

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
МОНІТОРИНГУ МЕРЕЖНОГО ТРАФІКУ ЗА ДОПОМОГОЮ
ВЕЙВЛЕТ-АНАЛІЗУ RESEARCH AND SOFTWARE
IMPLEMENTATION OF MONITORING THE NETWORK
TRAFFIC BY USING WAVELET ANALYSIS

Виконав: студент(ка) 2 курсу, групи КНТ-119м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Копієвська І.В.

(прізвище та ініціали)

Керівник Дубровін В.І.

(прізвище та ініціали)

Рецензент Неласа Г.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Копієвської Ірини Валеріївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація моніторингу мережного трафіку за допомогою вейвлетаналізу. Research And Software Implementation Of Monitoring The Network Traffic By Using Wavelet Analysis

керівник проєкту (роботи) Дубровін Валерій Іванович, к.т.н., професор,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 10 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми. 5. Експлуатація, тестування та експериментальне дослідження програми. 6. Економіко-організаційна частина. 7. Охорона праці та безпека в надзвичайних ситуаціях. 8. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Дубровін В.І., професор		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Калініна М.В., асистент		

7. Дата видачі завдання “4” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Розробка економіко-організаційної частини	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

(підпис)

Копієвська І.В.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Дубровін В.І.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
105 с., 9 табл., 35 рис., 2 дод., 31 джерело.

МЕРЕЖЕВІ АТАКИ, ВИЯВЛЕННЯ АНОМАЛІЙ, МЕРЕЖЕВИЙ
ТРАФІК, DOS-АТАКА, ВЕЙВЛЕТАНАЛІЗ.

Об'єкт дослідження – програмні засоби для моніторингу мережного трафіку.

Предмет дослідження – комп'ютерні програми для моніторингу мережного трафіку за допомогою вейвлетаналізу.

Мета роботи – створення програмної системи для моніторингу мережного трафіку та аналізу даних.

Матеріали, методи та технічні засоби: мови програмування Python, JavaScript, персональний комп'ютер з процесором Intel Core i3 під управлінням операційної системи Ubuntu 20.04.

Результати. Було розроблено програмне забезпечення, що дозволяє здійснювати моніторинг мережі, проводити політику доступу пристроїв в мережу Інтернет, відображення підключених пристроїв до локальної мережі та відображення їх IP та MAC адресу, функція логування підключення/відключення пристроїв до локальної мережі.

Висновки. Розроблено програмну систему для моніторинга мережі за допомогою вейвлетаналізу.

Галузь використання – для домашнього користування або для компаній, які потребують контроль та моніторинг трафіку.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 105 pages, 9 tables, 35 figures, 2 appendixes, 31 sources.

NETWORK ATTACKS, DETECTION OF ANOMALIES, NETWORK TRAFFIC, DOS-ATTACK, WAVELET ANALYSIS.

Object of study is a network traffic monitoring software.

Subject of study are computer programs for monitoring network traffic using wavelet analysis.

The purpose of the work is to create a software system for network traffic monitoring and data analysis.

Materials, methods, and tools: Python programming languages, JavaScript, a personal computer with an Intel Core i3 processor running Ubuntu 20.04. Results. The Windows software application is created which powered by the Kinect device, which interacts with the child in the form of a three-dimensional interactive game and motivates it to motion.

Conclusions A software system for network monitoring using wavelet analysis has been developed.

The field of use is home use or for companies that need traffic control and monitoring.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблем та постановка завдань дослідження	10
1.1 Вейвлетаналіз даних	10
1.2 2D та 3D представлення даних про мережевий трафік.....	12
1.3 Постановка проблеми	15
1.4 Постановка завдання роботи.....	16
1.5 Висновки за розділом 1	17
2 Матеріали і методи.....	18
2.1 Мова програмування Python	18
2.2 Мова програмування JavaScript.....	21
2.3 Каскадна таблиця стилів CSS	22
2.4 Мова гіпертекстової розмітки HTML	24
2.5 Інструментарій для управління ізольованими	25
Linux-контейнерами Docker.....	25
2.6 Висновки за розділом 2	27
3 Розробка архітектури програми.....	28
3.1 Діаграма класів	28
3.2 Діаграма класів для опису логічних з'єднань.....	30
3.3 Дерево контекстів	33
3.4 Висновки за розділом 3	34
4 Розробка програми	36
4.1 Основні розроблені алгоритми	36
4.2 Висновок за розділом 4	44
5 Експлуатація, тестування та експериментальне дослідження програми	45
5.1 Тестування WEB User Interface	45
5.2 Тестування скрипту для пошуку активних і неактивних IP.....	46
5.3 Тестування сервера	47

5.4 Симуляція DoS і DDoS атак.....	47
5.5 Висновок за розділом 5	48
6 Економіко-організаційна частина.....	49
6.1 Планування розробки експертної системи.....	49
6.2 Планування розробки додатку.....	49
6.3 Визначення витрат на розробку.....	51
6.4 Розрахунок капітальних вкладень	61
6.5 Оцінка економічної ефективності	62
7 Охорона праці та безпека в надзвичайних ситуаціях	65
7.1 Аналіз потенційних небезпек	65
7.2 Заходи із забезпечення безпеки.....	67
7.3 Заходи із забезпечення виробничої санітарії та гігієни праці.....	69
7.4 Заходи безпеки в надзвичайних ситуаціях	72
Висновки	77
Перелік джерел посилання	78
Додаток А Текст програми.....	81
Додаток Б Слайди презентації	97

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ANN	– штучна нейронна мережа;
CWT	– безперервне вейвлет-перетворення;
DDos	– Distributed Denial of Service;
DWT	– дискретне вейвлет-перетворення;
FFT	– перетворення Фур'є;
IP	– Internet Protocol address;
NN	– нейронна мережа;
STFT	– короткочасне перетворення Фур'є;
TCP	– Transmission Control Protocol.

ВСТУП

Моніторинг та аналіз мереж дуже важливі для того, щоб зрозуміти ефективність мереж, надійність мереж, безпеку мереж та виявити потенційні проблеми. За допомогою аналізу мережевого трафіку служба безпеки мережі зможе ідентифікувати будь-які шкідливі або підозрілі пакети всередині трафіку, тоді як адміністратори мережі можуть контролювати швидкість завантаження, пропускну здатність тощо, а отже, мати краще розуміння мережеві операції. На сьогоднішній день в аналізі трафіку даних мережі було використано багато методів. Нейронна мережа (NN), також відома як штучна нейронна мережа (ANN), використовувалася для прогнозування, а також для виявлення наявності аномалій [1]–[3]. Розпізнавання шаблонів було використано для класифікації даних про дорожній рух та теорія хаосу була використана для кореляції та прогнозування даних часових рядів, а також для ідентифікації нелінійних динамічних поведінок даних про трафік у реальному часі.

Перетворення Фур'є (FFT) та вейвлет-перетворення використовувались для аналізу частотних складових даних про трафік [4]. Основна відмінність між FFT і вейвлет-перетворенням полягає в тому, що вейвлет-перетворенням є локалізовані як за часом, так і за частотою, тоді як стандартне перетворення Фур'є лише локалізоване за частотою. Іншими словами, при вейвлет-перетворенні, коли сталася певна частотна подія, ми можемо знати як про те, яка частотна складова, так і коли це сталося.

1 АНАЛІЗ ПРОБЛЕМ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Вейвлетаналіз даних

Вейвлет - це мала хвиля. Вейвлетаналіз даних базується на вейвлетперетворенні, яке використовувалось для численних досліджень з геофізики, включаючи тропічну конвекцію [5]. Існує багато типів вейвлетів, більшість з яких використовуються для ортогонального вейвлет-аналізу [6], цілеспрямовано створені з особливими властивостями, які роблять їх корисними для обробки сигналів. На рисунку 1.1 показані деякі приклади часто використовуваних вейвлетів.

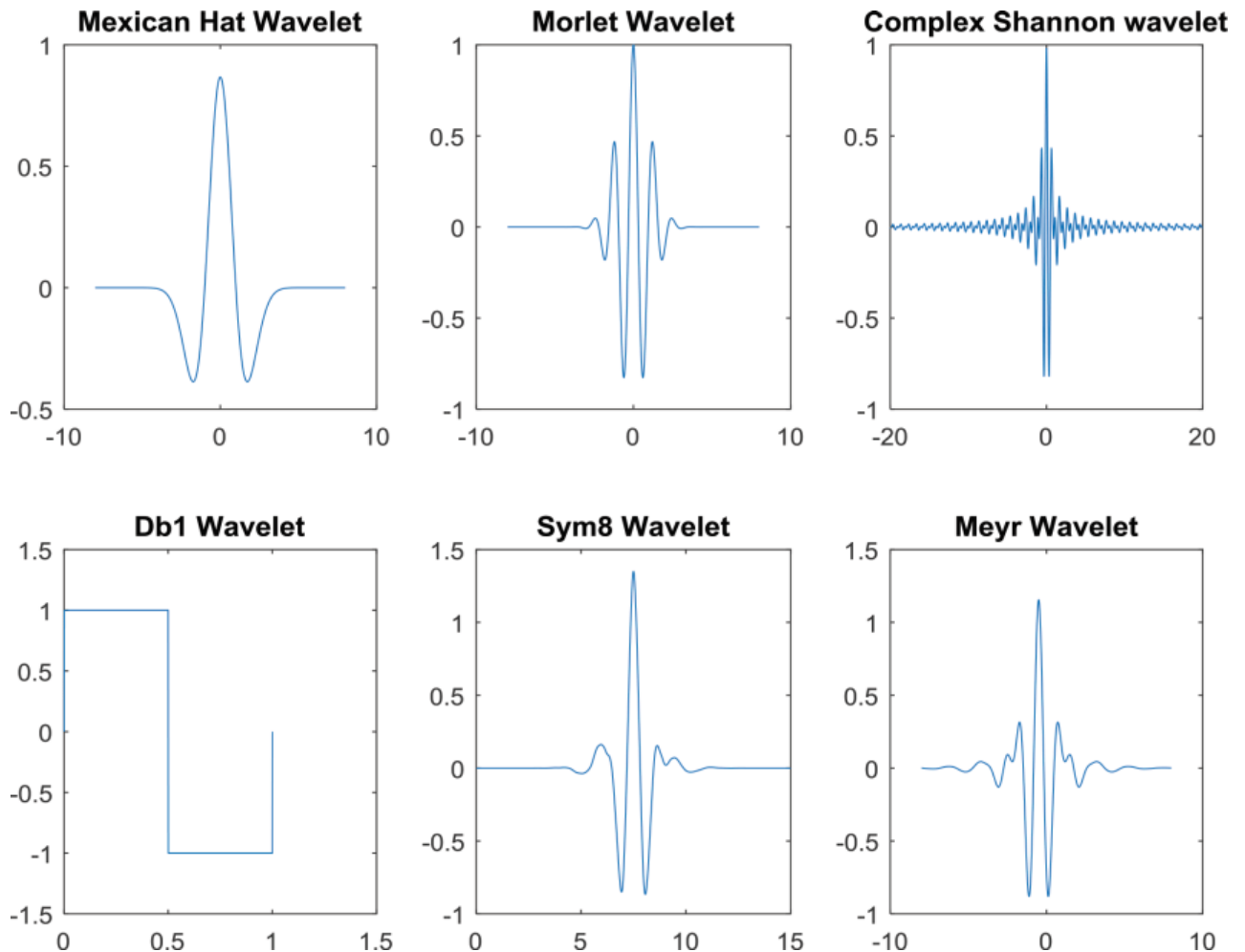


Рисунок 1.1 – Приклади різних типів вейвлетів

Вейвлетперетворення можна використовувати для аналізу часових рядів, що містять нестационарну потужність на багатьох різних частотах. На відміну від традиційних хвиль $\sin(t)$ або $\cos(t)$, які переходять від негативної нескінченності до позитивної нескінченності, вейвлети завжди починаються з нуля, збільшуються, а потім зменшуються назад до нуля. Вейвлетперетворення - це згортка даних часової послідовності та вейвлетів, і воно може бути загалом виражене у формулі (1.1) [7]:

$$F(a, b) = \frac{1}{\sqrt{a}} \cdot \int_{-\infty}^{\infty} f(t) \cdot \psi \left(\frac{t-b}{a} \right) \cdot dt, \quad (1.1)$$

де a - масштаб ($a > 0$);

b – поступальна величина;

t - час, $f(t)$ – дані;

$\psi(t)$ – вейвлет-функція;

* – комплексний спряжений символ.

Вейвлет-перетворення можна загалом поділити на дискретні вейвлет-перетворення (DWT) і безперервні вейвлет-перетворення (CWT).

Дискретне вейвлетперетворення (DWT) - це реалізація вейвлетперетворення з використанням дискретного набору вейвлетшквал. DWT розкладає сигнал на взаємно ортогональний набір вейвлетів, що є основною відмінністю від безперервного вейвлет-перетворення (CWT). DWT може використовуватися для вейвлет-декомпозиції та легкого та швидкого зменшення шумного сигналу.

Безперервне вейвлетперетворення (CWT) - це реалізація вейвлет-перетворення з використанням довільних масштабів та майже довільних вейвлетів. Використовувані вейвлети не є ортогональними, і дані, отримані в результаті цього перетворення, мають високу кореляцію. Щоб наблизити безперервне вейвлет-перетворення, згортку слід зробити N разів для кожної шкали, де N - кількість точок у часовому ряді. Вейвлет-перетворення подібне до перетворення Фур'є, але на відміну від перетворення Фур'є (яке

локалізується лише у частотному просторі), вейвлет-перетворення, локалізоване як у часовому просторі, так і в частотному просторі, див. рис. 1.2 CWT дозволяє користувачам мати змінну роздільну здатність, тобто або високу точність у часі та низьку точність у частоті, або високу точність у частоті та низьку точність у часі. Хоча віконне перетворення, таке як короткочасне перетворення Фур'є (STFT), яке також є кабелем для створення локального аналізу частоти, недоліком STFT є те, що розмір вікна є фіксованим.

У цьому дослідженні вейвлетперетворення використовується для декомпозиції даних, деноізації даних та аналізу частотних компонент, що залежать від часу, за допомогою безперервного вейвлет-перетворення (CWT).

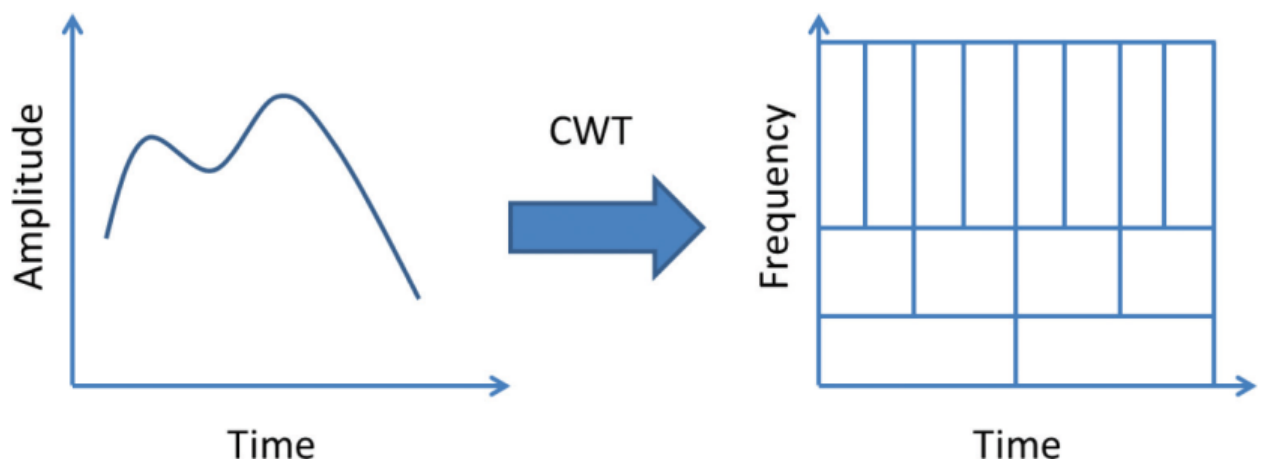


Рисунок 1.2 – Безперервне вейвлетперетворення

1.2 2D та 3D представлення даних про мережевий трафік

На рисунку 1.3 показані загальнодоступні дані про трафік мережі LSBU, записані з інтервалом 1 год, у форматі 3D (вгорі) та 2D-форматі (внизу), вісь X представляє час доби, з 01:00 до 24:00, а вісь Y - день року від 1 до 365, а вісь Z - загальний трафік у Гбіт / с. Дані реєстрували з інтервалом 1 год протягом 1 року, з листопада 2016 р. по листопад 2017 р. Представляючи дані про мережевий трафік у 2D та 3D формуванні, ми можемо краще зрозуміти використання та характеристики мережі.

Результати показують, що загальний мережевий трафік змінюється від сезону до сезону протягом року, а також час від часу змінюється протягом дня. Розуміючи загальну структуру мережевого трафіку, ми можемо краще спланувати мережеві операції, оптимізувати мережу використання та виявлення потенційно підозрілих трафіків.

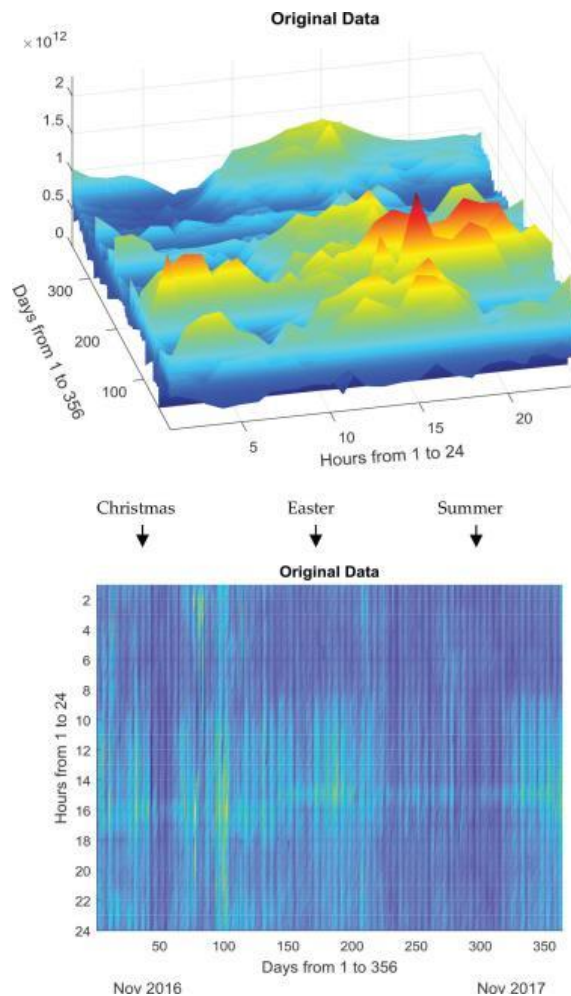


Рисунок 1.3 – 3D-представлення (вгорі) та відповідне 2D-представлення (внизу) загальних даних мережі за 1 рік у режимі щоденного використання (листопад 2016 р. - листопад 2017 р.)

На рисунку 1.4 показані дані про трафік LSBU за 1 рік Всесвітньої павутини (WWW) у форматі 3D (зверху) та відповідна 2D-представлення (знизу). Результати показують, що трафік WWW є дуже сезонним. Він має сильний ефект на тиждень та вихідні, це добре узгоджується з нашими

попередніми дослідженнями [8]. Це також сильно впливає на різдвяні, великодні та літні канікули. Дані про трафік WWW суттєво змінюються протягом доби, найвища - з 10:00 до 19:00, а найнижча - з 06:00 до 09:00, а не опівночі! Крім того, здається, що більше трафіку протягом осіннього семестру (вересень – січень), ніж весняного семестру (лютий – червень).

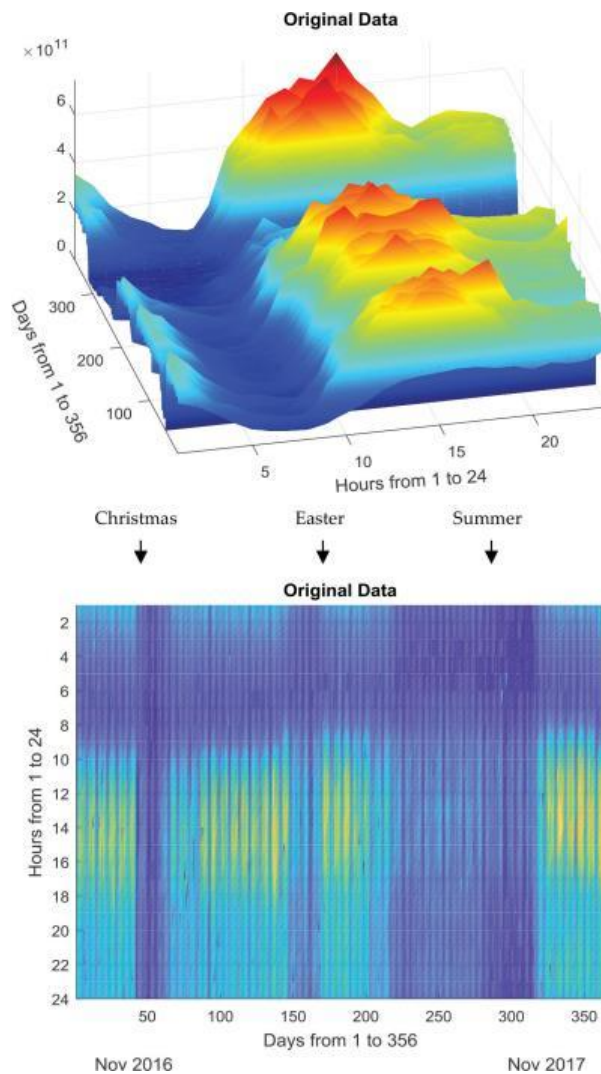


Рисунок 1.4 – 3D-представлення (вгорі) та відповідне 2D-представлення (внизу) однорічних даних про трафік WWW у режимі щоденного використання (листопад 2016 р. - листопад 2017 р.)

На рисунку 1.5 наведені дані про 1-річний трафік електронної пошти у форматі 3D (зверху) та відповідний формат 2D (знизу). Подібно до даних WWW, дані електронної пошти також відображають ефект тижня та

вихідних, а також сезонний ефект. Однак, на відміну від даних WWW, основний з них трафік електронною поштою був між 09:00 та 18:00, ввечері та рано вранці трафіку дуже мало. Тож у ці періоди люди переглядали Інтернет, але не надсилали багато електронних листів. Значний пік в середині графіку обумовлений оновленням електронної пошти, де було надіслано та отримано багато електронних листів.

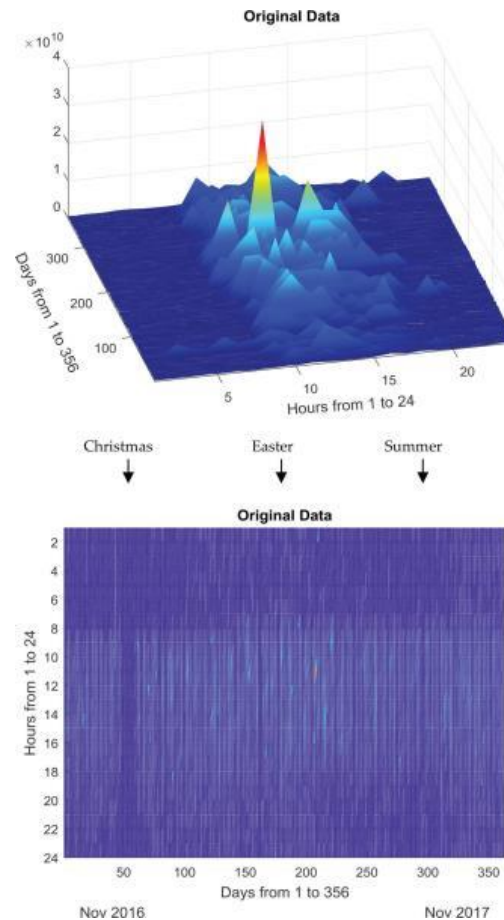


Рисунок 1.5 – 3D-представлення (вгорі) та відповідне 2D-представлення (внизу) 1-річних даних електронної пошти у режимі щоденного використання (листопад 2016 р. - листопад 2017 р.)

1.3 Постановка проблеми

Моніторинг мережі – важливе практичне завдання. Адміністратори постійно прагнуть підтримувати безперебійну роботу своєї мережі. Якщо мережа “впаде” хоча б на короткий період часу, продуктивність компанії

скоротиться і (у випадках організацій, які надають державні послуги) сама можливість надання основних послуг буде поставлена під загрозу. Тому адміністратори мають стежити за рухом мережевого трафіку і продуктивністю на всій мережі, перевіряти та знаходити прогалини в мережевій безпеці.

Одне зі завдань моніторингу – це своєчасне знешкодження DDoS-атак. Проблема боротьби з DDoS-атаками є актуальною, оскільки глобальна мережа Інтернет дедалі ширше використовується у діяльності людини. Розвиток IT-індустрії, розширення комп'ютерних мереж та зростання їх швидкостей спричинили збільшення кількості комп'ютерних атак, зокрема DDoS атак. Із суто інформаційної мережі, призначеної для обміну інформацією електронною поштою і забезпечення доступу до різноманітних віддалених файлових архівів, Інтернет стрімко перетворюється на серйозний ринок послуг, в який інвестуються великі суми грошей. В результаті атаки порушується або повністю блокується обслуговування законних користувачів, мереж, систем та інших ресурсів. Більшість DDoS-атак використовують уразливості в основному протоколі Internet (TCP / IP), а саме спосіб обробки системами запиту, що здійснюється від користувачів.

1.4 Постановка завдання роботи

Метою роботи є експериментальне дослідження, моніторинг і аналіз трафіку та аналіз типів DDoS атак й методів боротьби з ними. У публікації розглянемо типові види DDoS атак, покажемо, як їх моделювати і викривати та розробимо методи захисту від них, відобразимо візуально навантаження на кінцевому обладнанні.

Під час виконання роботи буде використано такі інструменти:

- hping3 – програма для генерації DoS та DDoS атак різних типів;
- Wireshark – програмне забезпечення для аналізу мережесих пакетів Ethernet та інших мереж (сніфер) із вільним вихідним кодом;

- `iptraf` – невелика програма, яка вміє моніторити всю мережеву активність комп'ютера;
- `iptables` – утиліта для командного рядка, стандартизований інтерфейс керування роботою міжмережевого екрана (брандмауєру) Netfilter для ядер Linux від версії 2.4;
- `netfilter` – це міжмережевий екран (брандмауєр), вбудований в ядро Linux з версії 2.4.

Аналізувати навантаження мережевого інтерфейсу планується для перевірки достовірності розроблених методів математичного моделювання мережі.

1.5 Висновки за розділом 1

У першому розділі представлено останнє дослідження щодо використання техніки вейвлет-перетворення для аналізу даних про трафік освітньої мережі. Дані про 2D та 3D представлення мережевого трафіку, тобто трафік за 24 год та 365 днів, допомагають нам краще зрозуміти схему мережевого трафіку. Завдяки вейвлет-перетворенню ми можемо виконувати декомпозицію даних мережевого трафіку та зменшення шуму даних. За допомогою безперервного вейвлет-перетворення (CWT) ми можемо аналізувати дані та показувати, як частотний вміст даних змінюється з часом. Аналіз CWT показує різні характеристики загальних даних про трафік, даних WWW та даних електронної пошти. Інформація про частоту, що залежить від часу, якої не вистачає іншим методам, таким як FFT, дуже корисна для аналізу мережевого трафіку. Використовуючи CWT, ми можемо легко визначити подію, яку в іншому випадку важко ідентифікувати в початковій часовій області.

2 МАТЕРІАЛИ І МЕТОДИ

Для реалізації проєкту я використала такі технології:

- мови програмування: Python, JavaScript;
- каскадна таблиця стилів: CSS;
- мова гіпертекстової розмітки: HTML;
- інструментарій для управління ізольованими Linux-контейнерами:

Docker.

2.1 Мова програмування Python

Python - це мова кодування загального призначення - це означає, що, на відміну від HTML, CSS та JavaScript, він може використовуватися для інших типів програмування та розробки програмного забезпечення, крім веброзробки [9].

Python можна використовувати для таких речей, як:

- розробка вебверсій та програм для мобільних пристроїв (або на стороні сервера);
- розробка настільних додатків та програмного забезпечення;
- обробка великих даних та виконання математичних обчислень;
- написання системних сценаріїв (створення інструкцій, що вказують комп'ютерній системі щось робити).

Переваги Python:

- легко читати, вивчати та писати. Python - це мова програмування високого рівня, що має англоподібний синтаксис. Це полегшує читання та розуміння коду. Python дійсно легко підібрати та вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того самого завдання, порівняно з іншими основними мовами, такими як C / C ++ та Java;

- підвищена продуктивність. Python - дуже продуктивна мова. Завдяки простоті Python розробники можуть зосередитись на вирішенні проблеми. Їм не потрібно витрачати занадто багато часу на розуміння синтаксису чи поведінки мови програмування. Ви пишете менше коду і отримуєте більше справ;

- інтерпретована мова. Python - це інтерпретована мова, що означає, що Python безпосередньо виконує код, рядок за рядком. У разі будь-якої помилки він зупиняє подальше виконання та повідомляє про помилку, що сталася. Python показує лише одну помилку, навіть якщо програма має кілька помилок. Це полегшує налагодження;

- динамічно набраний. Python не знає тип змінної, поки ми не запустимо код. Він автоматично призначає тип даних під час виконання. Програмісту не потрібно турбуватися про оголошення змінних та їх типів даних;

- безкоштовні та з відкритим кодом. Python підпадає під ліцензію з відкритим кодом, затверджену OSI. Це дозволяє використовувати та розповсюджувати безкоштовно. Ви можете завантажити вихідний код, змінити його і навіть поширити свою версію Python. Це корисно для організацій, які хочуть змінити певну поведінку та використовувати свою версію для розробки;

- підтримка величезних бібліотек. Стандартна бібліотека Python величезна, ви можете знайти майже всі функції, необхідні для вашого завдання. Отже, вам не потрібно залежати від зовнішніх бібліотек. Але навіть якщо ви це зробите, менеджер пакетів Python (pip) полегшує імпорт інших чудових пакетів з індексу пакетів Python (PyPi). Він складається з понад 200000 упаковок;

- переносимість. У багатьох мовах, таких як C / C ++, вам потрібно змінити код, щоб запускати програму на різних платформах. Це не те саме з Python. Ви пишете лише один раз і запускаєте його де завгодно. Однак слід бути обережним, щоб не включати будь-які системно залежні функції.

Недоліки Python:

- повільна швидкість. Вище ми обговорювали, що Python - це інтерпретована мова та динамічно набрана мова. Повторне виконання коду часто призводить до повільного виконання. Динамічний характер Python також відповідає за повільну швидкість Python, оскільки він повинен виконувати додаткову роботу під час виконання коду. Отже, Python не використовується для цілей, де швидкість є важливим аспектом проєкту;

- неефективний з пам'яттю. Щоб надати розробнику простоту, Python повинен зробити невеликий компроміс. Мова програмування Python використовує великий обсяг пам'яті. Це може бути недоліком під час створення додатків, коли ми віддаємо перевагу оптимізації пам'яті;

- слабкий рівень мобільних обчислень. Python зазвичай використовується в програмуванні на стороні сервера. Ми не бачимо Python на стороні клієнта чи мобільних додатках з наступних причин. Python не є ефективною пам'яттю і має повільну обробну потужність порівняно з іншими мовами;

- доступ до бази даних. Програмування на Python є простим і без напруги. Але коли ми взаємодіємо з базою даних, вона відстає. Рівень доступу до бази даних Python є примітивним і недостатньо розвиненим у порівнянні з такими популярними технологіями, як JDBC та ODBC. Величезні підприємства потребують плавної взаємодії складних застарілих даних, і тому Python рідко використовується на підприємствах;

- помилки виконання. Як ми знаємо, Python - це мова, що динамічно набирається, тому тип даних змінної може змінюватися будь-коли. Змінна, що містить ціле число, може містити рядок у майбутньому, що може призвести до помилок виконання. Тому програмістам Python потрібно провести ретельне тестування програм.

2.2 Мова програмування JavaScript

JavaScript - це динамічна мова комп'ютерного програмування. Вона легка і найчастіше використовується як частина вебсторінок, реалізації яких дозволяють клієнтському сценарію взаємодіяти з користувачем та створювати динамічні сторінки. Це інтерпретована мова програмування з об'єктно-орієнтованими можливостями.

JavaScript спочатку була відома як LiveScript, але Netscape змінив свою назву на JavaScript, можливо, через хвилювання, породжене Java. JavaScript вперше з'явилася в Netscape 2.0 у 1995 році під назвою LiveScript. Ядро мови загального призначення було вбудовано в Netscape, Internet Explorer та інші веббраузери [10].

Специфікація ECMA-262 визначила стандартну версію основної мови JavaScript.

- JavaScript - це легка, інтерпретована мова програмування;
- призначена для створення мережево орієнтованих додатків;
- доповнює та інтегрує Java;
- доповнює та інтегрує HTML;
- відкритий і крос-платформний.

Переваги JavaScript:

- швидкість. Клієнтський JavaScript дуже швидкий, оскільки його можна негайно запустити в клієнтському браузері. Якщо не потрібні зовнішні ресурси, JavaScript не перешкоджає мережевим викликам до серверного сервера;
- простота. JavaScript порівняно простий у вивченні та реалізації;
- популярність. JavaScript використовується повсюдно в Інтернеті;
- сумісність. JavaScript чудово грається з іншими мовами і може бути використаний у величезній кількості різноманітних додатків;
- навантаження сервера. Будучи на стороні клієнта, зменшується попит на сервері вебсайту;

- дає можливість створювати розширені інтерфейси.

Недоліки JavaScript:

- клієнтська безпека. Оскільки код виконується на комп'ютері користувачів, в деяких випадках його можна використовувати для зловмисних цілей. Це одна з причин, чому деякі люди вирішили відключити Javascript;
- підтримка браузера. JavaScript іноді інтерпретується різними браузерами. Це дещо ускладнює написання коду між браузерами.

2.3 Каскадна таблиця стилів CSS

Каскадні таблиці стилів, які залюбки називають CSS, - це проста мова дизайну, призначена для спрощення процесу зробити вебсторінки презентабельними.

CSS обробляє зовнішній вигляд частини вебсторінки. За допомогою CSS ви можете керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення або кольори використовуються, дизайном макета, варіаціями відображення для різних пристроїв та розмірами екрану а також безліч інших ефектів.

CSS легко вивчити і зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Найчастіше CSS поєднується з мовами розмітки HTML або XHTML [11].

Переваги CSS:

- послідовність. Головною перевагою CSS є те, що стиль послідовно застосовується на багатьох вебсторінках. Один командний рядок може одночасно керувати декількома областями, що є досить вигідним, якщо потрібно внести зміни. Вам потрібно змінити лише одне, а решта піде далі. Оскільки вам не потрібно міняти кожну сторінку по одному, вебдизайнери можуть бути дуже ефективними у створенні та зміні вебсайту лише за допомогою декількох рядків коду;

- покращена швидкість вебсайту. Вебдизайнерам потрібно використовувати лише невелику кількість рядків програмування для кожної сторінки. А якщо коду менше, читати менше рядків, що призводить до швидшого завантаження кожної сторінки. Оскільки користувачі Інтернету не бажають чекати, поки вебсайт завантажиться, покращення швидкості сайту буде найвигіднішим. Власники, які роблять ставку на ефективність вебсайтів для покращення рейтингу пошукових систем та клієнтської бази, отримають вигоду від CSS;

- легкий в обслуговуванні. Каскадна таблиця стилів не тільки спрощує розробку вебсайтів, але й обслуговування. Всі коди розміщені на одній сторінці, а це означає, що вдосконалення або зміна кількох рядків не потребуватиме перегляду кількох сторінок. І оскільки на вебсайті застосовується зміна в одному рядку коду, час і зусилля на технічне обслуговування значно скорочуються.

Недоліки CSS:

- вхід на різні рівні. Існує CSS, CSS 1 до CSS3, що спричинило плутанину серед розробників та веббраузерів. Одного типу CSS має бути достатньо. Було б кращим, ніж вибирати, який рівень CSS використовувати;

- фрагментація. У CSS те, що працює з одним браузером, не завжди може працювати з іншим. Ось чому веброзробники повинні перевірити на сумісність, запустивши програму в декількох браузерах, перш ніж вебсайт буде запущено в реальному часі. Якщо тільки люди використовують Mozilla або Chrome, але вони цього не роблять;

- відсутність безпеки. Оскільки це відкрита текстова система, CSS не має вбудованого захисту, який захищатиме її від перевизначення. Будь-хто, хто має доступ для читання / запису на вебсайті, може змінити файл CSS, змінити посилання або порушити форматування, випадково чи за дизайном.

2.4 Мова гіпертекстової розмітки HTML

HTML розшифровується як мова розмітки гіпертексту. Це дозволяє користувачеві створювати та структурувати розділи, абзаци, заголовки, посилання та цитати для вебсторінок та програм.

HTML не є мовою програмування, тобто він не має можливості створювати динамічну функціональність. Натомість це дає можливість упорядковувати та формувати документи, подібно до Microsoft Word.

Під час роботи з HTML ми використовуємо прості структури коду (теги та атрибути) для розмітки сторінки вебсайту. Наприклад, ми можемо створити абзац, помістивши вкладений текст у початковий тег `<p>` і закриття `</p>` [12].

Переваги HTML:

- HTML широко використовується;
- кожен браузер підтримує мову HTML;
- легкий у вивченні та використанні;
- не потрібно купувати додаткове програмне забезпечення, оскільки воно за замовчуванням є у кожному вікні.

Недоліки HTML:

- HTML може створювати лише статичні та звичайні сторінки, тому, якщо нам потрібні динамічні сторінки, тоді HTML не корисний;
- мені потрібно написати багато коду для створення простої вебсторінки;
- функції безпеки погано працюють з HTML;
- якщо нам потрібно написати довгий код для створення вебсторінки, це створює певну складність.

2.5 Інструментарій для управління ізольованими Linux-контейнерами Docker

Docker - це інструмент, спрощений для створення, розгортання та запуску програм за допомогою контейнерів. Контейнери дозволяють розробнику упакувати програму з усіма необхідними частинами, такими як бібліотеки та інші залежності, та розгорнути її як один пакет. Завдяки цьому, завдяки контейнеру, розробник може бути впевнений, що програма працюватиме на будь-якій іншій машині Linux, незалежно від будь-яких налаштованих параметрів, які може мати машина, які можуть відрізнятися від машини, яка використовується для написання та тестування коду [13].

Переваги Docker:

- рентабельність інвестицій та економія коштів. Перша перевага докерів - це рентабельність інвестицій. Особливо для великих, заснованих компаній, яким необхідно отримувати стабільний дохід у довгостроковій перспективі, рішення є лише кращим, якщо воно може зменшити витрати, одночасно отримуючи прибуток;
- швидке розгортання. Це може зменшити розгортання до секунд. Це через той факт, що він може створити контейнер для кожного процесу і навіть не завантажує ОС. Отже, навіть не турбуючись про витрати на їх повторне збільшення, вони були б вищими, ніж доступні. Дані можна створювати, а також знищувати;
- безпека. Docker гарантує, що програми, що працюють на контейнерах, повністю відокремлені та ізольовані один від одного, з точки зору безпеки, надаючи нам повний контроль над потоком руху та управлінням;
- простота та швидка конфігурація. Те, як Docker спрощує питання, є однією з ключових переваг цього. Це надає користувачам гнучкість для того, щоб взяти власну конфігурацію, вставити її в код і розгорнути її без будь-яких проблем. Однак вимоги до інфраструктури більше не пов'язані із

середовищем програми, оскільки Docker може використовуватися в самих різних середовищах;

- ефективність CI. За допомогою Docker ми можемо створити образ контейнера і надалі використовувати це саме зображення на кожному кроці процесу розгортання. Перевагою цього є можливість розділяти незалежні кроки, а також виконувати їх паралельно. Крім того, тривалість часу, який потрібно від побудови до виробництва, може помітно прискоритись;

- постійна інтеграція. Поки мова йде про безперервну інтеграцію, Docker добре працює як частина своїх конвеєрів разом із інструментами, такими як Тревіс, Дженкінс та Веркер. Ці інструменти можуть зберегти нову версію як образ Docker, кожного разу, коли наш вихідний код оновлюється, просто позначте його номером версії та натисніть на Docker Hub, а потім розгорніть у виробництві.

Недоліки Docker:

- відсутні функції. Зараз виконується маса запитів на функції, таких як самореєстрація контейнера та самоперевірка, копіювання файлів з хосту в контейнер та багато іншого;

- дані в контейнері. Бувають випадки, коли контейнер падає, тому після цього йому потрібна стратегія резервного копіювання та відновлення, хоча у нас є кілька рішень, які не є автоматизованими або ще не дуже масштабованими;

- запускате програми так само швидко, як подача з оголеного металу. У порівнянні з віртуальними машинами, контейнери Docker мають менші накладні витрати, але не нульові. Якщо ми запускаємо додаток безпосередньо на чистому металевому сервері, ми отримуємо справжню швидкість чистого металу навіть без використання контейнерів або віртуальних машин. Однак контейнери не працюють із швидкістю металу;

- забезпечити сумісність між платформами. Основна проблема полягає в тому, що якщо програма, призначена для запуску в контейнері Docker в Windows, то вона не може працювати в Linux або навпаки. Однак

віртуальні машини не підпадають під це обмеження. Отже, це обмеження робить Docker менш привабливим у деяких дуже різномірних середовищах, які складаються як із серверів Windows, так і Linux;

- запускайте програми з графічними інтерфейсами. Загалом, Docker призначений для розміщення програм, які працюють у командному рядку. Хоча у нас є кілька способів (наприклад, пересилання X11), за допомогою яких ми можемо зробити можливим запуск графічного інтерфейсу всередині контейнера Docker, однак це незграбно. Звідси ми можемо сказати, що для програм, які потребують розширених інтерфейсів, Docker не є хорошим рішенням;

- вирішіть усі ваші проблеми безпеки. Простими словами, нам потрібно оцінити ризики безпеки, характерні для Docker, і переконатися, що ми можемо з ними впоратися, перш ніж переносити робочі навантаження в Docker. Причиною цього є те, що Docker створює нові виклики безпеки, такі як складність відстеження декількох рухомих частин у масштабованому динамічному середовищі Docker.

2.6 Висновки за розділом 2

В якості мови програмування для серверної частини проєкту було використано мову програмування Python за її зручність та універсальність.

В якості фреймворку для розроблення клієнтської частини вебсайту було обрано Express.js за його мінімалістичність, але в той же час за велику кількість можливих підключень плагінів.

Для розробки вебпроєкту було обране інтегроване середовище розробки PhpStorm та PyCharm - інтегроване середовище розробки для мови програмування Python.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Діаграма класів

Вводиться поняття логічного з'єднання. Описується модель представлення даних, що задовольняє перерахованим раніше вимогам. На рис. 3.1 зображена діаграма класів, за допомогою яких представляється розібраний мережевий пакет.

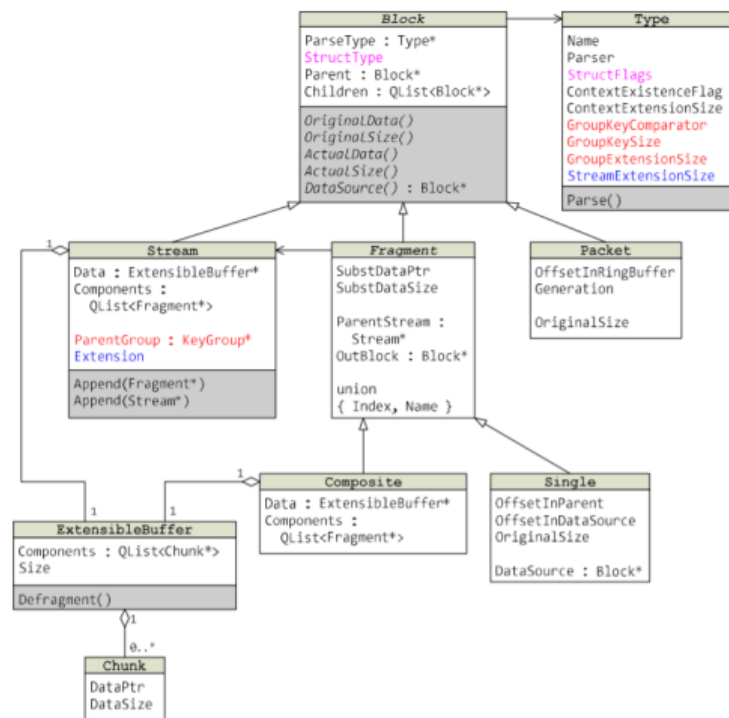


Рисунок 3.1 – Діаграма класів для опису розібраних мережевих пакетів

Виділені в процесі розбору структурні одиниці (мережеві пакети і їх поля) описуються за допомогою блоків (клас Block). Блок представляє масив байт деякого фіксованого розміру. При аналізі блоку може потрібно окремо обробити яку-небудь його частину. Для цього буде створений новий блок, що являється дочірнім по відношенню до початкового батьківського блоку. В процесі аналізу кожному блоку ставиться у відповідність тип розбору (поле ParseType класу Block), що визначає функцію (поле Parser класу Type) для

розбору даних цього блоку. Розбір полягає в розділенні буфера на іменовані поля згідно із заданим форматом пакету. Для блоку визначається базовий формат даних (поле StructType класу Block). Основні базові формати:

- struct - блок містить різнорідні поля, аналог типу struct мови Си (ProtoStruct для випадків, коли описується заголовок мережевого протоколу);
- fieldSeq - послідовність однорідних полів, аналог типу масив мови Си;
- packetSeq - послідовність мережевих пакетів (можливо різних форматів) одного протоколу.

Вводиться два види блоків: блок-потік - для опису відновлених потоків даних, і блок-фрагмент - для представлення цілих пакетів і окремих полів, виділених в потоках даних. Відновлення потоку має на увазі об'єднання полів (як правило, це поле з корисним навантаженням) з різних мережевих пакетів. З цією метою вводиться поняття накопичувального буфера (клас ExtensibleBuffer), що дозволяє об'єднувати як дані блоків-фрагментів, так і дані блоків-потоків. Відмітимо, що для блоку-потіку блок-батько не визначений, оскільки відношення "батько-нащадок" не застосовується для відновлених потоків.

Для розміщення в пам'яті розшифрованих (чи розпакованих) даних вводиться поняття заміщаючого буфера : його розмір фіксований, на відміну від накопичувального буфера, що збільшується у міру додавання даних. Блок-фрагмент з непорожнім заміщаючим буфером називатимемо заміщаючим блоком-фрагментом: подальшому розбору підлягають дані заміщаючого буфера. Дочірні блоки заміщаючого блоку-фрагмента характеризуються зміщенням відносно відповідного заміщаючого буфера. Відмітимо, що розмір заміщаючого буфера ніяк не пов'язаний з розміром початкових (зашифрованих або стислих) даних блоку-фрагмента. За визначенням, блок-потік не може містити заміщаючий буфер.

Заміщаючий блок-фрагмент, таким чином, характеризується первинними (зашифрованими) і фактичними (розшифрованими) даними.

Узагальнюючи ці поняття для блоку, називатимемо фактичними дані блоку, що підлягають подальшому розбору, і первинними дані без урахування можливого заміщення. Для блоку-потoku первинні дані співпадають з фактичними.

Вводиться два види блоків-фрагментів : складений (клас Composite) - для опису дефрагментированных PDU, і поодинокий (клас Single) - для представлення простих полів. Скорочено далі називатимемо складений блок-фрагмент складеним фрагментом, а поодинокий блокфрагмент - поодиноким фрагментом. Дефрагментація PDU, як і відновлення потоку, має на увазі об'єднання полів з різних пакетів, тому складений фрагмент володіє розширюваним буфером (поле Data класу Composite). В той же час, складений фрагмент призначений для опису однієї PDU, тоді як у буфері блоку-потoku, як правило, міститься декілька PDU.

Називатимемо джерелом даних блоку (поле DataSource класу Block) блок, що містить буфер, в якому містяться первинні дані першого. Значить, джерелом даних може виступати блок-потік, складений фрагмент, що заміщає блок-фрагмент. Поодинокий блок локалізується в даних свого джерела за допомогою зміщення (поле OffsetInDataSource класу Single).

Відповідно до структурного типу батьківського блоку блок-фрагмент описує або поле структури (характеризується ім'ям Name), або елемент послідовності (характеризується індексом Index).

3.2 Діаграма класів для опису логічних з'єднань

На рис. 3.2 представлена діаграма класів, за допомогою яких описуються логічні з'єднання.

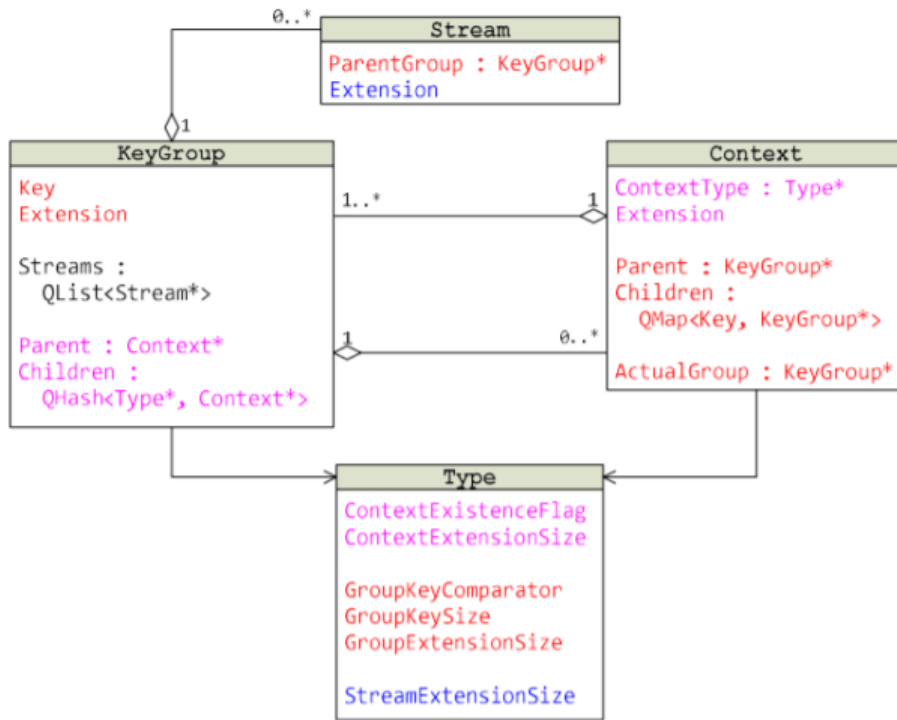


Рисунок 3.2 – Діаграма класів для опису логічних з'єднань

Кожне логічне з'єднання організовано за допомогою одного мережевого протоколу і представляється у вигляді послідовності пакетів цього протоколу. В той же час, на одному рівні стека протоколів можуть бути представлені пакети різних протоколів. Для виділення послідовностей пакетів, що відносяться до свідомо різних логічних сполук, необхідно згрупувати пакети заданого рівня стека відповідно до протоколів цих логічних з'єднань. Група пакетів деякого протоколу у рамках заданого рівня стека описується контекстом (клас Context). Мережевий протокол, на підставі якого був сформований контекст, за визначенням задає тип контексту (поле ContextType класу Context). У розширенні (поле Extension класу Context) контексту може зберігатися додаткова інформація, необхідна для проведення розбору пакетів, що відносяться до цього контексту (наприклад, стан декомпресора при аналізі пакетів протоколу PPP). Тип контексту використовується, зокрема, при побудові ідентифікаторів логічних з'єднань.

Логічне з'єднання протоколу деякого рівня має бути представлене послідовністю відновлених пакетів. Для відновлення (витягання корисного

навантаження) необхідно заздалегідь виділити відповідну підпоследовність з последовності пакетів цього протоколу. Для угруповання пакетів в підпоследовності використовується адресна інформація із службових полів. Кожна підпоследовність описується ключовою групою (клас KeyGroup). Така група в контексті ідентифікується за допомогою унікального ключа (Key), вміст і розмір якого визначаються відповідно до типу контексту. Контекст виступає в ролі контейнера по відношенню до таких груп і надає до них доступ по ключу. Ключова група має розширення (Extension) для збереження інформації, необхідної при проведенні розбору блоків (пакетів) цієї групи. Ключова група фактично є логічним з'єднанням.

Називатимемо відновленим потоком протоколу Р блок-потік, в накопичувальний буфер якого додані дані блоків-фрагментів однієї ключової групи (можливе переупорядковування і/або перекриття блоківфрагментов, відповідно до семантики протоколу Р) контексту з типом, що описує протокол Р. Відповідна ключова група називається батьківською (поле ParentGroup класу Stream) по відношенню до відновленого потоку. У рамках однієї ключової групи підтримується одночасне існування декількох потоків (поле Streams класу KeyGroup) - необхідність в цьому може бути обумовлена як семантикою відповідного протоколу, так і втратою окремих пакетів при передачі.

В результаті відновлення логічних з'єднань різних застосувань, що взаємодіють по мережі, буде побудовано дерево контекстів, ключових груп і відновлених потоків (далі, дерево контекстів) з вершинами трьох типів :

- вершина-контекст задає протокол;
- вершина-група характеризується ключем і описує логічне з'єднання на рівні протоколу батьківської вершини;
- вершина-потік описує відновлений потік даних протоколу вищерозміщеного рівня.

3.3 Дерево контекстів

На рис. 3.3 наведений приклад дерева контекстів, в якому в якості протоколу нижнього рівня, спостережуваного в мережевому з'єднанні, фігурує Ethernet. Ключ для Ethernet- групи будується шляхом конкатенації MACадресов посилача і одержувача (розмір ключа дорівнює 12 байтам). Для протоколу IPv4 ключ групи складається з пари IP- адрес (8 байт), для протоколу TCP - з пари портів (4 байти).

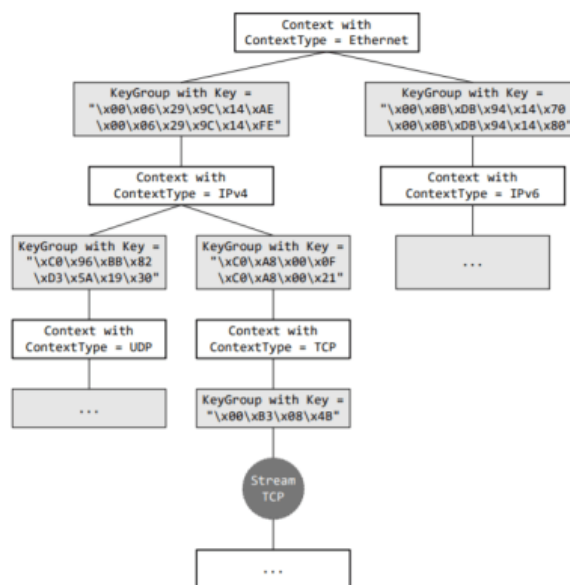


Рисунок 3.3 – Приклад дерева контекстів

Для забезпечення розбору пакетів довільного стека протоколів вводиться поняття розпізнавача. Розпізнавач - це функція, яка за даними блоку і, можливо, деякої додаткової інформації визначає тип розбору цього блоку. Зазвичай функціональність по розпізнаванню потрібно для полів в заголовках протоколів, що допускають дані різних форматів. Наприклад, корисне навантаження в пакеті протоколу IPv4 може містити пакети протоколів TCP, UDP, GRE, ICMP та ін. Службове поле, що визначає вищерозміщений протокол, міститься в IPv4- заголовку. Таким чином, для визначення типу розбору корисного навантаження використовуються дані

блоку-батька. Відповідну категорію розпізнавачів називатимемо розпізнавачами полів.

Для визначення типу розбору відновлених потоків використовується інша категорія розпізнавачів - розпізнавачі потоків з відомим типом складання. Розпізнавач отримує на вхід вміст накопичувального буфера потоку і інформація про тип розбору блоків, дані яких були додані в цей накопичувальний буфер. Розпізнавачі, що приймають на вхід тільки дані буфера, називатимемо універсальними. Такі розпізнавачі застосовуються до блокупотоків, якщо його тип складання невідомий, а також блоку-фрагменту, за умови, що за допомогою розпізнавачів полів не вдалося визначити тип розбору.

Називатимемо потік відновленим повністю, якщо виконані наступні три умови:

- потік має ознаку початку (наприклад, прапор SYN для TCP);
- потік має ознаку кінця (наприклад, прапор FIN для TCP);
- буфер потоку містить усі дані потоку без пропусків. Інакше називатимемо потік частково відновленим.

Називатимемо блок b1 префіксом блоку b2, якщо виконані наступні три умови:

- b1 і b2 належать одній ключовій групі;
- b1 і b2 описують корисне навантаження деякого протоколу P;
- відповідно до семантики протоколу P дані b1 безпосередньо передують даним b2.

3.4 Висновки за розділом 3

У третьому розділі представлені діаграма класів для опису розібраних мережових пакетів, діаграма класів для опису логічних з'єднань та приклад дерева контекстів. Виділені в процесі розбору структурні одиниці (мережеві пакети і їх поля) описуються за допомогою блоків (клас Block). Блок

представляє масив байт деякого фіксованого розміру. Для аналізу блоку створили новий блок, що являється дочірним по відношенню до початкового батьківського блоку. Логічне з'єднання протоколу деякого рівня представлене послідовністю відновлених пакетів. Для відновлення виділили відповідну підпослідовність з послідовності пакетів цього протоколу. Для угруповання пакетів в підпослідовності використовується адресна інформація із службових полів. Наведений приклад дерева контекстів, в якому в якості протоколу нижнього рівня, спостережуваного в мережевому з'єднанні, фігурує Ethernet. Ключ для Ethernet- групи побудували шляхом конкатенації MACадресов посилача і одержувача (розмір ключа дорівнює 12 байтам). Для протоколу IPv4 ключ групи складається з пари IP- адрес (8 байт), для протоколу TCP - з пари портів (4 байти).

4 РОЗРОБКА ПРОГРАМИ

4.1 Основні розроблені алгоритми

В ході розроблення ПП було реалізовано наступні функції:

- авторизація користувача;
- пошук активних IP;
- відображення активних пристроїв за допомогою вебінтерфейсу;
- можливість заблокувати сторонній IP;
- докеризація проєкту.

4.1.1 Авторизація користувача

В моєму ПП проходить наступний процес аутентифікації:

- користувач вводить ім'я та пароль;
- додаток перевіряє, чи є вони коректними;
- якщо ім'я і пароль коректні, додаток відправляє заголовок Set-Cookie, який буде використовуватися для аутентифікації подальших сторінок.

На рисунку 4.1 зображений фрагмент коду авторизації користувача.

```

17
18 // http authentication
19 let auth = require("basic-auth");
20 let admins = { // define admins here
21   'admin': { password: 'password' }
22 };
23
24
25 // index page
26 app.get('/', function(req, res) {
27   let user = auth(req);
28   if (!user || !admins[user.name] || admins[user.name].password !== user.pass) {
29     res.set('WWW-Authenticate', 'Basic realm=Network Monitor');
30     return res.status(401).send();
31   } else {
32     ConnectionHandler.get_connections(redisClient, function(err, connections) {
33       res.render('index', {
34         connections: connections
35       });
36     });
37   }
38 });
39

```

Рисунок 4.1 – Авторизація користувача

Коли користувач відвідує сторінки в тому ж домені, раніше встановлений cookie буде додано до всіх запитів. Аутентифікація на закритих сторінках відбувається за допомогою цього файлу cookie.

4.1.2 Пошук активних IP

Для пошуку активних IP я використовувала язык програмування Python. При запуску проекту запускається python script, який сканує мережу, в якій підключений додаток. Цей скрипт працює у фоновому режимі, який постійно моніторить мережу та в разі наявності нового активного IP, додає його до файлу mac_addresses.json. Цей файл представляє собою звичайний json та має наступний вигляд, див. рис. 4.2.

```
1  [
2    {
3      "device": "Home computer",
4      "MAC": "11:22:33:44:55:66"
5    }
6  ]
7  |
```

Рисунок 4.2 – Файл mac_addresses.json

Для пошуку активних IP в локальній мережі, я використовувала бібліотеку scapy. З її допомогою я сканувала всі можливі IP в моїй локальній мережі. Якщо IP відповідав на запит, то я обробляла це IP та додавала його в файл mac_addresses.json. Для запиту до IP, я розробила функцію monitor_loop. Вона має наступний вигляд, див.рис. 4.3.

```

154
155 def monitor_loop(gateway_ip, ip_range):
156     """
157     main monitor loop of the program. checks for new connections and
158     disconnections.
159     """
160     global loop_seconds
161     while True:
162         try:
163             new_connections = check_connected(gateway_ip, ip_range)
164             new_disconnections = check_for_disconnections()
165             if new_connections or new_disconnections:
166                 write_connections()
167                 time.sleep(loop_seconds)
168         except KeyboardInterrupt:
169             print "shutting down..."
170             sys.exit()
171

```

Рисунок 4.3 – Функція для пошуку IP

Функція приймає два параметри: `gateway_ip`, `ip_range`, де `gateway_ip` – це IP з якого запущено додаток, а `ip_range` – це діапазон сканування IP адресів. Функція сканує всі IP адреси та визначає підключені/відключені IP адреси. Коли функція знаходить нову IP адресу, вона її обробляє та додає до файлу `mac_addresses.json` новий об'єкт. Цей об'єкт містить в собі наступні поля:

- а) Device – назва пристрою;
- б) MAC – MAC адреса пристрою.
- в) скрипт `check_connection.py` має змінні для налаштування конфігурації роботи скрипту, а саме:

- 1) `loop_seconds` – період для запуску циклу;
- 2) `disconnect_time` - час очікування, перш ніж клієнт буде відключений;

- 3) `__checking` - прапорець, який вказує, помічати нові підключення чи ні;
- 4) `__broadcast` - перемикач для трансляції на вебслужбу;
- 5) `__writing` - перемикач для запису підключень до файлу;
- 6) `__speak` - прапор для розмови під час підключення нового пристрою;
- 7) `__log_unknown` - щоб вказати, чи слід реєструвати невідомі пристрої;
- 8) `__printing` - щоб вказати, чи сценарій друкує вихідні дані;
- 9) `connected_file` - файл для реєстрації з'єднань;
- 10) `log_file` - файл журналу для невідомих пристроїв;
- 11) `mac_file` - файл для mac-адрес;
- 12) `web_service_url` - URL вебслужби.

4.1.3 Відображення активних пристроїв за допомогою вебінтерфейсу

Для вебінтерфейсу та серверу я використовувала NodeJS. Для NodeJS я використала фреймворк Express. Express - це мінімальна та гнучка платформа веб-додатків Node.js, яка забезпечує надійний набір функцій для веб- і мобільних додатків. Налаштування Express та NodeJS серверу зображено на рисунку 4.4.

```

1 // server.js
2
3 // dependencies
4 let bodyParser = require('body-parser');
5 let express = require('express');
6 let fs = require('fs');
7 let logger = require('morgan');
8 let redis = require('redis');
9 let app = express();
10 let server = require('http').createServer(app);
11
12
13 server.listen(3000);
14
15

```

Рисунок 4.4 – Налаштування серверу NodeJS з фреймворком Express

Для того щоб сервер мав змогу коректно обробляти відповіді в форматі Json я використала бібліотеку body-parser. Бібліотека body-parser зображена на рисунку 4.5.

```

27 // body parser
28 app.use(bodyParser.json());
29 app.use(bodyParser.urlencoded({ extended: true }));
30

```

Рисунок 4.5 – Бібліотека body-parser

Для того щоб відобразити вебінтерфейс користувача я використовувала NodeJS pug, який представлено на рисунку 4.6.

```

1  div(class="container")
2    head
3      <meta charset="utf-8">
4      <meta http-equiv="X-UA-Compatible" content="IE=edge">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      title Network Monitor
7      link(rel="stylesheet", href="stylesheets/style.css")
8      link(rel="stylesheet", href="https://bootswatch.com/cosmo/bootstrap.min.css")
9      link(rel="stylesheet", href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js")
10     script(src="http://code.jquery.com/jquery-latest.min.js" type="text/javascript")
11     script(src="/socket.io/socket.io.js")
12     script(src="/scripts/mysocket.js" type="text/javascript")
13   body
14     div(class="navbar navbar-default navbar-fixed-top")
15       div(class="container")
16         div(class="navbar-header")
17           a(href="#" class="navbar-brand") Network Monitor
18     div(class="page")
19       div(class="well")
20         h1 Network
21         div(class="container")
22           div(class="row-fluid")
23             div(class="col" id="devices")
24               each connection in connections
25                 div(class=connection['connected'] == "1"? 'device connected' : 'device disconnected')
26                 div(class="alias")
27                   p= connection["alias"]
28                 div(class="ip")
29                   p= connection["ip"]
30                 div(class="mac")
31                   p= connection["MAC"]
32                 div(class="timestamp")
33                   p= connection["timestamp"]
34               else
35                 div(class="col" id="devices_error")
36                   p No devices connected
37

```

Рисунок 4.6 – Index pug файл

Pug компілюється в HTML і CSS та відображається користувачеві. Pug - це шаблонізатор і html-препроцесор, написаний на javascript для node.js. Pug має цілий ряд переваг у порівнянні зі звичайним HTML.

4.1.4 Блокування сторонніх IP

У моєму додатку користувач має функцію блокування IP адрес. Ця функція потрібна для того, щоб блокувати сумнівні IP адреси або якщо користувач хоче заблокувати конкретну IP адресу.

Всі заблоковані IP адреси повинні знаходитись у файлі blacklist.txt. Файл чорного списку повинен бути звичайним текстовим файлом з IP-адресою в кожному рядку, наприклад:

- 192.168.0.107;
- 192.168.0.109;
- 192.168.0.110.

Заблокування IP адреси відповідає функція block_ip_addr. Вона має наступний вигляд, який представлено на рисунку 4.7.



```
def block_ip_addr(ip_addr):  
    file = open('./resources/blacklist.txt', "w")  
    file.write(ip_addr + "\n")  
    file.close()
```

Рисунок 4.7 – Функція block_ip_addr

Також додаток автоматично запускає скрипт, який працює в фоновому режимі та перевіряє вхідний трафік. Цей скрипт перевіряє наявність DoS та DDoS атак і в разі їх виявлення повідомляє про це користувача. Цей скрипт має наступний вигляд, представлено на рисунку 4.8.

```

1 import socket
2 import struct
3
4 from datetime import datetime
5
6
7 s = socket.socket(socket.PF_PACKET, socket.SOCK_RAW, 8)
8
9 dict = {}
10
11 file_txt = open("attack_DDoS.txt", 'a')
12 t1 = str(datetime.now())
13
14 file_txt.writelines(t1)
15 file_txt.writelines("\n")
16
17 No_of_IPs = 15
18 R_No_of_IPs = No_of_IPs + 10
19 while True:
20     pkt = s.recvfrom(2048)
21     ipheader = pkt[0][14:34]
22     ip_hdr = struct.unpack("!8sB3s4s", ipheader)
23     IP = socket.inet_ntoa(ip_hdr[3])
24     print "The Source of the IP is:", IP
25
26 if dict.has_key(IP):
27     dict[IP] = dict[IP] + 1
28     print dict[IP]
29
30 if (dict[IP] > No_of_IPs) and (dict[IP] < R_No_of_IPs) :
31     line = "DDoS attack is Detected: "
32     file_txt.writelines(line)
33     file_txt.writelines(IP)
34     file_txt.writelines("\n")
35 else:
36     dict[IP] = 1

```

Рисунок 4.8 – Скрипт для виявлення DoS та DDoS атак

Насправді DDoS-атаку трохи складно виявити, тому що ми не знаємо, хост, який відправляє трафік, підроблений або справжній. Наведений вище Python скрипт допоможе виявити DDoS-атаку.

Якщо кількість запитів для одного IP в об'єкті dict перебільшує 15, то ми припускаємо, що це атака. Згідно зі сценарієм, якщо IP-адреса потрапить понад 15 разів, то додаток повідомить про це користувача спливаючим вікном, в якому буде вказана IP адреса та цю адресу буде додано до чорного списку.

4.1.5 Докеризація проєкту

Для зручності розробки проєкту та зручності розгорнення цього проєкту на іншому комп'ютері, я використала Docker. Docker — інструментарій для управління ізольованими Linux-контейнерами. Docker доповнює інструментарій LXC більш високорівневим API, що дозволяє керувати контейнерами на рівні ізоляції окремих процесів.

Мій додаток має три докер контейнера:

- докер контейнер для node.js серверу;
- докер контейнер для check_connection.py скрипту;
- докер контейнер для сервісу Redis.

Для того щоб об'єднати всі контейнери разом я створила файл docker-compose.yml, який містить інформацію для створення перерахованих вище контейнерів. Цей файл має наступний вигляд, який представлено на рисунку 4.9.

```

1  script:
2    build: ./src
3    volumes:
4      - ./src:/src
5    net: "host" # so that the python script can POST data to localhost on host machine
6
7  redis:
8    image: redis
9    ports:
10     - "6379"
11
12  mynodeapp:
13    build: ./node_app
14    ports:
15     - "3000:3000"
16    links:
17     - redis

```

Рисунок 4.9 – Файл docker-compose.yml

У файлі docker-compose.yml поле build – це Dockerfile. Dockerfile містить інформацію про те, як конкретний контейнер має бути зібраний. Наприклад Dockerfile для script контейнеру зображено на рисунку 4.10.

```

1  FROM ubuntu:14.04
2  MAINTAINER Iryna Kopyievska
3  RUN apt-get update
4  RUN apt-get install -y python python-pip tcpdump
5  RUN pip install --upgrade pip
6  RUN pip install scrapy
7  RUN pip install pytsx
8  ENV TERM xterm
9
10 CMD ["/src/check_connections.py","-f", "/src/resources/mac_addresses.json","-e","/src/resources/blacklist.txt"]

```

Рисунок 4.10 – Dockerfile для script контейнеру

В цьому файлі я спочатку завантажую образ ubuntu:14.04 з DockerHub.com. Це є офіційний образ системи ubuntu для Docker. Потім я встановлюю необхідні залежності для ubuntu. Після цього я встановлюю необхідні пакети для Python за допомогою pip. Pip – це система керування пакунками, яка використовується для встановлення та управління програмними пакетами, які написані на Python. В кінці файлу ми запускаємо наш скрипт з аргументами *-f* та *-e*.

4.2 Висновок за розділом 4

Для реалізації мого проєкту я використала сучасні технології, так як: NodeJS, Express, Python, Redis, Docker. Для серверу NodeJS реалізовано авторизацію. Для написання скрипту, що зчитує IP адреси, я використовувала модуль програмування Python та для зручності бібліотеку scrapy. Для того, щоб проєкт було легше покращувати та передавати іншому розробнику, я докеризувала мій додаток. Docker містить контейнери для основних складових додатку, таких як: NodeJs сервер, операційна система Ubuntu, в якій запущено скрипт для зчитування IP адрес та сервісу redis для зберігання інформації.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ
ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Тестування WEB User Interface

Після завершення розробки додатку, я приступила до його тестування. Першим кроком було тестування Web UI. Перш за все я перевірила вигляд додатку на всіх браузерях, приділяючи достатньо уваги Edge та Safari. Для тестування Web UI я використовувала Chrome dev tools. За допомогою цього інструменту, я перевірила адаптивність для мобільних телефонів, планшетів, десктопу та дисплею з високим розширенням.

Web UI зображено на рисунку 5.1.

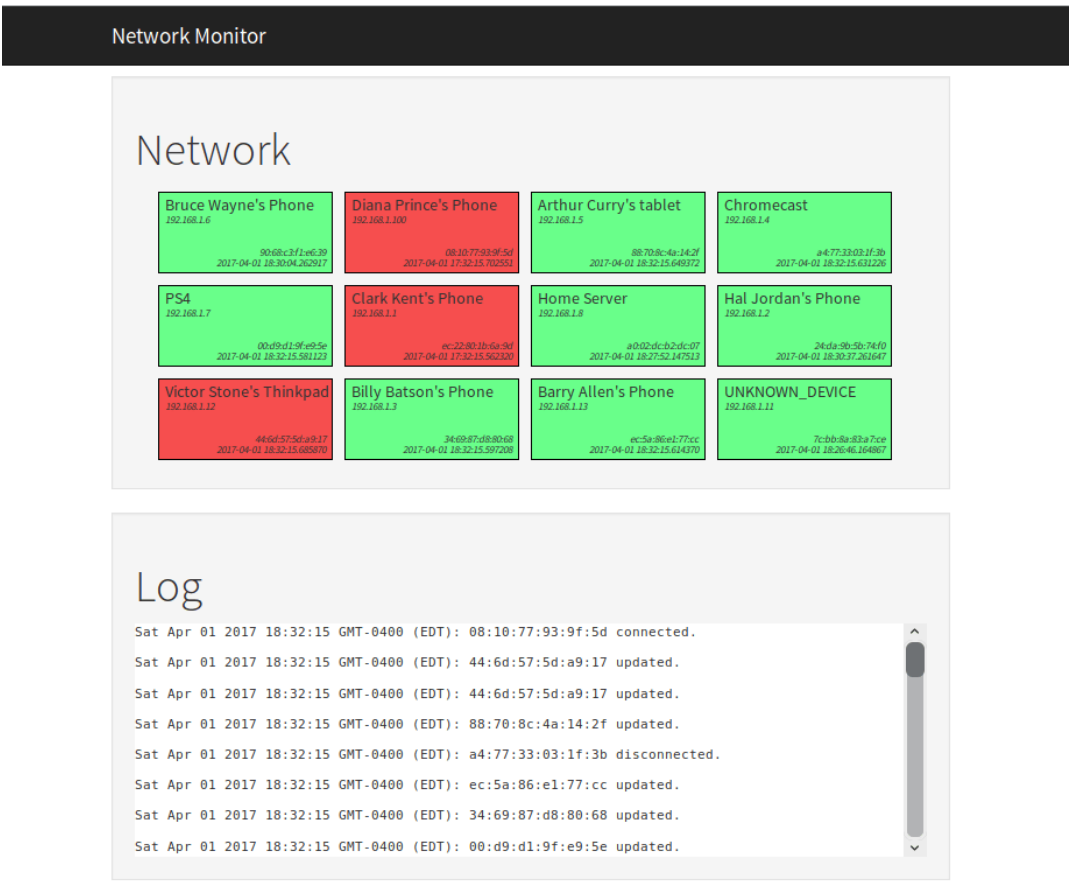


Рисунок 5.1 – Web UI

На наступному екрані відображається активні та неактивні IP адреси в локальній мережі. Також відображається назва пристрою, їх MAC адреса й дата останньої активності.

Після блоку активних та неактивних IP, відображається лог активності. Він зберігається в файлі log.txt. Для його тестування я виконала наступні кроки:

- запустила додаток;
- підключила новий пристрій до локальної мережі;
- відключила пристрій від локальної мережі.

Після цього в логові з'явилися два нових записи. Перший – це був запис про знайдений новий пристрій та його підключення до локальної мережі. Другий – це запис про його відключення від локальної мережі. Після відключення пристрою від локальної мережі, колір його блоку змінився на червоний.

5.2 Тестування скрипту для пошуку активних і неактивних IP

Після того, як ми запустили додаток, автоматично запускається скрипт, який в автоматичному режимі сканує локальну мережу та знаходить активні й неактивні IP. Для тестування цього скрипту, я перевірила, що докер контейнер створився коректно. Для цього я використала наступну команду – `sudo docker ps -a`. Ця команда відобразила інформацію, зображену на рисунку 5.2.

```
developer@developer-user:~/projects/Network-Monitor$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
702c65735332	network-monitor_mynodeapp	"docker-entrypoint.s..."	7 days ago	Up 22 seconds	0.0.0.0:3000->3000/tcp	network-monitor_mynodeapp_1
f674b3d4eebf	network-monitor_script	"/src/check_connecti..."	7 days ago	Up 23 seconds		network-monitor_script_1
5b8dc2261933	redis	"docker-entrypoint.s..."	7 days ago	Up 23 seconds	0.0.0.0:32768->6379/tcp	network-monitor_redis_1

```
developer@developer-user:~/projects/Network-Monitor$
```

Рисунок 5.2 – Функція для відображення активних докер контейнерів

На цьому скріншоті ми бачимо інформацію про те, що контейнер зі скриптом був запущений та працює.

Для перевірки того, що функції скрипту та сам скрипт знаходяться в докер контейнері, ми використали наступну команду – `sudo docker exec -it f6 bash`. За допомогою цієї команди, ми заходимо в контейнер з скриптом, після цього ми переконались в тому, що скрипт знаходиться в контейнері.

Для тестування коректної роботи скрипта, ми підключили новий пристрій до локальної мережі після запуску додатку. Пристрій з'явився у вебдодатку та у файлах `src/resource/mac_adresses.json`.

5.3 Тестування сервера

Мій сервер відповідає за відображення вебдодатку та зв'язку з даними, які обробляються та записуються до необхідних файлів за допомогою скрипту. Для тестування сервера, мені потрібно впевнитись, що докер контейнер коректно почав роботу. Для цього я використала команду – `sudo docker ps -a` та впевнилась, що контейнер піднявся та працює. Після перевірки коректності роботи докер контейнеру, потрібно перевірити, що сервер працює на `localhost:3000`. Для цього я скористалася веббраузером Google Chrome, ввела в пошукову строку `localhost:3000` і мені відобразився мій додаток. Це означає, що сервер коректно працює.

5.4 Симуляція DoS і DDoS атак

Для того, щоб симулювати DoS і DDoS атаки, я написала наступний скрипт. Цей скрипт симулює DoS і DDoS атаки з різних IP та різних портів. Після запуску скрипту, до чорного списку були додані IP, з яких була атака.

На рисунку 5.3 зображена функція, яка потрібна для відображення ативних докер контейнерів.

```

import random
from scapy.all import *
target_IP = input("Enter IP address of Target: ")
i = 1

while True:
    a = str(random.randint(1,254))
    b = str(random.randint(1,254))
    c = str(random.randint(1,254))
    d = str(random.randint(1,254))
    dot = "."
    Source_ip = a + dot + b + dot + c + dot + d

    for source_port in range(1, 65535):
        IP1 = IP(source_IP = source_IP, destination = target_IP)
        TCP1 = TCP(srcport = source_port, dstport = 80)
        pkt = IP1 / TCP1
        send(pkt, inter = .001)

    print ("packet sent ", i)
    i = i + 1

```

Рисунок 5.3 – Функція для відображення активних докер контейнерів

5.5 Висновок за розділом 5

Для тестування додатку мені знадобились наступні інструменти: google chrome dev tools та команди докер для перевірки коректної роботи контейнерів та можливості входу до контейнеру. Додаток пройшов перевірку на працезданість web ці, скрипту для пошуку IP адресів та роботи серверу. В ході перевірки вебдодатку не було знайдено помилок, які впливають на роботу основних функцій додатку.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

6.1 Планування розробки експертної системи

Приводиться економічне обґрунтування програмної реалізації додатку, який дозволить людині відстежувати мережний трафік, здійснювати пошук активних IP, блокувати сторонні IP та виконувати моніторинг активних пристроїв.

Цей програмний продукт має підвищити якість і швидкість отримання інформації про мережний трафік, підключені безпечні IP та можливі атаки зі сторонніх IP. Крім того користувач має змогу додавати активні пристрої та видаляти їх.

Очікується отримання прибутку за рахунок купівлі підписки на розроблюваний продукт користувачів.

6.2 Планування розробки додатку

Всю сукупність планування, аналізу та розробки додатку можна розділити на декілька головних етапів. Для кожного із цих етапів потрібно вказати трудомісткість проєкту, кількість учасників, які виконували розробку додатку та їх тривалість роботи.

У дослідженні та розробці представленого вебдодатку беруть участь два виконавця: програміст, робота якого буде оцінюватись в два з половиною місяців та консультант, який пів місяця буде виконувати свою частину роботи. Початок дослідження та розробки вебдодатку припадає на 2 вересня і весь процес роботи триватиме до 15 листопада 2020 року.

Етапи дослідження та розробки продукту представлені нижче в таблиці 6.1.

Таблиця 6.1 – Етапи розробки продукту

Назва роботи	Трудомісткість		Виконавці	Тривалість, днів
	люд.— дні	% до підсумку		
1. Аналіз предметної області	8	9,23	Програміст, консультант	6
2. Визначення вимог до програмного продукту	5	6,15	Програміст, консультант	3
3. Аналіз принципів побудови експертних систем	9	10,77	Програміст	7
4. Дослідження методів вирішення задач	8	10,77	Програміст	7
5. Побудова моделі системи	5	9,23	Програміст	5
6. Створення програмного коду	20	38,46	Програміст	19
7. Тестування і налагодження програмного продукту	5	9,23	Програміст, консультант	4
8. Складання програмної документації	5	6,15	Програміст, консультант	4
Разом	65	100		55

Дані, які аналізували та привели у таблиці 6.1, ми можемо дослідити та за їх допомогою створити стрічковий графік планування аналізу і розробки вебдодатку.

У таблиці 6.2 наведений стрічковий графік аналізу та планування розробки продукту (сірий колір – консультант, чорний – програміст).

Таблиця 6.2 – Стрічковий графік

Дата / Найменування роботи	02.09.2020	09.09.2020	16.09.2020	23.09.2020	30.09.2020	07.10.2020	14.10.2020	21.10.2020	28.10.2020	04.11.2020	11.11.2020
1.Аналіз предметної області	■										
2. Визначення вимог до програмного продукту	■										
3. Аналіз принципів побудови експертних систем		■									
4. Дослідження методів вирішення задач			■								
5. Побудова моделі системи					■						
6. Створення програмного коду						■	■	■	■	■	
7. Тестування і налагодження програмного продукту											■
8. Складання програмної документації											■

6.3 Визначення витрат на розробку

Для того, щоб визначити всі витрати на розробку додатку потрібно підрахувати кошти вартості робіт, яка містить такі статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок;
- матеріали і комплектуючі вироби;
- витрати на спеціальне обладнання;
- накладні витрати.

Для правильного розрахунку заробітної плати, потрібно мати на увазі час, витрачений на розробку додатку, який становить 55 днів.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – кількість днів;

Q – трудомісткість;

R – кількість робітників (люд.).

6.3.1 Розрахунок основної заробітної плати

У наведеній нижче таблиці 6.3 представлений підрахунок основної заробітної плати.

Таблиця 6.3 – Основна заробітна плата

Посада виконавця	Чисельність, осіб	Місячний оклад, грн	Кількість місяців	Сума ЗП, грн
Програміст	1	12000	2,5	30000
Консультант	1	8000	0,5	4000
Підсумок	2			34000

6.3.2 Розрахунок додаткової заробітної платні

Додаткова заробітна плата – це 10 частина основної заробітної плати працівника і обчислюється вона за формулою (6.2):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1. \quad (6.2)$$

Проведемо розрахунок:

$$ЗП_{\text{дод}} = 34000 \cdot 0,1 = 3400 \text{ грн.}$$

6.3.3 Відрахування на єдиний соціальний внесок

Відрахування на єдиний соціальний внесок – це 22 відсотки від суми основної та додаткової заробітної плати. І обчислювати його потрібно за такою формулою (6.3):

$$ВІД = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22, \quad (6.3)$$

де $ЗП_{\text{осн}}$ – основна заробітна плата (грн.);

$ЗП_{\text{дод}}$ – додаткова заробітна плата (грн.).

Нижче наведена сума відрахування на єдиний соціальний внесок за формулою (6.3).

$$ВІД = (30000 + 3000) \cdot 0,22 = 7260 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становлять 7260 грн.

6.3.4 Визначення витрат на матеріали

Додаткові витрати на матеріали підраховуються за нижче наведеною формулою (6.4):

$$M = \sum_{i=1}^n (Ц_i N_i (1 + K_{\text{т.з.}}) - Ц_{io} N_{io}), \quad (6.4)$$

де M – витрати на матеріали, покупні напівфабрикати та комплектуючі (грн.);

$K_{\text{т.з.}}$ – коефіцієнт, що враховує транспортно–заготівельні витрати;

$Ц_i$ – ціна i -го найменування чи комплектуючого (грн.);

N_i – потреба в i -му матеріалі, напівфабрикаті, комплектуючих;

Π_{io} – ціна зворотних відходів i -го найменування матеріалу (грн.);

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів та комплектуючих [14].

Підрахунок витрат на матеріал наведений нижче:

$$M = 1,05 \cdot 1835 = 1926,75 \text{ грн.}$$

Витрати на матеріали для розробки наведені в табл. 6.4:

Таблиця 6.4 – Витрати на матеріали

Найменування, вид, тип, марка	Одиниця виміру	Ціна за одиницю вим., грн.	Кількість од. вим.	Сума витрат, грн.
Канцелярія	Набір	330	2	660
Картридж для принтера	шт.	700	1	700
Папір формату А4 для принтера	Листів	0,25	500	125
USB Flash накопичувач об'ємом 32Gb	шт.	350	1	350
Всього				1835

6.3.5 Витрати на ПК

Далі потрібно підрахувати витрати на придбання обладнання, його перевезення і налагодження. Після цього потрібно порахувати витрати на оплату машинного часу ЕОМ (амортизаційні витрати на обслуговування та вартість спожитої електроенергії) [15].

Рахувати дані витрати необхідно за наступною формулою (6.5):

$$A = \Phi_B \frac{H_a}{100}, \quad (6.5)$$

де Φ_B – балансова вартість обчислювальної техніки (грн.);

N_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки (%).

Підставивши у формулу (6.5) значення балансової вартості використовуваних Mi RedmiBook 14, маніпулятору типу миша та принтеру Canon PIXMA G1411 (разом 27500 грн.) отримаємо суму:

$$A = 27500 \cdot 0,25 = 6875 \text{ грн.}$$

Амортизаційні відрахування становлять 6875 грн.

Для знаходження вартості користування обладнання використаємо дану формулу (6.6):

$$C_3 = N_n \cdot \Phi_{ef} \cdot K_{зч} \cdot K_{зп} \cdot C_e, \quad (6.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ;

$K_{зм}$ – середній коефіцієнт завантаження за часом;

$K_{зп}$ – коефіцієнт завантаження за потужністю;

C_e – ціна одного кВт*год. Електроенергії, грн./(кВт*год.).

Загальна кількість часу роботи ЕОМ (ПК працівника) за рік становить 2000 годин. Завантаження за часом становить 0,8, а потужності – 0,6. Знайдемо вартість користування обладнанням:

$$C_{31} = 0,2 \cdot 2000 \cdot 0,8 \cdot 0,6 \cdot 2,11 = 405,1 \text{ грн.}$$

Заробітна плата обслуговуючого персоналу рахується за формулою (6.7):

$$ЗП_{\text{обсл}} = ФЗП_p (1 + K_{\text{від}}) \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.7)$$

де $ФЗП_p$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робіт (грн.);

$K_{\text{від}}$ – коефіцієнт відрахувань на соціальне страхування і до інших фондів (%);

$t_{\text{обсл}}$ – час протягом року, необхідний на технічне обслуговування ЕОМ, (ч/рік);

$\Phi_{\text{еф. обсл.}}$ – річний ефективний фонд часу обслуговуючого персоналу.

Заробітна плата обслуговуючого персоналу за місяць складає 4500 грн., а за рік вона становить 54000 грн. Робочий час обслуговуючого персонального комп'ютера робітника дорівнює 2000 годин на рік. На проведення сервісу одного ПК працівнику потрібно 3 години, а за рік це дорівнює 36 годин. Як наслідок отримуємо результат:

$$ЗП_{\text{обсл}} = 54000 \cdot (1 + 0.22) \cdot \frac{36}{2000} = 1185,84 \text{ грн.}$$

Витрата «Поточний ремонт обладнання» є 3 відсотки від вартості придбаного обладнання і дорівнює 825 грн.

«Інші витрати» являє собою п'ять відсотків від додавання всіх інших витрат на утримання та користування обладнання. Сума всіх указаних витрат дорівнює 9223 грн., 5% від цієї суми – 461,15 грн.

Кошти на оплату машинного часу ЕОМ для написання та полагодження програми потрібно шукати за даною формулою (6.8):

$$C_{\text{мс}} = P_{\text{екс}} t_{\text{мч}}, \quad (6.8)$$

де $C_{\text{мс}}$ – витрати на сплату машинного часу (грн.);

$P_{\text{екс}}$ – експлуатаційні витрати на один час машинного часу ЕОМ;

$t_{мч}$ – машинний час ЕОМ для написання та полагодження програмного засобу.

Знайдено експлуатаційні витрати на один час машинного часу ЕОМ:

$$P_{\text{екс}} = \frac{9753}{2000} = 4,87 \text{ грн./машино-час.}$$

ЕОМ використовується протягом 55 днів в одну зміну, що в сумі дорівнює 440 годин. Як наслідок кошти на оплату машинного часу дорівнюють сумі:

$$C_{\text{мс}} = 4,87 \cdot 440 = 2142,8 \text{ грн.}$$

Кошторис витрат, такі як амортизація обладнання, експлуатація ПК, заробітна плата основна та додаткова, поточний ремонт обладнання та інші витрати описані у таблиці 6.5.

Таблиця 6.5 – Кошторис витрат

Найменування статей витрат	Сума, грн.
Амортизація обладнання	6 875
Експлуатація ПК (окрім витрат на поточний ремонт)	405,1
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	1185,84
Поточний ремонт обладнання	825
Інші витрати	461,15
Разом	9753

6.3.6 Витрати на сервер

У якості сервера використовується ПК Mi RedmiBook 14. Підставивши у формулу (6.9) значення балансової вартості використовуваних та маніпулятору типу миша (разом 20000 грн.) отримаємо :

$$A = 20000 \cdot 0,25 = 5000 \text{ грн.} \quad (6.9)$$

Сума амортизаційних відрахувань дорівнює 5000 грн.

За поданою нижче формулою (6.10) час роботи ЕОМ (сервера) на рік становить 2000 годин. Середні коефіцієнти завантаження за часом та за потужністю становлять 0,8 та 0,6.

$$C_{\Sigma 2} = 0,2 \cdot 2000 \cdot 0,8 \cdot 0,6 \cdot 2,11 = 405,1 \text{ грн.} \quad (6.10)$$

Заробітна плата обслуговуючого персоналу розраховується за формулою (6.11).

Обслуговуючий персонал за місяць отримує 4500 грн заробітної плати, а за рік плата становить 54000 грн. Робочий час обслуговуючого ПК працівника за рік становить 2000 годин. На обслуговування одного персонального комп'ютера потрібно 3 години на місяць, а за рік це становить 36 годин. З чого виходить, що:

$$ЗП_{\text{обсл}} = 54000 \cdot (1 + 0.22) \cdot \frac{36}{2000} = 1185 \text{ грн.} \quad (6.11)$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і складає 600.

Стаття «Інші витрати» складає п'ять відсотків від суми всіх попередніх статей витрат на утримання та експлуатацію устаткування. Сума всіх попередніх статей дорівнює 7190 грн., 5% від суми становлять 360 грн.

Кошторис витрат на утримання та експлуатацію серверу на рік описано у таблиці 6.6.

Таблиця 6.6 – Кошторис витрат

Найменування статей витрат	Сума, грн.
Амортизація обладнання	5000
Експлуатація (окрім витрат на поточний ремонт)	405,1
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	1185
Поточний ремонт обладнання	600
Інші витрати	360
Разом	7550

Для розробки ЕС також експлуатується сервер протягом 55 діб. Таким чином витрати на сплату машинного часу становлять число, знайдене за формулою (6.12):

$$C_{mc} = \frac{55}{365} \cdot 7550 = 1137,6 \text{ грн.} \quad (6.12)$$

6.3.7 Накладні витрати

Накладні витрати – витрати на загальне управління, таке як заробітна плата працівникам, додаткові канцелярські витрати, утримання будівель та їх експлуатацію. Накладні витрати додають до коштів для розробки додатку у відсотках до основної заробітної плати програмістів. В даному випадку накладні витрати становлять сорок відсотків до основної заробітної плати розробників, що становить 13600 грн.

В таблиці 6.7 наведені витрати на розробку додатку.

Таблиця 6.7 – Витрати на розробку програми

Стаття витрат	Сума (грн.)	Питома вага до підсумку (%)
Основна заробітна плата	34000	51,90
Додаткова заробітна плата	3400	5,19
ЄСВ	7260	12,56
Вартість матеріалів	1926,75	3,28
Витрати на ПК	2142,8	3,96
Витрати на сервер	1137,6	2,35
Накладні витрати	13600	20,76
Разом	63467	100,00

6.3.8 Розрахунок експлуатаційних витрат

Кошти, витрачені на експлуатацію програми протягом року, визначають за формулою (6.13):

$$И = P_{\text{экс.кТС}} + T_{\text{кТС}} \frac{S_{\text{рп}}}{n \cdot T_c}, \quad (6.13)$$

де $P_{\text{экс.кТС}}$ – експлуатаційні витрати, що припадають на 1 ч машинного часу КТС споживача програмного продукту, грн / машино–год;

$T_{\text{кТС}}$ – сумарний машинний час КТС протягом року, потрібний даному споживачеві програми для задач, що вирішуються за її допомогою, машино–год;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

n – кількість споживачів даної програми;

T_c – термін служби даної програми до її морального зносу, років.

Кошти, витрачені на експлуатацію програми протягом року, визначили:

$$И = 7550 + \frac{63467}{1 \cdot 10} = 13896,7 \text{ грн.}$$

6.3.9 Розрахунок витрат на адміністратора

Адміністратор за місяць заробляє 5500 грн, а за рік ця сума становить, див. формулу (6.14):

$$ЗП_{\text{адм}} = 5500 \cdot 12 \cdot 1,22 = 80520 \text{ грн.} \quad (6.14)$$

6.4 Розрахунок капітальних вкладень

Обсяг капітальних вкладень споживача програми K , пов'язаних з її розробкою / купівлею / і впровадженням, визначають за формулою (6.15):

$$K = K_{\text{пп}} + K_{\text{пф}}, \quad (6.15)$$

де $K_{\text{пп}}$ – виробничі витрати, грн.;

$K_{\text{пф}}$ – капітальні вкладення в виробничі фонди, необхідні для впровадження програми, грн.

Виробничі витрати охоплюють витрати на програмне налагодження, його інформаційне та математичне забезпечення, і проєктування системи. Знаходяться ці витрати за допомогою формули (6.16) [16]:

$$K_{пп} = \frac{S_{рп}}{n} + K_{пр.пп}, \quad (6.16)$$

де $S_{рп}$ – сумарні витрати на розробку або придбання програмного продукту, грн;

n – кількість споживачів даної програми;

$K_{пр.пп}$ – інші виробничі витрати, грн.

Капітальні вкладення в виробничі фонди включають витрати на придбання, установку, монтаж і налагодження комплексу технічних засобів (КТЗ) з урахуванням можливого використання КТЗ як для вирішення завдань для яких призначена розробляється програма, так і для завдань, що вирішуються за допомогою інших програм. Питомі капітальні вкладення в виробничі фонди, припадаючі на частку даного програмного продукту визначають за формулою (6.17):

$$K_{пф} = \frac{K_{ктс} \cdot T_{ктс}}{\Phi_{эф.ктс}}, \quad (6.17)$$

де $K_{ктс}$ – витрати на придбання, установку, монтаж і налагодження КТЗ, грн.;

$T_{ктс}$ – машинний час КТЗ, потрібне даному споживачеві програми для завдань, що вирішуються за допомогою даної програми, машино–год / рік;

$\Phi_{эф.ктс}$ – річний ефективний фонд часу роботи КТЗ, машино–год / рік.

Знайдено суму питомих капітальних вкладень:

$$K = \frac{63467}{1} + \frac{20000 \cdot 2000}{2000} = 83467 \text{ грн.}$$

6.5 Оцінка економічної ефективності

Створення даного застосунку направлено на отримання прибутку, залучення нових клієнтів і збільшення обсягу послуг.

За рахунок продажу програмного продукту та консультацію по ньому очікується продаж 2000 додатків за мінімальною вартістю 200 гривень за копію.

Кошти отримані від покупки додатку будуть вважатись прибутком, окрім заробітної плати адміністратору та інших витрат. Адміністратор буде проводити контроль над правильності функціонування, та буде вносить зміни, якщо це буде необхідно.

Прибуток розраховується за формулою (6.18).

$$\Pi = \text{Пост}_{\text{від продажу}} - 3\Pi_{\text{адм}} - \text{И}. \quad (6.18)$$

Таким чином сума прибутку дорівнює:

$$\Pi = 2000 \cdot 200 - 80520 - 13896,7 = 305588,3 \text{ грн.}$$

Річна економія розраховується за формулою (6.19).

$$\mathcal{E} = \Pi - E_{\text{н}}^{\text{а}} \cdot K, \quad (6.19)$$

де $\mathcal{E}$ – річна економія від впровадження системи, грн .;

Π – річний прибуток від впровадженої системи;

$E_{\text{н}}^{\text{а}}$ – нормативний коефіцієнт ефективності капітальних вкладень в засоби автоматизації;

K – обсяг капітальних витрат, пов'язаних зі створенням ПП.

Річна економія від впровадження системи дорівнює 293038,25 грн.

Коефіцієнт ефективності вкладених коштів в програму E і термін їх окупності $T_{\text{ок}}$ розраховують по формулам (6.20) та (6.21):

$$E = \frac{\Pi}{K}, \quad (6.20)$$

$$T_{\text{ок}} = \frac{K}{\Pi}. \quad (6.21)$$

Таким чином:

$$E = 293038,25 / 83467 = 3,5,$$

$$T_{\text{ок}} = 83467 / 293038,25 = 0,28 \text{ року.}$$

Отже у такий спосіб і таким чином економічна доцільність розробки даного програмного забезпечення, тому що окупність додатку відбувається за 0,28 року, що менше за допустиме значення часу на окупність продукту, що дорівнює 6,67 року.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

Використання програмного забезпечення, розробленого під час проєкту, передбачає використання комп'ютерів (персональних комп'ютерів, периферійних пристроїв тощо) та, як і будь-яка виробнича діяльність, пов'язана з впливом небезпечних та шкідливих виробничих факторів. Небезпечні та шкідливі фактори визначені в ДСанПіН "Гігієнічна класифікація праці за показниками шкідливості та безпеки факторів виробничого середовища, тяжкості та напруженості трудового процесу" №0472-14 [17].

Звертаючи увагу на потенційні небезпеки, які виникають на виробництві, де розроблюють програмне забезпечення, створено та відтворено низку заходів, які спрямовані на забезпечення безпечних умов праці.

Усі потенційні небезпеки, що виникають під час роботи в офісі чи комп'ютерному центрі, при роботі з комп'ютерами, можна розділити на такі групи:

- потенційні фізичні небезпеки;
- потенційні небезпеки психофізіологічного характеру;
- потенційні небезпеки санітарно-гігієнічного характеру;
- потенційна небезпека, пов'язана з порушенням правил пожежної безпеки;
- потенційні небезпеки, пов'язані з проявом наслідків надзвичайних ситуацій.

Потенційні фізичні небезпеки включають механічні травми, які можуть виникнути внаслідок неправильного планування виробничих приміщень (розташування електричних або комунікаційних проводів у підлозі у приміщенні великої прохідності, неправильні офісні меблі, погано

прикріплені полиці та шафи, інше), і навіть можуть бути можливий удар струмом під час роботи на комп'ютері або використання периферійного комп'ютерного обладнання (внаслідок порушення правил безпеки під час роботи або використання несправного обладнання), тощо. Відомо про те, що робота на комп'ютері забезпечує значні статичні навантаження на опорно-руховий апарат та організацію. Професійним лікуванням фахівців, які входять до системи при тривалому використанні комп'ютерів (зокрема, інженери-програмісти, оператори комп'ютерів), є синдром зап'ястного каналу або синдром зап'ястя тунелю, що може бути наслідком хронічної травми. Причиною такого захворювання є тривала робота з мишею: постійні стреси та напруга призводять до мікротравм, здавлення нервів навколишніми тканинами, що спричинює набряки.

Потенційні психофізіологічні небезпеки пов'язані з незадовільною організацією робочого місця (нераціональне розташування комп'ютерів та периферійних пристроїв, захаращеність робочої зони, неправильно підібране робоче місце з урахуванням параметрів людини та надмірне спілкування в робочий час, що вимагає концентрації уваги і розумову працю, тощо).

Потенційна небезпека для здоров'я включає порушення санітарних норм у приміщеннях, де безпосередньо здійснюється виробничий процес (наприклад, недостатня або підвищена освітленість робочої зони, недостатній повітрообмін, високий рівень шуму через роботу комп'ютера, недостатня або підвищена вологість в приміщенні).

Група небезпек, пов'язаних з порушенням правил пожежної безпеки, може включати потенційну небезпеку, пов'язану з недотриманням або неналежним дотриманням правил пожежної безпеки при плануванні виробничого об'єкта, виробничого процесу або при використанні небезпечного обладнання або оснащення (комп'ютерна техніка, периферійна техніка, тощо).

Врешті-решт, до потенційних небезпек, пов'язаних із наслідками надзвичайних ситуацій, належать такі, що пов'язані з неготовністю персоналу до надзвичайних ситуацій (таких як пожежа чи стихійне лихо).

7.2 Заходи із забезпечення безпеки

Для запобігання потенційним фізичним, психофізіологічним, санітарно-гігієнічним небезпекам, пожежній та надзвичайній небезпеці, а також для мінімізації впливу шкідливих виробничих факторів діє низка заходів безпеки. При плануванні виробничого процесу та розробці заходів безпеки застосовуються норми та вимоги, визначені в "Правилах охорони праці під час експлуатації електронно-обчислювальних машин", затверджених наказом директора управління державного нагляду в сільському господарстві та соціально-культурній сфері управління державного нагляду №65 від 26 березня 2010 року [18], ГОСТ 12.1.019-79 «Электробезопасность. Общие требования и номенклатура видов защиты» [19], «ПУЕ» («Правила устрою електроустановок»), ДСанПІН 3.3.2.007 – 98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [20], СНиП 2.04.05-91*У «Отопление, вентиляция и кондиционирование» [21], ГОСТ 12.1.005-88 (1991) «ССБТ. Общие санитарно – гигиенические требования к воздуху рабочей зоны» [22], а також норми зазначені у ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [23] та інші.

Передбачені заходи можна розділити на два розділи: організаційні та технічні.

Проти травмування персоналу проведено проектування ергономіки приміщення, розташування офісних меблів і виробничого обладнання в офісах. Розводка крученої пари виконана згідно стандарту EIA/TIA 568 (Американський стандарт проводки у комерційних приміщеннях), а також міжнародному стандарту ISO 11801 [24]. Офіс оснащений потрібними

меблями, що відповідають ДСТУ EN 527-2:2007 «Меблі для адміністративних приміщень. Робочі столи. Частина 2. Вимоги механічної безпеки» [25].

Для того, щоб захистити працівників від уражень електричним струмом під час проведення службових обов'язків проведено технічні заходи відповідно ГОСТ 12.1.019-79 [26] і «ПУЕ» [27], згідно з яким захисне заземлення наявного електрообладнання не перевищує 4 Ом.

Проведена підготовка з правил електробезпеки, екзамнування знать працівників на кваліфікаційну групу з електробезпеки, згідно НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці» [28], для застереження всіляких травм, проведення інструктажів з правил електробезпеки.

Для запобігання синдрому зап'ястного каналу оператори OEM, що використовують програмний додаток, розроблений під час проєкту, пройшли інструктаж з персоналом та внесли правила до внутрішньої трудової дисципліни, а саме перерви на декілька хвилин кожні 45 хвилин безперервної роботи на комп'ютері.

Для запобігання впливу факторів, що обумовлюють психофізіологічні небезпеки було реалізовано ряд організаційних заходів, а саме:

- оптимізація та обмеження системи зв'язку (вертикальної та горизонтальної) відповідно до потреб існуючих бізнес-процесів організації;
- запровадження системи управління часом;
- проведення корпоративних розважальних заходів для працівників організації;
- запровадження системи контролю за дотриманням вимог внутрішнього трудового розпорядку.

7.3 Заходи із забезпечення виробничої санітарії та гігієни праці

До санітарно-гігієнічних умов відносять:

- мікрокліматичні умови та їх забезпечення (відповідно до вимог ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» та ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування»);
- освітленість робочої зони та її забезпечення (відповідно до вимог ДБН В.2.5-28:2018 «Природне та штучне освітлення»);
- рівень шуму в робочому просторі (відповідно до вимог ДСН 3.3.6.037- 99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку»);
- виробничі вібрації (відповідно до ДСН 3.3.6.039-99 «Державні санітарні норми виробничої загальної локальної вібрації» та ДСТУ ГОСТ 12.1.012:2008 «ССБП. Вібраційна безпека. Загальні вимоги»);
- електромагнітні поля та випромінювання (відповідно до ДСанНіП 3.3.6.096-2002 «Державними санітарними нормами і правилами і правилами при роботі з джерелами електромагнітних полів», ДСН 239-96 «Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань» і ГОСТ 12.1.045-84 «ССБТ. Электростатические поля. Допустимые уровни на рабочих местах и требования к проведению контроля»);
- ергономіка робочого місця (відповідно до ДСанПіН 33.2.007-98 "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин").

При організації роботи операторів ЕОМ було враховано вимоги ДСанПіН 3.3.2.007 – 98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Інструкції щодо режиму роботи працівників організації були розроблені та узгоджені з урахуванням специфіки роботи, регулюючи час

початку та закінчення робочого дня, тривалість обідньої перерви, наявність та тривалість додаткових перерв, тощо.

Згідно з ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» робота з розробки програмного забезпечення належить до легкої категорії фізичної роботи (категорія Ia). Тож норми мікроклімату повинні відповідати параметрам визначених в ГОСТ 12.1.005-88 (1991) «ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны» (табл. 7.1) [29].

Для того, щоб гарантувати прийнятний рівень повітряного середовища у офісі, призначена система водяного опалення приміщення, що гарантує потрібну температуру повітря в холодні пори року відповідно СНиП 2.04.05-91*У «Отопление, вентиляция и кондиционирование».

В жаркі пори року навпаки введено систему охолодження повітря, для збереження прийнятного температурного режиму.

Таблица 7.1 – Допустима температура приміщення

Період року	Температура, °C	Відносна вологість, %	Швидкість руху, м/с
Холодний	22-24	70	0,2
Теплий	25-27	50	0,1-0,4

Потрібний рівень повітрообміну у офісі визначається за формулою (7.1):

$$L = \frac{Q}{C \cdot \gamma \cdot (t - t_{\text{пр}})}, \quad (7.1)$$

де Q – сумарна кількість теплоти, що утворюється в приміщенні, кДж/год;

C – питома теплоємність повітря, що дорівнює $C = 1 \text{ кДж}/(\text{кг} \cdot ^\circ\text{C})$;

γ - густина зовнішнього повітря, що дорівнює $\gamma = \frac{353}{(273+t_{\text{пр}})}$, кг/м³;

t – температура повітря, що видаляється, °С;

$t_{\text{пр}}$ – температура приточного повітря.

Загальна кількість тепла, що виробляється в приміщенні за годину, очікується на рівні 5300 кДж, а температура повітря, що відводиться, становить 25 °С. Встановлена система охолодження гарантує температуру припливного повітря на рівні 18 °С.

Таким чином, потрібний повітрообмін у приміщенні буде, див. формулу (7.2):

$$L = \frac{5300}{1 \cdot \frac{353}{(273 + 18)} \cdot (25 - 18)} = 623,5 \text{ м}^3/\text{год} , \quad (7.2)$$

Таким чином для знешкодження надмірного тепла у офісі в теплі пору року потрібно впровадити систему охолодження повітря, що гарантуватиме повітрообмін на рівні 623,5 м³ за годину. Щоб дотриматись цього запиту було впроваджено кондиціонер HYUNDAI ARN07HSSUA/ARU07HSSUA Inverter з потужністю охолодження 2,2 кВт та теплопродуктивністю 2,3 кВт.

Освітлення в офісі мусить бути 250-300 лк згідно ДБН В.2.5-28:2018 "Природне і штучне освітлення". Якщо немає змоги це зробити, можливе використання локального освітлення. Проте його потрібно встановити так, щоб не було відсвіту на екрані, а освітлювання самого екрану було більше за 280 лк.

Очікується, що в офісі застосовуватися обладнання, яке може перевищувати прийнятний рівень шуму (згідно ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку»). Зважуючи те, що принтер може стати причиною перевищення рівня шуму, призначено змінити його місцезнаходження якнайдалі від робочих місць.

Електромагнітне випромінювання згідно ДСН 239-96 "Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань" має подальші дані (табл. 7.2):

Таблиця 7.2 - Електромагнітне випромінювання моніторів робочих комп'ютерів

Вид електромагнітного випромінювання	Діапазон випромінювання	
	5...3000 Гц	3000...500000 Гц
Напруженість змінного електричного поля, В/м	30	3,5
Щільність магнітного потоку, нТл	150	30

Якщо електромагнітне випромінювання не буде відповідати бажаному рівню, то будуть впроваджені такі заходи: фільтри проти відсвіту на моніторах та зовнішній фільтр, що заземляється.

Усі робочі дисплеї в офісі задовольняють вимоги безпеки і енергозбереження ТСО'99 і, таким чином не перевищують рівень електромагнітного випромінювання (згідно ДСанПІН 3.3.2.007 – 98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»).

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Для правильної організації протипожежних заходів і оснащення офісу установками пожежогасіння, необхідно спочатку визначити категорію приміщення по вибухопожежонебезпечності. Віднесення до однієї з категорій здійснюється шляхом перевірки приміщення і розрахунку за допомогою таблиць і формул, зазначених в нормативних склепіннях.

Зазвичай офіси відносяться до категорії В і вважаються пожежонебезпечними. Така класифікація пов'язана з наявністю в офісних приміщеннях горючих і важкогорючих матеріалів і речовин, які при контакті з повітрям горять без утворення вибухонебезпечних сумішей.

Аналіз отриманих даних дозволяє виробити ряд вимог, що висуваються до офісних приміщень:

- наявність інструкції про заходи пожежної безпеки;
- обладнання системою пожежної сигналізації;
- оснащення офісу системами і засобами пожежогасіння згідно з класом приміщення;
- організація евакуаційних шляхів.

Підготовка та реалізація методів пожежної безпеки здійснено у відповідності до Закону України Про пожежну безпеку, Правил пожежної безпеки в Україні та вимогам відповідних нормативних актів.

Усі працівники були проінструктовані щодо пожежної безпеки під час прийому на роботу. Факт проведення інструктажів зафіксовано у спеціальному журналі, який був прошнурований, пронумерований та скріплений печаткою підприємства та підписом керівника.

Види протипожежних інструктажів, а також порядок організації та проведення протипожежних інструктажів, навчання і перевірки знань з пожежно-технічного мінімуму встановлено постановою КМУ від 26.06.2013 №444 «Про затвердження Порядку здійснення навчання населення діям у надзвичайних ситуаціях».

Періодичність проведення інструктажів встановлюється керівником підприємства, однак повинна становити не рідше одного разу в рік. Інструктажі проводяться працівником, який пройшов навчання з пожежної безпеки в навчальному центрі.

Побудований графік перевірки та контролю вогнегасників і назначено працівника, який буде відповідати за це. Особа, яка стала відповідальною за пожежну безпеку, перевірила вогнегасники перед їх розміщенням на об'єкті,

також переконалась в цілісності пломб на запірних пристроях, перевірила наявність або відсутність зовнішніх пошкоджень на самих вогнегасниках, положення стрілок індикаторів тиску закачних вогнегасників та наявність у маркуванні та в експлуатаційній документації відомостей про виробника, дату виготовлення (продажу) і технічного обслуговування. Також зауважила те, що розташування вогнегасників не повинно перевищувати відстань 25 метрів до самого дальнього місця в офісі.

Також встановили пожежну сигналізацію згідно до основних вимог ДБН В.2.5-56:2014 «Системи протипожежного захисту» і захисні пристрої, які вимикають джерело живлення, де трапилось замикання.

Всі офісні пристрої та гарнітура розміщені на відстані метру одне від одного, щоб у екстрених ситуаціях був достатній прохід для евакуації.

Крім того створено план евакуації, який розташований на видимому місці, проведено інструктаж з працівниками на тему дотримання всіх правил пожежної безпеки в офісі й умілість використання засобів пожежогасіння.

7.4.2 Заходи з цивільного захисту. Складові та режими діяльності єдиної державної системи цивільного захисту

Для покращення та модернізації контролю і управління системою цивільного захисту затверджено велику кількість важливих нормативно-правових актів України. Базовим документом, що забезпечив оптимізацію законодавства у сфері цивільного захисту і заклав нормативну базу для створення єдиної державної системи цивільного захисту, став прийнятий Верховною Радою України Кодекс цивільного захисту України, який затверджено Указом Президента України від 02 жовтня 2012 року № 5403-VI та введено в дію з 01 липня 2013 року.

Кодексом цивільного захисту України:

- покращено правові, економічні та організаційні засади реалізації державної політики у сфері цивільного захисту, зокрема щодо забезпечення

безпеки та захисту населення і територій від надзвичайних ситуацій та їх наслідків;

- створено єдину систему цивільного захисту;
- розподілено повноваження і функції у сфері цивільного захисту між центральними і місцевими органами виконавчої влади, органами місцевого самоврядування та іншими суб'єктами;
- підвищено ефективність заходів, спрямованих на запобігання виникненню надзвичайних ситуацій;
- усунуто протиріччя та дублювання, що містилися у нормативноправових актах у сфері цивільного захисту.

У зв'язку з його прийняттям низка законодавчих актів України, що регулювали відносини у відповідній сфері, визнано такими, що втратили чинність, серед них закони України: «Про Цивільну оборону України», «Про пожежну безпеку», «Про війська Цивільної оборони України», «Про аварійно-рятувальні служби», «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», «Про правові засади цивільного захисту».

Цивільний захист – це функція держави, спрямована на захист населення, територій, навколишнього природного середовища та майна від надзвичайних ситуацій шляхом запобігання таким ситуаціям, ліквідації їх наслідків і надання допомоги постраждалим у мирний час та в особливий період [30].

Цивільний захист забезпечується з урахуванням особливостей, визначених Законом України «Про основи національної безпеки України», суб'єктами, уповноваженими захищати населення, території, навколишнє природне середовище і майно, Кодексом цивільного захисту – у мирний час, а також в особливий період – у межах реалізації заходів держави щодо оборони України.

Основні принципи здійснення цивільного захисту:

- гарантування та забезпечення державою конституційних прав громадян на захист життя, здоров'я та власності;
- комплексного підходу до вирішення завдань цивільного захисту;
- пріоритетності завдань, спрямованих на рятування життя та збереження здоров'я громадян;
- максимально можливого, економічно обґрунтованого зменшення ризику виникнення надзвичайних ситуацій;
- централізації управління, єдиноначальності, підпорядкованості, статутної дисципліни Оперативно-рятувальної служби цивільного захисту, аварійно-рятувальних служб;
- гласності, прозорості, вільного отримання та поширення публічної інформації про стан цивільного захисту, крім обмежень, встановлених законом;
- добровільності – у разі залучення громадян до здійснення заходів цивільного захисту, пов'язаних з ризиком для їхнього життя і здоров'я;
- відповідальності посадових осіб органів державної влади та органів місцевого самоврядування за дотримання вимог законодавства з питань цивільного захисту;
- виправданого ризику та відповідальності керівників сил цивільного захисту за забезпечення безпеки під час проведення аварійно-рятувальних та інших невідкладних робіт [31].

ВИСНОВКИ

В ході розробки створено додаток для моніторингу мережного трафіку за допомогою вейвлетаналізу. В проєкті були використані наступні технології: мови програмування Python, JavaScript, Node.js, Docker, Hypertext Markup Language, Cascading Style Sheets. Дипломний проєкт розроблювався на основі персонального комп'ютера з процесором Intel Core i3 під управлінням операційної системи Ubuntu 20.04. Кожна частина проєкту запускається в окремому докер контейнері. Таким чином додаток добре зкомпонований для розробки. Для реалізації додатку було використано IDE: PHPStorm, PyCharm.

Було розроблено програмне забезпечення, що дозволяє здійснювати моніторинг мережі, проводити політику доступу пристроїв в мережу Інтернет, відображення підключених пристроїв до локальної мережі та відображення їх IP та MAC адресу, функція логування підключення/відключення пристроїв до локальної мережі. За допомогою простого та зручного дизайну вебдодатком легко користуватися. Він не потребує великих знань, ним може користуватись навіть початковий користувач ПК.

Для коректної роботи вебдодатку було оброблено багато інформації про локальні мережі, DoS (denial of service) та DDoS (distributed denial of service) атаки, документації та специфікації роботи мов програмування, які були використані в дипломному вебдодатку. Для підтвердження коректної роботи вебдодатку було проведено тестування, яке показало, що вебдодаток може правильно працювати в локальній мережі та виконувати свої поставленні функції.

Мій дипломний проєкт охоплює велику сферу галузей, де він може допомогти моніторити локальну мережу. Вебдодаток може використовуватися, як для домашнього використання, так і для компаній, які потребують контроль та моніторинг трафіку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шелухін, О. І. Виявлення мережевих аномальних викидів трафіку методом розладнання Бродського-Дарховського / О. І. Шелухін, А. С. Філінова // Т-Comm – Телекомунікації та Транспорт. – 2013. – №10. – С. 116-118.
2. Шелухін, О. І. Виявлення мережевих аномальних викидів трафіку методами на основі суми та максимуму квадратів вейвлеткоєфіцієнтів / О. І. Шелухін, А. С. Філінова // Т-Comm – Телекомунікації та Транспорт. – 2013. – №8. – С. 50-56.
3. Шелухін, О. І. Виявлення DOS і DDOS-атак методом дискретного вейвлет-аналізу / О. І. Шелухін, Ю. А. Іванов, В. Ю. Рігов // Т-Comm – Телекомунікації та Транспорт. – 2011. – №1. – С. 44-46.
4. Приклади застосування вейвлетаналізу / А. Н. Павлов // Огляди актуальних проблем нелінійної динаміки. – 2012. – №5. – С. 99.
5. Аналіз та класифікація методів виявлення вторгнень в інформаційну систему / В. В. Берковський, О. С. Безсонов // Кібернетична безпека. – 2017. - №2. - С. 57-62.
6. Преобразование Фурье [Электрон. ресурс]. – Режим доступа: <https://nplus1.ru/material/2017/03/24/abel-prize/>.
7. Огляд алгоритмів виявлення мережевих атак / У. В. Тремасова, С. В. Голубев // Економічні науки. – 2017. - №1. - С. 165-167.
8. Модифікація методики вейвлет-аналізу для виявлення аномалій у трафіку комп'ютерної мережі / Віталій Литвинов, Ігор Скітер, Олена Трунова, Едуард Сідін // Технічні науки та технології. – 2017. – №2(8). – С. 99-109.
9. What is python? Plus, 13 reasons why you should learn it [Electronic resource]. – Access mode: <https://skillcrush.com/blog/what-is-python/>.
10. An introduction to JavaScript [Electronic resource]. – Access mode: <https://javascript.info/intro>.

11. What is CSS, and why is it important? [Electronic resource]. – Access mode: <https://www.bigcommerce.com/ecommerce-answers/what-css-and-why-it-important/>.
12. What is HTML? [Electronic resource]. – Access mode: <https://www.hostinger.com/tutorials/what-is-html>.
13. What is docker? [Electronic resource]. – Access mode: <https://opensource.com/resources/what-docker>.
14. Економічна оцінка виробництва виробу на підприємстві / О. М. Гаврись, А. Ю. Подхалюзін. - Харків: НТУ «ХПІ». – 2017. – С. 25.
15. Економіка виробничого підприємства / За ред. Й. М. Петровича. – К.: Т-во „Знання”, КОО. – 2002. – С. 405.
16. Примак Т. О. Економіка підприємства. – К.: ВІКАР. – 2001. – С. 178.
17. Бедрій Я. І. / Охорона праці: Навчальний посібник. – Львів: Магнолія. – 2007. – С. 238.
18. Грішнова О. А. Економіка праці та соціально-трудова відносини: підручник / О. А. Грішнова. 4-те вид., оновлене. – К.: Знання. – 2009. – С. 390.
19. Жидецький В. Ц., Джигирей В. С, Мельников О. В. Основи охорони праці. – Вид 2-е, стереотипне. – Львів: Афіша. – 2002. – С. 320.
20. Керб Л. П. Основи охорони праці: Навч. посібник. – К.: КНЕУ. – 2003. – С. 215.
21. Щедрій Я. І. Охорона праці. – К.: ЦУЛ, – 2002. – С. 322.
22. Ткачук К. Н., Зацарний В. В., Сабарно Р. В. та інші. Охорона праці та промислова безпека: Посібник. – Київ: Лібра, – 2010. – С. 559.
23. Основи охорони праці: / В. В. Березуцький, Т. С. Бондаренко, Г. Г. Валенко та ін.; за ред. проф. В. В. Березуцького. – Х.:Факт, – 2005. – С. 480.
24. Русаловський А. В. Правові та організаційні питання охорони праці: Навч. посіб. – 4- те вид., допов. і перероб. – К.: Університет «Україна»,

– 2009. – С. 295.

25. НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електроннообчислювальних машин». Наказ Держгірпромнагляду від 26.03.2010р. № 65.

26. Охорона праці (практикум): Навч. посіб. / За заг. ред.. к.т.н., доц.. І. П. Пістуна. – Львів: «Тріада плюс», – 2011. – 436с.

27. Серіков Я. О. Основи охорони праці: Навч. посіб. – Харків, ХНАМГ. – 2007. – 227с.

28. Ткачук К. Н., Филипчук В. Л., Каштанов С. Ф., Зацарний В. В., Полукаров О. І. та ін. Виробнича санітарія: Навчальний посібник. – Рівне. – 2012. – 443с.

29. Офіційний сайт Держгірпромнагляду [Електрон. ресурс]. – Режим доступу: <http://www.dnopr.kiev.ua>.

30. Офіційний сайт Міністерства надзвичайних ситуацій України [Електрон. ресурс]. – Режим доступу: <http://www.mns.gov.ua>.

31. Энциклопедия по охране и безопасности труда МОТ. [Электрон. ресурс]. – Режим доступа: <http://base.safework.ru/iloenc>.

ДОДАТОК А
Текст програми

A. 1 Скрипт `check_connection.py`

```
#!/usr/bin/env python2.7

import time
import datetime
import os
import sys
import select
import subprocess
import json
from pprint import pprint

# foreign libraries to install for python2.7
import argparse
from scapy.all import *
import pytsx
import requests
# from urllib.parse import urlencode
# from urllib.request import Request, urlopen
# from twilio.rest import TwilioRestClient

# terminal colors
OKGREEN = '\033[92m'
FAIL = '\033[91m'
ENDC = '\033[0m'
ERROR = FAIL + "ERROR: " + ENDC
OKBLUE = '\033[94m'

connected = {} # { "MAC": "time connected" }
```

```

mac_defs = {} # { "MAC": "device name"}
black_list = [] # MAC addresses to exclude from monitoring

loop_seconds = 30 # seconds to wait in between each monitor loop
disconnect_time = 1800 # time to wait before client is rendered
disconnected

checking = False # flag that indicates whether or not to notice new
connections

__broadcast = True # toggle for broadcasting to a web service
__writing = False # toggle for writing connections to a file
__speak = False # flag for speaking when a new device connects
__log_unknown = True # flag to indicate whether to log unknown devices
__printing = False # flag to indicate if script prints output
connected_file = "resources/connected_file.json" # file to log connections
log_file = "/src/resources/unknown_connections.log" # log file for unknown
devices

mac_file = "resources/mac_addresses.json" # file for mac addresses
web_service_url = "http://127.0.0.1:3000/api/connections" # web service
url

def speak(say_this):
    """
    Will speak the given message
    """
    global __speak
    if __speak:
        os.system('aplay resources/blip.wav -R 1')
        speech_engine = pyttsx.init()
        speech_engine.say(say_this)

```

```
speech_engine.runAndWait()
```

```
def get_lan_ip():
    """
    Gets the current LAN's IP by connecting to google and extracting the IP
from
    the socket.
    """
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.connect(("google.com", 80))
    ip = s.getsockname()
    s.close()
    return ip[0]
```

```
def get_gateway_ip(gateway_ip=None):
    """
    Gets the gateway ip. Returns [gateway_ip, ip_range]
    """
    myip = get_lan_ip()
    ip_list = myip.split('.')
    del ip_list[-1]
    ip_list.append('*')
    ip_range = '.'.join(ip_list)
    del ip_list[-1]
    ip_list.append('1') # assumed default gateway is "XXX.XXX.XXX.1"
    if gateway_ip is not None:
        return [gateway_ip, ip_range]
    else:
```

```
return ['.'.join(ip_list), ip_range]
```

```
def get_ip_macs(ips):
```

```
    """
```

```
    Returns a list of tuples containing the (ip, mac address)
    of all of the computers on the network
```

```
    """
```

```
    answers, uans = arping('192.168.0.0/24', verbose=0)
```

```
    res = []
```

```
    for answer in answers:
```

```
        mac = answer[1].hwsrc
```

```
        ip = answer[1].psrc
```

```
        res.append((ip, mac))
```

```
        # res.append(specific_target)
```

```
    return res
```

```
def load_macs(filename):
```

```
    """
```

```
    Loads devices and their corresponding MAC addresses from a given json
```

file

```
    """
```

```
    try:
```

```
        global mac_defs
```

```
        with open(filename) as data_file:
```

```
            data = json.load(data_file)
```

```
        for each in data:
```

```
            mac_defs[each['MAC'].lower()] = each['device']
```

```
    except IOError:
```

```

    print filename+" not found."
    sys.exit(1)

def check_connected(gateway_ip, ip_range):
    global mac_defs, connected, __checking, black_list, __printing
    __new_connection = False
    devices = get_ip_macs(ip_range)
    #subprocess.call('clear', shell=True) # clear the shell
    i = 0
    for device in devices:
        if device[1] not in black_list:
            if device[1] in mac_defs.keys():
                if __printing:
                    print '%s)\t%s\t%s\t%s%s' % (i, device[0], device[1],
mac_defs[device[1]], ENDC)
                if device[1] not in connected.keys():
                    __new_connection = True
                    # notify
                    if __checking:
                        if __printing:
                            print OKGREEN + mac_defs[device[1]] + " just
connected!" + ENDC

    speak(mac_defs[device[1]] + "\n ")
        # update time stamp
        connected[device[1]] = datetime.now()
    else:
        # unkown device
        if __printing:

```

```

        print '%s)\t%s\t%s' % (i, device[0], device[1])
        speak("foreign device detected")
        log_foreign_access(datetime.now().strftime(
            "%Y-%m-%d %H:%M:%S") + " " + device[1])
        mac_defs[device[1]] = "UNKNOWN_DEVICE"
        connected[device[1]] = datetime.now()

    i += 1
    if __broadcast:
        broadcast_connection(device[1], mac_defs[
            device[1]], connected[device[1]], device[0])
    __checking = True
    return __new_connection

def monitor_loop(gateway_ip, ip_range):
    """
    main monitor loop of the program. checks for new connections and
    disconnections.
    """
    global loop_seconds
    while True:
        try:
            new_connections = check_connected(gateway_ip, ip_range)
            new_disconnections = check_for_disconnections()
            if new_connections or new_disconnections:
                write_connections()
                time.sleep(loop_seconds)
        except KeyboardInterrupt:
            print "shutting down..."

```

```
sys.exit()
```

```
def check_for_disconnections():
    """
    checks for disconnections. returns a boolean indicating whether a new
    disconnection has occurred.s
    """
    global connected, mac_defs, disconnect_time, __broadcast, __printing
    __new_disconnection = False
    to_be_removed = []
    for each in connected:
        diff = datetime.now() - connected[each]
        diff = diff.total_seconds()
        if __printing:
            print(OKBLUE + "%20s%20s" + ENDC) % (mac_defs[each],
str(diff))
        if diff >= disconnect_time:
            to_be_removed.append(each)
            __new_disconnection = True
    for each in to_be_removed:
        t = connected.pop(each)
        if __broadcast:
            broadcast_disconnection(each)
        if __printing:
            print FAIL + mac_defs[each] + " disconnected " + ENDC

    return __new_disconnection
```

```

def load_blacklist(filename):
    """
    loads MAC addresses from a file will be added to the black_list. entries
in
    the text file should be 1 MAC address per line
    """
    try:
        global black_list
        count = 0
        with open(filename) as data_file:
            for each in data_file:
                black_list.append(each.strip())
                count = count + 1
    except IOError:
        print filename+" does not exist."
        sys.exit(1)

def block_ip_addr(ip_addr):
    file = open('./resources/blacklist.txt', "w")
    file.write(ip_addr + "\n")
    file.close()

def broadcast_connection(mac, alias, timestamp, ip):
    """
    sends a http POST request with the MAC address of a newly connected
    device to a specified url
    """
    try:

```

```

        global web_service_url
        post_fields = {"mac": mac, "alias": alias, "timestamp": timestamp,
"ip":ip}

        requests.post(web_service_url, data=post_fields)

        # request = Request(web_service_url,
urlencode(post_fields).encode())

        # res = json.loads(urlopen(request).read().decode())
    except requests.exceptions.ConnectionError:
        pass

```

```

def broadcast_disconnection(mac):
    """
    sends a http DELETE request with the MAC address of a newly
disconnected
    device to a specified url
    """
    try:
        global web_service_url
        post_fields = {"mac": mac}
        requests.delete(web_service_url, data=post_fields)
        # request = Request(web_service_url+"/"+str(order_id),
method='DELETE')

        # res = json.loads(urlopen(request).read().decode())
    except requests.exceptions.ConnectionError:
        pass

# def send_text(msg):
#     client = TwilioRestClient()
#     client.messages.create(from_='+16313537388',

```

```

#             to='+15162590083',
#             body=msg)

def write_connections():
    """
    writes out connections to a JSON file
    """
    global connected, mac_defs, __writing
    if __writing:
        data = {}
        data['connections'] = []
        for mac in connected:
            json_obj = {}
            json_obj['MAC'] = mac
            json_obj['alias'] = mac_defs[mac]
            json_obj['timestamp'] = str(connected[mac])
            # data = data + mac + ", "+mac_defs[mac]+", "
+str(connected[mac])+ "\n"
            data['connections'].append(json_obj)
        write_to_file(connected_file, json.dumps(data, indent=4), "w")

def log_foreign_access(entry):
    """
    logs access from foreign devices to a log file with timestamp and mac
address
    """
    global log_file
    # create file if not exists
    write_to_file(log_file, entry, "a")

```

```

def write_to_file(file, data, write_option):
    """
    writes data to a file.
    """
    try:
        if not os.path.exists(file):
            open(file, 'w').close()
        with open(file, write_option) as myfile:
            os.chmod(file, 0777)
            myfile.write(data + '\n')
    except IOError:
        print file+" not found."

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-g", help="specify a gateway. default gateway will be 192.168.1.1")
# gateway
    # file with mac addresses
    parser.add_argument(
        "-f", help='specify a JSON file with device names and their
corresponding MAC addresses. Format should be [{"device": "some device",
"MAC": "some mac"}]. default will be mac_addresses.json')
    # text file with mac address exclusions
    parser.add_argument(

```

"-e", help='specify a text file containing MAC addresses to exclude from monitoring. text file must have one MAC address per line. default file will be blacklist.txt')

```
# parser.add_argument("-t", help="time to wait before a given client is
# considered disconnected. (seconds)") #time to wait before client is
# considered discon.
```

```
args = parser.parse_args()
```

```
# check for root access
```

```
if os.getuid() is not 0:
```

```
    print "This script requires root access."
```

```
    sys.exit(1)
```

```
ip_range = None
```

```
gateway_ip = None
```

```
if args.g is not None:
```

```
    gateway_ip = args.g
```

```
    ip_range = ".".join(gateway_ip.split('.')[0:-1]) + '.' + '*'
```

```
else:
```

```
    gateway_ip, ip_range = get_gateway_ip()
```

```
if args.f is not None:
```

```
    load_macs(args.f)
```

```
else:
```

```
    load_macs(mac_file)
```

```
if args.e is not None:
```

```
    load_blacklist(args.e)
```

```
# if args.t is not None:
```

```
#     global disconnect_time
```

```
#     disconnect_time = args.t
```

```
print("monitoring connections to network...")
monitor_loop(gateway_ip, ip_range)
```

A. 2 Server.js

```
// server.js

// dependencies
let bodyParser = require('body-parser');
let express = require('express');
let fs = require('fs');
let logger = require("morgan");
let redis = require("redis");
let app = express();
let server = require('http').createServer(app);

// websocket
let WebSocket = require('./my_modules/websocket');
let io = require('socket.io')(server);
io = new WebSocket(io);

// http authentication
let auth = require("basic-auth");
let admins = { // define admins here
  'admin': { password: 'password' }
};

// view engine
```

```

app.set('view engine', 'pug');

// body parser
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// public folder
app.use(express.static(__dirname + '/public'));

// morgan logger
app.use(logger('dev', {
  stream: fs.createWriteStream('./access.log', {
    flags: 'a'
  })
}));

// redis (for docker: ('6379', 'redis') )
let redisClient = redis.createClient('6379', 'redis');
redisClient.on('connect', function() { console.log('Redis server connected');
});

redisClient.on("error", function(err) {
  console.error("Error connecting to redis", err);
  process.exit(1);
});

// API handlers
let ConnectionHandler = require('./routes/ConnectionHandler.js');
  ConnectionHandler = new ConnectionHandler(redisClient, io);
let handlers = {
  connections: ConnectionHandler

```

```

};

// ROUTES

let routes = require('./routes/routes.js');
routes.init_routes(app, handlers);

// index page
app.get('/', function(req, res) {
  let user = auth(req);
  if (!user || !admins[user.name] || admins[user.name].password !==
user.pass) {
    res.set('WWW-Authenticate', 'Basic realm=Network Monitor');
    return res.status(401).send();
  } else {
    ConnectionHandler.get_connections(redisClient,      function(err,
connections) {
      res.render('index', {
        connections: connections
      });
    });
  }
});

// START THE SERVER
server.listen(3000);
console.log('Running on port 3000');

```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Дослідження та програмна реалізація моніторингу мережного трафіку за допомогою вейвлетаналізу

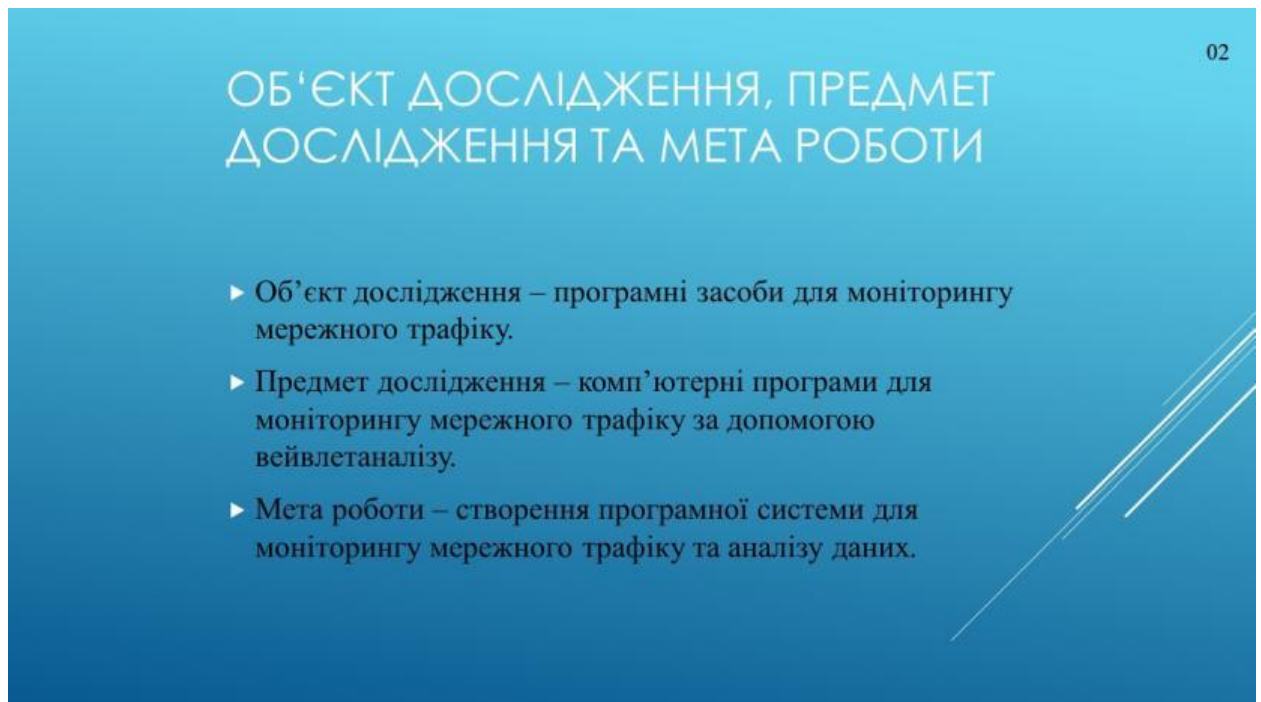


Рисунок Б.2 – Об'єкт дослідження, предмет дослідження та мета роботи

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

- Моніторинг та аналіз мереж дуже важливі для того, щоб зрозуміти ефективність мереж, надійність мереж, безпеку мереж та виявити потенційні проблеми. За допомогою аналізу мережевого трафіку служба безпеки мережі зможе ідентифікувати будь-які шкідливі або підозрілі пакети всередині трафіку, тоді як адміністратори мережі можуть контролювати швидкість завантаження, пропускну здатність тощо, а отже, мати краще розуміння мережеві операції.

Рисунок Б.3 – Аналіз предметної області

ВЕЙВЛЕТ

- Вейвлет - це мала хвиля. Вейвлетаналіз даних базується на вейвлет-перетворенні, яке використовувалось для численних досліджень з геофізики, включаючи тропічну конвекцію. Вейвлет-перетворення можна використовувати для аналізу часових рядів, що містять нестационарну потужність на багатьох різних частотах. На відміну від традиційних хвиль $\sin(t)$ або $\cos(t)$, які переходять від негативної нескінченності до позитивної нескінченності, вейвлети завжди починаються з нуля, збільшуються, а потім зменшуються назад до нуля.
- Вейвлет-перетворення - це згортка даних часової послідовності та вейвлетів, і воно може бути загалом виражене як:

$$F(a, b) = \frac{1}{\sqrt{a}} \cdot \int_{-\infty}^{\infty} f(t) \cdot \psi \left(\frac{t - b}{a} \right) \cdot dt.$$

Рисунок Б.4 – Вейвлет

БЕЗПЕРЕРВНЕ ВЕЙВЛЕТПЕРЕТВОРЕННЯ

- Вейвлет-перетворення подібне до перетворення Фур'є, але на відміну від перетворення Фур'є (яке локалізується лише у частотному просторі), вейвлет-перетворення, локалізоване як у часовому просторі, так і в частотному просторі

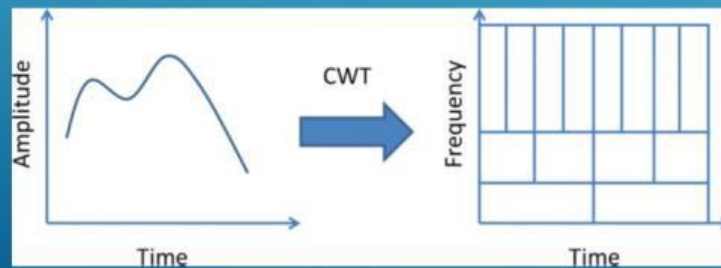
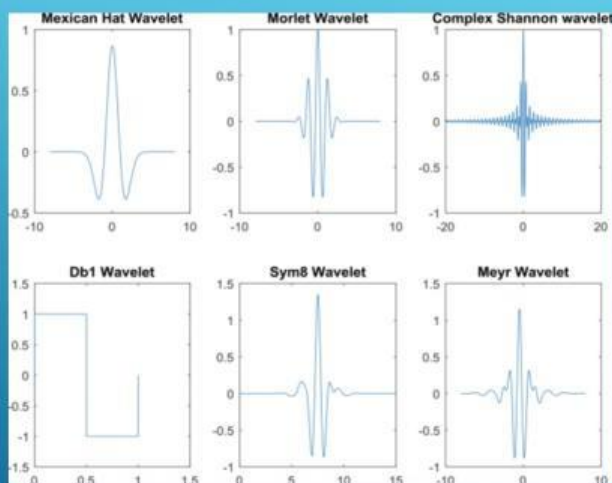


Рисунок Б.5 – Безперервне вейвлетперетворення



ПРИКЛАДИ ВЕЙВЛЕТІВ

- Вейвлет «Мексиканський капелюх»
- Вейвлети Морлі
- Вейвлет Шеннона
- Вейвлети Добеши
- Вейвлет Гаусса
- Вейвлет Мейєра

Рисунок Б.6 – Приклади вейвлетів

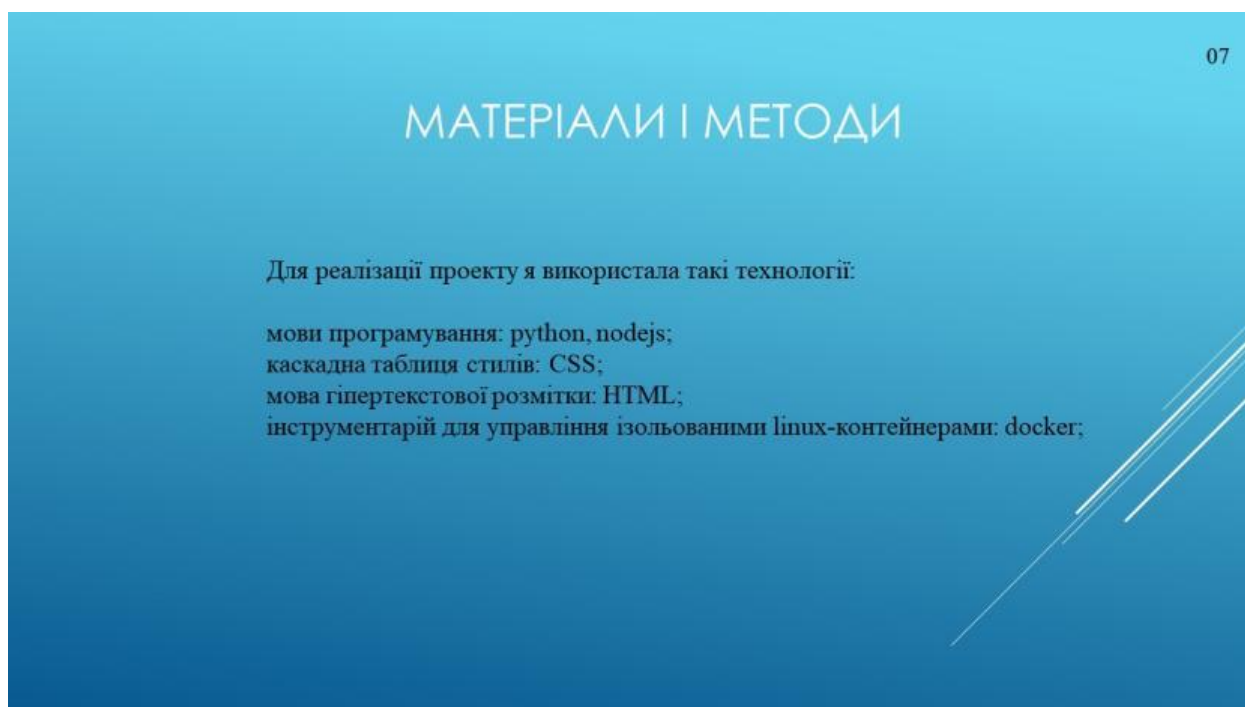


Рисунок Б.7 – Матеріали і методи

08

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

Критерій	Python	C++	Java	PHP
Наявність літератури та простота вивчення	+	-	-	+
Використання мови для написання бекенду	+	+	+	+
Продуктивність роботи	+	+	+	-
Паралельна обробка запитів та реалізація	+	+	+	+
Простота розробленого графічного інтерфейсу	-	-	-	+
Зручність роботи з локальною мережею Інтернет	+	+	-	-

Рисунок Б.8 – Порівняння мов програмування

ПОРІВНЯННЯ СЕРЕДОВИЩ РОЗРОБКИ

Критерій	PyCharm	Visual Code	PHPStorm
Наявність безкоштовної версії	+	+	+
Підтримка бібліотек та налаштувань	+	-	+
Крос-платформність	-	+	+
Швидкість роботи	-	+	-
Підтримка та оновлення	+	+	+
Написання Python коду	+	-	-
Розробка вебдодатку	-	+	+

Рисунок Б.9 – Порівняння середовищ розробки

- ▶ Node.js (або просто Node) - це серверна платформа для роботи з JavaScript через двигун V8. JavaScript виконує дію на стороні клієнта, а Node - на сервері. За допомогою Node можна писати повноцінні програми. Node вміє працювати з зовнішніми бібліотеками, викликати команди з коду на JavaScript і виконувати роль веб-сервера.
- ▶ Docker - це інструмент, спрощений для створення, розгортання та запуску програм за допомогою контейнерів. Контейнери дозволяють розробнику упакувати програму з усіма необхідними частинами, такими як бібліотеки та інші залежності, та розгорнути її як один пакет. Завдяки цьому, завдяки контейнеру, розробник може бути впевнений, що програма працюватиме на будь-якій іншій машині Linux, незалежно від будь-яких налаштованих параметрів, які може мати машина, які можуть відрізнятися від машини, яка використовується для написання та тестування коду.

Рисунок Б.10 – Node.js та Docker

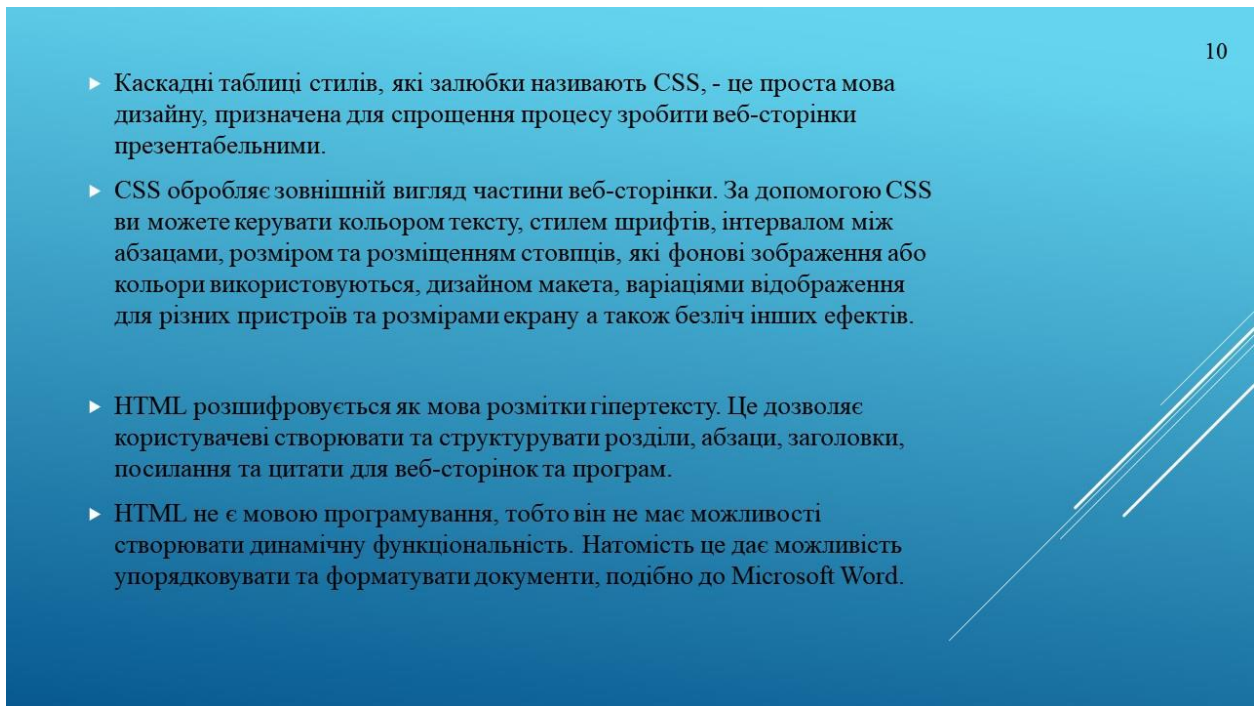


Рисунок Б.11 – Каскадна таблиця стилів CSS та розмітка гіпертексту HTML

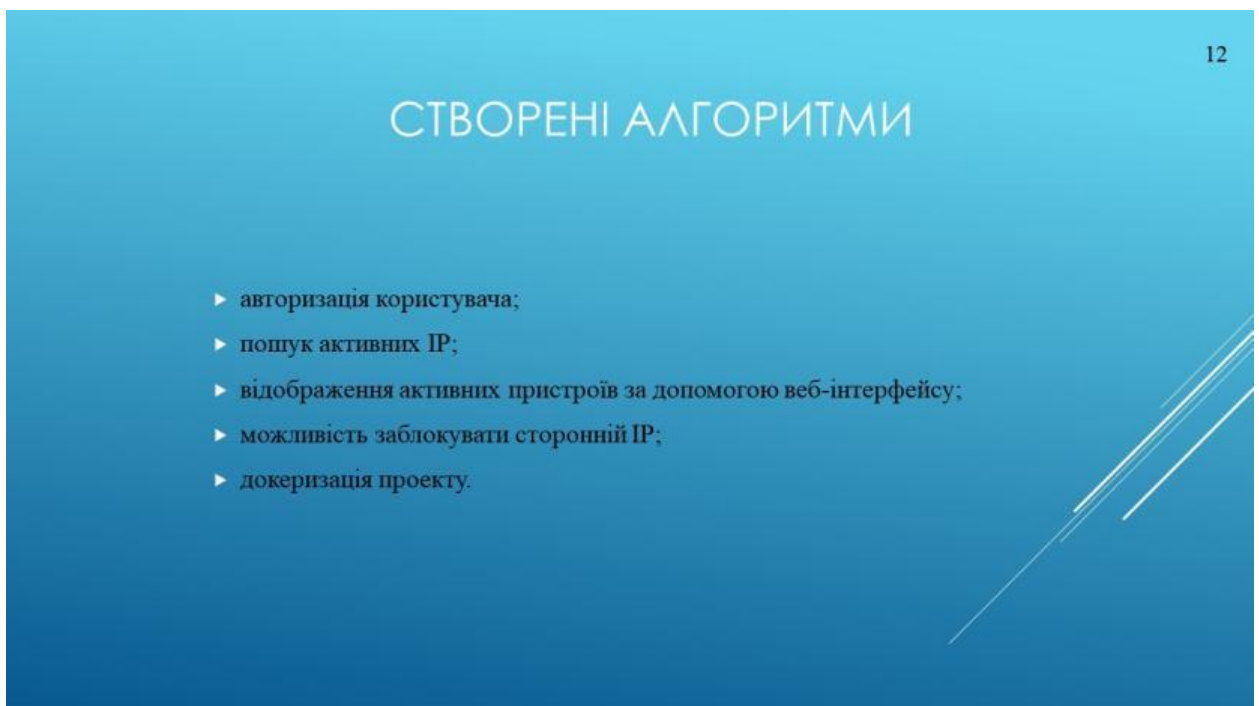


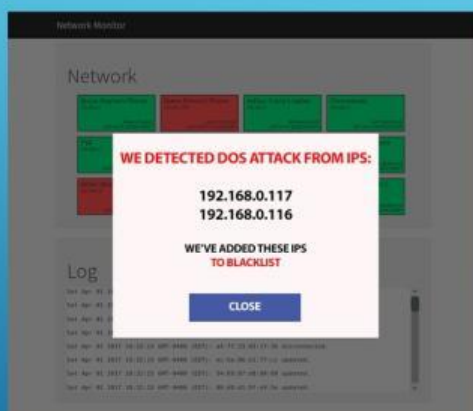
Рисунок Б.12 – Створені алгоритми

РОБОТА ПРОГРАМИ



- ▶ На наступному екрані відображається активні та неактивні IP адреси в локальній мережі. Також відображається назва пристрою, їх MAC адреса й дата останньої активності.

Рисунок Б.13 – Робота програми



- ▶ Коли відбувається DoS і Ddos атаки, користувач отримує повідомлення, яке зображене на скриншоті.

Рисунок Б.14 – Повідомлення про атаки

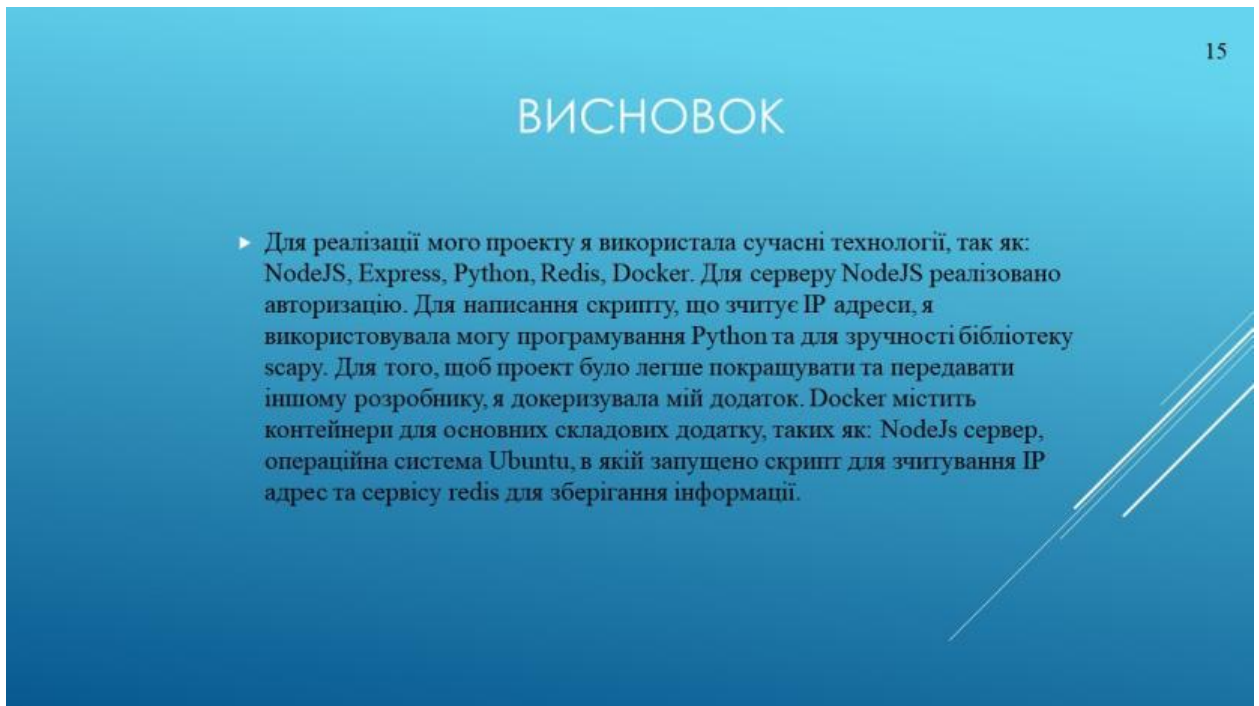


Рисунок Б.15 – Висновок