Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного модулю для аудиту захищеності локальних комп'ютерних підмереж»

Виконав: студент 4 курсу
групи 1ПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Малініч П.П.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Коваленко О.О.
(прізвище та ініціали)

Рецензент: к.т.н., ст. викл. каф. КН Озеранський. В.С.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"_25_" __березня__2022 року

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

Малінічу Павлу Павловичу

1. Тема роботи – «Розробка програмного модулю для аудиту захищеності локальних комп'ютерних підмереж»

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "_24_" _березня_2022 року №_66_

2. Строк подання студентом роботи _13 червня 2022 року__

3. Вихідні дані до роботи: Середовище розробки – Visual Studio Code, Мови розробки – Python та PHP, Операційна система – GNU/Linux.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; розробка архітектури та алгоритмів програмного додатка; розробка програмного додатку; тестування додатку, висновки; список використаних джерел, додатки.

5. Перелік графічного матеріалу: структурна схема програмного модулю, ER-діаграма та діаграма сутностей бази даних, робота методу багаторівневого сканування мережі та методу побудови карти, алгоритм роботи програмного модулю та тестування додатку

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Коваленко О. О., к.т.н., доцент кафедри ПЗ		

7. Дата видачі завдання _____25 березня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
	Аналіз проблеми, обґрунтування актуальності розробки системи та постановка задачі	26.03.2022 – 10.04.2022	Вик.
	Розробка архітектури та алгоритмів роботи системи	11.04.2022 – 20.04.2022	Вик.
	Вибір середовища та мови розробки	21.04.2022 – 24.04.2022	Вик.
	Розробка програмного продукту	25.04.2022 – 22.05.2022	Вик.
	Тестування роботи системи	23.05.2022 – 03.06.2022	Вик.
	Оформлення матеріалів до захисту БДР	04.06.2022 – 10.06.2022	Вик.

Студент _____ Маліниіч П. П.

(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи_____ Коваленко О. О.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 106 сторінок формату А4, на яких є 42 рисунків, 17 таблиць, список використаних джерел містить 30 найменувань.

У бакалаврській дипломній роботі проведено детальний аналіз методів і засобів аудиту захищеності локальних комп'ютерних підмереж. Сформульовано мету досліджень – удосконалення процесу аудиту захищеності локальних комп'ютерних підмереж, який можна було б легко масштабувати додаючи підтримку різних сервісів.

Розроблено метод побудови карти мережі на основі їх класу захищеності, який дозволяє візуально оцінити загальний стан мережі організації. Запропоновано підхід зростаючої складності карти відповідно до масштабів мережі та наявних у ній класів безпеки.

Розроблено алгоритми та програми автоматизованого аудиту захищеності локальних комп'ютерних підмереж на їх основі розроблених моделей і методів.

Отримані в бакалаврській дипломній роботі результати можна використовувати для автоматизованого аудиту корпоративних комп'ютерних мереж.

Ключові слова: аудит безпеки, комп'ютерні мережі, маршрутизатори, моніторинг комп'ютерних мереж, сканери комп'ютерних мереж.

ABSTRACT

The bachelor's thesis consists of 106 A4 pages, which have 42 figures, 17 tables, the list of source papers used contains 30 items.

A detailed analysis of the methods and means of auditing the security of local computer subnets was conducted in the bachelor's thesis. The aim of the research is to find the most flexible approach for the basic automated audit of the security of local computer subnets, which could be easily scaled by adding support for various services.

A method of constructing a network map based on their security class has been developed, which allows to visually assess the general state of the organization's network. An approach of increasing complexity of the network map according to the scale of the network is proposed and the security classes available in it.

Algorithms and programs for automated security audit of local computer subnets based on the developed models and methods have been developed.

The results gained in the bachelor's thesis can be used for automated audit of corporate computer networks.

Keywords: security audit, computer networks, routers, computer network monitoring, computer network scanners.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. З розвитком технологій та збільшенням їх доступності зростає потреба в періодичному аудиті та моніторингу локальних мереж у різних організаціях. Кожна організація бажає мати об’єктивну оцінку стану мережі, за допомогою якої мережеві спеціалісти цієї організації змогли б виконувати свою роботу більш ефективно, забезпечуючи вищий рівень безпеки співробітників за рахунок швидкого виявлення проблем [1]. Відповідно до даної потреби почали розвиватися програмні застосунки для моніторингу мережевого обладнання та побудови згідно отриманих даних результатів аудиту [2]. Дані системи надають можливість компаніям забезпечувати своїх співробітників достатнім рівнем безпеки та швидким вирішенням мережевих проблем [3]. Це у свою чергу значно підвищує ефективність працівників організації.

Аудит безпеки покладається на дві складові: на кваліфікованих спеціалістів та потужні інструменти за допомогою яких можна зібрати інформацію про мережу [4]. Першим чином при початку процесу аудиту безпеки комп’ютерної мережі проводиться збір інформації, аналіз елементів мережі та всіх зв'язків між ними. Наступним чином необхідно здійснити загальний рівень захищеності. Подальшим чином відбувається виявлення потенційно вразливих елементів мережі. На основі цих даних спеціалісти можуть провести аналіз ризиків атак ззовні. Під час цього вони можуть спробувати змоделювати атаку зовні [5]. На основі отриманої інформації виробляються рекомендації яким саме чином можна поліпшити ситуацію з безпекою.

Сучасні системи для аудиту безпеки корпоративних комп’ютерних мереж досить часто є досить складними для початківців, а також вузьконаправленими:

– для аудиту комп’ютерних мереж на базі домену Windows [6];
– для аудиту комп’ютерних мереж на базі серверів GNU/Linux;

– для аудиту комп’ютерних мереж по налаштуванням мережевого обладнання [7];
– для аудиту комп’ютерних мереж через цільове сканування під мереж;
– для аудиту комп’ютерних мереж через аналіз лог-файлів [8].

Серед об'єктів яким приділяється увага при аудиту мережі можна виділити наступні: маршрутизатори, комутатори, сервери, дискові масиви, бездротові точки доступу, IoT-пристрої [9], а також мережеві служби які працюють у мережі.

Також варто розуміти, що самі по собі ці додатки не можуть собою замінити кваліфікованих спеціалістів з інформаційної безпеки. Система може бути у змозі виявити певні вразливості та надати певні типові рекомендації, однак для проведення їх аналізу: яким чином вони виникли, яким чином впливають на систему та які ризики вони можуть являти – ці питання нині є досить складними у безпомилковому вирішенні програмних шляхом. Саме тому нове рішення має бути максимально зручним як для рядових співробітників, так і для більш професійних спеціалістів.

Основним недоліком систем аудиту комп’ютерних мереж є необхідний рівень знань та навичок роботи із ними [10]. Спеціалістам необхідно мати досить істотний рівень знань у різних технологіях, щоб повністю справитись із завданням. Комплексний безпековий аудит комп’ютерної мережі має здійснюватись як мінімум командою, яка розбирається у всіх задіяних на підприємстві цифрових технологіях [11].

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є удосконалення процесу аудиту захищеності локальних комп’ютерних підмереж, який дозволяє покращити процес усунення неполадок та загроз.

Основними задачами дослідження є:

– провести аналіз та постановку задачі;

- розробити архітектуру та алгоритми програмного модулю;
- розробити метод багаторівневого сканування мережі;
- розробити метод побудови карти мережі;
- розробити API-інтерфейс програмного модулю;
- розробити інструмент збору даних з обладнання;
- розробити інструмент сканування та тестування підмереж;
- розробити інструмент контролю за авторизацією;
- розробити графічний веб-інтерфейс для програмного модулю;
- провести тестування інструменту збору даних з обладнання та інших інструментів з використанням справжнього обладнання.

Об’єкт дослідження – процес розробки програмного модулю для аудиту захищеності локальних комп’ютерних підмереж.

Предмет дослідження – програмний модуль для аудиту захищеності локальних комп’ютерних підмереж.

Методи дослідження. У процесі досліджень використовувались методи дослідження:

- метод опитування внутрішнього стану керованого обладнання (поллінг);
- метод зовнішнього збору даних про доступні хости та мережеві сервіси (маппинг);
- метод створення карт на основі візуалізації даних про мережу та з’єднання між її елементами.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод побудови карти мережі на основі їх класу захищеності; метод багаторівневого сканування мережі на основі даних зібраних безпосередньо з обладнання (поллінг) та з даних зовнішнього сканування (маппинг).
2. Подальшого розвитку отримав метод зростаючої складності карти відповідно до масштабів мережі та наявних у ній класів безпеки. Даний

метод дозволяє створювати логічні карти комп'ютерних мереж ускладнюючи поділ елементів різних рівнів з ростом мережі.

Практична цінність отриманих результатів. Практична цінність полягає у кінцевій реалізації програмного модулю для аудиту захищеності локальних комп'ютерних підмереж.

Впровадження. Впровадження результатів досліджень підтверджуються відповідними актами та використовуються на таких підприємствах і організаціях:

– Громадська організація «Радіоланс Україна»
– Центр електронних комунікацій «ІнтерЦЕК» (ВНТУ)

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій роботі, опублікованій у співавторстві, автору належать такі результати: базу даних; інструмент збору даних з обладнання; інструмент сканування та тестування підмереж; інструмент контролю за авторизацією; графічний інтерфейс для веб-додатку.

Апробація матеріалів бакалаврської дипломної роботи. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на LI Науково-технічній конференції підрозділів Вінницького національного технічного університету НТКП ВНТУ (2022) [1].

Публікації. Основні результати дослідження опубліковані в науковій роботі – в тезах доповідей:

– Науково-технічна конференція підрозділів Вінницького національного технічного університету НТКП ВНТУ (2022).
– Міжнародна наукова інтернет-конференція "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення, м. Тернопіль (2019).

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз стану систем для аудиту комп’ютерних мереж

Сучасні корпоративні комп’ютерні мережі постійно зростають за рахунок розвитку IoT-технологій. Як результат суттєво збільшується кількість пристроїв у мережах. Крім того, з кожним роком зростає кількість вразливостей у різних програмних продуктах та мікропрограмах різних пристроїв, однак не менш критичними є безпекові проблеми у мережевій інфраструктурі. Питання безпеки фізичних комп’ютерних мереж є більш комплексним, оскільки майже завжди охоплює більше ніж один пристрій [11]. Мережеві атаки з отриманням фізичного доступу дають значно більші можливості для порушення роботи мережі, полегшують доступ до внутрішніх ресурсів мережі, а також можуть дати змогу перехоплювати трафік, однак є більш складними для виконання.

Існує досить багато рішень захисту мереж корпоративного класу, які наприклад використовуються банками, медичними закладами, а також великим бізнесом. Як відомо різні рішення аудиту захищеності мережі сфокусовані на різних аспектах IT-інфраструктури [12]. Наприклад як одні рішення можуть бути основані на пролідковуванні лог-файлів, інші можуть базуватись на аналізі мережевих сервісів або на каталозі користувачів. У той же самий час небагато систем тримають фокус на стані захищеності фізичної мережі .

Досить великий ризик зіткнутись із повністю справжнім фізичним несанкціонованим доступом є у провайдерів доступу до мережі Інтернет, для яких насамперед важливо вберегтись від проявів недобросовісної конкуренції та спроб самовільно підключитись зі сторони недобросовісних жителів. Як результат існують наступні можливі види взлому:

– несанкціоноване приєднання;
– несанкціоноване розширення мережі;
– тегування та перехід VLAN;
– приєднання до VLAN;

1.2 Порівняльний аналіз аналогів

Існує досить багато різних інструментів для аудиту захищеності мережевої інфраструктури підприємства, серед них можна виділити наступні:

– SolarWinds Network Configuration Manager;
– N-able RMM;
– ManageEngine Log360;
– Kaseya VSA An RMM;

SolarWinds Network Configuration Manager (NCM, рис. 1.1) призначений для системних адміністраторів для аудиту своєї мережі, а також для розгортання змін конфігурації на пристроях по всій мережі. Ця комбінація функцій дозволяє не тільки вносити зміни в конфігурацію, пов’язані з безпекою, але й відстежувати нові та несанкціоновані зміни на ваших пристроях.

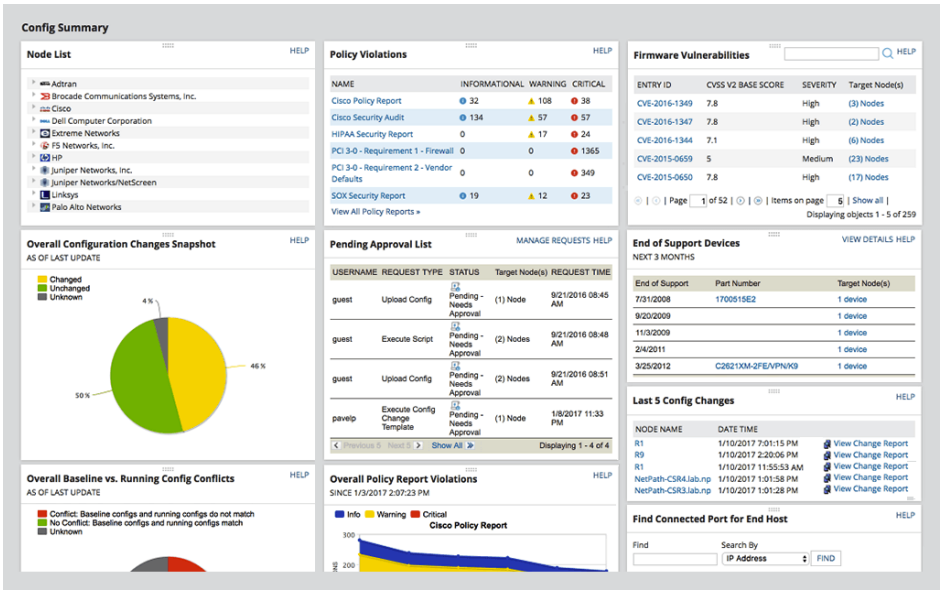


Рисунок 1.1 – Веб-інтерфейс SolarWinds NCM

Інструмент автоматично сканує та відстежує мережу на наявність пристроїв, а також дозволяє користувачу керувати безпекою своєї мережі та пристроїв, які в ній знаходяться. За допомогою централізованої інформаційної панелі Network Configuration Manager може відразу виявляти й попереджати користувача про найактуальніші події безпеки, тому не можна здогадуватися, який надати

пріоритет. SolarWinds підтримує десятки інтеграцій, тому перенесення сповіщень у вашу систему підтримки абонентів також є підтримуваним варіантом, якщо є потреба його використовувати у NOC або у службі підтримки. Нарешті, звітність може бути налаштована на створення квартальних звітів або детальної інформації про те, що було виявлено аудитом безпеки [13]. SolarWinds Network Configuration Manager є платним продуктом, і найбільш базова його версія коштує $1500 в рік.

Інший аналог, N-able RMM (рис. 1.2) більше призначений для постачальників керованих послуг (MSP), які керують кількома клієнтами і хочуть запропонувати аудит як послугу. Цей хмарний інструмент забезпечує віддалений моніторинг, а також управління ризиками та виявлення загроз на кількох сайтах або клієнтах одночасно. На централізованій інформаційній панелі можна переглядати ризики для кожної компанії, об’єкта або в цілому.

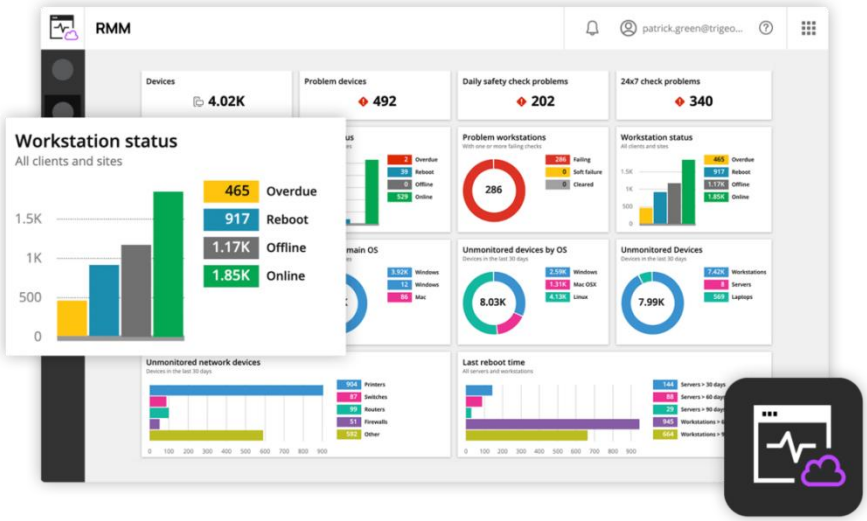


Рисунок 1.2 – Панель керування N-able RMM

Такі деталі, як кількість проблемних пристроїв, статус резервного копіювання та перевірки стану, можна побачити за допомогою простого дайджесту безпеки, який надає інформаційна панель. Вся платформа повністю налаштовується, що дозволяє створювати унікальні представлення панелі інструментів для вашого мережевого операційного центру та інших відділів за потреби. Шаблони аудиту допомагають зробити сканування простим, а також

виявляти конкретні проблеми відповідності. Наприклад, є вбудовані інструменти, які можуть спеціально сканувати та підтверджувати, чи ваша мережа в даний момент сумісна з HIPAA або PCI [14], і надавати допоміжний звіт. Розділ управління ризиками N-able RMM може сканувати та знаходити всю персональну ідентифікаційну інформацію (PII) і відстежувати, як і куди вона переміщається по мережі.

Цей рівень управління ризиками може запобігти виходу певної інформації з мережі, а також попередити про неналежний доступ до інформації. Дозволи безпеки можна сканувати для файлів і папок, щоб виявити неправильні дозволи для облікових записів користувачів на основі записів компанії. Нарешті, N-able RMM має потужну систему керування виправленнями, яка дозволяє створити шаблон процесу виправлення. Отже, якщо є оновлення, які, як ви знаєте, заважають певному програмному забезпеченню, ви можете скопіювати ці шаблони виправлень на всіх своїх клієнтів у своєму MSP [14]. Оскільки N-able RMM є SaaS, її процес встановлення відбувається у хмарі, а виставлення рахунків здійснюється за допомогою моделі на основі підписки.

ManageEngine Log360 видобуває журнали для отримання інформації про події, тому якість її роботи дуже залежить від ретельності системи збору журналів. ManageEngine забезпечує подачу даних журналу високої якості.

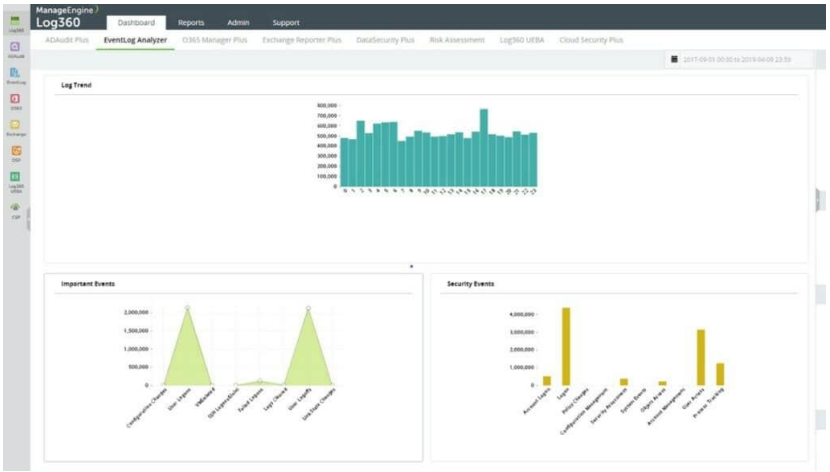


Рисунок 1.3 – ManageEngine Log360

Збірник журналів збирає дані з більш ніж 700 систем, включаючи події Windows і повідомлення Syslog з операційних систем. Журнали надсилаються на центральний сервер, де вони перетворюються в загальний формат і зберігаються.

Панель інструментів Log360 містить засіб перегляду даних, і він відображає записи у режимі реального часу, коли вони обробляються через сервер журналів. Переглядач даних включає аналітичні інструменти, а також можна читати записи з файлів журналів. Можна налаштувати власні автоматичні правила пошуку, які можуть працювати постійно та викликати сповіщення. Наприклад, цей сервіс легко використовувати для моніторингу цілісності файлів. Упорядковане зберігання файлів журналів робить це досить відповідним інструментом для аудиту відповідності [15].

Kaseya VSA — це програмне забезпечення для віддаленого керування та моніторингу, яке може виконувати аудит ризиків безпеки, а також здійснювати виявлення мережі та керування кінцевими точками. Компонент виявлення мережі автоматично зберігає інформацію про пристрій і мережу, а також інформацію про безпеку та виправлення. Глобальні політики безпеки та виправлення можуть бути встановлені для кожного клієнта та розгорнуті в масштабі, що робить Kaseya VSA чудовим варіантом для постачальників керованих послуг та великих підприємств із кількома мережами.

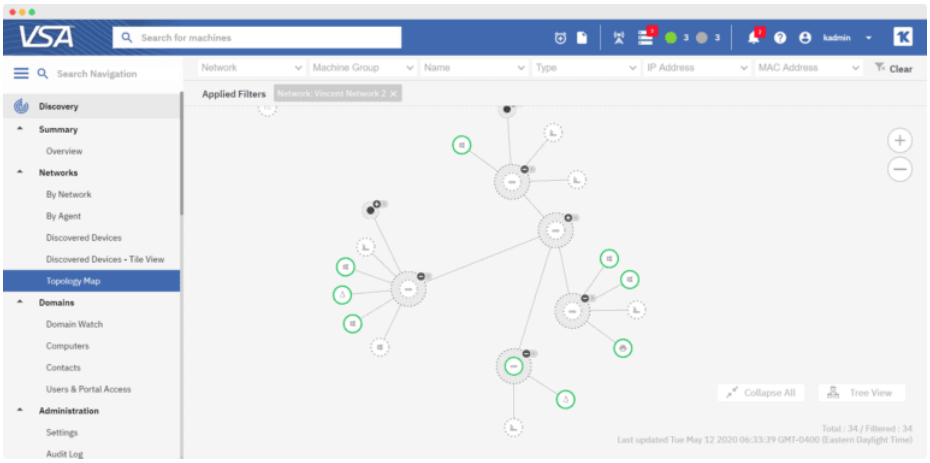


Рисунок 1.4 – Панель Kaseya VSA

Автоматизація на основі умов і політик дає змогу автоматично виконувати завдання з виправлення та безпеки. Платформа має портал Automation Exchange, на якій є 500 різних сценаріїв і готових конфігурацій, які доступні до розгортання.

Результати порівняння аналогів зведено в табл. 1.1. Критерії визначені відносно ключових потреб захисту низькобюджетних комп'ютерних мереж [11]. Оцінювання виконане за 100-бальною шкалою.

Таблиця 1.1 – Порівняльні характеристика систем аудиту захищеності комп'ютерних мереж

Критерії	SolarWinds NCM	N-able RMM	ManageEngine Log360	Kaseya VSA	Власний модуль
Адаптивний інтерфейс	–	+	+	+	+
Сканування підозрілих хостів	+	–	–	+	+
Розподіл мереж за класами захищеності	+	–	–	+	+
Інтеграція з Active Directory	+	+	–	+	–
Підтримка низькобюджетного обладнання	–	–	+	–	+
Загальна оцінка	60%	40%	40%	80%	80%

Відповідно до таблиці порівняльних характеристик розробка власного програмного модулю для аудиту захищеності локальних комп'ютерних під мереж має сенс. Отриманий продукт може покрити недоліки існуючих рішень та забезпечити новий функціонал для аудиту захищеності локальних комп'ютерних підмереж.

1.3 Аналіз методів розв'язання поставленої задачі

У основі нової розробки мають бути закладені методи, які у подальшому дадуть змогу позиціонувати її як конкурентоздатне рішення:

- Метод побудови карти мережі на основі їх класу захищеності. Карта мережі будується за ознаками належності мережі та конкретного хосту до того чи іншого класу. Для кожного класу задаються окремі правила відображення об’єктів та їх визначення як підозрілих.
- Метод багаторівневого сканування мережі на основі даних зібраних безпосередньо з обладнання (поллінг) та з даних зовнішнього сканування (маппинг). Даний метод полягає у тому що дані про хост надходять із різних джерел і для того щоб визначити які з них більш достовірні власне і потрібен даний метод.

1.4 Постановка задач для програмного модулю для аудиту захищеності локальних комп’ютерних підмереж

Проаналізувавши переваги та недоліки існуючих систем для аудиту захищеності локальних комп’ютерних підмереж, сформульовано такі задачі бакалаврської дипломної роботи:

- провести аналіз та постановку задачі;
- розробити архітектуру та алгоритми програмного модулю;
- розробити метод багаторівневого сканування мережі;
- розробити метод побудови карти мережі;
- розробити API-інтерфейс програмного модулю;
- розробити інструмент збору даних з обладнання;
- розробити інструмент сканування та тестування підмереж;
- розробити інструмент контролю за авторизацією;
- розробити графічний веб-інтерфейс для програмного модулю;
- провести тестування інструменту збору даних з обладнання та інших інструментів з використанням справжнього обладнання.

1.5 Висновки

У першому розділі розглянуто сучасний стан систем для аудиту комп’ютерних мереж. Проведено порівняння відомих систем для аудиту комп’ютерних мереж, таких як: SolarWinds NCM, N-able RMM, ManageEngine Log360 та Kaseya VSA. При порівнянні виявлено, що всі перелічені системи надаються виключно на платній основі. Проаналізувавши їх переваги та недоліки, виявлено, що такі опції як сканування підозрілих хостів, розподіл мереж за класами захищеності та підтримка низькобюджетного обладнання потребують подальшого розвитку. Завдяки цьому доведено доцільність розробки власного програмного модулю. На основі отриманої інформації сформовано перелік задач, які необхідно виконати для розробки власного програмного модулю.

2 РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ДОДАТКУ

2.1 Розробка архітектури програмного модулю

Для сучасних стартапів можливість швидко розвивати програмний продукт є надзвичайно важливою для боротьби у висококонкурентному середовищі. Для цього процес розробки програмного продукту має бути надзвичайно гнучким. Різні гнучкі підходи до організації процесу розробки, зокрема і ітераційний [16] передбачають ефективне початкове планування подальших процесів розробки, у результаті якого виробляється архітектура проєкту. Тобто розробці архітектури має передувати вироблення бачення процесів подальшого розширення функціоналу проєкту.

Розробка архітектури є найбільш важливим етапом розробки базового прототипу, який у разі успішного розвитку проекту повинен мати можливість швидкого росту до рівня MVP. Minimal Viable Product (MVP, мінімально життєздатний продукт) – тестова версія товару, послуги або сервісу з мінімальним набором функцій (іноді навіть однієї), яка має цінність для кінцевого споживача. MVP створюють для тестування гіпотез та перевірки життєздатності задуманого продукту, наскільки він справді буде цінним та необхідним на ринку [17]. Результати тестування мінімально життєздатного продукту та зворотний зв'язок від цільової аудиторії допомагають зрозуміти, чи варто розвивати проект далі, які зміни слід внести до стратегії, а що залишити у початковому вигляді [18,19].

Для розробки MVP-версії програмного модулю для аудиту захищеності локальних комп’ютерних підмереж необхідно зрозуміти які перспективи розвитку може мати дане рішення. На етапі створення MVP-рішення надзвичайно важко остаточно визначити який кінцевий формат додатку буде найбільш зручним для клієнта. Важливо розуміти що кінцеве рішення, яке буде створене на базі модулю може мати наступні варіанти розвитку:

- перетворення програмного модулю у повноцінну систему моніторингу та аудиту безпеки мережі;
- реалізація програмного модулю у вигляді API-сервісу, який даватиме іншим розробникам вбудовувати модуль у власні рішення;
- адаптація програмного модулю у вигляді розширення до існуючих систем моніторингу, таких як Zabbix, Nagios або UserSide.

Водночас при розробці архітектури програмного модулю необхідно розуміти що рішення, яке буде далі розроблятись на базі MVP-версії може бути будь-яким із трьох можливих варіантів. Але разом із тим, сама MVP-версія має найбільш вигідно демонструвати всі можливості програмного модулю, для чого може знадобитись самостійний веб-інтерфейс.

З врахуваннях всіх трьох сценаріїв розвитку проєкту відповідно до поставленої задачі визначено основні компоненти програмного модуля: база даних, API-інтерфейс, веб-інтерфейс, планувальник завдань, API-оболонка для запуску скриптів та набір скриптових інструментів програмного модулю (рис. 2.1).

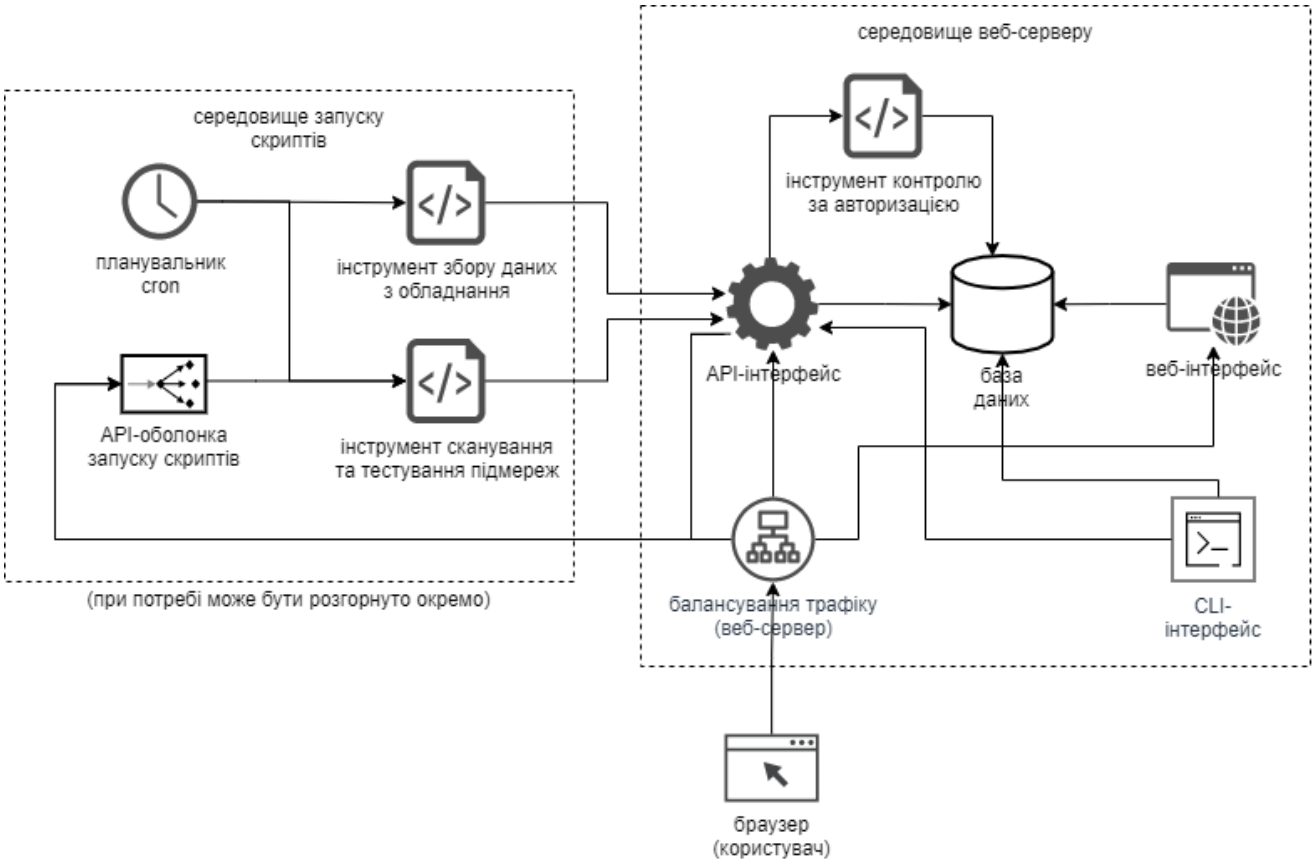


Рисунок 2.1 – Структурна схема програмного модулю

Основні компоненти програмного модулю мають наступне призначення:

База даних – один із основних компонентів використаних у роботі програмного модулю. У базі даних містяться всі робочі дані про комп'ютерні підмережі та їхні складові, а також дані про роботу скриптових інструментів. Зокрема у базі даних містяться дані про події та параметри адміністративного доступу до веб-інтерфейсу.

Веб-інтерфейс – один із компонентів та основний способів управління програмним модулем. Серверна частина веб-інтерфейсу, бекенд, переважно забезпечує розмежування прав доступу та виконання логіки роботи веб-інтерфейсу з базою даних. Фронтенд являє собою візуальну частину веб-інтерфейсу, яка працює у браузері користувача. Фронтенд, крім зв'язку з бекендом, має також зв'язок з API-інтерфейсом, звідки може отримувати дані таблиць та керувати процесами запуску скриптів API-оболонка для запуску скриптів.

API-інтерфейс – один із основних компонентів програмного модулю, який містить основний функціонал програмного модулю, має безпосередній зв'язок із базою даних. Надає методи доступу до модулю скриптовим інструментам, а також користувачам за допомогою CLI- та веб-інтерфейсу. Працює на базі технології HTTP REST, за допомогою якої стає можливо легко вбудовувати функціонал різних додатків, що працюють на стороні сервера із веб-інтерфейсами.

CLI-інтерфейс – засіб управління модулем із інтерфейсу командного рядка. Дозволяє оперативно змінювати адміністративні паролі, а також вручну запускати різні скриптові інструменти програмного модулю. Як і веб-інтерфейс має одночасно як безпосередній доступ до бази даних, так і доступ до API-інтерфейсу. Є досить корисним інструментом для виявлення проблем у роботі програмного модулю.

Планувальник завдань – скриптовий засіб що дозволяє запускати періодичні завдання, зокрема завдання для запуску скриптових інструментів. Планувальник завдань працює на базі системного інструментарію cron.

API-оболонка для запуску скриптів – спеціальне середовище, яке забезпечує можливість запуску скриптів та термінальний доступ до них з веб-інтерфейсу. Подібні засоби дозволяють перенаправляти потоки STDIN, STDOUT та STDERR через протокол HTTP, завдяки чому стає можливим управління інструментальними компонентами у яких є лише CLI-інтерфейс. Передбачається реалізація даного компоненту на базі бібліотеки Hyperwatch [20].

Скриптові інструменти програмного модулю – це скриптові файли, кожен із яких виконує лише одну певну функцію. Скрипти працюють як CLI-інструменти, звітуючи про свою роботу API-сервісу програмного модулю. Розширення функціоналу програмного модулю і його можлива еволюція до повноцінної системи моніторингу аудиту передбачаються завдяки написанню більшої кількості скриптових інструментів у майбутньому. Також при розробці API-сервісу програмного модулю планується передбачити можливість додання підтримки багатьох філіалів, і запуску скриптових інструментів на їх стороні (рис. 2.2).

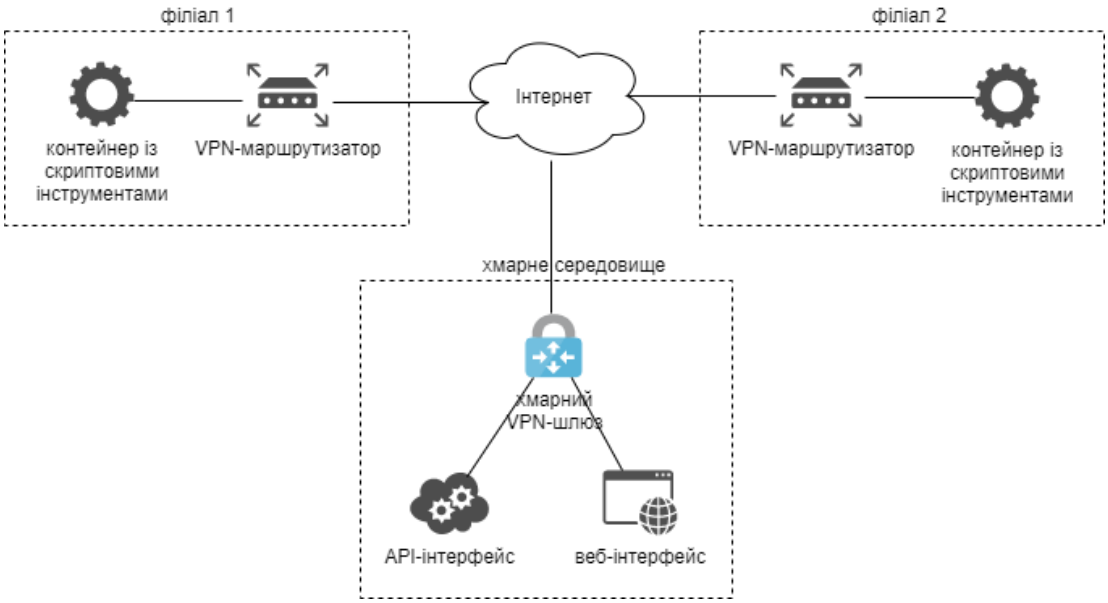


Рисунок 2.2 – Варіант хмарного розгортання модулю із кількома філіалами

2.2 Розробка структури бази даних

Розроблена архітектура програмного модулю передбачає використання бази даних для збереження інформації програмного модулю про комп’ютерні мережі та їх елементи, з якими працюватиме модуль. Для того, щоб зрозуміти якою має бути

структура його бази даних, необхідно знати з якими даними буде працювати сам додаток.

Оскільки в першу чергу програмний модуль працюватиме з мережами – він зберігатиме інформацію про підмережі, їх належність до того чи іншого класу, а також параметри виявлення підозрілих хостів. Основними сутностями при аналізі захищеності мережі є:

- підмережі;
- хости;
- порти (сервіси).

Скриптові інструменти та веб-інтерфейс повинні мати можливість зчитувати та записувати інформацію про стан цих об’єктів у базу даних. Перелік типів доступу компонентів програмного модуля до конкретних сутностей, які зберігатимуться у базі даних наведено у таблиці 2.1.

Таблиця 2.1 – Перелік типів об’єктів, до яких компоненти програмного модуля здійснюють доступ

Компонент	**Сутність**	**Тип доступу**
Інструмент збору даних з обладнання	Хости	Зчитування даних про параметри доступу до хосту
	Хости	Запис даних про належність хосту до мережі
	Підмережі	Запис даних про мережу
Інструмент сканування та тестування підмереж	Підмережі	Зчитування даних про мережу
	Хости	Запис даних про належність хосту до мережі
	Порти (сервіси)	Запис даних про сервіс хосту
Інструмент контролю за авторизацією	Події	Запис подій
Веб-інтерфейс	Підмережі	Перегляд, додавання, верифікація, редагування, видалення
	Хости	
	Порти (сервіси)	Перегляд, верифікація

Як видно найбільш очевидним варіантом реалізації бази даних є реляційна база даних. Реляційна база даних є сукупністю елементів даних, організованих у вигляді набору формально описаних таблиць, з яких дані можуть бути доступними

або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних.

Крім трьох ключових сутностей – підмереж, хостів та портів існують також інші сутності, такі як події, користувачі та сесії. Поділу привілеїв користувачів за групами доступу не передбачено у MVP-версії, оскільки дана сутність необхідна лише при розвитку даного рішення у повноцінну систему.

У відповідності до розробленої архітектури програмного модуля в універсальне відношення потрібно включити атрибути з наступних інформаційних об’єктів:

– підмережі (<код запису підмережі>, IP-адреса мережі, довжина маски, основний хост, клас безпеки мережі, параметри сканування);
– хости (<код запису хосту>, хостнейм, IP-адреса хосту, тип хосту, MAC-адреса хосту, параметри доступу);
– порти (<код запису порту>, час виявлення, час існування, номер порту, протокол порту, банер);
– події (<код запису події>, час виникнення, клас події, рівень важливості, ідентифікатор додатку, локальна IP-адреса, віддалена IP-адреса, опис);
– користувачі (<код запису користувача>, логін, хеш паролю, повне ім’я, електронна пошта, рівень доступу, активність);
– сесії (<код запису сесії>, токен доступу, дескриптор браузера, час входу, IP-адреса входу);

В таблиці 2.2 наведено перелік атрибутів універсального відношення.

Таблиця 2.2 – Перелік атрибутів універсального відношення

№	Назва атрибута	Ім'я поля (SQL)	Коментар
1	код запису підмережі	networks.net_id	ідентифікатор запису мережі у БД
2	IP-адреса мережі	networks.net_ip	IPv4- або IPv6-адреса мережі (без маски)
3	довжина маски	networks.length	довжина маски (кількість біт)

Продовження таблиці 2.2

№	Назва атрибута	Ім'я поля (SQL)	Коментар
4	основний хост	networks.master_host	ідентифікатор пристрою, який містить дані про наявні пристрої мережі (наприклад маршрутизатор або DHCP-сервер)
5	клас безпеки мережі	networks.net_class	буквено-числове значення, на основі якого визначається належність мережі до того чи іншого класу
6	параметри сканування	networks.scan_type	параметри сканування мережі: частота, протокол та порти
7	код запису хосту	known_hosts.host_id	ідентифікатор запису хосту у БД
8	хостнейм	known_hosts.host_name	ім’я хосту, отримана із DHCP-серверу або контролеру домену Active Directory
9	IP-адреса хосту	known_hosts.host_ip	IPv4- або IPv6-адреса хосту
10	тип хосту	known_hosts.class	тип хосту: маршрутизатор, комутатор, DHCP-сервер, контролер домену Active Directory або інше
11	MAC-адреса хосту	known_hosts.mac_addr	MAC-адреса хосту, отримана із DHCP-серверу або маршрутизатора
12	параметри доступу	known_hosts.snmp	параметри доступу до хосту по протоколу SNMP
13	код запису порту	discover_ports.record_id	ідентифікатор запису порту у БД
14	час виявлення	discover_ports.record_dt	час виявлення порту
15	час існування	discover_ports.expires	час через який інформація про сервіс на порту втратить свою актуальність
16	номер порту	discover_ports.port_number	номер порту TCP або UDP, 1-65535
17	протокол порту	discover_ports.port_proto	номер протоколу: TCP (6) або UDP (17)
18	банер	discover_ports.banner	банер сервісу, який використовує даний порт
19	код запису події	threat_events.record_id	ідентифікатор запису події у БД
20	час виникнення	threat_events.record_dt	час та дата у числовому форматі, коли сталась подія
21	клас події	threat_events.class	клас категорії, який відповідає за даний тип подій
22	рівень важливості	threat_events.level	рівень критичності події

Продовження таблиці 2.2

№	Назва атрибута	Ім'я поля (SQL)	Коментар
23	ідентифікатор додатку	threat_events.application_id	ідентифікатор додатку або скриптового інструменту, завдяки якому виявлено подію
24	локальна IP-адреса	threat_events.local_ip	якщо подія спричинена мережевим з'єднанням, тоді вказується IP-адреса серверу задіяного додатку
25	віддалена IP-адреса	threat_events.remote_ip	якщо подія спричинена мережевим з'єднанням, тоді вказується IP-адреса клієнту задіяного додатку
26	опис	threat_events.description	текстовий опис подіїї
27	код запису користувача	admin_users.id	ідентифікатор запису користувача у БД
28	логін	admin_users.login	логін користувача, який використовується для адміністративного входу
29	хеш паролю	admin_users.pwd_hash	хеш паролю користувача, який використовується для адміністративного входу
30	повне ім'я	admin_users.full_name	повне ім'я користувача
31	електронна пошта	admin_users.email	електронна пошта користувача
32	рівень доступу	admin_users.level	рівень адміністративного доступу користувача, 0-5
33	активність	admin_users.active	визначає чи здатен користувач заходити
34	код запису сесії	admin_session.record_id	ідентифікатор запису сесії користувача у БД
35	токен доступу	admin_session.session_id	токен PHP-сесії session_id
36	дескриптор браузера	admin_session.useragent	рядок User-Agent браузера, за допомогою якого заходитиме користувач
37	час входу	admin_session.login_time	час, коли заходитиме користувач
38	IP-адреса входу	admin_session.login_ip	IPv4- або IPv6-адреса з якої заходитиме користувач

Окремо варто виділити сутність хосту через необхідність прослідковувати зміну його станів та стану його портів. База даних має містити як верифіковані дані хосту, які змінюються лише у момент верифікації, так і дані про те, як змінювався стан хосту та його сервісів у той чи інший момент часу. База даних міститиме по

більшій мірі різні види даних про верифіковані хости та їх стан у різні моменти часу (рис. 2.3).

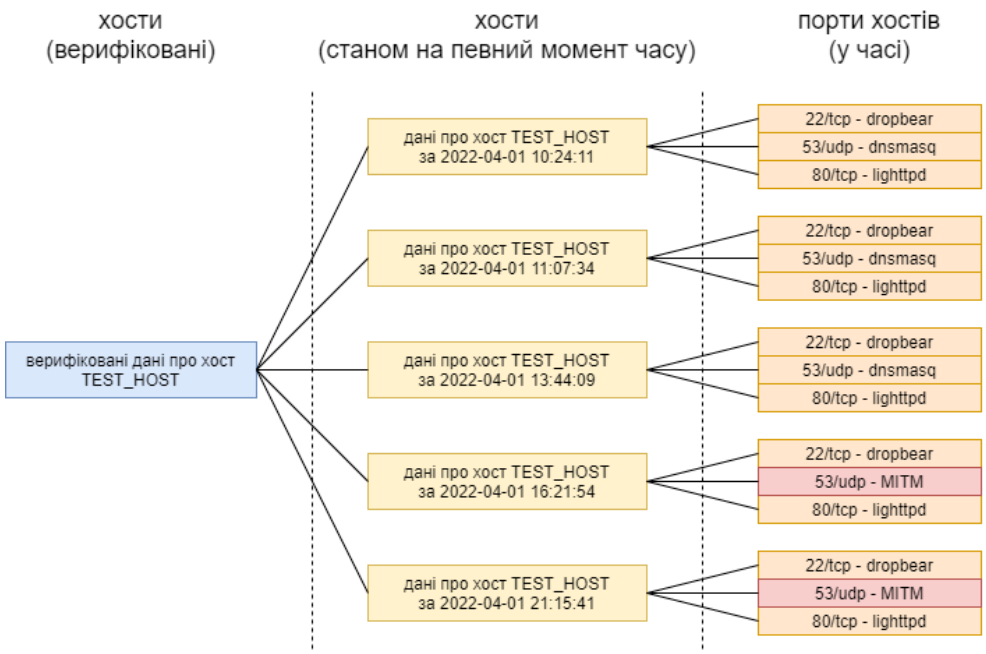


Рисунок 2.3 – Виявлення компрометації сервісу хосту з плином часу

Складність реалізації опису сутності хосту у структурі бази даних передбачає, що замість оптимізації дана сутність навпаки стане складнішою. Крім того інформація про хости та мережеві сервіси які на них працюють може досить швидко застарівати. Для того, щоб контролювати даний процес, необхідно буде визначити яким чином у базі даних будуть прослідковуватись неактуальні дані, і яким чином це впливатиме на верифіковані хости. Одним із способів контролювати застарілість інформації – визначати коли дані були створені, і видаляти занадто старі дані. Однак це може бути більш складно коли різні мережі матимуть різну частоту оновлення даних. Саме тому необхідно визначити які поля будуть мати лише верифіковані хости, а які будуть лише у їх знімків (таблиця 2.3).

У подальшій реалізації бази даних необхідно звернути увагу на можливість на збереження тимчасових даних хостів та портів у тимчасових таблицях, які при необхідності можна буде легко очищувати.

Таблиця 2.3 – Відмінності даних про верифіковані хости та їх змін у часі

Подібність полів	**Верифікований хост**	**Знімок хосту**
Спільні поля	код запису хосту (або знімку хосту)	
	ім’я хосту	
	IP-адреса	
	MAC-адреса	
	належність до мережі	
Індивідуальні поля	тип хосту	-
	параметри доступу	-
	-	*дата зняття інформації*
	-	*час актуальності інформації*

На основі інформаційних об’єктів бази даних в ER-моделі можна виділити такі сутності та ключі:

– підмережі (код запису підмережі);

– верифіковані хости (код запису верифікованого хосту);

– знімки хостів (код запису знімку хосту);

– порти (код запису порту);

– події (код запису події);

– користувачі (код запису користувача);

– сесії (код запису сесії).

Зв’язки екземплярів сутностей наведено у таблиці 2.4.

Таблиця 2.4 – Зв’язки екземплярів сутностей

Суть 1	**Суть 2**	**Тип зв’язку**	**Ім’я зв’язку**	**Клас належності**
верифіковані хости	підмережі	N:1	належать	обов.; обов.;
знімки хостів	підмережі	N:1	виявлені	обов.; обов.;
знімки хостів	верифіковані хости	N:1	описують стан на момент часу	обов.; обов.;
знімки хостів	порти	1:N	мають	обов.; обов.;
події	верифіковані хости	N:1	стосуються	обов.; обов.;
користувачі	верифіковані хости	1:N	верифікують	обов.; обов.;
сесії	користувачі	N:1	надають доступ	обов.; обов.;

Перший зв’язок являє собою поєднання верифікованих хостів та підмереж. Подібно верифікованим хостам, знімки також можуть бути виявлені у тій чи іншій мережі. Наявність дублюючого зв’язку у верифікованих хостів та їхніх знімків може забезпечити виявлення зміни мережі хосту. Користувач зможе верифікувати зміни у належності верифікованого хосту до тієї чи іншої мережі, і тоді запис верифікованого хосту прийме значення із знімку.

Записи про стан портів має прив’язуватись до знімків стану хостів, оскільки відкриті порти часто залежать від сервісів, які можуть працювати тимчасово або періодично.

Зв’язки між сутностями можна побачити на ER-діаграмі на рисунку 2.4.

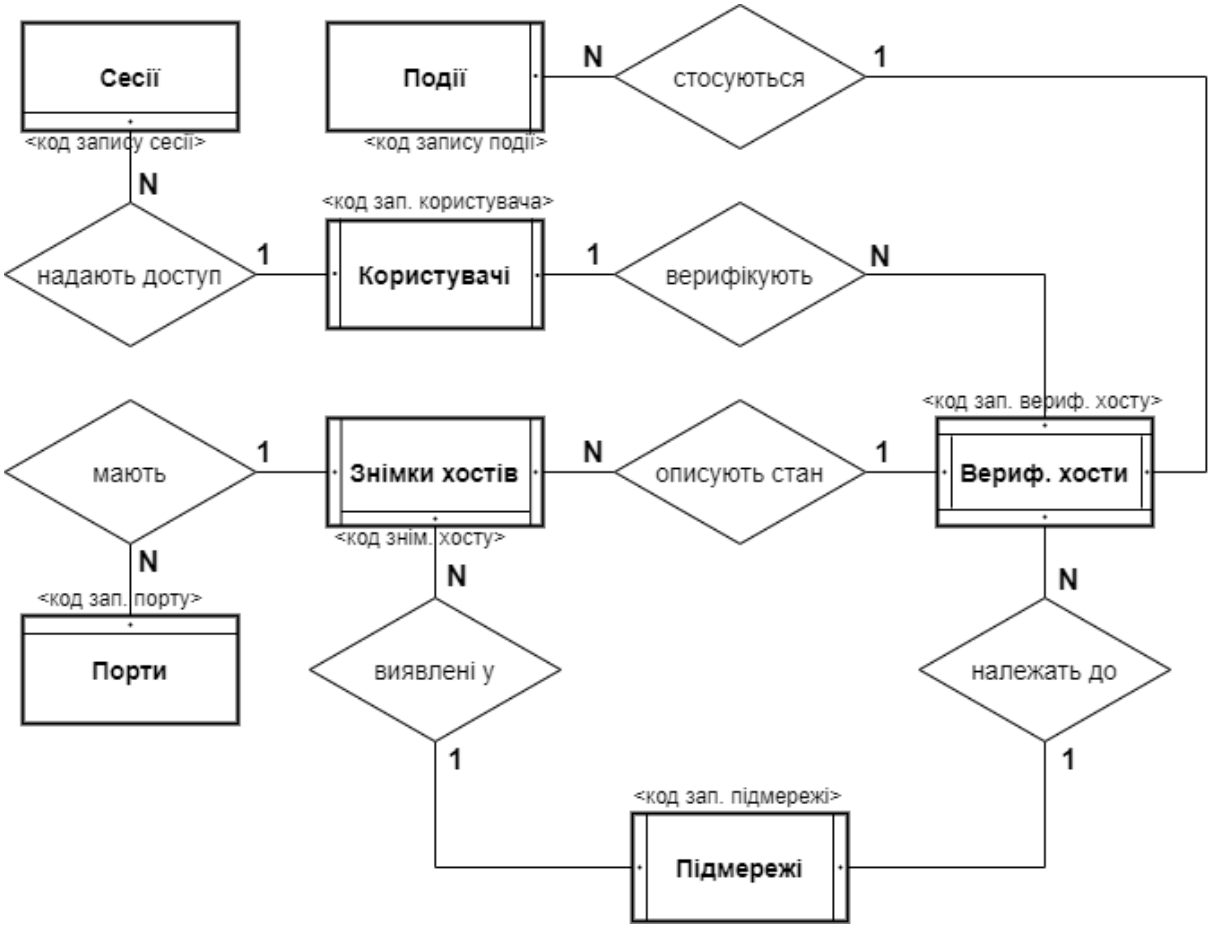


Рисунок 2.4 – ER-діаграма бази даних програмного модулю для аудиту захищеності локальних комп’ютерних підмереж

Для кращого сприйняття структури бази даних розроблено діаграму сутностей зображену на рисунку 2.5.

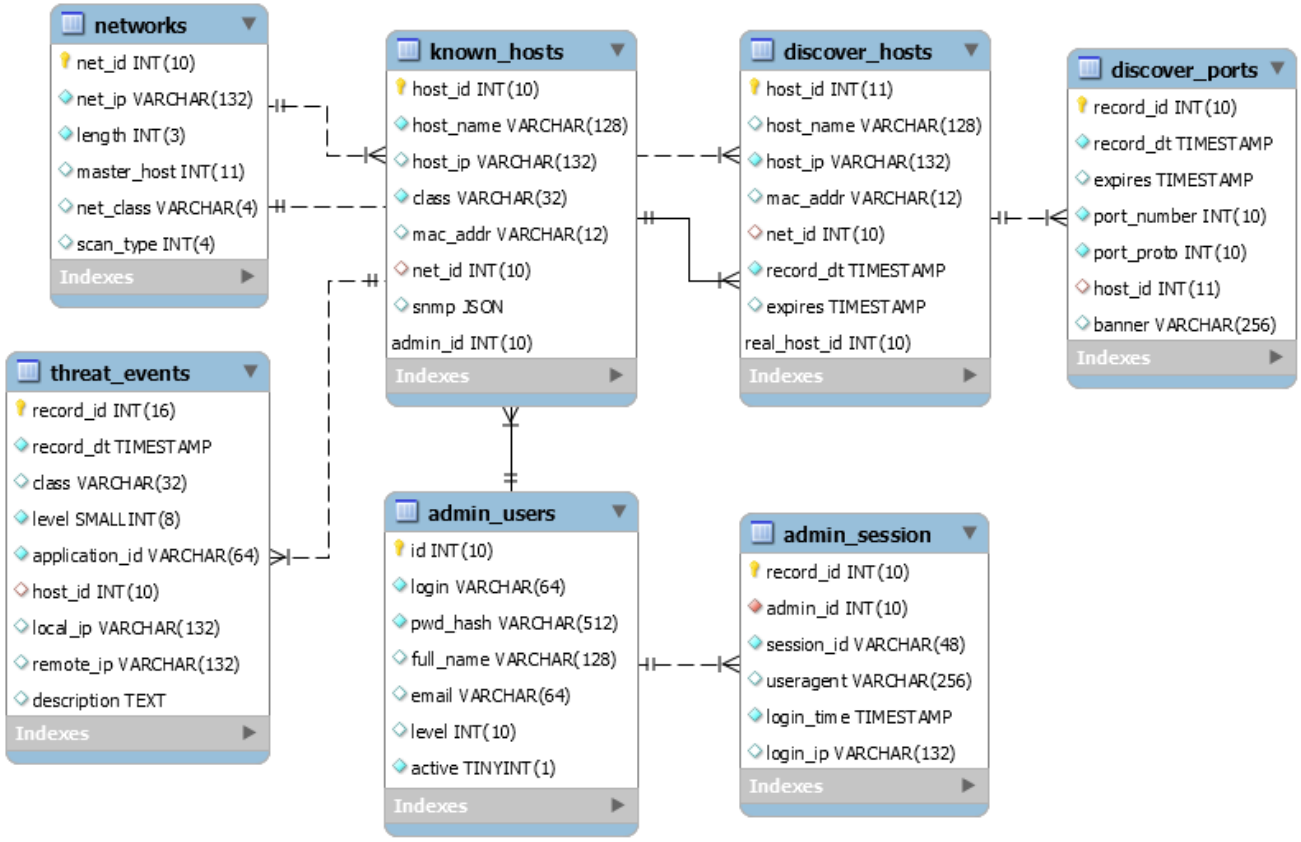


Рисунок 2.5 – Діаграма сутностей бази даних

Деякі види даних, такі як кеш даних SNMP, параметри даних, кеш згенерованих карт недоцільно зберігати у реляційних SQL базах даних. Їх більш доцільно зберігати у базах даних типу ключ-значення. Для розробки подібної бази даних необхідно спочатку обрати засоби для її реалізації, після чого буде більш доцільно застосувати підхід code-first при розробці API-інтерфейсу у 3-му розділі.

2.3 Розробка методу багаторівневого сканування мережі

Метод багаторівневого сканування мережі поєднує отримання даних зібраних безпосередньо з обладнання (поллінг) та з даних зовнішнього сканування (маппинг). Поєднання цих двох підходів дозволить розширити покриття більшої кількості рівнів у моделі TCP/IP.

Модель TCP/IP є підмножиною моделі OSI і є більш зручною з точки зору розробника, хоча для безпекового аудиту звісно модель OSI є більш практичною. Зважаючи на відмінності у представленні цих моделей (таблиця 2.5), варто зважати, що деякі рівні моделі OSI залишатимуться абстрактними з токи зору розробки методу багаторівневого сканування мережі [21-23].

Таблиця 2.5 – Порівняння моделей OSI та TCP/IP

Модель TCP/IP	**Модель OSI**	**Приклад**
Прикладний рівень	Прикладний рівень	FTP, HTTP, Telnet
	Представлення рівень	JPEG, HTML
	Сеансовий рівень	NFS, SQL, PAP
Транспортний рівень	Транспортний рівень	TCP, UDP
Рівень Інтернету	Мережевий рівень	IPv4, IPv6
Рівень мережевого доступу	Канальний рівень	ARP, CDP, STP
	Фізичний рівень	Ethernet, Wi-Fi

Метод багаторівневого сканування мережі передбачає поєднання методів пасивного та активного сканування мереж. Активне сканування здійснюється шляхом надсилання кількох тестових запитів та запису тестових відповідей на порти TCP та UDP [24]. Приклади активного сканера: Masscan, Nmap та Zmap. Пасивне сканування виконується не шляхом активного зондування, а шляхом простого прослуховування будь-яких даних, що надсилаються по мережі.

Зазвичай пасивне сканування передбачає використання обладнання, яке вміє віддзеркалювати трафік, наприклад з підтримкою протоколу NetFlow. Однак у разі авторизованого доступу до мережі у цьому немає необхідності, оскільки всі необхідні дані для роботи даного методу зазвичай вже наявні на мережевому обладнанні та/або серверах, і до них зазвичай є доступ по протоколу SNMP.

Метод багаторівневого сканування мережі дозволяє покрити канальний рівень завдяки інформації, яка буде отримуватись під час поллінгу скриптовим інструментом збору даних з обладнання. Дані про сканування під час першого

етапу зберігаються у реляційну БД, а також у сховище ключ-значення для подальшого використання під час маппингу (рис. 2.6).

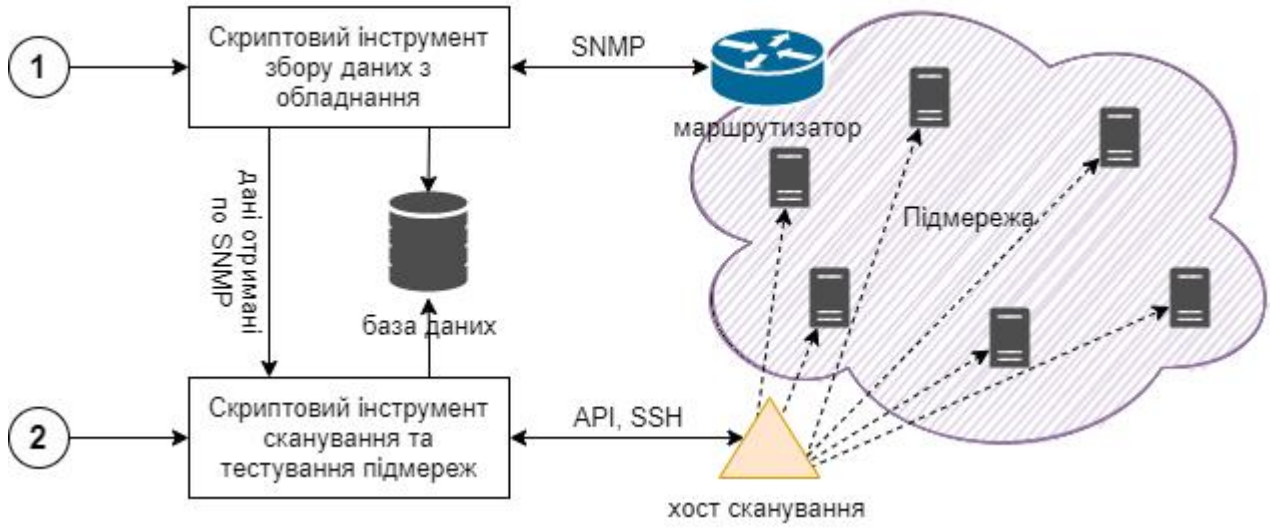


Рисунок 2.6 – Робота методу багаторівневого сканування мережі

Під час виконання другого етапу багаторівневого сканування згідно інформації, отриманої від обладнання здійснюється активне сканування. Активне сканування дозволяє легше покрити більш вищі мережеві рівні (мережевий і вище). Результати активного сканування записуються у таблиці баз даних для даних знімків стану хостів. При використанні багатьох хостів сканування збільшуються можливість виявлення компрометації доступу до хостів. Після сканування виявляються потенційні у сервісах (банери, відкриті ключі шифрування та SSL-сертифікати), після чого інформація про це потрапляє у таблицю подій як потенційно загрозлива.

2.4 Розробка методу побудови карти мережі

Побудова графічної карти хостів є однією із найбільш важливих функцій програмного модулю для аудиту захищеності локальних комп’ютерних підмереж. Метод побудови карти мережі будує карту, на які буде видно всі мережі, пристрої які їх об’єднують, класи безпеки, а також загальний стан мереж. Процес побудови карти розбитий на наступні кроки:

1. Згідно інформації про сканування виявити маршрутизатори які мають безпосередні зв’язок між собою. Наповнити масив зв’язків між маршрутизаторами;
2. Створити карту кожної мережі – визначити внутрішні та граничні хости (маршрутизатори чи ті які пов’язані з іншими мережами);
3. Виконати поділ хостів на повністю верифіковані, частково верифіковані та невідомі. Верифіковані матимуть менший радіус, а невідомі – більший. Згідно класу мережі визначити невідомі як вільно приєднані або потенційно небезпечні.
4. Побудова "островів": завдяки інформації про маршрути маршрутизаторів визначити які мережі безпосередньо з’єднані за допомогою тих чи інших маршрутизаторів і об’єднати у групи всі мережі між якими можлива маршрутизація (рис. 2.7).

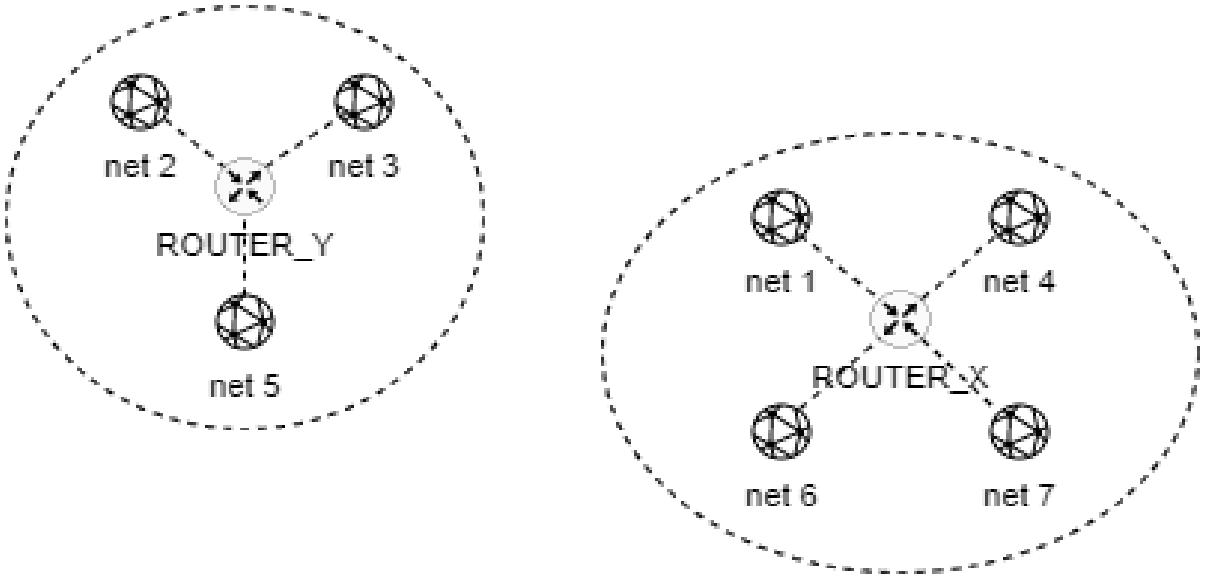


Рисунок 2.7 – Метод побудови карти мережі

5. Розподіл за класами безпеки. Розбити групи мереж на сектори по класам безпеки і зафарбувати однаковим для всього класу кольором (рис. 2.8).

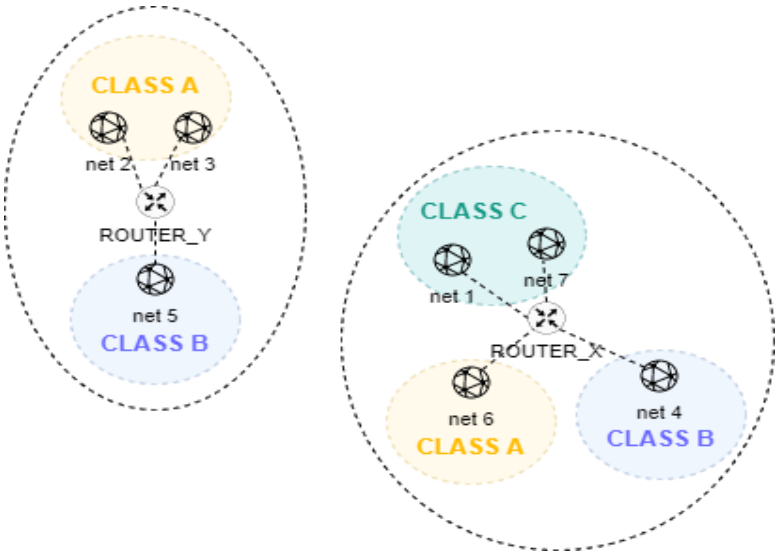


Рисунок 2.8 – Метод побудови карти мережі

6. Побудова "океану": визначити які маршрутизатори мають безпосередній доступ до мережі Інтернет,

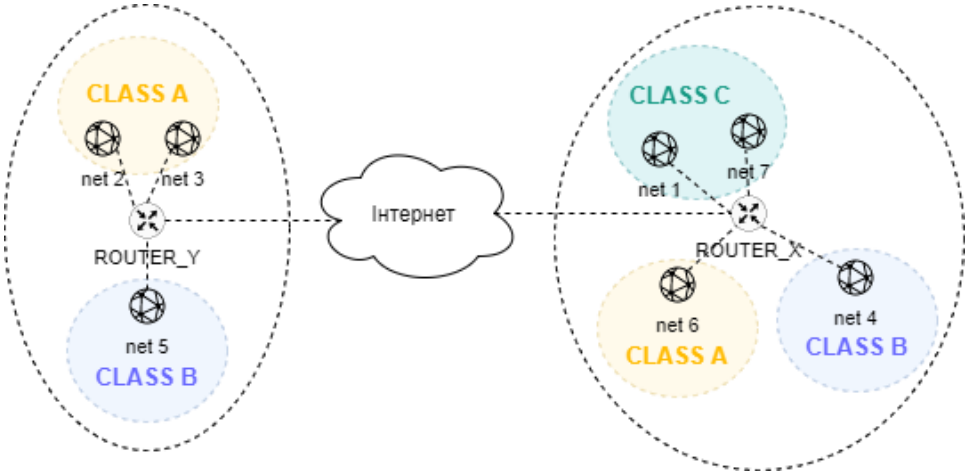


Рисунок 2.9 – Метод побудови карти мережі

2.5 Розробка алгоритмів роботи програмного модулю

Початок роботи з програмним модулем передбачає такі фази його запуску: встановлення, конфігурація та ініціалізація. Алгоритм початкового використання передбачає від користувача додаткове налаштування, якщо користувач цього не зробив раніше (рис. 2.10). Далі слідує повсякденне використання, в ході якого є можливою зміна його конфігурації.

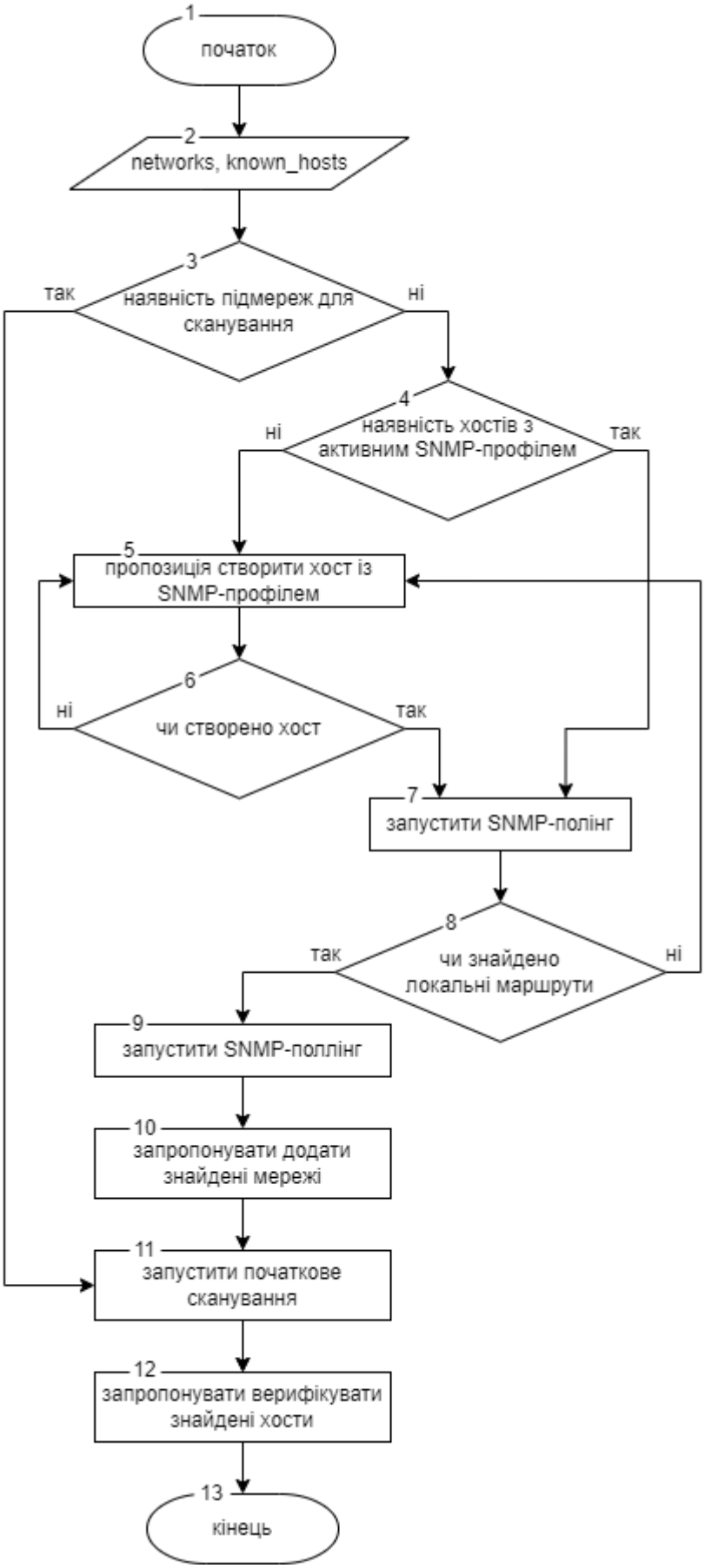


Рисунок 2.10 – Блок-схема ініціалізації програмного модуля

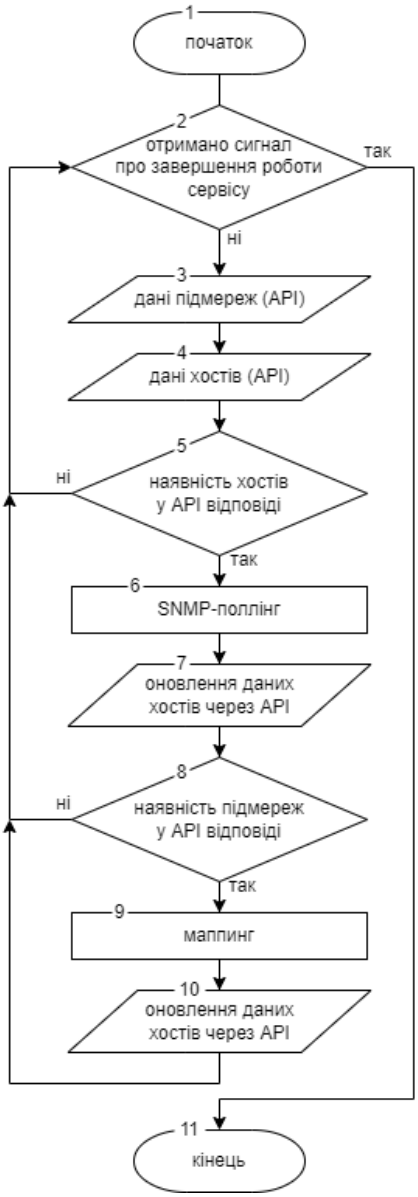


Рисунок 2.11 – Блок-схема звичайної роботи програмного модуля

2.6 Висновки

Розроблено архітектуру програмного програмного модулю для аудиту захищеності локальних комп’ютерних підмереж, а також можливості її розширення для різних варіантів розвитку. Розроблено структуру бази даних та принцип збереження даних про хости із плином часу. Розроблено метод опитування внутрішнього стану керуємого обладнання (поллінг); метод зовнішнього збору даних про доступні хости та мережеві сервіси (маппинг), а також розроблено алгоритми роботи програмного модулю.

3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Вибір засобів розробки для створення програмного модулю має насамперед базуватись на підтримці мережевих протоколів та технологій які безпосередньо необхідні для аудиту захищеності локальних комп'ютерних підмереж, а також для забезпечення швидкого розгортання, оновлення та зручного користування веб-інтерфейсом. Для початку необхідно проаналізувати можливості основних мов програмування, які використовуються для розробки програмного модулю.

Python – динамічна інтерпретована об'єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Має досить потужну пакетну базу з бібліотеками. Python, основні бібліотеки та його пакетний менеджер pip входять до програмних пакетів стандартних дистрибутивів GNU/Linux. Завдяки цьому його досить легко розгортати у контейнерних середовищах. Крім цього існує досить потужна підтримка опитування обладнання по протоколу SNMP за допомогою бібліотеки pysnmp. Через це прийнято рішення реалізувати на мові Python 3-ої версії скриптові інструменти для збору даних з обладнання і сканування підмереж.

PHP (рекурсивний акронім словосполучення PHP: Hypertext Preprocessor) – це поширена мова програмування загального призначення з відкритим кодом. PHP спеціально сконструйований для веб-розробок, і його код може впроваджуватися безпосередньо в HTML. На момент написання роботи PHP 7.0 має переваги на Python у високонавантажених веб-додатках завдяки своєму потужному движку Zend Engine 3.0 [25]. Саме тому прийнято рішення реалізувати на мові PHP 7-ої версії API-інтерфейс, інструмент контролю за авторизацією та backend-частину графічного веб-інтерфейсу програмного модулю.

Для зберігання даних обрано систему управління базами даних MariaDB. Ця система працює на базі MySQL та повністю з нею сумісна щодо синтаксису,

основних процесів та клієнтських бібліотек. Її перевагою є більш оптимізована реалізація таблиць, які знаходяться у пам’яті. Також досить потужною є підтримка зберігання json-об’єктів. Раніше розроблена структура бази даних для програмного моду є повністю сумісною з MariaDB. Реалізацію зв’язку з базою даних передбачається реалізувати на базі API-інтерфейсу.

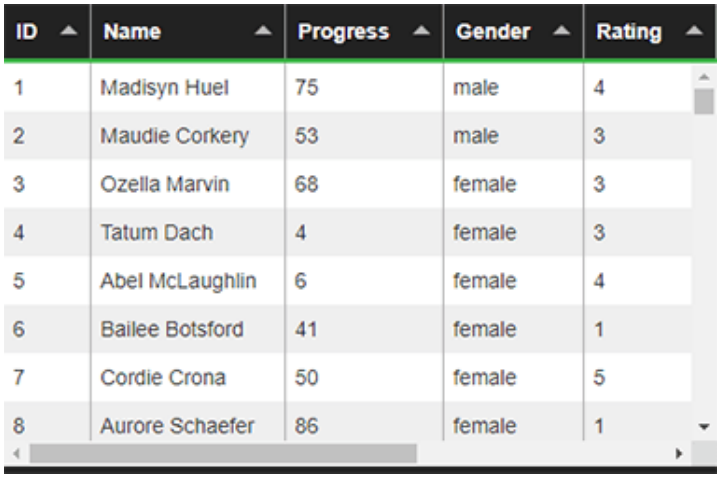
Для зберігання пар ключ-значення нетривалого збереження використовувати реляційну базу даних MariaDB є недоцільним. Для забезпечення зберігання цих даних обрано систему зберігання даних Redis. Подібне рішення дає більший виграш у швидкодії, за рахунок його високої оптимізованості. Redis надає схожі на Memcached функції для зберігання даних в форматі ключ/значення, розширені підтримкою структурованих даних, таких як списки, хеші і множини. На відміну від Memcached, Redis забезпечує постійне зберігання даних на диску і гарантує збереження БД у разі аварійного завершення роботи. Клієнтські бібліотеки доступні для більшості популярних мов, включаючи Perl, Python, PHP, Java, Ruby і Tcl, що робить його ідеальним рішенням для зберігання пар ключ-значення програмного модулю для аудиту захищеності локальних комп’ютерних підмереж.

Вибір frontend-бібліотек для графічного веб-інтерфейсу програмного модулю зосереджено на Bootstrap, jQuery та Tabulator. Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебдодатків. Frontend-бібліотека Bootstrap реалізована на базі CSS та JavaScript. Він спрощує розробку динамічних вебсайтів і веб-додатків.

jQuery – це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також дана бібліотека надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Tabulator – це frontend-бібліотека що дозволяє створювати інтерактивні таблиці за лічені секунди з будь-якої таблиці HTML, масиву Javascript або даних у форматі JSON. Дана бібліотека надає наступні можливості: застосування фільтрів по рядкам таблиці за певним параметром, сортування записів, групування за

певною ознакою, вибір рядків, підтримка responsible-розмітки та Ajax-запитів, редагування даних, пагінація, локалізація, можливість повернення змін, а також конвертація таблиць у форматі CSV та XLSX.



ID	Name	Progress	Gender	Rating
1	Madisyn Huel	75	male	4
2	Maudie Corkery	53	male	3
3	Ozella Marvin	68	female	3
4	Tatum Dach	4	female	3
5	Abel McLaughlin	6	female	4
6	Bailee Botsford	41	female	1
7	Cordie Crona	50	female	5
8	Aurore Schaefer	86	female	1

Рисунок 3.1 – Приклад використання бібліотеки Tabulator

Для розробки обрано Visual Studio Code, який поєднує в собі простоту редактора коду з тим, що потрібно розробникам для їх основного циклу редагування, збирання та налагодження. Він забезпечує повну підтримку редагування коду, навігацію та розуміння, а також легке налагодження, багату модель розширюваності та легку інтеграцію з наявними інструментами [26].

Visual Studio Code є крос платформним, він працює на Windows, macOS та Linux, а також має потужну підтримку Python, PHP, JavaScript, HTML/CSS, а також SSH, Docker та GitHub.

3.2 Розробка API-інтерфейсу програмного модулю

API-інтерфейс є ядром програмного модулю, оскільки він має безпосередній зв’язок зі всіма компонентами модулю, зокрема базою даних. Під час розробки архітектури у 2-му розділі даному компоненту відведено ключову роль, як це можна побачити на рисунку 2.1.

Існують наступні основні типи API:

– API бібліотек та фреймворків;

– API операційних систем;

– Віддалені API;

– Web API.

API-інтерфейс програмного модулю належить до типу Web API (які працюють на базі протоколу HTTP), а саме до REST API (орієнтованого на доступ до конкретних ресурсів).

Згідно ролі компонента, визначеного архітектурою, він має безпосередній доступ до баз даних. Модуль встановлює безпосередній зв'язок із базою даних MariaDB. Крім реляційних баз даних у API-інтерфейсі також має бути використана база даних ключ-значення. У базі такій базі Redis будуть зберігатись наступні дані:

– результат роботи скриптових інструментів (звіти);

– кеш відповідей SNMP;

– користувацькі параметри інтерфейсу.

Згідно з розробленою раніше структурою реляційної бази даних (рис. 2.5) розроблено ряд основних API-методів, які забезпечують взаємодію скриптових інструментів з реляційною та нереляційною базами даних, а також із графічним веб-інтерфейсом (таблиця 3.1).

Таблиця 3.1 – Перелік основних API-методів програмного модулю

Шлях URI	**Метод доступу**	**Опис**
`/api/networks/list.do`	GET	Даний метод забезпечує отримання масиву всіх активних підмереж, у яких необхіно проводити сканування. Дані отримуються з таблиці БД `networks`
`/api/snmp/list.do`	GET	Отримує масив хостів, що мають підтримку доступу через SNMP. Кожен елемент масиву містить IP-адресу, версію та параметри доступу SNMP. Дані отримуються з таблиці БД `known_hosts`
`/api/snmp/update.do`	POST	Публікує інформацію про дані SNMP у структурі ключ-значення
`/api/hosts/list.do`	GET	Отримує масив хостів, як тих, що пройшли верифікацію, так і тих, що ще не верифіковані. Дані отримуються з таблиць БД `known_hosts` та `discover_hosts`

Продовження таблиці 3.1

Шлях URI	Метод доступу	Опис
`/api/hosts/update.do`	POST	Додає виявлені хости у таблицю БД `discover_hosts`
`/api/reports/getid.do`	GET	Шукає чи існує ключ звіту у структурі ключ-значення
`/api/reports/push.do`	POST	Публікує звіт у структурі ключ-значення
`/api/events/push.do`	POST	Публікує подію (інформативну або таку, яка може бути потенційно загрозливою) у таблиці `threat_events`

Для доступу до API-методів необхідно створити один або кілька токенів доступу для скриптових інструментів. Токени зберігаються у вигляді json-масиву у файлі /app/config/api-hash.json (створюється при розгортанні).

Параметри доступу доступу до бази даних знаходяться у файлі /app/config/database.json (створюється при розгортанні). Код для розгортання структури бази даних MariaDB знаходиться у додатку Г (файл netmaster.sql).

3.3 Розробка інструменту збору даних з обладнання

Скриптовий інструмент збору даних працює на базі процесу поллінгу через протокол SNMP та написаний на мові програмування Python. Існують три основні компоненти архітектури управління мережею, які необхідно розуміти перед розробкою скрипкового інструменту для збору даних з обладнання: керуючий об’єкт, керований об’єкт і протокол управління мережею [27].

– Керуючий об’єкт – це програма, що працює на централізованій станції керування мережею (NMS). Це керуючий об’єкт, який контролює збір, обробку, аналіз та/або відображення інформації керування мережею. Саме тут ініціюються дії для контролю поведінки мережі, і тут адміністратор мережі взаємодіє з мережевими пристроями. У даному випадку це скриптовий інструмент збору даних з обладнання.
– Керований об’єкт, як правило, є апаратним або програмним додатком, який знаходиться в керованій мережі. Він перераховує та формалізує деякі його

властивості та стани, важливі для здорової роботи, таким чином роблячи їх доступними для керуючого суб’єкта. Наприклад, керованим об’єктом може бути хост, маршрутизатор, комутатор, принтер або будь-який інший пристрій.

– Третя частина системи управління мережею – це протокол управління мережею. Протокол працює між керуючою сутністю та керованою сутністю, дозволяючи керуючому об’єкту запитувати статус керованого об’єкта та змушувати останній виконувати дії через своїх агентів.

Протокол SNMP складається з чотирьох частин:

– Визначення об’єктів керування мережею, відомих як об’єкти MIB. Управлінська інформація представлена як сукупність керованих об’єктів, які разом утворюють віртуальне сховище інформації, відоме як база інформації управління (MIB). Об'єкт MIB може бути лічильником, описовою інформацією, наприклад версією програмного забезпечення; інформація про стан, наприклад, чи справний пристрій, або інформація про протокол, наприклад, шлях маршрутизації до місця призначення. Таким чином, об’єкти MIB визначають інформацію керування, яку підтримує керований вузол. Пов'язані об'єкти MIB збираються в так звані модулі MIB.
– Мова визначення даних, що називається SMI (Структура управлінської інформації), яка вводить базові типи даних, дозволяє створювати їх підтипи та більш складні структури даних. Об'єкти MIB виражаються цією мовою визначення даних.
– Протокол (SNMP) для передачі інформації та команд між керуючими та керованими об’єктами. SNMP розроблено на основі моделі клієнт-сервер. Важливо розуміти, що і керовані, і керовані об’єкти містять клієнтські та серверні компоненти.
– Розширювана структура безпеки та можливості системного адміністрування.

Отже, для потреб роботи скриптового інструменту збору даних з обладнання необхідні дані із таких OID (об’єктів MIB):

– 1.3.6.1.2.1.4.22 (ipNetToMediaTable) – таблиця ARP, де міститься прив’язка IP-адрес до MAC-адрес. Подібна прив’язка дозволяє отримувати інформацію про наявні у активних хостів MAC-адреси, і виявляти їх навіть у випадку якщо їх неможливо виявити методом сканування;
– 1.3.6.1.2.1.4.24 (ipForward) – таблиця з даними про мережеві маршрути пристрою. Завдяки метриці можна визначити які маршрути безпосередньо приєднані до хоста. Як правило у всіх мережевих інтерфейсів метрика є рівною нулю. Це дозволить визначити до яких мереж пристрій має безпосередній доступ;
– 1.3.6.1.4.1.9.10.102.1.4.1 (cDhcpv4ServerSharedNetTable) – дозволяє отримувати дані про використання мереж DHCP-сервером, і таким чином мати інформацію у яких мережах скільки видано IP-адрес.

Коли скриптовий інструмент розпочинає роботу, він зчитує адресу серверу, на якому встановлений API-інтерфейс та токен із файлу /app/config/api-server.json (створюється при розгортанні), що зображено на рисунку 3.2.

```
with open('app/config/api-server.json') as api_server:
    config = json.load(api_server)

api_vars = {
    "api_key": config['api_key'],
    "toolkit_ip": socket.gethostbyname( socket.gethostname() )
}
```

Рисунок 3.2 – Лістинг інструменту, що зчитує параметри доступу до API

Наступним чином відбувається підключення до серверу, в ході якого отримуються дані із методу /api/snmp/list.do, рисунок 3.3.

```
api_url = config['api_base'] + "snmp/list.do" + "?" +
urllib.parse.urlencode( api_vars )
api_request = urllib.request.urlopen(api_url)
request_body = api_request.read().decode("utf-8")
api_data = json.loads(request_body)
```

Рисунок 3.3 – Лістинг отримання даних із API-методу

Якщо дані підключення успішно отримані, тоді запускається цикл, в ході якого відбувається перевірка даних підключення до SNMP-хостів, а також встановлюється версія протоколу, що зображено на рисунку 3.4.

```
for snmp_host in api_data:
    arp_mib = {}
    if 'port' in snmp_host: snmp_host['port'] = int(snmp_host['port'])
    else: snmp_host['port'] = 161
    snmp_conn = UdpTransportTarget( (snmp_host['ip'], snmp_host['port']) )
    if 'version' in snmp_host: snmp_host['version'] =
int(snmp_host['version'])
    else: snmp_host['version'] = 2
    if snmp_host['version'] == 1 or snmp_host['version'] == 2:
        snmp_cred = CommunityData( snmp_host['community'],
mpModel=snmp_host['version']-1 )
    if snmp_host['version'] == 3:
        snmp_cred = UsmUserData(snmp_host['user'],
authKey=snmp_host['password'])
```

Рисунок 3.4 – Лістинг встановлення версії протоколу та верифікації параметрів

При коректності отриманих даних, скриптовий інструмент підключається до SNMP-хосту. Одночасно з підключенням запускається відповідна команда, у першому випадку це буде отримання таблиці ipNetToMediaTable, рисунок 3.5.

```
mac_poller = bulkCmd(
    SnmpEngine(),
    snmp_cred,
    snmp_conn,
    ContextData(),
    1, 25,
    ObjectType(ObjectIdentity('IP-MIB', 'ipNetToMediaTable')),
    lexicographicMode=False
)
```

Рисунок 3.5 – Лістинг команди отримання таблиці

Метод bulkCmd створений спеціально для роботи із таблицями. З допомогою нього можна отримувати одразу багато SNMP-полів. Протокол SNMP працює на базі UDP, що дозволяє отримувати дані у асинхронному режимі, що при необхідності дозволяє робити бібліотека pysnmp. Помилки зі з'єднанням та виконанням команд також обробляються у коді скриптового інструменту, що зображено на рисунку 3.6.

```
for errorIndication, errorStatus, errorIndex, varBinds in mac_poller:
    if errorIndication:
        print(errorIndication)
        break
    elif errorStatus:
        print('%s at %s' % (errorStatus.prettyPrint(),
            errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
        break
```

Рисунок 3.6 – Лістинг обробки помилок SNMP

Існує також цикл для обробки полів SNMP-таблиці, що зображений на рисунку 3.7. Цикл формує з них словник для подальшого перетворення їх у формат JSON.

```
for varBind in varBinds:
    mib_num = str(varBind[0]).split('.')
    for x in range(11): mib_num.pop(0)
    ip_num = '.'.join(mib_num)
    if (
        varBind[1].__class__.__name__ == 'IpAddress' or
        varBind[1].__class__.__name__ == 'PhysAddress'
    ) and (
        ip_num not in arp_mib
    ):
        arp_mib[ip_num] = {}

    if varBind[1].__class__.__name__ == 'IpAddress':
        arp_mib[ip_num]['ip'] = varBind[1].prettyPrint()
    if varBind[1].__class__.__name__ == 'PhysAddress':
        arp_mib[ip_num]['mac'] = varBind[1].prettyPrint()
```

Рисунок 3.7 – Форматування таблиці у словник

Після отримання даних із кожного хосту запускається метод update_host_data(), в якому дані перетворюються у формат JSON для подальшої передачі, що зображено на рисунку 3.8.

```
def update_host_data(snmp_host_ip, data):
    arp_out = []
    for pair in data.values(): arp_out.append(pair)
    json_data = json.dumps( { 'arp_table': arp_out } )
    api_url = config['api_base'] + "snmp/update.do"
    api_vars = urllib.parse.urlencode( {
        "api_key": config['api_key'],
        "toolkit_ip": socket.gethostbyname( socket.gethostname() ),
        "host_data": json_data
    } ).encode()
    req = urllib.request.Request(api_url, api_vars)
    resp = urllib.request.urlopen(req)
    print('[POLLER] Host ' + snmp_host_ip + ': collected ' +
str(len(arp_mib)) + ' ARP records')
    return resp.read(), resp.getheaders()
```

Рисунок 3.8 – Форматування даних у JSON та їх передача на сторону API

Отриманий список прив’язок MAC- та IP-адрес передається на серверну сторону через API. Серверний API-інтерфейс препарує їх та вписує у відповідні таблиці реляційної БД та а також бази ключ–значення.

3.4 Розробка інструменту сканування та тестування підмереж

Основною задачею інструменту сканування та тестування підмереж є оцінка захищеності сервісів у мережі, а також прослідковування чи не були вони скомпрометовані при доступі із інших точок мережі. Завдяки роботі по методу багаторівневого сканування, можливості виявлення різних прихованих хостів збільшуються. Компонент реалізовано на мові програмування Python

Nmap ("Network Mapper") – це утиліта і бібліотека з відкритим вихідним кодом для дослідження мережі та перевірки безпеки. Вона була розроблена для швидкого сканування великих мереж, хоча чудово справляється і з одиничними цілями. Nmap використовує "сирі" IP пакети оригінальним способом, щоб визначити які хости доступні в мережі, які служби (назва програми та версію) вони пропонують, які операційні системи (і версії ОС) вони використовують, які типи пакетних фільтрів/брандмауерів використовуються і ще багато інших параметрів. У той час, як Nmap зазвичай використовується для перевірки безпеки, багато системних адміністраторів знаходять її корисною для звичайних завдань, таких як контроль структури мережі, управління розкладами запуску служб та облік часу роботи хоста або служби [28].

У проекті використовується його версія у вигляді бібліотеки Python – python-nmap. Код з'єднання компонента з API-інтерфейсом програмного модулю та відправлення отриманих даних відбувається схожим чином, як у скриптовому інструменті збору даних з обладнання, відрізняється лише API-метод. Це зображено на рисунку 3.9.

```
available_hosts = []
api_url = config['api_base'] + "hosts/list.do" + "?" +
urllib.parse.urlencode( api_vars )
api_request = urllib.request.urlopen(api_url)
request_body = api_request.read().decode("utf-8")
api_data = json.loads(request_body)
if not len(api_data) > 0:
    raise RuntimeError('Blank API output')
```

Рисунок 3.9 – Отримання цілей сканування

API-метод надає скриптовому інструменту перелік хостів для сканування, частоту їх сканування, належність до тієї чи іншої мережі, а також клас захищеності. Після цього безпосередньо запускається процес сканування, який зображено на рисунку 3.10.

```
for scan_host in api_data:
    if 'ports' in scan_host: scan_host['ports'] = str(scan_host['ports'])
    nm = nmap.PortScanner()
    if 'ports' not in scan_host:
        nm.scan(scan_host['ip'], arguments='-A')
    else:
        nm.scan(scan_host['ip'], scan_host['ports'], arguments='-A')
```

Рисунок 3.10 – Запуск процесу сканування хостів

Після проходження сканування на API-сервер відправляються всі дані про стан хосту та працюючих на ньому сервісів, що зображено на рисунку 3.11.

```
available_hosts.append( {
    'ip': scan_host['ip'],
    'state': nm[scan_host['ip']].state(),
    'hostname': nm[scan_host['ip']].hostname(),
    'ports': {
        'tcp': nm[scan_host['ip']]['tcp'],
        'udp': nm[scan_host['ip']]['udp'],
    }
} )
```

Рисунок 3.11 – Формування структури відповіді для API

Всі подальші дії з аналізом даних відбуваються на стороні серверу, та включають роботу з даними про конкретний хосту у більш широкому часовому проміжку завдяки збереженню у базі даних більш ранніх їх станів.

3.5 Розробка інструменту контролю за авторизацією

Оскільки програмний модуль є засобом для аудиту захищеності локальних комп'ютерних підмереж, необхідно щоб і сам модуль мав достатній захист даних від несанкціонованого доступу. Є наступні обмеження, які дають можливість виключити несанкціонований доступ:

– API-токени – засіб API-інтерфейсу від несанкціонованого доступу. Він захищає API-методи. Хеші токенів зберігаються у файлі /app/config/api-hash.json (створюється при розгортанні). Оригінали токенів знаходяться у конфігураційних файлах скриптових інструментів.

– PHP-сесії – засіб обмеження доступу лише для користувачів, які вже виконали вхід через форму входу. Здатен захищати як веб-інтерфейс, так і API-методи, призначені для перегляду з браузера. Дані сесій зберігаються у базі ключ–значення Redis.

Виконання функції реєстрації сесій, перевірки правильності даних користувача та їх паролів покладено на PHP-клас NetLogin. Він має безпосередній зв'язок із базою даних У нього є три найбільш важливі методи. Перший із них, SignIn(), призначений для перевірки правильності даних користувача та їх паролів. Цей метод зображений на рисунку 3.12.

```
public function SignIn($user, $password) {
    $hash = NetLogin::PwdHash($password);
    $user = $this->link->escape_string($user);
    $sql  = $this->link->query("SELECT id AS user_id,login AS user_login,
full_name, email AS user_email, active AS user_active FROM admin_users
WHERE login='$user' AND pwd_hash='$hash' AND active=1");
    if($sql === false) return false;
    if($sql->num_rows <= 0) return false;
    $this->user_data = $sql->fetch_assoc();
    return true;
}
```

Рисунок 3.12 – Метод для авторизації користувача

Наступний метод дозволяє прослідковувати активних користувачів у базі даних з метою створення додаткового шару перевірки їх прав доступу при write-операціях. Цей метод зображено на рисунку 3.13.

```
public function RegSession() {
    $login_ip = $_SERVER['REMOTE_ADDR'];
    $user_id = intval($this->user_data['user_id']);
    $session_id = $this->link->escape_string(session_id());
    $useragent = $this->link->escape_string($this->session-
>session_useragent);
    $sql  = $this->link->query("INSERT INTO admin_session (admin_id,
session_id, login_ip, useragent) VALUES ('$user_id', '$session_id',
'$login_ip', '$useragent')");
}
```

Рисунок 3.13 – Метод реєстрації сесії корстувача

Також існує метод, який анулює роботу сесії, приведений на рисунку 3.14.

```
public function UnregSession() {
    $session_id = $this->link->escape_string(session_id());
    $sql  = $this->link->query("DELETE FROM admin_session WHERE session_id
= '$session_id'");
}
```

Рисунок 3.14 – Метод знищення сесії

Модуль NetLogin є окремим модулем який працює разом із API-інтерфейсом та графічним інтерфейсом у спільному середовищі php-fpm.

3.6 Розробка графічного веб-інтерфейсу програмного модулю

Розробку графічного інтерфейсу програмного модулю виконано у середовищі розробки Visual Studio Code. Веб-інтерфейс поєднує у собі наступні технології: PHP, node.js, Bootstrap, Font Awesome та Tabulator. Збірка фронтенд-бібліотек відбувається завдяки пакетному менеджеру npm. Як видно із файлу конфігурації меню приведеному на рисунку 3.15, виділено такі основні розділи веб-інтерфейсу: головна сторінка, хости, підмережі, карта мережі та події.

```php
<?php if(!defined('NET_APP_BASE'))
    die();

return array(
    'frontpage' => [
        'href' => NET_APP_BASE_URI,
        'descr' => 'Загальне',
        'icon' => 'far fa-fw fa-user',
    ],
    'host' => [
        'href' => NET_APP_BASE_URI . 'host/',
        'descr' => 'Хости',
        'icon' => 'fas fa-fw fa-server',
    ],
    'network' => [
        'href' => NET_APP_BASE_URI . 'network/',
        'descr' => 'Підмережі',
        'icon' => 'fas fa-fw fa-network-wired',
    ],
    'map' => [
        'href' => NET_APP_BASE_URI . 'map/',
        'descr' => 'Карта мережі',
        'icon' => 'fas fa-fw fa-map-marked-alt',
    ],
    'log' => [
        'href' => NET_APP_BASE_URI . 'log/',
        'descr' => 'Події',
        'icon' => 'fas fa-fw fa-history',
    ],
);
```

Рисунок 3.15 – Код побудови меню графічного веб-інтерфейсу

Завдяки API-серверу виникла можливість створити більш якісну фроненд частину. Спростити її створення вдалось з використанням фреймворку Bootstrap 4. Завдяки можливостям бібліотеки Tabulator вдалось зробити акцент на розробці досить потужного табличного графічного інтерфейсу. На рисунку 3.16 приведена JavaScript-функція, що забезпечує роботу майже цілої сторінки перегляду подій:

```
function update_log_table(json_params) {
    json_url = $('#log-table').data('json');
    start_page = $('#log-table').data('start-page');
    var table = new Tabulator("#log-table", {
        layout:"fitColumns",
        pagination:"remote",
        paginationSize:6,
        ajaxURL:json_url,
        ajaxParams:json_params,
        paginationSize:10,
        paginationSizeSelector:[10, 25, 50, 75, 100],
        paginationInitialPage:start_page,
        columns:[
            {title:"Подія", field:"action", headerSort:false,
formatter:action_formatter},
            {title:"Користувач", field:"login", widthGrow:2,
headerSort:false, formatter:user_formatter},
            {title:"Додаток", field:"app", headerSort:false,
formatter:secondary_formatter},
            {title:"IP-Адреса", field:"remote_ip", headerSort:false,
formatter:secondary_formatter},
            {title:"Час", field:"dt", headerSort:false,
formatter:secondary_formatter},
        ],
        dataLoaded:function(data){
            if(table_firstload) table.setPage(start_page); table_firstload
= false;
        },
        pageLoaded:function(page_no){
            if(!search_tag || search_tag == '') left_url =
location.protocol + '//' + location.host + location.pathname + '?page=' +
page_no;
            else left_url = location.protocol + '//' + location.host +
location.pathname + '?search=' + encodeURIComponent( search_tag ) +
'&page=' + page_no;
            history.replaceState(null, "Page " + page_no, left_url);
        },
    });
}
```

Рисунок 3.16 – Функція перегляду подій на стороні браузеру

Інші частини коду графічного веб-інтерфейсу програмного модулю для аудиту захищеності локальних комп’ютерних підмереж знаходяться у додатку Г.

3.7 Налаштування розгортання програмного модулю

Для можливості швидкого розгортання прийнято рішення використовувати контейнеризацію. Docker – це програмна платформа для швидкої розробки, тестування і розгортання додатків. Дана платформа упаковує програмне забезпечення у контейнери. За допомогою даної платформи можна налаштувати автоматичне розгортання наступних компонентів: бібліотеки, системні

інструменти, код і середовище виконання. Таким чином налаштовано розгортання програмного модулю.

Для запуску таких сервісів як lighttpd, php-fpm та скриптових інструментів використовується система cron. Головні сервіси запускаються при старті системи, а додаткові компоненти запускаються по таймеру cron чи за потреби.

Бази даних MariaDB та Redis налаштовуються за допомогою інструментарію Docker Compose. Із контейнером програмного модулю їх поєднують спільні ENV-змінні. Конфігураційний файл Docker приведено на рисунку 3.17:

```
FROM ubuntu:20.04
COPY etc/php-fpm.conf /etc/php/7.4/fpm/pool.d/www.conf
COPY etc/lighttpd.conf /etc/lighttpd/lighttpd.conf
COPY package.json requirements.txt /var/www/
COPY etc/cron.conf /etc/cron.d/web-cron
WORKDIR /var/www

RUN apt update && apt install -y \
    libpng-dev \
    libjpeg62-turbo-dev \
    libfreetype6-dev \
    snmp-mibs-downloader \
    build-essential \
    net-tools \
    lighttpd \
    locales \
    unzip \
    snmp \
    nmap \
    zip \
    git \
    cron \
    curl \
    bash \
    python3 \
    nodejs \
    pip \
    npm \
    php \
    php-cli \
    php-fpm \
    php-json \
    php-common \
    php-mysql \
    php-zip \
    php-gd \
    php-curl \
    php-mbstring \
    php-bcmath \
    php-snmp \
    php-pear \
    php-xml

RUN groupadd -g 1000 netmaster
RUN useradd -u 1000 -ms /bin/bash -g netmaster netmaster
RUN chmod 0644 /etc/cron.d/web-cron
RUN crontab /etc/cron.d/web-cron

USER netmaster
COPY . /var/www
COPY --chown=netmaster:netmaster . /var/www

RUN pip install -r requirements.txt
RUN npm install
RUN setup.py

EXPOSE 80 8080
CMD ["cron", "-f"]
```

Рисунок 3.17 – Код конфігурації розгортання контейнеру Docker

3.8 Висновки

У третьому розділі обґрунтовано вибір мови програмування та технологій для розробки програмного продукту. Проаналізувавши потреби програмного продукту обрано мови програмування Python та PHP. В якості системи управління базою даних обрано MariaDB та Redis. Фронтенд-частину графічного веб-інтерфейсу програмного модулю реалізовано на базі фреймворку Bootstrap 4, бібліотек jQuery та Tabutator, а також бібліотеки шрифтів FontAwesome. Також створено можливість автоматизованого розгортання програмного модулю для аудиту захищеності локальних комп’ютерних підмереж у середовищі Docker.

4 ТЕСТУВАННЯ ДОДАТКУ

4.1 Тестування програмного модулю

Робота програмного модулю для аудиту захищеності локальних комп'ютерних підмереж передбачає взаємодію із мережевим обладнанням, зокрема по протоколу SNMP. Повноцінне тестування системи не є можливим без побудови віртуального мережевого стенду. Перед тестуванням необхідно визначити особливості її структури та її можливих змін[29].

Вирішено зробити стандартний набір мережевого обладнання для навчання: 3 маршрутизатори, 3 комутатори та робочі станції (рис. 4.1). Для того щоб дана мережева конфігурація була сумісна із програмним забезпеченням, на ній необхідно налаштувати сервіс SNMP.

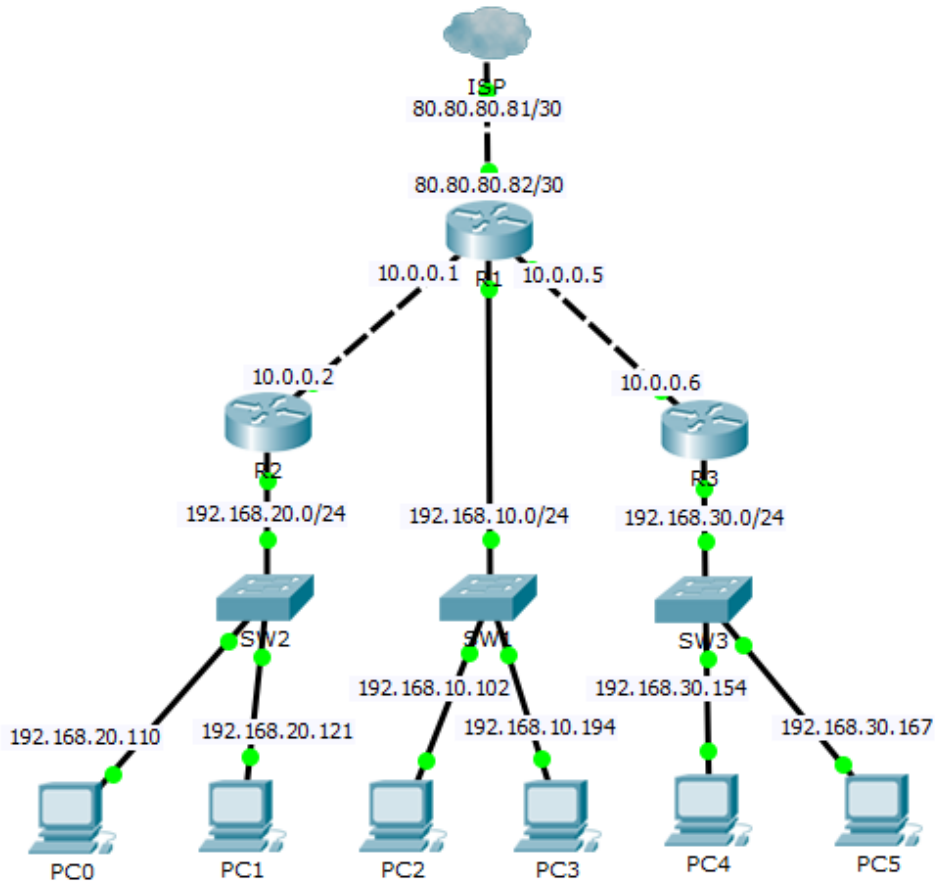


Рисунок 4.1 – Структура мережі для тестування

Існує чимало програмного забезпечення для моделювання комп'ютерних мереж. Найбільш відомим з них є Cisco Packet Tracer, однак його можливостей не буде достатньо для цілей тестування програмного модулю для аудиту захищеності локальних комп'ютерних підмереж, оскільки на момент написання роботи він не підтримує інтеграції із середовищем контейнеризації Docker.

Іншим програмним забезпеченням для моделювання та симуляції комп'ютерних мереж є пакет GNS3. Graphical Network Simulator-3 (скорочено GNS3) — програмний емулятор мережі, що дозволяє комбінувати віртуальні та реальні пристрої, що використовуються для моделювання складних мереж. Він використовує програмне забезпечення емуляції Dynamips для імітації Cisco IOS. GNS3 використовується багатьма великими компаніями, включаючи Exxon, Walmart, AT&T і NASA, а також популярний для підготовки мережевих професійних сертифікаційних іспитів [30]. Його суттєвою перевагою, яка дозволяє використати його для тестування є його підтримка інтеграції із середовищем контейнеризації Docker.

Для того, щоб підключити контейнер до GNS3, необхідно імпортувати раніше підготовлений його образ у середовище контейнерів GNS3 за допомогою Docker Compose (рис. 4.2)

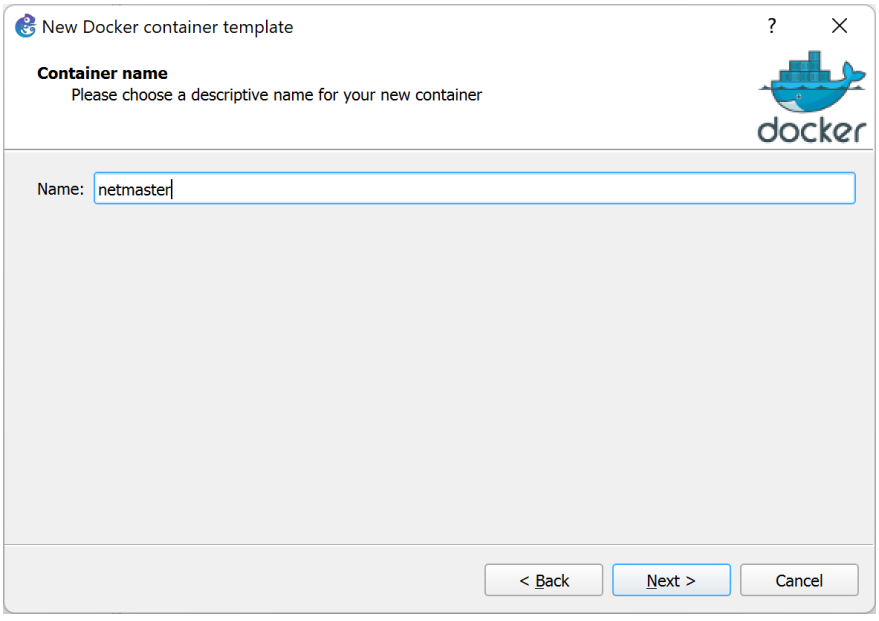


Рисунок 4.2 – Імпорт контейнеру у середовище GNS3

В ході імпорту необхідно налаштувати мережевий інтерфейс для його з’єднання із віртуальною мережею (рис. 4.3).

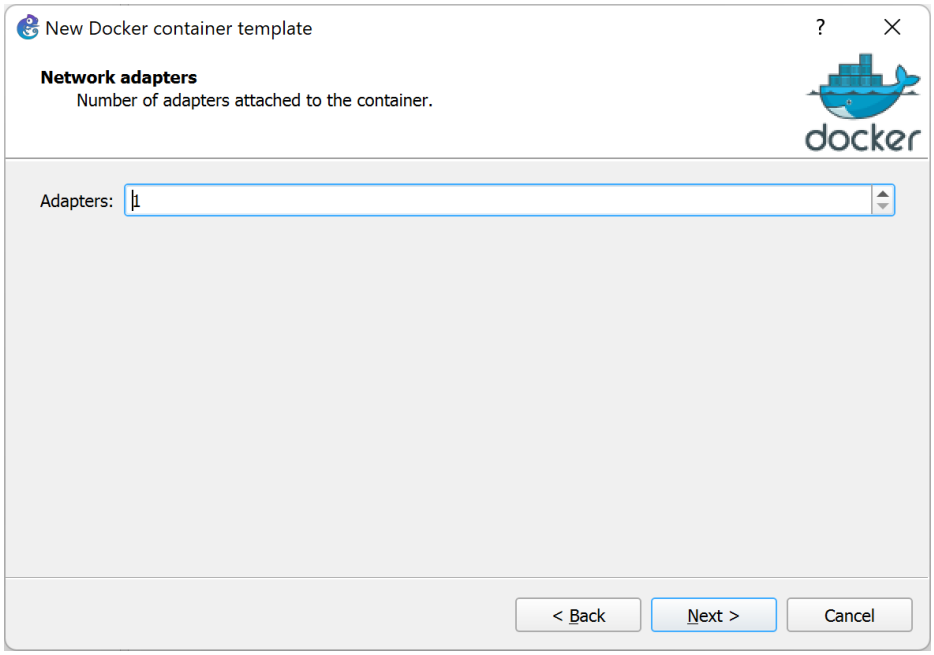


Рисунок 4.3 – Створення мережевих інтерфейсів

Можливі варіанти, коли контейнер не вдасться імпортувати вручну. Для цього необхідно використати засіб Docker Compose для виконання експорту образу (рис. 4.4). Після образ можна буде імпортувати у графічному інтерфейсі GNS.

```
PS D:\> cd D:\Projects\Docker\netmaster
PS D:\Projects\Docker\netmaster>
PS D:\Projects\Docker\netmaster> Set-Variable -Name DOCKER_HOST -Value "ssh://gns3@192.168.56.101"
PS D:\Projects\Docker\netmaster> docker-compose up -d
```

Рисунок 4.4 – Сторінка реєстрації з даними

Щоб отримати доступ до консолі контейнеру у середовищі GNS3 необхідно налаштувати параметри дистанційного контексту у оболонки Docker (рис. 4.5). IP-адресу, логін та пароль необхідно вказати від віртуальної машини GNS3, які можна переглянути, підключившись до її екрану.

```
PS D:\Projects\Docker\netmaster> docker context ls
NAME                TYPE                DESCRIPTION                               DOCKER ENDPOINT
      KUBERNETES ENDPOINT   ORCHESTRATOR
default *           moby                Current DOCKER_HOST based configuration   npipe:////./pipe/docker_engine
                                        swarm
desktop-linux       moby                                                          npipe:////./pipe/dockerDesktopLinuxEng
ine
PS D:\Projects\Docker\netmaster> docker context create remote --docker "host=ssh://gns3@192.168.56.101"
remote
Successfully created context "remote"
PS D:\Projects\Docker\netmaster> docker context ls
NAME                TYPE                DESCRIPTION                               DOCKER ENDPOINT
      KUBERNETES ENDPOINT   ORCHESTRATOR
default *           moby                Current DOCKER_HOST based configuration   npipe:////./pipe/docker_engine
                                        swarm
desktop-linux       moby                                                          npipe:////./pipe/dockerDesktopLinuxEng
ine
remote              moby                                                          ssh://gns3@192.168.56.101

PS D:\Projects\Docker\netmaster> docker --context remote ps
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ECDSA key fingerprint is SHA256:zSU0rsBYjQBph70FSCI6EttKxPn8K7UxP5IxEZEkaPc.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
gns3@192.168.56.101's password:
```

Рисунок 4.5 – Підключення до віддаленого середовища контейнерів GNS3

Якщо використовується зовнішній мережевий інтерфейс або переадресація портів, тоді стає можливим доступ до віртуальної машини через браузер (рис. 4.6). Логін та пароль адміністратора береться із змінних ENV або генерується при створенні нового контейнеру Docker.

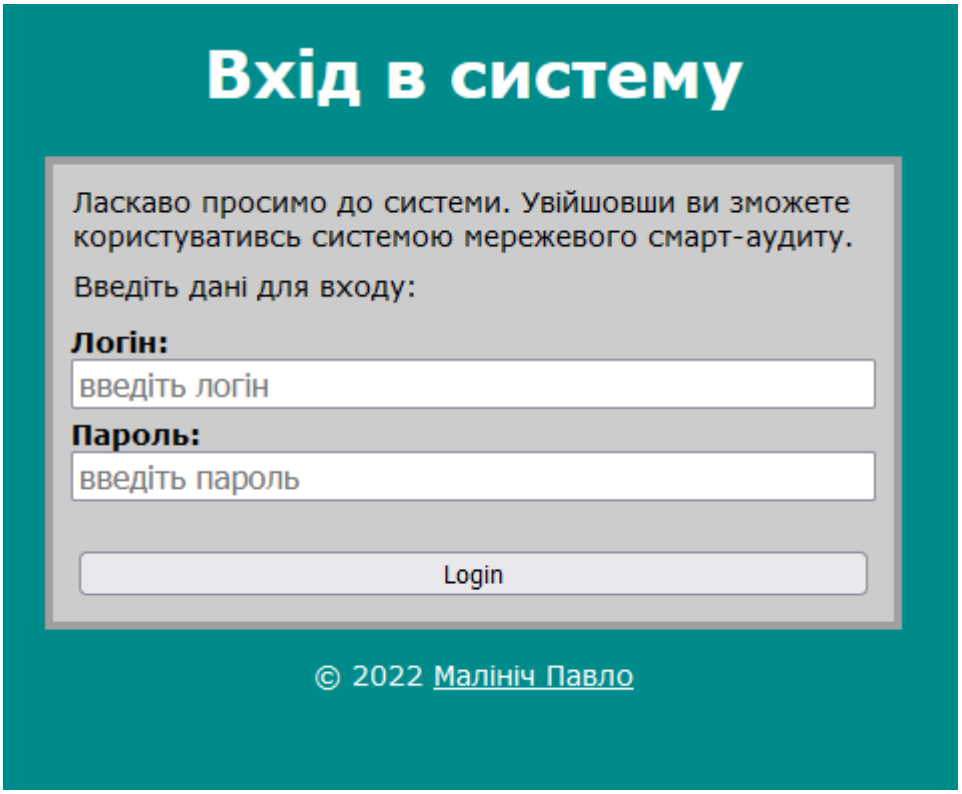


Рисунок 4.6 – Сторінка входу користувачів

Якщо пароль за вмовчанням не було вказано чи скопійовано згенерований – тоді його можна збити вручну за допомогою команди netmaster-cli, запустивши через оболонку Docker (рис. 4.7).

```
PS D:\Projects\Docker\netmaster> docker exec netmaster netmaster-cli webuser delete admin
User admin successfully deleted
PS D:\Projects\Docker\netmaster> docker exec netmaster netmaster-cli webuser create admin
Enter user's password:
Enter user's password again:

User admin successfully created
PS D:\Projects\Docker\netmaster>
```

Рисунок 4.7 – Збиття паролю користувача за вмовчанням

Після успішного входу відобразиться головне меню (рис. 4.8). Меню містить 5 пунктів: загальне, хости, підмережі, карта мережі та події.

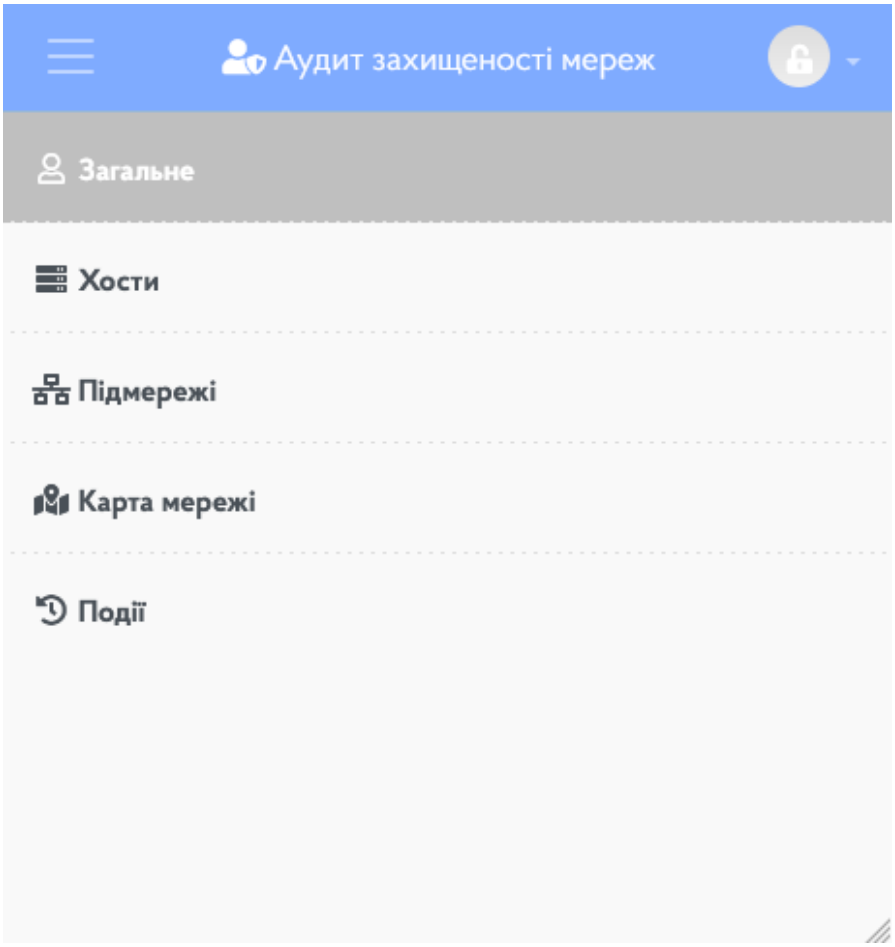


Рисунок 4.8 – Головна сторінка із меню

Процес додавання хосту здійснюється на сторінці "Хости" –> "Додати хост". Для додавання необхідно ввести IP-адресу, тип пристрою (рис 4.9), а також ввести параметри SNMP. При успішній синхронізації зі всіма хостами, скриптовий інструмент зможе побудувати карту (рис. 4.10).

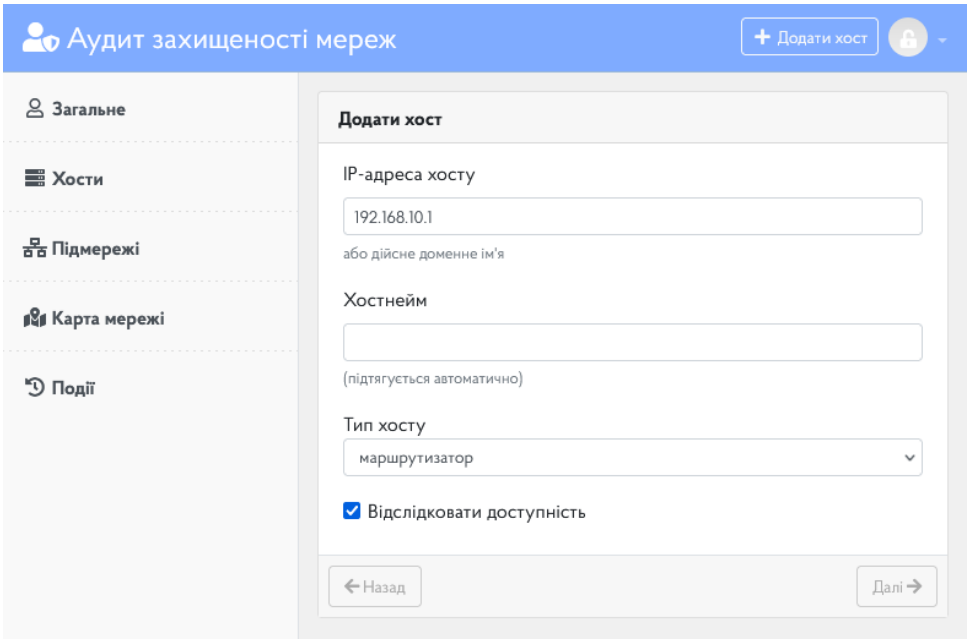


Рисунок 4.9 – Сторінка додавання нових пристроїв

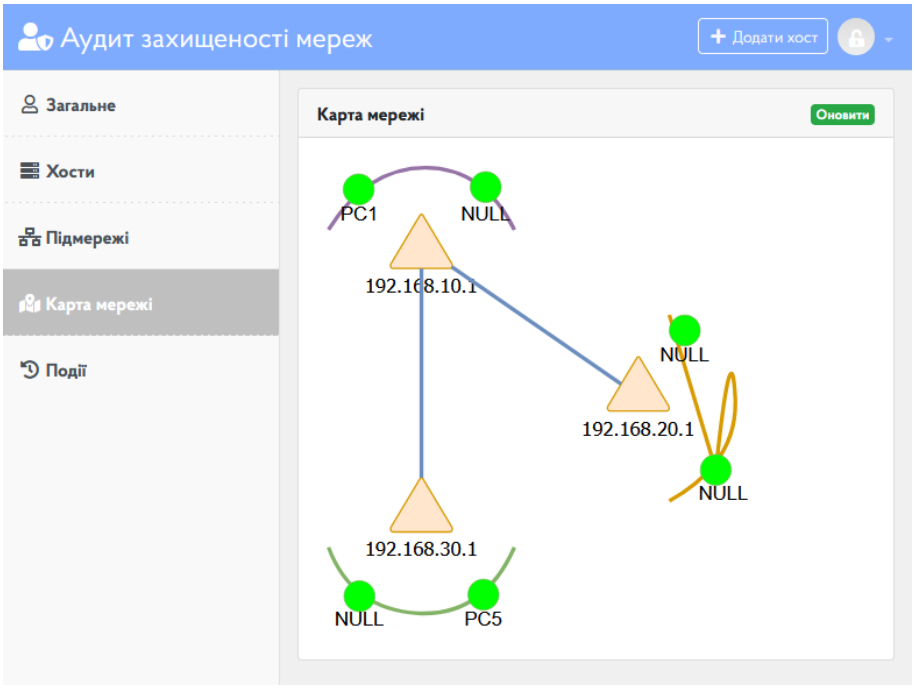


Рисунок 4.10 – Сторінка додавання нових пристроїв

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп'ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	1 віртуальне ядро CPU
Об'єм оперативної пам'яті	1 ГБ
Місце на жорсткому диску	8 ГБ
Операційна система	Ubuntu 18.04

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	4 віртуальних ядра CPU
Об'єм оперативної пам'яті	2 ГБ або більше
Розмір жорсткого диску	20 ГБ
Операційна система	Ubuntu 20.04

Рекомендується розгортання програмного модулю для аудиту захищеності локальних комп'ютерних підмереж у контейнерному середовищі Docker. Використання технології розгортання контейнерів Docker підтримується "з коробки" та дозволяє досить швидко розгортати всі компоненти програмного модулю.

Після встановлення програмного модулю для подальшої конфігурації необхідно підключитись до веб-інтерфейсу за допомогою браузеру. Якщо адміністративний пароль не задавався і не був згенерований за участі користувача – тоді його необхідно перевстановити за допомогою команди netmaster-cli.

Після входу у веб-панель програмного модулю для аудиту захищеності локальних комп'ютерних підмереж необхідно додати хости із SNMP-доступом. При додаванні хосту необхідно обрати версію протоколу SNMP та вказати параметри доступу. Для маршрутизаторів є можливість обрати які мережі мають підключення до мережі Інтернет. Після успішного збору даних про маршрути

можна буде імпортувати їх у розділі підмереж і таким чином почати їх прослідковування.

При імпорті мереж необхідно зазначити параметри їх сканування, зокрема частоту, порти, а також клас безпеки мережі. На базі цих параметрів генерується карта мережі.

4.3 Висновки

У четвертому розділі проведено тестування модулю для аудиту захищеності локальних комп’ютерних підмереж. Перевірено можливість розгортання у контейнерному середовищі Docker у зв’язці з симулятором мереж GNS3. Також розроблено інструкцію користувача по встановленню та початковому налаштуванню програмного модулю.

ВИСНОВКИ

У бакалаврській дипломній роботі розроблено програмний модуль для аудиту захищеності локальних комп’ютерних підмереж. Для розробки використано середовище програмної розробки Visual Studio Code.

Під час виконання бакалаврської дипломної роботи проаналізовано теперішній стан проблеми. Розглянуто програмні рішення, які є альтернативами. Проаналізувавши їх переваги та недоліки, виявлено, що такі опції як сканування підозрілих хостів, розподіл мереж за класами захищеності та підтримка низькобюджетного обладнання потребували подальшого розвитку. Визначено яким чином можливо створити більш досконале програмне рішення для моніторингу захищеності локальних підмереж. Встановлено основні задачі бакалаврської дипломної роботи.

Під час аналізу технологій розробки обрано мови програмування Python та PHP, а також фронтенд-бібліотек Bootstrap, jQuery та FontAwesome. Також розглянуто переваги використання реляційних та нереляційних баз даних, таких як MariaDB та Redis для збереження даних програмного модулю.

У бакалаврській дипломній роботі зроблено наступне:

– проведено аналіз та постановку задачі;
– розроблено архітектуру та алгоритми програмного модулю;
– розроблено метод багаторівневого сканування мережі;
– розроблено метод побудови карти мережі;
– розроблено API-інтерфейс програмного модулю;
– розроблено інструмент збору даних з обладнання;
– розроблено інструмент сканування та тестування підмереж;
– розроблено інструмент контролю за авторизацією;
– розроблено графічний веб-інтерфейс для програмного модулю;
– проведено тестування інструменту збору даних з обладнання та інших інструментів з використанням справжнього обладнання.

Також розроблено схеми загального алгоритму роботи програмного модулю та скриптових інструментів. Розроблено загальну архітектуру програмного модулю, структуру бази даних та принцип збереження даних хостів із плином часу, метод опитування внутрішнього стану керуємого обладнання (поллінг) та метод зовнішнього збору даних про доступні хости та мережеві сервіси (маппинг).

Тестування програмного модулю показало дієвість запропонованих концепцій даного програмного модулю та відповідність поставленому технічному завданню. Перевірено можливість розгортання у контейнерному середовищі Docker у зв’язці з симулятором мереж GNS3. Також розроблено інструкцію користувача по встановленню та початковому налаштуванню програмного модулю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Малініч П. П. Негативні безпекові чинники у локальних Ethernet-мережах та абонентських мереж останньої милі [Електронний ресурс] / П. П. Малініч, О. О. Коваленко, І. П. Малініч // Матеріали LI науково-технічної конференції підрозділів ВНТУ, Вінниця, 31 квітня - 1 червня 2022 р. - Електрон. текст. дані. - 2017. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15614/13205.
2. Ott, J. L. Managed Security Services. Inf. Secur. J. A Glob. Perspect., [Електронний ресурс] – Режим доступу до ресурсу: https://www.tandfonline.com/doi/abs/10.1201/1086/43317.10.4.20010901/31769.1?tab=permissions&scroll=top.
3. Conteh N. Y. Cybersecurity: risks, vulnerabilities and countermeasures to prevent social engineering attacks / Conteh N. Y., International Journal of Advanced Computer Research, 2017. – 68-74 с.
4. Radack S. M., Computer Security Division. Security in open systems networks. / Radack S. M., Computer standards & interfaces, 1990. – 213-218 с.
5. Rezgui Y., Information security awareness in higher education: An exploratory study. / Rezgui Y., Computers & security, 2008. – 241-253 с.
6. Binduf A., Active directory and related aspects of security. / Binduf A., 21st Saudi Computer Society National Computer Conference, 2018. – 4474-4479 с.
7. Kaeo M. Designing network security. / Kaeo M., Cisco Press, 2004. – 32 с.
8. Landauer M., Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. / Landauer M., computers & security, 2018. – 94-116 с.
9. Xu T., Security of IoT systems: Design challenges and opportunities. / Xu T., International Conference on Computer-Aided Design IEEE, 2014. – 417-423 с.
10. Ciampa M. Security+ guide to network security fundamentals. / Ciampa M., Cengage Learning, 2012. – 186 с.

11. Furnell S. The cybersecurity workforce and skills. / Furnell S., Computers & Security, 2021. – 208 с.
12. Banasiński C. Cybersecurity of consumer products against the background of the EU model of cyberspace protection. / Banasiński C., Journal of Cybersecurity, 2021. – 211 с.
13. Hale B. Why every it practitioner should care about network change and configuration management. / Hale B., SolarWinds, 2012.
14. Cantelmi R. Reviewing qualitative research approaches in the context of critical infrastructure resilience. / Cantelmi R., Environment Systems and Decisions, 2021. – 341-376 с.
15. Kavanagh K. M., Magic quadrant for security information and event management. / Kavanagh K. M., Gartner Group Research Note, 2015. – 31-37 с.
16. Larman C. Iterative and Incremental Development: A Brief History. / Larman C., Computer, 2003. – 47–56 с.
17. Peer I. Startup Mastery / Peer I., Tel Aviv-Yafo, 2020. – 220 с.
18. Münch J. Creating minimum viable products in industry-academia collaborations. / Münch J., International Conference on Lean Enterprise Software and Systems, 2013. – 137-151 с.
19. Коваленко О.О. Сучасні інформаційні системи – інвестиції в розвиток підприємства / Коваленко О.О., Інвестиції: практика та досвід, 2009. – 10–13 с.
20. Thorsten Lorenz GitHub – thlorenz/hyperwatch: Streams server side log messages to the browser and renders them inside your page. [Електронний ресурс] – Режим доступу: https://github.com/thlorenz/hyperwatch.
21. OSI model and TCP/IP network models – a must have concept to understand, before you move deeper in your networking adventures. [Електронний ресурс] – Режим доступу: https://www.ad-net.com.tw/osi-model-tcp-ip-network-models-must-concept-understand-move-deeper-networking-adventures/.

22. Edwards J. Networking self-teaching guide: OSI, TCP/IP, LANs, MANs, WANs, implementation, management, and maintenance. / Edwards J., John Wiley & Sons. 2015. – 157-165 с.
23. Piscitello D.M. Open systems networking: TCP/IP and OSI / Piscitello D.M., Addison-Wesley, 1993. – 349-351 с.
24. Chikohora E. A Study on the Impact of Network Vulnerability Scanners on Network Security. / Chikohora E., 3rd International Multidisciplinary Information Technology and Engineering Conference, 2021. – 1-4 с.
25. Butterfly Thoughts – PHP vs Python: Features & Performance Comparison in 2022. [Електронний ресурс] – Режим доступу: https://geekflare.com/php-vs-python-comparison.
26. Visual Studio Code. GitHub – microsoft/vscode: Open Source ("Code - OSS"). [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/microsoft/vscode.
27. SNMP design. [Електронний ресурс] – Режим доступу до ресурсу: https://pysnmp.readthedocs.io/en/latest/docs/snmp-design.html#terminology-and-entities.
28. Nmap Reference Guide – Description. [Електронний ресурс] – Режим доступу до ресурсу: https://nmap.org/book/man.html#man-description.
29. Чернишов К. А. Методи збору даних досвіду взаємодії користувача для випробувального етапу розробки через тестування [Текст] / К. А. Чернишов, І. П. Малініч, П. П. Малініч // Міжнародна наукова інтернет-конференція "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення, м. Тернопіль, 5 лютого 2019 р. : збірник тез доповідей. – Тернопіль, 2019. – Вип. 35. – 43 с.
30. Getting Started with GNS3 – Description. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.gns3.com/docs.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"_25_" _березня_2022 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка програмного модулю для аудиту захищеності локальних комп'ютерних підмереж»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

_____ к.т.н., доц. О.О. Коваленко

"____"_____2022 р.

Виконав:

_____студент гр. 1ПІ-18б П.П. Малініч

"____"_____2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного модулю для аудиту захищеності локальних комп'ютерних підмереж».

Галузь застосування – комп'ютерні мережі.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 12 від « 07 » лютого 2022 р.

3. Мета та призначення розробки.

Метою бакалаврської дипломної роботи є удосконалення процесу аудиту захищеності локальних комп'ютерних підмереж, який дозволяє покращити процес усунення неполадок та загроз.

Призначення роботи – спрощення процесу аудиту захищеності локальних комп'ютерних підмереж.

4. Вихідні дані для проведення НДР

1. Малініч П. П. Негативні безпекові чинники у локальних Ethernet-мережах та абонентських мереж останньої милі [Електронний ресурс] / П. П. Малініч, О. О. Коваленко, І. П. Малініч // Матеріали LI науково-технічної конференції підрозділів ВНТУ, Вінниця, 31 квітня - 1 червня 2022 р. - Електрон. текст. дані. - 2017. - Режим доступу : https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15614/13205.
2. Ott, J. L. Managed Security Services. Inf. Secur. J. A Glob. Perspect., 2001. – 1-3 с.
3. Conteh, N. Y., & Schmick, P. J. Cybersecurity: risks, vulnerabilities and countermeasures to prevent social engineering attacks. International Journal of Advanced Computer Research, 2017. – 68-74 с.

4. Radack S. M., & Staff, Computer Security Division. Security in open systems networks. Computer standards & interfaces, 1990. – 213-218 с.

5. Rezgui, Y., & Marks, A. Information security awareness in higher education: An exploratory study. Computers & security, 2008. – 241-253 с.

6. Binduf, A., Alamoudi, H. O., Balahmar, H., Alshamrani, S., Al-Omar, H., & Nagy, N. Active directory and related aspects of security. In: 2018 21st Saudi Computer Society National Computer Conference (NCC), 2018. – 4474-4479 с.

7. Kaeo, M. Designing network security. Cisco Press, 2004.

8. Landauer, M., Wurzenberger, M., Skopik, F., Settanni, G., & Filzmoser, P. Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. computers & security, 2018. – 94-116 с.

9. Xu, T., Wendt, J. B., & Potkonjak, M. Security of IoT systems: Design challenges and opportunities. In: 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2014. 417-423 с.

10. Ciampa, M. Security+ guide to network security fundamentals. Cengage Learning, 2012.

11. Furnell, S. The cybersecurity workforce and skills. Computers & Security, 2021.

12. Banasiński, C., & Rojszczak, M. Cybersecurity of consumer products against the background of the EU model of cyberspace protection. Journal of Cybersecurity, 2021.

13. Hale, B. Why every it practitioner should care about network change and configuration management. 2012.

14. Cantelmi, R., Di Gravio, G., & Patriarca, R. Reviewing qualitative research approaches in the context of critical infrastructure resilience. Environment Systems and Decisions, 2021. – 341-376 с.

15. Kavanagh, K. M., Rochford, O., & Bussa, T. Magic quadrant for security information and event management. Gartner Group Research Note, 2015.

16. Larman, C. Iterative and Incremental Development: A Brief History. Computer. 2003. – 47–56 с.

5. Технічні вимоги

Вхідні дані – інформація про комп'ютерну мережу, параметри доступу; вихідні дані – програмний модуль із побудованою картою мережі.

6. Конструктивні вимоги.

Дизайн рішення має відповідати естетичним та ергономічним вимогам, повинен бути зручним в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б – Акт впровадження

УЗГОДЖЕНО	ЗАТВЕРДЖУЮ
Науковий керівник Центру електронних комунікацій «ІнтерЦЕК»	Завідувач кафедри ПЗ
к.т.н., доцент Розводюк М.П.	д.т.н., професор Романюк О.Н.
_____	_____
«____» _____2022 року	«____» _____2022 року

АКТ ВПРОВАДЖЕННЯ № 7/ 27.05.2022
результатів науково-дослідних робіт

Замовник Центр електронних комунікацій «ІнтерЦЕК»
(найменування організації)

Цим актом підтверджується, що результати роботи – «Розробка програмного модулю для аудиту захищеності локальних комп'ютерних підмереж»,
(найменування теми)

що виконав студент гр. 1ПІ – 18б Малініч П.П.
(виконавець)

на громадських засадах в якості волонтерської допомоги.
(строки виконання)

Та впроваджено у Вінницькому національному технічному університеті
(найменування організації, де здійснювалося впровадження)

1. Вид впроваджених результатів: експлуатація програмного забезпечення для аудиту захищеності локальних комп'ютерних підмереж *(експлуатація виробу, роботи, технології)*

2. Характеристика масштабу впровадження: одиничне
(унікальне, одиничне, партія, масове, серійне)

3. Форма впровадження: дослідний зразок

4. Новизна результатів науково-дослідної роботи модернізація старих розробок
(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)

5. Впроваджені: у процес експлуатації комп'ютерних мереж ВНТУ

6. Соціальний та науково-технічний ефект: удосконалення процесів захищеності локальних комп'ютерних підмереж
(охорона навколишнього середовища, поліпшення й оздоровлення умов праці, удосконалення структури керування, науково-технічних напрямків, спеціальне призначення)

Від виконавця:	Від Центру електронних комунікацій «ІнтерЦЕК»:
студент групи 1ПІ-18б	
_____ Малініч П.П.	_____ Розводюк М.П.
Керівник: к.т.н., доцент кафедри ПЗ	_____ Коваленко О.О.

Додаток В – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка програмного модулю для аудиту захищеності локальних комп'ютерних підмереж»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. каф. ПЗ Коваленко О. О.

Оригінальність	96,4%
Схожість	3,6%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Малініч Павло Павлович

Керівник роботи _____ Коваленко Олена Олексіївна

Додаток Г – Лістинг програми

Код структури бази даних

Файл netmaster.sql

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: botmaster
--
CREATE DATABASE IF NOT EXISTS botmaster DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci;
USE botmaster;

--
-- Table structure for table admin_session
--

CREATE TABLE admin_session (
  record_id int(10) UNSIGNED NOT NULL,
  admin_id int(10) NOT NULL,
  session_id varchar(48) CHARACTER SET ascii NOT NULL,
  useragent varchar(256) CHARACTER SET ascii DEFAULT NULL,
  login_time timestamp NOT NULL DEFAULT current_timestamp(),
  login_ip varchar(132) CHARACTER SET ascii DEFAULT NULL
) ENGINE=MEMORY DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table admin_users
--

CREATE TABLE admin_users (
  id int(10) UNSIGNED NOT NULL,
  login varchar(64) CHARACTER SET ascii NOT NULL,
  pwd_hash varchar(512) CHARACTER SET ascii NOT NULL,
  full_name varchar(128) CHARACTER SET utf8 DEFAULT NULL,
  email varchar(64) CHARACTER SET utf8 DEFAULT NULL,
  level int(10) NOT NULL DEFAULT 0,
  active tinyint(1) NOT NULL DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table discover_hosts
--
```

```
CREATE TABLE discover_hosts (
  host_id int(11) UNSIGNED NOT NULL,
  host_name varchar(128) CHARACTER SET utf8 DEFAULT NULL,
  host_ip varchar(132) CHARACTER SET ascii NOT NULL,
  mac_addr varchar(12) DEFAULT NULL,
  real_host_id int(10) UNSIGNED DEFAULT NULL,
  record_dt timestamp NOT NULL DEFAULT current_timestamp(),
  expires timestamp NULL DEFAULT NULL
) ENGINE=MEMORY DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table discover_ports
--

CREATE TABLE discover_ports (
  record_id int(10) UNSIGNED NOT NULL,
  record_dt timestamp NOT NULL DEFAULT current_timestamp(),
  expires timestamp NULL DEFAULT NULL,
  port_number int(10) UNSIGNED NOT NULL,
  port_proto int(10) UNSIGNED NOT NULL,
  host_id int(11) DEFAULT NULL,
  banner varchar(256) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table known_hosts
--

CREATE TABLE known_hosts (
  host_id int(10) UNSIGNED NOT NULL,
  host_name varchar(128) CHARACTER SET utf8 NOT NULL,
  host_ip varchar(132) CHARACTER SET ascii DEFAULT NULL,
  class varchar(32) CHARACTER SET ascii NOT NULL,
  mac_addr varchar(12) CHARACTER SET ascii DEFAULT NULL,
  net_id int(10) DEFAULT NULL,
  snmp json  CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table networks
--

CREATE TABLE networks (
  net_id int(10) UNSIGNED NOT NULL,
  net_ip varchar(132) NOT NULL,
  length int(3) NOT NULL,
  master_host int(11) DEFAULT NULL,
  net_class varchar(4) NOT NULL,
  scan_type int(4) DEFAULT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Table structure for table threat_events
--

CREATE TABLE threat_events (
  record_id int(16) UNSIGNED NOT NULL,
  record_dt timestamp NOT NULL DEFAULT current_timestamp(),
  class varchar(32) CHARACTER SET ascii DEFAULT NULL,
  level smallint(8) NOT NULL,
  application_id varchar(64) CHARACTER SET ascii NOT NULL,
  host_id int(10) DEFAULT NULL,
  local_ip varchar(132) CHARACTER SET ascii DEFAULT NULL,
  remote_ip varchar(132) CHARACTER SET ascii DEFAULT NULL,
  description text CHARACTER SET utf8 DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Indexes for table admin_session
--
ALTER TABLE admin_session
  ADD PRIMARY KEY (record_id) USING HASH;

--
-- Indexes for table admin_users
--
ALTER TABLE admin_users
  ADD PRIMARY KEY (id);

--
-- Indexes for table discover_hosts
--
ALTER TABLE discover_hosts
  ADD PRIMARY KEY (host_id);

--
-- Indexes for table discover_ports
--
ALTER TABLE discover_ports
  ADD PRIMARY KEY (record_id);

--
-- Indexes for table known_hosts
--
ALTER TABLE known_hosts
  ADD PRIMARY KEY (host_id);
```

```
--
-- Indexes for table networks
--
ALTER TABLE networks
  ADD PRIMARY KEY (net_id);

--
-- Indexes for table threat_events
--
ALTER TABLE threat_events
  ADD PRIMARY KEY (record_id);

-- --------------------------------------------------------

--
-- AUTO_INCREMENT for table admin_session
--
ALTER TABLE admin_session
  MODIFY record_id int(10) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table admin_users
--
ALTER TABLE admin_users
  MODIFY id int(10) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table discover_hosts
--
ALTER TABLE discover_hosts
  MODIFY host_id int(11) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table discover_ports
--
ALTER TABLE discover_ports
  MODIFY record_id int(10) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table known_hosts
--
ALTER TABLE known_hosts
  MODIFY host_id int(10) UNSIGNED NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table networks
--
ALTER TABLE networks
  MODIFY net_id int(10) UNSIGNED NOT NULL AUTO_INCREMENT;
```

```
--
-- AUTO_INCREMENT for table threat_events
--
ALTER TABLE threat_events
  MODIFY record_id int(16) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Код інструменту збору даних з обладнання

Файл device-poller.py

```
import json
import socket
import urllib.parse
import urllib.request
from pysnmp.hlapi import *

def update_host_data(snmp_host_ip, data):
    arp_out = []
    for pair in data.values(): arp_out.append(pair)
    json_data = json.dumps( { 'arp_table': arp_out } )
    api_url = config['api_base'] + "snmp/update.do"
    api_vars = urllib.parse.urlencode( {
        "api_key": config['api_key'],
        "toolkit_ip": socket.gethostbyname( socket.gethostname() ),
        "host_data": json_data
    } ).encode()
    req = urllib.request.Request(api_url, api_vars)
    resp = urllib.request.urlopen(req)
    print('[POLLER] Host ' + snmp_host_ip + ': collected ' +
str(len(arp_mib)) + ' ARP records')
    return resp.read(), resp.getheaders()

with open('app/config/api-server.json') as api_server:
    config = json.load(api_server)

api_vars = {
    "api_key": config['api_key'],
    "toolkit_ip": socket.gethostbyname( socket.gethostname() )
}

api_url = config['api_base'] + "snmp/list.do" + "?" +
urllib.parse.urlencode( api_vars )
api_request = urllib.request.urlopen(api_url)
request_body = api_request.read().decode("utf-8")
```

```
api_data = json.loads(request_body)
if not len(api_data) > 0:
    raise RuntimeError('Blank API output')
for snmp_host in api_data:
    arp_mib = {}
    if 'port' in snmp_host: snmp_host['port'] = int(snmp_host['port'])
    else: snmp_host['port'] = 161
    snmp_conn = UdpTransportTarget( (snmp_host['ip'], snmp_host['port']) )

    if 'version' in snmp_host: snmp_host['version'] =
int(snmp_host['version'])
    else: snmp_host['version'] = 2
    if snmp_host['version'] == 1 or snmp_host['version'] == 2:
        snmp_cred = CommunityData( snmp_host['community'],
mpModel=snmp_host['version']-1 )
    if snmp_host['version'] == 3:
        snmp_cred = UsmUserData(snmp_host['user'],
authKey=snmp_host['password'])

    mac_poller = bulkCmd(
        SnmpEngine(), snmp_cred, snmp_conn, ContextData(), 1, 25,
ObjectType(ObjectIdentity('IP-MIB', 'ipNetToMediaTable')),
lexicographicMode=False
    )

    for errorIndication, errorStatus, errorIndex, varBinds in mac_poller:
        if errorIndication:
            print(errorIndication)
            break
        elif errorStatus:
            print('%s at %s' % (errorStatus.prettyPrint(),
                                errorIndex and varBinds[int(errorIndex) -
1][0] or '?'))
            break
        else:
            for varBind in varBinds:
                mib_num = str(varBind[0]).split('.')
                for x in range(11): mib_num.pop(0)
                ip_num = '.'.join(mib_num)
                if (
                    varBind[1].__class__.__name__ == 'IpAddress' or
                    varBind[1].__class__.__name__ == 'PhysAddress'
                ) and (
                    ip_num not in arp_mib
                ):
                    arp_mib[ip_num] = {}

                if varBind[1].__class__.__name__ == 'IpAddress':
                    arp_mib[ip_num]['ip'] = varBind[1].prettyPrint()
                if varBind[1].__class__.__name__ == 'PhysAddress':
```

```
                    arp_mib[ip_num]['mac'] = varBind[1].prettyPrint()

    update_host_data(snmp_host['ip'], arp_mib)
```

Код Backend-частини веб-інтерфейсу

Файл webapp.php

```
<?php namespace InterCecApp;

class WebApp {
    private $active = false;
    private $locale;
    private $application;
    private $front_buffer;
    public $front_templates;
    public $front_container;
    public $front_title;
    public $front_pages;
    public $front_menu;
    public $front_var;
    public $front_css;
    public $front_js;
    public $menu;

    public function FrontInit($full_load = true) {
        $this->front_buffer = '';
        ob_start();
        $this->application->AppFrontInit();
    }

    public function FrontRender( $front_container = true ) {
        $this->front_container = $front_container;
        $this->front_buffer.= ob_get_clean();

        if( $this->front_var['title'] != null && $this-
>front_var['title_sub'] != null &&
            $this->front_var['title'] != '' && $this-
>front_var['title_sub'] != '') {
            $this->front_title = $this->front_var['title_sub'] .
            ' - ' .
            $this->front_var['title'];
        } else if( $this->front_var['title_sub'] != '' ) {
            $this->front_title = $this->front_var['title_sub'];
        } else {
            $this->front_title = $this->front_var['title'];
        }

        $template_path = $this->GetTemplateLoader();
        if($template_path != null) {
```

```
            include( $template_path );
        }
        else
            echo( $this->front_buffer );
    }

    public function FrontRenderBlob( $blob, $front_container = true ) {
        $this->front_container = $front_container;
        $this->FrontInit();
        echo($blob);
        $this->FrontRender();
    }

    public function FrontRenderFunction( $function, $front_container = true
) {
        $this->front_container = $front_container;
        $this->FrontInit();
        $function();
        $this->FrontRender();
    }

    public function FrontRenderScript( $script, $front_container = true ) {
        $this->front_container = $front_container;
        $this->FrontInit();
        $this->application->LocalExec( $script );
        $this->FrontRender();
    }

    public function FrontRenderPage( $page, $front_container = true ) {
        if( !isset($this->front_pages[$page]) )
            return null;
        $this->front_container = $front_container;
        $this->FrontInit();
        $this->application->LocalExec($this->front_pages[$page]);
        $this->FrontRender();
    }

    public function UseTemplate( $name ) {
        $this->front_var['active_template'] = $name;
    }

    public function UseErrorTemplate( $name ) {
        $this->front_var['error_template'] = $name;
    }

    public function PushTemplate( $name, $href ) {
        $this->front_templates[$name] = $href;
    }

    public function SetTitle($str) {
```

```
        $this->front_var['title'] = $str;
    }

    public function SetTitleSub($str) {
        $this->front_var['title_sub'] = $str;
    }

    protected function GetTemplateLoader() {
        if(
            $this->front_var['active_template'] == null or
            $this->front_var['active_template'] == ''
        ) return null;

        $template = $this->front_var['active_template'];
        if(!isset($this->front_templates[$template]))
            return null;

        $path = dirname( $this->front_templates[$template] ) .
'/template_html.php';
        return $path;
    }

    public function GetLabel($label_id) {
        return $this->application->locale->GetLabel($label_id);
    }

    public function PrintLabel($label_id) {
        echo $this->application->locale->GetLabel($label_id);
    }

    public function LoadFrontLib( $name ) {
        array_push($this->front_var['front_libs'], $name);
    }

    public function PushCSS( $name, $href ) {
        $this->front_css[$name] = $href;
    }

    public function PushJS( $name, $href ) {
        $this->front_js[$name] = $href;
    }

    protected function PrintCSS( $tab = 4 ) {
        $buffer = ''; $cnt = 0;

        foreach($this->front_css as $key => $href) {
            $cnt++;
            if(in_array($key, $this->front_var['front_libs'])) {
                for ($i = 1; $i <= $tab; $i++) $buffer.= ' ';
```

```
                $buffer.= '<link rel="stylesheet" href="' . $href . '">' .
"\r\n";
            }
        }

        if($cnt != 0) {
            echo("\r\n");
            echo($buffer);
        }
    }

    protected function PrintJS( $tab = 4 ) {
        $buffer = ''; $cnt = 0;

        foreach($this->front_js as $key => $src) {
            $cnt++;
            if(in_array($key, $this->front_var['front_libs'])) {
                for ($i = 1; $i <= $tab; $i++) $buffer.= ' ';
                $buffer.= '<script src="' . $src . '"></script>' . "\r\n";
            }
        }

        if($cnt != 0) {
            echo("\r\n");
            echo($buffer);
        }
    }

    protected function PrintBody() {
        echo($this->front_buffer);
    }

    function __construct(&$parent) {
        $this->application = &$parent;
        $this->locale = &$parent->locale;

        if(isset($this->application->front_js)) {
            $this->front_js = $this->application->front_js;
            unset($this->application->front_js);
        } else $this->front_js = array();

        if(isset($this->application->front_css)) {
            $this->front_css = $this->application->front_css;
            unset($this->application->front_css);
        } else $this->front_css = array();

        if(isset($this->application->front_var)) {
            $this->front_var = $this->application->front_var;
            unset($this->application->front_var);
        } else $this->front_var = array(
```

```
            'title' => '',
            'title_sub' => '',
            'active_template' => null,
            'error_template' => null,
            'front_libs' => array(),
        );

        if(isset($this->application->front_templates)) {
            $this->front_templates = $this->application->front_templates;
            unset($this->application->front_templates);
        } else $this->front_templates = array();

        if(isset($this->application->front_pages)) {
            $this->front_pages = $this->application->front_pages;
            unset($this->application->front_pages);
        } else $this->front_pages = array();

        if(isset($this->application->front_menu)) {
            $this->front_menu = &$this->application->front_menu;
            unset($this->application->front_menu);
        } else $this->front_menu = array();
        $this->menu = new Menu( $this->front_menu );
    }
}
```

Код Frontend-частини веб-інтерфейсу

Файл login.css

```
html,
body {
    margin: 0;
    padding: 0;
}

body {
    background-color: darkcyan;
    font-family: Verdana, Arial, Helvetica, sans-serif;
}

input {
    font-size: 16px;
}

.full-width {
    width: 100%;
}

.header-line h1 {
```

```
    color: #fff;
}

.error-message {
    color: #ff0000;
    font-weight: bold;
}

.login-form {
    max-width: 400px;
    background-color: #ccc;
    border: 4px solid #9f9f9f;
    text-align: left;
    font-size: 14px;
    padding: 10px;
}

.login-form p {
    margin-top: 0;
    margin-bottom: 8px;
}

.login-form label,
.login-form input[type=text],
.login-form input[type=password],
.login-form input[type=number] {
    margin-left: -4px;
}

.asp-credits {
    font-size: 14px;
    color: #fff;
}

.asp-credits a,
.asp-credits a:hover,
.asp-credits a:visited {
    color: #fff;
}
```

Файл app.css

```
html {
    font-family: sans-serif;
    line-height: 1.15;
    -ms-text-size-adjust: 100%;
    -webkit-text-size-adjust: 100%
}

body,
```

```
html {
    width: 100%;
    height: 100%;
}

body {
    margin: 0;
    padding: 0;
    font-family: "Proba Pro", sans-serif;
}

ul.dashed {
    list-style-type: none;
}

ul.dashed > li {
    text-indent: -5px;
}

ul.dashed > li:before {
    content: "-";
    text-indent: -5px;
    margin-right: 3px;
}

.page-wrapper {
    overflow-x: hidden;
}

.id-nav {
    border-bottom: 1px solid #9aa9d7;
    background-color: #7f81ff;
    position: absolute;
    min-height: 57px;
    z-index: 10;
    top: 0;
    right: 0;
    left: 0;
}

.id-nav h1 {
    margin: 0;
    padding: 0;
    font-size: 1.5em;
}

.id-nav h1 a {
    color: #fff !important;
    text-decoration: none !important;
}
```

```
.id-nav #btnReturnToCab {
    display: none;
}

.id-nav .navbar-nav {
    -ms-flex-direction: row;
    flex-direction: row;
}

.id-nav .id-acc-button {
    padding: 0;
    margin: 0;
}

.id-nav .id-acc-button::after {
    vertical-align: .1em;
}

.id-nav .navbar-nav .dropdown-menu {
    position: absolute;
    float: none;
}

.id-nav avatar-element {
    color: #fff;
    width: 32px;
    height: 32px;
    border-radius: 50%;
    display: inline-block;
    line-height: 34px !important;
    background: linear-gradient(#ddd, #eee);
    text-align: center;
    font-size: 15px;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    text-transform: uppercase;
    font-weight: 700;
}

.id-nav avatar-element[data-user-type=student] {
    background: linear-gradient(#008044, #00c86b);
}

.id-nav avatar-element[data-user-type=teacher] {
    background: linear-gradient(#7500aa, #9f00e6);
}
```

```
.id-nav-right .dropdown-menu {
    padding: .25rem 0;
}

.sidebar-toggler {
    padding: .25rem .25rem;
    font-size: 1.25rem;
    line-height: 1;
    background-color: transparent;
    border: 1px solid transparent;
    border-radius: .25rem;
}

.sidebar-wrapper {
    min-height: 100vh;
    padding-top: 57px;
    margin-left: -15rem;
    transition: margin 0.25s ease-out;
    border-right: 1px solid #dee2e6 !important;
    background-color: #f9f9f9;
}

.sidebar-wrapper .list-group {
    width: 15rem;
}

.sidebar-wrapper .list-group-item {
    font-weight: bold;
    border-bottom: 1px dashed #ddd;
    background-color: #f9f9f9;
}

.sidebar-wrapper .list-group-item:focus,
.sidebar-wrapper .list-group-item:hover {
    color: #495057;
    text-decoration: none;
    background-color: #e8e8e8;
    border-color: #ddd;
}

.dropdown-item:focus,
.dropdown-item:hover {
    color: #fff;
    background-color: #4875cc;
}

.list-group-item.active {
    color: #fff;
    border-color: #e6e6e6;
    background-color: #bfbfbf;
```

```
}

.list-group-item.active:focus,
.list-group-item.active:hover {
    color: #fff;
    border-color: #e6e6e6;
    background-color: #a8a8a8;
}

.page-content-wrapper {
    background-color: #f0f0f0;
    min-width: 100vw;
    padding-top: 72px;
    padding-bottom: 16px;
}

.sidebar-toggle .sidebar-wrapper {
    margin-left: 0;
}

.alert-dismissible .close {
    padding: .8rem 1rem;
    font-size: 18px;
}

.card-header {
    padding: .50rem 1.0rem;
    border-bottom: 1px solid rgba(0,0,0,.125);
}

.card-header .help-button {
    margin: 0;
    padding: 0;
    line-height: 0.8em;
    position: relative;
    right: -6px;
}

.card-body {
    padding-top: 0.8em;
}

.card-search {
    padding: 6px;
}

.card-row {
    display: -ms-flexbox;
    display: flex;
    -ms-flex-wrap: wrap;
```

```
    flex-wrap: wrap;
    margin-right: -5px;
    margin-left: -5px;
}

.card-row .col-md-4 {
    padding-right: 5px;
    padding-left: 5px;
}

.card-row .card-body {
    padding-top: 1em;
    text-align: justify;
    text-align: justify;
    line-height: 1.2em;
    font-size: 14px;
}

.card-buttons {
    border-top: 1px solid #dee2e6;
    background-color: #f7f7f7;
    padding: .50rem;
}

.table-account tr:hover > td {
    background: #eff4f0 !important;
}

.thead-account th {
    padding: .20rem .75rem;
}

.table-buttons-row {
    background-color: rgba(0,0,0,.05);
}

.text-purple {
    color: #9b619b !important;
}

a.text-purple:focus, a.text-purple:hover {
    color: #573557 !important;
}

.btn-outline-oauth {
    color: #aaa;
    font-size: 0.85em;
    background-color: transparent;
    background-image: none;
    border-color: #ccc;
```

```
}

.btn-outline-oauth:hover,
.btn-outline-oauth:active {
    color: #fff;
    background-color: #ddd;
    border-color: #ddd;
}

.btn-outline-oauth.focus,
.btn-outline-oauth:focus {
    outline: 0;
    box-shadow: 0 0 0 .2rem rgba(181, 181, 181, 0.25);
}

.btn-outline-oauth .icon-oauth-google,
.btn-outline-oauth .icon-oauth-microsoft,
.btn-outline-oauth .icon-oauth-corporate,
.btn-outline-oauth .icon-oauth-user {
    width: 16px;
    height: 16px;
    content: '\a0';
    display: inline-block;
    margin-bottom: -3px;
    background-size: auto 100%;
}

.input-number-plain::-webkit-inner-spin-button,
.input-number-plain::-webkit-outer-spin-button {
  -webkit-appearance: none;
  margin: 0;
}

@media (max-width: 444px) {
    .id-nav h1 {
        display: none;
    }
}

@media (min-width: 366px) and (max-width: 444px) {
    .id-nav #btnReturnToCab {
        display: inline-block;
    }
}

@media (max-width: 544px) {
    .id-nav h1 {
        font-size: 1.1em;
    }
```

```
    .id-nav-right .dropdown-item {
        font-size: 20px;
    }
}

@media (min-width: 768px) {
    .id-nav #btnReturnToCab {
        display: inline-block;
    }

    .id-nav-right .dropdown-item {
        font-size: 14px;
    }

    .sidebar-wrapper {
        margin-left: 0;
    }

    .page-content-wrapper {
        min-width: 0;
        width: 100%;
    }

    .sidebar-toggler {
        display: none;
    }

    .text-md-left {
        text-align: left !important;
    }

    .text-md-center {
        text-align: center !important;
    }

    .text-md-right {
        text-align: right !important;
    }
}

@media (min-width: 960px) {
    .text-responsive-lg {
        display: inline-block !important;
    }
}

@media (min-width: 1368px) {
    .d-mega-block {
        display: block !important;
    }
```

```
}
```

Файл log.js

```
var table; var search_tag; var table_firstload = true;

function secondary_formatter(cell){
    var avatar_id = cell.getValue();
    $( cell.getElement() ).css({"color":"#888"});
    return avatar_id;
}

function action_formatter(cell){
    var action = cell.getValue();
    switch( action ) {
        case 'fail': action = 'Помилка'; break;
        case 'auth': action = 'Авториз.'; break;
        case 'login': action = 'Вхід'; break;
        case 'logout': action = 'Вихід'; break;
        case 'rogue_login': action = 'Підбір'; break;
        case 'session_setup': action = 'Автентиф.'; break;
    }
    switch( cell.getData().status ){
        case 'warning': return '<i class="fas fa-exclamation-triangle text-
warning"></i> <span class="text-warning">' + action + '</span>';
        default: return '<i class="fas fa-info-circle text-info"></i> <span
class="text-info">' + action + '</span>';
    }
}

function user_formatter(cell){
    var user_login = cell.getValue();
    if( !cell.getData().user_id && user_login == '' ) return '-';
    if( !cell.getData().user_id && user_login == '#NULL' ) return '-';
    if( !cell.getData().user_id && user_login == '#BAD_AVID' ) return
'(підбір)';
    if( !cell.getData().user_id || !cell.getData().tou ||
!cell.getData().u_name ) {
        $( cell.getElement() ).css({"color":"#888"});
        return user_login;
    };
    var base_uri = $('#log-table').data('base-uri');
    var user_slug = 'user_name';
    var user_class = 'text-success';
    return '<a class="' + user_class + '" href="' + base_uri + user_slug +
'/' + cell.getData().user_id + '/' + '">'
           + cell.getData().u_name
           + '</a>';
}
```

```
function update_log_table(json_params) {
    json_url = $('#log-table').data('json');
    start_page = $('#log-table').data('start-page');
    var table = new Tabulator("#log-table", {
        layout:"fitColumns",
        pagination:"remote",
        paginationSize:6,
        ajaxURL:json_url,
        ajaxParams:json_params,
        paginationSize:10,
        paginationSizeSelector:[10, 25, 50, 75, 100],
        paginationInitialPage:start_page,
        columns:[
            {title:"Подія", field:"action", headerSort:false,
formatter:action_formatter},
            {title:"Користувач", field:"login", widthGrow:2,
headerSort:false, formatter:user_formatter},
            {title:"Додаток", field:"app", headerSort:false,
formatter:secondary_formatter},
            {title:"IP-Адреса", field:"remote_ip", headerSort:false,
formatter:secondary_formatter},
            {title:"Час", field:"dt", headerSort:false,
formatter:secondary_formatter},
        ],
        dataLoaded:function(data){
            if(table_firstload) table.setPage(start_page); table_firstload
= false;
        },
        pageLoaded:function(page_no){
            if(!search_tag || search_tag == '') left_url =
location.protocol + '//' + location.host + location.pathname + '?page=' +
page_no;
            else left_url = location.protocol + '//' + location.host +
location.pathname + '?search=' + encodeURIComponent( search_tag ) +
'&page=' + page_no;
            history.replaceState(null, "Page " + page_no, left_url);
        },
    });
}

$(document).ready(function() {
    update_log_table();
});
```

Додаток Д

(*обов'язковий*)

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ ДЛЯ АУДИТУ ЗАХИЩЕНОСТІ ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ ПІДМЕРЕЖ

Додаток Д – Ілюстративна частина

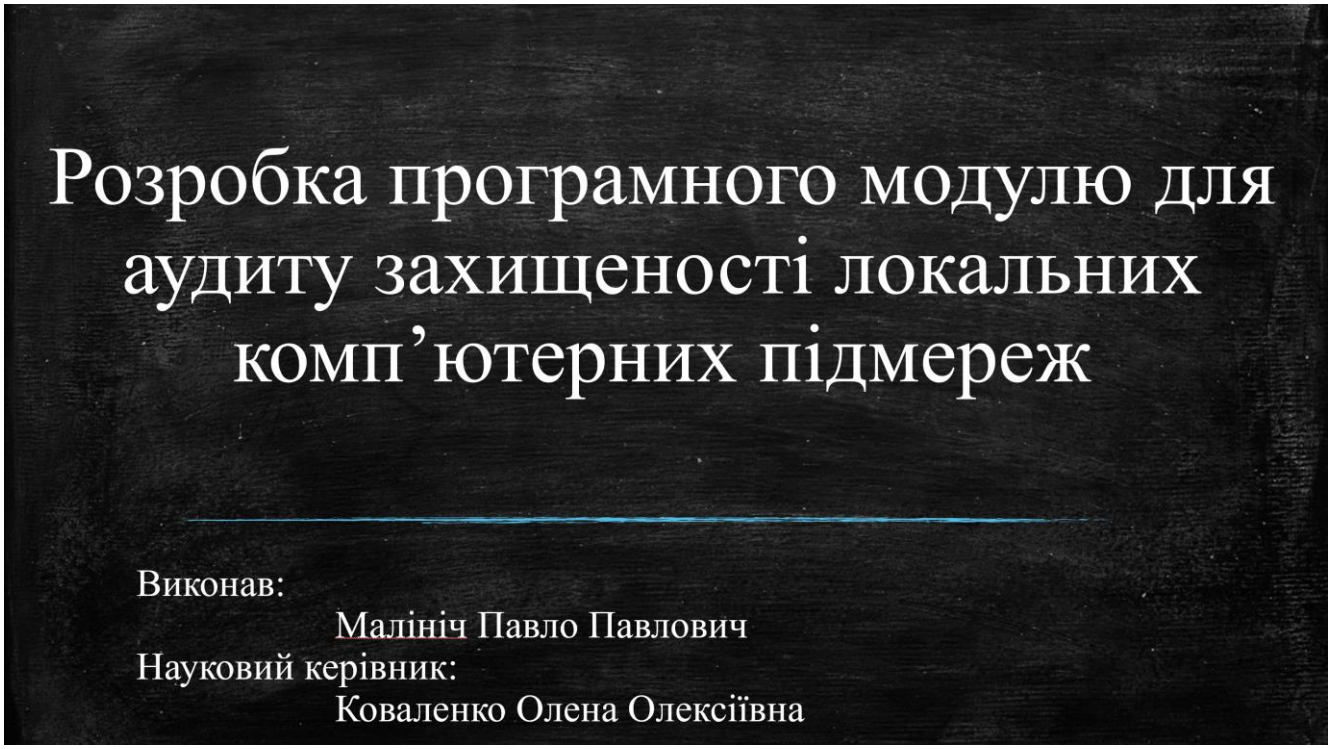


Рисунок Д.1 – Назва роботи

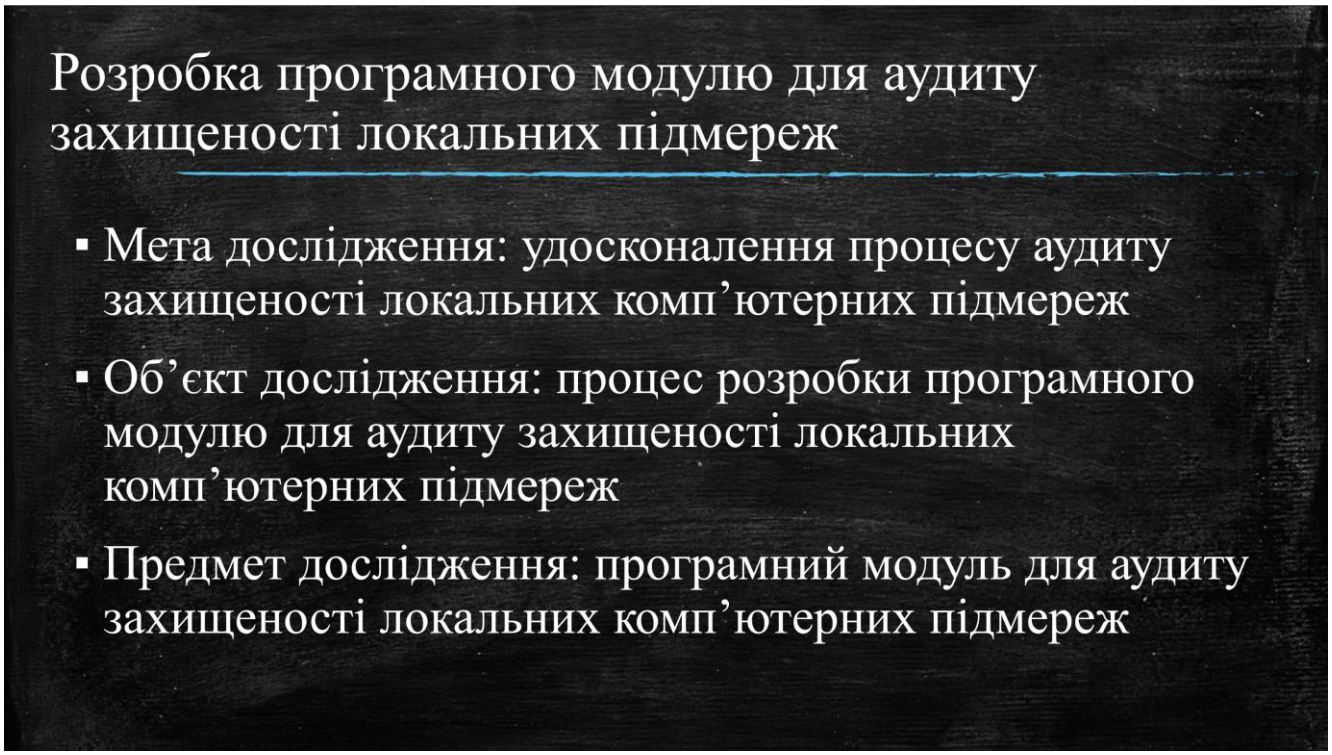


Рисунок Д.2 – Мета, об’єкт і предмет дослідження

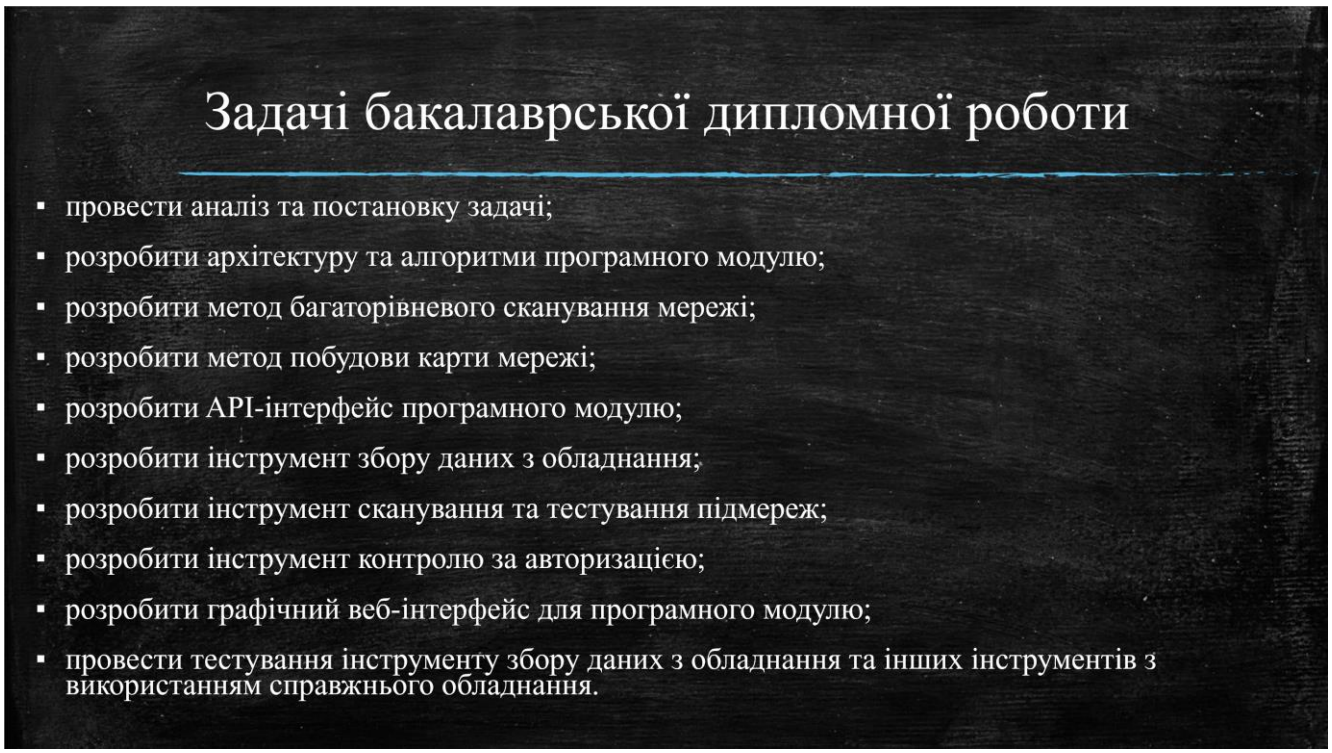


Рисунок Д.3 – Задачі

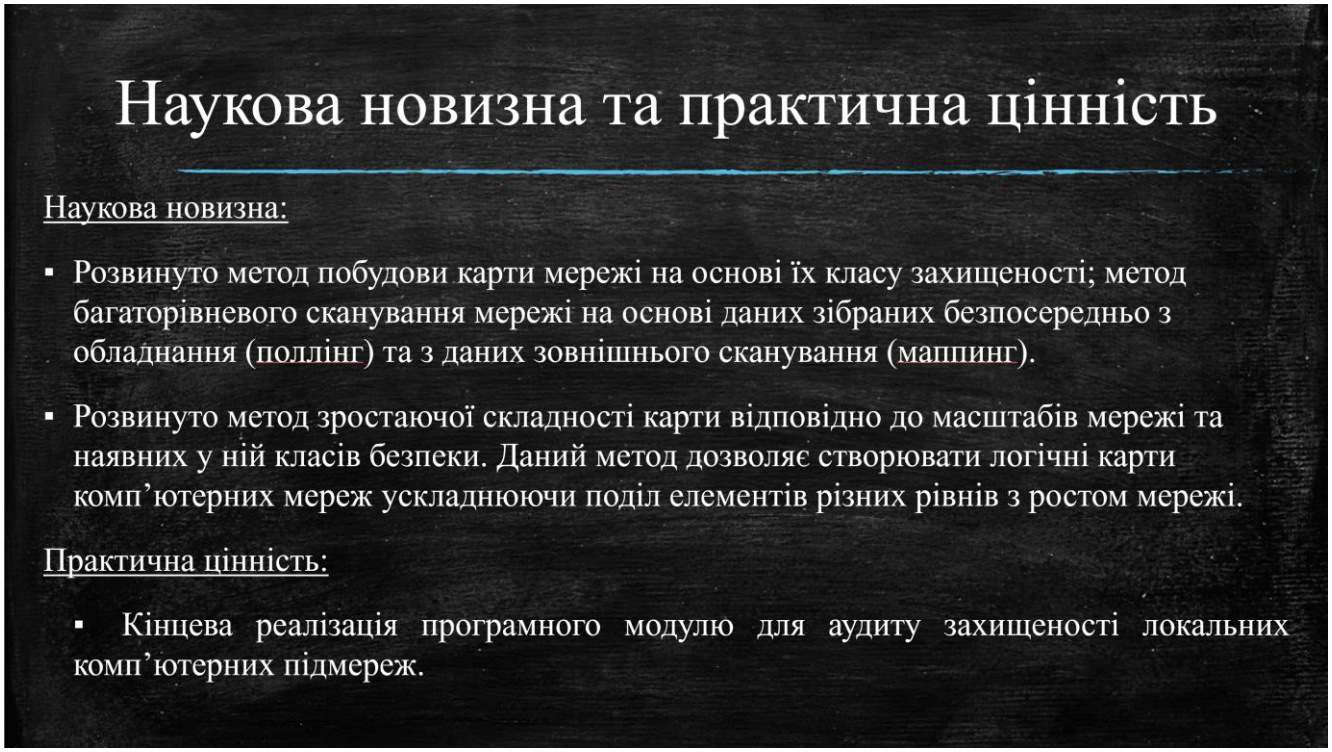


Рисунок Д.4 – Наукова новизна та практична цінність

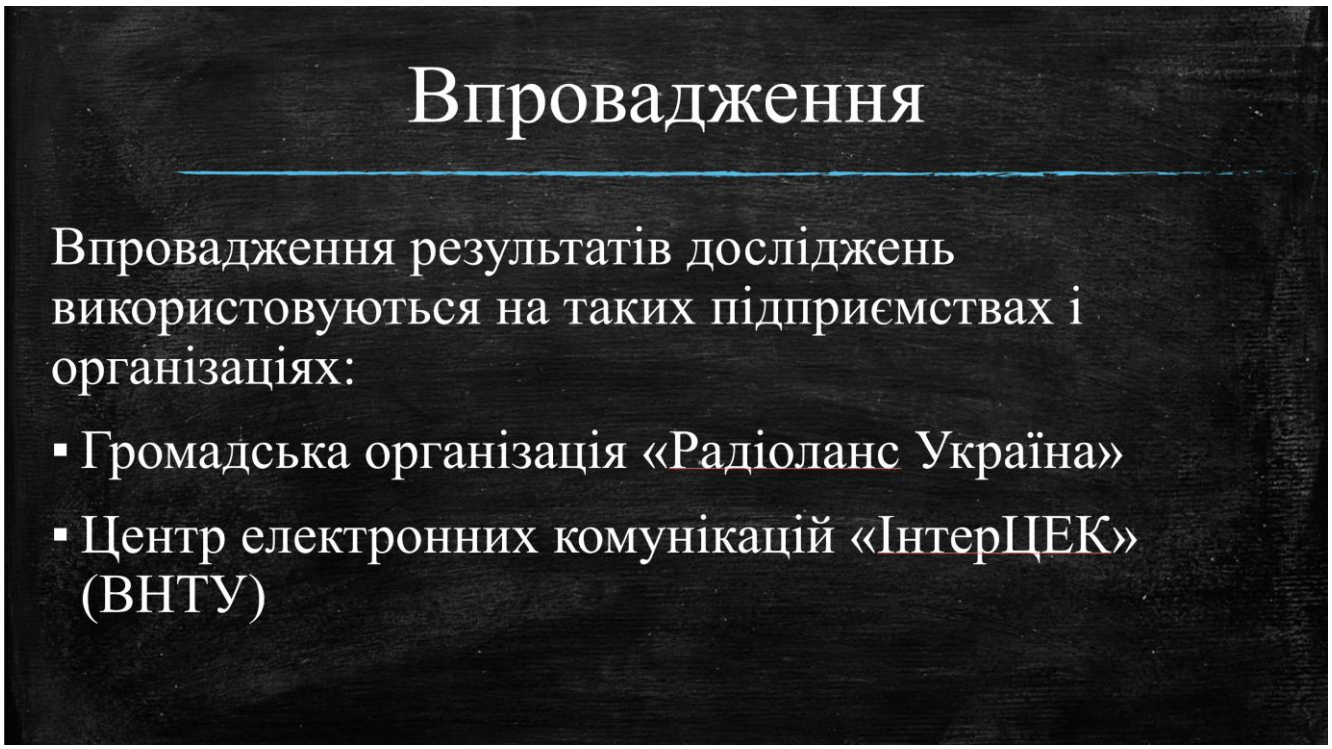


Рисунок Д.5 – Впровадження програмного модулю

Критерії	SolarWinds NCM	N-able RMM	ManageEngine Log360	Kaseya VSA	Власний модуль
Адаптивний інтерфейс	–	+	+	+	+
Сканування підозрілих хостів	+	–	–	+	+
Розподіл мереж за класами захищеності	+	–	–	+	+
Інтеграція з Active Directory	+	+	–	+	–
Підтримка низькобюджетного обладнання	–	–	+	–	+
Загальна оцінка	60%	40%	40%	80%	80%

Рисунок Д.6 – Порівняння аналогів

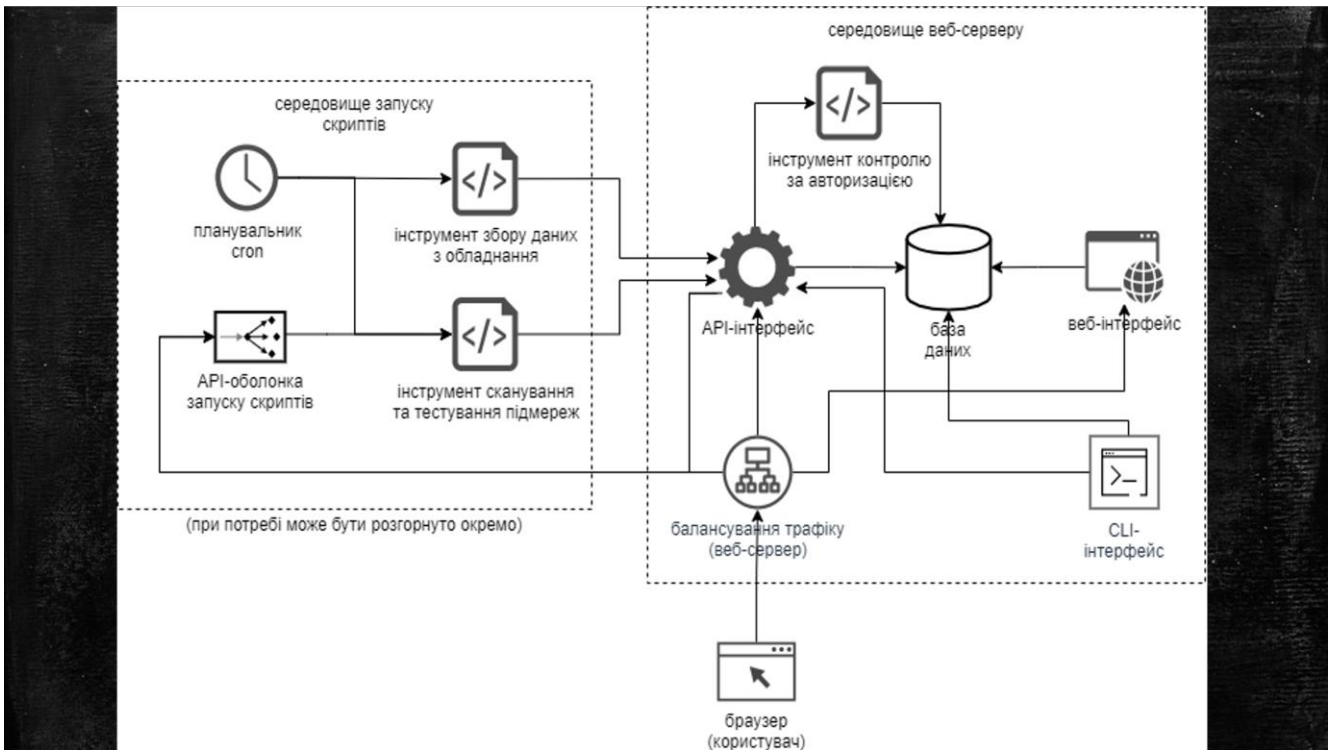


Рисунок Д.7 – Структурна схема програмного модулю

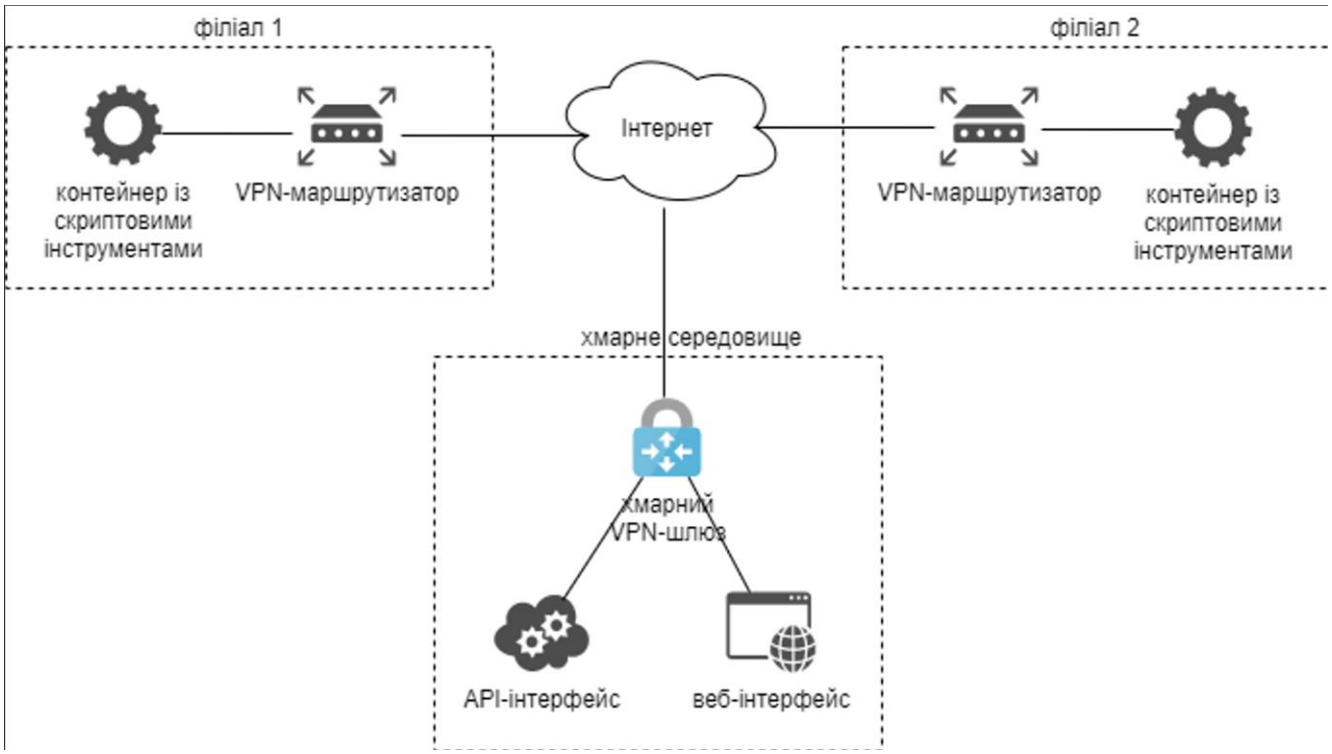


Рисунок Д.8 – Варіант хмарного розгортання модулю із кількома філіалами

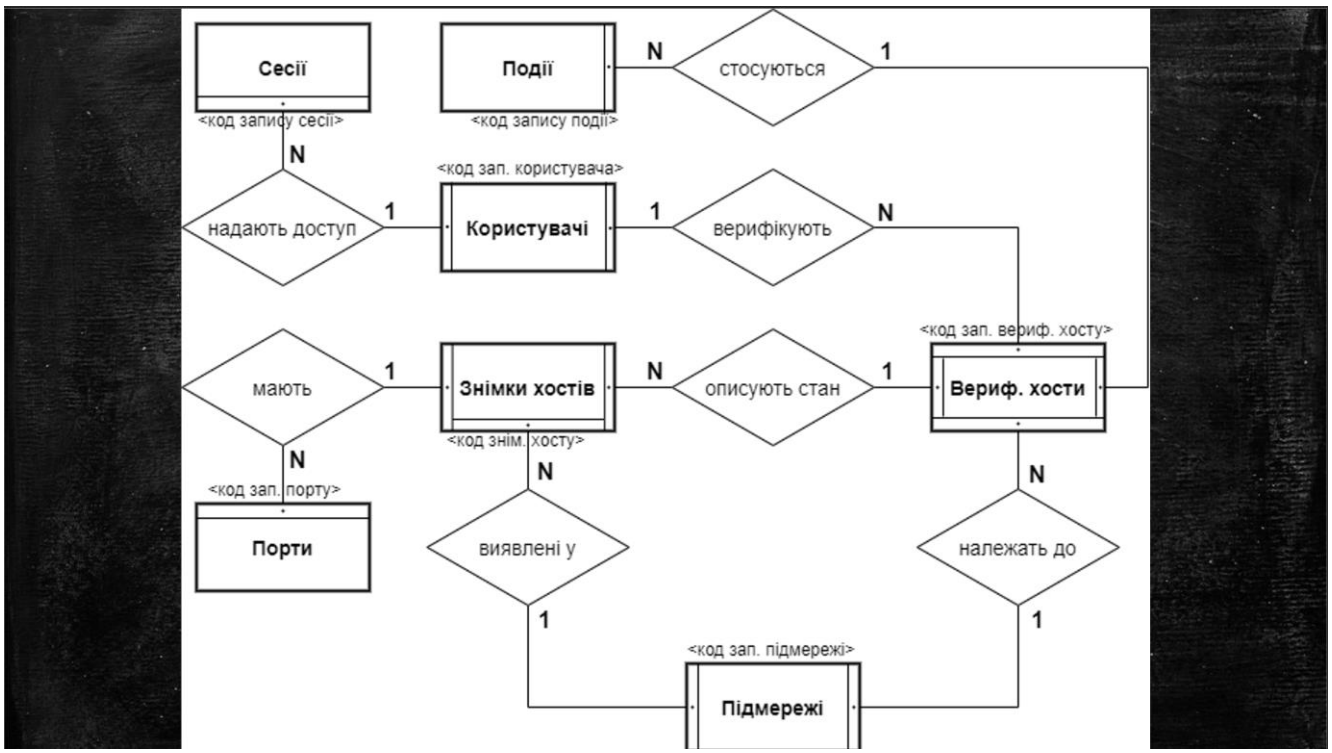


Рисунок Д.9 – ER-діаграма бази даних програмного модулю для аудиту захищеності локальних комп’ютерних підмереж

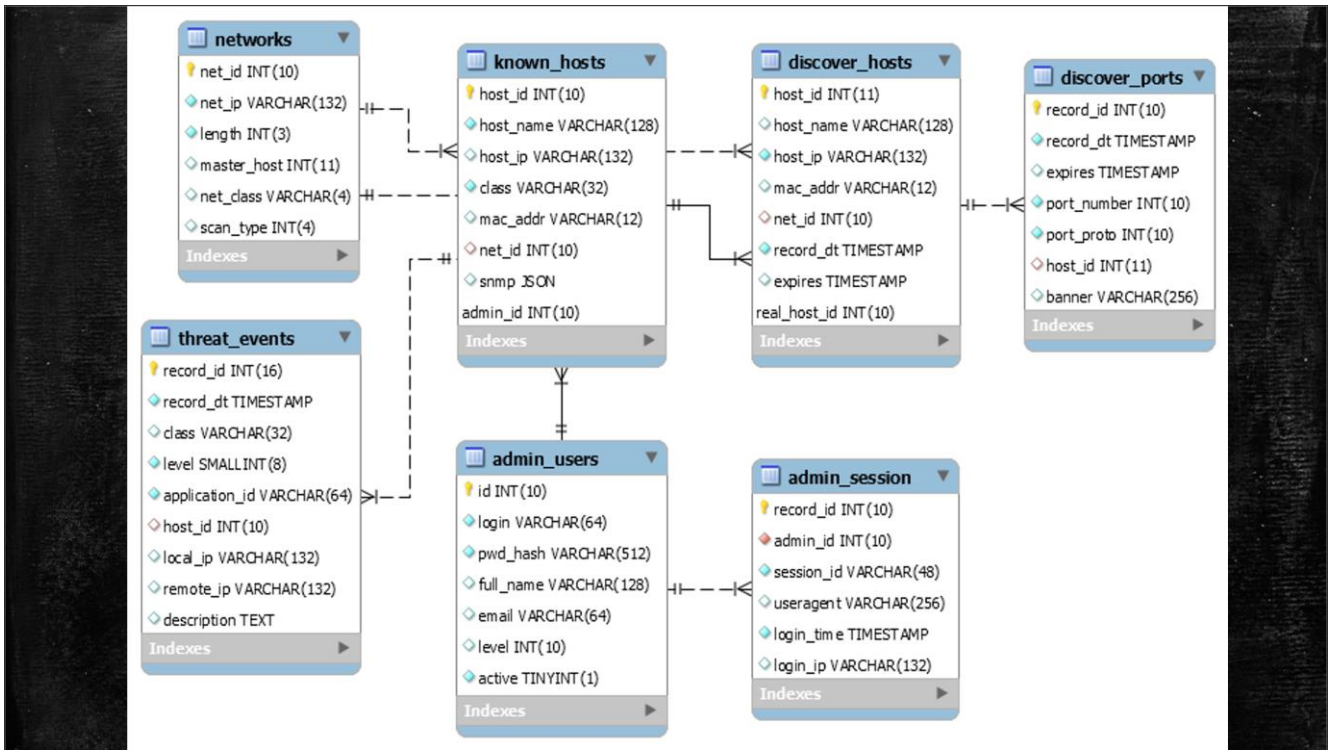


Рисунок Д.10 – Діаграма сутностей бази даних

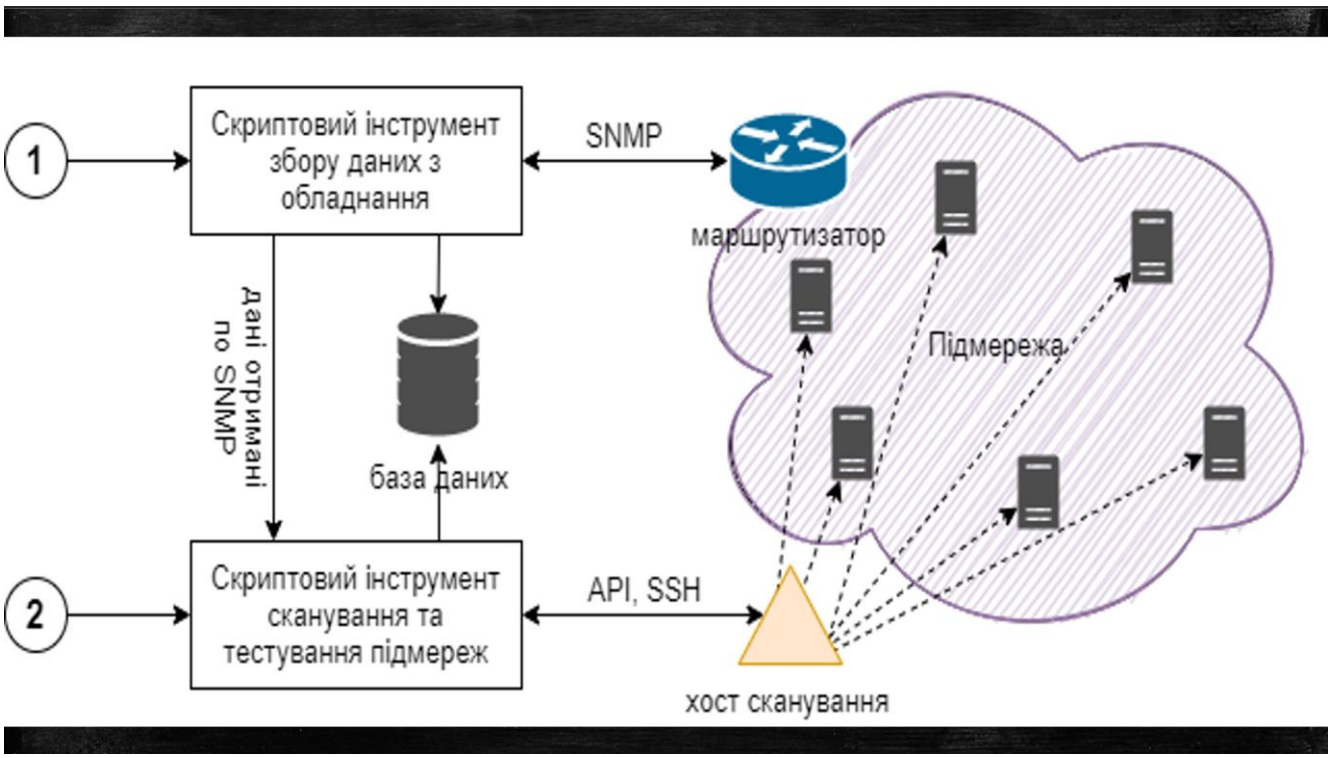


Рисунок Д.11 – Робота методу багаторівневого сканування мережі

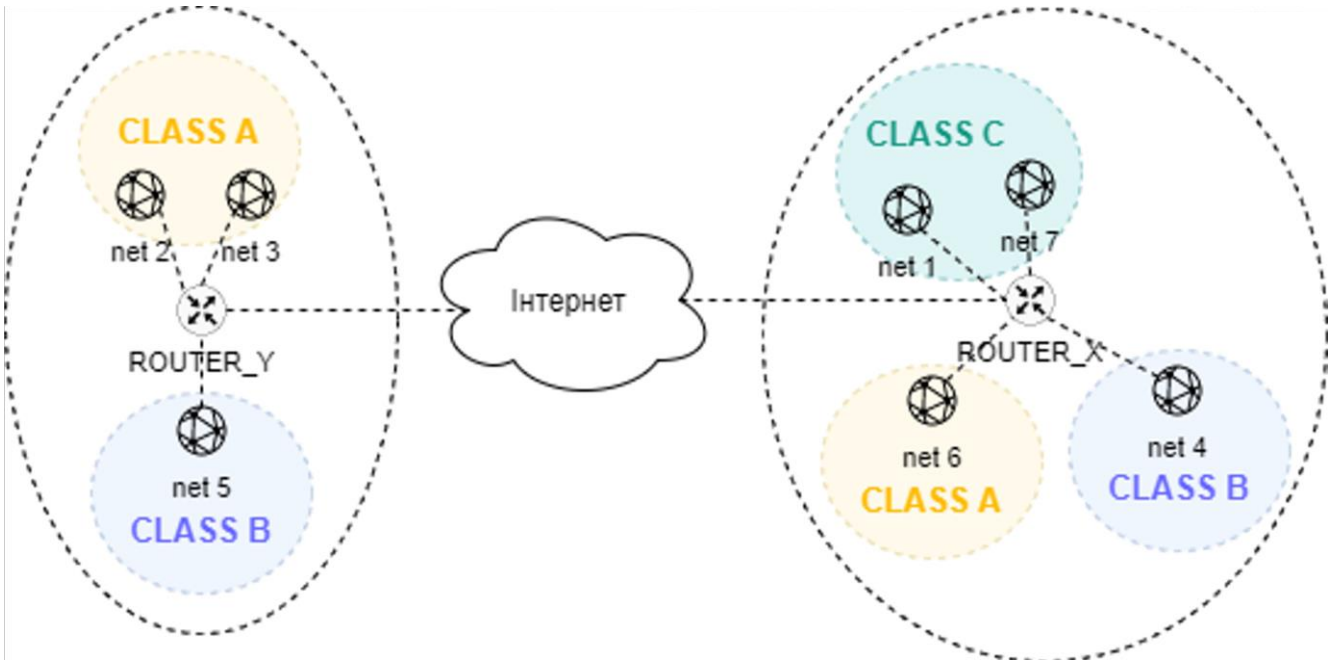


Рисунок Д.12 – Метод побудови карти мережі

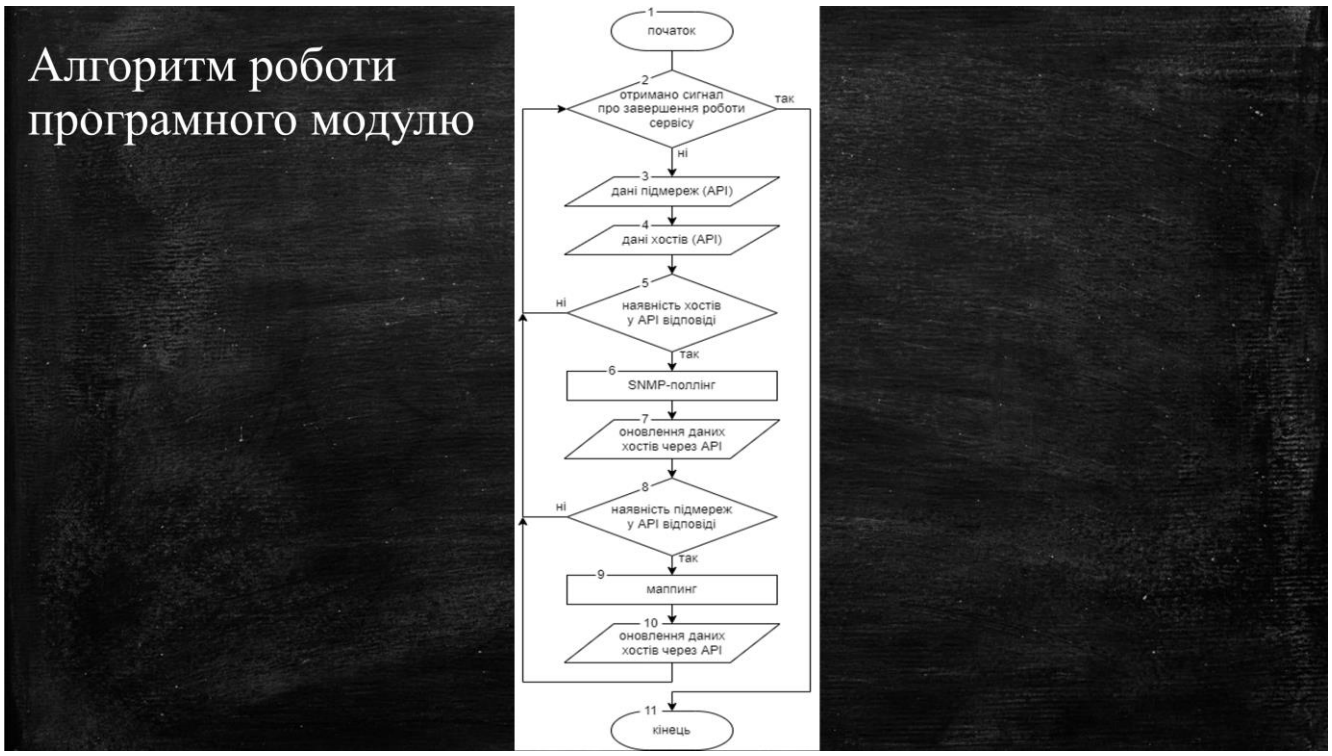
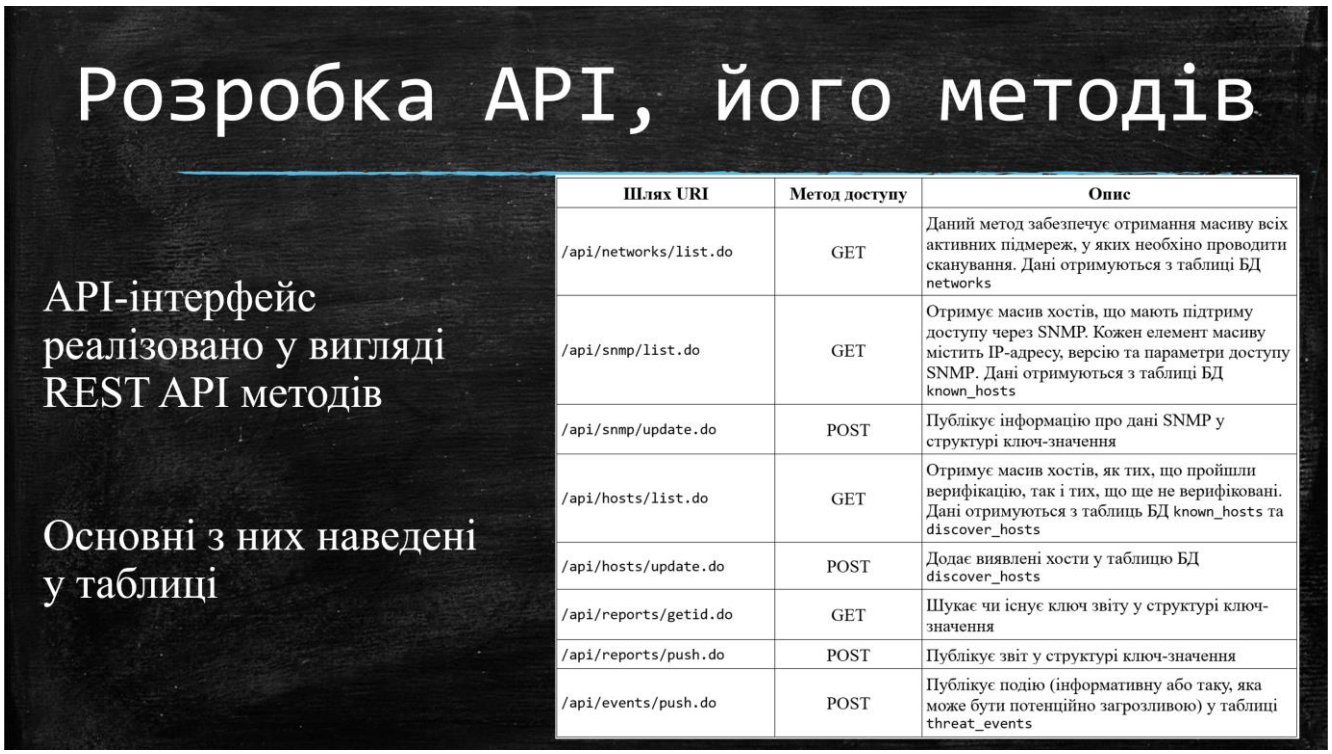


Рисунок Д.13 – Алгоритм роботи програмного модулю



Шлях URI	Метод доступу	Опис
/api/networks/list.do	GET	Даний метод забезпечує отримання масиву всіх активних підмереж, у яких необхідно проводити сканування. Дані отримуються з таблиці БД networks
/api/snmp/list.do	GET	Отримує масив хостів, що мають підтриму доступу через SNMP. Кожен елемент масиву містить IP-адресу, версію та параметри доступу SNMP. Дані отримуються з таблиці БД known_hosts
/api/snmp/update.do	POST	Публікує інформацію про дані SNMP у структурі ключ-значення
/api/hosts/list.do	GET	Отримує масив хостів, як тих, що пройшли верифікацію, так і тих, що ще не верифіковані. Дані отримуються з таблиць БД known_hosts та discover_hosts
/api/hosts/update.do	POST	Додає виявлені хости у таблицю БД discover_hosts
/api/reports/getid.do	GET	Шукає чи існує ключ звіту у структурі ключ-значення
/api/reports/push.do	POST	Публікує звіт у структурі ключ-значення
/api/events/push.do	POST	Публікує подію (інформативну або таку, яка може бути потенційно загрозливою) у таблиці threat_events

Рисунок Д.14 – Розробка API методів

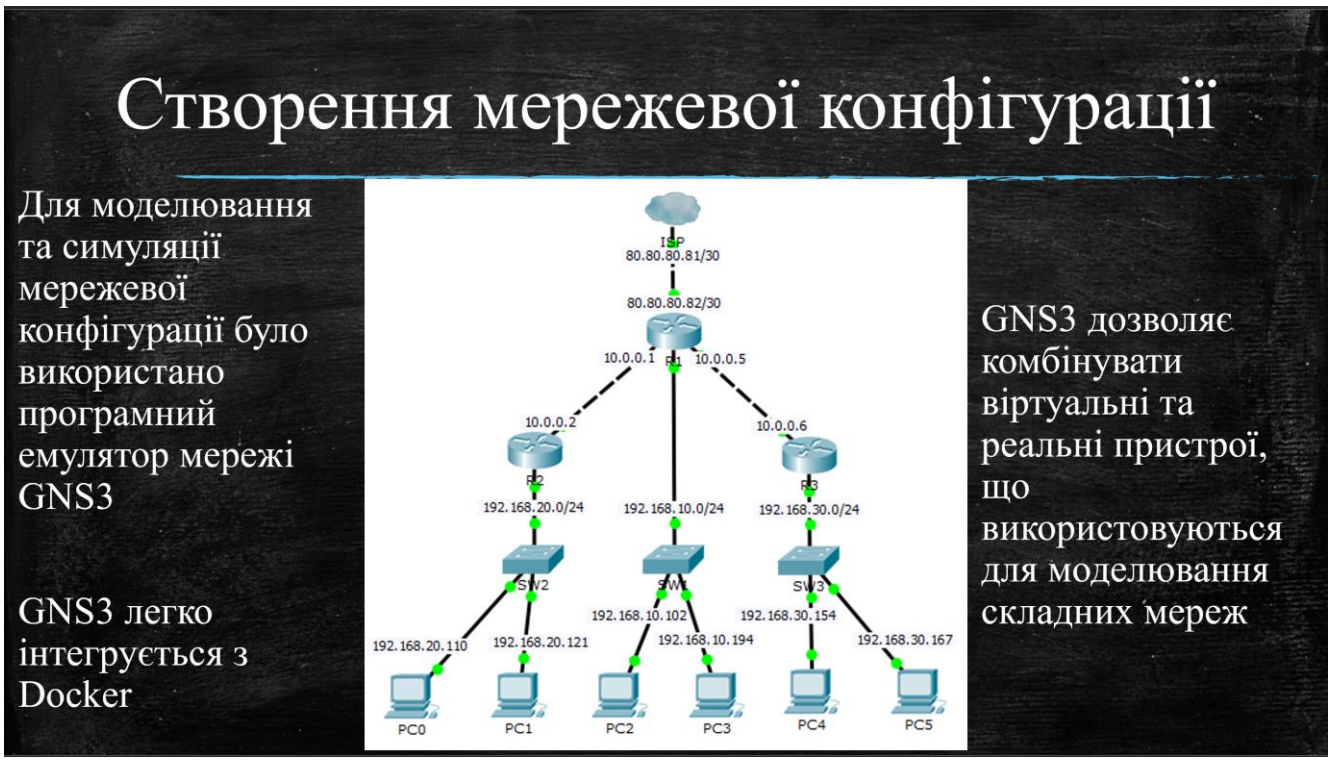


Рисунок Д.15 – Створення мережевої конфігурації

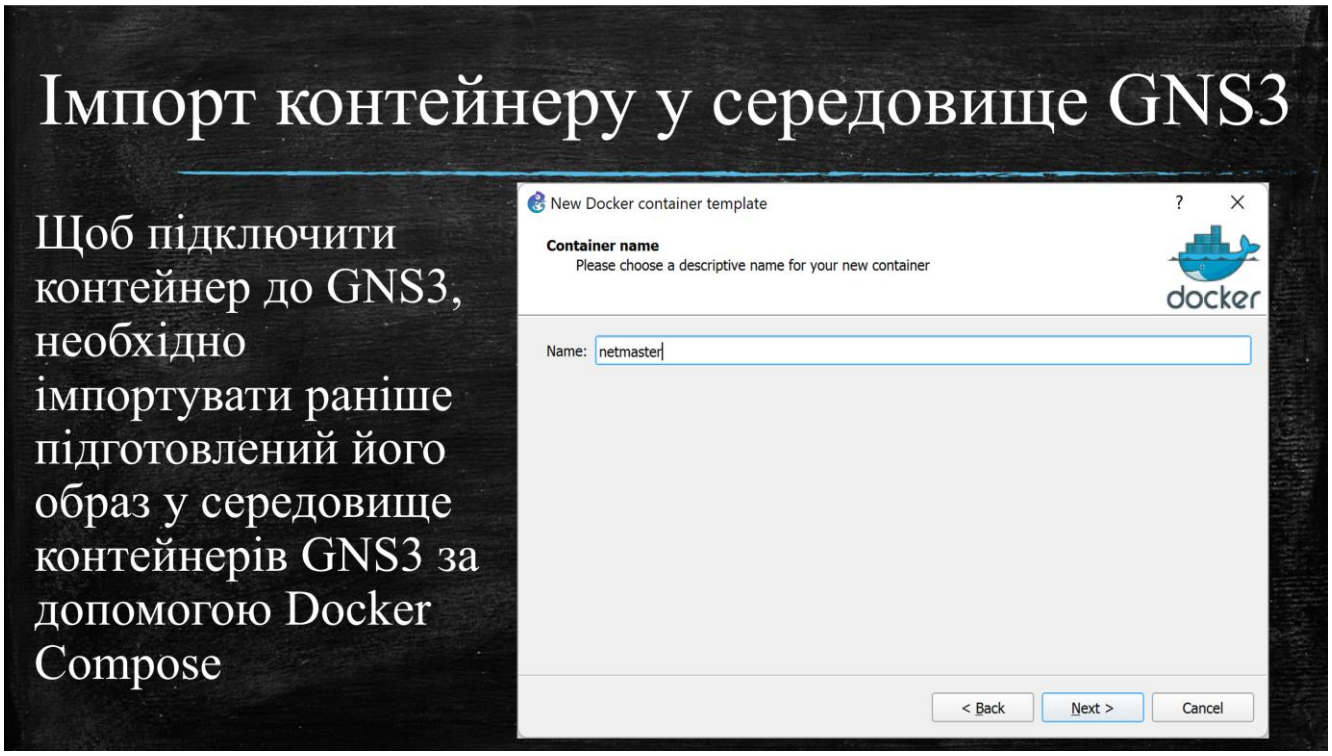


Рисунок Д.16 – Імпорт контейнеру у середовище

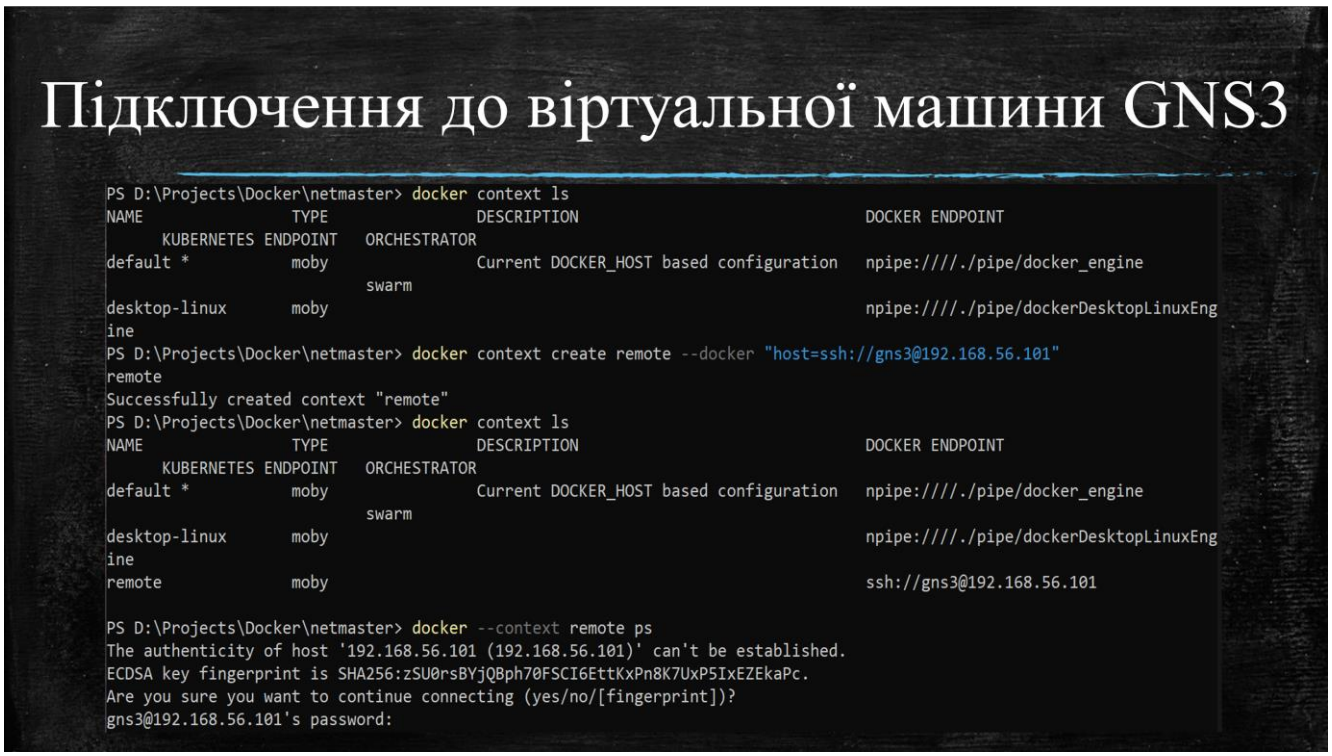


Рисунок Д.17 – Підключення до віртуальної машини

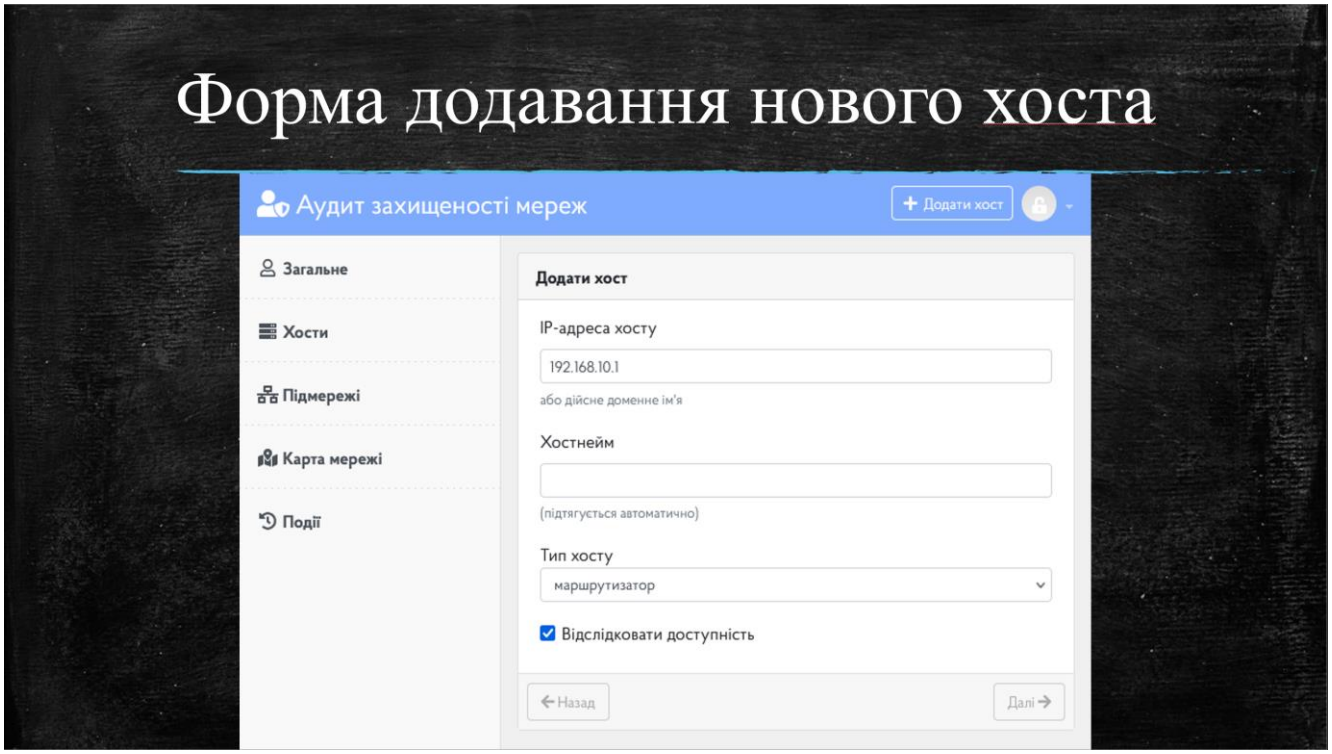


Рисунок Д.18 – Форма додавання нового хоста

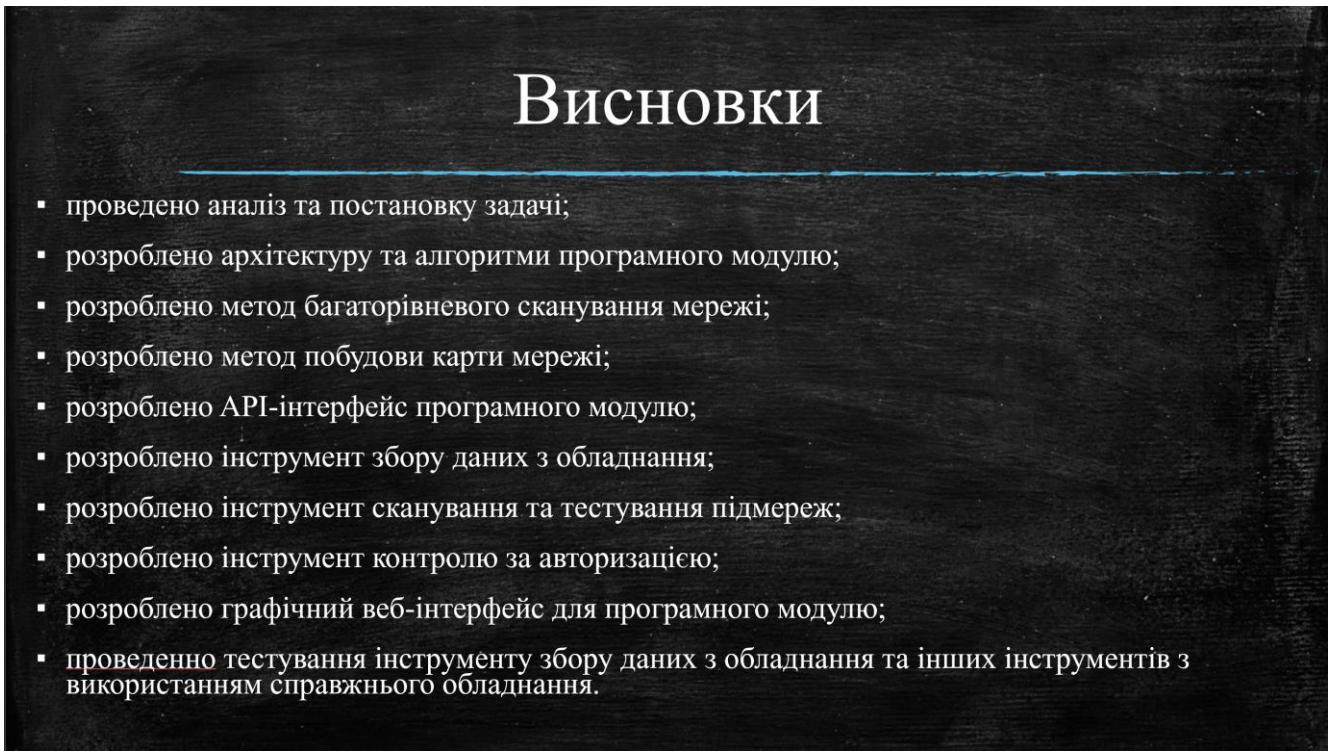


Рисунок Д.19 – Висновки