ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАТИЗАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ МОНІТОРИНГУ ЗАВАНТАЖЕНОСТІ МЕРЕЖЕВИХ ПОРТІВ У РОЗРІЗІ ГЕОГРАФІЇ ЗАПИТІВ»

Виконав: студент 5 курсу, групи ПП3–52

спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Шевчук В.М.

(прізвище та ініціали)

Керівник Негоденко О.В.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

2021 р.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти – «Бакалавр»
Спеціальність – 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Завідувач кафедри
Інженерії програмного забезпечення
_____ Негоденко О.В.
"____"_____2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ
Шевчуку В'ячеславу Миколайовичу

1. Тема проекту (роботи) " Розробка програмного засобу моніторингу завантаженості мережевих портів у розрізі географії запитів*"* затверджена наказом вищого навчального закладу від „12" березня 2021 р., № 65
2. Строк подання студентом роботи „01" червня 2021 р.
3. Вихідні дані до роботи: *Використовувати ОС Windows, СУБД SQLite, середовище об'єктно-орієнтованого проектування Visual Studio, мова програмування С++*
4. Зміст розрахунково-пояснювальної записки *(перелік питань, що які потрібно розробити):*
   4.1. Об'єктний аналіз поставленої задачі.
   4.2. Розробка моделі взаємодії даних.
   4.3. Розробка структури зберігання даних.
   4.4. Створення коду програми.
5. Перелік графічного матеріалу:
   5.1. *Інтерфейси існуючих програмних систем моніторингу мережи*
   5.2. *Інтерфейс розробленої програмної системи.*
   5.3. *Результати тестування розробленої програмної системи.*
6. Дата видачи завдання: 19.04.2021р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка:
1	Об'єктний аналіз поставленої задачі	19.04.2021	
2	Розробка моделі взаємодії даних	26.04.2021	
3	Розробка структури зберігання даних	03.05.2021	
4	Створення коду програми	10.05.2021	
5	Тестування і налагодження програми	17.05.2021	
6	Підготовка пояснювальної записки.	22.05.2021	
	Спецчастина		
7	Підготовка презентації та доповіді	23.05.2021	
8	Попередній захист	25.05.2021	
9	Подання роботи в деканат	01.06.2021	

Студент _____ Шевчук В.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Негоденко О.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 57 с., 29 рис., додатки, 15 джерел.

Мета і завдання дослідження. Метою роботи є розробка програмного засобу моніторингу завантаженості мережевих портів у розрізі географії запитів.

Об'єкт дослідження – моніторинг завантаженості мережевих портів.

Предмет дослідження – мережеві порти у розрізі географії запитів

Мета роботи – створення безкоштовного програмного забезпечення для моніторингу завантаженості мережевих портів.

Методи дослідження – методи проектування, розробки , тестування, валідації та верифікації програмного забезпечення.

Визначено – функціональні вимоги до розробки системи.

Здійснено – підготовчу роботу до розробки системи моніторингу.

На основі результатів виконаних досліджень розроблено: систему для моніторингу завантаженості мережевих портів у розрізі географії запитів.

Упровадження розробленої схеми дозволяє відслідковувати у реальному часі запити до мережевого пристрою з відображенням інформації у графічному вигляді.

ЗМІСТ

ВСТУП

Сучасне суспільство вже не може обійтися без інформаційних технологій. Інформаційні технології проникли в усі сфери життя людини. Невід'ємною частиною інформаційних технологій є глобальна мережа інтернет. Інтернет повністю змінює те, як ми працюємо, живемо, розважаємося і вчимося. Ці зміни будуть відбуватися в уже відомих нам галузях, а також в галузях, про які ми поки не підозрюємо.

Однак разом з колосальним зростанням популярності глобальної мережі інтернет виникає безпрецедентна загроза розголошення персональних даних, критично важливих корпоративних ресурсів, і т.п. Кожен день хакери піддають загрозі ці ресурси, намагаючись отримати до них доступ за допомогою спеціальних атак. Звичайно ж, необхідно захистити інформації всередині мережі тому, одним з головних завдань є забезпечення безпеки цієї інформації. Однією з небезпек є мережеві атаки. Для початку встановимо, що таке мережева атака. Мережева атака - дія, метою якого є захоплення контролю, підвищення прав, відмова в обслуговуванні, або її дестабілізація над віддаленою, або локальною обчислювальною системою, а також отримання даних користувачів які користуються цією віддаленою або локальною обчислювальною системою.

Один з найважливіших методів для боротьби з усіма видами інтернет-атак є грамотний моніторинг сайтів. Моніторинг дозволяє:

- швидко виявити і блокувати більшість атак на проникнення;
- визначити початок і напрямок атаки на відмову в обслуговування;
- сканування сайтів на цілісність і на відомі уразливості;
- дозволить звести до мінімуму ефективність більшості атак на проникнення;
- регулярне тестування навантаження сайтів дозволить спрогнозувати поведінку сайту при напливі відвідувачів після початку рекламної компанії;

- аналіз трафіку і сканування на ресурсомісткість кожної сторінки дозволить виявити потенційні напрямки атак на відмову і заздалегідь вжити необхідних заходів до оптимізації сторінок;
- регулярні моніторинг, сканування та аналіз сайту знижує вартість захисту і підвищує вартість атаки.

Оцінити їх ефективність допомагає регулярне тестування навантаження сайтів.

Моніторинг на швидке виявлення і визначити початок і напрямок атаки на відмову в обслуговуванні і є завданням “ програмного засобу моніторингу завантаженості мережевих портів у розрізі географії запитів ”.

Метою даної бакалаврської роботи є створення безкоштовного програмного забезпечення для моніторингу завантаженості мережевих портів, який допоможе користувачу моніторити мережевий трафік.

Завдання розробки є:

1. Провести аналіз предметної галузі та існуючих аналогів.
2. Провести проектування та обрати засоби для розробки програми.
3. Реалізувати програмний продукт за допомогою обраних технологій.
4. Провести тестування та оцінку програмного продукту.

Об’єкт дослідження – моніторинг завантаженості мережевих портів .

Предмет дослідження – мережеві порти у розрізі географії запитів.

Методи дослідження – методи проектування, розробки , тестування, валідації та верифікації програмного забезпечення.

Практичне значення отриманих результатів: Практичне значення отриманих результатів полягає

Дане програмне забезпечення буде розроблено для того щоб полегшити і покращити моніторинг мережевої безпеки та маршрутизації.

1. ВІДОМОСТІ ПРО ОБ’ЄКТ РОЗРОБКИ

1.1. Аналіз проблеми мережевої безпеки

Для організації комунікацій в неоднорідному мережевому середовищі застосовується набір протоколів TCP / IP, забезпечуючи сумісність між комп'ютерами різних типів. Сумісність - одна з основних переваг TCP / IP, тому більшість комп'ютерних мереж підтримує ці протоколи. Крім того, протоколи TCP / IP надають доступ до ресурсів глобальної мережі Інтернет.

Завдяки своїй популярності TCP / IP став стандартом для міжмережевої взаємодії. Однак повсюдне поширення стека протоколів TCP / IP оголило і його слабкі сторони. Створюючи своє дітище, архітектори стека TCP / IP не бачили причин для занепокоєння про захист мереж, що будуються на його основі. Тому в специфікаціях ранніх версій протоколу IP були відсутні вимоги безпеки, що призвело до початкової уразливості реалізації цього протоколу.

Проблеми забезпечення інформаційної безпеки в комп'ютерних мережах обумовлені загрозами безпеці для локальних робочих станцій, локальних мереж і атаками на корпоративні мережі, що мають вихід в загальнодоступні мережі передачі даних.

Мережеві атаки настільки ж різноманітні, як і системи, проти яких вони спрямовані. Одні атаки відрізняються великою складністю, інші може здійснити звичайний оператор.

Цілі порушника, який здійснює атаку:

• порушення конфіденційності інформації, що передається;

• порушення цілісності та достовірності передаваємої інформації;

• порушення працездатності всієї системи або окремих її частин.

Розподілені системи схильні до віддалених атак, оскільки компоненти розподілених систем зазвичай використовують відкриті канали передачі даних, і зловмисник може не тільки проводити пасивне прослуховування переданої інформації, а й активно впливати та модифікувати переданий трафік. І якщо

активний вплив на трафік може бути зафіксовано, то пасивне вплив практично не піддається виявленню. Але оскільки в ході функціонування розподілених систем обмін службовою інформацією між компонентами системи здійснюється теж по відкритим каналам передачі даних, то службова інформація стає таким же об'єктом атаки, як і дані користувача.

Труднощі виявлення факту проведення віддаленої атаки виводять цей вид неправомірних дій на перше місце за ступенем небезпеки і перешкоджає своєчасному реагуванню на здійснену загрозу, в результаті чого у порушника збільшуються шанси успішної реалізації атаки.

Безпека локальної мережі відрізняється від безпеки міжмережевої взаємодії тим, що на перше за значимістю місце виходять порушення зареєстрованих користувачів, оскільки в цьому випадку канали передачі даних локальної мережі знаходяться на контрольованій території і захист від несанкціонованого підключення до яких реалізується адміністративними методами.

На практиці IP-мережі уразливі для багатьох способів несанкціонованого вторгнення в процес обміну даними. У міру розвитку комп'ютерних та мережевих технологій список можливих типів мережевих атак на IP-мережі постійно розширюється.

Виникає питання, що ж дає порушникові можливість проникнути в чужі системи. До причин, що викликають вразливість системи можна віднести:

- відкритість системи, вільний доступ до інформації з організації мережної взаємодії, протоколам і механізмам захисту;
- наявність помилок в програмному забезпеченні, операційних системах і утиліти, які відкрито публікуються в мережі;
- різнорідність використовуваних версій програмного забезпечення і операційних систем;
- складність організації захисту міжмережевої взаємодії;
- помилки конфігурації системи і засобів захисту;
- неправильне адміністрування системи;

- "економія" на засобах забезпечення безпеки або ігнорування цих систем;
- замовчування про випадки порушення безпеки свого хоста.

Також може виникнути питання, які причини появи мережевих атак. Відповідь на це питання доволі проста. З розвитком бізнесу з'являються і різні недоброзичливці. Конкуренти, невдалі клієнти, скривджені колишні працівники - атаку може організувати будь-хто, завдяки відносній анонімності в інтернеті. Але основна причина атаки - це не бажання її організувати, а можливість.

Атаки мають різні види і способи їх проведення. В основному атаки розрізняються на відмову в обслуговуванні і в проникненні. Атака на відмову в обслуговуванні в даний час найбільш популярна, її мета - вивести з ладу сайт компанії в найбільш відповідний час. Дуже часто застосовується під час різних рекламних акцій, коли втрати від невчасно відключеного сайту найбільш істотні. Зазвичай атаки проводяться на вичерпання ресурсів сайту, або пропускної здатності інтернет-каналу до сайту.

Проникнення зазвичай використовується для поширення шкідливого коду і створення підпорядкованої мережі комп'ютерів. Зазвичай відбувається в автоматичному режимі, непомітно для власників сайту. Рідше використовується спеціально для репутаційних втрат компанії, шляхом зміни вмісту сайту для демонстрації вразливостей. Ще рідше використовується для розкрадання приватних даних компанії або її клієнтів - дуже серйозна атака, яка веде до серйозних втрат репутації та можливого судового переслідування з боку постраждалих клієнтів.

Для боротьби з усіма видами атак є грамотний моніторинг сайтів. Моніторинг дозволяє дуже швидко виявити і блокувати більшість атак на проникнення, визначити початок і напрямок атаки на відмову в обслуговування. Сканування сайтів на цілісність і на відомі уразливості дозволить звести до мінімуму ефективність більшості атак на проникнення. Регулярне тестування навантаження сайтів дозволить спрогнозувати поведінку сайту при напливі відвідувачів після початку рекламної компанії. Аналіз трафіку і сканування на ресурсомісткість

кожної сторінки дозволить виявити потенційні напрямки атак на відмову і заздалегідь вжити необхідних заходів до оптимізації сторінок.

Але існують і інші способи захисту від мережевих атак:

- Шифрування даних. Чи не є захистом як такої, але в разі витоку інформації зловмисник не прочитає її.
- Установка антивірусів і їх своєчасне оновлення.
- Застосування програм, які блокують дію сніфферов і руткітів.
- Використання брандмауера. Цей елемент виконує роль фільтра всього трафіку, що проходить через нього.

Важливим питанням є організація взаємодії систем активного аудиту (моніторингу) і загального управління. Активний аудит виконує типові керуючі функції: аналіз даних про активність в інформаційній системі, відображення поточної ситуації, автоматичне реагування на підозрілу активність. Подібним чином функціонує система мережевого управління. Активний аудит і загальне управління доцільно інтегрувати, використовуючи загальні програмно-технічні та організаційні рішення.

1.2. **Порівняльний аналіз сучасного програмного забезпечення для мережевої безпеки**

Аналіз трафіку є процесом, важливість якого відома будь-якому IT-професіоналу, не залежно від того, чи працює він в невеликій компанії або у великій корпорації. Адже виявлення і виправлення проблем з мережею - це справжнє мистецтво, яке безпосередньо залежить як від інстинкту самого фахівця, так і від глибини і якості оперованих їм даних. І аналізатор трафіку є саме тим інструментом, який ці дані надає вам. Обраний з розумом рішення для аналізу мережевого трафіку може не тільки допомогти вам з'ясувати, як пакети відправляються, приймаються і наскільки безпечно передаються по вашій мережі, але і дозволить зробити набагато-набагато більше!

Зараз на ринку представлена велика кількість варіацій програмного забезпечення для аналізу мережевого трафіку. Деякі використовують термінальний шрифт і інтерфейс командного рядка, і на перший погляд здаються складними у використанні. Інші рішення, навпаки, - виділяються простотою установки і орієнтовані на аудиторію з візуальним сприйняттям (вони буквально перенасичені різними графіками). Ціновий діапазон цих рішень також має велике значення відрізняється - від безкоштовних до рішень з дуже дорогою корпоративною ліцензією.

Для того, щоб в залежності від задач і переваг вибрати краще рішення для аналізу мережевого трафіку, список з найбільш цікавих з доступних зараз на ринку програмних продуктів для аналізу трафіку, а також короткий огляд вбудованої в них функціональності для вилучення, обробки і візуального надання різної мережевої інформації. Частина цих функцій у всіх наведених в цьому огляді рішень для аналізу мережевого трафіку схожа - вони дозволяють з тим чи іншим рівнем деталізації побачити відправлені і отримані мережеві пакети, - але практично всі з них мають деякі характерні особливості, які роблять їх унікальними при використанні в певних ситуаціях або мережевих середовищах. Зрештою, до аналізу мережевого трафіку ми вдаємося тоді, коли у нас з'явилася мережева проблема, але ми не можемо швидко звести її до певної машині, влаштуванню або протоколу, і нам доводиться проводити більш глибокий пошук.

В результаті аналізу програмних засобів для моніторингу мережи вибрав найбільш відповідні для цих цілей програмні рішення для аналізу мережевого трафіку.

SolarWinds Network Bandwidth Analyzer

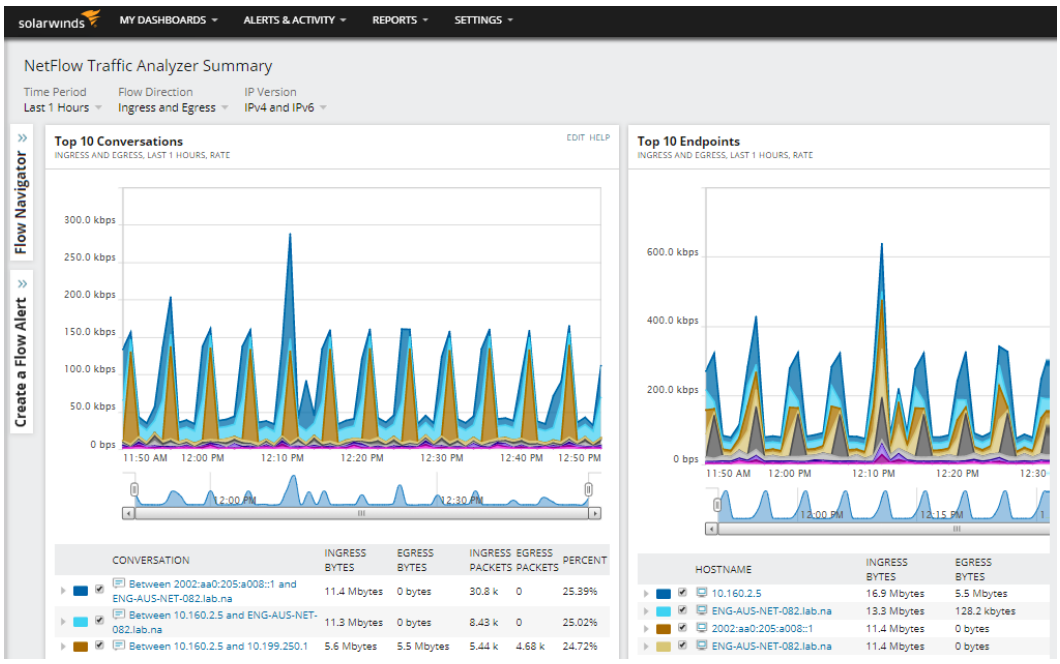


Рисунок 1.1 — SolarWinds Network Bandwidth Analyzer

Дане рішення позиціонується виробником як програмний пакет з двох продуктів - Network Performance Monitor (базове рішення) і NetFlow Traffic Analyzer (модульне розширення). Як заявляється, вони мають схожі, але все ж відрізняються функціональними можливостями для аналізу мережевого трафіку, що доповнюють один одного при спільному використанні відразу двох продуктів.

Network Performance Monitor, як випливає з назви, здійснює моніторинг продуктивності мережі і стане привабливим вибором, якщо ви хочете отримати загальне уявлення про те, що відбувається у вашій мережі. Купуючи це рішення, ви платите за можливість контролювати загальну працездатність вашої мережі: спираючись на величезну кількість статистичних даних, таких як швидкість і надійність передачі даних і пакетів, в більшості випадків ви зможете швидко ідентифікувати несправність в роботі вашої мережі. А просунуті інтелектуальні можливості програми по виявленню потенційних проблем і широкі можливості по візуальному представленню результатів у вигляді таблиць і графіків з чіткими попередженнями про можливі проблеми, ще більше полегшить цю роботу.

Модульне розширення NetFlow Traffic Analyzer більше сконцентровано на аналізі самого трафіку. У той час, як функціональність базового програмного рішення Network Performance Monitor більше призначена для отримання загального

уявлення про продуктивність мережі, в NetFlow Traffic Analyzer фокус уваги спрямований на більш детальний аналіз процесів, що відбуваються в мережі. Зокрема, ця частина програмного пакета дозволить проаналізувати перевантаження або аномальні скачки смуги пропускання і надасть статистику, відсортовану по користувачах, протоколів або додатків. Зверніть увагу, що дана програма доступна тільки для середовища Windows.

Wireshark

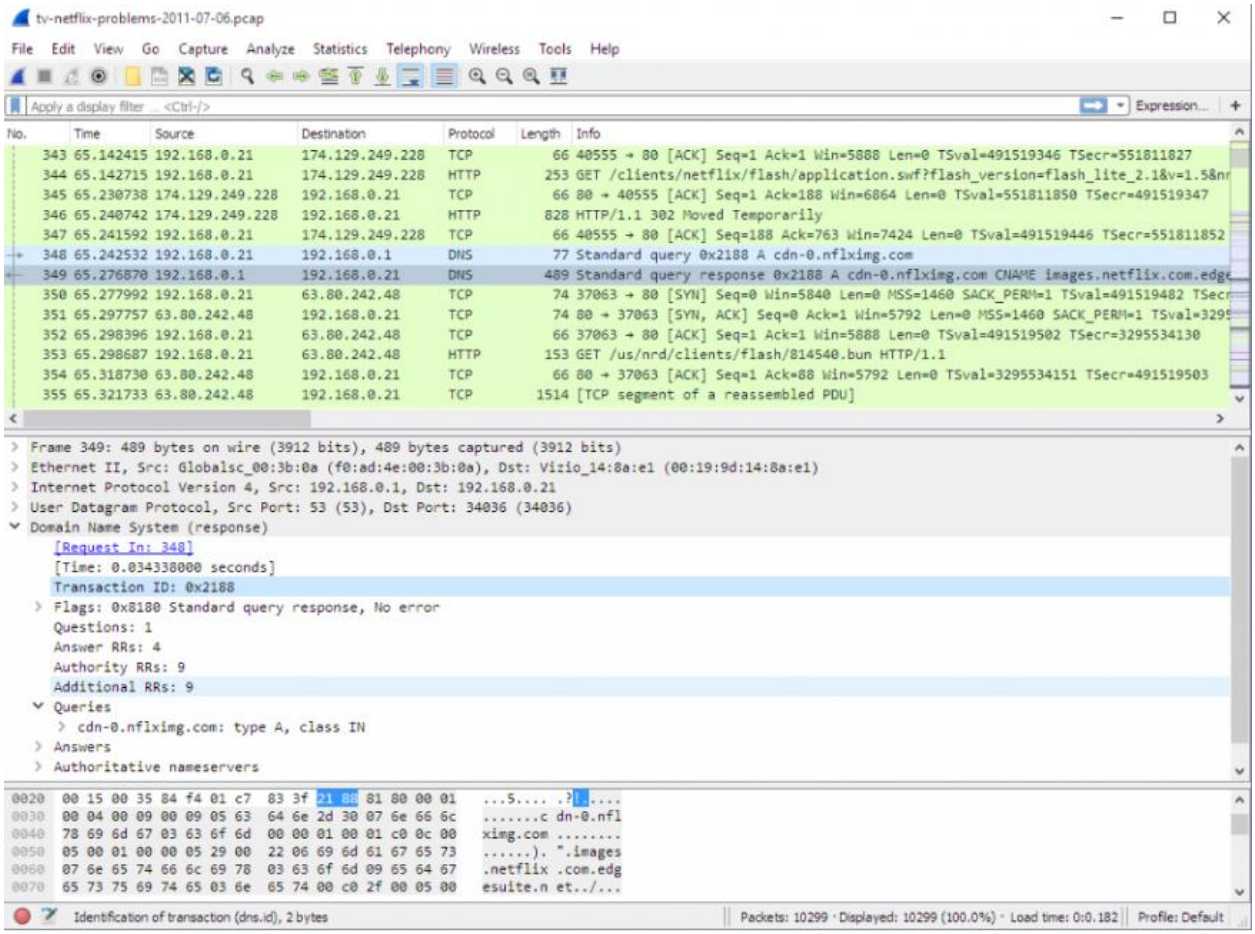


Рисунок 1.2 — Wireshark

WireShark є відносно новим інструментом у великій родині рішень для мережевої діагностики, але за цей час він вже встиг завоювати собі визнання і повагу з боку IT-професіоналів. З аналізом трафіку WireShark справляється чудово, прекрасно виконуючи для вас свою роботу. Розробники змогли знайти золоту середину між вихідними даними і візуальним представленням цих даних, тому в

WireShark ви не знайдете перекосів в ту чи іншу сторону, яким грішать більшість інших рішень для аналізу мережевого трафіку. WireShark простий, сумісний і портативний. Використовуючи WireShark, ви отримуєте саме те, що очікуєте, і отримуєте це швидко.

WireShark має прекрасний користувальницький інтерфейс, безліч опцій для фільтрації і сортування, і аналіз трафіку WireShark прекрасно працює з будь-яким з трьох найпопулярніших сімейств операційних систем - * NIX, Windows і macOS. Додайте до всього вищепереліченого той факт, що WireShark - програмний продукт з відкритим вихідним кодом і розповсюджується безкоштовно, і ви отримаєте прекрасний інструмент для проведення швидкої діагностики вашої мережі.

Tcpdump

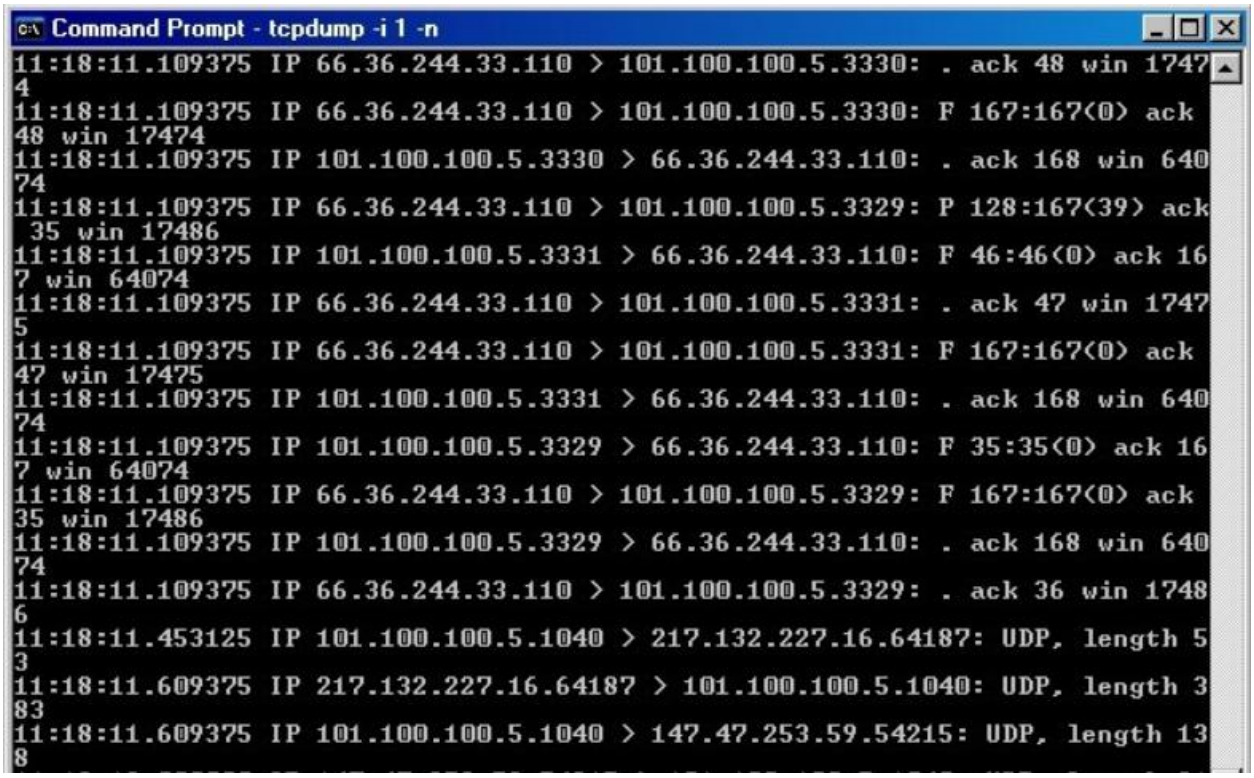


Рисунок 1.3 — tcpdump

Аналізатор трафіку tcpdump виглядає як якийсь древній інструмент, і, якщо вже бути до кінця відвертими, з точки зору функціональності працює він також. Незважаючи на те, що зі своєю роботою він справляється і справляється добре, причому використовуючи для цього мінімум системних ресурсів, наскільки це

взагалі можливо, багатьом сучасним фахівцям буде складно розібратися у величезній кількості «сухих» таблиць з даними. Але бувають в житті ситуації, коли використання настільки обрізаних і невибагливих до ресурсів рішень може бути корисно. У деяких середовищах або на ледве працюють ПК мінімалізм може виявитися єдиним прийнятним варіантом.

Спочатку програмне рішення tcpdump розроблено для середовища * NIX, але на даний момент він також працює з декількома портами Windows. Він має всю базову функціональність, яку ви очікуєте побачити в будь-якому аналізаторі трафіку - захоплення, запис і т.п., - але вимагати чогось більшого від нього не варто.

EtherApe

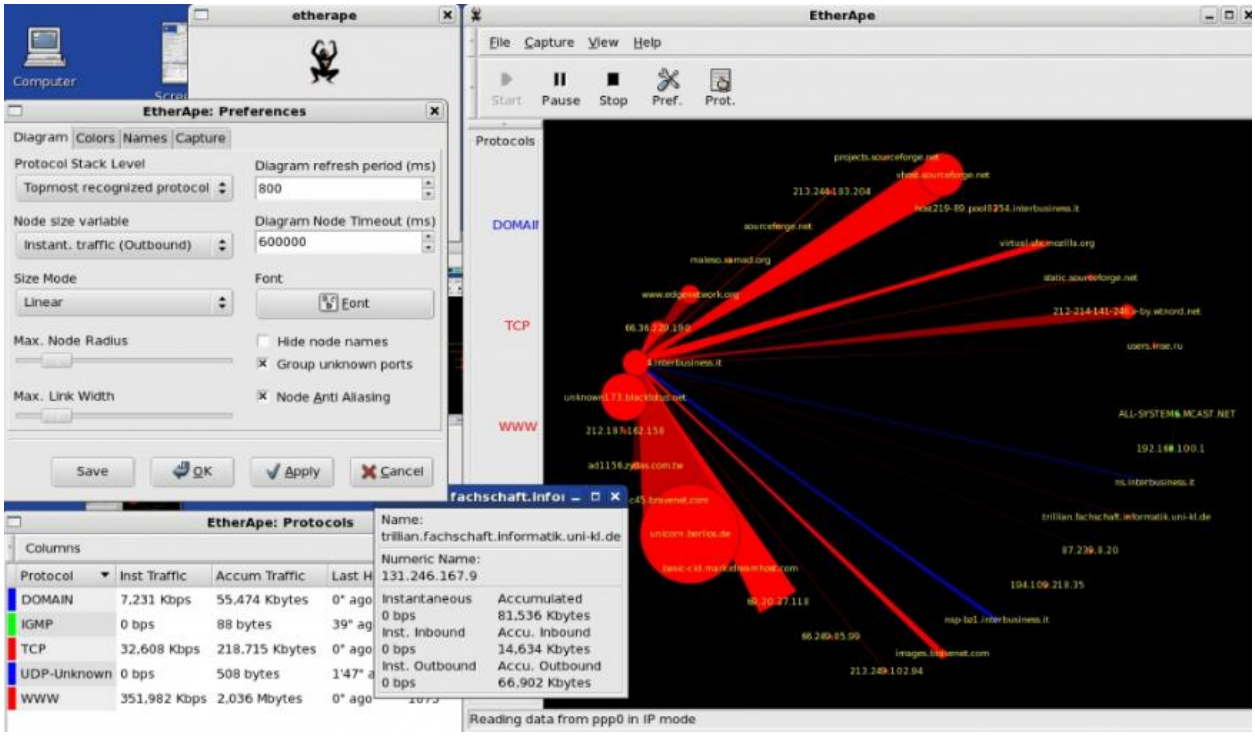


Рисунок 1.4 — EtherApe

За своїми функціональними можливостями EtherApe багато в чому наближається до WireShark, і він також є програмним забезпеченням з відкритим вихідним кодом і розповсюджується безкоштовно. Однак те, чим він дійсно виділяється на фоні інших рішень - це орієнтація на графіку. І якщо ви, наприклад, результати аналізу трафіку WireShark переглядаєте в класичному цифровому

вигляді, то мережевий трафік EtherApe відображається за допомогою просунутого графічного інтерфейсу, де кожна вершина графа являє собою окремий хост, розміри вершин і ребер вказують на розмір мережевого трафіку, а кольором відзначаються різні протоколи. Для тих людей, хто віддає перевагу візуальному сприйняттю статистичної інформації, аналізатор EtherApe може стати найкращим вибором. Доступний для середовищ * NIX і macOS.

Kismet

Рисунок 1.5 — Kismet

Аналізатор трафіка Kismet - ще один приклад програмного забезпечення з відкритим вихідним кодом, заточеного для вирішення конкретних завдань. Kismet не просто аналізує мережевий трафік, він пропонує більш широкі функціональні можливості. Наприклад, він може провести аналіз трафіку прихованих мереж і навіть безпроцесорних мереж, які не транслюють свій ідентифікатор SSID! Подібний інструмент для аналізу трафіку може бути звичайно корисним, коли у вашій безпровідній мережі є щось таке, що викликає проблеми, але швидко знайти їх джерело ви не можете. Kismet допоможе вам знайти неавторизовану мережу або точку доступу, яка працює, але має не зовсім правильні настройки.

Cain and Abel

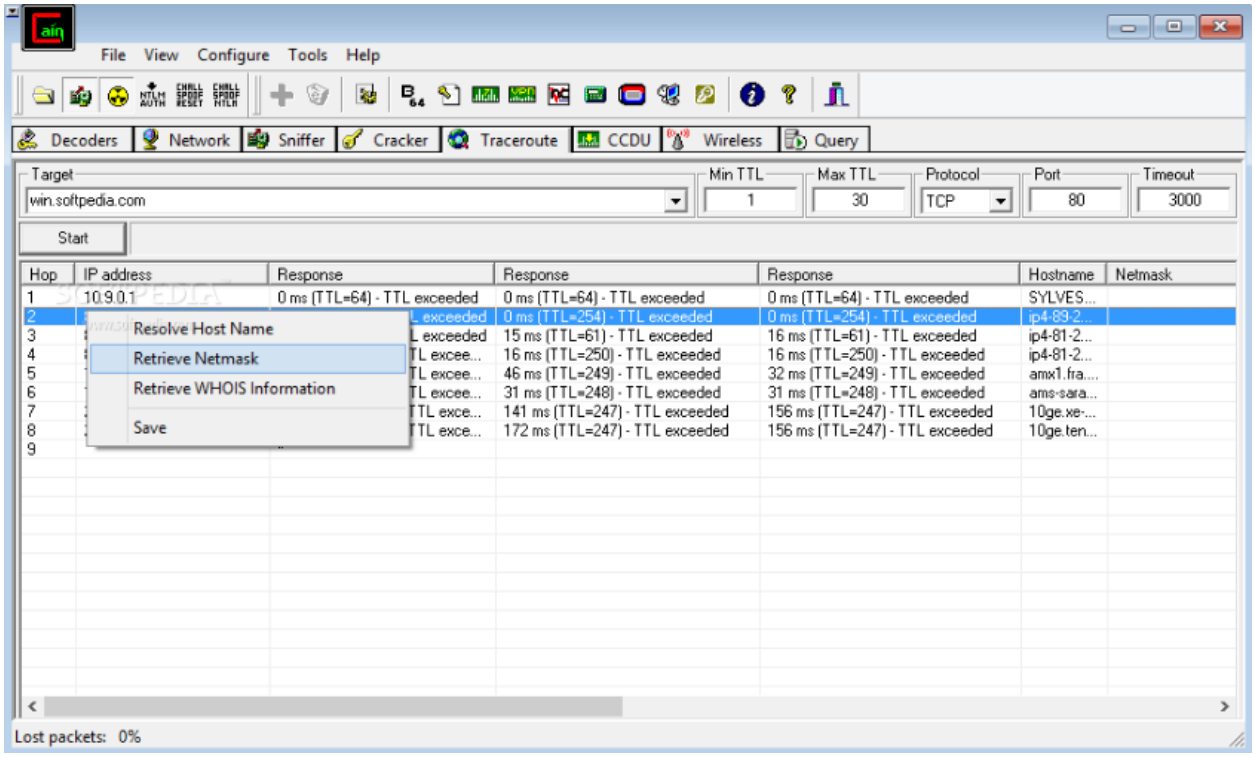


Рисунок 1.6 — Cain and Abel

У даного програмного забезпечення з вельми цікавою назвою можливість аналізу трафіку є скоріше допоміжною функцією, ніж основною. Якщо ваші завдання виходять далеко за межі простого аналізу трафіку, то вам варто звернути увагу на цей інструмент. З його допомогою ви можете відновлювати паролі для ОС Windows, проводити атаки для отримання втрачених облікових даних, вивчати дані VoIP в мережі, аналізувати маршрутизацію пакетів і багато іншого. Це дійсно потужний інструментарій для системного адміністратора з широкими повноваженнями. Працює тільки в середовищі Windows.

NetworkMiner

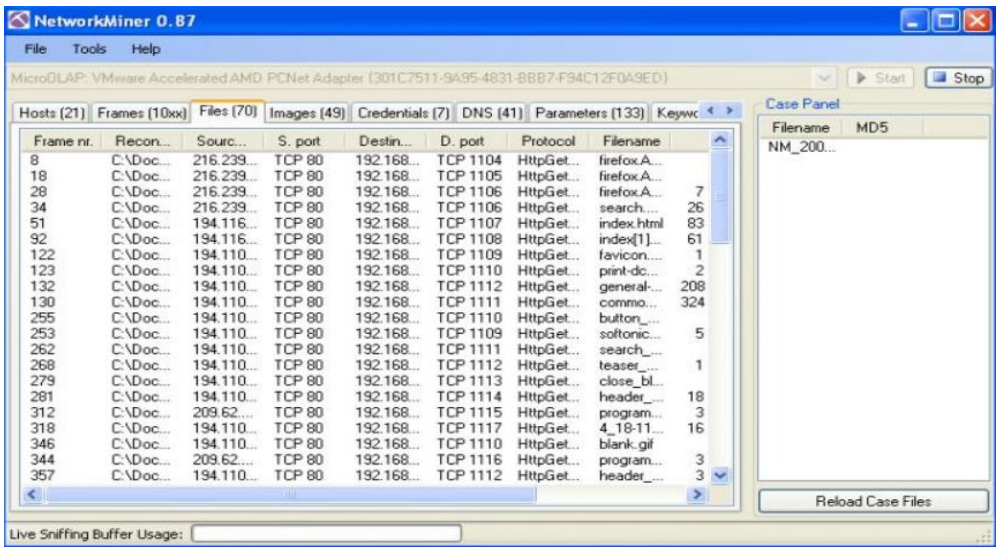


Рисунок 1.7 — NetworkMiner

NetworkMiner - ще одне програмне рішення, чия функціональність виходить за рамки звичайного аналізу трафіку. У той час як інші аналізатори трафіку зосереджують свою увагу на відправку та отримання пакетів, NetworkMiner стежить за тими, хто безпосередньо здійснює цю відправку та отримання. Цей інструмент більше підходить для виявлення проблемних комп'ютерів або користувачів, ніж для проведення загальної діагностики або моніторингу мережі як такої. NetworkMiner розроблений для ОС Windows.

В результаті аналізу цих програмних засобів для моніторингу мережевого трафіку виявив необхідний функціонал для охоплення найбільшої кількості користувачів.

Функціонал:

1. Безкоштовний
2. Відсутність реклами та платного контенту
3. Україномовний інтерфейс
4. Графічне зображення даних(за допомогою діаграм, гістограм, графіків і т.п.)
5. Простота у функціоналі

Порівняння даних додатків наведено в Таблиці 1.1

Як ми можемо бачити на даний момент не існує підходящого програмного засобу який би задовільнив всі потреби.

Підсумовуючи ми можемо зробити висновок, що програмного додатку моніторингу мережевої активності з прив’язкою до координат та відображенні на мапі світу із потрібними для нас критеріями не існує.

Таблиця 1.1 – Порівняння програмних засобів по функціоналу

Порівняння програмних засобів для моніторингу мережі					
Назва онлайн сервісу	Безкоштовна	Відсутність реклами та платного контенту	Україномовний інтерфейс	Графічне зображення даних	Простота у функціоналі
SolarWinds Network Bandwidth Analyzer	+	-	-	+	+
Wireshark	+	+	-	-	+
tcpdump	+	+	-	-	-
EtherApe	+	+	-	+	-
Kismet	+	+	-	-	+
Cain and Abel	+	-	-	-	+

2. ОПИС МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

2.1. Запропоновані методи та засоби вирішення задачі

Проаналізувавши ринок було виявлено що на даний момент відсутні програмні які б задовольняли всі потреби, а саме:

1. Безкоштовний
2. Відсутність реклами та платного контенту
3. Україномовний інтерфейс
4. Графічне зображення даних(за допомогою діаграм, гістограм, графіків і т.п.)
5. Простота у функціоналі

На основі вище перерахованого функціоналу було вирішено створити даний програмний засіб для моніторингу мережі

Щоб вирішити дану поставлену задачу було вирішено створити програмний засіб на персональний комп’ютер. Тому що, основними користувачами нашого програмного засобу спеціалісти по мережевій безпеці та мережеві адміністратори, як правило, вони користуються персональними комп’ютерами.

Фахівець з інформаційної безпеки - це людина, яка займається аналізом інформаційних ризиків компанії, розробляють і впроваджують заходи щодо їх запобігання. В його обов'язки також входить установка, настройка і супровід технічних засобів по захисту даних. Фахівці з безпеки також проводять заходи з навчання та консультації співробітників з питань забезпечення інформаційної захисту, розробляють нормативно-технічну документацію.

Фахівець з інформаційної безпеки повинен проводити збір і аналіз матеріалів організації для вжиття заходів щодо забезпечення інформаційної безпеки, а саме:

- встановлювати і налаштовувати технічні та програмні засоби захисту;
- знаходити можливі канали витоку відомостей, що становлять державну, військову, службову або комерційну таємницю;

- брати участь в розробці нових засобів автоматизації контролю, схем апаратури контролю, моделей і систем захисту інформації;
- встановлювати, налаштовувати і обслуговувати технічні та програмно-апаратні засоби захисту інформації

Головною суттю цього програмного засобу є надання користувачу можливості моніторингу мережевої активності з прив’язкою до координат та відображенні на мапі світу.

Моніторинг мережі - це аналіз трафіку і його діагностика, завантаження мережевих пристроїв і з'єднань. Перераховані процеси сприяють:

- Отриманню ефективного контролю над проблемами, що виникають в мережі. До числа таких проблем відносяться: сильне навантаження вузлів мережі і зниження їх пропускної здатності.
- Своєчасного інформування власника мережі про перевантаження системи і інших помилках, пов'язаних з діяльністю мережевих пристроїв.
- Безперервного контролю над пропускною спроможністю мережі.
- Вибору оптимальних способів вирішення проблеми. Наприклад, власник мережі може збалансувати навантаження в мережі за рахунок розподілу трафіку або вимкнути непрацююче пристрій для зменшення навантаження на мережу.
- Отриманню своєчасної інформації про те, яке обладнання слід поліпшити і де його можна придбати.

Як операційну систему для програмного засобу можна обрати:

- Windows
- Linux
- macOS

Мета будь-якої операційної системи (ОС) - допомогти спілкуванню людини з комп'ютером, через зручний і інтуїтивно зрозумілий графічний інтерфейс. З іншого боку, сама по собі ОС, це програма фон, для установки інших допоміжних програм і різних сервісів.

Почну с переваг ОС Windows:

- Простота експлуатації. У реальності, навіть непідготовленій людині дуже легко розібратися в принципах роботи операційної системи, просто недовго поспостерігавши за роботою іншої людини. Все просто і інтуїтивно зрозуміло.
- Дуже багато програм сторонніх компаній, випускають саме під цю операційну систему. Як платні, так і безкоштовні версії.
- Велика частина устаткування, що випускається для персонального ПК випускається з підтримкою ОС Windows. Веб-камери, сканери, принтери, ігрові маніпулятори і так далі і тому подібне. Всі драйвера, в першу чергу, виходять під дану ОС.
- Велика поширеність.
- При виникненні проблем, набагато простіше знайти відповідь в інтернеті, так як люди люблять ділитися своїм досвідом знайденого рішення.
- Різні форми і види оформлення. Крім стандартних варіантів, ОС дозволяє стороннім програмам вносити зміни в зовнішній вигляд робочого столу, папок, заставок і так далі.
- Можна легко відновити втрачену інформацію, як з самого комп'ютера, так і з знімних флеш-носіїв.

Недоліки ОС Windows:

- Малий набір програм в самій ОС.
- Віруси, трояни, черв'яки і так далі. Саме через масовість даної операційної системи, хакерами пишуться дуже багато програм для злому в великій кількості під Windows.
- Якщо не чистити ОС вчасно від різних тимчасових файлів, і зайвих програм, дуже швидко засмічується і починає тупити, викидати з програм, втрачати файли.

Linux - це нащадок операційних систем сімейства UNIX, спроектованих продумано і лаконічно. UNIX та потім Linux завжди розробляли не в одній компанії, а в різних лабораторіях і університетах, обмінюючись вихідними текстами програм та ідеями. Тому Linux - не монолітна система, а компонентна.

Він пристосований до того, що різні його компоненти написані незалежно різними людьми.

- Звідси його стійкість: неполадки в одній програмі не зроблять непрацездатною всю систему. Чи не станеться конфлікт і нестабільність через те, що різні сторонні додатки принесли з собою в систему один і той же компонент різних версій.
- Звідси його ефективність: різні програми використовують одні і ті ж стандартні системні засоби для стандартних операцій, а не реалізують їх самі. Це ж - вигода при розробці програм для Linux.
- Звідси його безпеку: оскільки в самій архітектурі системи передбачено обмеження доступу. Тут не потрібно витрачати ресурси на додаткові антивірусні програми, як грошові, так і системні і людські (на адміністрування самого антивіруса).

У Linux є два джерела принципових переваг перед будь-якими ОС Windows, пов'язані з принципами його поширення.

- Всі компоненти системи є вільно поширюваними.
- Відкритий вихідний текст дає можливість будь-якому користувачеві і фахівця виявити помилку і виправити її. Чим більше користувачів у відкритій програми і чим довше вона використовується, тим надійніше і стабільніше стає робота системи. Linux і системоутворюючі утиліти до нього багато років використовуються мільйонами фахівців.
- Прозорість. У Linux повністю видно всі нутрощі: всі компоненти системи і їх взаємодія не тільки доступні для вивчення, а й докладно і повно задокументовані. Це означає, що розібратися в причинах будь виниклої проблеми може не тільки вже готовий фахівець з Linux, але будь-хто, хто уважно прочитає документацію і проаналізує ситуацію.
- Гнучкість. Linux – може виглядати як завгодно. Він може виглядати як Windows, як MacOS, як щось зовсім своєрідне. Змінити можна все, тому будь-які рівні інтерфейсу відкриті і доступні для зміни.

Тепер про недоліки Linux:

- Неефективне управління електроживленням в Linux. Windows пропонує більше можливостей управління електроживленням, ніж може запропонувати Linux на даний момент. Так, Linux вміє переходити в режим сну і глибокого сну, вміє гасити екран, але пропонує вельми скромні можливості по оптимізації енергоспоживання в режимі роботи. І то за умови установки і настройки додаткових милиць - Laptop Mode Tools або Linux Advanced Power Management.
- Багато програм для Windows, мало для Linux. Існує маса прекрасних програм як під Windows, так і під Linux. Але вільні Linux-програми легко портувати під Windows, а Windows-програми під Linux - немає. Звичайно, можна спробувати змусити їх працювати під емуляторами, але результат часто сумний. І виходить, що вибір програм під Windows значно ширше. Але ж комп'ютером користуються заради доступу до прикладних програм, а не заради операційної системи. Тут Linux сильно програє.
- Марна трата часу на переучування. І KDE та GNOME, і Unity не схожі на робочий стіл Windows. Користувач повинен буде перенавчатися. Сумна правда в тому, що витрати часу на перенавчання не окупаються.
- Підтримка обладнання в Windows і Linux. Підтримка нового обладнання додається в ядро Linux досить швидко, але саме ядро може з'явитися в вашому улюбленому дистрибутиві не відразу. Для Windows доступні драйвера на момент покупки. Крім того, часто Linux-версії драйверів менш продуктивні або мають урізаною функціональністю. Приклад - драйвери відеокарт Nvidia і особливо ATI.
- Своєрідне співтовариство. Участь в співтоваристві не для всіх. Зазвичай співтовариство підноситься як величезний плюс Open Source. Але правда в тому, що зателефонувати в професійну технічну підтримку і отримати допомогу набагато простіше і швидше, ніж вислуховувати сумнівні репліки на форумах некваліфікованих користувачів.
- Документація. Не завжди документація до Linux-програм орієнтована на новачків. Правильніше навіть сказати, що вона рідко коли пишеться для

користувачів без серйозних знань в Linux. А іноді не пишеться зовсім. У разі комерційних програм ставлення до документації зовсім інше. Вона строго обов'язкове. І пишеться професійними технічними письменниками з адаптацією до різного рівня досвідченості читача.

- Мережеве взаємодія. Якщо у вашій локальній мережі переважають комп'ютери з Windows, то для обміну файлами з іншими користувачами доведеться встановити пакет Samba, який працює не завжди так, як він нього очікується і часто має проблеми з безпекою.
- Відсутність цілісності. Операційна система Windows написана однією компанією. FreeBSD написана однією командою розробників. Mac OS - те ж саме. Це добре продумані ОС, всі компоненти яких писалися відповідно до єдиного стилем. Різні розробники, існує велика кількість ліцензій, різні стилі програмування, різні підходи до зберігання файлів і конфігурацій.

Mac OS X - це операційна система на базі платформи UNIX, розроблена корпорацією Apple для лінійки комп'ютерів Macintosh. Серед користувачів ця операційна система (ОС) стає все більш популярною. Розглянемо її переваги і недоліки в порівнянні з Windows.

Переваги Mac OS:

- Графічна оболонка, з одного боку, красива, з іншого боку, не «перевантажена» спецефектами.
- Зручний для користувача інтерфейс з безліччю оригінальних функцій, що полегшують роботу з комп'ютером.
- Висока безпека в зв'язку з меншою кількістю вірусів через нешрокої поширеності цієї операційної системи.
- Поставка дистрибутивів додатків у вигляді образу диска (по подвійному кліку по файлу ОС виробляє монтування образу, після чого їм можна користуватися як звичайним розділом жорсткого диска).
- Сумісність основних форматів (DOC, RTF, PDF, JPEG, MP3, HTML і ін.) З іншими комп'ютерами.

- Стабільна робота (Mac OS розроблена спеціально для комп'ютерів Macintosh; інтегрування в операційну систему більшості драйверами для стороннього обладнання; висока сумісність компонентів, мінімальні труднощі з охолодженням).
- Простота в установці і видаленні додатків.
- Можливість установки Windows в якості другої операційної системи (за допомогою утиліти Boot Camp на всіх комп'ютерах Macintosh з процесором Intel).

Недоліки Mac OS:

- Менший в порівнянні з Windows вибір програм.
- Порівняно слабкі відеокарти.
- Менша гнучкість в управлінні розмірами і розташуванням панелей.
- Вища вартість. Комп'ютери Apple коштують дорожче приблизно в 2 рази, ніж PC аналогічної конфігурації.

Проаналізувавши переваги и недоліки операційних систем обрав варіант Windows, як найбільш підходящий для програмного забезпечення.

2.2 **Архітектуру програмного забезпечення**

Для створення програмного забезпечення обрав набір функцій Win32 API. Тому що, вона має легкий та зрозумілий функціонал для створення форм

Win32 API - це набір функцій (API - Application Programming Interface), що працюють під управлінням ОС Windows. Вони містяться в бібліотеці windows.h.

За допомогою WinAPI можна створювати різні віконні процедури, діалогові вікна, програми і навіть ігри. Ця, скажімо так, бібліотека є базовою в освоєнні програмування Windows Forms, MFC, тому що ці інтерфейси є надбудовами цієї бібліотеки.

Основними компонентами WinAPI є:

- WinBase: функції ядра, CreateFile, CreateProcess і т.п.

- WinUser: функції GUI, CreateWindow, RegisterClass і т.п.
- WinGDI: графічні функції, Ellipse, SelectObject і т.п.
- Загальні елементи управління: стандартні елементи управління, списки, повзунки і т.п.

При знайомстві з Win API виявляється, що багато вбудовані VB-функції - не що інше, як звернення до відповідних системних процедур, але тільки реалізоване у вигляді синтаксису даної мови. З огляду на це необхідність використання API визначається наступним варіантами:

- API-функції, які повністю реалізовані у вигляді вбудованих VB-функцій. Проте іноді і в цьому випадку буває корисним перейти до застосування API, так як це дозволяє часом істотно підвищити продуктивність (зокрема, за рахунок відсутності непотрібних перетворень переданих параметрів).
- Вбудовані VB-функції реалізують лише окремий випадок відповідної API-функції. Це досить типовий випадок. Наприклад, API-функція CreateDirectory має ширші можливості в порівнянні з вбудованим VB-оператором MkDir.
- Величезне число API-функцій взагалі не має аналогів в існуючому сьогодні варіанті мови VB. Наприклад, видалити каталог засобами VB не можна - для цього потрібно використовувати функцію DeleteDirectory.

Набір Win API реалізований у вигляді динамічних DLL-бібліотек. Однак, говорячи про DLL, необхідно зробити кілька важливих зауважень.

В даному випадку під DLL маю на увазі традиційний варіант довічних динамічних бібліотек, які забезпечують пряме звернення додатків до потрібних процедур - підпрограм або функцій (приблизно так само, як це відбувається при виклику процедур всередині VB-проекту). Такі бібліотеки можуть створюватися за допомогою різних інструментів: VC ++, Delphi, Fortran, крім VB (подивимося, що з'явиться у версії 7.0) - останній може робити тільки ActiveX DLL, доступ до яких можна отримати через інтерфейс OLE Automation.

Зазвичай файли динамічних бібліотек мають розширення .DLL, але це зовсім не обов'язково (для Win16 часто застосовувалося розширення .EXE); драйвери зовнішніх пристроїв позначаються за допомогою .DRV.

Windows API складається з декількох тисяч викликаються функцій, які розбиті на наступні основні категорії:

- Базові служби (Base Services).
- Служби компонентів (Component Services).
- Служби призначеного для користувача інтерфейсу (User Interface Services).
- Графічні та мультимедійні служби (Graphics and Multimedia Services).
- Обмін повідомленнями та спільна робота (Messaging and Collaboration).
- Мережа (Networking).
- Веб-служби (Web Services).

Win32 - 32х розрядний API для сучасних версій Windows. Найпопулярніша нині версія. Базові функції цього API реалізовані в DLL kernel32.dll і advapi32.dll; базові модулі GUI - в user32.dll і gdi32.dll. Win32 з'явився разом з Windows NT і потім був перенесений (в кілька обмеженому вигляді) в системи серії Windows 9x. У сучасних версіях Windows, що походять від Windows NT, роботу Win32 GUI забезпечують два модуля: csrss.exe (Client / Server Runtime Subsystem), що працює в режимі користувача, і win32k.sys в режимі ядра. Роботу ж системних Win32 API забезпечує ядро - ntoskrnl.exe

2.3 **Обрані програмні засоби**

2.3.1 **Середовище розробки**

Мною було обрана найпопулярніша платформа сімейства Windows 10 як найбільш розповсюджена платформа на території України. Пристрої на macOS занадто дорогі і мало розповсюджені на території України. А Linux не користуються великою популярністю та під нього мало інтегрованих середовищ розробки. Тому ці платформи були відкинуті для розробки мого програмного забезпечення.

Інтегроване середовище розробки - комплексне засіб, що включає все необхідне програмісту для створення програмного забезпечення.

Під інтегрованим середовищем розробки зазвичай розуміють комплексний засіб, що включає все необхідне програмісту для створення програмного забезпечення. Чіткої дефініції для цього терміна не існує. Проте, існує певний набір компонентів, які повинні бути присутніми в інтегрованих середовищах розробки:

- компілятор або інтерпретатор;
- редактор вихідного коду програм (обов'язково хоча б з підтримкою підсвічування синтаксису того мови програмування, для якого призначена середа),
- відладчик.

Відладчик - це, мабуть, навіть більше істотна частина інтегрованого середовища розробки, ніж компілятор або інтерпретатор, оскільки нерідко саме налагодження програми стає самим складним і дорогим етапом її створення.

Звичайно, сучасні інтегровані середовища розробки пропонують програмістам набагато більше можливостей, ніж входять в описаний вище необхідний мінімум. Наприклад, багато сучасних IDE є візуальними - вони дозволяють створювати інтерфейс програми за допомогою мишки, точно в такому вигляді, в якому він постане потім користувачеві. IDE, які не є візуальними, вимагають від програміста писати спеціальний код, відповідальний за створення призначеного для користувача інтерфейсу програми.

Залежно від того, для яких платформ можна писати програми і на яких платформах працює сама IDE, середовища розробки поділяються на крос-платформні (підтримують роботу з різними платформами) або переносних залежні (ті, які працюють тільки з однією платформою). Класичний приклад крос-платформної середовища розробки - Eclipse, переносних залежною - Delphi.

Залежно від кількості підтримуваних мов програмування, середовища можуть бути багатомовними або одномовними.

Список популярних середовищ розробки великий, і всі значущі продукти цього класу динамічно розвиваються в сторону все більшої зручності для розробників.

Для розробки додатків в наш час актуальними є такі середовища розробки:

Рисунок 2.3.1.1 – Логотип Microsoft Visual Studio

Microsoft Visual Studio - це інтегроване середовище розробки, ціна якої варіюється від $ 699 до $ 2900. Безліч версій цієї IDE здатні створювати всі типи програм, починаючи від веб-додатків і закінчуючи мобільними додатками, відеоіграми. Ця лінійка програмного забезпечення включає в себе безліч інструментів для тестування сумісності. Завдяки своїй гнучкості Visual Studio є відмінним інструментом для студентів та професіоналів.

Підтримувані мови: Ajax, ASP.NET, DHTML, JavaScript, JScript, Visual Basic, Visual C #, Visual C ++, Visual F #, XAML і інші.

Особливості:

- величезна бібліотека розширень, яка постійно збільшується;
- IntelliSense;
- настроюється панель і закріплюються вікна;
- простий робочий процес і файлова ієрархія;
- статистика моніторингу продуктивності в режимі реального часу;
- інструменти автоматизації;
- легкий рефакторинг і вставка фрагментів коду;
- підтримка розділеного екрану;
- список помилок, який спрощує налагодження;
- перевірка затвердження при розгортанні додатків за допомогою ClickOnce, Windows Installer або Publish Wizard.

Недоліки: оскільки Visual Studio дуже важка IDE, для відкриття і запуску додатків потрібні значні ресурси. Тому на деяких пристроях внесення простих змін може зайняти багато часу. Для простих завдань доцільно використовувати компактний редактор або засіб розробки PHP.

Рисунок 2.3.1.2 – Логотип NetBeans

Безкоштовне середовище розробки з відкритим вихідним кодом. Підходить для редагування існуючих проектів або створення нових. NetBeans пропонує простий drag-and-drop інтерфейс, який поставляється з великою кількістю зручних шаблонів проектів. Середовище в основному використовується для розробки Java додатків, але можна встановлювати пакети, які підтримують інші мови.

Мови програмування: C, C ++, C ++ 11, Fortan, HTML 5, Java, PHP і інші.

Особливості:

- інтуїтивний drag-and-drop інтерфейс;
- динамічні і статичні бібліотеки;
- інтеграція декількох сесій GNU-відладчика з підтримкою коду;
- можливість здійснювати віддалене розгортання;
- сумісність з платформами Windows, Linux, OS X і Solaris;
- підтримка Qt Toolkit;
- підтримка Fortan і Assembler;

- підтримка цілого ряду компіляторів, включаючи CLang / LLVM, Cygwin, GNU, MinGW і Oracle Solaris Studio.

Недоліки: це безкоштовне середовище розробки споживає багато пам'яті, тому може працювати повільно на деяких ПК.

Рисунок 2.3.1.3 – Логотип PyCharm

PyCharm розроблений командою Jet Brains. Користувачам надається безкоштовна версія Community Edition, 30-денна безкоштовна ознайомча версія Professional Edition і річна підписка за $ 213 - $ 690 на версію Professional Edition. Комплексна підтримка коду і аналіз роблять PyCharm кращої IDE для Python-програмістів.

Мови: AngularJS, Coffee Script, CSS, Cython, HTML, JavaScript, Node.js, Python, TypeScript.

Особливості:

- сумісність з операційними системами Windows, Linux і macOS;
- поставляється з Django IDE;
- легко інтегрується з Git, Mercurial і SVN;
- налаштовується з емуляцією VIM;

- відладчики JavaScript, Python і Django;
- підтримка Google App Engine.

Недоліки: користувачі скаржаться, що це середовище розробки Python містить деякі помилки, такі як періодично не працююча функція автоматичного заповнення, що може визвати певні незручності.

Рисунок 2.3.1.4 – Логотип IntelliJ IDEA

Ще одна IDE, розроблена Jet Brains. Вона пропонує користувачам безкоштовну версію Community Edition, 30-денну безкоштовну ознайомлювальну версію Ultimate Edition і річну підписку на версію Ultimate Edition за $ 533 - $ 693. IntelliJ IDEA підтримує Java 8 і Java EE 7, має великий інструментарієм для розробки мобільних додатків і корпоративних технологій для різних платформ. Якщо говорити про ціну, IntelliJ є прекрасним варіантом через величезного списку функцій.

Мови програмування: AngularJS, CoffeeScript, HTML, JavaScript, LESS, Node JS, PHP, Python, Ruby, Sass, TypeScript і інші.

Особливості:

- розширений редактор баз даних і дизайнер UML;
- підтримка декількох систем збірки;

- інтерфейс тестового запуску додатків;
- інтеграція з Git;
- підтримка Google App Engine, Grails, GWT, Hibernate, Java EE, OSGi, Play, Spring, Struts і інших;
- засоби розгортання і налагодження для більшості серверів додатків;
- інтелектуальні текстові редактори для HTML, CSS і Java;
- інтегрований контроль версій;
- AIR Mobile з підтримкою Android і iOS.

Недоліки: це середовище розробки JavaScript вимагає часу і зусиль на вивчення, тому може виявитися не кращим варіантом для початківців. У ній є багато поєднань гарячих клавіш, які потрібно просто запам'ятати. Деякі користувачі скаржаться на незграбний інтерфейс.

Рисунок 2.3.1.5 – Логотип Eclipse

Безкоштовний і гнучкий редактор з відкритим вихідним кодом. Він може виявитися корисним, як для новачків, так і для професіоналів. Спочатку створюється як середовище для Java-розробки сьогодні Eclipse має широкий діапазон можливостей завдяки великій кількості плагінів і розширень. Крім засобів

налагодження і підтримки Git / CVS, стандартна версія Eclipse поставляється з інструментами Java і Plugin Development Tooling. Якщо вам цього недостатньо, є багато інших пакетів: інструменти для побудови діаграм, моделювання, складання звітів, тестування і створення графічних інтерфейсів. Клієнт Marketplace Eclipse відкриває користувачам доступ до сховища плагінів та інформації.

Мови: C, C ++, Java, Perl, PHP, Python, Ruby та інші.

Особливості:

- безліч пакетних рішень, що забезпечують багатомовну підтримку;
- покращення Java IDE, такі як ієрархічні уявлення вкладених проектів;
- інтерфейс, орієнтований на завдання, включаючи повідомлення в системному треї;
- Автоматичне створення звітів про помилки;
- Параметри інструментарію для проектів JEE;
- Інтеграція з JUnit.

Недоліки: багато параметрів цього середовища розробки можуть залякати новачків. Eclipse не володіє всіма тими функціями, що і IntelliJ IDEA, але є IDE з відкритим вихідним кодом.

2.3.2 **Мови програмування**

Мови програмування які використовуються для створення програмного забезпечення в інтегрованих середовищах розробки, які наведені вище:

Рисунок 2.3.2.1 – Логотип C++

C++ є потужною мовою, успадкувавши від Сі багаті можливостей по роботі з пам'яттю. Тому нерідко C++ знаходить своє застосування в системному програмуванні, зокрема, при створенні операційних систем, драйверів, різних утиліт, антивірусів і т.п. До слова сказати, ОС Windows здебільшого написана на C++. Але тільки системним програмуванням застосування даного мови не обмежується. C++ можна використовувати в програмах будь-якого рівня, де важливі швидкість роботи і продуктивність. Нерідко він застосовується для створення графічних додатків, різних прикладних програм. Також особливо часто його використовують для створення ігор з багатою насиченою візуалізацією. Крім того, останнім часом набирає хід мобільний напрямок, де C++ теж знайшов своє застосування. І навіть в веб-розробці також можна використовувати C++ для створення веб-додатків або якихось допоміжних сервісів, які обслуговують веб-додатки. Загалом C++ - мову широкого користування, на якому можна створювати практично будь-які види програм.

C++ є компільовані мовою, а це значить, що компілятор транслює вихідний код на C++ в виконуваний файл, який містить набір машинних інструкцій. Але різні платформи мають свої особливості, тому скомпільовані програми можна просто перенести з однієї платформи на іншу і там вже запустити. Однак на рівні вихідного коду програми на C++ по більшою мірою володіють переносимість, якщо не використовуються якісь специфічні для поточної ос функції. А наявність компіляторів, бібліотек та інструментів розробки майже під всі поширені платформи дозволяє компілювати один і той же вихідний код на C++ в додатки під ці платформи.

На відміну від Сі мову C ++ дозволяє писати програми в об'єктно-орієнтованому стилі, представляючи програму як сукупність взаємодіючих між собою класів і об'єктів. Що спрощує створення великих додатків.

C++ - надзвичайно потужна мова, що містить засоби створення ефективних програм практично для будь-якого призначення, від низькорівневих утиліт і

драйверів до складних програмних комплексів самого різного призначення. Зокрема:

- висока сумісність з мовою С, що дозволяє використовувати весь існуючий С-код (код С може бути з мінімальними переробками скомпільована компілятором С++; бібліотеки, написані на С, зазвичай можуть бути викликані з С++ безпосередньо без будь-яких додаткових витрат, в тому числі і на рівні функцій зворотного виклику, дозволяючи бібліотекам, написаним на С, викликати код, написаний на С++);
- підтримуються різні стилі і технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, метапрограмування (шаблони, макроси);
- є можливість роботи на низькому рівні з пам'яттю, адресами, портами;
- можливість створення узагальнених контейнерів і алгоритмів для різних типів даних, їх спеціалізація і обчислення на етапі компіляції, використовуючи шаблони;
- кросплатформеність. Доступні компілятори для великої кількості платформ, на мові С++ розробляють програми для найрізноманітніших платформ і систем;
- ефективність. Мова спроектований так, щоб дати програмісту максимальний контроль над усіма аспектами структури і порядку виконання програми.

Частково недоліки С++ успадковані від мови-предка - Сі, - і викликані спочатку заданим вимогою можливо більшої сумісності з Сі. Це такі недоліки, як:

- синтаксис, що провокує помилки;
- препроцесор, успадкований від С, дуже примітивний;
- погана підтримка модульності (по суті, в класичному Сі модульність на рівні не доступна, її забезпечення перекладено на компонувальник). Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку заголовка (#include)

серйозно уповільнює компіляцію при підключенні великої кількості модулів (бо результуючий файл, який обробляється компілятором, виявляється дуже великий).

Рисунок 2.3.2.2 – Логотип C#

На сьогоднішній момент мова програмування C # один з найпотужніших, що швидко розвиваються і затребуваних мов в IT-галузі. На даний момент на ньому пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. І C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональність, як, наприклад, лямбда, динамічне зв'язування, асинхронні методи і т.п.

Переваги C#:

- підтримки корпорації Майкрософт. На відміну від Java, якої не пішов на користь перехід у власність Oracle, C# добре розвивається завдяки зусиллям Microsoft;

- останнім часом багато вдосконалюється. Так як C# був створений пізніше, ніж Java та іншими мовами, то потрібно дуже багато доопрацювати. Також це стосується популяризації і безкоштовності - було обіцяно відкрити вихідний код, а інструменти (Visual Studio, Xamarin) стали безкоштовними для приватних осіб і невеликих компаній;
- багато синтаксичного цукру. Синтаксичний цукор - це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;
- середній поріг входження. Синтаксис схожий на C, C++ або Java полегшує перехід для інших програмістів. Для новачків це також один з найперспективніших мов для вивчення;
- Xamarin. Завдяки покупці Xamarin на C# тепер можна писати під Android та iOS;
- додано функціональне програмування (F#);
- велика спільнота програмістів.

Недоліки C#:

- орієнтованість, в основному, тільки на .NET (на Windows платформу);
- безкоштовність тільки для невеликих компанії, учнів і програмістів-одинаків. Для великих команд покупка ліцензій обійдеться недешево;
- зберегли оператор go to.

Рисунок 2.3.2.3 – Логотип Python

Python - це високорівнева мова програмування загального призначення, яка використовується в тому числі і для розробки веб-додатків. Мова орієнтована на підвищення продуктивності розробника і читання коду.

Python підтримує кілька парадигм програмування: структурний, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване. У мові присутня динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних. Програмний код на Python організовується у функції та класи, які можуть об'єднуватися в модулі, а вони в свою чергу можуть бути об'єднані в пакети. Python зазвичай використовується як інтерпретується, але може бути скомпільован в байт-код Java і в MSIL (в рамках платформ .NET).

Переваги Python:

- низький поріг входження: людині, знайомому з програмуванням, досить півгодини, щоб почати писати на ньому корисні для себе скрипти, а не знайомому - Python дозволяє легко відкрити для себе програмування і спробувати свої сили в ньому;
- добре спроектований: Python увібрав в себе сучасні тенденції в програмуванні «з нуля». Крім того, він динамічно розвивається: процес включення нових конструкцій в мову добре налагоджений, і він продовжує вбирати в себе

прийоми функціонального програмування, аспектно-орієнтованого програмування та іншого, залишаючись при цьому назад-сумісним і внутрішньо несуперечливим;

- легко читається синтаксис (в порівнянні з C ++, Perl, PHP): дозволяє легко читати чужий код, розбиратися в давно написаному власному коді. У поєднанні зі сказаним вище це налаштовує творців бібліотек на простоту і логічність інтерфейсів;
- величезна кількість бібліотек з кодом на будь-який випадок життя: будь то робота з таблицями Excel, зображеннями або мережею Twitter;
- переносимість: Python реалізований під усіма поширеними операційними системами і на безлічі архітектур - Windows, Linux, MacOS, навіть на міні-комп'ютерах Arduino. Система залежностей добре продумана, і розгортання додатків на іншій машині відбувається легко і без сюрпризів.

Недоліки Python:

- Python не найшвидша серед мов програмування. Швидкість виконання програм може бути нижче;
- не найзручніший мову для мобільних розробок;
- проблеми з потоками. Global Interpreter Lock (GIL) допускає виконання лише одного потоку в кожен окремий момент. Це створює помітні обмеження для використання мови Python;
- через гнучкості типів даних споживання пам'яті Python не мінімальна.

Рисунок 2.3.2.4 – Логотип Java

Java - це технологія, яку використовують для розробки онлайн-додатків, тобто програм, що запускаються і працюють прямо в вашому браузері.

Ключовою особливістю мови Java є те, що його код спочатку транслюється в спеціальний байт-код, сумісний із різними платформами. А потім цей байт-код виконується віртуальною машиною JVM (Java Virtual Machine). В цьому плані Java відрізняється від стандартних різних мов як PHP або Perl, код яких відразу ж виконується інтерпретатором. У той же час Java не є і чисто компільовані мовою, як C або C ++.

Подібна архітектура забезпечує багатоплатформність і апаратну переносимість програм на Java, завдяки чому подібні програми без перекомпіляції можуть виконуватися на різних платформах - Windows, Linux, macOS і т.п. Для кожної з платформ може бути своя реалізація віртуальної машини JVM, але кожна з них може виконувати один і той же код.

Ще однією ключовою особливістю Java є те, що вона підтримує автоматичну збірку сміття. А це означає, що не треба звільняти вручну пам'ять від раніше використаних об'єктів, як в C++, так як збирач сміття це зробить автоматично за вас.

Java є об'єктно-орієнтованою мовою. Вона підтримує поліморфізм, успадкування, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків.

Переваги Java:

- об'єктно-орієнтована: в Java все є об'єктом. Доповнення може бути легко розширено, тому що він заснований на об'єктної моделі;
- платформно незалежний: на відміну від багатьох інших мов, включаючи C і C++, Java, коли була створена, не компілювалась в платформі конкретної машини, а в незалежній від платформи байт-коді. Цей байт код

поширюється через інтернет і інтерпретується в Java Virtual Machine (JVM), на якій він в даний час працює;

- простий: процеси вивчення та введення в мову програмування Java залишаються простими. Якщо Ви розумієте основні концепції об'єктно-орієнтованого програмування, то він буде простий для Вас в освоєнні;
- безпечним: методи перевірки автентичності засновані на шифруванні з відкритим ключем;
- архітектурно-нейтральним: компілятор генерує архітектурно-нейтральні об'єкти формату файлу, що робить скомпільований код виконуваним на багатьох процесорах, з наявністю системі Java Runtime;
- портативний: архітектурно-нейтральний і не має залежності від реалізації аспектів специфікацій - все це робить Java портативним. Компілятор в Java написаний на ANSI C з чистою переносимість, який є підмножиною POSIX;
- багатопоточність: функції багатопоточності, можна писати програми, які можуть виконувати безліч завдань одночасно. Введення в мову Java цієї конструктивної особливості дозволяє розробникам створювати налагоджені інтерактивні додатки;
- інтерпретований: Java байт-код переводиться на льоту в машинні інструкції та ніде не зберігається. Роблячи процес більш швидким і аналітичним, оскільки зв'язування відбувається як додаткове з невеликою вагою процесу;
- високопродуктивний: введення Just-In-Time компілятора, дозволило отримати високу продуктивність;
- поширений: призначений для розподіленої середовища інтернету;
- динамічний: програмування на Java вважається більш динамічним, ніж на C або C ++, так як він призначений для адаптації до мінливих умов. Програми можуть виконувати велике кількість під час обробки інформації, яка може бути використана для перевірки і дозволу доступу до об'єктів на час виконання.

Недоліки Java:

- низьке, в порівнянні з іншими мовами, швидкодія, підвищені вимоги до обсягу оперативної пам'яті (ОП);
- великий обсяг стандартних бібліотек і технологій створює складності у вивченні мови;
- постійний розвиток мови викликає наявність як застарілих, так і нових засобів, що мають одне і те ж функціональне призначення.

2.3.3 **База даних**

База даних (БД) - це організована структура, призначена для зберігання, зміни і обробки взаємозалежної інформації, переважно великого обсягу. Бази даних активно використовуються для динамічних сайтів зі значними обсягами даних - часто це інтернет-магазини, портали, корпоративні сайти. Такі сайти зазвичай розроблені за допомогою серверного мови програмування (як приклад, PHP) або на основі CMS (як приклад, WordPress), і не мають готових сторінок з даними за аналогією з HTML-сайтами. Сторінки динамічних сайтів формуються «на льоту» в результаті взаємодії скриптів і баз даних після відповідного запиту клієнта до веб-сервера.

Система управління базами даних (СУБД) - це комплекс програмних засобів, необхідних для створення структури нової бази, її наповнення, редагування вмісту і відображення інформації.

Рисунок 2.3.3.1 – Логотип SQLite

SQLite - компактна вбудована реляційна база даних. Вихідний код бібліотеки переданий в суспільне надбання. Є чисто реляційною базою даних.

Слово «вбудовується» означає, що SQLite не використовує парадигму клієнт-сервер. Тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а надає бібліотеку, з якої програма компонується і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується; ACID [1] -функції досягаються в тому числі за рахунок створення файлу журналу.

Переваги SQLite:

- файлова структура - вся база даних складається з одного файлу, тому її дуже легко переносити на різні машини;
- відсутність необхідності настройки сервера СУБД;
- повністю вільна ліцензія;
- кросплатформеність;
- висока швидкість простих операцій вибірки даних;
- підтримка транзакцій, тригерів, уявлень (views), вкладених запитів;
- безпека. БД зберігається в одному файлі, права доступу до якого можна контролювати стандартними засобами ОС;
- дуже економічна, в плані ресурсів, архітектура.

Недоліки SQLite:

- відсутність системи користувачів - більші СУБД включають в свій склад системи управління правами доступу користувачів. Зазвичай застосування цієї функції не так критично, так як ця СУБД використовується в невеликих додатках.
- відсутність можливості збільшення продуктивності - знову, виходячи з проектування, досить складно вичавити щось більш продуктивне з цієї СУБД.

Рисунок 2.3.3.2 – Логотип MySQL

MySQL - це реляційна система управління базами даних з відкритим вихідним кодом. В даний час ця СУБД одна з найбільш популярних в веб-додатках - переважна більшість CMS використовує саме MySQL (часто тільки її, без альтернатив), а майже всі веб-фреймворки підтримують MySQL вже на рівні базової конфігурації (без додаткових модулів).

Як це випливає з назви, в даній бібліотеці використовується формальна мова SQL (Structured Query Language), на якому створюються запити до баз даних. Основний інструмент для роботи з базами даних MySQL - phpMyAdmin.

Переваги MySQL:

- простота у використанні. MySQL досить легко інсталюється, а наявність безлічі плагінів і допоміжних додатків спрощує роботу з базами даних;
- великий функціонал. Система MySQL має практично всі необхідні інструменти, які можуть знадобитися в реалізації практично будь-якого проекту;
- безпека. Система спочатку створена таким чином, що безліч вбудованих функцій безпеки в ній працюють за замовчуванням;

- масштабованість. Будучи досить універсальної СУБД, MySQL в рівній мірі легко може бути використана для роботи і з малими, і з великими об'ємами даних;
- швидкість. Висока продуктивність системи забезпечується за рахунок спрощення деяких використовуваних в ній стандартів.

Недоліки MySQL:

- недостатня надійність. У питаннях надійності деяких процесів по роботі з даними (наприклад, зв'язок, транзакції, аудит) MySQL поступається деяким іншим СУБД;
- Низька швидкість розробки. Як і багатьом іншим програмним продуктам з відкритим кодом, MySQL бракує деякого технічної досконалості, що часом позначається на ефективності процесів розробки.

Рисунок 2.3.3.2 – Логотип MongoDB

MongoDB - це крос-платформна, документо-орієнтована база даних, яка забезпечує високу продуктивність і легку масштабованість. В основі даної БД лежить концепція колекцій і документів.

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційним базам даних.

На відміну від реляційних баз даних MongoDB пропонує документо-орієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати.

Але, навіть враховуючи всі недоліки традиційних баз даних і гідності MongoDB, важливо розуміти, що завдання бувають різні і методи їх вирішення бувають різні. В якійсь ситуації MongoDB дійсно поліпшить продуктивність вашої програми, наприклад, якщо треба зберігати складні за структурою дані. В іншій же ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішані підхід: зберігати один тип даних в MongoDB, а інший тип даних - в традиційних БД.

Вся система MongoDB може представляти не тільки одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на декількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними і зберігати цілісність.

Переваги MongoDB:

- схема менше. Якщо у вас гнучка схема, ідеально підходить для сховища документів;
- легкість масштабування. Масштабування читається з використанням наборів реплік. Додавання більшої кількості машин збільшує кількість ОЗУ для поширення робочого набору;
- вартість. MongoDB є безкоштовним і може працювати на Linux;
- можна вибрати, який рівень узгодженості ви хочете в залежності від значення даних.

Недоліки MongoDB:

- розмір даних в MongoDB зазвичай вище через, наприклад, що кожен документ має імена полів, які зберігаються в ньому;
- менше гнучкості при запиті;
- немає підтримки транзакцій - підтримуються певні атомарні операції на одному рівні документа;
- менш актуальна інформація доступна тому що, швидко розвивається продукт.

2.3.4. **Висновки**

Проаналізувавши інтегровані середовища розробки, прийняв рішення що, найкращим для мене вибором середовища розробки є Visual Studio тому що, в нього присутній такі переваги:

- інтуїтивний стиль кодування. Visual Studio форматує код у міру його введення, автоматично вставляючи необхідні відступи і застосовуючи колірне кодування для виділення елементів типу коментарів;
- можливості налагодження. Пропоновані в Visual Studio інструменти налагодження є найкращим засобом для відстеження помилок і діагностування дивної поведінки;
- підтримка безлічі мов при розробці.

Головною мовою програмування для даного програмного забезпечення була обрана мова C++ адже вона має такі переваги порівняно з іншими мовами програмування:

- підтримує різні парадигми програмування;
- доступність літератури, документації;
- висока продуктивність;
- багатопоточність.

Як СУБД мною була обрана SQLite бо вся база даних складається з одного файлу, тому її дуже легко переносити на різні машини в неї відсутня необхідності настройки сервера СУБД, має повністю вільну ліцензію та висока швидкість простих операцій для вибірки даних.

2.4 **Структура даних**

При розробці даного програмного забезпечення використав базу даних SQLite в якій було створено такі таблиці:

1. GeoLite2-City-Blocks-IPv4_1 — дані про IP,

2. GeoLite2-City-Locations-ru_2 — дані про країну та місто.

Таблиця GeoLite2-City-Blocks-IPv4_1 містить такі поля як:

- string network — IP адрес;
- int geoname_id — ключ к даним таблиці;
- float latitude — широта;
- float longitude — довгота;
- int accuracy_radius — радіус точності.

Таблиця GeoLite2-City-Locations-ru_2 містить такі поля як:

- int geoname_id — ключ к даним таблиці;
- string continent_name — назва континенту;
- string country_name — назва країни;
- string city_name — назва міста.

2.5 **Особливості реалізації**

Процес реалізації програмного забезпечення включає в себе такі етапи як

- створення та реалізація локальної бази даних;
- створення інтерфейсу;
- написання коду який взаємодіє з візуальною частиною та опрацювання логіки програмного забезпечення.

Для створення та реалізації локальної бази була обрана СУБД SQLite її треба встановити та завантажити базу даних GeoIP з геолокацією прив’язаною до IP для використання.

```
sqlite> .mode csv
sqlite> .import GeoIP2.csv GeoIP2
GeoIP2.csv:3714: expected 10 columns but found 1 - filling the rest with NULL
sqlite> SELECT * FROM GeoIP2;
1.0.0.0/24,2078025,2077456,"",0,0,5000,-34.9281,138.5999,1000
1.0.1.0/24,1814991,1814991,"",0,0,"",34.7725,113.7266,50
1.0.2.0/23,1814991,1814991,"",0,0,"",34.7725,113.7266,50
1.0.4.0/22,2077456,2077456,"",0,0,"",-33.4940,143.2104,1000
1.0.8.0/21,1814991,1814991,"",0,0,"",34.7725,113.7266,50
1.0.16.0/20,1861060,1861060,"",0,0,"",35.6900,139.6900,500
1.0.32.0/19,1814991,1814991,"",0,0,"",34.7725,113.7266,50
1.0.64.0/24,1863018,1861060,"",0,0,738-0051,34.3830,132.5459,10
1.0.65.0/24,1862415,1861060,"",0,0,734-0011,34.3401,132.4439,10
1.0.66.0/23,1863018,1861060,"",0,0,738-0041,34.3426,132.3071,20
1.0.68.0/24,6822110,1861060,"",0,0,732-0011,34.2982,133.0858,50
1.0.69.0/24,1862415,1861060,"",0,0,731-5151,34.3831,132.5851,50
1.0.70.0/24,1862415,1861060,"",0,0,731-0111,34.4501,132.4884,10
1.0.71.0/24,1863018,1861060,"",0,0,738-0051,34.3830,132.5459,10
1.0.72.0/23,1862413,1861060,"",0,0,736-0014,34.3668,132.5529,50
1.0.74.0/24,1862415,1861060,"",0,0,735-0001,34.2848,132.4777,10
1.0.75.0/24,1862415,1861060,"",0,0,731-0121,34.4593,132.4731,1
1.0.76.0/22,1862415,1861060,"",0,0,731-0111,34.4501,132.4884,10
1.0.80.0/23,1857550,1861060,"",0,0,690-0121,35.4774,133.0383,200
1.0.82.0/24,1849892,1861060,"",0,0,680-0011,35.5000,134.2333,200
1.0.83.0/24,1849892,1861060,"",0,0,680-0031,35.4917,134.2333,200
1.0.84.0/23,1849892,1861060,"",0,0,680-0001,35.5314,134.2118,100
```

Рисунок 2.5.1 – завантаження бази даних

GeoIP це проект визначення географічного положення по IP адресі. Проект являє собою набір баз даних і інструментів роботи з ними.

GeoIP2.csv

```
network,geoname_id,registered_country_geoname_id,represented_country_geoname_id,is_anonymous_proxy,is_satellite_provider,postal_code,latitude,longitude,accuracy_radius
1.0.0.0/24,2078025,2077456,,0,0,5000,-34.9281,138.5999,1000
1.0.1.0/24,1814991,1814991,,0,0,,34.7725,113.7266,50
1.0.2.0/23,1814991,1814991,,0,0,,34.7725,113.7266,50
1.0.4.0/22,2077456,2077456,,0,0,,-33.4940,143.2104,1000
1.0.8.0/21,1814991,1814991,,0,0,,34.7725,113.7266,50
1.0.16.0/20,1861060,1861060,,0,0,,35.6900,139.6900,500
1.0.32.0/19,1814991,1814991,,0,0,,34.7725,113.7266,50
1.0.64.0/24,1863018,1861060,,0,0,738-0051,34.3830,132.5459,10
1.0.65.0/24,1862415,1861060,,0,0,734-0011,34.3401,132.4439,10
1.0.66.0/23,1863018,1861060,,0,0,738-0041,34.3426,132.3071,20
1.0.68.0/24,6822110,1861060,,0,0,732-0011,34.2982,133.0858,50
1.0.69.0/24,1862415,1861060,,0,0,731-5151,34.3831,132.5851,50
1.0.70.0/24,1862415,1861060,,0,0,731-0111,34.4501,132.4884,10
1.0.71.0/24,1863018,1861060,,0,0,738-0051,34.3830,132.5459,10
1.0.72.0/23,1862413,1861060,,0,0,736-0014,34.3668,132.5529,50
1.0.74.0/24,1862415,1861060,,0,0,735-0001,34.2848,132.4777,10
1.0.75.0/24,1862415,1861060,,0,0,731-0121,34.4593,132.4731,1
1.0.76.0/22,1862415,1861060,,0,0,731-0111,34.4501,132.4884,10
1.0.80.0/23,1857550,1861060,,0,0,690-0121,35.4774,133.0383,200
1.0.82.0/24,1849892,1861060,,0,0,680-0011,35.5000,134.2333,200
1.0.83.0/24,1849892,1861060,,0,0,680-0031,35.4917,134.2333,200
1.0.84.0/23,1849892,1861060,,0,0,680-0001,35.5314,134.2118,100
1.0.86.0/23,1848277,1861060,,0,0,683-0851,35.4611,133.6389,20
1.0.88.0/23,1853174,1861060,,0,0,684-0061,35.5295,133.2102,50
1.0.90.0/23,1857550,1861060,,0,0,690-0121,35.4774,133.0383,20
1.0.92.0/24,1862415,1861060,,0,0,730-0031,34.3926,132.4543,200
1.0.93.0/24,1857550,1861060,,0,0,690-0041,35.4507,133.0787,200
1.0.94.0/23,1862415,1861060,,0,0,730-0031,34.3926,132.4543,200
1.0.96.0/25,1849892,1861060,,0,0,680-0001,35.5314,134.2118,100
1.0.96.128/25,1857550,1861060,,0,0,690-2101,35.4000,133.1167,100
1.0.97.0/24,1853174,1861060,,0,0,684-0061,35.5295,133.2102,20
1.0.98.0/23,1857550,1861060,,0,0,699-0101,35.4326,133.1625,50
1.0.100.0/22,1857550,1861060,,0,0,690-0861,35.4498,133.0435,20
1.0.104.0/23,1857550,1861060,,0,0,690-0121,35.4774,133.0383,5
1.0.106.0/24,1857550,1861060,,0,0,690-0821,35.4498,133.0435,50
1.0.107.0/24,1849892,1861060,,0,0,682-0711,35.5417,134.1000,100
1.0.108.0/23,1849892,1861060,,0,0,680-0941,35.5362,134.1851,100
1.0.110.0/24,1857550,1861060,,0,0,690-0131,35.4774,133.0383,100
1.0.111.0/24,1849892,1861060,,0,0,680-0941,35.5362,134.1851,100
1.0.112.0/23,1849892,1861060,,0,0,680-0901,35.5362,134.1851,50
1.0.114.0/23,1849892,1861060,,0,0,680-0011,35.5000,134.2333,10
1.0.116.0/24,1848277,1861060,,0,0,683-0051,35.4333,133.3500,50
1.0.117.0/25,1858301,1861060,,0,0,689-2111,35.4689,133.5862,50
1.0.117.128/25,1848277,1861060,,0,0,683-0051,35.4333,133.3500,50
```

Рисунок 2.5.2 – база даних GeoIP

Щоб програмне забезпечення могло працювати з SQLite необхідно підключити заголовок sqlite3.h в якому описані методи роботи з sqlite3.lib.

sqlite3.h

C_NetSniffer (Глобальная область)

```
31  ** part of the build process.
32  */
33  #ifndef SQLITE3_H
34  #define SQLITE3_H
35  #include <stdarg.h>     /* Needed for the definition of va_list */
36
37  /*
38  ** Make sure we can call this stuff from C++.
39  */
40  #ifdef __cplusplus
41  extern "C" {
42  #endif
43
44
45  /*
46  ** Provide the ability to override linkage features of the interface.
47  */
48  #ifndef SQLITE_EXTERN
49  # define SQLITE_EXTERN extern
50  #endif
51  #ifndef SQLITE_API
52  # define SQLITE_API
53  #endif
54  #ifndef SQLITE_CDECL
55  # define SQLITE_CDECL
56  #endif
57  #ifndef SQLITE_APICALL
58  # define SQLITE_APICALL
59  #endif
```

Рисунок 2.5.3 – заголовок sqlite3.h

Інтерфейс програмного забезпечення складається з головного вікна на якому зображена мапа світу з географічною прив’язкою IP адрес. Приклад такого вікна представлений на рисунку 2.5.4

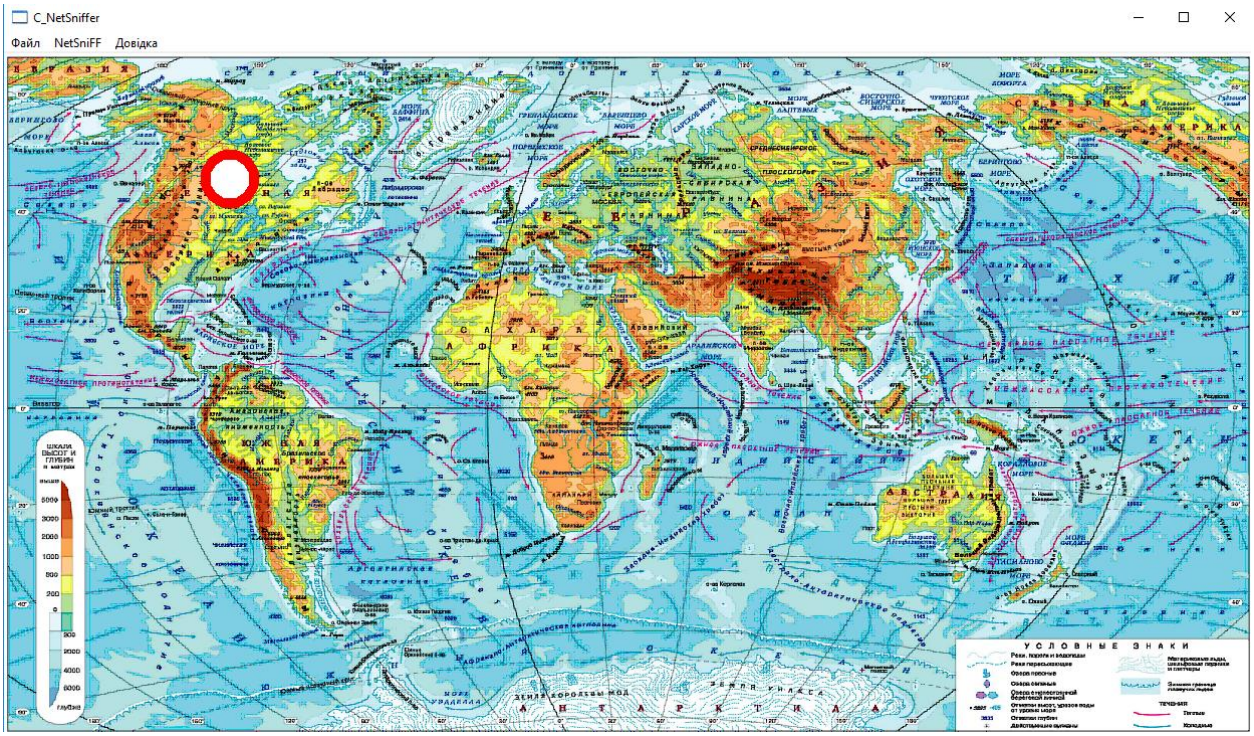


Рисунок 2.5.4 – відображення головного вікна форми

Допоміжного вікна для прослуховування на якому зображені всі дані. Він знаходиться на шляху NetSniFF – Monitor.

Рисунок 2.5.5 – відображення вибору допоміжного вікна

Допоміжне вікно містить такі параметри як:

- Прослуховуємий хост – обраний для прослуховування хост;
- Кількість пакетів –кількість пакетів які будуть прослуховуванні;
- Джерело – відправник пакетів;
- Одержувач – отримувач пакетів;
- Номер пакета – номер прослуханого пакету.

Допоміжне вікно зображено на рисунку 2.5.6.

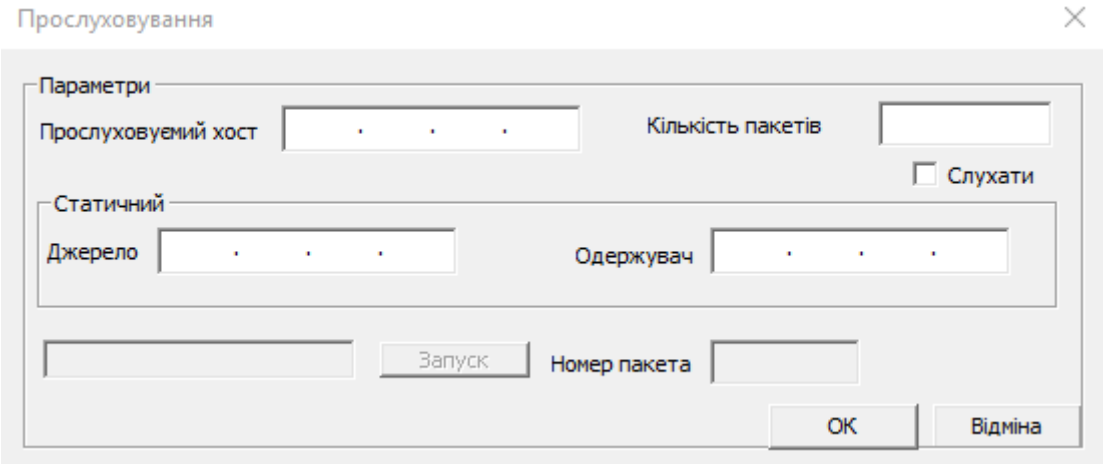


Рисунок 2.5.6 – відображення допоміжного вікна

Після створення всіх меню можна перейти до розробки логіки програми її поведінку те що вона має виконувати.

Код програми пишеться на мові програмування C++ Список класів в середовищі Visual Studio мають наступний вигляд представлений на Рисунку 2.5.7

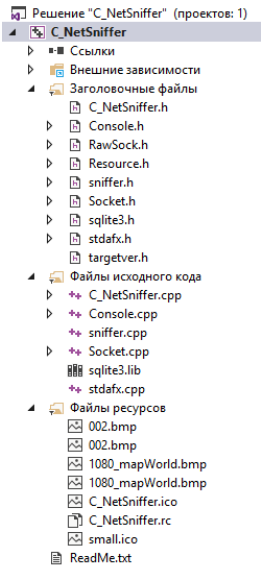


Рисунок 2.5.7 – всіх класів додатку в Visual Studio

Оскільки програма починається з головного вікна але воно використовується для візуального відображення даних, опису створення вікна.

Функція MyRegisterClass() реєструє клас вікна. Код цієї функції.

```
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEX wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style          = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc    = WndProc;
    wcex.cbClsExtra     = 0;
    wcex.cbWndExtra     = 0;
    wcex.hInstance      = hInstance;
    wcex.hIcon          = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_C_NETSNIFFER));
    wcex.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
    wcex.lpszMenuName   = MAKEINTRESOURCE(IDC_C_NETSNIFFER);
    wcex.lpszClassName  = szWindowClass;
    wcex.hIconSm        = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

    return RegisterClassEx(&wcex);
}
```

Функція InitInstance(HINSTANCE, int) зберігає обробку примірника і створює головне вікно. Код цієї функції.

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
   HWND hWnd;

   hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

   hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
      CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

   if (!hWnd)
   {
      return FALSE;
   }

   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);

   return TRUE;
}
```

Допоміжне вікно містить основний функціонал в ньому описано що буде виконувати програма при своєму відкритті чи при натисканні різних кнопок, заповненню текстових полів і т.п. Код цієї функції.

```
264   //Обработчик окна Монитор
265   INT_PTR CALLBACK dMonitor(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
266   {
267       UNREFERENCED_PARAMETER(lParam);
268           // Получаем HWND нашего chechbox'а.
269           HWND hwndCheck = GetDlgItem(hDlg, IDC_CHECK1);
270           HWND hwndButton = GetDlgItem(hDlg, IDC_BUTTON1);
271           //////////
272       switch (message)
273       {
274       case WM_INITDIALOG:
275           hDlg_Monitor = hDlg;
276           SetWindowText(hDlg,_T("Прослуховування"));
277           return (INT_PTR)TRUE;
278
279       case WM_COMMAND:
280           if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
281           {
282               EndDialog(hDlg, LOWORD(wParam));
283               return (INT_PTR)TRUE;
284           }
285
286           /// Обработка нажатия кнопки Старт
287           if (LOWORD(wParam) == IDC_BUTTON1)
288           {
289               SetWindowText(hDlg, _T("Слухаемо"));
290               SetDlgItemText(hDlg,IDC_EDIT1,_T("Слухаемо"));
291               main_old();
292               SendMessage (hwndCheck, BM_SETCHECK,BST_UNCHECKED, 0);
293               EnableWindow(hwndButton, 0);
294               SetDlgItemText(hDlg,IDC_EDIT3,_T(""));
295               return (INT_PTR)TRUE;
296           }
297
298           ///Обработка поля количества пакетов
299           if (LOWORD(wParam) == IDC_EDIT3)
300           {
301               wchar_t text[100];
302
303               LRESULT res = HIWORD(wParam);
304                if(res == EN_CHANGE)
305                {
306                   GetDlgItemTextW(hDlg,IDC_EDIT3, text, 100);
307                   console_out = convertWChar_TArrayToInt(text);
308                   SetDlgItemText(hDlg,IDC_EDIT1,text);
309                }
310
311               return (INT_PTR)TRUE;
312           }
313
314           ///обработка нажатия чекбокса
315           if(LOWORD(wParam) ==(LPARAM)IDC_CHECK1) {
316
317
318            // Выясняем текущее состояние chechbox'а.
319            LRESULT res = SendMessage (hwndCheck, BM_GETCHECK, 0, 0);
320            // Если галочка стоит.
321                if(res == BST_CHECKED)
322                {
323                    EnableWindow(hwndButton, 1);
324                }
325                // Если галочка не стоит.
326                if(res == BST_UNCHECKED)
327                {
328                   SetWindowText(hDlg, _T("Не слухаемо"));
329                   EnableWindow(hwndButton, 0);
330                   SetDlgItemText(hDlg,IDC_EDIT1,_T("Не слухаемо"));
331                }
332
333               return (INT_PTR)TRUE;
334             }
335           break;
336
337       }
338       return (INT_PTR)FALSE;
339   }
```

Функція main_old() це головна функція в якій і відбувається сканування портів та реалізовано весь основний функціонал. Код цієї функції.

```
413  int main_old()
414  {
415   char        buf[64] = "192.168.0.1";
416
417   FILE*       f = NULL;
418   DWORD       packet_res = 100;
419   u_long      watch_host = 0;
420
421
422  #ifdef NET_SERVER_2000
423      InitializeCriticalSection(&critsect);
424  #endif
425
426      // инициализация
427      RS_Init();
428
429      // проверка на успешность создания сокета
430      if (RS_SSocket != SOCKET_ERROR)
431      {
432
433      }
434      else
435      {
436          Sleep(5000);
437          return -1;
438      }
439
440
441      // Включение promiscuous mode
442      RS_SetPromMode(1);
443      SetDlgItemText(hDlg_Monitor,IDC_EDIT2, _T("Змішаний режим ввімкнено."));
444
445
446
447  #ifndef ALPHA
448
449      watch_host = inet_addr(buf);
450
451  #endif
452
453
454      RS_InitStat();
455      SetDlgItemText(hDlg_Monitor,IDC_IP_HOST,convertCharArrayToLPCWSTR(buf));
456      UpdateWindow(hDlg_Monitor);
457      do
458      {
459          IPHeader* hdr = RS_Sniff();
460          // обработка IP-пакета
461          if (hdr)
462          {
463              char *packet_str = 0, *char_src, *char_dest;
464              char *new_packet_str = 0;
465              packets_count++;
466              // штамп времени
467              time(&rawtime);
468              timeinfo = localtime (&rawtime);
469              // пакет в строку
470              //
471              new_packet_str = RS_IPHeaderToStr_new(hdr);
472
473              char_src = nethost2str(hdr->src);
474              char_dest = nethost2str(hdr->dest);
475
476
477              // печать заголовка IP-пакета
478              LPCWSTR  str1,str2;
479              str1 = convertCharArrayToLPCWSTR (char_src);
480              str2 = convertCharArrayToLPCWSTR(char_dest);
481
482              packet_str = RS_IPHeaderToStr(hdr);
483              SetWindowText(hDlg_Monitor,convertCharArrayToLPCWSTR(RS_Hostname));
484
485              LPWSTR str3 = 0;
486              WCHAR wr2[5];
487              wsprintf(wr2, L"%d",packets_count);
488
489              SetDlgItemText(hDlg_Monitor,IDC_IPADDRESS1,str1);
490              SetDlgItemText(hDlg_Monitor,IDC_IPADDRESS2,str2);
491              SetDlgItemText(hDlg_Monitor,IDC_EDIT1, wr2);
492
493              SetDlgItemText(hDlg_Monitor,IDC_STATIC9,str2);
494
495              HDC hdc = GetDC(hDlg_Monitor);
496              TextOut(hdc, 30, 270, str2, 14);
```

```
497                 ReleaseDC(hDlg_Monitor, hdc);
498                 UpdateWindow(hDlg_Monitor);
499                 // для посылки строки заголовка в сеть
500                 sCurrPacket = packet_str;
501                 bNewPacket = true;
502
503                 free((void*)packet_str);
504                 free((void*)hdr);
505
506                 if (packet_res > 0)
507                     Sleep(packet_res);
508                 console_out--;
509             }
510         }while (console_out > 0);
511
512
513
514         // Конец работы
515         RS_Free();
516         packets_count = 0;
517         return 0;
518     }
```

2.6 **Особливості застосування**

Особливостями даного програмного забезпечення можна назвати:

- Візуалізація на мапі світу IP адрес які проходять через скануємий хост;
- Простий та зрозумілий дизайн;
- Можливість за допомогою GeoIP по IP адресі дізнатися країну та місто без необхідності інтернет підключення.

Основною особливістю мого програмного продукту порівняно з іншими є можливість візуалізації IP адрес які проходять через скануємий хост та показує їх на мапі світу. Дана програма може бути застосована мережевими адміністраторами для діагностики уразливості своєї мережі і попередження її злому.

GeoIP це проект визначення географічного положення по IP адресу. Проект являє собою набір баз даних і інструментів роботи до ними. Його використання має головну перевагу в порівнянні з різними API - швидкість. Так само не залежить від доступності зовнішнього сервісу. Невеликий мінус це те, що базу даних розширення (у вигляді окремого файлу) бажано періодично оновлювати, особливо якщо потрібна інформація не тільки по країнах але і по містах.

3. ТЕСТУВАННЯ

Тестування програмного забезпечення - перевірка відповідності між реальним і очікуваним поведінкою програми, що здійснюється на кінцевому наборі тестів, обраному певним чином. У більш широкому сенсі, тестування - це одна з технік контролю якості, що включає в себе активності з планування робіт, проектування тестів, виконання тестування і аналізу отриманих результатів.

Тестування здійснюється за рахунок моделювання різних ситуацій, спираючись на спеціальні методи:

Чорний ящик - "Black box"

Нам невідома внутрішня структура коду і не маємо доступу до бази даних.

Процес дослідження ПО на баги здійснюється шляхом генерації тестових випадків виходячи з аналізу функціональної специфікації (документ, який описує бажані характеристики системи і підсумковий результат) або певних елементів системи.

Плюси "Black box":

- можливість визначення помилок, які не покриває метод "White box". Наприклад, при розробці було упущено якусь функціональність. Код працює добре, а ось відсутній елемент в системі, таким чином, щоб вважається вагомим багом.
- аналіз ПО на наявність дефектів здійснюється як би на призначеному для користувача рівні (з позиції користувача) без поглиблення у внутрішню структуру системи.
- потрібно значно менше часу, завдяки тому, що тест-кейси можна складати відразу після складання специфікації.

Мінуси:

- пропуск моментів, які не прописані в специфіці, але присутні в функціональності коду (саме цей факт спонукає тестувальників використовувати "білий ящик").
- невелике охоплення - тестування піддаються лише кілька вступних значень.

- При нечіткої специфікації можлива труднощі в складанні тест-кейсів.

Отже, Метод "Black box" варто використовувати, якщо тобі потрібно знайти баги:

- реалізації функцій ПЗ;
- призначеного для користувача інтерфейсу;
- функціональної специфікації.

Білий ящик - "White box"

Наявність детальної інформації про внутрішню складової системи - структурі коду.

Плюси методу "білого ящика":

- завдяки наявності інформації про внутрішню структуру системи значно легше визначити дані, які допоможуть в ефективності тестування.
- сприяє оптимізації коду.
- можливість усувати дефекти в коді.
- тестувати можна на ранніх етапах розробки, ще без наявності призначеного для користувацького інтерфейсу.

Мінуси методу "білого ящика":

- Потрібен досвід роботи в даній сфері. Новачкові буде складно розібратися з функціональністю коду.

Сірий ящик - "Grey box"

Відомі тільки деякі елементи реалізації ПО.

Варто, зазначити, що даний метод поєднує в собі підходи до тестування "білого і чорного ящиків".

Плюси:

- можливість застосування складних сценаріїв;
- тестувальник комунікує з розробниками, що дозволяє уникнути використання надлишкових тест-кейсів і значно скорочує час реалізації завдання.

Мінуси:

- Обмеження в аналізі тестового покриття і відсутність доступу до вихідного коду.

Для тестування даного програмного забезпечення було обрано метод експерименту тобто розроблене програмне забезпечення тестувалось кількома добровольцями, які користувались програмним забезпеченням на протязі певного часу. Нижче представлені результати тестування.

При запуску програми спочатку відображається головне вікно з мапою світу. Потім в верхньому меню необхідно обрати випадаючий список NetSniFF та в ньому обрати вкладку Monitor.

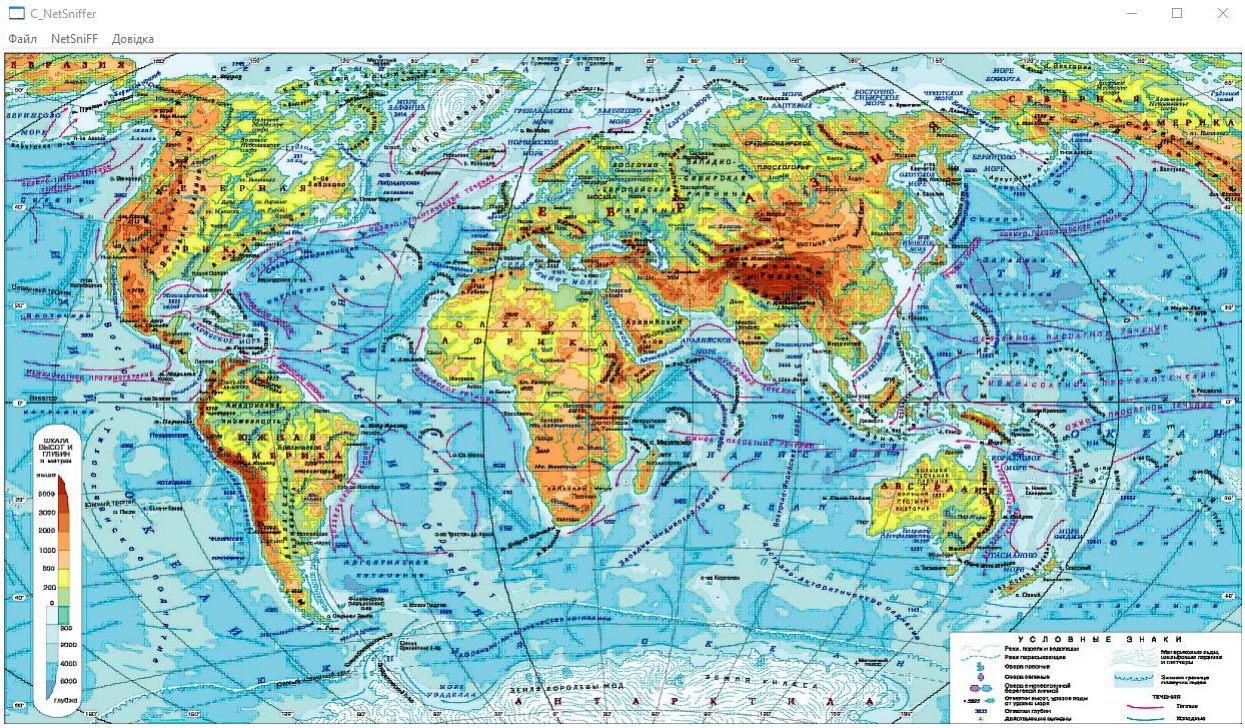


Рисунок 3.1 – головна сторінка з мапою світу

Щоб почати прослуховувати хост необхідно натиснути кнопку

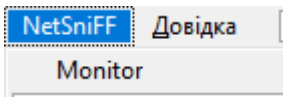


Рисунок 3.2 – вибір допоміжного вікна

В допоміжному вікні необхідно ввести прослуховуємий хост потім ввести кількість пакетів які будуть прослуховувати та поставити галочку слухати та натиснути кнопку запуск після чого програма прослухає вказану кількість пакетів. Якщо не ввести кількість пакетів за замовчення програма прослухає один пакет. А якщо не ввести прослуховуємий хост він буде вибраний за замовченням.

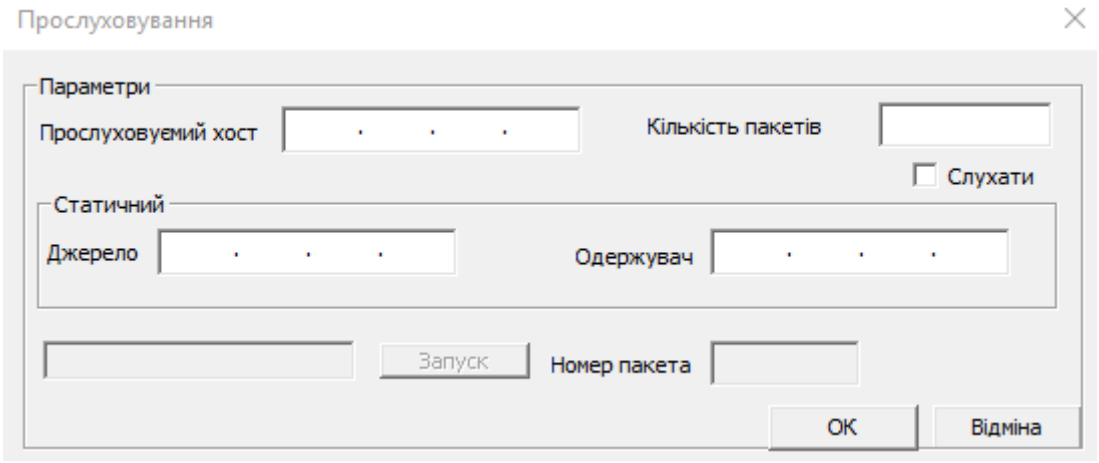


Рисунок 3.3 – допоміжна сторінка

Після введення необхідних даних та натиснення кнопки запуск вікно прийме вигляд представлений на Рисунок 3.4.

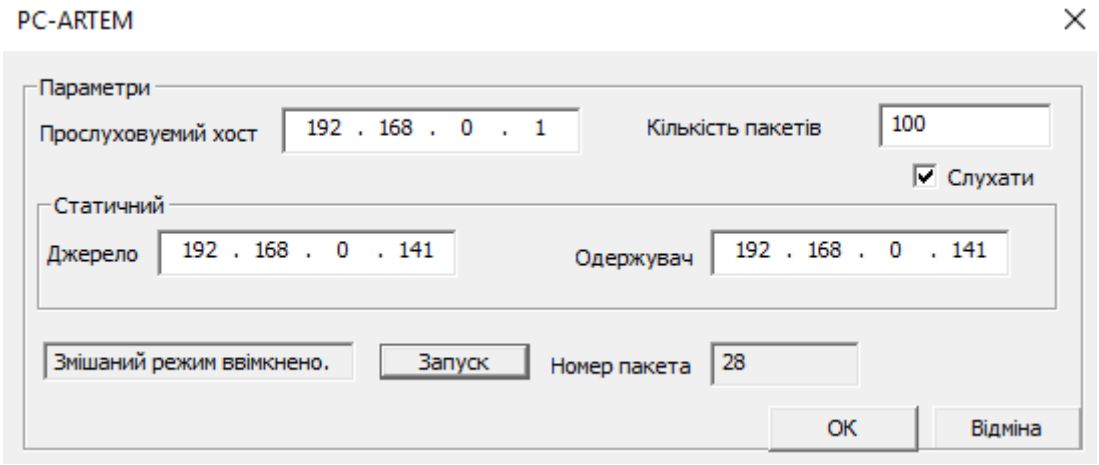


Рисунок 3.4 – допоміжна сторінка під час прослуховування

IP адреси в джерелі і одержувачі вказуються на мапі світу позначається точкою розмір якої залежить від значення параметра точності в базі даних GeoIP визначається розмір точки на мапі світу, як зображено на Рисунок 3.5.

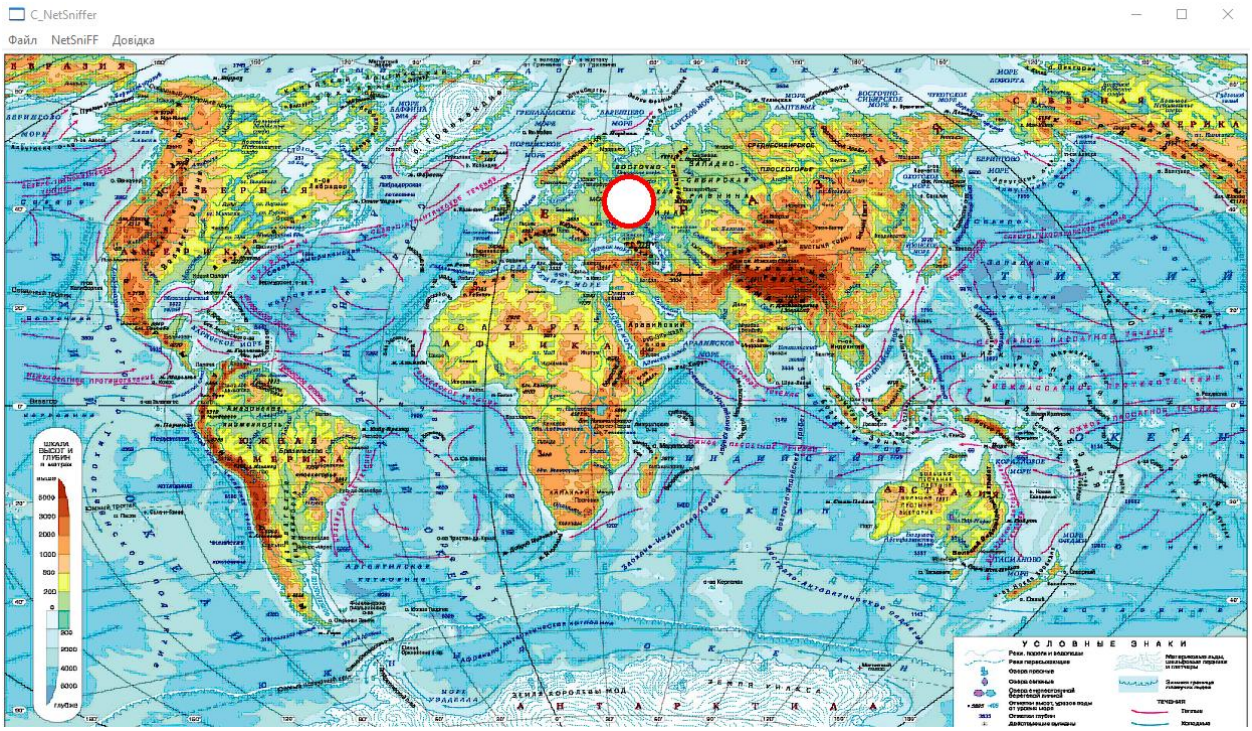


Рисунок 3.5 – головна сторінка з мапою світу

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмний засіб моніторингу завантаженості мережевих портів у розрізі географії запитів.

Було проаналізовано предметну галузь, вибір тематики та було виявлено що розробка програмного забезпечення є доцільною. Розглянуті аналоги програмного забезпечення визначені їх недоліки та переваги. За допомогою отриманих результатів було вирішено, що розробка даного програмного забезпечення доцільна.

Проведено аналіз проблем мережевої безпеки та існуючих аналогів програм моніторингу та встановлено, що є необхідність у створенні безкоштовного програмного засобу для графічного відображення завантаження мережевих портів у розрізі географії запитів, були обрані необхідні засоби для реалізації програмного забезпечення та проведено проектування проекту, програмний продукт було реалізовано за допомогою обраних технологій.

Тестування даного програмного забезпечення було проведено мною та добровольцями, які погодились допомогти мені з тестуванням.

Розроблено функціонал для прослуховування портів та їх відображенню на мапі світу;

Дана робота допомогла закріпити досвід, який було отримано в результаті розробки програмного забезпечення.

Результати отримані в процесі розробки стануть основою для подальшого розвитку програмного забезпечення для моніторингу мережевих портів у розрізі географії запитів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Типы сетевых атак, их описания и средства борьбы [Електронний ресурс] // Режим доступу до ресурсу: http://lagman-join.narod.ru/spy/CNEWS/cisco_attacks.html

2. Сетевые атаки. Виды. Способы борьбы [Електронний ресурс] // Режим доступу до ресурсу: https://moluch.ru/conf/tech/archive/5/1115/

3. Виды сетевых атак [Електронний ресурс] // Режим доступу до ресурсу: http://lib.kstu.kz:8300/tb/books/2016/TSS/Mehtiev%20i%20dr%201/Теория/01.2.htm

4. Защита от сетевых атак [Електронний ресурс] // Режим доступу до ресурсу: https://xserver.a-real.ru/support/useful/zashchita-ot-setevykh-atak/#title_4

5. Различные виды сетевых атак и методы противодействия им [Електронний ресурс] // Режим доступу до ресурсу: https://su27.ru/details_of_internet_attacks

6. Анализ угроз сетевой безопасности [Електронний ресурс] // Режим доступу до ресурсу: http://ypn.ru/138/analysis-of-threats-to-network-security/

7. Специалист по информационной безопасности [Електронний ресурс] // Режим доступу до ресурсу: http://buduguru.org/profession/17

8. ПРЕИМУЩЕСТВА И НЕДОСТАТКИ MAC OS X ПО СРАВНЕНИЮ С WINDOWS [Електронний ресурс] // Режим доступу до ресурсу: https://scienceforum.ru/2014/article/2014006938

9. Преимущества Linux [Електронний ресурс] // Режим доступу до ресурсу: http://heap.altlinux.org/modules/why_linux/index.html

10. Что такое интегрированная среда разработки [Електронний ресурс] // Режим доступу до ресурсу: https://www.kv.by/archive/index2008361105.htm

11. Достоинства и недостатки языка [Електронний ресурс] // Режим доступу до ресурсу: https://studfile.net/preview/5618019/page:2/

12. Введение в C++ [Електронний ресурс] // Режим доступу до ресурсу: https://metanit.com/cpp/tutorial/1.1.php

13. Язык программирования Python [Електронний ресурс] // Режим доступу до ресурсу: https://web-creator.ru/articles/python

14. Python: простое лучше, чем сложное [Електронний ресурс] // Режим доступу до ресурсу: http://suhorukov.com/news_akademy/python-prostoe-luchshe-chem-slozhnoe

15. Что это за программа Java и для чего она нужна [Електронний ресурс] // Режим доступу до ресурсу: https://pcsecrets.ru/internet/chto-eto-za-programma-java-i-dlya-chego-ona-nuzhna.html

16. Достоинства и недостатки языка Java [Електронний ресурс] // Режим доступу до ресурсу: https://javatalks.ru/topics/question/46747

ДОДАТОК А

Лістинг файлів коду

```
// C_NetSniffer.cpp: визначає точку входу для додатка.
//

#include "stdafx.h"
#include "RawSock.h"
#include "Console.h"
#include "Socket.h"
#include "resource.h"
#include "sniffer.h"


using namespace std;

#define MAX_LOADSTRING 100

// Глобальні змінні:
HINSTANCE hInst;                                // поточний екземпляр
TCHAR szTitle[MAX_LOADSTRING];                  // Текст рядка заголовка
TCHAR szWindowClass[MAX_LOADSTRING];            // ім'я класу головного вікна
HWND hDlg_Monitor;
sqlite3 *db = 0;
////////////////////////////////////
CRITICAL_SECTION critsect;
static string sCurrPacket = "";
static bool        bNewPacket = true;
char const Hbr_Name[] = "1080_Map";
////////////////////////////////////
// Надіслати оголошення функцій, включених в цей модуль коду:
ATOM                MyRegisterClass(HINSTANCE hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    dMonitor(HWND, UINT, WPARAM, LPARAM);
int convertWChar_TArrayToInt(wchar_t []);
////////////////////////////////////////////////////////////////
int APIENTRY _tWinMain(HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPTSTR    lpCmdLine,
                       int       nCmdShow)
{
        UNREFERENCED_PARAMETER(hPrevInstance);
        UNREFERENCED_PARAMETER(lpCmdLine);

                MSG msg;
                HACCEL hAccelTable;

        // Ініціалізація глобальних рядків
        LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
```

```
	LoadString(hInstance, IDC_C_NETSNIFFER, szWindowClass, MAX_LOADSTRING);
	MyRegisterClass(hInstance);

	// Виконати ініціалізацію додатка:
	if (!InitInstance (hInstance, nCmdShow))
	{
		return FALSE;
	}

	hAccelTable = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDC_C_NETSNIFFER));

	// Цикл основного повідомлення:
	while (GetMessage(&msg, NULL, 0, 0))
	{
		if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
		{
			TranslateMessage(&msg);
			DispatchMessage(&msg);
		}
	}

	return (int) msg.wParam;
}

//  ФУНКЦІЯ: MyRegisterClass ()
// ПРИЗНАЧЕННЯ: реєструє клас вікна.
//
// КОМЕНТАРІ:
//
// Ця функція і її використання необхідні тільки в разі, якщо потрібно, щоб даний код
// був сумісний з системами Win32, що не мають функції RegisterClassEx '
// яка була додана в Windows 95. Виклик цієї функції важливий для того,
// щоб додаток отримало "якісні" дрібні значки і встановило зв'язок
//    з ними.
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
	WNDCLASSEX wcex;
	wcex.cbSize = sizeof(WNDCLASSEX);
	wcex.style			= CS_HREDRAW | CS_VREDRAW;
	wcex.lpfnWndProc	= WndProc;
	wcex.cbClsExtra		= 0;
	wcex.cbWndExtra		= 0;
	wcex.hInstance		= hInstance;
	wcex.hIcon			= LoadIcon(hInstance,
MAKEINTRESOURCE(IDI_C_NETSNIFFER));
	wcex.hCursor		= LoadCursor(NULL, IDC_ARROW);
	wcex.hbrBackground	= (HBRUSH)(COLOR_WINDOW+1);
	wcex.lpszMenuName	= MAKEINTRESOURCE(IDC_C_NETSNIFFER);
	wcex.lpszClassName	= szWindowClass;
```

```
	wcex.hIconSm		= LoadIcon(wcex.hInstance,
MAKEINTRESOURCE(IDI_SMALL));

	return RegisterClassEx(&wcex);
}

//   ФУНКЦІЯ: InitInstance (HINSTANCE, int)
// ПРИЗНАЧЕННЯ: зберігає обробку примірника і створює головне вікно.
// КОМЕНТАРІ:
// У даній функції дескриптор екземпляра зберігається в глобальній змінній, а також
// створюється і виводиться на екран головне вікно програми.

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
  HWND hWnd;

  hInst = hInstance; // Зберегти дескриптор екземпляра в глобальній змінній

  hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

  if (!hWnd)
  {
    return FALSE;
  }

  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);

  return TRUE;
}

// ФУНКЦІЯ: WndProc (HWND, UINT, WPARAM, LPARAM)
// ПРИЗНАЧЕННЯ: обробляє повідомлення в головному вікні.
// WM_COMMAND - обробка меню програми
// WM_PAINT -Закрасити головне вікно
// WM_DESTROY - ввести повідомлення про вихід і повернутися.

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
	int wmId, wmEvent;
	PAINTSTRUCT ps;
	HDC hdc, hCompatibleDC;
	HANDLE hBitmap, hOldBitmap;
   RECT Rect;
   BITMAP Bitmap;
	char chFileNameDB[] = "geoip";
	int result;
	switch (message)
	{
```

```
        case WM_CREATE:
                HMODULE hDll;
                //Підключаємо базу даних (бібліотеку) sqlite3.dll

                result = sqlite3_open(chFileNameDB,&db);
                break;
        case WM_COMMAND:
                wmId    = LOWORD(wParam);
                wmEvent = HIWORD(wParam);
                // Разобрать выбор в меню:
                switch (wmId)
                {
                case IDM_ABOUT:
                        DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                        break;
                case IDM_EXIT:
                        DestroyWindow(hWnd);
                        break;
                case ID_NETSNIFF_START:
                        main_old();
                        break;
                case ID_NETSNIFF_J:
                        DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG_MONITOR), hWnd,
dMonitor);
                        break;
                default:
                        return DefWindowProc(hWnd, message, wParam, lParam);
                }
                break;
        case WM_PAINT:

                // Отримуємо індекс контекста пристрою
    hdc = BeginPaint(hWnd, &ps);

                hBitmap = (HBITMAP)LoadImage(GetModuleHandle(NULL),
MAKEINTRESOURCE(IDB_BITMAP3), IMAGE_BITMAP, 0, 0, LR_DEFAULTCOLOR);
                GetObject(hBitmap, sizeof(BITMAP), &Bitmap);
      hCompatibleDC = CreateCompatibleDC(hdc);
      hOldBitmap = SelectObject(hCompatibleDC, hBitmap);
      GetClientRect(hWnd, &Rect);
      StretchBlt(hdc, 0, 0, Rect.right, Rect.bottom, hCompatibleDC, 0, 0,
Bitmap.bmWidth,Bitmap.bmHeight, SRCCOPY);
      SelectObject(hCompatibleDC, hOldBitmap);
      DeleteObject(hBitmap);
      DeleteDC(hCompatibleDC);

                HPEN hPenDot;
                hPenDot = CreatePen(PS_SOLID, 5, RGB(255,0,0));
                SelectObject(hdc, hPenDot);
                Ellipse(hdc, Rect.left +600, Rect.top +125, Rect.left +650, Rect.top +175);
                ValidateRect(hWnd, NULL);
                //Віддаємо індекс контексту пристрою
```

```
            EndPaint(hWnd, &ps);

            break;
        case WM_DESTROY:
            sqlite3_close(db);
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}

// Оброблювач повідомлень для вікна "Про програму".
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
        UNREFERENCED_PARAMETER(lParam);


        switch (message)
        {
        case WM_INITDIALOG:
            //Вставимо фото автора
            HBITMAP bmp;
            bmp = (HBITMAP)LoadImage(GetModuleHandle(NULL),
MAKEINTRESOURCE(IDB_BITMAP1), IMAGE_BITMAP, 0, 0, LR_DEFAULTCOLOR);
            SendDlgItemMessage(hDlg, IDC_STATIC10, STM_SETIMAGE, IMAGE_BITMAP,
(LPARAM)bmp);

            return (INT_PTR)TRUE;

        case WM_COMMAND:
            if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
            {
                EndDialog(hDlg, LOWORD(wParam));
                return (INT_PTR)TRUE;
            }
            if(LOWORD(wParam) ==(LPARAM)IDC_BUTTON1) {
      //обробка натискання кнопки
                DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG_MONITOR), hDlg,
About);
                return (INT_PTR)TRUE;
              }
            break;

        }
        return (INT_PTR)FALSE;
}

//Оброблювач вікна Монітор
INT_PTR CALLBACK dMonitor(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
```

```
{
	UNREFERENCED_PARAMETER(lParam);
		// Отримуємо HWND нашого chechbox'a.
		HWND hwndCheck = GetDlgItem(hDlg, IDC_CHECK1);
		HWND hwndButton = GetDlgItem(hDlg, IDC_BUTTON1);
		//////////
	switch (message)
	{
	case WM_INITDIALOG:
		hDlg_Monitor = hDlg;
		SetWindowText(hDlg,_T("Прослуховування"));
		return (INT_PTR)TRUE;

	case WM_COMMAND:
		if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
		{
			EndDialog(hDlg, LOWORD(wParam));
			return (INT_PTR)TRUE;
		}

		/// Обробка натискання кнопки Старт
		if (LOWORD(wParam) == IDC_BUTTON1)
		{
			SetWindowText(hDlg, _T("Слухаемо"));
			SetDlgItemText(hDlg,IDC_EDIT1,_T("Слухаемо"));
			main_old();
			SendMessage (hwndCheck, BM_SETCHECK,BST_UNCHECKED, 0);
			EnableWindow(hwndButton, 0);
			SetDlgItemText(hDlg,IDC_EDIT3,_T(""));
			return (INT_PTR)TRUE;
		}

		///Обробка поля кількості пакетів
		if (LOWORD(wParam) == IDC_EDIT3)
		{
			wchar_t text[100];

			LRESULT res = HIWORD(wParam);
			 if(res == EN_CHANGE)
			 {
				GetDlgItemTextW(hDlg,IDC_EDIT3, text, 100);
				console_out = convertWChar_TArrayToInt(text);
				SetDlgItemText(hDlg,IDC_EDIT1,text);
			 }

			return (INT_PTR)TRUE;
		}

		///обробка натискання чекбокса
		if(LOWORD(wParam) ==(LPARAM)IDC_CHECK1) {
```

```
	// З'ясовуємо поточний стан chechbox'a.
	LRESULT res = SendMessage (hwndCheck, BM_GETCHECK, 0, 0);
	// Якщо галочка стоїть.
				if(res == BST_CHECKED)
				{
					EnableWindow(hwndButton, 1);
				}
				// Якщо галочка не стоїть.
				if(res == BST_UNCHECKED)
				{
					SetWindowText(hDlg, _T("Не слухаемо"));
					EnableWindow(hwndButton, 0);
					SetDlgItemText(hDlg,IDC_EDIT1,_T("Не слухаемо"));
				}

				return (INT_PTR)TRUE;
			}
			break;

	}
	return (INT_PTR)FALSE;
}

////////////////////////////////////////////////////////////////////////////

unsigned __stdcall Connection(void* a)
{
	Socket* s = (Socket*) a;
	if (a == NULL)
		return SOCKET_ERROR;
	while (1)
	{
		if (bNewPacket)
		{
			EnterCriticalSection(&critsect);

			s->SendLine(sCurrPacket+(char)13+(char)10);
			bNewPacket = false;

			LeaveCriticalSection(&critsect);
		}
	}

	delete s;
	return 0;
}

unsigned __stdcall AcceptConnections(void* a)
{
	SocketServer in(2000, 20);

	while (1)
```

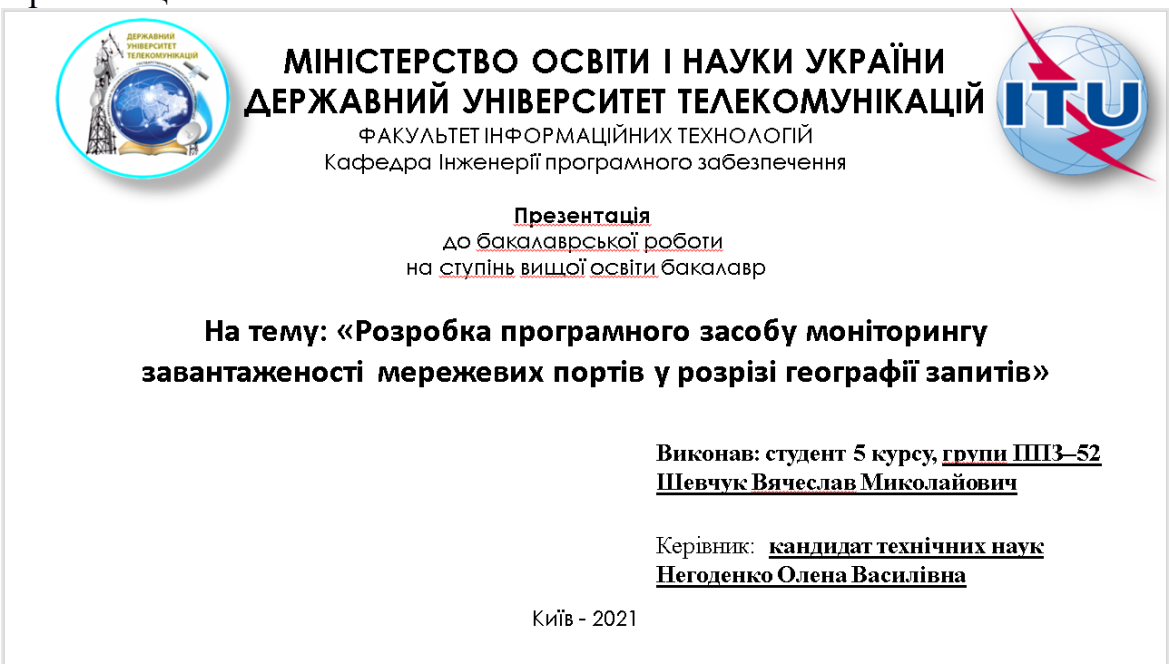
```
        {
                Socket* s = in.Accept();

                unsigned ret;
                _beginthreadex(0, 0, Connection, (void*)s, 0, &ret);
        }
}

wchar_t *convertCharArrayToLPCWSTR(const char* charArray)
{
   wchar_t* wString=new wchar_t[16];
   MultiByteToWideChar(CP_ACP, 0, charArray, -1, wString, 4096);
   return wString;
}

int miniPow(int a, int b)    // power with small number -
{
   int r = 1;
   for (int i = 0; i < b; i++)
   {
      r *= a;
   }
   return r;
}

int convertWChar_TArrayToInt(wchar_t text[])
{
   int i = 0;
   int len = wcslen(text);
   int coefficient = 0;
   int RawNumber = 0;
   int Number = 0;

   for (int k = 0; k < len; k++)
   {
      coefficient = miniPow(10, len - 1 - k);
      RawNumber = (int)text[k];
      Number = RawNumber - 48;
      i += coefficient * Number;
   }
   return i;
}

int main_old()
{
 char            buf[64] = "192.168.0.1";

 FILE*           f = NULL;
 DWORD                packet_res = 100;
 u_long               watch_host = 0;
```

```
#ifdef NET_SERVER_2000
	InitializeCriticalSection(&critsect);
#endif

	// ініціалізація
	RS_Init();

	// перевірка на успішність створення сокета
	if (RS_SSocket != SOCKET_ERROR)
	{

	}
	else
	{
		Sleep(5000);
		return -1;
	}


	// включення promiscuous mode
	RS_SetPromMode(1);
	SetDlgItemText(hDlg_Monitor,IDC_EDIT2, _T("Змішаний режим ввімкнено."));



#ifndef ALPHA

	watch_host = inet_addr(buf);

#endif


	RS_InitStat();
	SetDlgItemText(hDlg_Monitor,IDC_IP_HOST,convertCharArrayToLPCWSTR(buf));
	UpdateWindow(hDlg_Monitor);
	do
	{
		IPHeader* hdr = RS_Sniff();
		// обробка IP-пакету
		if (hdr)
		{
			char *packet_str = 0, *char_src, *char_dest;
			char *new_packet_str = 0;
			packets_count++;
			// штамп часу
			time(&rawtime);
			timeinfo = localtime (&rawtime);
			// пакет в рядок
			//
			new_packet_str = RS_IPHeaderToStr_new(hdr);

			char_src = nethost2str(hdr->src);
```

```
                    char_dest = nethost2str(hdr->dest);


                    // друк заголовка IP-пакета
                    LPCWSTR  str1,str2;
                    str1 = convertCharArrayToLPCWSTR (char_src);
                    str2 = convertCharArrayToLPCWSTR(char_dest);

                    packet_str = RS_IPHeaderToStr(hdr);

SetWindowText(hDlg_Monitor,convertCharArrayToLPCWSTR(RS_Hostname));

                    LPWSTR str3 = 0;
                    WCHAR wr2[5];
                    wsprintf(wr2, L"%d",packets_count);

                    SetDlgItemText(hDlg_Monitor,IDC_IPADDRESS1,str1);
                    SetDlgItemText(hDlg_Monitor,IDC_IPADDRESS2,str2);
                    SetDlgItemText(hDlg_Monitor,IDC_EDIT1, wr2);

                    SetDlgItemText(hDlg_Monitor,IDC_STATIC9,str2);

                    HDC hdc = GetDC(hDlg_Monitor);
                    TextOut(hdc, 30, 270, str2, 14);
                    ReleaseDC(hDlg_Monitor, hdc);
                    UpdateWindow(hDlg_Monitor);
                    // для посилки рядку заголовка в мережу
                    sCurrPacket = packet_str;
                    bNewPacket = true;

                    free((void*)packet_str);
                    free((void*)hdr);

                    if (packet_res > 0)
                            Sleep(packet_res);
                    console_out--;
            }
        }while (console_out > 0);



        // Кінець роботи
        RS_Free();
        packets_count = 0;
        return 0;
}
```

ДОДАТОК Б

Презентація



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Інженерії програмного забезпечення

Презентація
до бакалаврської роботи
на ступінь вищої освіти бакалавр

На тему: «Розробка програмного засобу моніторингу завантаженості мережевих портів у розрізі географії запитів»

Виконав: студент 5 курсу, групи ШІЗ–52
Шевчук Вячеслав Миколайович

Керівник: **кандидат технічних наук**
Негоденко Олена Василівна

Київ - 2021

Актуальність роботи

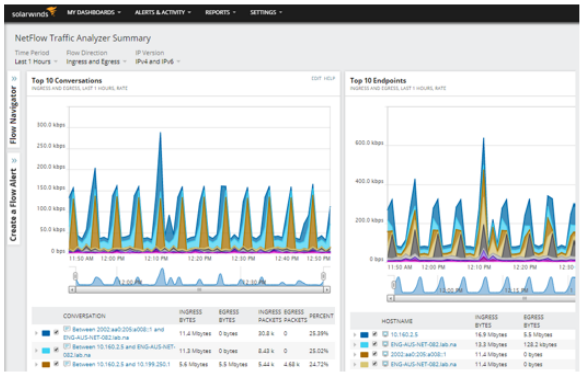
Зараз майже всі підприємства мають сайт на якому можна ознайомитись та замовити послуги, які надає дане підприємство, або працюють на віддалених робочих столах. Але підприємства, як правило працюють на певну категорію споживачів, які проживають певній території, або робітники підприємства підключаються до робочих столів з певної території. Існує не велика кількість програм, яка надає можливість відслідковувати звідки відбувається підключення до мережевого обладнання саме тому моя дипломна робота зараз актуальна.

Мета роботи

- *Об'єкт дослідження* – моніторинг завантаженості мережевих портів.
- *Предмет дослідження* – мережеві порти у розрізі географії запитів
- *Мета роботи* – створення безкоштовного програмного забезпечення для моніторингу завантаженості мережевих портів.

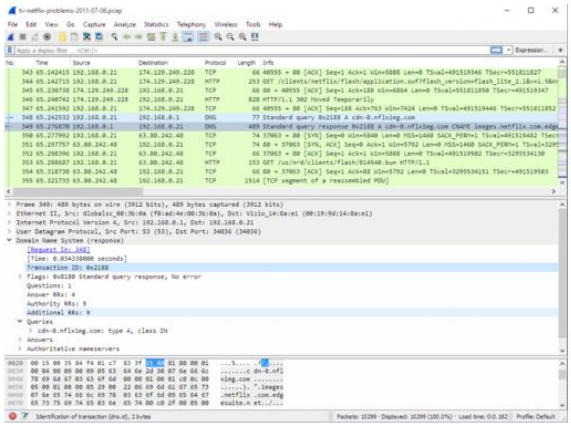
Аналіз аналогів

- SolarWinds Network Bandwidth Analyzer – це програмний пакет з двох продуктів Network Performance Monitor здійснює моніторинг продуктивності мережі. NetFlow Traffic Analyzer більше сконцентровано на аналізі самого трафіку.



Аналіз аналогів

- Wireshark – це програма для аналізу мережевих пакетів Ethernet і інших мереж з вільним вихідним кодом. Має графічний інтерфейс користувача.



Виявлені проблеми

- Тільки англомовний інтерфейс.
- Програмні рішення мають пробний період, після чого становляться платними, або програми безкоштовні, але виводять тільки цифрові дані, без візуалізації.
- В даних, які виводять програми не показують звідки йдуть запити до мережевих портів.
- Розглянуті програми нагромаджені інтерфейсом та можливостями

Середовище розробки

Microsoft Visual Studio

- Єдині засоби API для розробки програм на різних мовах.
- Простота стикування різномовних модулів.
- Багато готових до вживання класів, що реалізують різні алгоритми.
- Установка програм під .NET не вимагає програм-інсталяторів.

- Помітно знижується швидкість роботи програм.
- Потрібно більше оперативної пам'яті.

Qt Creator

- Легкість. Це ПО є гнучким і простим у використанні.
- Настроюється UX. Це ПО дозволяє налаштовувати і створювати сучасний інтерфейс.
- Активно розвивається.

- Бібліотека рідко оновлюється.
- Розташування пошукової системи. Користувач витрачає багато часу на пошук даного інструменту.
- Відключення. Користувачі скаржаться на несподіване завершення роботи.

Мова програмування

- C ++ - статично типізований мова програмування загального призначення. Підтримує такі парадигми програмування як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, забезпечує модульність, роздільну компіляцію, обробку винятків, абстракцію даних, оголошення типів (класів) об'єктів, віртуальні функції.

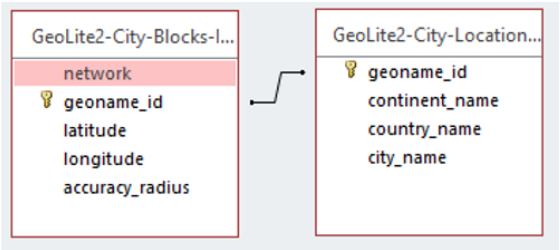
СУБД

- SQLite - це вбудована бібліотека, яка реалізує автономний, безсерверний, нульової конфігурації, транзакційний механізм СУБД SQL. Це база даних, яка налаштована на нуль, що означає, як і інші бази даних, її не потрібно налаштовувати в вашій системі.
- SQLite не є автономним процесом, як інші бази даних, можна пов'язати його статично або динамічно відповідно до вашим вимогою з вашим додатком. SQLite безпосередньо звертається до своїх файлів зберігання.

База даних

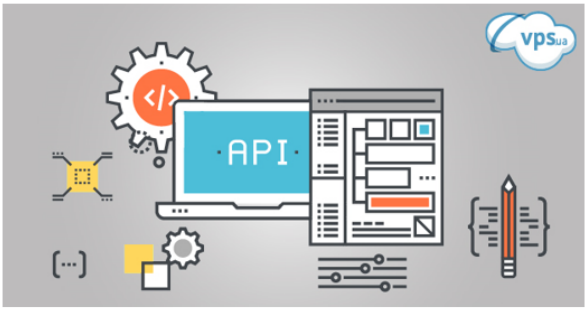
- MaxMind GeoIP2 визначають місцеположення та інші характеристики користувачів Інтернету для широкого спектру застосувань, включаючи персоналізацію вмісту, виявлення шахрайства, націлювання оголошень, аналіз трафіку, відповідність, гео-націлювання, гео-огородження та управління цифровими правами.

База даних

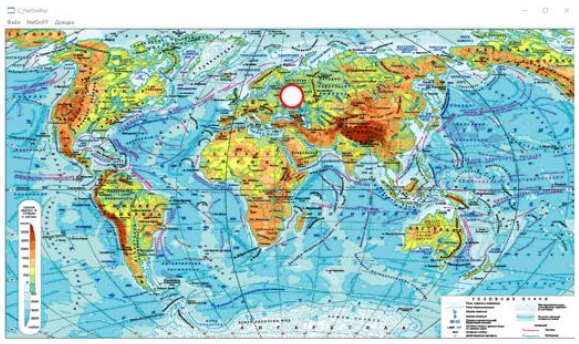


Інтерфейс

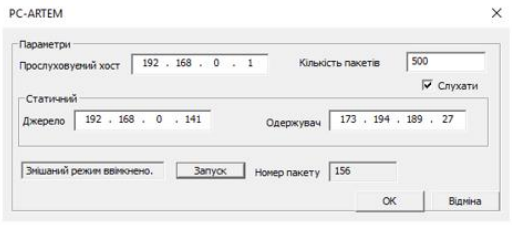
- Windows API (англ. Application programming interfaces) - загальна назва набору базових функцій інтерфейсів програмування додатків операційних систем сімейств Microsoft Windows корпорації «Майкрософт». Надає прямий спосіб взаємодії додатків користувача з операційною системою Windows.



Зовнішній вигляд програмного засобу



Головне вікно



Допоміжне вікно

Висновки

У даній бакалаврській роботі було розроблено програмний засіб для моніторингу завантаженості мережевих портів у розрізі географії запитів та візуалізацією даних на мапі світу. Був проведено аналіз предметної галузі, обґрунтована тематика, та було вирішено, що розробка програмного засобу є доцільною. Були розглянуті основні аналоги програмного забезпечення, визначені їх недоліки та переваги. На основі отриманих результатів було прийнято рішення, що розробка власного програмного засобу, який би вирішував проблеми, що присутні в аналогах.