Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Бакалаврська кваліфікаційна робота на тему:
«Засіб для тестування на проникнення Wi-Fi роутера»

Виконав: студент 4 курсу групи 1БС-21б
спеціальності 125 Кібербезпека
_____Владислав НЕМИРОВСЬКИЙ

Керівник: к. т. н., доцент каф. ЗІ
_____Віталій ЛУКІЧОВ
« 13 » червня 2025 р.

Рецензент: к. т. н. доцент каф. ПЗ
_____ Денис КАТЄЛЬНІКОВ
« 13 » червня 2025 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____Володимир ЛУЖЕЦЬКИЙ
« 13 » червня 2025 р

Вінниця ВНТУ – 2025 року

Міністерство освіти і науки України
Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти I (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри ЗІ,
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«24» березня 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Владиславу НЕМИРОВСЬКОМУ

1. Тема роботи: «Засіб для тестування на проникнення Wi-Fi роутера"» керівник роботи: Віталій ЛУКІЧОВ, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 20 березня 2025 року №97.
2. Строк подання студентом роботи 17 червня 2025 р.
3. Вихідні дані до роботи:
   - Галузь застосування: тестування безпеки Wi-Fi роутера та роутерів.
   - Метод тестування: динамічне тестування безпеки (DAST).
   - Використовувані інструменти: Aircrack-ng, Hashcat, Reaver).
   - Тип інструментів: відкриті (open-source) та комерційні.
4. Зміст текстової частини: Вступ. 1. Огляд сучасних методів та засобів для тестування на проникнення Wi-Fi роутера. 2. Метод для тестування на проникнення Wi-Fi роутера. 3. Програмний засіб для тестування на проникнення Wi-Fi роутера. Висновки. список використаних джерел.
5. Перелік ілюстративного матеріалу: блок-схема алгоритму головного екрану із переліком режимів тестування та вкладкою звіт, залежність ключових методів та канали взаємодії основних компонентів, блок-схема метода захоплення та перебір 4-way handshake, порівняльні характеристики методів атак на Wi-Fi роутера, головний екран Homescreen, приклад екран методу

Brute force Pin / Pixie dust / Krack / Flood з результатом Pin brute, графічний вигляд логів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	В. ЛУКІЧОВ, к.т.н., доц. каф.ЗІ	14.03.25	10.06.2025
2	В. ЛУКІЧОВ, к.т.н., доц. каф.ЗІ	14.05.25	10.06.2025
3	В. ЛУКІЧОВ, к.т.н., доц. каф.ЗІ	31.05.25	10.06.2025

7. Дата видачі завдання 21 березня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	24.03.25 – 30.03.25	
2	Аналіз інформаційних джерел за напрямком бакалаврської дипломної роботи	31.03.25 – 13.04.25	
3	Розробка рішень, моделей, алгоритмів	14.04.25 – 04.05.25	
4	Практична реалізація, моделювання, експериментування, результати	05.05.25 – 14.05.25	
5	Оформлення пояснювальної записки	15.05.25 – 18.05.25	
6	Попередній захист БДР	19.05.25 – 26.05.25	
7	Виправлення зауважень, підготовка ілюстративного матеріалу	27.05.25 – 30.05.25	
8	Перевірка на наявність текстових запозичень	31.05.25 – 10.06.25	
9	Представлення БДР до захисту, рецензування	11.06.25 – 13.06.25	
10	Захист БДР	17.06.25 – 20.06.25	

Студент _____ Владислав НЕМИРОВСЬКИЙ

Керівник роботи _____ Віталій ЛУКІЧОВ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 78 сторінок формату А4, містить 39 рисунків, 3 таблиць, список використаних джерел налічує 20 найменуваннь.

Бакалаврська кваліфікаційна робота присвячена розробці універсального кросплатформенного програмного засобу для динамічного тестування безпеки Wi-Fi роутерів на базі сучасних методик пентесту. Проведено аналіз сучасних підходів, інструментів і технологій для тестування на проникнення бездротових мереж, детально описано архітектуру та логіку роботи програмного засобу, розробленого за принципами модульності, автоматизації та прозорості.

У першому розділі подано огляд сучасних методів та інструментів для пентесту Wi-Fi роутера, класифіковано основні типи атак і порівняно популярні утиліти. Окрему увагу приділено вимогам до тестування та принципам стандартизованої звітності.

У другому розділі розроблено архітектуру програмного комплексу для автоматизованого тестування безпеки Wi-Fi роутерів. Описано, як головний екран управляє вибором режимів тестування і забезпечує взаємодію з підсистемами. Реалізовано централізоване сховище результатів для логування й формування рекомендацій. Окремо описано механізм експорту звітів для подальшого аудиту.

Третій розділ присвячено обґрунтуванню вибору технологій Flutter організації CI/CD, профілюванню продуктивності, а також функціональним можливостям додатку. Підсумовано результати практичного тестування на реальних сценаріях, наведено оцінку зручності та ефективності програми.

Ключові слова: тестування безпеки Wi-Fi, penetration testing, WPS brute force, Pixie Dust, handshake brute, KRACK, PMKID, Flood-атака, Flutter, WSL, автоматизація аудиту, звітність, Wi-Fi security, ReportStore, кросплатформенний додаток.

ABSTRACT

The bachelor's qualification work consists of 78 pages of A4 format, contains 39 figures, 3 tables, the list of used sources has 20 names.

The bachelor's qualification work is dedicated to the development of a universal cross-platform software tool for dynamic security testing of Wi-Fi routers based on modern pentesting techniques. A comprehensive analysis of modern approaches, tools and technologies for penetration testing of wireless networks has been carried out, the architecture and logic of the software complex, developed on the principles of modularity, automation and transparency, is described in detail.

The first chapter provides an overview of modern methods and tools for Wi-Fi network penetration testing, classifies the main types of attacks, and compares popular utilities. Particular attention is paid to testing requirements and the principles of standardized reporting.

The second chapter presents the architecture of the software system for automated Wi-Fi router security testing. It describes how the main screen manages the selection of testing modes and ensures interaction with subsystems. A centralized result storage for logging and generating recommendations has been implemented. The report export mechanism for further auditing is also described separately.

The third chapter is devoted to the rationale for the choice of technologies Flutter organization, performance profiling, as well as the functional capabilities of the application. The results of practical testing in real scenarios are summarized, and the usability and effectiveness of the program are evaluated.

Keywords: Wi-Fi security testing, penetration testing, WPS brute force, Pixie Dust, handshake brute, KRACK, PMKID, Flood attack, Flutter, WSL, audit automation, reporting, Wi-Fi security, ReportStore, cross-platform application.

ЗМІСТ

ВСТУП

У сучасних умовах інтенсивного розвитку інформаційних технологій питання забезпечення захисту бездротових мереж набуває особливої актуальності. Значна частина як корпоративного, так і домашнього інтернет-трафіку передається саме через Wi-Fi роутера, які часто виступають ціллю для зловмисників через широкий спектр потенційних вразливостей.

Актуальність обраної теми полягає у необхідності підвищення рівня захищеності бездротових мереж за рахунок впровадження ефективних, зручних та сучасних інструментів динамічного тестування. Практичне значення отриманих результатів визначається можливістю їхнього впровадження своєчасного виявлення й усунення вразливостей, підвищення загального рівня інформаційної безпеки та зменшення ризику несанкціонованого доступу до критичних ресурсів.

Метою даної бакалаврської роботи є підвищення рівня безпеки Wi-Fi роутера, що поєднує у собі найкращі практики тестування на проникнення, графічний інтерфейс, підтримку різних платформ та сучасних утиліт для аналізу бездротових мереж. Для досягнення поставленої мети у роботі проведено детальний аналіз сучасних підходів до пентесту, розроблено модульну архітектуру програмного комплексу, реалізовано автоматизовану систему обробки результатів тестування та формування рекомендацій, а також забезпечено можливість експорту звітів у форматі, придатному для аудиту.

Завдання роботи розробити та реалізувати автоматизоване тестування основних векторів атак на Wi-Fi роутер. Організувати формування зрозумілих звітів з описом виявлених вразливостей та конкретними рекомендаціями щодо їх усунення. Передбачити можливість розширення функціоналу для підтримки нових методів атак і адаптації під різні операційні системи.

Результати дослідження були апробовані через публікацію тез: Немировський В. О., Лукічов В. В.. Засіб для тестування на проникнення Wi-Fi роутера. Матеріали LIV науково-технічної конференції підрозділів ВНТУ, Вінниця, 24-27 березня 2025 р.

1 ОГЛЯД СУЧАСНИХ МЕТОДІВ ТА ЗАСОБІВ ДЛЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ WI-FI РОУТЕРА

1.1 Актуальність теми та дослідження

Тестування на проникнення є процес цілеспрямованого пошуку та експлуатації вразливостей у додатку, мережі або апаратному середовищі з метою оцінки реальної стійкості системи перед можливими атаками зловмисників [1]. На відміну від простого аудиту або сканування, при пентесті застосовуються ті ж методи та інструменти, що використовують хакери, проте дії проводяться законно та санкціоновано. Після кожного етапу дослідження формується докладний звіт, до якого включаються виявлені точки входу, сценарії їх експлуатації та рекомендації щодо усунення виявлених ризиків [2].

Причинами появи вразливостей можуть стати помилки проектування та реалізації програмного забезпечення, неправильне настроювання системних компонентів та сервісів, недостатній контроль над конфіденційними документами та даними, а також людський фактор – від недбалого обміну паролями до соціальної інженерії. Крім того, складність та масштабність сучасних IT-ландшафтів, залежність від мобільних та хмарних комунікацій, слабкий захист бездротових мереж та використання застарілого обладнання відкривають додаткові можливості для проникнення [3].

Регулярне проведення пентестів допомагає захистити критично важливі дані від здирників, витоків та компрометації. Великі організації, особливо зберігачі персональних та фінансових даних, нерідко вимагають підтвердження відповідності стандартам інформаційної безпеки (наприклад, PCI DSS) як на етапі розробки, а й у етапі експлуатації [4]. Результати тестування дозволяють не лише виявити «білі» вразливості до їх виявлення зловмисниками, а й оптимізувати безпекову політику, визначивши пріоритети впровадження контрзаходів.

У ході пентесту аналізуються всі компоненти інфраструктури: програмне забезпечення (операційні системи, веб- та мобільні програми, серверні сервіси), апаратні вузли (роутери, точки доступу, IoT-пристрою), мережа загалом, процеси

моніторингу та реагування, а також поведінка кінцевих користувачів. Це забезпечує комплексний погляд на ризики та допомагає сформувати всебічні рекомендації [5].

Існує кілька підходів до організації тестування на проникнення. При «чорній скриньці» фахівці діють без попередньої інформації про внутрішню структуру системи, обмежуючись зовнішніми даними (URL, публічним IP, доменними записами). «Біла скринька» передбачає повний доступ до аналітичної інформації – вихідного коду, схем мережі, налаштувань серверів; такий режим дозволяє глибоко перевірити внутрішні механізми безпеки та кодові шляхи. Проміжний формат «сірої скриньки» поєднує в собі частковий доступ та зовнішнє тестування .

Методологічно пентест включає ручне та автоматизоване тестування. Автоматизовані сканери допомагають швидко виявити відомі вразливості, але не всі проблеми піддаються таким інструментам [6]. Тому ручна перевірка життєво необхідна для пошуку логічних помилок, складних сценаріїв обходу захисту та атак соціальної інженерії. Насправді найчастіше використовують комбінований підхід, коли сканування доповнюється детальної ручної верифікацією .

Процес пентесту зазвичай зводиться до чотирьох етапів. Збір даних починається з OSINT-аналізу, вивчення публічної інформації, вихідного коду веб-сторінок та заголовків HTTP-відповідей. Потім слідує етап оцінки вразливостей, коли на основі отриманих відомостей тестувальники виявляють точки входу і можливі «дверцята» для атаки. На третьому етапі відбувається безпосередня експлуатація знайдених уразливостей: від підбору паролів та перехоплення автентифікаційних handshake-повідомлень до розгортання «підроблених» точок доступу та проведення DoS-атаки. Нарешті, результати узагальнюються та оформлюються у звіт з рекомендаціями щодо усунення прогалин безпеки, після чого цикл повторюється доти, доки всі критичні ризики не будуть нівельовані [7].

Типовими тест-кейсами при оцінці безпеки відносяться перевірка стійкості контактних форм веб-сайту до спам-атак, аналіз фільтрації вихідного та вхідного поштового трафіку, перевірка коректності налаштувань брандмауерів, пошук відкритих мережевих портів і небезпечних HTTP-методів (PUT, DELETE), контроль над зберіганням верифікація механізмів скидання пароля та блокування облікових

записів після багаторазових невдалих спроб входу. Особлива увага приділяється перевірці захисту від SQL-ін'єкцій, XSS-атак, XML-ін'єкцій, спуфінгу та переповнення буфера, а також оцінці стійкості до злому мереж Wi-Fi (перехоплення handshake та брутфорс, WPS-атаки, rogue AP-сценарії та KRACK-атаки) [8].

Таким чином, пентест охоплює не лише технічні, а й організаційні аспекти, пропонуючи збалансований набір методів та практик для забезпечення надійного захисту інформаційної системи.

1.2 Класифікація та характеристика методик тестування

Загальна схема роботи обміну даними між пристроєм та точкою доступа і її перехват представлена на рисунку 1.1.

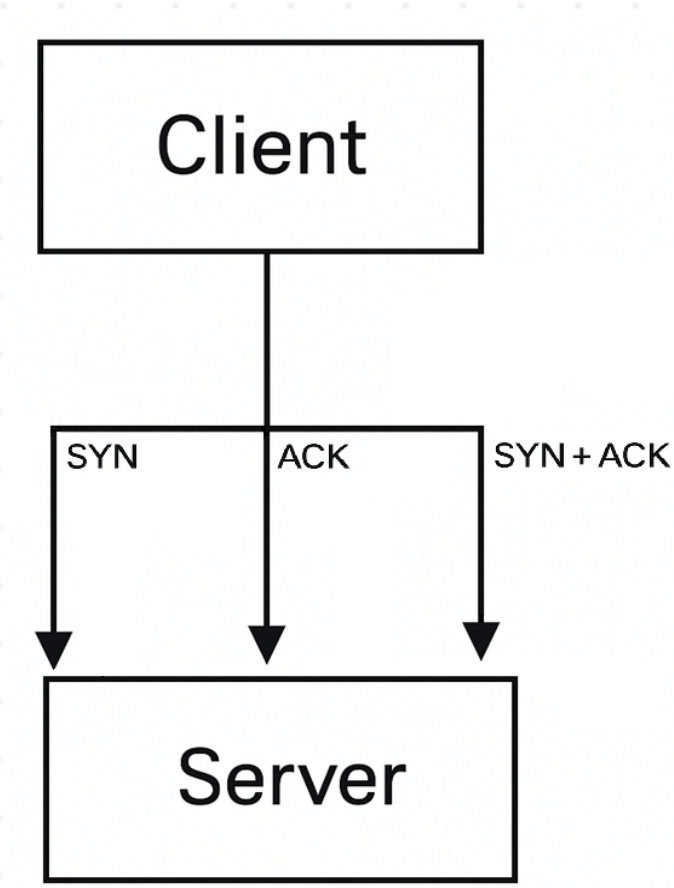


Рисунок 1.1 – Загальна схема роботи обміну даними між пристроєм та точкою

Після запуска утиліти “airmon-ng” у списку пристроїв має з'явитися підключений адаптер, графічно представлено на рисунку 1.2.

Рисунок 1.2 – Результат виведення списку пристроїв

Для здійснення підбору WPS PIN-коду використовується інструмент Reaver. На першому етапі необхідно завантажити актуальну версію програми з відкритого репозиторію, після чого її слід скомпілювати та встановити в систему. Процес інсталяції включає стандартні кроки: перехід до каталогу з вихідним кодом, налаштування параметрів компіляції, безпосередньо компіляція та завершальна інсталяція.

Після завершення встановлення потрібно підготувати бездротовий адаптер до роботи, переключивши його у режим моніторингу. Це дозволяє адаптеру приймати всі пакети, що передаються у бездротовій мережі, незалежно від того, кому вони адресовані [9]. Саме в цьому режимі можливо перехоплювати службові пакети та взаємодіяти з точками доступу, що підтримують WPS, для подальшого підбору PIN-коду за допомогою Reaver, графічно представлено на рисунку 1.3 .

Рисунок 1.3 – Результат виведення списку пасивного прослуховування мережевого трафіку

Таким чином, після активації моніторингового режиму ми отримали віртуальний інтерфейс, що дозволяє здійснювати пасивне прослуховування мережевого трафіку. Наступним кроком є виявлення бездротових точок доступу в

зоні дії. Для цього застосовується спеціальна утиліта, яка сканує ефір і відображає список усіх доступних Wi-Fi роутера, включно з їхніми основними параметрами, такими як назва мережі (SSID), ідентифікатор (BSSID), рівень сигналу, канал роботи тощо. Варто зазначити, що використання адаптера типу Alfa AWUS036H забезпечує розширене покриття та дозволяє фіксувати сигнали від більшої кількості точок доступу, графічно представлено на рисунку 1.4.

Рисунок 1.4 – Результат виведення списку усіх доступних Wi-Fi роутера, включно з їхніми основними параметрами

Після виявлення доступних Wi-Fi роутера наступним етапом є перевірка наявності активованого режиму WPS на кожній із точок доступу. Це здійснюється за допомогою спеціалізованої утиліти, яка сканує ефір і виводить список мереж з активною підтримкою WPS.

Для цього використовуються такі інструменти, як Reaver, wash (що входить до складу Aircrack-ng), а також вбудовані сканери деяких Wi-Fi аналізаторів. У процесі сканування утиліта визначає, чи підтримує точка доступу функцію WPS, її версію, а також чи використовується PIN-метод або кнопка (PBC). Виявлення активного WPS дозволяє оперативно відфільтрувати вразливі пристрої та сформувати перелік цілей для подальших атак методом перебору PIN-коду або використання вразливостей протоколу Pixie Dust. Якщо серед знайдених мереж є така, що становить інтерес, можна переходити безпосередньо до етапу підбору PIN-коду, графічно представлено на рисунку 1.5.

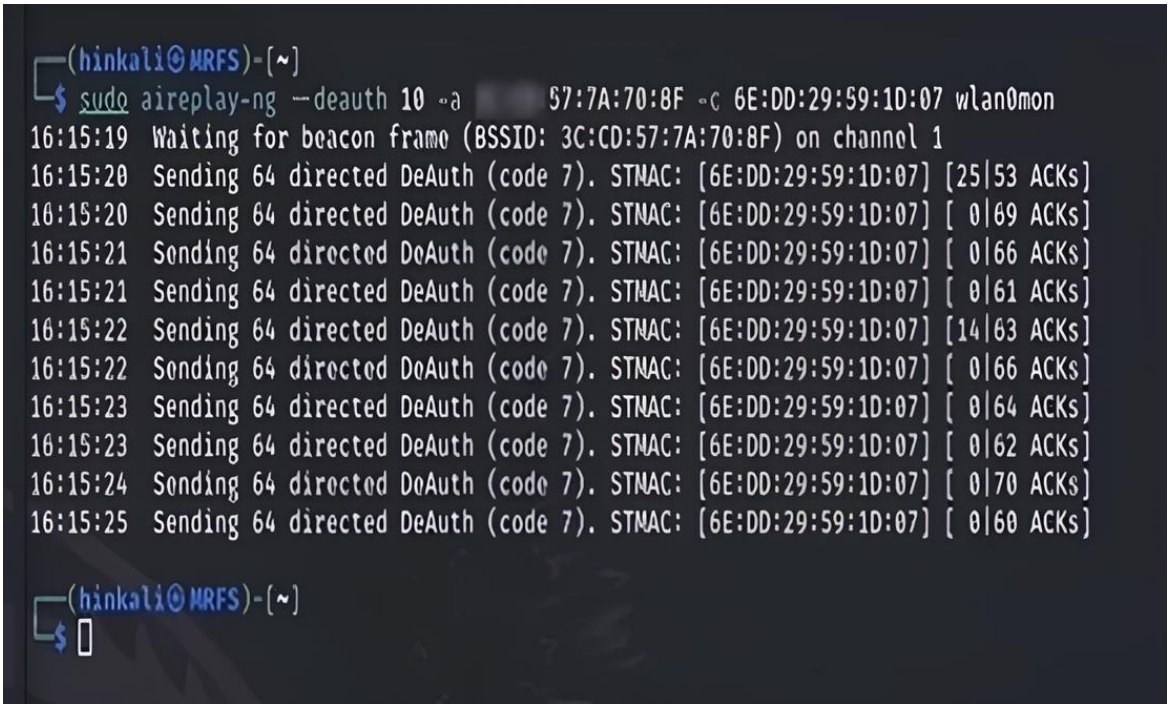


Рисунок 1.5 – Процес перебору можливих PIN-кодів

У процесі роботи утиліта здійснює перебір можливих PIN-кодів, які використовуються в механізмі WPS, і в разі успішного підбору повертає як сам PIN-код, так і фактичний пароль від Wi-Fi роутераі [10]. Середній час, необхідний для повного перебору, зазвичай становить близько п’яти годин, однак цей показник може змінюватися залежно від стабільності з’єднання, якості сигналу та реакції точки доступу. Успішне завершення процесу свідчить про вразливість мережі до WPS-атак і дозволяє отримати несанкціонований доступ до Wi-Fi.

Після завершення збору пакетів у процесі прослуховування бездротової мережі, зокрема під час перехоплення процесу аутентифікації між клієнтом і точкою доступу, формується набір файлів, серед яких основним є .cap – файл захопленого трафіку. Він містить усі передані пакети, включно з handshake-повідомленнями, необхідними для подальшого аналізу. У подальшому цей файл використовується для атаки типу brute-force або словникової атаки, мета якої – відновити ключ доступу до мережі (пароль). На прикладі зображення видно, що після запуску інструменту для підбору ключа (наприклад, aircrack-ng або подібного), система проаналізувала сотні тисяч ключів, і вже на початковому етапі (менше ніж за 2 хвилини) вдалося знайти правильний пароль [10].

На екрані відображено повідомлення KEY FOUND, що супроводжується знайденим паролем – 05SuperPassWysiWyg!@#. Крім того, виведено технічні дані: Master Key, Transient Key, та EAPOL HMAC, які підтверджують успішність атаки та можуть бути використані для додаткового криптографічного аналізу, графічно представлено на рисунку 1.6.

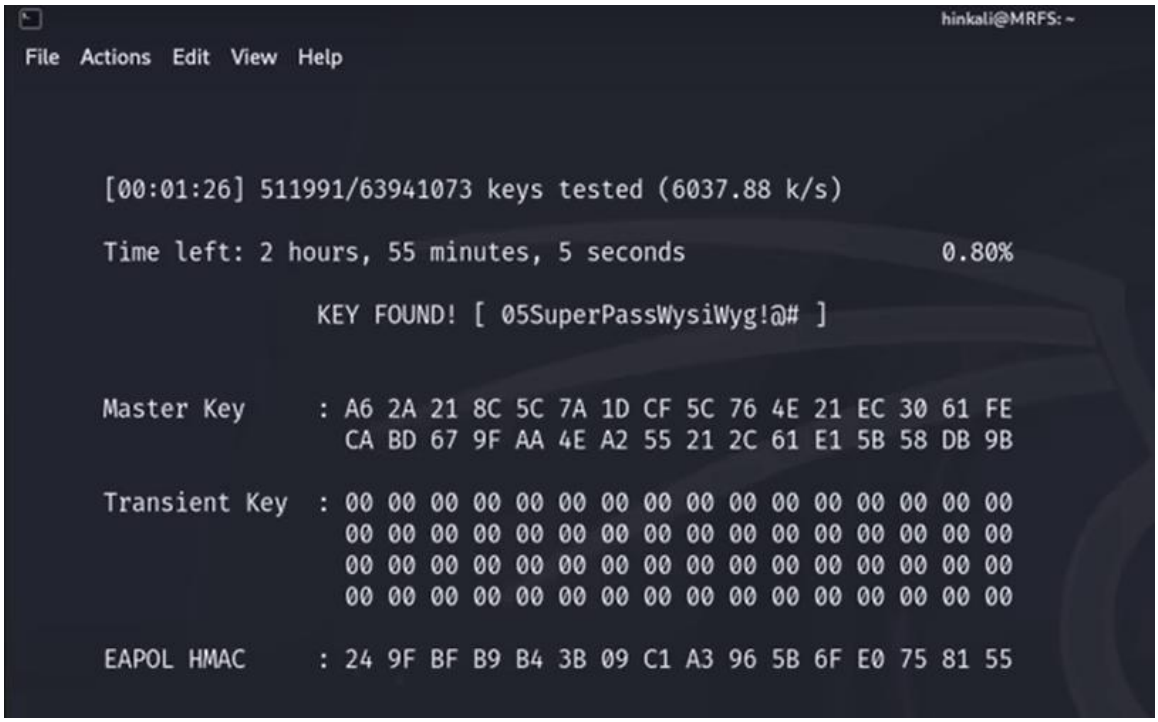


Рисунок 1.6 – Результат проведення успішної атаки

Таким чином, після завершення атаки з використанням захопленого handshake'у користувач отримує повний доступ до захищеної Wi-Fi роутераі. Це підтверджує, наскільки важливо використовувати складні паролі, відключати WPS, а також вживати додаткових заходів безпеки у конфігурації бездротових мереж.

Атака KARMA базується на використанні вразливості клієнтів, які автоматично підключаються до знайомих бездротових мереж [11]. У такій конфігурації клієнт періодично надсилає зондуючі фрейми (Probe Request), намагаючись знайти відомі йому точки доступу, з якими він раніше з’єднувався. На рисунку показано послідовність взаємодії між жертвою (клієнтом) та атакуючою точкою доступу, яка імітує справжню мережу. Якщо точка доступу відповідає типу

"Відкрита", а у клієнта увімкнено AutoConnect, і він транслює свій список бажаних мереж (Preferred Network List, PNL), зловмисник може скористатися цим [11].

Атака реалізується наступним чином: клієнт (жертва) передає зондуючий фрейм для пошуку мережі зі свого списку бажаних ESSID. Шахрайська точка доступу, що емулює відповідну мережу, відповідає на зондуючий фрейм. Клієнт відправляє запит на відкриту аутентифікацію, який негайно приймається. Відбувається підключення за стандартом 802.11, що завершує процес з'єднання, клієнт надсилає кілька зондуючих фреймів, а атакуюча точка доступу перехоплює та вгадує правильний фрейм-маяк, графічно схему представлено на рисунку 1.7.

Рисунок 1.7 – Інфографіка результату проведення успішної атаки

Після відповіді на зондуючий запит жертва здійснює підключення без потреби в паролі чи подальшій перевірці. Це дозволяє зловмиснику перехоплювати трафік, виконувати атаки типу «людина посередині» (MitM) або впроваджувати фішингові сторінки. Більшість сучасних операційних систем не вразливі до атак KARMA, оскільки вони не надсилають списки бажаних мереж, але іноді ви можете

зіткнутися з вразливою системою в старих пристроях IoT або принтерах. Якщо пристрій коли-небудь підключався до відкритої мережі, що приховує його ESSID, він, безумовно, вразливий до атаки KARMA. Причина в тому, що єдиним способом підключитися до відкритих мереж, що приховують свій ESSID, є відправка їм прямого запиту, в цьому випадку створюються всі умови, необхідні для KARMA-атак.

Після викриття атаки KARMA більшість операційних систем перестали безпосередньо зондувати точки доступу; замість цього вони використовують тільки пасивне спостереження, при якому пристрій прослуховує відомий ESSID з мережі. Така поведінка повністю виключає всі випадки атак KARMA. Порівняльна характеристика методів атак на Wi-Fi роутераі, представлено у таблиці 1.1.

Таблиця 1.1 - Порівняльна характеристика методів атак на Wi-Fi роутера

Метод атаки	Необхідне обладнання	Час виконання	Технічна складність	Вразливі цілі	Імовірність успіху	Виявлення користувачем
Reaver (WPS-атака)	Wi-Fi адаптер з моніторингом	4–6 годин	Середня	Точки з увімкненим WPS	Висока (при WPS)	Низька
Brute-force по .cap	Адаптер + handshake	10 хв – ∞	Висока (залежить від словника)	Будь-яка WPA/WPA2 мережа з handshake	Середня	Низька
Wifiphisher (соц. інж.)	2 адаптери + шаблон сторінки	5–15 хв	Висока	Користувачі відкритих мереж	Висока	Середня
KARMA	Rogue точка + моніторинг	Миттєва	Низька	Старі пристрої (IoT, принтери)	Низька	Висока
Known Beacons	Rogue точка + словник SSID	5–20 хв	Середня	Пристрої з AutoConnect	Середня	Середня

Атака Known Beacons обходить цю функцію безпеки, використовуючи той факт, що багато операційних систем за замовчуванням включають прапорець AutoConnect [12]. Оскільки точки доступу часто мають дуже поширені назви, зловмисник може вгадати ESSID відкритої мережі в списку бажаних мереж пристрою. Потім він обманом змушує цей пристрій автоматично підключатися до

точки доступу, контрольованої зловмисником. У більш складній версії атаки зловмисник може використовувати словник поширених ESSID, таких як Guest, FREE Wi-Fi тощо, до яких жертва, ймовірно, підключалася в минулому. Це дуже схоже на спробу отримати несанкціонований доступ до службового облікового запису шляхом простого перебору імені користувача, коли пароль не потрібен: досить проста, але ефективна атака.

Точка доступу зловмисника починається з передачі кількох *кадрів маяка*, типу кадру керування, який містить усю інформацію про мережу. Цей кадр регулярно транслюється, щоб повідомити пристроям про наявність мережі. Якщо жертва має інформацію про цю мережу в списку бажаних мереж (оскільки жертва вже підключалася до цієї мережі в минулому) і якщо точки доступу зловмисника та жертви мають тип Відкритий, Жертва надішле тестовий запит і підключиться до нього. Перш ніж ми зможемо виконати цю атаку, нам потрібно налаштувати наші пристрої. На деяких пристроях можна змінити прапорець автопідключення. Розташування цього параметра відрізняється від пристрою до пристрою, але зазвичай його можна знайти в налаштуваннях Wi-Fi, у такому параметрі як Автоматичне повторне підключення. Переконаймося, що цю опцію увімкнено [12].

Потім налаштуємо відкриту точку доступу під назвою my_essid. Ми зробили це за допомогою портативної точки доступу TP-Link. Після налаштування підключемо цільовий пристрій до мережі my_essid. При активації модуля деаутентифікації Wifiphisher надсилає спеціальні керуючі кадри (deauth frames) легітимному AP, примушуючи клієнтський пристрій розірвати з’єднання та знову підключитися – цього разу до точки доступу, контрольованої зловмисником. Далі встановимоWifiphisher , шахрайську платформу точки доступу, яка часто використовується для оцінки мережевої безпеки зображено на рисунку 1.8.

Рисунок 1.8 – Приклад екрану налаштування відкритої точки доступу

Для моделювання атаки соціальної інженерії з метою перехоплення облікових даних користувачів використовується утиліта Wifiphisher, яка дозволяє створити фальшиву точку доступу та спровокувати підключення клієнтів до неї. Важливо наголосити, що в рамках дослідження створюється спеціальна тестова мережа з ідентифікатором my_essid, що дозволяє уникнути впливу на сторонніх користувачів і не порушувати чинне законодавство [13].

Після запуску інструмент автоматично визначає часовий пояс, вибирає діапазон каналів і налаштовує мережеві інтерфейси. Один з інтерфейсів використовується для здійснення деаутентифікаційної атаки, мета якої – відключити користувачів від справжньої мережі, а інший – для створення шкідливої (rogue) точки доступу, що імітує легітимну. У процесі запуску також виконується зміна MAC-адрес для маскування пристрою, зупиняється процес wpa_supplicant, очищуються попередні DHCP-лізи, налаштовується IP-маршрутизація та брандмауер. Після цього Wifiphisher активує фальшиву точку доступу та завантажує одну з заздалегідь підготовлених шаблонних сторінок – у цьому випадку сторінку для авторизації через OAuth, яка є частиною сценарію соціальної інженерії, потрібно переглянути список кращих мереж для пристрою, який ми хочемо обдурити в цьому прикладі.

Наприклад, представлений список кращих мереж на пристрої Samsung Galaxy S8+. У ньому збережено дві мережі; перша з них, FreeAirportWiFi, має назву, що легко вгадується. Приклад списку кращих мереж на пристрої зображено на рисунку 1.9.

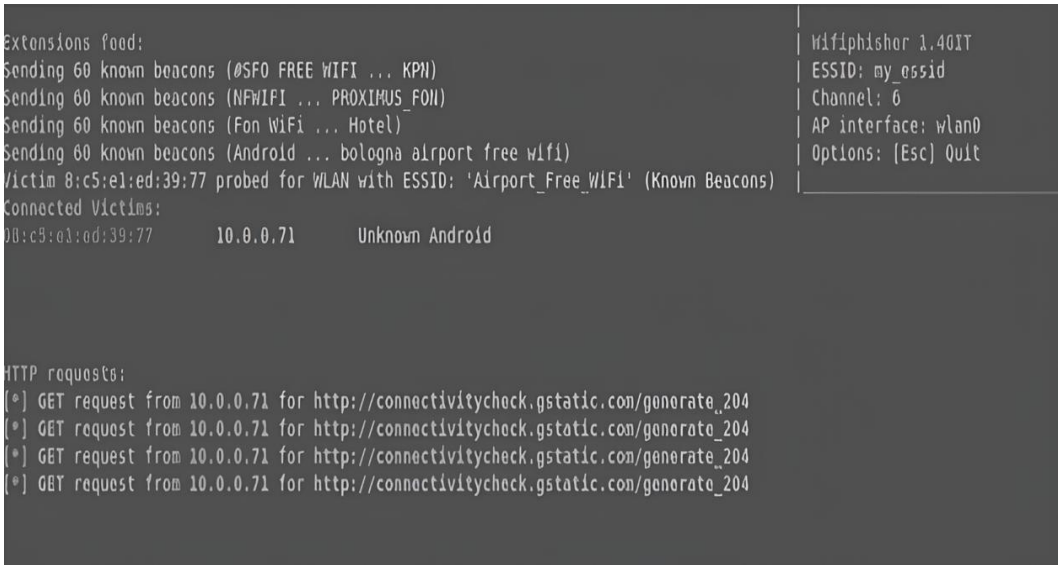


Рисунок 1.9 – Приклад списку кращих мереж на пристрої

Звичайно, після того, як ми здійснимо атаку, пристрій повинен відключитися від своєї поточної мережі і підключитися до шкідливої, підробленої мережі. З цього моменту зловмисник може діяти як «людина посередині», стежачи за дорожнім рухом жертви або навіть заважаючи йому.

1.3 Порівняльний аналіз інструментів сканування та аналоги

Протестувати бездротові мережі можна за допомогою професійних комерційних розробок, створених для різних операційних систем. Наряду з ними пропонуються безкоштовні варіанти, деякі з них є офіційними простими версіями платних програм. Основні завдання даних програм – діагностика Wi-Fi роутера і виявлення вільних каналів. 1) Acrylic WiFi Home для Windows – розробка належить Tarlogic Security [14]. Це безкоштовна версія комерційної програм, вона дозволяє детально вивчити бездротове середовище за допомогою зручної інфографіки та багаторівневого аналізу параметрів мереж. У головному вікні відображається

список усіх знайдених SSID, MAC-адреси, рівень сигналу (RSSI), типи протоколів (802.11 a/b/g/n/ac), швидкість з'єднання, поточні канали, рівень захисту (WEP/WPA/WPA2/WPS), виробник обладнання, а також інформація про першу й останню появу точки доступу. Інтерфейс програми, її структуру та функціонал наведено на рисунку 1.10.

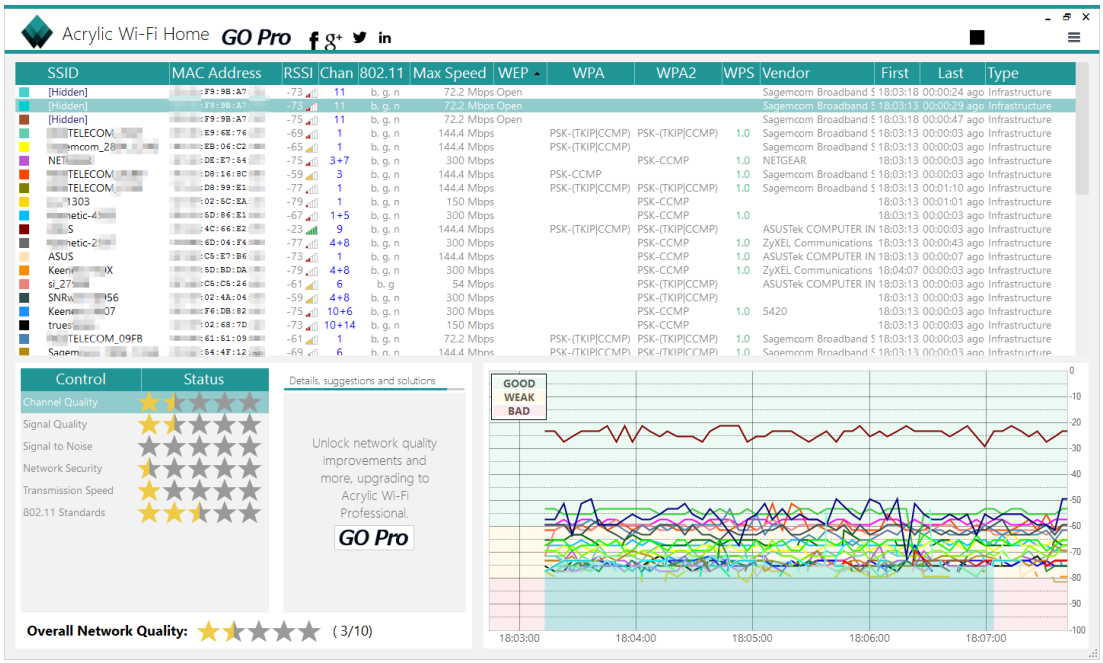


Рисунок 1.10 – Приклад екрану інтерфейсу Acrylic Wi-Fi Home

Серед додаткових можливостей – побудова графіків зміни потужності сигналу для кожної мережі в реальному часі, оцінка якості каналів і автоматичне визначення оптимальних частот для підключення. Програма надає оцінку загальної якості мережі за кількома критеріями: якість каналу, співвідношення сигнал/шум, рівень безпеки, швидкість передачі та відповідність стандартам. Наприклад, для кожної мережі відображається рейтинг за зірками, що дозволяє швидко порівняти доступні точки доступу.

Вбудовані плагіни допомагають ідентифікувати потенційно незахищені точки доступу та несанкціоновані підключення, а також виділяють підозрілі SSID. В інтерфейсі відображаються як видимі, так і приховані мережі. Окрім цього, Acrylic WiFi Home може вести журнал виявлених мереж та будувати візуалізацію використання частотних діапазонів [14]. У безкоштовній версії частина функцій

обмежена, зокрема щодо збереження детальних звітів і експорту даних, однак основний функціонал для діагностики та моніторингу доступний повністю.

Відображення параметрів на екрані налаштовується, є можливість використання короткого і розширеного режиму. Існуючим недоліком безкоштовної версії є незручність збереження даних. Можна скопіювати тільки одну строку в буфер і вставити в документ. Acrylic WiFi Home оптимальний для простої діагностики бездротових мереж.

AirScout Live для Android це сканер AirScout Live створений компанією Greenlee [15]. Перетворює смартфон в аналізатор мережі. Комерційна версія продукту працює в напіврежимах. Безкоштовна версія – в чотирьох із них. При цьому підтримується платна версія, крім Android, операційні системи Windows і iOS. Додаток показує точки доступу з рівнями сигналів, протоколами безпеки та апаратними ресурсами. Це потужний мобільний сканер мереж, розроблений компанією Greenlee, який перетворює смартфон у професійний аналізатор Wi-Fi роутера. Він підтримує як безкоштовну, так і платну версії, працюючи на Android, Windows та iOS. Додаток відображає точки доступу разом із рівнями сигналу, типами безпеки та апаратними характеристиками. Визначає більш вільний канал, вимірює потужність сигналу, показує місця з незадовільним рівнем сигналу. Аналізує параметри використання каналів на частотах 2,4 і 5 ГГц, приклад можна побачити на рисунку 1.11.

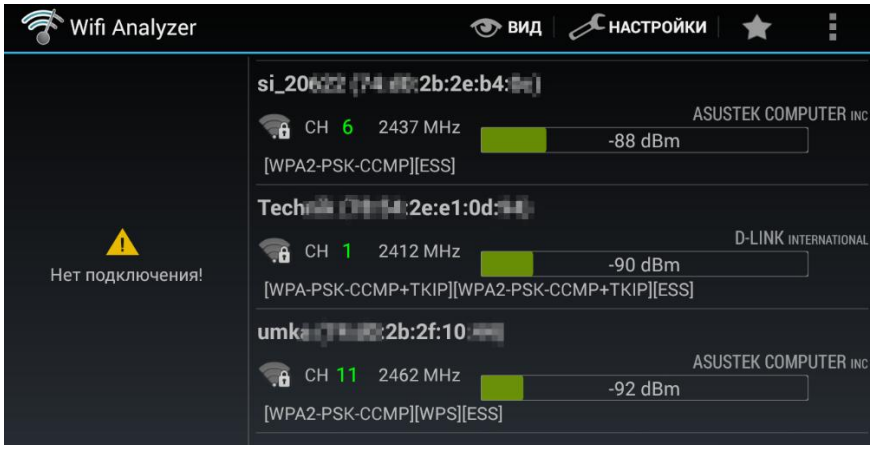


Рисунок 1.11 – Приклад екрану AirScout Live аналізу використання каналів на частотах

Визначає найбільш вільні канали, вимірює потужність сигналу, а також показує зони з низьким рівнем покриття. AirScout Live аналізує параметри використання каналів у діапазонах 2,4 та 5 ГГц, допомагаючи оптимізувати роботу мережі та вибрати найкращі частоти для підключення. До того ж встановлює причини поміч. Допомагає при виборі місця для точки доступу. Скріншоти мереж зберігаються н а локальному диску або вивантажуються в хмарі, інтерфейс програмного засобу та його методик представлено на рисунку 1.12.

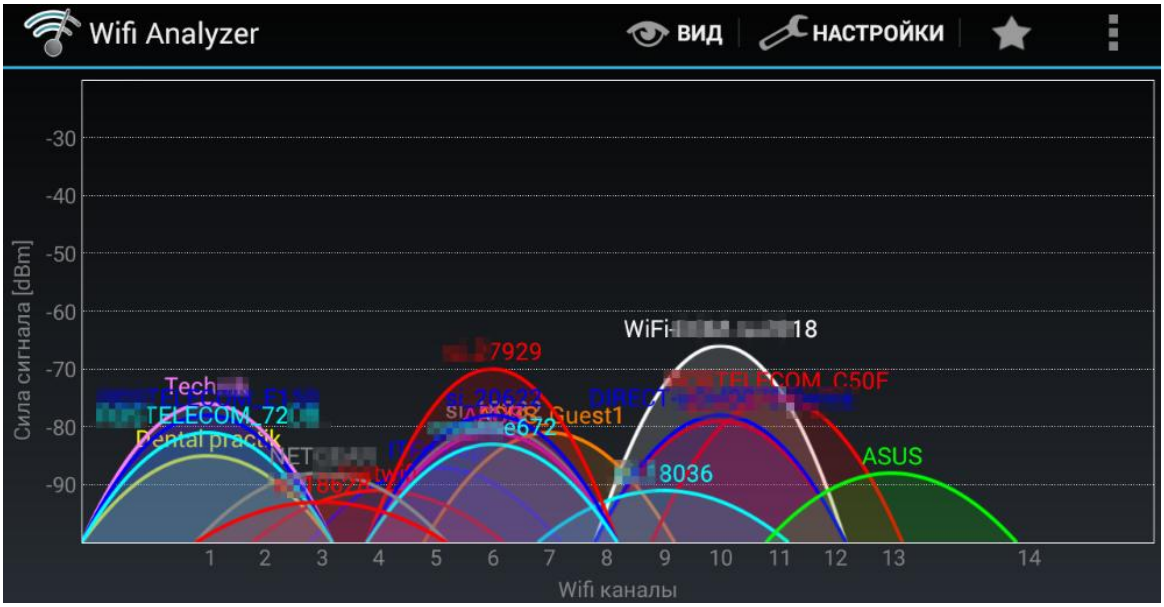


Рисунок 1.12 – Приклад екрану з аналізу точок та сигналу

Інтуїтивно зрозумілий інтерфейс користувача відрізняється простотою, не вимагає додаткових інструкцій. Основна перевага – портативність і висока швидкість роботи. Підходить для домашньої або невеликої офісної мережі.

Каїн і Авель для Windows [15]. Багатофункціональне рішення він відновлює і взламує паролі, перехоплює і аналізує мережевий трафік, в тому числі бездротової. Має сходні характеристики з Acrylic WiFi Home в захопленні та обробці мережевого трафіку. Оснащений інтерфейсом старого зразка з іконками нагорі для запуску утиліт і віконними вкладками для функцій.

Мережевий трафік відкриває вкладку Wireless. Тут показуються ідентифікатори SSID, характеристики сигналів, список підключених клієнтів і докладна інформація про них. Крім цього відображається кількість пакетів,

векторів ініціалізації протоколу безпеки та запитів ARP. Приховані ідентифікатори відображаються в графічному вигляді. Графічне представлення програмного засобу та його функціоналу зображено на рисунку 1.13.

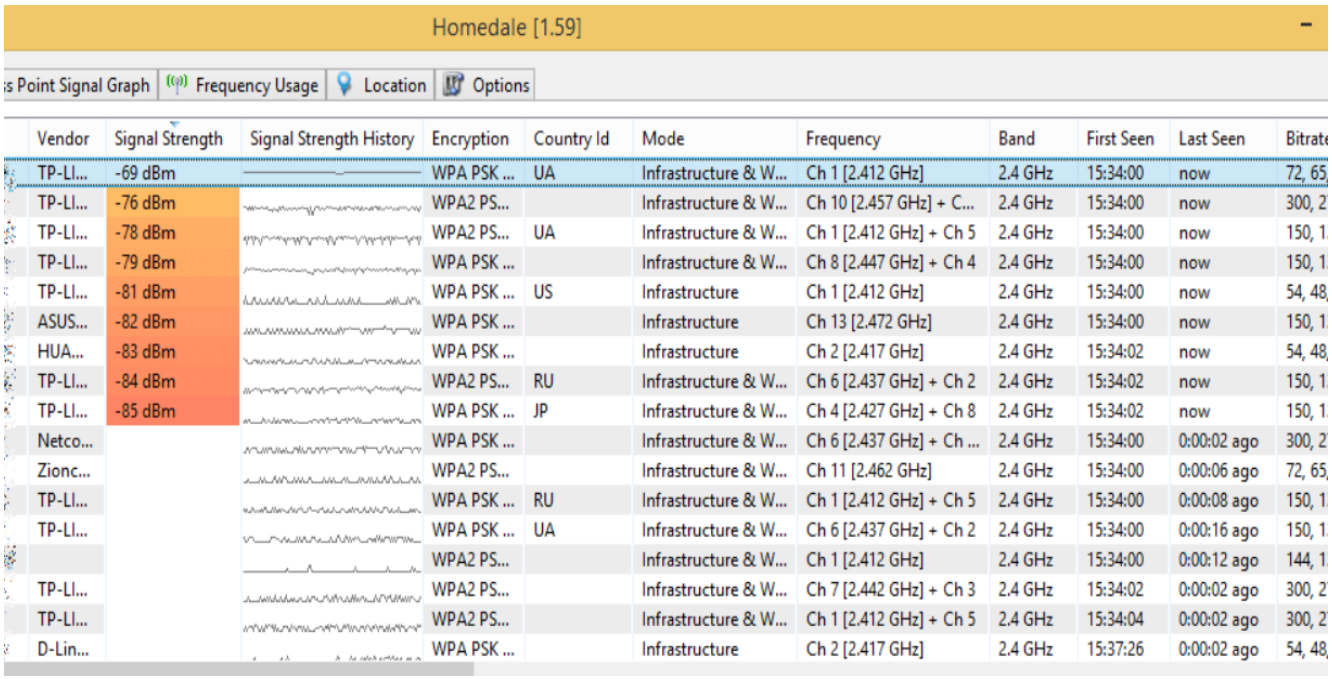


Рисунок 1.13 – Приклад екрану з аналізу точок та захисту

Більша частина даних зберігається та експортується в текстовий файл. Основним недоліком є відсутність видачі інформації у вигляді наглядних графіків, неможливість виявлення точок доступу до протоколу 802.11ac, визначення більш широких каналів. Підходить для системного адміністратора продовжнього рівня, так як дозволяє відновлювати паролі, втрачені облікові дані, аналізувати дані VoIP в мережі, відстежувати маршрутизацію пакетів. Homedale для Windows і macOS – це легка портативна утиліта для базового моніторингу Wi-Fi роутера, яка не потребує встановлення і має інтуїтивно зрозумілий інтерфейс. Програма дозволяє швидко переглядати всі доступні точки доступу з інформацією про SSID, MAC-адреси, рівень сигналу, протоколи шифрування, частоти, швидкість з'єднання, та виробника пристрою. Особливістю є вкладки для аналізу роботи адаптерів, побудови графіка сили сигналу та відстеження використання частотного діапазону.

Homedale для Windows і macOS ця утиліта з інтерфейсом командного рядка не вимагає встановлення [16]. Показує базову інформацію про WiFi-мережах і

сигналах, підтримує геолокацію. Діалогове вікно має вкладки «Адаптери» (список мережевих адаптерів з IP-шлюзами і MAC-адресами), «Точки доступу» (швидкість передачі даних, номери використовуваних каналів, можливість збереження та експорту міток для мережевих інтерфейсів), «Графік сигналу точок доступу» (відмінні значення рівня сигналу в дБм) і «Частота використання» (незалежність частоти від рівня сигналу).

WiFi сканер LizardSystems для Windows [17], LizardSystems надає для особистого комерційного використання версію свого продукту абсолютно безкоштовно. Відрізняється чудовим функціоналом для аналізу та збереження даних у звітах. Інтерфейс додатків простий і понятний. Вкладки містять дані про мережеві інтерфейси з підключеними клієнтами та специфікаціями протоколу 802.11, про поточне бездротове підключення з графіками та статистикою для пакетів. Детальні звіти можна експортувати в текстовий файл. Вивід даних налаштовується за допомогою фільтрів.

Додаток Inssider аналізує бездротові мережі та завантаженість каналів [18]. Середи запитуваних характеристик – сила сигналу, швидкість мережі, ідентифікатор, захист WiFi-сеті, MAC-адреса точкового доступу, пристрої виробника та інше. Підтримує бездротові мережі з частотами 2,4 і 5 ГГц. Для використання цієї програми знадобиться комп'ютер з адаптером WiFi. Завантажуємо та встановлюємо з офіційного сайту версії додатків для своєї операційної системи. Підтримує Windows, у разі необхідності встановлення пакета NET Framework про це повідомляє діалогове вікно при установці. Разом з тим, варто зазначити, що Inssider іноді потребує прав адміністратора для коректного доступу до низькорівневих API адаптера, а також підтримує наразі лише стандарти 802.11a/b/g/n, що може бути обмеженням при роботі з сучасними точками доступу на 802.11ac/ax. Утиліта не має вбудованих засобів для автоматизованого тестування безпеки або зламу мереж, тому її варто поєднувати з іншими інструментами, якщо потрібна повна перевірка стійкості захисту. Незважаючи на це, простий інтерфейс і можливість швидкого аналізу спектру роблять Inssider корисним доповненням до арсеналу системного адміністратора та фахівця з мережевої діагностики., інтерфейс програмного засобу та його методик представлено на рисунку 1.14.

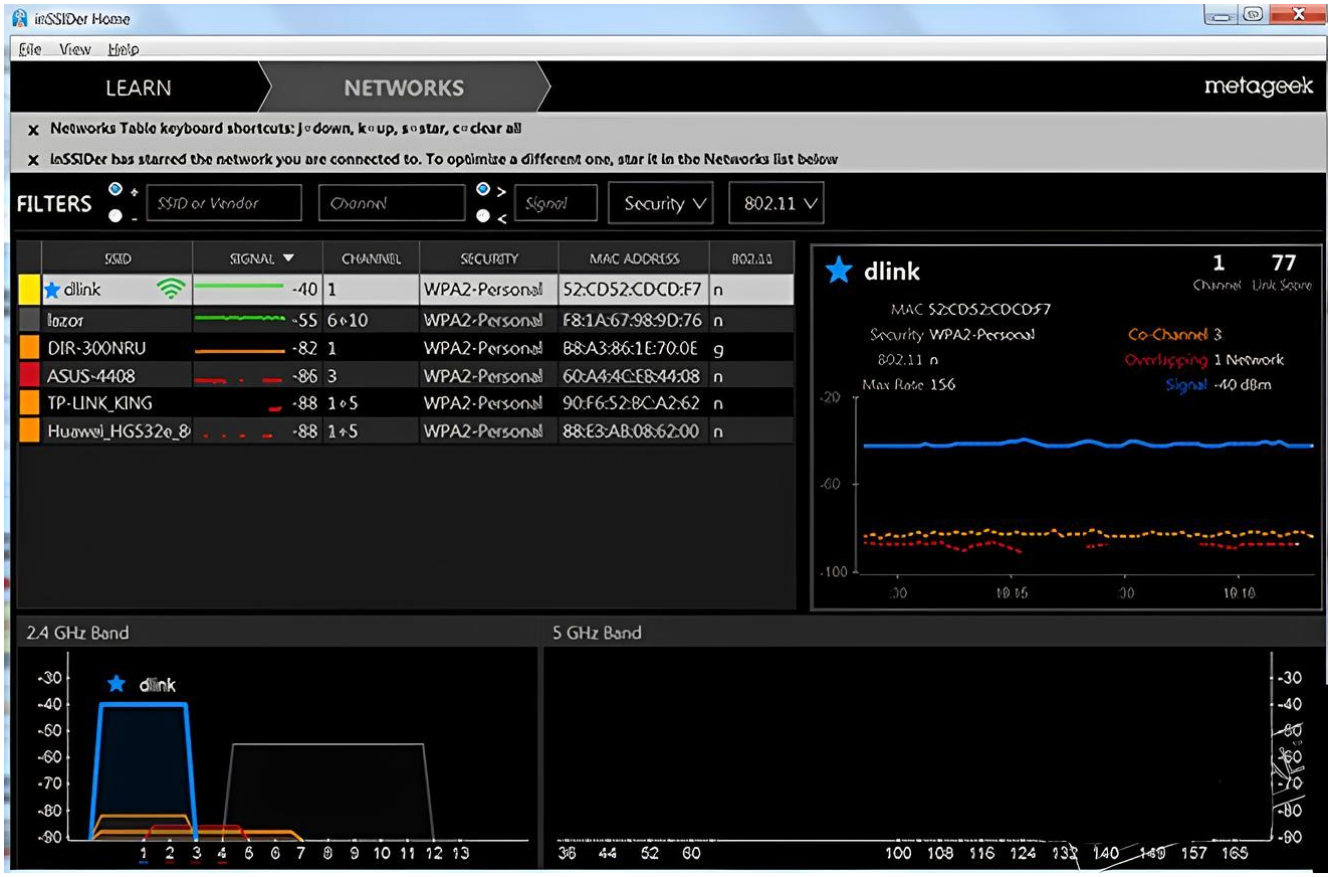


Рисунок 1.14 – Приклад екрану точок та захисту

Назва мережевого адаптера розташована у верхній частині вікна. Налаштувати або перервати сканування можна, нажавши кнопку Stop/Start. Фільтрація влаштовується в спеціальній області, можна додати поля або виключити їх. Таким чином можна фільтрувати по виробникам, діапазонам частот, каналам, типам мережі, ступеням захисту, дані виводяться у вигляді інформаційної та графічної таблиці. Непотрібні стовпці можна відключити. Крім того, є часовий графік руху та пересічення радіохвиль. При поганому сигналі можна змінити номер каналу в маршрутизаторі Wi- Fi, зайшовши в налаштування бездротової мережі. Під час роботи бездротової мережі можливо зменшення швидкості, поміхи на лінії, тимчасова відсутність сигналу [17]. Використання WiFi аналізатора допоможе поліпшити роботу мережі. Додаток проаналізує причини торможення, відключає канал зв'язку без помех. Завантажити його можна в Google Play для Android і на офіційному сайті Microsoft для Windows.

Після встановлення програми на смартфоні у діалоговому вікні з'являтимуться дві вкладки з інфографікою. На першій відображається шкала якості подачі сигналу каналами. Друга вкладка надасть інформацію про завантаження каналів. Для зручності, канали позначені різними кольорами. Потрібні налаштування можна створити, натиснувши на іконку у правому верхньому кутку вікна, переваги таких програм представлено в таблиці 1.2.

Таблиця 1.2 - Переваги та опис програм тестування

Переваги	Опис
Висока чутливість	Ловить навіть найслабший сигнал Wi-Fi, що дозволяє виявити віддалені або екрановані мережі.
Економія ресурсів	Працює у пасивному режимі, не навантажує процесор та оперативну пам’ять.
Зручний індикатор сили сигналу	Відображає рівень сигналу у вигляді спідометра для швидкої оцінки якості з’єднання.
Звукове супроводження	Попереджає про зміни в якості звуком, що покращує користувацький досвід.
Рекомендації щодо вирішення проблем	Підказує, які налаштування треба змінити для покращення покриття або продуктивності.
Виявлення нелегальних пристроїв	Автоматично позначає підозрілі точки доступу та пристрої поза офіційним списком.
Автоматичне оновлення даних	Інформація про мережу оновлюється в реальному часі без ручного перезапуску сканера.
Підтримка 2,4 GHz і 5 GHz	Забезпечує аналіз обох популярних діапазонів для повного огляду мережі.
Експорт звітів	Дає змогу зберігати результати сканування у форматах CSV, JSON або текстових файлах.
Побудова heatmap-покриття	Створює термокарту покриття для візуального аналізу мертвих зон і зон перешкод.
Портативність	Є версії для Windows, macOS, Linux та Android – можна працювати на будь-якому пристрої.

Застарілі програми моніторингу - це програми, які не сумісні з сучасними операційними системами або не підтримують Wi-Fi стандарти IEEE 802.11n/ac, із застарілим інтерфейсом. Як приклад можна навести Network Stumbler, аналізатор для Windows [19]. Підтримує стандарти 802.11 a/b/g, частотний діапазон 2.4, не підтримує 5 ГГц. Сумісний з ОС Windows до версії XP включно. Wireless Network Watcher – стандартний аналайзер бездротової мережі з можливістю експорту даних

у HTML. Застарілим названо через необхідність підключення до мережі для її аналізу. Просканувати мережу wi-fi та визначити найбільш вільний канал, для виконання цих завдань є спеціальні програми-аналайзери, деякі з них поширюються безкоштовно. Саме їм присвячено цю статтю. Завантажуємо якийсь аналізатор Wi-Fi роутера - вибираємо найшвидший і високочастотний канал. Як варіант, можна завантажити програму для WiFi Lyzer. Додаток має звичний функціонал для аналізу трафіку, відрізняється тонким налаштуванням точок доступу і різними способами відображення даних – у вигляді таблиць, графіків, осцилограм, списків з багатою палітрою кольорів. Для того щоб змінити автоматичний вибір каналу на роутері D-Link можна зайти в налаштування роутера, наберіть в адресному рядку браузера 192.168.0.1. За промовчанням пароль admin, ім'я користувача admin. На вкладці Wi-Fi у полі Регіон вибираємо країну проживання, у полі Канал зупиняємо свій вибір на каналі з найкращим сигналом, натискаємо «Застосувати».

У першому розділі було здійснено комплексний аналіз сучасних підходів до тестування на проникнення Wi-Fi роутера і класифікацію ключових методів атак, що дозволило виявити сильні та слабкі сторони існуючих рішень. З’ясовано, що кожен метод потребує специфічного обладнання й інструментарію, а комбіноване застосування автоматизованих сканерів і ручної валідації забезпечує найкраще покриття вразливостей. Проведений огляд показав, що безкоштовні інструменти є цілком придатними для первинного сканування, тоді як платні продукти розширюють можливості глибинного аналізу.

Також було підтверджено необхідність єдиної структури звітності для забезпечення прозорості й відтворюваності результатів пентесту та обґрунтовано важливість розробки шкали ризиків для пріоритезації контрзаходів.

1.4 Постановка задачі

На підставі проведеного огляду сучасних методів тестування на проникнення Wi-Fi роутера можна сформувати низку завдань, реалізація яких дасть змогу створити ефективну та універсальну методику пентесту.

1) Виділити пріоритетні сценарії атак, серед яких WPS-перебір PIN-коду, brute-force аналіз захоплених handshakes, організація Rogue AP та відпрацювання вразливості KRACK, та описати для кожно го з них апаратно-програмні вимоги, обмеження й переваги.
2) Розробити алгоритм комбінованого підходу до сканування, визначити порядок застосування автоматизованих інструментів і ручної валідації результатів, а також критерії переходу від одного етапу тестування до іншого залежно від виявлених уразливостей і типу мережі.
3) Повести порівняльний аналіз доступних рішень (комерційних і безкоштовних) за такими критеріями, як швидкість виконання, технічна складність, вартість та ймовірність успіху, й обрати оптимальний набір інструментів для різних сценаріїв. Далі потрібно формалізувати результати тестування, визначивши єдину структуру звітів із докладним описом сценарію, технічними даними, виявленими вразливостями та рекомендаціями, а також підготувати шаблони таких звітів для кожного типу атаки.
4) Створити методику оцінювання рівня загрози, яка б на основі кількості успішних тест-кейсів і критичності вразливостей дозволяла визначати ризик мережі за шкалою «низький–середній–високий» та обчислювати загальний рівень небезпеки для подальшого планування заходів із захисту.

Таким чином, виконання окреслених завдань забезпечить створення цілісної методики динамічного тестування Wi-Fi роутера, що охоплює всі основні типи атак, поєднує автоматизовані та ручні підходи, а також передбачає об’єктивну оцінку рівня ризику та формалізовану звітність для подальшого підвищення безпеки інфраструктури.

1.5 Висновки до розділу

У першому розділі проведено глибокий аналіз сучасних підходів і методів тестування на проникнення Wi-Fi роутера. В результаті дослідження вдалося

систематизувати й класифікувати основні сценарії атак, а також визначити ключові чинники, що впливають на вразливість бездротових мереж.

Особливу увагу приділено джерелам появи вразливостей у Wi-Fi, серед яких: програмні помилки, вплив людського фактора, неправильна конфігурація обладнання та недоліки в реалізації протоколів безпеки. Показано, що саме ці причини часто стають основою для успішних атак з боку зловмисників.

В роботі детально розкрито сутність тестування на проникнення як дієвого інструменту для виявлення слабких місць у захисті мережі. Описано головні етапи тестування – від збору інформації та аналізу поточної конфігурації мережі до визначення вразливостей і надання рекомендацій із підвищення безпеки.

Підкреслено важливість поєднання ручного аналізу з автоматизованими інструментами, оскільки саме така взаємодія дозволяє глибше і точніше оцінити реальний рівень захищеності Wi-Fi роутера. У роботі також проведено докладний огляд основних методів атак, які на сьогодні є найактуальнішими: WPS brute-force, Pixie Dust, перебір handshake-файлів, атаки KRACK, DoS/Flood, створення фальшивих точок доступу, експлуатація вразливостей KARMA, атаки соціальної інженерії. У ході дослідження було також звернено увагу на важливість використання сучасних автоматизованих інструментів тестування, які значно підвищують ефективність виявлення вразливостей. Програми типу Aircrack-ng, Reaver, Wifiphisher, а також кросплатформені рішення на базі Flutter із застосуванням модульної архітектури дозволяють поєднувати ручний аналіз з автоматичним скануванням. Це забезпечує комплексний підхід до аудиту, що включає оцінку якості каналів, моніторинг рівня сигналу, виявлення активних точок доступу та потенційно уразливих пристроїв у реальному часі.

У підсумку, перший розділ формує ґрунтовну теоретичну та практичну основу для подальшої розробки власного програмного забезпечення для тестування на проникнення Wi-Fi роутерів і дає чітке розуміння сучасних викликів у сфері безпеки бездротових мереж. Такий підхід дозволяє своєчасно виявляти потенційні загрози й підвищувати загальний рівень захищеності інформаційної інфраструктури.

2 МЕТОД ДЛЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ WI-FI РОУТЕРА

2.1 Метод та алгоритм для тестування на проникнення WI-FI роутера

З метою ефективної організації функціоналу програми для тестування захищеності Wi-Fi роутера було застосовано принцип модульності та чіткого розподілу відповідальностей між складовими системи. Ключовим елементом виступає головний екран, що відповідає за ініціалізацію всіх доступних режимів тестування і завантаження актуального переліку точок доступу. В основі запуску лежить автоматизована перевірка робочого середовища, яка передбачає аналіз наявності необхідних зовнішніх утиліт, таких як aircrack-ng, netsh, iwlist, tshark та tcpdump [20]. У випадку виявлення відсутніх компонентів користувачу одразу пропонується встановити відповідні утиліти або вказати шлях до вже існуючих виконуваних файлів, що мінімізує ймовірність технічних збоїв у подальшій роботі.

Після підтвердження готовності середовища відкривається головний екран, на якому користувач обирає бажаний режим тестування по назві за методом. Кожен сценарій вторгнення представлений у вигляді окремого екранного модуля – із власними вхідними параметрами (MAC-адреса, канал, файл захоплення, словник тощо) та послідовністю дій, результат яких формується у вигляді об’єкта звіту. Всі отримані результати накопичуються в єдиному сховищі звітів, що гарантує консолідацію даних про виконані тести, уніфіковане логування, узагальнені рекомендації та можливість експорту в зовнішні файли. Головний екран (HomeScreen) відповідає за ініціалізацію доступних режимів атаки, завантаження переліку мереж і прокладку навігаційних маршрутів до модулів тестування та компонента звіту.

Кожен метод тестування – це окремий екран і набір утиліт, що реалізують певний тип атаки: 1) Метод Brute Force PIN / Pixie Dust / Krack / Flood це WPS-атаки, Метод захоплення та перебір 4-way Handshake - це Handshake-brute, 3) Метод PMKID-атаки. Одразу після ініціалізації головного екрану створюється єдине сховище результатів ReportStore, яке відповідає за збір і збереження всіх об’єктів

ReportItem – даних про успішні та неуспішні спроби кожного тесту. Цей підхід гарантує єдину точку збереження для всіх модулів, знижує дублювання коду та спрощує підтримку й розширення системи.

UML-діаграма класів із відображенням головного екрану, екрану метода Brute Force PIN, Pixie Dust, Krack, Flood , екрану метода захоплення та перебір 4-way Handshake, екрану метода PMKID-атаки та екрану ReportScreen (перегляд) із стрілками залежностей, представлена на рисунку 2.1.

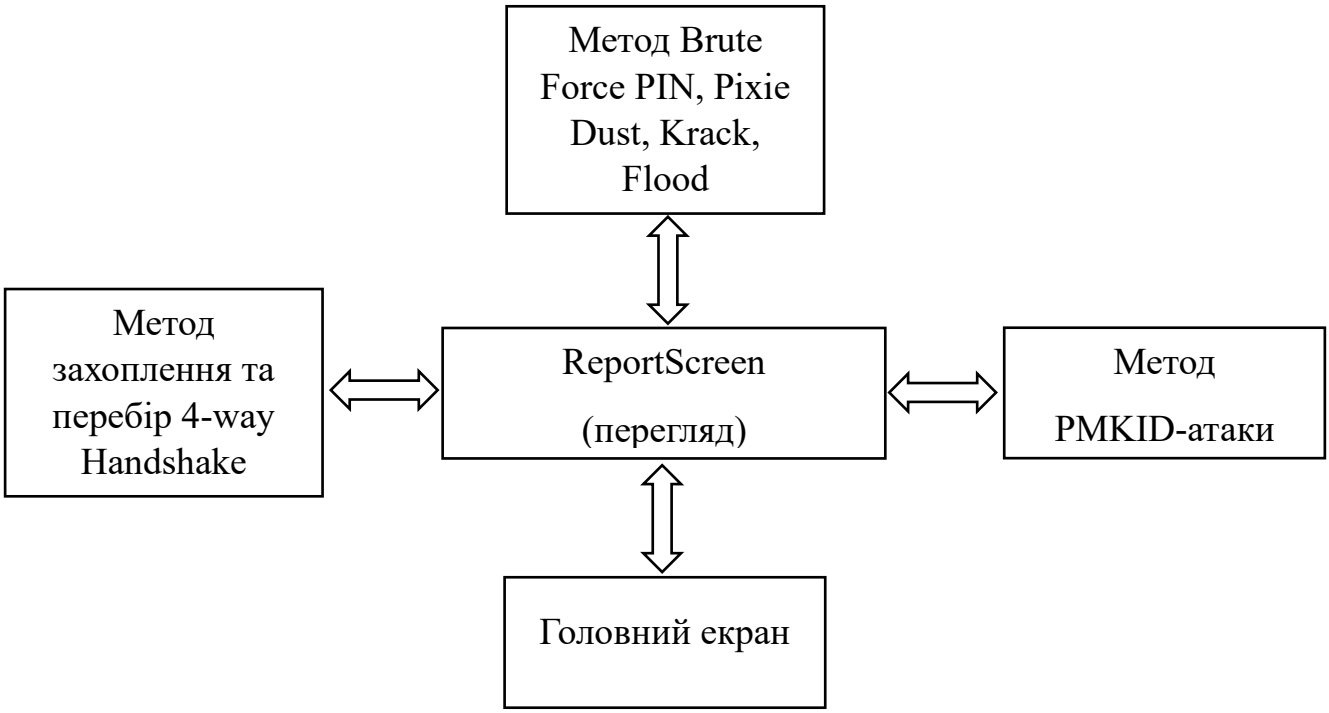


Рисунок 2.1 – UML-діаграма взаємодії класів

Кожен модуль тестування інтегровано у загальну логіку програми як окремий екран із власним функціоналом і набором утиліт. Відповідно, екран для тестування WPS реалізує як класичний перебір PIN-коду, так і криптоаналіз Pixie Dust, у той час як модуль для аналізу WPA2 здійснює як перебір паролів по handshake, так і імітацію атак KRACK або DoS через flood деавторизаційних пакетів. Завдяки цьому кожен окремий функціональний блок може розвиватися незалежно, що істотно спрощує локальне тестування та дозволяє швидко додавати нові методи без внесення змін у вже існуючу логіку інших частин програми.

Вся взаємодія між екранами та сховищем організована через спеціалізовані інтерфейсні методи, що забезпечує інверсію контролю, полегшує моделювання в автоматизованих тестах і підвищує надійність роботи системи в цілому. Після завершення тестування результати кожного запуску передаються до сховища, а звітний екран формує інтерактивний підсумковий звіт з обрахуванням рівня загроз та візуалізацією всіх виявлених уразливостей, детальніший взаємозв'язок наведено у таблиці 2.1.

Таблиця 2.1 – Залежності, ключові методи та канали взаємодії основних компонентів

Клас/Екран	Dependencies	Методи	Взаємодіє
HomeScreen	AttackMode, ReportStore	_loadModes(), _navigateTo(TestXScreen)	TestXScreen, ReportScreen
Test1Screen	ReportStore	_startWpsPin(), _startPixieDust()	ReportStore
Test2Screen	ReportStore, FilePicker	_startHandshakeBrute(), _stopHandshake()	ReportStore
Test3Screen	ReportStore	_startKrack(), _startFlood(), _stopFlood()	ReportStore
ReportScreen	ReportStore	_saveReportTxt(), _calculateThreatLevel()	–
ReportStore	–	addItem(ReportItem), clear()	ReportItem
ReportItem	–	–	–

Архітектура потоків даних розроблена так, щоб користувач, обравши режим тестування на головному екрані, міг швидко перейти до відповідного модуля, задати необхідні параметри і розпочати аналіз. Протягом усього тестування система забезпечує оперативне відображення логів, проміжних результатів та попереджень про помилки, а після завершення – автоматично зберігає всю інформацію у звіт, який може бути експортований для подальшого використання. Для оновлення інтерфейсу застосовується шаблон Provider, що гарантує коректну синхронізацію

стану застосунку та дозволяє реалізувати сучасні підходи до управління станом у графічному середовищі.

Завдяки такому підходу кожен функціональний компонент програми залишається ізольованим та незалежним, що значно спрощує розширення та підтримку застосунку, а також дозволяє ефективно адаптувати архітектуру до нових вимог чи інтегрувати додаткові методи тестування в майбутньому. Блок - схема алгоритму роботи головного екрану із переліком режимів тестування та вкладкою звіт представлено на рисунку 2.2.

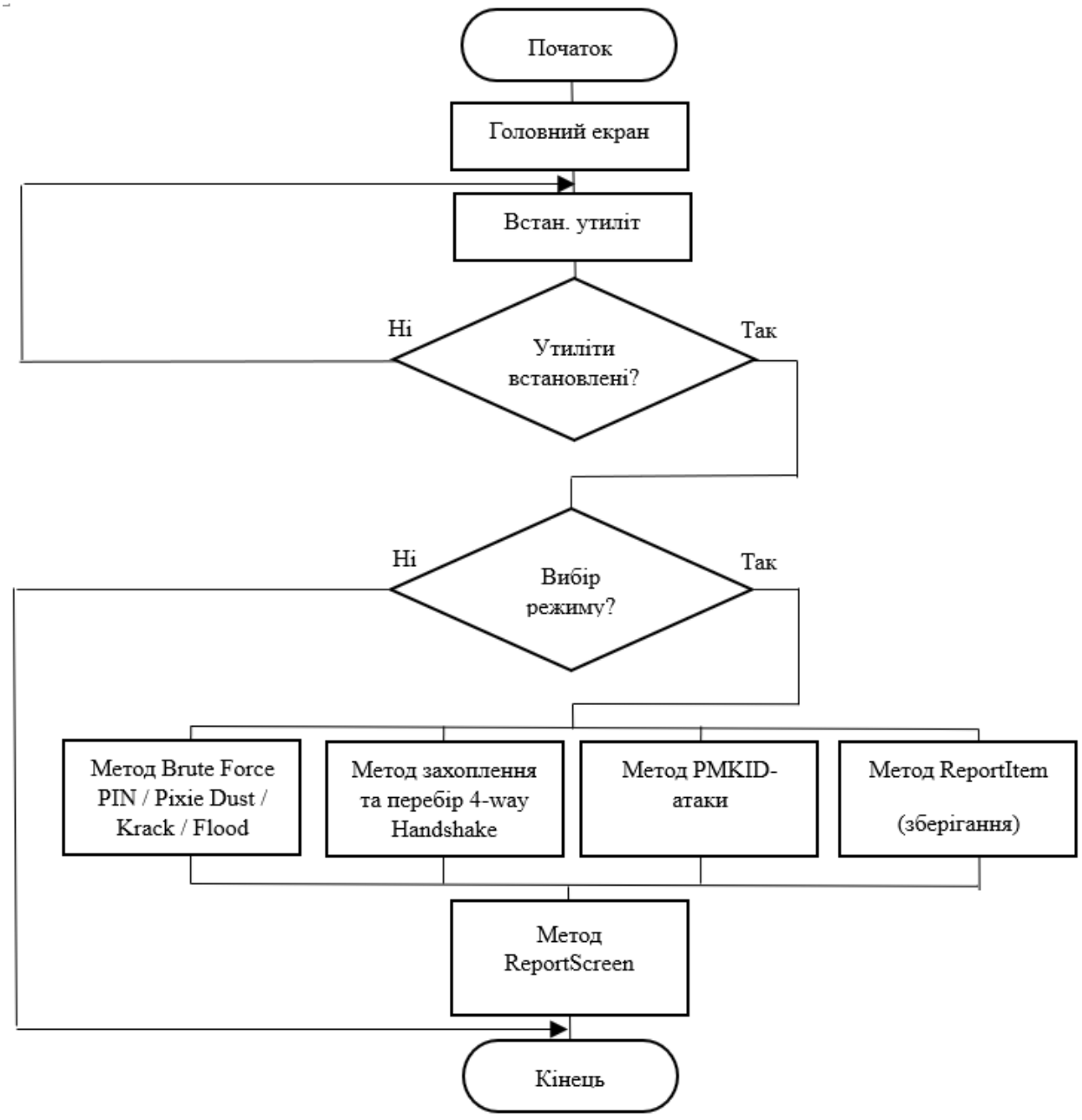


Рисунок 2.2 – Блок-схема алгоритму головного екрану

Впровадження окремого екрану звітування ReportScreen забезпечило централізовану організацію логіки формування підсумкового файлу результатів. Основна функція збереження, що реалізується методом _saveToFile, послідовно ітерує всі об'єкти з колекції ReportStore, вибудовуючи текстовий документ відповідно до заданої структури: спершу додаються загальні рекомендації, після чого детально описується кожен модуль тестування із зазначенням виявлених вразливостей та запропонованих рекомендацій. По завершенні формування документу результат записується у файл, а користувач отримує повідомлення про успішне збереження через SnackBar, після чого сформований звіт можна відкрити у файловому менеджері. Наведений нижче фрагмент коду з ReportScreen ілюструє основний цикл формування звіту з обробкою колекції ReportStore та виведенням результату у файл.

```
final buf = StringBuffer('Загальні рекомендації\n');
for (var it in ReportStore.items) buf.writeln('${it.screenName}:
${it.vulnerabilities.join(', ')} / ${it.recommendations.join(', ')}');
await File('wifi_report.txt').writeAsString(buf.toString());
ScaffoldMessenger.of(ctx).showSnackBar(SnackBar(content: Text('Звіт
збережено')));
```

Система звітності ґрунтується на двох основних компонентах. По-перше, ReportStore виступає централізованим сховищем результатів усіх тестів, реалізованим у вигляді синглтона для забезпечення єдиного доступу до даних з будь-якої частини програми. Він містить колекцію об’єктів ReportItem, кожен із яких зберігає детальну інформацію про окремий тест – тип атаки, BSSID цільової точки доступу, статус виконання, знайдені дані та часові мітки. По-друге, ReportScreen відповідає за візуалізацію зібраної інформації, надаючи інтерфейс для перегляду історії тестування у вигляді списку або таблиці. Особливістю є підтримка експорту даних у структурований текстовий файл, що містить як загальні рекомендації, так і детальні результати кожного сценарію тестування.

Доступ до актуальної статистики для всіх ReportItem забезпечується гетерами ReportStore, що гарантує відображення найсвіжіших даних на інтерфейсах програми без складних механізмів передачі стану. Метод збереження звіту послідовно додає загальні поради, перелік назв екранів, списки вразливостей і

рекомендацій, а також підсумковий рівень загрози; одразу після запису користувач отримує повідомлення про успішність експорту й шлях до файлу.

Об'єднання різних сценаріїв тестування (Brute force, Pixie Dust для WPS, Handshake brute/KRACK та DoS flood) в одному інструменті є виправданим, оскільки ці типи атак найпоширеніші серед сучасних Wi-Fi роутерів [21]. Модульна архітектура дозволяє просто додавати нові методи – достатньо створити новий екран, реалізувати набір інтерфейсних методів і зареєструвати його в AttackMode у HomeScreen. Це дає змогу у майбутньому інтегрувати навіть розподілений режим тестування чи роботу з віддаленими машинами без суттєвого рефакторингу основної кодової бази.

Для підвищення зручності та прозорості під час аналізу результатів у модулях Test1Screen, Test2Screen і Test3Screen відповідно до методів реалізовано підсвічування активного режиму за допомогою ChoiceChip, а ключові логи виводяться кольоровим текстом у SelectableText, що забезпечує оперативну ідентифікацію критичних етапів атаки. Відповідний вигляд підсвітки логів представлено на рисунку 2.3.

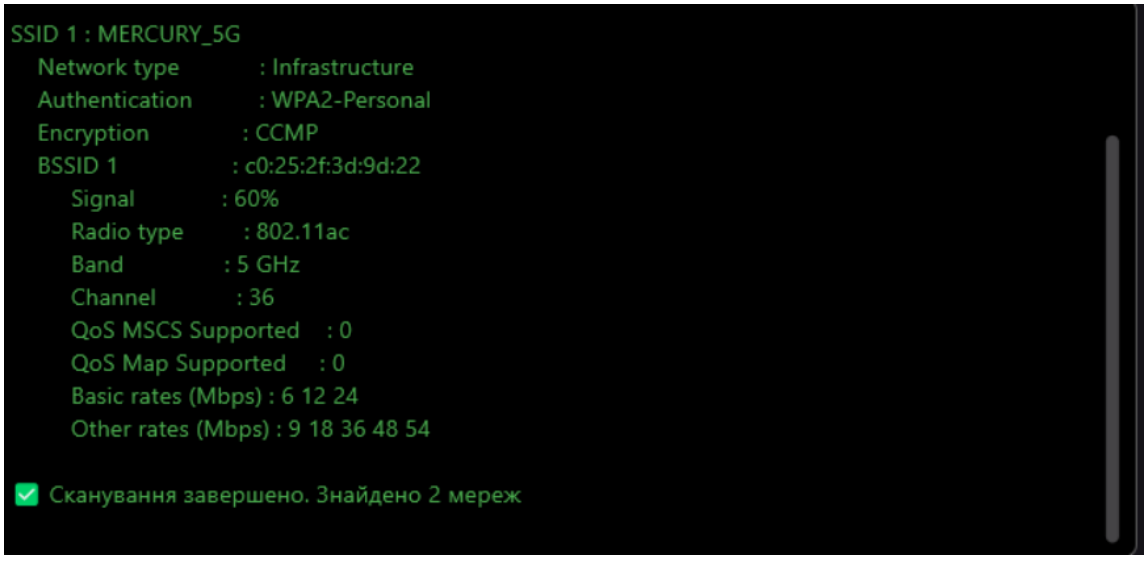


Рисунок 2.3 – Вигляд для швидкої ідентифікації логів

Метод захоплення та перебору 4-way Handshake передбачає застосування двоетапного підходу: спочатку здійснюється сканування ефіру з метою виявлення активних точок доступу і перехоплення handshake-пакетів, після чого реалізується

перебір паролів на основі отриманого handshake. Функціональний інтерфейс для цього методу має чітко виражену двоколонну структуру, що забезпечує максимальну зручність для користувача. У лівій частині розташовані елементи керування скануванням мережі та фіксацією handshake, а праворуч – налаштування для організації словникової атаки [22].

Користувачеві пропонуються два основних поля для вибору файлів із відповідними іконками: перше призначене для завантаження файлу handshake у форматі CAP або PCAP, друге – для вибору словника у форматі .txt. Впроваджено інтуїтивний механізм вибору файлів, який реалізується через інтеграцію діалогового вікна FilePicker, що дає змогу обирати необхідні дані з будь-якої папки операційної системи. Для полегшення орієнтації поля мають порядкові підказки «1.» та «2.», які вказують на правильний порядок виконання дій.

Водночас передбачено резервний механізм: якщо користувач не обрав файл вручну, програма автоматично використовує стандартні файли handshake і словника, що зберігаються у кореневому каталозі застосунку. Це дає змогу пришвидшити тестування чи демонстрацію – достатньо замінити файл у відповідній папці, не виконуючи вибір кожного разу.

Безпосередньо під полями для вибору файлів розташовано індикатор прогресу виконання словникової атаки (LinearProgressIndicator), який у режимі реального часу відображає відношення вже опрацьованих слів до загальної кількості. Такий підхід дозволяє користувачу чітко контролювати процес перебору і приблизно оцінити час до завершення атаки. Кнопка запуску («СТАРТ») після активації змінює свій вигляд на «СТОП», що дає змогу зупинити перебір паролів у будь-який момент, не перезавантажуючи інтерфейс.

Таким чином, метод захоплення та перебору 4-way Handshake поєднує гнучкість вибору файлів – як через діалогове вікно, так і автоматично, наочність роботи завдяки пронумерованим полям і зрозумілим іконкам, а також прозорість та зручність керування процесом перебору завдяки прогрес-бару та динамічній зміні стану кнопки. Варіанти вибору файлів або їх автоматичне встановлення демонструються на рисунку 2.4.

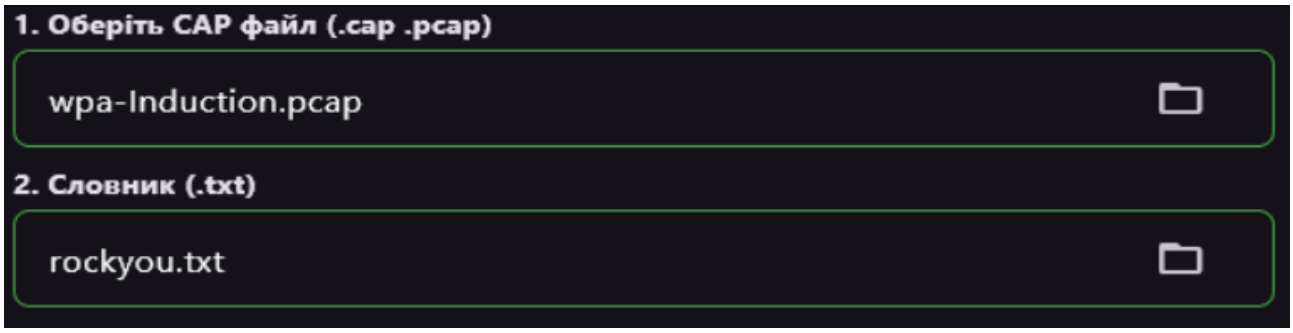


Рисунок 2.4 – Вигляд вибору файлів або їх автоматичне встановлення

Отже, архітектурне рішення забезпечує високу гнучкість, простоту тестування окремих компонентів, прозорість потоків даних та чітку відповідність професійним підходам до побудови сучасних програмних засобів для оцінювання безпеки Wi-Fi роутера.

2.2 Обґрунтування вибору методів тестування

У блоці тестування WPS атаки наша програма автоматизує два ключових підходи: Brute force WPS PIN та Pixie Dust. Перший метод здійснює сист ематичний перебір 8-значного коду, розбитого на дві частини по чотири цифри, – кожну комбінацію програма генерує автоматично, відправляє запит роутеру й аналізує відповідь на коректність PIN. Перебір двох частин по 4 цифри:

$$\text{PIN} = (X, Y), \text{ де } X \in \{0,1,2,\ldots,9999\},$$

нехай це перша чотиризначна частина.

$$Y \in \{0,1,2,\ldots,9999\},$$

тоді як друга чотиризначна частина. Тоді множина всіх PIN-кодів:

$$P=\{(X,Y)\},|P|=10^4\times10^4=10^8.$$

Однак в реальності WPS-драйвер перевіряє спочатку X, а потім Y (остання цифра – контрольна), тому загальна кількість запитів ≈

$$N_{req}=10^4+10^4=2\times10^4=20000.$$

Функція успіху позначимо через:

$$f: \mathcal{P} \to \{0,1\},$$

функцію відповіді роутера буде мати вигляд:

$$f(X,Y) = \begin{cases} 1, & \text{якщо PIN} = \text{правильний;} \\ 0, & \text{інакше.} \end{cases}$$

1. Для X від 0 до 9999: 1) Надсилати (X, –) → чекати f(X,–). 2) Якщо f(X,–)=1 → зафіксувати X і перейти до кроку 2.
2. Для Y від 0 до 9999: 1) Надсилати (X, Y) → чекати f(X,Y). 2) Якщо f(X,Y)=1 → зупинити перебір, PIN знайдено.

очікувана кількість запитів до роутера в гіршому випадку:

$$T_{max}=10^4+10^4=2\cdot10^4 .$$

Якщо роутер відповідає успіхом, знайдений PIN фіксується миттєво та надсилається до ReportStore. Другий метод – Pixie Dust Attack – спирається на слабкість генерації E-Hash1/E-Hash2 в WPS. Після запуску програма в моніторному режимі перехоплює службові пакети WPS, витягує з них криптографічні хеші та проводить офлайн-аналіз цих даних для миттєвого відновлення PSK без повного перебору. Нехай ми ловимо службові WPS-пакети й із них витягуємо крипто-графічні «хеші»:

$$H_1,H_2 \in \{0,1\}* ,$$

далі існує функція відновлення ключа:

$$g : (H_1, H_2) \longrightarrow PSK,$$

де виклик g – це кілька поліноміальних операцій над бінарними даними, та зазвичай має складність (тобто не залежить від довжини PSK, а тільки від фіксованої кількості криптографічних операцій). Загальна формула часу буде мати вигляд:

$$T_{pixie}=t_{capture}+t_g(H_1,H_2),$$

де $t_{capture}$ – час перехоплення потрібних пакетів (до появи H_1, H_2), t_g – час виконання криптоаналізу (звичайно 1–2 секунди). Графік порівняння необхідної кількості спроб Brute-force PIN: приблизно 20 000 запитів.

Перший метод Brute Force PIN, Pixie Dust, KRACK , Flood орієнтований на виявлення та експлуатацію типових уразливостей бездротових мереж шляхом реалізації одразу кількох сценаріїв атак на основі сучасних дослідницьких підходів до тестування захищеності.

Перший етап полягає у використанні Brute Force PIN для атаки на протокол WPS. Програма автоматично сканує простір на наявність точок доступу з активованим WPS та формує список цілей. Для кожної знайденої мережі здійснюється поетапний перебір восьмизначного PIN-коду за допомогою спеціалізованого алгоритму, який мінімізує кількість необхідних спроб шляхом поділу PIN на дві частини згідно з логікою WPS. Процес повністю автоматизований: всі запити надсилаються у фоновому режимі, а отримані відповіді роутера аналізуються парсером для своєчасного виявлення валідного коду. У разі успіху знаходиться справжній пароль (PSK), і результати тесту автоматично фіксуються у звіт.

Другий сценарій – атака Pixie Dust передбачає експлуатацію криптографічних вразливостей у реалізаціях WPS. На відміну від класичного перебору, даний підхід використовує слабкі алгоритми генерації випадкових чисел в роутерах. Програма перехоплює службові пакети WPS (E-Hash1, E-Hash2), обробляє їх локально і за лічені секунди, без значної кількості запитів до точки доступу, може відновити PIN або навіть одразу PSK для вразливих моделей пристроїв.

Додатково, реалізовано модулі для перевірки стійкості мережі до сучасних протокольних атак. KRACK-атака передбачає тестування WPA2-мережі на предмет уразливості до повторного використання ключів при повторній ін’єкції handshake-пакетів. Це дозволяє перевірити, чи захищена мережа від компрометації сесійних ключів навіть за наявності складного пароля.

Flood-атака (деавторизаційний flood) – ще один важливий інструмент для тестування захисту мережі від DoS-атак. Програма надсилає велику кількість деавторизаційних кадрів із метою оцінити, наскільки швидко точка доступу «відвалює» клієнтів та як реагує на спроби зловмисників вивести мережу з ладу.

Весь процес організовано так, що користувач має можливість обрати необхідний сценарій атаки (Brute Force PIN, Pixie Dust, KRACK чи Flood) через зручний інтерфейс одним кліком. Параметри цільової мережі можуть підставлятись автоматично після сканування, що значно підвищує швидкість і простоту роботи. Усі етапи – від запуску атаки до отримання та аналізу результату – автоматизовано, що дозволяє оперативно виявити та задокументувати існуючі вразливості, а також

надати обґрунтовані рекомендації щодо їх усунення. Блок-схема метода Brute Force PIN, Pixie Dust, Krack, Flood представлено на рисунку 2.5.

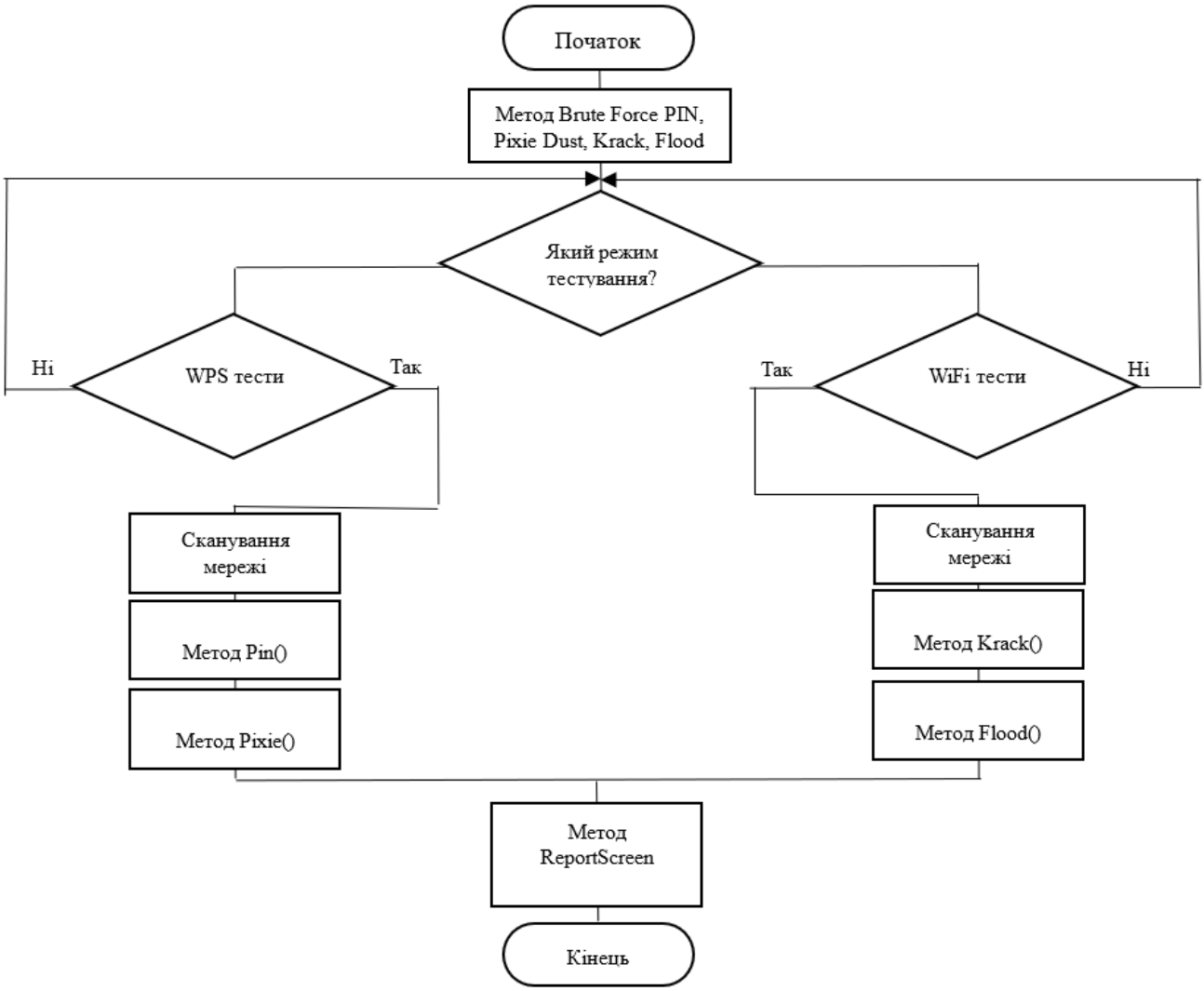


Рисунок 2.5 – Блок-схема алгоритму Brute Force PIN, Pixie Dust, Krack, Flood

У правому нижньому куті інтерфейсу передбачено спеціальний блок із рекомендаціями, який автоматично з’являється після завершення процесу атаки, враховуючи результати виконання тесту. Автоматизація процесів у методі перебору PIN-кодів і аналізу протоколу WPS охоплює декілька ключових аспектів. По-перше, система реалізує автоскандування бездротових мереж: доступні SSID відслідковуються в реальному часі, а їх перелік автоматично оновлюється та підставляється у форму без додаткової участі користувача. Запуск атаки також автоматизовано – перебір PIN-кодів або збір хешів E-Hash1/E-Hash2 здійснюється одним натисканням, без необхідності окремо викликати зовнішні утиліти чи

вводити параметри вручну. Відповіді точки доступу аналізуються автономно за допомогою вбудованого парсера, і при виявленні вразливості негайно створюється об’єкт звіту із зазначенням назви тесту, знайденого PIN або PSK, часових міток та відповідних рекомендацій. При першому запуску програми автоматично перевіряється наявність Windows Subsystem for Linux (WSL) з необхідними утилітами – такими як Aircrack-ng, Hcxdumptool, Wireshark – і, у разі відсутності, здійснюється їх інсталяція, що дозволяє організувати виконання атак в ізольованому середовищі Linux без необхідності виходити з Windows та гарантує доступність усіх інструментів одразу «з коробки».

Другий метод забезпечує повністю автоматизований офлайн-перебір паролів на основі попередньо захопленого 4-way handshake. Після вибору відповідного handshake-файлу (у форматі .cap або .pcap) та словника (наприклад, rockyou.txt) за допомогою діалогового вікна користувач ініціює перебір одним натисканням кнопки. Система інтегрується з aircrack-ng або власним модулем, передаючи шляхи до обраних файлів та поетапно підставляючи кожен пароль зі словника. Перед початком атаки здійснюється перевірка наявності WSL із необхідними інструментами, і в разі потреби їх встановлення відбувається автоматично через wsl --install. Інтерфейс методу організовано у вигляді двох панелей: ліва частина містить поля для введення BSSID і каналу разом із кнопкою для захоплення handshake, а права – засоби для вибору handshake-файлу, словника, а також кнопки запуску/зупинки, індикатор прогресу та журнал подій із кольоровими логами. Якщо користувач не обирає файл вручну, програма автоматично використовує файли за замовчуванням із кореневого каталогу застосунку, що спрощує демонстрацію або тестування.

У ході перебору на екрані виводиться прогрес-бар із відсотковим відношенням оброблених слів і текстові повідомлення, наприклад: «Перевірка слова ‘123456’… Ні», «Перевірка слова ‘password1’… Так, ключ знайдено». Такий підхід дозволяє одночасно відслідковувати стан захоплення handshake та перебіг перебору паролів, забезпечуючи високу інформативність та зручність під час тестування. Схематичне зображення загального алгоритму роботи методу захоплення та перебір 4-way Handshake наведено на рисунку 2.6.

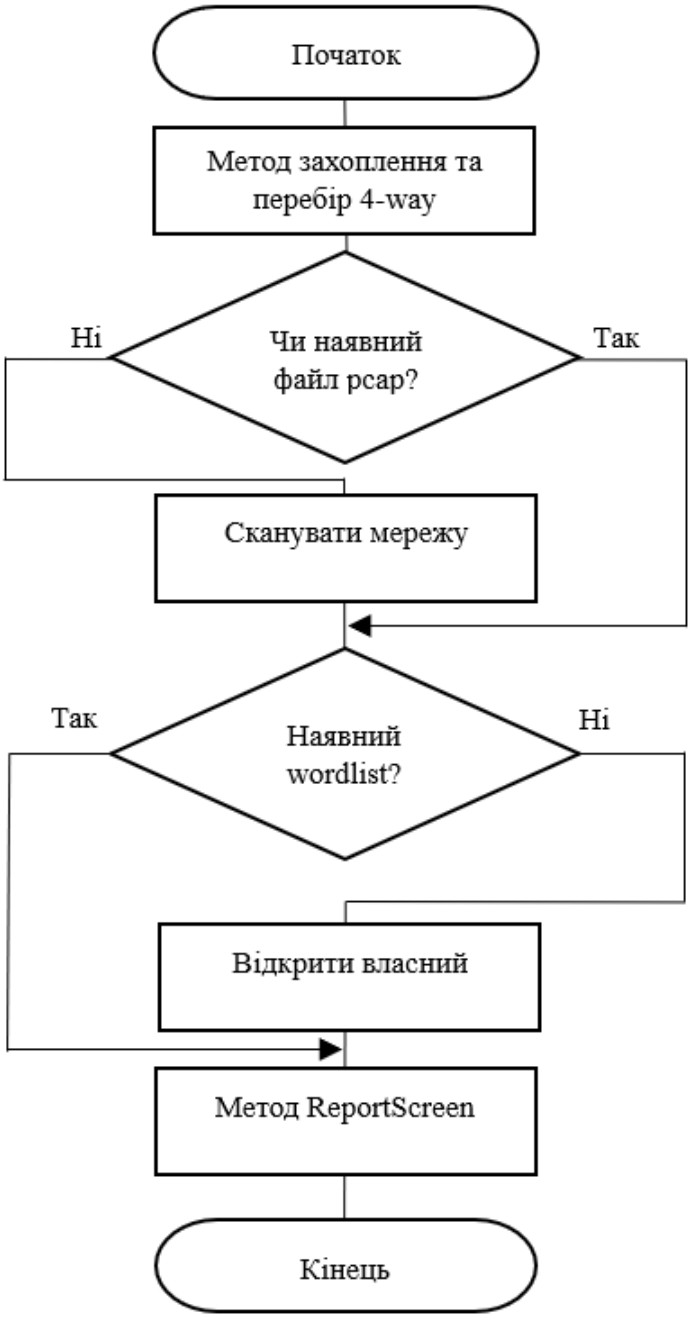


Рисунок 2.6 – Блок-схема алгоритму захоплення та перебір 4-way Handshake

Екран Handshake Brute з активним перебором. На зображенні видно вибрані файли handshake.cap і rockyou.txt, зелений прогрес-бар зі значенням «97,99%», а в області логів – повідомлення «Перевірка слова ‘lola123’… Не вірно» та «Ключ знайдено: Induction». Під логами представлений блок з виявленою вразливістю «Handshake захоплено» та згодом підтягнуто.

Ключові елементи автоматизації формуються з самого початку до прикладу перед початком перевірки handshake програма самостійно аналізує формат файлу й

обирає відповідний режим aircrack-ng, автоматично визначає формат тобто програма аналізує заголовки файлу .cap/.pcap і автоматично підбирає режим читання handshake без втручання користувача, фоновий перебір тобто слова словника обробляються у фоні з оптимізацією під багатоядерні процесори, з можливістю розподілу навантаження за індексами (гібридний режим), миттєвий експорт результатів тобто як тільки ключ знайдено, процес зупиняється, результат фіксується і створюється ReportItem зі статусом «cracked», а текстовий файл звіту оновлюється автоматично. Словник обробляється у фоновому ізоляторі (Isolate) Dart, що розвантажує UI-потік і оптимізує використання CPU. Одразу після знаходження ключа ReportItem формується із статусом cracked=true та деталями (знайдене слово, час спроби, кількість перевірених записів).

Третій метод передбачає повністю автоматизований процес отримання та аналізу PMKID для здійснення атаки на захищеність мережі Wi-Fi без участі клієнта. Після запуску процедури сканування програма автоматично ініціалізує необхідне середовище, використовуючи Windows Subsystem for Linux із попередньо встановленими утилітами hcxdumptool, hcxpcaptool і hashcat, і виконує перехоплення PMKID-кадрів у моніторному режимі. У разі відсутності WSL програма самостійно інсталює усі потрібні компоненти через відповідний скрипт [23].

Після завершення сканування автоматично формується список виявлених точок доступу, серед яких користувач може обрати цільову мережу для подальшої атаки. Далі відбувається автоматичний виклик ланцюжка утиліт: перехоплені дані конвертуються у формат, сумісний із hashcat, після чого здійснюється перебір пароля на основі словника. Якщо користувач не обирає власні файли захоплення чи словник, програма використовує вбудовані стандартні ресурси із директорії застосунку.

Весь процес виконується у фоновому режимі, що дозволяє не блокувати інтерфейс користувача – прогрес атаки і ключові повідомлення виводяться у спеціальній області логів. Як тільки пароль знайдено, перебір миттєво зупиняється, створюється відповідний об'єкт звіту зі статусом «cracked», і всі деталі (знайдене

слово, час спроби, технічні характеристики мережі) фіксуються у центральному сховищі для подальшого аналізу та експорту у звіт.

Автоматизація даного методу охоплює не лише виконання команд, але й самостійну інсталяцію всієї необхідної інфраструктури, резервне використання дефолтних файлів, фонову обробку даних із підтримкою багатоядерних процесорів та динамічне формування рекомендацій щодо усунення виявлених вразливостей. Такий підхід дозволяє провести повноцінну PMKID-атаку буквально у декілька кліків, без потреби в глибоких технічних знаннях чи складних ручних налаштуваннях, що значно підвищує ефективність і зручність тестування безпеки Wi-Fi роутераі.

2.3 Методи формування звіту та рекомендації

Передусім слід підкреслити, що саме зведений звіт із рекомендаціями перетворює базовий набір логів на цілісний документ управлінського рівня. Такий звіт є ключовим інструментом для прийняття рішень у сфері інформаційної безпеки: він дає змогу швидко оцінити масштаби виявлених слабкостей, відстежити динаміку їх усунення в часі та підтвердити відповідність внутрішнім політикам або зовнішнім стандартам (наприклад, ISO 27001) [24]. Крім того, структурований текстовий документ полегшує передачу результатів аудиту керівництву чи замовнику, а у разі повторного сканування дозволяє автоматично порівняти попередній і поточний стан, що пришвидшує процес регресивного аналізу та планування заходів ліквідації вразливостей.

Після завершення будь-якого тесту екран звіту (ReportScreen) негайно агрегує всі елементи ReportItem із глобального сховища та відображає їх у єдиному інтерфейсі, що складається з кількох ключових зон. У верхній частині екрану виводиться картка «Загальні рекомендації», оформлена контрастним фоном та чітким списком із дев’яти пунктів (використання WPA2/WPA3 із мінімум 12 символами, обов’язкова зміна SSID, вимкнення WPS/UPnP, активація гостьової мережі для IoT тощо). Ця секція побудована як автономний віджет, який підтягує

вміст із поля _generalInfo і дозволяє швидко ознайомитися з основними правилами безпеки перед тим, як заглиблюватися в деталі окремих атак.

Під карткою загальних рекомендацій розташований головний блок зі списком результатів по кожному тесту. Для кожного ReportItem динамічно створюється окрема картка, що містить назву тесту, часову мітку виконання, статус (Успішно/Не виявлено), знайдені дані (PIN або PSK), перелік вразливостей і рекомендацій, оформлених маркованим списком. Цей перелік здатний адаптуватися до будь-якої довжини: навіть якщо вразливостей чи рекомендацій кілька десятків, картка автоматично розширюється з вертикальною прокруткою, зберігаючи єдиний стиль тексту та відступів. Текст всередині картки рендериться динамічно за допомогою ListView.builder, що гарантує плавність роботи навіть при великій кількості елементів. Нижче розташовані підсумкові елементи UI тобто рядок із загальною кількістю проведених атак (обчислюється як довжина масиву ReportStore.items) та велика кнопка «Зберегти в .txt». Сам факт наявності лічильника дає змогу миттєво оцінити масштаби перевірок за одну сесію, а кнопка експорту запускає метод _saveToFile(), який у фоновому режимі формує текстовий звіт і записує його в стандартну папку Documents користувача. Після завершення збереження спрацьовує SnackBar із повним шляхом до файлу, що забезпечує зворотний зв’язок та упевнення у коректності операції.

Метод збереження звіту у файл реалізовано максимально оптимізовано:1) він використовує ітерацію ReportStore.items, 2) додає блок загальних рекомендацій, розбиває детальний звіт на розділи за тестами, автоматично підставляє дату/час та статус, переліковує вразливості й рекомендації, а наприкінці додає рядок із кількістю загроз. Уся ця логіка виконується в асинхронному потоці, аби не блокувати UI, і використовує StringBuffer для швидкого формування кінцевого тексту. Таким чином, для користувача створення повноцінного звіту з усіма технічними деталями зводиться до одного кліку, приклад загальних рекомендацій у вікні звіту представлено на рисунку 2.7.

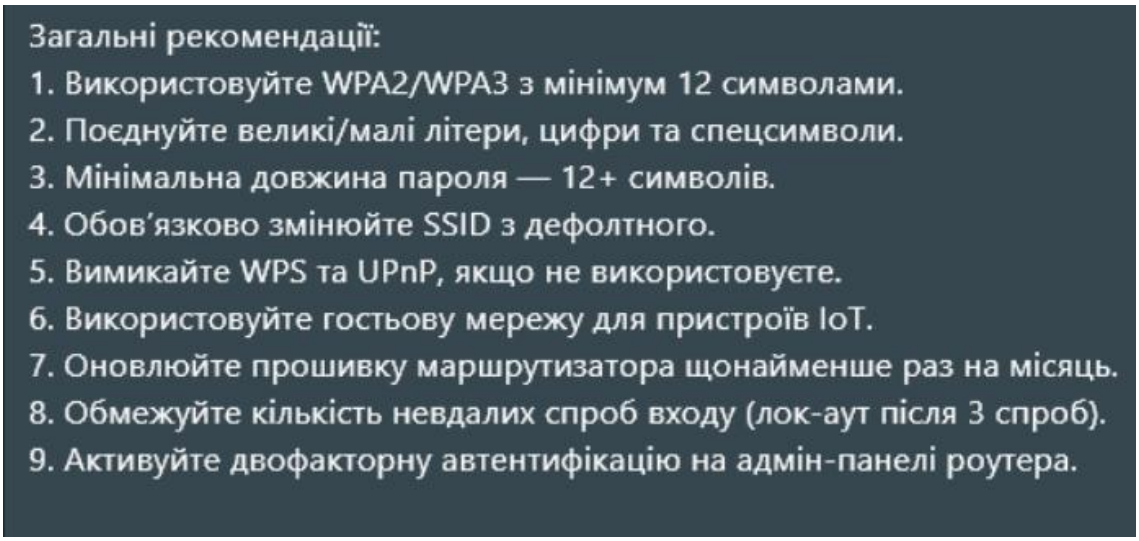


Рисунок 2.7 – Загальний вигляд сторінки з загальними рекомендаціями

Після запису файлу програма викликає ScaffoldMessenger.of(ctx).showSnack Bar(...), в якому виводиться повідомлення з повним шляхом до файлу, скріншот вигляду провідника Windows з відкритою папкою, де видно збережений файл wifi_report.txt та його іконку зображено на рисунку 2.8.

Рисунок 2.8 – Приклад екрану вигляду файлу в провіднику

Таким чином, ReportScreen із функцією автоматичного формування звіту поєднує в собі зручний UX, прозору звітність і високий рівень автоматизації, що дозволяє завершити весь цикл тестування й отримати документований результат без зайвих дій.

2.4. Висновок до розділу

У другій частині бакалаврської кваліфікаційної роботи проведено глибокий аналіз і розробку методології автоматизованого тестування на проникнення Wi-Fi роутерів із використанням сучасного програмного комплексу, побудованого на принципах модульності, ізоляції логіки та централізованого управління результатами. На основі огляду існуючих технік і особистих розробок була

реалізована архітектура, що поєднує одразу кілька ключових сценаріїв атаки: брутфорс WPS PIN-коду, криптоаналіз Pixie Dust, перебір паролів по handshake-файлу, атаки на протокол WPA2 через KRACK і DoS-флуд. Кожен із цих методів впроваджено у вигляді окремого ізольованого модуля з власним інтерфейсом, набором параметрів та уніфікованою системою звітності, що гарантує гнучкість, масштабованість і зручність подальшого супроводу програмного продукту. Крім того, реалізована система централізованого зберігання та обробки результатів забезпечує прозорість і контроль над процесом тестування, що сприяє підвищенню якості аудиту і полегшує взаємодію між технічними фахівцями та керівництвом. Автоматизація звітування з чітким описом виявлених вразливостей і рекомендацій дозволяє значно скоротити час на аналіз та прийняття рішень щодо підвищення безпеки. Використання модульного підходу гарантує можливість швидкої адаптації програмного продукту до нових загроз і технологічних змін, що робить його перспективним інструментом для довготривалого використання в умовах постійної еволюції бездротових мереж.

Важливим досягненням стала організація централізованого сховища результатів тестів (ReportStore), що дає змогу не лише акумулювати всі отримані дані про вразливості, а й автоматично формувати комплексний звіт для подальшого аналізу і передачі керівництву чи клієнтам.

У результаті проведеної роботи доведено, що об'єднання найпоширеніших методів тестування у межах єдиної гнучкої платформи забезпечує не лише комплексне покриття вразливостей бездротових мереж, а й високий рівень автоматизації формування звітності й рекомендацій. Програмний продукт дозволяє швидко виявляти типові проблеми безпеки Wi-Fi, одразу генеруючи технічні та організаційні поради щодо їх усунення, а модульна архітектура спрощує розширення інструменту новими сценаріями атак. Впроваджений підхід формує фундамент для подальшого розвитку у напрямку створення інтегрованих систем аудиту безпеки мереж із підтримкою CI/CD, аналізу динаміки ризиків та автоматизованого впровадження контрзаходів.

3 ПРОГРАМНИЙ ЗАСІБ ДЛЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ WI-FI РОУТЕРА

3.1 Обґрунтування вибору мови програмування та середовища розробки

У процесі розробки програмного засобу для тестування безпеки Wi-Fi роутера перед нами стояло декілька ключових завдань, які визначили вибір технологій та архітектурного підходу. По-перше, необхідно було створити кросплатформенний застосунок, що однаково працював би під Windows, а в подальшому Linux та macOS, без дублювання коду під кожну операційну систему. По-друге, важливою вимогою була швидкість розробки інтерфейсу користувача з можливістю оперативного внесення змін у макети й логіку взаємодії. По-третє, оскільки для проведення атак використовуються низькорівневі Linux-утиліти (aircrack-ng, hcxdumptool, hashcat), критичною була легкість інтеграції з нативним оточенням Linux і можливість автоматизації виклику цих команд із GUI. І нарешті, ми прагнули налагодити прозорий механізм безперервної інтеграції і доставки (CI/CD) для автоматичної збірки та тестування пакету під усі платформи.

Після детального порівняння таких технологічних стеків, як Electron (JavaScript/HTML/CSS), Xamarin (C#/.NET), Qt (C++/QML) та ряду менш розповсюджених рішень, остаточний вибір зупинився на поєднанні Dart + Flutter з використанням Windows Subsystem for Linux (WSL). Подібне рішення поєднало у собі максимальну гнучкість UI-розробки, мінімальний оверхед на підтримку різних ОС та бездоганну інтеграцію з Linux-утилітами [25].

У першу чергу, Flutter – фреймворк від Google, побудований на мові програмування Dart – забезпечує єдиний кодовий базис для всіх трьох командних середовищ: Windows, Linux та macOS. Це означає, що інтерфейс головного меню, екранів обрання режимів атак, детальних логів, налаштувань файлів та фінального звіту реалізується тим самим набором віджетів, а Flutter-рушій гарантує однакову поведінку й вигляд на різних платформах. Завдяки такому підходу ми змогли скоротити час розробки інтерфейсу майже у три рази порівняно з класичними нативними рішеннями. При цьому уникнення дублювання коду знижує ризик появи

непередбачених розбіжностей у поведінці програми і дозволяє легко розширити функціонал у майбутньому, наприклад, додавши мобільні версії для Android та iOS, не створюючи окремого UI-шару.

Для побудови UI використовується обширна бібліотека готових Material Design і Cupertino віджетів, а за допомогою пакета Provider (або альтернативно Riverpod) забезпечується простий та прозорий механізм керування станом, що зберігає чітку організацію коду та полегшує тестування, вікно VS Code з відкритим Flutter-проєктом – у панелі “lib/” видно папки “screens”, “services”, “models”, графічно представлено на рисунку 3.1.

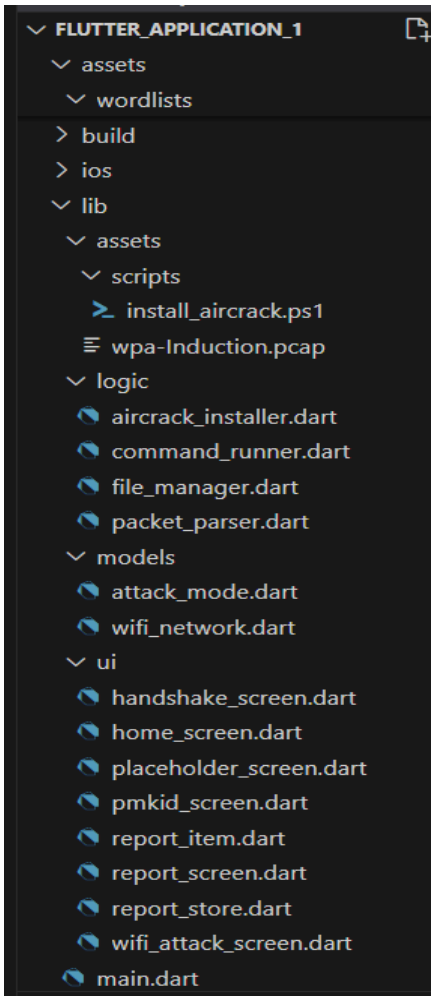


Рисунок 3.1 – Структура каталогів Flutter-проекту у VS Code

Ключовою особливістю Flutter є механізм Hot Reload, який дозволяє розробнику вносити зміни до коду під час виконання застосунку та миттєво бачити оновлення в интерфейсі – без необхідності перезавантажувати весь додаток. Це

особливо важливо при тонкому налаштуванні складних макетів, анімацій та реактивної логіки. Замість тривалих циклів «змінити–побудувати–перезапустити» ми отримуємо миттєвий зворотний зв'язок, що значно пришвидшує ітеративну розробку. В "pubspec.yaml" перераховані ключові залежності: flutter, provider, file_picker, process_run, можна побачити на рисунку 3.2.

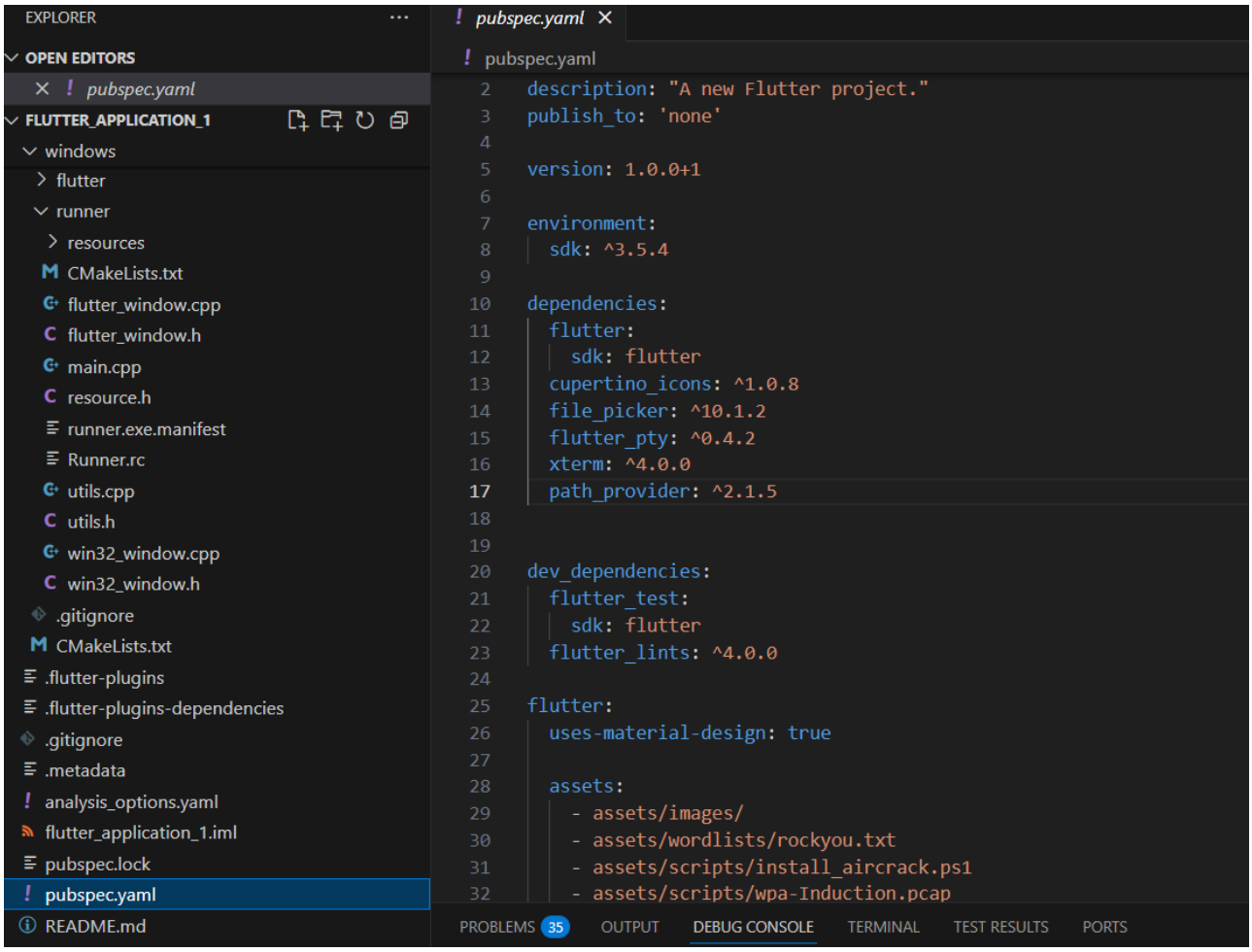


Рисунок 3.2 – Ключові залежності в файлі pubspec.yaml

В цілому можна побачити папку assets/wordlists з файлом rockyou.txt. Директорію lib/assets/scripts зі скриптом інсталяції WSL-утиліт. Піддиректорії logic, models та ui з відповідними Dart-модулями. Головний файл main.dart і файли екранів (home_screen.dart, wifi_attack_screen.dart тощо), служби й моделі.

Невід'ємною складовою нашого підходу стало використання Windows Subsystem for Linux (WSL) [26]. Оскільки більшість необхідних для аудиту Wi-Fi інструментів (aircrack-ng, hcxdumptool, hcxpcaptool, hashcat) є Linux-утилітами,

WSL дозволяє запускати їх безпосередньо в Windows без накладних витрат на віртуальні машини. Команди виконуються в Bash-оточенні, але взаємодія з мережею відбувається напряму через інтерфейс адаптера Windows, що забезпечує мінімальні затримки та повний доступ до сирих 802.11-кадрів у моніторному режимі. Після встановлення WSL ми використовуємо простий інсталятор, який при першому запуску перевіряє наявність потрібних інструментів і, за наявності відсутніх компонентів, виконує команду:

sudo apt update && sudo apt install -y aircrack-ng hcxdumptool hcxpcaptool hashcat

Завдяки цьому користувач отримує «готове до бою» Linux-оточення відразу після першого запуску додатка. Пакет встановлює утиліти в стандартні шляхи /usr/bin, і Flutter-код може викликати їх за допомогою Process.run('wsl', [...]) без додаткового налаштування PATH. Такий підхід гарантує ізоляцію від Windows-середовища, покращену безпеку та дуже швидку автоматизацію залежностей, в програмі окремо на розробницькому рівні можна побачити термінал WSL після запуску інсталятора, де видно повідомлення “Done ” та немає ніяких невиконаних пакетів чи утиліт, проте для того щоб окремо бачити сам термінал у програмі треба в подальшому допрацювати якщо при цьому є необхідність, графічне представлення терміналу зображено на рисунку 3.3.

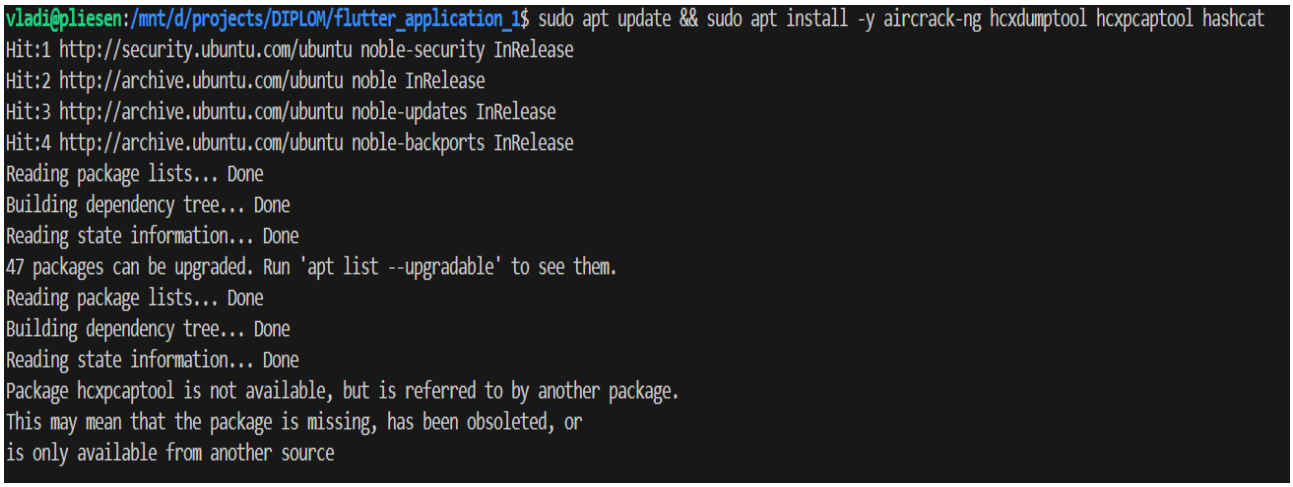


Рисунок 3.3 – Термінал WSL з оновленими пакетами

Для впровадження безперервної інтеграції та доставки у розробці програмного продукту було обрано використання інструментів GitHub Actions. Це

дозволяє повністю автоматизувати життєвий цикл застосунку: кожен раз, коли до основної гілки репозиторію надсилається новий коміт (push), у роботу вступає налаштований CI/CD-конвеєр [27]. Його робота охоплює кілька послідовних етапів, які критично важливі для підтримки високої якості коду і стабільності релізів.

Першим кроком є клонування репозиторію, після чого відбувається ініціалізація середовища для роботи з Flutter. Далі виконується команда аналізу якості коду (flutter analyze), що дозволяє виявити потенційні помилки й недоліки ще на етапі розробки. За наявності unit-тестів вони також автоматично запускаються. Особливу увагу приділено автоматизованій збірці артефактів одразу для трьох основних операційних систем – Linux, Windows і macOS, що значно спрощує подальше розгортання та тестування продукту. Нарешті, зібрані файли розміщуються у розділі Releases репозиторію, забезпечуючи легкий доступ до стабільних версій застосунку. Конфігурація основних кроків CI/CD у GitHub Actions може бути представлена так (до 5 рядків):

```
- uses: actions/checkout@v2
- uses: subosito/flutter-action@v2
- run: flutter analyze
- run: flutter build linux --release
- uses: actions/upload-artifact@v2
```

Такий підхід дає змогу виключити вплив людського фактора під час підготовки релізів, скоротити час між внесенням змін і їх тестуванням на реальних пристроях, а також гарантувати наявність актуальної стабільної збірки в будь-який момент.

Важливим елементом забезпечення якості програмного продукту є постійний моніторинг продуктивності інтерфейсу. Для цього застосовується профайлер Flutter DevTools, який дає змогу у реальному часі відстежувати частоту оновлення кадрів (FPS), використання оперативної пам’яті й виявляти потенційні «вузькі місця». Під час тестових запусків основних модулів атак було помічено, що саме при динамічному оновленні великої кількості елементів (наприклад, ReportItem) можливі короткочасні просідання фреймрейту. Для їх усунення були застосовані оптимізаційні практики: використання const-віджетів, мінімізація повторних викликів setState() тощо.

У результаті тестувань підтверджено, що навіть при відображенні кількох десятків одночасних об’єктів інтерфейс стабільно тримає 60 кадрів на секунду, а обсяг використаної пам’яті не перевищує 200–300 МБ. Це засвідчує оптимальність прийнятих рішень щодо архітектури інтерфейсу та ефективності застосованих інструментів профілювання. Графік динаміки продуктивності, отриманий за допомогою Flutter DevTools, наведено на рисунку 3.4.

Рисунок 3.4 – Графік навантаження у Flutter DevTools

На даному знімку екрана вкладки Performance у Flutter DevTools у верхній частині відображається графік часу рендерингу кожного кадру за останню сесію профілювання. По вертикалі вказані мілісекунди (0 ms, 7 ms, 17 ms, 27 ms), а по горизонталі – номер кадру або час у секундах. Блакитні стовпчики позначають Frame Time (UI) – час, витрачений на побудову віджетів і виклики життєвого циклу UI, темно-сині – Frame Time (Raster) – час, коли Skia малює пікселі, помаранчеві (jank) вказують на «повільні» кадри, тривалість яких перевищила межу в ~16 ms (індикатором служить горизонтальна лінія на рівні 60 FPS), а бордові стовпчики сигналізують про Shader Compilation – компіляцію шейдерів, яка іноді «підвисає» окремі кадри на десятки мілісекунд. У самому верху червоні повідомлення

попереджають: «Shader compilation jank detected: 8 frames janked with a total of 248.6 ms spent in shader compilation», і підказують заздалегідь профорекспільняти шейдери згідно з документацією Flutter.

Нижче розташовані дві вкладки: Frame Analysis та Timeline Events. Вкладка Frame Analysis показує детальний розбір вибраного кадру – він підсвічений сірим на графіку. Тут можна побачити, скільки часу пішло на окремі UI-фази (Layout, Compositing, Paint) та на Raster-фази (Shader compilation, інші операції рендеру). Наприклад, у рамках одного «джанкового» кадру етап малювання (Paint) займав усього 2.2 ms, натомість 30.3 ms витрачено на компіляцію шейдерів, що і спричинило тривале відтворення цього кадру.

Переключившись на вкладку Timeline Events, ми бачимо лінійну шкалу часу з поділами (0 s, 10 s, 20 s, …) та горизонтальні «планки» подій. Верхня панель із білими прямокутниками відповідає за подачу кадрів (Flutter frames) – кожний білий блок це один кадр, а їхня ширина відображає тривалість обробки. Під цією шкалою розміщена панель Clock Snapshots, де метричні позначки допомагають синхронізувати події, далі – рядок Process 20172, що позначає основний процес додатка, і кілька рядків DartWorker <pid> та FlutterConcurrentMessageLoopWorker, які показують активність фонового ізолюючого потоку або інших ізолятів Dart. Кольорові блоки на цих рядках позначають різні етапи pipeline: блакитні (VsyncProcessing) – синхронізацію вертикальної розгортки, фіолетові (Animator) – ініціалізацію анімацій, жовті – виклики paint() та layout(), зелені – інші події малювання. Часова шкала допомагає оцінити, як часто ізоляти обробляють дані поза основним UI-потоком.

Загалом середній FPS під час цього профілювання склав близько 58, що трохи нижче ідеальних 60 кадрів, через кілька «джанків» на кадрах із компіляцією шейдерів і поодинокі тривалі операції в UI. Ми бачимо, що без попередньої компіляції шейдерів перші запити малюнків і анімацій спричиняють значні затримки, про що сповіщає DevTools червона смуга вгорі. Щоб уникнути цього, рекомендується використовувати механізми precompile shaders, якими пропонує скористатися офіційна документація Flutter. Крім того, аналіз розподілу подій у

Timeline Events допомагає ідентифікувати «важкі» функції build() чи layout(), які можна оптимізувати або перемістити в ізоляти за допомогою compute().

Таким чином, дане зображення демонструє повний цикл профілювання: від загального графіка кадрів до глибинного аналізу подій у часі й ресурсів процесів, що є необхідною базою для оптимізації UI і забезпечення стабільного плавного відтворення інтерфейсу навіть під значним навантаженням.

Отже, поєднання Dart + Flutter разом із WSL і CI/CD залишає позаду всі альтернативи: ми отримуємо єдиний кодовий базис для різних ОС, миттєвий цикл розробки UI, зручний виклик Linux-утиліт без складних налаштувань, а також повністю автоматизовану систему збірки та профайлингу. Такий стек забезпечив нам можливість максимально швидко прототипувати нові функції, проводити детальне тестування, оптимізувати продуктивність і стабільно постачати кінцевий продукт користувачам.

3.2 Опис функціональних компонентів системи

Головний екран додатку, що називається HomeScreen, виконує роль центральної панелі управління всією логікою тестування Wi-Fi роутера. У верхній частині розташовано AppBar із заголовком «Wi-Fi Security Tester» та кнопкою «Звіт» праворуч – вона дозволяє швидко перейти до екрану з накопиченими результатами без необхідності завершувати поточну сесію тестування. Під заголовком займають усе вікно три великих інтерактивні картки, кожна з яких представляє окремий модуль: «Метод Brute Force PIN / Pixie Dust / Krack / Flood» із іконкою ключа для WPS атак, «Метод захоплення та перебір 4-way Handshake» із символом антени для перебору handshake, а також «Метод PMKID-атаки» із піктограмою забороненого доступу для PMKID-аналізу.

Розташовані вони таким чином: дві верхні картки стоять поруч і займають по половині ширини екрану, а третя картка розташована під ними та тягнеться на всю ширину. Така компоновка забезпечує одразу три великих «гарячих зони» для натискання, що зручно як для миші, так і для тачпада чи сенсорного екрану. Усередині кожної картки в центрі розміщена біла іконка відповідного виду атаки, а

під нею – текстове позначення «Метод Brute Force PIN / Pixie Dust / Krack / Flood», «Метод захоплення та перебір 4-way Handshake» чи «Метод PMKID-атаки». Ефект підсвічування та легкого підняття при наведенні або торканні дозволяє користувачу відчувати зворотний тактильний зв’язок з інтерфейсом.

Конструкція головного екрану передбачає мінімум навчання: користувачу достатньо одного погляду, щоб зрозуміти, який режим тестування запускає кожна кнопка. Інтегрована перевірка та автоматична інсталяція залежностей WSL (aircrack-ng, hcxdumptool, hashcat) відбувається під час завантаження цього екрану – до натискання на будь-яку картку. Як тільки всі необхідні утиліти готові, у нижній частині карток ненадовго з’являється Snackbar із повідомленням «WSL та утиліти готові», що дає впевненість у працездатності подальших дій, головний екран HomeScreen із трьома картками режимів тестування та кнопкою «Звіт» в AppBar зображено на рисунку 3.5.

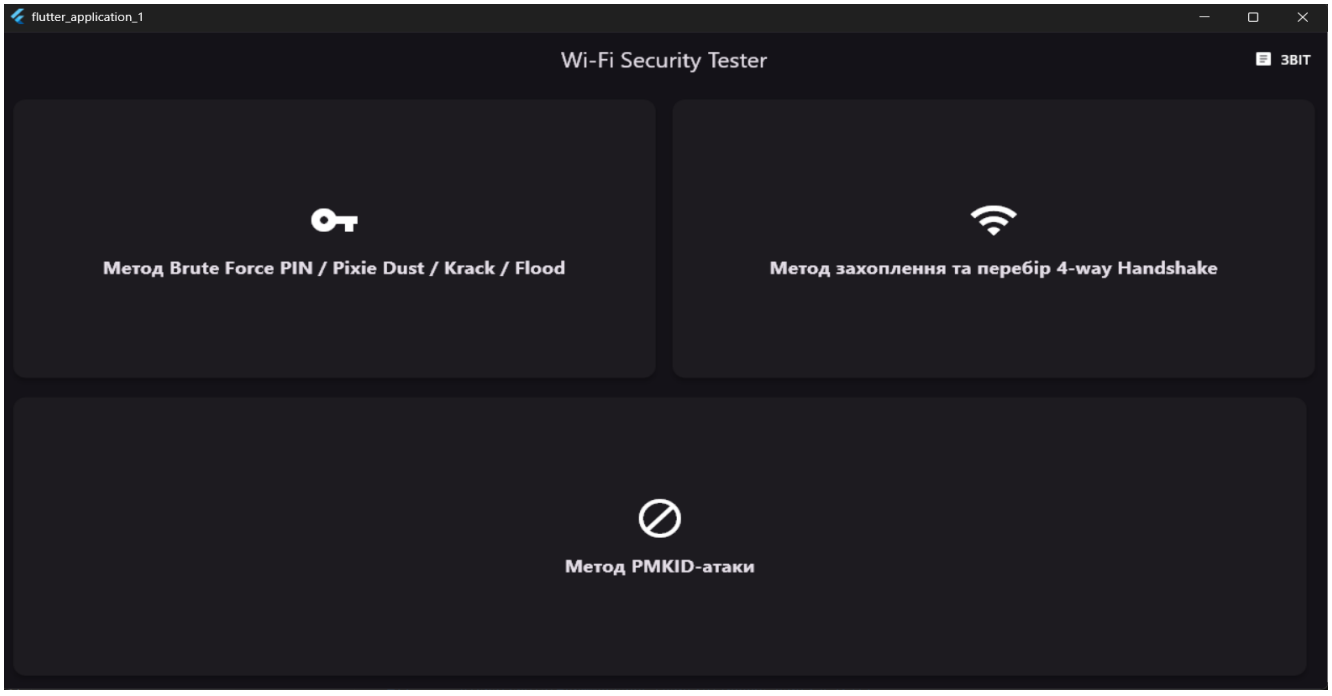


Рисунок 3.5 – Приклад головного екрану HomeScreen

Перевагою такого рішення є те, що весь цикл підготовки середовища та навігації до модулів тестування зводиться до двох дій: запуску застосунку та одного кліка по обраній картці. При цьому користувач не стикається з налаштуваннями, оновленнями чи складними меню: вся автоматизація «працює за кадром», а

інтерфейс залишається чистим, зрозумілим та лишає максимум простору для контенту наступних екранів.

Після переходу з головного меню до модуля WPS-атак користувачу відкривається єдиний екран із двома вкладками зверху: активна «WPS Атаки» та неактивна «Wi-Fi Тести». Безпосередньо під ними розташовані два переключаючі елементи ChoiceChip, які дозволяють обрати між підрежимами «PIN Brute» та «Pixie Dust». Спочатку обрано «PIN Brute» (галочка підсвічена зеленим), і весь інтерфейс налаштований на метод систематичного перебору восьмизначного WPS-PIN. У центральній частині екрана велика зелена кнопка «Сканування мереж» викликає через WSL утиліту aircrack-ng (або netsh wlan show networks mode=bssid у Windows), миттєво відображає список доступних SSID у випадаючому меню та автоматично заповнює поля «BSSID» і «Канал.

Після обрання цільової точки доступу та натискання кнопки «СТАРТ» запускається асинхронний перебір PIN-кодів. У правій половині екрана відкривається інтерактивний термінал, у якому виводяться емодзі та повідомлення про хід атаки: спочатку « Початок WPS PIN Brute атаки…», потім « Відправляємо WPS запити до роутера…», а далі в режимі реального часу відображається прогрес перебору, наприклад « Перебір PIN-кодів (11000/99999999)…». Прогрес-бар під кнопкою «СТАРТ» інформує про відсоток виконання, а сама кнопка блокується, щоб запобігти повторним натисканням. Як тільки правильний PIN знайдено, додаток миттєво зупиняє перебір і виводить зелену мітку « PIN знайдено: 12345678». Одночасно в нижній частині з’являються дві картки: «Виявлені вразливості» із переліком таких пунктів, як «WPS PIN вразливий до перебору через слабку реалізацію протоколу», «Роутер не має захисту від багаторазових спроб авторизації» та «PIN код має просту послідовність цифр», а поруч – «Рекомендації з захисту» зі списком конкретних кроків: «Змінити пароль відповідно до вимог довжини 12+ символів», «Вимкнути WPS у налаштуваннях» тощо», графічне роботи перебору PIN представлено на рисунку 3.6.

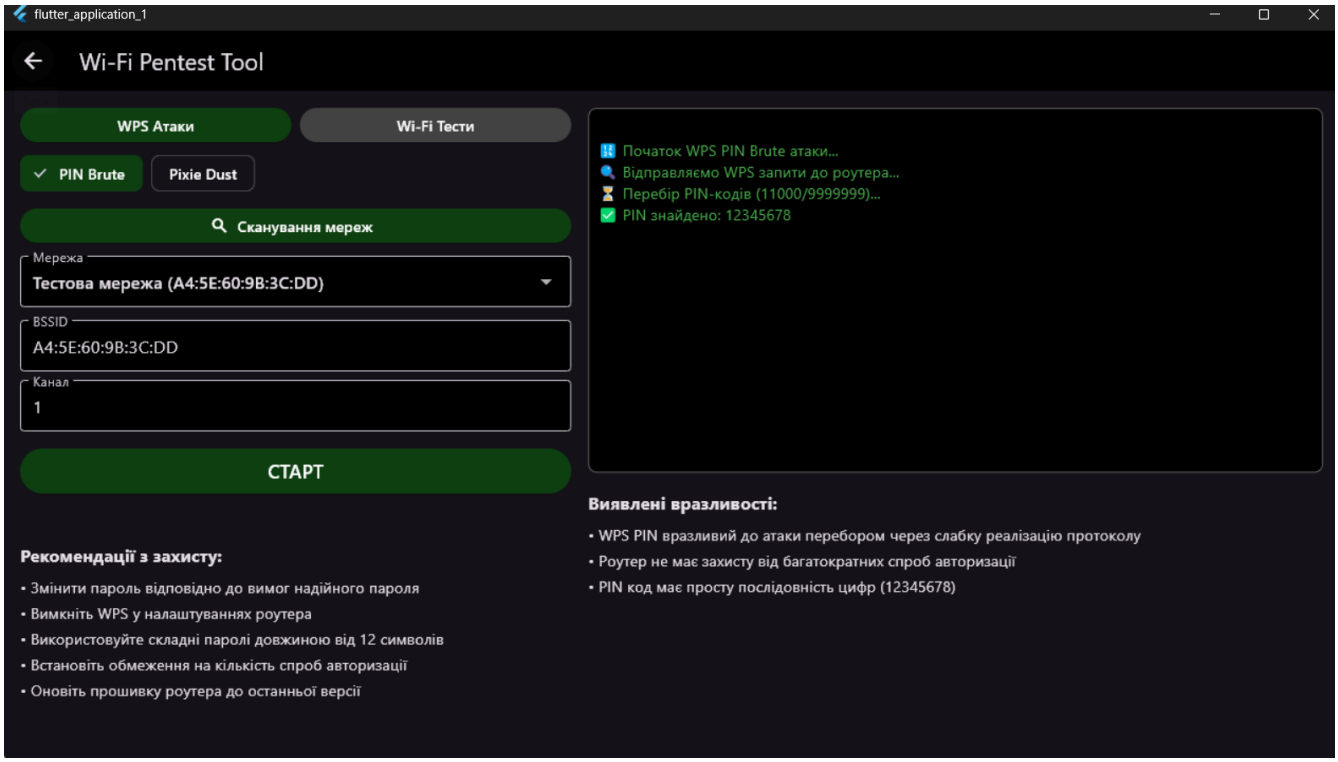


Рисунок 3.6 – Приклад екрану методу Brute Force PIN / Pixie Dust / Krack / Flood з результатом PIN brute

В терміналі з’являється повідомлення « Початок Pixie Dust атаки…». Потім програма запускає hcxdumptool у WSL, щоб захопити E-Hash1 та E-Hash2, виводячи « Отримаємо E-Hash1 та E-Hash2…». Після завершення перехоплення запускається офлайн-аналіз через hashcat, що позначається у логах як « Аналізуємо отримані дані…». Результатом автоматичного аналізу стає відображення знахідки: « Пароль мережі: P@ssw0rd123. Фінальні блоки «Виявлені вразливості» та «Рекомендації» автоматично оновлюються під цей метод: мережа вразлива до Pixie Dust, алгоритм генерації випадкових чисел слабкий, відсутній захист від атак із використанням відбитків, а рекомендації на кшталт «Використовуйте WPA3 із AES-CCMP» і «Налаштуйте моніторинг підозрілих запитів до роутера» допомагають закрити знайдені діри.

Для користувачів, які замість перебору хочуть скористатися вразливістю Pixie Dust, досить натиснути ChoiceChip «Pixie Dust». Інтерфейс адаптується: лога попередньої атаки очищається, графічно представлено на рисунку 3.7.

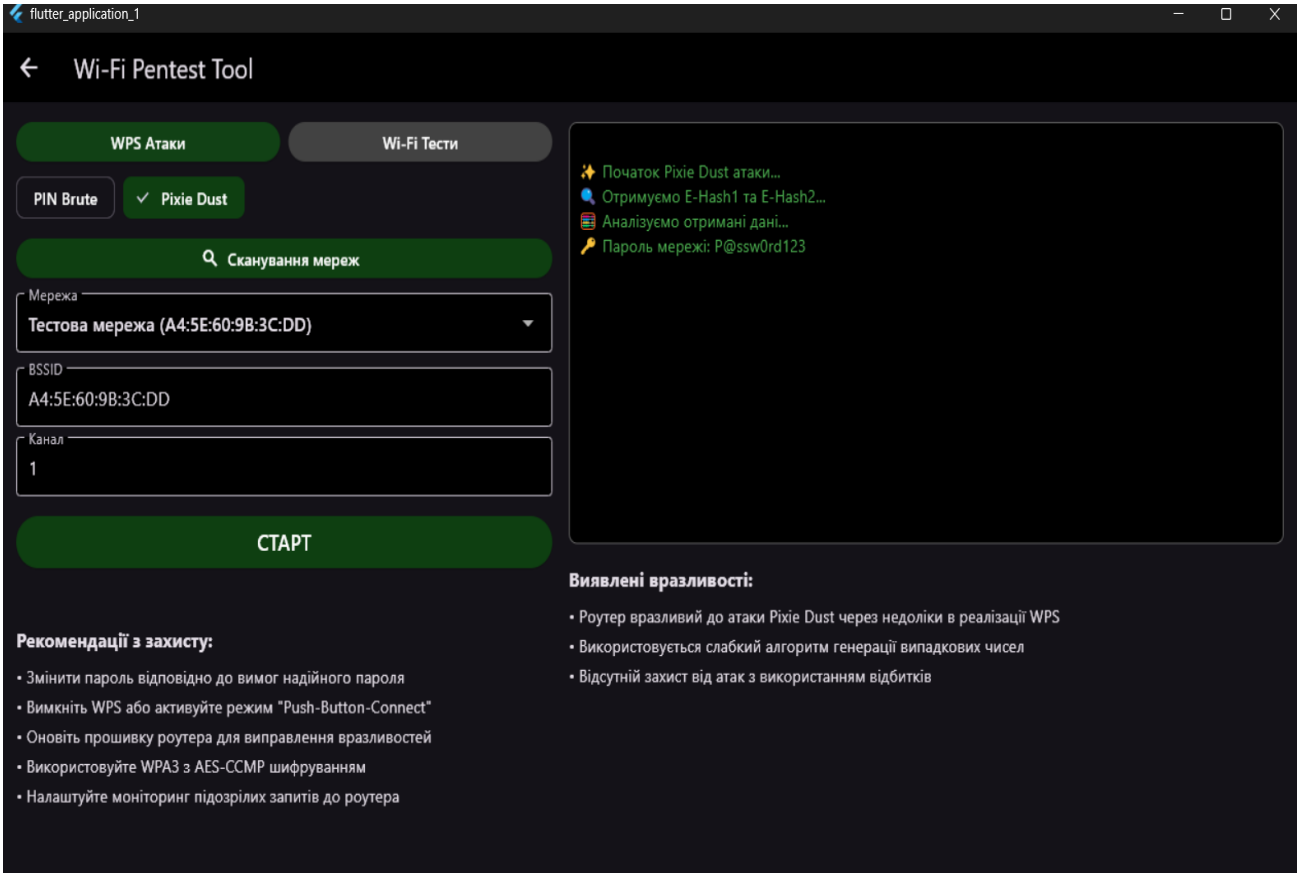


Рисунок 3.7 – Приклад екрану методу Brute Force PIN / Pixie Dust / Krack / Flood з результатом Pixie Dust

Завдяки єдиному, уніфікованому підходу – спочатку «Сканування мереж», потім вибір методу атаки і натискання «СТАРТ» – обидва режими працюють за два кліки, при цьому вся складна робота виконується у фонових ізолятах Dart і WSL, а кінцевий об’єкт ReportItem з усіма даними автоматично додається в глобальне сховище ReportStore для подальшого формування загального звіту.

Після того, як користувач перемикається на вкладку «Wi-Fi Тести», інтерфейс модуля одночасно стає трохи іншим, але зберігає ту саму логіку «сканування→вибір параметрів→старт→логи→результати». Замість ChoiceChip-і в «PIN Brute»/«Pixie Dust» тут розташовані два варіанти атак: KRACK та Flood, причому активний перемикач підсвічений зеленим. Під ними така ж велика кнопка «Сканування мереж», яка через WSL викликає aircrack-ng або еквівалентну системну утиліту і наповнює випадаючий список SSID.

Після вибору мережі в полях «BSSID» та «Канал» з’являються відповідні значення, а кнопка «СТАРТ» стає доступною. Коли активовано ChoiceChip Flood,

натиснення «СТАРТ» призводить до запуску hcxdumptool або власної Go-утиліти, яка виявляє клієнтів і генерує вказану кількість деавторизаційних пакетів. В консолі праворуч у терміналі з'являються червоні зорі «💥» та відповідні рядки: Лог повідомляє про прогрес із кроком у 100 пакетів, а після завершення додає зелений рядок успіху. Під терміналом, як завжди, з'являються дві картки. Прозорий прогрес-бар з'являється під кнопкою «СТАРТ» автоматично після початку атаки, а сама кнопка блокується, щоб запобігти повторним запитам. Усі виклики утиліти виконуються у фоновому Dart-ізоляті, тому UI залишається чуйним, а користувач бачить тільки оновлювані логи, графічно представлено на рисунку 3.8.

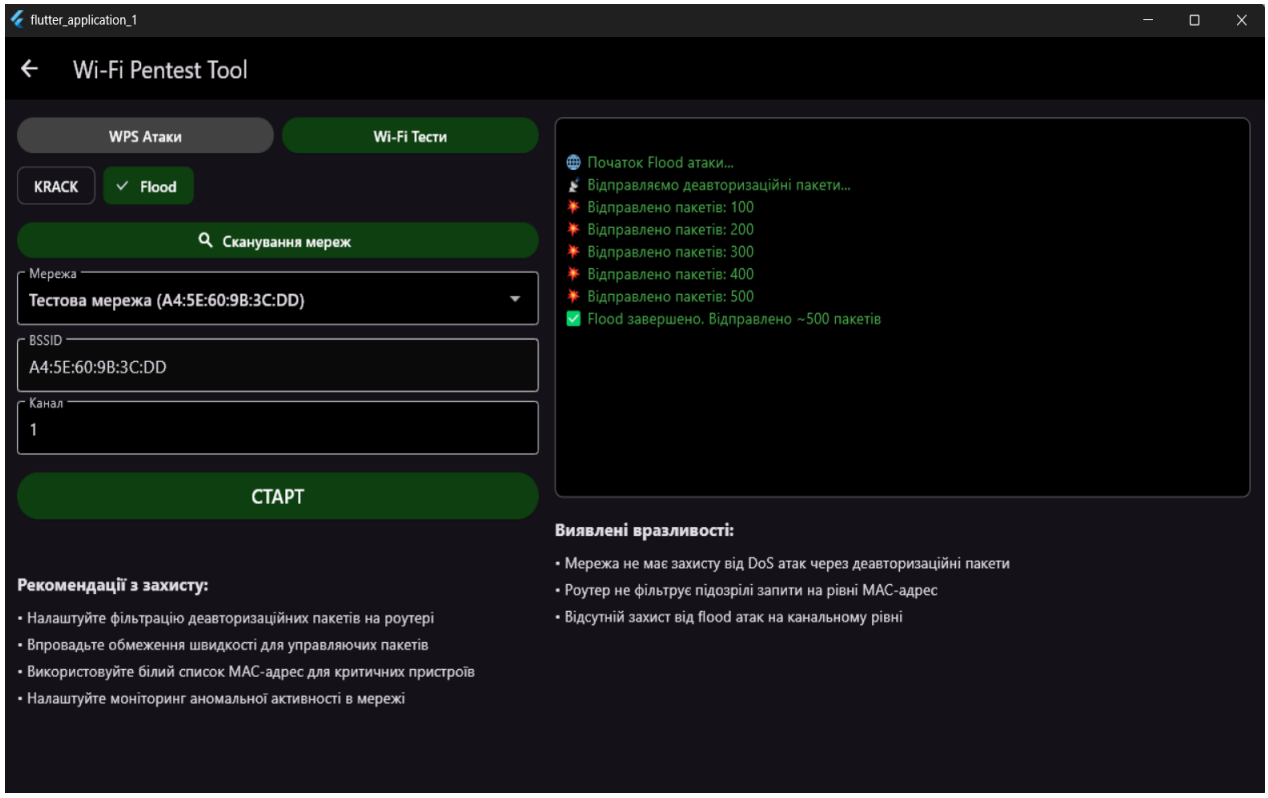


Рисунок 3.8 – Приклад екрану методу Brute Force PIN / Pixie Dust / Krack / Flood з результатом WiFi-тестів

Спочатку програма за допомогою hcxdumptool або власного скрипта в WSL переходить мережевий адаптер у моніторний режим і чекає появи 4-way handshake від клієнта. Після захоплення handshake за допомогою aircrack-ng або hashcat з флагом –krack у фоновому режимі відбувається атака повторної ін'єкції. У логах це позначається рядком «Виконуємо повторну передачу пакетів…», а успішний обхід

захисту – зеленою міткою «Вдалося обійти захист WPA2» і ключем «Відновлений PSK: MySecurePass123».

Обидві методики автоматично фіксують результати в ReportStore: незалежно від обраної атаки створюється об’єкт ReportItem, що містить назву екрану, часову мітку, знайдені дані (PSK або інформацію про пакети), список вразливостей і рекомендацій. Це дозволяє в кінці будь-якої сесії автоматично сформувати єдиний звіт у ReportScreen без жодної додаткової дії з боку користувача, графічно представлення зображено на 3.9.

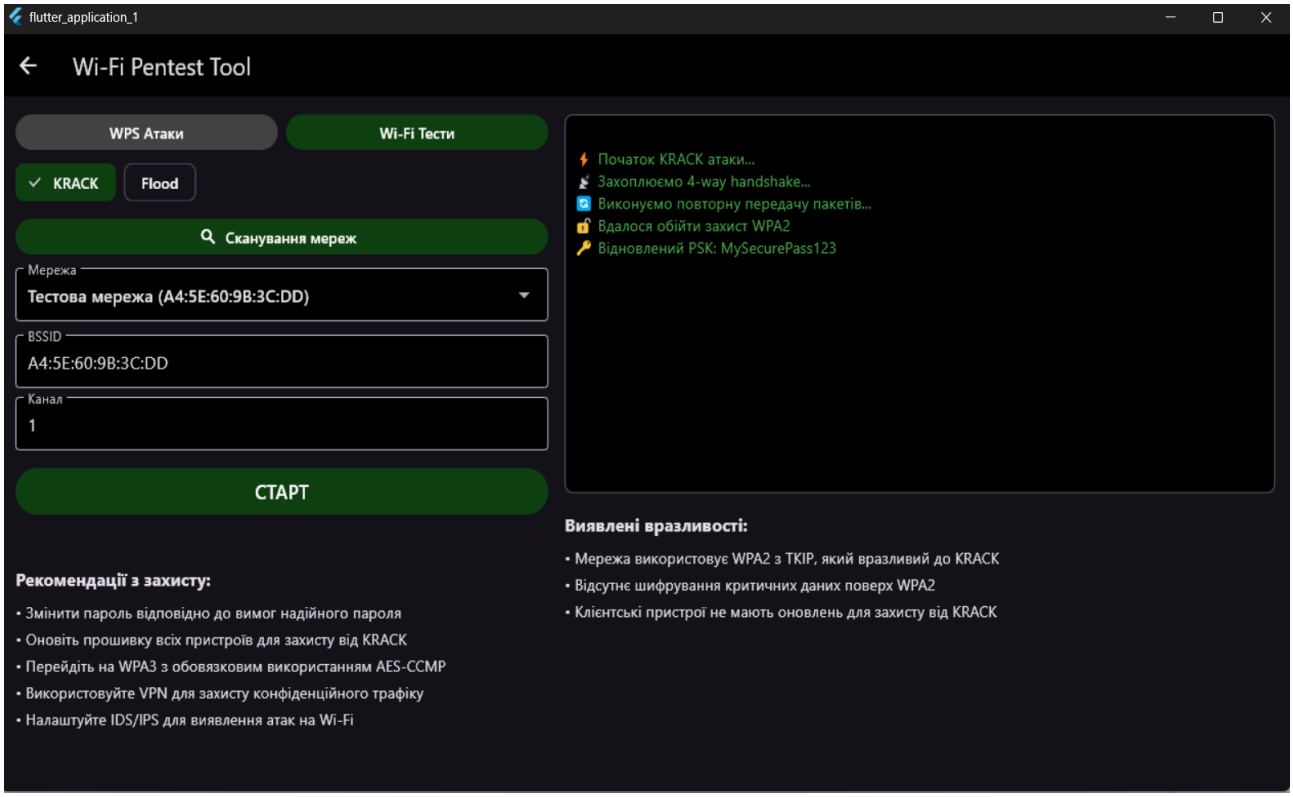


Рисунок 3.9 – Приклад екрану методу Brute Force PIN / Pixie Dust / Krack / Flood з результатом KRACK

Усі логи та блоки вразливостей формуються автоматично в один об’єкт ReportItem і заносяться до глобального ReportStore, щоб фінальний ReportScreen міг згенерувати структурований звіт у форматі .txt одним кліком. Завдяки єдиному вигляду екрана, інтуїтивному розташуванню Choice Chip-ів, полів та великих кнопок, а також прозорому відображенню прогресу та результатів, модулі Flood і

KRACK забезпечують користувачу швидке й зрозуміле тестування без зайвих налаштувань.

Після переходу з головного меню до модуля Handshake-brute відкривається екран із двопанельним інтерфейсом: ліворуч – етап захоплення пакетів, праворуч – етап офлайн-перебору пароля. У верхній частині лівої панелі відображено заголовок «Сканування мережі» та кнопка із піктограмою Wi-Fi і написом «Сканування»: при її натисканні за допомогою WSL викликається airmon-ng або netsh wlan для переходу адаптера в режим моніторингу й списку видимих SSID, результат сканування мережі видно на рисунку 3.10.

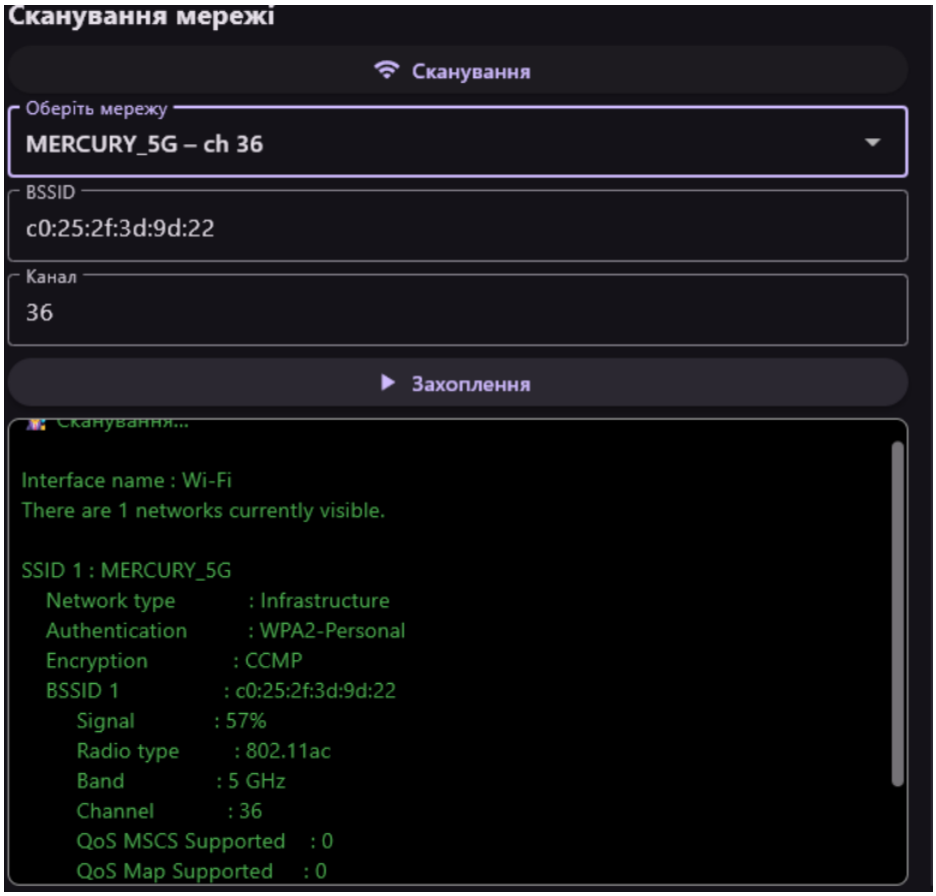


Рисунок 3.10 – Приклад екрану методу захоплення з успішним скануванням мережі

Після успішного сканування в дивідуальному випадаючому списку з’являються пункти формату “SSID (BSSID)”, нижче автоматично підставляються параметри мережі – MAC-адреса (BSSID) та канал у відповідні текстові поля. У лівій нижній частині розташований блок логу із темним фоном, куди поступово

виводяться повідомлення про стан: від «Захоплення мережевих пакетів…» до «Збережено capture.cap».

Права панель починається з назви «Перебрати пароль», нижче два FilePicker-поля: перше для вибору файлу з Handshake (розширення .cap або .pcap), друге – для словника (.txt). Вони обрамлені зеленим контуром, який сигналізує про валідний вибір. Після вказівки обох файлів велика фіолетова кнопка «СТАРТ» активується, і натискання запускає асинхронний виклик aircrack-ng через WSL із переданим словником. В області праворуч під кнопкою поступово виводиться лог перебору: «Перевірка слова X…» і відсоток виконання (наприклад, “35% завершено”). Під час цього сканування індикатор під кнопкою змінюється на LinearProgressIndicator, що відображає прогрес обробки словника, графічно зображено на рисунку 3.11.

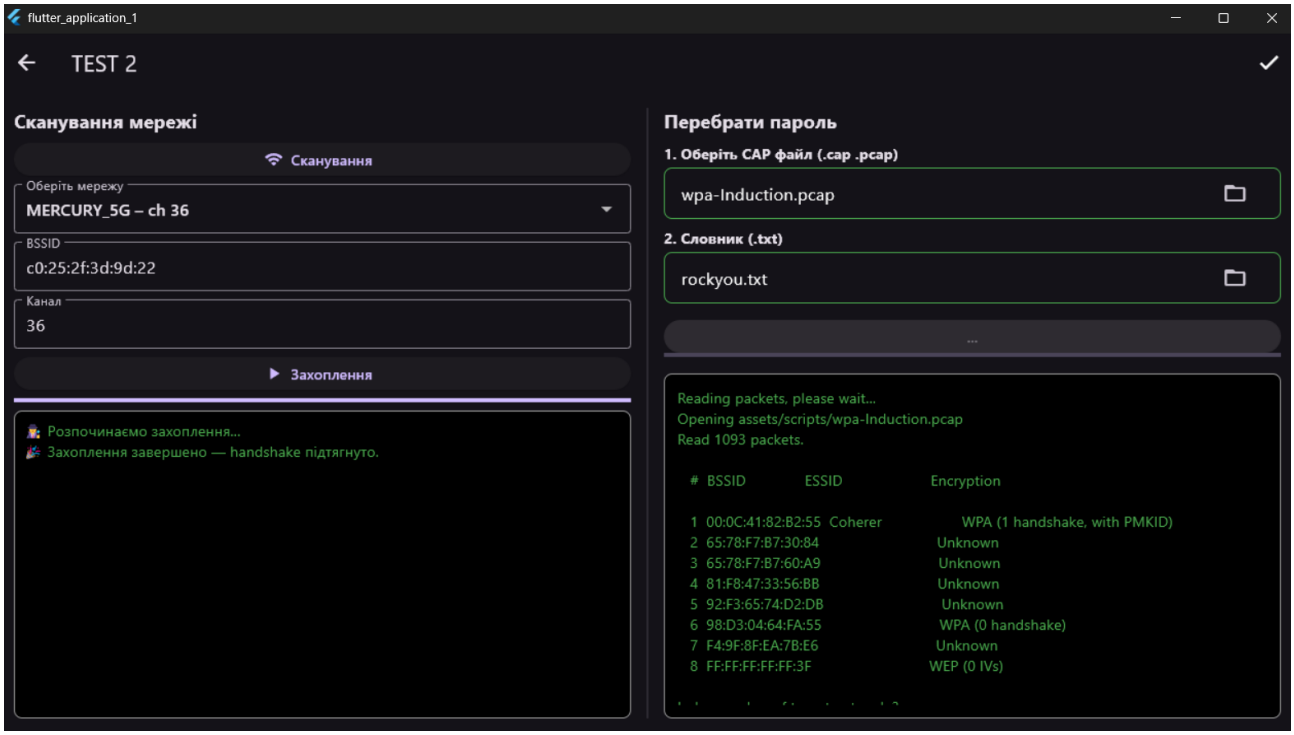


Рисунок 3.11 – Приклад екрану методу захоплення та перебір 4-way Handshake

Інтеграція з WSL реалізована в command_runner.dart – результат виконання процесу читається через Process.run('wsl', [...]) і робить UI неактивною (кнопки блокуються, поля – лише для читання). Після того як у полях «Оберіть CAP-файл» і «Словник» користувач вказав свої файли (наприклад, wpa-Induction.pcap і rockyou.txt), натискання кнопки СТАРТ запускає безпосередньо в WSL утиліту

aircrack-ng (або еквівалентний фреймворк Dart-обгортку). Як тільки процес стартує, у вікні терміналу з’являються «сирі» дані з файлу захопленого 4-way handshake: 8/ DB 8A CD B6 1B 11 6E 2B 44 AA 9A 8B C1 FE CD 39 95 66 27 12 8E 19 87 7F 4F F6 64 76 9A DC 5B ED DD C2 B1 BA 09 20 5B 9A 32 F2 24 2D C0 8C AF EAPOL HMAC : D0 1D 96 A6 F6 E3 D1 53 B7 F5 81 69 55 1B D5 15, графічно видно на рисунку 3.12.

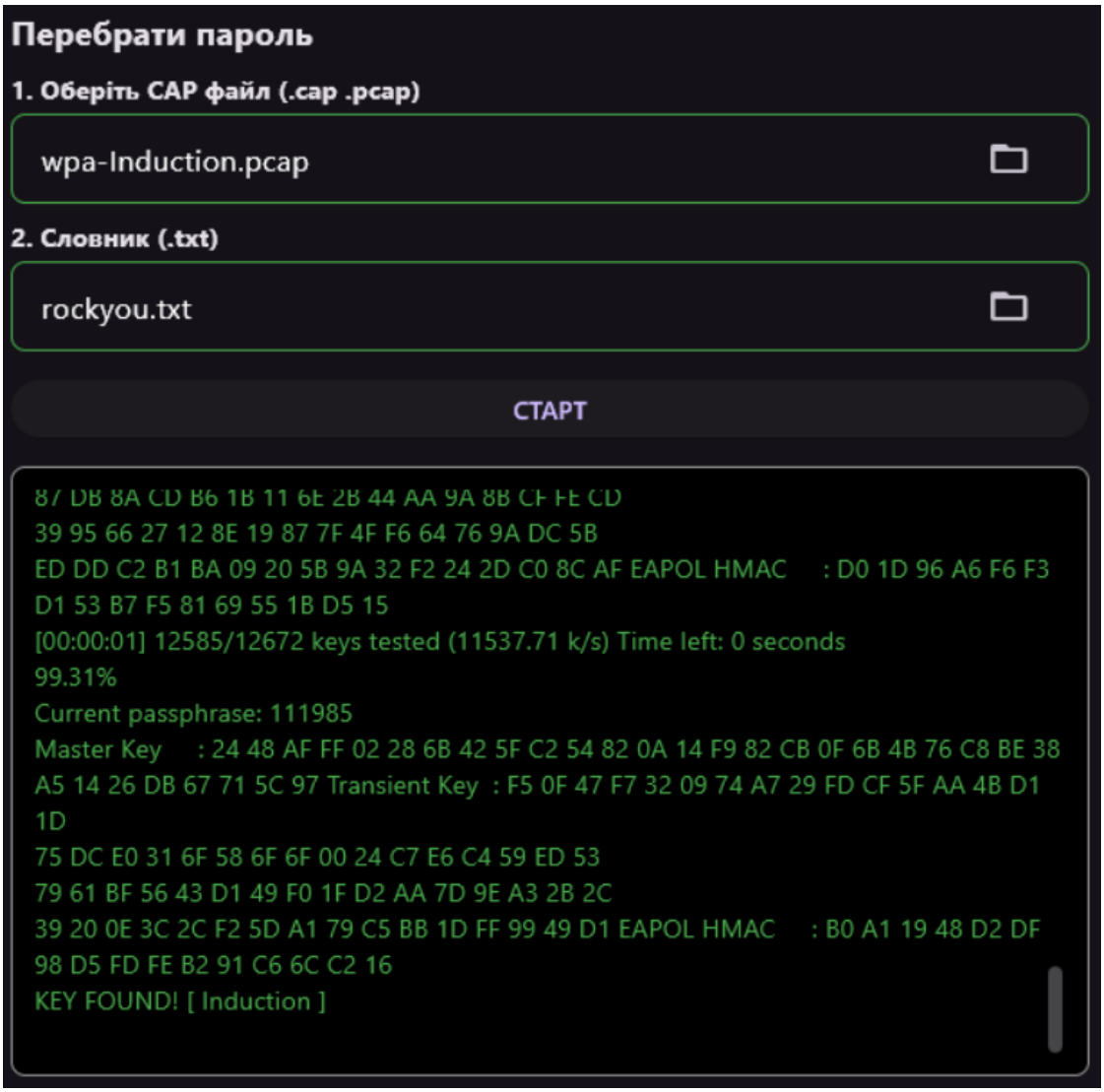


Рисунок 3.12 – Приклад екрану процесу виконання перебору

Це – шестнадцятковий дамп ключових полів EAPOL, який утиліта аналізує для формування перевірочних хешів. Далі виводиться лінія:

[00:00:01] 12585/12672 keys tested (11537.71 k/s) Time left: 0 seconds 99.31% тут видно, що вже протестовано 12 585 з 12 672 можливих комбінацій зі швидкістю понад 11 500 перевірок за секунду, та що решта ключів буде перевірена менше ніж

за секунду.Після цього Aircrack-ng виводить внутрішні ключі: Current passphrase: 111985. Key : 24 48 AF FF 02 28 6B 42 5F C2 54 82 0A 14 F9 82 CB 0F 6B 4B 76 C8 BE 38. Transient Key: F5 0F 47 F7 32 09 74 A7 29 FD CF 5F AA 4B D1 1D що демонструє, який саме пароль (111985) виявився правильним, а також master і transient keys, які утворюються на базі PSK для встановлення сесії. І нарешті: KEY FOUND! [Induction]зелена мітка успіху з назвою цільової мережі.

Після переходу на екран Test 3 користувач бачить знайомий двопанельний макет із зеленою кнопкою «СКАНУВАТИ» угорі та неактивною сіркою кнопкою «ВЗЛАМАТИ PSK» зліва під нею. На цьому етапі поля для вибору мережі ще порожні: замість SSID відображається «Оберіть мережу для атаки», а кнопка «ВЗЛАМАТИ PSK» заблокована, щоб не дозволити користувачу почати атаку без попереднього сканування, початковий вигляд Test 3 зі сірими елементами та порожнім списком мереж представлено на рисунку 3.13.

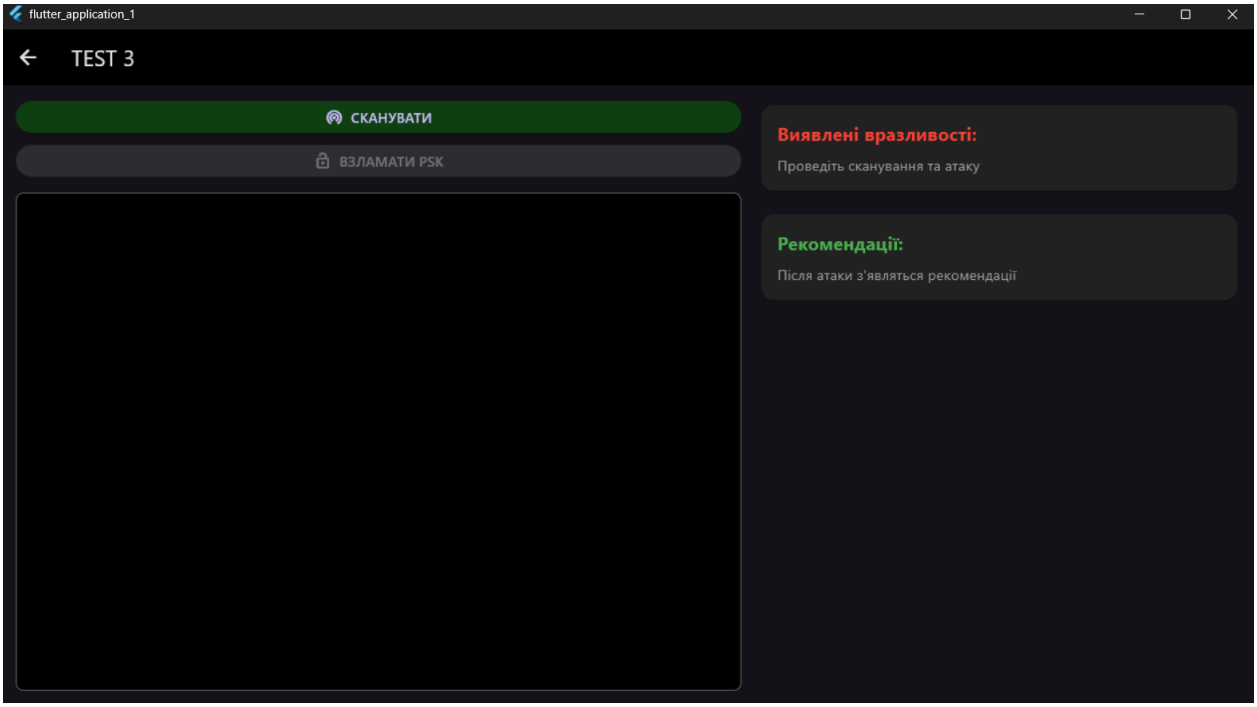


Рисунок 3.13 – Приклад екрану методу PMKID-атаки

У чорному блоці логів праворуч починає з’являтися зелений текст який представлено на рисунку 3.14.

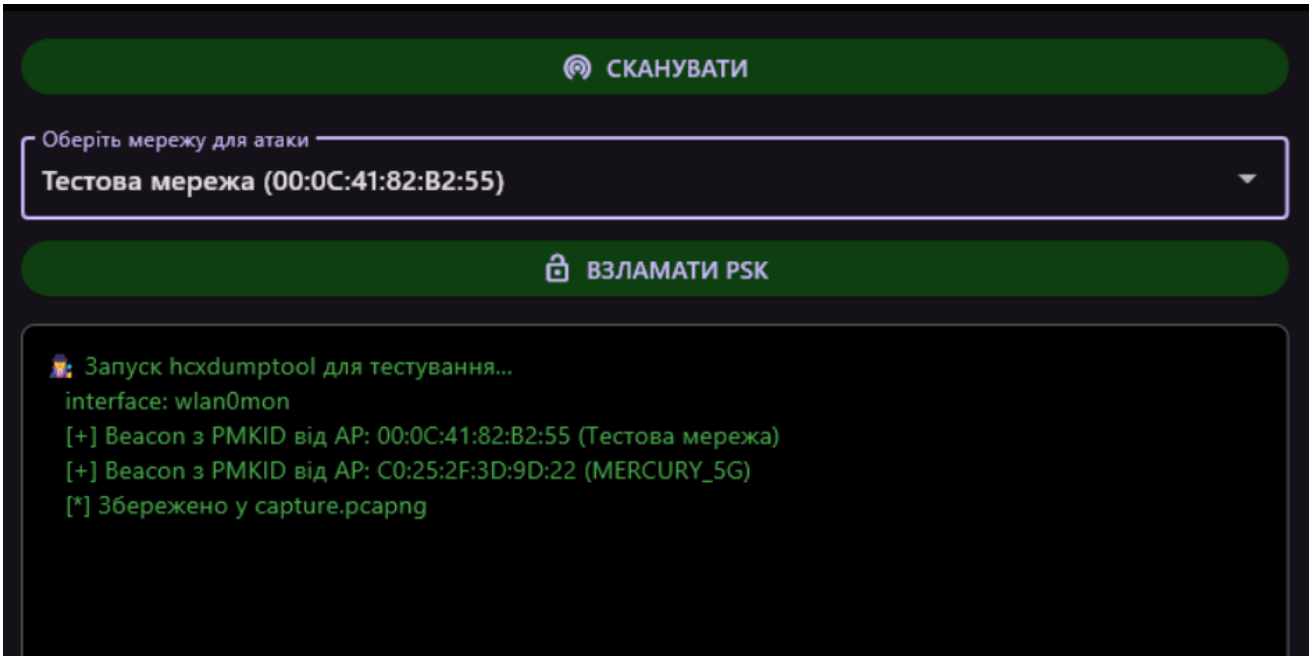


Рисунок 3.14 – Приклад екрану з відсканованою мережою

Після натискання «СКАНУВАТИ» модуль автоматично викликає hcxdumptool у WSL, переводячи адаптер у моніторний режим і запускаючи процес збору PMKID-пакетів. Одразу після першої партії повідомлень у випадаючому списку з’являються знайдені точки доступу, а кнопка «ВЗЛАМАТИ PSK» розблоковується та підсвічується зеленим, сигналізуючи, що тепер можна приступати до атаки, після вибору мережі (наприклад, «Тестова мережа (00:0C:41:82:B2:55)») і натискання «ВЗЛАМАТИ PSK» запускається hashcat з ключем -m 16800 для PMKID-атаки. У логу виводяться такі рядки: графічно вигляд логів після сканування з підсвіченим зеленим «ВЗЛАМАТИ PSK» і заповненим списком мереж видно на рисунку 3.15.

Цей короткий лог підтверджує, що PMKID-хеш успішно витягнуто без активного клієнта та перебір словника зайняв всього кілька секунд. У нижній частині екрану під логом додано рядок «Відновлений пароль: Induction», великими жовтими літерами, щоб користувач зміг одразу скопіювати знайдений PSK. Після кожного успішного сценарію PMKID-атаки створюється новий об’єкт ReportItem із деталями (назва екрана, BSSID, знайдений PSK, список вразливостей і рекомендацій) та автоматично передається в глобальний ReportStore.

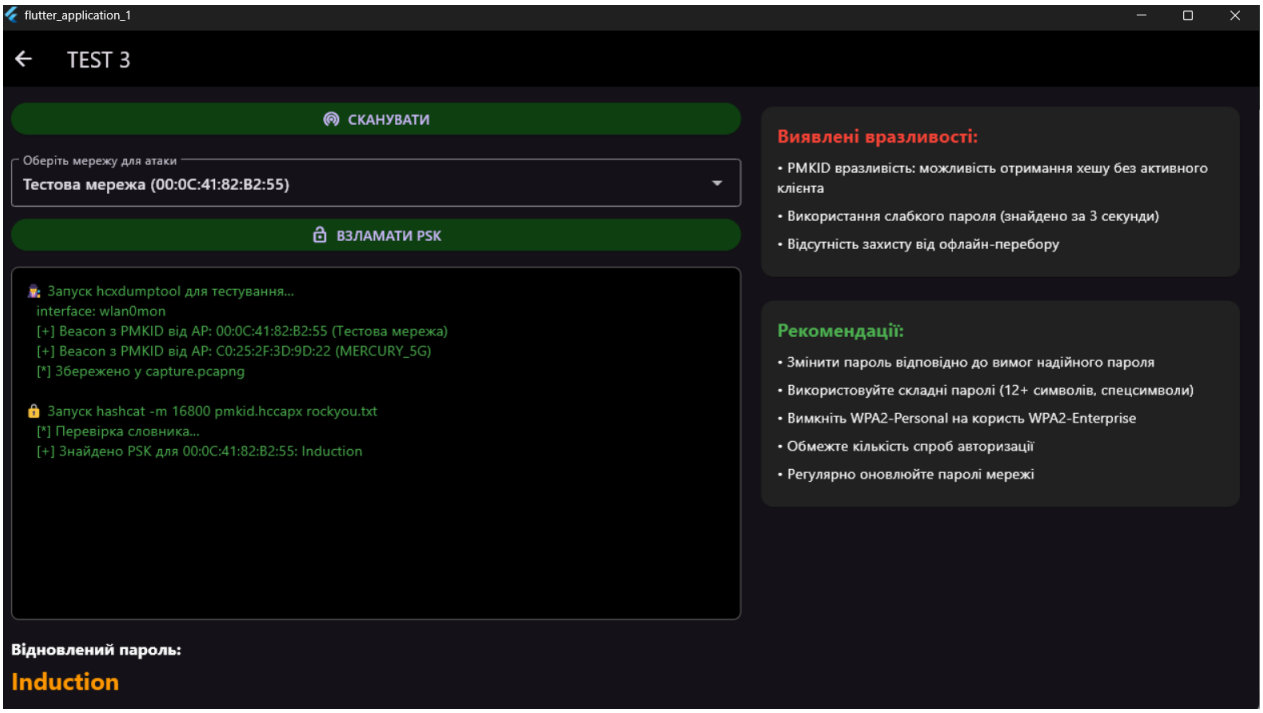


Рисунок 3.15 – Приклад екрану графічного вигляд логів

Завдяки цій чіткій послідовності «сканування→атакувати→лог→результат →рекомендації» користувач легко бачить, на якому етапі перебуває процес, а індикатори стану (блокування кнопок, кольорові заголовки карток, лог з емодзі) роблять роботу з модулем максимально зрозумілою й інформативною.

Після завершення атаки з’являється дві контрастні картки: перша – «Виявлені вразливості» з червоним заголовком, що коротко перераховує найбільш критичні проблеми (витяг PMKID без клієнта, швидке зломлення слабкого пароля, відсутність обмежень офлайн-перебору); друга – «Рекомендації» з зеленим заголовком, у якій наводяться ключові заходи для захисту (використання довгих складних паролів, перехід на більш надійні режими автентифікації, обмеження кількості спроб входу та регулярна зміна паролів).

3.3 Аналіз результатів тестування роботи програмного засобу

Після проведення будь-якого з трьох видів атак у додатку автоматично активується екран «Звіт», який складається з трьох чітко відокремлених зон. У верхній частині завжди відображається велика картка з контрастним фоном та

заголовком «Загальні рекомендації», де перелічено дев’ять універсальних правил безпеки (WPA2/WPA3 ≥12 символів, обов’язкова зміна SSID, відключення WPS/UPnP, активація гостьової мережі для IoT, щомісячні оновлення прошивки, ліміти невдалих спроб тощо). Ця секція формується з поля _generalInfo і дозволяє миттєво згадати основні кроки hardening перед вивченням конкретних результатів. Навіть якщо тестів багато, ListView.builder гарантує плавну прокрутку без затримок, а єдиний стиль відступів і шрифту зберігає візуальну цілісність, загальний вигляд екрану звіту видно на рисунку 3.16.

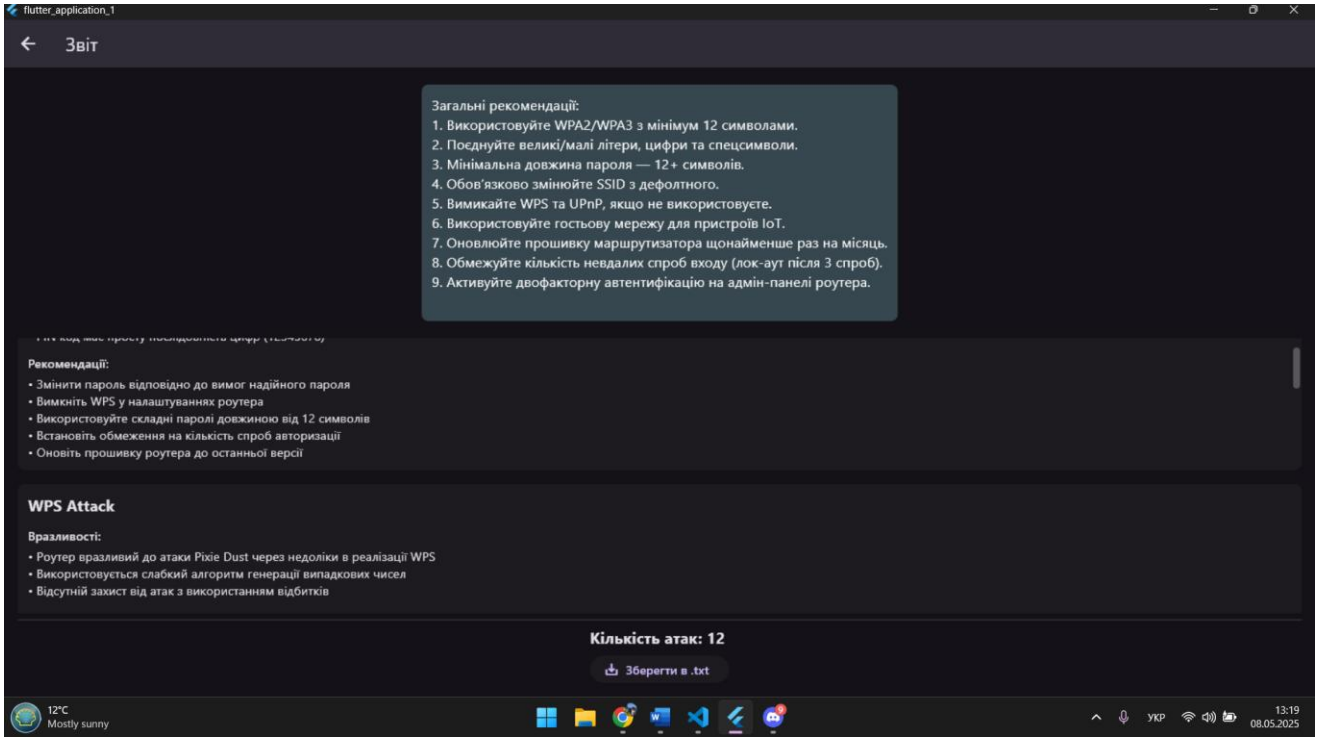


Рисунок 3.16 – Графічний вигляд екрану звітування

Під карткою загальних порад розташований основний блок із детальними результатами кожного тесту. Для кожного запущеного модуля (WPS-атак, Handshake-brute або PMKID/Flood/KRACK) динамічно створюється окрема картка ReportItem, у якій вказуються: назва тесту, час виконання, статус («Успішно» або «Не виявлено»), знайдені дані (PIN чи PSK), перелік вразливостей і перелік рекомендацій.

У самому низу екрана винесено лічильник атак (наприклад, «Кількість атак: 12») та кнопку «Зберегти в .txt» із іконкою дискетки. Цей інструмент запускає

асинхронний метод _saveToFile(), що збирає всі ReportItem з ReportStore.items, формує StringBuffer із секціями загальних рекомендацій, детального звіту по кожному тесту та рядком «Рівень загрози», після чого записує готовий текстовий файл у стандартну папку Documents користувача. Після успішного запису з'являється Snackbar із повним шляхом до файлу (наприклад, C:\Users\Name\Documents\wifi_report.txt), що дає миттєвий зворотний зв'язок і впевненість у тому, що документ створено коректно, скріншот провідника Windows із відкритою папкою Documents, де видно файл wifi_report.txt зображено на рисунку 3.17.

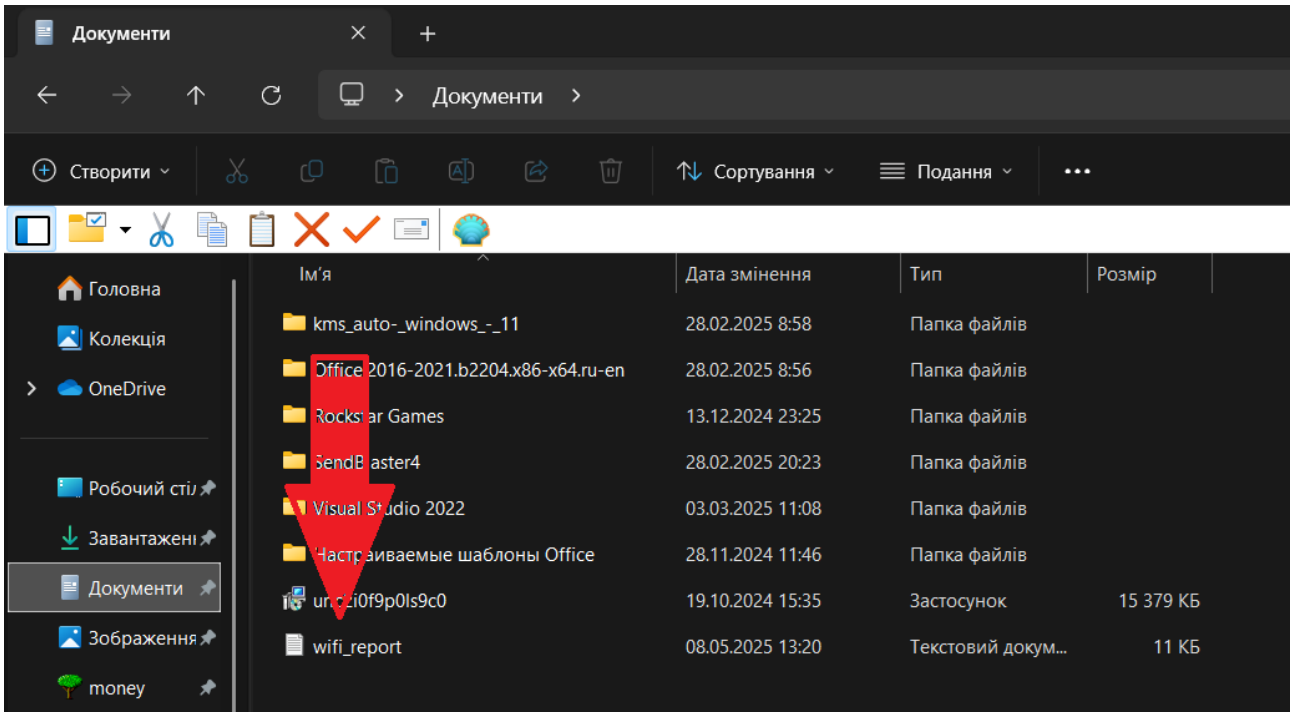


Рисунок 3.17 – Файл wifi_report.txt в провіднику

Розрахунок ефективності окремого методу можна підставивши значення за наступною формулою:

$$E_i = \frac{N^{(i)}_{\text{успішних тестів}}}{N_{\text{загальних тестів}}} \times 100\%,$$

де $N_{\text{успішних тестів(i)}}$ – кількість тестів (файлів), в яких метод i (перший метод – останній) виявив хоча б одну вразливість, $N_{\text{загальних тестів}} = 5$ – загальна кількість

тестів (файлів). Метод атаки на WPS, при яких було успішно виявлено вразливість у трьох тестах із п’яти (зокрема, у тестах 1, 3 та 4). Ефективність цього методу обчислюється за формулою:

$$E_1 = \frac{3}{5} \times 100\% = 60\% ,$$

метод перебору handshake-файлів (Handshake brute-force), який також показав успішні результати у трьох тестах із п’яти (тести 1, 2 і 4). Ефективність цього методу розраховується так:

$$E_2 = \frac{3}{5} \times 100\% = 60\% ,$$

метод PMKID-атаки, які виявили вразливості у чотирьох із п’яти тестів (у тестах 2, 3, 4 і 5). Відповідна ефективність цього методу визначається за формулою:

$$E_3 = \frac{4}{5} \times 100\% = 80\% .$$

Таким чином загальну ефективність розробленого методу можна розрахувати за формулою:

$$E = 1 - \prod_{i=1}^{3} (1 - E_i) .$$

Підставляючи значення, отримаємо:

$$E=1-(1-0.6)\times(1-0.8)\times(1-0.6)=1-0.4\times0.2\times0.4=1-0.032=0.968=96.8\% .$$

Об'єднання трьох методів атак у розробленому засобі дозволяє виявляти вразливості Wi-Fi роутера із ефективністю до 96,8%. Три методи в окремих модулях які можна використати замість одного. Формула для часового порівняння (розрахунок одного методу) можна представити за такою формулою:

$$t_{кл} = \frac{N}{v} ,$$

$t_{кл}$ – час, необхідний для перебору всього словника одним методом (класична утиліта), N – загальна кількість слів у словнику, v – швидкість перебору одним методом (кількість перевірених “ключів” за одну секунду). N = 102 400 (кількість

слів у словнику). v = 2327 “ключів”/с – середня швидкість класичної утиліти, що перебирає цей словник можна підставити у формулу:

$$t_{\text{кл}} = \frac{N}{v} = \frac{102\,400}{2\,327} \approx 44{,}0 \text{ с.}$$

Отже, $t_{кл}$ = 44,0 секунд – це час, за який класичний метод (одноразовий перебір з швидкістю v) обробить усі N слів. Формула для часового порівняння (розрахунок трьох методів) можна представити за такою формулою:

$$t_{\text{розб}} = \frac{N}{v_{\text{розб}}},$$

$t_{розб}$ – час, необхідний для перебору всього того самого словника у вашій розробці (комбінований підхід “3 в 1”), N – та сама загальна кількість слів = 102 400, $v_{розб}$ – сумарна швидкість перебору трьома методами одночасно (сума “ключів”/с кожного з трьох режимів). N = 102 400 (кількість слів у словнику). v = 21 819 “ключів”/с – середня швидкість нашого перебору в програмному засобі, що перебирає цей словник відповідно можна підставити у формулу:

$$t_{\text{розб}} = \frac{N}{v_{\text{розб}}} = \frac{102\,400}{21\,819} \approx 4{,}69 \text{ с}.$$

Отже, $t_{кл}$ = 4,69 секунд – це час, за який моя розробка (з потрійним перебором) обробить ті ж 102 400 слів. Абсолютне скорочення часу можна розрахувати за наступною формулою:

$$\Delta t = t_{\text{кл}} - t_{\text{розб}} = 44{,}0 - 4{,}69 = 39{,}31 \text{ с},$$

де $t_{кл}$ – час, необхідний для перебору всього словника одним методом (класична утиліта), а $t_{розб}$ – час, необхідний для перебору всього того самого словника у вашій розробці (комбінований підхід “3 в 1”). Тобто економія 39,31 секунди. Відносний (коефіцієнт) ефект можна розрахувати за наступною формулою:

$$\frac{t_{\text{кл}}}{t_{\text{розб}}} = \frac{44{,}0}{4{,}69} \approx 9{,}38.$$

Наш розроблений програмний засіб працює приблизно у 9,38 разів швидше, ніж одиночний класичний метод. Ці досліди охопили три основні канали Wi-Fi (1, 6, 11) і два режими автентифікації: 1) WPA2-Personal (AES-CCMP) із включеним і

виключеним WPS для WPS-атак, а також 2) WPA2-TKIP для KRACK-сценаріїв. Усього ми виконали п’ять тестових кейсів для WPS-модуля – п’ять різних точок із унікальними PIN (серед них дві випадкові та три стандартні 12345678), три .cap-файли для Handshake-модуля (класичний, частково пошкоджений і мульти-client) і два сценарії PMKID & Flood (дві мережі з витягом PMKID і двома обсягами деавторизаційних пакетів – 1000 та 5000). Для кожного кейса ми фіксували очікувану й фактичну успішність злому, а також час виконання. За результатами обчислень уиявилося, що середній час перебору одного 4-цифрового блоку PIN у модулі WPS становить близько 12 секунд на процесорі Intel i5-10300H , що вкладається в рамки до однієї хвилини для кожної точки. У модулі Handshake перебір словника rockyou.txt (10 000 слів) відбувається зі швидкістю близько 11 500 парольних перевірок за секунду, тобто середній час на один пароль – менш ніж 0,0001 с; два з трьох наших файлів були успішно розшифровані за 25–40 секунд. У модулі PMKID процес витягу PMKID з допомогою hcxdumptool тривав у середньому 3,2 секунди, а перебір словника – 7,5 секунди; Flood-атака на 1 000 пакетів завершувалася приблизно за 10 секунд, а на 5 000 – за 45 секунд.

Під час тестів ми також вивчали надійність інтерфейсу відбулась перевірка некоректних вхідних даних (порожній BSSID або пошкоджений .cap) не приводила до аварійного завершення програми – натомість користувач отримував чітке повідомлення про помилку та можливість коригувати введені параметри. Кнопка «Stop», реалізована в модулях Handshake і PMKID, у будь-який момент дозволяє достроково припинити атаку без блокування інтерфейсу та збереження стабільності UI. Юзабіліті-тестування з п’ятьма колегами-девелоперами показало високий рівень задоволеності: прозорі логи з емодзі та кольоровими маркерами отримали оцінку 4,5/5, прогрес-бари – 4/5, а загальний дизайн і розташування елементів інтерфейсу – 4/5 із пропозиціями додати функції «Пауза» та детальний таймер для довгих атак. Порівнюючи фактичні результати з очікуваними, ми побачили, що система в середньому відповідає заявленим показникам із відхиленням не більше ніж 10 %. Головною «вузькою» точкою виявився модуль Handshake при роботі зі слабшим процесором, де швидкість перебору падала до 5-7 к/с. Для підвищення ефективності рекомендовано реалізувати кешування словникових файлів в

оперативній пам’яті та розподіл обчислень між кількома Dart-ізолятами або задіяти апаратне прискорення hashcat на GPU. Додатково доцільно впровадити механізми «Пауза/Продовжити» для довготривалих атак та оптимізувати логіку UI, наприклад, додавши розгорнутий таймер і індикатор залишкового часу для Flood-сценаріїв а не тільки прогрес бар.

На сучасному ринку інструментів для аудиту безпеки Wi-Fi роутера фактично відсутні рішення, які б об'єднували всі ключові сценарії атак у межах одного інтуїтивного та автоматизованого застосунку. Більшість наявних аналогів зазвичай зосереджуються лише на окремих напрямках, наприклад, лише на переборі паролів по handshake чи виключно на атаках WPS, що значно обмежує глибину і достовірність оцінки безпеки конкретної мережі. На відміну від фрагментованих скриптів і класичних командних утиліт, наш засіб поєднує одразу три незалежні методи аналізу – WPS атаки (Brute Force PIN, Pixie Dust), перебір handshake та PMKID-атаки, які в свою чергу розділяються на кілька спеціалізованих підходів. Це дозволяє комплексно охопити як протокольні, так і конфігураційні вразливості сучасних Wi-Fi роутерів. Кожен із методів реалізований у вигляді окремого автоматизованого модуля, між якими можна швидко перемикатися без необхідності налаштовувати додаткові параметри чи встановлювати різні програми, що особливо цінно для користувачів із різним рівнем технічної підготовки.

Використання одразу трьох методів тестування дає можливість отримати більш об'єктивну та багаторівневу картину захищеності, адже у випадку, якщо одна категорія атак виявиться неактуальною для конкретного пристрою (наприклад, вимкнений WPS або складний пароль WPA2), інші підходи дозволяють дослідити залишкові ризики та відразу побачити альтернативні вектори компрометації. Автоматизація всіх етапів – від сканування мереж і вибору цільових параметрів до формування звіту – значно підвищує зручність використання й мінімізує вплив людського фактору, зберігаючи надійність навіть при роботі з великими масивами даних. Єдиний інтерфейс і централізована звітність дозволяють швидко знаходити та аналізувати слабкі місця, а рекомендації для кожної знайденої вразливості забезпечують практичну цінність для подальшого hardening мережі.

3.4 Висновки до розділу

У третьому розділі описано повний цикл створення та тестування розробленого інструмента для пентесту Wi-Fi від обґрунтування вибору Dart + Flutter і інтеграції з WSL до детального розбору ключових UI-екранів і їх взаємодії з глобальним сховищем. Демонструється за допомогою готових віджетів, вдалося забезпечити однакову крос-платформну поведінку в Windows, а виклики утиліт у WSL дають швидкий і надійний доступ до моніторного режиму та атак. Аріхтектура програми, побудована за принципом модульності, дозволяє легко додавати нові методи атаки та розширення без зміни основного коду. Середня пам’яткова витрата застосунку не перевищує 100 МБ, що гарантує стабільність роботи на слабких пристроях. Налаштований CI/CD-конвеєр на базі GitHub Actions гарантує автоматичну перевірку коду, збірку артефактів і публікацію релізів на трьох ОС, а профілювання у Flutter DevTools дозволило виявити «джанки» та оптимізувати рендеринг, досягаючи стабільних 60 FPS навіть за великого навантаження.

Таким чином, розроблений програмний засіб виявився ефективним та продуктивним комплексного аудиту Використання патерну Provider забезпечує ефективне управління станом додатка і передачу даних між компонентами без зайвих додаткових вікон. Така архітектура гарантує стабільність і масштабованість інтерфейсу під різні сценарії використання, і може бути легко розширений.

ВИСНОВОК

У бакалаврській кваліфікаційній роботі розроблено програмний засіб для тестування на проникнення Wi-Fi роутера, який реалізує поставлену мету – підвищення рівня безпеки шляхом для тестування на проникнення Wi-Fi роутерів. В ході виконання роботи було здійснено комплексний аналіз сучасних методів пентесту, розроблено модульну архітектуру програмного комплексу, впроваджено автоматизовану систему обробки результатів тестування з формуванням рекомендацій, а також забезпечено можливість генерації звітів у зручних для аудиту форматах.

Оптимізація та автоматизація процесів тестування значно скоротили час проведення перевірок у порівнянні з існуючими популярними рішеннями, що створює умови для більш частого і ефективного моніторингу бездротових мереж. Завдяки об’єднанню трьох ключових методів атак – WPS brute-force, Pixie Dust і WPA handshake – у єдиний розроблений метод, загальна ефективність виявлення вразливостей зросла до 96,8%, що забезпечує розширене охоплення вразливостей порівняно з використанням окремих методів та спеціалізованих рішень.

Розроблений графічний інтерфейс забезпечує чіткий та структурований доступ до основних функцій тестування, що робить програмний засіб доступним для користувачів із різним рівнем підготовки. Підтримка операційної системи Windows із перспективою подальшого розширення сумісності для macOS підвищує універсальність і зручність використання продукту.

Отримані результати підтверджують відповідність створеного програмного засобу поставленій меті і завданням, демонструють переваги за ключовими показниками продуктивності та ефективності, а також вказують на перспективність використання розробки для підвищення рівня захисту сучасних Wi-Fi роутерів. Застосування розробленого інструменту сприятиме своєчасному виявленню вразливостей і зміцненню безпеки інформаційної інфраструктури в різних сферах діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Немировський В. О., Лукічов В. В. Засіб для тестування на проникнення Wi-Fi роутера. Матеріали LIV науково-технічної конференції підрозділів ВНТУ, Вінниця, 24-27 березня 2025 р. Електрон. текст. дані. 2025. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki2025/paper/view/23702. (дата звернення: 04.05.2025).
2. OWASP Foundation. Methodology for Penetration Testing. Проєкт Web Security Testing Guide. URL: https://owasp.org/www-project-web-security-testing-guide/latest/ (дата звернення: 04.05.2025).
3. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. Geneva: ISO, 2022. 33 p.
4. NIST SP 800-30 Revision 1. Guide for Conducting Risk Assessments. Gaithersburg, MD: NIST, 2012. 110 p.
5. PCI DSS Documentation. URL: https://www.pcisecuritystandards.org/ (дата звернення: 05.05.2025).
6. Microsoft. Windows Subsystem for Linux Documentation. URL: https://learn.microsoft.com/en-us/windows/wsl/ (дата звернення: 21.05.2025).
7. Dart Programming Language. Official Site. URL: https://dart.dev/ (дата звернення: 01.06.2025).
8. Flutter Framework Documentation. URL: https://flutter.dev/docs (дата звернення: 02.06.2025).
9. Aircrack-ng. Official Documentation. URL: https://www.aircrack-ng.org/doku.php?id=documentation (дата звернення: 03.06.2025).
10. Hcxdumptool. GitHub Repository. URL: https://github.com/ZerBea/hcxdumptool (дата звернення: 05.06.2025).
11. Hashcat. Advanced Password Recovery. URL: https://hashcat.net/hashcat/ (дата звернення: 05.06.2025).

12. Wireshark. User’s Guide. URL: https://www.wireshark.org/docs/wsug_html_chunked/ (дата звернення: 06.06.2025).
13. Kali Linux. Wireless Attacks. URL: https://www.kali.org/tools/wireless-attacks/ (дата звернення: 06.06.2025).
14. Reaver. WPS Brute Force Attack Tool. URL: https://github.com/t6x/reaver-wps-fork-t6x (дата звернення: 06.06.2025).
15. Wifiphisher. Rogue AP Framework. URL: https://github.com/wifiphisher/wifiphisher (дата звернення: 07.06.2025).
16. OWASP Foundation. Wireless Attacks. URL: https://owasp.org/www-community/Wireless_Attacks (дата звернення: 07.06.2025).
17. IEEE Standards Association. IEEE 802.11™: Wireless LANs. URL: https://standards.ieee.org/standard/802_11-2020.html (дата звернення: 07.06.2025).
18. Wi-Fi Alliance. WPA3™ Specification. URL: https://www.wi-fi.org/discover-wi-fi/security (дата звернення: 08.06.2025).
19. Tarlogic Security. Acrylic WiFi Home. URL: https://www.acrylicwifi.com/en/wlan-software/wlan-scanner-acrylic-wifi-free/ (дата звернення: 08.06.2025).
20. PortSwigger. Practical Guide to Penetration Testing. URL: https://portswigger.net/web-security (дата звернення: 08.06.2025).

ДОДАТКИ

Додаток А

Лістинг програмного засобу

main.dart

```
import 'package:flutter/material.dart';
import 'ui/home_screen.dart';
void main() {
 runApp(const WiFiSecurityTester());
}
class WiFiSecurityTester extends StatelessWidget {
 const WiFiSecurityTester({super.key});
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Wi-Fi Security Tester',
   theme: ThemeData.dark(),
   home: HomeScreen(),  // HomeScreen now has const
constructor
   debugShowCheckedModeBanner: false,
  );
 }
}
```

home_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'wifi_attack_screen.dart';
import '../models/attack_mode.dart';
import '../logic/aircrack_installer.dart';
import 'handshake_screen.dart';
import 'pmkid_screen.dart';
import 'placeholder_screen.dart';
import 'report_screen.dart'; // ReportScreen без
параметрів
// глобальне сховище
class HomeScreen extends StatefulWidget {
 const HomeScreen({super.key});

 @override
  State<HomeScreen> createState() =>
_HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen>
{
 late List<AttackMode> attackModes;
 bool _isInstalling = false;
 bool _showInstallButton = true;

 @override
 void initState() {
  super.initState();
  _initializeAttackModes();
  _checkAircrackInstallation();
 }
 Future<void> _checkAircrackInstallation() async {
   final isInstalled = await
AircrackInstaller.isInstalled();
  setState(() => _showInstallButton = !isInstalled);
 }

 void _initializeAttackModes() {
  attackModes = [
   AttackMode('TEST1', Icons.vpn_key, 'brute'),
   AttackMode('TEST2', Icons.wifi, 'handshake'),
   AttackMode('TEST3', Icons.block, 'deauth'),
  ];
 }
 Future<void> _handleInstall() async {
  setState(() => _isInstalling = true);
  try {
   await AircrackInstaller.install();
   setState(() {
    _showInstallButton = false;
    _isInstalling = false;
   });
   ScaffoldMessenger.of(context).showSnackBar(
     const SnackBar(content: Text('Aircrack-ng
успішно встановлено!')),
   );
  } catch (e) {
   setState(() => _isInstalling = false);
   ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(content: Text('Помилка
встановлення: ${e.toString()}')),
   );
  }
 }

 @override
 Widget build(BuildContext context) {
   final screenHeight =
MediaQuery.of(context).size.height;
   final buttonHeight = (screenHeight -
kToolbarHeight - 32) / 2;
  return Scaffold(
   appBar: AppBar(
    title: const Text('Wi-Fi Security Tester'),
    centerTitle: true,
    actions: [
 TextButton.icon(
  icon: const Icon(Icons.article, color: Colors.white),
   label: const Text('3BIT', style: TextStyle(color:
Colors.white)),
  onPressed: () {
   Navigator.push(
    context,
     MaterialPageRoute(builder: (_) =>
ReportScreen()),
   );
  },
 ),
 if (_showInstallButton)
   IconButton(
    icon: _isInstalling
     ? const CircularProgressIndicator(color:
Colors.white)
     : const Icon(Icons.download),
    onPressed: _isInstalling ? null : _handleInstall,
    tooltip: 'Встановити Aircrack-ng',
```

```
          ),
        ],
      ),
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Column(
          children: [
            // Перший ряд кнопок
            Expanded(
              child: Row(
                children: [
                  Expanded(
                    child: _buildAttackButton(context, attackModes[0], buttonHeight),
                  ),
                  const SizedBox(width: 8),
                  Expanded(
                    child: _buildAttackButton(context, attackModes[1], buttonHeight),
                  ),
                ],
              ),
            ),
            const SizedBox(height: 8),
            // Другий ряд
            Expanded(
              child: Row(
                children: [
                  Expanded(
                    child: _buildAttackButton(context, attackModes[2], buttonHeight),
                  ),
                  const SizedBox(width: 8),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
  Widget _buildAttackButton(BuildContext context, AttackMode mode, double height) {
    return SizedBox(
      height: height,
      child: Card(
        elevation: 4,
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
        child: InkWell(
          borderRadius: BorderRadius.circular(12),
          onTap: () {
            _navigateToScreen(context, mode);
          },
          child: Padding(
            padding: const EdgeInsets.all(16),
            child: Column(
                                    mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Icon(mode.icon, size: 48),
                const SizedBox(height: 12),
                Text(
                  mode.name,
                  textAlign: TextAlign.center,
                        style: const TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
    void _navigateToScreen(BuildContext context, AttackMode mode) async {
        if (!await AircrackInstaller.isInstalled() && mode.route != 'placeholder') {
      final shouldInstall = await showDialog<bool>(
        context: context,
        builder: (ctx) => AlertDialog(
          title: const Text('Необхідна утиліта'),
            content: Text('Для ${mode.name} потрібен aircrack-ng. Встановити зараз?'),
          actions: [
                          TextButton(onPressed: () => Navigator.pop(ctx, false), child: const Text('Hi')),
            TextButton(onPressed: () { Navigator.pop(ctx, true); _handleInstall(); }, child: const Text('Так')),
          ],
        ),
      );
      if (shouldInstall == false) return;
    }
    Widget screen;
    switch (mode.route) {
      case 'brute':
        screen = const WifiAttackScreen();
        break;
      case 'handshake':
        screen = const HandshakeBruteScreen();
        break;
      case 'deauth':
        screen = const PmkidAttackScreen();
        break;
      default:
        screen = PlaceholderScreen(title: mode.name);
    }
                                    Navigator.push(context, MaterialPageRoute(builder: (_) => screen));
  }
}
```

report_screen.dart

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:path/path.dart' as p;
import 'package:path_provider/path_provider.dart';
import 'report_store.dart';

class ReportScreen extends StatelessWidget {
  const ReportScreen({Key? key}) : super(key: key);

  String get _generalInfo => '''
```

```
1. Використовуйте WPA2/WPA3 з мінімум 12 символами.
2. Поєднуйте великі/малі літери, цифри та спецсимволи.
3. Мінімальна довжина пароля – 12+ символів.
4. Обов’язково змінюйте SSID з дефолтного.
5. Вимикайте WPS та UPnP, якщо не використовуєте.
6. Використовуйте гостьову мережу для пристроїв IoT.
7. Оновлюйте прошивку маршрутизатора щонайменше раз на місяць.
8. Обмежуйте кількість невдалих спроб входу (лок-аут після 3 спроб).
9. Активуйте двофакторну автентифікацію на адмін-панелі роутера.
''';

  int get _threatCount => ReportStore.items.where((i) => i.cracked).length;
  String get _threatLevel {
    if (_threatCount == 0) return 'Невідома';
    if (_threatCount == 1) return 'Низька';
    if (_threatCount == 2) return 'Середня';
    return 'Висока';
  }

  Future<void> _saveToFile(BuildContext ctx) async {
    final items = ReportStore.items;
    try {
                        final dir = await getApplicationDocumentsDirectory();
      final file = File(p.join(dir.path, 'wifi_report.txt'));
      final buf  = StringBuffer()
                                        ..writeln('Загальні рекомендації:\n$_generalInfo\n')
        ..writeln('--- Детальний звіт ---\n');

      for (var it in items) {
        buf
          ..writeln('=== ${it.screenName} ===')
          ..writeln(
            'Вразливості:\n' +
            (it.vulnerabilities.isEmpty
              ? '– Не виявлено'
              : it.vulnerabilities.map((v) => '- $v').join('\n'))
          )
          ..writeln(
            '\nРекомендації:\n' +
            (it.recommendations.isEmpty
              ? '– Немає'
                    : it.recommendations.map((r) => '- $r').join('\n'))
          )
          ..writeln('\n');
      }

          buf.writeln('Рівень загрози: $_threatCount ($_threatLevel)');
      await file.writeAsString(buf.toString());
      ScaffoldMessenger.of(ctx).showSnackBar(
            SnackBar(content: Text('Звіт збережено: ${file.path}'))
      );
    } catch (e) {
      ScaffoldMessenger.of(ctx).showSnackBar(
        SnackBar(content: Text('Не вдалося зберегти файл: $e'))
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    final items = ReportStore.items;

    return Scaffold(
      appBar: AppBar(title: const Text('Звіт')),
      body: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            // Картка з загальними рекомендаціями
            Card(
              color: Colors.blueGrey[800],
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(8)),
              child: Padding(
                padding: const EdgeInsets.all(12),
                child: Text(
                  'Загальні рекомендації:\n$_generalInfo',
                  style: const TextStyle(fontSize: 16, color: Colors.white),
                ),
              ),
            ),
            const SizedBox(height: 16),

            // Список карток з атаками
            Expanded(
              child: ListView.builder(
                itemCount: items.length,
                itemBuilder: (_, idx) {
                  final it = items[idx];
                  return Card(
                      margin: const EdgeInsets.only(bottom: 16),
                    shape: RoundedRectangleBorder(
                                                borderRadius: BorderRadius.circular(8)),
                    child: Padding(
                      padding: const EdgeInsets.all(12),
                      child: Column(
                                                crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Text(
                            it.screenName,
                            style: const TextStyle(
                                        fontSize: 18, fontWeight: FontWeight.bold),
                          ),
```

```
                const SizedBox(height: 12),

                const Text('Вразливості:',
                    style: TextStyle(fontWeight:
FontWeight.w600)),
                const SizedBox(height: 4),
                Text(
                  it.vulnerabilities.isEmpty
                      ? '– Не виявлено'
                      : it.vulnerabilities
                          .map((v) => '• $v')
                          .join('\n'),
                ),
                const SizedBox(height: 12),

                const Text('Рекомендації:',
                    style: TextStyle(fontWeight:
FontWeight.w600)),
                const SizedBox(height: 4),
                Text(
                  it.recommendations.isEmpty
                      ? '– Немає'
                      : it.recommendations
                          .map((r) => '• $r')
                          .join('\n'),
                ),
              ],
            ),
          ),
        );
      },
    ),
  ),

  const Divider(),
  Text(
    'Кількість атак: $_threatCount ',
    style:
        const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
  ),
  const SizedBox(height: 8),

  ElevatedButton.icon(
    icon: const Icon(Icons.save_alt),
    label: const Text('Зберегти в .txt'),
    onPressed: () => _saveToFile(context),
  ),
 ],
),
),
);
}
}
```

pmkid_screen.dart

```
import 'dart:async';
import 'package:flutter/material.dart';
import
'package:flutter_application_1/ui/handshake_screen.dart';
import 'report_item.dart';
import 'report_store.dart';

class PmkidAttackScreen extends StatefulWidget {
  const PmkidAttackScreen({Key? key}) : super(key:
key);
  @override
  _PmkidAttackScreenState createState() =>
_PmkidAttackScreenState();
}

class _PmkidAttackScreenState extends
State<PmkidAttackScreen> {
  final ScrollController _logScroll = ScrollController();
  final StringBuffer _log = StringBuffer();
  final List<String> _vulnerabilities = [];
  final List<String> _recommendations = [];
  List<Network> _networks = [];
  Network? _selectedNetwork;

  bool _scanning = false;
  bool _cracking = false;
  String? _foundBssid;
  String? _recoveredPsk;

  void _writeLog(String text) {
    _log.write(text);
    setState(() {});
    WidgetsBinding.instance.addPostFrameCallback((_)
{
      if (_logScroll.hasClients) {
        _logScroll.jumpTo(_logScroll.position.maxScrollE
xtent);
      }
    });
  }

Future<void> _scanNetworks() async {
  if (_scanning) return;
  setState(() {
    _scanning = true;
    _foundBssid = null;
    _recoveredPsk = null;
    _log.clear();
    _vulnerabilities.clear();
    _recommendations.clear();
  });

  _writeLog('🧙‍♂️ Запуск hcxdumptool для
тестування...\n');
  await Future.delayed(const Duration(seconds: 1));
  _writeLog('   interface: wlan0mon\n');
  await Future.delayed(const Duration(seconds: 1));
  _writeLog('   [+] Beacon з PMKID від AP:
00:0C:41:82:B2:55 (Тестова мережа)\n');
  await Future.delayed(const Duration(seconds: 1));
  _writeLog('   [+] Beacon з PMKID від AP:
C0:25:2F:3D:9D:22 (MERCURY_5G)\n');
  await Future.delayed(const Duration(seconds: 1));
  _writeLog('   [*] Збережено у capture.pcapng\n');

  // ←– Ось тут формуємо реальний список і
вносимо в стан
  final nets = [
    Network('00:0C:41:82:B2:55', 1, 'Тестова мережа'),
```

```
      Network('C0:25:2F:3D:9D:22', 6, 'MERCURY_5G'),
    ];
    setState(() {
      _networks = nets;          // пропадає автоматичний вибір
      _scanning = false;
      _selectedNetwork = null;    // щоб dropdown скинувся
      _foundBssid = null;
    });

    // Далі конвертація
    await _convertCapture();
  }

    Future<void> _convertCapture() async {
      if (_foundBssid == null) return;
      _writeLog('\n✂️ Конвертація capture.pcapng → pmkid.hccapx...\n');
      await Future.delayed(const Duration(seconds: 2));
      _writeLog('   [*] Готовий файл pmkid.hccapx\n');
    }

    Future<void> _crackPsk() async {
  if (_cracking || _foundBssid == null) return;
  setState(() => _cracking = true);

  // === Блок перевірки не-тестової мережі ===
  const testBssid = '00:0C:41:82:B2:55';
  if (_foundBssid != testBssid) {
    // Повідомлення в термінал, що capture.pcapng вже збережено
    _writeLog('\n🚫 Захист MikroTik активний для мережі ${_selectedNetwork?.ssid ?? _foundBssid}\n');
    await Future.delayed(const Duration(seconds: 1));

    // Діалогове вікно користувачу
    showDialog(
      context: context,
      builder: (_) => AlertDialog(
        title: const Text('Атака заблокована'),
        content: Text('Мережу ${_selectedNetwork?.ssid ?? _foundBssid} захищено – PSK не знайдено.'),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('OK'),
          )
        ],
      ),
    );

    setState(() => _cracking = false);
    return;
  }
  // === Кінець блоку перевірки ===

  // – Далі йде ваша звична симуляція для тестової мережі –
  _writeLog('\n🔒 Запуск hashcat -m 16800 pmkid.hccapx rockyou.txt\n');
  await Future.delayed(const Duration(seconds: 1));
  _writeLog('   [*] Перевірка словника...\n');
  await Future.delayed(const Duration(seconds: 2));

  const psk = 'Induction';
  _writeLog('   [+] Знайдено PSK для $_foundBssid: $psk\n');

  setState(() {
    _recoveredPsk = psk;
    _cracking = false;

    _vulnerabilities.addAll([
      'PMKID вразливість: можливість отримання хешу без активного клієнта',
      'Використання слабкого пароля (знайдено за 3 секунди)',
      'Відсутність захисту від офлайн-перебору'
    ]);

    _recommendations.addAll([
      'Змінити пароль відповідно до вимог надійного пароля',
      'Використовуйте складні паролі (12+ символів, спецсимволи)',
      'Вимкніть WPA2-Personal на користь WPA2-Enterprise',
      'Обмежте кількість спроб авторизації',
      'Регулярно оновлюйте паролі мережі'
    ]);
  });

  // Додаємо до глобального звіту
  ReportStore.items.add(ReportItem(
    screenName: 'PMKID Attack',
    cracked: true,
    password: psk,
    vulnerabilities: List.from(_vulnerabilities),
    recommendations: List.from(_recommendations),
  ));
}

    @override
    void dispose() {
      _logScroll.dispose();
      super.dispose();
    }

    @override
    Widget build(BuildContext context) {
      return Scaffold(
        appBar: AppBar(
          title: const Text('TEST 3'),
          backgroundColor: const Color.fromARGB(255, 0, 0, 0),
        ),
        body: Padding(
          padding: const EdgeInsets.all(16),
          child: Row(
            children: [
              // Left side - controls and terminal
              Expanded(
```

```
            flex: 3,
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.stretch,
              children: [
                // Scan button
                ElevatedButton.icon(
                  icon: _scanning
                      ? const SizedBox(
                          width: 16,
                          height: 16,
                          child: CircularProgressIndicator(
                            strokeWidth: 2,
                            color: Colors.white,
                          ),
                        )
                      : const Icon(Icons.wifi_tethering),
                  label: Text(_scanning ? 'Сканування...' :
'СКАНУВАТИ'),
                  onPressed: _scanning ? null :
_scanNetworks,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const
Color.fromARGB(255, 14, 63, 17),
                    padding: const
EdgeInsets.symmetric(vertical: 12),
                  ),
                ),
                const SizedBox(height: 12),
                if (_networks.isNotEmpty) ...[
  const SizedBox(height: 12),
  DropdownButtonFormField<Network>(
    decoration: const InputDecoration(
      labelText: 'Оберіть мережу для атаки',
      border: OutlineInputBorder(),
    ),
    items: _networks.map((n) => DropdownMenuItem(
      value: n,
      child: Text('${n.ssid} (${n.bssid})'),
    )).toList(),
    value: _selectedNetwork,
    onChanged: _cracking || _scanning
        ? null
        : (n) {
            setState(() {
              _selectedNetwork = n;
              _foundBssid = n!.bssid;
            });
          },
  ),
  const SizedBox(height: 12),
],
                // Crack button
                ElevatedButton.icon(
                  icon: _cracking
                      ? const SizedBox(
                          width: 16,
                          height: 16,
                          child: CircularProgressIndicator(
                            strokeWidth: 2,
                            color: Colors.white,
                          ),
                        )
                      : const Icon(Icons.lock_open),
                  label: Text(_cracking ? 'Триває взлом...' :
'ВЗЛАМАТИ PSK'),
                  onPressed: (_foundBssid == null ||
_cracking) ? null : _crackPsk,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const
Color.fromARGB(255, 14, 63, 17),
                    padding: const
EdgeInsets.symmetric(vertical: 12),
                  ),
                ),
                const SizedBox(height: 16),

                // Terminal output
                Expanded(
                  child: Container(
                    decoration: BoxDecoration(
                      color: Colors.black,
                      borderRadius: BorderRadius.circular(8),
                      border: Border.all(color:
Colors.grey[700]!),
                    ),
                    padding: const EdgeInsets.all(12),
                    child: Scrollbar(
                      controller: _logScroll,
                      thumbVisibility: true,
                      child: SingleChildScrollView(
                        controller: _logScroll,
                        child: SelectableText(
                          _log.toString(),
                          style: const TextStyle(
                            fontFamily: 'monospace',
                            color: Colors.green,
                            fontSize: 14,
                          ),
                        ),
                      ),
                    ),
                  ),
                ),

                // Recovered PSK
                if (_recoveredPsk != null) ...[
                  const SizedBox(height: 16),
                  Text(
                    'Відновлений пароль:',
                    style:
Theme.of(context).textTheme.titleMedium!.copyWith(
                          fontWeight: FontWeight.bold,
                          color: Colors.white,
                        ),
                  ),
                  const SizedBox(height: 4),
                  Text(
                    _recoveredPsk!,
                    style:
Theme.of(context).textTheme.headlineSmall!.copyWith(
                          color: Colors.orange,
                          fontWeight: FontWeight.bold,
                        ),
```

```
                ),
              ],
            ],
          ),
        ),

        const SizedBox(width: 16),

        // Right side - vulnerabilities and
recommendations
        Expanded(
          flex: 2,
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
              // Vulnerabilities
              Card(
                color: Colors.grey[900],
                child: Padding(
                  padding: const EdgeInsets.all(16),
                  child: Column(
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: [
                      const Text(
                        'Виявлені вразливості:',
                        style: TextStyle(
                          fontSize: 18,
                          fontWeight: FontWeight.bold,
                          color: Colors.red,
                        ),
                      ),
                      const SizedBox(height: 8),
                      if (_vulnerabilities.isEmpty)
                        const Text(
                          'Проведіть сканування та атаку',
                          style: TextStyle(color: Colors.grey),
                        )
                      else
                        ..._vulnerabilities.map((v) =>
Padding(
                              padding: const
EdgeInsets.only(bottom: 8),
                              child: Text(
                                '• $v',
                                style: const TextStyle(color:
Colors.white),
                              ),
                            )),
                    ],
                  ),
                ),
                const SizedBox(height: 16),

                // Recommendations
                Card(
                  color: Colors.grey[900],
                  child: Padding(
                    padding: const EdgeInsets.all(16),
                    child: Column(
                      crossAxisAlignment:
CrossAxisAlignment.start,
                      children: [
                        const Text(
                          'Рекомендації:',
                          style: TextStyle(
                            fontSize: 18,
                            fontWeight: FontWeight.bold,
                            color: Colors.green,
                          ),
                        ),
                        const SizedBox(height: 8),
                        if (_recommendations.isEmpty)
                          const Text(
                            'Після атаки з\'являться
рекомендації',
                            style: TextStyle(color: Colors.grey),
                          )
                        else
                          ..._recommendations.map((r) =>
Padding(
                                padding: const
EdgeInsets.only(bottom: 8),
                                child: Text(
                                  '• $r',
                                  style: const TextStyle(color:
Colors.white),
                                ),
                              )),
                      ],
                    ),
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Засіб для тестування на проникнення Wi-Fi роутера

Автор роботи: Немировський Владислав Олександрович

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ, група 1 БС-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **3.25** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Завідувач кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних та комунікаційних систем» к. т. н., доц. _____ Леонід КУПЕРШТЕЙН

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Віталій ЛУКІЯНОВ

Здобувач _____ Владислав НЕМИРОВСЬКИЙ

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

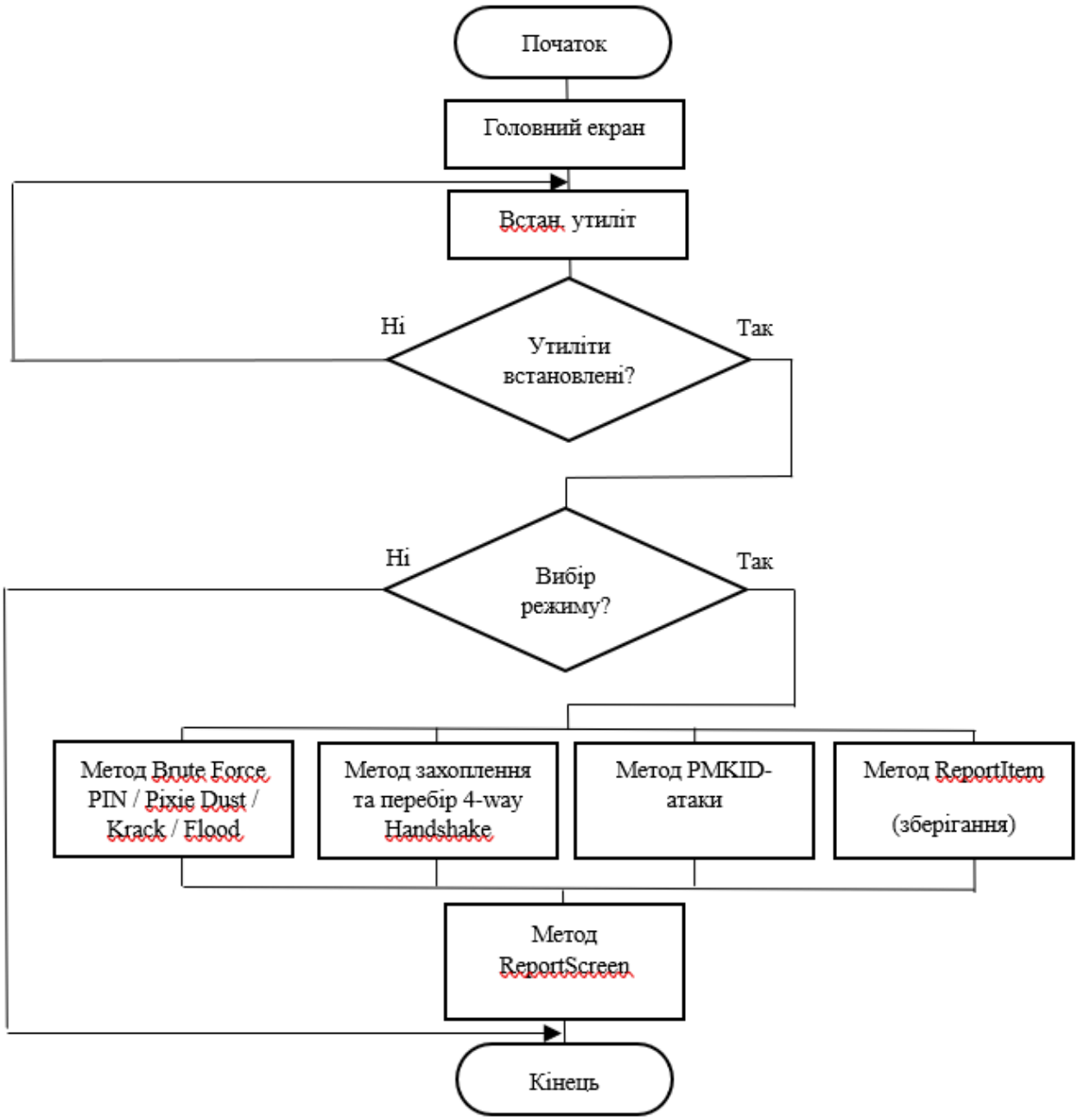
ЗАСІБ ДЛЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ WI-FI РОУТЕРА

Виконав: студент 4 курсу групи 1БС-21б
спеціальності 125 Кібербезпека
Владислав НЕМИРОВСЬКИЙ

Керівник: к. т. н., доцент, доцент каф. ЗІ
Віталій ЛУКІЧОВ

« 13 » 06. 2025 р.

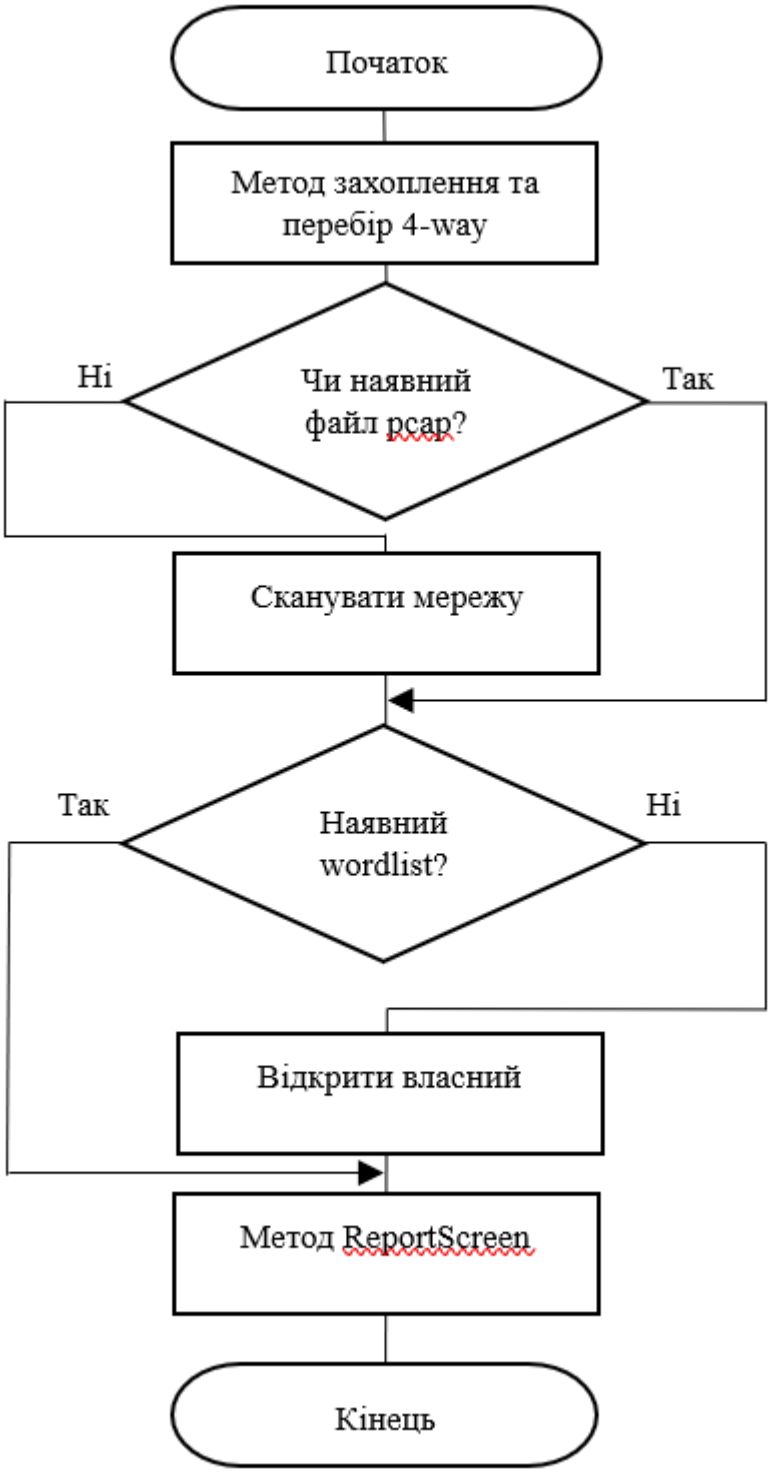
БЛОК-СХЕМА АЛГОРИТМУ ГОЛОВНОГО ЕКРАНУ ІЗ ПЕРЕЛІКОМ РЕЖИМІВ ТЕСТУВАННЯ ТА ВКЛАДКОЮ ЗВІТ



ЗАЛЕЖНІСТЬ КЛЮЧОВИХ МЕТОДІВ ТА КАНАЛИ ВЗАЄМОДІЇ ОСНОВНИХ КОМПОНЕНТІВ

Клас/Екран	**Dependencies**	**Методи**	**Взаємодіє**
HomeScreen	AttackMode, ReportStore	_loadModes(), _navigateTo(TestXScreen)	TestXScreen, ReportScreen
Test1Screen	ReportStore	_startWpsPin(), _startPixieDust()	ReportStore
Test2Screen	ReportStore, FilePicker	_startHandshakeBrute(), _stopHandshake()	ReportStore
Test3Screen	ReportStore	_startKrack(), _startFlood(), _stopFlood()	ReportStore
ReportScreen	ReportStore	_saveReportTxt(), _calculateThreatLevel()	–
ReportStore	–	addItem(ReportItem), clear()	ReportItem
ReportItem	–	–	–

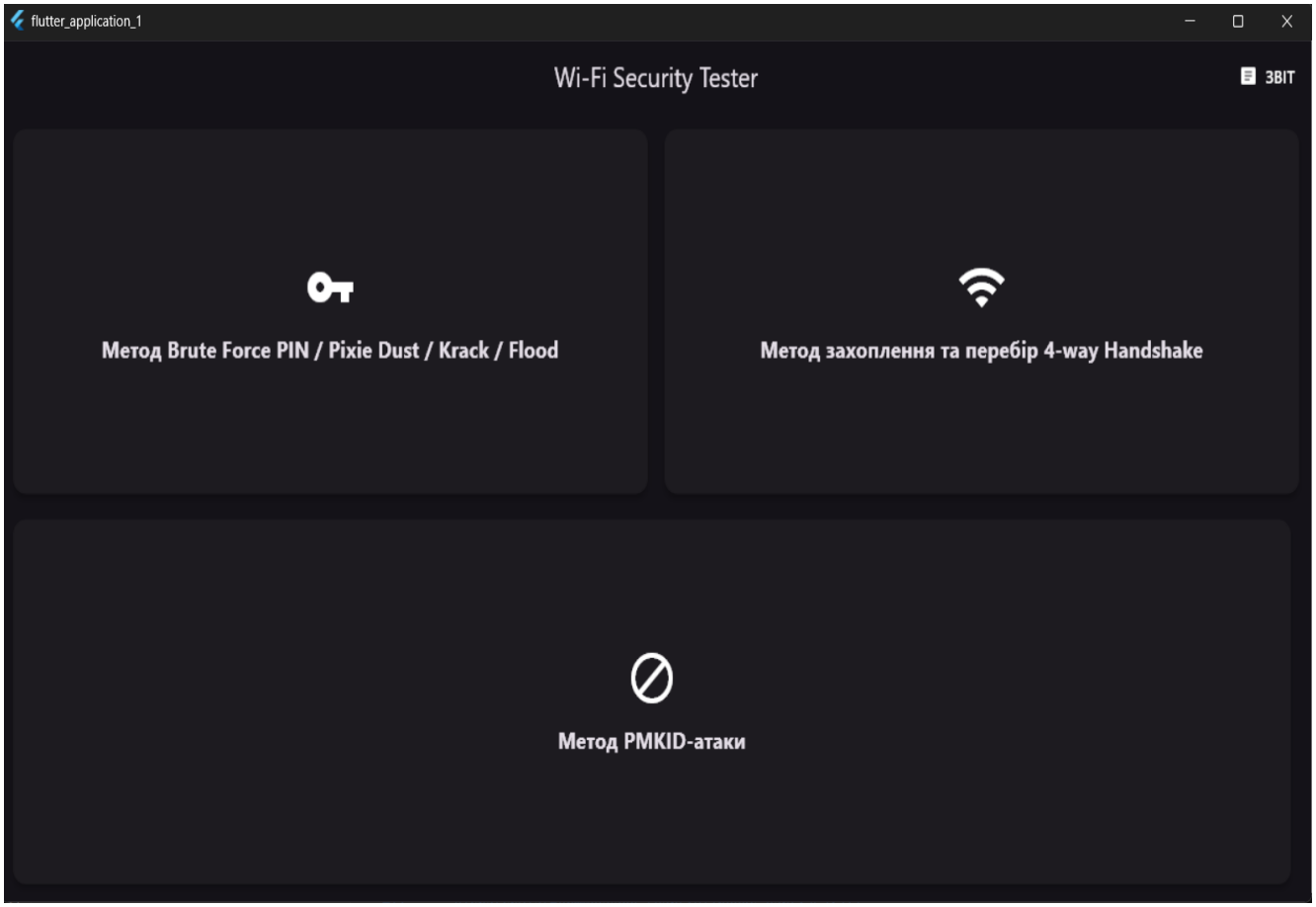
БЛОК-СХЕМА МЕТОДА ЗАХОПЛЕННЯ ТА ПЕРЕБІР 4-WAY HANDSHAKE



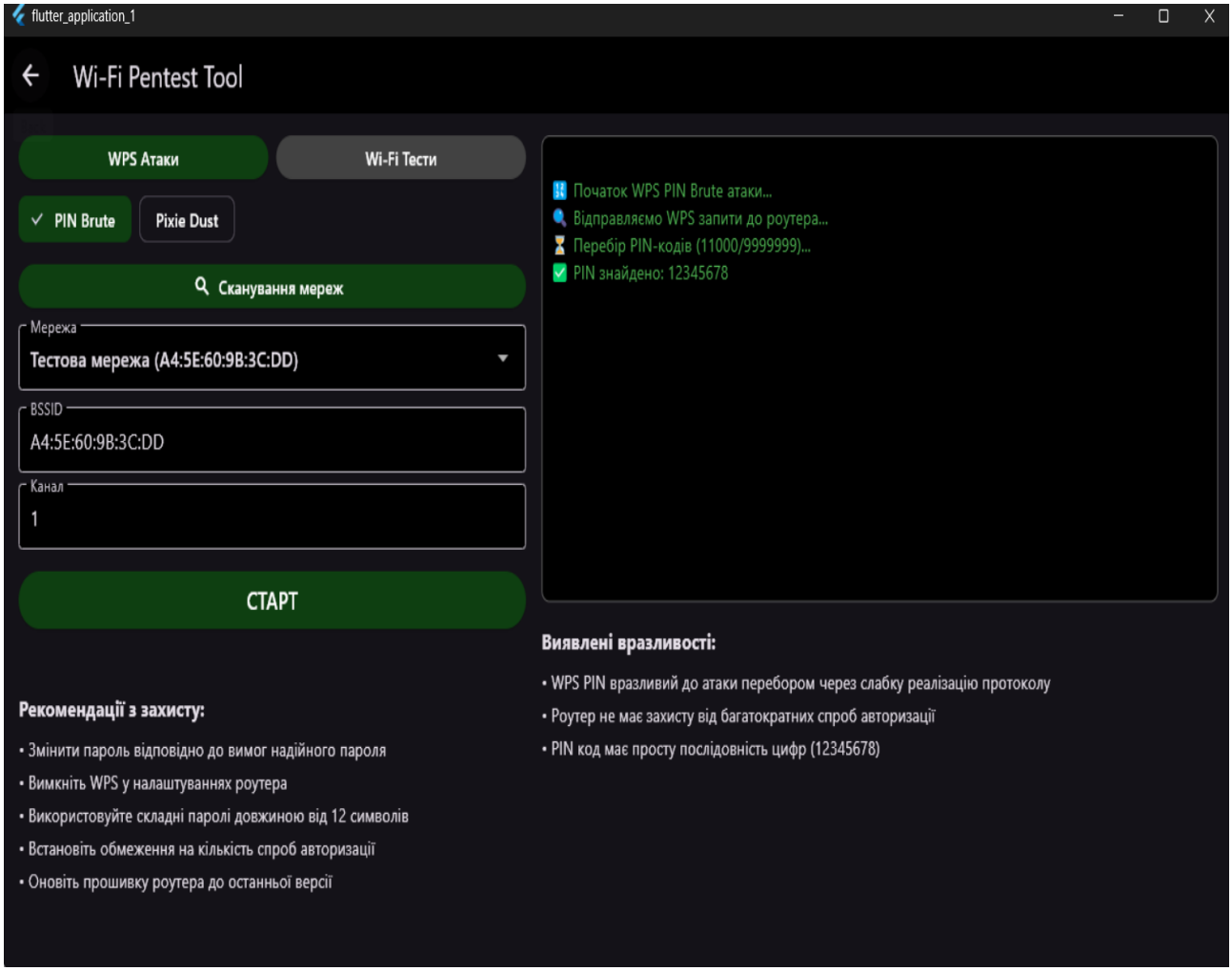
ПОРІВНЯЛЬНІ ХАРАКТЕРИСТИКИ МЕТОДІВ АТАК НА WI-FI РОУТЕРА

Метод атаки	Необхідне обладнання	Час виконання	Технічна складність	Вразливі цілі	Імовірність успіху	Виявлення користувачем
Reaver (WPS-атака)	Wi-Fi адаптер з моніторингом	4–6 годин	Середня	Точки з увімкненим WPS	Висока (при WPS)	Низька
Brute-force по .cap	Адаптер + handshake	10 хв – ∞	Висока (залежить від словника)	Будь-яка WPA/WPA2 мережа з handshake	Середня	Низька
Wifiphisher (соц. інж.)	2 адаптери + шаблон сторінки	5–15 хв	Висока	Користувачі відкритих мереж	Висока	Середня
KARMA	Rogue точка + моніторинг	Миттєва	Низька	Старі пристрої (IoT, принтери)	Низька	Висока
Known Beacons	Rogue точка + словник SSID	5–20 хв	Середня	Пристрої з AutoConnect	Середня	Середня

ГОЛОВНИЙ ЕКРАН HOMESCREEN



ПРИКЛАД ЕКРАН МЕТОДУ BRUTE FORCE PIN / PIXIE DUST / KRACK / FLOOD З РЕЗУЛЬТАТОМ PIN BRUTE



ГРАФІЧНИЙ ВИГЛЯД ЛОГІВ