Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

МЕТОД ЗАХИСТУ ВІД ARP-СПУФІНГУ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

Виконав студент 2 курсу, групи 1КІ-24м
Спеціальності — 123 «Комп'ютерна інженерія»
_____ Паламарчук К.О.
Керівник к.т.н., доц.каф. ОТ
_____ Войцеховська О.В.
«12» 12 2025р.
Опонент к.т.н, доц. каф. ПЗ
_____ Чехместрук Р.Ю.
«12» 12 2025р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«15» 12 2025р.
_____

ВНТУ 2025

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. О.Д. Азаров
« 25 » вересня 2025 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту **Паламарчук Катерині Олександрівні**

1 Тема роботи: «Метод захисту від ARP-спуфінгу в локальних комп'ютерних мережах», керівник Войцеховська Олена Валеріївна. к.т.н., доцент кафедри ОТ затверджено наказом вищого навчального закладу від «24» вересня 2025 року № 313.

2 Термін подання студентом роботи 04.12.2025 р.

3 Вихідні дані до роботи: сучасні технології для захисту локальних комп'ютерних мереж від ARP-спуфінгу, документація ARP протоколу, технології віртуалізації.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз сучасних методів захисту від ARP-спуфінгу, розробка методу захисту від ARP-спуфінгу, реалізація методу захисту від ARP-спуфінгу, тестування методу на моделі локальної комп'ютерної мережі, економічна частина, висновки, перелік джерел посилань.

5 Перелік графічного матеріалу: структурна схема компонентів методу, блок-схема алгоритму встановлення довіри, блок-схема алгоритму обміну пакетами з використанням сеансових ключів, блок-схема алгоритму роботи криптографічного модулю при взаємодії з незахищеним вузлом, блок-схема алгоритму роботи поведінкового модулю.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Войцеховська О. В. к.т.н., доц. каф. ОТ		
5	Ратушняк О. Г. к.т.н., доц. каф. ЕПВМ		
Нормоконтроль	Швець С. І. асистент каф. ОТ		

7 Дата видачі завдання 25.09.2025 р.

8 Календарний план виконання МКР приведений у таблиці 2

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Термін виконання		Примітка
		початок	закінчення	
1	Постановка задачі роботи	25.09.2025	29.09.2025	виконано
2	Огляд та порівняння існуючих технологій	30.09.2025	02.10.2025	виконано
3	Визначення функціональних вимог до методу та проєктування	03.10.2025	05.10.2025	виконано
4	Розробка алгоритмів роботи методу	06.10.2025	16.10.2025	виконано
5	Програмна реалізація методу	17.10.2025	26.10.2025	виконано
6	Створення моделі комп'ютерної мережі, тестування методу і виправлення помилок	27.10.2025	02.11.2025	виконано
7	Розрахунок економічної частини	03.11.2025	04.11.2025	виконано
8	Оформлення пояснювальної записки	05.11.2025	11.11.2025	виконано
9	Попередній захист	12.11.2025	12.11.2025	виконано
10	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	12.11.2025	12.12.2025	виконано

Студент Катерина ПАЛАМАРЧУК

Керівник роботи Олена ВОЙЦЕХОВСЬКА

АНОТАЦІЯ

УДК 004.7

Паламарчук К.О. Метод захисту від ARP-спуфінгу в локальних комп’ютерних мережах. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2025 — 95с.

На укр. мові. Бібліогр.: 30 назв; рис.: 14; табл. 7.

У магістерській кваліфікаційній роботі розроблено метод захисту від ARP-спуфінгу в локальних комп’ютерних мережах.

Проведено огляд технології ARP, визначено можливі типи атак, а також порівняльний аналіз існуючих методів захисту від ARP-спуфінгу. Досліджено технології для реалізації захисту та засоби для моніторингу атак.

У роботі розроблено структурну схему методу захисту, алгоритми роботи криптографічного і поведінкового модулів. Виконано програмну реалізацію цих модулів, створено та налаштовано сервер довіри і базу даних криптографічних ключів. Проведено моделювання локальної комп’ютерної мережі та тестування роботи методу захисту.

Ключові слова: ARP, MITM, ARP-спуфінг, комп’ютерні мережі, захист комп’ютерних мереж, цифровий підпис, поведінкова детекція, VirtualBox, Linux.

ABSTRACT

UDC 004.7

Palamarchuk K.O. A Method for Protecting Local Computer Networks Against ARP-Spoofing. Master's qualification thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2025 — 95p.

In ukrainian. Bibliography: titles: 30; figures: 14; tables: 7.

This master’s thesis presents a method for protecting local computer networks against ARP spoofing attacks. The work includes an overview of the ARP technology, an analysis of potential attack types, and a comparative study of existing ARP-spoofing mitigation techniques. Technologies suitable for implementing protection mechanisms, as well as tools for attack monitoring, were examined.

The thesis introduces a structural diagram of the proposed protection method, along with algorithms for the cryptographic and behavioral detection modules. A full software implementation of these modules was developed, and a trust server with a cryptographic key database was configured. A virtual local network environment was modeled, followed by testing of the method's effectiveness.

Keywords: ARP, MITM, ARP-spoofing, computer networks, network security, digital signature, behavioral detection, VirtualBox, Linux.

ЗМІСТ

ВСТУП

Сучасні комп’ютерні мережі є основою для обміну даними, управління інформаційними системами та надання цифрових послуг. З розвитком технологій зростає не лише обсяг переданої інформації, а й кількість і складність загроз, спрямованих на порушення цілісності, конфіденційності та доступності даних, перехоплення або підміну мережевого трафіку. Одною з найпоширеніших атак є атаки типу Man-in-the-Middle (MITM), що дозволяють зловмиснику втручатися у комунікацію між двома вузлами мережі.

Актуальність роботи полягає в зростанні необхідності розробки методів захисту від кібератак, спрямованих на порушення цілісності та безпеки мережевих комунікацій. Особливо вразливими до MITM є локальні мережі (LAN), де використовується протокол Address Resolution Protocol (ARP) для встановлення відповідності між IP- та MAC-адресами. Цей протокол не передбачає механізмів автентифікації чи перевірки достовірності повідомлень, що робить його вразливим для ARP-спуфінгу. Це дозволяє зловмиснику перенаправляти мережевий трафік через власний вузол, перехоплювати, змінювати або блокувати передані дані.

Попри існування різноманітних методів захисту більшість рішень мають істотні обмеження: спеціальне обладнання, значне ускладнення адміністрування мережі або неможливість забезпечення належного рівня надійності. У зв’язку з цим актуальним є розроблення нових ефективних методів захисту ARP-протоколу, які можна легко інтегрувати у вже існуючу інфраструктуру без значних витрат ресурсів.

Запропонований підхід підвищення безпеки ARP шляхом використання цифрового підпису та поведінкової детекції дозволяє забезпечити автентичність ARP-повідомлень і знизити ризик реалізації атак типу ARP-спуфінг. Це дає можливість побудови більш надійних локальних мереж і підвищення рівня інформаційної безпеки без потреби в дорогому обладнанні чи складних конфігураціях.

Метою магістерської кваліфікаційної роботи є підвищення рівня захисту від ARP-спуфінгу у локальних комп’ютерних мережах та виявлення спроб підміни MAC-адрес і блокування їх у реальному часі шляхом застосування цифрового підпису та моніторингу мережевої активності.

Для досягнення поставленої мети, необхідно вирішити наступні **задачі**:

— проаналізувати атаки, що призводять до ARP-спуфінгу, та провести порівняльний аналіз сучасних методів протидії атакам;

— провести вибір стеку технологій для програмної реалізації методу захисту;

— розробити структурну схему методу захисту від ARP-спуфінгу;

— розробити алгоритм роботи методу захисту з використанням цифрового підпису та виявлення і блокування підозрілого ARP-трафіку;

— розробити програмне забезпечення для реалізації методу захисту та серверу довіри;

— провести моделювання комп’ютерної мережі з використанням запропонованого методу для перевірки коректності його роботи.

Об'єктом дослідження є процеси забезпечення захисту даних у локальних комп’ютерних мережах від атак типу ARP-спуфінг.

Предметом дослідження є методи передавання інформації в локальних комп’ютерних мережах, методи здійснення атак типу ARP-спуфінг та засоби підвищення захищеності протоколу ARP.

Новизна одержаних результатів полягає в розробці дворівневого захисту локальних комп’ютерних мереж від атак типу ARP-спуфінг, який базується на використанні цифрового підпису для автентифікації ARP-повідомлень та виявленні можливої атаки, та, на відміну від традиційних способів, забезпечує криптографічну перевірку достовірності джерела ARP-повідомлення та відстежує зміни у ARP-таблиці, що дозволяє підвищити рівень захисту локальних мереж та виявляти спроби підміни MAC-адрес і блокувати їх у реальному часі.

Практичне значення полягає у розробці алгоритму виявлення атак типу

ARP-спуфінг та у розробці програмного забезпечення для перевірки цілісності ARP пакетів і моніторингу мережевих з’єднань, яке може бути інтегроване у локальні комп’ютерні мережі без необхідності використання спеціалізованого апаратного забезпечення чи складних змін у мережевій інфраструктурі.

Апробація результатів роботи: опублікована доповідь на міжнародній науково-технічній конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)».

Публікація: «Додавання цифрового підпису до протоколу ARP для запобігання ARP-spoofing» / К. О. Паламарчук, О. В. Войцеховська // Матеріали міжнародної науково-технічної конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)». Вінниця, 2025 р. [Електронний ресурс]. — Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/viewFile/26761/21971 [1].

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ЗАХИСТУ ВІД АТАК ARP-СПУФІНГУ

1.1 Механізм роботи протоколу ARP у локальних мережах

Address Resolution Protocol (ARP) — це мережевий протокол, який використовується для визначення фізичної (MAC) адреси пристрою в локальній мережі на основі відомої IP-адреси. Його основне завдання — забезпечити взаємодію між мережевим (IP) і канальним (Ethernet) рівнями моделі OSI.

ARP працює за принципом запиту та відповіді. Коли комп'ютеру потрібно надіслати пакет до іншого вузла в тій самій локальній мережі, він спочатку перевіряє свою ARP-таблицю — кеш відповідностей IP та MAC. Якщо відповідний запис знайдено, пакет надсилається безпосередньо. Якщо ні — комп'ютер генерує широкомовний ARP-запит, який надсилається всім пристроям у мережі.

Усі пристрої отримують запит, але відповідає лише вузол, чия IP-адреса збігається з запитуваною. Він надсилає ARP-відповідь, яка містить свою MAC-адресу, і цей запис додається до ARP-таблиці відправника. У подальших комунікаціях це дозволяє уникати повторних запитів, поки запис дійсний.

Записи до цієї таблиці створюються протоколом ARP на льоту, не вимагаючи ручного втручання адміністраторів. Кеш ARP є динамічним (хоча існує можливість налаштувати статичну ARP-таблицю) і обмеженим за розміром. Зазвичай адреси залишаються в кеші лише кілька хвилин, і він регулярно очищується для звільнення місця.

Використання протоколу ARP має ряд вагомих переваг та недоліків. Серед переваг можна виділити:

— простота реалізації та універсальність, оскільки ARP підтримується усіма мережевими пристроями;

— автоматичне оновлення відповідностей IP та MAC-адрес без участі користувача;

До недоліків відносять відсутність механізмів безпеки. У протоколі не

передбачено автентифікації, будь-який пристрій може надсилати підроблені ARP-відповіді, навіть якщо ARP-запит ніколи не надсилався. Це дозволяє будь-якому хосту в локальній мережі LAN підробити повідомлення, що містить шкідливу інформацію.

Оскільки ARP покладається на широкомовні повідомлення для визначення MAC-адрес, у великих середовищах це може перевантажувати мережу, призводячи до погіршення продуктивності [2].

1.2 Аналіз можливих атак, що призводять до ARP-спуфінгу

Атака типу Man-in-the-Middle (MITM) належить до категорії активних мережевих атак, що спрямовані на перехоплення, модифікацію або підміну інформації, яка передається між двома вузлами мережі. Сутність цієї атаки полягає у втручанні зловмисника в процес обміну даними між двома легітимними сторонами комунікації таким чином, що жодна з них не підозрює про присутність стороннього посередника (рисунок 1.1).

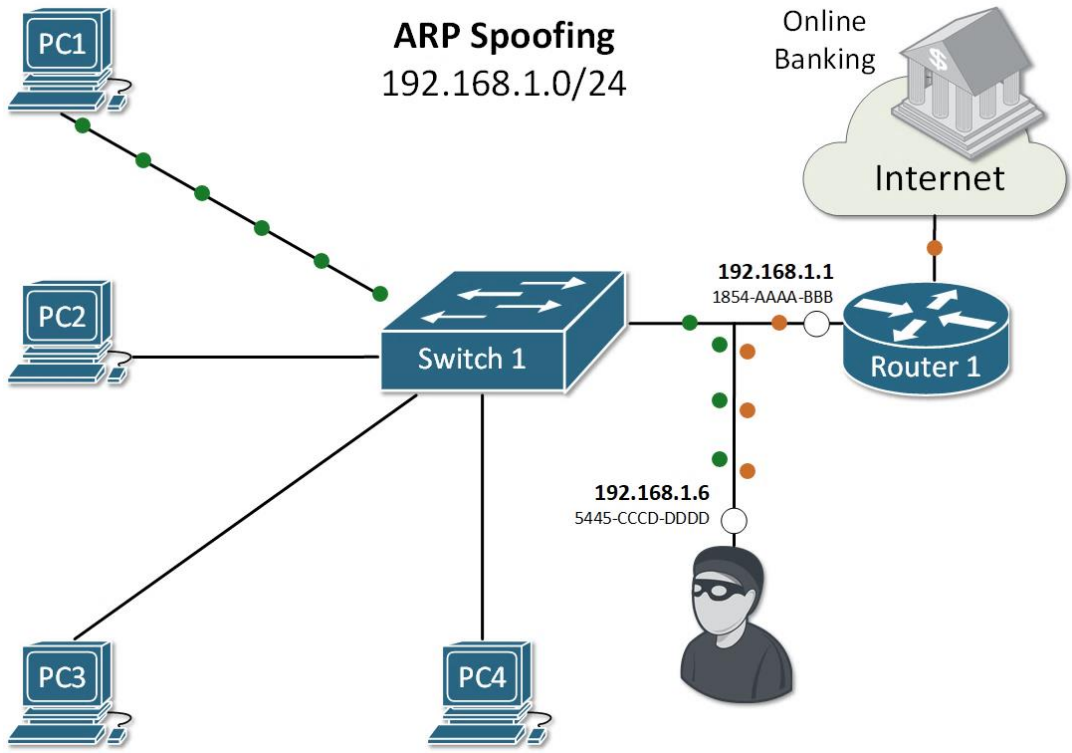

Рисунок 1.1 — Схема здійснення атаки типу Man-in-the-Middle у локальній мережі

Під час здійснення атаки MITM зловмисник може як пасивно прослуховувати передану інформацію, так і активно змінювати її в процесі передачі. У пасивному варіанті атаки зловмисник здійснює моніторинг мережевого трафіку з метою збору конфіденційних даних, таких як автентифікаційні облікові записи, сесійні токени або вміст переданих повідомлень. Активний варіант передбачає можливість модифікації, затримки або перенаправлення трафіку, що дозволяє підмінювати дані, маніпулювати запитами користувача чи перенаправляти його на фальшиві ресурси без помітних ознак втручання.

Особливу небезпеку атаки типу MITM становлять у незахищених або погано сегментованих локальних мережах, де передача даних здійснюється у відкритому вигляді без використання криптографічних механізмів шифрування та автентифікації. У таких середовищах зловмисник може реалізувати атаку без спеціальних привілеїв доступу, використовуючи стандартні інструменти аналізу трафіку та підміни адрес [3].

Ключовою умовою успішної реалізації атаки Man-in-the-Middle є здатність зловмисника контролювати або впливати на маршрутизацію мережевого трафіку між двома сторонами. У локальних мережах це зазвичай досягається через експлуатацію вразливостей протоколів канального рівня, зокрема ARP.

ARP-спуфінг (ARP-poisoning) є різновидом атаки типу Man-in-the-Middle, що реалізується на канальному рівні моделі OSI через експлуатацію вразливостей протоколу ARP. Суть цієї атаки полягає у навмисній підміні відповідей ARP, що призводить до внесення неправдивих записів у ARP-таблиці вузлів локальної мережі [4].

Протокол ARP є протоколом без збереження стану (stateless protocol), що означає, що відповідь може бути оброблена, навіть якщо відповідний запит ніколи не був отриманий. Завдяки підміні ARP-відповідей зловмисник змушує жертву надсилати пакети не безпосередньо до отримувача, а через свій пристрій, який виступає у ролі посередника. У цьому випадку атакуючий отримує змогу одночасно обмінюватися даними з обома сторонами, виконуючи роль прозорого

ретранслятора, який може змінювати або фільтрувати інформацію. У типових локальних мережах ARP-спуфінг є особливо небезпечним через широкомовний характер ARP-запитів і відсутність централізованого контролю над ARP-відповідями. Зловмисник може здійснити атаку навіть без спеціальних прав доступу, використовуючи лише базові утиліти аналізу трафіку або спеціалізовані інструменти, такі як arpspoof чи Bettercap. Атака може бути виконана в межах кількох секунд, а внесені зміни в ARP-таблицях зберігаються доти, доки їх не буде перезаписано або очищено.

1.3 Огляд існуючих методів захисту від ARP-спуфінгу

Захист від ARP-спуфінгу є складним завданням, оскільки сама архітектура протоколу ARP не передбачає механізмів перевірки достовірності джерела ARP-повідомлень. У зв’язку з цим більшість сучасних методів протидії спрямовані не на зміну протоколу, а на виявлення та запобігання фальсифікованим ARP-відповідям на рівні операційної системи, мережевого обладнання або програмного забезпечення моніторингу.

1.3.1 Використання статичних ARP-таблиць

Суть методу полягає у ручному формуванні відповідностей між IP- та MAC-адресами на кожному мережевому вузлі. У цьому випадку таблиця ARP створюється адміністратором та зберігається у пам’яті системи як незмінна. Усі динамічні ARP-запити і відповіді ігноруються, що унеможливлює підміну даних у процесі роботи мережі.

Основна перевага цього підходу полягає у його абсолютній стійкості до ARP-спуфінгу: дані не оновлюються автоматично, атакуючий комп’ютер не може змінити асоціацію IP–MAC. Недоліком є низька масштабованість та складність адміністрування, особливо у великих або динамічних комп’ютерних мережах, де кількість пристроїв постійно змінюється. Крім того, у разі заміни обладнання або зміни конфігурації мережі необхідно вручну оновлювати усі відповідні записи.

1.3.2 Використання механізму Port Security

Механізм Port Security реалізується на рівні канального рівня комутаторів і покликаний обмежити можливість зловмисника або випадкового пристрою використовувати фізичний порт комутатора для несанкціонованого доступу до мережі. Концептуально Port Security встановлює політику допустимих MAC-адрес для кожного фізичного порту і контролює трафік на підставі цієї політики; у випадку виявлення трафіку з невідомих або заборонених MAC-адрес механізм ініціює попередньо визначену реакцію, яка може включати відкидання кадрів, логування події або повне відключення порту. Таким чином, Port Security працює як перший бар’єр на канальному рівні, що обмежує можливість атак, пов’язаних з підміною MAC-адрес або підключенням несанкціонованих пристроїв.

Принцип функціонування засновано на двох базових режимах формування списку дозволених MAC-адрес: статичному й динамічному. У статичному режимі адміністратор вручну задає конкретні MAC-адреси, які можуть передавати трафік через порт; у динамічному режимі комутатор самостійно «вивчає» MAC-адреси при першому прийомі кадрів і фіксує їх у локальній прив’язці до порту. Часто реалізується опція «sticky», яка поєднує динамічне навчання з можливістю збереження отриманих прив’язок у конфігурації комутатора — це дає гібридну модель, яка спрощує адміністрування при збереженні контролю. Крім того, Port Security зазвичай підтримує параметри обмеження кількості одночасних MAC-адрес на порті та механізми старіння записів, що дозволяє адаптувати політику під середовища з мобільними або тимчасовими підключеннями.

Прикладна поведінка комутатора при виявленні порушення політики Port Security залежить від налаштованого його режиму реакції. Найпоширеніші стратегії включають «protect» (неприйнятні кадри просто відкидаються, при цьому порт залишається активним для дозволених адрес), «restrict» (кадри відкидаються, генерується системний журнал і повідомлення SNMP) та «shutdown» (порт переводиться в вимкнений стан), що вимагає ручного або

автоматичного відновлення. Вибір стратегії обґрунтовується політикою безпеки мережі.

У контексті захисту від ARP-спуфінг Port Security виконує роль попередньої перешкоди: обмеження MAC-адрес на фізичному порту унеможливлює для атакуючого легке привласнення MAC-адреси жертви шляхом підключення свого пристрою до того самого порту або заміни кабелю у неконтрольованих точках доступу. Однак Port Security не перевіряє зміст ARP-повідомлень і не контролює динамічні відповідності IP та MAC у межах мережевого сегмента, тому цей механізм сам по собі не здатен повністю запобігти отруєнню ARP-кешу, коли атакуючий знаходиться в іншому фізичному місці того самого сегмента або має доступ до кількох портів. Тобто Port Security зменшує площу атаки, пов’язану з фізичним доступом та підміною MAC-адрес на конкретних портах, але потребує інтеграції з іншими заходами, щоб забезпечити всебічний захист від ARP-спуфінгу [5].

1.3.3 Використання динамічної ARP інспекції

Механізм Dynamic ARP Inspection (DAI) є сучасним засобом апаратного рівня захисту від атак типу ARP-спуфінг у комутаторах корпоративного класу. Його принцип дії базується на контролі достовірності ARP-повідомлень, які проходять через комутатор, із метою запобігання підміні MAC-адрес у локальній мережі. DAI працює у тісній інтеграції з механізмом DHCP Snooping, що дозволяє комутатору створювати базу даних динамічних відповідностей між IP, MAC та VLAN, отриманих під час процесу розподілу IP-адрес через DHCP. Ця база використовується як еталон для перевірки всіх ARP-запитів і відповідей, що циркулюють у мережі.

У системі DAI усі порти комутатора поділяються на довірені (trusted) і недовірені (untrusted). Як правило, довіреними вважаються порти, що підключені до інших комутаторів або шлюзів маршрутизації, тобто до інфраструктурного обладнання, яке формує легітимний ARP-трафік. Усі інші порти, призначені для підключення користувацьких пристроїв, визначаються як недовірені. DAI

здійснює перевірку лише трафіку, що надходить з недовірених портів, що мінімізує навантаження на систему і підвищує ефективність фільтрації.

DAI функціонує на канальному рівні моделі OSI та реалізує механізм фільтрації ARP-пакетів у реальному часі. Кожен ARP-пакет, який проходить через комутатор, аналізується і перевіряється на відповідність записам у таблиці DHCP Snooping Binding Table. Якщо IP-адреса, зазначена у ARP-повідомленні, не відповідає зареєстрованій MAC-адресі у таблиці, або якщо пакет надходить із порту, що не визначений як довірений, такий пакет блокується. Таким чином, комутатор виступає активним фільтром, який не допускає потрапляння фальсифікованих ARP-відповідей до вузлів мережі, усуваючи саму можливість отруєння ARP-таблиць на кінцевих пристроях.

У випадках, коли у мережі використовуються статично призначені IP-адреси (тобто DHCP Snooping не формує записів для DAI), адміністратор може додавати записи прив’язки вручну до так званого static binding table. Це забезпечує підтримку гібридних схем адресації та дозволяє зберігати повний контроль над автентичністю ARP-повідомлень навіть без DHCP.

Крім базової перевірки відповідності IP–MAC, DAI також може виконувати додаткову валідацію ARP-полів, таких як ARP opcode, довжина полів або правильність вмісту кадру Ethernet. Це дозволяє виявляти та блокувати не лише підміну адрес, але й спроби створення нестандартних або модифікованих ARP-повідомлень, які використовуються у складних атаках для обходу звичайних систем фільтрації. Комутатор може логувати події виявлення підроблених пакетів і надсилати SNMP-повідомлення для централізованого моніторингу, що підвищує оперативність реагування на інциденти безпеки.

Перевага Dynamic ARP Inspection полягає у повній прозорості для користувачів і відсутності необхідності у встановленні додаткового програмного забезпечення на кінцевих пристроях. Усі перевірки здійснюються на рівні комутатора, що перевіряє достовірності ARP-трафіку в межах VLAN. Завдяки апаратній реалізації обробка ARP-пакетів не створює суттєвого навантаження на продуктивність мережі, навіть у великих інфраструктурах із сотнями вузлів.

До недоліків DAI можна віднести його залежність від коректної роботи DHCP Snooping — у разі помилкових або неповних записів у таблиці прив’язок легітимні ARP-повідомлення можуть бути відкинуті. Також складність конфігурації та необхідність підтримки цієї функції комутатором певного класу роблять її менш доступною для невеликих або бюджетних мереж. У складних топологіях із кількома VLAN адміністратор має забезпечити коректну синхронізацію бази DHCP Snooping між усіма комутаторами, щоб уникнути розсинхронізації даних.

Попри зазначені обмеження, Dynamic ARP Inspection залишається одним із найнадійніших засобів захисту від ARP-спуфінг, оскільки забезпечує активне превентивне блокування підроблених ARP-повідомлень на апаратному рівні, ще до їх потрапляння до кінцевих вузлів. У корпоративних середовищах DAI зазвичай використовується у комплексі з іншими механізмами контролю, такими як Port Security і DHCP Snooping, що разом формують багаторівневу систему безпеки.

1.3.4 Програмні засоби для відслідковування атак

Програмні засоби моніторингу та запобігання ARP-спуфінг є одним із напрямів захисту локальних мереж, орієнтованих на виявлення та запобігання атакам у процесі мережевої взаємодії. Вони реалізують функції активного або пасивного контролю ARP-трафіку на рівні кінцевих вузлів, на відміну від апаратних методів, які застосовуються на комутаторах. Такі системи відстежують підозрілі зміни у відповідностях IP–MAC, виявляють аномальні ARP-повідомлення та попереджають адміністратора або блокують шкідливий трафік.

Програма Arpwatch є класичною програмною утилітою для моніторингу ARP-трафіку в локальних мережах. Вона належить до пасивних систем виявлення мережевих аномалій і орієнтована на реєстрацію змін у відповідностях IP–MAC, що циркулюють у сегменті локальної мережі. Програма спостерігає за широкомовними ARP-повідомленнями, зберігає інформацію про

нові MAC-адреси, зміни існуючих відповідностей і підозрілі дії, такі як повторювані ARP-відповіді з різними MAC-адресами для одного IP.

Arpwatch працює на рівні операційної системи і прослуховує канальний інтерфейс у режимі promiscuous, тобто приймає всі кадри, що проходять через мережевий адаптер, незалежно від призначення. Кожне ARP-повідомлення аналізується та зіставляється з локальною базою даних, яка містить відомі відповідності IP–MAC. При виявленні нової MAC-адреси або зміненої відповідності IP–MAC утворюється запис у журналі, а також може бути надіслане сповіщення адміністратору (електронною поштою або через системні логи).

Архітектура Arpwatch дозволяє працювати як у локальному режимі (для одного сегмента мережі), так і у централізованому, коли декілька вузлів надсилають дані у спільну базу для моніторингу великої інфраструктури. Програма підтримує ведення журналів у текстовому або структурованому форматі (наприклад, CSV), що полегшує їх інтеграцію з іншими системами безпеки або SIEM.

Програма не потребує змін у мережевій інфраструктурі чи додаткового обладнання. Програма пасивно прослуховує трафік і не генерує додаткових пакетів [7].

XArp — популярний програмний засіб активного захисту від атак типу ARP-спуфінг та Man-in-the-Middle у локальних комп’ютерних мережах. На відміну від пасивних утиліт, таких як Arpwatch, XArp застосовує активну перевірку достовірності ARP-повідомлень, використовуючи алгоритми аналізу поведінки мережевих вузлів, інтегровану базу довірених IP–MAC пар і методи детекції аномалій.

Програма XArp контролює ARP-трафік у реальному часі, формуючи локальну базу довірених IP та MAC відповідностей, яка може наповнюватися автоматично при першому підключенні пристроїв або задаватися адміністратором вручну. Всі ARP-пакети, що надходять у мережевий адаптер, аналізуються щодо відповідності записам у базі. При виявленні невідповідності,

повторюваних ARP-відповідей з різними MAC-адресами для одного IP або аномальної поведінки вузлів, XArp може виконати одну з реакцій: блокування трафіку від підозрілого вузла, ізоляцію вузла на рівні ОС або повідомлення адміністратора.

Програма підтримує активне сканування мережі для перевірки достовірності ARP-відповідей і може виявляти складні сценарії атак, включно з підміною MAC-адрес та MITM через маршрутизовані сегменти. Для цього XArp застосовує методи інтелектуального аналізу аномалій: порівняння часу відповіді, частоти ARP-повідомлень, співставлення поведінки пристроїв з шаблонами нормальної активності. При великому обсязі ARP-трафіку активний аналіз і сканування можуть створювати помітне навантаження на процесор та оперативну пам’ять.

Програма здатна детектувати атаки з використанням ланцюжка вузлів, MITM через маршрутизовані сегменти та інші нестандартні сценарії ARP-спуфінгу [8].

Програма Ettercap є багатоцільовим мережевим інструментом для перехоплення, аналізу та маніпулювання трафіком у локальній мережі. Це програмне забезпечення реалізує набір можливостей для як пасивного моніторингу, так і активного втручання в мережеві сесії. Ettercap поєднує функціональні можливості аналізатора протоколів, механізми перехоплення кадрів на канальному рівні та засоби для обробки і модифікації пакетів у реальному часі, що робить його корисним як для досліджень безпеки, так і для тестування захищеності мережі.

Принцип роботи програми базується на перехопленні та обробці Ethernet-кадрів у режимі прослуховування, із подальшою кореляцією та розбором протокольних стеків. Архітектура включає ядро обробки пакетів, набір протокольних дисекторів для інтерпретації вмісту (ARP, IP, TCP, UDP, DNS, HTTP тощо), а також механізм фільтрів, що дозволяє програмно змінювати поля заголовків або payload перед передачею пакета далі по мережі. Розширюваність Ettercap реалізовано через плагінну систему, що дає змогу додавати нові методи

аналізу, валідації або реагування на підозрілу активність.

У контексті ARP-спуфінг Ettercap виступає як інструмент для демонстрації вразливостей, оскільки він містить механізми підміни ARP-відповідей і ретрансляції трафіку для моделювання MITM-сценаріїв. Проте його функціональність не обмежується лише експлуатацією вразливостей. Ettercap також може використовуватися для детектування аномалій ARP-повідомлень, аналізу шаблонів трафіку, верифікації правильності ARP-відповідностей у контрольованому середовищі та перевірки коректності роботи апаратних засобів захисту (наприклад, налаштувань DAI або Port Security). Інструмент дозволяє проводити детальне логування та кореляцію подій на рівні сесій, що корисно для аудиту та розслідування інцидентів.

Наявність фільтрів для зміни пакетів у реальному часі і протокольних дисекторів дає можливість аналізувати складні сценарії взаємодії протоколів та виявляти нетипову поведінку.

Однак, робота в режимі перехоплення потребує підвищеного рівня доступу на хості, а інтенсивний аналіз великих обсягів трафіку може значно навантажувати ресурси системи [9].

1.4 Порівняльний аналіз методів захисту

Проведений огляд методів захисту показав, що статичні ARP-таблиці ефективні у невеликих і стабільних мережах.

Port Security слід впроваджувати там, де існує необхідність жорстко контролювати фізичний доступ до мережі на рівні точок підключення кінцевих пристроїв. Цей механізм корисний для робочих станцій у відкритих офісних просторах, для портів у зонах гостьового доступу та для портів, що обслуговують критичні сервери, де небажані або сторонні MAC-адреси мають бути моментально виявлені й обмежені. Port Security є менш ефективним у середовищах з частою зміною пристроїв або при наявності підключених IP-телефонів, Wi-Fi точок доступу з кількома клієнтами або віртуальних машин із динамічною генерацією MAC-адрес.

Механізм DAI забезпечує апаратну превентивну блокаду підроблених ARP-повідомлень і є найнадійнішим засобом у великих корпоративних мережах, де є комутатори підприємницького класу, що підтримують відповідні функції.

Програмні засоби відслідковування атак (наприклад Arpwatch, XArp, оборонні конфігурації Ettercap) є найбільш гнучким інструментом, який можна швидко впровадити у систему, для середніх і малих мереж.

Порівняльна характеристика методів захисту від ARP-спуфінгу наведена у таблиці 1.1.

Таблиця 1.1 — Порівняння методів захисту від ARP-спуфінгу

Метод захисту	Простота впровадження та масштабування	Захист від атак	Потреба у спеціальному обладнання	Вплив на продуктивність
Статичні ARP-таблиці	Низька на одиничних вузлах, висока складність при масштабуванні	Високий у статичних мало змінних мережах	Не потребує спеціального обладнання	Мінімальний
Port Security	Добре масштабується	Високий рівень захисту	Потрібні керовані комутатори	Невеликий
Dynamic ARP Inspection	Складна інтеграція, легке масштабування	Високий рівень захисту за умови правильної конфігурації	Спеціалізовані комутатори з підтримкою DAI/DHCP Snooping	Мінімальний, можливе навантаження при атаці
Програмні засоби	Висока швидкість впровадження	Ефективні в детекції, але не можуть блокувати атаку на апаратному рівні	Не потребує спеціального обладнання	Може бути суттєвим на кінцевих хостах при великому трафіку

Отже, порівняння показало, що статичні ARP-таблиці та Port Security доцільні у менших або фізично контрольованих мережах, тоді як Dynamic ARP Inspection є оптимальним рішенням для масштабних корпоративних мереж із централізованим керуванням. Програмні інструменти моніторингу доповнюють апаратні механізми, забезпечуючи гнучкість і швидке впровадження.

2 РОЗРОБКА МЕТОДУ ЗАХИСТУ ВІД ARP-СПУФІНГУ

2.1 Аналіз S-ARP як модифікації протоколу ARP

Науковцями з Університету Каліфорнії був запропонований протокол, названий Secure ARP (S-ARP), як модифікація стандартного протоколу ARP з метою усунення його вразливості перед атаками типу ARP-спуфінг [10]. S-ARP призначений розв’язати цю проблему шляхом прив’язки пари IP-MAC до криптографічних підписів і централізованої моделі довіри.

Основна ідея S-ARP полягає в тому, що кожен хост у мережі має пару криптографічних ключів, а кожен ARP-пакет містить цифровий підпис, який дозволяє перевірити цілісність і справжність відправника. Для цього у протоколі передбачено існування довіреного централізованого сервера, який зберігає публічні ключі всіх хостів і видає сертифікати. Усі ARP-повідомлення супроводжуються сертифікатом від цього сервера, а отже, отримувач пакета може впевнитися, що IP-MAC відповідність походить від легітимного джерела. Такий підхід забезпечує криптографічний захист від підміни ARP-відповідей, оскільки кожне повідомлення підписане приватним ключем відправника й може бути перевірене за його сертифікованим публічним ключем.

Робота протоколу S-ARP передбачає використання декількох компонентів. У мережі повинен існувати Сервер реєстрації та сертифікації, який виконує функції довіреного центру. Він створює ключі для нових хостів або приймає їхні запити на реєстрацію, перевіряє їх та видає сертифікати, що містять публічний ключ і MAC-адресу. Кожен хост повинен мати підтримку криптографічних операцій — генерування ключів, підписування ARP-пакетів, перевірка підписів, обробка сертифікатів. Мережевий стек або додатковий модуль має бути здатним вбудовувати цифровий підпис у ARP-пакет і коректно обробляти розширений заголовок.

Повідомлення S-ARP подібне до звичайного ARP-повідомлення, але містить додаткову секцію в кінці, що забезпечує сумісність з оригінальним протоколом. Ця додаткова частина складається з 12-байтового заголовка S-ARP

і змінної за довжиною корисної частини, як показано на рисунку 2.1. ARP-відповіді доповнюються заголовком S-ARP, тоді як ARP-запити залишаються незмінними.

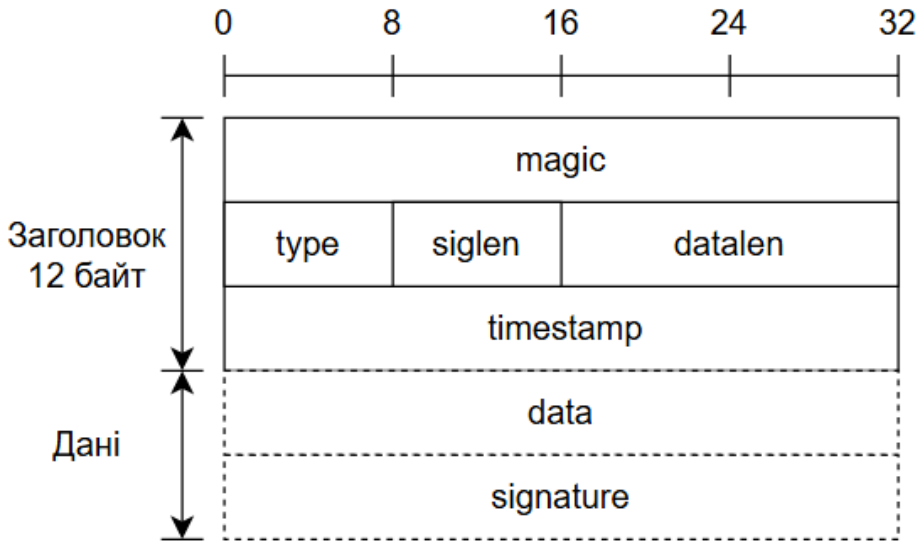

Рисунок 2.1 — Структура S-ARP-пакета

Заголовок S-ARP містить цифровий підпис відправника, часову мітку, тип повідомлення та його довжину. Поле magic призначене для визначення наявності S-ARP-заголовка. Якщо він присутній, то це поле має значення 0x7599e11e. Оскільки ARP-пакети мають довжину лише 42 байти, а мінімальна довжина Ethernet-кадру становить 60 байтів, пакети зазвичай доповнюються довільними даними. Через це отримана довжина кадру не може бути індикатором наявності додаткових частин, таких як заголовок S-ARP.

Поле type дозволяє розрізняти кілька типів повідомлень:

— підписане ARP (лише відповідь);

— керування відкритими ключами (запит/відповідь);

— синхронізація часу (запит/відповідь).

Підписаними повідомленнями обмінюються лише хости локальної мережі. Інші типи повідомлень передаються виключно між хостом і Сервером довіри.

Поля siglen і datalen вказують довжину цифрового підпису та довжину корисних даних у S-ARP-навантаженні відповідно. Поле timestamp містить

значення локального S-ARP-годинника на момент формування пакета. Нарешті, поле signature містить хеш заголовків ARP і S-ARP.

Хост спершу перевіряє сертифікат, потім витягує публічний ключ відправника, перевіряє підпис і лише після цього довіряє отриманим IP-MAC зв’язкам. Така структура забезпечує повноцінну криптографічну цілісність ARP-даних.

На рівні реалізації S-ARP потребує внесення змін у мережевий стек. У експериментальних реалізаціях модуль S-ARP інтегрується у ядро Linux як окремий протокол поверх ARP. Оскільки кожен ARP-пакет вимагає криптографічної обробки, накладні витрати у S-ARP значно більші порівняно зі звичайним ARP [10].

Попри високий рівень безпеки, протокол S-ARP має низку недоліків. Використання цифрових підписів для кожного ARP-пакета створює значне навантаження на процесор, особливо в мережах із великою кількістю вузлів. Іншим недоліком є несумісність S-ARP зі деякими мережевими пристроями (мережеві принтери, IP-телефони), які не підтримують реалізацію протоколу, що унеможливлює його застосування у більшості корпоративних мереж. Саме через ці фактори S-ARP не був впроваджений як стандарт.

2.2 Визначення функціональних вимог до методу захисту та розробка структурної схеми методу захисту

Запропонований метод захисту від атак типу ARP-спуфінг поєднує криптографічні механізми автентифікації з поведінковим аналізом мережевого трафіку. Він повинен виконувати верифікацію легітимності ARP-повідомлень та одночасно виявляти аномальну активність, яка свідчить про спроби атаки на локальну комп’ютерну мережу.

Метод складається з двох частин: сервера довіри та клієнтського агенту. Клієнтський агент має у собі два модуля: криптографічний модуль та поведінковий модуль. Така структура дозволяє нівелювати як вразливості самого протоколу ARP, так і виявляти аномалії, пов'язані з незахищеними вузлами.

Структурну схему компонентів методу захисту подано у додатку В.

Сервер довіри займає центральне місце системи. Його призначення полягає у фіксації авторитетної інформації щодо мережевих вузлів, зокрема відповідності між IP-адресами, MAC-адресами та їхніми публічними криптографічними ключами. Така база даних виступає джерелом істини, на яке орієнтуються всі інші компоненти системи.

Сервер працює у захищеному сегменті мережі та надає інтерфейси (на базі HTTPS) для реєстрації нових вузлів, отримання публічних ключів та обробки сповіщень про можливі інциденти. Завдяки цьому він виконує роль довіреного центру, який підтверджує справжність походження ARP-повідомлень, запобігаючи підробленню цифрових підписів при перших контактах між хостами.

Кожний захищений хост у мережі оснащується клієнтським агентом. Цей агент формує криптографічну ідентичність пристрою, генерує пару відкритого та закритого ключів і передає відкритий ключ на сервер довіри разом із власними мережевими параметрами.

Основна функція агента полягає у перевірці достовірності ARP-трафіку. Він перехоплює вхідні та вихідні ARP-пакети до того, як вони потрапляють у системний ARP-кеш операційної системи. Власні ARP-відповіді агент підписує, а вхідні пакети перевіряє за підписом відправника. Якщо публічний ключ відправника невідомий, агент отримує його з сервера довіри. У випадку відсутності підпису або невідповідності цифрової перевірки пакет блокується, а інформація про інцидент повідомляється на сервер. Таким чином, агент забезпечує захист конкретної машини, не дозволяючи здійснити підміну відповідей у її ARP-кеші.

Функції криптографічного модуля полягають у гарантуванні автентичності відправника та цілісності переданих даних у кожному ARP-пакеті. Для оптимізації продуктивності та мінімізації навантаження на вузли мережі, метод використовує подвійний підхід до автентифікації. Він успадковує принципи S-ARP, але вдосконалює їх шляхом використання гібридної схеми та можливості

динамічної заміни сеансових ключів.

Під час першого встановлення довіри між двома хостами застосовується алгоритм асиметричної криптографії RSA. Цей етап слугує для безпечного обміну даними та встановлення спільного сеансового ключа. Для всіх подальших комунікацій між цими хостами, з метою зниження обчислювальної складності та затримок, використовується симетричний механізм, а саме HMAC. Пакет, що не проходить криптографічну перевірку (або не має підпису в середовищі, де він очікується), відкидається хостом-отримувачем, запобігаючи отруєнню його ARP-кешу.

Поведінковий модуль функціонує як система виявлення вторгнень, що працює паралельно з криптографічним модулем на випадок компрометації криптографічного підпису та для виявлення атак, що не націлені на конкретний хост. Його призначення — виявлення атак, які криптографічна підсистема може пропустити, зокрема, спрямованих на незахищені пристрої (IP-телефони, принтери, IoT) або атак типу ARP-flood.. Цей компонент додає додатковий рівень захисту від ситуацій, де зловмисник генерує нетипові патерни трафіку чи масово надсилає запити, не порушуючи криптографічну структуру пакета.

Цей модуль працює у нерозбірливому режимі, пасивно аналізуючи весь ARP-трафік у сегменті. Він шукає аномальну поведінку трафіку, таку як конфлікти IP та MAC, аномально висока частота ARP-запитів, або раптову зміну MAC-адреси для ключового вузла. При виявленні аномалії, підсистема логує інцидент, надсилає тривожне сповіщення на сервер довіри та ініціює активні контрзаходи у вигляді розсилки ARP-пакетів з коректними даними для відновлення цілісності мережі

Функції, які повинен виконувати модуль поведінкової детекції:

— постійний моніторинг мережевого трафіку в реальному часі для виявлення підозрілої активності, наприклад, раптової зміни MAC-адрес для критичних вузлів, таких як шлюз або аномально високої частоти ARP-запитів з одного джерела;

— сповіщення адміністраторів системи про потенційну атаку;

— детальне логування всіх виявлених інцидентів з метою подальшого аналізу;

— блокування атаки.

Механізм блокування повинен протидіяти спуфінгу шляхом відкидання всіх пакетів від джерела атаки та надсиланню пакетів з коректними даними.

Поведінкова підсистема повинна аналізувати багато індикаторів аби відрізняти легітимні події від зловмисної активності. Для неї доступні користувацькі налаштування порогів чутливості.

2.3 Обґрунтування вибору технологій розробки методу

Вибір стеку технологій для реалізації методу захисту ґрунтується на критеріях гнучкості розробки та доступності спеціалізованих бібліотек для мережевої взаємодії та криптографії.

2.3.1 Обґрунтування вибору систем керування базами даних

Для зберігання публічних ключів клієнтських агентів та інформації про відповідності IP-MAC адрес на сервері можуть застосовуватися різні типи систем керування базами даних. Серед найбільш поширених варіантів використовують реляційні СКБД, такі як PostgreSQL та MySQL/MariaDB.

PostgreSQL є однією з найбільш розвинених реляційних систем керування базами даних з відкритим вихідним кодом. Її архітектура розроблена таким чином, щоб забезпечити стабільність, надійність та безпеку зберігання інформації навіть у високонавантажених системах. PostgreSQL підтримує розширені можливості SQL, включаючи складні вкладені запити, віконні функції, транзакції з гарантією ACID, тригери, функції та збережені процедури, що дозволяє реалізовувати логіку обробки даних на стороні серверу. Важливою перевагою є підтримка різноманітних типів індексів і можливість створення власних типів даних та операторів, що робить систему гнучкою для спеціалізованих задач.

PostgreSQL добре масштабується як по вертикалі, так і по горизонталі, що

дозволяє застосовувати її у корпоративних системах, банківських та фінансових платформах, великих веб-сервісах, аналітичних сховищах та наукових дослідженнях, де важливо працювати з великими обсягами інформації та складними зв’язками між даними [11].

SQLite суттєво відрізняється за концепцією — це легка вбудована система керування базами даних, яка функціонує без виділеного серверного процесу. Усі дані зберігаються в одному файлі, що робить SQLite надзвичайно зручною для застосування у програмних продуктах, де важлива автономність і простота розгортання. Вона не потребує конфігурації чи адміністрування та може використовуватися без мережевої взаємодії, що значно спрощує її інтеграцію в мобільні застосунки, IoT-пристрої, десктопні програми, локальні утиліти або системи з обмеженими ресурсами.

Незважаючи на компактність, SQLite забезпечує підтримку транзакцій та гарантує цілісність даних, що дозволяє використовувати її для зберігання оперативної інформації, тимчасових файлів або структурованих локальних даних користувача [12].

Вибір SQLite у даному випадку зумовлений архітектурою сервера, який виконує роль централізованого сховища публічних ключів та метаданих, але не здійснює обробку великих масивів даних або інтенсивні конкурентні транзакції. Обсяг операцій запису та читання є відносно невеликим, а вимоги до швидкого розгортання і мінімізації залежностей мають пріоритет. Використання SQLite дозволяє зменшити складність системи, уникнути налаштування окремого серверного процесу СКБД, забезпечити портативність та спростити резервне копіювання, оскільки база представлена єдиним файлом. Таким чином, SQLite є оптимальним рішенням для компактної та автономної серверної компоненти з чітко визначеним і невеликим навантаженням.

2.3.2 Обгрунтування вибору криптографічних алгоритмів

Криптографічні алгоритми відіграють ключову роль у забезпеченні автентичності та цілісності переданих даних, особливо у відкритих або

потенційно небезпечних мережевих середовищах. У загальному випадку їх поділяють на симетричні та асиметричні. Симетричні алгоритми використовують один спільний ключ для шифрування та розшифрування. Такі алгоритми характеризуються високою швидкодією, що робить їх доцільними для тривалих або інтенсивних сеансів обміну даними. Проте виникає необхідність безпечного розповсюдження цього ключа між сторонами, що ускладнює їх застосування у середовищі без первинного захищеного каналу.

Асиметричні алгоритми базуються на використанні пари ключів: приватного (закритого) та публічного (відкритого). Серед найбільш відомих алгоритмів цього класу — RSA, DSA та алгоритми на основі еліптичних кривих (ECDSA, Ed25519). DSA забезпечує цифрове підписування, однак менш гнучкий у параметризації та гірше масштабується при збільшенні довжини ключа. Алгоритми на базі еліптичних кривих відзначаються меншою довжиною ключа при тій самій стійкості та високою продуктивністю, але потребують обережного вибору математичних параметрів для запобігання вразливостей і складніші у реалізації [13].

RSA є одним з основних алгоритмів асиметричної криптографії, що ґрунтується на складності факторизації великих чисел та використанні модульної арифметики. Алгоритм забезпечує захист інформації завдяки використанню двох ключів: публічного для шифрування та приватного для розшифрування (рисунок 2.2).

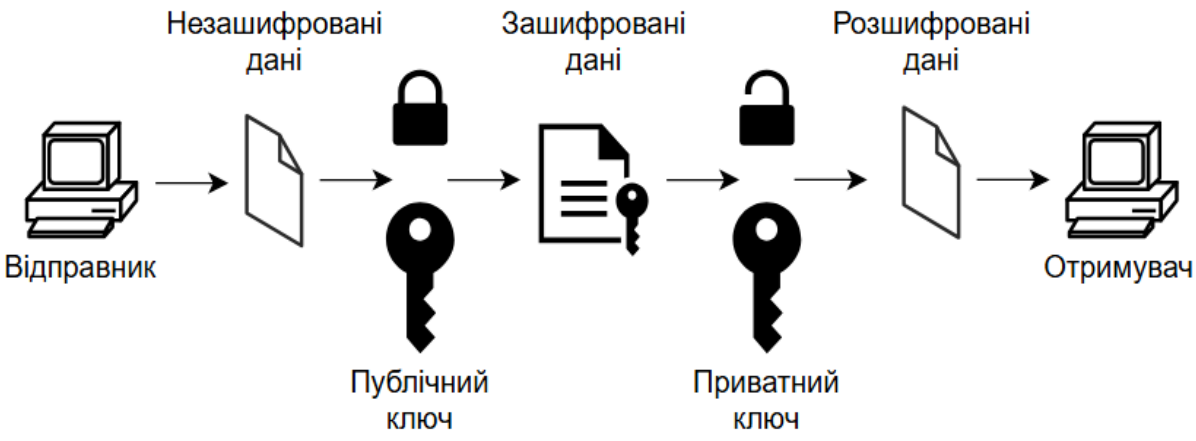

Рисунок 2.2 — Схема роботи алгоритму RSA

Його робота поділяється на три основні етапи: генерування ключів, шифрування та дешифрування. На етапі генерування ключів обираються два великі прості числа p та q, які зберігаються в таємниці. Обчислюється їхній добуток, що стає складовою як публічного, так і приватного ключа. Потім визначається значення функції Ейлера за формулою 2.1.

$$\Phi(n) = (p - 1) \cdot (q - 1). \quad (2.1)$$

Далі обирається показник шифрування e таким чином, щоб він був взаємно простим із $\Phi(n)$. Потім обчислюється показник дешифрування d, який є мультиплікативно оберненим до e за модулем $\Phi(n)$. Таким чином, формується пара ключів: публічний ключ (n, e) та приватний ключ (n, d).

Шифрування здійснюється за допомогою публічного ключа. Вихідне повідомлення M попередньо переводиться у числову форму, після чого формується шифротекст C за формулою 2.2.

$$C = M^e \bmod n. \quad (2.2)$$

Для дешифрування використовується приватний ключ: вихідне значення відновлюється за формулою 2.3.

$$M = C^d \bmod n. \quad (2.3)$$

Завдяки цьому лише власник приватного ключа може відновити повідомлення, що забезпечує конфіденційність переданої інформації.

Стійкість RSA ґрунтується на складності факторизації великого числа n, яке є добутком двох великих простих чисел. Хоча значення n і e є відкритими, для знаходження приватного ключа необхідно обчислити $\Phi(n)$, для чого мають бути відомими числа p і q [14].

HMAC (Hash-based Message Authentication Code) належить до симетричних

криптографічних механізмів автентифікації, що використовують криптографічну хеш-функцію разом із секретним ключем. Застосування HMAC забезпечує перевірку достовірності та цілісності даних при використанні спільно узгоджених секретів, що відрізняє його від методів, побудованих на цифрових підписах та асиметричних алгоритмах шифрування.

Основна ідея HMAC полягає у тому, що перед відправленням повідомлення до нього обчислюється хеш зі спеціальним включенням секретного ключа. Отриманий код автентифікації передається разом із повідомленням, а приймаюча сторона повторює аналогічне обчислення. Якщо обидві сторони використовують однаковий секретний ключ та той самий алгоритм хешування, а отримане та обчислене значення збігаються, повідомлення вважається достовірним та незмінним (рисунок 2.3).

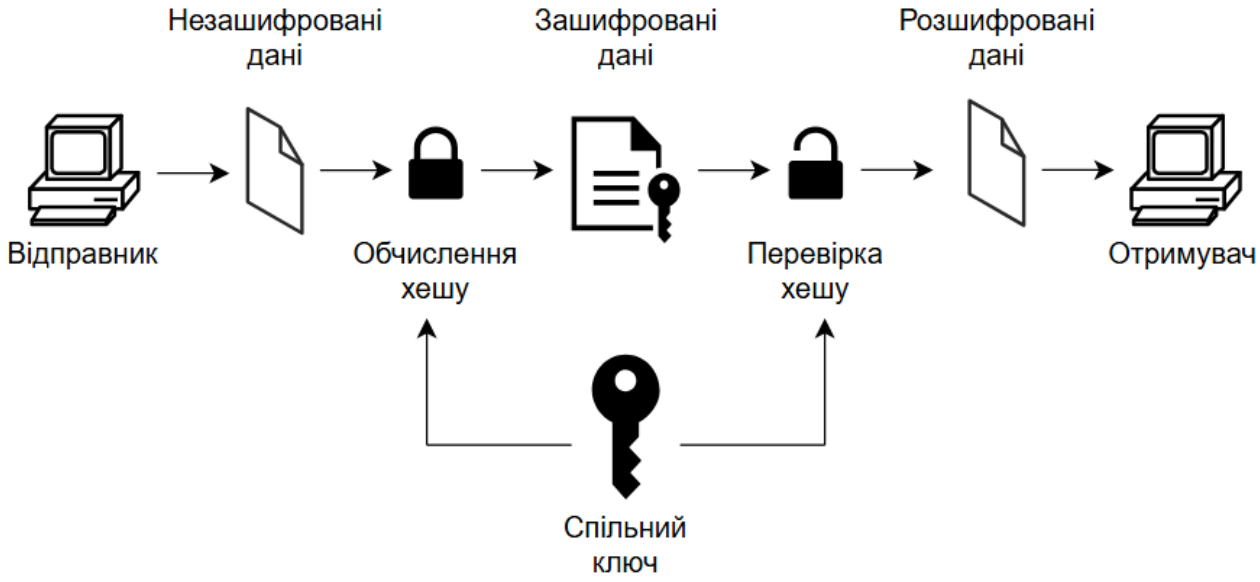

Рисунок 2.3 — Алгоритм HMAC

HMAC використовує два основні компоненти. Перший — секретний ключ, який повинен бути відомий обмеженому колу сторін та не підлягати розголошенню. Другий — криптографічна хеш-функція, що може бути реалізована на основі алгоритмів MD5, SHA-1, SHA-256 або інших функцій. Важливим аспектом безпеки є узгодження як секретного ключа, так і вибраного хеш-алгоритму, оскільки від цього залежить стійкість схеми до атак.

Результат обробки за HMAC є одностороннім та необернеником, тому зловмисник, який перехопив зашифроване значення, не може визначити початкові дані або ключ. Навіть при відомому тексті повідомлення відновлення секретного ключа залишається обчислювально складним завданням. Така властивість дозволяє використовувати HMAC у середовищах із недовірою до каналів зв'язку, де необхідно забезпечити, що дані не були змінені під час передавання.

Практична реалізація HMAC потребує наявності спільного секретного ключа та інструмента обчислення хеш-функцій. Цей ключ повинен зберігатися у безпечному середовищі, оскільки компрометація ключа дає можливість створювати фальшиві повідомлення від імені довіреної сторони. Тому системи, що використовують HMAC, часто включають додаткові механізми захисту ключового матеріалу.

HMAC застосовується у випадках, коли необхідне підтвердження цілісності даних у поєднанні з автентифікацією джерела. Це особливо актуально для організацій, що працюють із конфіденційною інформацією або персональними даними, а також для систем, що потребують захищеної передачі службових команд та протоколів взаємодії [15].

Для первинного встановлення довіри між хостами, обрано алгоритм RSA. Оскільки факторизація числа, що складається із сотень або тисяч бітів, є обчислювально складною задачею, алгоритм забезпечує високий рівень криптографічної стійкості. Саме тому RSA широко використовується в мережевих протоколах безпеки, електронному підписі, обміні ключами та у системах захисту комунікацій. Завдяки простоті реалізації та високій ефективності HMAC використано для подальшого обміну ARP-повідомленнями для вузлів, які встановили зв'язок.

2.4 Формування підписаного ARP-повідомлення

У стандартному протоколі ARP не передбачено механізму вбудованого криптографічного підпису, тому при реалізації захисту необхідно адаптувати

структуру пакета так, щоб додаткові дані не заважали роботі мережевого стеку операційної системи, але залишалися доступними для аналізу спеціальним агентом. Базовий ARP-пакет має фіксовану довжину 28 байт і містить інформацію про типи протоколів, довжини адрес та пари IP-MAC, які використовуються для оновлення ARP-кешу (рисунок 2.4).

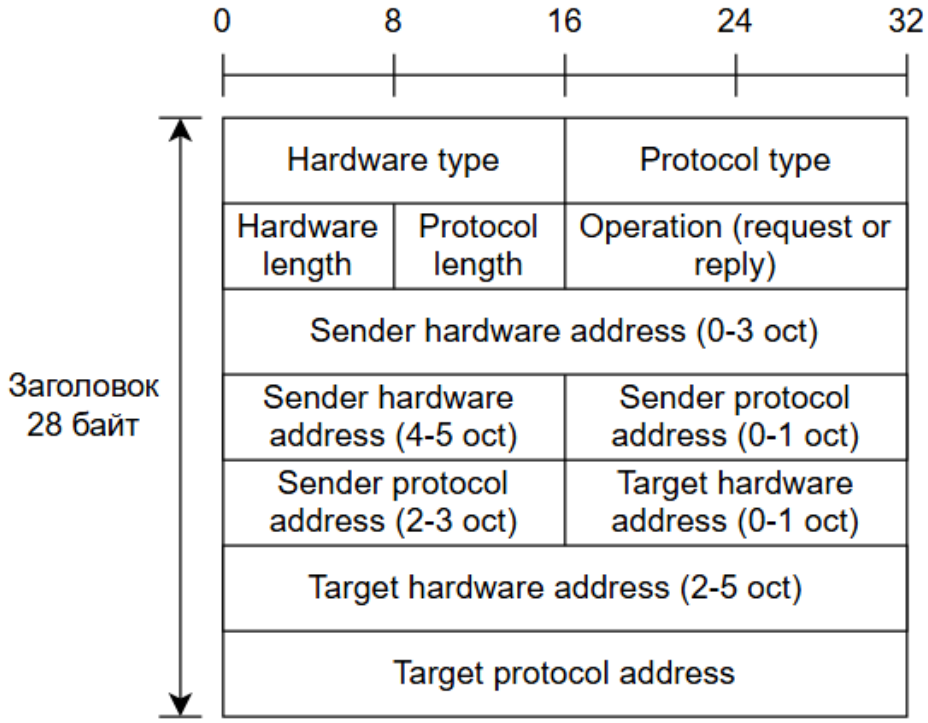

Рисунок 2.4 — Структура стандартного ARP-пакета

ARP-повідомлення містить кілька полів. Перше поле, Hardware Type (HTYPE), визначає тип фізичної мережі, у межах якої виконується ARP-запит або відповідь. У випадку Ethernet використовують значення 1. Це поле забезпечує універсальність ARP-протоколу. Друге поле, Protocol Type (PTYPE), вказує на протокол мережного рівня, для якого здійснюється встановлення відповідності адрес. У сучасних мережах майже завжди присутнє значення 0x0800, що позначає IPv4. Поля Hardware Address Length (HLEN) та Protocol Address Length (PLEN) визначають довжину апаратної та мережевої адрес відповідно. У стандартних Ethernet/IPv4-мережах HLEN дорівнює 6, що відповідає 48-бітній MAC-адресі, а PLEN дорівнює 4, що відповідає 32-бітній IPv4-адресі. Поле Operation (OPER) визначає тип ARP-повідомлення, що

передається. Значення 1 — позначає ARP-запит, і 2 — означає ARP-відповідь.

Поле Sender Hardware Address (SHA) містить MAC-адресу відправника, а Sender Protocol Address (SPA) — IP-адресу. Аналогічно, поля Target Hardware Address (THA) та Target Protocol Address (TPA) описують MAC- та IP-адреси цільового пристрою [16].

Для реалізації цифрового підпису до стандартного ARP-пакету не вносились зміни, але додано поле навантаження (payload), яке не впливає на базову обробку ARP ядром, але аналізується клієнтським агентом. Такий підхід забезпечує сумісність зі стандартним мережевим стеком, оскільки основна частина пакета не змінюється, а підпис розміщується наприкінці кадра. Перевагою такого підходу є часткова зворотна сумісність. Звичайні мережеві пристрої, які не мають встановленого агента, отримавши модифікований пакет, прочитають тільки стандартні 28 байт заголовка ARP. Але клієнтський агент не прийматиме непідписані пакети від незахищених пристроїв.

У полі навантаження розміщується криптографічний підпис, який може мати різний формат залежно від етапу взаємодії між вузлами. Під час першого встановлення довіри використовується підпис на основі RSA-2048, який займає 2048 біт — 256 байт, у сумі розмір підписаного пакету становитиме 284 байти (рисунок 2.5).

Рисунок 2.5 — Структура ARP-пакета з RSA-підписом

ARP з HMAC-SHA256 додає до повідомлення 256 біт, тобто 32 байти, сумарно — 60 байт (рисунок 2.6).

Рисунок 2.6 — Структура ARP-пакета з HMAC-підписом

Стандартний MTU (Maximum Transmission Unit) для Ethernet становить 1500 байт. Максимальний розмір пакету з RSA-підписом має розмір 298 байт, враховуючи заголовок Ethernet-кадру. Це значно менше встановленого ліміту, тому пакети не будуть фрагментуватися.

2.5 Розробка алгоритму криптографічного модулю

Функціонування криптографічної підсистеми базується на принципі примусової автентифікації кожного ARP-повідомлення до того, як воно буде допущене до обробки. Алгоритм роботи реалізовано через клієнтські агенти, що перехоплюють трафік канального рівня, та їх взаємодію з центральним сервером довіри.

Роботу модуля можна описати, розглянувши основні сценарії взаємодії у мережі. Під час першого запуску клієнтський агент на хості створює пару асиметричних RSA-ключів необхідної довжини. Приватний ключ зберігається локально у захищеному вигляді та ніколи не покидає меж пристрою, що унеможливлює його перехоплення або підміну. Публічний ключ, навпаки, передається на сервер довіри разом із відповідними мережевими атрибутами хоста. Сервер зберігає цю інформацію у базі даних, формуючи джерело даних для подальшої перевірки цифрових підписів. Усі звернення хостів щодо

отримання ключів здійснюються через цей сервер, що дозволяє виключити можливість підміни ключів у процесі обміну.

Алгоритм функціонування модулю цифрового підпису розпочинається з ініціалізації підсистеми перехоплення мережевого трафіку. Кожен пакет, що проходить через мережевий стек, перехоплюється та піддається структурному аналізу для визначення протоколу та типу операції. На цьому етапі алгоритм відфільтровує ARP-запити, які пропускаються без модифікацій, та обробляє виключно ARP-відповіді. Далі відбувається визначення напрямку руху пакету. У випадку обробки вихідного трафіку криптографічний модуль генерує цифровий підпис і відправляє пакет.

При обробці вхідного трафіку алгоритм першочергово перевіряє наявність поля цифрового підпису у структурі отриманого пакету. Відсутність підпису кваліфікується як порушення безпеки і призводить до блокування пакету, реєстрації інциденту в системному журналі та відправки тривожного сповіщення на сервер. Якщо підпис присутній, він відокремлюється від даних заголовка ARP для подальшої верифікації. Залежно від наявності сеансового ключа в локальному кеші, виконується або швидка перевірка HMAC, або, у разі першого контакту, ініціюється запит до сервера довіри для отримання публічного ключа відправника та подальшої верифікації RSA-підпису. Блок-схему алгоритму встановлення довіри подано у додатку Г.

Система перевіряє локальний кеш сеансів на наявність попередньо узгодженого симетричного ключа для хоста-отримувача. Якщо такий ключ знайдено, для забезпечення високої продуктивності генерується підпис на основі алгоритму HMAC-SHA256. Алгоритм обміну пакетами з використанням сеансових ключів наведено у блок-схемі у додатку Д.

Сеансовий HMAC-ключ генерується одним із хостів після успішної перевірки RSA-підпису і надсилається іншому хосту у зашифрованому вигляді, використовуючи його публічний RSA-ключ. Одержувач розшифровує ключ своїм приватним ключем і зберігає його локально. Таким чином формується спільний криптографічний секрет, відомий тільки двом сторонам. Цей ключ

використовується для підпису та перевірки наступних ARP-повідомлень за алгоритмом HMAC, що суттєво знижує навантаження на систему без втрати рівня довіри.

У разі успішної верифікації криптографічного підпису, алгоритм видаляє службові дані з тіла пакету, відновлюючи його оригінальний формат, та передає ARP-пакет мережевому стеку ОС для легітимного оновлення ARP-кешу. Якщо ж перевірка виявляє невідповідність підпису, використання відкликаного ключа або помилку цілісності даних, пакет відкидається, запобігаючи отруєнню кешу, а інформація про спробу атаки передається на сервер (додаток Е).

2.6 Розробка алгоритму роботи модуля поведінкової детекції

Модуль поведінкової детекції, або мережевий сенсор, виконує роль системи спостереження та реагування на аномалії в ARP-трафіку. Його робота ґрунтується на безперервному аналізі мережевих повідомлень у канальному рівні, що дозволяє своєчасно виявляти ознаки ARP-підміни чи навмисного перевантаження мережі. Сенсор отримує доступ до всього локального трафіку, а не лише того, який було адресовано безпосередньо йому. Завдяки цьому він може сформувати повну картину поточного стану мережі та контролювати критичні зв’язки між IP- і MAC-адресами. Блок-схема роботи алгоритму поведінкового модулю поданий у додатку Ж.

На початковому етапі сенсор формує власну базу стану, що відображає відповідність IP-адрес їхнім MAC-адресам. Принцип, що лежить в основі цього етапу, можна охарактеризувати як «довіра при першому використанні»: перше отримане ARP-повідомлення для певної адреси фіксується як еталон. Паралельно система відстежує часові характеристики потоку ARP-пакетів, що дає можливість контролювати частотність їх появи. Особлива увага приділяється критичним вузлам мережі, таким як шлюз та DNS-сервери, оскільки їх підміна має найбільш руйнівні наслідки.

Другий етап полягає в аналізі зібраних даних. Сенсор порівнює кожен новий ARP-пакет із своєю базою стану, перевіряючи, чи не змінилася MAC-

адреса для вже відомої IP-адреси. Якщо така невідповідність виявляється, система розцінює її як спробу ARP-спуфінгу. Окремо аналізується частота появи пакетів від кожної IP-адреси. Якщо кількість ARP-повідомлень перевищує встановлений поріг за певний часовий інтервал, це розглядається як спроба атаки ARP-flood, що може бути спрямоване на перевантаження мережевих пристроїв або сенсора. У випадку відсутності аномалій база стану оновлюється відповідно до реального стану трафіку.

Коли сенсор виявляє атаку, система переходить до фази активного реагування. Першим кроком є фіксація інциденту та інформування Сервера Довіри, що забезпечує централізоване логування та можливість оповіщення адміністратора. Далі сенсор переходить до відновлення коректного стану мережі. Для цього він генерує та розсилає ARP-пакети, які містять правильні відповідності IP- і MAC-адрес. Метою цих дій є перезапис помилкових або навмисно змінених записів у ARP-кешах інших вузлів, що дозволяє повернути мережу до коректного стану без необхідності втручання користувача.

Останнім етапом алгоритму є визначення моменту, коли атака справді припинилася. Сенсор продовжує спостерігати трафік і, якщо у встановлений проміжок часу не фіксується нових шкідливих ARP-повідомлень, активні контрзаходи припиняються, але моніторинг триває постійно. Якщо ж зловмисник поновлює спроби підміни або флуду, сенсор відразу повертається до стадії реагування. Таким чином, модуль забезпечує безперервний захист мережі, поєднуючи виявлення, відновлення та стійке подальше спостереження.

2.7 Розробка алгоритму роботи сервера довіри та бази даних

Сервер довіри виступає центральним джерелом інформації про легітимні вузли мережі та використовується для підтвердження їхньої автентичності. Сервер працює у режимі пасивного очікування запитів і не ініціює зв’язків самостійно. Комунікація з іншими компонентами завжди відбувається з боку агентів, що знижує ризик неконтрольованого розповсюдження даних. Передача даних виконується по HTTPS, що виключає можливість підслуховування або

підміни переданої інформації.

Основою роботи сервера становить база даних, у якій зберігається повна інформація про зареєстровані хости. Сервер не покладається на тимчасові структури в оперативній пам’яті та не формує власних припущень щодо стану мережі. Уся актуальна інформація зберігається у вигляді записів, що описують вузол за IP-адресою, MAC-адресою та його публічним криптографічним ключем. Додатково фіксується час останньої активності. Така модель дозволяє підтримувати стан системи та робить сервер незалежним від тривалості своєї роботи або перезавантажень.

Публічні ключі не генеруються сервером. Відповідальність за створення криптографічної пари повністю лежить на агенті, який працює на вузлі. Приватний ключ не повинен залишати меж комп’ютера, який його використовує для підписування ARP-повідомлень. Сервер лише зберігає публічні ключі у форматі, що дозволяє видавати їх іншим учасникам при необхідності. Такий підхід можна розглядати як аналог камери схову, де сервер не має можливості відкрити сейф, а лише гарантує його наявність та незмінність.

Реєстрація нового хоста виконується під час першого запуску його агента. Сервер перевіряє отримані IP- та MAC-адреси, щоб унеможливити дублювання або спроби видати себе за інший вузол. У випадку повторної реєстрації система повертає повідомлення про конфлікт, що не дозволяє зловмиснику витіснити запис законного пристрою. Після успішної верифікації дані вносяться до бази, і вузол вважається довіреним.

Коли один вузол потребує перевірити отримане ARP-повідомлення, він звертається до сервера з запитом на отримання публічного ключа відправника. Сервер у відповідь надає ключ разом з MAC-адресою, яка закріплена за відповідною IP-адресою. Це дозволяє агенту переконатися в автентичності пакету та виявити підміну без необхідності довіряти самому ARP-протоколу. Таким чином, у мережі з’являється додатковий рівень контролю, не залежний від самого принципу роботи ARP.

Для зберігання даних серверу застосовується одна таблиця hosts, яка

виконує роль центрального сховища достовірних даних про кожного зареєстрованого учасника мережі.

Першим полем таблиці є id, унікальний внутрішній ідентифікатор, що створюється автоматично та використовується виключно для внутрішнього керування записами.

Основою для пошуку й ідентифікації вузлів є поле ip_address, яке зберігає IP-адресу хоста та має обмеження унікальності. Це гарантує відсутність дублювання адрес. Поле mac_address також має унікальні та обов'язкові обмеження. MAC-адреса виступає додатковим фактором підтвердження походження пристрою, дозволяючи запобігати випадкам подвійної реєстрації одного й того самого хоста або навмисного дублювання записів.

Поле public_key містить публічний RSA-ключ, представлений у форматі PEM. Це поле необхідне для перевірки цифрових підписів отриманих ARP-пакетів. Саме за цим полем сервер надає відповідь на запит GET.

Додаткову інформаційну роль відіграють поля часових міток registered_at та last_seen. Перше фіксує момент реєстрації вузла, забезпечуючи можливість подальшого аудиту та аналізу поведінки мережі. Друге може використовуватися для реалізації механізмів періодичного очищення застарілих записів або моніторингу активності хостів.

Додавання нового хоста зводиться до виконання INSERT-запиту, тоді як перевірка його ключа здійснюється через SELECT-запит за IP-адресою. Така модель забезпечує мінімальні затримки при обробці запитів і не створює зайвого навантаження на сервер, що є важливим у контексті високочастотних ARP-операцій.

Окремим елементом є система фіксації повідомлень про підозрілу активність. Сервер не аналізує ці повідомлення, а лише приймає та зберігає їх як журнальні записи. Це дозволяє будувати поведінкову аналітику на стороні інших компонентів і виявляти складніші атаки, що виходять за межі простої підміни MAC-адрес.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ЗАХИСТУ ВІД ARP-СПУФІНГУ

3.1 Вибір мови програмування

Для реалізації методу захисту ARP можуть застосовуватися різні мови програмування, кожна з яких має власні особливості щодо продуктивності, зручності розробки та доступності бібліотек.

C++ — мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Базується на мові C. [17].

C++ забезпечує прямий доступ до системних викликів і raw-сокетів, що дозволяє створювати високопродуктивні рішення з мінімальними накладними витратами. Однак робота з пам’яттю у вимагає особливої обережності, оскільки помилки керування пам’яттю здатні призвести до критичних вразливостей безпеки.

Java — об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle). Програми Java зазвичай компілюються в спеціальний байт-код, тому вони можуть працювати на будь-якій віртуальній Java-машині (JVM) незалежно від комп'ютерної архітектури. Програмне забезпечення Java дозволяє грати в мережеві ігри, спілкуватися з людьми по всьому світу, підраховувати відсотки за іпотечними кредитами та переглядати тривимірні зображення. Програми, написані мовою програмування Java та доступ до яких можна отримати з браузера, називаються аплетами. Великі компанії також використовують Java для розробки додатків та систем електронної комерції.

Універсальність, ефективність, портативність платформ та безпека технології Java роблять цю технологію ідеальним вибором для мережевих обчислень [18].

Мова Go — компільована, суворо типізована, Open Source мова програмування. Go можна розглядати як сучасний компроміс між швидкістю

виконання та простотою створення мережевих сервісів.

Мова має вбудовані засоби конкурентності, розвинені стандартні бібліотеки для роботи з мережевими протоколами та досить просту модель компіляції та деплойменту. Це робить її зручною для реалізації багатопотокових сенсорів або серверних модулів. Водночас, бібліотека засобів для гнучкого аналізу та конструювання «raw» мережевих пакетів у Go є менш розвиненою, ніж у мов, орієнтованих на швидке прототипування [19].

Мова Rust — це мова програмування загального призначення, яка поєднує в собі швидкість і ефективність C++ з безпекою пам’яті та продуктивністю мови високого рівня. Мова добре підходить для створення безпечних і високопродуктивних модулів, які працюють у середовищах з підвищеними вимогами до надійності.

Rust дозволяє уникати багатьох типових помилок пам’яті завдяки механізмам контролю володіння ресурсами та статичної типізації. Недоліком є відносно стрімка крива навчання та збільшений час розробки при створенні складних прототипів або дослідницьких рішень [20].

Для розробки методу захисту від ARP-спуфінгу обрано мову програмування Python, оскільки вона забезпечує усіма зручними засобами для роботи з мережевими пакетами, криптографічні бібліотеки та можливість оперативного модифікування алгоритмів аналізу без значних витрат на проєктування системи.

Порівняно з мовами C++ або Java, Python характеризується більш лаконічним та інтуїтивним синтаксисом, що зменшує обсяг коду для реалізації типових операцій. Це дозволяє зосередитися на логіці завдання, а не на формальностях синтаксису.

Однією з основних особливостей Python є динамічна типізація. На відміну від статично типізованих мов, де типи змінних визначаються на етапі компіляції й залишаються незмінними, Python дозволяє гнучко працювати з типами під час виконання програми. Такий підхід спрощує експериментування з даними та прискорює процес розробки. Ця властивість особливо актуальна для систем, у

яких потрібно обробляти неоднорідні або змінні за структурою дані.

Додатковою перевагою є зручність тестування і розгортання програм. Код, написаний на Python, зазвичай легко читати та модифікувати, що зменшує витрати часу на аналіз, супровід та налагодження.

Python також вирізняється великою кількістю бібліотек, що підтримують різноманітні напрямки розробки. Наявність спеціалізованих бібліотек, таких як Scapy для перехоплення, створення та обробки мережевих пакетів та cryptography для реалізації сучасних криптографічних алгоритмів, значно спрощує розробку системи поведінкової детекції.

Python дозволяє швидко змінювати логіку аналізу, порівнювати різні методи визначення аномалій та легко інтегрувати додаткові компоненти, зокрема модулі журналювання, взаємодію з API або формування звітів. Недоліком є нижча продуктивність у порівнянні з компільованими мовами, однак у завданнях моніторингу локального сегмента мережі та обробки ARP-трафіку цей фактор зазвичай не є критичним, особливо за умови оптимального використання бібліотек [21].

3.2 Початкове налаштування та встановлення залежностей

Для забезпечення коректної взаємодії між компонентами методу необхідно виконати початкове налаштування залежностей, встановити потрібні пакети та підготувати середовище виконання.

Сервер реалізовано на основі веб-фреймворку FastAPI, що дозволяє легко працювати з REST API, а агенти розроблені з використанням Python і бібліотек для криптографічних операцій.

Система передбачає роботу на Linux Ubuntu як для сервера, так і для клієнтських вузлів, оскільки ці системи мають прямий доступ до ARP-таблиць і мережевих інтерфейсів.

На кожному вузлі повинна бути встановлена актуальна версія Python із менеджером пакетів pip. Для роботи з мережевими інтерфейсами агенту потрібен доступ до команд ip, arp або iproute2. Встановити ці інструменти можна

виконавши команди, які подано у лістингу 3.1.

Лістинг 3.1 — Встановлення Python та інструментів для роботи з пакетами

```
sudo apt update
sudo apt install python3 python3-pip
sudo apt install net-tools iproute2
```

Сервер використовує FastAPI як основу для обробки HTTPS-запитів та SQLite для зберігання даних про хости. Бібліотека uvicorn виконує роль HTTP-сервера, а sqlalchemy використовується як ORM для роботи з базою даних (лістинг 3.2).

Лістинг 3.2 — Встановлення бібліотек для роботи сервера

```
pip install fastapi uvicorn sqlalchemy pydantic cryptography
```

Створення бази даних відбувається автоматично при першому запуску сервера. У базі будуть зберігатись дані про всі зареєстровані хости у локальній комп’ютерній мережі. Програмний код, який створює таблицю наведено у лістингу 3.3.

Лістинг 3.3 — Ініціалізація таблиці hosts

```
Base = declarative_base()
class Host(Base):
    __tablename__ = "hosts"
    id = Column(Integer, primary_key=True)
    ip_address = Column(String, unique=True, index=True)
    mac_address = Column(String, unique=True)
    public_key = Column(String)
    last_seen = Column(DateTime)
engine = create_engine("sqlite:///trust.db")
Base.metadata.create_all(engine)
```

Для підписування ARP-повідомлень та перевірки підписів, отриманих від інших вузлів використовується бібліотека cryptography, що забезпечує реалізацію RSA та HMAC. Модуль поведінкової аналітики використовує scapy для перехоплення ARP-пакетів. Встановлення всіх бібліотек виконується командами, які подано у лістингу 3.4.

Лістинг 3.4 — Встановлення залежностей для клієнта

```
pip install cryptography requests
sudo apt install python3-scapy
pip install scapy
```

3.3 Реалізація логіки роботи цифрового підпису

3.3.1 Генерація ключів та механізм розповсюдження

Для того, щоб забезпечити криптографічний захист ідентифікації хостів у мережі, необхідно створити механізм генерації пари ключів — приватного та публічного. Приватний ключ використовується виключно на стороні агента для підпису ARP-пакетів, тоді як публічний передається на Сервер Довіри для подальшої верифікації підписів іншими учасниками мережі. У лістингу 3.5 наведено приклад реалізації функції, яка виконує генерацію ключа до збереження ключів і підготовки публічного ключа разом із IP/MAC до відправлення на сервер довіри.

Лістинг 3.5 — Генерація RSA-ключів

```
KEY_SIZE = 2048
def generate_and_register_keys(interface="eth0"):
    print("Генерація RSA-ключів...")
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=KEY_SIZE,
    )
```

За допомогою функції rsa.generate_private_key() створюється нова пара ключів — приватний і відповідний йому публічний. Параметр public_exponent=65537 є стандартним вибором для більшості криптосистем, оскільки поєднує високу безпеку і швидкість операцій.

Публічний ключ, навпаки, може передаватися іншим компонентам системи. Під час початкової реєстрації хост відправляє свій публічний ключ на Сервер Довіри, де той зберігається у відповідності до IP-адреси (лістинг 3.6). Функція requests.post() надсилає POST-запит до Сервера Довіри за адресою, вказаною у змінній SERVER_URL. Параметр verify=False дозволяє виконати

запит навіть без перевірки SSL-сертифіката (для тестового середовища). У робочих умовах цей параметр необхідно замінити на перевірку дійсного сертифіката.

Якщо сервер повертає статус-код 200, реєстрація вважається успішною, і агент додається до таблиці hosts у базі даних сервера. В інших випадках виводиться код помилки та текст відповіді для діагностики.

Лістинг 3.6 — Реєстрація агента на сервері

```
ip_addr = get_if_addr(interface)
mac_addr = get_if_hwaddr(interface)
print(f"Реєстрація агента {ip_addr} на сервері...")
registration_data = {"ip_address": ip_addr, "mac_address":
mac_addr, "public_key": public_pem.decode('utf-8')}
try:
    response = requests.post(f"{SERVER_URL}/register", json =
registration_data, verify=False)
    if response.status_code == 200:
        print("Успішна реєстрація на Сервері Довіри.")
    else:
        print(f"Помилка реєстрації {response.status_code}")
except requests.exceptions.RequestException as e:
    print(f" Помилка підключення до сервера: {e}")
```

Коли інший агент вперше отримує ARP-пакет, підписаний цим хостом, він звертається до Сервера Довіри по захищеному каналу HTTPS та отримує відповідний публічний ключ через запит формату GET.

Для того, щоб створити унікальний криптографічно стійкий ключ сеансу, який буде використовуватися двома хостами після встановлення довіри, застосовується системна функція os.urandom(). Вона генерує послідовність випадкових байтів, які неможливо передбачити (лістинг 3.7).

Лістинг 3.7 — Генерація HMAC-ключа

```
import os
def generate_hmac_key():
    return os.urandom(32)
```

Згенерований ключ зберігається в оперативній пам’яті агента і не має потрапляти на диск або в лог-файли. Його життєвий цикл обмежений часом дії

сеансу між двома хостами.

Найскладнішим є передача HMAC-ключа іншому хосту, адже його потрібно відправити по мережі, але водночас не допустити перехоплення. Передавати ключ у відкритому вигляді не можна — це одразу зруйнує модель довіри. Тому перед відправкою його потрібно зашифрувати (лістинг 3.8).

Лістинг 3.8 — Шифрування HMAC-ключа перед відправленням

```
def encrypt_hmac_key(receiver_public_key, hmac_key):
    encrypted_key = receiver_public_key.encrypt( hmac_key,
padding.OAEP( mgf = padding.MGF1(algorithm=hashes.SHA256()),
algorithm=hashes.SHA256(), label=None))
    return encrypted_key
```

3.3.2 Перехоплення ARP-пакетів та інтеграція з ядром

Для того щоб агенти могли перехоплювати та аналізувати ARP-пакети у режимі реального часу, необхідно організувати зв’язок між мережевим стеком ядра Linux і користувацьким простором, де виконується код програми. Цього досягають за допомогою iptables у поєднанні з модулем NetfilterQueue.

Підсистема Netfilter у Linux дозволяє встановлювати правила обробки пакетів, а iptables виступає інтерфейсом для їх конфігурації. За допомогою спеціальної цілі NFQUEUE можна перенаправити певні пакети в чергу користувацького простору, де вони будуть оброблені Python-скриптом із використанням бібліотеки netfilterqueue. Саме там виконується логіка підпису, перевірки, фільтрації та прийняття рішень щодо кожного ARP-пакета (лістинг 3.9 та лістинг 3.10).

Лістинг 3.9 — Налаштування правил ядра та створення черги

```
sudo iptables -I INPUT -p arp -j NFQUEUE --queue-num 1
sudo iptables -I OUTPUT -p arp -j NFQUEUE --queue-num 1
```

Створюються два правила:

— INPUT — усі вхідні ARP-пакети направляються до черги номер 1;

— OUTPUT — усі вихідні ARP-пакети також прямують до цієї черги.

Таким чином, програма отримує повний контроль над обробкою ARP-трафіку як у напрямку “вхід”, так і “вихід”.

Лістинг 3.10 — Створення черги NetfilterQueue та прив’язка обробника

```
from netfilterqueue import NetfilterQueue
def process_packet(pkt):
    pass
nfq = NetfilterQueue()
nfq.bind(1, process_packet)
nfq.run()
```

У цьому коді імпортується модуль NetfilterQueue, який дозволяє працювати з мережевими чергами, створеними iptables. Оголошується функція process_packet(pkt), яка є зворотним викликом (callback), що буде викликаний для кожного перехопленого пакета. Усередині цієї функції реалізується логіка перевірки підписів і прийняття рішення (наприклад, pkt.accept() або pkt.drop()). Створюється екземпляр NetfilterQueue(), який прив’язується до черги з номером 1 (що відповідає налаштуванню iptables).

Виклик nfq.run() запускає нескінченний цикл обробки, у якому кожен перехоплений ARP-пакет передається функції process_packet, де описана основна логіка.

Вихідні ARP-відповіді потребують підпису перед відправленням. Якщо між вузлами вже встановлено HMAC-сеанс, додається HMAC-підпис. Якщо це перший контакт, підпис формується за допомогою RSA.

Вхідні ARP-відповіді перевіряються на наявність правильного підпису. Спочатку перевіряється HMAC (якщо є активний ключ сеансу). Якщо ж ключ відсутній або підпис не збігається, виконується RSA-верифікація.

Фрагмент коду, що подано у лістингу И.1 додатка И реалізує головну логіку агента, який аналізує та підписує ARP-пакети, забезпечуючи захист від ARP-спуфінгу. Основна функція process_packet(packet) викликається для кожного перехопленого пакета, переданого з ядра Linux через NetfilterQueue. Вона визначає тип пакета (вихідний чи вхідний ARP-відповідь) і відповідно або підписує його, або перевіряє підпис.

На початку відбувається імпорт необхідних бібліотек. Бібліотеки hmac та hashlib використовуються для обчислення HMAC-підписів, requests — для обміну з Сервером Довіри, scapy — для розбору ARP-пакетів, а cryptography — для операцій із ключами RSA. Після цього приватний ключ завантажується з локального файлу private_key.pem, який зберігається на хості агента. Також створюється словник SESSION_KEYS, де тимчасово зберігаються сеансові ключі HMAC для кожного перевіреного вузла.

Коли агент обробляє вихідний ARP-пакет (визначається за власною MAC-адресою), він додає до нього криптографічний підпис. Якщо між агентом і отримувачем уже встановлений сеансовий HMAC-ключ, підпис обчислюється за допомогою алгоритму HMAC-SHA256, що є швидким симетричним методом перевірки цілісності. Якщо ж сеансового ключа ще немає, використовується RSA-підпис, тобто створюється цифровий підпис вмісту пакета приватним ключем агента. Після цього підпис додається до поля навантаження пакета (arp_pkt.load), і модифікований пакет повертається в мережевий стек для відправлення.

Для вхідних ARP-пакетів логіка зворотна. Якщо пакет не містить підпису, він відкидається як потенційно шкідливий. Якщо підпис є, програма спершу визначає, чи має агент сеансовий HMAC-ключ для цього відправника. Якщо ключ знайдено, то обчислюється новий HMAC із вхідного пакета та порівнюється з отриманим підписом. Якщо підпис співпадає, то пакет вважається автентичним і пропускається далі, якщо ні — відкидається.

Якщо сеансовий ключ для відправника відсутній, агент звертається до Сервера Довіри через HTTPS-запит (GET /get_key/{source_ip}), щоб отримати публічний RSA-ключ цього вузла. Після отримання ключа виконується перевірка RSA-підпису. Якщо перевірка успішна, агент створює новий випадковий 256-бітний HMAC-ключ (os.urandom(32)) і зберігає його у словнику SESSION_KEYS. Надалі цей ключ використовується для швидшої перевірки пакетів між тими ж двома вузлами. У разі невдалої верифікації або помилок доступу до сервера пакет відкидається.

3.4 Реалізація логіки поведінкової детекції

3.4.1 Відстеження подій та прийняття рішень

Для розмежування легітимних і шкідливих подій система підтримує внутрішню карту відповідностей IP–MAC. Перший побачений зв’язок IP та MAC вважається коректним, якщо атака не почалася раніше за запуск агенту. У нормальному режимі відповідності змінюються рідко, зазвичай у результаті DHCP-переназначення. В свою чергу, спуфінг характеризується раптовою зміною MAC-адреси для вже відомого IP. Окремо система відстежує ARP-флуд — якщо кількість пакетів від одного джерела перевищує встановлений поріг за короткий проміжок часу, це інтерпретується як атака ARP Flooding. Особливо ретельно контролюється IP шлюзу.

Програмний модуль IDS реалізовано як клас BehavioralIDS, що отримує перехоплені пакети від бібліотеки Scapy та проводить аналіз відповідно до визначених критеріїв. Клас підтримує дві ключові структури даних: карту відповідностей ip_mac_map, у якій зберігається базовий стан мережі, та словник packet_history, де кожному IP відповідає черга часових міток останніх отриманих пакетів.

Уся логіка поведінкового модулю сконцентрована в класі BehavioralIDS, що виконує роль автономного сенсора. Його лістинг подано у лістингу И.2 додатку И. Конструктор формує дві внутрішні структури — карту відповідностей IP та MAC і історію часових міток пакетів, організовану через deque, яка дозволяє швидко видаляти застарілі записи. Як тільки кількість активних записів у вікні перевищує заданий поріг, система одразу реєструє атаку. Таким чином модуль здатний фіксувати раптові сплески трафіку навіть при невеликих часових інтервалах.

Метод process_packet викликається автоматично Scapy для кожного ARP-пакета. Спочатку виконується базова фільтрація, тобто аналізуються лише пакети, що містять ARP-рівень. Далі обробляються основні поля: IP-адреса, MAC-адреса та код операції. Аналізується частота надходжень — якщо пакет

стає частиною атаки переповненням, подальші кроки пропускаються, оскільки цього достатньо для класифікації атаки. Далі виконується перевірка на виявлення підміни MAC. Якщо система вже знає правильну MAC-адресу для певного IP, і в новому пакеті вміщено інший MAC, то подія визначається як ARP Spoofing, і формується відповідне сповіщення. Якщо ж IP зустрічається вперше, він додається до бази TOFU, що дозволяє IDS вивчити нормальний стан мережі.

3.4.2 Механізм блокування атак

Процес протидії складається з двох етапів. Після виявлення конфлікту система формує Gratuitous ARP — спеціальний «лікувальний» пакет, що містить справжню MAC-адресу відповідного IP. Це повідомлення транслюється широкомовно, тому кожен пристрій у локальному сегменті отримує оновлений запис. Оскільки ARP-кеш завжди оновлюється за останнім повідомленням, коригуючий пакет заміщує хибну інформацію зловмисника, чим фактично нейтралізує атаку та відновлює справжню відповідність IP–MAC.

Реалізовано основні компоненти механізму активної протидії на рівні ARP. Конструктор класу створює початкову базу знань про вузли мережі.

Метод deploy_countermeasures (лістинг 3.11) виконує функцію активного захисту. Він формує ARP-відповідь типу is-at, у якій джерелом вказується справжній IP-адресат, а MAC-адреса заповнюється коректним значенням, що зберігається у базі стану мережі. Надсилання пакета широкомовною адресою гарантує його доставку до всіх вузлів, а багаторазова передача створює високу ймовірність того, що «лікувальне» повідомлення буде оброблене перед тим, як зловмисник зможе повторно транслювати хибні дані.

Лістинг 3.11 — Метод deploy_countermeasures

```
def deploy_countermeasures(self, target_ip, correct_mac):
   pkt = Ether(dst="ff:ff:ff:ff:ff:ff") / ARP(
       op=2, psrc=target_ip, hwsrc=correct_mac,
       hwdst="ff:ff:ff:ff:ff:ff", pdst="ff:ff:ff:ff:ff:ff"
   )
   sendp(pkt, count=5, inter=0.1, verbose=False)
```

Метод analyze_traffic (лістинг 3.12) відповідає за аналіз кожного перехопленого ARP-пакета. За наявності запису у локальній базі система перевіряє відповідність MAC-адреси тому значенню, яке вважається еталонним. У випадку невідповідності ініціюється формування коригуючих пакетів, що миттєво повертають вузли до правильного стану ARP-кешів. Якщо ж IP-адреса зустрічається вперше, вона додається в базу й надалі використовується як частина моделі нормальної поведінки. Завершальним елементом є функція sniff, яка запускає безперервний моніторинг ARP-трафіку й передає кожний пакет на обробку модулю аналізу.

Лістинг 3.12 — Метод analyze_traffic

```
    def analyze_traffic(self, packet):
        if not packet.haslayer(ARP):
            return
        sender_ip = packet[ARP].psrc
        sender_mac = packet[ARP].hwsrc
        if sender_ip in self.network_state:
            if self.network_state[sender_ip] != sender_mac:
                self.deploy_countermeasures(sender_ip,
self.network_state[sender_ip])
        else:
            self.network_state[sender_ip] = sender_mac
```

3.4.3 Логування та відправка тривожних повідомлень на сервер

Локальний рівень логування реалізовано за допомогою стандартного модуля logging. Події записуються у окремий текстовий файл, який відіграє роль «чорної скриньки» системи. Запис інцидентів здійснюється з рівнем попередження WARNING або помилки ERROR, що дозволяє легко фільтрувати критичні події під час аналізу. Локальні логи зберігаються на хості незалежно від працездатності мережевої інфраструктури, що гарантує збереження інформації навіть при атаках типу DoS або мережевих відмовах.

Другим рівнем механізму є передача тривог на сервер. Для цього використовується HTTP-інтерфейс, а дані відправляються у форматі JSON, що забезпечує сумісність із будь-якими серверними системами обробки. Запит виконується через бібліотеку requests з урахуванням можливих мережевих

помилок. Використання конструкції try…except дозволяє уникати зависання агента: у разі недоступності сервера відбувається тільки локальна фіксація помилки, після чого робота захисного модуля продовжується у штатному режимі. Обмеження timeout гарантує, що блокування на відповіді сервера не вплине на обробку трафіку.

3.5 Розробка серверу довіри

Розроблений сервер довіри реалізує центральну точку керування безпекою, забезпечуючи облік зареєстрованих хостів, розповсюдження їхніх відкритих ключів та приймання тривожних повідомлень про виявлені аномалії у мережі. Його робота ґрунтується на поєднанні веб-фреймворку FastAPI,

Підсистема приймання HTTP-запитів реалізується за допомогою FastAPI, який забезпечує автоматичну валідацію отриманих даних та формування структурованих відповідей.

Основні функції сервера реалізовані трьома ендпоінтами FastAPI, які забезпечують життєвий цикл управління ключами та реагування на інциденти.

Ендпоінт /register є точкою входу для легітимних агентів безпеки. Агент викликає цей метод при першому запуску, передаючи свою IP-адресу, MAC-адресу та згенерований публічний RSA-ключ у форматі JSON. Сервер виконує перевірку цілісності: він запобігає спробам реєстрації, якщо в базі даних вже існує запис із цим самим IP або MAC. Це забезпечує унікальність кожного хоста.

Якщо перевірки успішні, створюється новий запис у таблиці hosts, ініціалізуються поля registered_at та last_seen, а запис фіксується у базі даних (db.commit). У випадку виявлення дубліката або помилки бази даних, повертається відповідний HTTP-код помилки.

Коли будь-який агент мережі бажає автентифікувати інший хост, він відправляє запит на ендпоінт /get_key, вказуючи IP-адресу цільового хоста. Сервер виконує пошук відповідного запису в базі даних. Якщо запис знайдено, сервер повертає публічний ключ хоста-власника, який використовується для верифікації цифрового підпису ARP-пакета. Як побічний ефект, цей метод

оновлює поле last_seen знайденого запису до поточного часу (datetime.utcnow()), що слугує механізмом аудиту та підтвердження активності хоста. Якщо запис не знайдено, повертається відповідь 404, що сигналізує агенту про неможливість перевірки довіри до цільового хоста.

Ендпоінт /alert є каналом зв'язку між сенсорами поведінкової детекції та центральним сервером. Він приймає JSON-об'єкти, що містять деталі інциденту: IP і MAC джерела та опис. Хоча в цьому коді тривога виводиться лише у консоль та системний лог, у розширеній версії саме тут мала б бути реалізована логіка запису інцидентів у окрему базу даних, активація автоматичних контрзаходів або надсилання сповіщень адміністратору мережі.

4 ТЕСТУВАННЯ МЕТОДУ НА МОДЕЛІ ЛОКАЛЬНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ

4.1 Вибір інструментів для віртуалізації

Для повноцінного тестування методу захисту потрібно побудувати модель локальної комп'ютерної мережі. У сучасній практиці моделювання та дослідження мережевих систем застосовуються програмні засоби для віртуалізації, які дозволяють створювати ізольовані середовища, розгортати різні операційні системи та відтворювати складні інфраструктури без потреби у фізичному обладнанні.

Віртуальна машина — це програмне середовище, яка створює імітацію реального комп'ютера. Віртуальні машини дозволяють запускати кілька операційних систем і застосунків на одному фізичному сервері, що дає змогу максимально ефективно використовувати його ресурси.

Одну віртуальну машину можна налаштувати для роботи з будь-якою операційною системою — Windows, Linux чи macOS — незалежно від того, яка ОС встановлена на основному комп'ютері. Операційну систему, яка запущена на віртуальній машині називають гостьовою [22].

Oracle VirtualBox — це кросплатформне програмне забезпечення для віртуалізації. Це означає, що він розширює можливості наявного комп'ютера, дозволяючи запускати кілька операційних систем одночасно, у межах кількох віртуальних машин.

Можливо встановлювати та запускати стільки віртуальних машин, скільки потрібно. Єдині практичні обмеження — це обсяг доступної пам'яті та дискового простору. Також є можливість зберігати, копіювати, експортувати та переносити віртуальні машини між різними хостами та у середовище Oracle Cloud Infrastructure.

VirtualBox може працювати на різних операційних системах хоста — від невеликих вбудованих пристроїв або настільних комп'ютерів до серверів у дата-центрах та хмарних середовищ. Крім того, у деяких випадках можливо

встановити застарілу операційну систему, яку фізичний комп’ютер вже не підтримує, оскільки у віртуальній машині можна гнучко налаштувати віртуальне обладнання відповідно до потреб [23].

VMware Workstation є програмним засобом для створення повноцінних віртуальних машин, що дозволяє запускати різні операційні системи та прикладні програми в ізольованому середовищі на одному фізичному комп’ютері. Продукт відображає апаратні ресурси хоста на ресурси віртуальної машини, завдяки чому кожна ВМ отримує власні процесор, пам’ять, дисковий простір і пристрої вводу-виводу, працюючи як повний еквівалент окремого фізичного ПК. Після встановлення VMware Workstation користувач може інсталювати у віртуальних машинах звичайні, немодифіковані операційні системи, а також дистанційно керувати ними через інтегрований веб-інтерфейс, який забезпечує доступ із будь-яких пристроїв без додаткових плагінів.

Перевагами VMware Workstation є можливість швидкого створення та відтворення тестових середовищ. Продукт також підтримує моделювання складних багаторівневих інфраструктур, забезпечує гнучке керування доступом, шифрування обмежених віртуальних машин та інтеграцію віртуальних застосунків у середовище робочого столу користувача. Це робить VMware Workstation потужним інструментом для навчальних, дослідницьких і професійних задач у сфері віртуалізації [24].

Для моделювання мережі обрано засіб віртуалізації VirtualBox. Це рішення забезпечує оптимальний баланс між функціональними можливостями, простотою використання та доступністю. VirtualBox є повністю безкоштовним, підтримує багато операційних систем, а мережеві режими дозволяють легко налаштовувати взаємодію між віртуальними хостами.

4.2 Вибір програмних засобів для середовища моделювання

Операційною системою для робочих хостів та сервера вибрано Ubuntu Desktop. Ubuntu — це популярний дистрибутив операційної системи Linux, який базується на відкритому програмному забезпеченні. Він є одним з

найпопулярніших і легко доступних варіантів Linux для користувачів по всьому світу. Ubuntu було розроблено з метою створення вільної, відкритої та простої операційної системи, яка могла би замінити існуючі комерційні операційні системи.

Одна з ключових особливостей Ubuntu — це його простий та інтуїтивно зрозумілий інтерфейс, що дозволяє новим користувачам швидко освоїти систему. Він має велику кількість програм і додатків, таких як офісні пакети, веб-браузери, мультимедійні інструменти та інше, що дозволяє використовувати комп'ютер у повному обсязі.

Крім того, Ubuntu активно підтримується спільнотою розробників, яка забезпечує оновлення, підтримку та безпеку системи. Користувачі можуть отримувати оновлення програмного забезпечення, включаючи виправлення безпеки і нові функції, що дозволяє системі залишатися актуальною [25].

Роль сервера ключів і централізованого сервісу атестації відведено Ubuntu Server. Ubuntu Server підходить для цієї задачі через великий набір серверних пакунків. Серверна версія полегшує автоматизацію, контроль доступу й інтеграцію з системами логування.

Для симуляції атакуючого вузла застосовано дистрибутив Kali Linux. Це безкоштовна операційна система заснована на дистрибутиві Debian GNU/Linux, призначена для фахівців з кібербезпеки, цифрової криміналістики та тестування на проникнення. Kali розроблений спеціально для тестування безпеки й надає уніфікований набір утиліт, що дозволяє швидко відтворювати різні типи ARP-атаки та сценарії вторгнення без додаткової інсталяції інструментів. Це спрощує експериментальну частину дослідження, дає змогу фокусуватися на оцінці ефективності захисту, а не на налаштуванні атакуючих засобів [26].

Для перехоплення та аналізу трафіку у випробуваннях використовуються Wireshark і tcpdump.

Wireshark — це інструмент для аналізу мережевого трафіку, який дозволяє детально переглядати та досліджувати дані, що передаються по мережі в реальному часі або збережені у вигляді файлу. Програма працює як «мережевий

аналізатор», тобто перехоплює пакети даних і розшифровує їх структуру згідно з підтримуваними протоколами, яких у Wireshark налічується кілька сотень.

Основна мета Wireshark — діагностика та аналіз роботи мережі. За його допомогою можна визначити причини затримок у передачі даних, некоректного маршрутизаційного налаштування, виявити спроби несанкціонованого доступу або ознаки мережевих атак. Програма дозволяє переглядати кожен пакет на різних рівнях мережевої моделі, від каналів передачі даних до застосункового рівня. Завдяки цьому користувач може побачити як загальну картину мережевої активності, так і конкретні деталі взаємодії між вузлами [27].

Tcpdump — це консольний інструмент для перехоплення та аналізу мережевого трафіку, який працює безпосередньо на рівні мережевого інтерфейсу. На відміну від графічних аналізаторів, tcpdump зосереджений на фіксації та відображенні пакетів у текстовому вигляді, що робить його зручним для швидкої діагностики та роботи на серверах або системах без графічного середовища. Програма дозволяє фільтрувати трафік за протоколами, адресами та портами, що дає можливість отримати потрібні дані [28].

Обидва інструменти широко використовуються адміністраторами мереж для виявлення і усунення проблем, інженерами з безпеки — для дослідження підозрілої активності та аналізу атак, а також викладачами та студентами для вивчення мережевих протоколів у практичному контексті.

4.3 Створення моделі локальної мережі

Для тестування роботи методу, потрібно чотири хости. Два з них будуть виконувати роль «жертви» атаки і один у ролі атакуючого. Четвертий вузол слугує сервером довіри. Для цього необхідно створити кілька віртуальних машин.

Схему локальної комп’ютерної мережі на основі якої буде здійснюватися тестування методу захисту подано на рисунку 4.1.

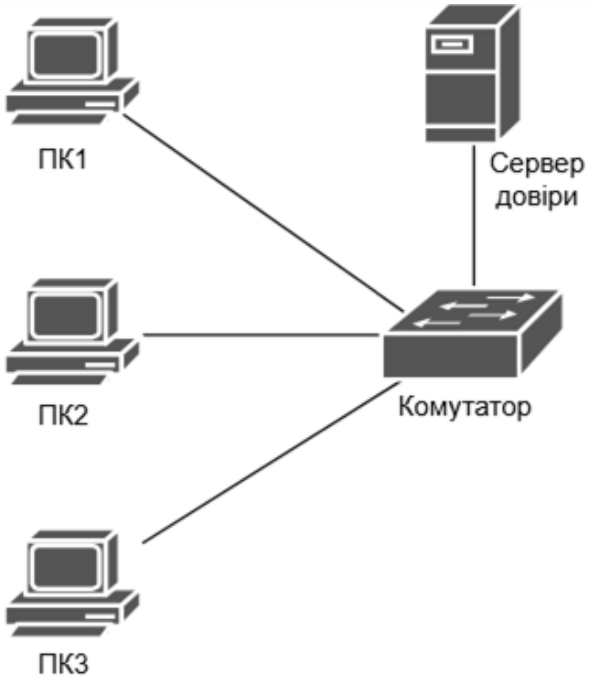

Рисунок 4.1 — Схема локальної комп’ютерної мережі

Щоб створити віртуальну машину у VirtualBox потрібно у головному вікні програми вибрати «Створити». Далі задається назва віртуальної машини та обирається тип і версія операційної системи, яку планується встановити. Після цього потрібно вказати ISO-образ або інший інсталяційний носій операційної системи, визначити обсяг оперативної пам’яті, що буде виділений для віртуальної машини, а також створити або обрати віртуальний диск (рисунок 4.2).

Після завершення початкового налаштування віртуальна машина з’являється у списку VirtualBox.

Об’єднання кількох віртуальних машин у спільну локальну мережу виконується шляхом налаштування їхніх мережевих інтерфейсів. У VirtualBox для цього використовується режим мережі Host-Only або Internal Network, залежно від того, чи потрібен доступ до хост-системи. Якщо потрібно створити окрему мережу лише між віртуальними машинами, то обирається режим Internal Network. Для цього у налаштуваннях кожної ВМ у розділі Network потрібно увімкнути один із мережевих адаптерів і встановити для нього тип підключення Internal Network, вказавши однакове ім’я мережі для всіх машин. У випадку Host-

Only Network віртуальні машини будуть бачити як одна одну, так і систему-хост, але матимуть ізольований доступ від зовнішньої мережі.

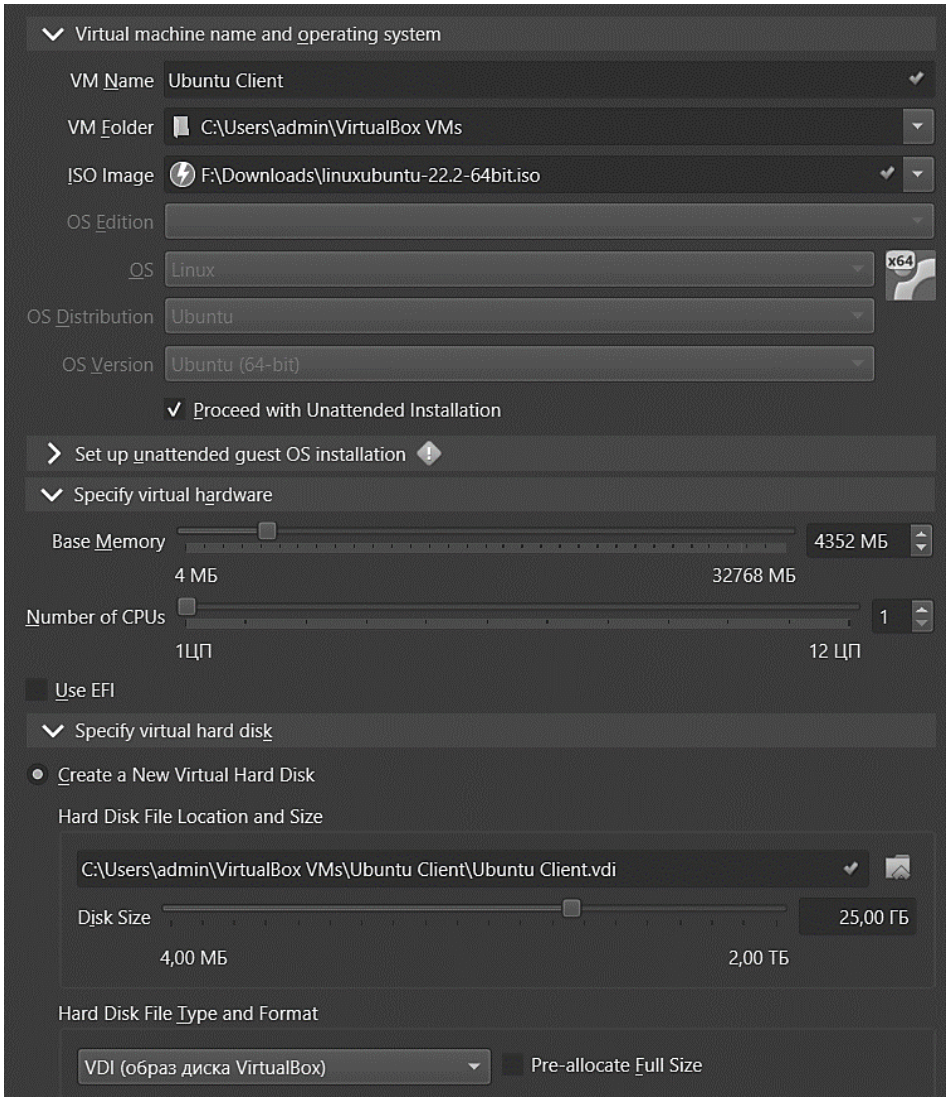

Рисунок 4.2 — Створення віртуальної машини

Після налаштування мережевого режиму необхідно встановити IP-адреси на віртуальних машинах, що належать до однієї підмережі. Обрано приватну мережу з IP-адресою 192.168.10.0/24, машинам призначаються адреси з діапазону 192.168.10.1 - 192.168.10.4 (рисунок 4.3). Після цього вузли зможуть обмінюватися пакетами безпосередньо, що дозволяє моделювати локальні мережні сценарії, досліджувати поведінку протоколів та тестувати мережеві сервіси в ізольованому середовищі.

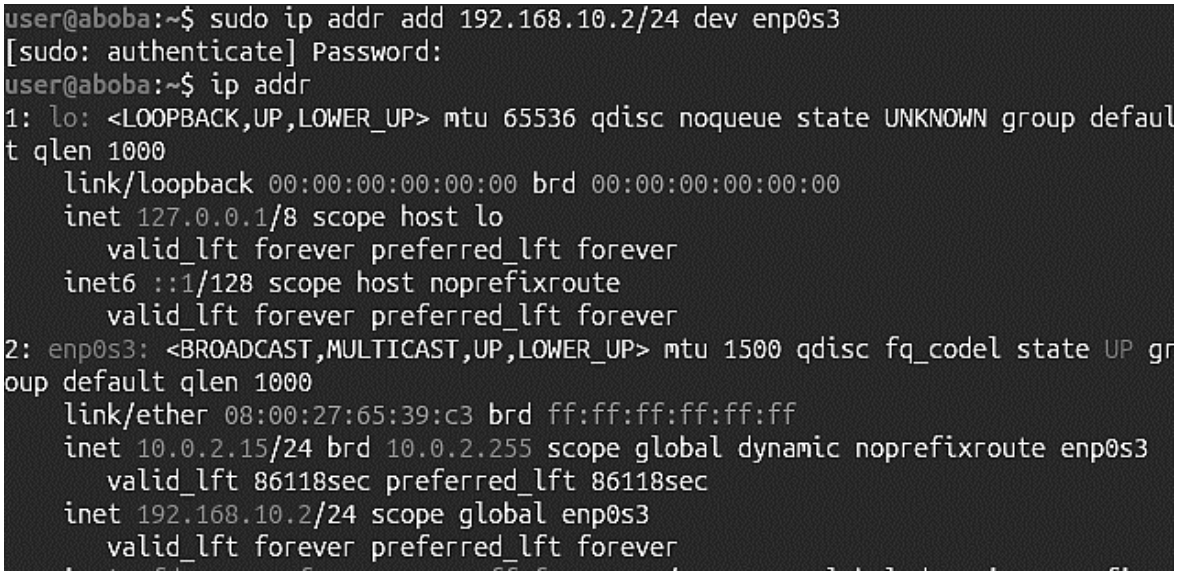

Рисунок 4.3 — Налаштування IP-адреси на хості

4.4 Моделювання атаки і тестування роботи методу захисту

Команда arpspoof — це інструмент, що дозволяє підміняти ARP-відповіді в локальній мережі з метою змінити відповідності IP та MAC-адрес, які зберігаються в ARP-кеші інших хостів. Команда може надсилати фальшиві ARP-повідомлення, унаслідок чого трафік, призначений для жертви або для шлюзу, може почати йти через машину того, хто виконав підміну, що відкриває можливості для перехоплення, моніторингу або зміни пакетів.

Команду arpspoof використовують фахівці з інформаційної безпеки під час тестування на проникнення і досліджень, для перевірки стійкості мережі та механізми виявлення атак. У контрольованому лабораторному середовищі він допомагає відтворити реальні сценарії ARP-спуфінгу і оцінити, чи коректно працюють системи виявлення та контрзаходи [28].

Щоб протестувати захист від ARP-спуфінгу, на атакуючій машині на системі запускається інструмент arpspoof з пакету dsniff і одночасно вмикається пересилання IP-пакетів, щоб трафік між жертвою і шлюзом проходив через атакуючий вузол.

Спочатку вмикається IP-forwarding, щоб машина-посередник могла пересилати пакети між вузлами.

Після цього запускається arpspoof у двох процесах, щоб отруїти кеш ARP як на жертві, так і на шлюзі; це дозволяє встановити двосторонній MITM (рисунок 4.4).

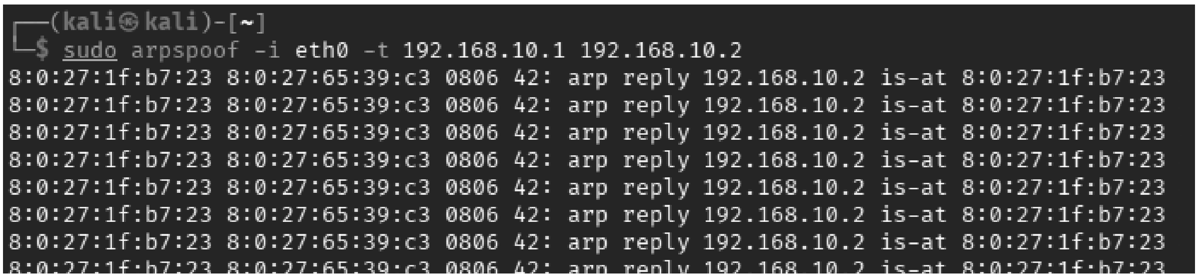

Рисунок 4.4 — Проведення атаки ARP-спуфінгу

У цих командах перший IP після -t — ціль (жертва або шлюз), а останній — адреса того, кого потрібно імітувати. Таким чином атакуючий повідомляє жертві, що його MAC відповідає за IP іншого пристрою, і повідомляє йому, що його MAC відповідає за IP жертви.

На машині-жертві через команду arp -a можна переглянути ARP-кеш і помітити, що MAC-адреса шлюзу змінилась на MAC-адресу атакуючої машини (рисунок 4.5).

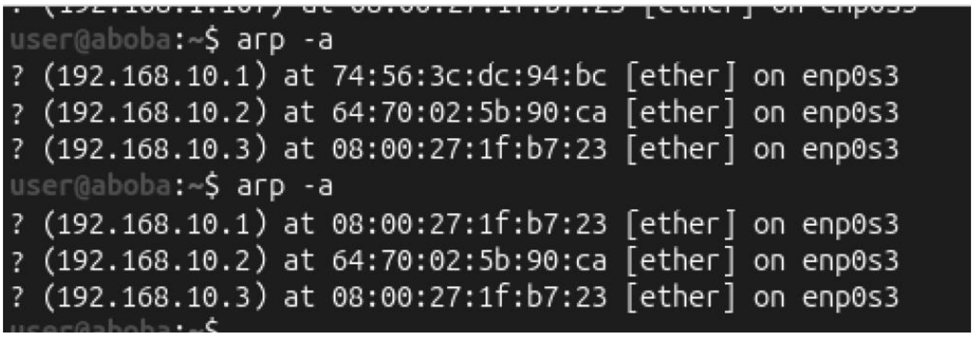

Рисунок 4.5 — Зміни у ARP-кеші після проведення атаки

На машині одночасно можна запускати захоплення пакетів командою tcpdump, щоб оцінити трафік, що проходить через неї (рисунок 4.5).

Щоб побачити відбитки ARP-повідомлень у Wireshark, потрібно увімкнути захоплення на інтерфейс атакуючого або жертви й застосувати фільтр відображення arp.

```
16:41:11.028689 ARP, Reply 192.168.10.2 is-at 08:00:27:1f:b7:23 (oui Unknown), length 46
16:41:13.030332 ARP, Reply 192.168.10.2 is-at 08:00:27:1f:b7:23 (oui Unknown), length 46
16:41:15.035035 ARP, Reply 192.168.10.2 is-at 08:00:27:1f:b7:23 (oui Unknown), length 46
16:41:17.040116 ARP, Reply 192.168.10.2 is-at 08:00:27:1f:b7:23 (oui Unknown), length 46
16:41:19.044602 ARP, Reply 192.168.10.2 is-at 08:00:27:1f:b7:23 (oui Unknown), length 46
```

Рисунок 4.5 — Перегляд захоплених пакетів у tcpdump

У списку пакетів потрібно шукати ARP-відповіді is-at від адреси, яку атакуючий підмінює — у полі Sender IP повинна стояти адреса іншої машини, а в полі Sender MAC — MAC атакуючої машини. Наявність таких is-at пакетів із MAC атакуючого означає, що жертва отримує підроблені ARP-відповіді (рисунок 4.6).

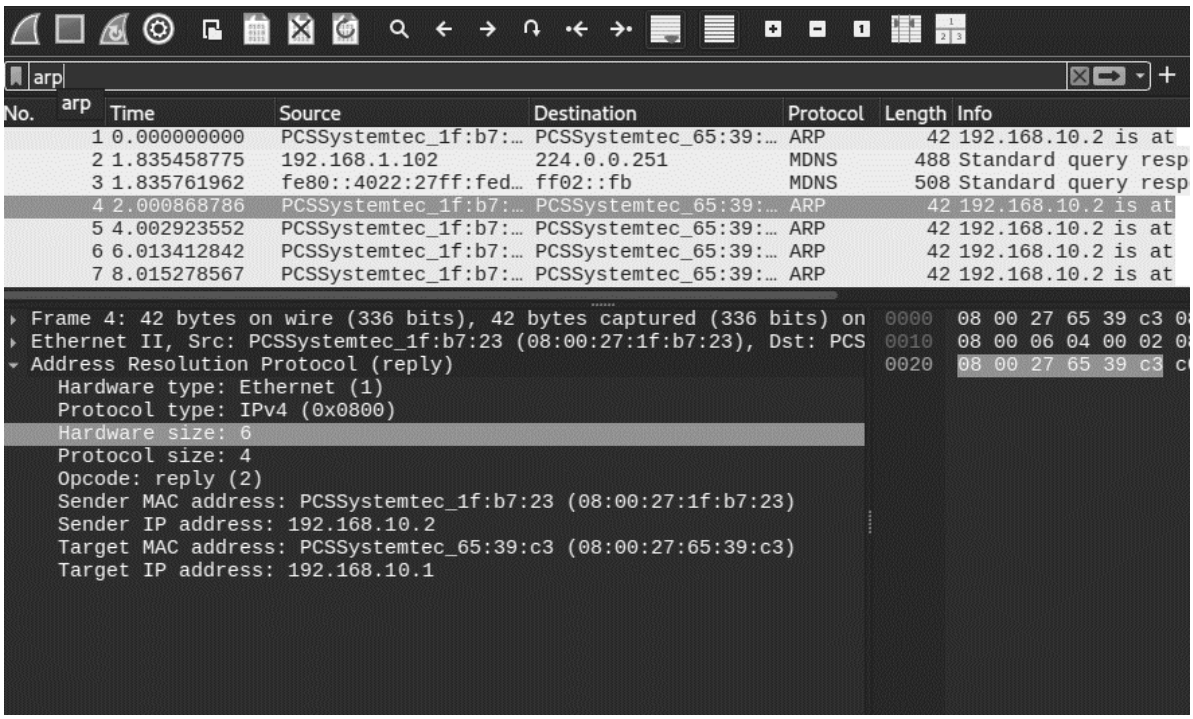

Рисунок 4.6 — Перегляд захоплених пакетів у Wireshark

Агент повинен виконуватись з підвищеними привілеями, оскільки взаємодіє з мережевим стеком і iptables. Після завершення роботи слід видалити додані правила iptables, щоб уникнути непередбачуваної маршрутизації пакетів. Тестування перевіряє сценарії першого контакту з новим хостом, відновлення після відкидання пакета, а також стійкість механізму встановлення сеансових ключів.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Основна мета проведення комерційного та технологічного аудиту є підвищення рівня захисту від ARP-спуфінгу у локальних комп'ютерних мережах та виявлення спроб підміни MAC-адрес і блокування їх у реальному часі шляхом додавання цифрового підпису до процесу обміну ARP-повідомленнями та виявлення і подальшого блокування підозрілого ARP-трафіку.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету з кафедри обчислювальної техніки: к.т.н., доцент Богомолов С. В., к.т.н., доцент Добровольстька Н. В., к.т.н., доцент Кадук О. В.

Для проведення технологічного аудиту було використано таблицю 4.1 [30] в якій за п'ятибальною шкалою, використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження табл. 5.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
01	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 — Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	2	2	2
2. Ринкові переваги (наявність аналогів)	2	1	2
3. Ринкові переваги (ціна продукту)	2	3	2
4. Ринкові переваги (технічні властивості)	1	2	2
5. Ринкові переваги (експлуатаційні витрати)	3	2	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	2
8. Практична здійсненність (наявність фахівців)	2	2	3
9. Практична здійсненність (наявність фінансів)	1	1	1
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	30	30	31
Середньоарифметична сума балів СБс	$\underline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{30+30+31}{3} = 30{,}33$		

Середньоарифметична оцінка, отримана на основі експертних висновків, становить 30,33 бали, і згідно з таблицею 5.2, це вказує на середній рівень комерційного потенціалу результатів проведених досліджень.

Результатом магістерської роботи є програмне забезпечення, яке визначає та блокує атаки типу ARP-спуфінг і дозволяє захищати ARP-повідомлення шляхом додавання цифрового підпису.

Шляхи реалізації цієї розробки передбачають впровадження у вигляді клієнт-серверної архітектури в межах локальної мережі (LAN). Серверна частина, що включає сервер довіри з базою даних ключів, розгортається на одному виділеному, статичному вузлі. Клієнтський агент, який містить криптографічний та поведінковий модулі встановлюється як фонова служба на кожній робочій станції, що потребує активного захисту.

Ця розробка буде реалізована як програмне забезпечення, яке не вимагає спеціалізованого дорогого мережевого обладнання. Вона функціонує на рівні операційної системи та мережевого трафіку, інтегруючись у вже існуючу інфраструктуру.

Основними користувачами даної системи є системні та мережеві адміністратори, а також фахівці з інформаційної безпеки. Вони будуть відповідати за розгортання сервера, встановлення агентів на критичні вузли та моніторинг журналу логів, який генерує система.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

Основна заробітна плата кожного із дослідників $З_о$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою 5.1:

$$З_0 = \frac{М}{Т_р} * t \text{ (грн)}, \quad (5.1)$$

де M — місячний посадовий оклад конкретного розробника (інженера,

дослідника, науковця тощо), грн.;

T_p — число робочих днів в місяці; приблизно T_p ~ 21…23 дні;

t — число робочих днів роботи дослідника.

Зведемо сумарні розрахунки до таблиці 5.4.

Таблиця 5.4 — Заробітна плата в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
1. Керівник проекту	17000	809,5	5	4048
2. Мережевий інженер	20000	952,4	19	18095
3. Розробник програмного забезпечення	35000	1666,7	16	26667
4. Тестувальник безпеки	25000	1190,5	10	11905
Всього				34048

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників (формула 5.2).

На даному підприємстві додаткова заробітна плата начисляється в розмірі 11% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (5.2)$$

$$З_д = 0{,}11 * (34048) = 3745{,}24 \text{ (грн)}$$

Нарахування на заробітну плату $Н_{зп}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.3):

(5.3)

$$Н_{ЗП} = (З_о + З_р + З_д) * \frac{\beta}{100} \text{ (грн)},$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

$З_р$ — основну заробітну плату робітників, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (34048 + 3745{,}24) * \frac{22}{100} = 8314{,}43 \text{ (грн)}$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою (5.4):

$$М = \sum_{i=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{i=1}^{n} В_j \cdot Ц_{вj}, \qquad (5.4)$$

де $Н_j$ — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

$Ц_j$ — вартість матеріалу j-го найменування, грн/кг;

$К_j$ — коефіцієнт транспортних витрат, ($К_j$ = 1,1 … 1,15);

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведені в таблицю 5.5.

Таблиця 5.5 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, шт	Вартість витраченого матеріалу, грн
Папір	185	1	185
Ручка	25	1	25
Блокнот	50	1	50
Флешка	260	1	260
З врахуванням коефіцієнта транспортування			572

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, можуть бути розраховані з використанням прямолінійного методу амортизації за формулою (5.5):

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.5)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки необхідно звести до таблиці 5.6.

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень (формула 5.6).

Таблиця 5.6 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	25000	2	3	1041,67
Всього				1041,67

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і}, \quad (5.6)$$

де W_{yt} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_і$ — коефіцієнт корисної дії обладнання, $\eta_і < 1$.

Для написання магістерської роботи використовується персональний комп’ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}5 \cdot 185 \cdot 12{,}69 \cdot 0{,}5}{0{,}8} = 733{,}64$$

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою (5.7):

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%}, \quad (5.7)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження».

$$В_{св} = 0,2 * (34048) = 6809,52$$

Накладні (загальновиробничі) витрати $В_{нзв}$, які розраховуються за формулою (5.8) охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо.

Накладні (загальновиробничі) витрати $В_{нзв}$ можна прийняти як (100…150)% від суми основної заробітної плати робітників, які виконували дану МКР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.8)$$

де $Н_{нзв}$ — норма нарахування за статтею «Інші витрати»

$$В_{нзв} = (34048) \cdot \frac{100}{100\%} = 34047,62 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКР:

В = 34048+3745,24+8314,43+572+1041,67+733,64+6809,52+34047,62 = 89311,74 грн

Прогнозування загальних втрат ЗВ на розробку та впровадження результатів виконаної МКР здійснюється за формулою (5.9):

$$ЗВ = \frac{В}{\eta}, \quad (5.9)$$

де η — коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт = 0,7.

Звідси:

$$ЗВ = \frac{89311,74}{0,7} = 127588,19 \text{ грн}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою (5.10).

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right), \qquad (5.10)$$

де $\Delta Ц_0$ — покращення основного оціночного показника від впровадження результатів розробки у даному році.

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

ρ — коефіцієнт, який враховує рентабельність продукту. ρ = 0,25;

υ — ставка податку на прибуток. У 2025 році – 18%.

Припустимо, що ціна зросте на 500 грн. Кількість одиниць реалізованої

продукції також збільшиться: протягом першого року на 40 шт., протягом другого року – на 50 шт., протягом третього року на 60 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до 80000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [500 \cdot 1 + (80000 + 500) \cdot 40] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 550146{,}74 \text{ грн.}$$

$$\Delta П_2 = [500 \cdot 1 + (80000 + 500) \cdot (40 + 50)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 1238138 \text{ грн.}$$

$$\Delta П_3 = [500 \cdot 1 + (80000 + 500) \cdot (40 + 50 + 60)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 2063230 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій *PV* (формула 5.11), які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \tag{5.11}$$

де $k_{\text{інв}}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}} = 2…5$).

$$PV = 2 \cdot 127588{,}19 = 255176{,}38$$

Розрахуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$ згідно наступної формули (5.12):

$$E_{абс} = (ПП - PV), \tag{5.12}$$

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн. (формула 5.13);

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.13}$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

T — період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t — період часу (в роках).

$$ПП = \frac{550146{,}74}{(1+0{,}2)^1} + \frac{1238138}{(1+0{,}2)^2} + \frac{2063230}{(1+0{,}2)^3} = 2517825{,}99 \text{ грн.}$$

$$Е_{абс} = (2517825{,}99 - 255176{,}38) = 2262649{,}6 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою (5.14):

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.14}$$

де $T_{ж}$ — життєвий цикл наукової розробки, роки.

$$\text{Е}_{\text{в}} = \sqrt[3]{1 + \frac{2262649{,}6}{255176{,}38}} - 1 = 1{,}66 = 1{,}66\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою (5.15):

$$\tau = d + f, \tag{5.15}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d= (0,14…0,2);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0{,}18+0{,}05=0{,}23$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою (5.16):

$$T_{ок} = \frac{1}{E_в} \tag{5.16}$$

$$\text{Т}_{\text{ок}} = \frac{1}{1{,}66} = 0{,}6 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки є доцільним.

Отже, результати здійсненого технологічного аудиту вказують на високий рівень комерційного потенціалу. У порівнянні з аналогічним виробом виявлено,

що нова розробка вищої якості і більш конкурентоспроможна, як з технічних, так і економічних позначень.

Вкладені інвестиції в даний проект окупляться через 6 місяців. Загальні витрати складають 127588,19 грн. Прогнозований прибуток за три роки 2517825,99 грн.

ВИСНОВКИ

В магістерській кваліфікаційній роботі було розроблено метод захисту від ARP-спуфінгу, який виявляє та блокує спроби підміни ARP-відповідей і забезпечує комплексний підхід до протидії атакам, які направлені на отруєння ARP-кешу. Метод базується на модифікації ARP протоколу S-ARP та покращує її шляхом додавання модулю поведінкової детекції та використанням симетричних ключів.

Проведений огляд сучасного стану питання та аналіз існуючих рішень підтвердив актуальність проблеми, а також продемонстрував недоліки поширених підходів, які переважно зосереджуються на частковому виявленні атаки або потребують додаткового обладнання. Проведений аналіз можливих атак показав вразливість протоколу ARP до атак типу Man-in-the-Middle та ARP-спуфінгу.

В роботі обґрунтовано вибір стеку технологій для реалізації програмних модулів криптографічного підпису та поведінкової детекції. Проведений аналіз криптографічних алгоритмів показав, що використання алгоритму RSA для обміну ключами та HMAC для контролю цілісності ARP-повідомлень дозволяє підвищити стійкість системи до підроблених мережевих пакетів. Для організації сервера довіри обрано SQLite, що спростило зберігання даних та забезпечило автономність рішення без потреби у розгортанні додаткових серверних систем.

Розроблено структурну схему методу захисту від ARP-спуфінгу, яка містить криптографічних та поведінковий модулі та описує їх взаємодію з сервером довіри. Також розроблено алгоритм роботи криптографічного модулю методу захисту та алгоритм роботи поведінкового модулю для виявлення і блокування підозрілого ARP-трафіку.

Виконано програмну реалізацію всіх компонентів методу захисту. Мова програмування Python, у поєднанні з бібліотеками Scapy та Cryptography, забезпечила можливість прямої роботи з пакетами та інтеграції криптографічних механізмів.

Проведене тестування підтвердило працездатність запропонованого методу. Моделювання мережі проведено з використанням програмного забезпечення для віртуалізації Oracle VirtualBox. Операційною системою для робочих станцій та серверу обрано дистрибутив Linux Ubuntu. Для симуляції атакуючого вузла застосовано дистрибутив Kali Linux. Для перехоплення та аналізу трафіку у випробуваннях використовуються Wireshark і tcpdump. Метод захисту виявляє сторонні спроби підміни ARP-таблиці, блокує шкідливі пакети та не впливає на роботу легітимного трафіку.

Усі завдання, поставлені в ході виконання роботи, реалізовано повністю, мета досягнута, а розроблене рішення може бути використане як окремий модуль безпеки у локальних комп’ютерних мережах або інтегроване до комплексних засобів захисту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. «Додавання цифрового підпису до протоколу ARP для запобігання ARP-spoofing» / К. О. Паламарчук, О. В. Войцеховська // Матеріали міжнародної науково-технічної конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)». Вінниця, 2025 р. [Електронний ресурс]. — Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/viewFile/26761/21971

2. What is ARP (Address resolution protocol)? [Електронний ресурс]. — Режим доступу до ресурсу: https://www.fortinet.com/uk/resources/cyberglossary/what-is-arp

3. Атаки типу Man-In-The-Middle: що треба знати кожному [Електронний ресурс]. — Режим доступу до ресурсу: https://www.imena.ua/blog/man-in-the-middle

4. What Is ARP Spoofing? [Електронний ресурс]. — Режим доступу до ресурсу: https://jumpcloud.com/it-index/what-is-arp-spoofing

5. What Is Port Security and Why Is It Important? [Електронний ресурс]. — Режим доступу до ресурсу: https://www.crowdsec.net/glossary/port-security

6. Understanding and Using Dynamic ARP Inspection (DAI) [Електронний ресурс]. — Режим доступу до ресурсу: https://www.juniper.net/documentation/us/en/software/junos/security-services/topics/topic-map/understanding-and-using-dai.html

7. Arpwatch – Monitor Ethernet Activity {IP and Mac Address} in Linux [Електронний ресурс]. — Режим доступу до ресурсу: https://www.tecmint.com/monitor-ethernet-activity-in-linux

8. XArp: Advanced ARP Spoofing Detection [Електронний ресурс]. — Режим доступу до ресурсу: https://filehippo.com/download_xarp-64-bit

9. Ettercap | Bugcrowd [Електронний ресурс]. — Режим доступу до ресурсу: https://www.bugcrowd.com/glossary/ettercap

10. D. Bruschi, A. Ornaghi, and E. Rosti, «S-ARP: A secure address resolution

protocol», in Proc. 19th Annu. Comput. Secur. Appl. Conf., 2003, pp. 66 –74.

11. What is PostgreSQL? [Електронний ресурс]. — Режим доступу до ресурсу: https://www.ibm.com/think/topics/postgresql

12. SQlite – визначення, основні характеристики та архітектура [Електронний ресурс]. — Режим доступу до ресурсу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite

13. Кібербезпека - Криптографія з відкритим ключем, різновидності алгоритм [Електронний ресурс]. — Режим доступу до ресурсу: https://sites.google.com/view/blog-ua/основні-поняття-криптографії-та-захисту-інформації/криптографія-з-відкритим-ключем-різновидності-алгоритм

14. Алгоритм шифрування RSA, види атак на нього. Реалізація мовою Python [Електронний ресурс]. — Режим доступу до ресурсу: https://dou.ua/forums/topic/43026/

15. HMAC (Hash-Based Message Authentication Codes) Definition [Електронний ресурс]. — Режим доступу до ресурсу: https://www.okta.com/identity-101/hmac

16. ARP Protocol Packet Format [Електронний ресурс]. — Режим доступу до ресурсу: https://www.geeksforgeeks.org/computer-networks/arp-protocol-packet-format

17. Вивчення C++: Особливості C++ [Електронний ресурс]. — Режим доступу до ресурсу: http://informatikpm11.blogspot.com/p/c-1-2.html

19. Let’s Go! Тисяча й одна причина полюбити Golang [Електронний ресурс]. — Режим доступу до ресурсу: https://campus.epam.ua/ua/blog/474

20. Що пишуть мовою програмування Rust? [Електронний ресурс]. — Режим доступу до ресурсу: https://foxminded.ua/rust-mova-prohramuvannia

21. Які особливості програмування мовою Python порівняно з іншими мовами програмування? [Електронний ресурс]. — Режим доступу до ресурсу: https://optima.study/blog/yaki-osoblyvosti-prohramuvannya-movoyu-python-porivnyano-z-inshymy-movamy-prohramuvannya

22. Що таке віртуальна машина та як вона працює [Електронний ресурс].

— Режим доступу до ресурсу: https://gigacloud.ua/articles/shho-take-virtualna-mashyna-ta-yak-vona-praczyuye

23. About Oracle VirtualBox [Електронний ресурс]. — Режим доступу до ресурсу: https://www.virtualbox.org/manual/topics/Introduction.html

24. DATASHEET VMware Workstation 9 [Електронний ресурс]. — Режим доступу до ресурсу: http://www.lojati.com.br/Customer/Datasheets/WS-3G-SSS-C.pdf

25. Що таке Ubuntu? Для чого використовують? [Електронний ресурс]. — Режим доступу до ресурсу: https://tseivo.com/b/memecode/t/en3oooxpvn

26. Kali Linux: історія, встановлення, налаштування [Електронний ресурс]. — Режим доступу до ресурсу: https://research.kr-labs.com.ua/kali-linux-guide

27. Wireshark: аналіз мережевого трафіку [Електронний ресурс]. — Режим доступу до ресурсу: https://itorakul.com.ua/wireshark

28. tcpdump Command [Електронний ресурс]. — Режим доступу до ресурсу: https://www.ibm.com/docs/en/aix/7.1.0?topic=t-tcpdump-command

29. github - smikims/arpspoof [Електронний ресурс]. — Режим доступу до ресурсу: https://github.com/smikims/arpspoof

30. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. — 42 с.

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. О.Д. Азаров
« 25 » вересня 2025 р.
_____

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
«Метод захисту від ARP-спуфінгу в локальних комп'ютерних мережах»

Науковий керівник: доцент к.т.н.
_____ Войцеховська О.В.
Студентка групи 1КІ-24м
_____ Паламарчук К.О.

м. Вінниця – 2025

1 Підстава для використання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність розробки методів захисту ARP-протоколу полягає у комплексному забезпеченні безпеки локальної комп’ютерної мережі шляхом виявлення та запобігання атакам типу ARP-спуфінг. Запропонований підхід підвищення безпеки ARP за допомогою цифрового підпису та поведінкової детекції забезпечує захист мережі без потреби у спеціальному обладнанні або складному адмініструванні.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета проєкту — розробка методу захисту від ARP-спуфінгу у локальних комп’ютерних мережах шляхом додавання механізму цифрового підпису до процесу обміну ARP-повідомленнями та виявлення і подальшого блокування підозрілого ARP-трафіку.

2.2 Призначення розробки — підвищити рівень захисту локальних комп’ютерних мереж від атаки типу ARP-спуфінг без необхідності використання спеціалізованого апаратного забезпечення чи складних змін у мережевій інфраструктурі.

3 Вихідні дані для виконання МКР

3.1 Проведення порівняльного аналізу існуючих методів захисту ARP-протоколу та технологій мережевої безпеки.

3.2 Проєктування структури методу, розробка алгоритмів роботи модулів цифрового підпису та поведінкового аналізу.

3.3 Програмна реалізація клієнтського агенту та серверу довіри.

3.4 Тестування розробленого методу у моделі локальної комп’ютерної мережі.

4 Вимоги до виконання МКР

Головна вимога — розробити метод захисту від ARP-спуфінгу з використанням цифрового підпису та поведінкової детекції, використовуючи усі здобуті навички та знання.

5 Етапи МКР та очікувані результати

5.1 Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ з/п	Назва етапів дипломної роботи	Термін виконання		Очікувані результати
		початок	кінець	
1	Постановка задачі роботи	25.09.2025	29.09.2025	Аналітичний огляд літературних джерел, задачі досліджень,
2	Огляд та порівняння існуючих технологій	30.09.2025	02.10.2025	Розділ 1
3	Визначення функціональних вимог до методу та проєктування	03.10.2025	05.10.2025	Розділ 2
4	Розробка алгоритмів роботи методу	06.10.2025	16.10.2025	Розділ 2
5	Програмна реалізація методу	17.10.2025	26.10.2025	Розділ 3
6	Створення моделі комп’ютерної мережі, тестування методу і виправлення помилок	27.10.2025	02.11.2025	Розділ 4
	Розрахунок економічної частини	03.11.2025	04.11.2025	Розділ 5
7	Оформлення пояснювальної записки	08.05.2024	19.05.2024	ПЗ, графічний матеріал
8	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	12.11.2025	12.12.2025	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, анотації до МКР українською та іноземною мовами, довідка про

відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

При оформлюванні МКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2023;

— документами на які посилаються у вище вказаних.

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Метод захисту від ARP-спуфінгу в локальних комп'ютерних мережах

Тип роботи: _____ магістерська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) __2_____%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

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

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., д.т.н., проф., зав. кафедри ОТ_ _____
(прізвище, ініціали, посада) (підпис)

_____Мартинюк Т. Б., д.т.н., проф. кафедри ОТ__ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С. М., проф. каф. ОТ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____Войцеховська О. В., доц. каф. ОТ____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____ Паламарчук К.О._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

Структурна схема методу захисту від ARP-спуфінгу

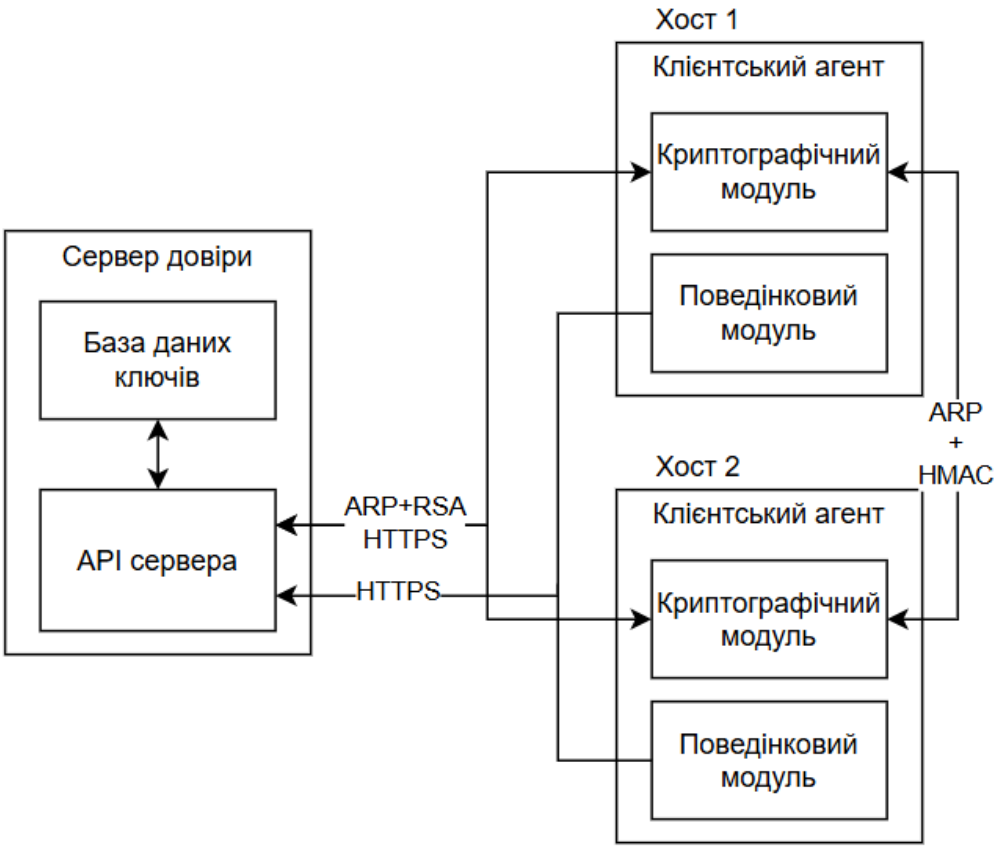

Рисунок В.1 — Структурна схема методу захисту від ARP-спуфінгу

ДОДАТОК Г

Блок-схема алгоритму встановлення довіри

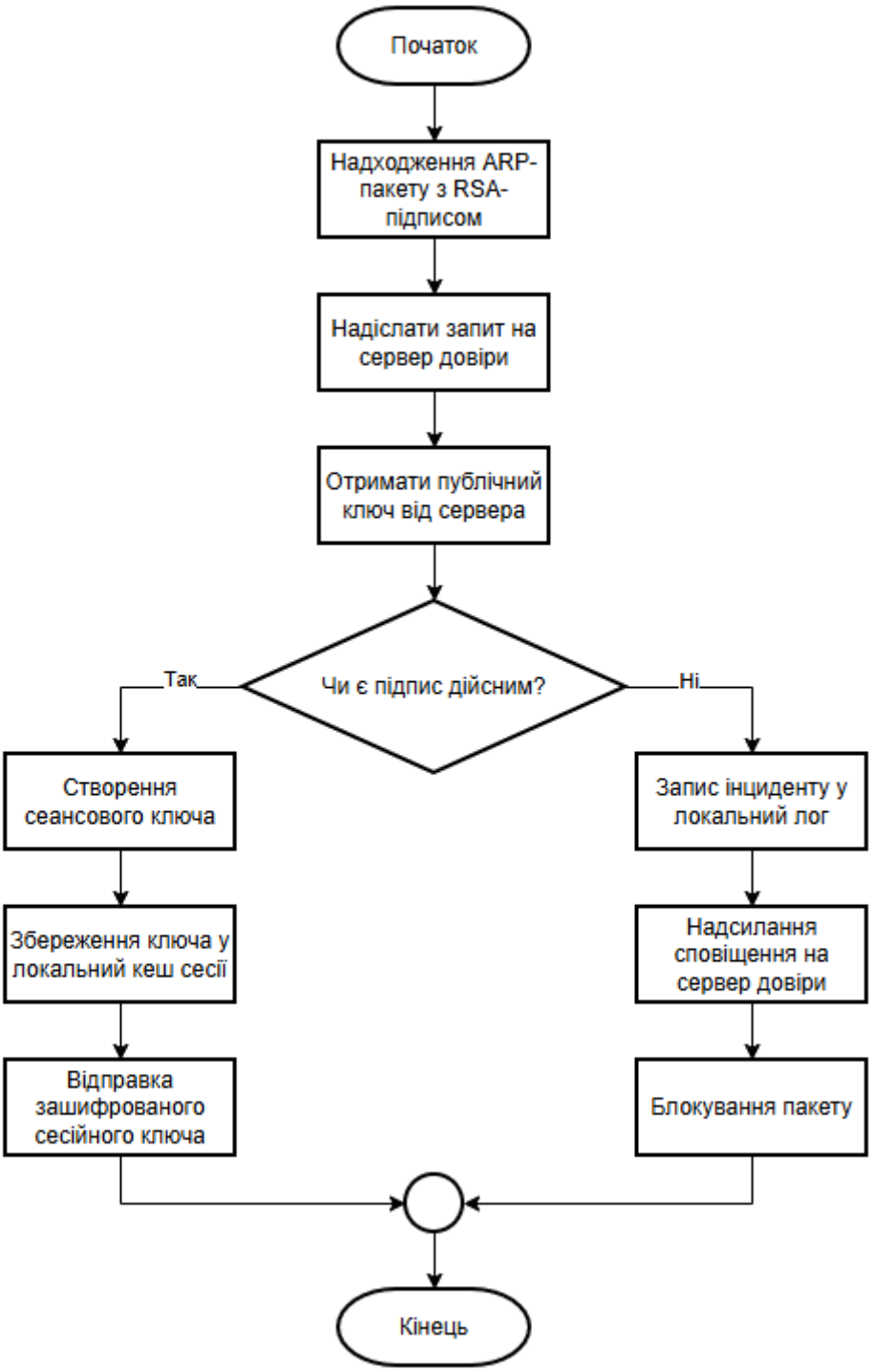

Рисунок Г.1 — Блок-схема алгоритму встановлення довіри

ДОДАТОК Д

Блок-схема алгоритму обміну пакетами з використанням сеансових ключів

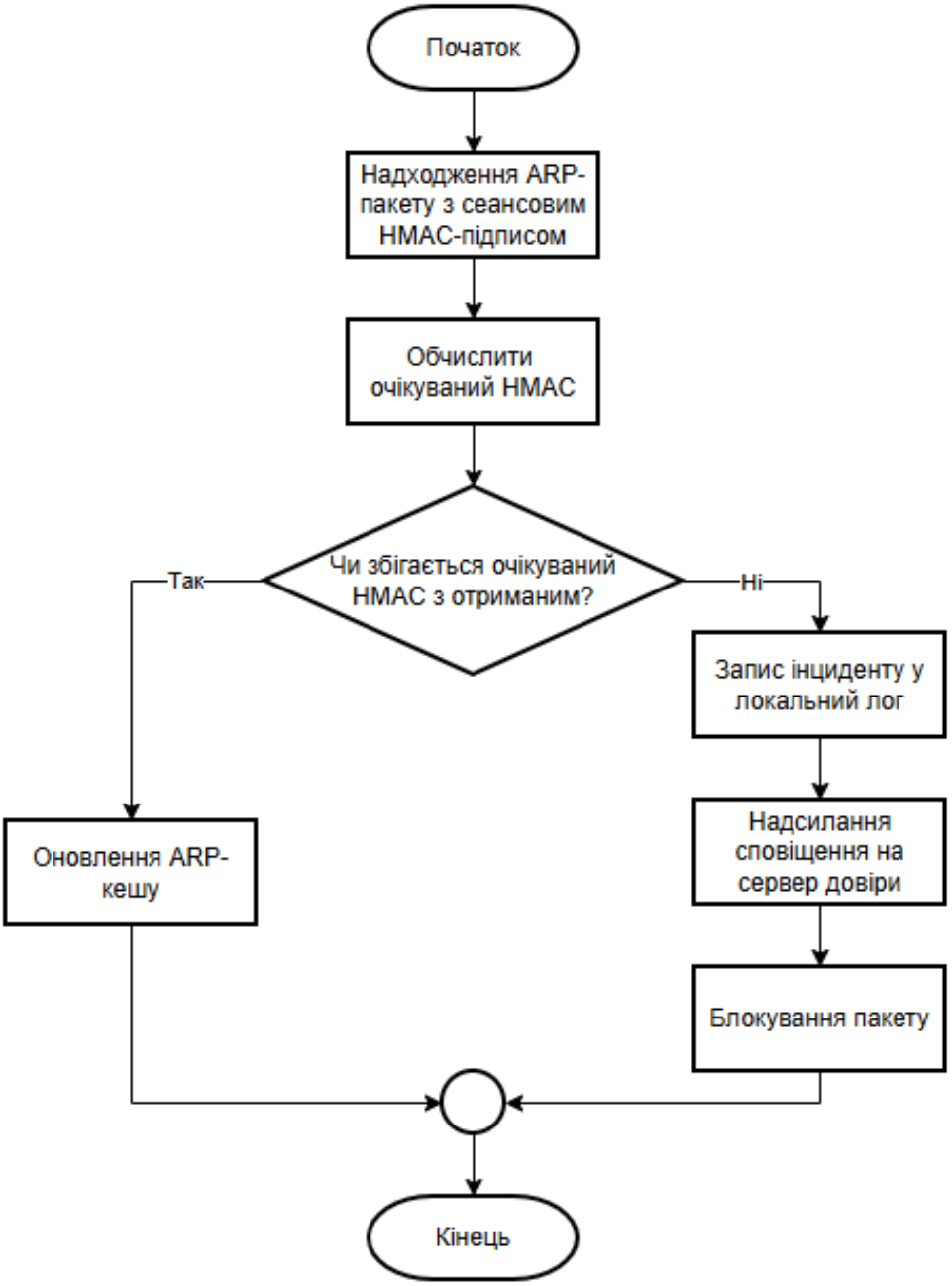

Рисунок Д.1 — Блок-схема алгоритму обміну пакетами з використанням сеансових ключів

ДОДАТОК Е

Блок-схема алгоритму роботи криптографічного модулю при взаємодії з незахищеним вузлом

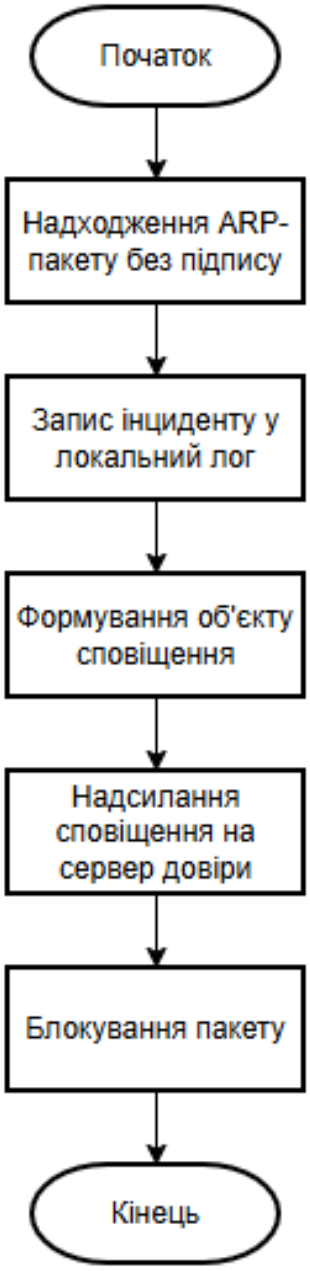

Рисунок Е.1 — Блок-схема алгоритму роботи криптографічного модулю при взаємодії з незахищеним вузлом

ДОДАТОК Ж

Блок-схема алгоритму роботи поведінкового модулю

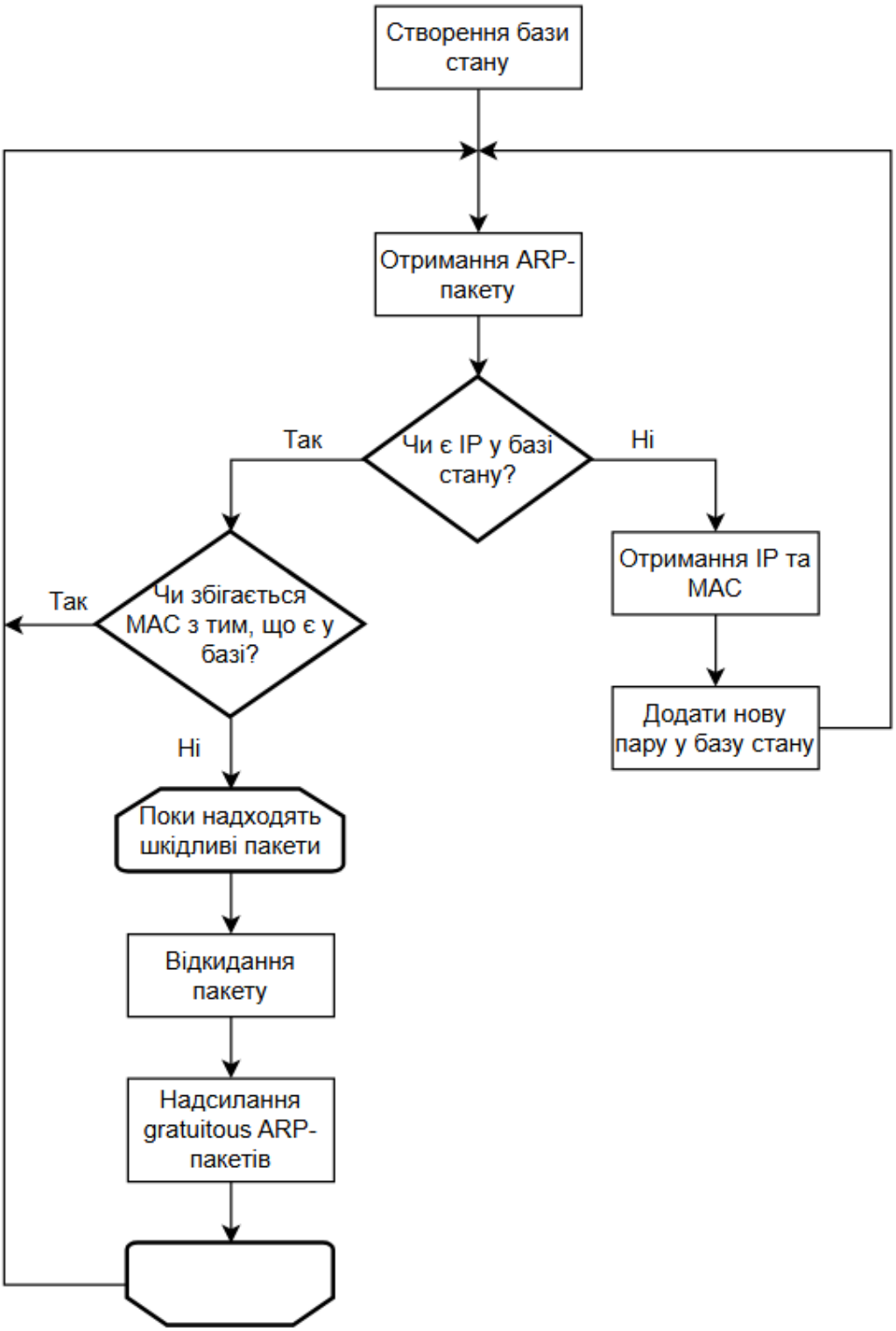

Рисунок Ж.1 — Блок-схема алгоритму роботи поведінкового модулю

ДОДАТОК И

Лістинг коду методу захисту

Лістинг И.1 — Обробник підписів RSA / HMAC

```
import hmac
import hashlib
import os
import requests
from scapy.all import ARP
from cryptography.hazmat.primitives import hashes,serialization
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives.serialization import
load_pem_private_key, load_pem_public_key
with open("private_key.pem","rb") as f:
    MY_PRIVATE_KEY = load_pem_private_key(f.read(), password=None)
SESSION_KEYS = {}
MY_MAC = "AA:BB:CC:DD:EE:FF"
def process_packet(packet):
    payload = packet.get_payload()
    try:
        arp_pkt = ARP(payload)
    except:
        packet.drop()
        return
    if arp_pkt.op == 2 and arp_pkt.hwsrc == MY_MAC:
        signature = b""
        target_ip = arp_pkt.pdst
        if target_ip in SESSION_KEYS:
            key = SESSION_KEYS[target_ip]
            h = hmac.new(key, payload, hashlib.sha256)
            signature = h.digest()
        else:
            signature = MY_PRIVATE_KEY.sign(payload,
padding.PSS(mgf=padding.MGF1(hashes.SHA256()),
salt_length=padding.PSS.MAX_LENGTH), hashes.SHA256())
        arp_pkt.load = signature
        packet.set_payload(bytes(arp_pkt))
        packet.accept()
    elif arp_pkt.op == 2 and arp_pkt.hwsrc != MY_MAC:
        if not arp_pkt.load:
            packet.drop()
            return
        signature = arp_pkt.load
        original_payload = bytes(arp_pkt)[:28]
        source_ip = arp_pkt.psrc
        if source_ip in SESSION_KEYS:
            key = SESSION_KEYS[source_ip]
            h = hmac.new(key, original_payload, hashlib.sha256)
            if hmac.compare_digest(h.digest(), signature):
                packet.set_payload(original_payload)
                packet.accept()
```

```
            else:
                packet.drop()
        else:
            r = requests.get(f"https://server-
ip:8000/get_key/{source_ip}", verify=False)
            if r.status_code != 200:
                packet.drop()
                return
            pub_key_str = r.json()['public_key']
            public_key =
load_pem_public_key(pub_key_str.encode('utf-8'))
            try:
                public_key.verify(signature, riginal_payload,
padding.PSS(mgf=padding.MGF1(hashes.SHA256()), salt_length =
padding.PSS.MAX_LENGTH), hashes.SHA256())
                new_session_key = os.urandom(32)
                SESSION_KEYS[source_ip] = new_session_key
                packet.set_payload(original_payload)
                packet.accept()
            except Exception:
                packet.drop()
    else:
        packet.accept()
```

Лістинг И.2 — Логіка поведінкового модулю

```
class BehavioralIDS:
    def __init__(self):
        self.ip_mac_map = {}
        self.packet_history = defaultdict(lambda: deque())
    def alert_server(self, attack_type, source_ip, source_mac,
details):
        alert_msg = f"[{attack_type}] Source: {source_ip}
({source_mac}). {details}"
        logging.warning(alert_msg)
        try:
            payload = {
                "source_ip": source_ip,
                "description": f"{attack_type}: {details}"
            }
            pass
        except Exception as e:
            logging.error(f"Failed to send alert: {e}")
    def mitigate_attack(self, target_ip, correct_mac):
        logging.info(f"MITIGATION: Broadcasting correct MAC for
{target_ip} is {correct_mac}")
        packet = Ether(dst="ff:ff:ff:ff:ff:ff") / ARP(
            op=2, psrc=target_ip, hwsrc=correct_mac,
hwdst="ff:ff:ff:ff:ff:ff", pdst="ff:ff:ff:ff:ff:ff")
        sendp(packet, count=3, verbose=False)
    def is_flood(self, ip_src):
        current_time = time.time()
```

```
        timestamps = self.packet_history[ip_src]
        timestamps.append(current_time)
        while timestamps and timestamps[0] < current_time -
TIME_WINDOW:
            timestamps.popleft()
        if len(timestamps) > FLOOD_THRESHOLD:
            return True
        return False
    def process_packet(self, packet):
           if not packet.haslayer(ARP):
            return
        op = packet[ARP].op
        psrc = packet[ARP].psrc
        hwsrc = packet[ARP].hwsrc
        if self.is_flood(psrc):
            self.alert_server("ARP_FLOOD", psrc, hwsrc, f"Rate
exceeded: >{FLOOD_THRESHOLD} pkts/{TIME_WINDOW}s")
            return
        if psrc in self.ip_mac_map:
            known_mac = self.ip_mac_map[psrc]
            if known_mac != hwsrc:
                attack_details = f"Conflict! Stored: {known_mac},
Received: {hwsrc}"
                self.alert_server("ARP_SPOOFING", psrc, hwsrc,
attack_details)
                if psrc == GATEWAY_IP:
                    self.mitigate_attack(psrc, known_mac)

        else:
            self.ip_mac_map[psrc] = hwsrc
            logging.info(f"New Host Discovered: {psrc} ->
{hwsrc}")
```