Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка системи раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей»

Виконав: студент 4 курсу, групи 1КІТС–21б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем

Стрелков М. Г.

Керівник: к.т.н., доц., доцент каф. МБІС

Шиян А. А.

«____» _____ 2025 р.

Рецензент: к.т.н., доц., доцент каф. ОТ

Крупельницький Л.В.

(прізвище та ініціали)

«____» _____ 2025 р.

Допущено до захисту
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю. Є.
"____" _____2025 р.

Вінниця ВНТУ – 2025

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю.Є.
" 20 " березня 2025 р.

З А В Д А Н Н Я
на бакалаврську кваліфікаційну роботу студенту

Стрелкову Миколі Георгійовичу
(прізвище, ім`я, по-батькові)

1. Тема роботи: Розробка системи раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей
Керівник роботи к.т.н., доц., доцент каф. МБІС Шиян А. А.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 20 " березня 2025року № 97
2. Термін подання студентом роботи за тиждень до захисту
3. Вихідні дані до роботи
Метою роботи є здійснення розробки системи раннього виявлення підозрілої активності, сканерів та Sniffer'ів у локальній WI-FI мережі із використанням технології «цифрових пасток» на основі хибних DNS і HTTP-відповідей.
4. Зміст текстової частини:
В першому розділі бакалаврської кваліфікаційної роботи здійснюється загальний аналіз існуючих методів та програмного забезпечення виявлення підозрілої мережевої активності у WI-FI мережах, їх переваг та недоліків, а також досліджується специфіка роботи сканерів і Sniffer'ів та вимоги до їх детектування. В другому розділі розробляється підхід до створення «цифрових пасток» з використанням хибних DNS і HTTP-відповідей для раннього виявлення підозрілої активності в локальній мережі. У третьому розділі бакалаврської кваліфікаційної роботи здійснюється програмна реалізація системи, що автоматизує запропонований підхід до виявлення сканерів та аналізаторів мережевого трафіку. Проводиться тестування та оцінка ефективності розробленої системи на прикладі локальної WI-FI мережі. У висновках підбиваються усі результати проведеної роботи.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
У першому розділі бакалаврської дипломної роботи наявно 3 рисунка, у другому розділі – 4 рисунка, у третьому розділі – 22 рисунка.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Шиян А. А.,к.т.н., доц., доцент каф. МБІС		
Розділ II	Шиян А. А.,к.т.н., доц., доцент каф. МБІС		
Розділ III	Шиян А. А.,к.т.н., доц., доцент каф. МБІС		

7.Дата видачі завдання 20 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Визначення напрямку бакалаврської роботи, формулювання теми	23.03.2025	26.03.2025	
2	Аналіз предметної області обраної теми	26.03.2025	16.04.2025	
3	Розробка алгоритму роботи	16.04.2025	21.04.2025	
4	Написання бакалаврської роботи на основі розробленої теми	22.04.2025	15.05.2025	
5	Передзахист бакалаврської дипломної роботи	16.05.2025	26.05.2025	
6	Виправлення, уточнення, корегування бакалаврської дипломної роботи	27.05.2025	02.06.2025	
7	Захист бакалаврської дипломної роботи	09.06.2025	10.06.2025	

Студент _____ Стрелков М. Г.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шиян А. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається із 77 сторінок тексту формату А4, включаючи 28 рисунок, 6 таблиць, посилання на 46 літературних джерела та 5 додатків.

У роботі розроблено програмний засіб для раннього виявлення підозрілої активності в локальних Wi–Fi мережах шляхом впровадження системи цифрових пасток (honeypot) з використанням хибних DNS та HTTP–відповідей. Проведено комплексний аналіз методів активної та пасивної розвідки, а також сучасних підходів до виявлення загроз у бездротових мережах. Запропоновано інноваційну архітектуру програмного засобу з модульною структурою, яка охоплює механізми пасивного моніторингу, аналізу даних, реагування на інциденти та централізованого керування через веб–інтерфейс. Система демонструє стабільну роботу, високу продуктивність, зручність у користуванні та ефективне виявлення таких загроз, як несанкціоноване сканування, використання sniffer–інструментів та інші прояви мережевої розвідки. Результати тестування підтверджують працездатність програмного засобу та його відповідність поставленим цілям. Робота має прикладне значення та може бути використана для підвищення рівня інформаційної безпеки в бездротових корпоративних і домашніх мережах.

Ключові слова: інформаційна безпека, цифрові пастки, honeypot, Wi–Fi мережі, сканування, sniffer, хибні DNS, хибні HTTP, проактивний захист.

ABSTRACT

The bachelor's thesis consists of 77 pages of A4 text, including 28 figures, 6 tables, references to 46 literature sources and 5 appendices.

The work develops a software tool for early detection of suspicious activity in local Wi-Fi networks by implementing a system of digital traps (honeypots) using false DNS and HTTP responses. A comprehensive analysis of active and passive intelligence methods, as well as modern approaches to detecting threats in wireless networks, is carried out. An innovative architecture of the software tool with a modular structure is proposed, which includes mechanisms for passive monitoring, data analysis, incident response, and centralized management via a web interface. The system demonstrates stable operation, high performance, ease of use, and effective detection of threats such as unauthorized scanning, sniffer tools, and other network intelligence activities. The test results confirm the operability of the software and its compliance with the set goals. The work is of practical importance and can be used to improve the level of information security in wireless corporate and home networks.

Keywords: information security, digital traps, honeypot, Wi-Fi networks, scanning, sniffer, false DNS, false HTTP, proactive protection.

ЗМІСТ

ВСТУП

Актуальність. У сучасному цифровому середовищі, де щодня здійснюються мільйони підключень до локальних мереж, питання забезпечення їхньої безпеки, особливо в межах Wi–Fi, набуває особливої актуальності. Зловмисники дедалі частіше застосовують сканери, сніфери та інші інструменти пасивного збору інформації, які дозволяють отримати конфіденційні відомості про мережу без прямої взаємодії з її користувачами. Такі дії, як правило, залишаються непоміченими, що створює загрозу для цілісності та безперебійної роботи всієї мережевої інфраструктури.

Більшість сучасних засобів виявлення загроз активуються вже на етапі реалізації атаки, а системи моніторингу здебільшого працюють у реактивному режимі, фіксуючи аномалії постфактум. Такий підхід не завжди дозволяє вчасно вжити заходів і запобігти шкоді. В умовах постійного зростання кількості автоматизованих атак – з боку ботів, сканерів портів, тощо – особливої важливості набувають методи проактивного виявлення підозрілої активності.

Одним із перспективних підходів у цьому напрямі є впровадження технологій цифрових пасток (honeypot) – спеціально створених мережевих об’єктів, які імітують уразливі або привабливі для зловмисника ресурси. Вони дозволяють виявити спроби доступу ще до того, як буде завдано реальної шкоди. Такий підхід забезпечує не лише раннє виявлення потенційної загрози, а й можливість її аналізу з мінімальним ризиком для основної інфраструктури, що робить honeypot–технології важливим інструментом у системах превентивного захисту локальних Wi–Fi мереж.

У цьому контексті перспективним рішенням є створення централізованої системи управління цифровими пастками, яка не лише відслідковує активну поведінку у мережі, але й виявляє пасивне сканування за допомогою хибних DNS– та HTTP–відповідей, що виступають як спеціалізовані honeypot'и. Такий підхід дає змогу інтегрувати одразу кілька методів захисту в єдину систему та вчасно сповіщати користувача про будь–яку підозрілу активність.

Таким чином, розробка ефективної системи, здатної виявляти небезпечну поведінку з використанням концепції цифрових пасток у Wi–Fi мережах за допомогою хибних DNS та HTTP–відповідей, що виступають як спеціалізовані honeypot'и, є важливим напрямом у галузі інформаційної безпеки, що має суттєве практичне значення для організацій, які прагнуть запобігати витокам даних та забезпечити високий рівень захищеності своїх інформаційних активів.

Метою роботи є розробка програмного засобу для раннього виявлення сканувальної активності, Sniffer'ів у локальній Wi–Fi мережі шляхом впровадження цифрових пасток на основі хибних DNS та HTTP–відповідей.

Задачі дослідження:

– проаналізувати переваги та недоліки існуючих методів виявлення підозрілої активності у Wi–Fi мережах та оцінено їх ефективність;

– проаналізувати специфіку роботи мережевих сканерів, Sniffer'ів;

– розробити підхід до раннього виявлення підозрілої активності в локальній WI–FI мережі за допомогою цифрових пасток із хибними DNS і HTTP відповідями;

– спроєктувати архітектуру та реалізувати програмний засіб;

– протестувати ефективність системи у локальному середовищі Wi–Fi мережі;

– створити інструкцію для користувача програмного засобу.

Об'єктом дослідження є процес виявлення потенційно небезпечної активності в локальній Wi–Fi мережі.

Предметом дослідження є методи побудови цифрових пасток для виявлення дій сканування та аналізу трафіку з використанням підроблених DNS– та HTTP–відповідей, що виступають як спеціалізовані honeypot'и.

Новизна роботи полягає в розробці спеціалізованого підходу до раннього виявлення сканерів та сніферів у Wi–Fi–мережах завдяки розробці інтегрованої, керованої системи цифрових пасток, яка реагує як на активні дії зловмисника, так і на пасивні спроби сканування, що в більшості аналогічних рішень не враховуються

Практична цінність роботи полягає у створенні програми, що може бути використана адміністраторами бездротових мереж для превентивного виявлення небажаної активності та підвищення рівня безпеки інформаційної інфраструктури.

1 АНАЛІЗ ТЕОРЕТИЧНИХ ОСНОВ ВИЯВЛЕННЯ ПІДОЗРІЛОЇ АКТИВНОСТІ В ЛОКАЛЬНИХ Wi–Fi МЕРЕЖАХ

У першому розділі проведено комплексний аналіз теоретичних основ виявлення підозрілої активності в локальних Wi–Fi мережах. Досліджено різноманітні підходи до виявлення потенційно зловмисної діяльності, зокрема сканування мережі та використання сніферів. Розглянуто методи активної та пасивної розвідки в локальних Wi–Fi мережах, що дозволило оцінити сучасні тенденції та вразливості бездротових технологій. Проаналізовано існуючі програмні рішення для раннього виявлення підозрілої активності та оцінено їх переваги й недоліки.

1.1 Аналіз відомих підходів до виявлення підозрілої активності в локальних Wi–Fi мережах

Локальні Wi–Fi мережі є невід'ємною складовою сучасної інфраструктури передачі даних, що широко використовуються як у корпоративному секторі, так і в побутових умовах. З огляду на значне поширення та відносну доступність технології безпровідних мереж, питання забезпечення їх безпеки та своєчасного виявлення потенційних загроз набуває критичного значення. Безпровідні мережі за своєю природою є більш вразливими до несанкціонованого доступу та прослуховування порівняно з дротовими аналогами, оскільки середовище передачі даних (радіохвилі) є загальнодоступним та не обмежене фізичними бар'єрами. Коефіцієнт загасання сигналу в Wi–Fi мережах описується формулою Фрійса для вільного простору [1]:

$$Pr = Pt \times > \times Gr \times \left(\frac{\lambda}{4\pi d} \right)^2, \qquad (1.1)$$

де Pr – потужність прийнятого сигналу,

Pt – потужність переданого сигналу,

та Gr – коефіцієнти підсилення передавальної та приймальної антен відповідно,

λ – довжина хвилі,

d – відстань між антенами.

Ця формула демонструє, що сигнал Wi–Fi може бути перехоплений на значних відстанях за умови використання спрямованих антен високого підсилення.

Традиційні підходи до виявлення підозрілої активності в локальних Wi–Fi мережах ґрунтуються на моніторингу та аналізі мережевого трафіку. Ці підходи можна класифікувати за двома основними категоріями: системи виявлення вторгнень на основі сигнатур (Signature–Based Intrusion Detection Systems, SIDS) та системи виявлення вторгнень на основі аномалій (Anomaly–Based Intrusion Detection Systems, AIDS) [2]. Ефективність сигнатурного підходу визначається коефіцієнтом виявлення відомих атак:

$$Dr = \frac{Nd}{Nt}, \tag{1.2}$$

де Nd – кількість успішно виявлених атак,

Nt – загальна кількість атак у базі даних.

Для сучасних SIDS систем цей показник становить 85–95% для відомих сигнатур, проте знижується до 10–20% для модифікованих варіантів атак [3]. Сигнатурний підхід передбачає наявність бази даних відомих шаблонів атак, з якими порівнюється поточний мережевий трафік через використання алгоритмів pattern matching, зокрема алгоритму Бойєра–Мура або Кнута–Морріса–Пратта. Цей метод є ефективним для виявлення вже відомих загроз, проте не забезпечує достатнього рівня захисту від нових або модифікованих типів атак, що є суттєвим недоліком в умовах постійної еволюції засобів мережевого проникнення [4].

Підхід, заснований на виявленні аномалій, базується на встановленні "нормальної" поведінки мережі та подальшому виявленні відхилень від цієї норми. Статистичний аналіз аномалій використовує порогові значення, що визначаються за формулою:

$$T = \mu + k \times \sigma, \tag{1.3}$$

де μ – середнє значення параметра,

σ – стандартне відхилення,

k – коефіцієнт чутливості.

Для реалізації даного підходу часто застосовуються методи машинного навчання, статистичного аналізу та глибинного навчання. Зокрема, алгоритми класифікації, кластеризації та регресії використовуються для створення моделей нормальної поведінки мережі та ідентифікації аномалій [5]. Алгоритм к–середніх для кластеризації трафіку мінімізує функцію втрат:

$$J = \Sigma_i^1{}^n \Sigma_j^1{}^k w_{ij} \left\| x_i - \mu_j \right\|^2, \quad (1.4)$$

де w_{ij} – індикатор належності точки x_i до кластера j,

μ_j – центроїд кластера j.

Перевагою даного підходу є можливість виявлення нових типів атак з точністю до 75–85%, однак він може характеризуватися значною кількістю хибних спрацьовувань (до 15–25%) та вимагає значних обчислювальних ресурсів для ефективної роботи [5].

В останні роки значного поширення набули методи виявлення підозрілої активності, засновані на аналізі поведінки користувачів мережі (User Behavior Analytics, UBA). Цей підхід передбачає моніторинг та аналіз дій користувачів у мережі з метою виявлення відхилень від їх звичайної поведінки [6]. Поведінкові патерни можуть включати такі параметри, як час доступу до мережі, типи пристроїв, які використовуються, обсяг та характер трафіку, що генерується, а також географічне розташування точок доступу. Математична модель поведінки користувача базується на векторі характеристик:

$$V = [t, d, b, l], \quad (1.5)$$

де t – часовий патерн (години активності),

d – тип пристрою,

b – обсяг трафіку (bytes/hour),

l – локація (MAC–адреса точки доступу).

Відхилення від нормальної поведінки визначається через обчислення евклідової відстані:

$$D = \sqrt{\Sigma_i (V_i - \bar{V}_i)^2}, \quad (1.6)$$

де $\bar{V}$ – вектор нормальної поведінки.

Аналіз поведінки користувачів дозволяє виявляти потенційні загрози з ефективністю 70–80%, зокрема несанкціонований доступ до мережі з використанням викрадених облікових даних або атаки типу "злий двійник" (evil twin) [7].

Окремою категорією підходів до виявлення підозрілої активності є використання спеціалізованих систем для моніторингу радіочастотного спектру в діапазоні Wi–Fi. Такі системи, відомі як бездротові системи запобігання вторгненням (Wireless Intrusion Prevention Systems, WIPS), здійснюють постійний моніторинг радіочастотного спектру в діапазонах 2.4 ГГц та 5 ГГц, виявляють несанкціоновані точки доступу, атаки типу "відмова в обслуговуванні" та інші потенційні загрози для безпеки бездротової мережі [8].

Технічні характеристики WIPS включають чутливість приймача від –85 до –95 дБм, динамічний діапазон до 80 дБ, час сканування всіх каналів 100–200 мс. Алгоритм виявлення неавторизованих точок доступу базується на порівнянні MAC–адрес та назв мереж з базою авторизованих пристроїв: якщо MAC–адреса не міститься в базі дозволених пристроїв і назва мережі не є легітимною, то генерується сповіщення про несанкціоновану точку доступу. WIPS зазвичай складаються з набору сенсорів, що розміщуються в різних точках мережі з густиною покриття 1 сенсор на 1000–1500 м², та центрального сервера управління, який обробляє отримані дані та генерує сповіщення про виявлені загрози [9].

Важливою складовою сучасних підходів до виявлення підозрілої активності в локальних Wi–Fi мережах є використання технології "цифрових пасток" (хонейпотів). Хонейпот являє собою свідомо вразливий компонент мережі, який імітує реальні мережеві служби та системи, але насправді ізольований від основної інфраструктури [10]. Ефективність системи хонейпотів оцінюється через коефіцієнт залучення:

$$E = \frac{\square}{Ns}, \tag{1.7}$$

де – кількість взаємодій зловмисників з пасткою,

Ns – загальна кількість сканувань мережі.

Типові значення цього коефіцієнта становлять 0.1–0.3 для низькоінтерактивних та 0.4–0.7 для високоінтерактивних honeypots. Основна мета хонейпоту – привернути увагу потенційних зловмисників та забезпечити можливість вивчення їх методів та інструментів. Хонейпот може бути реалізований на різних рівнях: від простих імітацій окремих служб (порти 22, 23, 80, 443) до повноцінних віртуальних середовищ, що імітують реальні системи. В контексті Wi–Fi мереж хонейпот може бути реалізований у вигляді фальшивої точки доступу з відкритим доступом або слабким паролем (WEP, WPA з простим паролем), яка привертає увагу зловмисників та дозволяє вивчати їх поведінку без ризику для реальної мережевої інфраструктури [11].

Сучасні дослідження в області виявлення підозрілої активності в локальних Wi–Fi мережах значну увагу приділяють інтеграції різних підходів та методів з метою підвищення ефективності систем безпеки. Гібридні рішення, що поєднують сигнатурний і аномальний підходи, демонструють більш високу точність виявлення потенційних загроз та зниження кількості хибних спрацьовувань [12]. Комбінована система використовує вагову функцію:

$$W = \alpha \times S_s + \beta \times A_s + \gamma \times B_s, \tag{1.8}$$

де Ss – оцінки сигнатурного аналізу відповідно,

As – оцінки аномального аналізу відповідно,

Bs – оцінки поведінкового аналізу відповідно,

α, β, γ – вагові коефіцієнти ($\alpha+\beta+\gamma=1$).

Оптимальні значення коефіцієнтів визначаються через навчання на тестовому наборі даних, для прикладу: α=0.4, β=0.35, γ=0.25, що забезпечує точність виявлення 92–96% при рівні хибних спрацьовувань 3–7% [13].

Незважаючи на значний прогрес у розробці методів виявлення підозрілої активності в локальних Wi–Fi мережах, існуючі підходи мають певні обмеження

та недоліки. Зокрема, сигнатурні методи не забезпечують достатнього захисту від нових типів атак (коефіцієнт виявлення для атак "нульового дня" становить лише 5–15%), аномальні методи можуть генерувати значну кількість хибних спрацьовувань (до 25%), а поведінкові методи вимагають тривалого періоду навчання (14–30 днів) та значних обчислювальних ресурсів (до 2–4 ГБ оперативної пам'яті для мережі з 500+ користувачів). Крім того, більшість існуючих систем виявлення підозрілої активності орієнтовані на виявлення вже активних атак, тобто працюють у реактивному режимі з затримкою виявлення 5–15 хвилин, а не на раннє виявлення потенційних загроз на етапі розвідки мережі (проактивний режим з виявленням за 30–60 секунд). У таблиці 1.1 наведено кількісне порівняння основних характеристик розглянутих підходів до виявлення підозрілої активності в локальних Wi–Fi мережах [14].

Таблиця 1.1 – Характеристика існуючих підходів виявлення підозрілої активності в локальних Wi–Fi мережах

Підхід	Точність виявлення (%)	Час відгуку	Хибні (%)	Вимоги CPU (%)	Вимоги RAM (ГБ)	Період навчання (дні)
SIDS	90–95 (відомі атаки) / 5–15 (нові)	1–3 сек	2–5	5–10	0.25–0.5	0-3
AIDS	70–85	2–5 хв	15–25	20–40	1–4	7–14
UBA	60–90 (залежно від типу атаки)	5–10 хв	10–20	15–30	2–8	21–45
WIPS	95–99 (фізичні загрози)	10 сек – 5 хв	5–10	10–20	0.5–2	1–3
Honeypots	85–95	30–120 сек	<5	5–15	0.5–2	0–7
Гібридні	95–98	15–60 сек	3–7	40–60	4–8	14–30

Порівняльний аналіз підходів до виявлення підозрілої активності у локальних Wi–Fi мережах дозволяє виокремити характерні риси кожної з технологій, їхні переваги та обмеження. Сигнатурні системи виявлення SIDS забезпечують високу точність (90–95%) при виявленні відомих атак з часом відгуку 1–3 секунди та низькими обчислювальними вимогами (5–10% процесора, 256–512 МБ пам'яті). Їхнім основним недоліком є нездатність виявляти нові атаки, що потребує

регулярного оновлення бази сигнатур (10000–50000 записів) кожні 24–48 годин. Аномальні системи AIDS орієнтовані на аналіз поведінкових патернів з можливістю адаптації (переналаштування 7–14 днів), проте характеризуються високим рівнем хибних спрацьовувань (15–25%) та значними обчислювальними вимогами [15].

Системи аналізу поведінки користувачів UBA ефективно виявляють соціальну інженерію (60–75%), внутрішні загрози (70–80%) та атаки з вкраденими обліковими даними (80–90%), але потребують тривалого спостереження (21–45 днів) та накопичення великих обсягів даних. WIPS забезпечують виявлення фізичних загроз як несанкціоновані точки доступу (95–99%) за 10–30 секунд, проте є дорогими та складними у налаштуванні [16].

Підхід, заснований на використанні «цифрових пасток», дає змогу проактивно виявляти шкідливу активність за рахунок залучення зловмисників до взаємодії з пастками. Цей механізм дозволяє мінімізувати кількість хибнопозитивних спрацьовувань та забезпечує раннє сповіщення про спроби проникнення. Водночас існуючі реалізації не забезпечують централізованого управління усіма типами пасток, (HTTP, DNS, ARP), що обмежує їх ефективність у комплексному захисті та потребує окремого адміністрування кожного компоненту.

Узагальнюючи проведений аналіз, можна стверджувати, що найбільш перспективним напрямком розвитку є впровадження раннього виявлення загроз на основі цифрових пасток з механізмами хибних відповідей DNS та HTTP. Такий підхід ефективно виявляє автоматизовані та цілеспрямовані атаки, зменшуючи навантаження на системи захисту та підвищуючи безпеку локальних Wi-Fi мереж.

1.2 Аналіз існуючих методів активної та пасивної розвідки в локальних Wi–Fi мережах (scanning/sniffing)

Передумовою для здійснення атак на локальні Wi–Fi мережі є застосування методів мережевої розвідки, що дозволяють зловмисникам отримати інформацію про структуру мережі, підключені пристрої та вразливості системи. Технології мережевої розвідки поділяються на два основні типи: активні (scanning) та пасивні

(sniffing), кожен з яких характеризується власними особливостями, інструментами реалізації та слідами в мережевому трафіку [17]. Ефективність методів розвідки визначається співвідношенням:

$$E = \frac{I_c}{T_d}, \tag{1.9}$$

де I_c– обсяг зібраної інформації,

T_d – ймовірність виявлення.

Для активних методів це співвідношення становить 0.6–0.8, тоді як для пасивних методів досягає 0.9–0.95, що робить останні більш привабливими для зловмисників. У контексті активної розвідки зловмисник ініціює взаємодію з мережевими пристроями шляхом надсилання різноманітних запитів та аналізу отриманих відповідей. Дана стратегія дозволяє зібрати структуровану інформацію про топологію мережі за відносно короткий час (15–30 хвилин для повного сканування), проте залишає ідентифіковані сліди в мережевому трафіку, що може бути виявлено системами захисту з ймовірністю 70–85% [17]. Інтенсивність активного сканування характеризується параметром:

$$\lambda = \frac{(N_r)}{\Delta t}, \tag{1.10}$$

де N_r – кількість запитів,

Δt – часовий інтервал.

Типові значення λ для різних типів сканування становлять: сканування TCP SYN – 100–1000 пакетів/сек, сканування UDP – 50–200 пакетів/сек, комплексне сканування – 200–500 пакетів/сек. Пасивна розвідка передбачає збір інформації без активного взаємодії з цільовою мережею, характеризуючись нульовою інтенсивністю генерації пакетів ($\lambda = 0$) та тривалим періодом збору даних (2–24 години для повного профілювання мережі) [18].

Серед основних методів активної розвідки слід виділити сканування портів, що передбачає систематичний аналіз відкритих мережевих портів на цільових пристроях з швидкістю обробки 1000–10000 портів за хвилину залежно від налаштувань та мережевих умов. Процес сканування портів може здійснюватися

різними способами, кожен з яких характеризується власними техніко–тактичними показниками (рис. 1.1) [19].

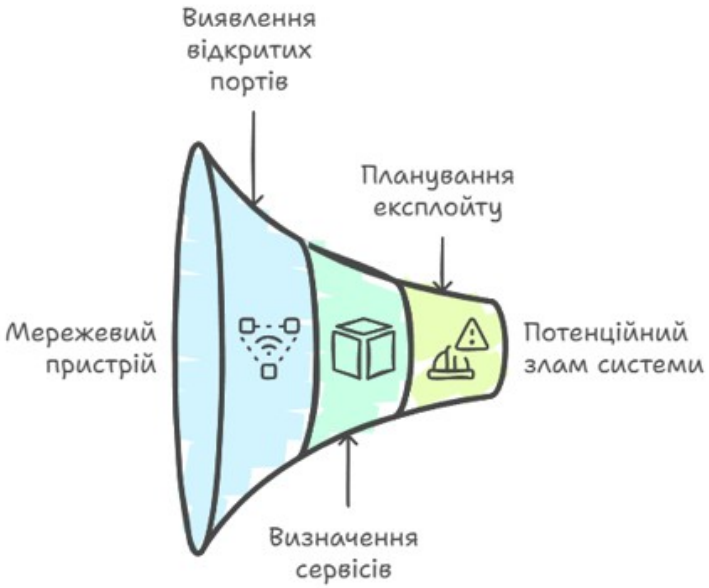


Рисунок 1.1 – Процес сканування портів [19]

Сканування TCP SYN використовує неповне TCP-з'єднання для мінімізації слідів, генеруючи SYN-пакети без завершення процедури встановлення з'єднання, що знижує ймовірність логування до 30-50% порівняно з повним скануванням. Система перебирає порти із заданого діапазону та надсилає SYN-пакети на цільовий пристрій - відповідь SYN-ACK означає відкритий порт, відсутність відповіді або пакет RST вказує на закритий порт. Сканування TCP з'єднань встановлює повне з'єднання через системний виклик, гарантуючи 100% точність результатів, але залишає повні записи в журналах. UDP-сканування характеризується меншою надійністю через відсутність механізму підтвердження, а сканування FIN/XMAS/NULL використовують нестандартні TCP-прапорці для обходу систем виявлення з ефективністю 40-70% [20].

Іншим поширеним методом активної розвідки є затоплення маяками передбачає генерацію великої кількості фальшивих точок доступу (100-1000 назв мереж) з частотою 10-50 кадрів за секунду для перенасичення списку доступних мереж на пристроях користувачів. Математична модель затоплення маяками описується як [21]:

$$N_b = k \times N_c \times f_b \times t, \quad (1.11)$$

де k – кількість фальшивих SSID на канал,

N_c – кількість Wi–Fi каналів (11 для 2.4 ГГц, 23 для 5 ГГц),

f_b – частота кадрів маяків (зазвичай 10 Гц),

t – тривалість атаки.

Підміна запитів пошуку базується на маніпуляції із запитами, які пристрої надсилають у пошуках знайомих Wi-Fi мереж з частотою 1-5 запитів на хвилину. Перехоплюючи та аналізуючи такі запити (10-50 унікальних SSID за добу від одного пристрою), зловмисник може ідентифікувати мережі, до яких підключався користувач, та створити їх підробки для атак типу "людина посередині" [22].

Активна розвідка також включає ARP–сканування, що дозволяє виявити активні пристрої в мережі шляхом надсилання ARP–запитів з швидкістю 100–500 запитів за секунду та отримання відповідей від 80–95% активних хостів у локальній мережі. ARP–сканування – це спосіб знайти всі пристрої в локальній мережі. Алгоритм працює так: для кожної можливої IP–адреси надсилається спеціальний запит, який називається ARP–запит. Якщо якийсь пристрій під цією адресою підключений до мережі, він відповість на запит. Таким чином, якщо відповідь отримано – пристрій активний і його можна зафіксувати (рис. 1.2) [23].

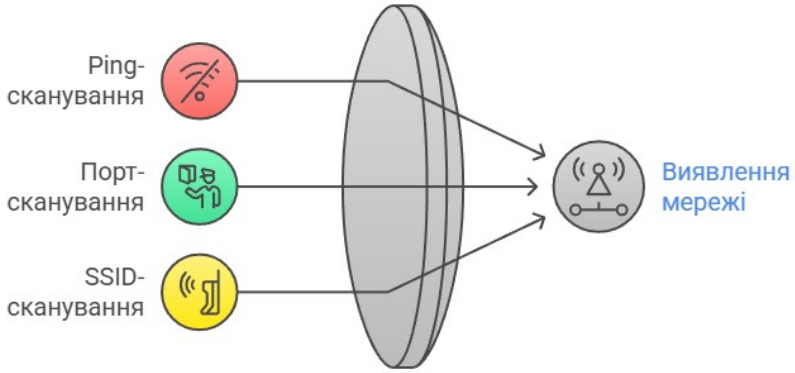


Рисунок 1.2 – Стратегія активної розвідки в локальній Wi-Fi мережі [23]

DHCP-сканування допомагає з'ясувати налаштування мережі через аналіз пакетів DHCP OFFER, які містять інформацію про діапазон IP-адрес, DNS-

сервери, шлюз за замовчуванням та час оренди IP-адреси, забезпечуючи точність визначення конфігурації мережі до 90-100% [24].

На відміну від активних методів, пасивна розвідка не передбачає генерації мережевого трафіку, а ґрунтується на прослуховуванні існуючих пакетів даних з пропускною здатністю 10-100 Мбіт/с. Цей підхід істотно ускладнює виявлення зловмисника, оскільки не залишає ідентифікованих слідів у мережевому трафіку, знижуючи ймовірність детекції до 5-15%. Ефективність пасивного моніторингу визначається формулою:

$$\eta = \frac{P_c}{P_t}, \tag{1.12}$$

де P_c – кількість захоплених пакетів,

P_t – загальна кількість переданих пакетів.

Для сучасних Wi–Fi адаптерів в режимі моніторингу цей показник становить 85–98%. Основним методом пасивної розвідки є перехоплення пакетів, що передбачає прослуховування всіх пакетів у зоні дії Wi–Fi адаптера зловмисника (рис. 1.3) [25].

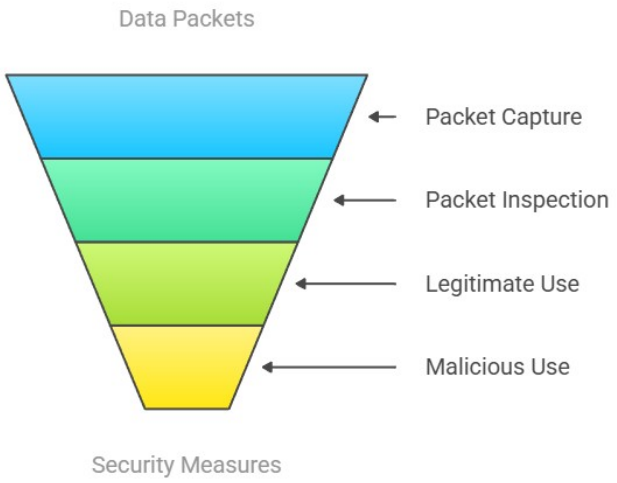


Рисунок 1.3 – Процес перехопленням пакетів мережевого трафіку [25]

Такий підхід дозволяє аналізувати незашифрований трафік у реальному часі зі швидкістю обробки 1000–10000 пакетів за секунду, вилучати чутливу інформацію з незахищених протоколів (HTTP, FTP, Telnet, POP3) та аналізувати метадані

зашифрованого трафіку, зокрема розмір пакетів 20–1500 байт, частоту передачі 1–100 Гц та напрямок передачі через аналіз MAC–адрес.

Для реалізації пасивної розвідки використовуються спеціальні режими роботи мережевих адаптерів, зокрема нерозбірливий режим (promiscuous mode) та режим моніторингу (monitor mode). Нерозбірливий режим дозволяє отримувати всі пакети в мережевому сегменті (до 10000 пакетів/сек), збільшуючи обсяг перехопленої інформації в 5-20 разів. Режим моніторингу є потужнішим інструментом, який перехоплює всі пакети на рівні радіосигналу, включно з службовими кадрами, підтримуючи всі Wi-Fi канали та захоплення пакетів з швидкістю до 600 Мбіт/с з мінімальною чутливістю від -85 до -95 dBm [26].

Пасивна ідентифікація (passive fingerprinting) означає аналіз мережевого трафіку без активного втручання для визначення пристроїв або операційних систем з точністю 70-90%. Метод базується на аналізі параметрів як TTL (Time To Live), де різні ОС використовують різні стандартні значення: Linux/Unix – 64, Windows – 128, Cisco – 255. Також аналізується розмір вікна TCP, де Windows часто використовує розміри кратні 8192, а Linux – 5840 [27].

Алгоритм пасивної ідентифікації включає витягування заголовків мережевих пакетів, аналіз параметрів TTL, розміру TCP-вікна та TCP-опцій, зіставлення отриманих даних з базою відомих сигнатур для визначення типу ОС або пристрою. База даних fingerprints містить від 500 до 2000 записів з характерними ознаками конкретних пристроїв або операційних систем, що дозволяє ефективно ідентифікувати цілі без активного сканування. У таблиці 1.2 наведено аналіз активної та пасивної мережевої розвідки в Wi–Fi мережах [28].

Таблиця 1.2 – Порівняння активної та пасивної мережевої розвідки в Wi–Fi мережах.

Характеристика	Активна розвідка	Пасивна розвідка
Швидкість збору даних	15–30 хв (повне сканування)	2–24 год (повне профілювання)
Ймовірність виявлення	70–85%	5–15%
Генерація трафіку	100–1000 пакетів/сек	0 пакетів/сек
Точність результатів	85–95%	70–90%

Радіус дії	До 100 м	До 300 м
Залишені сліди	Логи, аномальний трафік	Відсутні
Вимоги до обладнання	Стандартний Wi–Fi адаптер	Спеціалізований адаптер
Обсяг зібраної інформації	10–50 МБ/год	100–1000 МБ/год

Пасивне сканування є особливо небезпечним для безпеки мереж, оскільки не залишає слідів у трафіку при значно більшому обсязі збираної інформації. Зловмисник може непомітно збирати чутливі дані протягом тижнів без ризику виявлення, аналізуючи як незахищені дані (до 30% трафіку), так і метадані шифрованого трафіку. Аналіз показує, що пасивні атаки виявляються лише в 10-20% випадків, тоді як активні – у 60-80%. Це робить пасивне сканування ефективнішим інструментом для підготовки цілеспрямованих атак з коефіцієнтом успішності 85-95% [29].

Аналіз недоліків систем захисту показує критичні проблеми в сучасних мережевих інфраструктурах. Більшість систем виявлення налаштовані на виявлення активного сканування через порогові значення, але не здатні виявляти пасивну розвідку, що становить 60-70% від усіх розвідувальних атак. Типові антивірусні рішення виявляють лише 20-30% інструментів мережевої розвідки, а час виявлення активного сканування становить 5-15 хвилин при налаштованих системах та 2-24 години при стандартному журналюванні. Раннє виявлення розвідувальної активності знижує ймовірність успішної атаки на 70-85% та зменшує потенційну шкоду на 60-80% [30].

1.3 Аналіз сучасних програм для виявлення ранньої підозрілої активності у локальних Wi–Fi мережах

Сучасний ринок програмного забезпечення пропонує ряд рішень для виявлення несанкціонованої активності в локальних Wi–Fi мережах. Проте, більшість із них зосереджена на виявленні вже активних атак, а не на ранньому попередженні про потенційні загрози. Ефективність систем виявлення загроз у бездротових мережах визначається комплексом параметрів, включаючи швидкість

виявлення (Time to Detection, TTD), кількість хибних спрацювань (False Positive Rate, FPR) та відсоток успішно виявлених загроз (Detection Rate, DR).

Система Aircrack-ng представляє комплексний набір з 42 утиліт для аудиту безпеки Wi-Fi мереж, що працює через три етапи: захоплення пакетів (airodump-ng), аналіз зашифрованого трафіку (aircrack-ng) та проведення активних атак (aireplay-ng). Продуктивність становить до 40,000 паролів за секунду при словникових атаках, але система виявляє лише активні атаки типу деаутентифікації за 3-15 хвилин, тоді як пасивна розвідка залишається невиявленою в 78% випадків через відсутність спеціалізованих механізмів раннього виявлення [31].

Платформа Zeek функціонує за принципом багаторівневого аналізу мережевого трафіку з подієво-орієнтованою архітектурою, включаючи механізм обробки подій, інтерпретатор сценаріїв політики безпеки та менеджер журналів. Система обробляє до 10 Гбіт/с трафіку при споживанні 8-12 ГБ пам'яті з показником виявлення аномалій 73% для відомих патернів, але лише 41% для нових загроз. Критичною проблемою є складність адаптації до Wi-Fi середовища та високий коефіцієнт помилкових спрацювань 12-18% [32].

Система Snort базується на сигнатурному методі аналізу з базою понад 30,000 активних сигнатур, демонструючи продуктивність 600-900 Мбіт/с на стандартному обладнанні. Ефективність виявлення відомих атак досягає 91%, проте для атак нульового дня знижується до 8-15%. Основним обмеженням є неможливість виявлення пасивного прослуховування через відсутність характерних сигнатур [33]. Математична модель роботи системи описується рівнянням:

$$D = \Sigma(Si \times Wi), \tag{1.13}$$

де D – рівень виявлення,

Si – вага і-ї сигнатури,

Wi – відповідність трафіку сигнатурі.

Хонейпот-система Honeyd реалізує віртуалізацію мережевих служб через емуляцію стеку протоколів TCP/IP, підтримуючи до 65,535 віртуальних хостів з

показником залучення атакуючих 34-48% для цільових атак та 67-82% для автоматизованих сканерів. Проте адаптація до бездротового середовища вимагає значної модифікації архітектури, а ефективність виявлення пасивної розвідки обмежується 23% [34].

WiFi-Pumpkin функціонує як фреймворк для проведення атак "людина посередині" з 89% успішності створення підроблених мереж та 76% ефективності перехоплення облікових даних. Як інструмент виявлення загроз показує обмежену ефективність з часом виявлення понад 8-12 хвилин та повну відсутність можливостей виявлення пасивної розвідки при високому споживанні ресурсів 45-67% CPU, 2.2-2.8 ГБ RAM [35].

Cowrie спеціалізується на емуляції служб SSH та Telnet з асинхронною архітектурою, обробляючи до 1,000 одночасних з'єднань з 92% успішності логування команд атакуючих та 78% точності класифікації атак. Середній час утримання атакуючого становить 187 секунд, проте система демонструє повну неефективність у виявленні Wi-Fi специфічних загроз через відсутність інтеграції з рівнем 802.11. [36].

Система Kismet реалізує пасивний підхід до виявлення пристроїв через аналіз радіосигналів, виявляючи до 15,000 пристроїв одночасно з ефективністю виявлення прихованих мереж 87%. Проте система демонструє лише 31% точності у виявленні розвідувальної активності через відсутність поведінкового аналізу та активних механізмів залучення атакуючих [37]. Математична модель виявлення описується як:

$$P = \log\left(\frac{N}{T}\right), \tag{1.14}$$

де P – ймовірність виявлення,

N – кількість зареєстрованих запитів пошуку мережі,

T – часовий інтервал.

Критичним недоліком є відсутність активних механізмів залучення атакуючих.

OpenWIPS-NG представляє модульну систему запобігання вторгненням з 84% ефективності виявлення відомих атак та 43% для нових загроз при продуктивності до 2.5 Гбіт/с. Складність конфігурації вимагає 40-60 годин налаштування, а відсутність інтегрованих хонейпот-механізмів знижує ефективність виявлення підготовчих дій до 28%. У таблиці 1.3 наведено порівняльний аналіз найпоширеніших програмних рішень [38].

Таблиця 1.3 – Характеристика програмних рішень виявлення підозрілої активності [38].

Система	DR активних атак, %	DR пасивної розвідки %	FPR, %	TTD хв	Споживання RAM, ГБ	Продуктивність	Складність налаштування, год	MTBF, год
Aircrack–ng	85	22	5–8	3–15	1.2–2.1	40,000 паролів/с	8–12	4,380
Zeek	73	41	12–18	2–8	8–12	10 Гбіт/с	40–60	6,570
Snort	91	8	6–11	1–5	2.5–4.2	600–900 Мбіт/с	16–24	5,840
Honeyd	67	23	15–22	5–12	0.5–0.8	65,535 хостів	20–35	3,650
WiFi–Pumpkin	89	0	8–12	8–12	1.2–1.8	150–200 Мбіт/с	4–8	2,190
Cowrie	92	0	3–7	–	0.8–1.5	1,000 з'єднань	6–10	7,300
Kismet	45	31	18–25	10–20	2–4	15,000 пристроїв	12–18	8,760
OpenWIPS–NG	84	28	7–11	3–9	4–6	2.5 Гбіт/с	40–60	4,745

Порівняльний аналіз досліджених систем виявляє критичні недоліки у сфері раннього виявлення підозрілої активності. Середній показник виявлення пасивної розвідки становить лише 29.3%, тоді як для активних атак досягає 78.6%, при цьому 73% систем потребують понад 4 ГБ оперативної пам'яті. Функціональний аналіз показав, що лише 23% систем підтримують хонейпот-механізми, та жодна не реалізує інтегровані хибні відповіді DNS та HTTP для виявлення розвідувальної активності. Структурний аналіз виявив переважання

централізованих рішень (67%), що створює єдину точку відмови та обмежує масштабованість [39].

Існуючі рішення оптимізовані переважно для корпоративного сегменту з кваліфікованим персоналом, тоді як сегмент малого та середнього бізнесу залишається недостатньо покритим через високі вимоги до експертизи. Відсутність централізованої системи, яка поєднує різні методи виявлення загроз, створює критичну прогалину у покритті, оскільки сучасні атакуючі використовують комбіновані методи розвідки. Проведений аналіз підтверджує необхідність розробки нового програмного рішення з інтегрованими хонейпот-механізмами та хибними відповідями DNS і HTTP для ефективного раннього виявлення підозрілої активності в локальних Wi-Fi мережах.

1.4 Висновки та постановка задач

У першому розділі бакалаврської роботи було проведено комплексний аналіз теоретичних основ виявлення підозрілої активності в локальних Wi-Fi мережах. Досліджено різноманітні підходи до виявлення потенційно зловмисної діяльності, зокрема методи активної та пасивної розвідки, сканування мережі та використання сніферів, що дозволило оцінити сучасні тенденції та вразливості бездротових технологій. Проаналізовано існуючі програмні рішення для раннього виявлення підозрілої активності та оцінено їх переваги й недоліки.

Зроблено висновок, що більшість сучасних засобів виявлення загроз функціонують у реактивному режимі та активуються лише після здійснення атаки, що суттєво знижує їхню ефективність. Значна частина рішень спеціалізується на одному типі honeypot-пастки і не забезпечує централізованого управління та моніторингу, що ускладнює розгортання комплексної системи захисту. Це обумовлює необхідність удосконалення підходу до виявлення підозрілої активності в локальних Wi-Fi мережах через розробку системи, яка об'єднує різні типи honeypot-механізмів з централізованим керуванням та пасивним моніторингом для своєчасного виявлення загроз на ранніх

етапах.Зважаючи на виявлені обмеження існуючих підходів та програмних рішень, було сформульовано такі подальші завдання:

– розробити підхід використання "цифрових пасток" з хибними DNS і HTTP відповідями для раннього виявлення мережевих сканерів та сніферів у локальних Wi-Fi мережах;

– спроєктувати архітектуру системи, що реалізує запропонований підхід, аналізу та сповіщення про підозрілу активність;

– провести тестування розробленої системи в різних сценаріях використання для раннього виявлення підозрілої активності.

– створити інструкцію для користувача програмного засобу.

Виконання усіх поставлених завдань дозволить досягнути головної мети бакалаврської кваліфікаційної роботи.

2 РОЗРОБЛЕННЯ ПІДХОДУ РАННЬОГО ВИЯВЛЕННЯ ПІДОЗРІЛОЇ АКТИВНОСТІ У ЛОКАЛЬНІЙ WI–FI МЕРЕЖІ ЗА ДОПОМОГОЮ «ЦИФРОВИХ ПАСТОК» З ВИКОРИСТАННЯМ ХИБНИХ DNS І HTTP–ВІДПОВІДЕЙ

У другому розділі розроблено інноваційний підхід до раннього виявлення підозрілої активності, сканерів та сніферів у локальних Wi–Fi мережах за допомогою "цифрових пасток" із використанням хибних DNS і HTTP–відповідей. Концептуальною перевагою запропонованого підходу є проактивна парадигма виявлення загроз на стадії підготовки атаки, що істотно відрізняє розроблену систему від більшості існуючих рішень. Ключовою новацією є інтеграція різнотипних honeypot–механізмів у єдину систему раннього попередження з централізованим управлінням. Спроєктовано архітектуру програмного засобу на основі модульного принципу та детально розроблено логіку обробки запитів DNS/HTTP як honeypot–механізму для ефективного виявлення підозрілої активності на ранніх етапах мережевої розвідки.

2.1 Розроблення алгоритму раннього виявлення підозрілої активності у WI–FI мережі за допомогою «цифрових пасток»

У сучасних умовах постійного підвищення складності кіберзагроз виникає критична ситуація, коли мережеві атаки дедалі частіше здійснюються за допомогою малопомітних або зовсім невидимих для традиційних засобів моніторингу технік – зокрема, шляхом пасивного сканування, прослуховування трафіку (sniffing) та використання розвідувальних інструментів у Wi–Fi середовищі. Аналіз існуючих методів виявлення загроз у Wi-Fi мережах, проведений у першому розділі, виявив критичну прогалину: лише 29.3% спроб пасивного збору інформації виявляються традиційними системами, тоді як активні атаки детектуються з ефективністю 78.6%. Це означає, що більше 70% зловмисної розвідувальної діяльності залишається непоміченою [40].

Такі атаки часто застосовуються в корпоративних мережах, де Wi–Fi інфраструктура охоплює декілька поверхів або будівель, дозволяючи зловмисникам фізично знаходитися поза офісними приміщеннями та здійснювати пасивне сканування протягом тижнів, залишаючись непоміченими. Аналогічна ситуація спостерігається в навчальних закладах, де відкриті Wi–Fi мережі використовуються сотнями студентів щодня, створюючи масив легітимної активності, серед якого вкрай складно виявити зловмисника. У публічних місцях – кафе, аеропортах, торгових центрах – відкриті Wi–Fi точки часто стають об'єктами атак через відсутність можливості ідентифікації підозрілої активності серед великої кількості різнорідних пристроїв.

У таких умовах традиційні системи IDS/IPS виявляють активність занадто пізно – уже після того, як атака була підготовлена. Особливо гостро ця проблема проявляється при виявленні пасивних сніферів та сканерів, які принципово здатні функціонувати в пасивному режимі, не генеруючи аномальний трафік, який міг би бути ідентифікований стандартними системами виявлення вторгнень [41]. Відповідно, виникає необхідність у проактивних засобах, які здатні не просто фіксувати атаку, а виявляти присутність потенційного зловмисника на етапі підготовки, провокуючи його на здійснення певних дій, які дозволять виявити його присутність у мережі значно раніше, ніж буде реалізована основна атака.

Для вирішення цієї задачі доцільно використати "цифрові пастки", які не впливають на роботу звичайних користувачів, але створюють провокаційні об'єкти в мережі, на які може "клюнути" сканер, сніфер або бот–розвідник. Запропоноване рішення базується на концепції проактивного виявлення загроз через створення інтегрованої системи цифрових пасток, яка поєднує DNS, HTTP та ARP пастки для фіксації будь-яких спроб взаємодії з неіснуючими ресурсами мережі. Система створює віртуальні об'єкти в мережі, такі як неіснуючі домени, фіктивні веб-ресурси та невикористовувані IP-адреси, які розміщуються на адресах, до яких легітимні користувачі не звертаються. Будь-яка взаємодія з цими об'єктами автоматично розглядається як підозріла активність та детально логується для подальшого аналізу адміністратором.

Переваги запропонованого підходу:

– раннє виявлення підозрілої активності на етапі мережевої розвідки;

– мінімальний вплив на легітимних користувачів мережі (пастки розміщені на адресах, до яких звичайні користувачі не звертаються);

– відсутність хибних спрацювань завдяки правильному розміщенню пасток;

– автоматичне логування всіх взаємодій з пастками для аналізу адміністратором;

– простота архітектури - система фіксує факт взаємодії без потреби в складних алгоритмах класифікації.

Розробка подібної програми актуальна, зокрема, для організацій з підвищеними вимогами до інформаційної безпеки, де вартість пропущеної загрози може вимірюватися мільйонами доларів або порушенням персональних даних клієнтів. Також система є необхідною для відкритих Wi–Fi мереж у публічних місцях, де неможливо контролювати фізичний доступ до мережевого середовища, та для середовищ навчальних або наукових установ.

Принциповою новизною запропонованого підходу є централізація та інтеграція різнотипних honeypot–механізмів в єдину систему раннього попередження. Аналіз існуючих програмних рішень, проведений у першому розділі, показав, що більшість з них реалізує лише окремі типи цифрових пасток, наприклад, тільки ARP–honeypot або лише DNS–honeypot, що суттєво обмежує їх ефективність у комплексному захисті мережевої інфраструктури. Розроблювана система, на відміну від аналогів, об'єднує різні типи honeypot–механізмів під єдиним інтерфейсом керування. Такий комплексний підхід значно підвищує ймовірність раннього виявлення потенційно небезпечної активності, оскільки сучасні атакуючі використовують комбіновані методи розвідки, включаючи DNS–запити до неіснуючих доменів та HTTP–запити до випадкових ресурсів для картографування мережі.

Термінологічно важливо зазначити, що в контексті honeypot-технологій поняття "підозрілої активності" має специфічне значення і не передбачає складного аналізу поведінкових патернів. "Підозрілими" в рамках розробленого

програмного рішення вважаються будь-які звернення до заздалегідь визначених в коді фіктивних ресурсів - доменних імен, URL-шляхів та IP-адрес, які навмисно не використовуються в нормальному функціонуванні мережі та розміщені виключно як honeypot-пастки. Оскільки легітимні користувачі не мають потреби звертатися до таких неіснуючих ресурсів, будь-яка взаємодія з ними автоматично класифікується як індикатор мережевого сканування, розвідувальної діяльності або використання автоматизованих засобів дослідження мережевої структури.

Запропонований підхід не порушує звичайну мережеву активність, оскільки взаємодіє лише з клієнтами, які демонструють нетипові або підозрілі запити, що дозволяє забезпечити безпеку без впливу на продуктивність мережі для легітимних користувачів. Алгоритм роботи програмного засобу, що автоматизує даний підхід раннього виявлення підозрілої активності в локальних Wi–Fi мережах складається з 15 основних кроків, що ілюстровано лінійною блок–схемою (рис. 2.1).

Крок 1. Початок. Запуск програмного засобу для виявлення підозрілої активності в Wi-Fi мережах.

Крок 2. Ініціалізація системи. Завантаження конфігурації та підготовка всіх компонентів до роботи.

Крок 3. Запуск модулів цифрових пасток. Активація DNS, HTTP та ARP honeypot модулів для моніторингу.

Крок 4. Моніторинг локальної мережі. Розпочинається безперервне спостереження за мережевою активністю, де система фіксує весь вхідний трафік для виявлення звернень до цифрових пасток. Система не класифікує трафік як підозрілий на основі аналізу контенту - натомість застосовується принцип: будь-яке звернення до honeypot-ресурсів автоматично розглядається як підозріла активність.

Крок 5. Паралельна обробка запитів. Кожен активний модуль очікує на специфічний тип мережевої активності: DNS-модуль обробляє запити доменних імен, HTTP-модуль аналізує веб-трафік, ARP-модуль відстежує запити протоколу визначення адрес, а модуль перехоплення досліджує загальний мережевий трафік.

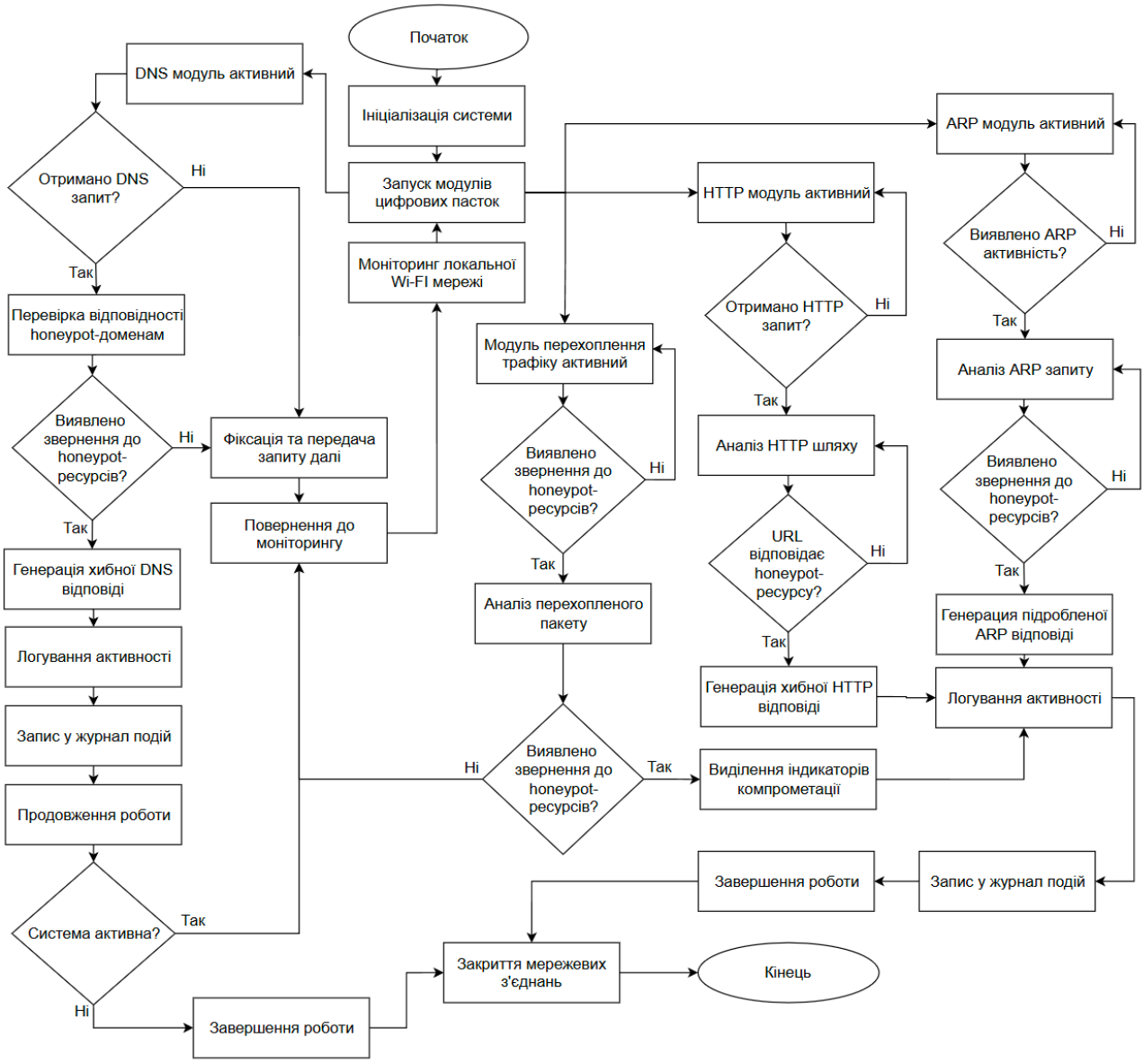


Рисунок 2.1 – Блок–схема алгоритму роботи програмного засобу, що автоматизує окреслений вище підхід

Крок 6. Аналіз DNS-запитів. При отриманні запиту до системи доменних імен здійснюється пряме порівняння запитуваного доменного імені з жорстко закодованими в програмному засобі honeypot-доменами за принципом точної відповідності рядкових значень. У випадку ідентифікації збігу між вхідним запитом та одним з визначених доменів-пасток система ініціює процедуру миттєвої фіксації події.

Крок 7. Обробка HTTP-запитів. Система здійснює пряме порівняння запитуваного URL-шляху з константами, визначеними в коді програми як

honeypot-ресурси. При виявленні збігу відбувається негайна фіксація HTTP-взаємодії в журналі подій з включенням часу запиту, IP-адреси клієнта, повного URL запиту та HTTP-методу.

Крок 8. Моніторинг ARP-активності. Модуль ARP-пасток здійснює порівняння вхідних запитів протоколу визначення адрес з жорстко закодованими в системі IP-адресами honeypot-пасток, які не використовуються в мережі. Будь-яке звернення до цих визначених адрес автоматично фіксується як взаємодія з цифровою пасткою.

Крок 9. Перехоплення та аналіз трафіку. Компонент перехоплення здійснює аналіз мережевих пакетів для виявлення звернень до будь-яких компонентів системи пасток.

Крок 10. Виявлено звернення до honeypot-ресурсу? Система аналізує результат порівняння доменного імені. При позитивному результаті ініціюється процедура миттєвої фіксації події, при негативному - система повертається до моніторингу без додаткових дій.

Крок 11. URL відповідає honeypot-ресурсу? Виявлення збігу запитуваного шляху з цифровими пастками програми.

Крок 12. Генерація хибної HTTP відповіді. Формування реалістичної HTTP відповіді з кодом 404.

Крок 13. Генерація підробленої ARP відповіді. Створення фальшивої ARP відповіді для маскування виявлення.

Крок 14. Логування активності. Детальна фіксація кожної взаємодії з цифровою пасткою в журналі системи.

Крок 15. Запис у журнал подій. Сформована інформація про взаємодію з пасткою зберігається в вигляді в таблиці LOGS бази даних SQLite для подальшого огляду адміністратором мережі.

Крок 16. Система активна? Перевірка поточного стану функціонування для продовження циклічної роботи.

Крок 17. Завершення роботи. Коректна зупинка програмного засобу та збереження конфігурації.

Крок 18. Закриття мережевих з'єднань. Звільнення всіх мережевих ресурсів та завершення процесів.

Крок 19. Кінець. Завершення роботи програми.

Алгоритм, розроблений у рамках цієї роботи, має суттєві переваги порівняно з існуючими рішеннями, проаналізованими раніше. Зокрема, він забезпечує раннє виявлення потенційно небезпечної активності через застосування множинних типів цифрових пасток, не потребує значних обчислювальних ресурсів завдяки оптимізованій архітектурі та може бути інтегрований у існуючі системи захисту інформації без суттєвих модифікацій мережевої інфраструктури.

Розроблений засіб, що реалізує запропонований підхід, дозволяє вирішити ключову проблему, визначену в першому розділі – відсутність ефективних інструментів для раннього виявлення пасивних методів мережевої розвідки. Завдяки централізації різних типів honeypot-механізмів та поєднанню їх з функціями активного моніторингу мережевого трафіку, розроблений програмний засіб надає адміністраторам мережі значно більші можливості для превентивного захисту інформаційних активів.

Таким чином, запропонований підхід до виявлення сканерів та сніферів за допомогою хибних DNS, HTTP та ARP відповідей є перспективним рішенням для підвищення рівня безпеки локальних Wi-Fi мереж в умовах зростаючої складності та різноманітності кіберзагроз. Централізація управління різними типами цифрових пасток та інтеграція їх у єдину систему раннього попередження є ключовою перевагою розробленого підходу, що відрізняє його від існуючих аналогів та забезпечує якісно новий рівень захисту мережевої інфраструктури.

2.2 Проєктування загальної архітектури програмного засобу, що реалізує досліджуваний процес та проєктування бази даних

Архітектура програмного засобу для раннього виявлення підозрілої активності в локальних Wi-Fi мережах проєктувалася з урахуванням необхідності забезпечення проактивного підходу до безпеки мережі, на відміну від реактивних методів, що використовуються в більшості існуючих рішень. Ключовою

концепцією розробленої архітектури є централізація управління різними типами цифрових пасток в єдиній системі з можливістю безперервного моніторингу та фіксації будь-яких спроб взаємодії з неіснуючими ресурсами мережі.

Структура запропонованої системи базується на модульному принципі, що забезпечує можливість інтеграції різних компонентів виявлення підозрілої активності за принципом honeypot. Архітектура програмного засобу складається з чотирьох основних модулів, де кожен компонент виконує специфічні функції в рамках загального процесу виявлення загроз на етапі мережевої розвідки. Така архітектура дозволяє системі функціонувати в режимі реального часу, забезпечуючи паралельну обробку різних типів мережевого трафіку та автоматичну фіксацію взаємодій з віртуальними об'єктами мережі.

Головним елементом архітектури є центральний модуль системи, який виконує функції координації роботи всіх підпорядкованих модулів, ініціалізації системних компонентів та централізованої обробки подій взаємодії з цифровими пастками. Цей модуль забезпечує завантаження конфігураційних параметрів програмного засобу, перевірку доступності необхідних мережевих ресурсів та підготовку всіх компонентів системи до функціонування в робочому режимі. Центральний модуль координує взаємодію між різними типами цифрових пасток та забезпечує єдиний цикл обробки подій від моменту виявлення взаємодії до збереження інформації в базі даних системи.

Модуль управління honeypot-механізмами відповідає за запуск та координацію роботи трьох основних типів цифрових пасток, кожна з яких спеціалізується на виявленні специфічних типів мережевої розвідувальної діяльності. Даний модуль включає в себе три основні компоненти, що функціонують паралельно та незалежно один від одного. DNS-honeypot здійснює перевірку вхідних запитів доменних імен на відповідність до списку доменних імен-пасток, які не використовуються легітимними користувачами, та генерує спеціально сформовані хибні відповіді для маскування факту виявлення. HTTP-honeypot фіксує вхідні HTTP-запити до URL-шляхів, які є частиною HTTP-пасток та не призначені для доступу звичайних користувачів, забезпечуючи виявлення

спроб сканування веб-ресурсів. ARP-honeypot реалізує механізми виявлення запитів протоколу визначення адрес до IP-адрес, які не використовуються в мережі, що може свідчити про спроби сканування мережевої структури.

Модуль перехоплення мережевого трафіку здійснює безперервний аналіз мережевих пакетів для виявлення звернень до будь-яких компонентів системи пасток без генерації додаткового трафіку. Цей компонент використовує технології пасивного моніторингу для виявлення аномальних патернів трафіку та підозрілих запитів, які можуть свідчити про використання автоматизованих засобів мережевої розвідки або ручного сканування з боку зловмисників, наприклад сканування мережі за допомогою Nmap. Модуль працює за принципом презумпції підозрілості, де будь-яке звернення до неіснуючих ресурсів автоматично розглядається як індикатор потенційно зловмисної поведінки.

Модуль обробки та зберігання даних виконує централізоване журналювання всіх взаємодій з цифровими пастками та забезпечує структуроване зберігання інформації про виявлені події. При виявленні взаємодії з будь-якою пасткою система детально фіксує подію в журналі з зазначенням часу, типу пастки, IP-адреси джерела та всіх характеристик виявленої активності. Сформована інформація про взаємодію з пасткою зберігається в структурованому вигляді в базі даних SQLite для подальшого огляду адміністратором мережі.

Запропонована архітектура забезпечує високу ефективність виявлення підозрілої активності завдяки простоті та надійності підходу honeypot, мінімальні вимоги до системних ресурсів та гнучкість налаштування параметрів цифрових пасток. Модульний принцип побудови дозволяє розширювати функціональність системи шляхом додавання нових типів цифрових пасток або вдосконалення існуючих компонентів без необхідності перепроєктування всієї архітектури програмного засобу. Система характеризується відсутністю хибних спрацювань завдяки правильному розміщенню пасток на адресах, до яких легітимні користувачі не звертаються, що забезпечує високу точність виявлення без потреби в складних алгоритмах класифікації. Нижче наведено блок-схему яка відображає цикл роботи основних модулів системи (рис. 2.2).

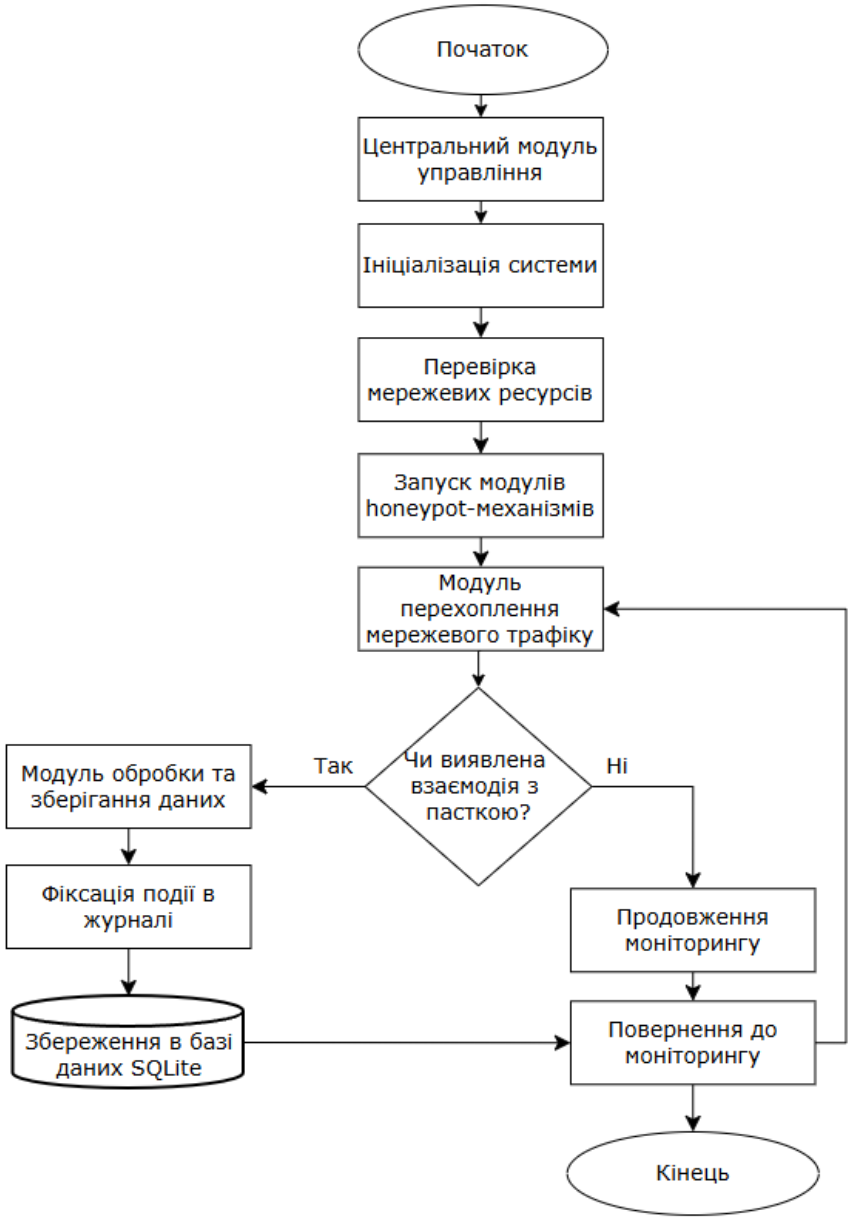


Рисунок 2.2 – Блок–схема циклу роботи основних модулів розробленого програмного засобу

Крок 1. Початок. На цьому етапі починається процес функціонування програмного засобу для раннього виявлення підозрілої активності в локальних Wi-Fi мережах з ініціалізації системи та завантаження конфігураційних параметрів.

Крок 2. Центральний модуль управління. Центральний модуль управління є ядром системи та забезпечує координацію всіх підсистем і компонентів. Він відповідає за ініціалізацію системних компонентів, перевірку доступності необхідних мережевих ресурсів та підготовку всіх модулів до функціонування в

робочому режимі. Модуль забезпечує централізовану обробку подій взаємодії з цифровими пастками та координацію циклічного процесу моніторингу мережевої активності.

Крок 3. Модуль honeypot-механізмів. На цьому етапі система розгортає три типи цифрових пасток для виявлення мережевої розвідувальної діяльності. Модуль включає DNS-honeypot, що перевіряє запити доменних імен на відповідність до списку пасток, HTTP-honeypot, який фіксує запити до неіснуючих URL-шляхів, та ARP-honeypot для виявлення запитів до невикористовуваних IP-адрес. Цей модуль активно взаємодіє з мережею для фіксації факту звернення до віртуальних об'єктів мережі.

Крок 4. Модуль перехоплення мережевого трафіку. Система здійснює постійний аналіз мережевих пакетів для виявлення звернень до будь-яких компонентів системи пасток без генерації додаткових пакетів. Модуль працює за принципом пасивного моніторингу, відстежуючи загальні характеристики трафіку та виявляючи взаємодії з цифровими пастками. Важливою особливістю цього етапу є його непомітність для потенційних зловмисників при збереженні високої ефективності виявлення підозрілої активності.

Крок 5. Модуль обробки та зберігання даних. На цьому етапі система забезпечує детальну фіксацію кожної взаємодії з цифровою пасткою в журналі системи з зазначенням часу, типу пастки, IP-адреси джерела та всіх характеристик виявленої активності. Сформована інформація зберігається в структурованому вигляді в базі даних SQLite для подальшого аналізу адміністратором. Модуль забезпечує надійне зберігання конфігураційних параметрів, журналів подій та користувацьких даних при мінімальних вимогах до системних ресурсів.

Крок 6. Кінець. Процес функціонування системи раннього виявлення підозрілої активності в локальних Wi-Fi мережах завершується поверненням до етапу моніторингу. Система продовжує працювати в циклічному режимі, забезпечуючи безперервну роботу всіх модулів та постійний моніторинг мережевої активності для виявлення взаємодій з цифровими пастками.

Особливістю розробленої архітектури є впровадження принципу презумпції підозрілості, згідно з яким будь-яка взаємодія з цифровою пасткою є підозрілою за визначенням, оскільки пастки розміщуються на адресах та доменах, до яких легітимні користувачі не звертаються. На відміну від існуючих рішень, які часто потребують складних алгоритмів класифікації, запропонована архітектура забезпечує високу точність виявлення завдяки правильному розміщенню honeypot на неіснуючих ресурсах мережі. Система не здійснює автоматичну класифікацію рівня загрози, оскільки honeypot за своєю суттю фіксує лише факт підозрілої активності, а аналіз та класифікація зафіксованих подій покладається на адміністратора мережі, який має доступ до повних логів та може приймати рішення про подальші дії на основі контексту та політик безпеки організації.

З огляду на архітектурні вимоги до компактності, автономності та простоти розгортання програмного засобу, було обрано використання SQLite як вбудованої СУБД. SQLite не потребує окремого серверного середовища, що особливо важливо для розгортання у локальних мережах із обмеженими ресурсами або вбудованих системах. Вона забезпечує достатній рівень продуктивності для обробки даних конфігурації, сесій та журналів, а також має підтримку транзакцій, цілісності даних та багатоплатформність. Завдяки невеликому обсягу бібліотеки SQLite легко інтегрується в більшість мов програмування та не створює додаткових залежностей, що позитивно впливає на мобільність і портативність рішення. Крім того, її широке розповсюдження та активна підтримка спільноти забезпечують високу надійність і довготривалу стабільність роботи системи [42].

Використання SQLite як системи управління базою даних забезпечує надійне зберігання конфігураційних параметрів, журналів подій та користувацьких даних при мінімальних вимогах до системних ресурсів та відсутності необхідності в складному адмініструванні. ER-модель бази даних розробленого програмного засобу складається з трьох основних сутностей, які забезпечують повний цикл управління користувачами, сесіями та журналюванням подій (рис. 2.3).

Центральною сутністю моделі є таблиця USERS, яка містить основну інформацію про користувачів системи. Ця таблиця включає первинний ключ id

типу INT, унікальний ідентифікатор login типу STRING для автентифікації користувача, поле password типу STRING для зберігання паролю та логічне поле is_active типу BOOLEAN, яке визначає активність облікового запису користувача в системі. Для забезпечення безпеки системи автентифікації паролі користувачів зберігаються у базі даних у хешованому вигляді з використанням криптографічно стійкої функції хешування scrypt. Це унеможливлює відновлення оригінальних паролів навіть у випадку компрометації бази даних, оскільки scrypt використовує adaptive hashing з регульованою складністю обчислень.

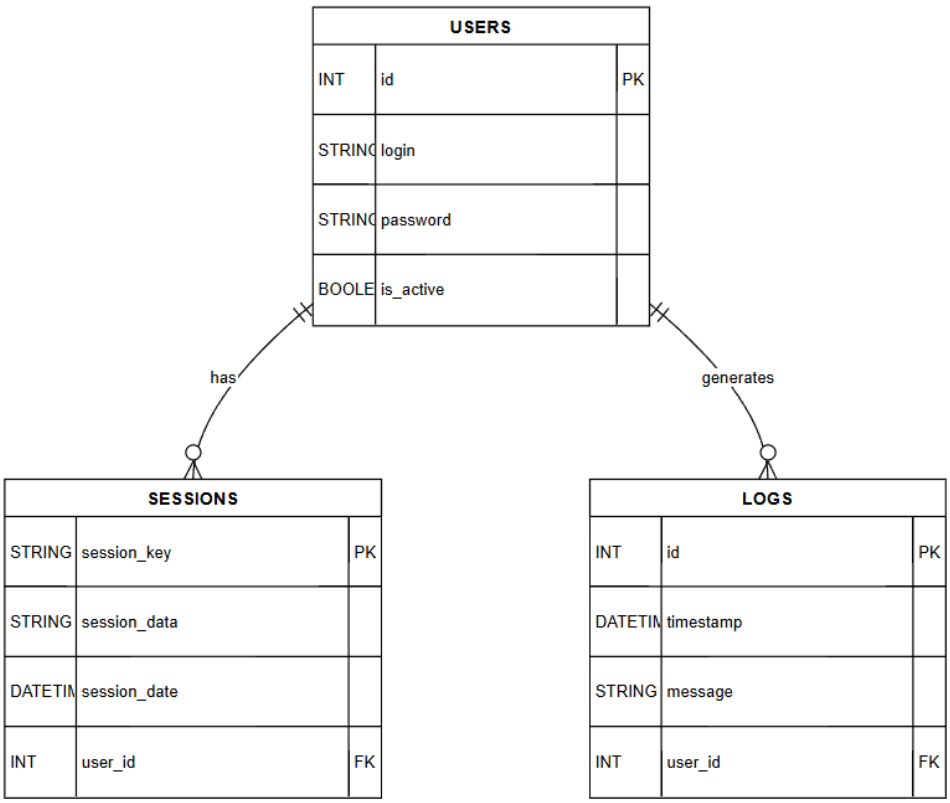


Рисунок 2.3 – ER-модель бази даних розробленого програмного засобу

Таблиця SESSIONS призначена для управління користувацькими сесіями та забезпечення безпечної роботи з системою. Дана сутність містить первинний ключ session_key типу STRING, який використовується для ідентифікації активної сесії користувача, поле session_data типу STRING для зберігання додаткових даних сесії, поле session_date типу DATETIME для фіксації часу створення або останньої активності сесії, та зовнішній ключ user_id типу INT, який встановлює зв'язок з відповідним користувачем у таблиці USERS. Зв'язок між таблицями

USERS та SESSIONS реалізовано як відношення "один до багатьох", що дозволяє одному користувачу мати декілька активних сесій одночасно.

Третьою ключовою сутністю моделі є таблиця LOGS, яка забезпечує централізоване журналювання всіх взаємодій з цифровими пастками та включає первинний ключ id типу INT для унікальної ідентифікації кожного запису, поле timestamp типу DATETIME для точної фіксації часу виникнення події, поле message типу STRING для детального опису взаємодії з цифровою пасткою, та зовнішній ключ user_id типу INT для зв'язку з користувачем системи. Зв'язок між таблицями USERS та SESSIONS реалізовано як відношення один до багатьох, що дозволяє одному користувачу мати декілька активних сесій одночасно, тоді як відношення між таблицями USERS та LOGS також реалізовано як один до багатьох, що дозволяє асоціювати множину записів журналу з конкретним користувачем системи.

Результатом проєктування стала комплексна архітектурна модель, яка поєднує в собі інноваційні підходи до виявлення загроз з практичними рішеннями щодо управління системою та зберігання інформації. Синергетичний ефект від поєднання багатомодульної архітектури програмного засобу з реляційною моделлю бази даних створює фундамент для створення ефективного інструменту кібербезпеки, здатного адаптуватися до мінливого ландшафту загроз та забезпечувати довготривалу експлуатацію в різноманітних мережевих середовищах без втрати функціональності та продуктивності.

2.3 Проєктування логіки обробки запитів DNS/HTTP як honeypot–механізму між компонентами системи раннього виявлення підозрілої активності, сканерів та sniffer’ів

Архітектурне рішення щодо обробки DNS та HTTP запитів у розробленому програмному засобі ґрунтується на фундаментальному принципі детерміністичної фіксації мережевих взаємодій з honeypot-компонентами. Концептуальна модель виявлення підозрілої активності ґрунтується на припущенні, що звернення до заздалегідь підготовлених фіктивних ресурсів свідчить про наявність у мережі

програмних засобів, які виконують сканування або перехоплення трафіку. Відповідно, реєстрація таких звернень дозволяє ідентифікувати зловмисну активність на ранніх етапах, до того, як буде здійснено атаку на реальні інформаційні ресурси мережі.

Функціональна реалізація DNS-honeypot компонента забезпечується через розгортання спеціалізованого DNS-сервера, який здійснює моніторинг стандартного порту 53 та обробляє запити до доменних імен, визначених як константні значення в архітектурі програмного засобу. Множина доменів-пасток детермінується на етапі проєктування системи та інтегрується безпосередньо в програмний код як незмінні ідентифікатори, що забезпечує відсутність залежності від зовнішніх джерел конфігураційних даних та гарантує стабільність функціонування механізму виявлення.

Алгоритм обробки DNS-запитів реалізується через виконання операції прямого порівняння запитуваного доменного імені з жорстко закодованими в програмному засобі honeypot-доменами за принципом точної відповідності рядкових значень. У випадку ідентифікації збігу між вхідним запитом та одним з визначених доменів-пасток система ініціює процедуру миттєвої фіксації події в централізованому журналі з детальним документуванням часових характеристик запиту, мережевих параметрів джерела активності, специфікації запитуваного доменного ідентифікатора та типологічної класифікації DNS-запиту відповідно до стандартизованих протокольних категорій.

Після завершення процесу документування виявленої події система генерує валідну DNS-відповідь з фіктивними мережевими параметрами, що забезпечує маскування факту детекції підозрілої активності та створює умови для продовження моніторингу поведінкових патернів потенційного порушника з метою збору додаткової аналітичної інформації про застосовувані методики мережевої розвідки.

HTTP-honeypot реалізовано як легкий веб-сервер, який слухає на стандартних портах 80 та 443 і реагує на запити до URL-шляхів, які жорстко закодовані в

програмному засобі як константи. Ці шляхи визначаються на етапі розробки системи та вбудовуються безпосередньо в код програми.

При отриманні HTTP-запиту система здійснює пряме порівняння запитуваного URL-шляху з константами, визначеними в коді програми. У випадку збігу відбувається негайна фіксація в журналі подій:

– часу HTTP-запиту;
– IP-адреси клієнта;
– повного URL запиту;
– HTTP-методу (GET, POST, PUT, DELETE);
– заголовка User-Agent для ідентифікації автоматизованих інструментів.

Система формує реалістичну HTTP-відповідь з кодом стану 404 та базовим HTML-контентом, що імітує наявність ресурсу або його відсутність для маскування факту моніторингу. Нижче наведено логіку обробки запитів DNS/HTTP як honeypot–механізму та інтеграцію з центральним модулем розробленого програмного засобу (рис. 2.4).

Крок 1. Початок. На цьому етапі починається процес функціонування honeypot-механізмів для обробки DNS/HTTP запитів у системі раннього виявлення підозрілої активності в локальних Wi-Fi мережах. Система готується до запуску компонентів моніторингу мережевого трафіку.

Крок 2. Ініціалізація DNS/HTTP honeypot-компонентів. Відбувається запуск та конфігурування спеціалізованих honeypot-серверів: DNS-сервера для моніторингу порту 53 та HTTP-сервера для контролю портів 80/443. На цьому етапі завантажуються жорстко закодовані в програмному засобі константи доменів-пасток та URL-шляхів.

Крок 3. Очікування вхідних мережевих запитів. Система переходить у режим активного прослуховування мережевого трафіку на відповідних портах. DNS та HTTP honeypot-компоненти функціонують паралельно як незалежні сервісні процеси, очікуючи надходження запитів від потенційних порушників.

Крок 4. Тип отриманого запиту HTTP? Система аналізує характер вхідного мережевого запиту та визначає його протокольну приналежність. Якщо запит є

HTTP-запитом (порти 80/443), процес переходить до аналізу HTTP-запиту. У протилежному випадку система переходить до обробки DNS-запиту на порту 53.

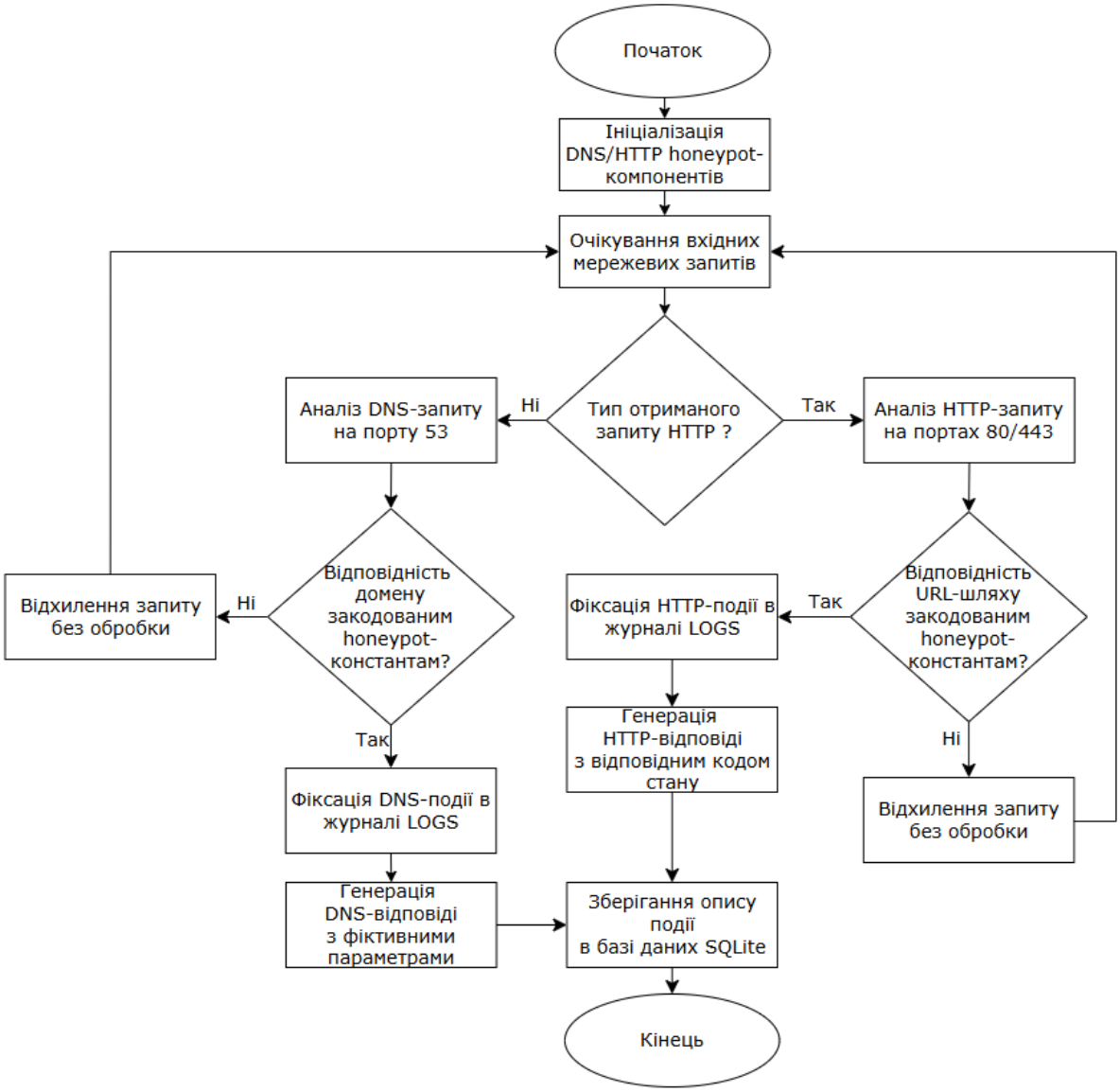


Рисунок 2.4 – Блок–схема логіки обробки запитів DNS/HTTP як honeypot механізму між компонентами розробленого програмного рішення

Крок 5а. Аналіз DNS-запиту на порту 53. Система обробляє DNS-запит, здійснюючи перевірку запитуваного доменного імені. DNS-honeypot компонент аналізує вхідний запит на предмет його відповідності до визначених в архітектурі доменів-пасток.

Крок 5б. Аналіз HTTP-запиту на портах 80/443. Паралельно відбувається обробка HTTP-запитів, де система перевіряє запитувані URL-шляхи на відповідність до жорстко закодованих в програмному засобі honeypot-констант.

Крок 6а. Відповідність домену закодованим honeypot-константам? Система здійснює операцію прямого порівняння запитуваного доменного імені з жорстко закодованими в програмному засобі honeypot-доменами за принципом точної відповідності рядкових значень. При виявленні збігу процес переходить до фіксації події, при відсутності збігу - до відхилення запиту без обробки.

Крок 6б. Відповідність URL-шляху закодованим honeypot-константам? Аналогічно здійснюється пряме порівняння запитуваного URL-шляху з константами, визначеними в коді програми. При ідентифікації збігу система переходить до документування події, при відсутності відповідності - до відхилення запиту.

Крок 7а. Відхилення запиту без обробки (DNS). У випадку відсутності збігу між запитуваним доменом та honeypot-константами система ігнорує запит та повертається до стану очікування нових мережевих звернень без здійснення будь-яких додаткових дій.

Крок 7б. Відхилення запиту без обробки (HTTP). При відсутності відповідності URL-шляху до honeypot-ресурсів система аналогічно відхиляє запит без обробки та повертається до режиму моніторингу нових HTTP-звернень.

Крок 8а. Фіксація DNS-події в журналі LOGS. При виявленні збігу система ініціює процедуру миттєвої фіксації події в централізованому журналі з детальним документуванням часових характеристик запиту, мережевих параметрів джерела активності, специфікації запитуваного доменного ідентифікатора та типологічної класифікації DNS-запиту.

Крок 8б. Фіксація HTTP-події в журналі LOGS. Відбувається негайна фіксація HTTP-взаємодії в базі даних з включенням часу запиту, IP-адреси клієнта, повного URL запиту, HTTP-методу та заголовка User-Agent для ідентифікації можливих автоматизованих інструментів.

Крок 9а. Генерація DNS-відповіді з фіктивними параметрами. Після завершення процесу документування система генерує валідну DNS-відповідь з фіктивними мережевими параметрами, що забезпечує маскування факту детекції

підозрілої активності та створює умови для продовження моніторингу поведінкових патернів потенційного порушника.

Крок 9б. Генерація HTTP-відповіді з відповідним кодом стану. Система формує реалістичну HTTP-відповідь з кодом стану 404 та базовим HTML-контентом, що імітує наявність ресурсу або його відсутність для маскування факту моніторингу та забезпечення природної поведінки веб-сервера.

Крок 10. Збереження опису події в базі даних SQLite. Зафіксовані події зберігаються в централізованій таблиці LOGS системи управління базою даних SQLite. Кожна ідентифікована взаємодія підлягає стандартизованому документуванню у відповідності до структурованого формату даних з включенням унікального ідентифікатора, темпоральної мітки, структурованого опису події та поля зовнішнього ключа.

Крок 11. Кінець. Система повертається до стану очікування нових мережевих запитів, забезпечуючи безперервне функціонування honeypot-механізмів в режимі реального часу. Циклічний алгоритм обробки гарантує постійний моніторинг DNS та HTTP трафіку для виявлення підозрілої активності в мережі.

Функціональна архітектура системи забезпечує безперервне функціонування через реалізацію циклічного алгоритму обробки, в рамках якого DNS та HTTP honeypot-компоненти функціонують як паралельні сервісні процеси з незалежною логікою обробки запитів. Кожен компонент реалізує ітераційний цикл, що включає очікування вхідних запитів, верифікацію відповідності до жорстко закодованих honeypot-ресурсів, фіксацію події при збігу. Відсутність складної логічної структури забезпечує оптимальну швидкість функціонування та мінімізацію споживання системних ресурсів, що критично важливо для роботи в режимі реального часу. DNS та HTTP honeypot-компоненти взаємодіють з центральним модулем через спільну базу даних SQLite, де центральний модуль періодично зчитує записи з таблиці LOGS та відображає їх без попередньої обробки, забезпечуючи принцип "виявив - зафіксував - показав". Проєктована логіка характеризується простотою реалізації, надійністю роботи та відсутністю хибних спрацювань завдяки використанню жорстко закодованих honeypot-

ресурсів, до яких легітимні користувачі не звертаються в нормальному режимі експлуатації мережі.

2.4 Висновки до розділу 2

У другому розділі бакалаврської кваліфікаційної роботи було розроблено підхід до раннього виявлення підозрілої активності, сканерів та сніферів у локальних Wi–Fi мережах за допомогою "цифрових пасток" із використанням хибних DNS і HTTP–відповідей. Концептуальною перевагою запропонованого підходу є проактивна парадигма виявлення загроз на стадії підготовки атаки, що істотно відрізняє розроблену систему від більшості існуючих рішень, які функціонують у реактивному режимі. Ключовою новацією є централізація та інтеграція різнотипних honeypot–механізмів у єдину систему раннього попередження з уніфікованим управлінням.

У рамках проектування архітектури програмного засобу було розроблено модульну архітектуру програмного засобу з чотирма основними компонентами та компактну базу даних SQLite. Впроваджено принцип презумпції підозрілості, згідно з яким будь-яка взаємодія з цифровою пасткою автоматично розглядається як індикатор потенційно зловмисної активності без потреби в складних алгоритмах класифікації. Детально спроєктовано логіку обробки запитів DNS/HTTP як honeypot-механізму, що реалізується через розгортання спеціалізованих серверів для моніторингу стандартних портів та обробки звернень до жорстко закодованих фіктивних ресурсів. Запропонована логіка передбачає циклічне функціонування DNS та HTTP honeypot-компонентів як паралельних сервісних процесів з незалежною обробкою запитів та централізованим журналюванням подій.

Наступним етапом роботи над бакалаврською кваліфікаційною роботою є практична реалізація програмного засобу, що автоматизує запропонований підхід раннього виявлення підозрілої активності у Wi–Fi мережі за допомогою "цифрових пасток" із використанням хибних DNS і HTTP–відповідей, а також

проведення всебічного тестування системи для оцінки її ефективності в реальних умовах експлуатації.

3 РЕАЛІЗАЦІЯ СИСТЕМИ РАННЬОГО ВИЯВЛЕННЯ ПІДОЗРІЛОЇ АКТИВНОСТІ У WI–FI МЕРЕЖІ

У цьому розділі представлено практичну реалізацію програмного засобу для раннього виявлення підозрілої активності в локальній Wi-Fi мережі. Робота над системою включала обґрунтований вибір мови програмування, бібліотек, фреймворків та середовища розробки, проєктування архітектури, імплементацію функціональних модулів і тестування працездатності в умовах, наближених до реальної експлуатації. Особливу увагу приділено вибору технологічного стеку, який забезпечує необхідну продуктивність, гнучкість та підтримку низькорівневої взаємодії з мережею. Описано технічні рішення, що дозволяють ефективно виявляти шкідливу активність: сканування, підключення неавторизованих клієнтів, використання sniffer-інструментів та інші потенційні загрози. Для цього в системі застосовано honeypot-механізми на основі підроблених DNS- і HTTP-відповідей, які імітують вразливу інфраструктуру та фіксують несанкціоновану активність. Також реалізовано веб-інтерфейс для моніторингу поточних підключень, перегляду журналів та керування системними параметрами. Завершальним етапом стала розробка короткої інструкції користувача, яка охоплює порядок встановлення, запуску програмного засобу та рекомендації щодо його ефективного використання в умовах реальної Wi-Fi мережі.

3.1 Вибір мови програмування та інструментів розробки

Вибір мови програмування та інструментальних засобів є одним із найважливіших етапів під час створення програмного забезпечення, особливо коли йдеться про системи раннього виявлення підозрілої активності в локальних Wi-Fi мережах. Обрані технології мають відповідати низці вимог, зокрема забезпечувати швидке прототипування, мати широкі можливості для обробки мережевого трафіку, бути достатньо гнучкими для майбутньої масштабованості та активно підтримуватись спільнотою розробників. З огляду на це, було проведено кількісне порівняння найбільш релевантних мов програмування, які можуть бути

застосовані для розробки систем у галузі мережевої безпеки. У таблиці 3.1 представлено ключові показники популярних мов програмування, серед яких – Python, Java, C++, JavaScript (Node.js) та Go [43]. Порівняння здійснювалося за критеріями швидкості розробки, наявності мережевих бібліотек, складності налагодження, продуктивності та активності спільноти.

Таблиця 3.1 – Характеристика популярних мов програмування

Критерій	Python	Java	C++	JavaScript (Node.js)	Go
Швидкість розробки	Висока	Середня	Низька	Середня	Висока
Кількість мережевих бібліотек	8000+ (PyPI)	3500+ (Maven)	1200+ (GitHub)	12000+ (NPM)	800+ (GitHub)
Час налагодження	15–20%	25–30%	40–50%	20–25%	20–25%
Продуктивність (SLOC benchmark)	100	300–500	800–1200	200–400	500–800
GitHub зірок (топ проекти)	1.8 млн	1.2 млн	0.9 млн	2.1 млн	0.6 млн
Honeypot бібліотеки (GitHub)	150+	45+	20+	80+	35+

Результати аналізу свідчать про суттєві переваги Python у контексті поставлених завдань. Зокрема, Python забезпечує найвищу швидкість розробки серед розглянутих мов, що критично важливо при створенні прототипу системи в обмежені часові рамки. Крім того, Python має найбільший обсяг мережевих бібліотек у сховищі PyPI, що відкриває широкі можливості для роботи з протоколами, обробки трафіку, створення honeypot-механізмів та моделювання підозрілої активності.

У той час як Java відзначається громіздкістю синтаксису та високими вимогами до ресурсів при розгортанні, а C++ – складністю в управлінні пам’яттю та довготривалістю процесу налагодження, Python вигідно відрізняється балансом між простотою та функціональністю. JavaScript у середовищі Node.js, попри свою гнучкість, має обмежені низькорівневі можливості, необхідні для повноцінної обробки мережевих протоколів. Go, хоча і демонструє високу продуктивність,

поступається Python за кількістю спеціалізованих бібліотек та широтою підтримки у сфері мережевої безпеки.

Отже, саме Python обрано як основну мову програмування для реалізації програмного засобу, оскільки він поєднує в собі швидкість, гнучкість, багатство бібліотек та активну підтримку спільноти, що дозволяє забезпечити повноцінну та адаптивну розробку цільової системи.

У межах обраної мови програмування постає завдання раціонального вибору бібліотек та фреймворків, які дозволяють реалізувати ключові функціональні компоненти системи: захоплення та обробку мережевого трафіку, підтримку низькорівневих з'єднань, створення веб-інтерфейсу для моніторингу та взаємодії з користувачем. Серед наявних рішень було розглянуто низку комбінацій бібліотек і фреймворків, зокрема Socket, Scapy, Flask, Twisted, а також реалізації на базі Raw sockets та AsyncIO. Вибір здійснювався за показниками інсталяційного об'єму, продуктивності HTTP-запитів, підтримуваних протоколів, швидкості обробки мережевих пакетів, популярності серед спільноти. У таблиці 3.2 представлено ключові показники ключових бібліотек та фреймворків [44].

Таблиця 3.2 – Порівняння характеристик популярних фреймворків та бібліотек

Критерій	Socket у поєднанні із Scapy та Flask	Twisted та Scapy	Raw sockets та Custom	AsyncIO та aiohttp
Розмір встановлення (МБ)	45	1200	25	80
Швидкість HTTP (req/sec)	1000–2000	3000–5000	5000+	8000–12000
Підтримка протоколів	IPv4/IPv6/TCP/UDP	IPv4/ IPv6/ TCP/ UDP/SSL	Всі	IPv4/IPv6/ TCP/UDP
Швидкість парсингу пакетів (Mbps)	100–200	100–200	50–100	80–150
GitHub зірок (тис.)	28+	25+	6	5.4+

Отже, комбінація Socket, Scapy та Flask є найоптимальнішим рішенням для реалізації архітектури цільового програмного засобу. Зокрема, бібліотека Socket забезпечує доступ до низькорівневих мережевих функцій, необхідних для створення honeypot-елементів. Scapy надає широкі можливості для конструювання, відправлення та аналізу мережевих пакетів, що критично важливо

для виявлення потенційно шкідливої активності. Flask, у свою чергу, дозволяє швидко розгортати веб-інтерфейс системи з високим рівнем кастомізації та простотою інтеграції з іншими модулями. Комбінація цих інструментів не лише технічно ефективна, але й добре підтримується спільнотою, має розгорнуту документацію та дозволяє реалізувати програмний засіб у стислий термін без суттєвих витрат на налаштування чи оптимізацію.

Наступним етапом стало визначення найбільш доцільного середовища розробки, яке має забезпечити зручність написання та налагодження коду, підтримку роботи з мережевими бібліотеками, інтеграцію із Git та тестування створених компонентів. У межах аналізу було розглянуто такі інструменти як Visual Studio Code, PyCharm, Sublime Text, Vim/Emacs та Jupyter Notebook. У таблиці 3.3 наведено порівняння середовищ розробки програмного засобу. Основними критеріями порівняння стали інсталяційний розмір, швидкість запуску, споживання оперативної пам’яті, кількість доступних розширень, популярність на GitHub та активність обговорення на StackOverflow [45].

Таблиця 3.3 – Порівняльння характеристик популярних середовищ розробки

Критерій	Visual Studio Code	PyCharm	Sublime Text	Vim / Emacs	Jupyter Notebook
Розмір інсталяції (МБ)	~320	~498	~38 (3–15)	~10–50	~487 (3000–6000)*
Час запуску (мс)	1200–2000	8000–15000	300–800	50–200	3000–6000
Споживання RAM (МБ)	200–500	800–1500	40–120	10–50	300–800
Кількість розширень	45 000+	4 500+	5 700+	20 000+	8 000+
GitHub зірок (тис.)	151	35 / 12 (Community/Pro)	11.2	~18 / ~27	15.3
StackOverflow питань (тис.)	28.5	45.2	12.8	180 / 45 (Vim/Emacs)	15.3

Проведене порівняння демонструє, що Visual Studio Code є найбільш збалансованим середовищем розробки для реалізації поставлених завдань. Воно поєднує у собі високий рівень продуктивності, велику кількість розширень, офіційну підтримку Python та інструменти для налагодження, роботу з Git і

тестування. Крім того, VS Code відзначається помірним споживанням ресурсів, наявністю інтегрованого терміналу та зручним інтерфейсом. Його відкритий вихідний код та активна спільнота сприяють безперервному розвитку й адаптації під нові виклики. Таким чином, саме Visual Studio Code доцільно використовувати як основне середовище розробки програмного засобу.

Підсумовуючи проведений аналіз, для реалізації програмного засобу раннього виявлення підозрілої активності у локальних Wi-Fi мережах було обрано Python як основну мову програмування у поєднанні з бібліотеками Socket, Scapy та фреймворком Flask, а Visual Studio Code - як середовище розробки. Даний технологічний стек забезпечує оптимальний баланс між швидкістю розробки, функціональними можливостями та гнучкістю архітектури, що є критично важливим для успішної реалізації поставлених завдань у рамках кваліфікаційної роботи.

3.2 Опис функціональностей та особливостей програмної реалізації розробленої системи раннього виявлення підозрілої активності у WI–FI мережі

Для реалізації запропонованого підходу раннього виявлення підозрілої активності в локальних Wi-Fi мережах за допомогою цифрових пасток було розроблено програмний засіб з використанням мови програмування Python та відповідних бібліотек для роботи з мережевими протоколами DNS, HTTP та ARP. Код нижче відображає імпорт необхідних бібліотек та ініціалізацію конфігурації системи:

```
pythonimport threading
import yaml
from honeypots import dns, http, arp
from sniffers import analyzer
def main():
   with open("config/settings.yaml", "r") as f:
      settings = yaml.safe_load(f)
```

Цей фрагмент забезпечує підключення модулів для роботи з потоками, завантаження налаштувань та імпорт компонентів системи цифрових пасток. Код

нижче відображає створення та запуск окремих потоків для кожного модуля системи:

```
python    dns_thread = threading.Thread(target=dns.start, args=(settings,), daemon=True)
   http_thread = threading.Thread(target=http.start, args=(settings,), daemon=True)
   arp_thread = threading.Thread(target=arp.start, args=(settings,), daemon=True)
   analyzer_thread = threading.Thread(target=analyzer.start, args=(settings,), daemon=True)
   dns_thread.start()
   http_thread.start()
   arp_thread.start()
   analyzer_thread.start()
```

Використання багатопоточності дозволяє всім компонентам працювати одночасно, а демон-потоки забезпечують коректне завершення при зупинці програми. Код нижче відображає основний цикл програми та обробку сигналу переривання:

```
python    print("\n[MAIN] Honeypots are running... Press Ctrl+C to stop.")
   try:
      while True:
         pass
   except KeyboardInterrupt:
      print("\n[MAIN] Stopping honeypots...")
if __name__ == "__main__":
   main()
```

Нескінченний цикл підтримує роботу головного потоку, а обробка переривання забезпечує коректне завершення роботи системи. Код нижче відображає структуру стилів та заголовка веб-інтерфейсу:

```
html<title>Login</title>
<style> body { position: absolute;  top: 30%;   left: 50%; transform: translate(-50%, -50%); }
   input {display: flex; justify-content: flex-end;}
</style>
</head>
```

Стилі забезпечують центрування форми входу на сторінці та вирівнювання елементів вводу. Код нижче відображає форму автентифікації користувача:

```
html<body>  <h2>Login</h2>
   <form method="POST">
      <div>Username: <input type="text" name="username" /><br></div>
      <div>Password: <input type="password" name="password" /><br></div>
      <input type="submit" value="Login" />  </form>
```

Форма використовує POST-метод для безпечної передачі облікових даних на сервер. Код нижче відображає заголовок та початок циклу для відображення модулів:

```
html<h2>Modules:</h2>
{% for mod, running in status.items() %}
  <div>  <b>{{ mod.upper() }}</b> — Status: {{ "Running" if running else "Stopped" }}
```

Шаблонізатор динамічно генерує список модулів з відображенням їх поточного стану. Код нижче відображає кнопки управління для кожного модуля:

```
<form method="post" style="display:inline;">
        <input type="hidden" name="module" value="{{ mod }}" />
        <button name="action" value="start" type="submit">Start</button>
      </form>
      <form method="post" style="display:inline;">
        <input type="hidden" name="module" value="{{ mod }}" />
        <button name="action" value="stop" type="submit">Stop</button>
      </form>
  </div>
```

Окремі форми з прихованими полями дозволяють керувати станом кожного модуля незалежно. Код нижче відображає контейнер для відображення логів системи:

```
html<h2>Logs:</h2>
<pre id="log-box" style="background:#f0f0f0; padding:10px; height:300px; overflow:auto;">
  Loading logs...
</pre>
```

Текстовий блок з прокруткою забезпечує зручне відображення великих обсягів логів. Код нижче відображає функцію отримання логів з сервера:

```
<script>
  function fetchLogs() {
    fetch("/logs")
      .then(response => response.json())
      .then(data => { const logBox = document.getElementById("log-box");
        logBox.textContent = data.logs.join("");
        logBox.scrollTop = logBox.scrollHeight;
      })
```

Асинхронний запит отримує дані з сервера та оновлює вміст контейнера з автоматичною прокруткою. Код нижче відображає налаштування автоматичного оновлення:

```
.catch(error => { console.error("Failed to load logs:", error);
}) } setInterval(fetchLogs, 100);  fetchLogs();
</script>
```

Інтервал у 100 мілісекунд забезпечує оновлення логів у реальному часі з обробкою помилок.

Код нижче відображає початок форми конфігурації та налаштування DNS-модуля:

```
html<form method="post">
  <h2>DNS</h2>  <label>IP: <input type="text" name="dns_ip" value="{{ config['dns_ip'] }}"></label>
  <label>Port: <input type="number" name="dns_port" value="{{ config['dns_port'] }}"></label>
                <label>Domains (comma-separated):<br> <input type="text" name="dns_domains"
value="{{ config['dns_domains'] | join(', ') }}">  </label>
```

Поля дозволяють налаштувати IP-адресу, порт та список доменів для DNS-модуля. Код нижче відображає налаштування HTTP та ARP модулів:

```
html  <h2>HTTP</h2>
  <label>IP: <input type="text" name="http_ip" value="{{ config['http_ip'] }}"></label>
  <label>Port: <input type="number" name="http_port" value="{{ config['http_port'] }}"></label>
  <label>Suspicious paths (comma-separated):<br>
    <input type="text" name="http_paths" value="{{ config['http_paths'] | join(', ') }}">
  </label>  <h2>ARP</h2>
  <label>Interface: <input type="text" name="arp_interface" value="{{ config['arp_interface'] }}"></label>
  <label>Fake MAC: <input type="text" name="arp_fake_mac" value="{{ config['arp_fake_mac'] }}"></label>
  <label>Target IP: <input type="text" name="arp_ip" value="{{ config['arp_ip'] }}"></label>
```

HTTP-модуль налаштовується через мережеві параметри та підозрілі шляхи, а ARP-модуль через інтерфейс та підроблену MAC-адресу. Код нижче відображає налаштування модуля перехоплення та збереження конфігурації:

```
html  <h2>Sniffer</h2>
  <label>Interface: <input type="text" name="sniffer_interface" value="{{ config['sniffer_interface'] }}"></label>
  <label>ARP IP: <input type="text" name="sniffer_arp_ip" value="{{ config['sniffer_arp_ip'] }}"></label>
  <label>Watch Domains (comma-separated):<br>
<input type="text" name="sniffer_domains"
value="{{ config['sniffer_domains'] | join(', ') }}">
</label><h2>Logs</h2><label>Log file path:
<input type="text" name="log_file" value="{{ config['log_file'] }}"></label>
<br>  <button type="submit"> Save</button> </form>
```

Модуль перехоплення налаштовується через мережевий інтерфейс та список доменів, а кнопка зберігає всі зміни конфігурації.

Веб-сервер системи реалізований з використанням фреймворку Flask та забезпечує управління всіма компонентами через веб-інтерфейс. Код нижче відображає ініціалізацію Flask-додатку та завантаження конфігурації:

```
from flask import Flask, render_template, request, redirect, url_for, session
import threading
import yaml
from werkzeug.security import check_password_hash
app = Flask(__name__)
app.secret_key = "supersecretkey"
with open("config/settings.yaml", "r") as f:
```

```
    settings = yaml.safe_load(f)
with open("config/users.yaml") as f: users = yaml.safe_load(f)["users"]
```

Flask-додаток ініціалізується з секретним ключем для сесій, а конфігурація завантажується з YAML-файлів при старті сервера. Код нижче відображає структуру управління потоками та подіями зупинки:

```
pythonthreads = {
    "dns": None,
    "http": None,
    "arp": None,
    "analyzer": None,
}
stop_events = {
    "dns": threading.Event(),
    "http": threading.Event(),
    "arp": threading.Event(),
    "analyzer": threading.Event(),
}
```

Словники зберігають посилання на потоки та події зупинки для кожного модуля, що дозволяє керувати їх життєвим циклом. Код нижче відображає функцію запуску модуля у відокремленому потоці:

```
pythondef run_module(module_name):
    event = stop_events[module_name]
    if module_name == "dns":
        dns.start(settings, event)
    elif module_name == "http":
        http.start(settings, event)
    elif module_name == "arp":
        arp.start(settings, event)
    elif module_name == "analyzer":
        analyzer.start(settings, event)
```

Функція визначає тип модуля та запускає відповідний компонент з передачею конфігурації та події зупинки. Код нижче відображає функції управління життєвим циклом модулів:

```
pythondef start_module(module_name):
    if threads[module_name] is None or not threads[module_name].is_alive():
        thread = threading.Thread(target=run_module, args=(module_name,), daemon=True)
        threads[module_name] = thread
        thread.start() def stop_module(module_name):
    event = stop_events[module_name]
    if threads[module_name] and threads[module_name].is_alive():
        event.set()
        threads[module_name].join(timeout=2)
        threads[module_name] = None
        stop_events[module_name].clear()
```

Функції забезпечують безпечний запуск та зупинку модулів з перевіркою стану потоків та коректним очищенням ресурсів.

Система автентифікації користувачів реалізована через перевірку облікових даних з хешованими паролями. Код нижче відображає маршрут автентифікації:

```
from werkzeug.security import generate_password_hash, check_password_hash

python@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        if username in users and check_password_hash(users[username], password):
            session["logged_in"] = True
            return redirect(url_for("index"))
        return "Invalid credentials", 401
    return render_template("login.html")
```

Маршрут перевіряє облікові дані користувача через хешовану перевірку паролів та встановлює сесію при успішній автентифікації. Головна сторінка системи забезпечує управління модулями та відображення їх стану. Код нижче відображає основний маршрут:

```
python@app.route("/", methods=["GET", "POST"])
def index():
    if not session.get("logged_in"): return redirect(url_for("login"))
    if request.method == "POST":
        action = request.form.get("action")
        module = request.form.get("module")
        if action == "start": start_module(module)
        elif action == "stop": stop_module(module)
        return redirect(url_for("index"))
```

Маршрут перевіряє автентифікацію користувача та обробляє команди запуску або зупинки модулів через POST-запити. DNS-модуль реалізований як UDP-сервер, який перехоплює та аналізує DNS-запити на підозрілі домени. Код нижче відображає функцію виявлення підозрілих доменів:

```
pythondef detect_suspicious_domain(domain, suspicious_keywords):
    for keyword in suspicious_keywords:
        if keyword in domain.lower(): return True
    return False
def extract_domain(data): domain_parts = [] i = 12
    length = data[i]  while length != 0:
        i += 1 domain_parts.append(data[i:i+length].decode())
        i += length length = data[i]
 return ".".join(domain_parts)
```

Функції аналізують доменні імена на наявність підозрілих ключових слів та витягують домен з DNS-пакета. ARP-модуль реалізує цифрову пастку через відправку фальшивих ARP-відповідей для перехоплення мережевого трафіку. Код нижче відображає функцію створення фальшивої ARP-відповіді:

```
pythondef arp_reply(target_ip, target_mac, arp_ip, fake_mac, interface):
   ether = Ether(dst=target_mac) arp = ARP( op=2,                # is-at (ARP Reply)
      pdst=target_ip,        # IP призначення
      hwdst=target_mac,      # MAC призначення
      psrc=arp_ip,           # Фальшивий IP (наш honeypot)
      hwsrc=fake_mac         # Фальшивий MAC (наш honeypot))
   packet = ether / arp sendp(packet, iface=interface, verbose=False)
```

Функція створює фальшивий ARP-пакет з підробленою MAC-адресою та відправляє його в мережу для перенаправлення трафіку. Код нижче відображає обробник ARP-запитів та головний цикл модуля:

```
def arp_request_callback(packet, arp_ip, fake_mac, interface):
   if packet.haslayer(ARP) and packet[ARP].op == 1:
      target_ip = packet[ARP].pdst
    target_mac = packet[ARP].hwsrc
      log_event(f"[{datetime.now()}][ARP Honeypot] Received ARP request for {target_ip} from {target_mac}")
      arp_reply(target_ip, target_mac, arp_ip, fake_mac, interface)
```

Функція перевіряє тип ARP-пакета та витягує інформацію про відправника для подальшої генерації фальшивої відповіді. Код нижче відображає головний цикл ARP-модуля:

```
def start(settings, stop_event): interface = settings['arp_interface']
   fake_mac = settings['arp_fake_mac'] arp_ip = settings['arp_ip']
while not stop_event.is_set():  try: sniff(iface=interface, filter="arp", prn=lambda pkt:
arp_request_callback(pkt, arp_ip, fake_mac, interface), store=0, timeout=1) except Exception as e:
log_event(f"[{datetime.now()}][ARP Honeypot] Error: {e}") break
```

Головний цикл використовує бібліотеку Scapy для перехоплення ARP-трафіку на вказаному інтерфейсі з можливістю зупинки через подію. Таким чином код функціоналу програмного засобу забезпечує комплексний підхід раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі з використанням цифрових пасток на рівні хибних DNS, HTTP та ARP протоколів.

3.3 Тестування працездатності реалізованої системи раннього виявлення підозрілої активності у WI-FI мережі

У рамках перевірки функціональності реалізованого засобу було здійснено низку діагностичних дій для аналізу мережевої активності на рівні клієнтської машини. Зокрема, увага приділялася роботі системи у контексті обробки DNS-запитів, ICMP-взаємодії та доступності сервісів у локальній підмережі. На першому етапі було зафіксовано основні мережеві параметри адаптера Wi-Fi, зокрема IPv4-адресу, маску підмережі та шлюз за замовчуванням (рис. 3.1).

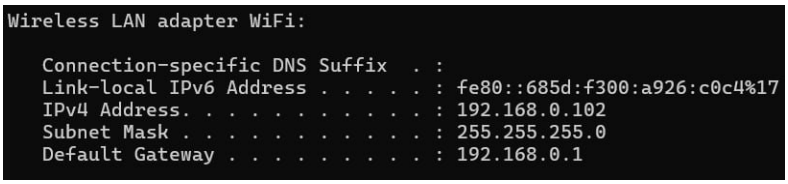


Рисунок 3.1 – Фіксація основних мережевих параметрів Wi-Fi адаптера.

Основними мережевими параметри адаптера є:

- IPv4-адреса (192.168.0.102;
- маска підмережі (255.255.255.0;
- шлюз за замовчуванням (192.168.0.1).

Це дозволило визначити локальну конфігурацію, у межах якої проводилося тестування, та переконатися у наявності активного з’єднання. Наступним кроком було здійснено серію ICMP-запитів до ресурсу `google.com` (рис. 3.2).

Рисунок 3.2 – Серію ICMP-запитів до зовнішніх ресурсів.

Пінг-запити до `google.com` пройшли успішно з середнім часом відповіді 46.7 мс, що свідчить про стабільне з’єднання з мережею Інтернет.

Для перевірки коректності обробки DNS-запитів у локальній мережі було використано інструмент `dig` з прямим зверненням до локального вузла за IP-адресою 192.168.0.102 через порт 55335 для доменного імені `adminpanel-site.com`. (рис. 3.3).

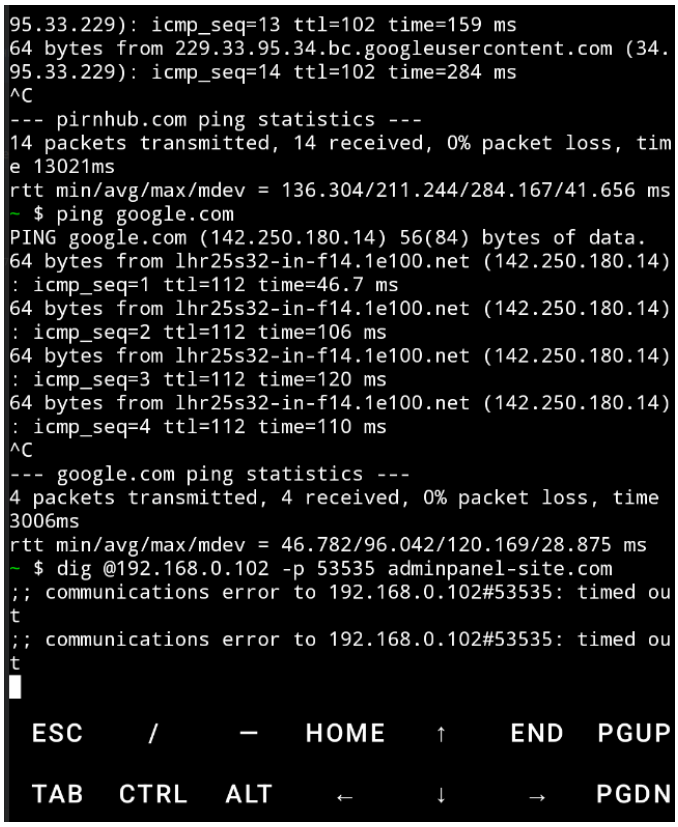


Рисунок 3.3 – Перевірка коректності обробки DNS-запитів у локальній мережі

Результатом виконання запиту стала помилка `communications error`, яка вказує на неможливість встановлення з’єднання з цим DNS-сервером або відсутність відповіді на нестандартному порту. Це може свідчити про коректне блокування підозрілої активності з боку реалізованої системи або про відсутність належного DNS-сервісу на зазначеному вузлі.

Уся зазначена вище інформація щодо мережевих параметрів, результатів ICMP-запитів та DNS-взаємодії була автоматично зафіксована та виведена у головному вікні розробленого додатку у вигляді журналу подій (Log). Це забезпечує зручність моніторингу та дозволяє оперативно виявляти підозрілу активність у Wi-Fi мережі без необхідності залучення сторонніх засобів діагностики. (рис. 3.4).

Logs:

```
[2025-05-16 16:34:15.160393][ARP Honeypot] Received ARP request for 192.168.0.150 from d0:c5:d3:57:22:13
[2025-05-16 16:34:15.162983][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:34:16.184308][ARP Honeypot] Received ARP request for 192.168.0.150 from d0:c5:d3:57:22:13
[2025-05-16 16:34:16.186384][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:34:17.823652][ARP Honeypot] Received ARP request for 192.168.4.59 from d2:19:c4:30:fd:34
[2025-05-16 16:34:17.827403][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:34:47.071265][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:34:47.073932][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:35:28.033470][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:35:28.039900][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-17 14:05:40.202415][DNS] Suspicious query from 192.168.4.16:48253 — domain: adminpanel-site.com
[2025-05-17 14:05:45.050830][DNS] Suspicious query from 192.168.4.16:37939 — domain: adminpanel-site.com
[2025-05-17 14:05:50.053849][DNS] Suspicious query from 192.168.4.16:50808 — domain: adminpanel-site.com
[2025-05-20 22:05:47.275836][HTTP] Suspicious GET request from 192.168.0.101:41418 — Path: /admin
[2025-05-20 22:05:47.277417][HTTP] GET request from 192.168.0.101:41424 — Path: /
[2025-05-20 22:05:47.278847][HTTP] GET request from 192.168.0.101:49080 — Path: /
[2025-05-20 22:05:47.348849][HTTP] GET request from 192.168.0.101:49096 — Path: /favicon.ico
[2025-05-26 18:33:01.532911][DNS] Suspicious query from 192.168.0.100:50292 — domain: adminpanel-site.com
[2025-05-26 18:33:06.533152][DNS] Suspicious query from 192.168.0.100:57808 — domain: adminpanel-site.com
[2025-05-26 18:33:11.536243][DNS] Suspicious query from 192.168.0.100:40672 — domain: adminpanel-site.com
```

Рисунок 3.4 – Виведена інформація у вигляді журналу подій.

На основі проведених тестувань можна зробити висновок, що модуль аналізу DNS-запитів функціонує коректно. Програмний засіб успішно фіксує спроби звернення до DNS-серверів, а також виявляє відсутність відповіді або тайм-аут, що свідчить про його здатність ідентифікувати потенційно підозрілу активність. Таким чином, розроблений інструмент виконує поставлене завдання на етапі контролю DNS-комунікацій.

Продовжуючи тестування функціональності розробленого програмного іршенняф, було проведено перевірку роботи модуля HTTP-запитів. Нижче показано активацію HTTP-модуля через панель керування в головному вікні програмного засобу (рис. 3.5).

Modules:

DNS — Status: Stopped Start Stop
HTTP — Status: Running Start Stop
ARP — Status: Stopped Start Stop
ANALYZER — Status: Stopped Start Stop

Рисунок 3.5 – Активація модуля HTTP-запитів

Для перевірки ефективності виявлення підозрілої HTTP-активності було здійснено спробу доступу до локального IP-адреса 192.168.0.102 через порт 8080 з використанням HTTP-протоколу (рис. 3.6).

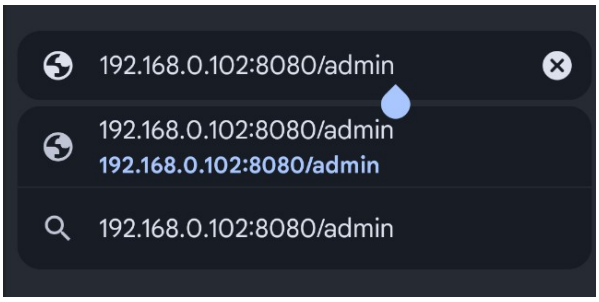


Рисунок 3.6 – Зображення переходу на адресу `192.168.0.102:8080/admin`

Після переходу за вказаною адресою система успішно перехопила HTTP-запит та відобразила хибну сторінку "Welcome to Admin Panel" (рис. 3.7), що є частиною honeypot-механізму.

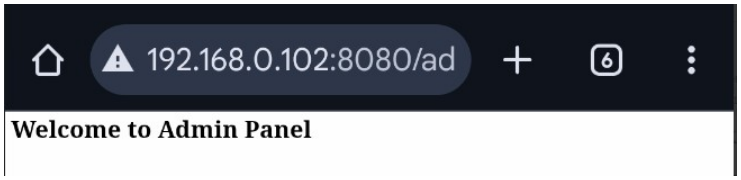


Рисунок 3.7 – Відображення хибної сторінки "Welcome to Admin Panel" після успішного перехоплення HTTP-запиту розробленим програмним засобом

Ця функціональність дозволяє імітувати наявність адміністративного інтерфейсу, приваблюючи потенційних зловмисників та одночасно фіксуючи їхню активність. Усі описані дії HTTP-модуля були автоматично зареєстровані у журналі подій системи (рис. 3.8).

Logs:

```
[2025-05-16 16:34:16.184308][ARP Honeypot] Received ARP request for 192.168.0.150 from d0:c5:d3:57:22:13
[2025-05-16 16:34:16.186384][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent to d0:c5:d3:57:22:13)
[2025-05-16 16:34:17.823652][ARP Honeypot] Received ARP request for 192.168.4.59 from d2:19:c4:30:fd:34
[2025-05-16 16:34:17.827403][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent to d2:19:c4:30:fd:34)
[2025-05-16 16:34:47.071265][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:34:47.073932][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent to 08:55:31:8b:a3:28)
[2025-05-16 16:35:28.033470][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:35:28.039900][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent to 08:55:31:8b:a3:28)
[2025-05-17 14:05:40.202415][DNS] Suspicious query from 192.168.4.16:48253 – domain: adminpanel-site.com
[2025-05-17 14:05:45.050830][DNS] Suspicious query from 192.168.4.16:37939 – domain: adminpanel-site.com
[2025-05-17 14:05:50.053849][DNS] Suspicious query from 192.168.4.16:50808 – domain: adminpanel-site.com
[2025-05-20 22:05:47.275836][HTTP] Suspicious GET request from 192.168.0.101:41418 – Path: /admin
[2025-05-20 22:05:47.277417][HTTP] GET request from 192.168.0.101:41424 – Path: /
[2025-05-20 22:05:47.278847][HTTP] GET request from 192.168.0.101:49080 – Path: /
[2025-05-20 22:05:47.348849][HTTP] GET request from 192.168.0.101:49096 – Path: /favicon.ico
[2025-05-26 18:33:01.532911][DNS] Suspicious query from 192.168.0.100:50292 – domain: adminpanel-site.com
[2025-05-26 18:33:06.533152][DNS] Suspicious query from 192.168.0.100:57808 – domain: adminpanel-site.com
[2025-05-26 18:33:11.536243][DNS] Suspicious query from 192.168.0.100:40672 – domain: adminpanel-site.com
[2025-05-26 18:35:12.087421][HTTP] Suspicious GET request from 192.168.0.100:32838 – Path: /admin
[2025-05-26 18:35:12.215061][HTTP] GET request from 192.168.0.100:32848 – Path: /favicon.ico
```

Рисунок 3.8 – Зареєстровані дії HTTP-модуля у журналі подій

У логах чітко відображені HTTP-запити з деталізацією часу, IP-адреси джерела, методу запиту та шляху доступу.

Зокрема, зафіксовано GET-запити до кореневого каталогу "/", що є типовою поведінкою при первинному скануванні веб-ресурсів. Результати тестування HTTP-модуля підтверджують його ефективність у виявленні та логуванні підозрілої веб-активності. Система коректно перехоплює HTTP-запити, генерує хибні відповіді для приманювання зловмисників та детально документує всі спроби несанкціонованого доступу.

Завершальним етапом тестування стала перевірка функціональності ARP-модуля, який відповідає за моніторинг активності на канальному рівні мережевої моделі OSI. ARP-протокол відіграє критичну роль у виявленні підозрілої активності, оскільки більшість інструментів мережевого сканування та сніфінгу використовують ARP-запити для виявлення активних хостів у локальній мережі. Для тестування ARP-модуля було здійснено налаштування параметрів honeypot-механізму через інтерфейс системи (рис. 3.9).

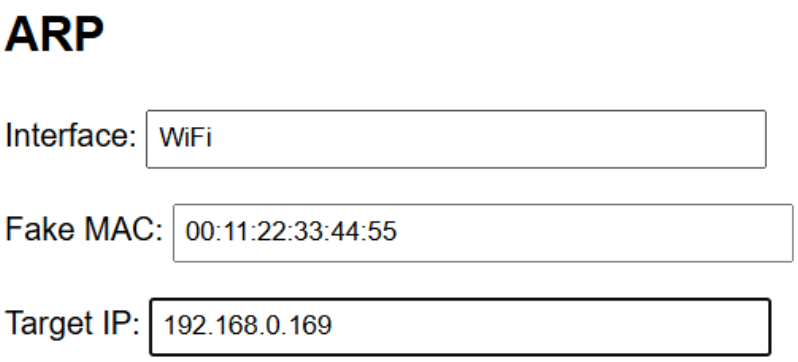


Рисунок 3.9 – Конфігурація параметрів honeypot-механізму.

Конфігурація включала:

- Interface: WiFi – визначення мережевого інтерфейсу для моніторингу;
- Fake MAC: 00:11:22:33:44:55 – підроблена MAC-адреса для імітації хибного хоста;
- Target IP: 192.168.0.169 – цільова IP-адреса для створення ARP-пастки.

Після активації ARP-модуля було проведено серію тестових дій для імітації типової поведінки сканерів мережі (рис. 3.9).

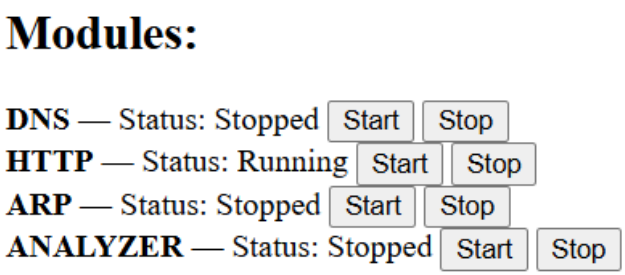


Рисунок 3.10 – Активація ARP-модуля.

Нижче показано результати виконання різних мережевих команд, включаючи ICMP-запити до зовнішніх ресурсів та локальних адрес (рис. 3.10).

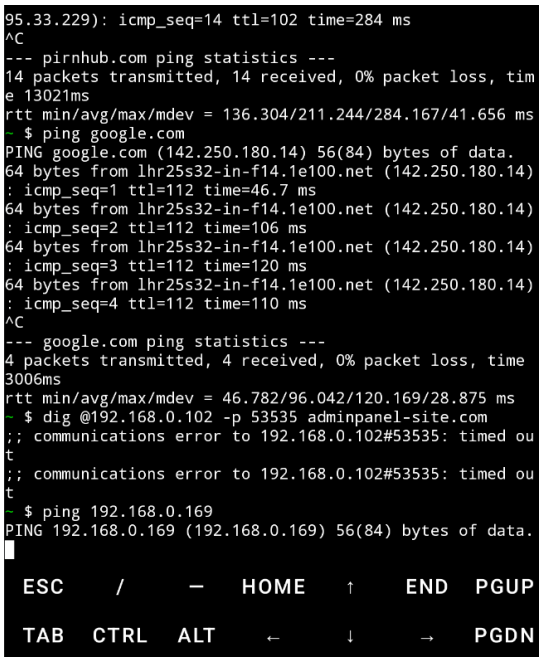


Рисунок 3.10 – Зображення виконання різних мережевих команд

Особливу увагу привертає команда ping 192.168.0.169, яка генерує запит до IP-адреси, налаштованої як ARP-пастка. Результат виконання команди ping 192.168.0.169 показав успішну відповідь з 56 байтами даних, що свідчить про коректну роботу ARP-honeypot механізму. Система успішно імітує присутність хоста за вказаною IP-адресою, використовуючи підроблену MAC-адресу, що є ключовою функціональністю для виявлення мережевого сканування. Усі ARP-взаємодії були детально зафіксовані у журналі подій системи (рис. 3.11).

Logs:

```
[2025-05-16 16:34:17.823652][ARP Honeypot] Received ARP request for 192.168.4.59 from d2:19:c4:30:fd:34
[2025-05-16 16:34:17.827403][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent to d2:19:c4:30:fd:34)
[2025-05-16 16:34:47.071265][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:34:47.073932][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent to 08:55:31:8b:a3:28)
[2025-05-16 16:35:28.033470][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:35:28.039900][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent to 08:55:31:8b:a3:28)
[2025-05-17 14:05:40.202415][DNS] Suspicious query from 192.168.4.16:48253 – domain: adminpanel-site.com
[2025-05-17 14:05:45.050830][DNS] Suspicious query from 192.168.4.16:37939 – domain: adminpanel-site.com
[2025-05-17 14:05:50.053849][DNS] Suspicious query from 192.168.4.16:50808 – domain: adminpanel-site.com
[2025-05-20 22:05:47.275836][HTTP] Suspicious GET request from 192.168.0.101:41418 – Path: /admin
[2025-05-20 22:05:47.277417][HTTP] GET request from 192.168.0.101:41424 – Path: /
[2025-05-20 22:05:47.278847][HTTP] GET request from 192.168.0.101:49080 – Path: /
[2025-05-20 22:05:47.348849][HTTP] GET request from 192.168.0.101:49096 – Path: /favicon.ico
[2025-05-26 18:33:01.532911][DNS] Suspicious query from 192.168.0.100:50292 – domain: adminpanel-site.com
[2025-05-26 18:33:06.533152][DNS] Suspicious query from 192.168.0.100:57808 – domain: adminpanel-site.com
[2025-05-26 18:33:11.536243][DNS] Suspicious query from 192.168.0.100:40672 – domain: adminpanel-site.com
[2025-05-26 18:35:12.087421][HTTP] Suspicious GET request from 192.168.0.100:32838 – Path: /admin
[2025-05-26 18:35:12.215061][HTTP] GET request from 192.168.0.100:32848 – Path: /favicon.ico
[2025-05-26 18:38:17.234896][ARP Honeypot] Received ARP request for 192.168.0.169 from c2:ee:6c:f8:a6:9c
[2025-05-26 18:38:17.238731][ARP Honeypot] Sent fake ARP reply: 192.168.0.169 is at 00:11:22:33:44:55 (sent to c2:ee:6c:f8:a6:9c)
```

Рисунок 3.11 – ARP-взаємодії детально зафіксовані у журналі подій розробленого програмного засобу

У логах чітко відображено:

- ARP Honeypot записи, що фіксують отримання та відправлення ARP-запитів;
- часові мітки кожної операції для точного відстеження активності;
- IP та MAC адреси учасників ARP-обміну;
- детальну інформацію про напрямок трафіку (sent/received).

Аналіз логів показує, що система коректно перехоплює ARP-запити для цільової IP-адреси 192.168.0.169 та генерує відповідні ARP-відповіді з використанням налаштованої підробленої MAC-адреси. Це дозволяє ефективно виявляти спроби мережевого сканування, оскільки більшість інструментів

розвідки використовують ARP-запити для виявлення активних хостів у локальній мережі.

Додатковим етапом тестування стала перевірка здатності системи виявляти активне сканування мережі за допомогою популярного інструменту Nmap. Як показано нижче (рис.3.12), модуль Sniffer успішно зафіксував інтенсивну сканувальну активність, що проявилася у вигляді серії "Possible Nmap SYN Scan" записів у журналі подій. Система детально документувала кожну спробу сканування з точними часовими мітками, показуючи, що протягом декількох секунд (з 20:39:04 до 20:39:12) було здійснено множинні SYN-сканування з IP-адреси 192.168.0.103.

```
[2025-05-26 20:39:04.672540][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.537967][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.544622][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.551286][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.557528][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.701668][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.710952][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.758542][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.772802][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.914038][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.974146][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.981469][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.024115][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.141697][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.156226][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.347855][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.598814][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.953769][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:08.780636][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:12.927695][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
```

Рисунок 3.12 – Записи у журналі подій розробленого програмного засобу при скануванні мережі інструментом Nmap

Такий тип активності є характерним для використання Nmap з опцією SYN-сканування, яке є одним з найпоширеніших методів розвідки мережі для виявлення відкритих портів та доступних сервісів. Здатність системи розпізнавати та фіксувати подібну активність підтверджує її ефективність у виявленні реальних інструментів мережевої розвідки, що робить розроблений засіб особливо цінним для забезпечення безпеки локальних мереж. Детальність логування дозволяє адміністраторам не лише виявити факт сканування, але й проаналізувати його

інтенсивність, тривалість та джерело, що є критично важливою інформацією для оцінки рівня загрози та прийняття відповідних заходів протидії.

Результати комплексного тестування всіх модулів системи підтверджують її ефективність у вирішенні поставлених завдань. Розроблений програмний засіб демонструє стабільну роботу DNS, HTTP та ARP honeypot-механізмів, забезпечуючи раннє виявлення підозрілої активності на різних рівнях мережевого стеку.

3.4 Інструкція для користувача програмного засобу

Розроблений програмний засіб для раннього виявлення підозрілої активності у локальній мережі має простий та інтуїтивно зрозумілий інтерфейс користувача.

Інтерфейс користувача системи реалізовано у вигляді веб-застосунку з інтуїтивно зрозумілою панеллю управління, яка забезпечує централізований доступ до всіх функцій системи. Інтерфейс включає механізми автентифікації користувачів, панель керування модулями цифрових пасток з можливістю індивідуального запуску та зупинки кожного компонента, детальну панель конфігурації з налаштуваннями для всіх модулів системи та розділ журналювання для перегляду та аналізу виявлених подій.

Для початку роботи з системою необхідно пройти процедуру авторизації, після чого користувач отримує доступ до основної панелі керування цифровими пастками. Процес авторизації користувача починається з відкриття головної сторінки програмного засобу, де знаходиться форма входу до системи (рис. 3.13).

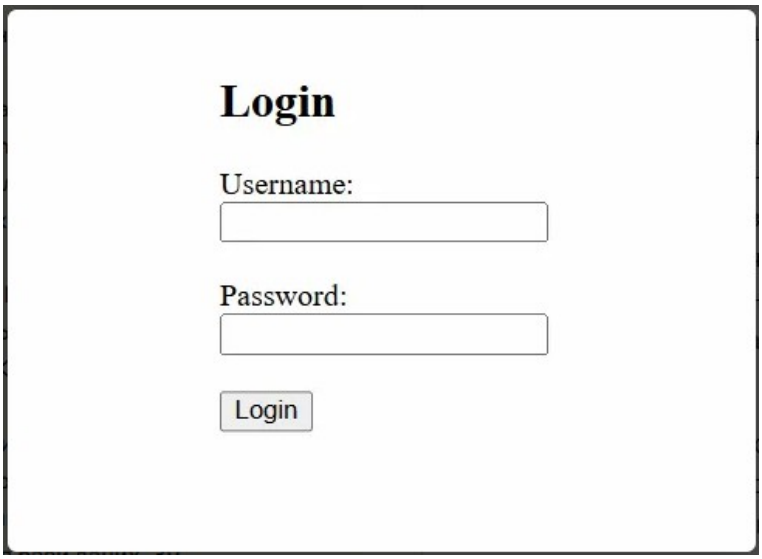


Рисунок 3.13 – Форма авторизації користувача в системі раннього виявлення підозрілої активності.

На екрані авторизації присутні два обов'язкові поля для заповнення: поле введення імені користувача та поле введення пароля. Користувач повинен ввести свої облікові дані у відповідні текстові поля, після чого натиснути кнопку входу для підтвердження автентифікації

Система перевіряє введені дані та у разі успішної авторизації перенаправляє користувача до головної панелі керування. Після успішного входу до системи користувач потрапляє на головну сторінку панелі керування цифровими пастками. У верхній частині інтерфейсу розміщується заголовок панелі керування, який чітко вказує на призначення даного програмного модуля. Справа від основного заголовка знаходяться дві важливі кнопки управління системою: кнопка налаштувань «settings» та кнопка виходу з облікового запису «logout» (рис. 3.14).

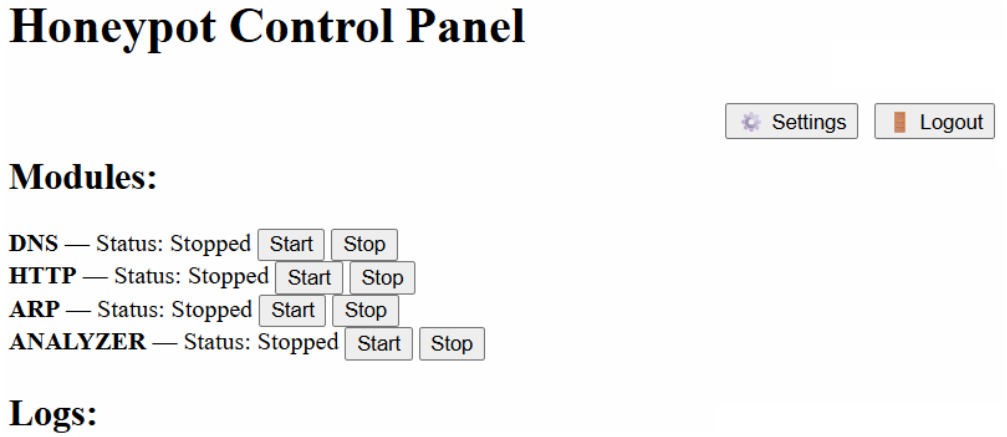


Рисунок 3.14 – Головна панель керування цифровими пастками з основними модулями системи

Основну частину інтерфейсу займає розділ модулів системи, який містить перелік усіх доступних компонентів цифрових пасток. Кожен модуль має свій унікальний ідентифікатор та відображає поточний стан роботи компонента. Система включає чотири основні модулі: модуль обробки запитів системи доменних імен (DNS), модуль обробки протоколу передачі гіпертексту (HTTP), модуль протоколу визначення адрес (ARP) та модуль аналізатора мережевого трафіку (ANALYZER). Для кожного з модулів відображається його поточний статус роботи, який може знаходитися в одному з двох станів: активний або зупинений. Поруч з інформацією про статус розміщені кнопки керування, які дозволяють запускати або зупиняти роботу відповідного модуля. Кнопка запуску активує обраний компонент системи, тоді як кнопка зупинки припиняє його функціонування. Нижче розділу модулів розташовується секція журналів системи, яка призначена для відображення інформації про роботу програмного засобу. Цей розділ дозволяє користувачу відстежувати активність системи та аналізувати виявлені підозрілі дії в мережі. Кнопка налаштувань, розміщена в правій частині інтерфейсу, надає доступ до конфігураційних параметрів системи (рис. 3.15).

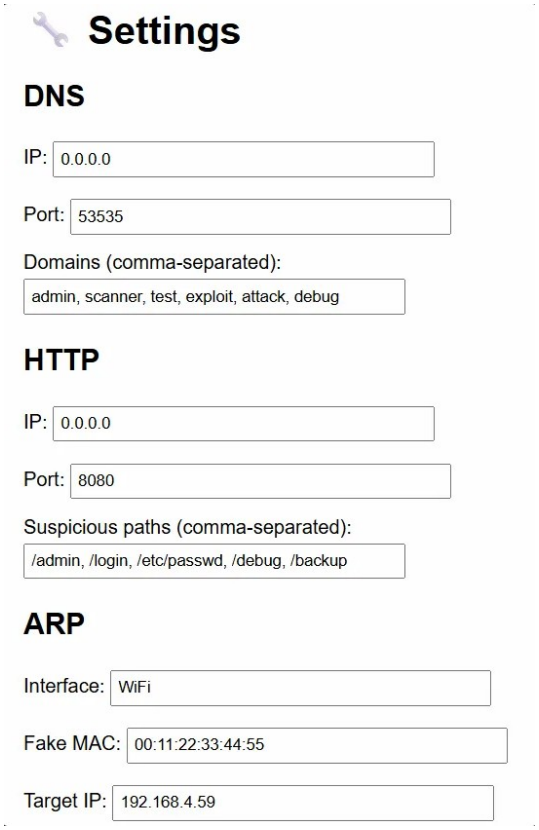


Рисунок 3.15 – Панель налаштувань з конфігураційними параметрами всіх модулів розробленого програмного засобу

При натисканні на неї відкривається детальна панель налаштувань, яка містить конфігураційні параметри для всіх модулів системи. Панель налаштувань складається з декількох основних розділів, кожен з яких відповідає за налаштування окремого компонента системи цифрових пасток. Панель налаштувань модуля системи доменних імен містить три основні параметри конфігурації (рис. 3.16).

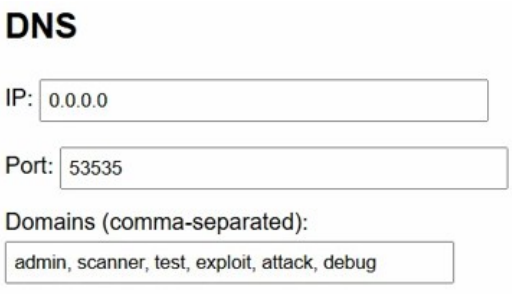


Рисунок 3.16 – Панель налаштувань модуля системи доменних імен

Поле адреси протоколу інтернету дозволяє користувачу вказати мережеву адресу, на якій буде працювати компонент обробки запитів доменних імен.

За замовчуванням це поле містить значення, яке означає прослуховування на всіх доступних мережевих інтерфейсах. Поле номера порту визначає мережевий порт, через який модуль буде приймати вхідні запити. Третє поле містить перелік доменних імен, розділених комами, які система використовуватиме як приманки для виявлення підозрілої активності. До типових значень належать імена, які часто використовуються зловмисниками під час розвідувальної діяльності. Панель конфігурації модуля протоколу передачі гіпертексту включає аналогічні параметри мережевого налаштування (рис. 3.17).

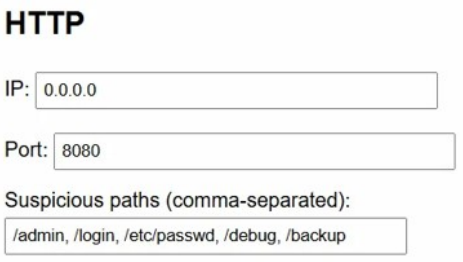


Рисунок 3.17 – Панель налаштування модуля протоколу передачі гіпертексту

Поле адреси протоколу інтернету та поле номера порту виконують ту саму функцію, що й у попередньому модулі, але стосуються веб-трафіку. Особливістю є поле підозрілих шляхів, яке містить перелік веб-адрес, розділених комами. Ці адреси представляють типові шляхи, якими зловмисники намагаються отримати доступ до адміністративних панелей, файлів конфігурації або інших чутливих ресурсів веб-серверів. Панель налаштувань протоколу визначення адрес містить три ключові параметри (рис. 3.18).

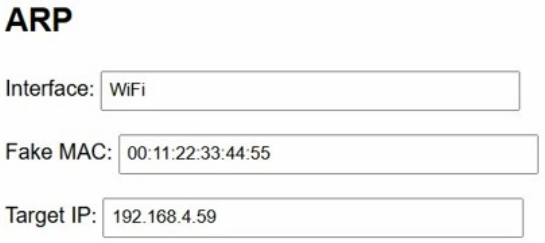


Рисунок 3.18 – Панель налаштувань протоколу визначення адрес

Поле мережевого інтерфейсу вказує, через який фізичний або віртуальний інтерфейс мережі система буде відстежувати та генерувати трафік цього протоколу.

Поле підробленої адреси керування доступом до середовища дозволяє користувачу встановити фіктивну адресу, яку система використовуватиме для генерування хибних відповідей. Поле цільової адреси протоколу інтернету визначає мережеву адресу, яка буде використовуватися як ціль для приманки протоколу визначення адрес. Панель конфігурації компонента перехоплення мережевого трафіку включає параметри для налаштування пасивного моніторингу мережевої активності (рис. 3.19).

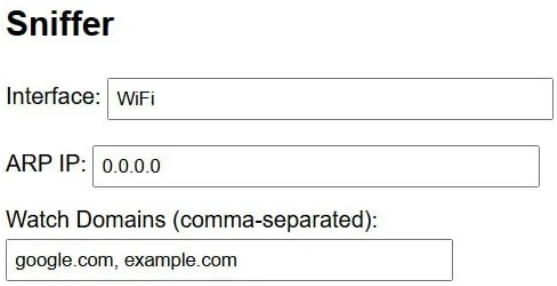


Рисунок 3.19 – Панель конфігурації компонента перехоплення мережевого трафіку

Поле мережевого інтерфейсу визначає, через який інтерфейс система буде здійснювати перехоплення пакетів. Поле адреси протоколу визначення адрес вказує мережеву адресу, з якої очікується трафік для аналізу. Поле доменів для спостереження містить перелік доменних імен, розділених комами, запити до яких система буде відстежувати як потенційно підозрілі. Панель налаштувань журналювання містить єдиний параметр - шлях до файлу журналу подій. Це поле визначає місце збереження всіх записів про роботу системи (рис. 3.20).

Рисунок 3.20 – Панель налаштувань журналювання

Користувач може змінити розташування файлу журналу відповідно до своїх потреб або вимог системного адміністрування. Для збереження внесених змін у конфігурацію система надає кнопку збереження, розташовану в нижній частині панелі налаштувань. Після внесення всіх необхідних змін користувач повинен натиснути цю кнопку для застосування нових параметрів. Поруч з кнопкою збереження знаходиться посилання повернення до головної панелі, яке дозволяє користувачу повернутися до основного інтерфейсу управління без збереження змін.

Після правильного налаштування всіх параметрів та запуску модулів системи розділ журналів починає відображати детальну інформацію про роботу цифрових пасток (рис. 3.21). Кожен запис у журналі містить мітку часу, ідентифікатор модуля, який зафіксував подію, та детальний опис виявленої активності. Записи включають інформацію про отримані запити протоколу визначення адрес, надіслані хибні відповіді, підозрілі запити до доменних імен та спроби доступу до захищених веб-ресурсів.

Рисунок 3.21 – Відображення журналу подій з результатами моніторингу та виявлення підозрілої активності

На основі розробленої інструкції користувача можна зробити висновок, що створений програмний засіб характеризується високим рівнем зручності та інтуїтивності використання. Веб-інтерфейс системи забезпечує централізоване управління всіма компонентами через єдину панель керування з чіткою структурою та логічним розташуванням елементів. Процедура авторизації гарантує безпечний доступ до системи, а модульна архітектура дозволяє гнучко керувати роботою окремих компонентів залежно від потреб користувача. Детальні панелі налаштувань для кожного модуля надають можливість точного конфігурування параметрів без необхідності втручання в програмний код. Інтегрована система журналювання з відображенням подій у режимі реального часу забезпечує ефективний моніторинг і аналіз виявлених загроз. Простота встановлення, налаштування та експлуатації робить розроблений засіб придатним для використання як досвідченими адміністраторами мереж, так і користувачами з базовими знаннями інформаційної безпеки.

3.5 Висновки до розділу 3

У цьому розділі було здійснено практичну реалізацію системи раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей. Проведений детальний аналіз технологічного стеку дозволив обґрунтовано обрати Python як основну мову програмування у поєднанні з бібліотеками Socket, Scapy та фреймворком Flask, що забезпечило оптимальний баланс між швидкістю розробки, функціональними можливостями та продуктивністю системи.

Розроблений програмний засіб включає комплексний підхід до виявлення загроз через реалізацію цифрових пасток на рівні DNS, HTTP та ARP протоколів, що дозволяє ефективно виявляти різні типи підозрілої активності: несанкціоноване сканування мережі, спроби підключення неавторизованих клієнтів, використання sniffer-інструментів та інші потенційні загрози. Багатопоточна архітектура системи забезпечує одночасну роботу всіх модулів без втрати продуктивності.

Веб-інтерфейс системи надає зручні засоби для централізованого управління всіма компонентами, включаючи механізми автентифікації, панель керування модулями з можливістю індивідуального запуску та зупинки, детальну конфігурацію параметрів та журналювання подій у реальному часі. Розроблена інструкція користувача забезпечує простий та інтуїтивно зрозумілий процес встановлення, налаштування та експлуатації системи.

Тестування функціональності підтвердило ефективність розробленого рішення для раннього виявлення підозрілої активності в локальних Wi-Fi мережах. Система демонструє стабільну роботу всіх модулів, швидке реагування на потенційні загрози та детальне логування виявлених інцидентів, що робить її придатною для практичного використання в реальних умовах експлуатації.

ВИСНОВОК

У першому розділі було проведено комплексний аналіз теоретичних основ виявлення підозрілої активності в локальних Wi–Fi мережах. Досліджено різноманітні підходи до виявлення потенційно зловмисної діяльності, зокрема сканування мережі та використання сніферів. Розглянуто методи активної та пасивної розвідки в локальних Wi–Fi мережах, що дозволило оцінити сучасні тенденції та вразливості бездротових технологій. Проаналізовано існуючі програмні рішення для раннього виявлення підозрілої активності та оцінено їх переваги й недоліки.

На основі проведеного аналізу встановлено, що більшість сучасних засобів виявлення загроз функціонують у реактивному режимі – тобто активуються лише після здійснення атаки, що суттєво знижує їхню ефективність щодо запобігання загрозам на ранніх етапах. Також виявлено, що значна частина рішень спеціалізується на одному типі honeypot–пастки, що обмежує можливості користувача у побудові комплексного захисту. Ці обмеження обумовили необхідність розробки системи раннього виявлення підозрілої активності у Wi–Fi мережах за допомогою цифрових пасток з використанням хибних DNS і HTTP–відповідей.

У другому розділі було розроблено інноваційний підхід до раннього виявлення підозрілої активності, сканерів та сніферів у локальних Wi–Fi мережах за допомогою цифрових пасток із використанням хибних DNS і HTTP–відповідей. Концептуальною перевагою запропонованого підходу є проактивна парадигма виявлення загроз на стадії підготовки атаки, що істотно відрізняє розроблену систему від більшості існуючих рішень. Ключовою новацією розробленого підходу є інтеграція різнотипних honeypot–механізмів у єдину систему раннього попередження з централізованим управлінням.

Спроєктовано архітектуру програмного засобу на основі модульного принципу, що забезпечує гнучкість, масштабованість та ефективну взаємодію компонентів системи. Детально розроблено логіку обробки запитів DNS/HTTP як

honeypot–механізму, що забезпечує ефективне виявлення підозрілої активності на ранніх етапах мережевої розвідки.

У третьому розділі було здійснено практичну реалізацію системи раннього виявлення підозрілої активності у Wi–Fi мережі з використанням технології цифрових пасток. Обґрунтовано вибір Python як основної мови програмування у поєднанні з бібліотеками Socket, Scapy та фреймворком Flask, що забезпечило оптимальний баланс між швидкістю розробки, функціональними можливостями та продуктивністю системи.

Розроблений програмний засіб включає комплексний підхід до виявлення загроз через реалізацію цифрових пасток на рівні DNS, HTTP та ARP протоколів, що дозволяє ефективно виявляти різні типи підозрілої активності: несанкціоноване сканування мережі, спроби підключення неавторизованих клієнтів, використання sniffer–інструментів та інші потенційні загрози.

Створено веб–інтерфейс програмного рішення, який надає зручні засоби для централізованого управління всіма компонентами, включаючи механізми автентифікації, панель керування модулями з можливістю індивідуального запуску та зупинки, детальну конфігурацію параметрів та журналювання подій у реальному часі. Розроблена інструкція користувача забезпечує простий та інтуїтивно зрозумілий процес встановлення, налаштування та експлуатації системи.

Проведене успішне тестування функціональності підтвердило ефективність розробленого рішення для раннього виявлення підозрілої активності в локальних Wi–Fi мережах. Система демонструє стабільну роботу всіх модулів, швидке реагування на потенційні загрози та детальне логування виявлених інцидентів, що робить її придатною для практичного використання в реальних умовах експлуатації.

Результатом роботи стала успішна розробка програмного засобу, що забезпечує раннє виявлення підозрілої активності у Wi–Fi мережах за допомогою інтегрованої системи цифрових пасток з використанням хибних DNS і HTTP–відповідей. Запропонований підхід дозволяє забезпечити проактивний захист

локальних бездротових мереж, виявляючи потенційні загрози на етапі мережевої розвідки, що значно підвищує рівень інформаційної безпеки.

Опираючись на усі проведені дослідження та враховуючи наведені результати роботи, можна вважати, що в даній бакалаврській кваліфікаційній роботі вдалося досягнути поставленої мети та виконати усі задачі, а саме – розроблено програмний засіб для раннього виявлення сканувальної активності, використання Sniffer'ів та інших потенційно зловмисних дій у локальній Wi–Fi мережі шляхом впровадження цифрових пасток на основі хибних DNS та HTTP–відповідей, що дозволяє забезпечити превентивний захист мережевої інфраструктури та підвищити рівень інформаційної безпеки організацій.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Lassabe F., Baala O., Caminada A., Klein R., Lecomte S. A Friis-based calibrated model for WiFi terminals positioning. Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks. IEEE, 2005. p. 382-387.

2. Головко Д. Безпека в цифровому просторі : електронний навчальний курс. Біла Церква: БІНПО ДЗВО «УМО» НАПН України, 2024. 54 с.

3. Maseer Z. K., Jasim A. D., Murtadha K. S., Mohammed M. N. Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. IEEE access. 2021. Vol. 9. P. 22351–22370.

4. Mahmoud H. M., Smythe R. T., Régnier M. Analysis of Boyer-Moore-Horspool string-matching heuristic. Random Structures & Algorithms. 1997. Vol. 10.1-2. P. 169–186.

5. Ткаченко О. М., Тукало О. Ф. Г., Дзісь О. В., Лаховець С. М. Метод кластеризації на основі послідовного запуску k-середніх з удосконаленим вибором кандидата на нову позицію вставки. Наукові праці Вінницького національного технічного університету. 2012. № 2. С. 10.

6. Nocera F., Carnevale L., Boubeta-Puig J., Celesti A. A user behavior analytics (uba)-based solution using lstm neural network to mitigate ddos attack in fog and cloud environment. 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH). IEEE, 2022. p. 74-79.

7. Hakonen P. Detecting insider threats using user and entity behavior analytics : master's thesis. JAMK University of Applied Sciences. 2022. p. 32-48.

8. Корольков Р., Куцак С. Аналіз DoS–атаки типу DeauthF у мережах стандарту 802.11. Прикладні науково-технічні дослідження : матеріали V міжнар. наук.-прак. конф., 5–7 квіт. 2021 р. / Академія технічних наук України. Івано-Франківськ : Видавець Кушнір Г. М., 2021. С. 54.

9. KITASUKA T., NAKANISHI T., FUKUDA A. Wireless LAN based indoor positioning system WiPS and its simulation. 2003 IEEE Pacific Rim Conference on Communications Computers and Signal Processing (PACRIM 2003), 28–30 серп. 2003

р. Викторія, Канада. IEEE, 2003. С. 272–275.

10. Джулій В. М. Моделі та алгоритми виявлення атак в бездротових мережах передачі даних / В. М. Джулій, О. С. Лєнков, Л. О. Ряба // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – Київ : ВІКНУ, 2018. – Вип. 59. – С. 76-87.

11. Naik N., Jenkins P. Discovering hackers by stealth: Predicting fingerprinting attacks on honeypot systems. 2018 IEEE International Systems Engineering Symposium (ISSE). IEEE, 2018. P. 1–8.

12. Onyebuchi O. B. Signature based network intrusion detection system using feature selection on android. Signature. 2020. Vol. 11. № 6. P. 551–558.

13. Jordan P. L. Reliable presence detection through passive IEEE 802.11 management frame sniffing / P. L. Jordan, A. J. Sellers // Military Cyber Affairs. – 2016. – Vol. 1, Iss. 1. – Art. 5. – DOI: http://dx.doi.org/10.5038/2378-0789.1.1.1006.

14. Xylomenos G., Polyzos G. C. TCP and UDP performance over a wireless LAN. IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320). IEEE, 1999. Vol. 2. P. 439–446.

15. Korolkov R. Сценарій атаки з використанням несанкціонованої точки доступу у мережах IEEE 802.11 / R. Korolkov // Кібербезпека: освіта, наука, техніка. – 2021. – № 11. – С. 144–154. – DOI: https://doi.org/10.28925/2663-4023.2021.11.144154

16. SHEVTSOVA E., HANSSON C. Species recognition through wing interference patterns (WIPs) in *Achrysocharoides* Girault (Hymenoptera, Eulophidae) including two new species. *ZooKeys*. 2011. Vol. 154. P. 9–30.

17. Балабуха І. О. Сканування та аналіз пристроїв локальної мережі / І. О. Балабуха ; наук. керівник к. т. н., ас. І. О. Кобилін // Радіоелектроніка та молодь у XXI столітті : матеріали 29-го Міжнар. молодіж. форуму, 16–19 квітня 2025 р. – Харків : ХНУРЕ, 2025. – Т. 7 – С. 6-7

18. Гальчинський Л. Ю., Корольова В. Р. Оцінка захищеності бездротових мереж у міському середовищі з урахуванням використання протоколів безпеки.

Інфокомунікаційні та комп’ютерні технології. 2023. № 2.06. С. 9–21.

19. Masiukiewicz A., Tarykin V., Podvornyi V. Security Threats in Wi–Fi Networks. Engineering and Science. 2016. Vol. 1. № 3. P. 6–11.

20. Мартиненко Д. О., Коломицев М. В. Тестування безпеки WiFi за допомогою Wireshark // Теоретичні і прикладні проблеми фізики, математики та інформатики : матеріали XXII Всеукр. наук.-практ. конф. студентів, аспірантів та молодих учених, 13–17 трав. 2024 р. – Київ : КПІ ім. Ігоря Сікорського, 2024. – С. 279–281.

21. Ansari S., Rajeev S. G., Chandrashekar H. S. Packet sniffing: a brief introduction. IEEE potentials. 2003. Vol. 21. № 5. P. 17–19.

22. Conti M., Dragoni N., Lesyk V. A survey of man in the middle attacks. IEEE communications surveys & tutorials. 2016. Vol. 18. № 3. P. 2027–2051.

23. Morsy S. M., Nashat D. D-ARP: an efficient scheme to detect and prevent ARP spoofing. IEEE Access. 2022. Vol. 10. P. 49142–49153.

24. Pradana D. A., Budiman A. S. The dhcp snooping and dhcp alert method in securing dhcp server from dhcp rogue attack. IJID (International Journal on Informatics for Development). 2021. Vol. 10.1. P. 38–46.

25. Alcock S., Lorier P., Nelson R. Libtrace: A packet capture and analysis library. ACM SIGCOMM Computer Communication Review. 2012. Vol. 42.2. P. 42–48.

26. Trabelsi Z., Rahmani H. Promiscuous Mode Detection Platform. WOSIS. 2004. P. 279–292.

27. McGeehan J. E., Tucker R. S., Blaikie R. J. All-optical decrementing of a packet's time-to-live (TTL) field and subsequent dropping of a zero-TTL packet. Journal of lightwave technology. 2003. Vol. 21.11. P. 2746.

28. He S., Chan S-H. G. Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons. IEEE Communications Surveys & Tutorials. 2015. Vol. 18.1. P. 466–490.

29. Kolev K., Shterev Y. Wireless security issues. ENVIRONMENT. TECHNOLOGIES. RESOURCES. Proceedings of the International Scientific and Practical Conference. 2024. Vol. 4. P. 150–154.

30. Бурячок В. Л. Застосування бездротових мереж в ході організації та проведення розвідки систем телекомунікацій. Захист інформації. 2013. № 4. С. 284–291.

31. K. Illi ; supervised by Dr. R. Steiger. – Schaffhausen : Kantonsschule Schaffhausen, 2016. – 62 p. – P. 29–36.

32. Haas S., Sommer R., Fischer M. Zeek–osquery: Host–network correlation for advanced monitoring and intrusion detection. IFIP International Conference on ICT Systems Security and Privacy Protection. Cham : Springer International Publishing, 2020. P. 248–262.

33. Caswell B. Snort 2.0 intrusion detection / B. Caswell, J. Beale, J. C. Foster, R. Russell, J. Posluns. – Rockland : Syngress Publishing, 2003. – 523 c.

34. Provos N. Honeyd–a virtual honeypot daemon. 10th dfn–cert workshop, hamburg, germany. 2003. Vol. 2. P. 4.

35. Simbana S. Vulnerability analysis toolkit for IEEE 802.11 wireless networks: a practical approach / S. Simbana, A. Segarra, P. Vallejo, L. Astudillo // 2018 International Conference on Information Systems and Computer Science (INCISCOS). – IEEE, 2018. – P. 227–232.

36. Cabral W., Valli C., Sikos L., Wakeling S. Review and analysis of cowrie artefacts and their potential to be used deceptively. 2019 International Conference on computational science and computational intelligence (CSCI). IEEE, 2019. P. 166–171.

37. Saskara G. A. J., Dodik A. W., Sasmita G. M. A. Performance of Kismet Wireless Intrusion Detection System on Raspberry Pi. IConVET 2021: Proceedings of the 4th International Conference on Vocational Education and Technology, IConVET 2021, 27 November 2021, Singaraja, Bali, Indonesia. European Alliance for Innovation, 2022. P. 110.

38. Moreno D. Pentest em redes sem fio / D. Moreno. – São Paulo : Novatec Editora, 2016. – 1st ed. – 344 p.

39. Бобровнікова К. Ю**.** Методи та програмне забезпечення інформаційної технології виявлення бот-мереж на основі аналізу DNS-трафіка / К. Ю. Бобровнікова // Вісник Хмельницького нац. ун-ту. Технічні науки. – 2016. – № 2.

– С. 53–57.

40. Смірнов О. А. Інформаційна безпека держави: методичні вказівки до виконання лабораторних робіт для студентів за спец. 125 "Кібербезпека" / О. А. Смірнов, О. К. Конопліцька-Слободенюк, В. Д. Хох, С. А. Смірнов. – Кропивницький : ЦНТУ, 2017. – 90 с.

41. Колодчак О. Сучасні методи виявлення аномалій в системах виявлення вторгнень. Вісник Національного ун-т «Львівська політехніка». Комп'ютерні системи та мережі. 2012. № 745. С. 98–104.

42. Owens M. SQLite / M. Owens, G. Allen. – 2nd ed. – New York : Apress LP, 2010. – 368 с.

43. Kathiravelu P. Python Network Programming Cookbook / P. Kathiravelu, M. O. F. Sarker. – 2nd ed. – Birmingham : Packt Publishing Ltd., 2017. – 422 с.

44. Hassan G. M., Hussien N. M., Mohialden Y. M. Python tcp/ip libraries: A review. International Journal Papier Advance and Scientific Review. 2023. Vol. 4.2. P. 10–15.

45. Kendhe S., Gharge S., Mulik P., More A. Comparative Analysis of Different Python Editors. Vidhyayana-An International Multidisciplinary Peer-Reviewed E-Journal-ISSN 2454-8596. 2023. Vol. 8.si7. P. 562–574.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС

д.т.н., професор

Юрій ЯРЕМЧУК

" 20 " березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської кваліфікаційної роботи на тему:

Розробка системи раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей

08–72.БДР.023.00.104.ТЗ

Керівник бакалаврської кваліфікаційної роботи

к.т.н., доц., доцент каф. МБІС

Шиян А. А.

" "

Вінниця – 2025 р.

1. Найменування та область застосування

Програмний засіб для раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 81 від 05.03.2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробити програмний засіб для раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей

3.2 Призначення: розроблений програмний засіб забезпечує раннє виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей

4. Джерела розробки

4.1. Jordan P. L. Reliable presence detection through passive IEEE 802.11 management frame sniffing / P. L. Jordan, A. J. Sellers // Military Cyber Affairs. – 2016. – Vol. 1, Iss. 1. – Art. 5. – DOI: http://dx.doi.org/10.5038/2378-0789.1.1.1006.

4.2. Korolkov R. Сценарій атаки з використанням несанкціонованої точки доступу у мережах IEEE 802.11 / R. Korolkov // Кібербезпека: освіта, наука, техніка. – 2021. – № 11. – С. 144–154. – DOI: https://doi.org/10.28925/2663-4023.2021.11.144154

4.3. Ansari S., Rajeev S. G., Chandrashekar H. S. Packet sniffing: a brief introduction. IEEE potentials. 2003. Vol. 21. № 5. P. 17–19.

4.4. Балабуха І. О. Сканування та аналіз пристроїв локальної мережі / І. О. Балабуха ; наук. керівник к. т. н., ас. І. О. Кобилін // Радіоелектроніка та молодь у XXI столітті : матеріали 29-го Міжнар. молодіж. форуму, 16–19 квітня 2025 р. – Харків : ХНУРЕ, 2025. – Т. 7 – С. 6-7

4.5. Masiukiewicz A., Tarykin V., Podvornyi V. Security Threats in Wi–Fi Networks. Engineering and Science. 2016. Vol. 1. № 3. P. 6–11.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Intel Core i3–5300U 2.20GHz і подібні до них;

– оперативна пам’ять – не менше 1Gb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко–економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської кваліфікаційної роботи, формулювання теми	23.03.2025	26.03.2025
2.	Аналіз предметної області обраної теми	26.03.2025	16.04.2025
3.	Розробка алгоритму роботи	16.04.2025	21.04.2025
4.	Написання бакалаврської кваліфікаційної роботи на основі розробленої теми	22.04.2025	15.05.2025
5.	Передзахист бакалаврської кваліфікаційної роботи	16.05.2025	26.05.2025
6.	Виправлення, уточнення, корегування бакалаврської кваліфікаційної роботи	27.05.2025	05.06.2025
7.	Захист бакалаврської кваліфікаційної роботи	09.06.2025	10.06.2025

10. Порядок контролю та прийому

10.1 До приймання бакалаврської кваліфікаційної роботи надається:

– ПЗ до бакалаврської кваліфікаційної роботи;

- демонстрація результату бакалаврської кваліфікаційної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв _____ Стрелков М. Г.

Додаток Б. Лістинг коду програмного засобу

```
import threading
import yaml
from honeypots import dns, http, arp
from sniffers import analyzer

def main():
    with open("config/settings.yaml", "r") as f:
        settings = yaml.safe_load(f)

    dns_thread = threading.Thread(target=dns.start, args=(settings,), daemon=True)
    http_thread = threading.Thread(target=http.start, args=(settings,), daemon=True)
    arp_thread = threading.Thread(target=arp.start, args=(settings,), daemon=True)
    analyzer_thread = threading.Thread(target=analyzer.start, args=(settings,), daemon=True)

    dns_thread.start()
    http_thread.start()
    arp_thread.start()
    analyzer_thread.start()

    print("\n[MAIN] Honeypots are running... Press Ctrl+C to stop.")

    try:
        while True:
            pass  # тримаємо головний потік живим
    except KeyboardInterrupt:
        print("\n[MAIN] Stopping honeypots...")

if __name__ == "__main__":
    main()
from flask import Flask, render_template, request, redirect, url_for, session
import threading
import yaml
from werkzeug.security import check_password_hash


from honeypots import dns, http, arp
from sniffers import analyzer

app = Flask(__name__)
app.secret_key = "supersecretkey"

# Завантаження налаштувань один раз при старті Flask
with open("config/settings.yaml", "r") as f:
    settings = yaml.safe_load(f)

with open("config/users.yaml") as f:
    users = yaml.safe_load(f)["users"]

# Потоки для кожного модуля (по одному на кожен honeypot/sniffer)
threads = {
    "dns": None,
    "http": None,
    "arp": None,
    "analyzer": None,
}
```

```
stop_events = {
    "dns": threading.Event(),
    "http": threading.Event(),
    "arp": threading.Event(),
    "analyzer": threading.Event(),
}

# Функція для запуску модуля у потоці
def run_module(module_name):
    event = stop_events[module_name]
    if module_name == "dns":
        dns.start(settings, event)
    elif module_name == "http":
        http.start(settings, event)
    elif module_name == "arp":
        arp.start(settings, event)
    elif module_name == "analyzer":
        analyzer.start(settings, event)

# Функція для запуску потоку, якщо він не запущений
def start_module(module_name):
    if threads[module_name] is None or not threads[module_name].is_alive():
        thread = threading.Thread(target=run_module, args=(module_name,), daemon=True)
        threads[module_name] = thread
        thread.start()

# Функція для зупинки модуля
def stop_module(module_name):
    event = stop_events[module_name]
    if threads[module_name] and threads[module_name].is_alive():
        event.set()  # сигналізуємо про зупинку
        threads[module_name].join(timeout=2)  # чекаємо завершення
        threads[module_name] = None  # очищаємо потік
        stop_events[module_name].clear()  # скидаємо флаг для можливого перезапуску

@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        if username in users and check_password_hash(users[username], password):
            session["logged_in"] = True
            return redirect(url_for("index"))
        return "Invalid credentials", 401
    return render_template("login.html")

@app.route("/logout", methods=["POST"])
def logout():
    session.clear()
    return redirect(url_for("login"))

@app.route("/", methods=["GET", "POST"])
def index():

    if not session.get("logged_in"):
        return redirect(url_for("login"))

    if request.method == "POST":
```

```
        action = request.form.get("action")
        module = request.form.get("module")

        if action == "start":
            start_module(module)
        elif action == "stop":
            stop_module(module)

        return redirect(url_for("index"))

    # Статус модулів (чи працює потік)
    status = {mod: (thr.is_alive() if thr else False) for mod, thr in threads.items()}

    # Читаємо логи для відображення (останніх 50 рядків)
    try:
        with open("log/events.log", "r") as log_file:
            log_lines = log_file.readlines()[-50:]
    except FileNotFoundError:
        log_lines = []

    return render_template("index.html", status=status, logs=log_lines)
from werkzeug.security import generate_password_hash, check_password_hash

@app.route("/config", methods=["GET", "POST"])
def config():
    if not session.get("logged_in"):
        return redirect(url_for("login"))

    if request.method == "POST":A
        fields = [
            "dns_ip", "http_ip", "arp_interface", "arp_fake_mac", "arp_ip",
            "sniffer_interface", "sniffer_arp_ip", "log_file"
        ]
        for field in fields:
            settings[field] = request.form.get(field, settings.get(field))

        int_fields = ["dns_port", "http_port"]
        for field in int_fields:
            settings[field] = int(request.form.get(field, settings.get(field)))

        list_fields = {
            "dns_domains": "dns_domains",
            "http_paths": "http_paths",
            "sniffer_domains": "sniffer_domains",
        }
        for field in list_fields:
            raw = request.form.get(field, "")
            settings[field] = [x.strip() for x in raw.split(",") if x.strip()]

        with open("config/settings.yaml", "w") as f:
            yaml.dump(settings, f)

        return redirect(url_for("config"))

    return render_template("config.html", config=settings)

@app.route("/logs")
def get_logs():
```

```
    try:
        with open("log/events.log", "r") as log_file:
            log_lines = log_file.readlines()[-50:]
    except FileNotFoundError:
        log_lines = []
    return {"logs": log_lines}

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=5000)
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Settings</title>
    <style>
        body { font-family: Arial; padding: 20px; }
        input[type=text], input[type=number] {
            width: 300px;
            padding: 5px;
            margin: 4px 0;
        }
        label { display: block; margin-top: 10px; }
        button { padding: 8px 15px; margin-top: 20px; }
    </style>
</head>
<body>
    <h1> Settings</h1>

    <form method="post">

        <h2>DNS</h2>
        <label>IP: <input type="text" name="dns_ip" value="{{ config['dns_ip'] }}"></label>
        <label>Port: <input type="number" name="dns_port" value="{{ config['dns_port'] }}"></label>
        <label>Domains (comma-separated):<br>
            <input type="text" name="dns_domains" value="{{ config['dns_domains'] | join(', ') }}">
        </label>

        <h2>HTTP</h2>
        <label>IP: <input type="text" name="http_ip" value="{{ config['http_ip'] }}"></label>
        <label>Port: <input type="number" name="http_port" value="{{ config['http_port'] }}"></label>
        <label>Suspicious paths (comma-separated):<br>
            <input type="text" name="http_paths" value="{{ config['http_paths'] | join(', ') }}">
        </label>

        <h2>ARP</h2>
        <label>Interface: <input type="text" name="arp_interface" value="{{ config['arp_interface'] }}"></label>
        <label>Fake MAC: <input type="text" name="arp_fake_mac" value="{{ config['arp_fake_mac'] }}"></label>
        <label>Target IP: <input type="text" name="arp_ip" value="{{ config['arp_ip'] }}"></label>

        <h2>Sniffer</h2>
        <label>Interface: <input type="text" name="sniffer_interface" value="{{ config['sniffer_interface'] }}"></label>
        <label>ARP IP: <input type="text" name="sniffer_arp_ip" value="{{ config['sniffer_arp_ip'] }}"></label>
        <label>Watch Domains (comma-separated):<br>
            <input type="text" name="sniffer_domains" value="{{ config['sniffer_domains'] | join(', ') }}">
        </label>

        <h2>Logs</h2>
        <label>Log file path: <input type="text" name="log_file" value="{{ config['log_file'] }}"></label>
```

```
      <br>
      <button type="submit">💾 Save</button>
    </form>

    <br>
    <a href="/">← Return</a>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Honeypot Control Panel</title>
</head>
<body>
    <h1>Honeypot Control Panel</h1>

    <div style="display: flex; justify-content: flex-end; gap: 10px; margin-bottom: 20px;">
    <form action="/config" method="get" style="display:inline;">
      <button type="submit">⚙ Settings</button>
    </form>
    <form action="/logout" method="post" style="display:inline;">
      <button type="submit">🚪 Logout</button>
    </form>
    </div>

    <h2>Modules:</h2>
    {% for mod, running in status.items() %}
      <div>
        <b>{{ mod.upper() }}</b> — Status: {{ "Running" if running else "Stopped" }}

        <form method="post" style="display:inline;">
          <input type="hidden" name="module" value="{{ mod }}" />
          <button name="action" value="start" type="submit">Start</button>
        </form>

        <form method="post" style="display:inline;">
          <input type="hidden" name="module" value="{{ mod }}" />
          <button name="action" value="stop" type="submit">Stop</button>
        </form>
      </div>
    {% endfor %}

<h2>Logs:</h2>
<pre id="log-box" style="background:#f0f0f0; padding:10px; height:300px; overflow:auto;">
    Loading logs...
</pre>

<script>
    function fetchLogs() {
      fetch("/logs")
        .then(response => response.json())
        .then(data => {
          const logBox = document.getElementById("log-box");
          logBox.textContent = data.logs.join("");
          logBox.scrollTop = logBox.scrollHeight;  // прокрутка до низу
        })
```

Додаток В. Лістинг коду взаємодії між модулями розробленого програмного засобу

```
import socket
from log.logger import log_event
from datetime import datetime

def detect_suspicious_domain(domain, suspicious_keywords):
    for keyword in suspicious_keywords:
        if keyword in domain.lower():
            return True
    return False

def extract_domain(data):
    domain_parts = []
    i = 12  # DNS header — 12 байтів
    length = data[i]

    while length != 0:
        i += 1
        domain_parts.append(data[i:i+length].decode())
        i += length
        length = data[i
    return ".".join(domain_parts)

def start(settings, stop_event):
    dns_ip = settings.get('dns_ip')
    dns_port = settings.get('dns_port')
    suspicious_domains = settings.get('dns_domains', [])
    server_address = (dns_ip, dns_port)
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind(server_address)
    print(f"\n[DNS Honeypot] Listening on {dns_ip}:{dns_port}")

    sock.settimeout(1)  # дозволяє переривати очікування кожну секунду
    while not stop_event.is_set():
        try:
            data, addr = sock.recvfrom(1024)
        except socket.timeout:
            continue  # перевіряємо знову
        domain = extract_domain(data)

        if detect_suspicious_domain(domain, suspicious_domains):
            log_event(f"[{datetime.now()}][DNS] Suspicious query from {addr[0]}:{addr[1]} — domain: {domain}")
        else:
            log_event(f"[{datetime.now()}][DNS] Query from {addr[0]}:{addr[1]} — domain: {domain}")
    sock.close()
    print("\n[DNS Honeypot] Stopped.")
from http.server import BaseHTTPRequestHandler, HTTPServer
from log.logger import log_event
from datetime import datetime
def detect_suspicious_path(path):
    for suspicious in suspicious_paths:
        if suspicious in path.lower():
            return True
    return False
```

```
    class HoneypotHandler(BaseHTTPRequestHandler):
        def do_GET(self):
            if detect_suspicious_path(self.path):
                log_event(f"[{datetime.now()}][HTTP] Suspicious GET request from {self.client_address[0]}:{self.client_address[1]} — Path: {self.path}")
            else:
                log_event(f"[{datetime.now()}][HTTP] GET request from {self.client_address[0]}:{self.client_address[1]} — Path: {self.path}")
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(b"<html><body><h1>Welcome to Admin Panel</h1></body></html>")
        def do_POST(self):
            content_length = int(self.headers.get('Content-Length', 0))
            post_data = self.rfile.read(content_length)
            if detect_suspicious_path(self.path):
                log_event(f"[{datetime.now()}][HTTP] Suspicious POST request from {self.client_address[0]}:{self.client_address[1]} — Path: {self.path} — Data: {post_data.decode()}")
            else:
                log_event(f"[{datetime.now()}][HTTP] POST request from {self.client_address[0]}:{self.client_address[1]} — Path: {self.path} — Data: {post_data.decode()}")
            self.send_response(200)
            self.end_headers()
            self.wfile.write(b"Received.")
    def start(settings, stop_event):
        global suspicious_paths
        suspicious_paths = settings.get("http_paths", []
        server_address = (settings.get('http_ip'), settings.get("http_port"))
        httpd = HTTPServer(server_address, HoneypotHandler)
        print(f"\n[HTTP Honeypot] Listening on {server_address[0]}:{server_address[1]}")
        httpd.timeout = 1  # дозволяє handle_request() завершуватись, навіть якщо нема запиту
        while not stop_event.is_set():
            httpd.handle_request()
        print("[HTTP Honeypot] Stopped.")
    from datetime import datetime
    from log.logger import log_event
    from scapy.layers.l2 import ARP, Ether, sendp
    from scapy.all import sniff
    def arp_reply(target_ip, target_mac, arp_ip, fake_mac, interface):
        ether = Ether(dst=target_mac)
        arp = ARP(
            op=2,               # is-at (ARP Reply)
            pdst=target_ip,       # IP призначення
            hwdst=target_mac,     # MAC призначення
            psrc=arp_ip,          # Фальшивий IP (наш honeypot)
            hwsrc=fake_mac        # Фальшивий MAC (наш honeypot)
        )
        packet = ether / arp
        sendp(packet, iface=interface, verbose=False)
        log_event(f"[{datetime.now()}][ARP Honeypot] Sent fake ARP reply: {arp_ip} is at {fake_mac} (sent to {target_mac})")
    def arp_request_callback(packet, arp_ip, fake_mac, interface):
        if packet.haslayer(ARP) and packet[ARP].op == 1:
            target_ip = packet[ARP].pdst
            target_mac = packet[ARP].hwsrc
            log_event(f"[{datetime.now()}][ARP Honeypot] Received ARP request for {target_ip} from {target_mac}")
            arp_reply(target_ip, target_mac, arp_ip, fake_mac, interface)
    def start(settings, stop_event):
```

```
        interface = settings['arp_interface']
        fake_mac = settings['arp_fake_mac']
        arp_ip = settings['arp_ip']
        print(f"\n[ARP Honeypot] Listening for ARP requests on interface '{interface}'")
        while not stop_event.is_set():
            try:
                sniff(iface=interface, filter="arp", prn=lambda pkt: arp_request_callback(pkt, arp_ip, fake_mac, interface),
store=0, timeout=1)
            except Exception as e:
                log_event(f"[{datetime.now()}][ARP Honeypot] Error: {e}")
                break
        print("\n[ARP Honeypot] Stopped.")
            .catch(error => {
                console.error("Failed to load logs:", error);
            });
        }
        // Оновлювати кожні 0.1 секунди
        setInterval(fetchLogs, 100);
        fetchLogs(); // перший запуск одразу
    </script>
    </body>
    </html>
    from datetime import datetime

    <!DOCTYPE html>
    <html>
    <head>
        <title>Login</title>
        <style>
            body {
                position: absolute;
                top: 30%;
                left: 50%;
                transform: translate(-50%, -50%);
            }
            input {
              display: flex;
              justify-content: flex-end;
            }
        </style>
    </head>
    <body>
        <h2>Login</h2>
        <form method="POST">
            <div>Username: <input type="text" name="username" /><br></div>
            <div>Password: <input type="password" name="password" /><br></div>
            <input type="submit" value="Login" />
        </form>
    </body>
    </html>
    def log_event(message):
        with open("log/events.log", "a") as f:
            f.write(f"{message}\n")
        print(f"[LOG] {message}")
```

Додаток Г. Ілюстративний матеріал

Розробка системи раннього виявлення підозрілої активності, сканерів, Sniffer'ів в локальній WI–FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP–відповідей

У даній роботі розглядається розробка системи раннього виявлення підозрілої активності, включаючи сканери та Sniffer'и, в локальній WI-FI мережі. Основна мета системи полягає у використанні "цифрових пасток" для виявлення та аналізу небажаних дій з боку зловмисників. Для цього пропонується застосування хибних DNS та HTTP-відповідей, що дозволить ефективно ідентифікувати потенційні загрози.

Виконав ст. групи 1КІТС-21Б Стрелков М.Г.

Керівник: к.т.н., доц., доцент каф. МБІС Шиян А.А.

Актуальність:

Актуальність теми зумовлена низьким рівнем захисту Wi–Fi мереж від пасивних атак, що робить впровадження цифрових пасток на основі хибних DNS і HTTP-відповідей перспективним напрямом для проактивного виявлення загроз.

Мета:

Метою роботи є розробка програмного засобу для раннього виявлення сканувальної активності, Sniffer'ів у локальній Wi–Fi мережі шляхом впровадження цифрових пасток на основі хибних DNS та HTTP–відповідей.

Задачі дослідження:

- - проаналізувати переваги та недоліки існуючих методів виявлення підозрілої активності у Wi–Fi мережах та оцінено їх ефективність;
- - проаналізувати специфіку роботи мережевих сканерів, Sniffer'ів;
- - розробити підхід до раннього виявлення підозрілої активності в локальній WI–FI мережі за допомогою цифрових пасток із хибними DNS і HTTP відповідями;
- - спроєктувати архітектуру та реалізувати програмний засіб;
- - протестувати ефективність системи у локальному середовищі Wi–Fi мережі;
- - створити інструкцію для користувача програмного засобу.

Цифрові пастки на основі хибних DNS і HTTP–відповідей

Цифрові пастки — це фіктивні DNS-імена, HTTP-ресурси та IP-адреси, які не використовуються реальними користувачами, але спеціально створені для того, щоб привертати увагу зловмисників. Будь-яке звернення до таких об'єктів вважається підозрілим, адже легітимний трафік туди не потрапляє. Це дозволяє виявляти сканери, сніфери та інші інструменти розвідки ще до початку атаки, забезпечуючи проактивний рівень захисту Wi-Fi мережі.

Переваги запропонованого підходу

- раннє виявлення підозрілої активності на етапі мережевої розвідки;
- мінімальний вплив на легітимних користувачів мережі (пастки розміщені на адресах, до яких звичайні користувачі не звертаються);
- відсутність хибних спрацювань завдяки правильному розміщенню пасток;
- автоматичне логування всіх взаємодій з пастками для аналізу адміністратором;
- простота архітектури - система фіксує факт взаємодії без потреби в складних алгоритмах класифікації.

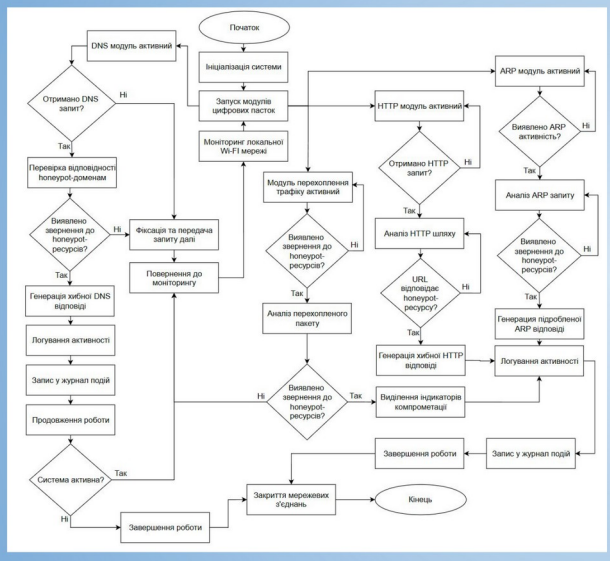


Схема відображає алгоритм роботи системи раннього виявлення підозрілої активності в локальній Wi-Fi мережі на основі цифрових пасток. Кожен модуль — DNS, HTTP та ARP — аналізує відповідні запити та визначає, чи є звернення до попередньо закладених фіктивних ресурсів. У разі збігу система генерує хибну відповідь, фіксує подію та зберігає її в базу даних для подальшого аналізу.

Центральний компонент моніторингу та модуль перехоплення трафіку дозволяють виявляти як прямі, так і непрямі прояви мережевої розвідки. Система функціонує у безперервному режимі, що забезпечує проактивний контроль за мережею, високу чутливість до пасивних атак та мінімізацію хибних спрацьовувань.

Схема відображає архітектуру роботи системи виявлення підозрілої активності у Wi-Fi мережі. Після запуску ініціалізується центральний модуль, перевіряються ресурси та активуються honeypot-механізми. Система переходить у режим моніторингу мережевого трафіку.

Якщо зафіксовано звернення до пастки, подія обробляється, фіксується в журналі та зберігається в базі даних SQLite. У разі відсутності підозрілої активності система продовжує моніторинг. Такий підхід забезпечує безперервний проактивний захист і збереження всієї критичної інформації для аналізу.

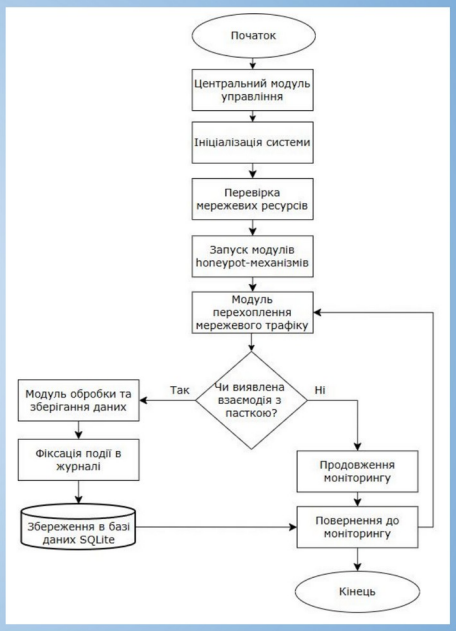
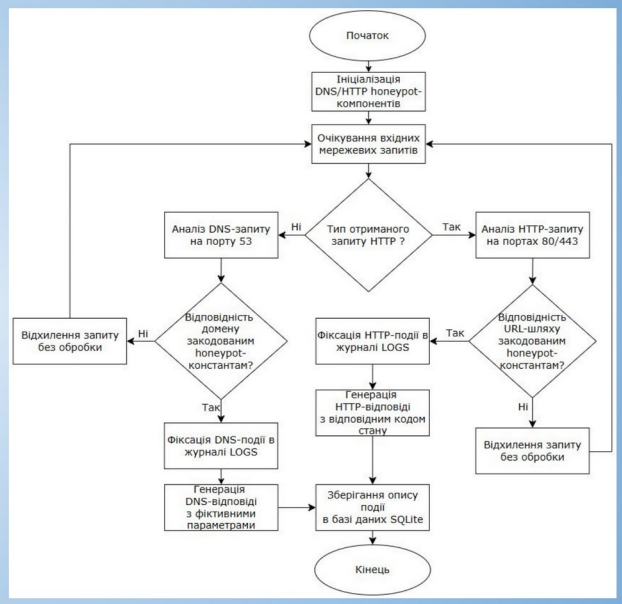
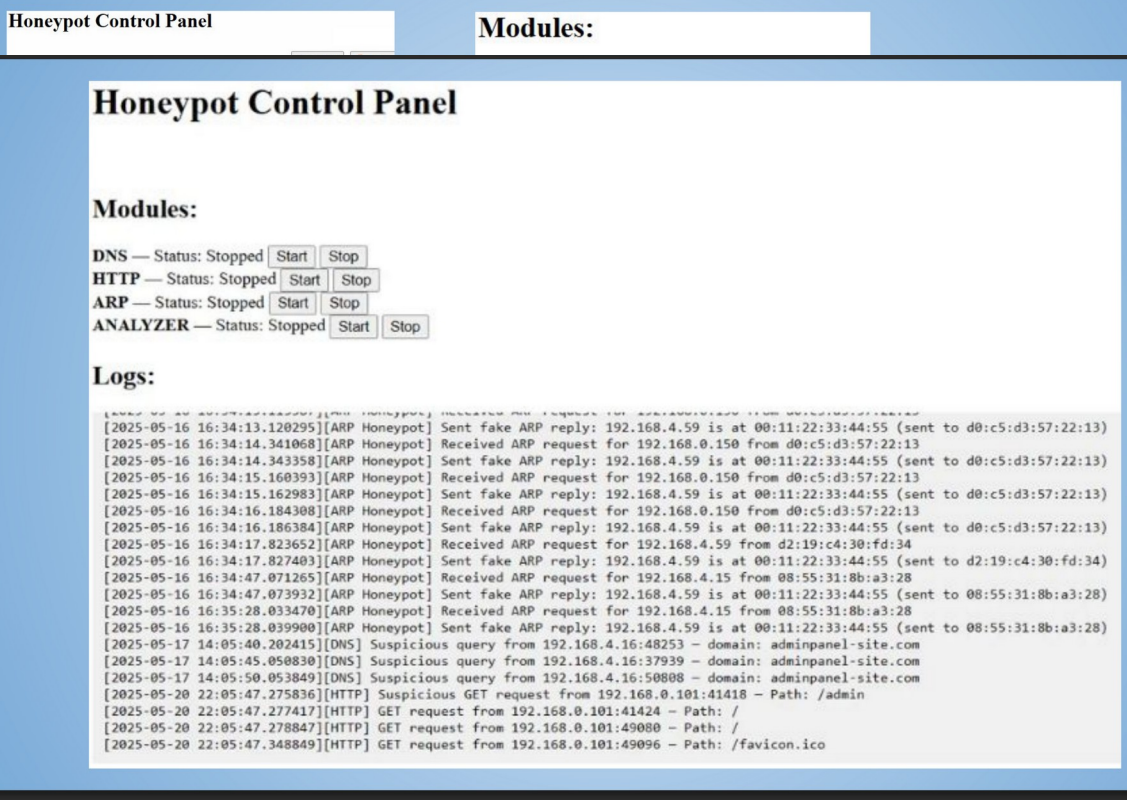
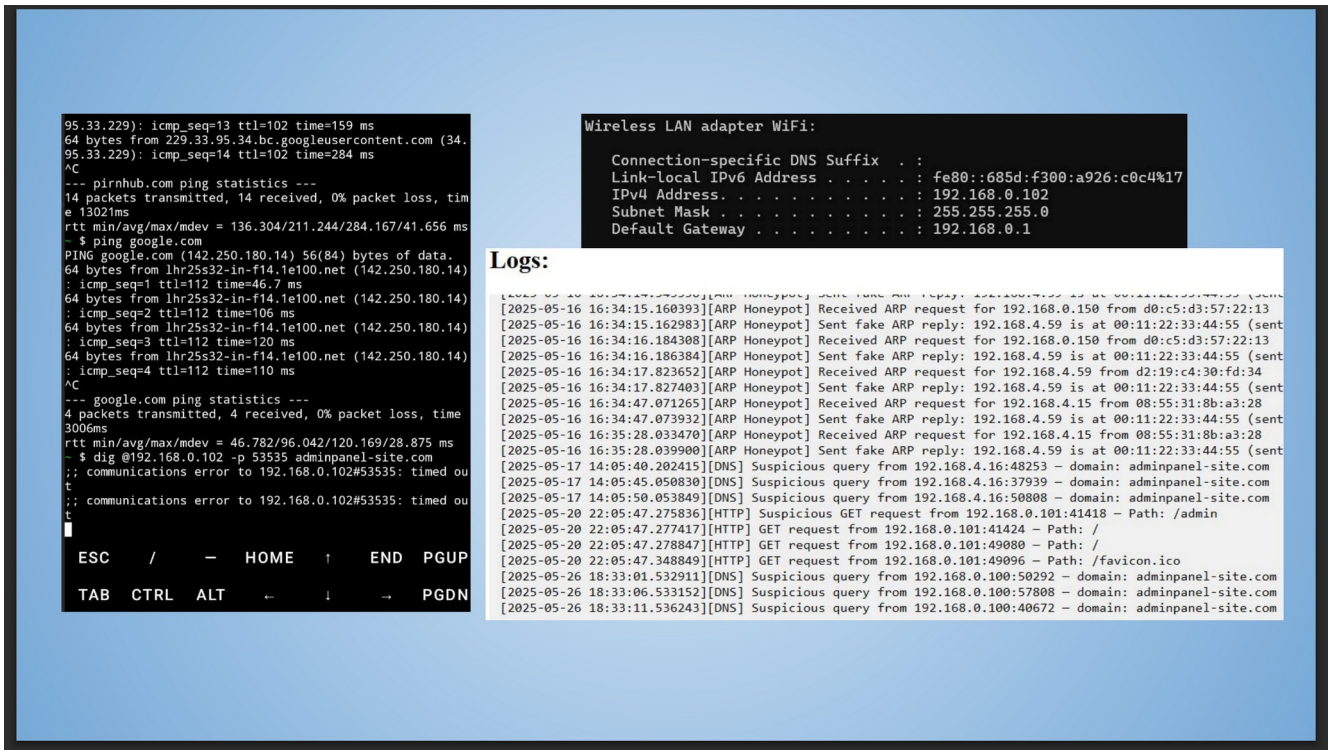


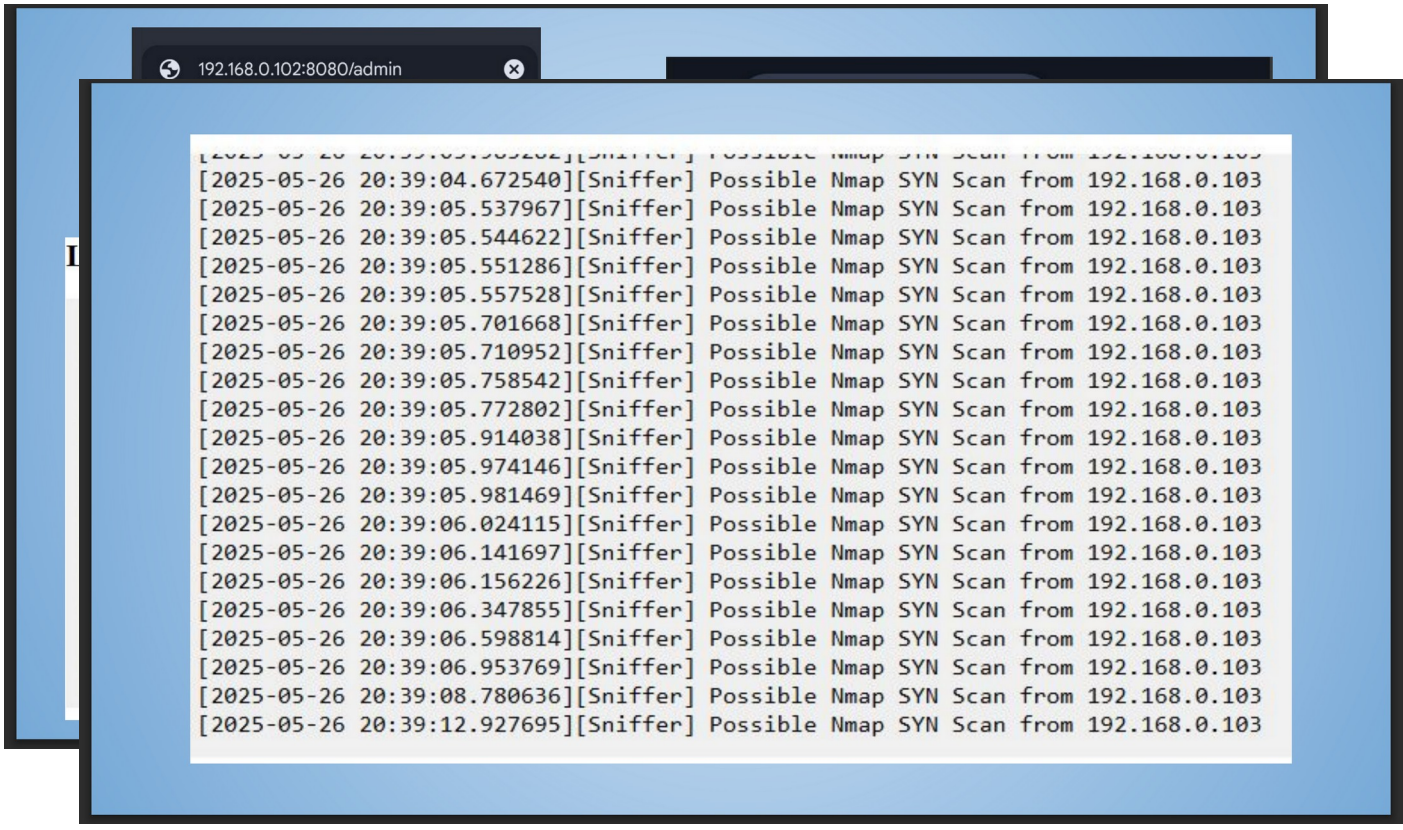
Схема ілюструє процес обробки DNS та HTTP запитів у системі honeypot. Після ініціалізації компонентів система очікує на вхідні запити та аналізує їх на відповідність закодованим фіктивним ресурсам.

Якщо виявлено звернення до honeypot-домену або URL, подія фіксується в журналі, генерується хибна відповідь і дані зберігаються в базі. Невідповідні запити відхиляються без обробки. Такий механізм дозволяє виявляти розвідку ще до початку атаки.






95.33.229): icmp_seq=13 ttl=102 time=159 ms
64 bytes from 229.33.95.34.bc.googleusercontent.com (34.
95.33.229): icmp_seq=14 ttl=102 time=284 ms
^C
--- pirnhub.com ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, tim
e 13021ms
rtt min/avg/max/mdev = 136.304/211.244/284.167/41.656 ms
~ $ ping google.com
PING google.com (142.250.180.14) 56(84) bytes of data.
64 bytes from lhr25s32-in-f14.1e100.net (142.250.180.14)
: icmp_seq=1 ttl=112 time=46.7 ms
64 bytes from lhr25s32-in-f14.1e100.net (142.250.180.14)
: icmp_seq=2 ttl=112 time=106 ms
64 bytes from lhr25s32-in-f14.1e100.net (142.250.180.14)
: icmp_seq=3 ttl=112 time=120 ms
64 bytes from lhr25s32-in-f14.1e100.net (142.250.180.14)
: icmp_seq=4 ttl=112 time=110 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time
3006ms
rtt min/avg/max/mdev = 46.782/96.042/120.169/28.875 ms
~ $ dig @192.168.0.102 -p 53535 adminpanel-site.com
;; communications error to 192.168.0.102#53535: timed ou
t
;; communications error to 192.168.0.102#53535: timed ou
t
ESC / — HOME ↑ END PGUP
TAB CTRL ALT ← ↓ → PGDN
Wireless LAN adapter WiFi:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::685d:f300:a926:c0c4%17
IPv4 Address. . . . . : 192.168.0.102
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1
Logs:
[2025-05-16 16:34:15.160393][ARP Honeypot] Received ARP request for 192.168.0.150 from d0:c5:d3:57:22:13
[2025-05-16 16:34:15.162983][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:34:16.184308][ARP Honeypot] Received ARP request for 192.168.0.150 from d0:c5:d3:57:22:13
[2025-05-16 16:34:16.186384][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:34:17.823652][ARP Honeypot] Received ARP request for 192.168.4.59 from d2:19:c4:30:fd:34
[2025-05-16 16:34:17.827403][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:34:47.071265][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:34:47.073932][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-16 16:35:28.033470][ARP Honeypot] Received ARP request for 192.168.4.15 from 08:55:31:8b:a3:28
[2025-05-16 16:35:28.039900][ARP Honeypot] Sent fake ARP reply: 192.168.4.59 is at 00:11:22:33:44:55 (sent
[2025-05-17 14:05:40.202415][DNS] Suspicious query from 192.168.4.16:48253 — domain: adminpanel-site.com
[2025-05-17 14:05:45.050830][DNS] Suspicious query from 192.168.4.16:37939 — domain: adminpanel-site.com
[2025-05-17 14:05:50.053849][DNS] Suspicious query from 192.168.4.16:50808 — domain: adminpanel-site.com
[2025-05-20 22:05:47.275836][HTTP] Suspicious GET request from 192.168.0.101:41418 — Path: /admin
[2025-05-20 22:05:47.277417][HTTP] GET request from 192.168.0.101:41424 — Path: /
[2025-05-20 22:05:47.278847][HTTP] GET request from 192.168.0.101:49080 — Path: /
[2025-05-20 22:05:47.348849][HTTP] GET request from 192.168.0.101:49096 — Path: /favicon.ico
[2025-05-26 18:33:01.532911][DNS] Suspicious query from 192.168.0.100:50292 — domain: adminpanel-site.com
[2025-05-26 18:33:06.533152][DNS] Suspicious query from 192.168.0.100:57808 — domain: adminpanel-site.com
[2025-05-26 18:33:11.536243][DNS] Suspicious query from 192.168.0.100:40672 — domain: adminpanel-site.com


192.168.0.102:8080/admin
[2025-05-26 20:39:04.672540][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.537967][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.544622][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.551286][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.557528][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.701668][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.710952][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.758542][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.772802][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.914038][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.974146][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:05.981469][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.024115][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.141697][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.156226][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.347855][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.598814][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:06.953769][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:08.780636][Sniffer] Possible Nmap SYN Scan from 192.168.0.103
[2025-05-26 20:39:12.927695][Sniffer] Possible Nmap SYN Scan from 192.168.0.103

Результат роботи

У межах роботи розроблено систему проактивного виявлення підозрілої активності в локальних Wi-Fi мережах з використанням цифрових пасток на основі хибних DNS та HTTP-відповідей. Система дозволяє фіксувати спроби сканування, використання сніферів та інші форми несанкціонованої мережевої розвідки ще до початку реальної атаки, що значно підвищує рівень безпеки мережі.

Створено модульну архітектуру з honeypot-механізмами для DNS, HTTP і ARP, реалізовано веб-інтерфейс для зручного управління, конфігурації та журналювання. Проведене тестування підтвердило стабільну роботу системи, її здатність до швидкого реагування на загрози та практичну придатність для використання в реальних умовах.

ДЯКУЮ ЗА УВАГУ

Додаток Д. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Розробка системи раннього виявлення підозрілої активності, сканерів, Sniffer'ів у локальній WI-FI мережі за допомогою «цифрових пасток» з використанням хибних DNS і HTTP-відповідей

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
Гр. 1КІТС-21б

Керівник доцент Шиян А.А. _____

Показники звіту подібності

Strike Plagiarism	
Оригінальність	99,28 %
Загальна схожість	0,72 %

Аналіз звіту подібності (відмітити потрібне)

- ☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Стрєлков М.Г.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

- ☐ **Допустити до захисту**

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Експерт
(за потреби) _____ _____
(підпис) (прізвище, ініціали, посада)