Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи управління мережею Wi-Fi та аналізу даних”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Мірошніченко А.С.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Мірошніченку Артему Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи управління мережею Wi-Fi та аналізу даних*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 48-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи управління мережею Wi-Fi та аналізу даних*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Мірошніченко А.С.
(прізвище та ініціали)

АНОТАЦІЯ

Мірошніченко А.С. Програмне забезпечення системи управління мережею Wi-Fi та аналізу даних. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління мережею Wi-Fi та аналізу даних.

Метою розробки є програмне забезпечення системи управління мережею Wi-Fi та аналізу даних.

Результат роботи – програмна реалізація системи управління мережею Wi-Fi та аналізу даних.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, управління мережею, Wi-Fi, аналіз даних

ABSTRACT

Miroshnichenko A.S. Software for Wi-Fi network management and data analysis. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a Wi-Fi network management and data analysis system.

The purpose of the development is software for a Wi-Fi network management and data analysis system.

The result of the work is a software implementation of a Wi-Fi network management and data analysis system.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program is developed in the Python environment.

Keywords: computer engineering, network management, Wi-Fi, data analysis

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електрона обчислювальна машина
КВ	–	коефіцієнт варіації
КЗ	–	канал зв'язку
НСД	–	несанкціонований доступ
ПС	–	програмна середа
СВВ	–	система виявлення вторгнень
СеМО	–	експонентна мережа масового обслуговування
СМО	–	система масового обслуговування
СПД	–	система передачі даних

ВСТУП

Актуальність теми. Мережа Wi-Fi стає обов'язковою для все більшого числа об'єктів різного призначення: торгових центрів, виставочних комплексів, об'єктів транспортної інфраструктури, медичних установ, стадіонів і т.д. Такі мережі забезпечують не тільки ефективний зв'язок і передачу інформації. Зібрані з їх допомогою аналітичні дані дозволяють підвищити ефективність використання площ, поліпшити бізнес-показники, підняти на більш високий рівень комфорт і безпеку людей, що перебувають на об'єктах. Розглянемо використання Wi-Fi на прикладі торгових центрів.

Традиційні магазини, що використовують фізичні площадки для взаємодії з покупцями, випробовують усе більше гостру конкуренцію з боку інтернет-магазинів, які активно застосовують самі передові інформаційні й цифрові технології. Щоб не програти, традиційним магазинам необхідно також максимально активно задіяти сучасні технології, які допомагають ефективно обслуговувати й утримувати покупців. Одним із ключових напрямків «цифрової трансформації» є підтримка мобільних пристроїв, з якими покупці приходять у магазин. Серед типових даних, видаваних системою управління мережею Wi-Fi та аналізу даних є наступні:

- загальне число покупців, що зайшли в той або інший магазин;
- відношення числа нових покупців до числа постійних клієнтів;
- час, проведений покупцями в магазині;
- тип використовуваних ними мобільних пристроїв;
- демографічні дані.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи управління мережею Wi-Fi та аналізу даних.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем управління мережею Wi-Fi та аналізу даних.

– Дослідження системи управління мережею Wi-Fi та аналізу даних.

– Програмна реалізація системи управління мережею Wi-Fi та аналізу даних.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі управління мережею Wi-Fi та аналізу даних.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління мережею Wi-Fi та аналізу даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Оскільки мережі Wi-Fi поширюються з промислових районів на житлові, потреба в надійній безпеці зросла. Розвиток інтелектуальних мереж, особливо в IoT, створив проблеми безпеки та вразливості даних. Унікальний метод, який використовує машинне навчання для виявлення аномалій і можливих порушень безпеки в мережах Wi-Fi, усуває ці труднощі. Ми збираємо, попередньо обробляємо та аналізуємо мережеві дані, щоб створити повний набір даних. Цей набір даних тренує алгоритми машинного навчання для виявлення та класифікації мережевих аномалій. Використовуючи гнучкі методи, аналіз даних і алгоритми машинного навчання, ми створили систему виявлення вторгнень у мережу Wi-Fi (WNIDS), яка може виявляти різноманітні мережеві загрози. Запропонований WNIDS містить два пов’язані етапи з конкретними моделями машинного навчання. Ці алгоритми точно класифікують мережеві дані як звичайні або специфічні для атаки. Наша технологія захищає від зловмисних атак і забезпечує надійну мережу Wi-Fi для користувачів у різних доменах за допомогою машинного навчання. Сучасні загрози безпеці мережі були повністю зрозумілі завдяки опитуванням і аналізу даних. WNIDS було впроваджено та розгорнуто через структурований життєвий цикл розробки системи. Цей інструмент усуває слабкі місця мережі та сприяє розвитку віддалених підприємств, пропонуючи безпечний і плавний доступ споживачам у всьому світі.

1.2 Область застосування

Розібравшись із призначенням систем глибокого аналізу трафіку (DPI), з тим, як далеко вона може забратися при аналізі минаючих через неї пакетів і як точно визначає застосунки, які не використовують заздалегідь відомі заголовки при обміні даними, подивимося, які реальні варіанти використання (use case) застосовні.

От 9 основних сценаріїв, які реалізуються на практиці:

1. Аналіз і класифікація трафіку. Моніторинг і тренди

Мережний трафік – це не ріка, що несе свої води коричневого кольору по каналах оператора, а веселка, тільки квітів у ній сотні й тисячі. Кожний колір показує протокол, додаток, абонента, розташування – величезний спектр інформації.

Будь-які втручання в трафік починаються з його аналізу, і DPI надає самі широкі можливості для цього, перевершуючи по функціоналі такі рішення, як NBAR від Cisco.

2. Пріоритизація трафіку

Незамінна функція керування трафіком для боротьби з «ненажерливими» протоколами й забезпечення високої швидкості роботи абонента в Інтернеті.

Самим популярним прикладом необхідності розподілу пріоритетів трафіку є p2 p-протокол (Torrent). Коли користувач запускає завантаження по даному протоколі, знижується якість всіх інших (voip, http, video і ін.), це пояснює повільну швидкість відкривання сторінок, лаги відео, погана якість голосу при дзвінках. Рішення цієї проблеми простої: робимо пріоритет трафіку p2p нижче інших, обмежуємо йому смугу пропущення й всі інші застосунки починають працювати швидко, як звичайно.

Ще одним простим прикладом необхідності пріоритизаціи є онлайн-кінотеатри, що набирають популярність. При збільшенні якості відео 720p – 1080p – 4K росте навантаження на канал оператора. В Австралії з появою сервісу

відео Netflix інтернет-трафік за 6 тижнів збільшився на 25 %. Активне керування трафіком і гнучке настроювання пріоритетів дозволяє, небагато знизивши якість відео в певний момент, забезпечити гарну якість таких чутливих сервісів, як, наприклад, VOIP. Якщо абонент хоче дивитися відео в максимальній якості, для нього персонально можна застосувати спеціальний тариф за додаткову плату, а отриманий прибуток використовувати для розвитку операторської мережі.

3. Оптимізація аплінків

Аплінки – канал оператора, той ресурс, що становить левову частину витрат оператора і є обмежником для надання абонентові найвищих швидкостей.

Статистично 50 % такого вечірнього трафіку – це Torrent, 20 % – відео, 30 % – все інше. Якщо виділити Torrent-трафік і понизити його пріоритет, то всі інші протоколи будуть працювати як і в будь-який інший час без необхідності збільшення загальної пропускної здатності каналу. Цей механізм починає працювати тільки в періоди збільшення навантаження (недостачі швидкості) і діє рівномірно на всіх абонентів. Клієнт не зауважує погіршення якості надання послуги, а оператор не несе додаткових витрат.

4. Розподіл каналу між абонентами

Можна дозволити користувачеві докуповувати опції підвищення швидкості або доступу до певних ресурсів без обмежень (безліміт-абонемент на Skype, доступ до відео на максимальній швидкості, необмежений доступ до соціальних мереж і т.п.). Гнучкість настроювань таких тарифів безмежна.

Мобільні оператори за допомогою системи глибокий аналіз трафіку можуть контролювати завантаження кожної базової станції окремо, розподіляючи навантаження рівномірно по всім, щоб абонент не відчував погіршення якості.

5. Кешування

Кеш-сервер працює у зв'язуванні із системою DPI і не вимагає забезпечення режиму проксірування.

По статистиці, 3 Тб сховище, задіяне під кешування Youtube контенту, для 100 000 абонентів дає зниження зовнішньої Youtube смуги на 30 %, а виходить, на 30 % збільшується швидкість завантаження іншого трафіку.

6. Поведінкова оцінка абонентів

Кожний користувач Мережі унікальний. Він виходить в Інтернет у певний час, користується одним із браузерів, ходить по цікавим для нього сайтам, дивиться комедії або жахи, сидить в «Фейсбуці користується торрентами або онлайн-кінотеатрами.

Якщо користувачі почав відвідувати сайти конкурентів оператора, можна запропонувати йому тариф дешевше.

У руках досвідченого маркетолога зібрана інформація може стати інструментом для збільшення числа абонентів і доходу компанії.

7. Повідомлення абонентів

Функція, що дозволяє операторові передавати повідомлення абонентові під час роботи в Інтернеті.

8. Заборона ресурсів (білий і чорний списки)

Закон жадає від провайдерів стежити за реєстрами заборонених веб-ресурсів і вчасно їх блокувати. Щоб виконувати ці вимоги, оператор повинен постійно перевіряти актуальні записи в реєстрі й фільтрувати трафік із заборонених сайтів.

9. Захист, перехоплення трафіку, передфільтр СОРМ

Тому що DPI пропускає через себе й фільтрує весь трафік, захист абонентів і обчислювальних систем у хмарі стає для неї однієї з безпосередніх завдань.

DoS- (Denial of Service – відмова в обслуговуванні) і DDoS-атаки (Distributed Denial of Service – розподілена атака типу «відмова в обслуговуванні») приводять до переповнення орендованої клієнтом смуги (забивання трафіком). Атакуючий звичайно намагається замаскувати свій IP-адреса, щоб його неможливо було заблокувати, або атака виробляється з великої

кількості комп'ютерів одночасно з організованої заздалегідь мережі ботів. Несподіване падіння мережного ресурсу внаслідок завантаження смуги пропущення – це фінансові втрати для його власника, невиконання умов надання безвідмовного доступу для хостера, погіршення репутації оператора.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління мережею Wi-Fi та аналізу даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Глобальний ринок систем глибокого аналізу трафіку (DPI) в 2024 році оцінювався в 741,7 мільйон доларів США, і, за прогнозом аналітиків, до 2027 виросте більш ніж в 6 разів і складе 4 711,3 мільйони. Даний метод в основному застосовують інтернет-провайдери й оператори зв'язку, які прагнуть захистити своїх абонентів від вірусних атак шляхом сканування трафіку, що передається через мережі. Також ця технологія допомагає поліпшити якість обслуговування (QoS) установлюючи пріоритети переданих даних для зниження навантаження на мережу.

Ріст ринку DPI обумовлений збільшенням числа використовуваних мобільних пристроїв (смартфонів) у сполученні з постійно зростаючим попитом на широкополосну передачу даних, на відміну від голосового зв'язку, доходи від якої постійно падають. За останні 5 років в Україні частка користувачів, що користуються інтернетом хоча б раз у добу виросла з 31% до 57% (66,5 млн чоловік). При цьому частка мобільних користувачів в останні роки сильно росте, так якщо наприкінці 2024 року вона становила близько 15%, те до кінця 2014 уже майже 25%. 29% всіх візитів на веб-сайти відбувається з мобільних пристроїв. Майже три чверті цих візитів доводиться на смартфони, і їхня частка продовжує рости.

Користувачі швидко звикають до більше високих швидкостей, змушують операторів уводити нові більше швидкі тарифи. Якщо кілька років назад швидкості технології EDGE були достатні й комфортні, то зараз значок «E» на смартфоні сприймається споживачем як відсутність інтернету зовсім. Оператор

обіцяє швидкості в 10 – 40 – 80 Мбіт у секунду, щоб задовольнити сучасні запити клієнта. От тільки умовчує, що завантаження файлів буде в 8 разів повільніше, тому що швидкість зазначена в мегабітах, а розмір файлу вказується в мегабайтах (1 Мбайт = 8 Мбіт, отже, швидкість 10 Мбіт у секунду – це 1,25 Мбайт у секунду. Оператор обіцяє ці швидкості клієнтові, знаючи, що пропускна здатність свого каналу обмежена. Завдання продати 5 000 користувачів по 10 Мбіт/с, що дає в сумі 50 Гбіт/с, якщо свій канал усього 10Гбіт/с вирішує система DPI, оптимізуючи смугу пропущення до клієнта за рахунок фільтрів, пріоритетів і кешу.

Зростаюча урбанізація поряд з підвищенням купівельної спроможності, згідно із прогнозами, викличе ще більший попит на мобільний трафік у порівнянні зі звітним періодом. Щоб упоратися з напливом нових клієнтів, зберігши якість надаваних послуг на високих швидкостях, оператором всі частіше прийде звертатися до технології DPI і нарощувати потужності наявних пристроїв. Ці фактори будуть рушійною силою росту попиту на DPI рішення в найближчі 6 років.

Крім забезпечення достатньої пропускної здатності, DPI дозволяє вирішити багато роблем безпеки при роботі з даними. В Україні інтерес до систем з погляду безпеки, підстьобнуло законодавство, що зажадало від операторів обмежувати доступ абонентів до протиправного контенту в інтернеті. DPI здійснює глибокий аналіз трафіку, забезпечує виконання даних вимог, а також запобігає виконання шкідливих програм, захищає від мережних атак ззовні (DDoS, сканування портів), дозволяє боротися з тероризмом.

Важливо розуміти, що в рішенні поставлених завдань, тільки DPI виконує аналіз всіх минаючих через неї пакетів аж до 7 рівня моделі OSI, а не тільки по стандартних номерах портів. Виконуючи поведінковий аналіз трафіку, вона розпізнає застосунки, що не використовують для обміну даними заздалегідь відомі заголовки й структури даних.

Проте відсутність розуміння принципів роботи даних систем і низький рівень довіри забезпечення конфіденційності користувача сповільнює ріст ринку. Споживач не знає, що використання системи DPI поліпшує якість надаваних йому послуг, але боїться, що доступ до його даних одержать треті особи.

З погляду операторів, навпроти, переваги використання таких технологічних рішень зрозумілі й виправдані, але наявні на ринку промислові DPI-Рішення, як правило, не вписуються за ціною в бюджети середніх і маленьких компаній.

Зараз на ринку присутні трохи вендорів, що робіть рішення DPI які займаються їхньою інтеграцією, це як світові лідери – Procera, Allot, Sandvine, так і розроблювачі – VAS Experts, Протей, Mfisoft, Napalabs. Однак жодна компанія із трійки закордонних лідерів не має офіційного представництва в Україні. Установка й настроювання їхніх рішень здійснюється компаніями інтеграторами і якість даної роботи прямо залежить від кваліфікації й досвіду фахівців інтегратора. Також інтегратор найчастіше виконує функції техпідтримки першої лінії для замовника, а у випадку серйозної проблеми звертається прямо до вендору, що вирішує проблему й віддає назад партнерові-інтегратору, а той замовникові. Такий ланцюжок не сама зручного й швидка для рішення виникаючих проблем. Тим більше що поки на ринку небагато фахівців, які мають достатню компетенцію по DPI, розуміють які завдання за допомогою неї можна вирішувати, базові бізнес-кейси всі знають і розуміють, але застосування стандартних рішень конкурентами позбавляє їхніх переваг друг перед іншому й створюють враження низької ефективності DPI.

Завдання інтегратора – продати готове рішення болванку від вендора, що зовсім не оптимізоване для українського ринку. А настроювання під існуючі реалії, особливості місцевого трафіку й протоколи передачі даних лягають на плечі фахівців оператора, для яких DPI це «чорний ящик» у настроюваннях якого розбиратися можна роками. Усунення 20% основних причин неправильного настроювання системи на етапі впровадження дозволяє поліпшити її

продуктивність на 80%, от тільки знайти ці ключові 20% у стані далеко не кожний фахівець інтегратора або клієнта.

Виробники й платформи

Allot Service Gateway

Allot NetXplorer – це єдиний засіб керування всіма платформами й сервісами Allot. Дозволяє з однієї консолі управляти конфігураціями й політиками виділення ресурсів, становити звіти, здійснювати пошук і усунення несправностей, вести облік якості обслуговування. Простий і наочний інтерфейс дозволяє здійснювати ці маніпуляції швидко й легко.

Procera Networks

Компанія Procera наголошує на своїй програмній платформі PacketLogic, ядром якої є DRDL (Datastream Recognition Definition Language), розроблювальний інженерами компанії протягом 15 років.

Зараз Procera обновила модельний ряд, зробивши його більше гнучким по продуктивності, але забравши найменш продуктивну серію PL5000.

Huawei Technologies Co. Ltd

Компанія Huawei в останні роки активно збільшує свою частку на мережному ринку Україні. Це зв'язано й із санкціями на продукцію американських виробників, і із ціновою політикою китайського вендора.

У даний момент пристрій не одержав широкого поширення серед операторів зв'язку, тому судити про якість і продуктивність рано.

Це короткий огляд платформ DPI. Функції в кожної з них практично ідентичні. Кожна компанія завоювала достатній авторитет на ринку, що й показує статистика з дуже близькими процентними показниками популярності. Говорити про швидкість обробки даних або зручності використання складно.

Пристрої різних виробників працюють на різних апаратних платформах і з різним ПЗ. Характеристики ми брали з офіційних специфікацій, але вони плаваючі, у кожній лінійці є різні по потужності рішення. А розроблювачі найчастіше встановлюють свої програмні комплекси на стандартні сервери, і отут

продуктивність залежить тільки від потужності заліза й числа мережних інтерфейсів.

Кожна платформа підбирається під технічне завдання проекту, що містить такі нюанси, які можуть скоротити вибір до 1-2 рішень. Наприклад, якщо компанія не має права купувати пристрою виробництва США, це скорочує коло вибору до вітчизняних і китайських фірм. Або оператор може вирішити об'єднати в одному пристрої системи DPI і CGNAT – таким вимогам відповідає СКАТ 5.0 від VAS Experts. А якщо мережа побудована на Cisco ASR, тобто можливість реалізувати рішення AVC, не купуючи додаткового встаткування.

Це дає переваги по потужності рішення за рахунок додавання мережних інтерфейсів, пам'яті і ядер процесора. Кожний додатковий модуль для Allot, Procera, Sandvine дуже доріг, до того ж, збільшуючи продуктивність, необхідно розширювати ліцензію.

Однак є й зворотна сторона. Рішення на власних апаратних платформах роблять готовий пристрій більше надійним і стабільним, навіть якщо в основі лежить звичайний x86 процесор. Зрівняти можна з комп'ютерами iMac від Apple з macOS і звичайним PC на Windows. Закрита платформа й оптимізація операційної системи під певне залізо робіть ПК від Apple більше продуктивне й стабільним, ніж аналогічне рішення на Windows.

Особливості виробників

Порівняння по цифрах не дає розуміння того, що буде вигідніше на практиці. Якщо виробник може надати встаткування на тест – це великий плюс, але найчастіше орієнтуватися доводиться по досвіду людей, які вже впровадили й використовують систему. Допоможуть тематичні форуми в Інтернеті, а також фахівці технічної підтримки розроблювача. Менеджери по продажах будуть розповідати про те, що їхній пристрій краще інших, але якщо сформувати список конкретних технічних питань і задати їх у підтримку, то подання про продукт буде більше повним.

Наприклад, компанія Sandvine є присутнім на українському ринку недавно, має мало представників і небагато реалізованих кейсів, здійснює підтримку тільки через інтегратор (що незручно), зате часто обновляє ПЗ своїх платформ і дозволяє їх гнучко налаштовувати.

Procera може похвастатися зручним клієнтом PacketLogic Client для настроювання й обслуговування, а також відмінною візуалізацією виконуваних функцій.

Рішення від Cisco може має обмеження смуги пропущення в 15 Гбіт/с на один Service Control модуль, зате функція Subscriber Manager не вимагає додаткових ліцензій, а працює в базовій поставці.

VAS Experts регулярно обновляє ПЗ СКАТ для виправлення помилок, додавання нових протоколів і сигнатур. З одного боку, це добре, тому що нові сигнатури дозволяють розділяти трафік більш докладно, з іншого боку – виправлення одних помилок може сприяти появі нових, система не володіє 100%-ний стабільністю.

Зразковий розрахунок

Продуктивність, функції, зручність використання – це дуже важливі фактори, однак принципове рішення про інтеграцію системи DPI залежить від OPEX (операційних витрат). Приведемо невеликий розрахунок ефективності (ROI) від впровадження системи керування трафіком DPI.

Вихідні дані:

- Кількість абонентів – 15 000.
- Вартість DPI рішення – 75 000 $.
- Пропонована ефективність DPI рішення – 35%.
- Вартість 1Гбіт/с – 3 000$ / мес.
- Смуга пропущення на абонента – 2 Мбіт/с.
- Коефіцієнт активності абонента – ½ (0,5).

Споживані абонентами трафік становить:

(2 Мбіт/с * 0,5) * 15 000 = 15 Гбіт/с на місяць

Використання системи DPI при заявленій ефективності по зменшенню аплінка в 35% дає нам 5,25 Гбіт/с економії смуги пропущення на місяць (15 Гбіт/с * 0,35 = 5,25 Гбіт/с). Монетизація економії 5,25 Гбіт/с * 3 000$ = 15 750$.

При такій щомісячній економії окупність впровадження DPI складе всього 5 місяців (75 000 $ / 15 750 $ = близько 5 місяців), звичайно, це зразковий розрахунок, але він дозволяє оцінити вигоду. А якщо додати до цього реалізацію додаткових функцій, за які не прийде мати додаткові витрати: блокування заборонених ресурсів, захист від мережних атак, CGNAT (у СКАТ від VAS Experts), те окупність наступить ще швидше, а якість надаваних послуг абонентові за тугіше абонентську плату – вище.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи управління мережею Wi-Fi та аналізу даних.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Система DPI, як правило, установлюється на границі мережі оператора в розрив існуючих аплінків, що йдуть від прикордонних маршрутизаторів. Тим самим, весь трафік, що залишає або входить у мережу оператора, проходить через DPI, що дає можливість його моніторингу й контролю. Для рішення специфічних завдань можна встановлювати цю систему не на границі мережі, а спускати її нижче, ближче до кінцевих користувачів, на рівень BRAS/CMTS/GGSN/... Це може бути корисно тим операторам, які з ряду причин крім утилізації зовнішніх каналів також хочуть вирішувати завдання контролю внутрішніх. Природно, тут мова йде про досить великі сервіси-провайдери з великою розподіленою мережею масштабів країни й з досить дорогими канальними ємностями.

На ринку DPI є моделі на самий різний гаманець. Продуктивність представлених на ринку пристроїв плаває в межах від сотень Мбіт/с до 160 Гбіт/с FDX у рамках однієї окремо взятої коробки, які, як правило, можна поєднувати в кластери. Відповідно, і вартість плаває досить серйозно – від декількох тисяч до мільйонів доларів США. У випадку з корпоративним сегментам рішення припускають низькошвидкістні підключення по мідних інтерфейсах типів 10/100/1000. Операторські рішення розраховані на підключення безлічі лінків 1GE і 10GE. Що стосується зовсім дорослих рішень, те поки що ринок 100GE інтерфейсів на мережному встаткуванні досить убогий і дорогий, але як тільки з'явиться перший реальний бізнес-кейс, вендори DPI запропонують відповідні рішення, тому що в деяких з них заготівлі вже є.

Subscriber Management

Важливим моментом є те, що правила, на підставі яких виконується шейпинг/блокування, можуть бути задані за допомогою двох основних базисів –

per-service або per-subscriber. У першому випадку найпростішим образом обмовляється, що конкретному застосунку дозволяється утилізувати певну смугу. У другому прив'язка застосунка до смуги здійснюється для кожного передплатника або групи передплатників незалежно від інших, що виробляється через інтеграцію DPI з існуючими OSS/BSS системами оператора. Тобто можна налаштувати систему таким чином, що передплатник, що за тиждень накачав торрентів на 100 гігабайт, до кінця місяця буде обмежений по швидкості завантаження цих же торрентів на рівні 70% від купленого їм тарифу. А в передплатника, що купив додаткову послугу за назвою «Skype без проблем», трафік застосунка Skype не буде блокуватися ні при яких умовах, але будь-який іншої – легко. Можна зробити прив'язку до User-Agent і дозволити браузинг тільки за допомогою браузерів, що рекомендуються, можна робіти хитрі редіректи залежно від типу браузера або ОС. Іншими словами, гнучкість тарифних планів і опцій обмежена лише здоровим глуздом. Якщо ж мова йде про трафіку мобільних операторів, то DPI дозволяє контролювати завантаження кожної базової станції окремо, справедливо розподіляючи ресурси БС таким чином, щоб всі користувачі залишилися задоволені якістю сервісу. Зрозуміло, дану завдання можна вирішувати силами мобільного ядра, але це не завжди бюджетно. Раз вуж я згадав мобільних операторів, то хотілося б відзначити, що кожний поважаючий себе виробник пакетного ядра EPC (Evolved Packet Core) для LTE інтегрує у свій PDN-GW функціонал DPI, заточений під рішення завдань мобільних операторів.

Звучить це все, звичайно, не дуже оптимістично, але для багатьох операторів по економічних причинах значно дешевше поставити систему DPI для контролю утилізації каналів, чим розширювати аплінки. Причому, зробити це без особливих втрат абонентської бази, тому що давно відомо, що більша частина трафіку генерується приблизно 5% найбільш активних абонентів. І в цьому випадку операторові економічно більш вигодно знизити абонентську базу, але платити менше грошей за аплінки, тому що підуть самі активні качальщики,

через які оператор змушений щомісяця платити немаленьку суму за аплінки. Це нічний кошмар будь-якого маркетолога, але в деяких випадках втратити клієнтів – вигідно. Делікатність ситуації полягає в тому, що рано або пізно наступить такий момент, коли всі оператори так чи інакше будуть що-небудь шейпить за допомогою DPI. Тобто якщо сьогодні один оператор почне рубати торренти, самі активні качальщики разом підуть до іншого. Після цього в того сильно скакне завантаження його каналів і клієнти почнуть скаржитися на те, що погано працює веб-браузинг. Оператор подумає, підрахує, і в підсумку купить DPI. І так доти, поки всі гравці на ринку не обзаведуться подібною системою. Зрозуміло, установка DPI не знімає з оператора завдання по періодичному розширенню аплінків і збільшенню швидкості доступу для передплатників. Просто тепер ці розширення не будуть безконтрольними. Тобто оператор завжди буде знати трафік якого типу й у якій кількості піде через його канали, це буде прогнозовано. Зрозуміло, коли мова йде про коробки вартістю $1M, справа не тільки в аплінках, необхідно це розуміти. Моя особиста думка в першому наближенні, як користувача послуги широкополосного доступу в інтернет, полягає в тому, що що-небудь різати й блокувати, звичайно ж, погано й зовсім неправильно. Але, дивлячись очами інженера на те, якими темпами ростуть обсяги трафіку, використання DPI стає порятунком для багатьох операторів, тому що торренти сьогодні здатні забити намертво практично будь-який аплінк.

Нова модель послуг

Ми плавно перейшли до завдання розвитку мережі і її послуг. Дивлячись на те, як передплатники користуються купленої ними смугою, які застосунки використовують, оператор може вивчати потреби кожної категорії передплатників і пропонувати їм більше гнучкі й зроблені тарифні плани. Приміром, ґрунтуючись на тім, що передплатники тарифу Silver активно користуються послугами сторонньої SIP-телефонії, можна запропонувати їм додатковий пакет, що дозволяє використовувати аналогічний сервіс, надаваний оператором, але й з знижкою. Інші передплатники при бажанні скористатися

більше дешевою телефонією будуть мотивовані переходити на більше дорогий тариф, здобуваючи додаткові бонуси у вигляді підвищення швидкості. Можна придумати багато кейсів, це лише один з них. Своє бачення персоналізованих сервісів представила компанія Allot. Підхід дуже цікавий, і вигідний як для користувача, так і для оператора. Тенденції розвитку телекомунікаційного ринку такі, що для операторів продавати трубу, як вони роблять зараз, незабаром буде просто невигідно, є маса досліджень, що підтверджують це. ARPU не збільшується, конкуренція висока, устаткування необхідно апгрейдити все частіше й частіше, витрати операторів ростуть, а бажання діставати прибуток нікуди не подінеться. Завдання DPI у даному розрізі – реалізувати нові моделі надання послуг кінцевому користувачеві. Деякі світові оператори маленькими кроками вже рухаються до даної ідеї. В Україні, мабуть, процес цей буде довгим і болісним, тому що для досягнення завдання необхідно перебудовувати мозки абонентів на іншу частоту, що дуже непросто, тому що відучити людини не качати торренти, а купувати легальний контент – непросто. Я б не хотів зараз запускати дискусію на тему «А де мені брати легальний контент?», це окрема пісня, і я дуже радий, що це зрушилося з мертвої точки (на прикладі ivi, omlet, zabava і т.п. разом зі зростаючими продажами Smart TV). Сподіваюся, дані проекти не стихнуть. Про Netflix я поки не мрію, але було б здорово.

DPI відмінно вміє працювати у зв'язуванні з різними VAS (Value Added Services) системами, такими як антиспам, антивірус, відеооптимізатори й т.п. Суть функціонала полягає у відводі частини трафіку за заданим адміністратором критеріями, на сторонні пристрої, для здійснення більше глибокого аналізу й обробки.

Досить легко можна організувати надання користувачам послуг по батьківському контролі, які стають усе більше й більше актуальними.

Спецслужби

Наприкінці хотілося б сказати пари слів про те, для чого також закупається DPI, крім як для знущань над абонентами. Устаткування DPI, у

зв'язку зі своїм умінням бачити геть усе, що відбувається на мережі, є досить цікавим пристроєм для товаришів у погонах, без яких зараз нікуди. За допомогою DPI спецслужби можуть вести спостереження за мережною активністю того або іншого користувача. Можна перекрити йому VPN, HTTPS та інші принадності, що роблять неможливим аналіз контенту. Зрозуміло, можна закривати доступ користувачів до неугодної влади сайтам, що дуже актуально у зв'язку з останніми подіями в законотворчій діяльності в Україні.

Мережний нейтралітет

І, нарешті, хотілося б сказати пари слів про багатостраждальний мережний нейтралітет, що існує в деяких країнах. Якщо коротко, то операторам під час відсутності перевантажень на аплінках нині заборонено блокувати трафік законних/легальних застосунків. Тобто почати вибіркове блокування будь-якого трафіку тепер дозволяється тільки у випадку виникнення перевантаження. Але, у той же час, ще немає чітких формулювань на тему того, які саме застосунки є законними, а які – немає. По логіці, незаконним може бути тільки контент, а не застосунка. Приміром, дитяча порнографія явно ставиться до незаконного контенту, але протоколи HTTP і Bittorrent, за допомогою яких можна здійснювати його передачу – цілком собі легальні. Так що отут є ще досить великий простір для суперечок, а тема, на мій погляд, досить цікава. Поки що в нас мережним нейтралітетом не пахне, тому в операторів на руках – всі карти для керування трафіком за допомогою DPI. Основною функцією систем DPI є фільтрація трафіку, і виконує її програмний комплекс, що передвстановлений на апаратну платформу. Добре написане й оптимізоване ПЗ дозволяє системі DPI виконувати багато функцій крім фільтрації (пріоритизація, заборона ресурсів, повідомлення абонентів і навіть CGNAT), але без спеціально підібраного заліза програмна частина системи не буде працювати максимально продуктивно. У попередніх статтях про іноземних і українських виробників DPI ми розповідали, що ПЗ може бути заточене під конкретне встаткування (Allot, Cisco, Procera) або поставлятися окремо для установки на сумісне (Sandvine, Vas Experts, «Протей»).

3.2 Розробка структурної схеми

Як показало дослідження 2024 Global Shopper Study, приблизно 44% покупців використовують мережу Wi-Fi у магазині для відвідування веб-сайту його власника або виробників товарів, що цікавлять. Вони задіють свої смартфони для безлічі цілей: від перегляду порівняльних оглядів обираних продуктів і відкликань інших покупців до використання скідочних купонів і властиво оплати товарів, що здобуваються.

Підтримка складної мережі Wi-Fi, що охоплює більші площі в десятках магазинів і обслуговуюча безліч різнотипних мобільних пристроїв, – непросте завдання для IT-відділу торгового центра. Її ефективне рішення практично неможливо без наявності сучасної системи керування й моніторингу, що могла б відслідковувати різні аспекти роботи мережі на всіх рівнях, аж до рівня застосунків.

Реалізуємо можливості такої системи, що розроблена в даній роботі. ПЗ одержує доступ до універсального засобу керування мережею Wi-Fi і аналізу даних інформації й представляє неї в зручному для перегляду через браузер виді в реальному часі або показує історичні дані. Універсальний засіб керування мережею Wi-Fi і аналізу даних, побудовано на основі принципів програмувальних мереж (SDN) і дозволяє при необхідності розподілити функціональність контролера бездротової ЛОМ на кожну точку доступу.

Одним із ключових компонентів рішення служить убудований в операційну систему Універсального засобу керування мережею Wi-Fi і аналізу даних движок Deep Packet Inspection (DPI), що здатний відслідковувати потоки трафіку кожного користувача, виділяти й ідентифікувати тисячі використовуваних застосунків. Відповідна інформація в реальному часі передається на платформу NSight. У результаті адміністратор мережі одержує повні відомості по всім семи рівнях моделі OSI, включаючи дані по типі користувальницьких пристроїв, їхнім операційним системам, використовуваним

застосункам та ін. Ці відомості надзвичайно корисні як для експлуатації мережі, так і для взаємодії з покупцями.

При наявності мереж Wi-Fi на декількох вилучених об'єктах система покаже їх на картах Google Maps. Клич мишею на об'єкті, що цікавить, – і система видасть інтерактивну схему приміщень, на якій відображені основні об'єкти мережі Wi-Fi, а також показані проблемні зони. Система в режимі онлайн видає загальну статистику за такими показниками, як число підключених користувачів, завантаження мережних ресурсів, що тече випускна здатність, дані по використанню тих або інших застосунків. Можна легко й просто «спуститися» на рівень конкретної точки доступу, подивившись параметри частотних каналів, рівень помилок, частку повторних запитів та ін. Зокрема, можна вивчити картину розподілу частотних каналів, перевірити, наскільки вони оптимально призначені, немає чи проблем з інтерференцією.

Інтерактивні схеми приміщень із вказівкою щільності мобільних пристроїв (а виходить, і покупців) – це можливість не тільки оптимізувати розподіл мережних ресурсів і спланувати модернізацію, але й одержати дані про щільність покупців у різних зонах торгового центра. Засоби DPI дозволяють контролювати розподіл трафіку різних застосунків, установлюючи більше високий пріоритет найбільш важливим для конкретного магазина або торгового центра. Крім того, завдяки DPI платформа дає можливість відслідковувати навіть те, які сайти конкурентів відвідує покупець, перебуваючи в магазині, щоб вчасно перехопити його увагу й сформувати привабливу пропозицію. Нарешті, аналіз трафіку на рівні застосунків – це гарний інструмент, для того щоб оперативно виявити можливі погрози або використання небажаних застосунків і вжити заходів ще до того, як вони заподіяли шкоду мережі або доступної через неї інформації. А це особливо важливо в мережі, у якій працює безліч «чужих» пристроїв.

Рішення типу – ефективні універсальні засоби керування мережею WI-Fi і аналізу даних, що збираються з допомогою такої мережі. І більшість виробників

бездротових систем обмежуються пропозицією лише універсальних засобів аналітики, що підходять для самих різних галузей. Мінусом такого підходу є необхідність проведення серйозних робіт з адаптації рішення для конкретної галузі, у нашім випадку – для торговельних підприємств. Це може зажадати значних витрат, особливо з обліком того, що торгові центри не завжди мають у своєму розпорядженні кваліфікований персонал, здатним виконати таку адаптацію й кастомизацію.

Система підтримує безліч варіантів, що налаштовуються адміністратором, реєстрації, у тому числі й через найбільш популярні соціальні мережі. Це не тільки дає покупцям можливість вибору найбільш зручного способу реєстрації, але й дозволяє торговельній організації одержувати через соціальні мережі масу інформації про віковий состав користувачів, їхньої гендерної приналежності, інші важливі характеристики.

Система здатна масштабуватися до підтримки 10 мільйонів покупців. Серед типових даних, видаваних системою:

– загальне число покупців, що зайшли в той або інший магазин;
– відношення числа нових покупців до числа постійних клієнтів;
– час, проведений покупцями в магазині;
– тип використовуваних ними мобільних пристроїв;
– демографічні дані.

На структурній схемі, що відображена рисунком 3.1, зазначено, які саме дані може збирати система.

В основі системи – потужний набір засобів з аналізу даних, що збираються, представлених у зручному для використання виді, що дозволяє фахівцям торгового центра зробити маркетингові кампанії й пропозиції більше персоніфікованими, а виходить, ефективними, в остаточному підсумку забезпечуючи більше високу якість обслуговування й підвищуючи задоволеність покупців.

Рисунок 3.1 – Структурна схема системи

Наявність у магазині або торговому центрі не просто точок доступу Wi-Fi, а дійсно повнофункціональної мережі бездротового зв'язку з ефективними засобами аналізу даних, що збираються – це важлива умова для підвищення конкурентоспроможності традиційних торговельних підприємств. Надавши покупцям можливість комфортно користуватися в магазині улюбленими ними смартфонами, у тому числі для уточнення інформації з обираного товару, і впровадивши сучасні засоби аналізу їхніх переваг і побажань, традиційні торгові центри зможуть успішно конкурувати з інтернет-магазинами. Сучасна мережа Wi-Fi – одна із ключової складової цифрової трансформації бізнесу ретейла.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Так, як функціональна схема є більш подобним описом функціональних можливостей структурної схеми, то вона буде представляти собою, більш детальний варіант структурної схеми.

З рисунку видно, що розроблена система складається з наступних частин:

– Блок визначення топології мережі Wi-Fi.

– Блок аналізу даних.

– Блок зберігання результатів.

– Блок перевірки та фільтрації пакетів у мережі Wi-Fi.

– Блок аналізу мережної статистики.

Блок визначення топології мережі Wi-Fi

Блок визначення топології мережі Wi-Fi:

– Блок використання відомостей із загальної системи моніторингу мережі Wi-Fi, а не опитування пристрою додатково.

– Блок складання списку пристроїв у мережі Wi-Fi, автоматично, ґрунтуючись на дані системи моніторингу.

– Блок побудови топології мережі Wi-Fi, за станом на задану дату й відстеження змін у топології протягом часу.

– Блок автоматичного визначення рівнів ієрархії пристроїв у мережі Wi-Fi, з виділенням периферійних, проміжних і центральних вузлів;

– Блок побудови топології мережі Wi-Fi, незалежно від використовуваної системи моніторингу й програмно-апаратних платформ;

– Блок комбінувати показників, на основі яких визначаються зв'язки між пристроями, і при їхньому обчисленні виконувати перевірку на значимість із використанням статистичних критеріїв.

Блок аналізу даних

Блок аналізу даних:

– загальне число покупців, що зайшли в той або інший магазин;

– відношення числа нових покупців до числа постійних клієнтів;

– час, проведений покупцями в магазині;

– тип використовуваних ними мобільних пристроїв;

– демографічні дані.

У період навчання Детектор аналізу трафіку збирає базову інформацію для розуміння нормальної роботи мережі Wi-Fi, куди входять:

– Інтенсивність пакетів для кожного типу пакетів, обмірювана як кількість пакетів у секунду (pps).

– Співвідношення пакетів, наприклад, співвідношення пакетів SYN і пакетів FIN.

– Кількість одночасних TCP-з'єднань, відкритих одним джерелом.

Базова інформація збирається по кожній цільовій адресі хост-ПК, цільовій підмережі Wi-Fi, вихідній адресі хост-ПК і вихідній підмережі Wi-Fi.

Після закінчення періоду навчання Детектор аналізу трафіку переводиться в режим моніторингу. Доти, поки немає атаки, що активно розвивається, вхідний трафік з мережі Wi-Fi Інтернет проходить через комутатор без якого-небудь втручання з боку Блоку усунення аномального трафіку. Копія вхідного трафіку посилає для аналізу на Детектор аналізу трафіку через зовнішній аналізатор протоколів (SPAN) або віртуальні списки ACL.

Якщо Детектор аналізу трафіку виявляє аномальне в порівнянні з базовою інформацією поводження трафіку, починається процес усунення:

– Детектор аналізу трафіку направляє в Блок усунення аномального трафіку команду почати процес зміни напрямку.

– Блок усунення аномального трафіку відхиляє (“захоплює”) трафік, адресований на атакуєму IP-адресу, переадресуючи його на самого себе.

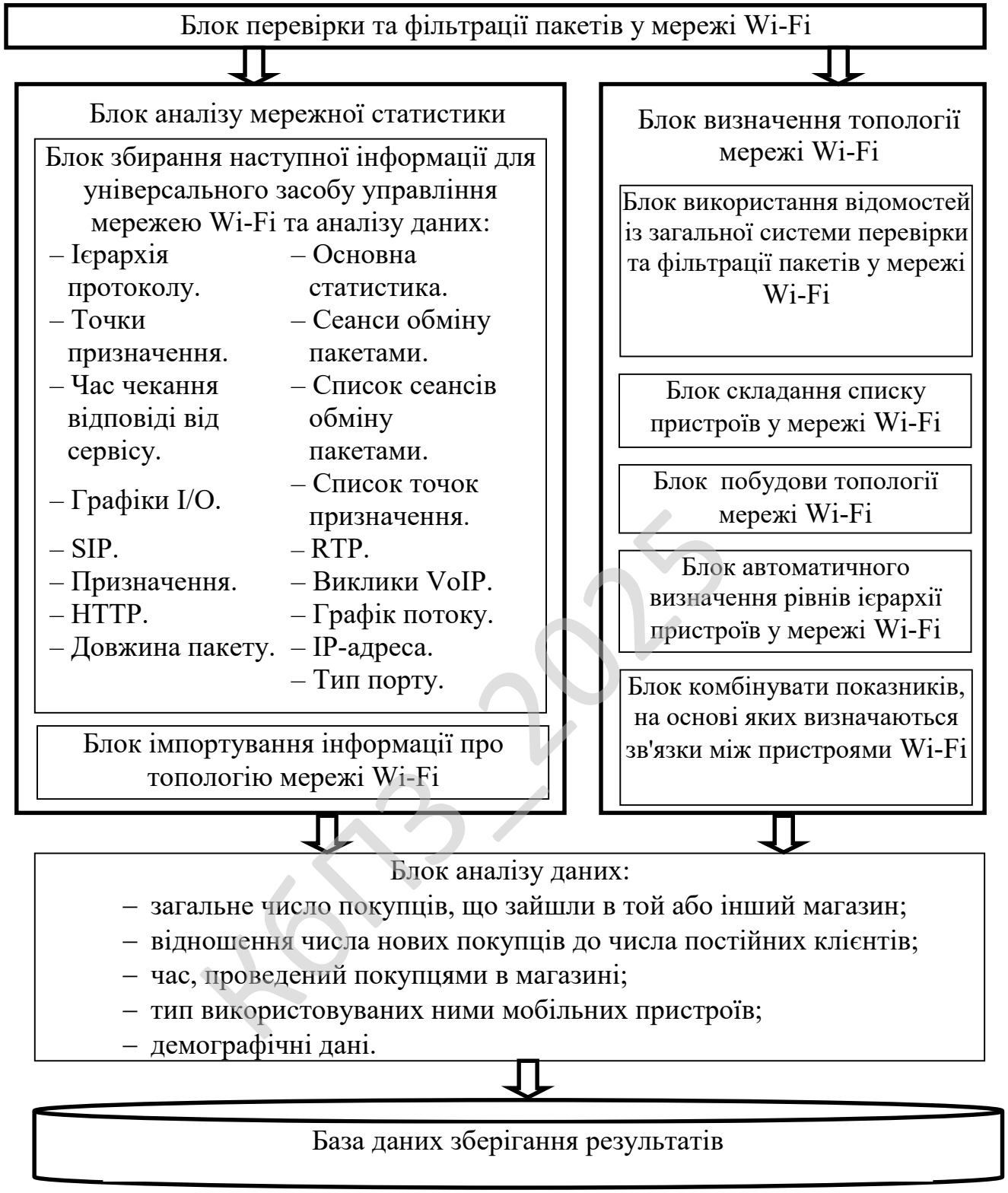

Рисунок 3.2 – Функціональна схема системи

– Блок усунення аномального трафіку піддає трафік багатоступінчастому аналізу й застосовує контрзаходи для відділення благонадійних джерел від джерел атаки. Цей процес іменується очищенням або вичищенням.

– Блок усунення аномального трафіку скидає трафік атаки й пересилає благонадійний трафік назад на нормальний маршрут проходження трафіку до мети. Цей процес іменується ін'єкцією.

Блок аналізу мережної статистики

Блок збирання наступної інформації:

– Основна статистика (Summary).

– Ієрархія протоколу (Protocol Hierachy).

– Сеанси обміну пакетами (Conversations).

– Точки призначення (Endpoints).

– Графіки I/O (IO Graphs).

– Список сеансів обміну пакетами (Conversation List).

– Список точок призначення (Endpoint List).

– Час чекання відповіді від сервісу (Service Response Time).

– RTP.

– SIP.

– Виклики VoIP (VoIP Calls).

– Призначення (Destination).

– Графік потоку (Flow Graph).

– HTTP.

– IP-адреса (IP address).

– Довжина пакету (Packet Length).

– Тип порту (Port Type).

Розпишемо їх більш детально.

1. Основна статистика. Доступні такі елементи основної статистики, як:

– Властивості захоплених файлів.

– Час захвату.

– Інформація про фільтр захвату.

– Інформація про фільтр відображення.

2. Ієрархія протоколу. Статистика ієрархії протоколу допомагає аналізувати пакети, розбиваючи відображені дані, які належать чинному рівню OSI.

3. Сеанси обміну пакетами. Якщо ви використовуєте протокол TCP/IP або програму, яка працює із цим протоколом, ви маєте побачити чотири активних вкладок для обміну пакетами за допомогою Ethernet, IP, TCP та UDP. «Діалог» між комп'ютерами відображає трафік між двома активними хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами, наприклад, «Ethernet:6».

4. Точки призначення. Точки призначення забезпечують статистику даними про відправку та прийом пакетів. Номер, зазначений на вкладці після назви протоколу, вказує на кількість точок призначення. Наприклад, «Ethernet:6».

5. Графіки I/O. Основний графік може бути отриманий за допомогою команди «IO graphs» (Графіки I/O). Ще декілька графіків можуть бути додані у тому ж вікні на основі фільтрів відображення.

6. Час чекання відповіді від сервісу. 13 протоколів доступні для глибокого аналізу.

7. RTP. RTP (Real-time Transport Protocol, протокол передачі у реальному часі, RFC 3550) – це протокол для передачі звука та відео через IP-мережу. Він працює у початку протоколу дейтаграм користувача (User Datagram Protocol, UDP). Він часто використовується у сукупності з протоколами SIP або H.233, забезпечуючи виконання сигнальних завдань.

8. SIP. SIP (Session Initiation Protocol, протокол встановлення сесії, RFC 3261) – це сигнальний протокол, який оголошує відео– або VoIP-сесії. Він працює разом із протоколом RTP, який використовується для передачі мультимедійних даних.

9. Виклики VoIP.

VoIP (Voice over IP, голосовий зв'язок за допомогою Інтернету) взагалі використовує два типи протоколів:

– сигнальні протоколи, такі, як SIP або H.323

– переносні протоколи, наприклад, RTP

10 Призначення. Відображення усіх IP-адрес призначення мережевих пакетів.

11. Графік потоків. Графіки потоків забезпечує послідовний аналіз TCP-з'єднань. Перші три строки містять оголошення TCP– з'єднання з послідовностями «SYN», «SYN ACK» та «ACK».

12 HTTP. HTTP (Hypertext Transfer Protocol, протокол передачі гіпертексту) – це протокол типу «клієнт-сервер», який використовується для передачі HTML-файлів. HTTP-клієнт (у більшості випадків це web-браузер) відсилає HTTP-запит до web-серверу із полем «URL», який допомагає знайти потрібний файл. Web-сервер відповідає HTTP-пакетом та забезпечує клієнт необхідною web-сторінкою.

Меню «HTTP» містить три підменю:

– «Load Distribution» (Розподіл пакетів).

– «Packet Counter» (Лічильник пакетів).

– «Requests» (Запити).

14 IP-адреса. Відображення IP-адреси джерела або призначення мережевих пакетів.

15. Довжина пакету.

16. Тип порту. Відображення статистики портів TCP або UDP.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Рисунок 3.3 – Діаграма взаємодії процесів

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю універсального засобу управління мережею Wi-Fi та аналізу даних.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

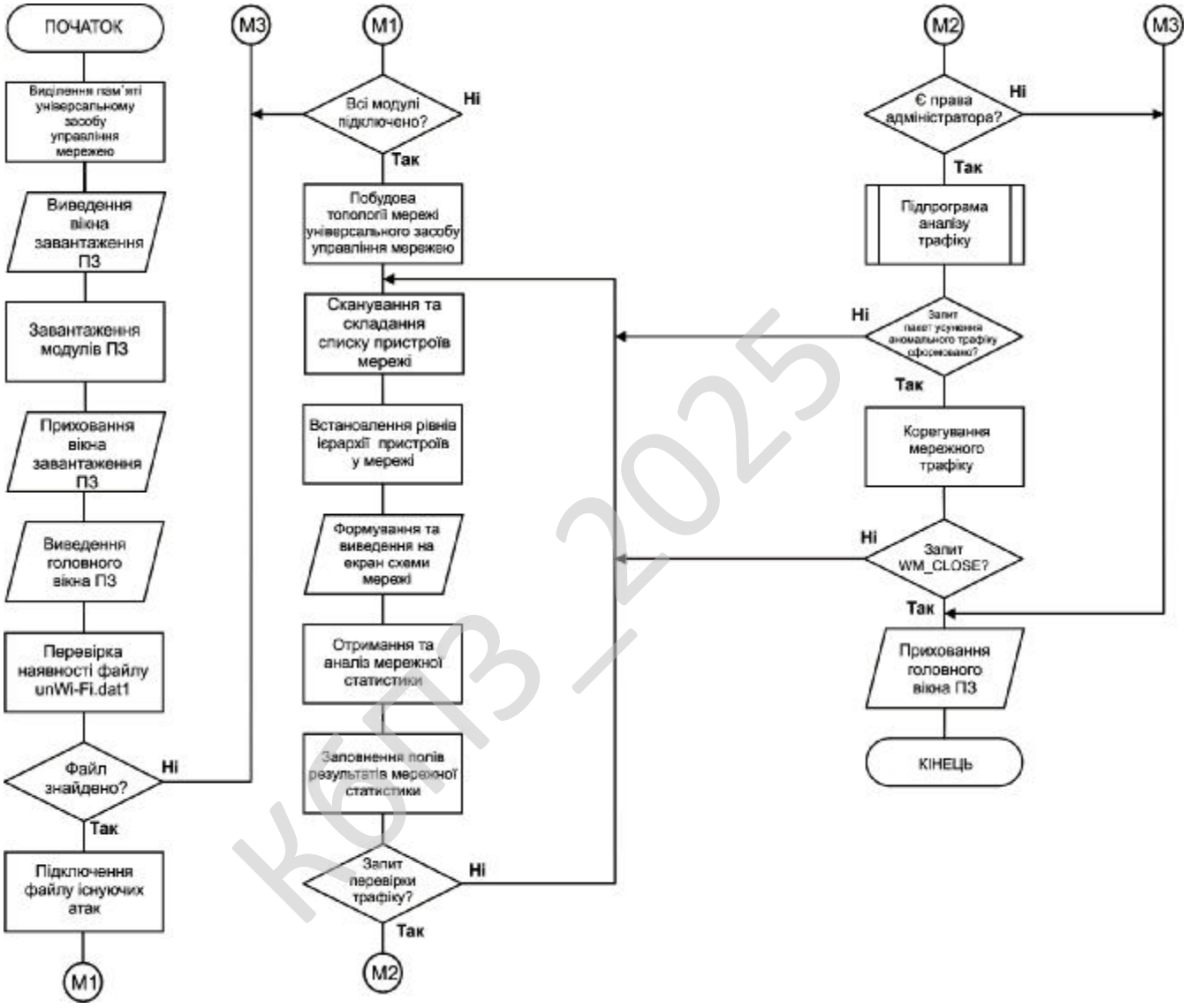

Рисунок 4.1 – Блок-схема основної програми

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

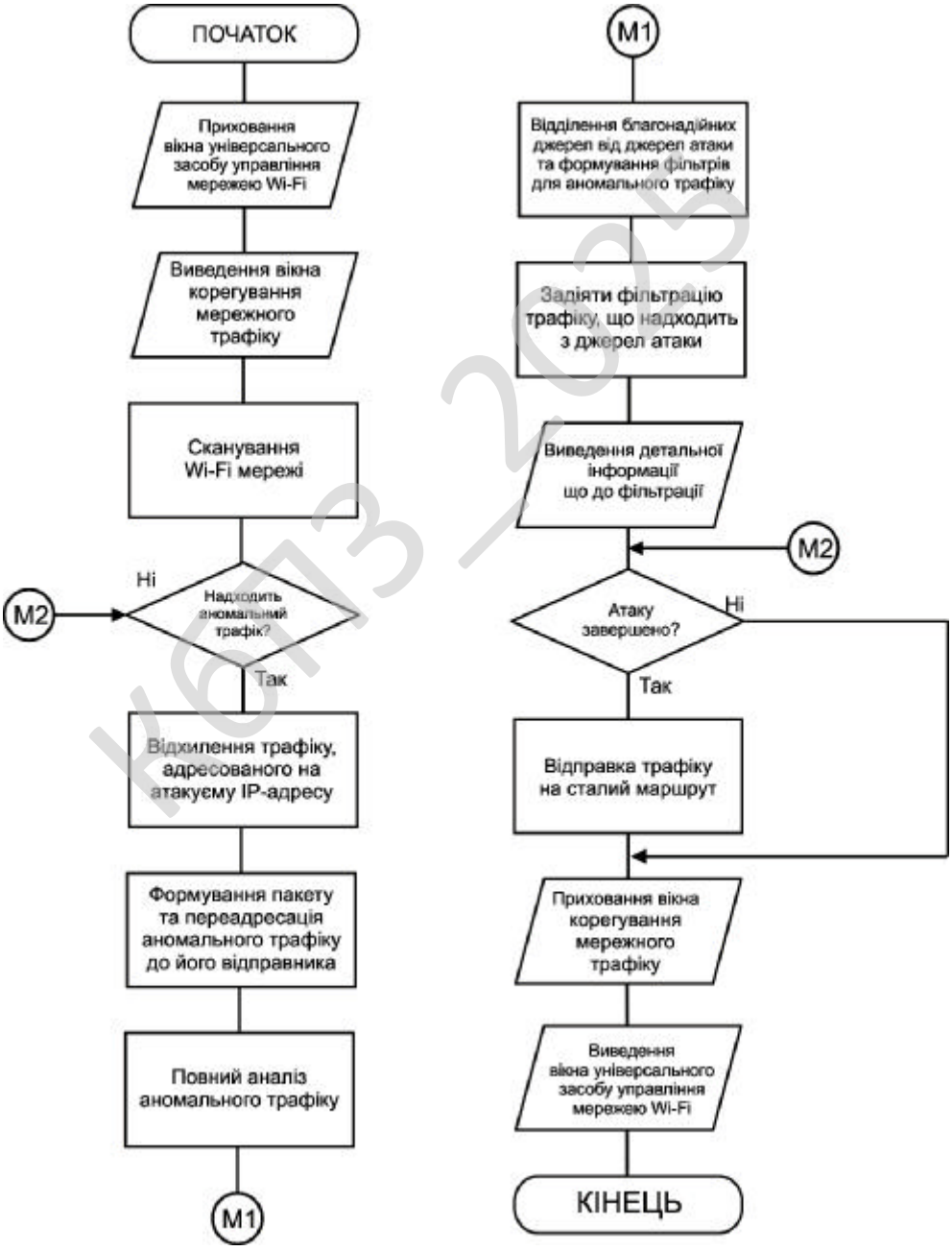

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики XP зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Технологія екстремального програмування була розроблена Кентом Беком, Уардом Каннінґхемом та Роном Джеффріесом під час роботи над Chrysler Comprehensive Compensation System (C3). У 1996 Кент Бек став лідером проекту і почав вдосконалювати методи розробки, що застосовувалися в роботі над проектом. Свій метод він виклав у книзі «Extreme Programming Explained», котру було видано у жовтні 1999. Після купівлі Крайслера компанією Даймлер–Бенц проект C3 було скасовано у лютому 2000.

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану (традиційна особливість для програмного забезпечення, що проектується).

XP використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

XP намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн.

Написання коду. Прихильники XP заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми XP. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

1.3. Замовник завжди поруч (Whole team, Onsite customer).

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1. Простота (Simple design).

3.2. Метафора системи (System metaphor).

3.3. Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).

3.4. Стандарт кодування (Coding standard or Coding conventions).

4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму SHA-3 (Keccak) – алгоритм гешування змінної розрядності, розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing. 2 жовтня 2012 року Keccak став переможцем конкурсу криптографічних алгоритмів, проведеним Національним інститутом стандартів і технологій США. 5 серпня 2015 року алгоритм

затверджено та опубліковано в якості стандарту FIPS 202[1]. У програмній реалізації автори заявляють про 12,5 циклах на байт при виконанні на ПК з процесором Intel Core 2. Проте в апаратних реалізаціях Keccak виявився набагато швидшим, ніж всі інші фіналісти.

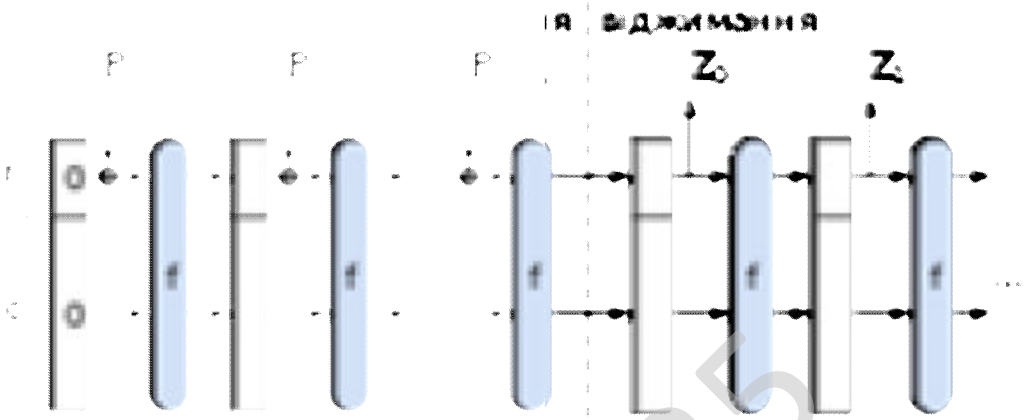

Рисунок 4.3 – Конструкція функції губки, використана в геш-функції

Конструкція функції губки, використана в геш-функції. P_i – вхідні блоки, Z_j – вихід алгоритму. Невикористаний для виведення набір бітів c («capacity») повинен мати значний розмір для досягнення стійкості до атак.

Алгоритм SHA-3 побудований за принципом криптографічної губки (дана структура криптографічних алгоритмів була запропонована авторами алгоритму Keccak раніше).

Геш-функції сімейства SHA-3 побудовані на основі конструкції криптографічної губки, в якій дані спочатку «вбираються» в губку, при якому початкове повідомлення M піддається багатораундовим перестановкам f, потім результат Z «віджимається» з губки. На етапі «вбирання» блоки повідомлення додаються за модулем 2 з підмножиною стану, який потім перетвориться з допомогою функції перестановки f. На етапі «віджимання» вихідні блоки зчитуються з одного і того ж підмножинного стану, зміненого функцією перестановок f. Розмір частини стану, який записується і зчитується, називається

«швидкістю» (англ. rate) і позначається r, а розмір частки, яка незаймана введенням / виведенням, називається «ємністю» (англ. capacity) і позначається c.

Алгоритм отримання значення хеш-функції можна розділити на кілька етапів:

– Вихідне повідомлення M додається до рядка P довжини, кратній r, за допомогою функції доповнення (pad-функції).

– Рядок P ділиться на n блоків довжини r: P_0, P_1,..., P_{n-1}

– «Всмоктування»: кожен блок P_i доповнюється нулями до рядка довжини b біт і підсумовується по модулю 2 з рядком стану S, де S – рядок довжини b біт (b = r + c). Перед використанням цієї функції всі елементи S дорівнюють нулю. Для кожного наступного блоку стан – рядок, отриманий застосуванням функції перестановок f до результату попереднього кроку.

– «Віджимання»: поки довжина Z менша d (d – кількість біт в результаті геш-функції), до Z додається r перших біт стану S, після кожного додавання до S, застосовується функція перестановок f. Потім S обрізається до довжини d біт

– Рядок Z довжини d біт повертається в якості результату

Завдяки тому, що стан містить c додаткових біт, алгоритм стійкий до атаки подовженням повідомлення, до якої прийняті алгоритми SHA-1 і SHA-2.

У SHA-3 стан S – це масив 5 × 5 слів довжиною w = 64 біта, всього 5 × 5 × 64 = 1600 біт. Також в Keccak можуть використовуватися довжини w, рівні меншим ступенями 2 (від w = 1 до w = 32).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ універсального засобу управління мережею Wi-Fi та аналізу даних яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Налаштування; Довідка.

– Функції обрання мережного інтерфейсу.

– Розділу виведення результату роботи системи – поточного стану.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

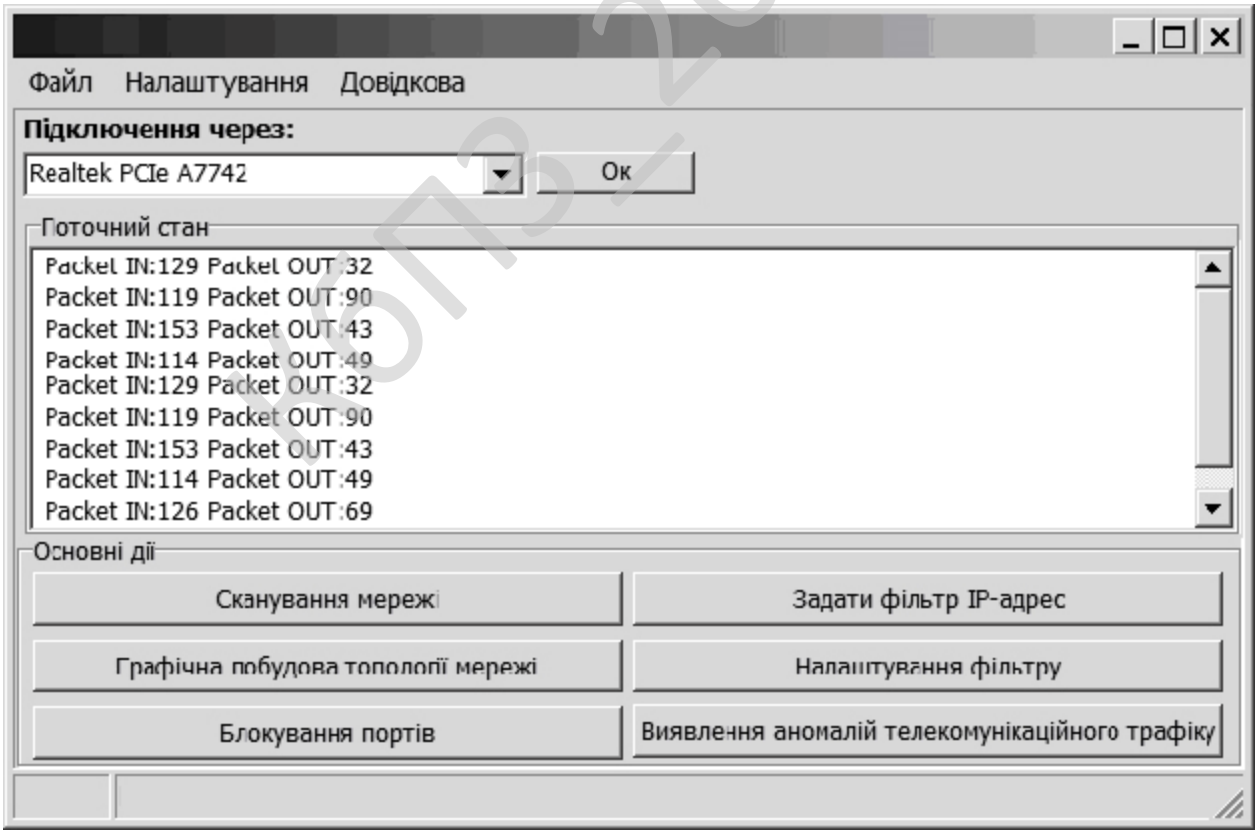

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

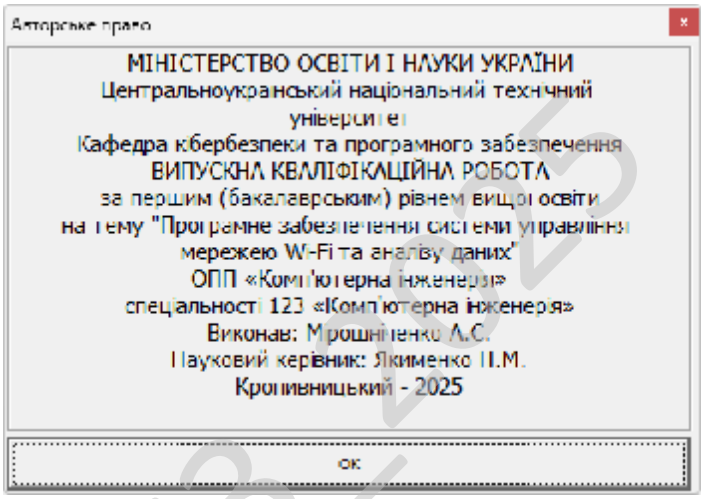

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача.

Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи управління мережею Wi-Fi та аналізу даних.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління мережею Wi-Fi та аналізу даних.

– Досліджена система управління мережею Wi-Fi та аналізу даних.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління мережею Wi-Fi та аналізу даних.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління мережею Wi-Fi та аналізу даних.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи управління мережею Wi-Fi та аналізу даних. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHA-3.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

2. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

3. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

4. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

5. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

6. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

7. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

8. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

9. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

10. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

11. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

12. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

13. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

14. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

15. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

16. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

17. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

18. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

19. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

20. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

21. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

22. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

23. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

24. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

25. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

26. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

27. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

28. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

29. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

30. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

31. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

32. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

33. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

34. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

35. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

36. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

37. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

38. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

39. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

40. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

41. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

42. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

43. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

44. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

45. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

46. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

47. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

48. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

49. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

50. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

51. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

52. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

53. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

54. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи управління мережею Wi-Fi та аналізу даних.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 48-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи управління мережею Wi-Fi та аналізу даних.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи управління мережею Wi-Fi та аналізу даних;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 64 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи управління мережею Wi-Fi та аналізу даних

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import os
import time
import subprocess
import threading
import json
import sqlite3
import psutil
import requests
import scapy.all as scapy
import matplotlib.pyplot as plt
from flask import Flask, request, redirect, render_template
from collections import defaultdict

# Ініціалізація бази даних для логування активності пристроїв
def init_db():
    conn = sqlite3.connect("network_logs.db")
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS logs (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            event TEXT,
            ip TEXT,
            mac TEXT,
            timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
        )
    """)
    conn.commit()
    conn.close()

# Логування подій у базу даних
def log_event(event, ip="", mac=""):
    conn = sqlite3.connect("network_logs.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO logs (event, ip, mac) VALUES (?, ?, ?)", (event,
ip, mac))
    conn.commit()
    conn.close()

# Функція сканування Wi-Fi мереж
def scan_wifi_networks():
    networks = []
    try:
        result = subprocess.run(["netsh", "wlan", "show", "network",
"mode=Bssid"], capture_output=True, text=True)
        output = result.stdout.split("\n")
        network_info = {}
        for line in output:
            if "SSID" in line:
                if network_info:
                    networks.append(network_info)
                    network_info = {}
                network_info["SSID"] = line.split(":")[1].strip()
            if "BSSID" in line:
                network_info["BSSID"] = line.split(":")[1].strip()
            if "Signal" in line:
                network_info["Signal"] = line.split(":")[1].strip()
            if "Channel" in line:
                network_info["Channel"] = line.split(":")[1].strip()
        if network_info:
            networks.append(network_info)
    except Exception as e:
```

```
        log_event("Error scanning networks", str(e))
    return networks

# Функція аналізу рівня сигналу
def plot_signal_strength():
    signal_data = {f"SSID_{i}": [psutil.net_io_counters().bytes_sent % 100 for _
in range(10)] for i in range(5)}
    plt.figure(figsize=(10, 6))
    for ssid, strength in signal_data.items():
        plt.plot(strength, label=ssid)
    plt.xlabel("Час")
    plt.ylabel("Рівень сигналу (dBm)")
    plt.title("Аналіз рівня сигналу Wi-Fi")
    plt.legend()
    plt.grid(True)
    plt.show()

# Функція виявлення підключених пристроїв
def detect_connected_devices():
    devices = []
    try:
        output = subprocess.run(["arp", "-a"], capture_output=True, text=True)
        for line in output.stdout.split("\n"):
            if "dynamic" in line:
                parts = line.split()
                if len(parts) > 1:
                    devices.append(parts[0])
    except Exception as e:
        log_event("Error detecting devices", str(e))
    return devices

# Функція блокування невідомих пристроїв
def block_unknown_device(mac):
    try:
        subprocess.run(["netsh", "wlan", "add", "filter", f"permission=block
ssid=all networktype=infrastructure mac={mac}"], capture_output=True, text=True)
        log_event("Blocked unknown device", mac=mac)
    except Exception as e:
        log_event("Error blocking device", str(e))

# Функція перевірки витоків DNS
def detect_dns_leak(packet):
    if packet.haslayer(scapy.DNS) and packet.haslayer(scapy.IP):
        log_event("DNS Leak Detected", packet[scapy.IP].src)

# Функція виявлення ботнетів
def detect_botnet(packet):
    if packet.haslayer(scapy.TCP) and packet[scapy.TCP].dport == 6667:
        log_event("Possible Botnet Activity", packet[scapy.IP].src)

# Функція інтеграції з SIEM-системами
def send_to_siem(log):
    siem_url = "http://siem.example.com/api/logs"
    headers = {"Content-Type": "application/json"}
    requests.post(siem_url, data=json.dumps(log), headers=headers)

# Функція автоматичного вимкнення Wi-Fi у разі загрози
def disable_wifi():
    os.system("netsh wlan disconnect")
    log_event("Wi-Fi Disconnected due to Threat")

# Функція запуску Captive Portal
app = Flask(__name__)
```

```
@app.route("/")
def index():
    mac = request.args.get("mac", "unknown")
    return render_template("login.html", mac=mac)

@app.route("/authorize", methods=["POST"])
def authorize():
    mac = request.form["mac"]
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO users (mac, status) VALUES (?, 'allowed')",
(mac,))
    conn.commit()
    conn.close()
    return redirect("http://example.com")

# Основна функція
def main():
    init_db()

    # Запуск Captive Portal у фоні
    captive_thread = threading.Thread(target=lambda: app.run(host="0.0.0.0",
port=8080))
    captive_thread.daemon = True
    captive_thread.start()

    while True:
        print("Сканування Wi-Fi...")
        networks = scan_wifi_networks()
        for net in networks:
            print(net)

        print("Перевірка підключених пристроїв...")
        devices = detect_connected_devices()
        for device in devices:
            print(f"Підключений пристрій: {device}")

        print("Запуск моніторингу DNS...")
        dns_thread = threading.Thread(target=lambda: scapy.sniff(filter="udp
port 53", prn=detect_dns_leak))
        dns_thread.daemon = True
        dns_thread.start()

        print("Запуск моніторингу ботнет-активності...")
        botnet_thread = threading.Thread(target=lambda:
scapy.sniff(filter="tcp", prn=detect_botnet))
        botnet_thread.daemon = True
        botnet_thread.start()

        print("Перевірка можливих загроз...")
        if "00:1A:2B:3C:4D:5E" in devices:
            disable_wifi()

        print("Очікування перед наступним скануванням...")
        time.sleep(30)

if __name__ == "__main__":
    main()
```

Файл scan_wifi.py

```
import subprocess
import time
import os

# Функція сканування доступних Wi-Fi мереж
def scan_wifi_networks():
    networks = []
    try:
        result = subprocess.run(["netsh", "wlan", "show", "network",
"mode=Bssid"], capture_output=True, text=True)
        output = result.stdout.split("\n")
        network_info = {}
        for line in output:
            if "SSID" in line:
                if network_info:
                    networks.append(network_info)
                    network_info = {}
                network_info["SSID"] = line.split(":")[1].strip()
            if "BSSID" in line:
                network_info["BSSID"] = line.split(":")[1].strip()
            if "Signal" in line:
                network_info["Signal"] = line.split(":")[1].strip()
            if "Channel" in line:
                network_info["Channel"] = line.split(":")[1].strip()
        if network_info:
            networks.append(network_info)
    except Exception as e:
        print(f"Error scanning networks: {e}")
    return networks

# Функція підключення до Wi-Fi
def connect_to_wifi(ssid, password):
    profile = f"""
    <WLANProfile>
        <name>{ssid}</name>
        <SSIDConfig>
            <SSID>
                <name>{ssid}</name>
            </SSID>
        </SSIDConfig>
        <connectionType>ESS</connectionType>
        <connectionMode>auto</connectionMode>
        <MSM>
            <security>
                <authEncryption>
                    <authentication>WPA2PSK</authentication>
                    <encryption>AES</encryption>
                    <useOneX>false</useOneX>
                </authEncryption>
                <sharedKey>
                    <keyType>passPhrase</keyType>
                    <protected>false</protected>
                    <keyMaterial>{password}</keyMaterial>
                </sharedKey>
            </security>
        </MSM>
    </WLANProfile>
    """
    try:
        profile_path = f"{ssid}.xml"
        with open(profile_path, "w") as file:
```

```
            file.write(profile)
        subprocess.run(["netsh", "wlan", "add", "profile",
f"filename={profile_path}"], capture_output=True, text=True)
        subprocess.run(["netsh", "wlan", "connect", f"name={ssid}"],
capture_output=True, text=True)
        time.sleep(5)
        os.remove(profile_path)
    except Exception as e:
        print(f"Error connecting to Wi-Fi: {e}")

if __name__ == "__main__":
    networks = scan_wifi_networks()
    for net in networks:
        print(net)
```

Файл wifi_signal_analysis.py

```
import matplotlib.pyplot as plt
import random
import time

# Симуляція рівня сигналу Wi-Fi для різних точок доступу
def generate_signal_data():
    return {f"SSID_{i}": [random.randint(30, 100) for _ in range(10)] for i in
range(5)}

# Побудова графіків рівня сигналу
def plot_signal_strength(signal_data):
    plt.figure(figsize=(10, 6))
    for ssid, strength in signal_data.items():
        plt.plot(strength, label=ssid)
    plt.xlabel("Час")
    plt.ylabel("Рівень сигналу (dBm)")
    plt.title("Динаміка рівня сигналу Wi-Fi")
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    data = generate_signal_data()
    plot_signal_strength(data)
```

Файл evil_twin_detector.py

```
# Виявлення атак "Evil Twin"

import subprocess
import time

# Виявлення дублікатів SSID
def detect_evil_twin():
    networks = {}
    try:
        result = subprocess.run(["netsh", "wlan", "show", "network",
"mode=Bssid"], capture_output=True, text=True)
        output = result.stdout.split("\n")
        for line in output:
            if "SSID" in line:
                ssid = line.split(":")[1].strip()
                if ssid in networks:
                    networks[ssid] += 1
                else:
                    networks[ssid] = 1

    except Exception as e:
        print(f"Помилка при скануванні: {e}")
    for ssid, count in networks.items():
        if count > 1:
            print(f"Попередження! Виявлено можливий Evil Twin: {ssid}")

if __name__ == "__main__":
    detect_evil_twin()
```

Файл network_monitor.py

```
# Моніторинг підключених пристроїв та визначення їх місцеположення


import subprocess
import re
import time

# Отримання списку підключених пристроїв
def get_connected_devices():
    devices = []
    try:
        output = subprocess.run(["arp", "-a"], capture_output=True, text=True)
        lines = output.stdout.split("\n")
        for line in lines:
            match = re.search(r"(\d+\.\d+\.\d+\.\d+)", line)
            if match:
                devices.append(match.group(1))
    except Exception as e:
        print(f"Помилка при отриманні підключених пристроїв: {e}")
    return devices

if __name__ == "__main__":
    while True:
        print("Підключені пристрої:")
        print(get_connected_devices())
        time.sleep(10)
```

Файл firewall_control.py

```
import subprocess

# Блокування пристрою за MAC-адресою
def block_device(mac_address):
    try:
        subprocess.run(["netsh", "wlan", "add", "filter", f"permission=block
ssid=all networktype=infrastructure mac={mac_address}"], capture_output=True,
text=True)
        print(f"Пристрій {mac_address} заблоковано!")
    except Exception as e:
        print(f"Помилка блокування: {e}")

if __name__ == "__main__":
    block_device("00:1A:2B:3C:4D:5E")
```

Файл traffic_analysis.py

```
import scapy.all as scapy
import sqlite3
import requests
import time

# Функція для аналізу мережевого трафіку
def packet_sniffer(packet):
    if packet.haslayer(scapy.IP):
        src_ip = packet[scapy.IP].src
        dst_ip = packet[scapy.IP].dst
        print(f"Пакет від {src_ip} до {dst_ip}")

# Функція збереження логів у базу даних
def save_log(src, dst):
    conn = sqlite3.connect("network_logs.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS logs (src TEXT, dst TEXT,
timestamp TEXT)")
    cursor.execute("INSERT INTO logs VALUES (?, ?, datetime('now'))", (src,
dst))
    conn.commit()
    conn.close()

# Функція перевірки IP-адреси через API VirusTotal
def check_ip_threat(ip):
    api_key = "YOUR_VIRUSTOTAL_API_KEY"
    url = f"https://www.virustotal.com/api/v3/ip_addresses/{ip}"
    headers = {"x-apikey": api_key}
    response = requests.get(url, headers=headers)
    return response.json()
if __name__ == "__main__":
    print("Запуск сніферу...")
    scapy.sniff(prn=packet_sniffer, store=0)
```

Файл channel_monitor.py

```
import subprocess
import re
import time

# Функція сканування каналів Wi-Fi
def scan_channels():
    channels = {}
    try:
        result = subprocess.run(["netsh", "wlan", "show", "network",
"mode=Bssid"], capture_output=True, text=True)
        output = result.stdout.split("\n")
        for line in output:
            if "Channel" in line:
                channel = line.split(":")[1].strip()
                if channel in channels:
                    channels[channel] += 1
                else:
                    channels[channel] = 1
    except Exception as e:
        print(f"Помилка сканування: {e}")
    return channels

# Функція визначення найкращого каналу
def find_best_channel():
    channels = scan_channels()
    if channels:
        best_channel = min(channels, key=channels.get)
        print(f"Рекомендований канал для Wi-Fi: {best_channel}")
    else:
        print("Не вдалося отримати інформацію про канали.")

if __name__ == "__main__":
    while True:
        find_best_channel()
        time.sleep(60)
```

Файл dns_monitor.py

```
import scapy.all as scapy
import re
import sqlite3

# Функція аналізу DNS-запитів
def analyze_dns(packet):
    if packet.haslayer(scapy.DNS) and packet.haslayer(scapy.IP):
        query = packet[scapy.DNS].qd.qname.decode("utf-8")
        src_ip = packet[scapy.IP].src
        print(f"DNS-запит: {query} від {src_ip}")
        save_log(src_ip, query)

# Функція запису логів у базу
def save_log(ip, query):
    conn = sqlite3.connect("dns_logs.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS logs (ip TEXT, query TEXT,
timestamp DATETIME DEFAULT CURRENT_TIMESTAMP)")
    cursor.execute("INSERT INTO logs (ip, query) VALUES (?, ?)", (ip, query))
    conn.commit()
    conn.close()

if __name__ == "__main__":
    print("Запуск моніторингу DNS-запитів...")
    scapy.sniff(filter="udp port 53", prn=analyze_dns, store=0)
```

Файл mac_spoof_detection.py

```
import subprocess
import re
import time

# Функція отримання MAC-адрес підключених пристроїв
def get_mac_addresses():
    macs = {}
    try:
        output = subprocess.run(["arp", "-a"], capture_output=True, text=True)
        lines = output.stdout.split("\n")
        for line in lines:
            match = re.search(r"(\d+\.\d+\.\d+\.\d+)\s+([a-fA-F0-9:-]+)", line)
            if match:
                ip, mac = match.groups()
                macs[ip] = mac
    except Exception as e:
        print(f"Помилка: {e}")
    return macs

# Функція виявлення змін MAC-адрес
def detect_mac_changes():
    known_macs = get_mac_addresses()
    while True:
        current_macs = get_mac_addresses()
        for ip, mac in current_macs.items():
            if ip in known_macs and known_macs[ip] != mac:
                print(f"Попередження! Змінена MAC-адреса: {ip} (було
{known_macs[ip]}, стало {mac})")
        time.sleep(30)

if __name__ == "__main__":
    detect_mac_changes()
```

Файл bandwidth_monitor.py

```
import psutil
import time

# Функція моніторингу мережевого трафіку
def monitor_bandwidth():
    old_data = psutil.net_io_counters()
    while True:
        time.sleep(1)
        new_data = psutil.net_io_counters()
        sent = new_data.bytes_sent - old_data.bytes_sent
        received = new_data.bytes_recv - old_data.bytes_recv
        old_data = new_data
        print(f"Передано: {sent} байт, Отримано: {received} байт")

if __name__ == "__main__":
    monitor_bandwidth()
```

Файл wifi_heatmap.py

```
import matplotlib.pyplot as plt
import random

# Генерація тестових даних покриття Wi-Fi
def generate_heatmap_data():
    return [[random.randint(30, 100) for _ in range(10)] for _ in range(10)]

# Відображення heatmap
def plot_heatmap(data):
    plt.imshow(data, cmap="coolwarm", interpolation="nearest")
    plt.colorbar(label="Рівень сигналу (dBm)")
    plt.title("Heatmap покриття Wi-Fi")
    plt.show()

if __name__ == "__main__":
    data = generate_heatmap_data()
    plot_heatmap(data)
```

Файл vpn_auto_connect.py

```
import subprocess
import time

# Функція автоматичного підключення до VPN
def connect_vpn():
    try:
        subprocess.run(["openvpn", "--config", "vpn_config.ovpn"],
capture_output=True, text=True)
        print("VPN підключено!")
    except Exception as e:
        print(f"Помилка: {e}")

if __name__ == "__main__":
    connect_vpn()
```

Файл trusted_networks.py

```
import json

# Завантаження довірених мереж
def load_trusted_networks():
    try:
        with open("trusted_networks.json", "r") as file:
            return json.load(file)
    except FileNotFoundError:
        return []

# Збереження нового списку мереж
def save_trusted_networks(networks):
    with open("trusted_networks.json", "w") as file:
        json.dump(networks, file, indent=4)

if __name__ == "__main__":
    trusted_networks = load_trusted_networks()
    print("Довірені мережі:", trusted_networks)
```

Файл dhcp_server.py

```
from scapy.all import *
import random

# Функція обробки DHCP-запитів
def dhcp_reply(packet):
    if DHCP in packet and packet[DHCP].options[0][1] == 1:
        requested_ip = "192.168.1." + str(random.randint(100, 200))
        print(f"Видається IP: {requested_ip} для {packet[Ether].src}")

if __name__ == "__main__":
    sniff(filter="udp and port 67", prn=dhcp_reply)
```

Файл captive_portal.py

```
from flask import Flask, request, redirect, render_template
import sqlite3

app = Flask(__name__)

# База даних користувачів
def init_db():
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS users (mac TEXT, status TEXT)")
    conn.commit()
    conn.close()

# Перевірка доступу до мережі
def check_access(mac):
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    cursor.execute("SELECT status FROM users WHERE mac=?", (mac,))
    result = cursor.fetchone()
    conn.close()
    return result is not None and result[0] == "allowed"

# Головна сторінка авторизації
@app.route("/")
def index():
    mac = request.args.get("mac", "unknown")
    if check_access(mac):
        return redirect("http://example.com")
    return render_template("login.html", mac=mac)

# Форма авторизації
@app.route("/authorize", methods=["POST"])
def authorize():
    mac = request.form["mac"]
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO users (mac, status) VALUES (?, 'allowed')",
(mac,))
    conn.commit()
    conn.close()
    return redirect("http://example.com")

if __name__ == "__main__":
    init_db()
    app.run(host="0.0.0.0", port=8080)
```

Файл device_activity.py

```
import subprocess
import time
import sqlite3

# Функція отримання підключених пристроїв
def get_connected_devices():
    devices = []
    output = subprocess.run(["arp", "-a"], capture_output=True, text=True)
    for line in output.stdout.split("\n"):
        if "dynamic" in line:
            parts = line.split()
            if len(parts) > 1:
                devices.append(parts[0])
    return devices

# Функція логування підключень
def log_device_activity():
    conn = sqlite3.connect("device_logs.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS logs (ip TEXT, timestamp DATETIME
DEFAULT CURRENT_TIMESTAMP)")

    while True:
        devices = get_connected_devices()
        for device in devices:
            cursor.execute("INSERT INTO logs (ip) VALUES (?)", (device,))
        conn.commit()
        time.sleep(60)

if __name__ == "__main__":
    log_device_activity()
```

Файл ad_blocker.py

```
import os

# Функція оновлення файлу hosts для блокування реклами
def block_ads():
    ad_domains = ["ads.example.com", "tracking.example.com"]
    with open("/etc/hosts", "a") as file:
        for domain in ad_domains:
            file.write(f"127.0.0.1 {domain}\n")
    os.system("systemctl restart network")

if __name__ == "__main__":
    block_ads()
```

Файл wifi_kill_switch.py

```
import os
import time

# Функція перевірки підключених пристроїв
def detect_intrusion():
    while True:
        output = os.popen("arp -a").read()
        if "00:1A:2B:3C:4D:5E" in output:
            print("Попередження! Виявлено несанкціонований пристрій. Відключення
Wi-Fi...")
            os.system("netsh wlan disconnect")
        time.sleep(10)

if __name__ == "__main__":
    detect_intrusion()
```

Файл botnet_detector.py

```
import scapy.all as scapy

# Функція аналізу підозрілих пакетів
def detect_botnet(packet):
    if packet.haslayer(scapy.IP) and packet.haslayer(scapy.TCP):
        if packet[scapy.TCP].dport == 6667:  # IRC-трафік
            print(f"Попередження! Можлива активність ботнету від
{packet[scapy.IP].src}")

if __name__ == "__main__":
    scapy.sniff(filter="tcp", prn=detect_botnet)
```

Файл protocol_analyzer.py

```
import scapy.all as scapy

# Функція аналізу мережевого трафіку
def analyze_traffic(packet):
    if packet.haslayer(scapy.IP):
        print(f"IP-пакет: {packet[scapy.IP].src} -> {packet[scapy.IP].dst}")

    if packet.haslayer(scapy.TCP):
        print(f"TCP-порт: {packet[scapy.TCP].sport} ->
{packet[scapy.TCP].dport}")
    if packet.haslayer(scapy.UDP):
        print(f"UDP-порт: {packet[scapy.UDP].sport} ->
{packet[scapy.UDP].dport}")

if __name__ == "__main__":
    scapy.sniff(prn=analyze_traffic)
```

Файл dns_leak_detector.py

```
import scapy.all as scapy

# Функція моніторингу DNS-запитів
def detect_dns_leak(packet):
    if packet.haslayer(scapy.DNS) and packet.haslayer(scapy.IP):
        print(f"DNS-запит до {packet[scapy.DNS].qd.qname.decode('utf-8')} від
{packet[scapy.IP].src}")

if __name__ == "__main__":
    scapy.sniff(filter="udp port 53", prn=detect_dns_leak)
```

Файл guest_wifi.py

```
import os

# Функція створення гостьової Wi-Fi мережі
def create_guest_wifi():
    os.system("netsh wlan set hostednetwork mode=allow ssid=GuestNetwork
key=guestpassword")
    os.system("netsh wlan start hostednetwork")
    print("Гостьова Wi-Fi мережа активована!")

# Функція відправки логів у SIEM
def send_to_siem(log):
    siem_url = "http://siem.example.com/api/logs"
    headers = {"Content-Type": "application/json"}
    requests.post(siem_url, data=json.dumps(log), headers=headers)

if __name__ == "__main__":
    sample_log = {"event": "unauthorized_access", "source_ip": "192.168.1.100"}
    send_to_siem(sample_log)

if __name__ == "__main__":
    create_guest_wifi()
```