МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проєкт

здобувача освіти денної форми навчання
КБ.02.14.000.ДП

ПОЛІЩУКА
ІГОРЯ ДМИТРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей

Проєктний матеріал складається з пояснювальної записки на 73 сторінках та графічного (презентаційного) матеріалу на 11 аркушах (слайдах)

Дипломник _____ (Поліщук І.Д.)

Керівник _____ (Гаджиєв М.М.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «26» червня 2025 р. Протокол ЕК № 5

Оцінка ЕК 5 (відмінно) / 90 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*
Спеціальність *123 «Комп'ютерна інженерія»*
Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:
Заст. дир. з НВР *Беркань І.В.*
"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Поліщука Ігоря Дмитровича*
(прізвище, ім'я, по батькові)

1. Тема проєкту *Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей*

затверджена наказом по коледжу від "*14*" *листопада* 202*4* р. № *246*

2. Термін здачі закінченого проєкту

3. Вихідні данні до проєкту *1. Розробити програмний застосунок для сканування мережевих портів з можливістю перевірки окремих портів або діапазонів портів на одному чи кількох IP-адресах. 2. Забезпечити ідентифікацію відкритих портів та визначення служб, що працюють на цих портах, з використанням локальної бази даних у форматі JSON. 3. Використати мову Python для реалізації програмного застосунку. 4.Забезпечити вивід результатів сканування у зручному для сприйняття вигляді з використанням кольорових позначень для відкритих портів.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Побудова моделі програмного застосунку для сканування мережевих портів; Аналіз існуючих програмних рішень для порт-сканування; Аналіз архітектури та структури програмного застосунку для сканування портів; Розробка логіки роботи застосунку для виявлення відкритих портів і визначення служб; Побудова діаграми взаємодії модулів програмного забезпечення для аналізу мережевих порт.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)
Блок-схема архітектури застосунку; Принцип роботи асинхронного сканування портів із використанням семафору; Діаграма послідовності взаємодії функцій у програмі; Приклад формату JSON-файлу зі списком портів і описами сервісів; Схема можливого розвитку застосунку; Приклад виводу результатів у командному рядку; Презентація з результатами тестування.

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Гаджиєв М.М.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник *Гаджиєв М.М.* _____
(підпис)

Завдання прийняв до виконання *Поліщук І.Д.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Вступ. Постановка задачі проєктування*	19.05.2025	Виконав
2	*Аналіз технічного завдання та пошук літератури*	19.05.2025	Виконав
3	*Побудова моделі програмного застосунку для сканування мережевих портів*	21.05.2025	Виконав
4	*Аналіз існуючих програмних рішень*	22.05.2025	Виконав
5	*Аналіз архітектури та структури програмного застосунку*	24.05.2025	Виконав
6	*Використання методів сканування портів та підходів до аналізу мережевої безпеки*	26.05.2025	Виконав
7	*Розробка структури програмного застосунку для виявлення відкритих портів і визначення служб*	27.05.2025	Виконав
8	*Розробка діаграми взаємодії модулів програмного застосунку*	28.05.2025	Виконав
9	*Розробка інтерфейсу командного*	30.05.2025	Виконав
10	*Реалізація програмного застосунку*	2.06.2025	Виконав
11	*Випробування застосунку та аналіз результатів*	4.06.2025	Виконав
12	*Виконання економічних розрахунків*	9.06.2025	Виконав
13	*Розробка питань з охорони праці та техніки безпеки*	11.06.2025	Виконав
14	*Підготовка мультимедійної презентації проєкту*	16.06.2025	Виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02.14 000. 00 ДП	Дипломний проект		
А4			КБ 02.14 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02.14 000. 00 ДП		
Розробив		Поліщук І.Д.		18.06	Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей	Літ.	Аркуш 4 / Аркушів 73
Перевірив		Гаджиєв М.М.		18.06		Н Д П	
Н. контр.		Петрашова В.І.		18.06.25		ВСП "ОТФК ОНТУ" гр.4КБ-02	
Затверд.		Кривченко Ю.В.		18.06			

ЗМІСТ

ВСТУП

Вміння знаходити вразливості комп'ютера є обов'язковою частиною для безпечного перебування в цифровому середовищі. Кожного дня прості користувачі та компанії можуть бути під загрозою втрати даних або стати жертвами кіберзлочинів і хакерських атак через вразливість їх комп'ютерів або комп'ютерних систем. Одними з потенційних цілей для атак є відкриті мережеві порти, які виконують обмін даних між комп'ютерами. Аналіз та виявлення відкритих портів є одним із найважливіших етапів для оцінки стану інформаційної безпеки.

Актуальність теми обумовлена в необхідності мати простий, гнучкий та ефективний інструмент для аналізу мережі. Хоч і на сьогоднішній день існує багато альтернативних рішень для сканування портів, але більшість з них або складні у використанні чи потребують додаткового налаштування. З цієї точки зору створення власного застосунку, який може сканувати порти та ідентифікувати служби, які працюють на цих портах, має сенс.

Мета роботи - створити простий у використанні та ефективний програмний застосунок для сканування мережевих портів, що дозволяє виявляти відкриті порти, ідентифікувати служби, які працюють на ньому та оцінювати потенційну небезпеку.

У дипломному проєкті також приділено увагу економічним аспектам розробки програмного забезпечення. Виконано орієнтовні розрахунки трудових і часових витрат на створення продукту та зроблено висновки щодо доцільності його реалізації з погляду витрат ресурсів.

Крім того, у роботі розглянуто основні вимоги з охорони праці при організації робочого місця розробника програмного забезпечення. Проаналізовано можливі небезпечні та шкідливі фактори, а також заходи для забезпечення безпечних і комфортних умов праці під час виконання проєкту.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз загроз, пов’язаних з відкритими портами

Відкриті порти, особливо ті, які не використовуються або на яких працюють застарілі та некоректно налаштовані служби, становлять значну загрозу для безпеки інформаційних систем. Зловмисники активно сканують мережі у пошуках таких відкритих точок входу, щоб використовувати їх для проведення атак. Однією з найпоширеніших загроз є несанкціонований доступ. Якщо на відкритому порті працює служба з відомою вразливістю, зловмисник може отримати контроль над системою. Наприклад, відкритий порт 22 (SSH) зі слабким паролем може дозволити неавторизованому користувачеві віддалено підключитися до сервера.

Іншою значною загрозою є атаки типу "відмова в обслуговуванні" (DoS) або розподілені DoS-атаки (DDoS). Зловмисники можуть використовувати відкриті порти для перевантаження служб великою кількістю запитів, що призведе до вичерпання ресурсів системи та відмови в обслуговуванні для користувачів. Наприклад, відкритий порт 80 (HTTP) може стати ціллю для флуду HTTP-запитами.

Крім того, відкриті порти можуть бути використані для впровадження шкідливого програмного забезпечення, такого як віруси, трояни, шпигунське ПЗ або програми-вимагачі. Після отримання доступу через відкритий порт, зловмисники можуть завантажити та виконати шкідливий код на цільовій системі, що призведе до компрометації даних, шифрування файлів або використання системи як частини ботнету.

Нарешті, відкриті порти можуть стати джерелом витоку конфіденційної інформації. Деякі служби, котрі працюють на відкритих портах, можуть надавати метадані, версії програмного забезпечення або навіть чутливі дані, які можуть бути використані зловмисниками для подальших атак або спостереженням. Усунення загроз, пов'язаних з відкритими портами, вимагає регулярного сканування мережі, ревізії налаштувань безпеки та своєчасного оновлення програмного забезпечення.

1.2 Аналіз існуючих інструментів для сканування портів

На сьогоднішній день на ринку існує багато альтернатив для сканування портів одними з таких є: Nmap, Masscan та Netcat, але вони не ідеальні у кожного є свої недоліки. Ціль цього етапу – це розібратися в перевагах і недоліках існуючих рішень та відштовхуючись від цього буде проєктуватися майбутній застосунок.

Одним з найпопулярніших і найбільш використовуваних сканерів портів вважається Nmap (Network Mapper), розроблений Гордоном Лайоном в 1997 році. Nmap підтримує широкий спектр технік сканування, включаючи SYN Scan, TCP Connect Scan, UDP Scan, а також різні stealth-методи (FIN, Null, Xmas), працює у командному рядку, визначає служби котрі працюють на портах, а також визначає ОС цілі, котру сканують.

Переваги:

— Універсальність. Підтримує безліч типів сканування (TCP, SYN, UDP, XMAS тощо);
— Визначення служб. Може виявляти версії сервісів та операційних систем;
— Гнучкість. Багато параметрів і можливостей для налаштування під конкретні завдання;
— Можливість використання NSE (Nmap Scripting Engine) для додаткового аналізу;
— Підтримується на Windows, Linux та macOS.

Недоліки:

— Високий поріг входу для новачків (для ефективного використання Nmap користувач повинен розуміти мережеві протоколи, типи сканування, параметри команд);
— Потребує знань параметрів командного рядка (для запуску базового сканування, користувачу потрібно знати правильну команду, а для більш складних операцій йому прийдеться вивчати десятки флагів);
— Надлишковий функціонал для базових завдань (Nmap має багато можливостей, які не потрібні при виконанні стандартного аналізу відкритих

портів, що ускладнює використання програми у простих задачах).

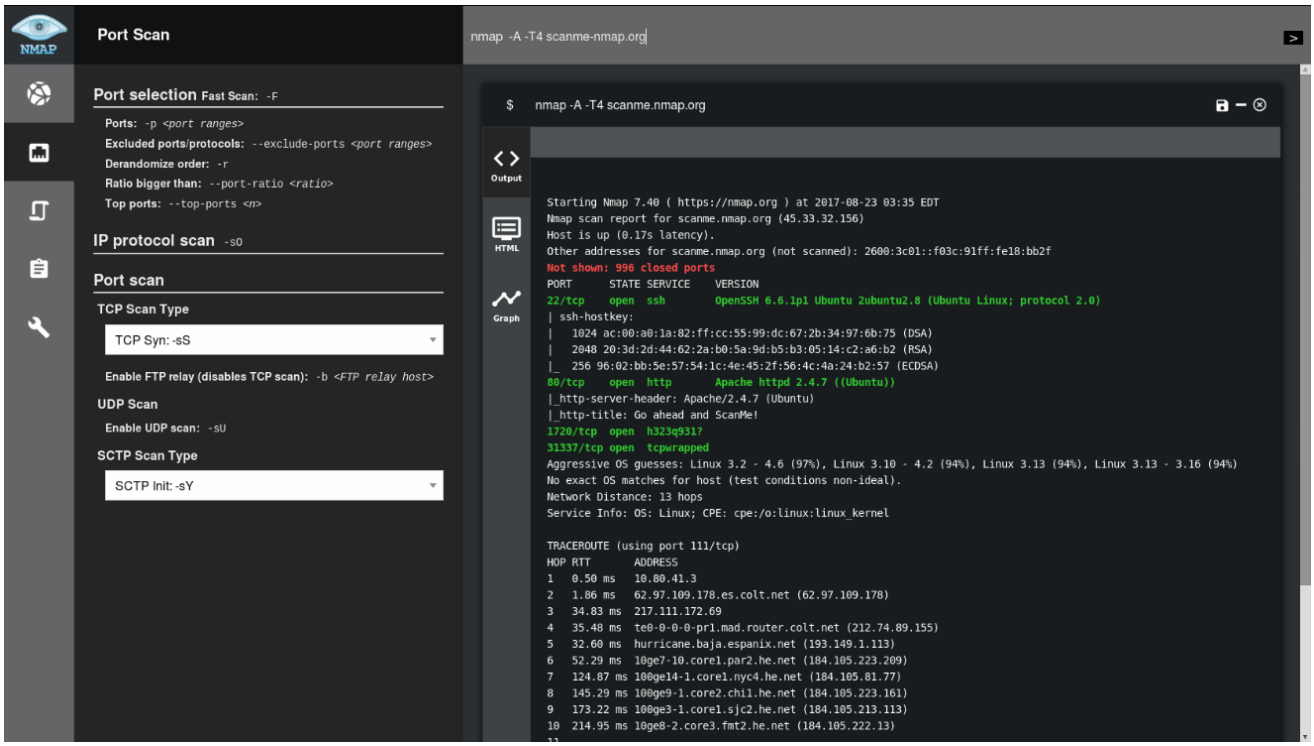


Рисунок 1.1. Інтерфейс програми Nmap

Немало відомим сканером є Masscan, розроблений Робертом Гремом в 2013 році. Masscan було розроблено лише для однієї цілі – максимально швидке сканування великих діапазонів IP адрес. Однією з важливих переваг цього сканеру – це велика швидкість сканування, він може відправляти мільйони пакетів на секунду. Його швидка робота реалізована за допомогою спеціально створеного TCP/IP стека.

Переваги:

— Висока швидкість сканування. Masscan здатний сканувати сотні тисяч або навіть мільйони портів за секунду;

— Простота використання для масового сканування. Оптимізований для перевірки великих діапазонів IP-адрес;

— Мінімальне споживання ресурсів при правильному налаштуванні. Може ефективно працювати навіть на стандартному обладнанні;

— Підтримка сканування великих підмереж;

— Підтримується на Windows, Linux та macOS.

Недоліки:

— Не показує версії сервісів або опис портів (програма повідомляє лише про те, що порт відкритий, не надаючи жодної інформації про те, який саме сервіс його використовує або яка версія програми запущена на цьому порті);

— Не визначає типи служб (не надає інформації яку службу використовує відкритий порт).

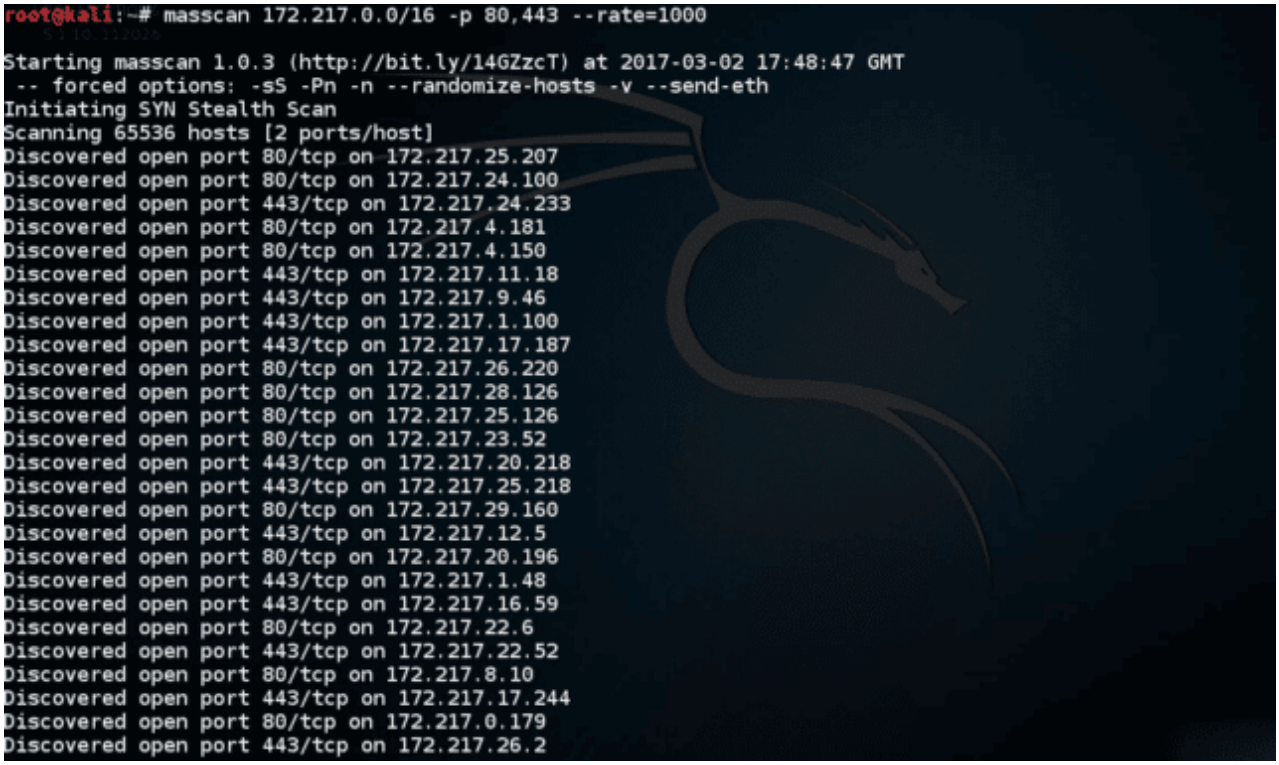


Рисунок 1.2. Інтерфейс програми Masscan в командному рядку

Netcat — це не зовсім сканер портів, а інструмент для низькорівневої роботи з мережею. Його можна використовувати для перевірки портів, передачі файлів, прослуховування сокетів тощо. Він може бути використаний як зловмисниками, так і аудиторами безпеки. Враховуючи сценарій атаки, цей крос-функціональний інструмент може керуватися скриптами, що робить його досить надійним, а також допоможе налагодити та дослідити мережу.

Переваги:

— Дуже гнучкий. може слухати, відправляти, з’єднуватися;

— Підтримується на Windows, Linux та macOS;

— Універсальність. Може використовуватись як сканер портів, тестер з'єднань, засіб для передачі даних.

Недоліки:

— Не автоматизований — не зручно для сканування діапазонів (Netcat не має вбудованих функцій для перевірки декількох портів одразу, тому користувачу потрібно вручну задавати кожен порт);

— Немає кольорового або структурованого виводу (Усі результати виводяться простим текстом, що ускладнює читання при великому обсязі даних);

— Не показує типи служб або статус порту без додаткової логіки (Netcat не ідентифікує, яка служба працює на порту - це потрібно реалізовувати самостійно);

— Потребує глибших знань мережі (Щоб ефективно працювати з Netcat, необхідно розуміти TCP/UDP протоколи та принципи з'єднання).

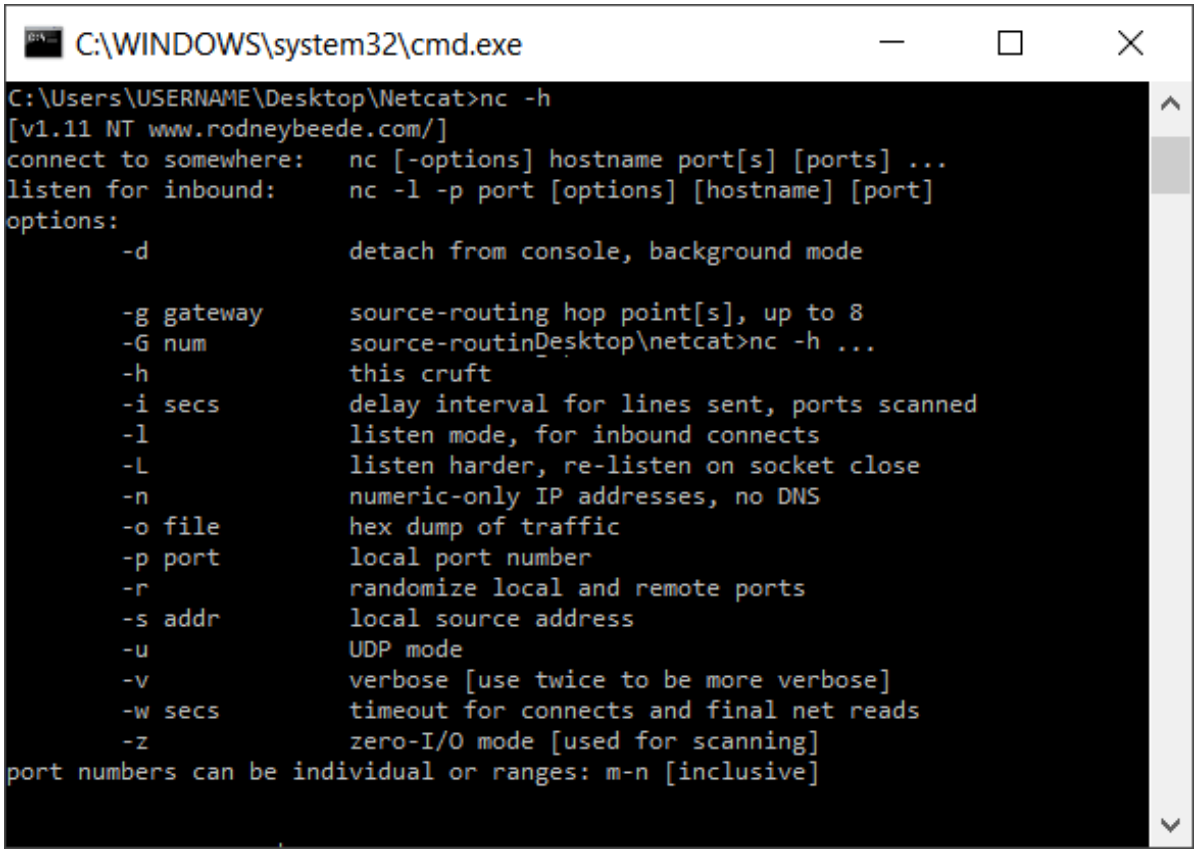


Рисунок 1.3. Інтерфейс програми Netcat в командному рядку

Програми, такі як Nmap, Masscan та інші, користуються високим попитом і мають славу потужних інструментів у професійному середовищі. Але функціональність у багатьох випадках або надлишкова, або занадто складна для базових типових завдань первинного аналізу безпеки. Отже, на основі

переглянутих програм можна створити список вимог з переваг та недоліків, котрі треба буде врахувати при реалізації програмного застосунку:

— Просте меню (створити інтуїтивно зрозуміле меню у котрому користувач зможе розібратися самостійно);

— Назви служб, котрі працюють на порті (коли програма буде показувати результат сканування, до номеру порта також буде додана назва служби, котра працює під цим номером);

— Висока швидкість (реалізувати високу швидкість сканування);

— Підтримка сканування однієї IP адреси, або декількох (додати можливість користувачу вказувати одну або декілька IP-адрес для сканування);

— Можливість сканувати один порт або в певному діапазоні (додати можливість вибирати користувачу між скануванням певного порта або в певному діапазоні).

1.3 Вибір методу сканування

TCP connect - це один із найпростіших і надійних способів сканування. Принцип його роботи полягає в тому, що програма відкриває повноцінне з’єднання за стандартною процедурою TCP. Це означає, що на кожному порту, який перевіряється, послідовно виконується трьохетапний процес встановлення з’єднання. Якщо з’єднання вдалося встановити, порт вважається відкритим і доступним для взаємодії. У разі, якщо порт закритий, система миттєво надсилає відповідь із повідомленням про відмову у з’єднанні.

Переваги:

— Простий в реалізації;

— Не потребує прав адміністратора;

— Гарантовано визначає, чи порт відкритий;

— Працює на будь яких платформах.

Недоліки:

— Легко виявляється фаєрволами та системами IDS;

— Створює навантаження на цільовий хост;

— Повільніший у порівнянні з іншими методами;

— Низька швидкість в порівнянні з іншими методами скануваннями.

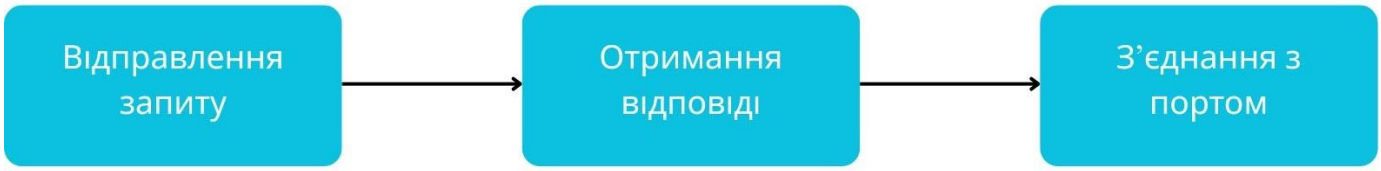


Рисунок 1.4 З'єднання за допомогою TCP connect до відкритого порта

UDP-сканування перевіряє, чи відкриті UDP-порти. Відправляється порожній або спеціальний пакет, і очікується відповідь. Відсутність відповіді може означати відкритий порт, ICMP повідомлення - закритий.

Переваги:

— Дає змогу перевірити UDP-служби (DNS, SNMP, DHCP);

— Обхід фаєрволів, які блокують лише TCP.

Недоліки:

— Сканування повільне, бо немає підтвердження;

— Більшість серверів не відповідають на невідомі UDP-запити;

— Складно інтерпретувати результат.

SYN-сканування (іноді називають "stealth scan") використовує першу фазу TCP - надсилання пакета SYN. Якщо хост відповідає SYN-ACK, порт вважається відкритим. Після цього програма не завершує з'єднання, а відразу скидає його.

Переваги:

— Швидше та менш помітне, ніж TCP connect;

— Не встановлює повного з'єднання;

— Зменшує ймовірність детектування.

Недоліки:

— Потребує права root/адміна;

— Не підтримується через стандартний socket.

ACK-сканування не визначає, чи порт відкритий, а використовується для виявлення фільтрації фаєрволом. Надсилається TCP ACK-пакет, і залежно від

відповіді можна зробити висновок, чи порт фільтрується.

Переваги:

— Добре підходить для виявлення правил фаєрвола;

— Корисне в пентестингу.

Недоліки:

— Не показує стан порту напряму (лише фільтрацію);

— Складна інтерпретація відповідей.

FIN-сканування — це метод сканування портів, при якому на порт надсилається TCP-пакет з прапорцем FIN без попереднього встановлення з’єднання. За стандартом RFC 793 на такий пакет відкритий порт не повинен відповідати, тоді як закритий порт повертає пакет з прапорцем RST (reset).

Переваги:

— Менш помітне для систем виявлення вторгнень (IDS), ніж TCP connect або SYN-сканування;

— Використовується для обхідних методів виявлення портів у мережах із базовими системами захисту.

Недоліки:

— Цей метод ефективно працює тільки на ОС, що суворо дотримуються RFC 793 (наприклад, Unix/Linux);

— Windows зазвичай ігнорує FIN-пакети незалежно від стану порту, що знижує ефективність сканування.

Для реалізації програми був обран метод TCP connect, оскільки він найбільш універсальний, стабільний і простий у реалізації з використанням стандартних інструментів Python. Він не потребує спеціальних прав адміністратора, дає точний результат та демонстрації принципів роботи сканера портів.

1.4 Розробка логіки роботи програмного застосунку

На цьому етапі розробляється чітка та зрозуміла логіка роботи програмного застосунку, котра буде спиратися на зазначені вимоги з попередніх етапів розробки. Формування логіки програми є ключовим елементом, оскільки вона визначає, як дані користувача перетворюються на функціональні дії застосунку.

Логіку роботи програми можна розбити на декілька модулів.

Перший модуль, введення параметрів. Цей модуль відповідає за взаємодію з користувачем. Функціонал першого модулю:

— Отримання списку IP-адрес цілей (одна або декілька);

— Вибір режиму сканування: одного порта або певного діапазону портів;

— Отримання номеру порта або числового діапазону.

Другий модуль, обробка IP-адрес. Цей модуль виконує аналіз введених даних і створює список задач на сканування. Функціонал другого модулю:

— Розділення введених IP-адрес на окремі елементи (якщо декілька адрес);

— Перевірення коректності форматів введених даних;

— Формування асинхронних задач на кожну IP-адресу з відповідним діапазоном портів.

Третій модуль, сканування портів. Головна частина програми, яка виконує сканування портів. Функціонал третього модулю:

— Встановлення з’єднання з IP-адресою через конкретний порт;

— У випадку, коли порт відкритий – зберігання до списку результатів;

— Програма паралельно обробці результатів продовжує сканування.

Четвертий модуль, зіставлення з локальною базою портів. Цей модуль відповідає за обробку відкритих портів. Функціонал четвертого модулю:

— Програма відкриває локальний JSON-файл, який містить назви служб портів;

— Програма шукає відповідність номеру порта з результату та з JSON файлу;

— Додає цю інформацію до фінального звіту.

П’ятий модуль, вивід результатів. Цей модуль відповідає за зручне та структуроване представлення результатів. Функціонал п’ятого модулю:

— Виділення кольорами інформації для виводу;

— Вивід результатів по кожній IP-адресі окремо.

Коли програма розроблялась, було прийнято рішення поділити її на декілька модулів. Такий підхід дуже зручний — кожна частина відповідає за щось своє, і це допомагає розібратися, як усе разом працює. Також, якщо щось піде не так,

легше знайти проблему чи додати нові функції з часом.

Рисунок 1.5 Послідовність дій роботи програми

Програма працює як послідовний ланцюг модулів. Спочатку користувач вводить IP-адреси (одну чи декілька), потім вибирає режим сканування: один порт чи діапазон портів. Модуль введення відповідає за перевірку коректності введених даних, фільтрує їх та формує список IP-адрес для подальшої обробки.

Далі дані передаються в модуль сканування, де завдяки асинхронним задачам перевірка портів відбувається швидко й одночасно для кількох адрес. Щоб система не перевантажувалася, використовується семафор, який обмежує кількість активних підключень до тисячі та знижує ризик помилок під час великої кількості запитів.

Результати сканування обробляються наступним модулем: він звертається до бази даних у форматі JSON, щоб додати опис знайдених відкритих портів і можливих служб на них. Під час роботи програма видає підказки у випадку помилок введення. Завершальний модуль виводить результати у зручному вигляді з кольоровими підсвічуваннями й групуванням за IP-адресами для кращого вигляду.

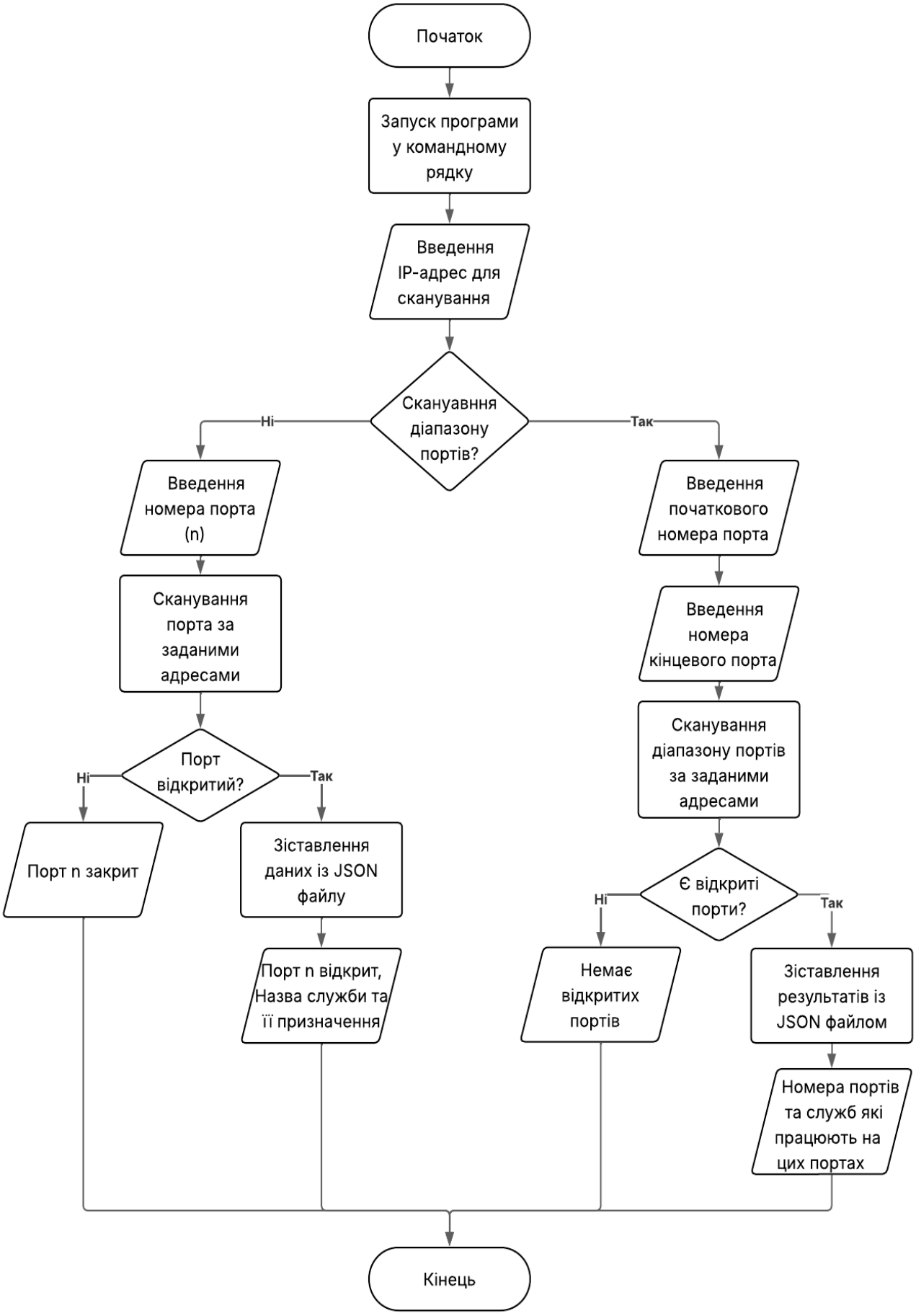


Рисунок 1.6. Блок схема алгоритму роботи програми

Програма працює як послідовність логічних етапів, які запускаються один за одним. На кожному етапі виконуються окремі функції:

1. Введення даних — користувач вказує IP-адресу або кілька, а також режим сканування (один порт або діапазон).
2. Обробка введення — програма перевіряє правильність IP, обробляє список адрес.
3. Сканування портів — для кожного IP виконується асинхронне підключення до заданих портів.
4. Фільтрація результатів — зберігаються лише відкриті порти.
5. Ідентифікація служб — за портами шукається інформація у локальному JSON-файлі.
6. Вивід результатів — на екран виводиться інформація для кожного хоста окремо, з кольоровим маркуванням.

Загалом, уся система працює просто: беруться дані, скануються порти, розбір, що до чого, і видача результатів. Модулі пов’язані між собою, але кожен із них можна спокійно підправити чи допрацювати. У перспективі можна додати можливість зберігати результати в файл, зробити простий графічний інтерфейс.

1.5 Вибір технологій для реалізації застосунку

1.5.1 Вибір мови програмування

Для створення програмного застосунку для сканування портів було обрано мову програмування Python, тому що ця мова має простий синтаксис, потужні інструменти для роботи з мережами та величезну кількість готових бібліотек. Python часто застосовують у кібербезпеці, для аналізу мереж, тестування на проникнення, а також у численних проєктах із відкритим кодом. Основними критеріями вибору були:

— доступність технологій;

— наявність бібліотек для роботи з мережею;

— продуктивність при скануванні великої кількості портів;

— зручність для виводу результатів;

— можливість подальшого розширення.

Причини використання Python:

— Синтаксис, який легко читати. Python вирізняється зрозумілістю навіть для тих, хто не є профі у програмуванні. Це дозволило зосередитися на розробці логіки сканування, а не боротися зі складними елементами мови. Наприклад, щоб налаштувати сокет чи підключитися до сервера, достатньо кількох рядків коду.

— Вбудовані можливості для роботи з мережею. У Python є стандартний модуль socket, який забезпечує роботу з протоколами TCP і UDP без необхідності встановлювати додаткове програмне забезпечення. Завдяки було отримано прямий доступ до мережевих функцій, що критично важливо для сканера портів.

— Асинхронність з asyncio. Для швидкого й ефективного сканування я скористався бібліотекою asyncio, яка підтримує асинхронний підхід. Це дає змогу одночасно перевіряти безліч портів, не чекаючи завершення кожного з'єднання. Починаючи з версії 3.4, Python має вбудовану підтримку асинхронності, що спрощує паралельну обробку без використання потоків.

— Багато корисних бібліотек. Крім asyncio, було додано termcolor для кольорового оформлення виводу в терміналі та json для взаємодії з базою даних портів. У Python усі ці інструменти або вже доступні, або їх можна легко встановити через менеджер пакетів pip.

— Працює на будь-якій системі. Python сумісний із Windows, Linux, macOS та іншими платформами. Завдяки цьому застосунок запускається без змін у коді на різних пристроях, що значно полегшує тестування й демонстрацію.

— Популярність у сфері кібербезпеки. Python є однією з популярних мов серед спеціалістів з інформаційної безпеки. Такі відомі інструменти, як Nmap, Wireshark, Scapy чи Metasploit, або написані на Python, або мають модулі для роботи з ним. Це підтвреджує, що мова ідеально підходить для створення сканерів, аналізаторів і автоматизації мережевих завдань.

Таблиця 1.1. Порівняння Python з іншими мовами програмування

	Python	C/C++	Java
Простота	Дуже простий код для читання	Складні мови синтаксично	Складна структура
Швидкість розробки	Швидка	Повільна	Середня
Підтримка асинхронності	Вбудована через asyncio	Потрібні додаткові бібліотеки	Є, але складно реалізувати
Документація, навчальні матеріали	Багато матеріалів, легко знайти	Є, але технічна	Багато офіційної документацій
Можливість подальшого розширення	Легко додати GUI, API	Треба писати вручну	Можливо, через фреймворки
Кросплатформеність	Так	Так, але потрібна компіляція	Так

1.5.2 Асинхронне програмування

Під час реалізації сканера портів була проблема, що програма послідовно сканувала велику кількість портів і це займало дуже багато часу. Це відбувалося тому, що програма перевіряла кожен порт по черзі, чекаючи відповіді від одного, перш ніж перейти до наступного. Особливо повільно це працювало, коли порт був закритий, адже тоді доводилося чекати таймауту — часу, поки система зрозуміє, що відповіді не буде.

Щоб вирішити цю проблему, було вирішено звернутися до асинхронного програмування. Цей підхід дозволяє виконувати кілька завдань одночасно без створення багатьох потоків чи процесів, які зазвичай потребують більше ресурсів. У Python для цього є бібліотека asyncio, що працює з так званими корутинами.

Для того щоб не виникло проблем при відкриванні багатої кількості з'єднань одночасно, було додано семафор із бібліотеки asyncio. asyncio.Semaphore — інструмент для обмеження кількості одночасно виконуваних асинхронних

задач. Його використання дозволяє стабілізувати роботу застосунку, уникнути перевантаження ресурсів та контролювати інтенсивність мережевого навантаження при обробці великих діапазонів портів.

Корутина — це спеціальна функція, яка може тимчасово зупинятися, дозволяючи іншим функціям працювати, а потім продовжувати свою роботу з того місця, де вона зупинилася. У сканері портів кожна функція, яка перевіряє окремий порт, є корутиною. Завдяки цьому, поки програма чекає відповіді від одного порту, вона не "зависає", а починає перевіряти інші порти.

Наприклад, використовується команда await asyncio.open_connection(ip, port), програма призупиняє роботу з цим конкретним портом, але не зупиняється повністю — вона одразу переходить до сканування інших портів. Це дозволяє перевіряти десятки чи навіть сотні портів одночасно.

Переваги цього підходу

Асинхронне програмування дало мені кілька важливих переваг:

— Швидкість

Завдяки одночасній перевірці багатьох портів час сканування значно скорочується. Наприклад, сканування 1000 портів на 5 IP-адресах виконується в рази швидше, ніж при послідовному підході.

— Економія ресурсів

На відміну від багатопоточності, де для кожного завдання створюється окремий потік, асинхронність працює в одному потоці. Це зменшує навантаження на пам’ять і процесор.

— Масштабованість

За допомогою функції asyncio.gather() можна легко сканувати не одну, а багато IP-адрес одночасно, просто додавши їх до списку завдань.

— Простота коду

У Python для асинхронного програмування достатньо кількох ключових слів: async, await і gather(). Це дозволяє писати зрозумілий і ефективний код навіть без глибокого знання потоків чи процесів.

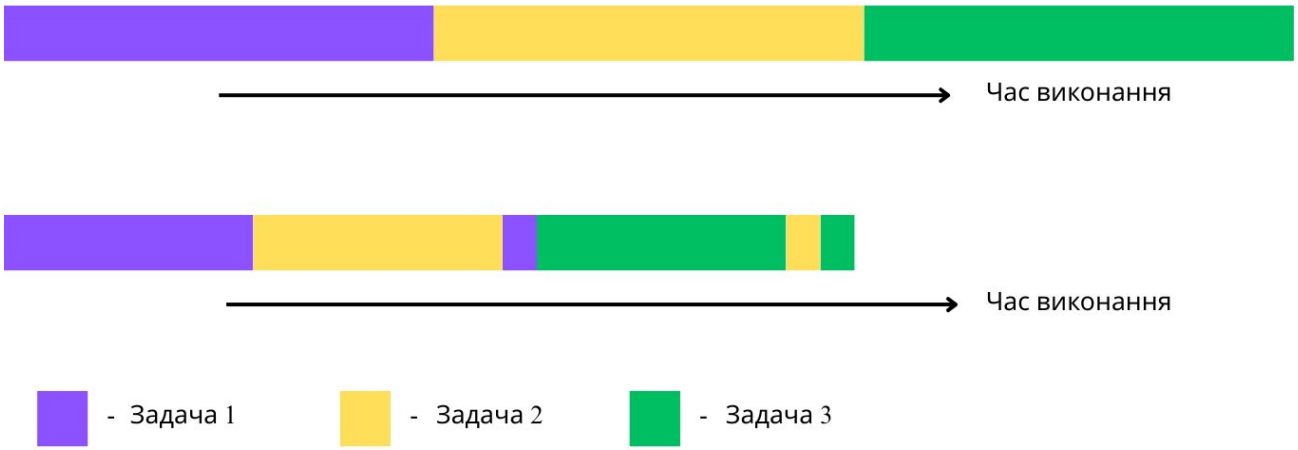


Рисунок 1.7. Порівняння синхронного і асинхронного програмування

У коді програми створено асинхронну функцію scan_port(), яка намагається підключитися до певного порту. Для кожної IP-адреси формується список таких корутин — по одній на кожен порт, який потрібно перевірити. Потім за допомогою asyncio.gather() вони запускаються усі одночасно. Це дозволяє сканувати багато портів на кількох хостах майже паралельно, що значно прискорює процес.

Недоліки та обмеження

Асинхронне програмування — не ідеальне рішення для всіх випадків. У нього є свої особливості:

— Сумісність

Не всі бібліотеки в Python підтримують asyncio, тож іноді доводиться шукати альтернативи або адаптувати код.

— Складність налагодження

Код із корутинами може бути важчим для розуміння та відлагодження.

— Обмеження за типом задач

Асинхронність найкраще працює там, де є багато очікування, наприклад, при мережевих операціях із таймаутами. Для задач із великими обчисленнями краще обрати інший підхід, наприклад, багатопоточність.

1.5.3 Бібліотеки: asyncio, termcolor, json, ipaddress, time

Для реалізації функціональності програмного застосунку було використано низку бібліотек Python, кожна з яких виконує конкретну роль у роботі програми.

asyncio

— Вбудована бібліотека Python для асинхронного виконання завдань;

— Дозволяє запускати сканування портів паралельно без використання потоків;

— Підвищує швидкість обробки великої кількості портів та IP-адрес;

— Працює разом із asyncio.Semaphore для контролю кількості одночасних підключень.

termcolor

— Забезпечує кольоровий вивід тексту в терміналі;

— Додає візуальні підказки: зелений колір — відкриті порти, червоний — закриті, синій — тривалість сканування;

— Покращує зручність роботи з CLI-інтерфейсом програми.

json

— Використовується для завантаження локального файлу з інформацією про порти: номери, служби, описи;

— Дозволяє виводити не просто номери портів, а також їх призначення, роблячи результати сканування більш змістовними;

— Підтримує гнучке розширення функціональності без змін у коді.

ipaddress

— Стандартна бібліотека для перевірки коректності IP-адрес;

— Використовується для валідації введених користувачем значень перед початком сканування;

— Дозволяє уникнути помилок сканування через неправильний формат IP.

time

— Застосовується для вимірювання тривалості сканування;

— Дозволяє виводити користувачу точний час, витрачений на обробку IP чи діапазону портів;

— Полегшує порівняння ефективності між різними конфігураціями сканування.

1.5.4 Методи оптимізації продуктивності сканера

У процесі розробки програмного застосунку важливо не лише забезпечити коректність роботи, а й досягти оптимальної продуктивності. При перевірці великої кількості портів або IP-адрес сканер може створювати значне навантаження на комп'ютер і мережу. Щоб запобігти цього, в проєкті були використані кілька методів оптимізації.

Одним із ключових рішень стало впровадження асинхронного програмування за допомогою модуля asyncio. Завдяки цьому програма використовує ресурси ефективніше, а сканування відбувається значно швидше, оскільки час очікування відповіді від одного порту не блокує перевірку інших.

Для запобігання великого навантаження на операційну систему і уникнення помилок типу «WinError 10055: неможливо виконати операцію на сокеті, оскільки черга переповнена». В застосунку є контроль кількості одночасних підключень за допомогою асинхронного семафора. Семафор дозволяє обмежити кількість паралельно активних задач сканування, встановивши верхню межу. На етапі розробці через те що відкривалось багато з'єднань водночас, то після 10000 порта програма могла не обробляти відповіді від портів, через переповнення списку результатів, але із введенням обмежень ця проблема зникла і програма працює коректно. Це забезпечує стабільну роботу навіть при великій кількості портів і зменшує ризик перевантаження системи або мережі.

Ще одним важливим аспектом оптимізації є налаштування таймаутів підключення. Надто короткий таймаут може призвести до пропуску відкритих портів через тимчасові затримки у відповіді від мережі. Занадто великий — сповільнить роботу сканера, оскільки програма буде довго чекати відповіді від кожного порту. У дипломному проєкті було обрано компромісний варіант — таймаут у 2 секунди, який забезпечує баланс між точністю результатів і швидкістю сканування.

Таким чином, поєднання асинхронних задач, обмеження кількості

одночасних підключень семафором і коректного вибору таймаутів дозволило підвищити продуктивність сканера й забезпечити його стабільну роботу навіть у складних мережевих умовах.

1.5.5 Вивід в командний рядок

На даному етапі застосунок реалізований у вигляді CLI (Command Line Interface) - тобто працює в терміналі. Це було свідомим вибором, з огляду на:

— Простоту реалізації та налагодження;

— Відсутність зайвих візуальних компонентів;

— Зручність демонстрації на будь-якому комп’ютері;

— Швидкий доступ до кольорового форматованого виводу.

CLI дозволяє зосередитись на головному — функціональності сканера, без потреби створювати складний інтерфейс. Однак логіка програми вже побудована так, що її можна легко розширити:

1.6 Архітектура програми

Після вибору технологій наступним важливим етапом стало проєктування архітектури програмного застосунку, що дозволяє ефективно розподілити функціональність, забезпечити гнучкість у розширенні, та зручно підтримувати код. Архітектура була побудована за принципами модульності, незалежності компонентів та підтримки паралельного оброблення кількох IP-адрес.

Програма реалізована у вигляді кількох логічно відокремлених функцій, кожна з яких відповідає за конкретний етап або дію. Це дозволяє:

— легко тестувати кожну функцію окремо;

— змінювати або вдосконалювати одну частину, не порушуючи інші;

— спростити читання й підтримку коду.

Програма складається з кількох основних функціональних блоків, реалізованих як окремі функції. Нижче описані їх ролі:

1. main()

— Це головна керуюча функція;

— Отримує введення від користувача

— Визначає режим роботи (один порт / діапазон);

— Запускає сканування для кожного IP.

2. scan_multiple(ip, ports)

— Приймає IP-адресу та верхню межу портів.

— Створює список портів у заданому діапазоні.

— Викликає асинхронне сканування кожного порту.

— Збирає відкриті порти у список.

3. scan_one(ip, port)

— Сканує лише один порт для вказаного IP.

— Застосовується, якщо користувач обрав точкове сканування.

4. scan_port(ip, port)

Основна функція сканування.

— Працює асинхронно через asyncio.open_connection.

— Якщо вдається підключитися — повертає номер порту.

— Інакше — повертає None.

5. lookup_port_info(port)

— Функція звертається до файлу ports.json.

— Знаходить відповідність: номер порту → служба → опис.

— Повертає результат для подальшого виводу.

6. Вивід у консоль

— За допомогою termcolor дані виводяться кольорово.

— Це дозволяє зручно візуально відрізнити відкриті порти від інших.

Архітектура побудована за принципом централізованого керування, коли головна функція main() викликає інші компоненти за потреби, залежно від обраного режиму користувачем.

Основні етапи взаємодії:

— Користувач запускає програму → main();

— main() обробляє введення → вибирає один чи кілька IP / портів;

— Для кожного IP-адреса викликається scan_multiple() або scan_one();

— Кожен порт передається до scan_port();

— Відкриті порти передаються у lookup_port_info() для отримання назви служби та опису;

— Результати виводяться з форматуванням через termcolor.

Архітектура програми спочатку передбачає роботу з кількома IP-адресами, які користувач може ввести, розділивши їх комами. Логіка побудована так, що:

— кожна IP-адреса обробляється окремо;

— результати для кожного хоста виводяться незалежно, щоб уникнути плутанини;

— виконання побудовано через asyncio.gather() — тобто IP скануються паралельно.

Це дає змогу швидко перевіряти невеликі локальні мережі або кілька віддалених серверів без потреби запускати програму кілька разів.

Архітектура застосунку побудована з урахуванням простоти, модульності та ефективності. Кожен компонент виконує окрему роль, усі елементи взаємодіють логічно, підтримується обробка кількох IP-адрес одночасно. Завдяки такому підходу, застосунок не лише легко підтримувати, а й розвивати, адаптуючи до нових завдань у сфері кібербезпеки та мережевого адміністрування.

Однією з ключових вимог до якісного програмного забезпечення є зрозуміла, передбачувана і чітка взаємодія між його модулями. У застосунку логіка побудована так, що кожна функція (модуль) виконує свою окрему задачу, а вся програма працює завдяки координації між цими частинами. Це дозволяє легко змінювати одну частину коду, не порушуючи інші, а також швидко розширювати програму.

Як дані переходять від введення до результату:

1. Користувач вводить IP та порт(и) → ці значення зберігаються в змінних у main();
2. Залежно від вибору режиму (один або кілька портів) запускається відповідна функція;
3. Вона передає порти в scan_port() для перевірки;
4. Успішні підключення повертають список відкритих портів;

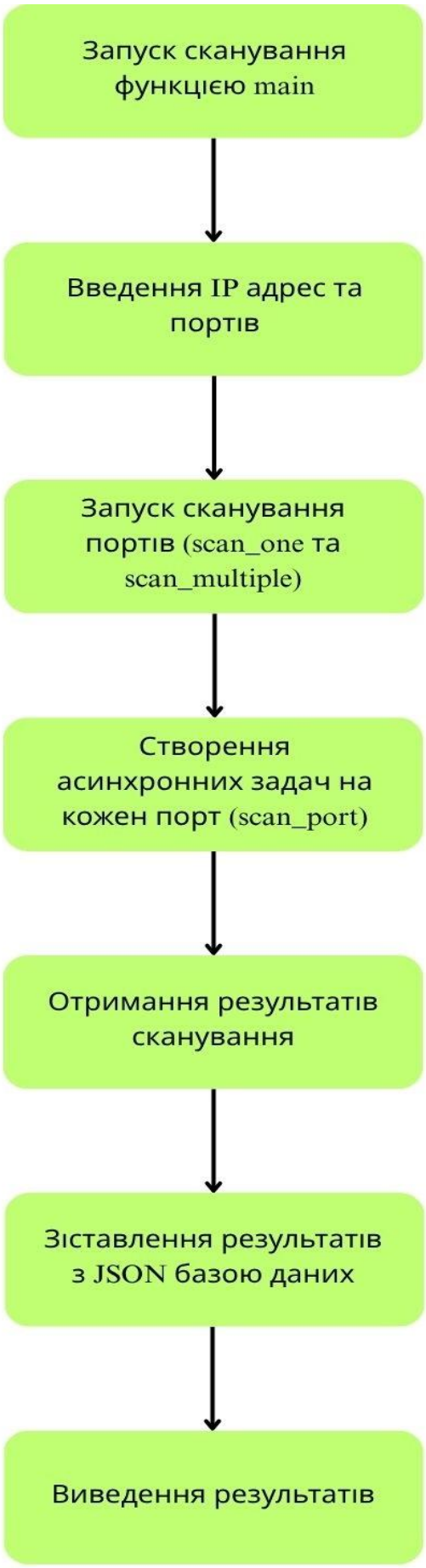


Рисунок 1.8 Взаємодія модулів програми

1.7 Створення JSON бази даних

Для створення локальної бази даних у вигляді JSON файлу було використано парсер, який був розроблений спеціально для цілей дипломного проєкту. Парсер - це програмний інструмент, розроблений для автоматичного збору структурованих даних із вебсторінок. У межах даної дипломної роботи парсер використовується для отримання інформації про мережеві порти з відкритого джерела - веб сторінки IANA.

Принцип роботи парсера:

1. Завантаження HTML-сторінки

   За допомогою HTTP-запиту програма отримує HTML-код вебсторінки, яка містить таблиці з інформацією про порти.

2. Обробка HTML-коду

   Код вебсторінки розбирається за допомогою бібліотеки BeautifulSoup, яка дозволяє знаходити й витягувати таблиці з даними.

3. Виділення таблиць з портами

   Парсер знаходить всі HTML-елементи з класом wikitable, які містять необхідну інформацію про порти.

4. Обхід рядків таблиці

   Кожен рядок таблиці розглядається окремо. З нього виділяються наступні поля:

   - Номер порту або діапазон портів (наприклад, 20–21, 443)
   - Опис сервісу
   - Протокол (TCP/UDP)
   - Примітки (додаткова інформація)

5. Обробка діапазонів портів

   Якщо вказано діапазон (наприклад, 6000–6063), парсер розгортає його до окремих записів для кожного порту.

6. Фільтрація та уникнення дублікатів

   Для унікальності створюється ключ із порту та протоколу. Це дозволяє уникнути повторів.

7. Збереження результатів

Витягнута інформація зберігається у форматі .json, що дозволяє легко використовувати її в інших програмних компонентах або аналізувати.

Розробка парсера веб сторінок була реалізована на мові програмування Python, і цьому є низка технічних та практичних причин:

1. Бібліотека BeautifulSoup - зручність у роботі з HTML

Python має просту та потужну бібліотеку BeautifulSoup, яка спеціалізується на парсингу HTML-коду. Вона дозволяє обробляти будь який HTML. Її API легко читається та не потребує складної конфігурації.

2. Бібліотека requests - простий HTTP-запит в один рядок

Мова Python надає інструменти для надсилання HTTP-запитів через бібліотеку requests. Вона дозволяє отримати HTML-вміст за один рядок коду, не вимагаючи додаткових налаштувань (як у Java або C++). Це значно пришвидшує процес розробки.

3. Швидкий старт і мінімальний поріг входження

Python - це високорівнева мова з простою синтаксичною структурою. Написати парсер на Python можна набагато швидше, ніж на інших популярних мовах. Це особливо важливо, де цінується швидкий результат і гнучкість.

4. Гнучка обробка тексту

Python має вбудовану підтримку для регулярних виразів (бібліотека re) та зручну роботу зі рядками, що критично важливо при парсингу даних, де потрібно виділяти окремі фрагменти тексту або числа з HTML.

5. Наявність великої спільноти та документації

Python має широку спільноту розробників, що активно створює документацію, навчальні матеріали та бібліотеки. Це дозволяє швидко знаходити рішення проблем або приклади реалізації.

6. Просте збереження результатів у форматі JSON

Python має вбудований модуль json, який дозволяє швидко і без додаткових залежностей серіалізувати (тобто перетворювати) дані у формат JSON. Це зручно для подальшої обробки, візуалізації або експорту до баз даних.

7. Легкість тестування та налагодження

У Python легко додавати логування, повідомлення про помилки, або запускати парсер частинами - що дозволяє зручно тестувати і контролювати процес парсингу.

1.8 Приклади роботи програмного застосунку

Щоб підтвердити працездатність та практичну ефективність розробленого застосунку, у цьому розділі наведено приклади реального запуску програми з різними вхідними даними. Такі приклади ілюструють, як саме відбувається процес сканування, які повідомлення виводяться на екран, та яким чином подається результат. Крім того, буде виконано короткий аналіз результатів для демонстрації реальної корисності інструменту.

Сценарій 1: Сканування одного порту на одній IP-адресі

```
C:\Users\minik\Desktop\diplom>py diplom.py
[*] Enter targets to scan (split them by ,): 127.0.0.1
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): one
[*] Enter port to scan: 135

[!] Starting scan for 127.0.0.1
Scanned port 135: epmap - DCE endpoint resolution
[+] Open port for 127.0.0.1:
    Port 135: epmap - DCE endpoint resolution
[√] Finished scanning 127.0.0.1
------------------------------------------------------------
```

Рисунок 1.9. Приклад роботи програми при скануванні порта

Сценарій 2: Сканування діапазону портів на одній IP-адресі

```
C:\Users\minik\Desktop\diplom>py diplom.py
[*] Enter targets to scan (split them by ,): 127.0.0.1
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): multiple
[*] Enter start port: 1
[*] Enter end port: 10000

[!] Starting scan for 127.0.0.1
Scanned port 135: epmap - DCE endpoint resolution
Scanned port 445: microsoft-ds - Microsoft-DS
Scanned port 5040: unknown - No description
Scanned port 6463: unknown - No description
Scanned port 9180: unknown - No description
[+] Open ports for 127.0.0.1:
    Port 135: epmap - DCE endpoint resolution
    Port 445: microsoft-ds - Microsoft-DS
    Port 5040: unknown - No description
    Port 6463: unknown - No description
    Port 9180: unknown - No description
[√] Finished scanning 127.0.0.1
```

Рисунок 1.10. Приклад роботи програми при скануванні діапазону портів

Сценарій 3: Сканування одного порту на кількох IP-адресах

```
C:\Users\minik\Desktop\diplom>py diplom.py
[*] Enter targets to scan (split them by ,): 127.0.0.1, 192.168.0.1
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): one
[*] Enter port to scan: 443
[*] Scanning multiple targets

[!] Starting scan for 127.0.0.1

[!] Starting scan for 192.168.0.1
Scanned port 443: https - http protocol over TLS/SSL
[+] Open port for 192.168.0.1:
    Port 443: https - http protocol over TLS/SSL
[✓] Finished scanning 192.168.0.1
------------------------------------------------------------
[-] Port 443 is closed for 127.0.0.1
[✓] Finished scanning 127.0.0.1
------------------------------------------------------------
```

Рисунок 1.11. Приклад роботи програми при скануванні одного порта

Сценарій 4: Сканування діапозону портів на кількох IP-адресах

```
C:\Users\minik\Desktop\diplom>py diplom.py
[*] Enter targets to scan (split them by ,): 127.0.0.1, 192.168.0.1
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): multiple
[*] Enter start port: 1
[*] Enter end port: 500
[*] Scanning multiple targets

[!] Starting scan for 127.0.0.1

[!] Starting scan for 192.168.0.1
Scanned port 135: epmap - DCE endpoint resolution
Scanned port 53: domain - Domain Name Server
Scanned port 80: http - World Wide Web HTTP
Scanned port 445: microsoft-ds - Microsoft-DS
Scanned port 443: https - http protocol over TLS/SSL
[+] Open ports for 127.0.0.1:
    Port 135: epmap - DCE endpoint resolution
    Port 445: microsoft-ds - Microsoft-DS
[✓] Finished scanning 127.0.0.1
------------------------------------------------------------
[+] Open ports for 192.168.0.1:
    Port 53: domain - Domain Name Server
    Port 80: http - World Wide Web HTTP
    Port 443: https - http protocol over TLS/SSL
[✓] Finished scanning 192.168.0.1
------------------------------------------------------------
```

Рисунок 1.12. Приклад роботи програми при скануванні діапазону портів

Сценарій 5. Користувач увів не правильну адресу

```
C:\Users\minik\Desktop\diplom>py diplom.py
[*] Enter targets to scan (split them by ,): abcd
[!] One or more IP addresses are invalid. Please try again.
[*] Enter targets to scan (split them by ,):
```

Рисунок 1.13. Приклад роботи програми, якщо не правильно введено IP-адрес

Сценарій 6. Користувач вибирає неправильний режим сканування

```
[*] Enter targets to scan (split them by ,): 127.0.0.1
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): abc
[!] Error: Please enter 'one' or 'multiple':
[*] Do you want to scan one port or multiple ports? (write one or multiple to select):
```

Рисунок 1.14. Приклад роботи програми при виборі неправильного режиму сканування

1.9 Порівняння результатів з іншими сканерами

Щоб оцінити ефективність розробленого застосунку для сканування портів, було проведено тестове порівняння його роботи з одним із раніше згаданих програмних застосунків - Nmap. Метою є перевірка точності, швидкості виконання та зручності інтерфейсу обох рішень.

Тестування №1. Було проведено сканування всіх портів по IP-адресі роутера. Було виявлено 5 відкритих портів, час сканування становить 48,70 секунд.

```
C:\Users\minik>nmap -p 1-65535 -T4 -A -v 192.168.0.1
Starting Nmap 7.97 ( https://nmap.org ) at 2025-06-05 11:20 +0300
NSE: Loaded 158 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 11:20
Completed NSE at 11:20, 0.00s elapsed
Initiating NSE at 11:20
Completed NSE at 11:20, 0.00s elapsed
Initiating NSE at 11:20
Completed NSE at 11:20, 0.00s elapsed
Initiating ARP Ping Scan at 11:20
Scanning 192.168.0.1 [1 port]
Completed ARP Ping Scan at 11:20, 0.11s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:20
Completed Parallel DNS resolution of 1 host. at 11:20, 0.50s elapsed
Initiating SYN Stealth Scan at 11:20
Scanning 192.168.0.1 [65535 ports]
Discovered open port 80/tcp on 192.168.0.1
Discovered open port 443/tcp on 192.168.0.1
Discovered open port 53/tcp on 192.168.0.1
Discovered open port 20001/tcp on 192.168.0.1
SYN Stealth Scan Timing: About 50.60% done; ETC: 11:21 (0:00:30 remaining)
Discovered open port 1900/tcp on 192.168.0.1
Completed SYN Stealth Scan at 11:21, 69.89s elapsed (65535 total ports)
```

Рисунок 1.15. Звіт після сканування всіх портів на одній IP-адресі

Тестування №2. Було проведено сканування всіх портів на декількох IP-адресах (IP-адреси локального хоста та роутера). На IP-адресі роутера було

виявлено 5 відкритих портів на IP-адресі роутера, час сканування становить 52,36 секунд. На IP-адресі локального хоста було виявлено 19 відкритих портів на IP-адресі роутера, час сканування становить 6,53 секунд.

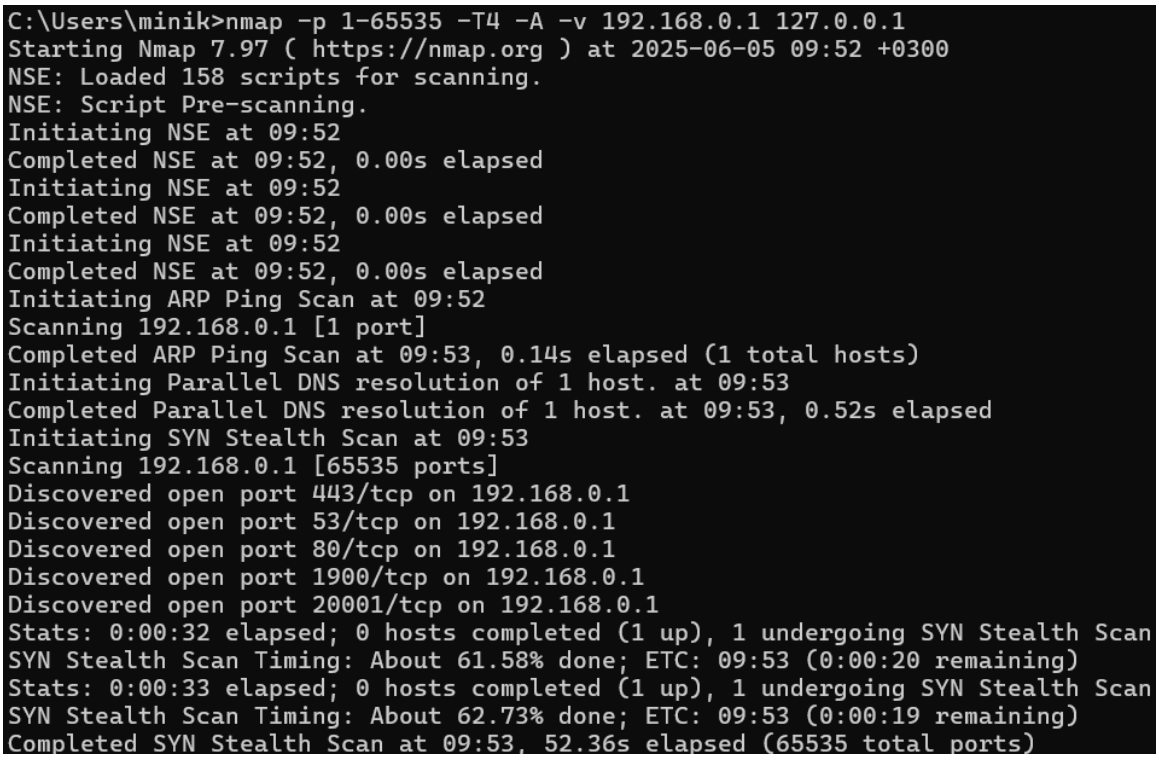


Рисунок 1.16. Звіт після сканування всіх портів на IP-адресі роутера

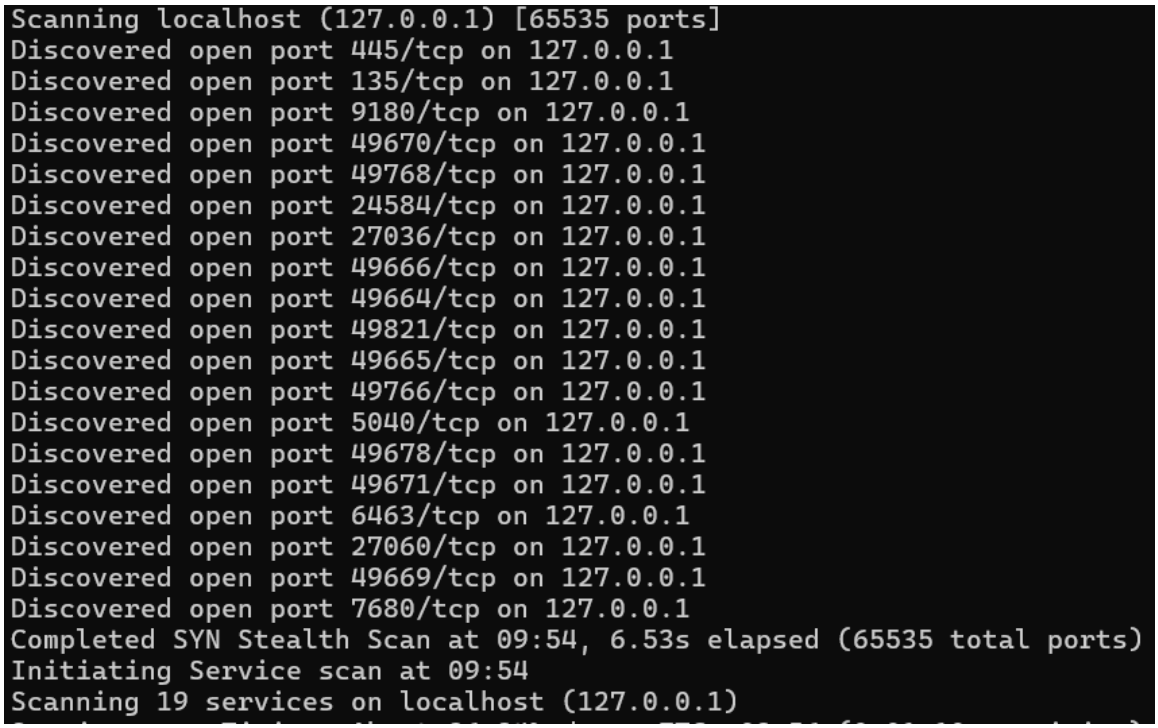


Рисунок 1.17. Звіт після сканування всіх портів на IP-адресі локального хоста

Тестування №3. Було проведено сканування одного порта на одній IP-адресі локального хоста. Було виявлено відкритий порт, час сканування становить 6,21 секунд.

Рисунок 1.18. Звіт після сканування одного порта

Таблиця 1.2. Результати сканування у Nmap

Цілі сканування та режим сканування	Число відкритих портів	Час сканування
Всі порти на одній IP-адресі	5 відкритих портів	48,70 секунд
Всі порти на декількох IP-адресах	5 відкритих портів на одній адресі та 19 на другій адресі	52,36 секунд на одну адресу та 6,53 секунд на другу адресі
Один порт на одній IP-адресі	1 відкритий порт	6,21 секунд

Тестові сканування у розробленому програмному застосунку

Тестування №1. Було проведено сканування всіх портів по IP-адресі роутера. Було виявлено 5 відкритих портів, час сканування становить 133,65 секунд.

Рисунок 1.19. Звіт після сканування всіх портів

Тестування №2. Було проведено сканування всіх портів на декількох IP-адресах (IP-адреси локального хоста та роутера). На IP-адресі роутера було виявлено 5 відкритих портів на IP-адресі роутера. На IP-адресі локального хоста було виявлено 19 відкритих портів на IP-адресі роутера. Час сканування обох адрес становить 136.05 секунд.

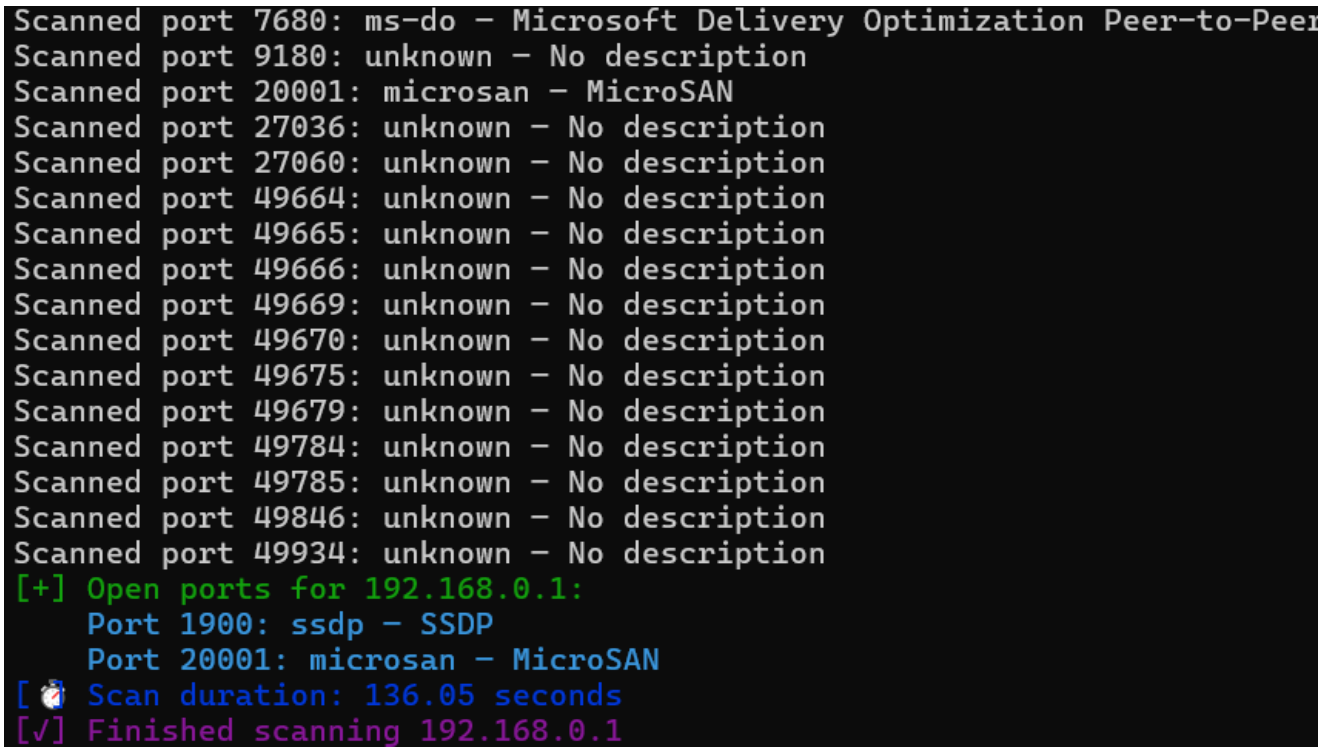


Рисунок 1.20. Звіт після сканування всіх портів

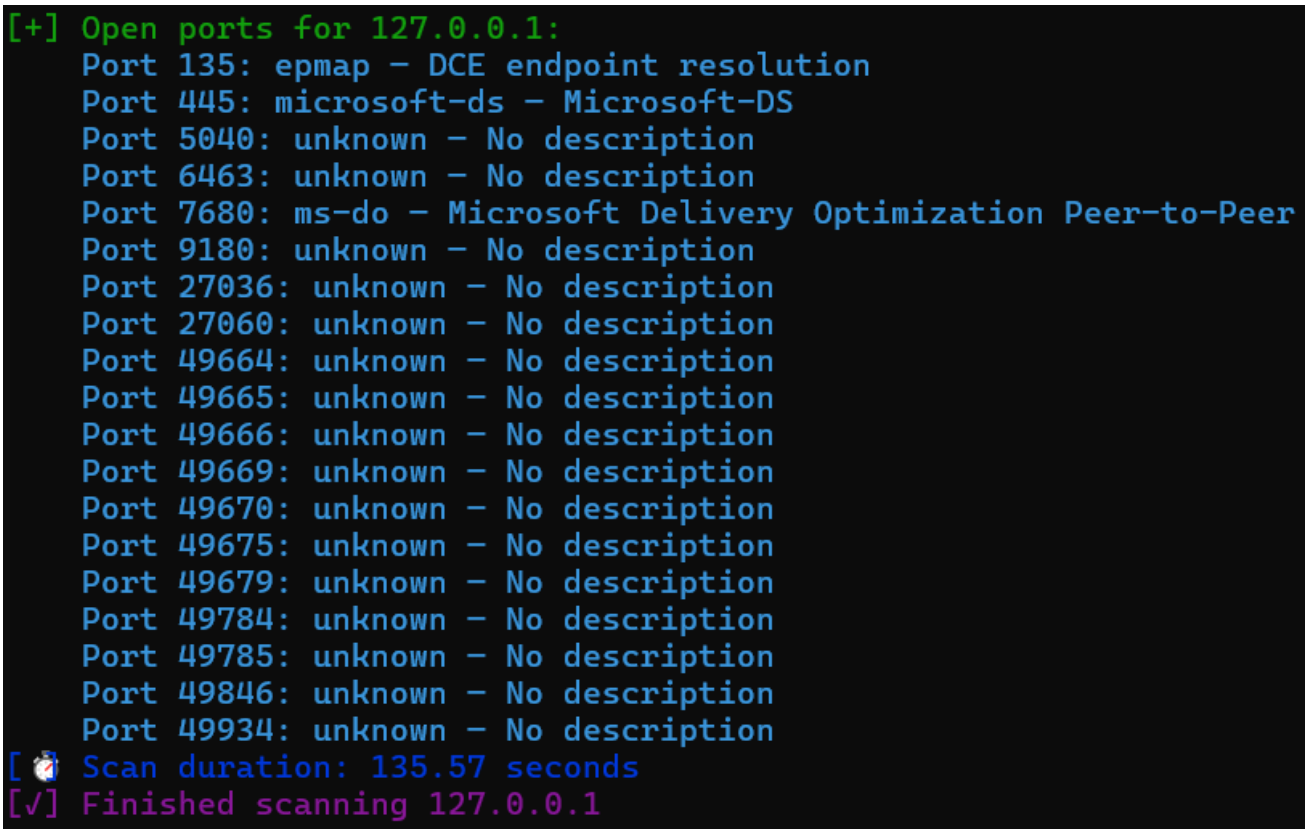


Рисунок 1.21. Звіт після сканування всіх портів

Тестування №3. Було проведено сканування одного порта на одній IP-адресі локального хоста. Було виявлено відкритий порт, час сканування становить менше секунди.

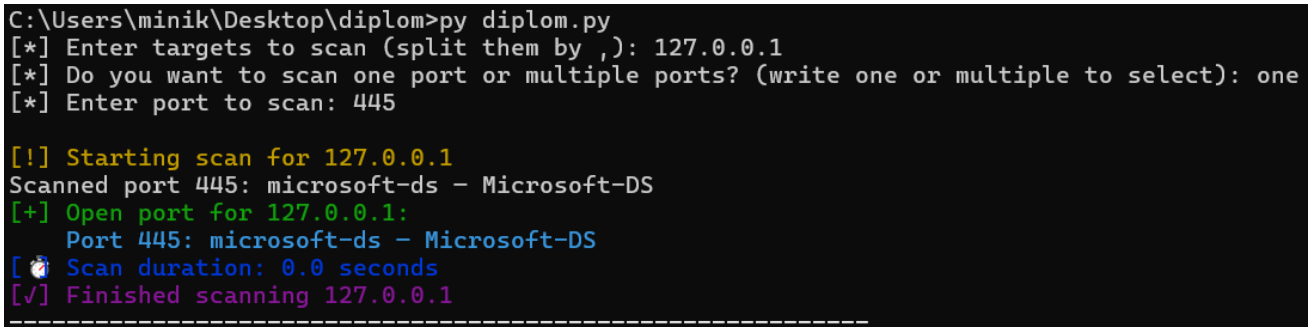


Рисунок 1.22. Звіт після сканування одного порта

Таблиця 1.3. Результати сканування у розробленій програмі

Цілі сканування та режим сканування	Число відкритих портів	Час сканування
Всі порти на одній IP-адресі	5 відкритих портів	133,65 секунд

Продовження таблиці 1.3.

Всі порти на декількох IP-адресах	5 відкритих портів на одній адресі та 19 на другій адресі	136,05 секунд
Один порт на одній IP-адресі	1 відкритий порт	менше секунди

Щоб оцінити ефективність та практичну цінність результатів та особливостей розробленого програмного застосунку та Nmap, доцільно порівняти їх у таблиці. Це дозволяє виявити як сильні сторони, так і потенційні обмеження власної реалізації, а також надати об'єктивну оцінку її точності, швидкодії та зручності.

Таблиця 1.4. Порівняння результатів програмного застосунку та Nmap

Критерії	Розроблена програма	Nmap
Швидкість	Висока	Дуже висока
Точність виявлення портів	Висока	Висока
Виявлення служб	Часткове	Автоматичне, з аналізом
Гнучкість налаштувань	Мінімальна	Висока
Порог входу для використання	Низький	Високий
Підсумковий звіт	Вивід з кольорами	Расширений, але тяжко розібратися

1.10 Тестування програмного застосунку на інших ОС

Усі тестування програмного застосунку проходили на платформі Windows, але він також може працювати на інших платформах, якщо буде встановлен інтерпритатор Python 3.10 чи вище, а також бібліотеки asyncio та termcolor.

На macOS застосунок запускався, все працювало, окрім того що він не знаходив відкритих портів. Скоріше за все, через особливості операційної системи

запити на з’єднання не відправлялися чи не надходили відповіді з портів на обробку.

З платформою Linux все інакше. Для тестування використовувалась Linux Ubuntu 24.04.2 LTS. Перед початком були встановлені бібліотеки asyncio та termcolor. Тестові сканування були проведені у двох режимах на адресі локального хоста.

```
(myenv) ubuntu@ubuntu-VirtualBox:~/diplom$ python3 diplom.py
[*] Enter targets to scan (split them by ,): 109.122.15.104
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): one
[*] Enter port to scan: 53

[!] Starting scan for 109.122.15.104
Scanned port 53: domain — Domain Name Server
[+] Open port for 109.122.15.104:
    Port 53: domain — Domain Name Server
[⏱] Scan duration: 0.01 seconds
[✓] Finished scanning 109.122.15.104
------------------------------------------------------------
```

Рисунок 1.23. Звіт сканування одного порта на Linux

```
(myenv) ubuntu@ubuntu-VirtualBox:~/diplom$ python3 diplom.py
[*] Enter targets to scan (split them by ,): 109.122.15.104
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): multiple
[*] Enter start port: 1
[*] Enter end port: 65535

[!] Starting scan for 109.122.15.104
Scanned port 53: domain — Domain Name Server
Scanned port 1900: ssdp — SSDP
Scanned port 20001: microsan — MicroSAN
[+] Open ports for 109.122.15.104:
    Port 53: domain — Domain Name Server
    Port 1900: ssdp — SSDP
    Port 20001: microsan — MicroSAN
[⏱] Scan duration: 133.95 seconds
[✓] Finished scanning 109.122.15.104
------------------------------------------------------------
```

Рисунок 1.24. Звіт сканування діапазону портів на Linux

1.11 Пояснення користуванням застосунку

Цей розділ присвячений поясненню як інструкція користуванням сканером портів. Напочатку треба запустити командний рядок (cmd), та перейти в дерикторію, де знаходиться папка з програмою командою cd. Запуск самої програми запускається командою py diplom.py.

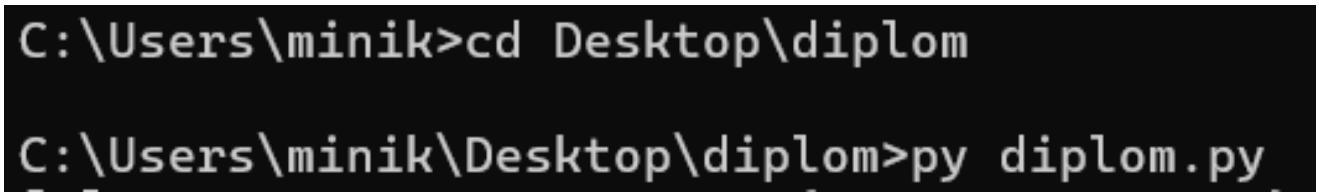


Рисунок 1.25. Перехід в дерикторію знаходження програми та її запуск

Далі у наступній строчці командного рядка треба ввести IP-адреси цілей, котрі будуть скануватися. Якщо буде скануватися більше однієї IP-адреси, то треба вводити їх через кому.

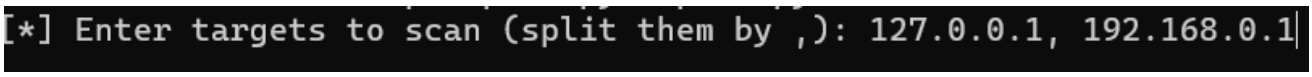


Рисунок 1.26. Введення IP-адрес цілей для сканування

Наступним кроком йде вибір режиму сканування, сканувати один порт або діапазон портів. Щоб вибрати режим сканування одного порта треба ввести one, а для сканування діапазону multiple.

Рисунок 1.27. Вибір методу сканування

Якщо ж був обран режим сканування одного порта, то наступною строчкою в командному рядку треба ввести номер порта для сканування.

Рисунок 1.28. Введення номера порта для сканування

Останнім етапом програма починає сканування, а потім виводить результати сканування.

Але якщо, був обран режим сканування діапазону портів, то програма спочатку просить ввести номер порта з якого почнеться сканування, а наступним номер того порта на котрому закінчиться сканування.

Рисунок 1.29. Введення даних для сканування діапазону портів

Далі як і раніше було сказано, програма почне сканування та виведе

результати сканування.

1.12 Методи захисту мереж

Закриття невикористовуваних портів

Один з найпростіших, але ефективних методів - закрити всі порти, які не використовуються. У типових мережах часто залишаються активними:

- службові порти;
- тестові інтерфейси;
- застарілі служби, про які адміністратор може навіть не знати.

Закриття портів передбачає:

- відключення непотрібних служб (наприклад, FTP, Telnet);
- вимкнення портів у налаштуваннях ОС або застосування правил брандмауера;
- моніторинг відкритих портів через сканери (включно з твоїм застосунком).

Переваги закриття невикористовуваних портів:

- Обмеження кількості доступних зовнішніх точок — зменшується кількість відкритих портів, через які потенційно може здійснюватися несанкціонований доступ.
- Підвищення рівня контролю — у мережі залишаються доступними лише ті служби, які дійсно необхідні для роботи.
- Зменшення кількості вразливих компонентів — вимкнення зайвих служб знижує ризики, пов’язані з їх потенційними уразливостями.

Використання брандмауерів

Брандмауер — це програмний застосунок, яке контролює всі вхідні та вихідні мережеві з’єднання, фільтруючи їх за заданими правилами. Це один з базових компонентів системи захисту.

Основні функції:

— Блокування підозрілих IP-адрес;

— Відсікання доступу до закритих портів;

— Фільтрація протоколів, портів, напрямків трафіку.

Переваги використання брандмауера:

— Можливість налаштування детальних правил фільтрації — адміністратор може точно визначати, який трафік дозволений або заборонений;

— Повноцінний контроль мережевого трафіку — забезпечується управління як вхідними, так і вихідними з’єднаннями на основі політик безпеки;

— Моніторинг та журналювання підозрілої активності — система фіксує спроби несанкціонованого доступу, що дозволяє оперативно реагувати на загрози.

IDS/IPS-системи (системи виявлення та запобігання вторгнень)

IDS (Intrusion Detection System) — система виявлення вторгнень, яка аналізує мережевий трафік і шукає підозрілі дії: спроби сканування, атак, несанкціонованих змін.

IPS (Intrusion Prevention System) — розширення IDS, яке не тільки виявляє, а й автоматично блокує вторгнення.

Як вони працюють:

— отримують копії трафіку з мережі;

— аналізують вміст пакетів (наприклад, SYN-сканування, спроби злому);

— в разі виявлення аномалій:

- IDS — сповіщає адміністратора;
- IPS — автоматично блокує джерело.

Приклади рішень:

— Snort;

— Suricata;

— Zeek;

— Cisco Secure IPS.

Переваги використання IDS/IPS-систем:

— Здатність до глибокого аналізу мережевого трафіку — системи розпізнають не лише поверхневу активність, а й структуру пакетів та нетипову поведінку;

— Ефективне виявлення складних та багатоступеневих атак - включаючи спроби сканування портів та експлуатації вразливостей;

— Автоматичне реагування на загрози — IPS-системи можуть самостійно блокувати трафік або IP-адреси в реальному часі без втручання адміністратора.

1.13 Етичні аспекти використання порт-сканера

Сканер портів є корисним інструментом для виявлення відкритих портів та аналізу безпеки мережі, однак його використання пов'язане з певними етичними обмеженнями. Будь-яке сканування чужих мереж або серверів без дозволу власника вважається порушенням норм етики та законодавства. Це може сприйматися як спроба несанкціонованого доступу або підготовка до атаки.

У більшості країн сканування портів без попереднього погодження прирівнюється до несанкціонованого втручання в роботу інформаційних систем. Тому етичне використання порт-сканера можливе лише у випадках, коли:

— власник мережі або серверу дав на це згоду (наприклад, під час проведення тестування безпеки);

— сканування виконується в навчальних цілях у рамках власної лабораторії чи тестового середовища;

— воно проводиться в процесі пентесту відповідно до офіційного договору або угоди.

Також важливо дотримуватися принципів прозорості: у разі сканування корпоративної мережі дії фахівця повинні бути погоджені з керівництвом, а результати тестування мають бути використані виключно для підвищення рівня захищеності.

Таким чином, сканер-портів є лише інструментом, а відповідальність за його використання завжди лежить на користувачі. Етичний підхід передбачає, що основна мета таких дій — підвищення безпеки та запобігання реальним загрозам, а не нанесення шкоди чи порушення прав інших осіб.

1.14 Перспективи розвитку застосунку

Розроблений програмний застосунок виконує базове TCP-сканування портів з можливістю обробки кількох IP-адрес, працює асинхронно та забезпечує зручний текстовий інтерфейс. Однак у перспективі він може бути суттєво

вдосконалений як у плані функціональності, так і у зручності використання. Нижче наведено основні напрямки можливого розвитку проєкту.

1. Додавання підтримки UDP-сканування

На даний момент реалізовано лише TCP-сканування, яке дає точні результати, але не покриває повністю усі можливі служби в мережі. UDP-протокол широко використовується у таких службах, як DNS (порт 53), SNMP (порт 161), DHCP (порт 67/68) тощо.

Переваги реалізації:

— Ширше охоплення портів і служб;

— Більш повна картина стану мережі;

— Підвищення цінності сканера як інструменту безпеки.

UDP-сканування технічно складніше (через відсутність підтвердження з’єднання), але можливо реалізувати на основі socket з ручною обробкою таймаутів.

2. Графічний інтерфейс користувача (GUI)

Інтерфейс командного рядка є легким і універсальним, але може бути складним для недосвідчених користувачів. Реалізація графічного інтерфейсу на основі бібліотек, таких як Tkinter, PyQt або customtkinter, дозволить значно підвищити зручність використання.

Що можна реалізувати в GUI:

— Форму для введення IP, діапазонів портів;

— Кольоровий звіт у вікні з таблицею;

— Можливість збереження результатів;

— Інтерактивну кнопку запуску сканування.

3. Логування результатів сканування

Наразі результати відображаються лише в терміналі. Запровадження логування дозволить:

— Зберігати звіти у файл (.txt, .json, .csv);

— Аналізувати історію сканувань;

— Інтегрувати результати з іншими системами.

Це особливо важливо при постійному моніторингу або проведенні аудитів.

4. Веб-інтерфейс (через Flask або FastAPI)

Створення веб-версії сканера дозволить:

— Запускати сканування з браузера;

— Працювати з будь-якого пристрою в мережі;

— Виводити інтерактивні таблиці та графіки.

Веб-інтерфейс можна реалізувати на основі Python-фреймворків (Flask, FastAPI) і JavaScript для фронтенду. Це також відкриває шлях до віддаленого моніторингу стану мережі.

5. Додавання SYN-сканування

Одним із напрямків розвитку є реалізація SYN-сканування (так званого «напіввідкритого»), яке використовується в професійних інструментах, як-от Nmap. Метод полягає в надсиланні лише SYN-пакета без встановлення повного TCP-з’єднання.

Переваги:

— Працює швидше за звичайне TCP-сканування;

— Менш помітне для цілі;

— Зменшує навантаження на систему.

Обмеження: потребує доступу до сирих сокетів і, як правило, адміністративних прав. Може реалізовуватись через scapy або низькорівневі засоби.

6. Розширення бази даних портів і сервісів

На поточному етапі застосунок використовує файл формату JSON, де зберігається список стандартних портів і відповідних їм сервісів. База містить основну інформацію - номер порту, назву служби та короткий опис. Проте у майбутньому цю базу даних можна значно розширити й зробити динамічною.

Зокрема, можливо:

— Додати нестандартні порти й сервіси, що використовуються специфічними програмами або у корпоративних мережах;

— Інтегрувати оновлення з офіційних джерел (наприклад, IANA або популярних відкритих баз знань про сервіси й порти);

— Додати додаткові атрибути: тип протоколу (TCP, UDP), рівень ризику, тип платформи (Windows, Linux);

— Передбачити можливість автоматичного оновлення бази під час запуску програми або за запитом користувача.

Незважаючи на те, що базова версія застосунку вже виконує основну функцію сканування портів, потенціал для розвитку проєкту значний. Розширення підтримки протоколів, додавання нових типів сканування, покращення інтерфейсу та додавання можливостей зберігання і візуалізації результатів дозволять зробити програму більш професійною та зручною як для технічних спеціалістів, так і для початківців.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

У рамках цього дипломного проєкту розроблено програмний застосунок для сканування мережевих портів. Програма створена з урахуванням сучасних підходів до розробки, що забезпечує її ефективність та зручність використання. Якість роботи сканера визначається точністю, надійністю та відповідністю функціональним вимогам при оптимальному використанні ресурсів. У розділі розглянуто економічний аспект розробки: витрати часу, зусиль та ресурсів, необхідних для створення продукту.

2.2 Розрахунок ціни програмного продукту нормативним методом

2.2.1 Визначення трудомісткості розробки програмного забезпечення

Час розробки програмного забезпечення залежить від складності, обсягу робіт, трудомісткості та кваліфікації розробників. Для оцінки масштабів проєкту часто використовують метод структурного порівняння, що дозволяє приблизно визначити обсяг коду за аналогами.

Найбільш відповідаючий у більшому ступені до функціоналу застосунку є ПП автоматизації засобів по каталогу (680-7000). На основі обраного аналогу, для якого відомий орієнтовний обсяг коду V_0 (3.84) в умовних машинних командах, далі виконується розрахунок трудомісткості розробки. Обсяг ПП становить 4.00 тис. умов. Машинних команд із нормою час 283 люд/год.

Виходячи з отриманого значення, встановлюється норма часу на створення аналогічного програмного забезпечення. З урахуванням умов реалізації проєкту, це значення коригується відповідним коефіцієнтом (Кк = 0,7÷0,8). Таким чином, розрахункова трудомісткість становить: $T^a_p = 283 \times 0,7 = 198,1$ люд./год.

Загальна трудомісткість розраховується окремо для кожного етапу розробки, базуючись на трудовитратах аналогічного програмного забезпечення з урахуванням складності реалізації, рівня новизни та ступеня використання

типових програмних компонентів:

— Розробка технічного завдання $T_{ТЗ} = T^{a}p \times L_1 \times K_Н$

— Розробка технічного проєкту $T_{ТП} = T^{a}p \times L_2 \times K_Н$

— Розробка робочого проєкту $T_{РП} = T^{a}p \times L_3 \times K_Н \times K_Т$

Для розрахунку необхідні наступні коефіцієнти:

Li - питома вага і-го етапу розробки (табл. 2.3);

Кн - поправочний коефіцієнт, що враховує ступінь новизни (табл.2.4);

Кт - поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (табл. 2.5).

Обрані варіанти в таблицях виділено жовтим кольором.

Таблиця 2.3 Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ(L_1)	0,15	0,12	0,12
ТП(L2)	0,16	0,15	0,11
РП(L3)	0,55	0,58	0,61

Таблиця 2.4 Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75-1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0-0,8
В	ПП маючий аналог	0,7

Таблиця 2.5 Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	Значення $К_т$
40-60	0,7

Розрахунок трудомісткості по етапах:

— Трудомісткість ТЗ $T_{тз}=T_а*L_1*К_н$=198,1 *0,12*0,7= 16,64

— Трудомісткість розробки ТП $T_{тп}=T_а*L_2*К_н$ = 198,1*0,11*0,7 = 15,25

— Трудомісткість розробки РП $T_{рп}=T_а*L_3*К_н$=198,1* 0,61*0,7*0,7= 59,21

Для подальших обчислень було встановлено обсяг документації, підготовленої на кожному етапі розробки: технічне завдання - 2 сторінки (Nтз = 2), технічний проєкт - 50 сторінок (Nтп = 50), робочий проєкт - 8 сторінок (Nрп = 8), пояснювальна записка - 20 сторінок (Nпз = 20).

Таблиця 2.6 Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1.ТЗ	$T_{Ртз}$=16,64	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}$= 0,15*2=0,30
2.Розробка ТП	$T_{Ртп}$=15,25	$T_{кк}=0,7*N_{тп}$= 0,7*50=35	$T_{нк}=0,15*N_{тп}$= 0,15*50=7,5
3.Розробка РП	$T_{Ррп}$=59,21	$T_{кк}=0,7*N_{рп}$= 0,7*8=5,6	$T_{нк}=0,15*N_{рп}$= 0,15*8=1,2
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}$= 1,5*20 =30,0	$T_{кк}=0,7*N_{тз}$= 0,7*20=14,00	$T_{нк}=0,15*N_{пз}$= 0,15*20 =3,00
Усього, в т.ч.:	189,1		
- на розробку	$T_р$= 121,1		
- контроль		$T_{кк}$= 56	
- нормоконтроль			$T_{нк}$= 12

На основі таблиці 2.6 розрахуємо тривалість розробки в роках:

$T_{пп}$=T/(8.0*0.73*360)= (р),

де

8,0 – тривалість робочого дня;

0,73 – коефіцієнт перекладу в календарні дні;

$T_{пп}$=189,1/(8.0*0.73*360)= 0.089 (р)

2.2.2 Розрахунок ціни програмного продукту.

У цьому розділі проводиться обчислення вартості програмного продукту: основної заробітної плати, матеріальних витрат, витрат на машино-години та загальних витрат. Зарплати наведено в таблиці 2.7. З 1 січня 2025 року мінімальна місячна зарплата в Україні становить 8000 грн, погодинна ставка — 46 грн.

Таблиця 2.7 Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, години	Погодинна траифна ставка, грн.	Розрахнуок грн
1. Розробка ПП	121,1	46,00	5570,6
2. Контроль керівника	56	55,00	3080
3. Нормконтроль	12	50,00	600
Усього ($З_о$)			$З_о$=9250,6

Виконаємо обчислення матеріальних витрат, пов'язаних зі створенням програмного продукту. Результати розрахунків подано в таблиці 2.8.

Таблиця 2.8 Розрахунок матеріальних витрат на розробку ПО

Найменування матеріаотних витрат	Тип, модель	Кількість	Ціна одиниці, грн	Вартість, грн
Папір	Лист А4	80	5,00	400
Транспортно-заготівельні Витарти 10%				$B_{тр\_з}$ = 0,1x$B_{м1}$= 0,1*400=40
Усього				$В_м$=$В_{м1}$+$В_{тр\_з}$= 440

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в табл. 2.9.

Таблиця 2.9 Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн	Формула розрахунку
1. Матеріали	440	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	9250,6	$З_о$ (див. табл. 2.7)
3. Додаткова заробітна плата	925,06	$З_д = 0,1 \times З_о = 0,1 \times 9250,6$
4. Відрахування до єдиного фонду соціального внеску	2238,65	$В_{є.с.в.} = 0,22 \times (З_о + З_д) = 0,22 \times (9250,6 + 925,06)$
5. Накладні витрати	5550,36	$В_{нак} = 0,6 \times З_о = 0,6 \times 9250,6$
6. Повна собівартість	18404,67	$С_{пов} = В_м + З_о + З_д + В_{є.с.в.} + В_{нак} = 440 + 9250,6 + 925,06 + 2238,65 + 5550,36$

Розмір прибутку, що включається в ціну, визначається по наступній формулі:

$П=(С_{пов}*Р)/100=(18404,67*10)/100=1840,47$

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о=С_{пов}+П=18404,67+1840,47=20245,14$ грн.

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р= Ц_о+ПДВ =20245,14 +20245,14 *0,2=24294$ грн.

3 РОЗДІЛ ОХОРОНИ ПРАЦИ ТА ТЕХНІКИ БЕЗПЕКИ

У охорони праці дипломної роботи розглянуто питання створення безпечних та комфортних умов праці під час розробки програмного забезпечення. Хоча така діяльність не пов’язана з виробничими шкідливими факторами в звичайному розумінні, але вона все одно потребує дотримання певних вимог щодо організації робочого місця, правильного використання комп’ютера та забезпечення електробезпеки.

Особливу увагу приділено умовам праці при роботі з комп’ютером, оскільки тривале сидіння за монітором і робота з клавіатурою та мишею можуть впливати на стан здоров’я користувача. Треба пам’ятати про фактори, які можуть викликати перевтому очей, напруження опорно-рухового апарату, а також загальне зниження працездатності. Тому розглядання питань охорони праці є важливою частиною проєктування програмного продукту й організації діяльності розробника.

Також у розділі розглядаються основні вимоги до мікроклімату в приміщенні, рівнів шуму, освітлення, а також описані заходи з пожежної та електричної безпеки. Це дуже важливо, оскільки навіть звичайне робоче місце розробника має відповідати нормам, встановленим законодавством і стандартами з охорони праці.

3.1 Аналіз шкідливих та небезпечних факторів, які впливають на розробника

Розробник програмного забезпечення під час роботи за комп’ютером піддається впливу кількох шкідливих факторів. Насамперед це навантаження на зір, що виникає через тривалу роботу перед монітором і може призводити до втоми очей та зниження гостроти зору.

Ще одним фактором є довге перебування в одній позі, що викликає напруження м’язів спини, шиї та рук. Це може призвести до дискомфорту та проблем з опорно-руховим апаратом, якщо не робити перерви та не дотримуватись правильної організації робочого місця.

Також важливо дотримуватись правил електробезпеки, оскільки робота з

комп’ютерною технікою завжди пов’язана з певними ризиками ураження струмом при несправності обладнання.

Додатково впливають умови мікроклімату: температура, вологість і шум. Якщо ці параметри не відповідають нормам, це знижує працездатність і негативно впливає на самопочуття.

3.2 Вимоги з гігієни до приміщення

3.2.1 Вимоги до приміщення

Приміщення для роботи розробника програмного забезпечення має відповідати встановленим санітарним та гігієнічним нормам для забезпечення комфортних і безпечних умов праці. Основні вимоги включають:

— Площа приміщення - не менше 6 м² на одного працівника, щоб забезпечити достатній простір для розміщення обладнання та вільного пересування.

— Висота стелі - не менше 3.2 метрів для нормальної циркуляції повітря.

— Передбачення вентиляції для забезпечення притоку свіжого повітря та видалення надлишків тепла, що виділяється комп’ютерною технікою.

3.2.2 Вимоги до шуму в приміщенні

Під час роботи за комп’ютером шум не основний шкідливий фактор, але його рівень все одно впливає на самопочуття й працездатність. Джерелом шуму може бути системні блоки, вентилятори, принтери, і т.д. Тривала дія високого рівня шуму викликає втому, знижує концентрацію й продуктивність праці.

Відповідно до санітарних норм, допустимий рівень шуму на робочому місці користувача ПК не повинен перевищувати 50 дБ. Цей рівень вважається безпечним і комфортним для тривалої роботи.

Щоб знизити вплив шуму, рекомендується використовувати комп’ютерне обладнання з низьким рівнем шуму, розташовувати його на відстані від робочого місця або використовувати шумопоглинальні матеріали в приміщенні.

3.2.3 Вимоги до освітлення в приміщенні

Освітлення робочого місця має важливе значення для збереження зору та підтримки комфортних умов праці. Недостатня або надмірна яскравість призводить до втоми очей, зниження продуктивності й погіршення самопочуття.

Відповідно до норм, рівень освітленості на робочій поверхні для роботи за комп’ютером повинен бути не менше 300 люксів. Освітлення має бути рівномірним, без різких тіней і бликів на моніторі.

Рекомендується поєднувати природне і штучне освітлення. При штучному освітленні слід використовувати люмінесцентні або LED-світильники з нейтральним відтінком світла (близько 4000-5000 К). Робоче місце потрібно розташовувати так, щоб уникати прямих сонячних променів на екран монітора.

3.2.4 Вимоги з електробезпеки

Робота розробника програмного забезпечення пов’язана з постійним використанням електричних приладів, зокрема комп’ютера, монітора, периферійного обладнання. Тому важливо дотримуватись правил електробезпеки для запобігання ураження електричним струмом та інших небезпечних ситуацій.

Основні вимоги:

- уся техніка повинна бути підключена до справних електромереж із заземленням;
- використовувати справні подовжувачі та розетки без механічних пошкоджень;
- не допускається прокладання кабелів і проводів через прохідні місця, де можливе їх пошкодження;
- у разі виявлення несправності обладнання його слід негайно вимкнути та повідомити відповідального за електрогосподарство;
- у приміщенні мають бути доступні засоби пожежогасіння, призначені для гасіння електрообладнання (наприклад, вогнегасник з вуглекислотою).
- Електрообладнання має бути справним і регулярно перевірятися на відсутність пошкоджень. Кабелі, розетки й подовжувачі повинні бути цілими, без ознак оплавлення або тріщин.
- Не допускається перевантаження електромережі, наприклад підключення великої кількості пристроїв до однієї розетки без відповідного розрахунку навантаження.

— Слід уникати прокладання кабелів і дротів у місцях, де можливе їх механічне пошкодження (під дверима, під ногами тощо).

3.2.5 Вимоги до мікроклімату

Під час роботи за комп'ютером важливо забезпечити комфортний мікроклімат, оскільки він впливає на самопочуття й працездатність людини. Основні вимоги стосуються температури, вологості та швидкості руху повітря.

Рекомендовані вимоги до мікроклімату:

— температура повітря - 18-24 °C;

— відносна вологість - 40-60 %;

— швидкість руху повітря - не більше 0,1 м/с.

— Приміщення має провітрюватися, а за потреби обладнуватися системами кондиціонування або вентиляції для підтримки зазначених вимогах. Відхилення від норм може призводити до швидкої втоми, зниження концентрації уваги та загального дискомфорту під час роботи.

Передбачення вентиляції для забезпечення притоку свіжого повітря та видалення надлишків тепла, що виділяється комп'ютерною технікою. Основні види вентиляції, які можуть застосовуватись:

— Природна вентиляція — здійснюється завдяки відкриванню вікон, фрамуг, вентиляційних решіток. Використовується там, де допустимий рівень тепловиділення та невелика кількість обладнання.

— Примусова (механічна) вентиляція — система вентиляторів і витяжок, яка забезпечує постійний обмін повітря незалежно від зовнішніх умов. Рекомендується для приміщень із великою кількістю комп'ютерів або потужного серверного обладнання.

— Комбінована вентиляція — поєднання природної та примусової вентиляції. Наприклад, коли більшу частину часу використовується природна вентиляція, але за потреби (літня спека, велике тепловиділення) вмикаються механічні системи.

Такі системи дозволяють уникнути перегріву обладнання, запобігти підвищенню температури та забезпечити комфортні умови для роботи персоналу.

Таблиця 3.1 Нормовані параметри мікроклімату в приміщенні з комп'ютерами

Період року	Температура повітря, ^0C	Відносна вологість, %	Швидкість руху повітря, м/с
Холодний	20-24	40-60	не більше 0,1
Теплий	22-25	40-60	Не більше 0,2

3.3 Пожежна безпека

Організація робочого місця потребує обов'язкового дотримання правил пожежної безпеки. Незважаючи на те, що основна діяльність виконується за комп'ютером, використання електротехнічного обладнання завжди супроводжується певним ризиком займання, особливо у випадку поломки чи порушення правил експлуатації.

Основні вимоги щодо пожежної безпеки:

— Первинні засоби пожежогасіння мають бути в легкодоступному місці. Найбільш доцільно використовувати вуглекислотні або порошкові вогнегасники, оскільки вони підходять для гасіння електроустановок, що перебувають під напругою.

— У приміщенні повинна бути передбачена пожежна сигналізація, а також вказані шляхи евакуації. Працівники повинні бути ознайомлені з планом евакуації й порядком дій у разі пожежі.

— Категорично заборонено використання легкозаймистих матеріалів поблизу комп'ютерної техніки (наприклад, паперових куп чи горючих рідин).

— Керівникам треба періодично проводити інструктажі з пожежної безпеки та перевіряти готовність приміщення до надзвичайних ситуацій.

Дотримання цих заходів дозволяє знизити ризик виникнення пожежі та мінімізувати можливі наслідки.

ВИСНОВКИ

В ході виконання дипломної роботи було створено програмний застосунок для сканування мережевих портів з метою виявлення потенційних вразливостей комп'ютерних мереж. Програма була реалізована на мові програмування Python з використанням бібліотек asyncio, json, time та termcolor. Такий вибір технологій дозволяє забезпечити зручність використання програми та її високу продуктивність під час роботи з великими діапазонами портів і кількома IP-адресами одночасно.

В процесі роботи було виконано детальний аналіз існуючих інструментів для сканування портів. Це дало змогу врахувати переваги та недоліки використання таких програм, як Nmap, Masscan та Netcat. На основі отриманих даних було знайдено найкращий підхід для реалізації програми. Логіка роботи сканера побудована на тому щоб сканування було швидким та стабільним. Для цього було використано асинхронне програмування, воно допомагає в обробці запитів та обмежує кількості одночасних з'єднань, що запобігає перевантаженню операційної системи.

Розроблений застосунок не лише перевіряє статус портів, а й надає користувачу додаткову інформацію про сервіси та служби, що працюють на порті за допомогою локальної бази даних у форматі JSON файлу. Вивід даних у термінал організований таким чином, щоб результати були зроумілими для аналізу при скануванні одної чи кількох адрес. Кольорове виділення відкритих портів спрощує візуальне сприйняття.

Програма показала працездатність під час тестування на прикладах локальних і зовнішніх IP-адрес, а також на діапазонах портів різного розміру. Вона може використовуватись як інструмент для людей, котрі не розбираються в скануванні портів. У майбутньому сканер портів можна вдосконалити шляхом додавання UDP сканування портів, впровадження нових методів прихованого сканування, розширення бази даних портів і сервісів, а також реалізації графічного або веб-інтерфейсу. Усі поставлені завдання дипломної роботи було успішно виконано, а результат підтвердив її ефективність.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Кузьменко С. В. Адміністрування комп'ютерних мереж: навч. посіб. — Харків: ХНУРЕ, 2020. — 284 с.
2. Ковальчук Ю. П., Шевченко Л. І. Основи інформаційної безпеки: навч. посіб. — Київ: НТУУ «КПІ», 2021. — 342 с.
3. Семенов А. П., Іваненко В. С. Мережева безпека та кіберзахист: монографія. — Харків: ХНЕУ, 2023. — 350 с.
4. FORTINET. Що таке сканування портів? [Електронний ресурс]. — Режим доступу: https://www.fortinet.com/resources/cyberglossary/what-is-port-scan (дата звернення: 23.03.2025).
5. NMAP.ORG. Port Scanning Techniques [Електронний ресурс]. — Режим доступу: https://nmap.org/book/man-port-scanning-techniques.html (дата звернення: 25.03.2025).
6. nBook. Що таке сканування портів, як воно працює і як захиститися від кібератак [Електронний ресурс]. — Режим доступу: https://nbookpart.com.ua/scho-take-skanuvannya-portiv-yak-vono-pratsyue-i-yak-zahystytys-vid-kiberatak/ (дата звернення: 23.03.2025).
7. IANA. Service Name and Transport Protocol Port Number Registry [Електронний ресурс]. — Режим доступу: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (дата звернення: 9.05.2025).
8. Python. Official site [Електронний ресурс]. — Режим доступу: https://www.python.org/ (дата звернення: 12.04.2025).
9. OWASP. Open Web Application Security Project [Електронний ресурс]. — Режим доступу: https://owasp.org/ (дата звернення: 25.03.2025).
10. **Wikipedia.** Сканер портів [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BD%D0%B5%D1%80_%D0%BF%D0%BE%D1%80%D1%82%D1%96%D0%B2 (дата звернення: 24.03.2025).

ДОДАТОК А. Код програмного застосунку для сканування мережевих портів мовою Python

```
import termcolor
    import asyncio
    import json
    import ipaddress
    import time  # для вимірювання часу

    # Завантаження сервісів із JSON-файлу
    def load_services(filename="ports_list.json"):
        with open(filename, "r") as file:
            return json.load(file)

    # Перевірка чи правильно введені IP-адреси
    def is_valid_ip(ip):
        try:
            ipaddress.ip_address(ip)
            return True
        except ValueError:
            return False

    # Отримання інформації про порт
    def get_service_info(port, services):
        for entry in services:
            try:
                if int(entry["Port Number"]) == port:
                    name = entry["Service Name"] or "unknown"
                    description = entry["Description"] or "No description"
                    return f"{name} — {description}"
            except:
```

```
            continue
    return "unknown — No description"

# Сканування одного порту
async def scan_port(ipaddress, port, services, semaphore):
    async with semaphore:
        try:
            reader, writer = await asyncio.wait_for(
                asyncio.open_connection(ipaddress, port),
                timeout=2
            )
            writer.close()
            await writer.wait_closed()
            service_info = get_service_info(port, services)
            print(f"Scanned port {port}: {service_info}")
            return (port, service_info)
        except asyncio.TimeoutError:
            return None
        except:
            return None

# Сканування кількох портів з контролем навантаження
async def scan_multiple(target, start_port, end_port, services):
    print(termcolor.colored(f'\n[!] Starting scan for {target}', 'yellow'))

    start_time = time.time()  # Початок часу

    semaphore = asyncio.Semaphore(1000)
    tasks = [
        scan_port(target, port, services, semaphore)
```

```
        for port in range(start_port, end_port + 1)
    ]
    results = await asyncio.gather(*tasks)
    open_ports = [res for res in results if res]

    if open_ports:
        print(termcolor.colored(f"[+] Open ports for {target}:", 'green'))
        for port, info in open_ports:
            print(termcolor.colored(f"    Port {port}: {info}", 'cyan'))
    else:
        print(termcolor.colored(f"[-] No open ports found for {target}", 'red'))

    end_time = time.time()
    duration = round(end_time - start_time, 2)
    print(termcolor.colored(f"[⏱] Scan duration: {duration} seconds", 'blue'))
    print(termcolor.colored(f"[✓] Finished scanning {target}", 'magenta'))
    print(termcolor.colored('-' * 60, 'white'))

# Сканування одного порту
async def scan_one(target, port, services):
    print(termcolor.colored(f'\n[!] Starting scan for {target}', 'yellow'))

    start_time = time.time()

    semaphore = asyncio.Semaphore(1)
    result = await scan_port(target, port, services, semaphore)

    if result:
        port, info = result
        print(termcolor.colored(f"[+] Open port for {target}:", 'green'))
```

```
        print(termcolor.colored(f"    Port {port}: {info}", 'cyan'))
    else:
        print(termcolor.colored(f"[-] Port {port} is closed for {target}", 'red'))

    end_time = time.time()
    duration = round(end_time - start_time, 2)
    print(termcolor.colored(f"[⏱] Scan duration: {duration} seconds", 'blue'))
    print(termcolor.colored(f"[✓] Finished scanning {target}", 'magenta'))
    print(termcolor.colored('-' * 60, 'white'))

# Основна функція
async def main():
    services = load_services()

    while True:
        targets = input("[*] Enter targets to scan (split them by ,): ")
        ip_list = [ip.strip() for ip in targets.split(',')]

        if all(is_valid_ip(ip) for ip in ip_list):
            break
        else:
            print(termcolor.colored("[!] One or more IP addresses are invalid. Please try again.", 'red'))

    while True:
        choice = input("[*] Do you want to scan one port or multiple ports? (write one or multiple to select): ").strip().lower()
        if choice in ('one', 'multiple'):
            break
        print("[!] Error: Please enter 'one' or 'multiple': ")
```

```
    if choice == 'one':
        port = int(input("[*] Enter port to scan: "))
        if ',' in targets:
            print("[*] Scanning multiple targets")
            tasks = [scan_one(ip.strip(), port, services) for ip in targets.split(',')]
            await asyncio.gather(*tasks)
        else:
            await scan_one(targets.strip(), port, services)
    else:
        start_port = int(input("[*] Enter start port: "))
        end_port = int(input("[*] Enter end port: "))

        if start_port < 1 or end_port > 65535 or start_port > end_port:
            print(termcolor.colored("[!] Invalid port range. Please enter values between 1 and 65535.", 'red'))
            return

        if ',' in targets:
            print("[*] Scanning multiple targets")
            tasks = [scan_multiple(ip.strip(), start_port, end_port, services) for ip in targets.split(',')]
            await asyncio.gather(*tasks)
        else:
            await scan_multiple(targets.strip(), start_port, end_port, services)

asyncio.run(main())
```

ДОДАТОК Б. Код парсера сайту IANA мовою Python

```
import csv
import requests
import io
import json

url = "https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.csv"
response = requests.get(url)
response.encoding = 'utf-8'

f = io.StringIO(response.text)
reader = csv.DictReader(f)

ports_dict = {}

for row in reader:
    service_name = row['Service Name'].strip()
    port_number = row['Port Number'].strip()
    protocol = row['Transport Protocol'].strip()
    description = row['Description'].strip()

    if not port_number:
        continue

    # Ключ без урахування протоколу
    key = (service_name, port_number, description)

    # Додаємо тільки якщо ще не додано
    if key not in ports_dict:
```

```
        ports_dict[key] = {
            'Service Name': service_name,
            'Port Number': port_number,
            'Description': description
        }

# Перетворюємо у список
merged_ports = list(ports_dict.values())

# Зберігаємо у JSON
with open("iana_ports_no_protocol.json", "w", encoding="utf-8") as f:
    json.dump(merged_ports, f, ensure_ascii=False, indent=4)

print(f"Збережено {len(merged_ports)} записів без поля Protocol у iana_ports_no_protocol.json")
```

ДОДАТОК В. Слайди мультимедійної презентації

Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей

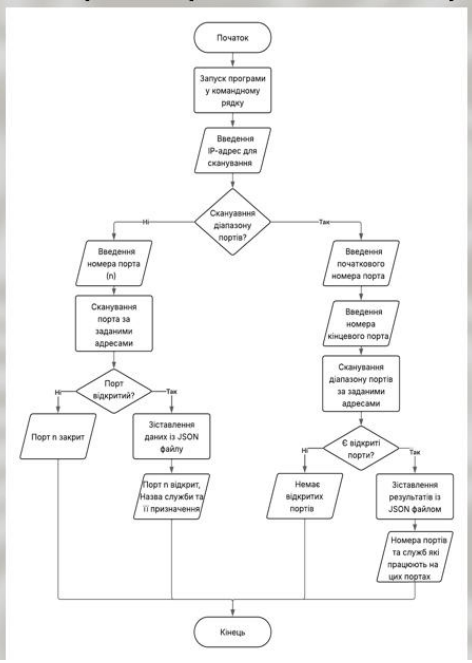
Поліщук Ігор 4КБ-02

Актуальність теми

Вибір методу сканування

Логіка роботи програмного застосунку

Алгоритм роботи сканера



Взаємодія модулів програми

Метод асинхронного програмування

Приклад роботи програми

```
C:\Users\minik\Desktop\diplom>py diplom.py
[*] Enter targets to scan (split them by ,): 192.168.0.1
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): multiple
[*] Enter start port: 1
[*] Enter end port: 65535

[!] Starting scan for 192.168.0.1
Scanned port 53: domain - Domain Name Server
Scanned port 80: http - World Wide Web HTTP
Scanned port 443: https - http protocol over TLS/SSL
Scanned port 1900: ssdp - SSDP
Scanned port 20001: microsan - MicroSAN
[+] Open ports for 192.168.0.1:
    Port 53: domain - Domain Name Server
    Port 80: http - World Wide Web HTTP
    Port 443: https - http protocol over TLS/SSL
    Port 1900: ssdp - SSDP
    Port 20001: microsan - MicroSAN
[ Scan duration: 133.65 seconds
[√] Finished scanning 192.168.0.1
------------------------------------------------------------
```

```
C:\Users\minik\Desktop\diplom>py diplom.py
[*] Enter targets to scan (split them by ,): 127.0.0.1
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): one
[*] Enter port to scan: 445

[!] Starting scan for 127.0.0.1
Scanned port 445: microsoft-ds - Microsoft-DS
[+] Open port for 127.0.0.1:
    Port 445: microsoft-ds - Microsoft-DS
[ Scan duration: 0.0 seconds
[√] Finished scanning 127.0.0.1
------------------------------------------------------------
```

Приклад роботи програми

```
(myenv) ubuntu@ubuntu-VirtualBox:~/diplom$ python3 diplom.py
[*] Enter targets to scan (split them by ,): 109.122.15.104
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): multiple
[*] Enter start port: 1
[*] Enter end port: 65535

[!] Starting scan for 109.122.15.104
Scanned port 53: domain — Domain Name Server
Scanned port 1900: ssdp — SSDP
Scanned port 20001: microsan — MicroSAN
[+] Open ports for 109.122.15.104:
    Port 53: domain — Domain Name Server
    Port 1900: ssdp — SSDP
    Port 20001: microsan — MicroSAN
[⏱] Scan duration: 133.95 seconds
[✓] Finished scanning 109.122.15.104
------------------------------------------------------------
```

```
(myenv) ubuntu@ubuntu-VirtualBox:~/diplom$ python3 diplom.py
[*] Enter targets to scan (split them by ,): 109.122.15.104
[*] Do you want to scan one port or multiple ports? (write one or multiple to select): one
[*] Enter port to scan: 53

[!] Starting scan for 109.122.15.104
Scanned port 53: domain — Domain Name Server
[+] Open port for 109.122.15.104:
    Port 53: domain — Domain Name Server
[⏱] Scan duration: 0.01 seconds
[✓] Finished scanning 109.122.15.104
------------------------------------------------------------
```

Порівняння роботи створеної програми з NMAP

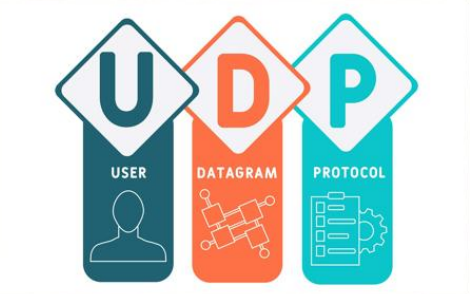
Nmap

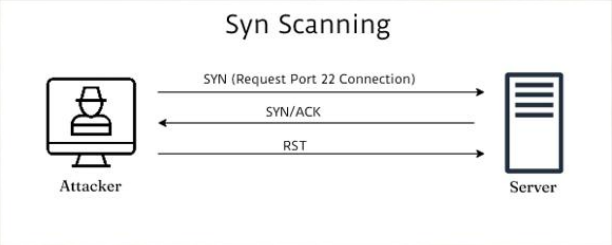
```
Scanning localhost (127.0.0.1) [65535 ports]
Discovered open port 445/tcp on 127.0.0.1
Discovered open port 135/tcp on 127.0.0.1
Discovered open port 9180/tcp on 127.0.0.1
Discovered open port 49670/tcp on 127.0.0.1
Discovered open port 49768/tcp on 127.0.0.1
Discovered open port 24584/tcp on 127.0.0.1
Discovered open port 27036/tcp on 127.0.0.1
Discovered open port 49666/tcp on 127.0.0.1
Discovered open port 49664/tcp on 127.0.0.1
Discovered open port 49821/tcp on 127.0.0.1
Discovered open port 49665/tcp on 127.0.0.1
Discovered open port 49766/tcp on 127.0.0.1
Discovered open port 5040/tcp on 127.0.0.1
Discovered open port 49678/tcp on 127.0.0.1
Discovered open port 49671/tcp on 127.0.0.1
Discovered open port 6463/tcp on 127.0.0.1
Discovered open port 27060/tcp on 127.0.0.1
Discovered open port 49669/tcp on 127.0.0.1
Discovered open port 7680/tcp on 127.0.0.1
Completed SYN Stealth Scan at 09:54, 6.53s elapsed (65535 total ports)
Initiating Service scan at 09:54
Scanning 19 services on localhost (127.0.0.1)
```

Розроблений застосунок

```
[+] Open ports for 127.0.0.1:
    Port 135: epmap — DCE endpoint resolution
    Port 445: microsoft-ds — Microsoft-DS
    Port 5040: unknown — No description
    Port 6463: unknown — No description
    Port 7680: ms-do — Microsoft Delivery Optimization Peer-to-Peer
    Port 9180: unknown — No description
    Port 27036: unknown — No description
    Port 27060: unknown — No description
    Port 49664: unknown — No description
    Port 49665: unknown — No description
    Port 49666: unknown — No description
    Port 49669: unknown — No description
    Port 49670: unknown — No description
    Port 49675: unknown — No description
    Port 49679: unknown — No description
    Port 49784: unknown — No description
    Port 49785: unknown — No description
    Port 49846: unknown — No description
    Port 49934: unknown — No description
[⏱] Scan duration: 135.57 seconds
[✓] Finished scanning 127.0.0.1
```

Розвиток і вдосконалення застосунку





ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Поліщука Ігоря Дмитровича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Гаджиєв Матін Магсудович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей*

Обсяг розрахунково-пояснювальної записки *73* сторінок

Обсяг графічної (презентаційної) частини *11* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений дипломний проєкт відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений створенню програмного застосунку сканування мережевих портів з метою виявлення вразливостей і складається з пояснювальної записки та мультимедійної презентації з відповідними схемами.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу, економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 11 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структурні, принципові та функціональні схеми, структурні моделі, блок-схеми алгоритмів, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту

Асинхронна архітектура на asyncio – дозволяє перевіряти тисячі портів одночасно в одному потоці, без важких потоків чи процесів. Модульність коду – чіткий поділ на функції scan_port, scan_one, scan_multiple, lookup_service, main тощо спрощує тестування й розширення.

д) основні недоліки дипломного проекту

Відсутність UDP-сканування – програма перевіряє лише TCP-порти, хоча багато критичних сервісів працюють по UDP (DNS, SNMP, DHCP). Незручне масштабування – при скануванні всього діапазону портів (1–65535) час роботи підскочив до понад 130 с, тоді як у Nmap це займає близько 50 с.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

«*13*» *06* *2025 р.*

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Поліщука Ігоря Дмитровича

(прізвище, ім`я та по батькові)

Спеціальність: *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 73 сторінки. У пояснювальній записці наведено етапи розробки програмного застосунку, тестування програмного застосунку, а також його порівняння з існуючими рішеннями. Графічна частина складається з 11 слайдів мультимедійної презентації, які також містять блок-схеми, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проектом: *Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Поліщук І. Д. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Поліщук І.Д.. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання *Під час дипломного проектування здобувач освіти Поліщук І.Д. мав змогу самостійно приймати окремі рішення з реалізації програмної застосунку та показав вміння організовано працювати над поставленим завданням, розробляти структурні схеми та архітектуру програмного забезпечення для сканування мережевих портів із використанням сучасних програмних засобів, таких як Python, засоби моделювання та візуалізації, а також готувати презентаційні та звітні матеріали для захисту проєкту.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту

проф. Д.Т.Н *Гаджиєв Матін Магсудович*

Місце роботи і посада керівника дипломного проекту

зав. каф. ІПЗ ДНУІТЗ

Підпис

«18» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Поліщук Ігор Дмитрович
здобувач освіти гр. 4КБ-02, та

Гаджиєв Матін Магсудович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей» (автор роботи – Поліщук І.Д., керівник роботи – Гаджиєв М.М.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Поліщук І.Д. /

Керівник _____ / Гаджиєв М.М./

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Поліщука Ігоря Дмитровича

на тему *Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з деякими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *18.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 18.06.2025 р. значення коефіцієнту подібності в роботі становить 13.23%, коефіцієнт цитування – 1.88%.*

(підпис) *18.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Поліщука І.Д.* (П.І.Б.)

проведена «*18*» *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/18/2025
Дата редагування 6/18/2025

Документ прийнятий

Звіт подібності

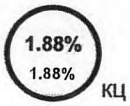
метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей
Автор Науковий керівник / Експерт
Поліщук Ігор ДмитровичГаджиєв Матин Магсудович
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КЦ

25
Довжина фрази для коефіцієнта подібності 2

11450
Кількість слів

91798
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	R	19
Інтервали	A→	0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	62

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	64 0.56 %
2	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	54 0.47 %
3	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	47 0.41 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	37 0.32 %
5	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	35 0.31 %

6	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	32 0.28 %
7	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	31 0.27 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	31 0.27 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	31 0.27 %
10	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	31 0.27 %

з домашньої бази даних (0.12 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка 3D-гри у жанрі survival-horror з налаштуваннями рівнів складності [redacted] **6**/12/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	14 (2) 0.12 %

з програми обміну базами даних (0.09 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Мокроусов Д_Кваліфікаційна робота частина 1 [redacted] **1**/10/2025 Kharkiv National University of Internal Affairs Course papers (Kharkiv National University of Internal Affairs Course papers)	10 (1) 0.09 %

з Інтернету (13.02 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download**	**391 (31) 3.41 %**
2	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content**	**264 (14) 2.31 %**
3	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	209 (20) 1.83 %
4	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**120 (6) 1.05 %**
5	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	91 (8) 0.79 %
6	**https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download**	**54 (1) 0.47 %**
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	52 (5) 0.45 %
8	**https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download**	**36 (2) 0.31 %**
9	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**32 (1) 0.28 %**
10	https://card-file.ontu.edu.ua/bitstreams/62baa43e-b968-4993-bb54-8cf8761a89b2/download	29 (3) 0.25 %
11	https://slidetodoc.com/1-http-comp-web3-ruaccessoriesmonitors-actfullidarticle5849-2-http/	26 (4) 0.23 %
12	https://card-file.ontu.edu.ua/server/api/core/bitstreams/3302e08a-9549-43ba-8861-728bff7dc7ff/content	22 (2) 0.19 %
13	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	19 (2) 0.17 %
14	https://metod.vntu.edu.ua/getfile.php/1195.pdf	15 (1) 0.13 %

15	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	14 (1) 0.12 %
16	https://ekmair.ukma.edu.ua/server/api/core/bitstreams/ddaa7270-dfd0-41c9-a6c0-15c5e14f9251/content	14 (2) 0.12 %
17	https://ep3.nuwm.edu.ua/18432/1/06-08-16.pdf	13 (2) 0.11 %
18	https://ela.kpi.ua/bitstreams/271943ff-5eb1-433b-8f11-c869145fa971/download	13 (2) 0.11 %
19	https://dkng.net.ua/doc/metod.material/baranchuk/Baranchuk_Standart_pidpryjemstva.pdf.pdf	11 (1) 0.10 %
20	https://habr.com/ru/articles/809901/	10 (1) 0.09 %
21	http://dspace.onua.edu.ua/bitstream/handle/11300/19940/%D0%A2%D0%BE%D0%BB%D0%BE%D0%BA%D0%BD%D0%BE%D0%B2%20%D0%9E%D0%B3%D0%BB%D1%8F%D0%B4.pdf?sequence=1	10 (1) 0.09 %
22	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	9 (1) 0.08 %
23	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	9 (1) 0.08 %
24	http://openarchive.nure.ua/bitstream/document/10776/1/2019_M_CITAM_Berezhetskyy_V_Ya.docx	7 (1) 0.06 %
25	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	6 (1) 0.05 %
26	https://card-file.ontu.edu.ua/bitstreams/7b1e10b9-0ac2-4b07-afc4-8cdfff7db780/download	5 (1) 0.04 %
27	https://card-file.ontu.edu.ua/bitstreams/bcb0d6f9-f464-4578-bda6-b5b2ce2349bb/download	5 (1) 0.04 %
28	https://conferences.vntu.edu.ua/public/files/mn/mn-2019_netpub.pdf	5 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж» Група: 4КБ- 02

Дипломний проєкт
здобувача **освіти денної форми навчання КБ.** 02.14.000.ДП

ПОЛІЩУКА
ІГОРЯ ДМИТРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»
Група: 4 КБ-02

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проєкту на тему:
Розробка програмного застосунку сканування
мережевих портів з метою виявлення вразливостей
Проєктний **матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах)**