

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»**

Інститут прикладного системного аналізу
Кафедра системного проектування

«До захисту допущено»

Завідувач кафедри

_____ А.І.Петренко
(підпис) (ініціали, прізвище)

“ ” _____ 2020 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 122 – комп’ютерні науки та інформаційні
(код і назва)

на тему: Використання мікросервісів в інформаційному забезпеченні охорони
здоров’я.

Виконав: студент 6 курсу, групи ДА-391мп
(шифр групи)

_____ Федоренко Даниїл Павлович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____ к.т.н., доцент Чкалов О.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту к.т.н., доцент Чкалов О.В. _____
(назва розділу) (посада, науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

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

Нормоконтроль _____ к.т.н., доцент, Кирюша Б.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

**Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»**

Інститут/факультет Інституту прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ **А.І.Петренко** _____
(підпис) (ініціали, прізвище)

«___» _____ 202_р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Федоренко Даниїлу Павловичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Використання мікросервісів в інформаційному забезпеченні охорони здоров'я.

науковий керівник дисертації Чкалов Олексій Валерійович к.т.н. _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___» _____ 20__ р. № _____

2. Строк подання студентом дисертації - _____

3. Об'єкт дослідження: мікросервіси в інформаційному забезпеченні охорони здоров'я.

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою): реалізація мікросервісів в інформаційному забезпеченні охорони здоров'я.

5. Перелік завдань, які потрібно розробити

1. Дослідити архітектуру мікросервісів.
2. Дослідити існуючі методології мікросервісів
3. Дослідити мікросервіси над монолітами.
4. Реалізувати мікросервіс.

6. Перелік графічного (ілюстративного) матеріалу: презентація на тему «Використання мікросервісів в інформаційному забезпеченні охорони здоров'я»

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Реалізація стартап-проекту	Чкалов О.В., доцент		

9. Дата видачі завдання 7 липня 2020_____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	7 липня 2020	
2	Ознайомлення з тех. літературою	11 вересня 2020	
3	Збір інформації	14 вересня 2020	
4	Дослідження предметної області та існуючих рішень	22 вересня 2020	
5	Розробка програмного додатку та його розгортання у хмарі	3 жовтня 2020	
6	Тестування додатку	14 жовтня 2020	
7	Оцінка результатів та опис практичних рекомендацій	26 жовтня 2020	
8	Оформлення дипломної роботи	15 грудня 2020	

Студент

(підпис)

Д. П. Федоренко
(ініціали, прізвище)

Науковий керівник дисертації

(підпис)

О. В. Чкалов
(ініціали, прізвище)

РЕФЕРАТ НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему “ Використання мікросервісів в інформаційному
забезпеченні охорони здоров'я ”
студентом Федоренко Даниїлом Павловичем

Робота виконана на 148 сторінках, містить 34 ілюстрації, 22 таблиці. При підготовці використовувалася література з джерела.

Актуальність. Організації охорони здоров'я, усвідомлюючи потенціал технології Інтернету речей (IoT), швидко застосовують цю технологію, щоб забезпечити значне покращення якості та ефективності послуги. Однак ці розумні та взаємопов'язані пристрої можуть виступати потенційними «чорними дверима» в IT-мережі лікарні, надаючи зловмисникам доступ до конфіденційної інформації. Як результат, кібератаки на медичні пристрої IoT зростають з останніх кількох років. Це зростаюче занепокоєння всіх зацікавлених сторін, оскільки вплив таких атак полягає не лише у втраті грошей або конфіденційності, але й життя багатьох пацієнтів також піддається ризику. Беручи до уваги різні типи пристроїв IoT, які можна виявити підключеними до мережі лікарні, традиційні орієнтовані на хост рішення безпеки (наприклад, антивірус, програмні виправлення) суперечать реалістичній інфраструктурі IoT (наприклад, обмежене обладнання, відсутність належних вбудованих заходів безпеки) . Існує потреба у рішеннях безпеки, які враховують такі проблеми пристроїв IoT, як неоднорідність використовуваних технологій та протоколів, обмежені ресурси з точки зору акумулятора та обчислювальної потужності тощо.

Мета. Ознайомитися з мікросервісами та вразливостями медичних пристроїв IoT; запровадити новий підхід, який використовує основу мікропослуг як адаптивне та гнучке рішення безпеки для вирішення проблеми.

Завдання. Дисертація зосереджена на атаках відбитків пальців ОС через їх важливість для зломисників зрозуміти мережу вцілому. У цій дипломній роботі було розроблено три мікросервіси, кожна з яких призначена для обслуговування конкретного функціоналу. Кожен з цих мікросервісів має невеликий розмір, використовуючи оперативну пам'ять приблизно 50 МБ.

Об'єкт дослідження. Мікросервіси.

Предмет дослідження. Мікросервіси.

Наукова новизна. Наукова новизна роботи полягає в дослідженні пристроїв IoT, як неоднорідність використовуваних технологій та протоколів, обмежені ресурси з точки зору акумулятора та обчислювальної потужності.

Практична цінність. Практична цінність роботи полягає у подальшому її використанні для лікарень.

Публікації.

Ключові слова. мікросервіс, ОС відбитків пальців, атака, архітектура мікросервісів.

РЕФЕРАТ НА МАГИСТЕРСКУЮ ДИССЕРТАЦИИ

выполненную на тему "Использование в информационном обеспечении
охраны здоровья"

студентом Федоренко Даниилом Павловичем

Работа выполнена на 148 страницах, содержит 34 иллюстрации, 22 таблицы.
При подготовке использовалась литература из источника.

Актуальность. Медицинские организации, осознавая потенциал технологии Интернета вещей (IoT), быстро используют эту технологию для значительного повышения качества и эффективности услуг. Однако эти умные и взаимосвязанные устройства могут выступать в роли потенциальных «черных дверей» в ИТ-сети больницы, предоставляя злоумышленникам доступ к конфиденциальной информации. В результате в последние несколько лет участились кибератаки на медицинские устройства IoT. Это вызывает растущую озабоченность всех заинтересованных сторон, поскольку последствия таких атак заключаются не только в потере денег или конфиденциальности, но и в жизнях многих пациентов. Учитывая различные типы устройств IoT, которые могут быть подключены к больничной сети, традиционные решения для обеспечения безопасности, ориентированные на хост (например, антивирус, программные исправления), работают вразрез с реальной инфраструктурой IoT (например, ограниченное оборудование, отсутствие надлежащей безопасности). Существует потребность в решениях безопасности, которые решают проблемы устройств IoT, такие как неоднородность используемых технологий и протоколов, ограниченные ресурсы с точки зрения времени автономной работы и вычислительной мощности.

Цель. Ознакомиться с микросервисами и уязвимостями медицинских устройств IoT; внедрить новый подход, использовать основу микрослужб как адаптивное и гибкое решение безопасности для решения проблемы.

Задание. Диссертация сконцентрирована на атаках отпечатков пальцев ОС за их важность для злоумышленников понять сеть в целом. В этой дипломной работе было разработано три микросервиса, каждая из которых предназначена для обслуживания конкретного функционала. Каждый из этих микросервисов имеет небольшой размер, использовали оперативную память приблизительно 50 МБ.

Объект исследования. Микросервисы.

Предмет исследования. Микросервисы.

Научная новизна. Научная новизна работы заключается в исследовании устройств IoT, таких как неоднородность используемых технологий и протоколов, ограниченные ресурсы с точки зрения батареи и вычислительной мощности.

Практическая ценность. Практическая ценность работы заключается в ее дальнейшем использовании в больницах.

Публикации.

Ключевые слова. микросервис, ОС по отпечатку пальца, атака, микросервисная архитектура.

ABSTRACT

ON MASTER'S THESIS

performed on the topic "Use of microservices in health information provision"
student Fedorenko Danyil Pavlovich

The work is done on 148 pages, contains 34 illustrations, 22 tables. Literature from the source was used in the preparation.

Topicality. Health organizations, recognizing the potential of Internet of Things (IoT) technology, are rapidly using this technology to significantly improve the quality and efficiency of the service. However, these smart and interconnected devices can act as potential "black doors" in a hospital's IT network, giving attackers access to confidential information. As a result, cyberattacks on IoT medical devices have been on the rise for the past few years. This is a growing concern for all stakeholders, as the impact of such attacks is not only in the loss of money or confidentiality, but also in the lives of many patients. Given the different types of IoT devices that can be found connected to a hospital network, traditional host-oriented security solutions (eg, antivirus, software patches) run counter to a realistic IoT infrastructure (eg, limited equipment, lack of proper security). There is a need for security solutions that address the challenges of IoT devices, such as the heterogeneity of technologies and protocols used, limited resources in terms of battery life, and computing power.

Goal. Get acquainted with microservices and vulnerabilities of medical devices IoT; introduce a new approach that uses the micro-services framework as an adaptive and flexible security solution to solve the problem.

Task. The dissertation focuses on OC fingerprint attacks because of their importance for attackers to understand the network as a whole. In this thesis, three microservices were developed, each of which is designed to serve specific

functionalities. Each of these microservices is small in size, using approximately 50 MB of RAM.

Object of study. Microservices.

Subject of study. Microservices.

Scientific novelty. The scientific novelty of the work lies in the study of IoT devices, such as the heterogeneity of the technologies and protocols used, limited resources in terms of battery and computing power.

Practical value. The practical value of the work lies in its further use for hospitals. Publications.

Keywords. microservice, fingerprint OS, attack, microservice architecture.

ЗМІСТ

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ЩОДО МІКРОСЕРВІСІВ	13
1.1 Мікросервіси над монолітами	13
1.2 Операційна система Fingerprinting	19
2 ПІДСУМОК.....	25
2.1 Мотивація	25
2.2 Пов'язані роботи	30
3 МЕТОДОЛОГІЯ.....	35
3.1 Інфраструктура розвитку	35
3.2 Архітектура системи	36
3.3 Поток процесу	39
4 ВПРОВАДЖЕННЯ	42
4.1 Fingerprinting ОС та оцінка вразливості (MS ONE)	42
4.2 Моніторинг атак (MS TWO)	43
4.3 Аналіз PCAP (MS THREE)	46
4.4 Шаблони атак Fingerprinting ОС.....	46
4.5 Бази даних.....	49
5 ДЕМОНСТРАЦІЯ І РЕЗУЛЬТАТИ	53
6 ОБГОВОРЕННЯ	63
7 ВИСНОВКИ ТА МАЙБУТНЯ РОБОТА.....	69
7.1 Майбутня робота	69
7.2 Висновок.....	70

	11
8 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ	72
8.1 Опис ідеї проекту	72
8.2 Технічний аудит ідеї проекту	74
8.3 Аналіз ринкових можливостей	75
8.4 Розробка ринкової стратегії проекту.....	85
8.5 Розробка маркетингової програми	89
ПЕРЕЛІК ПОСИЛАНЬ	95
ДОДАТОК А	100
ДОДАТОК В	133

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

JS – JavaScript

API - application programming interface

CVE - Common Vulnerabilities and Exposures

Nmap - Network Mapper

FDA - Food and Drug Administration

IoT - internet of things

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ЩОДО МІКРОСЕРВІСІВ

1.1 Мікросервіси над монолітами

У недалекому минулому корпоративні додатки розроблялися великою групою розробників, що працюють над єдиним монолітним додатком. У [9] монолітний додаток визначається як програмний додаток, модулі якого не можуть виконуватися самостійно. Ці програми, як правило, складаються з 3 частин: бази даних, інтерфейсу користувача на стороні клієнта та додатка на стороні сервера [25]. Серверна програма відповідає за обробку HTTP-запитів від клієнта, реалізацію основної бізнес-логіки, зчитування або модифікацію даних з бази даних та надання відповідей клієнту (зазвичай з HTML, JSON або XML). Логічно, що це модульна архітектура, але програма упакована та розгорнута як моноліт - єдиний логічний виконуваний файл. Архітектура моноліту має кілька переваг [33]:

- Простий у розробці
- Простий у тестуванні
- Простий у розгортанні - просто скопіюйте упаковану програму на сервер
- Простий у масштабі - просто запустіть кілька копій програми за балансиrom навантаження

Однак цей, здавалося б, простий підхід стикається з кількома обмеженнями [33]:

- Важко зрозуміти: по мірі зростання програми, після декількох ітерацій, ця проста програма стає монолітною системою, що складається з мільйонів рядків коду. По суті, це переростає у велику кульку бруду - стає занадто складним, щоб будь-який окремий розробник міг його повністю зрозуміти.

- Складніше в обслуговуванні: обслуговування програми (виправлення помилок) стає надзвичайно складним та занадто трудомістким. Через відсутність повного розуміння кодової бази, шанси на неправильні зміни також вищі.
- Перешкоджання модифікаціям: Постійна розробка таких програм важка, оскільки часто ці програми мають величезний розмір і, отже, довший час запуску. Це призводить до втрати продуктивності, оскільки для оновлення будь-якої частини програми буде потрібно перерозгортання програми, а отже, розробникам доведеться почекати. Навіть після передислокації, проблема ручного тестування всієї програми все ще стоїть.
- Непропорційне масштабування: Масштабування монолітних програм є складним, оскільки різні модулі можуть мати суперечливі вимоги до ресурсів. Наприклад, одному модулю може знадобитися більша швидкість обробки, а іншому - високопродуктивна пам'ять. Але оскільки обидва модулі повинні працювати разом, можна зробити лише один із 2 варіантів.
- Погана надійність: надійність - це проблема монолітних застосунків. Помилка будь-якого одного модуля може призвести до збиття всієї програми. Крім того, оскільки всі випадки програми однакові, це призводить до повної недоступності програми.
- Неможливість використання нових технологій: нарешті, розробка монолітного додатку означає назавжди відданість одному стеку технологій. Це створює істотну проблему відсутності можливості використовувати відповідний інструмент для роботи. Очевидно, що переписати цілого «звіра» в нові рамки неможливо. Для вирішення цих питань багато технологічних лідерів, таких як Amazon, Google, eBay та Netflix, прийняли так званий шаблон архітектури мікросервісів. [32] визначає архітектуру мікросервісів як "архітектуру, яка структурує додаток як набір вільно пов'язаних служб, що

співпрацюють". Замість побудови одного моноліта, додаток складається з набору менших, автономних служб, які можуть працювати у власному просторі пам'яті та масштабувати незалежно один від одного. Малий означає, що кожна служба несе одну відповідальність. Автономний означає, що служба має свою незалежну інфраструктуру. Архітектура мікросервісів базується на концепції композиції [34]. Додаток побудований на основі декількох мікросервісів. Композиція працює успішно, оскільки потрібно зробити лише одне - додавати компоненти разом. Не потрібно міняти компоненти; просто побудуйте нові для розгляду більш конкретних випадків. Це дозволяє швидше кодувати, оскільки нам не потрібно змінювати старий код.

Рисунок 1.1 підкреслює різницю між архітектурним стилем монолітного додатка та додатком, заснованим на мікросервісах. З одного боку, моноліт використовує однакову інфраструктуру для всіх функціональних можливостей. Однак мікросервіси повністю розділяють весь свій стек технологій - кожна служба працює як окремий процес (може також працювати в окремих середовищах). Зв'язок через HTTP / HTTPS є єдиним зв'язком між мікросервісами. Кожна мікросервісна служба також може мати власне сховище бази даних або може також спільно використовувати між іншими мікросервісами.

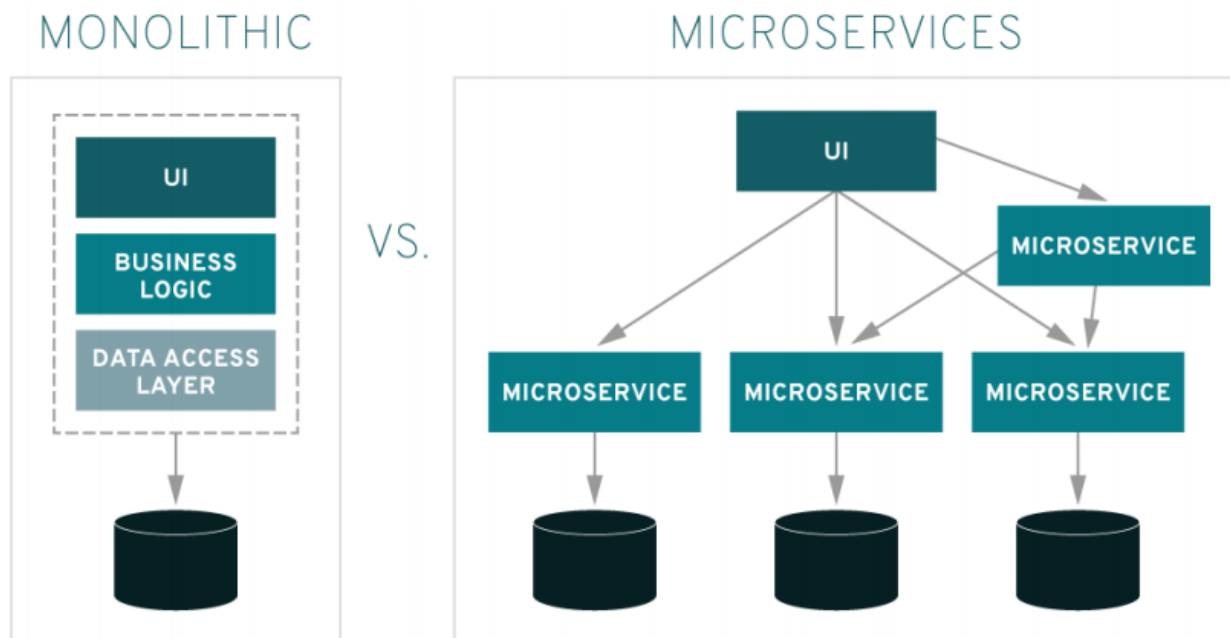


Рис. 1.1: Монолітна архітектура проти архітектури мікропослуг [31]

Архітектура мікросервісів має наступні переваги [25]:

- Послуги мінімальні та дрібнозернисті, призначені для виконання однієї функції, тому їх може створити кілька розробників.
- Коли інтерфейси мікросервісів виставляються за допомогою, скажімо, RESTful API, вони також можуть бути використані та використані іншими службами та програмами без необхідності прямого з'єднання.
- Послуги існують як незалежно розгортаються одиниці і можуть масштабуватися незалежно від інших служб. Таким чином, це дозволяє постійно розгортати складні програми, тому організації можуть вдосконалювати свої програми в дорозі.
- Це усуває бар'єр у прийнятті останніх технологій за допомогою програмування поліглотів. Розробники можуть використовувати будь-яку з найкращих технологій, що підходить для реалізації функціональних можливостей служби.

- Моніторинг безпеки набагато простіший за допомогою цієї архітектури, оскільки всі служби ізольовані, тому будь-яка проблема з однією службою не може впливати на інші служби.
- Він забезпечує хороший захисний механізм. Навіть якщо одна служба перестає працювати, інші служби продовжують працювати.

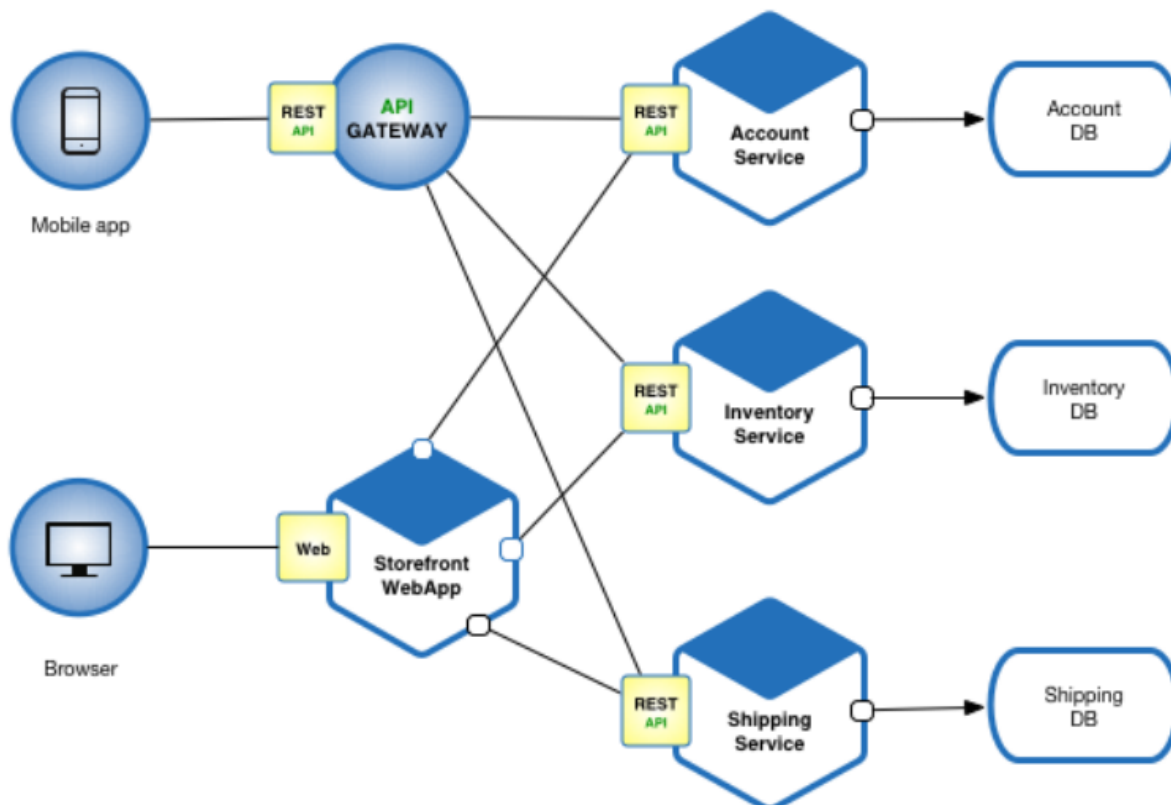


Рис. 1.2: Впровадження архітектури мікропослуг [32]

На рисунку 1.2 наведено приклад програми, реалізованої за допомогою архітектури мікросервісів. Програма, що показана тут - це програма для електронної комерції, яка має як веб-сайт, так і мобільний додаток для своїх клієнтів. Кожна функціональна область програми реалізована за допомогою власного мікросервісу. Мікросервіси мають API, який споживається іншими мікросервісами, а також клієнтами програми (тут - мобільний додаток та браузер). У сучасному світі розробки програмного забезпечення одна з

основних існуючих проблем полягає в тому, що ми, розробники, не можемо кодувати досить швидко. Ця проблема існує, оскільки ми не можемо зменшити наш технічний борг. Відповідно до Wikipedia, технічний борг - це концепція розробки програмного забезпечення, яка представляє приховану вартість додаткової переробки, спричинену вибором простого рішення зараз, а не кращого рішення, яке займе більше часу. Це трапляється тому, що багато разів ми повинні стикатися з компромісами між якістю та своєчасністю доставки. На додаток до цього, багато разів вимоги бізнесу змінюються в середині проекту. Отже, ми змушені використовувати швидкі хаки та клуби для досягнення цілей. Щоб зробити програмне забезпечення ідеальним, ми часто вдаємося до рефакторингу, що в підсумку призводить до величезного технічного боргу. Архітектура мікросервісів пропонує більш просте рішення для зменшення технічного боргу. Компромісом цієї гнучкості є складність, яка пов'язана з управлінням значною кількістю розподілених послуг [25]. Управління складно з наступних причин:

- Щоб потенційно повторно використовувати послуги, їх потрібно легко знайти. Крім того, інтерфейси служби повинні бути належним чином задокументовані, щоб їх можна було легко використовувати повторно.
- Взаємозалежності між службами слід підтримувати та контролювати, оскільки час простою послуг або відключення послуг може мати ефект доміно на всю програму.

Архітектура мікросервісів приділяє багато уваги, настільки, що стрімко рухається до піку завищених очікувань на циклі Гартнера Нуре [33]. Цикли Гартнера Нуре надають графічне зображення зрілості та впровадження технологій та додатків, а також те, як вони потенційно мають значення для вирішення реальних бізнес-проблем та використання нових можливостей [14].

Архітектура мікросервісів також пропонує багато історій успіху, а також деякі помилки. Отже, деякі компанії готові піти на ризик, інші - ні.

1.2 Операційна система Fingerprinting

Першим кроком у будь-якому вторгненні завжди є розвідка. Fingerprinting ОС (Відбитки пальців або Дактилоскопія) - один із кроків, здійснених під час розвідки мережі цілі. Подібно до того, як відбитки пальців людини є найвищим інструментом для точної ідентифікації людини, операційна система також має унікальні характеристики у способі спілкування з іншими сутностями, що допомагають ідентифікувати його в мережі. Це можливо, оскільки різні операційні системи мають різну реалізацію набору протоколів TCP / IP. Протоколи задокументовані у відповідному Запиті на коментарі (RFC), який розробники можуть використовувати при їх реалізації. Однак існують певні сфери, де є лише рекомендації, і рішення залишається за розробником. Наприклад, розробники різних ОС вибирають різні значення для початкового Time to Live (TTL) та розміру вікна TCP. Переглядаючи та аналізуючи певні прапорці протоколів, параметри та дані в пакетах, які ОС відправляє через мережу, можна зробити обґрунтовані здогадки про ОС [3]. Fingerprinting ОС - це процес аналізу. Fingerprinting ОС є важливою, оскільки адміністратори мережі можуть використовувати її для кращого проектування та впровадження засобів контролю в мережах та локальних машинах [2]. За допомогою відбитків пальців ОС легко відрізнити ПК від серверів, маршрутизаторів та принтерів. Регулярне сканування мережі з виявленням ОС може допомогти підтримувати мережевий інвентар чистим та сучасним. Крім того, ефективне дактилоскопічне дослідження ОС є життєво важливою навичкою тестування на проникнення. Дактилоскопія ОС є особливо важливою для зловмисних користувачів (або чорних капелюхів), оскільки, якщо зловмисник може ідентифікувати ОС, яка працює на цільових машинах, вони можуть використовувати її, щоб визначити, які саме вразливості

використовувати. Кожна ОС, що використовується сьогодні, має свої унікальні вразливості. Потенційно здогадуючись про працюючу ОС, надзвичайно легко з'ясувати специфічні для ОС вразливі місця. Ці знання надзвичайно допомагають зловмиснику, оскільки зловмисник тепер може усунути значну кількість вразливостей, яких немає в цілі. Наприклад, немає сенсу намагатися використовувати службу IIS на сервері Apache (оскільки сервер Apache не може мати вразливості Microsoft IIS). Також часто трапляється, що публічно виявляється помилка або вразливість, про яку навіть виробники ОС досі не знають, і, отже, відповідні виправлення ще не розроблені. Отже, зміцнення вашої мережі проти відбитків пальців ОС може, іноді, запобігти її 0-денним атакам. Дактилоскопія ОС класифікується на 2 категорії, активну та пасивну [2].

- Активна відбитка пальців - це процес надсилання пакетів на віддалений хост та аналіз відповідних пакетів відповідей [1]. Активні дактилоскопії зазвичай дають точніші результати за коротший проміжок часу, ніж пасивні дактилоскопії, оскільки вони мають доступ до більшої кількості даних. Однак є шанси отримати неточні результати. Але у випадку з активним відбитком пальців, якщо на цільовому хості чи в мережі працює IDS або брандмауер, існує більша ймовірність виявлення та блокування.

Nmap (Network Mapper) - найпоширеніший активний інструмент відбитків пальців. Nmap - це безкоштовна утиліта з відкритим кодом для виявлення мережі та аудиту безпеки. Адміністратори системи та мережі часто використовують Nmap для інвентаризації мережі або для контролю за безпекою своїх мереж. Основною метою Nmap є швидке сканування великих сегментів мережі на наявність активних пристроїв, а також звіт про відкриті порти та служби на цих пристроях [3]. Nmap - це в основному мережевий сканер, але він також має модуль відбитків пальців ОС. У ньому

використовується текстова база даних із понад 1500 записів, внесених громадою протягом часу. Однак при використанні Nmap неможливо гарантувати схованість. Що стосується відбитків пальців ОС, Nmap надсилає численні зонди TCP, UDP та ICMP до знайдених відкритих і закритих портів цілі, а потім аналізує відповідні відповіді. Як вже згадувалося раніше, різні операційні системи залишають різну інформацію в своїх пакетах TCP, UDP та ICMP. Nmap аналізує ці частини інформації, щоб повернути потенційні здогадки ОС. У цьому проекті ми використовували Nmap в одному з наших мікросервісів для відбитків пальців ОС. Xprobe - ще один активний інструмент відбитків пальців. Ми спробували використати це у своєму проекті, але у нас не було останньої версії, і ми не змогли отримати підтримку від спільноти з деяких питань. Тому ми вирішили не використовувати його.

- Пасивне відбиток пальців, навпаки, не генерує трафіку під час відбитків пальців, але воно прослуховує (або нюхає) та аналізує вже існуючий мережевий трафік між хостом сканування та цільовою машиною. Отже, це крадіжка, ніж активна дактилоскопія, і, отже, найкращий інструмент, коли хочеться уникнути виявлення або блокування IDS або брандмауером. Багато разів пасивні інструменти для відбитків пальців працюють повністю в автономному режимі, аналізуючи раніше захоплений мережевий трафік. Результати пасивних відбитків пальців, як правило, менш точні, оскільки даних для аналізу обмежено. Деякі з часто використовуваних інструментів для пасивних відбитків пальців - це Network Miner, p0f та Satori. Для цього проекту ми розробили спеціальну програму NodeJS, використовуючи існуючий проект C++ на GitHub, розроблений Ghali [15], для аналізу файлів даних, що містять раніше захоплений мережевий трафік (відомі як файли PCAP) та виявлення спроб атак на відбитки пальців ОС.

Більшість методів дактилоскопії використовують різницю в пакетах, що генеруються різними операційними системами (див. Рисунок 1.3). Деякі із загальноприйнятих методів аналізують наступне [1]:

- Значення IP TTL
- Значення IP-ідентифікатора
- Розмір вікна TCP
- Параметри TCP (пакетів TCP SYN та SYN-ACK)
- Пакети ICMP Echo
- Поле агента користувача пакета HTTP

Operating System (OS)	IP Initial TTL	TCP window size
Linux (kernel 2.4 and 2.6)	64	5840
Google's customized Linux	64	5720
FreeBSD	64	65535
Windows XP	128	65535
Windows 7, Vista and Server 2008	128	8192
Cisco Router (IOS 12.4)	255	4128

Рис. 1.3: Значення TTL та розміру вікна для різних платформ ОС [17]

1.3 Спільні вразливості та ризики (CVES)

CVE - це список записів офіційно відомих вразливих мір кібербезпеки, причому кожен запис складається з унікального ідентифікатора, опису та принаймні одного посилання на відповідні звіти про вразливість або

рекомендації. Програма загальних вразливостей та ризиків (CVE) - це програма, започаткована в 1999 році некомерційною організацією MITER, яка управляє науково-дослідними та дослідницькими центрами, що фінансуються федеральним урядом [37]. Метою програми є виявлення та каталогізація уразливостей програмного забезпечення чи мікропрограмного забезпечення у „безкоштовний” словник для організацій для підвищення їхньої безпеки. Основна мета словника - забезпечити стандартизований спосіб ідентифікації кожної

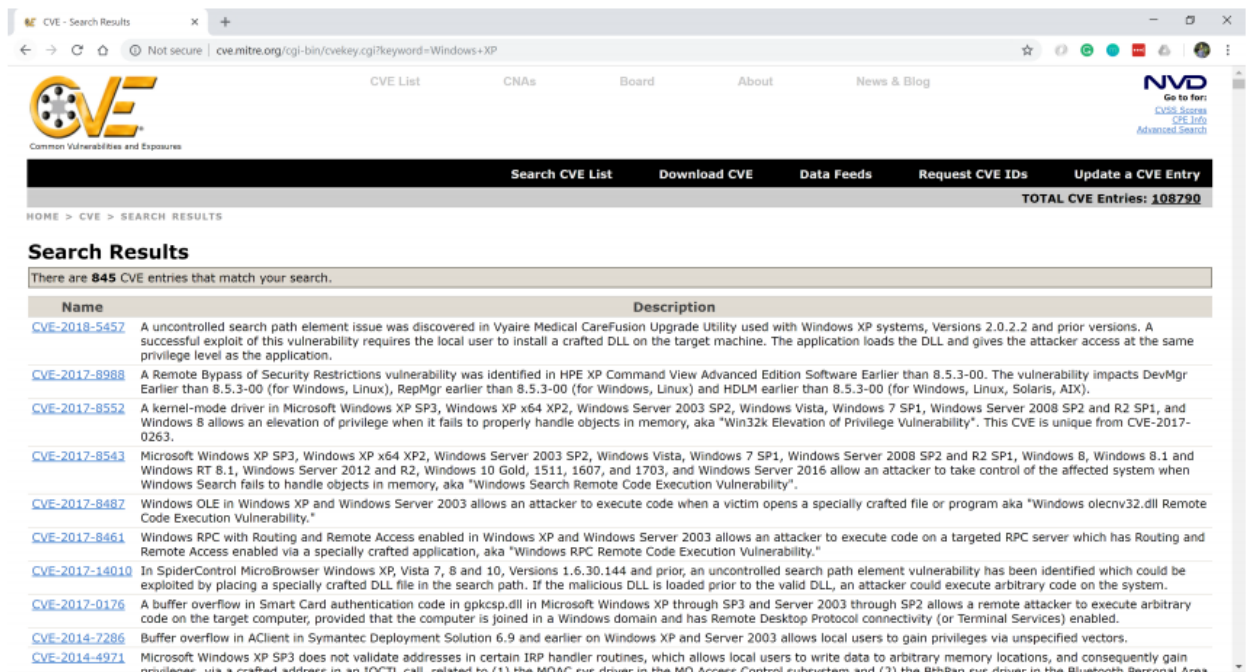


Рис. 1.4: Записи про загальні вразливості та вплив (CVE) для Windows XP

відомої вразливості чи впливу. За допомогою цих стандартних ідентифікаторів адміністратори безпеки можуть отримати доступ до технічних деталей щодо конкретної вразливості в декількох сумісних із CVE джерелах. CVE не надають жодної інформації щодо ризику, впливу, виправлення або детальної технічної інформації. На малюнку 1.4 нижче показано деякі записи CVE для операційної системи Windows XP. Для цього проекту ми шукали загальнодоступні API для доступу до цієї багатої бази

вразливостей. Однак жодного не знайдено, тому ми розробили веб-додаток для вишкрібання веб-файлів NodeJS, який скасовує результати пошукового запиту, що складаються з CVE для даної операційної системи.

2 ПІДСУМОК

2.1 Мотивація

Як вже згадувалося раніше в главі 1, галузь охорони здоров'я швидко впроваджує Інтернет речей. Існує багато застосувань IoT в охороні здоров'я, починаючи від пристроїв моніторингу здоров'я в Інтернеті, закінчуючи розумними датчиками та радіологічними пристроями. Ця інтеграція функцій IoT у медичні вироби приносить значні покращення якості та ефективності служби охорони здоров'я [5] [12]. За даними Всесвітньої організації охорони здоров'я (ВООЗ), медичним приладом є „будь-який інструмент, прилад, прилад, машина, прилад, імплантат, реагент або калібратор *in vitro*, програмне забезпечення, матеріали або інші подібні або пов'язані з ними вироби, призначені для використання в діагностиці, профілактика, моніторинг та лікування захворювань чи інших станів »[8]. Ці пристрої можуть виконувати деякі обчислення, отримувати вхідні дані із навколишнього середовища через датчики, виводити значення в навколишнє середовище через виконавчі механізми та також спілкуватися з іншими пристроями. Медичні вироби стають дедалі розумнішими та взаємопов'язані [35]. Однак ця швидкозростаюча технологія орієнтована в першу чергу на вирішення бізнес-проблем, а безпека - це лише додаткова думка. Для досягнення найкоротшого часу виходу на ринок та низьких виробничих витрат ці медичні вироби виробляються без урахування безпеки. Отже, збільшується ризик зловмисних нападів на лікарні та клініки. Такі зловмисні атаки та порушення даних мають пряму загрозу для життя пацієнтів.

Якщо хакери можуть зламати такі пристрої, вони отримують необмежений доступ до медичної інформації пацієнта, а також можуть отримати контроль над самим пристроєм, щоб змінити основні функції пристрою. Для кращої оцінки впливу порушень зверніться до Рисунку 2.1, який показує витрати на

порушення даних на душу населення для різних галузей. Як видно чітко, вартість порушення даних найвища для галузі охорони здоров'я.

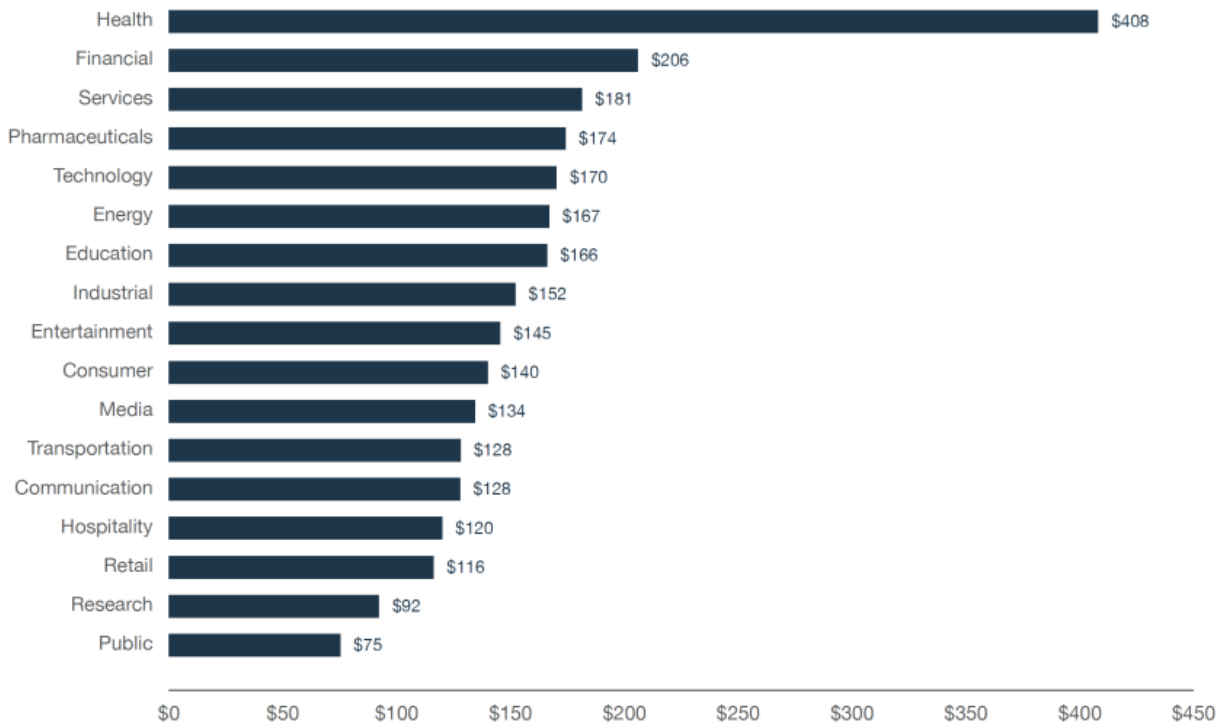


Рис. 2.1: Вартість порушення даних на душу населення для різних галузей [28]

З багатьох досліджень та звітів про інциденти [18] [38] видно, що кібератаки на медичні пристрої IoT зростають. Загрози для медичних виробів не виникли за одну ніч. За останні 5 років багато дослідників безпеки докладають зусиль, щоб з'ясувати недоліки, які призводять до отримання контролю над медичними пристроями. З різних досліджень, проведених Ервеном [40], Редкліффом [29] та Джеком [22], наведено кілька прикладів різних атак, які можливі на медичні прилади чи лікарняне обладнання:

- Інсулінові насоси можна зламати, щоб віддалено доставляти смертельні дози інсуліну пацієнтам.

- Кардіостимулятори можна зламати, щоб доставити смертельний удар напругою 830 вольт, використовуючи ноутбук на відстані 50 футів від пацієнта.
- Дефібриляторами з підтримкою Bluetooth можна маніпулювати, щоб доставити випадкові поштовхи в серце пацієнта або запобігти виникненню медично необхідного шоку.
- До рентгенівських променів можуть отримати доступ сторонні особи, які ховаються в мережі лікарні.
- Налаштування температури в холодильниках, що зберігають кров та наркотики, можуть бути скинуті, що спричинить псування.
- Цифрові медичні записи можуть бути змінені, щоб змусити лікарів поставити неправильний діагноз, виписати неправильні препарати або надати небажану допомогу.

Згідно з доповіддю Trend Micro (охоронної фірми) [13], вони провели дослідження за допомогою пошукової системи Shodan, яка індексує підключені до Інтернету пристрої, з якої вони виявили, що понад 100 000 медичних пристроїв та лікарняних комп'ютерів у всьому світі відкрито схильні та потенційно вразливі до ризику. Важливо розуміти, яку інформацію ми можемо отримати з цих різних дослідницьких робіт. Слабкі сторони медичних виробів, як правило, діляться на дуже декілька загальних категорій: [21].

- Дефекти кодування: Дефекти кодування - це помилки, виявлені в програмному забезпеченні або коді прошивки, що дозволяють зловмисникам створювати помилкові ситуації, які можуть допомогти їм розпочати атаку. Наприклад, переповнення буфера, введення SQL, міжсайтовий сценарій (XSS) тощо.

- Недоліки дизайну програмного забезпечення: Сюди входять помилкові конфігурації програмного забезпечення, такі як існування жорстко закодованих паролів або стандартних та слабких паролів, а також відсутність шифрування даних, навіть коли вони не використовуються, і вони просто перебувають у стані спокою в певному сховищі пристрою.
- Відсутність захисту від несанкціонованого доступу: Це відсутність заходів захисту від несанкціонованого доступу в апаратному забезпеченні, мікропрограмі та стеку програмного забезпечення пристрою. Захист пристроїв від фальсифікації або простої зворотної інженерії необхідний, щоб зловмисникам було важче отримати root-доступ до пристрою.
- Сторонні вразливості: не тільки програмне забезпечення, розроблене на пристрої, може бути ризикованим, але й сторонні програми, якими може користуватися пристрій, можуть мати вразливості. Сюди входять фреймворки веб-програм, прошивка та платформи операційної системи.

Ми хотіли б наголосити на важливості захисту медичних виробів. Йдеться не лише про пряму загрозу життю пацієнта за допомогою медичного пристрою. Хоча це віддалене явище, це можливо. Захист медичних виробів означає захист критичної інфраструктури. Це означає запобігання зловмисним людям взяти під контроль цю інфраструктуру, запобігання потенційним шантажам лікарень та виробників пристроїв, а також безпеку пацієнтів. Однією з цілей, що стоять за розробкою медичних виробів, повинно бути те, що захищене програмне забезпечення повинно продовжувати працювати належним чином під час зловмисної атаки. Це включає апаратні та програмні аспекти, а також навмисні та ненавмисні зовнішні загрози.

Через відсутність твердого доручення виробники та організації охорони здоров'я намагаються дотримуватися вказівок FDA [11], щоб зменшити ризики безпеки пристроїв. Згідно з дослідженням, проведеним Synopsys та

Ponemon [36], лише 51% виробників медичних виробів та 44% організацій охорони здоров'я дотримуються вказівок FDA. Виробники пристроїв повинні забезпечити вбудований захист своїх пристроїв, регулярно їх виправляти та оновлювати. Для організацій охорони здоров'я необхідна цілісна та гнучка стратегія безпеки, щоб не відставати від вразливостей IoMT. Але оскільки лікарняні системи мають усі види медичних пристроїв, їм важко забезпечити безпеку кожного з цих пристроїв. Ми вважаємо, що використання шаблону архітектури мікросервісів може виявитися корисним у цьому сценарії. Диференційовані характеристики пристроїв IoMT представляють важливі проблеми для розробки повного рішення безпеки. Проблеми включають обмеження потужності, неоднорідність використовуваних технологій і протоколів (Bluetooth, інфрачервоний порт, Wi-Fi тощо), відсутність стандартної операційної системи для медичних виробів тощо. Також, за даними Управління з контролю за продуктами та ліками [10], більшість медичних пристроїв використовують вразливе готове програмне забезпечення. Оскільки кожен тип пристроїв може мати різний механізм взаємодії, створення лише одного рішення для розміщення всіх цих пристроїв з часом перетвориться на складний моноліт. За допомогою архітектури мікросервісів можна створити спеціалізовані мікросервіси, фокус яких буде зосереджений на взаємодії з певним типом медичних пристроїв IoT (наприклад, монітором серцевого ритму, датчиком температури тощо). Послуги можна використовувати для фіксації та моніторингу трафіку, який передається через пристрій. З послугою, присвяченою лише один тип пристрою, це дозволяє нам використовувати правильну технологію / мову для розробки необхідних функцій взаємодії та захоплення для цього пристрою. Наприклад, C має потужні бібліотеки для нюху трафіку з пристроїв, що використовують різні комунікаційні протоколи, такі як Bluetooth або Інтернет. Як тільки трафік буде зібраний, служба може направити його іншій службі,

яка спеціалізується на аналізі трафіку на шкідливі дії. Служба аналізу може посылатися на окрему базу даних, яка зберігає відомі підписи атаки. Для проведення аналізу та взаємодії з базою даних, можна використовувати обширний набір інструментів Python. Як тільки виявляється зловмисна діяльність, окрема служба попередження може надсилати попередження відповідним органам. Тут JavaScript можна використовувати для створення інтерактивних візуалізацій. Різні мови програмування мають свої переваги та недоліки. Такі мови, як Java, C, C++ або C#, високоефективні у використанні ресурсів і забезпечують високу продуктивність, але, з іншого боку, вартість розробки, як правило, висока для цих мов через обмежену кількість розробників, які добре володіють цими мовами. Тоді як Python та JavaScript зручні для користувача, а отже, розробників легко знайти. Але ці мови також можуть мати проблеми, пов'язані з паралельністю. Детальний приклад такого впровадження ми представляємо у розділі 6.

2.2 Пов'язані роботи

Кібербезпека IoT є основним напрямком досліджень як в академічному, так і в корпоративному світі. Недавнє дослідження Juniper Research підрахувало, що витрати на рішення Інтернету речей з питань кібербезпеки досягнуть понад 6 млрд доларів у всьому світі до 2023 р. Безкоштовний у користуванні проект OWASP Internet of Things (IoT) Internet2 розроблений, щоб допомогти виробникам, розробникам та споживачам краще зрозуміти проблеми безпеки, пов'язані з використанням технології IoT, та допомогти користувачам у прийнятті рішень щодо безпеки при розробці чи використанні технологій IoT. Проект також опублікував список 10 найкращих уразливих місць IoT у 2014 році [26]. На ринку безпеки IoT існує багато комерційних рішень, пропонованих як традиційними постачальниками безпеки, так і новими стартапами, орієнтованими виключно на безпеку IoT [4]. Однак в академічному світі дослідницька робота в галузі безпеки IoT обмежена. Це

пов'язано з наявністю технічних проблем, таких як відсутність емпіричних даних, пов'язаних з різними пристроями IoT, та нестача дієвих підписів атак, характерних для IoT. Обсяг дослідницької роботи звужується, коли ми спеціально шукаємо кібербезпеку пристроїв IoT у галузі охорони здоров'я. Що стосується пристроїв IoT у галузі охорони здоров'я, більшість дослідницьких робіт [35], [40], [29], [22], [13] та [21] стосуються таких питань, як необхідність кібербезпеки та загальні вказівки щодо того, як виробники, а також організації охорони здоров'я повинні думати про захист екосистеми. Нам не вдалося знайти жодної опублікованої роботи з відкритим кодом, яка б передбачала реальну реалізацію такого рішення. Отже, натомість ми розширили наш спектр до рішень безпеки для пристроїв IoT загалом. Ми розглянули два різні підходи, розглянуті для рішень IoT з кібербезпеки. У роботі [30] Rahman et al. забезпечує комплексний огляд апаратних механізмів захисту для запобігання атакам на пристрої IoT. Автори припускають, що апаратні методи є перспективною альтернативою лише програмним механізмам, оскільки останні часто самі вразливі до віддалених атак. Крім того, програмні рішення вимагають регулярних оновлень та виправлень, щоб захистити систему.

Апаратні методи збирають апаратні функції для аналізу переважаючих вразливостей програмного рівня. Дослідження аналізує різні техніки, що підтримуються апаратним забезпеченням для забезпечення безпеки IoT. Однак автори також зазначають, що апаратні методи дійсно стикаються з декількома проблемами, і існують обмеження щодо ефективності виявлення або запобігання. Отже, вони припускають, що необхідні подальші дослідження. У роботі [39] Ю та ін. висловлює два ключові спостереження щодо захисту IoT: (1) поточні рішення на базі хоста (наприклад, антивірус, програмні виправлення) неефективні, і існує потреба залежати від мережевих рішень, оскільки пристрої IoT мають обмежені ресурси і можуть взаємодіяти з кожним іншим, що призводить до залежностей між пристроями; (2) традиційні

механізми захисту від статичного периметра (наприклад, IDS та брандмауер на шлюзах) не можуть захистити пристрої IoT, оскільки пристрої вбудовані глибоко всередині мереж. Для вирішення цих питань автори запропонували мову абстракції політики, яка може охопити ключові контексти навколишнього середовища, взаємодії між пристроями та безпеки пристроїв IoT. Для застосування, автори застосовують концепцію програмно визначених мереж (SDN) до реалізацій IoT, де можна: (а) швидко розробити та розгорнути нові захисні мережі, призначені для випадків використання IoT та (b) динамічно змінювати політики безпеки мереж відповідно до поточного робочого контексту різних пристроїв та середовища. Автори окреслюють архітектуру безпеки під назвою IoTSec, як показано на малюнку 2.2, яка використовує спеціальні мікро-середні скриньки (функції мікромережевої безпеки), які виконують роль шлюзів безпеки для кожного пристрою IoT. Централізована платформа керування відстежує контекст різних пристроїв IoT та робочого середовища і на основі цього створює загальний вигляд для забезпечення дотримання політики на різних пристроях. Виходячи з цього подання, платформа управління створює і конфігурує кожен з мікро-середніх

ящиків та застосовує необхідні механізми переадресації для маршрутизації

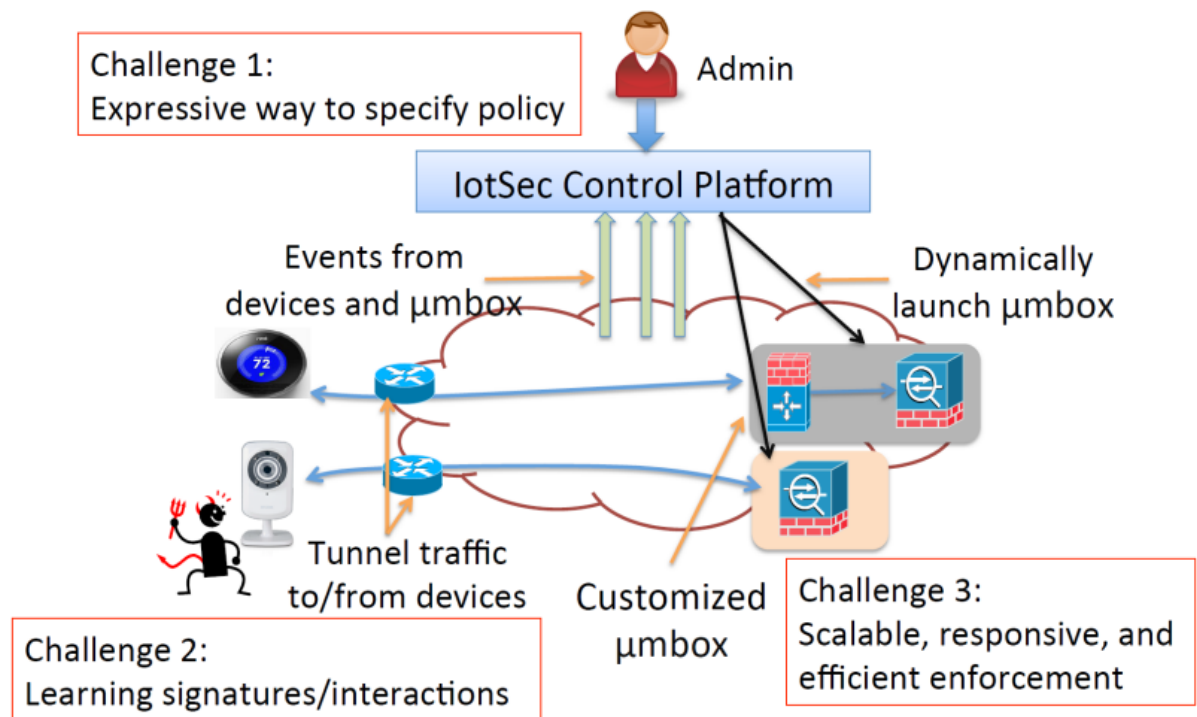


Figure 2: High-level vision of IoTSec

Рис. 2.2: Запропонована архітектура безпеки з [39]

пакетів до цих мікро-проміжних ящиків. З нашого дослідження літератури ми виявили, що не було проведено жодної відповідної роботи, принаймні в академічному світі, який використовує додаток на основі мікропослуг як рішення для кібербезпеки. Це не означає, що наша ідея використання мікросервісів у сфері IoT є першою у своєму роді. Шаблони архітектури Інтернету речей (IoT) та архітектури мікропослуг - це останні концепції ІТ-галузі, і їм успішно вдалося набутися значного ажіотажу в цій галузі. Було проведено кілька обговорень та дослідницьких робіт [23], [16], [7], [24], [6] та [27], які намагалися застосувати шаблон архітектури мікросервісів при розробці систем IoT. Одним з таких прикладів є [23], в якому Lu et al. запропонував використовувати шаблони архітектури мікросервісів для розробки систем IoT. Вони пропонують переосмислити IoT - замість того, щоб розглядати фундаментальну одиницю конструкції як "річ", ми повинні

сприймати системи IoT як побудовані з "послуг". Автори показують, як шаблони для шлюзів API, розподілу послуг, виявлення послуг, утримання та контролю доступу можуть поєднуватися для вирішення проблем у розробці систем IoT, використовуючи два приклади як приклади.

3 МЕТОДОЛОГІЯ

У цьому розділі ми починаємо з надання інфраструктури та технологій, що використовуються для розвитку. Далі ми представимо архітектурну схему розробленої системи (що складається із Сервера та Клієнта). Слідуючи діаграмі архітектури системи, ми обговоримо процес виконання системи.

3.1 Інфраструктура розвитку

Додаток Server було розроблено та розгорнуто на віртуальній машині під управлінням дистрибутиву Kali Linux. Дистрибутив Kali Linux призначений для вдосконаленого тестування на проникнення та цифрової криміналістики [20]. Kali Linux має кілька інструментів, корисних для різних завдань інформаційної безпеки, таких як тестування на проникнення, дослідження безпеки та комп'ютерна криміналістика. Ми вибрали Kali Linux, щоб скористатися такими інструментами, як Nmap, які поставляються попередньо встановленими та налаштованими. Крім того, оскільки приналежність до сімейства Linux, мережеве програмування, таке як збір пакетів даних та синтаксичний аналіз, менш рестриктоване, ніж у Windows. Віртуальну машину надала Група технічних служб (TSG) при FAU College of Engineering and Computer Science. Клієнтська програма була розроблена на персональному комп'ютері Windows 10. І Сервер, і Клієнт були розроблені виключно з використанням JavaScript у NodeJS, відкритому коді, міжплатформеному середовищі виконання JavaScript, побудованому на механізмі JavaScript Chromes V8. Мікропослуги були розроблені з використанням фреймворку SenecaJS, інструментарію мікропослуг для NodeJS. Додаток Client було розроблено з використанням фреймворку Electron, бібліотеки з відкритим кодом, розробленої GitHub для створення настільних додатків з використанням HTML, CSS та JavaScript.

3.2 Архітектура системи

На рисунку 3.1 показана архітектурна схема системи. Як показано, Сервер складається переважно з 3 мікросервісів, які піддаються використанню RESTful API.

- Першою мікросервісом є Служба оцінки відбитків пальців та вразливостей ОС, яка надає відбитки відбитків пальців ОС будь-якої цілі за допомогою інструмента Nmap. Служба виконує утиліту Nmap у фоновому режимі на отриманій IP-адресі цілі. Після ідентифікації запущеної ОС на цільовому пристрої служба також надає загальновідомі вразливості ОС, витягуючи їх із бази даних загальних вразливостей та ризиків (CVE), як описано у Розділі 1.3. Для демонстрації масштабування ми запускаємо два екземпляри першої служби, оскільки Nmap зазвичай займає значний час виконання (приблизно 3 хв на запит).

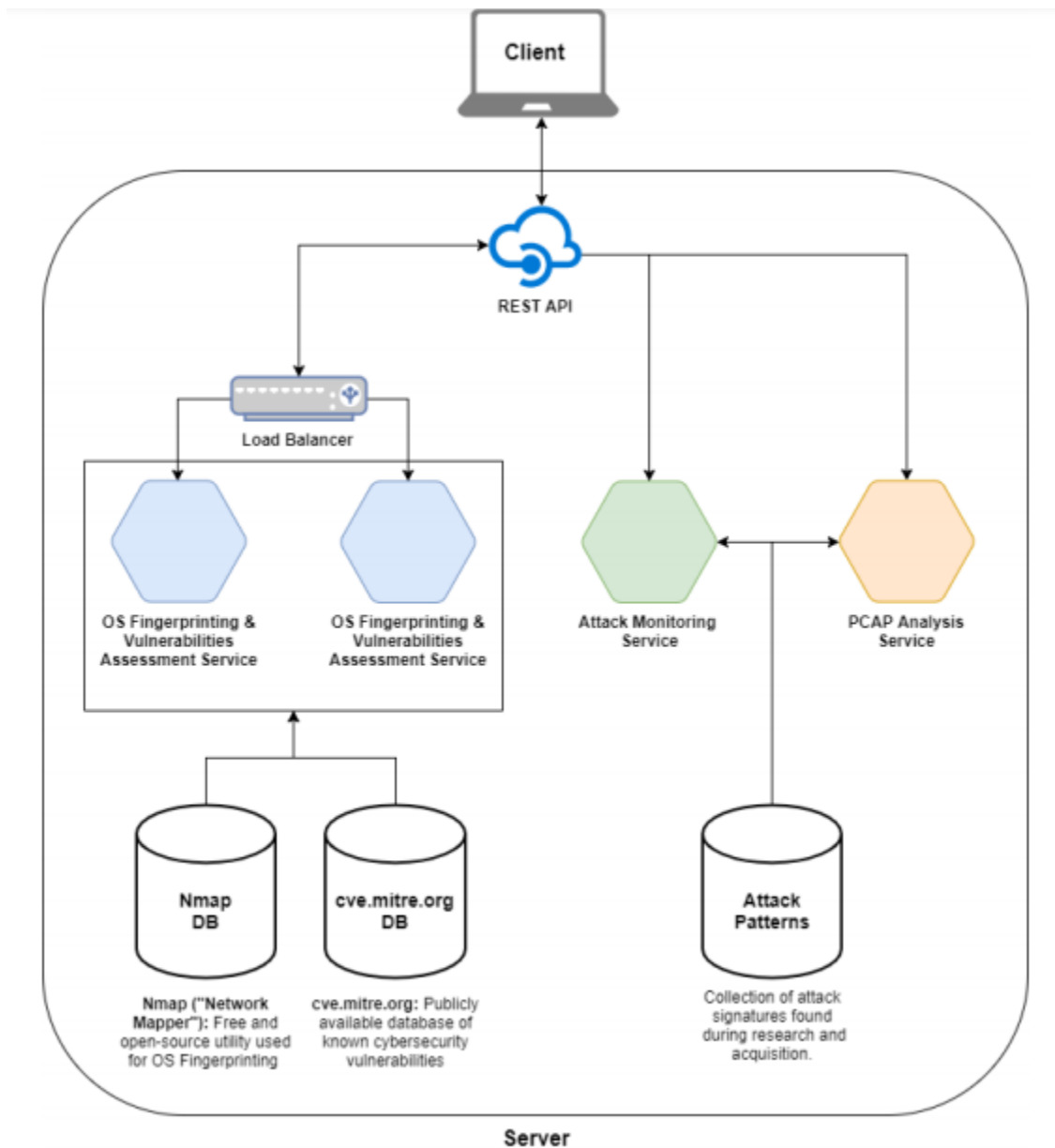


Рис. 3.1: Архітектура системи

Обидва екземпляри розміщені за балансиrom навантаження, який розподіляє отримані запити між двома екземплярами відповідно до алгоритму Round-Robin.

- Друга служба, служба моніторингу атак, забезпечує можливість моніторингу заданої цілі в режимі реального часу для атак відбитків пальців ОС. Послуга надає клієнтам три функції - розпочати сеанс моніторингу в реальному часі на хост-машині, отримувати оновлення з сеансу та зупиняти поточний сеанс.

- Остання послуга, служба аналізу PCAP, може виявити будь-які спроби атак відбитків пальців ОС із даного файлу PCAP із раніше захоплених мережових пакетів. Файл PCAP завантажується на сервер клієнтом.

Всі три служби прослуховують запити на різних портах і чекають переадресованих клієнтських запитів від RESTful API. Отже, API в основному є точкою входу програми. На даний момент у нашому впровадженні всі три мікросервіси та API працюють від одного хоста, але при необхідності їх можна виділити на різні сервери.

Ми також створили власну базу даних MongoDB, яка в даний час зберігає різні шаблони атаки відбитків пальців ОС, які були виявлені під час нашого дослідження. Як служба моніторингу атак, так і служба аналізу PCAP спочатку зчитують ці шаблони з бази даних, а потім використовують їх для ідентифікації атак відбитків пальців ОС. Під час нашого дослідження ми виявили, що схеми атак не мають фіксованої структури або формату. Це пояснюється тим, що, як вже згадувалося раніше, відбитки пальців ОС здійснюються шляхом аналізу різних полів та прапорів переданих пакетів даних. Отже, у цих шаблонах не використовується фіксована кількість полів. В результаті ми обрали MongoDB, оскільки, будучи базою даних NoSQL, досить легко зберігати таку неоднорідну структуру шаблонів. Також, маючи на увазі майбутнє, буде досить просто розширити колекцію без потреби в зміні існуючих зразків. Ми називаємо нашу поточну систему одиницею збору підписів атак, оскільки, використовуючи цю архітектуру системи, ми можемо шукати нові шаблони, а також збирати їх у базі даних. Пізніше, як ми побачимо у розділі 6, ми можемо використовувати цю багату колекцію шаблонів під час моніторингу фактичних пристроїв IoT.

Нарешті, у нашій системі Клієнт - це міжплатформова настільна програма, яка може працювати з будь-яких машин Windows, Linux або Mac. Клієнт

розроблений як багатопрофільна програма з головним екраном та трьома окремими екранами, що відповідають трьом мікропослугам. Клієнт використовується як додаток для користувача для надсилання запитів та отримання результатів від мікросервісів.

3.3 Поток процесу

У цьому розділі ми опишемо потік виконання системи, як показано на малюнку 3.2. Клієнт надає 3 загальні функції для користувача:

- Операційна система відбитків пальців заданої цілі та отримання списку відомих вразливостей визначеної ОС.
- Моніторинг сервера в реальному часі на наявність атак відбитків пальців.
- Аналіз заданого файлу PCAP та виявлення спроб атак відбитків пальців ОС.

Клієнт робить різні REST-запити до API, що працює на Сервері, залежно від вибраних користувачем функціональних можливостей. Коли API отримує запити, він аналізує URL-адресу запиту, а потім, використовуючи визначений набір маршрутів, робить відповідний виклик відповідній службі. Служби виконують своє завдання і після завершення відповідають на API. Нарешті API надсилає відповідь клієнту. На малюнку 3.3 нижче наведено перелік доступних функцій для користувача, відповідні запити REST та дії, вжиті конкретною мікрослужбою.

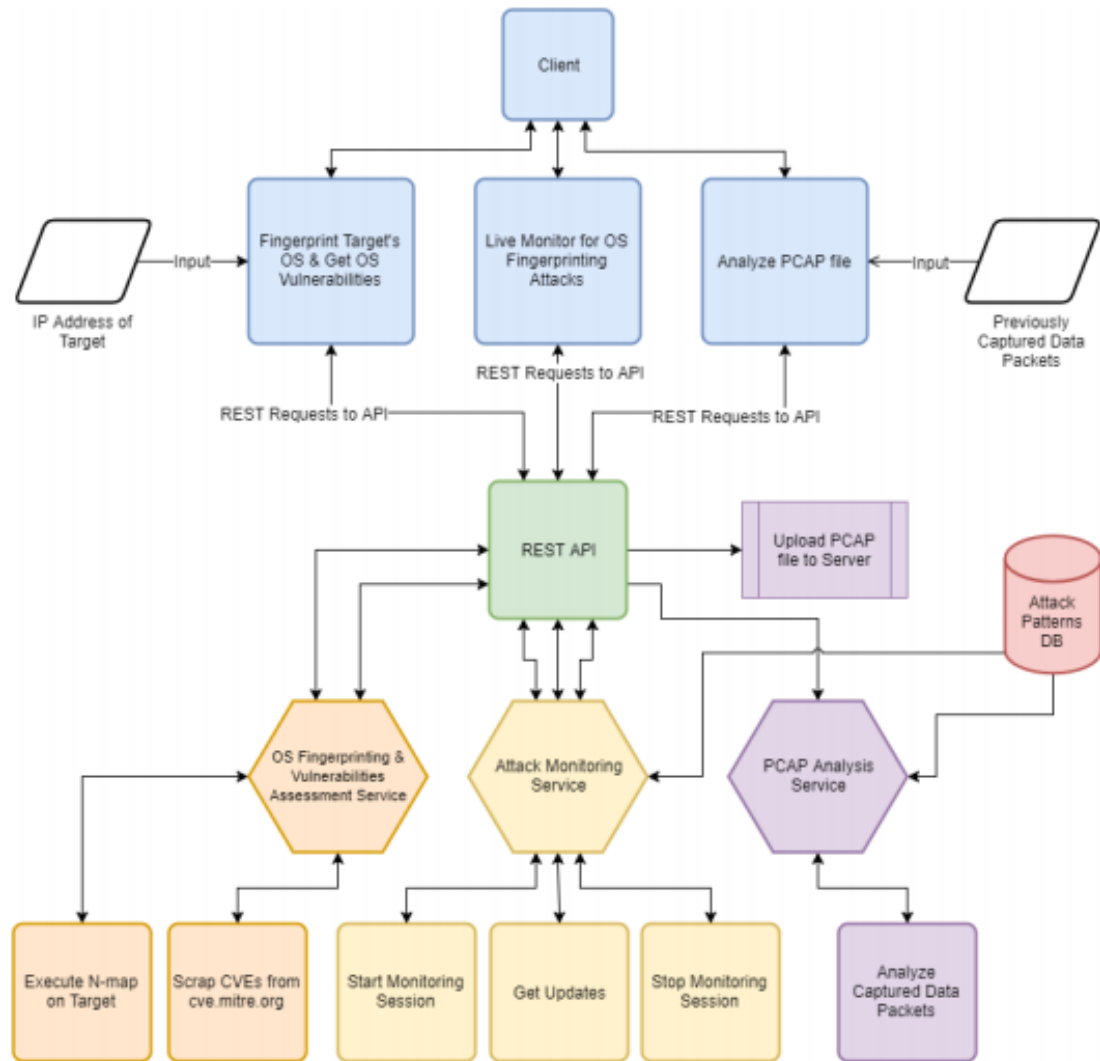


Рис. 3.2: Виконання потік системи

Table 3.1 List of existing functionalities and corresponding REST Requests

Function	REST Request	Actions performed
Fingerprint OS of remote target	GET http://\${API_URL}/api/msOne/fpOS?target=\${queryIP}	Server calls the OS Fingerprinting & Vulnerabilities Assessment Service which then executes 'Execute N-map on Target' function
Get OS Vulnerabilities	GET http://\${API_URL}/api/msOne/getCVEs?target=\${osName}	Server calls the OS Fingerprinting & Vulnerabilities Assessment Service which then executes 'Scrap CVEs from cve.mitre.org' function
Start live monitoring for OS Fingerprinting Attacks	GET http://\${API_URL}/api/msTwo/start	Server calls the Attack Monitoring Service which first reads in the Attack Patterns from 'Attack Patterns DB' and then executes 'Start Monitoring Session' function
Get updates from monitoring session	GET http://\${API_URL}/api/msTwo/updateSFA GET http://\${API_URL}/api/msTwo/updateSLA	Server calls the Attack Monitoring Service which then executes 'Get Updates' function
Stop monitoring session	GET http://\${API_URL}/api/msTwo/stop	Server calls the Attack Monitoring Service which then executes 'Stop Monitoring Session' function
Upload PCAP file	POST http://\${API_URL}/fileupload	The Server saves the uploaded PCAP file to a pre-determined location
Analyze PCAP	GET http://\${API_URL}/api/msThree/analyzePCAP?fileName=\${name}	Server calls the PCAP Analysis Service which first reads in the Attack Patterns from 'Attack Patterns DB' and then executes 'Analyze Captured Data Packets' function

Рис. 3.3: Список існуючих функціональних можливостей та відповідні запити REST

4 ВПРОВАДЖЕННЯ

У цьому розділі ми обговоримо реалізацію трьох мікропослуг. Ми показуємо тут кілька фрагментів коду, але читачам слід звернутися до Додатку А для всього вихідного коду. Ми не надаємо деталей реалізації для Клієнта через його простоту. Однак ми також надаємо весь його вихідний код у Додатку В.

4.1 Fingerprinting ОС та оцінка вразливості (MS ONE)

Ця служба має 2 обов'язки. Першою відповідальністю є визначення ОС даної цілі за її IP-адресою. Для ідентифікації запущеної ОС служба виконує інструмент командного рядка Nmap за вказаною IP-адресою (див. Малюнок 4.1). Ми витягуємо можливі здогадки щодо ОС з результатів Nmap, і служба відповідає першим вгадуванням ОС (оскільки вона має найвищу впевненість). Якщо Nmap не може зробити припущення, що трапляється з багатьох причин, таких як відсутність відкритих / закритих портів на цілі або занадто велика кількість відбитків пальців збігаються для чіткого вгадування, служба відповідає з відповідною причиною. Друга відповідальність - отримати відомі вразливості для ідентифікованої ОС зі списку CVE. Для цього служба викликає функцію веб-обміну (див. Рис4.2), яку ми розробили для вилучення записів CVE з <http://cve.mitre.org>.

```
seneca.add({ role: 'msOne', cmd: 'fpOS' }, function (msg, respond) {
  console.log('Msg received here!');
  const { remoteIP } = msg;
  let commandToExec = `sudo nmap -sV -T5 -O -Pn ${remoteIP} | grep -m 1 -e "Aggressive OS guesses" -e "OS details" -e "No OS"`;
  exec(commandToExec, (error, stdout, stderr) => { ...
  });
});
```

Рис. 4.1: Команда для виконання Nmap для заданої цілі

```
seneca.add({ role: 'msOne', cmd: 'getCVEs' }, async function (msg, respond) {
  let { osQuery } = msg;
  let osCVEs = await scrapCVEs(osQuery);
  console.log(`Response: Found ${osCVEs.cve_id.length} CVEs.`);
  respond(null, osCVEs);
});
```

Рис. 4.2: Команда для отримання списку вразливостей

4.2 Моніторинг атак (MS TWO)

Ця служба спеціалізується на моніторингу в режимі реального часу хосту, на якому працює служба, та виявленні можливих випадків атак відбитків пальців в ОС. На схеми ідентифікації атак відбитків пальців в ОС посилалися з існуючого GitHub проект, розроблений Цезарем Гали [15]. Ці шаблони зберігаються окремо в базі даних MongoDB, щоб створити спільне сховище, яке можна розширити або навіть змінити в майбутньому. Шаблони, які використовуються для узгодження, коротко описані в розділі 4.4. Служба відповідає за відповідь на 3 команди:

- Почати сеанс моніторингу: Отримавши цю команду, служба розгалужує новий дочірній процес, який розпочне новий сеанс захоплення пакетів на пристрої мережевого інтерфейсу за замовчуванням хоста (наприклад, порт Ethernet або порт Wi-Fi) (див. Малюнок 4.3). Коли пакети захоплюються, для кожного пакета виконується мало перевірки стану розумності (як дійсний пакет IPv4). Якщо пакет проходить всі перевірки надійності, ми додатково відокремлюємо пакети TCP та ICMP та аналізуємо їх окремо. Далі ми шукаємо певні прапори або значення полів як у сегментах TCP, так і в ICMP, які можуть вказувати на можливі спроби відбитків пальців ОС. Через обмежений час та ресурси, ми зосередились лише на закономірностях у сегментах TCP та ICMP. Крім того, збираються пакети, що належать до тієї самої IP-адреси джерела, щоб ми могли виявити спроби відбитків пальців ОС, що відбуваються

протягом певного інтервалу часу (або атаки, що викликають стан), а не лише миттєво (або атаки без громадянства).

```
seneca.add({ role: 'msTwo', cmd: 'start' }, function (msg, respond) {
  newSession = fork('./liveFPDetection.js', [], { silent: false });
  newSession.send('start');
  newSession.on('message', (status) => {
    respond(null, { status });
  });
});
```

Рис. 4.3: Команда для запуску нового сеансу моніторингу в режимі реального часу

- Отримувати оновлення: Отримавши цю команду, служба надсилає повідомлення журналу клієнту як оновлення із сеансу, який працює на хості (див. Рис. 4.4). Розгалужений дочірній процес надає журнали, що описують появу підозрілих пакетів, які можуть бути спробами відбитків пальців ОС. У кожному повідомленні журналу вказується мітка часу появи пакета, IP-адреса джерела пакета, відповідний шаблон та IP-адреса призначення (в даному випадку IP-адреса хоста) та порт, на який пакет був відправлений. Розгалужений процес надає 2 різні типи журналів, один для атак без громадянства та інший для атак. Ось чому в команді ми маємо додатковий параметр, який визначає тип повідомлень журналу, для яких зроблено запит. Наш Клієнт налаштований так, що кожні 5 секунд робить запит на журнал атак без стану та кожні 70 секунд запитує журнали атак.

- Зупинити сеанс моніторингу: Отримавши цю команду, служба вбиває сеанс захоплення пакетів і відповідає підсумком, який складається з підрахунку різних виявлених атак відбитків пальців, а також повідомлень журналу, які можна використовувати для отримання додаткової інформації. Повідомлення журналу також зберігаються в окремому файлі журналу на сервері (див. Рис. 4.5).

```
seneca.add({ role: 'msTwo', cmd: 'getLogs' }, function (msg, respond) {
  if (msg.type == 'StatelessAttack') {
    newSession.send('getLogs:StatelessAttack');
    newSession.on('message', (consoleLogs) => {
      respond(null, { consoleLogs });
    });
  } else if (msg.type == 'StatefullAttack') {
    newSession.send('getLogs:StatefullAttack');
    newSession.on('message', (analysisResults) => {
      respond(null, { analysisResults });
    });
  }
});
```

Рис. 4.4: Команда для отримання оновлень із сеансу моніторингу в реальному часі

```
seneca.add({ role: 'msTwo', cmd: 'stop' }, function (msg, respond) {
  newSession.send('stop');
  newSession.on('message', (response) => {
    newSession.kill();
    respond(null, response);
  });
});
```

Рис. 4.5: Команда для зупинки сеансу моніторингу в реальному часі

```
seneca.add({ role: 'msThree', cmd: 'analyzePCAP' }, function (msg, respond) {
  const { fileName } = msg;
  let filePath = `${__dirname}/uploads/${fileName}`;

  let newSession = fork('./passiveFPDetection.js', [], { silent: false });
  newSession.send(filePath);
  newSession.on('message', (results) => {
    let { pktCounts, consoleLogs, analysisResults } = results;
    newSession.kill();
    console.log(`Response: Analyzed ${pktCounts.totalPkts} packets.`);
    respond(null, { pktCounts, consoleLogs, analysisResults });
  });
});
```

Рис. 4.6: Команда для аналізу існуючого файлу PCAP

4.3 Аналіз PCAP (MS THREE)

Ця послуга зосереджена на аналізі заданого файлу даних раніше захоплених мережевих пакетів (також відомого як файл PCAP) та виявленні випадків будь-яких атак на відбитки пальців ОС. Тут ми шукаємо ті самі закономірності, що і в сеансі моніторингу в режимі реального часу. Шаблони, які використовуються для узгодження, коротко описані в розділі 4.4. Служба відповідає за відповідь лише на одну команду - Аналіз захоплених пакетів даних. Отримавши цю команду, служба розгалужує новий дочірній процес, який розпочне синтаксичний аналіз вхідного файлу PCAP (див. Рис. 4.6). По мірі синтаксичного аналізу кожного пакету ми спочатку виконуємо кілька перевірок стану обґрунтованості, як дійсний пакет IPv4. Далі ми окремо аналізуємо сегменти TCP та ICMP та шукаємо певні прапори або значення полів, які можуть вказувати на можливі спроби відбитків пальців ОС. Через обмежений час та ресурси, ми зосередились лише на закономірностях у сегментах TCP та ICMP. Нарешті, коли весь файл проаналізований, служба відповідає зведенням, яке складається з кількох типів атак без відбитків пальців ОС, а також атак стану, якщо вони були виявлені.

4.4 Шаблони атак Fingerprinting ОС

Щоб виявити спроби відбитків пальців ОС, ми нюхаємо потрібну мережу (або аналізуємо наявний файл PCAP) і пасивно фіксуємо та аналізуємо пакети, передані через мережу. Ми класифікуємо атаки відбитків пальців ОС на дві категорії:

- Атака без громадянства: Атаки, здійснені окремими спеціально створеними пакетами, які мають незвичні прапори або значення полів. Такі напади виникають миттєво.

- Атака зі станом: це атаки, які відбуваються протягом певного інтервалу часу. Зловмисник аналізує значну кількість пакетів, щоб зробити детерміновану здогадку.

Ми хотіли б зауважити, що ці моделі не на 100% свідчать про зловмисні наміри. Оскільки RFC для різних протоколів містять лише рекомендації щодо того, як слід впроваджувати певні частини протоколу, розробники ОС мають свободу вибору, що їм подобається. Отже, існують нетипові закономірності, які при використанні генерують нестандартні відповіді з різних цільових ОС, що допомагає в їх ідентифікації. Але не можна гарантувати, що лише зловмисний користувач використовуватиме ці нетипові шаблони, іноді пакети можуть пошкодитися під час транзиту, або це може бути просто доброякісною помилкою відправника. Ось чому, коли такі шаблони узгоджуються, це може бути можливою атакою відбитків пальців ОС, але не точно. Крім того, ми просто використовуємо шаблони, як це є у посиланому проекті GitHub. Нам не вдалося знайти виправдання для всіх закономірностей. Ми шукаємо такі моделі для виявлення атак без громадянства:

- Пакет TCP без встановлених прапорів. У народі відомий як сканування NULL.
- Пакет TCP із встановленими прапорами SYN та FIN.
- Пакет TCP із встановленими прапорами RST та FIN.
- TCP-пакет із встановленими прапорами RST та SYN.
- TCP-пакет із встановленим лише прапором PSH.
- TCP-пакет із встановленими прапорами FIN та URG.
- Пакет TCP із встановленими прапорами SYN та URG.
- Пакет TCP із встановленими прапорами SYN, FIN та URG.

- Пакет TCP із встановленими прапорами FIN, PSH та URG. У народі відомий як сканування XMAS.
- Пакет TCP із встановленими прапорами SYN, FIN, PSH та URG.
- Пакет TCP із встановленими прапорами SYN, ECE та CWR.
- Пакет TCP із встановленими прапорами DF та ACK та розміром вікна = 1024.
- TCP-пакет із незвичною комбінацією опцій TCP. Наприклад, однією з таких комбінацій є: Масштабування вікна = 10, MSS (Максимальний розмір сегмента) = 1460, Вибіркове підтвердження = True, Розмір вікна = 1, Відмітка часу (Echo) = 0, Відмітка часу (Відправник) = 0xFFFFFFFF.
- Пакет ICMP з недійсним значенням коду або зарезервованим значенням типу.

Для виявлення атак, що містять стан, ми збираємо пакети з тієї самої IP-адреси джерела (або іншими словами, потенційного зловмисника), щоб ми могли проаналізувати їх як одну безперервну спробу, зроблену з використанням декількох наступних пакетів, а не лише одного окремого пакета. Поряд із просто групуванням пакетів за IP-адресою зловмисника, ми також відстежуємо унікальні спроби підключення TCP, а також встановлені зв'язки між однаковими номерами портів джерела та призначення. Зверніть увагу, що існує різниця між спробою з'єднання та встановленим з'єднанням, оскільки кожне нове TCP-з'єднання встановлюється лише після успішного тристороннього рукошлякування, тоді як спроба передбачає лише відправлення пакету SYN до порту призначення. Вся ця збережена інформація (як "стан") передається загальному сценарію як між службою моніторингу атак, так і службою аналізу PCAP, яка виявляє такі атаки стану, сприйнятливі до відбитків пальців ОС:

- Кілька спроб підключення TCP з одного порту джерела до одного порту призначення. Ми використали 5 як орієнтир для максимально дозованих TCP-з'єднань між парою портів джерела та призначення. Але це лише експериментальне значення, і фактичне значення може встановити адміністратор мережі / безпеки залежно від обсягу очікуваного мережевого трафіку.
- Сканування портів - процес, при якому клієнт надсилає запити на діапазон номерів портів на хості, щоб знайти відкритий порт. Сканування портів не завжди є шкідливою діяльністю, але це може бути обов'язковим кроком перед спробою відбитків пальців ОС, оскільки для багатьох методів потрібен відкритий порт, щоб отримати відповіді.
- Кілька пар TCP-пакетів із встановленими прапорами SYN та RST, що надсилаються на декілька портів призначення. Знову ж таки, ми використали 5 як еталон для максимально дозованих пар таких TCP-пакетів. Але це лише експериментальне значення, і фактичне значення може встановити адміністратор мережі / безпеки залежно від обсягу очікуваного мережевого трафіку. На рисунку 4.7 показані шаблони, що зберігаються в нашій базі даних.

4.5 Бази даних

Для тестування мікропослуг ми використали 2 набори даних, надані професором Еліасом БуХарбом¹. Набори даних є частиною його дослідницької роботи з кібербезпеки, проведеної в лабораторії розвідки кіберзагроз в Коледжі інженерії та обчислювальної техніки, Атлантичний університет Флориди. Одним із наборів даних є перелік зруйнованих пристроїв, розповсюджених в Інтернеті. Ми використовуємо цей набір даних, щоб спробувати відбитки пальців ОС за допомогою послуги MS One.

```

1 { "type": "stateless", "attack": "Null Scan", "sign": { "tcpFlags": 0 } }
2 { "type": "stateless", "attack": "SYN-FIN", "sign": { "tcpFlags": 3 } }
3 { "type": "stateless", "attack": "RST-FIN", "sign": { "tcpFlags": 5 } }
4 { "type": "stateless", "attack": "RST-SYN", "sign": { "tcpFlags": 6 } }
5 { "type": "stateless", "attack": "PSH", "sign": { "tcpFlags": 8 } }
6 { "type": "stateless", "attack": "FIN-URG", "sign": { "tcpFlags": 33 } }
7 { "type": "stateless", "attack": "SYN-URG", "sign": { "tcpFlags": 34 } }
8 { "type": "stateless", "attack": "SYN-FIN-URG", "sign": { "tcpFlags": 36 } }
9 { "type": "stateless", "attack": "XMAS Scan", "sign": { "tcpFlags": 41 } }
10 { "type": "stateless", "attack": "SYN-FIN-PSH-URG", "sign": { "tcpFlags": 43 } }
11 { "type": "stateless", "attack": "SYN-ECE-CWR", "sign": { "tcpFlags": 194 } }
12 { "type": "stateless", "attack": "ACK-DF-WS1024", "sign": { "tcpFlags": 33, "ipFlag_DF": 64, "windowSize": 1024 } }
13 { "type": "statefull", "attack": "FIN Scan", "sign": { "tcpFlags": 1 } }
14 { "type": "ICMP", "attack": "Invalid Code", "sign": { "type": 8, "codeGT": 0 } }
15 { "type": "ICMP", "attack": "Invalid Type",
16   "sign": {
17     "typeIN": [ 1, 2, 7, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29 ],
18     "typeGTE": 44
19   }
20 }
21

```

Рис. 4.7: Шаблони атак, що наразі зберігаються в базі даних

In [15]: healthIoT

Out[15]:

	ip	attack_type	packets_num	start_time	end_time	city	country	orgtype	iot	product	devicetype
87	216.229.60.121	Horizontal	11161	04/16/2018 17:00	04/16/2018 18:00	Marina Del Rey	United States	Medical And Dental Services	IoT	OpenSSH	undefined
518	112.94.226.232	Strobe	158	04/16/2018 17:41	04/16/2018 17:55	Guangzhou	China	Hospital	IoT	undefined	undefined
2469	80.191.174.5	Horizontal	100	04/16/2018 17:02	04/16/2018 17:59	Zahedan	Iran	Medical And Dental Services	IoT	undefined	undefined
5683	200.163.138.178	Strobe	177	04/16/2018 17:00	04/16/2018 17:59	Santo Andre	Brazil	Hospital	IoT	Postfix smtpd Apache httpd	undefined
7202	64.60.11.34	Strobe	571	04/16/2018 17:00	04/16/2018 17:56	Beverly Hills	United States	Medical And Dental Services	IoT	SonicWALL firewall http config OpenSSH	firewall
11307	72.15.20.5	Strobe	347	04/16/2018 17:00	04/16/2018 17:59	Saint Johnsbury	United States	Hospital	IoT	undefined	undefined

Connectra Check Point

Рис. 4.8: Небагато скомпрометованих жертв в Інтернеті з наборів даних

Одним із наборів даних є перелік зруйнованих пристроїв, розповсюджених в Інтернеті. Ми використовуємо цей набір даних, щоб спробувати відбитки пальців ОС за допомогою служби MS One. Набір даних складався з приблизно 340 000 унікальних IP-адрес, що належать жертвам атак, що відбулись у всьому світі протягом одногодинного інтервалу 16.04.2018. Жертви належали до різних організацій, таких як телекомунікації, виробництво, провайдери Інтернет-послуг, уряд, страхування, охорона здоров'я тощо. З них у нас було загалом 116 пристроїв IoMT. На рисунку 4.8 показано декілька цільових IP-адрес, а також відповідну інформацію про них. У нас також був набір даних darknet, що представляє один день небажаного інтернет-трафіку. Darknet - це

частина простору IP-адрес, яка є маршрутизованою, але не запускає жодних служб.

```
E:\Workspace\msThesis\Datasets\14>capinfos -K 14.pcap
File name:      14.pcap
File type:      Wireshark/... - pcapng
File encapsulation: Ethernet
File timestamp precision: microseconds (6)
Packet size limit: file hdr: (not set)
Packet size limit: inferred: 1498 bytes
Number of packets: 67 M
File size:      8103 MB
Data size:      5853 MB
Capture duration: 104399.997867 second
First packet time: 2013-11-30 19:00:00.001930
Last packet time: 2013-12-01 23:59:59.999797
Data byte rate: 56 kbps
Data bit rate: 448 kbps
Average packet size: 86.60 bytes
Average packet rate: 647 packets/s
SHA256:         94a7e4c70606264528380b798d348c888f932875ce086641ea3ab8ff20b17859
RIPEMD160:      17bbbed5a656c2644a034ea21ca8c37b001627bda
SHA1:           e6d19c7ffc2200219d157ec8d1edb9d0ba86376e
Strict time order: True
Capture application: mergecap
Number of interfaces in file: 1
Interface #0 info:
    Encapsulation = Ethernet (1 - ether)
    Capture length = 1500
    Time precision = microseconds (6)
    Time ticks per second = 1000000
    Time resolution = 0x06
    Number of stat entries = 0
    Number of packets = 67589771
```

Рис. 4.9: Файл PCAP, використаний при тестуванні для MS Three

Отже, будь-який трафік, спрямований на даркнет, можна вважати зловмисним, оскільки законний трафік туди ніколи не потрапить. Цей набір даних також надав професор Еліас БуХарб. Це файл PCAP розміром 8 ГБ із приблизно 67 мільйонами пакетів. Ми використовуємо цей набір даних для тестування служби MS Three. Завдяки величезному розміру, ми розділяємо файли на менші файли по 100 000 пакетів кожен, щоб ми мали швидший час завантаження та аналізу. Будучи набором даних darknet, це надзвичайно корисний ресурс для пошуку спроб відбитків пальців в ОС. На рисунку 4.9

показана основна інформація про файл PCAP, отриманий за допомогою capinfos2 інструмент від Wireshark.

5 ДЕМОНСТРАЦІЯ І РЕЗУЛЬТАТИ

У цьому розділі ми надамо знімки екрану Клієнта, що демонструють роботу нашої системи. Ми також проводимо аналіз використання пам'яті для мікросервісів, щоб краще зрозуміти переваги підходу до мікросервісів. Сервер розміщений на віртуальній машині Kali Linux. REST API працює на порту 80 і доступний за URL-адресою: <http://ohm.eng.fau.edu>. 2 екземпляри Служби оцінки відбитків пальців та оцінки вразливостей ОС прослуховують порти 10101 та 10102, а решта дві служби, Моніторинг атак та Аналіз PCAP, прослуховують порти 10103 та 10104 відповідно. Клієнт - це міжплатформова настільна програма, яка надсилатиме запити до API. Ми уявляємо Клієнта як інформаційну панель, з якої користувач може робити запити до мікрослужб і де відповіді можуть бути представлені у будь-якій формі як бажаний текстовий чи візуальний. Через обмеження часу ми використовуємо лише текстове представлення. На панелі інструментів Клієнт надає опцію Меню, за допомогою якої користувач може відвідати сховище проекту GitHub, а також Вийти із програми. Параметр розробника призначений для використання під час розробки для доступу до інструментів розробника Chromes для налагодження. Щоб продовжити далі, користувач може вибрати будь-яку з 3 кнопок, кожна з яких відкриває новий екран, що відповідає кожній з трьох мікропослуг. На малюнках 5.1 - 5.8 показані екрани, що відповідають кожній мікропослуді. Щоб зробити зручний інтерфейс та максимально інформативним, ми включили такі функції, які є такими:

- Для кращої читабельності користувачі можуть перейти в повноекранний режим, натиснувши клавішу F11, і перейти в звичайний режим вікна, натиснувши клавішу ESC.
- Як показано на малюнку 5.3, якщо користувач надає недійсну IP-адресу, Клієнт повідомляє про це, не ініціюючи запит на Сервер.

- На малюнку 5.3 ми також можемо побачити, що якщо Nmap не може зробити певну припущення ОС, він відповідає належним чином, і клієнт не відображає кнопку Отримати CVE.
- Як показано на малюнку 5.7, коли користувач подає файл PCAP для завантаження, відображається повідомлення про успішне або невдале завантаження файлу разом із деякими основними деталями про PCAP. Для відображення корисної інформації про файл PCAP ми використовуємо утиліту командного рядка capinfos з набору інструментів Wireshark. Повідомлення журналу виконання від REST API та служб наведені на рисунках 5.9 - 5.12. Зокрема, на малюнку 5.10 ми бачимо, як множинні запити до MS One збалансовані між двома запущеними екземплярами MS One. Запити пересилаються до будь-якого з 2-х примірників за допомогою алгоритму Round-Robin. Навантаження

балансування використовується лише для MS One, оскільки Nmap займає значну кількість часу на виконання. На рисунках 5.13 та 5.14 ми представляємо використання пам'яті кожного мікросервісу як уявлення про розмір. Для цього аналізу ми використали топ - програму диспетчера завдань, яку можна знайти у багатьох Unix-подібних операційних системах, що надає інформацію про використання процесора та пам'яті. Ми показали використання пам'яті у два моменти часу, один відразу після запуску кожного з мікросервісів і один після того, як мікросервіси подали по одному запиту від Клієнта. Після численних випробувань ми спостерігали, що мікросервіси використовують ОЗУ від 40 до 50 Мб під час її виконання, загалом, з декількома випадками, коли використання досягає 85 Мб. Ці нечасті випадки трапляються з кількох причин, наприклад, коли Nmap займає більше часу на виконання, оскільки він не може отримати збіг на відбитках пальців, або коли більший файл PCAP подається на аналіз.



Рис.5.1:Домашній екран клієнта, який працює під управлінням Windows10



Рис. 5.2: Домашній екран клієнта, що працює на Ubuntu 18.04

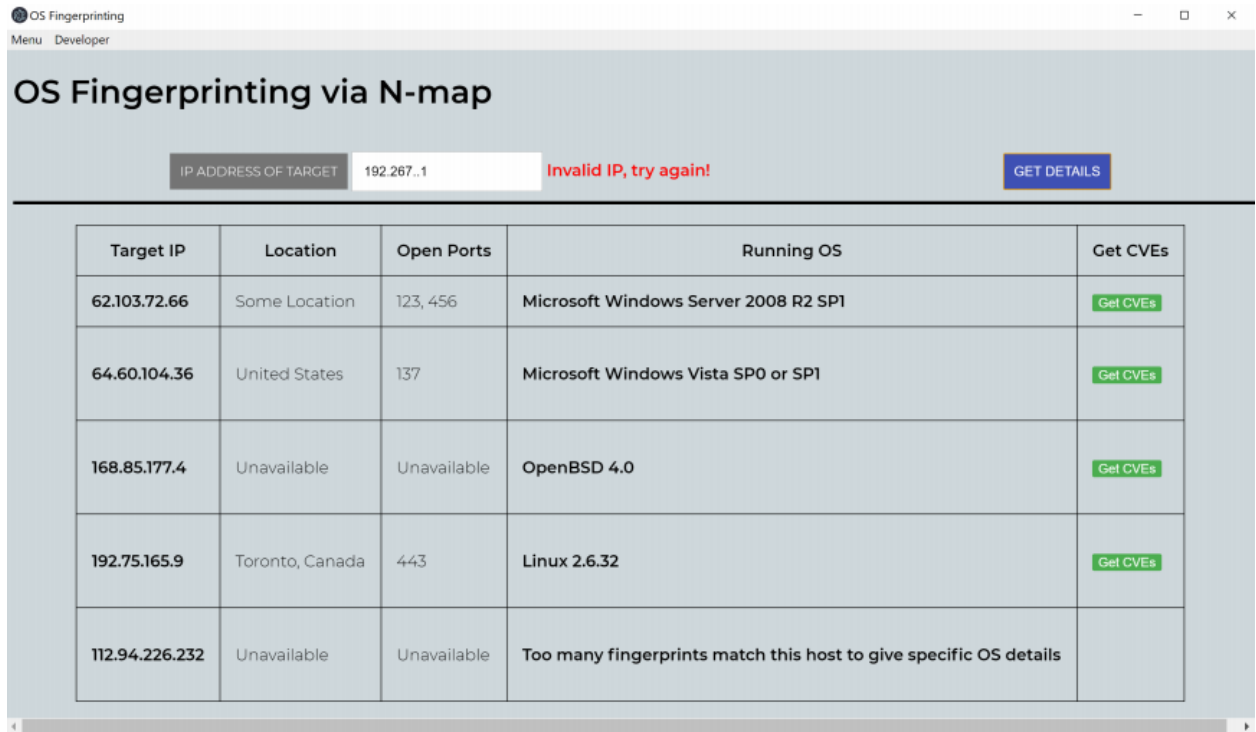


Рис. 5.3: Екран для відбитків пальців ОС за допомогою Nmap

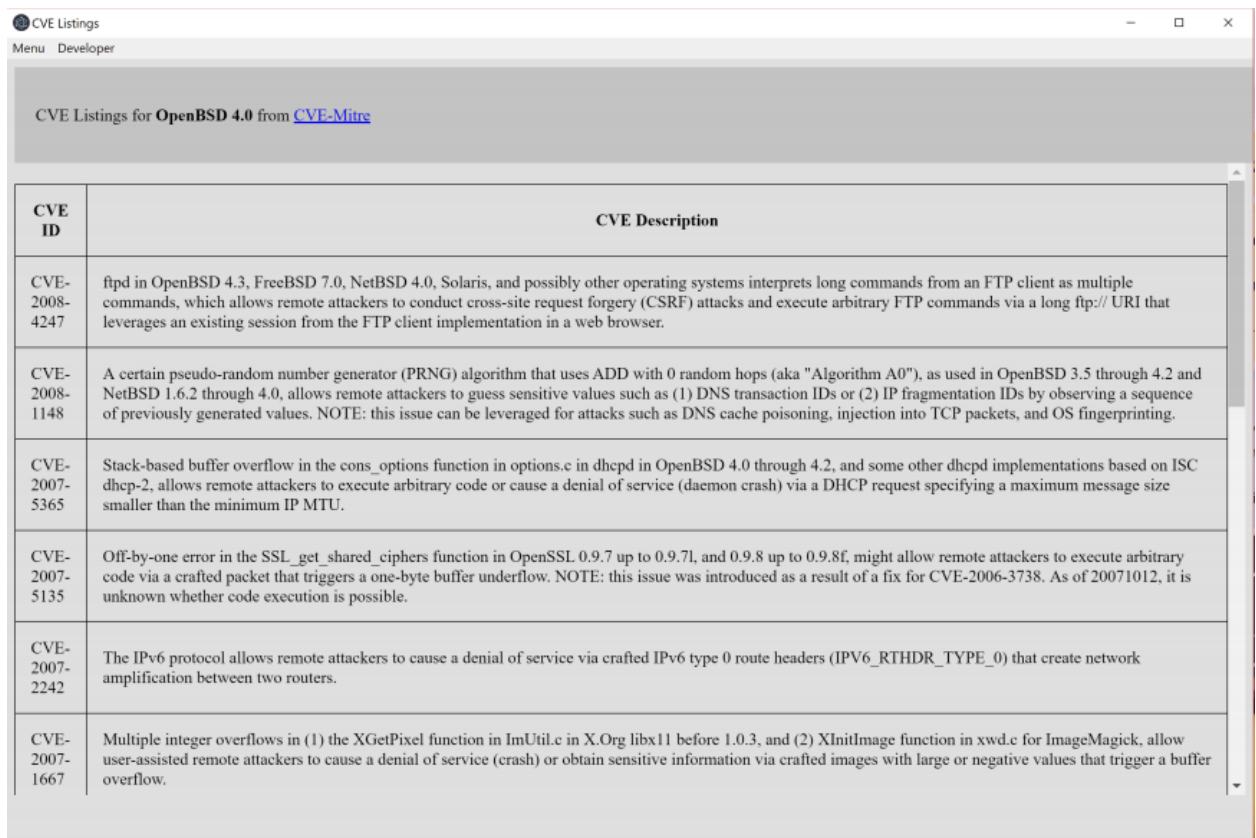


Рис. 5.4: Екран для списків CVE

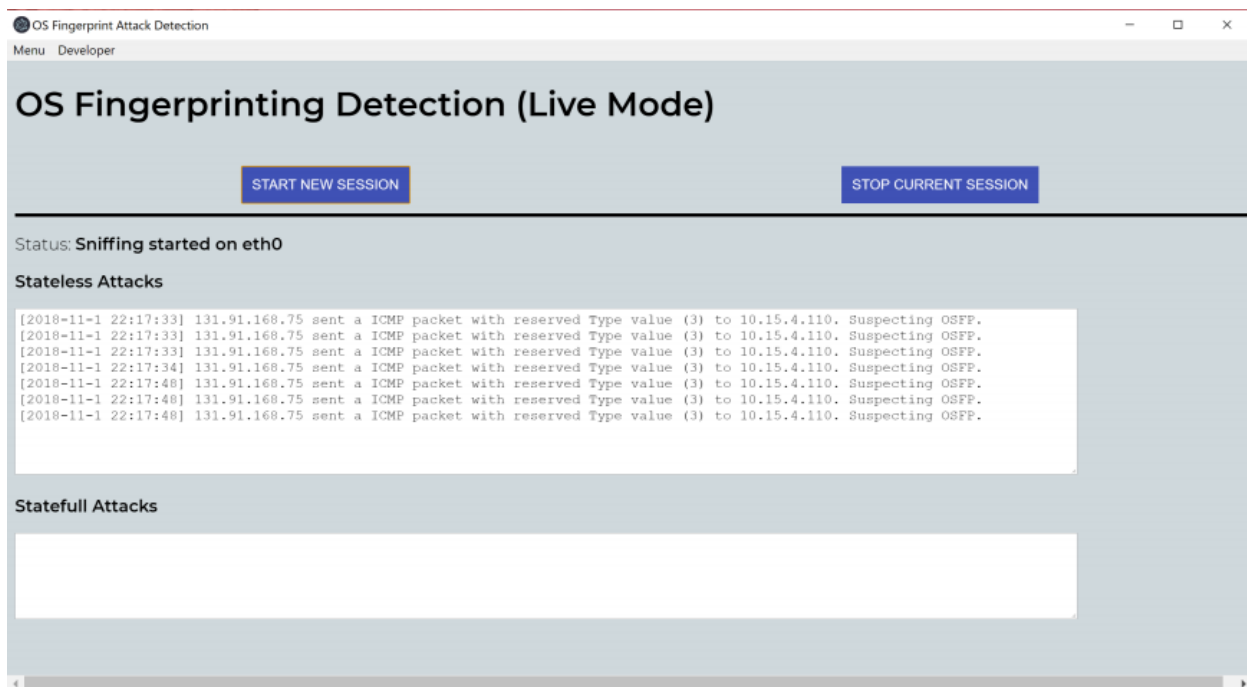


Рис. 5.5: Служба моніторингу атак у режимі реального часу на сесії

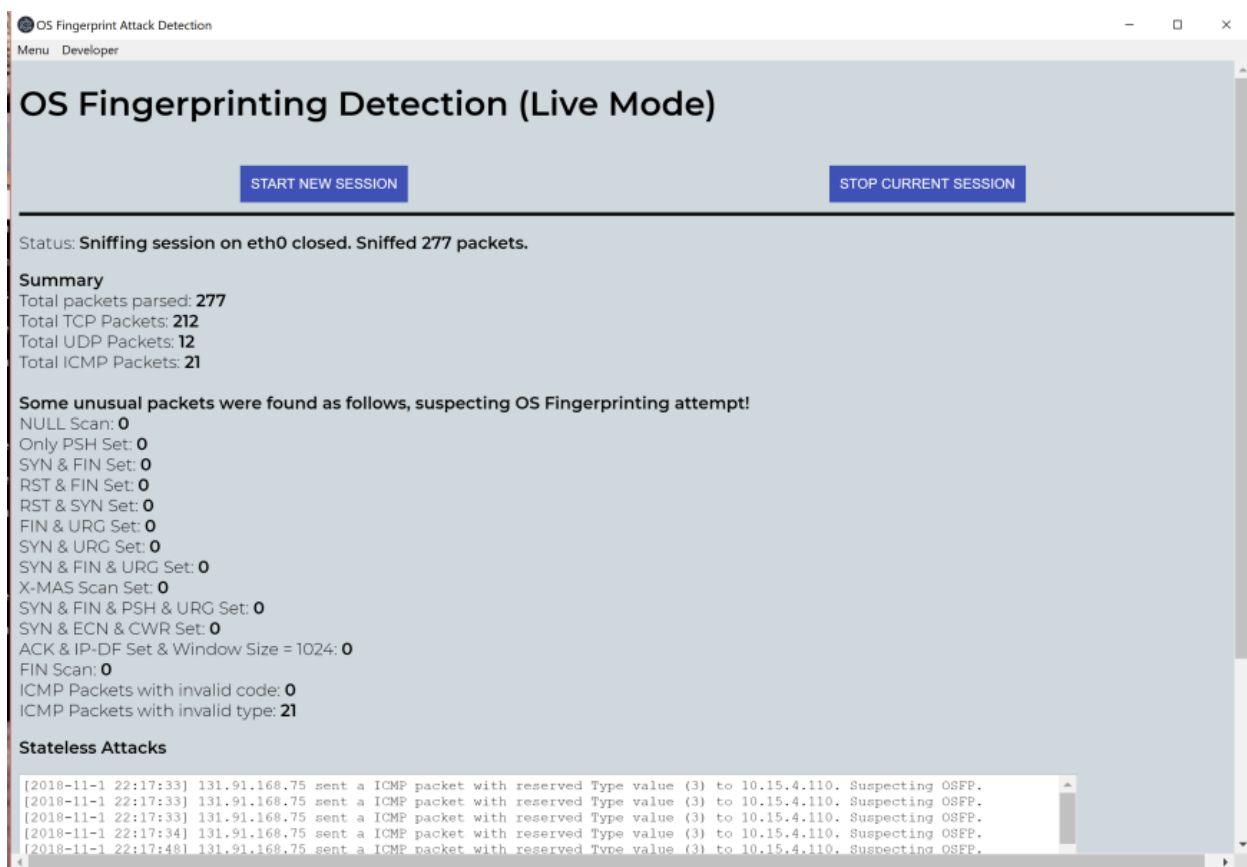


Рис. 5.6: Підсумок результатів після закінчення сеансу моніторингу атак

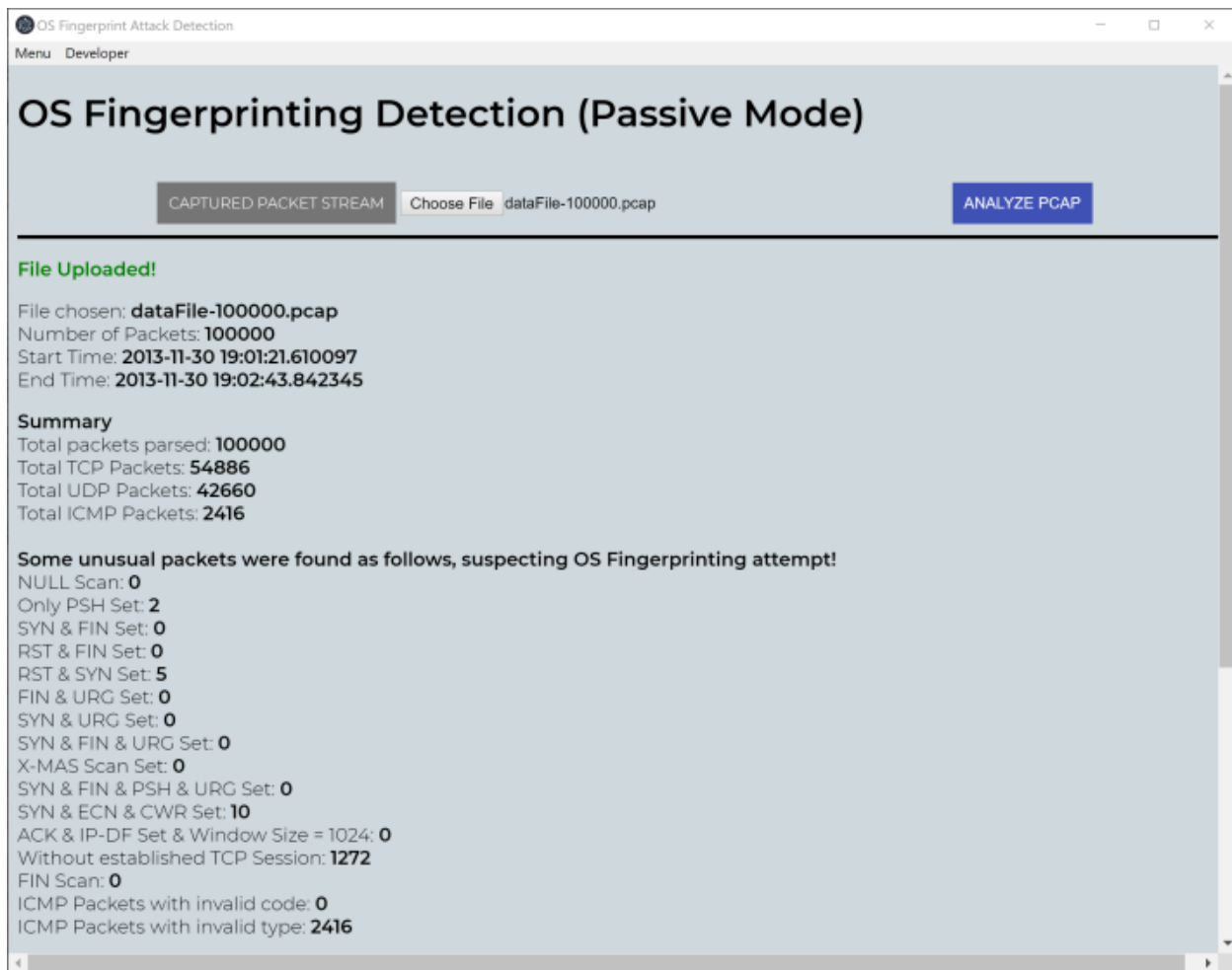


Рис. 5.7: Підсумок результатів аналізу файлу PCAP

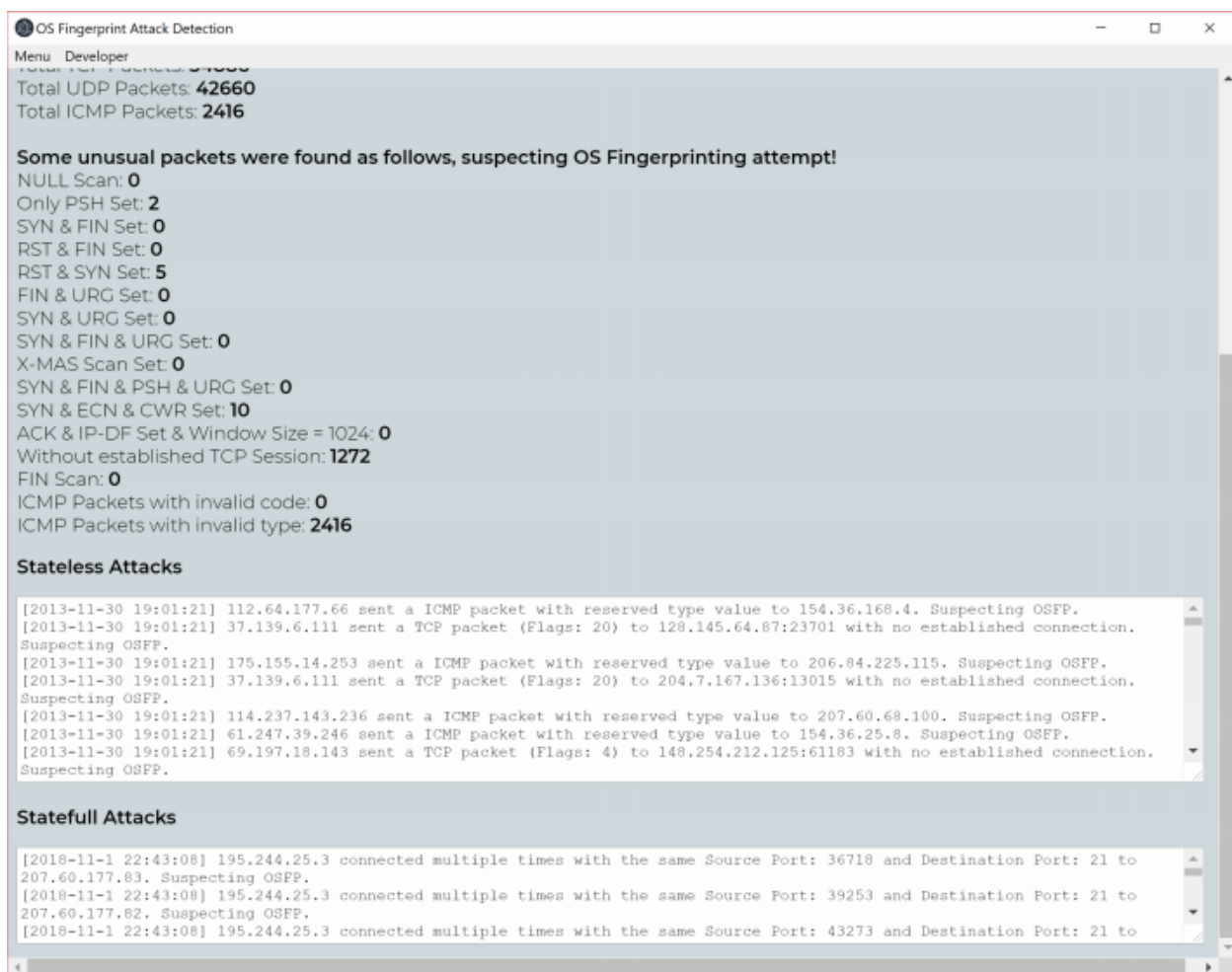


Рис. 5.8: Журнал повідомлень про атаки без громадянства та державного статусу

```

root@kali: ~
root@kali:~/Documents/msThesis/theBackEnd# npm start

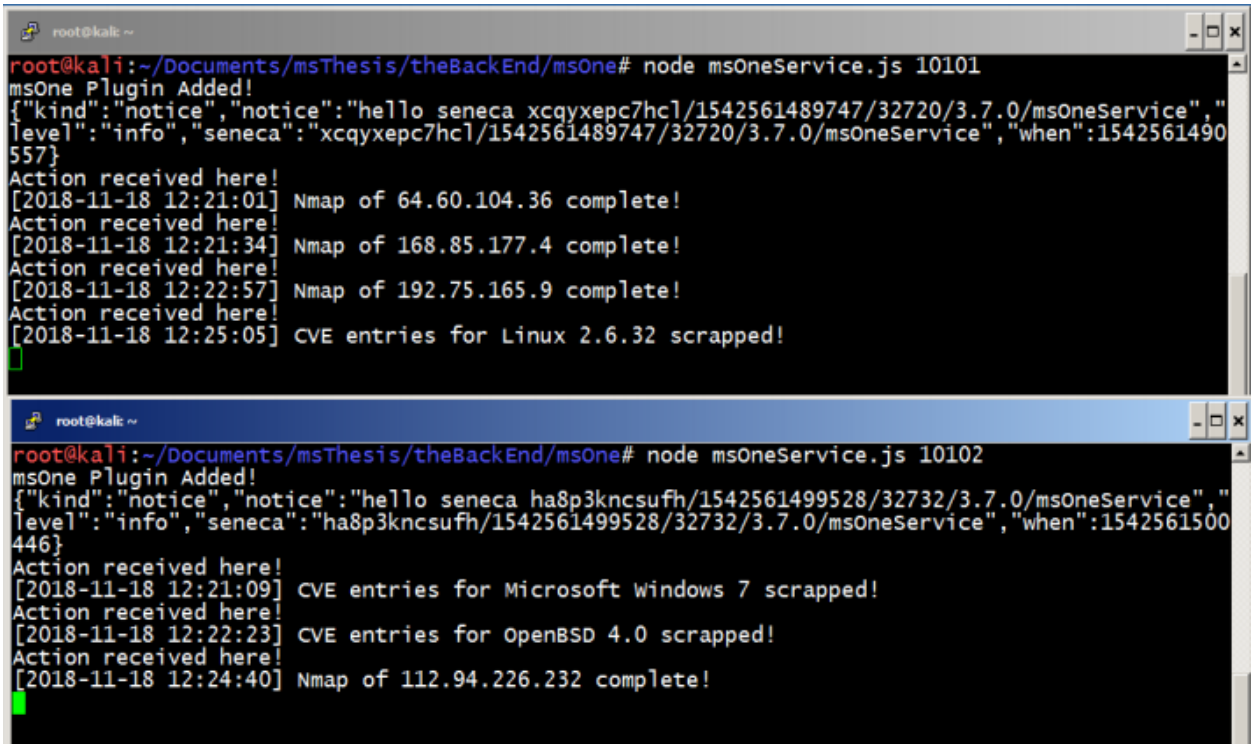
> thesis-app@1.0.0 start /root/Documents/msThesis/theBackEnd
> node index

{"kind":"notice","notice":"hello seneca umsvula24xxd/1541895793980/22177/3.7.0/expressAPI","level":"info","seneca":"umsvula24xxd/1541895793980/22177/3.7.0/expressAPI","when":1541895794149}
App started on port 80
[2018-11-10 19:24:27] msOne -> fpOS request completed.
[2018-11-10 19:24:45] msOne -> fpOS request completed.
[2018-11-10 19:25:40] msOne -> fpOS request completed.
[2018-11-10 19:27:31] msOne -> fpOS request completed.
[2018-11-10 19:27:46] msOne -> getCVEs request completed.
[2018-11-10 19:27:59] msOne -> getCVEs request completed.
[2018-11-10 19:28:06] msOne -> getCVEs request completed.
[2018-11-10 19:28:40] msTwo -> start request completed.
[2018-11-10 19:28:44] msTwo -> updateSLA request completed.
[2018-11-10 19:28:49] msTwo -> updateSLA request completed.
[2018-11-10 19:28:54] msTwo -> updateSLA request completed.
[2018-11-10 19:28:59] msTwo -> updateSLA request completed.
[2018-11-10 19:29:04] msTwo -> updateSLA request completed.
[2018-11-10 19:29:09] msTwo -> updateSLA request completed.
[2018-11-10 19:29:14] msTwo -> updateSLA request completed.
[2018-11-10 19:29:19] msTwo -> updateSLA request completed.
[2018-11-10 19:29:24] msTwo -> updateSLA request completed.
[2018-11-10 19:29:29] msTwo -> updateSLA request completed.
[2018-11-10 19:29:34] msTwo -> updateSLA request completed.
[2018-11-10 19:29:39] msTwo -> updateSLA request completed.
[2018-11-10 19:29:44] msTwo -> updateSLA request completed.
[2018-11-10 19:29:49] msTwo -> updateSFA request completed.
[2018-11-10 19:29:49] msTwo -> updateSLA request completed.
[2018-11-10 19:29:50] msTwo -> stop request completed.
[2018-11-10 19:30:46] File dataFile-100000.pcap uploaded and moved successfully.
[2018-11-10 19:30:49] msThree -> analyzePCAP request completed.

[API-Sessi0:npm* "kali" 19:31 10-Nov-18]

```

Рис. 5.9: Журнали з REST API



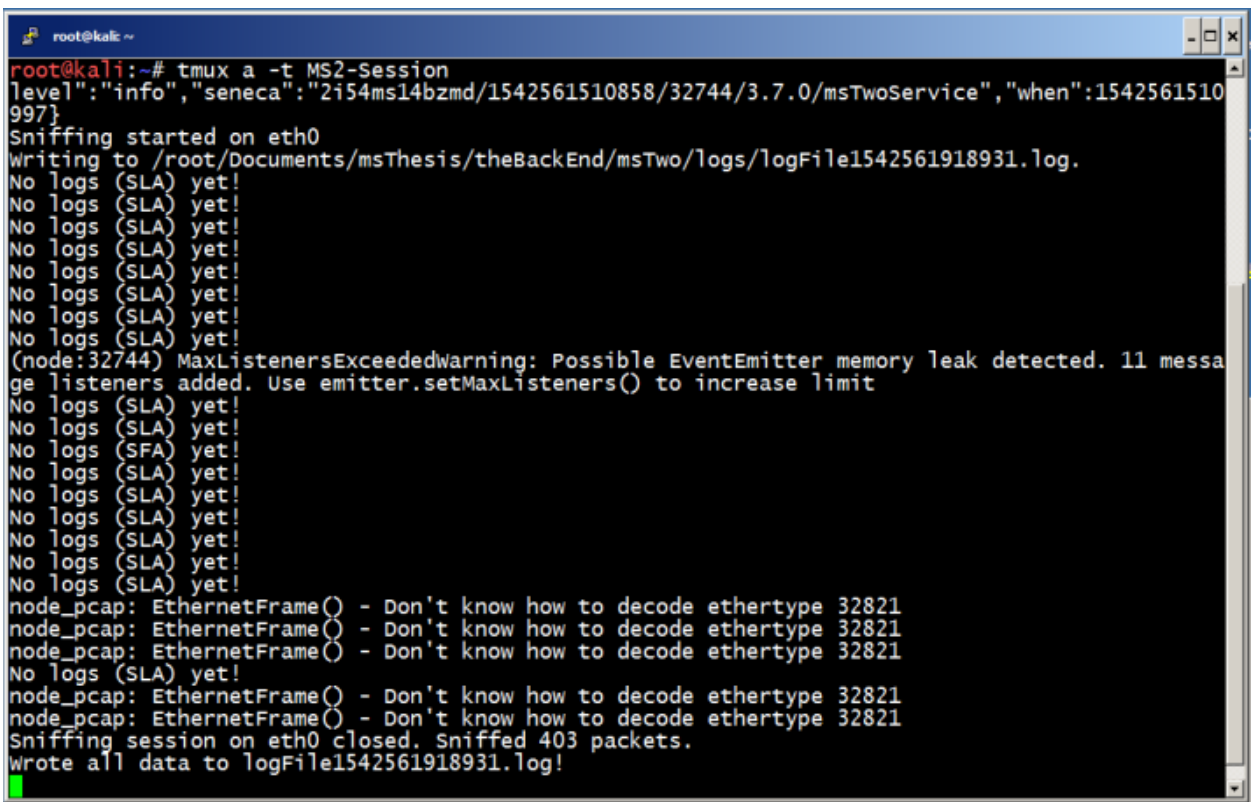
```

root@kali: ~/Documents/msThesis/theBackEnd/msOne# node msOneService.js 10101
msOne Plugin Added!
{"kind":"notice","notice":"hello seneca xcqyxepc7hcl/1542561489747/32720/3.7.0/msOneService","level":"info","seneca":"xcqyxepc7hcl/1542561489747/32720/3.7.0/msOneService","when":1542561490557}
Action received here!
[2018-11-18 12:21:01] Nmap of 64.60.104.36 complete!
Action received here!
[2018-11-18 12:21:34] Nmap of 168.85.177.4 complete!
Action received here!
[2018-11-18 12:22:57] Nmap of 192.75.165.9 complete!
Action received here!
[2018-11-18 12:25:05] CVE entries for Linux 2.6.32 scrapped!

root@kali: ~/Documents/msThesis/theBackEnd/msOne# node msOneService.js 10102
msOne Plugin Added!
{"kind":"notice","notice":"hello seneca ha8p3kncsufh/1542561499528/32732/3.7.0/msOneService","level":"info","seneca":"ha8p3kncsufh/1542561499528/32732/3.7.0/msOneService","when":1542561500446}
Action received here!
[2018-11-18 12:21:09] CVE entries for Microsoft Windows 7 scrapped!
Action received here!
[2018-11-18 12:22:23] CVE entries for OpenBSD 4.0 scrapped!
Action received here!
[2018-11-18 12:24:40] Nmap of 112.94.226.232 complete!

```

Рис. 5.10: Журнали служби оцінки відбитків пальців та оцінки вразливості ОС



```

root@kali: ~# tmux a -t MS2-Session
level":"info","seneca":"2i54ms14bzmd/1542561510858/32744/3.7.0/msTwoService","when":1542561510997}
Sniffing started on eth0
Writing to /root/Documents/msThesis/theBackEnd/msTwo/logs/logFile1542561918931.log.
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
(node:32744) MaxListenersExceededWarning: Possible EventEmitter memory leak detected. 11 message listeners added. Use emitter.setMaxListeners() to increase limit
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
No logs (SLA) yet!
node_pcap: EthernetFrame() - Don't know how to decode ethertype 32821
node_pcap: EthernetFrame() - Don't know how to decode ethertype 32821
node_pcap: EthernetFrame() - Don't know how to decode ethertype 32821
No logs (SLA) yet!
node_pcap: EthernetFrame() - Don't know how to decode ethertype 32821
node_pcap: EthernetFrame() - Don't know how to decode ethertype 32821
Sniffing session on eth0 closed. Sniffed 403 packets.
Wrote all data to logFile1542561918931.log!

```

Рис. 5.11: Журнали служби моніторингу атак

```

root@kali: ~
root@kali:~/Documents/msThesis/theBackEnd/msThree# node msThreeService.js 10104
msThree Plugin Added!
{"kind":"notice","notice":"hello seneca w2wd2mnj4qny/1542561521080/32756/3.7.0/msThreeService",
"level":"info","seneca":"w2wd2mnj4qny/1542561521080/32756/3.7.0/msThreeService","when":154256
1521192}
Analyzed 100000 packets.
Response: Analyzed 100000 packets.

```

Рис. 5.12: Журнали служби аналізу PCAP

```

root@kali: ~
top - 19:36:29 up 35 days, 4:54, 7 users, load average: 0.00, 0.00, 0.00
Tasks: 152 total, 1 running, 151 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.1 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1991.9 total, 257.9 free, 594.2 used, 1139.7 buff/cache
MiB Swap: 2045.0 total, 2017.5 free, 27.5 used. 1205.4 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 30244 root        20   0  965.4m  46.7m  22.7m S   0.3   2.3   0:01.82 node msOneService.js 10101
 30278 root        20   0  951.0m  42.6m  22.4m S   0.3   2.1   0:00.75 node msThreeService.js 10104
 30255 root        20   0  965.9m  48.9m  22.7m S   0.0   2.5   0:01.63 node msOneService.js 10102
 30267 root        20   0  951.0m  42.7m  22.5m S   0.0   2.1   0:00.73 node msTwoService.js 10103

```

Рис. 5.13: Використання оперативної пам'яті мікросервісами відразу після запуску

```

root@kali: ~
top - 19:59:24 up 35 days, 5:17, 7 users, load average: 0.05, 0.03, 0.00
Tasks: 155 total, 1 running, 154 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.2 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 30278 root        20   0  951.3m  39.7m  22.5m S   0.3   2.0   0:02.26 node msThreeService.js 10104
 30244 root        20   0  965.9m  49.3m  24.4m S   0.0   2.5   0:03.49 node msOneService.js 10101
 30255 root        20   0 1178.5m  54.6m  24.6m S   0.0   2.7   0:04.72 node msOneService.js 10102
 30267 root        20   0  951.5m  41.8m  22.7m S   0.0   2.1   0:02.26 node msTwoService.js 10103
 30309 root         0    0    2.3m   0.8m   0.7m S   0.0   0.0   0:00.00 sh -c node index
 30310 root        20   0 1147.8m  48.5m  24.4m S   0.0   2.4   0:04.11 node index

```

Рис. 5.14: Використання оперативної пам'яті мікросервісами після обслуговування запитів

6 ОБГОВОРЕННЯ

У цьому розділі ми розпочнемо з опису фактичного сценарію використання запропонованої основи мікросервісів. Після цього ми приймаємо та перераховуємо деякі обмеження роботи, проведеної в цій дипломній роботі. Щоб проілюструвати практичне використання основи, яку ми вже обговорювали, ми представляємо правдоподібну реалізацію, як показано на малюнку 6.1. На малюнку ми показуємо типову мережу лікарні з її системою IoT та те, як пропонується запропонована нами структура. Подібно до архітектури IoTSec, представленої в [39] Ю та ін., Ми передбачаємо, що можемо використовувати наш фреймворк як мережеве рішення безпеки. Окремі мікросервіси можуть бути розроблені для взаємодії з певним типом пристрою IoT та моніторингу потоку мережевого трафіку через нього. Кожна з цих служб захоплює та аналізує всі пакети даних, які передаються через певний пристрій. Використовуючи за шаблонами, отриманими з підрозділу збору підписів атак (див. розділ 3.2), мікросервіси можуть аналізувати захоплені пакети та спостерігати за відповідними атаками. Мікросервіси будуть розміщені на централізованому підрозділі моніторингу мережі, який працює на внутрішньому сервері, безпосередньо підключеному до основного мережевої точки, маршрутизатор. Оскільки пристрій знаходиться в тій самій мережі, що і пристрої IoT, мікросервіси зможуть отримати доступ до мережевого трафіку за допомогою фільтрів для кожного пристрою. Блок також матиме шлюз API для обміну даними з окремими мікросервісами. У цьому ж блоці ми можемо також мати інформаційну панель (як Клієнт у нашому додатку), яка може бути використана адміністратором мережі для перегляду результатів з мікросервісів.

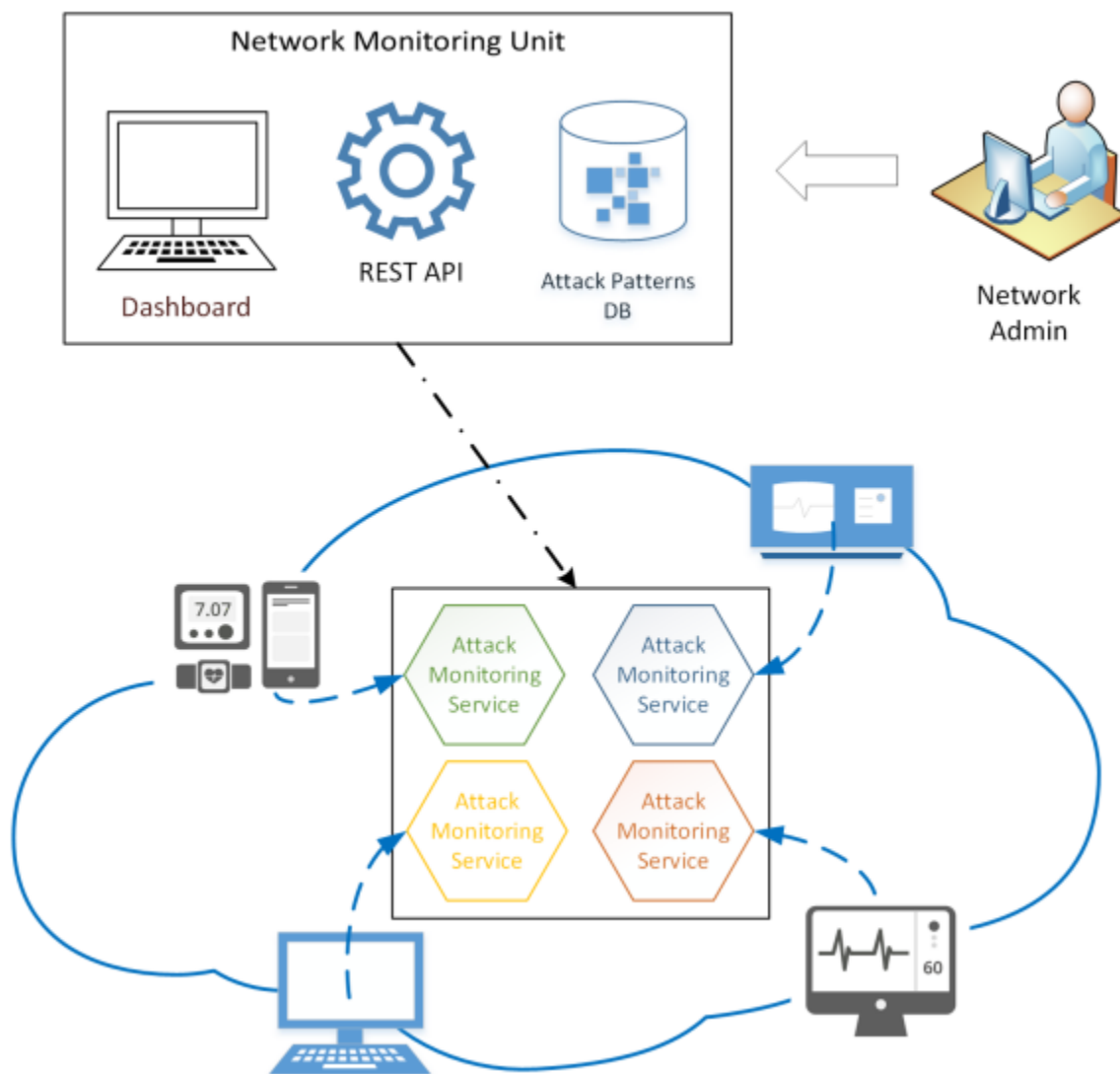


Рис. 6.1: Імовірний сценарій використання запропонованої основи

Запропонована нами структура дозволяє ІТ-команді лікарні впроваджувати механізми моніторингу для конкретних пристроїв і навіть для конкретної ОС. Однак необов'язково, щоб мікросервіси розроблялися лише ІТ-командою лікарні. Навіть ІТ-компанії можуть розробити такі індивідуальні мікропослуги для стандартних пристроїв IoT, які використовуються в багатьох лікарнях. У цьому випадку лікарні можуть придбати ці готові мікропослуги та використовувати їх, як пакет "підключи і працюй". Це можливо, оскільки мікросервіси розгортаються за допомогою контейнерів, що забезпечують портативність. ІТ-команді лікарні потрібно лише увімкнути необхідний

контейнер, а контейнер також піклується про залежності бібліотеки та фреймворку. Тепер ми перерахуємо деякі проблеми, які існують із запропонованим нами підходом:

- Оскільки фреймворк є програмним захисним механізмом, він може виявити додаткові вразливості, що існують у самому програмному забезпеченні. Прикладом таких уразливостей є дефекти кодування, нерегулярні оновлення програмного забезпечення, вразливі бази даних та веб-сервери тощо.
- Успішна реалізація підходу до мікросервісів вимагає глибокого розуміння області застосування, щоб визначити найкращий обсяг кожної послуги. Ми лише подряпали поверхню проблем кібербезпеки з пристроями IoT. Щоб створити розширене рішення, потрібно більше знань у домені про те, як відстежувати всі зв'язки до пристрою IoT та з нього та виявляти всі шкідливі дії.
- Рішення має невід'ємну складність. Існує потреба у багатьох розробниках та технологіях через відсутність загального середовища програмування або операційної системи на пристроях IoT.
- Зловмисники завжди шукають нових та винахідливих способів атакувати пристрої, не торкаючись їх фізично. Отже, для постійного оновлення моніторингу будуть потрібні часті оновлення. Щоб вирішити цю проблему, ми задумали вести окрему базу даних шаблонів. Але все-таки типи пристроїв IoT, доступних на ринку, - це не малий ринок. Таким чином, ми очікуємо зіткнутися зі складністю за допомогою цього підходу.
- Ми не мали доступу до випробувального стенду пристроїв IoT, на якому ми могли б проводити експерименти. Виконання зловмисних дій в Інтернеті, навіть для дослідницьких цілей, вимагає схвалення / дозволів та більшої кількості ресурсів, як ізольоване та безпечне середовище. Через обмежений

час та ресурси ми змогли протестувати лише спроби відбитків пальців ОС на нашому хостинговому сервері та наборі даних darknet. Отже, шаблонів, які ми використовували, достатньо лише для ідентифікації спроб відбитків пальців ОС, і нам не вдалося знайти шаблони, які вказують на будь-які інші види атак на конкретні пристрої IoT. Архітектура мікросервісів дозволяє використовувати програмування поліглотів. Однак у своїй роботі ми використовували лише JavaScript. В результаті, хоча JavaScript є однією з найбільш широко використовуваних мов, ми все ж дотримувались деяких обмежень під час його використання:

- Ми виявили, що JavaScript не є найбільш підходящою мовою, коли мова йде про мережеве програмування. Ми не змогли знайти жодного способу в JavaScript, що дозволило нам надсилати спеціальні створені пакети до віддалених цілей і фіксувати / аналізувати відповідні відповіді. Ось чому ми використовували пасивний підхід для відбитків пальців ОС, замість активного. Для обнюхування та аналізу пакетів, захоплених на нашому сервері, ми використовували пакет rpsar, доступний з npm. Для синтаксичного аналізу даного файлу PCAP ми використовували пакет rpsar-parser, доступний з npm.
- Для активних відбитків пальців ОС також потрібна багатопотоковість, один потік для відправки пакетів до цілі, а інший потік, який може одночасно фіксувати та аналізувати відповіді. Але NodeJS - це парадигма, заснована на подіях, тому ми виявили, що багатопотоковість із доступом до спільних даних була занадто складною для вивчення та використання протягом часового циклу дисертації. Ось чому ми вирішили використати пасивний підхід для Fingerprinting OS.
- Також необхідна багатопотоковість для синтаксичного аналізу та аналізу захоплених пакетів для атак із станом. Отже, для вирішення проблеми ми

зібрали необхідну інформацію в структурі словника, і вона була передана дочірньому процесу, який виконує сценарій аналізу після повного аналізу. Ми також думали про іншу реалізацію запропонованої нами основи. Реалізація показана на малюнку 6.2. Реалізація схожа, за винятком того, що тут ми розглядаємо мікросервіси, що розгортаються на самому пристрої IoT, як додаткову функціональність. Однак ми зрозуміли, що це впровадження неможливо. Це пояснюється тим, що більшість пристроїв IoT використовують однопоточний мікроконтролер (8051, MSP430, серія ATMEGA) із оперативною пам'яттю менше 2 Мб [39]. Розроблені нами мікросервіси мають значно більше використання оперативної пам'яті (50 МБ), що, очевидно, недостатньо мало, щоб його можна було розгорнути на самому пристрої IoT. Що ще важливіше, постачальники IoT можуть не надавати доступ третім сторонам для розгортання будь-якого додаткового програмного забезпечення на своїх пристроях. Ми представляємо цю реалізацію для кращого розуміння читачів.

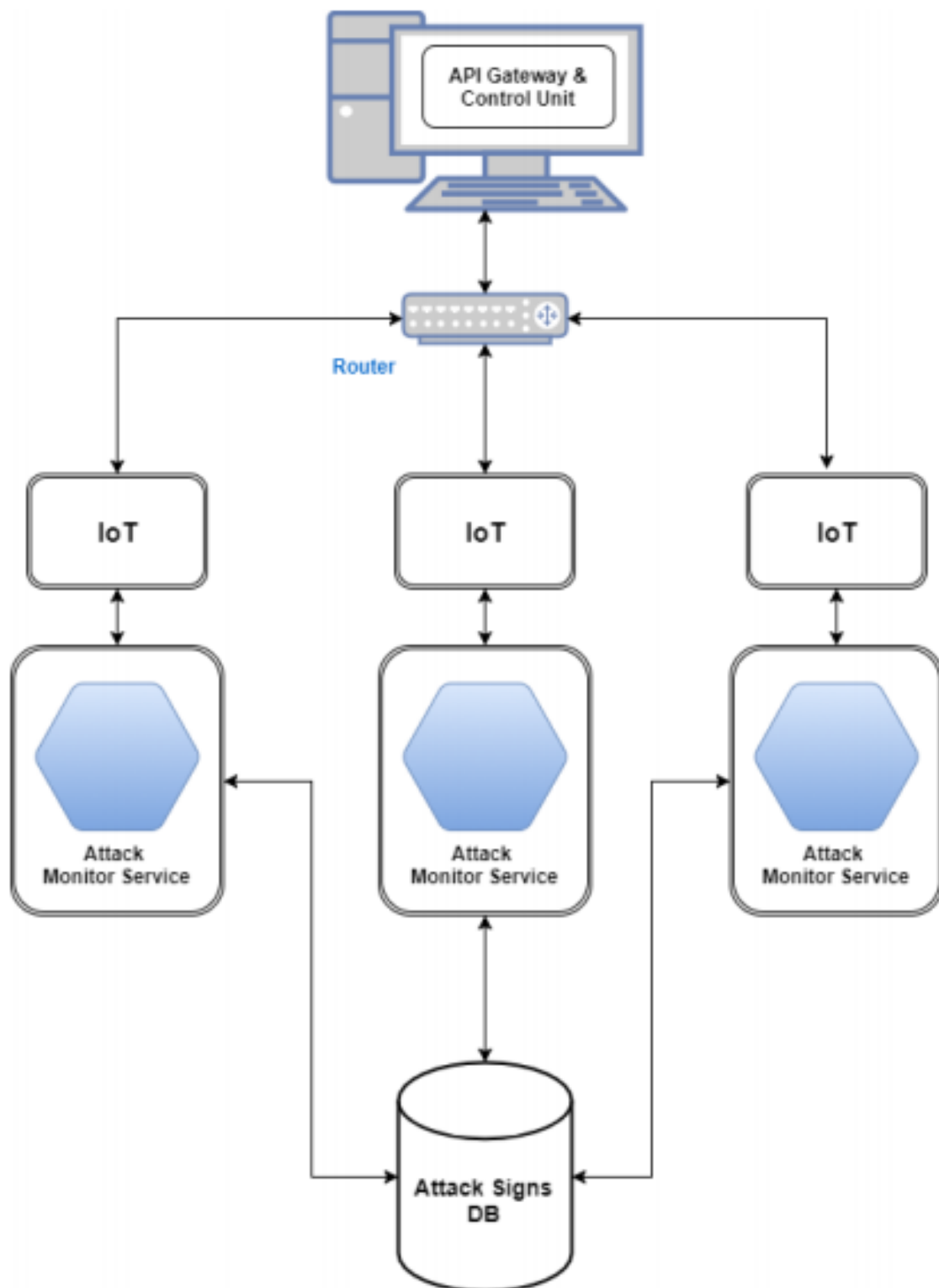


Рис. 6.2: Альтернативна реалізація, неможлива для сучасних IoT

7 ВИСНОВКИ ТА МАЙБУТНЯ РОБОТА

7.1 Майбутня робота

Існує декілька способів розширення запропонованої системи.

- З метою демонстрації ця теза обмежила сферу атак кібербезпеки лише відбитками пальців ОС. За допомогою експертів у галузі кібербезпеки, структуру можна розширити, розробивши мікросервіси з іншими функціями, пов'язаними з безпекою, такими як виявлення шкідливого програмного забезпечення, моніторинг вразливостей програмного забезпечення та дефекти кодування, несанкціонований доступ тощо.
- Завжди є місце для вдосконалення існуючих мікропослуг, або шляхом застосування оптимізації, або подальшого розкладання існуючих послуг. Прикладом декомпозиції може бути, коли Служба оцінки відбитків пальців та оцінки вразливості ОС розділена на 2 служби, одна, яка надає відбитки пальців ОС, і інша, яка отримує список вразливостей, характерних для ОС. Обидва мікросервіси можуть бути зв'язані між собою, тому, коли відбитки пальців ОС завершуються, вони надсилають ідентифіковану ОС Службі оцінки вразливості, яка отримуватиме вразливості з веб-сайту CVE.
- Структуру можна додатково вдосконалити, використовуючи окремий словник правил моделей атак. Потім словник стає пошуковою таблицею, яку експерт з кібербезпеки може оновити, не розбираючи та не вивчаючи весь код програми для шаблонів. Ця функціональність забезпечує розділення турботи між розробниками та експертами доменів. У нашій існуючій реалізації схеми атак вбудовані в один і той же код із реалізацією мікросервісів. Ми плануємо включити цю функцію в майбутньому, і оновлена версія буде доступна через сховище GitHub.

- Цей проект використав лише одну мову програмування, JavaScript. Оскільки мікросервіси дозволяють використовувати програмування поліглот, цікаво буде знайти різницю в продуктивності, коли інші мікросервіси будуються з використанням інших мов. Наприклад, C і C++ мають потужні бібліотеки для розпізнавання та аналізу пакетів. Використовуючи інші мови, ми також можемо досягти менших розмірів.
- У наших існуючих мікросервісах ми контролюємо та аналізуємо всі цілі індивідуально. Однак цікавим завданням дослідження було б визначити, чи використовуються однакові схеми атак на групу цілей. Це може допомогти у виявленні спільних атак з того самого джерела.
- Також можна додати компонент машинного навчання до фреймворку, використовуючи потужність мов Python / R. Існує багато існуючих досліджень щодо виявлення вразливостей та ідентифікації атак на основі алгоритмів машинного навчання зі значно вищою точністю.

7.2 Висновок

У цій дипломній роботі ми запропонували структуру, яка використовує шаблон архітектури мікросервісів, щоб забезпечити адаптивне та гнучке рішення з кібербезпеки для зменшення ризику атак на вразливі пристрої IoT. Ми показали його застосування, розробивши додаток на основі мікропослуг із трьома мікропослугами, кожна з яких обслуговує певну функціональність. Будучи вирішальним кроком у розвідці мережі, дисертація зосередила свою увагу на відбитках пальців ОС. Ми виявили, що розроблені нами мікросервіси мають невеликий розмір із використанням оперативної пам'яті приблизно 50 МБ. Ми також намагалися підвищити обізнаність щодо вразливості пристроїв IoT, особливо тих, що використовуються у галузі охорони здоров'я, та про те, як вони забезпечують легкий доступ із внутрішньої двері до мережі організації, що призводить до збільшення кількості атак на пристрої IoT.

Беручи до уваги обговорені обмеження, ми все ще вважаємо перспективним підхід до мікросервісів для побудови рішень з кібербезпеки. Для вирішення проблем, що спостерігалися під час цієї дипломної роботи, ми закликаємо до подальших зусиль.

8 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

8.1 Опис ідеї проекту

У цьому розділі описується економічне обґрунтування реалізації попереднього проекту на тему "Використання мікропослуг в охороні здоров'я". Описана технологія застосовуватиметься у випадку веб-програми, тому її можна використовувати в будь-який час. Процес включає:

- Реалізація програми за цією назвою роботи;
- Розробити стратегію розвитку продукту, яка здатна конкурувати з іншими на галузевому рівні.

Мета цього розділу - створити творчу ідею, оцінити ринкові очікування та ділові можливості для бізнесу та наукових досягнень та визначити ключові підходи, розроблені в першій частині дослідження, для проектування на економічно конкурентних ринках. Шлях до зростання.

Таблиця 8.1 - суть ідеї стартапу

Зміст ідеї	Напрямки застосування	Переваги для користувача
Додаток, що зможе допомогти захитити данні користувачів системи	запровадити новий підхід, який використовує основу мікропослуг як адаптивне та гнучке рішення безпеки для вирішення проблеми з кібератаками	Захищенні данні користувачів системи.

	Полегшення роботи мед. працівників	розуміння вразливостей медичних пристроїв IoT
--	------------------------------------	---

Тому додаток можна використовувати як сумісне та гнучке рішення безпеки для вирішення проблеми кібер-атаки.

Таблиця 8.2 - Нейтральні, слабкі та сильні характеристики ідеї застосунку

№ п \ п	Техніко- економічні характеристики	Потенційні конкуренти				W слабка сторон а	N нейтра льна сторон а	S сильна сторон а
		Мій прое кт	К-т 1	К-т 2	К-т 3			
1.	Форма виконання	Веб-сервіс, додаток	Веб-сервіс	Додаток	Додаток			+
2.	Собівартість	Низька	Середня	Висока	Висока			+
3.	Наявність адміністратора	Обов'язкова	Обов'язкова	Не обов'	Обов'язкова		+	

	для налаштування			язков а				
4.	Наявність інтернету	Не треб а	Треба	Треба	Треба	+		
5.	Крос- платформеність	Так	Ні	Так	Так			+

Було визначено спеціальний перелік слабких, сильних та нейтральних особливостей та потенційну концепцію товару, які є основою для цього порівняння. Сила цього проекту полягає в тому, що програма встановлена, і для її використання не потрібно підключатися до Інтернету, оскільки доступ надається з локального пристрою.

Відсутність особистої веб-програми та адміністратора необхідні для контролю належної роботи програми.

До недоліків відноситься необхідність користуватися Інтернетом, але така поведінка забезпечується суб'єктом проекту та запобігає попередньому завантаженню інформації на локальній машині.

8.2 Технічний аудит ідеї проекту

Обмеження цього розділу необхідні для розрахунку технології, за допомогою якої ви можете перевірити концепцію проекту (техніка створення продукту).

Таблиця 8.3 - технологічний аудит проекту

№ п\п	Ідея проекту	Технології	Наявність технології	Доступність технології
1	Створення веб-сервісу	JS	Наявна	Безкоштовна, доступна
		Rest API	Наявна	Безкоштовна, доступна
Для створення веб-сервісу обрані технології (JS, Rest API), які є безкоштовними, доступними та добре дослідженими потенційними розробниками.				

У таблиці 5.3 наведені попередні результати основних технологічних пакетів, які можуть бути використані для реалізації первинної реалізації проекту, описаної вище. Слід вибрати технологічне обладнання, яке не потребує додаткового обслуговування та вартості.

8.3 Аналіз ринкових можливостей

Визначення ринкових можливостей, які ринок може використовувати під час проекту, та ринкових ризиків, які можуть порушити проект, не зважаючи на ринкове середовище, майбутні потреби споживачів та прогнозування конкуренції. Дозволяє планувати.

Ми аналізуємо потреби, такі як: наявність потреб, обсяг, діяльність з розвитку ринку.

Таблиця 8.4 - Базова характеристика ринку стартапу

№ п\п	Показники ринку	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум. од	10000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (характер)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Конфіденційність оброблюваних даних
6	Середня норма рентабельності в галузі (або по ринку), %	$R = (3000000 * 100) / (1000000 * 12) = 25\%$

Таким чином, середня промислова віддача нижча за інтерес банку до інвестицій. Отже, немає обмежень щодо інвестицій, доступу до ринку та вимог до якості та сертифікації цього проекту, як це було б застосовано у випадку мережевої послуги.

Наступним кроком є огляд груп потенційних користувачів, створення їх функцій та списків, які визначають вимоги до продуктів для кожної групи.

Таблиця 8.5 - Характеристика потенційних клієнтів

№ п\п	Формуюча ринок потреба	Цільова аудиторія	Відмінності у поведінці різних цільових груп	Вимоги споживачів до продукту
1	Система сервісів, що захищають данні користувачів системи.	Міністерств о охорони здоров'я, лікарні	Цільова група займається захистом данні користувачів системи	Зручний інструмент для захисту системи

Визначено характеристики початкового проекту; найбільша потреба у формуванні ринку – це веб-сервіси; представлені основні цільові сегменти ринку – лікарні; різноманітність поведінки цільових груп різних клієнтів – захист системи; основні потреби від замовника – захист системи.

Таблиця 8.6 - Аналіз загроз

№ п\п	Фактор	Зміст загрози	Можлива реакція компанії
1	Динаміка ринку	Уповільнення росту ринку	Співпраця з іншими компаніями
			Розширення на інші ринки
2	Конкуренція	Вихід на ринок великої компанії	Акцентувати увагу на надійність нашої системи

3	Технології	Поява нових технологій	Програма передбачає розширення, потрібно розробити новий функціонал
4	Держава	Збільшення податків	Оптимізація діяльності для скорочення витрат

Стартапам може загрожувати ринкова діяльність, поява нових конкурентів, впровадження нових технологій та зміни державної політики.

Таблиця 8.7 - Фактори можливостей

№ п\п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання на ринку покупців	Поява великої кількості клієнтської бази, які готові та бажають використовувати новий продукт	Вплив на нових клієнтів через рекламну компанію
2	Зниження ефективності конкурента 1	Конкурент 1 втрачає жагу до роботи	Переконання покупців в ефективності нашого веб-сервісу

3	Зменшення витрат на технічну підтримку	Збільшення продуктивності роботи штату компанії за рахунок підвищення їхнього професійного рівня	Підвищувати рівень кваліфікації своїх співробітників
---	--	--	--

У наведеній вище таблиці ми наводимо приклади можливих причин збільшення нашої продуктивності, таких як: Збільшення частки ринку покупця, зменшення ефективності впливу конкурента, зменшити вартість технічної підтримки.

Далі проведемо аналіз пропозицій:

Таблиця 8.8 - Ступеневий аналіз конкуренції на ринку

Назва характеристики	Особливість конкурентного середовища	В чому проявляється характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип	Досконале	Існують фірми конкуренти	Підрахувати ціни на конкурентні товари, вирахувати спроможність пропонувати якісні товари на більш вигідніших умовах

Рівень конкурентної боротьби	Міжнародне	Є зарубіжний конкурент	Додати у веб-сервіс можливість вибору мови для привернення уваги зарубіжних користувачів
Галузева ознака	Фокус на галузі	Усі конкуренти займаються однією справою	Створити таке ПЗ, щоб можна було легко переробити його під час розширення або перехід до нових технологій
Вид товарів	Товарно-видове	Однаковий вибір запропонованих послуг і товарів, програмного забезпечення	Створити ПЗ, у якому враховані мінуси конкурентів
Характер конкурентних переваг	Нецінове	Вдосконалення технології створення ПЗ, для зменшення його собівартості	Використання ефективних та не дорогих технологій
Інтенсивність	Немарочне	Бренди відсутні	-

Аналізувались оцінки конкурентоспроможності ринку. Ми змогли з'ясувати, що конкуренція є світовим класом, і конкуренти мають сильну позицію. Це потрібно враховувати при оцінці нашої вартості. Ми також виявили, що конкуренти спеціалізуються на певних галузях, які можуть бути частиною нашого виробництва.

Далі було розроблено перелік факторів ринкової конкуренції на основі аналізу складових 5-сильної моделі М. Портера.

Таблиця 8.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Фактор сили постачальників	Фактор сили споживачів	Фактор загроз з боку товарів-замінників
Висновки	Існує 3 конкуренти на ринку. Найбільш схожим являється конкурент 1, має відсутність потреби	Так, можливості для входу на ринок є, бо наше ПЗ дає можливість пришвидшення вирішення задач лінійної алгебри	Постачальники відсутні	Важливим для користувача є зручність у користуванні та можливість використовувати	Товаризації можуть використати більш дешеву технологію створення додатку та зменшити

	інтернету та адміністрат ора			увати без інтернету	собівартіст ь товару
--	---------------------------------------	--	--	------------------------	-------------------------

Тому з конкурентної точки зору можна впевнено стверджувати, що проект має можливість працювати на ринку, оскільки розроблене рішення спрощує та пришвидшує роботу експерта. Ви можете визначити ключові особливості товару, які допоможуть вам бути конкурентоспроможними на ринку - надійність, простота використання, простота використання та безпека даних.

На основі аналізу вищезазначених рішень визначається та уточнюється перелік причин конкуренції.

Таблиця 8.10 - Обґрунтування факторів конкурентоспроможності

№ п\п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	У майбутньому реалізація веб додатку	Дозволить використовувати службу тільки при наявності веб браузеру
2	Простота користувацького інтерфейсу	Доступ відбувається через веб-сайт або додаток на ПК зі схожим інтерфейсом

Основними конкурентами, представленими на ринку, є: введення двох платформ, з якими розвиток взаємодії з користувачем не викликає зайвих проблем, а отже, не має ключової складової програма змінилася, простота користувача дозволяє програмі навчитися.

Сильні та слабкі сторони першого проекту аналізуються відповідно до конкурентних факторів.

Таблиця 8.11 - Порівняльний аналіз сильних та слабких сторін проекту

№ п\п	Фактор конкурен-тоспроможності	Бали 1-20	Рейтинг товарів конкурентів у порівнянні з нашим						
			-3	-2	-1	0	+1	+2	+3
1	Реалізація як локального додатку та веб сайту	20	+						
2	Простота користувацького інтерфейсу	19			+				

Подано порівняльний аналіз потенціалу проекту щодо конкурентних результатів та нашої компанії. Найвищі оцінки були отримані в конкурентних цілях - наявність платформ двох типів, як мережових, так і локальних.

Таблиця 8.12 - SWOT-аналіз стартапу

S	Простий користувацький інтерфейс, надійність, безпечність	Необхідність доступу до інтернету	W
O	Падіння конкурентоспроможності конкурента, успішна реклама з акцентом на надійність	Збільшення конкуренції, зміна пріоритетів користувачів, податки	T

Таблиця SWOT допомогла нам зібрати плюси і мінуси, щоб ви почали. Позитивні: слабкі сторони зв'язку, безпека та надійність: потрібні робочі запити. Можливості: реклама та помилки конкурентів. Ризики: конкурентний перерозподіл, спад ринку.

На основі SWOT-аналізу було розроблено іншу реалізацію для першого проекту.

Таблиця 8.13 - Альтернативи ринкового впровадження стартап-проекту

№ п\п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
-------	---------------------------------	--------------------------------	-------------------

1	Створення ПЗ використовуючи машинне навчання	88%	3 місяців
2	Створення ПЗ використовуючи власні методи	42%	16 місяців

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими.

8.4 Розробка ринкової стратегії проекту

Перший крок у розробці маркетингової стратегії передбачає визначення ринкової стратегії страхування: опис цільових груп майбутніх клієнтів.

Таблиця 8.14 - Вибір цільових груп потенційних користувачів

№ п\п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовн ий попит в межах цільової групи	Інтенсивні сть конкуренці ї в сегменті	Простота входу в сегмент
1	Компанії по плануванню виробництва	Можливість автоматичної категоризації , що зменшує	Великий	Існує 3 конкурент и, які надають такіж самі	Швидкодія , зручний користувац ький інтерфейс,

		роботу людей		товари, але менш якісні	точність класифікац ії
2	Статистичні центри	Можливість автоматичної категоризації , що зменшує роботу людей	Великий		Швидкодія , зручний користувац ький інтерфейс, точність класифікац ії
Які цільові групи обрано: компанії по плануванню виробництва та статистичні центри					

Результати розгляду потенційних цільових груп користувачів для першого проекту наведені в таблиці 5.14 і дозволяють перевірити переваги продукту для використання на цей сегмент ринку, а також чи доцільно витратити кошти на джерела впливу певної групи.

На основі результатів аналізу потенційних груп користувачів (категорій) було обрано дві основні цільові групи, якими буде запропонований проект. Для досягнення ефективних результатів для вибраних організацій необхідно розробити стратегію охоплення ринку.

Таблиця 8.15 - Визначення базової стратегії розвитку

№ п\п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Створення ПЗ використовуючи машинне навчання	Ринкове позиціонування	Швидкодія, простота у користуванні, безпечність, надійність	Диференціація

Була обрана наступна альтернатива для розробки проєктів: розробка програмного забезпечення за допомогою машинного навчання.

Таблиця 8.16 - Визначення базової стратегії конкурентної поведінки

№ п\п	«Першопроходець» на ринку	Агресивний пошук нових споживачів	Копіювання основних характеристик конкурентів	Стратегія конкурентної поведінки
-------	---------------------------	-----------------------------------	---	----------------------------------

1	Ні	Так	Буде, а саме: буде копіюватися ідея щодо використання підходу машинного навчання для побудови ПЗ (конкуренти 1, 2, 3), корисний інтерфейс (конкурент 3).	Зайняття конкурентної ніші
---	----	-----	--	----------------------------

Таким чином, визначається ключова стратегія конкурентної поведінки - підтримка конкурентних позицій.

Далі викладається стратегія ділового часу, яка допоможе користувачам ідентифікувати програмний продукт.

Таблиця 8.17 - Визначення стратегії позиціонування

№ п\п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Простота інтерфейсу,	Диференціація	Простота користувацьког	Швидкодія, безпека, простота,

	швидкодія, точність результатів		о інтерфейсу дозволить отримувати необхідні дані і відслідковуват и події в режимі реального часу	точність результатів
--	---------------------------------------	--	---	-------------------------

Таким чином, визначається стратегія позиціонування, а головною вимогою до виходу цільової аудиторії є: легка взаємодія, швидкість завантаження, надійність, безпека даних; ключова стратегія розвитку: різноманітність. Ключові сфери конкуренції для стартап-проекту: зручність користування дозволяє отримати доступ до необхідних даних і відстежувати події в режимі реального часу, а також повний спектр факторів проекту: швидкість, безпека, простота.

8.5 Розробка маркетингової програми

Першим кроком є розробка маркетингової ідеї для товару, який клієнт отримає на виході. Для цього в таблиці 5.18 узагальнено результати першого аналізу конкуренції на ринку.

Таблиця 8.18 - Визначення ключових переваг концепції потенційного товару

№ п\п	Потреба	Вигода, яку пропонує продукт	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
----------	---------	---------------------------------	---

1	Швидкодія	ПЗ працює досить швидко, приведений граф можна отримати до 1 сек	Перевага у швидкості
2	Простота користувацького інтерфейсу	Простота роботи додатку	Користувачі мають зручний інтерфейс для взаємодії з додатком

Таким чином, можна бачити, що проект має суттєві переваги перед конкурентами та повністю відповідає цільовим потребам. Першим кроком було розроблення маркетингової ідеї для товару, який замовник використовував би для своїх цілей. У таблиці наведені результати першого дослідження.

Далі в наступній таблиці описується модель збуту для трьох рівнів товарів: поняття товарів та / або послуг, фізичні компоненти, особливості способу доставки.

Таблиця 8.19 - Трирівнева модель товару

Рівні товару	Сутність та складові		
1. Товар за задумом	Додаток будувати з симетричної матриці граф		
2. Товар у реально	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор

му виконанні	1. Зручність та простота користувацького інтерфейсу	Нм	Технологічна
	2. Швидкість роботи	Нм	Технологічна
	3. Безпека згідно до світових стандартів	Нм	Технологічна
	Якість: згідно до стандарту ISO 4444 буде проведено тестування		
	Маркування відсутнє		
	Моя компанія: “Fedora”		
3. Товар із підкріпленням	10-денна пробна версія		
	Постійна підтримка для користувачів		
За рахунок чого потенційний товар буде захищено від копіювання: патент			

Описуючи три рівні виробничих моделей, з яких можна зробити висновок, що ключові особливості виробничого продукту дуже зручні та технологічні.

Також наводиться зміст тієї частини концептуального продукту цього товару з акцентом.

Наступним кроком є визначення діапазону цін (табл. 5.20). Аналіз зроблений експертно.

Таблиця 8.20 - Визначення меж встановлення ціни

№ п\п	Рівень цін на товари-замінники, грн	Рівень цін на товари-аналоги, грн	Рівень доходів цільової групи споживачів, грн	Верхня та нижня межі встановлення ціни на товар/послугу, грн
1	11000	12000	5000	8000-10000

Далі визначається найкращий процес продажу для подальшого прийняття рішення (табл. 5.21)

Таблиця 8.21 - Формування системи збуту

№ п\п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
-------	---	---	----------------------	--------------------------

1	Купують підписку та роблять щорічні внески для подовження ліцензії	Продаж	0(напрямую), 1(через одного посередника)	Власна та через посередників
---	--	--------	--	------------------------------

Тому система продажів встановлюється у формі щорічного членства (ліцензії).

Продаж буде проводитись самостійно та за допомогою посередників, прямий та односторонній у формі продажу товарів.

Тоді була розвинена ідея маркетингових комунікацій.

Таблиця 8.22 - Концепція маркетингових комунікацій

№ п\п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Використання за веб браузерів	Інтернет	Швидкодія, простота у використанні, безпека	Показати переваги сервісу, у тому числі і перед конкурентами	Документація у якій з прикладами описане використання та встановлення програми

Отже, в Таблиці 5.22 наведено концепцію маркетингового спілкування, вирішено, що придбання використаної ліцензії буде поновлюватися щомісячно, використання сервісу можливе в хмарах або інших послугах.

Висновки до розділу 8

У цьому розділі ми дізнаємося про найважливіші речі щодо виходу товарів на подібний ринок. Рівень виходу на ринок не дуже високий, хоча існує глобальна конкуренція.

Успіх у проекті вимагає впровадження практики машинного навчання. Структура цього дослідження окреслила ключові економічні особливості проекту, а також управління потенційними ризиками. Вивчивши результати, ми можемо зробити висновок, що подальша реалізація є доцільною.

Під час аналізу були оцінені найважливіші фінансово-економічні параметри проекту, а також проаналізовані ризики. Як результат, впровадження буде дуже конкурентоспроможним.

ПЕРЕЛІК ПОСИЛАНЬ

- [1] OS Fingerprinting, 2011. [Online] Retrieved from: http://forensicswiki.org/wiki/OS_fingerprinting [Accessed: 18-Oct-2018].
- [2] What you must know about OS Fingerprinting, 2014. [Blog] Retrieved from: <https://resources.infosecinstitute.com/must-know-os-fingerprinting/> [Accessed: 18-Oct-2018].
- [3] J. M. Allen. OS and Application Fingerprinting Techniques. SANS Technology Institute, 2007.
- [4] K. Alspach. 2018 Internet of Things 50: 10 coolest IoT Security Companies, 2018. [Online] Available: <https://www.crn.com/slide-shows/security/300101861/2018-internet-of-things-50-10-coolest-iot-security-companies.htm> [Accessed: 18-Oct-2018].
- [5] Aruba. State of IoT Healthcare, 2018. [Online] Available: <https://bit.ly/2t6vgSm> [Accessed: 15-Oct-2018].
- [6] U. Babaria. Why IoT Development needs Microservices and Containerization, 2018. [Blog] Available: <http://sandhill.com/article/why-iot-development-needs-microservices-and-containerization/> [Accessed: 18-Oct-2018].
- [7] B. Butzin, F. Golatowski, and D. Timmermann. Microservices approach for the internet of things. In Emerging Technologies and Factory Automation (ETFa), 2016 IEEE 21st International Conference on, pages 1–6. IEEE, 2016.
- [8] M. Cheng. Medical Device Regulations: Global overview and guiding principles. World Health Organization, 2003.
- [9] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. Microservices: yesterday, today, and tomorrow. In Present and Ulterior Software Engineering, pages 195–216. Springer, 2017.

- [10] FDA. Guidance for Industry, FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices. U.S. Food & Drug Administration, 1999.
- [11] FDA. FDA outlines cybersecurity recommendations for medical device manufacturers, 2016. [Online] Available: <https://www.fda.gov/NewsEvents/Newsroom/PressAnnouncements/ucm481968.htm> [Accessed: 18-Oct-2018].
- [12] Frost and Sullivan. Internet of Medical Things, Forecast to 2021, 2017. Available: <https://store.frost.com/internet-of-medical-things-forecast-to-2021.html> [Accessed: 15-Oct-2018].
- [13] M. Fuentes. Cybercrime and other threats faced by the healthcare industry, 2017. [PDF] Available: <https://bit.ly/2PWJtPv> [Accessed: 18-Oct-2018].
- [14] Gartner. Gartner Hype Cycle, 2018. Retrieved from: <https://www.gartner.com/en/research/methodologies/gartner-hype-cycle> [Accessed: 18-Oct-2018].
- [15] C. Ghali. cesarghali/os-fingerprinting, 2013. [GitHub] Available: <https://github.com/cesarghali/OS-Fingerprinting> [Accessed: 10-May-2018].
- [16] R. Hill, D. Shadija, and M. Rezai. Enabling community health care with microservices. In Ubiquitous Computing and Communications (ISPA/IUCC), 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on, pages 1444–1450. IEEE, 2017.
- [17] E. Hjelmvik. Passive OS Fingerprinting, 2011. [Blog] Retrieved from: <https://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting> [Accessed: 18-Oct-2018].
- [18] J. Jay. Healthcare sector suffered more than half of all cyber-attacks in 2017, 2018. [Blog] Available: <https://www.scmagazineuk.com/healthcare-sector-suffered-half-cyber-attacks-2017/article/1472744> [Accessed: 18-Oct-2018].

- [19] Juniper. Iot Security spend to reach \$6 Billion by 2023, growing 300% from 2018, 2018. [Online] Available: <https://bit.ly/2Dv6pQ1> [Accessed: 18-Oct-2018].
- [20] Kali. What is Kali Linux?, 2018. [Online] Available: <https://docs.kali.org/introduction/what-is-kali-linux> [Accessed: 18-Oct-2018].
- [21] M. Khera. Think like a hacker: Insights on the latest attack vectors (and security controls) for medical device applications. *Journal of diabetes science and technology*, 11(2):207–212, 2017.
- [22] J. Kirk. Pacemaker hack can deliver deadly 830- volt jolt, 2012. [Blog] Available: <https://www.computerworld.com/article/2492453/malware-vulnerabilities/pacemaker-hack-can-deliver-deadly-830-volt-jolt.html> [Accessed: 18-Oct-2018].
- [23] D. Lu, D. Huang, A. Walenstein, and D. Medhi. A secure microservice framework for iot. In *Service-Oriented System Engineering (SOSE)*, 2017 IEEE Symposium on, pages 9–18. IEEE, 2017.
- [24] L. Martins, R. T. J’uniior, W. F. Giozza, J. P. C. da Costa, et al. Increasing the dependability of iot middleware with cloud computing and microservices. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, pages 203–208. ACM, 2017.
- [25] Mulesoft. Microservices vs Monolithic Architecture, 2018. Retrieved from: <https://www.mulesoft.com/resources/api/microservices-vs-monolithic> [Accessed: 16-Oct-2018].
- [26] OWASP. Top IoT Vulnerabilities, 2016. [Online] Available: https://www.owasp.org/index.php/Top_IoT_Vulnerabilities [Accessed: 18-Oct-2018].
- [27] N. Patel. Five things to know about the future of Microservices and IoT, 2017. [Blog] Available: <https://bit.ly/2AYrQ9P> [Accessed: 18-Oct-2018].

- [28] Ponemon. 2018 Cost of a Data Breach Study: Global Overview (p. 18), 2018. [Image] Retrieved from: <https://www.ibm.com/security/data-breach> [Accessed: 18-Oct-2018].
- [29] J. Radcliffe. Hacking medical devices for fun and insulin: breaking the human SCADA system, 2014. [PDF] Available: <https://vdocuments.mx/radcliffe-hacking-medical-devices-wp.html> [Accessed: 29-Oct-2018].
- [30] F. Rahman, M. Farmani, M. Tehranipoor, and Y. Jin. Hardware-assisted cybersecurity for iot devices. In 2017 18th International Workshop on Microprocessor and SOC Test and Verification (MTV), pages 51–56. IEEE, 2017.
- [31] Redhat. What are microservices?, 2017. [Image] Retrieved from: <https://www.redhat.com/en/topics/microservices/what-are-microservices> [Accessed: 18-Oct-2018].
- [32] C. Richardson. Pattern: Microservice Architecture, 2017. Retrieved from: <https://microservices.io/patterns/microservices.html> [Accessed: 18-Oct-2018]. [33] C. Richardson and F. Smith. Microservices from design to deployment. NGINX Inc, pages 3–4, 2016.
- [34] R. Rodger. The Tao of Microservices. Manning Publications Company, 2017.
- [35] J. Sametinger, J. Rozenblit, R. Lysecky, and P. Ott. Security challenges for medical devices. Communications of the ACM, 58(4):74–82, 2015.
- [36] Synopsys. Synopsys and Ponemon Study Highlights Critical Security Deficiencies in Medical Devices, 2017. [Online] Available: <https://prn.to/2QwyObb> [Accessed: 18-Oct-2018].
- [37] A. Taylor. What is CVE, its definition and purpose?, 2017. [Blog] Retrieved from: <https://www.csoononline.com/article/3204884/application-security/what-is-cve-its-definition-and-purpose.html> [Accessed: 18-Oct-2018].

- [38] A. Wendy. The biggest healthcare breaches of 2017, 2017. [Blog] Available: <https://www.healthcareitnews.com/slideshow/biggest-healthcare-breaches-2017-so-far> [Accessed: 18-Oct-2018].
- [39] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, page 5. ACM, 2015.
- [40] K. Zetter. It's insanely easy to hack hospital equipment, 2014. [Blog] Available: <https://www.wired.com/2014/04/hospital-equipment-vulnerable/> [Accessed: 18-Oct-2018].

ДОДАТОК А

КОД ДЖЕРЕЛА ДЛЯ СЕРВЕРА

Основний сценарій для запуску REST API

```
'use
strict'

const electron = require('electron');

const { app, BrowserWindow, Menu, ipcMain, shell } = electron;

let mainWindow; // For ./src/screens/index.html
let osfpWindow; // For ./src/screens/msOne-OSFP.html
let cvesWindow; // For ./src/screens/msOne-CVEs.html
let liveAttackWindow; // For ./src/screens/msTwo.html
let passiveAttactWindow; // For ./src/screens/msThree.html

let targetOS; // For CVEs listings


function createMainWindow () {
  mainWindow = new BrowserWindow({
    width: 1000,
    height: 600
  });
  mainWindow.loadURL(`file://${__dirname}/src/screens/home.html`);
  // mainWindow.webContents.openDevTools();
  mainWindow.on('closed', () => app.quit());

  const mainMenu = Menu.buildFromTemplate(menuTemplate);
```

```

    Menu.setApplicationMenu(mainMenu);
}

app.on('ready', createMainWindow);

ipcMain.on('window-OSFP:open', (event, arg) => {
  if (arg) {
    // console.log('main.js receives open-win-osfp msg & opens osfp window');
    osfpWindow = new BrowserWindow({
      // width: 1200,
      // height: 600
      fullscreen: true
    });
    osfpWindow.loadURL(`file://${__dirname}/src/screens/msOne-OSFP.html`);
    // osfpWindow.webContents.openDevTools();
    osfpWindow.on('closed', () => osfpWindow = null);
  }
});

ipcMain.on('targetOS:set', (event, arg) => {
  // console.log('main.js receives set-target-os msg and opens cves window');
  targetOS = arg;
  cvesWindow = new BrowserWindow({
    width: 1200,
    height: 800
  });
  cvesWindow.loadURL(`file://${__dirname}/src/screens/msOne-CVEs.html`);
  // cvesWindow.webContents.openDevTools();
  cvesWindow.on('closed', () => cvesWindow = null);
});

ipcMain.on('window-CVEs:loaded', (event, arg) => {
  if (arg) {
    cvesWindow.webContents.send('targetOS:get', targetOS);
    // console.log('main.js receives cves-page-loaded msg and sends
targetOS');
  }
});

ipcMain.on('window-LiveAttack:open', (event, arg) => {
  if (arg) {
    liveAttackWindow = new BrowserWindow({
      width: 1200,
      height: 600
      // fullscreen: true

```

```

    });
    liveAttackWindow.loadURL(`file://${__dirname}/src/screens/msTwo.html`);
    liveAttackWindow.on('closed', () => liveAttackWindow = null);
  }
});

ipcMain.on('window-PassiveAttack:open', (event, arg) => {
  if (arg) {
    passiveAttactWindow = new BrowserWindow({
      width: 1200,
      height: 600
      // fullscreen: true
    });

    passiveAttactWindow.loadURL(`file://${__dirname}/src/screens/msThree.html`);
    passiveAttactWindow.on('closed', () => passiveAttactWindow = null);
  }
});

const menuTemplate = [{
  label: 'Menu',
  submenu: [
    {
      label: 'GitHub Repo',
      click() {
        shell.openExternal('https://github.com/ohmtrivedi/FAU-MS-Thesis-Front-End');
      }
    },
    { type: 'separator' },
    {
      label: 'Quit',
      accelerator: process.platform === 'darwin' ? 'Command+Q' : 'Ctrl+Q',
      click() {
        app.quit();
      }
    }
  ]
}
]);

if (process.platform === 'darwin') {
  menuTemplate.unshift({});
}

if (process.env.NODE_ENV !== 'production') {

```

```

    menuTemplate.push({
      label: 'Developer',
      submenu: [
        { role: 'reload' },
        {
          label: 'Toggle Developer Tools',
          accelerator: process.platform === 'darwin' ? 'Command+Alt+I' :
'Ctrl+Shift+I',
          click(item, focusedWindow) {
            focusedWindow.toggleDevTools();
          }
        }
      ]
    });
  }
}

```

Seneca API Plugin

```

module.exports
= function
api(options) {

  let seneca = this;

  seneca.add('role:api,path:msOne', function (msg, respond) {
    // console.log(msg);
    let { operation } = msg.args.params;
    let { target } = msg.args.query;
    // let operation = msg.args.body.operation;
    // let target = msg.args.body.remote_ip;
    if (operation == 'fpOS') {
      seneca.act({ role: 'msOne', cmd: 'fpOS', remoteIP: target },
function (err, result) {

        if (err) throw err;
        // result.then(output => respond(output));
        let curTimestamp = new Date();

        console.log(`[${curTimestamp.toLocaleString()}] msOne -> fpOS
request completed.`);

        respond(result);

      });
    } else if (operation == 'getCVEs') {
      seneca.act({ role: 'msOne', cmd: 'getCVEs', osQuery: target },
function (err, result) {

        if (err) throw err;
        // result.then(output => respond(output));
        let curTimestamp = new Date();

```

```

        console.log(`[${curTimestamp.toLocaleString()}] msOne -> getCVEs
request completed.`);

        respond(result);

    });

}

});

seneca.add('role:api,path:msTwo', function (msg, respond) {
    let { operation } = msg.args.params;

    if (operation == 'start') {
        seneca.act({ role: 'msTwo', cmd: 'start' }, function
(err, result) {

            if (err) throw err;
            let curTimestamp = new Date();

            console.log(`[${curTimestamp.toLocaleString()}] msTwo -> start
request completed.`);

            respond(result);

        });
    } else if (operation == 'updateSLA') {
        seneca.act({ role: 'msTwo', cmd: 'getLogs', type:
'StatelessAttack' }, function (err, result) {
            if (err) throw err;
            let curTimestamp = new Date();

            console.log(`[${curTimestamp.toLocaleString()}] msTwo -> updateSLA
request completed.`);

            respond(result);

        });
    } else if (operation == 'updateSFA') {
        seneca.act({ role: 'msTwo', cmd: 'getLogs', type:
'StatefullAttack' }, function (err, result) {
            if (err) throw err;
            let curTimestamp = new Date();

            console.log(`[${curTimestamp.toLocaleString()}] msTwo -> updateSFA
request completed.`);

            respond(result);

        });
    } else if (operation == 'stop') {
        seneca.act({ role: 'msTwo', cmd: 'stop' }, function
(err, result) {

            if (err) throw err;

```

```

        let curTimestamp = new Date();

        console.log(`[${curTimestamp.toLocaleString()}] msTwo -> stop
request completed.`);

        respond(result);
    });
}
});

seneca.add('role:api,path:msThree', function (msg, respond) {
    let { operation } = msg.args.params;
    let { fileName } = msg.args.query;

    if (operation == 'analyzePCAP') {
        seneca.act({ role: 'msThree', cmd: 'analyzePCAP',
fileName }, function (err, result) {
            if (err) throw err;
            let curTimestamp = new Date();

            console.log(`[${curTimestamp.toLocaleString()}] msThree ->
analyzePCAP request completed.`);

            respond(result);
        });
    }
});
}

```

Seneca Plugin for MS One Service

```

'use
strict'

```

```

const { exec } = require('child_process'),
    rp = require('request-promise'),
    cheerio = require('cheerio');
// util = require('util');

module.exports = function msOne (options) {
    let seneca = this;
    // let identifiedOS = '';

    seneca.add({ init: 'msOne' }, function(pluginInfo, respond) {
        console.log(options.message);
        respond();
    });
}

```

```

});

seneca.add({ role: 'msOne', cmd: 'fpOS' }, function (msg, respond) {
  console.log('Msg received here!');
  const { remoteIP } = msg;
  let commandToExec = `sudo nmap -T5 -O ${remoteIP} | grep -m 1 -e
"Aggressive OS guesses" -e "OS details" -e "No OS"`;
  exec(commandToExec, (error, stdout, stderr) => {
    let outputSplit, osGuess;
    let curTimestamp = new Date();
    console.log(`[${curTimestamp.toLocaleString()}] Nmap of ${remoteIP}
complete!`);
    if (error) {
      console.log(error);
    } else if (stderr) {
      console.log(stderr);
    } else if (stdout.startsWith('Aggressive')) {
      outputSplit = stdout.slice(22).split(',');
      osGuess = outputSplit[0].trim();
    } else if (stdout.startsWith('OS')) {
      outputSplit = stdout.slice(11).split(',');
      osGuess = outputSplit[0].trim();
    } else {
      osGuess = stdout.trim();
    }
    if (osGuess.indexOf('(') !== -1) {
      osGuess = osGuess.substring(0, osGuess.indexOf('(')-1);
    }
    respond(null, { osGuess });
  });
});

seneca.add({ role: 'msOne', cmd: 'getCVEs' }, async function (msg, respond) {
  let { osQuery } = msg;
  let osCVEs = await scrapCVEs(osQuery);
  console.log(`Response: Found ${osCVEs.cve_id.length} CVEs.`);
  respond(null, osCVEs);
});

}

let scrapCVEs = (osQuery) => {
  let osName = osQuery.split(' ');
  let url = 'http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=' +
osName.join('+');
  let os_cves = { cve_id: [], cve_desc: [] };

```

```

const options = {
  uri: url,
  transform: function(body) {
    return cheerio.load(body);
  }
};

return rp(options)
  .then($ => {
    let desiredTableElem = $('#TableWithRules').children('table');
    desiredTableElem.find('td').each((i, elem) => {
      // console.log(i + elem.name);
      if (elem.children[0].type === 'tag') {
        // console.log(elem.children[0].name);
        os_cves.cve_id.push(elem.children[0].children[0].data);
        // console.log(elem.children[0].children[0].data);
      }
      else if (elem.children[0].type === 'text') {
        os_cves.cve_desc.push(elem.children[0].data);
        // console.log(elem.children[0].data);
      }
    });
    console.log('Scrapping complete!');
    return os_cves;
  })
  .catch(err => {
    console.log(err);
  });
}

```

Seneca Plugin for MS Two Service

```

'use
strict'

```

```

const { fork } = require('child_process');
// util = require('util');

```

```

process.setMaxListeners(0);

```

```

let newSession, sessionData;

```

```

// let addSessionListener = () => {
//   sessionData = null;
//   newSession.on('message', (data) => {

```

```

//      sessionData = data;
//    });
//  }

module.exports = function msTwo (options) {
  let seneca = this;

  seneca.add({ init: 'msTwo' }, function(pluginInfo, respond) {
    console.log(options.message);
    respond();
  });

  seneca.add({ role: 'msTwo', cmd: 'start' }, function (msg, respond) {
    newSession = fork('./liveFPDetection.js', [], { silent: false });
    newSession.send('start');
    newSession.on('message', (status) => {
      respond(null, { status });
    });
  });

  seneca.add({ role: 'msTwo', cmd: 'getLogs' }, function (msg, respond) {
    if (msg.type == 'StatelessAttack') {
      newSession.send('getLogs:StatelessAttack');
      newSession.on('message', (SLALogs) => {
        respond(null, { SLALogs });
      });
    } else if (msg.type == 'StatefullAttack') {
      newSession.send('getLogs:StatefullAttack');
      newSession.on('message', (SFALogs) => {
        respond(null, { SFALogs });
      });
    }
  });

  seneca.add({ role: 'msTwo', cmd: 'stop' }, function (msg, respond) {
    newSession.send('stop');
    newSession.on('message', (response) => {
      newSession.kill();
      respond(null, response);
    });
  });
}

```

```
'use
strict'
```

```
const pcap = require('pcap'),
      { fork } = require('child_process'),
      fs = require('fs'),
      { MongoClient } = require('mongodb'),
      { USERNAME, PASSWORD, SERVER_URL, DATABASE } =
require('../common/mongoDBConfig');

process.setMaxListeners(0);

let attackSigns, newSession, attackers, writeStream, logFile;
let SLALogs = '', SFALogs = '';
let pktCounts = {
  totalPkts: 0,
  tcpPkts: 0,
  udpPkts: 0,
  icmpPkts: 0,
  // SYN_ACK: 0,
  statelessAttacks: {
    NULLScan: 0, // No flags set
    PSH: 0,
    SYN_FIN: 0,
    RST_FIN: 0,
    RST_SYN: 0,
    FIN_URG: 0,
    SYN_URG: 0,
    SYN_FIN_URG: 0,
    XMASScan: 0, // URG && PSH && FIN
    SYN_FIN_PSH_URG: 0,
    SYN_ECE_CWR: 0,
    ACK_DF_WS1024: 0,
  },
  statefullAttacks: {
    noSession: 0,
    FINScan: 0, // FIN only
  },
  icmpAttacks: {
    invalidCode: 0,
    invalidType: 0
  }
};

function getSignatures() {
```

```

const MONGO_URL = `mongodb://${USERNAME}:${PASSWORD}@${SERVER_URL}/`;

let opts = {
  useUrlParser: true,
  authSource: 'admin'
};

return new Promise((resolve, reject) => {
  MongoClient.connect(MONGO_URL, opts, (err, db) => {
    if (err) throw err;
    let mydb = db.db(DATABASE);

    let query = {};

    mydb.collection('attackSigns')
      .find(query, { projection: { _id: 0, attack: 1, sign: 1 } })
      .toArray((err, result) => {
        if (err) reject(err);
        attackSigns = result.reduce((map, obj) => {
          map[obj.attack] = obj.sign;
          return map;
        }, {});
        // console.log(attackSigns);
        resolve(attackSigns);
        db.close();
      });
  });
});

function parsePacket(raw_packet) {
  pktCounts.totalPkts++;

  let packet = pcap.decode.packet(raw_packet);
  // console.log(packet);
  let packetHeader = packet.pcap_header;
  let pktTimestamp = new Date(packetHeader.tv_sec*1000);

  let ethernetPacket = packet.payload;

  // Check for IPv4
  if (ethernetPacket.ethertype != 2048) { // ipPacket.constructor.name !=
    'IPv4'
    // console.log(`[${pktTimestamp.toLocaleString()}] Packet
    #${pktCounts.totalPkts}: Not IPv4, actual - ${ethernetPacket.ethertype}`);
  }
}

```

```

        return;
    }

    // IP packet
    let ipPacket = ethernetPacket.payload;

    // Validate IPv4 Header
    if (ipPacket.version !== 4 || (ipPacket.headerLength < 20 &&
ipPacket.headerLength > 60)) {
        // console.log(`[${pktTimestamp.toLocaleString()}] Packet
#${pktCounts.totalPkts}: Invalid IPv4 header`);
        return;
    }

    // Protocol
    // let protocol;
    switch (ipPacket.protocol) {
        case 1: // protocol = 'ICMP'
            pktCounts.icmpPkts++;
            parseICMP(ipPacket, pktTimestamp);
            break;
        case 6: // protocol = 'TCP'
            pktCounts.tcpPkts++;
            parseTCP(ipPacket, pktTimestamp);
            break;
        case 17: // protocol = 'UDP'
            pktCounts.udpPkts++;
            // parseUDP();
            break;
        default: // protocol = 'Other'
            // console.log(`[${pktTimestamp.toLocaleString()}] Packet
#${pktCounts.totalPkts}: Other Protocol - ${ipPacket.protocol}`);
            return;
    }
    return;
}

function parseTCP(ipPacket, pktTimestamp) {
    // Extracting Source & Destination IP
    let sourceIP = ipPacket.saddr['addr'].join('.');
    let destinationIP = ipPacket.daddr['addr'].join('.');
    // IP Flags
    let ipFlag_DF = ipPacket.flags.doNotFragment;

    // TCP packet

```

```

let tcpPacket = ipPacket.payload;
// Source Port
let sourcePort = tcpPacket.sport;
// Destination Port
let destinationPort = tcpPacket.dport;
// Windows size
let windowSize = tcpPacket.windowSize;
// TCP Flags
let tcpFlags = (tcpPacket.flags.fin ? 1 : 0) + (tcpPacket.flags.syn ? 2 : 0)
               + (tcpPacket.flags.rst ? 4 : 0) + (tcpPacket.flags.psh ? 8 : 0)
               + (tcpPacket.flags.ack ? 16 : 0) + (tcpPacket.flags.urg ? 32 : 0)
               + (tcpPacket.flags.ece ? 64 : 0) + (tcpPacket.flags.cwr ? 128 :
0);

/* Check for Stateless Attacks */
if (tcpFlags == attackSigns['Null Scan'].tcpFlags) {
    pktCounts.statelessAttacks.NULLScan++;
    writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet with no flags ` +
        `set to ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
    SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with no flags ` +
        `set to ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
    return;
} else if (tcpFlags == attackSigns['SYN-FIN'].tcpFlags) {
    pktCounts.statelessAttacks.SYN_FIN++;
    writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet with SYN and ` +
        `FIN flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`);
    SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with SYN and ` +
        `FIN flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`;
    return;
} else if (tcpFlags == attackSigns['RST-FIN'].tcpFlags) {
    pktCounts.statelessAttacks.RST_FIN++;
    writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet with RST and ` +
        `FIN flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`);
    SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with RST and ` +
        `FIN flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`;

```

```

        return;
    } else if (tcpFlags == attackSigns['RST-SYN'].tcpFlags) {
        pktCounts.statelessAttacks.RST_SYN++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet with RST and ` +
            `SYN flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with RST and ` +
            `SYN flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`;
        return;
    } else if (tcpFlags == attackSigns['PSH'].tcpFlags) {
        pktCounts.statelessAttacks.PSH++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet containing ` +
            `only PSH flag to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet containing ` +
            `only PSH flag to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`;
        return;
    } else if (tcpFlags == 18) {
        // pktCounts.SYN_ACK++;
        return;
    } else if (tcpFlags == attackSigns['FIN-URG'].tcpFlags) {
        pktCounts.statelessAttacks.FIN_URG++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet with FIN ` +
            `and URG flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with FIN ` +
            `and URG flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`;
        return;
    } else if (tcpFlags == attackSigns['SYN-URG'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_URG++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet with SYN ` +
            `and URG flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with SYN ` +

```

```

        `and URG flags set to ${destinationIP}:${destinationPort}. Suspecting
        OSFP.\n`;
        return;
    } else if (tcpFlags == attackSigns['SYN-FIN-URG'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_FIN_URG++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
        TCP packet with RST ` +
        `and URG flags set to ${destinationIP}:${destinationPort}. Suspecting
        OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
        packet with RST ` +
        `and URG flags set to ${destinationIP}:${destinationPort}. Suspecting
        OSFP.\n`;
        return;
    } else if (tcpFlags == attackSigns['XMAS Scan'].tcpFlags) {
        pktCounts.statelessAttacks.XMASScan++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
        TCP packet with FIN, ` +
        `PSH and URG flags set [X-MAS Tree Scan] to
        ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
        packet with FIN, ` +
        `PSH and URG flags set [X-MAS Tree Scan] to
        ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
        return;
    } else if (tcpFlags == attackSigns['SYN-FIN-PSH-URG'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_FIN_PSH_URG++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
        TCP packet with SYN, ` +
        `FIN, PSH and URG flags set to ${destinationIP}:${destinationPort}.
        Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
        packet with SYN, ` +
        `FIN, PSH and URG flags set to ${destinationIP}:${destinationPort}.
        Suspecting OSFP.\n`;
        return;
    } else if (tcpFlags == attackSigns['SYN-ECE-CWR'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_ECE_CWR++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
        TCP packet with SYN, ` +
        `ECN and CWR flags set to ${destinationIP}:${destinationPort}.
        Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
        packet with SYN, ` +

```

```

        `ECN and CWR flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
        return;
    } else if ((ipFlag_DF == attackSigns['ACK-DF-WS1024'].ipFlag_DF) &&
        (tcpFlags == attackSigns['ACK-DF-WS1024'].tcpFlags) &&
        (windowSize == attackSigns['ACK-DF-WS1024'].windowSize)) {
        pktCounts.statelessAttacks.ACK_DF_WS1024++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP ACK packet with ` +
            `IP DF and a window size of 1024 to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP ACK
packet with ` +
            `IP DF and a window size of 1024 to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
        return;
    } else { }

    /* Check for Statefull Attacks */

    if (!attackers.has(sourceIP)) {
        let attackerState = {
            // attackerIP: sourceIP,
            pktCounts: 0,
            tcpConnections: [],
            tcpConnectionsTotal: 0,
            // tcpConnectionsIndex: 0,
            pktCounts_SYNARST: 0,
            // udpConnections: [],
            // udpConnectionsTotal: 0,
            // udpConnectionsIndex: 0,
            allPortsScan: false,
            tcp_SYNARST: false
        };
        attackers.set(sourceIP, attackerState);
    }

    // Get current state of attacker
    let attacker = attackers.get(sourceIP);
    attacker.pktCounts++;

    // Retrieving index of connection, if a TCP connection already exists
    let tcpConn_ix = attacker.tcpConnections.findIndex(item => {
        return (item.srcPort == sourcePort && item.dstPort == destinationPort)
    });

```

```

// New Connection, First Packet
if (tcpConn_ix == -1) {
    if (tcpFlags == 2) { // SYN Packet trying to establish connection
        let newTCPConn = {
            dstIP: destinationIP,
            srcPort: sourcePort,
            dstPort: destinationPort,
            flags: {
                SYN: true,
                ACK: false,
                FIN: false,
                RST: false,
            },
            pktCounts_SYN: 1,
            timestamp: pktTimestamp,
            portScan: false,
            halfOpenScan: false
        }
        attacker.tcpConnections.push(newTCPConn);
        attacker.tcpConnectionsTotal++;
        // attacker.tcpConnectionsIndex++;
    } else if (tcpFlags == attackSigns['FIN Scan'].tcpFlags) {
        pktCounts.statefullAttacks.FINScan++;
    } else {
        pktCounts.statefullAttacks.noSession++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP}
sent a TCP packet (Flags: ` +
            `${tcpFlags}) to ${destinationIP}:${destinationPort} with no
established connection. Suspecting OSFP.\n`);
        SFALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet (Flags: ` +
            `${tcpFlags}) to ${destinationIP}:${destinationPort} with no
established connection. Suspecting OSFP.\n`;
        // return;
    }
} else { // Connection already exists (Established Connection)
    if (tcpFlags == 2) { // SYN Packet
        attacker.tcpConnections[tcpConn_ix].pktCounts_SYN++;
    } else if (tcpFlags == 16) { // ACK Packet
        attacker.tcpConnections[tcpConn_ix].flags.ACK = true;
    } else if (tcpFlags == 4) { // RST Packet
        attacker.tcpConnections[tcpConn_ix].flags.RST = true;
        if (attacker.tcpConnections[tcpConn_ix].flags.SYN &&
            !(attacker.tcpConnections[tcpConn_ix].flags.ACK)) {

```

```

        attacker.pktCounts_SYNARST++;
    }
} else if (tcpFlags == 1) { // FIN Packet
    attacker.tcpConnections[tcpConn_ix].flags.FIN = true;
}
}

/* Check for received TCP Options */
let tcpOpts_MSS = tcpPacket.options.mss;
let tcpOpts_WinScale = tcpPacket.options.window_scale;
let tcpOpts_SACK_OK = tcpPacket.options.sack_ok;
let tcpOpts_TS_Sender = tcpPacket.options.timestamp;
let tcpOpts_TS_Echo = tcpPacket.options.tcpOpts_TS_Echo;

if (tcpOpts_WinScale == 10 && tcpOpts_MSS == 1460 && tcpOpts_SACK_OK
    && tcpOpts_TS_Sender == 0xFFFFFFFF && tcpOpts_TS_Echo == 0 && windowSize
== 1) {
    writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP}
sending strange TCP options ` +
        `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
SACK_OK: ${tcpOpts_SACK_OK}, ` +
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize})) to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
    SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sending
strange TCP options ` +
        `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
SACK_OK: ${tcpOpts_SACK_OK}, ` +
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize})) to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
}

if (tcpOpts_WinScale == 0 && tcpOpts_MSS == 1400 && tcpOpts_SACK_OK
    && tcpOpts_TS_Sender == 0xFFFFFFFF && tcpOpts_TS_Echo == 0 && windowSize
== 63) {
    writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP}
sending strange TCP options ` +
        `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
SACK_OK: ${tcpOpts_SACK_OK}, ` +
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
${tcpOpts_TS_Echo}, ` +

```

```

        `Window Size: ${windowSize}))) to
    ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sending
    strange TCP options ` +
        `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
    SACK_OK: ${tcpOpts_SACK_OK}, ` +
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
    ${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize}))) to
    ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
    }

    if (tcpOpts_WinScale == 5 && tcpOpts_MSS == 640
        && tcpOpts_TS_Sender == 0xFFFFFFFF && tcpOpts_TS_Echo == 0 && windowSize
    == 4) {
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP}
    sending strange TCP options ` +
            `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
    Timestamp (Sender): ` +
                `${tcpOpts_TS_Sender}, Timestamp (Echo): ${tcpOpts_TS_Echo},
    Window Size: ${windowSize}))) ` +
            `to ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sending
    strange TCP options ` +
            `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
    Timestamp (Sender): ` +
                `${tcpOpts_TS_Sender}, Timestamp (Echo): ${tcpOpts_TS_Echo},
    Window Size: ${windowSize}))) ` +
            `to ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
    }

    if (tcpOpts_WinScale == 10 && tcpOpts_SACK_OK
        && tcpOpts_TS_Sender == 0xFFFFFFFF && tcpOpts_TS_Echo == 0 && windowSize
    == 4) {
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP}
    sending strange TCP options ` +
            `combination (WS: ${tcpOpts_WinScale}, SACK_OK:
    ${tcpOpts_SACK_OK}, ` +
                `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
    ${tcpOpts_TS_Echo}, ` +
                    `Window Size: ${windowSize}))) to
    ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sending
    strange TCP options ` +

```

```

        `combination (WS: ${tcpOpts_WinScale}, SACK_OK:
${tcpOpts_SACK_OK}, ` +
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize}))) to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
    }

    if (tcpOpts_WinScale == 10 && tcpOpts_MSS == 536 && tcpOpts_SACK_OK
        && tcpOpts_TS_Sender == 0xFFFFFFFF && tcpOpts_TS_Echo == 0 && windowSize
        == 16) {
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP}
sending strange TCP options ` +
        `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
SACK_OK: ${tcpOpts_SACK_OK}, ` +
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize}))) to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sending
strange TCP options ` +
        `combination (WS: ${tcpOpts_WinScale}, MSS: ${tcpOpts_MSS},
SACK_OK: ${tcpOpts_SACK_OK}, ` +
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize}))) to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
    }

    if (tcpOpts_MSS == 256 && tcpOpts_SACK_OK
        && tcpOpts_TS_Sender == 0xFFFFFFFF && tcpOpts_TS_Echo == 0 && windowSize
        == 512) {
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP}
sending strange TCP options ` +
        `combination (MSS: ${tcpOpts_MSS}, SACK_OK: ${tcpOpts_SACK_OK}, `
+
        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize}))) to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`);
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sending
strange TCP options ` +
        `combination (MSS: ${tcpOpts_MSS}, SACK_OK: ${tcpOpts_SACK_OK}, `
+

```

```

        `Timestamp (Sender): ${tcpOpts_TS_Sender}, Timestamp (Echo):
        ${tcpOpts_TS_Echo}, ` +
        `Window Size: ${windowSize}))) to
        ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
    }

    // Updating attacker's state
    attackers.set(sourceIP, attacker);
}

function parseICMP(ipPacket, pktTimestamp) {

    // Extracting Source & Destination IP
    let sourceIP = ipPacket.saddr['addr'].join('.');
    let destinationIP = ipPacket.daddr['addr'].join('.');

    let icmpPacket = ipPacket.payload;
    // Extracting ICMP Type and ICMP Code
    let icmpType = icmpPacket.type;
    let icmpCode = icmpPacket.code;

    if (icmpType == attackSigns['Invalid Code'].type && icmpCode >
    attackSigns['Invalid Code'].codeGT) {
        pktCounts.icmpAttacks.invalidCode++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
        ICMP packet with invalid ` +
        `Code value to ${destinationIP}. Suspecting OSFP.\n`;
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a ICMP
        packet with invalid ` +
        `Code value to ${destinationIP}. Suspecting OSFP.\n`;
    } else if (icmpType in attackSigns['Invalid Type'].typeIN || icmpType >=
    attackSigns['Invalid Type'].typeGTE) {
        pktCounts.icmpAttacks.invalidType++;
        writeStream.write(`[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
        ICMP packet with reserved ` +
        `Type value to ${destinationIP}. Suspecting OSFP.\n`;
        SLALogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a ICMP
        packet with reserved ` +
        `Type value to ${destinationIP}. Suspecting OSFP.\n`;
    }
}

function forkAnalysis(attackers) {
    const analysisProcess = fork('../common/attackersAnalysis.js');

```

```

// console.log(`\nSent ${attackers.size} potential attackers for analysis.`);
let data = JSON.stringify([...attackers]);
analysisProcess.send(data);

analysisProcess.on('message', (results) => {
  writeStream.write(results);
  SFALogs += results;
  analysisProcess.kill();
});
}

process.on('message', (args) => {
  if (args == 'start') {
    getSignatures().then(() => {
      const MY_IP = '10.13.200.148';
      const HOST_IP = '10.15.4.110';
      // Filter for packets sent to my system
      newSession = pcap.createSession('', `src host ${MY_IP}`);
      // No filter
      // newSession = pcap.createSession('');
      console.log(`Sniffing started on ${newSession.device_name}`);
      attackers = new Map();
      let curTimestamp = new Date();
      logFile = `logFile${curTimestamp.valueOf()}.log`;
      writeStream = fs.createWriteStream(`${__dirname}/logs/${logFile}`);
      console.log(`Writing to ${__dirname}/logs/${logFile}`);
      process.send(`Sniffing started on ${newSession.device_name}`);
      newSession.on('packet', parsePacket);
      setInterval(forkAnalysis, 60000, attackers);
    });
  } else if (args == 'getLogs:StatelessAttack') {
    if (SLALogs.length == 0) {
      console.log('No logs (SLA) yet!');
      process.send('');
    } else {
      process.send(SLALogs);
      SLALogs = '';
    }
  } else if (args == 'getLogs:StatefullAttack') {
    if (SFALogs.length == 0) {
      console.log('No logs (SFA) yet!');
      process.send('');
    } else {
      process.send(SFALogs);
      SFALogs = '';
    }
  }
});

```

```

    }
  } else if (args == 'stop') {
    newSession.close();
    console.log(`Sniffing session on ${newSession.device_name} closed.
    Sniffed ${pktCounts.totalPkts} packets.`);
    writeStream.on('finish', () => {
      console.log(`Wrote all data to ${logFile}!`);
    });
    writeStream.end();
    process.send({ status: `Sniffing session on ${newSession.device_name}
    closed. Sniffed ${pktCounts.totalPkts} packets.`, pktCounts, SLALogs, SFALogs });
  }
});

```

Seneca Plugin for MS Three Service

```

'use
strict'

const { fork } = require('child_process');
// util = require('util');

module.exports = function msThree (options) {
  let seneca = this;

  seneca.add({ init: 'msThree' }, function(pluginInfo, respond) {
    console.log(options.message);
    respond();
  });

  seneca.add({ role: 'msThree', cmd: 'analyzePCAP' }, function (msg, respond) {
    const { fileName } = msg;
    let filePath = `${__dirname}/uploads/${fileName}`;

    let newSession = fork('./passiveFPDetection.js', [], { silent: false });
    newSession.send(filePath);
    newSession.on('message', (results) => {
      let { pktCounts, consoleLogs, analysisResults } = results;
      newSession.kill();
      console.log(`Response: Analyzed ${pktCounts.totalPkts} packets.`);
      respond(null, { pktCounts, consoleLogs, analysisResults });
    });
  });
};

```

Script for PCAP Analysis

```
'use
strict'

const pcapp = require('pcap-parser'),
      { fork } = require('child_process'),
      { MongoClient } = require('mongodb'),
      { USERNAME, PASSWORD, SERVER_URL, DATABASE } =
require('../common/mongoDBConfig');

let attackSigns, pcapParser, attackers;
let consoleLogs = '';
let pktCounts = {
  totalPkts: 0,
  tcpPkts: 0,
  udpPkts: 0,
  icmpPkts: 0,
  // SYN_ACK: 0,
  statelessAttacks: {
    NULLScan: 0, // No flags set
    PSH: 0,
    SYN_FIN: 0,
    RST_FIN: 0,
    RST_SYN: 0,
    FIN_URG: 0,
    SYN_URG: 0,
    SYN_FIN_URG: 0,
    XMASScan: 0, // URG && PSH && FIN
    SYN_FIN_PSH_URG: 0,
    SYN_ECE_CWR: 0,
    ACK_DF_WS1024: 0,
  },
  statefullAttacks: {
    noSession: 0,
    FINScan: 0, // FIN only
  },
  icmpAttacks: {
    invalidCode: 0,
    invalidType: 0
  }
};

function getSignatures() {
  const MONGO_URL = `mongodb://${USERNAME}:${PASSWORD}@${SERVER_URL}/`;
```

```

let opts = {
  useNewUrlParser: true,
  authSource: 'admin'
};

return new Promise((resolve, reject) => {
  MongoClient.connect(MONGO_URL, opts, (err, db) => {
    if (err) throw err;
    let mydb = db.db(DATABASE);

    let query = { };

    mydb.collection('attackSigns')
      .find(query, { projection: { _id: 0, attack: 1, sign: 1 } })
      .toArray((err, result) => {
        if (err) reject(err);
        attackSigns = result.reduce((map, obj) => {
          map[obj.attack] = obj.sign;
          return map;
        }, {});
        resolve(attackSigns);
        db.close();
      });
  });
});

}

function parsePacket(packet) {
  let packetHeader = packet.header;
  let packetContent = packet.data;

  let pktTimestamp = new Date(packetHeader.timestampSeconds*1000);
  pktCounts.totalPkts++;

  // Parsing Ethernet Header...
  // Check for IPv4 (EtherType)
  if (packetContent.readUInt8(12) != 8 && packetContent.readUInt8(13) != 0) {
    // console.log('Not IPv4 Packet');
    return;
  }

  // Parsing IPv4 Header....
  // Check for IPv4 (Version 4) (Version)
  let ipv4_VHL = packetContent.readUInt8(14);

```

```

if ( ipv4_VHL < 69 || ipv4_VHL > 79) {
    // console.log('Invalid IPv4 Packet');
    return;
}

// Extracting Protocol
let protocol = packetContent.readUInt8(0x17);

switch (protocol) {
    case 1:
        pktCounts.icmpPkts++;
        parseICMP(packetContent, pktTimestamp);
        break;
    case 6:
        pktCounts.tcpPkts++;
        parseTCP(packetContent, pktTimestamp);
        break;
    case 17:
        pktCounts.udpPkts++;
        break;
    default:
        // console.log('Unknown Protocol');
        break;
}
}

function parseTCP(packetContent, pktTimestamp) {
    // Extracting Fragmentation Info
    let dfFlag = packetContent.readUInt16LE(0x14);

    // Extracting Source & Destination IP
    let sourceIP = packetContent.readUInt8(0x1A).toString() + '.' +
        packetContent.readUInt8(0x1B).toString() + '.' +
        packetContent.readUInt8(0x1C).toString() + '.' +
        packetContent.readUInt8(0x1D).toString();

    let destinationIP = packetContent.readUInt8(0x1E).toString() + '.' +
        packetContent.readUInt8(0x1F).toString() + '.' +
        packetContent.readUInt8(0x20).toString() + '.' +
        packetContent.readUInt8(0x21).toString();

    // Parsing TCP Header...
    // Extracting Source & Destination Port
    let sourcePort = packetContent.readUInt16BE(0x22);
    let destinationPort = packetContent.readUInt16BE(0x24);

```

```

// Extracting Window Size
let windowSize = packetContent.readUInt16BE(0x30);

/* Check for Stateless Attacks */

if (packetContent.readUInt8(0x2F) == attackSigns['Null Scan'].tcpFlags) {
    pktCounts.statelessAttacks.NULLScan++;
    consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with no flags set to ${destinationIP}:${destinationPort}. Suspecting
OSFP.\n`;
    return;
} else if ((packetContent.readUInt8(0x2F) & 3) == attackSigns['SYN-
FIN'].tcpFlags) {
    pktCounts.statelessAttacks.SYN_FIN++;
    consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with SYN and FIN flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
    return;
} else if ((packetContent.readUInt8(0x2F) & 5) == attackSigns['RST-
FIN'].tcpFlags) {
    pktCounts.statelessAttacks.RST_FIN++;
    consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with RST and FIN flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
    return;
} else if ((packetContent.readUInt8(0x2F) & 6) == attackSigns['RST-
SYN'].tcpFlags) {
    pktCounts.statelessAttacks.RST_SYN++;
    consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with RST and SYN flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
    return;
} else if ((packetContent.readUInt8(0x2F) & 8) ==
attackSigns['PSH'].tcpFlags) {
    pktCounts.statelessAttacks.PSH++;
    consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet containing only PSH flag to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
    return;
} else if ((packetContent.readUInt8(0x2F) & 18) == 18) {
    // pktCounts.SYN_ACK++;
    return;
} else if ((packetContent.readUInt8(0x2F) & 33) == attackSigns['FIN-
URG'].tcpFlags) {
    pktCounts.statelessAttacks.FIN_URG++;

```

```

        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with FIN and URG flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
        return;
    } else if ((packetContent.readUInt8(0x2F) & 34) == attackSigns['SYN-
URG'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_URG++;
        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with SYN and URG flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
        return;
    } else if ((packetContent.readUInt8(0x2F) & 36) == attackSigns['SYN-FIN-
URG'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_FIN_URG++;
        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with RST and URG flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
        return;
    } else if ((packetContent.readUInt8(0x2F) & 41) == attackSigns['XMAS
Scan'].tcpFlags) {
        pktCounts.statelessAttacks.XMASScan++;
        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with FIN, PSH and URG flags set [X-MAS Tree Scan] to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
        return;
    } else if ((packetContent.readUInt8(0x2F) & 43) == attackSigns['SYN-FIN-PSH-
URG'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_FIN_PSH_URG++;
        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with SYN, FIN, PSH and URG flags set to
${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
        return;
    } else if ((packetContent.readUInt8(0x2F) & 194) == attackSigns['SYN-ECE-
CWR'].tcpFlags) {
        pktCounts.statelessAttacks.SYN_ECE_CWR++;
        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
packet with SYN, ECN and CWR flags set to ${destinationIP}:${destinationPort}.
Suspecting OSFP.\n`;
        return;
    } else if ((dfFlag == attackSigns['ACK-DF-WS1024'].ipFlag_DF) &&
        ((packetContent.readUInt8(0x2F) & 16) == attackSigns['ACK-DF-
WS1024'].tcpFlags) &&
        (windowSize == attackSigns['ACK-DF-WS1024'].windowSize)) {
        pktCounts.statelessAttacks.ACK_DF_WS1024++;
    }

```

```

        consoleLogs += `[$${pktTimestamp.toLocaleString()}] ${sourceIP} sent a TCP
        ACK packet with IP DF and a window size of 1024 to
        ${destinationIP}:${destinationPort}. Suspecting OSFP.\n`;
        return;
    } else { }

    /* Check for Statefull Attacks */

    if (!attackers.has(sourceIP)) {
        let attackerState = {
            // attackerIP: sourceIP,
            pktCounts: 0,
            tcpConnections: [],
            tcpConnectionsTotal: 0,
            // tcpConnectionsIndex: 0,
            pktCounts_SYNARST: 0,
            // udpConnections: [],
            // udpConnectionsTotal: 0,
            // udpConnectionsIndex: 0,
            allPortsScan: false,
            tcp_SYNARST: false
        };
        attackers.set(sourceIP, attackerState);
    }

    // Get current state of attacker
    let attacker = attackers.get(sourceIP);
    attacker.pktCounts++;

    // Retrieving index of connection, if a TCP connection already exists
    let tcpConn_ix = attacker.tcpConnections.findIndex(item => {
        return (item.srcPort == sourcePort && item.dstPort == destinationPort)
    });

    let tcpFlags = packetContent.readUInt8(0x2F);
    // New Connection, First Packet
    if (tcpConn_ix == -1) {
        if (tcpFlags == 2) { // SYN Packet trying to establish connection
            let newTCPConn = {
                dstIP: destinationIP,
                srcPort: sourcePort,
                dstPort: destinationPort,
                flags: {
                    SYN: true,
                    ACK: false,

```

```

        FIN: false,
        RST: false,
    },
    pktCounts_SYN: 1,
    timestamp: pktTimestamp,
    portScan: false,
    halfOpenScan: false
}
attacker.tcpConnections.push(newTCPConn);
attacker.tcpConnectionsTotal++;
// attacker.tcpConnectionsIndex++;
} else if (tcpFlags == attackSigns['FIN Scan'].tcpFlags) {
    pktCounts.statefullAttacks.FINScan++;
} else {
    pktCounts.statefullAttacks.noSession++;
    consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
TCP packet (Flags: ${tcpFlags}) to ${destinationIP}:${destinationPort} with no
established connection. Suspecting OSFP.\n`;
    // return;
}
} else { // Connection already exists (Established Connection)
    if (tcpFlags == 2) { // SYN Packet
        attacker.tcpConnections[tcpConn_ix].pktCounts_SYN++;
    } else if (tcpFlags == 16) { // ACK Packet
        attacker.tcpConnections[tcpConn_ix].flags.ACK = true;
    } else if (tcpFlags == 4) { // RST Packet
        attacker.tcpConnections[tcpConn_ix].flags.RST = true;
        if (attacker.tcpConnections[tcpConn_ix].flags.SYN &&
!(attacker.tcpConnections[tcpConn_ix].flags.ACK)) {
            attacker.pktCounts_SYNARST++;
        }
    } else if (tcpFlags == 1) { // FIN Packet
        attacker.tcpConnections[tcpConn_ix].flags.FIN = true;
    }
}
}

/* TODO: Check for received TCP Options
    Comment: Cannot be done, difficult to retrieve individual options because
of NOP complication */
let tcpOptionsIx = 0x36;

// Updating attacker's state
attackers.set(sourceIP, attacker);
}

```

```

function parseICMP(packetContent, pktTimestamp) {

    // Extracting Source & Destination IP
    let sourceIP = packetContent.readUInt8(0x1A).toString() + '.' +
        packetContent.readUInt8(0x1B).toString() + '.' +
        packetContent.readUInt8(0x1C).toString() + '.' +
        packetContent.readUInt8(0x1D).toString();

    let destinationIP = packetContent.readUInt8(0x1E).toString() + '.' +
        packetContent.readUInt8(0x1F).toString() + '.' +
        packetContent.readUInt8(0x20).toString() + '.' +
        packetContent.readUInt8(0x21).toString();

    // Extracting ICMP Type and ICMP Code
    let icmpType = packetContent.readUInt8(0x22);
    let icmpCode = packetContent.readUInt8(0x23);

    if (icmpType == attackSigns['Invalid Code'].type && icmpCode >
attackSigns['Invalid Code'].codeGT) {
        pktCounts.icmpAttacks.invalidCode++;
        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
ICMP packet with invalid code value to ${destinationIP}. Suspecting OSFP.\n`;
    } else if (icmpType in attackSigns['Invalid Type'].typeIN || icmpType >=
attackSigns['Invalid Type'].typeGTE) {
        pktCounts.icmpAttacks.invalidType++;
        consoleLogs += `[${pktTimestamp.toLocaleString()}] ${sourceIP} sent a
ICMP packet with reserved type value to ${destinationIP}. Suspecting OSFP.\n`;
    }
}

function forkAnalysis(attackers) {
    const analysisProcess = fork('../common/attackersAnalysis.js');

    // console.log(`Sent ${attackers.size} potential attackers for analysis.`);
    let data = JSON.stringify([...attackers]);
    analysisProcess.send(data);

    return new Promise((resolve, reject) => {
        analysisProcess.on('message', (results) => {
            resolve(results);
            analysisProcess.kill();
        });
    });
}

```

```

process.on('message', (filePath) => {
  getSignatures().then(() => {
    pcapParser = pcapp.parse(filePath);
    attackers = new Map();

    pcapParser.on('packet', (packet) => {
      parsePacket(packet);
    });

    pcapParser.on('end', () => {
      forkAnalysis(attackers).then((analysisResults) => {
        // console.log(analysisResults);
        console.log(`Analyzed ${pktCounts.totalPkts} packets.`);
        process.send({ pktCounts, consoleLogs, analysisResults });
      });
    });

    pcapParser.on('error', (error) => {
      console.log(error);
    });
  });
});

```

Script for Attackers Analysis

```

const
MAX_ALLOWED_CONNECTIONS
= 5;

const MAX_ALLOWED_SYN_AND_RST = 5;

let analysisResults = '';

function attackersAnalysis(attackers) {
  // console.log(util.inspect(attackers, false, null, true));
  attackers.forEach((attackerState, attacker) => {
    // console.log(attacker + ": " +
    attackerState.tcpConnections.length);
    let openConnections = 0; // All open connections to the
server
    // console.log(attackerState.tcpConnections);
    attackerState.tcpConnections.forEach(connection => {
      openConnections += connection.pktCounts_SYN;
      if (openConnections >= MAX_ALLOWED_CONNECTIONS &&
!connection.portScan) {
        connection.portScan = true;
        let curTimestamp = new Date();

```

```

        analysisResults +=
`[${curTimestamp.toLocaleString()}] ${attacker} connected
multiple times with the same Source Port: ${connection.srcPort}
and Destination Port: ${connection.dstPort} to
${connection.dstIP}. Suspecting OSFP.\n`;
    }
});

    if (openConnections > (MAX_ALLOWED_CONNECTIONS*3) &&
!attackerState.allPortsScan) {
        attackerState.allPortsScan = true;
        let curTimestamp = new Date();
        analysisResults +=
`[${curTimestamp.toLocaleString()}] ${attacker} attempting Port
Scan. Suspecting OSFP.\n`;
    }

    if (attackerState.pktCounts_SYNARST >=
MAX_ALLOWED_SYN_AND_RST && !attackerState.tcp_SYNARST) {
        attackerState.tcp_SYNARST = true;
        let curTimestamp = new Date();
        analysisResults +=
`[${curTimestamp.toLocaleString()}] ${attacker} sending multiple
SYN then RST TCP packets to multiple ports. Suspecting OSFP.\n`;
    }
});
process.send(analysisResults);
}

process.on('message', (args) => {
    let attackers = new Map(JSON.parse(args));
    // console.log(attackers.size);
    // console.log('\n*****Analyzing the collected traffic of all
attackers*****\n');
    attackersAnalysis(attackers);
});

```

ДОДАТОК В

КОД ДЖЕРЕЛА КЛІЄНТА

Основний сценарій, який відображає Клієнта

```
'use
strict'

const electron = require('electron');

const { app, BrowserWindow, Menu, ipcMain, shell } = electron;

let mainWindow; // For ./src/screens/index.html
let osfpWindow; // For ./src/screens/msOne-OSFP.html
let cvesWindow; // For ./src/screens/msOne-CVEs.html
let liveAttackWindow; // For ./src/screens/msTwo.html
let passiveAttactWindow; // For ./src/screens/msThree.html

let targetOS; // For CVEs listings

function createMainWindow () {
  mainWindow = new BrowserWindow({
    width: 1000,
    height: 600
  });
  mainWindow.loadURL(`file://${__dirname}/src/screens/home.html`);
  // mainWindow.webContents.openDevTools();
  mainWindow.on('closed', () => app.quit());

  const mainMenu = Menu.buildFromTemplate(menuTemplate);
  Menu.setApplicationMenu(mainMenu);
}

app.on('ready', createMainWindow);

ipcMain.on('window-OSFP:open', (event, arg) => {
  if (arg) {
    // console.log('main.js receives open-win-osfp msg & opens osfp window');
    osfpWindow = new BrowserWindow({
      // width: 1200,
      // height: 600
      fullscreen: true
    });
    osfpWindow.loadURL(`file://${__dirname}/src/screens/msOne-OSFP.html`);
```

```

        // osfpWindow.webContents.openDevTools();
        osfpWindow.on('closed', () => osfpWindow = null);
    }
});

ipcMain.on('targetOS:set', (event, arg) => {
    // console.log('main.js receives set-target-os msg and opens cves window');
    targetOS = arg;
    cvesWindow = new BrowserWindow({
        width: 1200,
        height: 800
    });
    cvesWindow.loadURL(`file://${__dirname}/src/screens/msOne-CVEs.html`);
    // cvesWindow.webContents.openDevTools();
    cvesWindow.on('closed', () => cvesWindow = null);
});

ipcMain.on('window-CVEs:loaded', (event, arg) => {
    if (arg) {
        cvesWindow.webContents.send('targetOS:get', targetOS);
        // console.log('main.js receives cves-page-loaded msg and sends
targetOS');
    }
});

ipcMain.on('window-LiveAttack:open', (event, arg) => {
    if (arg) {
        liveAttackWindow = new BrowserWindow({
            width: 1200,
            height: 600
            // fullscreen: true
        });
        liveAttackWindow.loadURL(`file://${__dirname}/src/screens/msTwo.html`);
        liveAttackWindow.on('closed', () => liveAttackWindow = null);
    }
});

ipcMain.on('window-PassiveAttack:open', (event, arg) => {
    if (arg) {
        passiveAttactWindow = new BrowserWindow({
            width: 1200,
            height: 600
            // fullscreen: true
        });
    }
});

```

```

passiveAttactWindow.loadURL(`file://${__dirname}/src/screens/msThree.html`);
    passiveAttactWindow.on('closed', () => passiveAttactWindow = null);
    }
});

const menuTemplate = [{
  label: 'Menu',
  submenu: [
    {
      label: 'GitHub Repo',
      click() {
        shell.openExternal('https://github.com/ohmtrivedi/FAU-MS-Thesis-Front-End');
      }
    },
    { type: 'separator' },
    {
      label: 'Quit',
      accelerator: process.platform === 'darwin' ? 'Command+Q' : 'Ctrl+Q',
      click() {
        app.quit();
      }
    }
  ]
}];

if (process.platform === 'darwin') {
  menuTemplate.unshift({});
}

if (process.env.NODE_ENV !== 'production') {
  menuTemplate.push({
    label: 'Developer',
    submenu: [
      { role: 'reload' },
      {
        label: 'Toggle Developer Tools',
        accelerator: process.platform === 'darwin' ? 'Command+Alt+I' :
'Ctrl+Shift+I',
        click(item, focusedWindow) {
          focusedWindow.toggleDevTools();
        }
      }
    ]
  })
}

```

```
    });
  }
}
```

Сценарій для візуалізації екрана MS One (1)

```
'use
strict'
```

```
const electron = require('electron'),
      axios = require('axios'),
      shodanClient = require('shodan-client'),
      net = require('net');

const { ipcRenderer, remote } = electron;

const API_URL = 'ohm.eng.fau.edu:80';
// const API_URL = '10.15.4.110:14000';
// const API_URL = '10.13.210.243:12001';
// const API_URL = '192.168.1.81:12001';
const SHODAN_API_KEY = 'Z7cRqljCHEczyRZQbuG3djUxikmDW6sT';

let newRow, cellRef_IP, cellRef_Location,
    cellRef_Ports, cellRef_OS, cellRef_CVES;

document.addEventListener('keydown', event => {
  let thisWindow = remote.getCurrentWindow();
  switch (event.key) {
    case 'Escape':
      if (thisWindow.isFullScreen()) thisWindow.setFullScreen(false);
      break;
    case 'F11':
      if (!thisWindow.isFullScreen()) thisWindow.setFullScreen(true);
      break;
  }
});

const ipValid = document.getElementById('ipValid');
const table = document.getElementById('results');
const targetIP = document.getElementById('targetIP');
let btnID = 1;

document.querySelector('#btn0').addEventListener('click', (event) => {
  ipcRenderer.send('targetOS:set', 'Microsoft Windows Server 2008 R2 SP1');
  // console.log('From osfp.js: Sending set-target-os msg');
```

```

});

function getIPInfo(queryIP, locationCell, portsCell) {
  shodanClient
    .host(queryIP, SHODAN_API_KEY, { minify: true })
    .then(result => {
      // console.log(result);
      let { city, country_name, ports } = result;
      locationCell.innerHTML = (city ? `${city}, ` : '') + (country_name ?
country_name : '');
      portsCell.innerHTML = (ports ? ports.join(',') : 'Unavailable');
    })
    .catch(err => {
      locationCell.innerHTML = 'Unavailable';
      portsCell.innerHTML = 'Unavailable';
      console.log(err);
    });
}

function getOS(queryIP, osCell, cvesCell) {
  axios.get(`http://${API_URL}/api/msOne/fpOS?target=${queryIP}`)
    .then(response => {
      // console.log(response);
      const { osGuess } = response.data;
      osCell.innerHTML = '<b>' + osGuess + '</b>';
      if (!osGuess.startsWith("Too many")) {
        cvesCell.innerHTML = '<button class="cve_btns" id=btn' + btnID
+ '>Get CVEs</button>';
        cvesCell.addEventListener('click', (event) => {
          ipcRenderer.send('targetOS:set', osGuess);
          // console.log('From osfp.js: Sending set-target-os msg');
        });
        btnID += 1;
      }
    })
    .catch(err => {
      osCell.innerHTML = '<u>Errr, error!</u>';
      console.log(err);
    });
}

document.querySelector('#submitBtn').addEventListener('click', (event) => {
  if(net.isIPv4(targetIP.value)) {
    ipValid.innerHTML = '';
    newRow = table.insertRow(-1);
  }
});

```

```

cellRef_IP = newRow.insertCell(0);
cellRef_Location = newRow.insertCell(1);
cellRef_Ports = newRow.insertCell(2);
cellRef_OS = newRow.insertCell(3);
cellRef_CVES = newRow.insertCell(4);

cellRef_IP.innerHTML = '<h4>' + targetIP.value + '</h4>';
// console.log('From osfp.js: Getting IP details from Shodan...');
getIPInfo(targetIP.value, cellRef_Location, cellRef_Ports);

// console.log('From osfp.js: Sending OS FP request of ' +
targetIP.value);
getOS(targetIP.value, cellRef_OS, cellRef_CVES);
} else {
    ipValid.style.color = 'red';
    ipValid.innerHTML = '<b>Invalid IP, try again!</b>';
}
});

```

Сценарій для візуалізації екрана MS One (2)

```

'use
strict'

const electron = require('electron'),
      axios = require('axios');

const { ipcRenderer, remote } = electron;

const API_URL = 'ohm.eng.fau.edu:80';
// const API_URL = '10.15.4.110:14000';
// const API_URL = '10.13.210.243:12001';
// const API_URL = '192.168.1.81:12001';

document.addEventListener('keydown', event => {
    let thisWindow = remote.getCurrentWindow();
    switch (event.key) {
        case 'Escape':
            if (thisWindow.isFullScreen()) thisWindow.setFullScreen(false);
            break;
        case 'F11':
            if (!thisWindow.isFullScreen()) thisWindow.setFullScreen(true);
            break;
    }
});

```

```

});

const osnameText = document.getElementById('osname');
const table = document.getElementById('cves-results');

document.addEventListener('DOMContentLoaded', function (event) {
  // console.log('From cves.js: DOM fully loaded & parsed');
  ipcRenderer.send('window-CVEs:loaded', true);
  // console.log('From cves.js: Sending did-finish-load msg');
});

ipcRenderer.on('targetOS:get', (event, osName) => {
  // console.log('From cves.js: Received targetOS');
  osnameText.innerHTML = osName;

  // axios.post(`http://${API_URL}/api/getCVEs/`, {
  //   role: 'msOne',
  //   cmd: 'getCVEs',
  //   target: osName
  // })
  axios.get(`http://${API_URL}/api/msOne/getCVEs?target=${osName}`)
    .then((response) => {
      console.log(response);
      let cve_ids = response.data.cve_id;
      let cve_descs = response.data.cve_desc;
      for (let i=0; i < cve_ids.length; i++) {
        let newRow = table.insertRow(-1);
        let cell_id = newRow.insertCell(0);
        let cell_desc = newRow.insertCell(1);
        cell_id.innerHTML = cve_ids[i];
        cell_desc.innerHTML = cve_descs[i];
      }
    })
    .catch(err => {
      console.log(err);
    });
});

```

Сценарій для візуалізації екрана MS Two

```

'use
strict'

```

```

const electron = require('electron'),

```

```

    axios = require('axios');

const { remote } = electron;

const API_URL = 'ohm.eng.fau.edu:80';
// const API_URL = '10.15.4.110:14000';
// const API_URL = '10.13.210.243:12001';
// const API_URL = '192.168.1.81:12001';

document.addEventListener('keydown', event => {
    let thisWindow = remote.getCurrentWindow();
    switch (event.key) {
        case 'Escape':
            if (thisWindow.isFullScreen()) thisWindow.setFullScreen(false);
            break;
        case 'F11':
            if (!thisWindow.isFullScreen()) thisWindow.setFullScreen(true);
            break;
    }
});

let timer1, timer2;

document.querySelector('#startBtn').addEventListener('click', (event) => {
    axios.get(`http://${API_URL}/api/msTwo/start`)
        .then((response) => {
            // console.log(response);
            let { status } = response.data;
            document.querySelector('#sessionStatus').innerHTML = `Status:
<b>${status}</b>`;
            document.querySelector('#summary').style.display = 'none';
            document.querySelector('#logsDiv1').style.display = 'block';
            document.querySelector('#consoleLogsSLA').value = '';
            document.querySelector('#logsDiv2').style.display = 'block';
            document.querySelector('#consoleLogsSFA').value = '';
        })
        .catch(err => {
            throw err;
        });
    timer1 = setInterval(getStatelessAttackLogs, 5000);
    timer2 = setInterval(getStatefullAttackLogs, 70000);
});

document.querySelector('#stopBtn').addEventListener('click', (event) => {
    axios.get(`http://${API_URL}/api/msTwo/stop`)

```

```

        .then((response) => {
            // console.log(response);
            let { status, pktCounts, SLALogs, SFALogs } = response.data;
            document.querySelector('#sessionStatus').innerHTML = `Status:
<b>${status}</b>`;
            document.querySelector('#summary').style.display = 'block';
            document.querySelector('#summary').innerHTML =
processResults(pktCounts);
            document.querySelector('#consoleLogsSLA').value += SLALogs;
            document.querySelector('#consoleLogsSFA').value += SFALogs;
            clearInterval(timer1);
            clearInterval(timer2);
        })
        .catch(err => {
            throw err;
        });
    });

let getStatelessAttackLogs = () => {
    axios.get(`http://${API_URL}/api/msTwo/updateSLA`)
        .then((response) => {
            // console.log(response);
            let { SLALogs } = response.data;
            if (SLALogs.length > 0) {
                document.querySelector('#consoleLogsSLA').value += SLALogs;
            } else {
                console.log('No update (SLA) this time!');
            }
        })
        .catch(err => {
            throw err;
        });
}

let getStatefullAttackLogs = () => {
    axios.get(`http://${API_URL}/api/msTwo/updateSFA`)
        .then((response) => {
            // console.log(response);
            let { SFALogs } = response.data;
            if (SFALogs.length > 0) {
                document.querySelector('#consoleLogsSFA').value += SFALogs;
            } else {
                console.log('No update (SFA) this time!');
            }
        })
}

```

```

        .catch(err => {
            throw err;
        });
    }

let processResults = (pktCounts) => {
    let results = '<b>Summary</b><br>';
    results += `Total packets parsed: <b>${pktCounts.totalPkts}</b><br>`;
    results += `Total TCP Packets: <b>${pktCounts.tcpPkts}</b><br>`;
    results += `Total UDP Packets: <b>${pktCounts.udpPkts}</b><br>`;
    results += `Total ICMP Packets: <b>${pktCounts.icmpPkts}</b><br><br>`;
    results += '<b>Some unusual packets were found as follows, suspecting OS
Fingerprinting attempt!</b><br>';
    results += `NULL Scan: <b>${pktCounts.statelessAttacks.NULLScan}</b><br>`;
    results += `Only PSH Set: <b>${pktCounts.statelessAttacks.PSH}</b><br>`;
    results += `SYN & FIN Set: <b>${pktCounts.statelessAttacks.SYN_FIN}</b><br>`;
    results += `RST & FIN Set: <b>${pktCounts.statelessAttacks.RST_FIN}</b><br>`;
    results += `RST & SYN Set: <b>${pktCounts.statelessAttacks.RST_SYN}</b><br>`;
    results += `FIN & URG Set: <b>${pktCounts.statelessAttacks.FIN_URG}</b><br>`;
    results += `SYN & URG Set: <b>${pktCounts.statelessAttacks.SYN_URG}</b><br>`;
    results += `SYN & FIN & URG Set:
<b>${pktCounts.statelessAttacks.SYN_FIN_URG}</b><br>`;
    results += `X-MAS Scan Set:
<b>${pktCounts.statelessAttacks.XMASScan}</b><br>`;
    results += `SYN & FIN & PSH & URG Set:
<b>${pktCounts.statelessAttacks.SYN_FIN_PSH_URG}</b><br>`;
    results += `SYN & ECN & CWR Set:
<b>${pktCounts.statelessAttacks.SYN_ECE_CWR}<br></b>`;
    results += `ACK & IP-DF Set & Window Size = 1024:
<b>${pktCounts.statelessAttacks.ACK_DF_WS1024}</b><br>`;
    // results += `Without established TCP Session:
<b>${pktCounts.statefullAttacks.noSession}</b><br>`;
    results += `FIN Scan: <b>${pktCounts.statefullAttacks.FINScan}</b><br>`;
    results += `ICMP Packets with invalid code:
<b>${pktCounts.icmpAttacks.invalidCode}</b><br>`;
    results += `ICMP Packets with invalid type:
<b>${pktCounts.icmpAttacks.invalidType}</b><br>`;

    return results;
}

```

Сценарій для візуалізації екрана MS Three

```
'use
strict'
```

```
const electron = require('electron'),
    { exec } = require('child_process'),
    request = require('request'),
    fs = require('fs'),
    axios = require('axios');
// util = require('util');

const { remote } = electron;

const API_URL = 'ohm.eng.fau.edu:80';
// const API_URL = '10.15.4.110:14000';
// const API_URL = '10.13.210.243:12001';
// const API_URL = '192.168.1.81:12001';

document.addEventListener('keydown', event => {
    let thisWindow = remote.getCurrentWindow();
    switch (event.key) {
        case 'Escape':
            if (thisWindow.isFullScreen()) thisWindow.setFullScreen(false);
            break;
        case 'F11':
            if (!thisWindow.isFullScreen()) thisWindow.setFullScreen(true);
            break;
    }
});

document.querySelector('form').addEventListener('submit', (event) => {
    event.preventDefault();
    const { name, path } = document.querySelector('input').files[0];

    let commandToExec = `capinfos -Traec ${path}`;
    // console.log('Executing: ' + commandToExec);
    exec(commandToExec, (error, stdout, stderr) => {
        if (error) throw error;
        else if (stderr) throw stderr;
        else {
            let details = stdout.split('\t');
            document.querySelector('#pcapInfo').innerHTML = `File chosen:
<b>${name}</b><br>`;
            document.querySelector('#pcapInfo').innerHTML += `Number of Packets:
<b>${details[1]}</b><br>`;
        }
    });
});
```

```

        document.querySelector('#pcapInfo').innerHTML += `Start Time:
<b>${details[2]}</b><br>`;
        document.querySelector('#pcapInfo').innerHTML += `End Time:
<b>${details[3]}</b><br>`;
    }
});

let req = request.post(`http://${API_URL}/fileupload`, (err, response) => {
    if (err) throw err;
    else {
        if (response.statusCode == 200) {
            document.querySelector('#fileUpload').style.display = 'block';
            document.querySelector('#fileUpload').style.color = 'green';
            document.querySelector('#fileUpload').innerHTML = '<b>File
Uploaded!</b>';

            axios.get(`http://${API_URL}/api/msThree/analyzePCAP?fileName=${name}`)
                .then((response) => {
                    // console.log(response.data);
                    let { consoleLogs, pktCounts, analysisResults } =
response.data;

                    document.querySelector('#summary').innerHTML =
processResults(pktCounts);
                    document.querySelector('#logsDiv1').style.display = 'block';
                    document.querySelector('#consoleLogsSLA').value =
consoleLogs;

                    document.querySelector('#logsDiv2').style.display = 'block';
                    document.querySelector('#consoleLogsSFA').value =
analysisResults;
                })
                .catch(err => {
                    document.querySelector('#summary').innerHTML = '<b><u>Errr,
error!</u></b>';
                    throw err;
                });
        } else {
            document.querySelector('#fileUpload').style.display = 'block';
            document.querySelector('#fileUpload').style.color = 'red';
            document.querySelector('#fileUpload').innerHTML = '<b>File Upload
failed!</b>';
        }
    }
});

let form = req.form();
form.append('file', fs.createReadStream(path));

```

```

});

let processResults = (pktCounts) => {
  let results = '<b>Summary</b><br>';
  results += `Total packets parsed: <b>${pktCounts.totalPkts}</b><br>`;
  results += `Total TCP Packets: <b>${pktCounts.tcpPkts}</b><br>`;
  results += `Total UDP Packets: <b>${pktCounts.udpPkts}</b><br>`;
  results += `Total ICMP Packets: <b>${pktCounts.icmpPkts}</b><br><br>`;
  results += '<b>Some unusual packets were found as follows, suspecting OS
Fingerprinting attempt!</b><br>';
  results += `NULL Scan: <b>${pktCounts.statelessAttacks.NULLScan}</b><br>`;
  results += `Only PSH Set: <b>${pktCounts.statelessAttacks.PSH}</b><br>`;
  results += `SYN & FIN Set: <b>${pktCounts.statelessAttacks.SYN_FIN}</b><br>`;
  results += `RST & FIN Set: <b>${pktCounts.statelessAttacks.RST_FIN}</b><br>`;
  results += `RST & SYN Set: <b>${pktCounts.statelessAttacks.RST_SYN}</b><br>`;
  results += `FIN & URG Set: <b>${pktCounts.statelessAttacks.FIN_URG}</b><br>`;
  results += `SYN & URG Set: <b>${pktCounts.statelessAttacks.SYN_URG}</b><br>`;
  results += `SYN & FIN & URG Set:
<b>${pktCounts.statelessAttacks.SYN_FIN_URG}</b><br>`;
  results += `X-MAS Scan Set:
<b>${pktCounts.statelessAttacks.XMASScan}</b><br>`;
  results += `SYN & FIN & PSH & URG Set:
<b>${pktCounts.statelessAttacks.SYN_FIN_PSH_URG}</b><br>`;
  results += `SYN & ECN & CWR Set:
<b>${pktCounts.statelessAttacks.SYN_ECE_CWR}<br></b>`;
  results += `ACK & IP-DF Set & Window Size = 1024:
<b>${pktCounts.statelessAttacks.ACK_DF_WS1024}</b><br>`;
  results += `Without established TCP Session:
<b>${pktCounts.statefullAttacks.noSession}</b><br>`;
  results += `FIN Scan: <b>${pktCounts.statefullAttacks.FINScan}</b><br>`;
  results += `ICMP Packets with invalid code:
<b>${pktCounts.icmpAttacks.invalidCode}</b><br>`;
  results += `ICMP Packets with invalid type:
<b>${pktCounts.icmpAttacks.invalidType}</b><br>`;

  return results;
}

```