МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
(повне найменування інституту, назва факультету)

Кафедра комп'ютерної інженерії та електромеханіки
(повна назва кафедри)

ДИПЛОМНА БАКАЛАВРСЬКА РОБОТА

на тему

КІБЕРСТІЙКА МУЛЬТИСЕРВІСНА КОРПОРАТИВНА МЕРЕЖА ПІДПРИЄМСТВА

Виконав: студент групи БКІ-19

спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

Андрощук А.В,
(прізвище та ініціали)

Керівник д.т.н., проф. Злотенко Б.М.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Інститут інженерії та інформаційних технологій
Кафедра комп'ютерної інженерії та електромеханіки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп"ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІЕМ

_____проф. Злотенко Б.М.

"_____" _____2023 року

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

Андрощуку Андрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема дипломної бакалаврської роботи **Кіберстійка мультисервісна корпоративна мережа підприємства**

Науковий керівник роботи Злотенко Борис Миколайович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
д.т.н., професор
затверджені наказом вищого навчального закладу від 15.03.2023 № 75-уч.
2. Строк подання студентом роботи 28 травня 2023 року
3. Вихідні дані до дипломної бакалаврської роботи: технічне завдання, технічна та патентна література.
4. Зміст дипломної бакалаврської роботи (перелік питань, які потрібно розробити): 1. Аналітичний огляд існуючих кіберстійких мультисервісних корпоративних мереж підприємства. 2. Вибір інструментів для кіберстійкої мультисервісної корпоративної мережі підприємства. 3. Розробка кіберстійкої мультисервісної корпоративної мережі підприємства.
5. Дата видачі завдання 10.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	06.03.2023	
2	Розділ 1. Аналітичний огляд існуючих кіберстійких мультисервісних корпоративних мереж підприємства	17.03.2023	
3	Розділ 2. Вибір інструментів для кіберстійкої мультисервісної корпоративної мережі підприємства	29.03.2023	
4	Розділ 3. Розробка кіберстійкої мультисервісної корпоративної мережі підприємства	12.04.2023	
6	Висновки	18.05.2023	
7	Оформлення дипломної бакалаврської роботи (чистовий варіант)	26.05.2023	
8	Здача дипломної бакалаврської роботи на кафедру для рецензування (за 14 днів до захисту)	28.05.2023	
9	Перевірка дипломної бакалаврської роботи на наявність співпадінь (за 10 днів до захисту)	01.06.2023	
10	Подання дипломної бакалаврської роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	04.06.2023	

Студент _____ Андрощук А.В.
(підпис) (прізвище та ініціали)

Науковий керівник роботи _____ Злотенко Б.М.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Андрощук А. В. Кіберстійка мультисервісна корпоративна мережа підприємства : дипломна бакалаврська робота за спеціальністю 123 Комп'ютерна інженерія / А. В. Андрощук ; наук. кер. Б. М. Злотенко. – Київ : КНУТД, 2023. – 67 с.

Робота присвячена дослідженню і розробленню кіберстійкої мультисервісної корпоративної мережі підприємства.

В першому розділі в повному обсязі описано, та проаналізовані кіберстійкі мультисервісні корпоративні мережі підприємства.

В другому розділі проаналізовано інструменти для виконання роботи.

В третьому розділі описана розробка проекту .

В четвертому представлений виготовлений прилад та інструкція щодо використання.

Пояснювальна записка виконана в текстовому редакторі Microsoft Word, в роботі використані програми Cisco Packet Tracer, IntelliJ IDEA.

Ключові слова: кібербезпека, корпоративна мережа, захист мережі, інтеграція системи, аналіз мережі, інформаційна безпека, протоколи мережі, система моніторингу мережі

ABSTRACT

Androschuk A.V. Cyber-resistant multi-service corporate network of the enterprise. - Manuscript.

Bachelor's thesis in the specialty 123 Computer Engineering, educational program "Computer Systems and Networks". - Kyiv National University of Technology and Design, Kyiv, 2023.

The work is devoted to the research and development of a cyber-resistant multi-service corporate network of the enterprise.

In the first chapter, cyber-resistant multi-service corporate networks of the enterprise are fully described and analyzed.

In the second section, the tools for performing the work are analyzed.

The third section describes the development of the project.

The fourth presents the manufactured device and instructions for use.

The explanatory note was made in the text editor Microsoft Word, Cisco Packet Tracer, IntelliJ IDEA programs were used in the work.

Keywords: cyber security, corporate network, network protection, system integration, network analysis, information security, network protocols, network monitoring system

ЗМІСТ

ВСТУП

Актуальність роботи. Тема "Кіберстійка мультисервісна корпоративна мережа підприємства" є дуже актуальною в сучасному світі. З розвитком технологій та залежності більшості підприємств від комп'ютерів та Інтернету, кібербезпека стала однією з ключових проблем у сфері інформаційної безпеки.

Розробка та забезпечення кіберстійкої мультисервісної корпоративної мережі підприємства включає в себе побудову надійної мережі, робота якої буде змодельована в програмі Cisco Packet Tracer, яка буде захищена такими методами і інструментами як: файрволи, захист від DDOS атак, використання міжсерверної аутентифікації.

Отже, дипломна робота на тему "Кіберстійка мультисервісна корпоративна мережа підприємства" буде актуальною і корисною, оскільки її результати дозволять розробити ефективні методи та інструменти для забезпечення кібербезпеки корпоративної мережі та захисту важливої інформації підприємства від кібератак. Дана робота буде корисною, так як її ціль - протистояти зростаючій кількісті кібератак та інших загроз для мережевої безпеки та безпеки корпоративних мереж.

Крім того, така робота буде сприяти розвитку нових технологій та методів в галузі кібербезпеки, що є актуальним в контексті зростаючої загрози кібератак та інших кіберзагроз.

Метою роботи є розроблення кіберстійкої мультисервісної корпоративної мережі підприємства на основі побудови ефективної та надійної моделі мережі, а також створення програмного забезпечення що дозволить зменшити ризики кібератак.

Отже, мета дипломної роботи полягає в тому, щоб дослідити та розробити ефективні методи та інструменти для забезпечення кібербезпеки корпоративної мережі підприємства що дозволить підвищити рівень захищеності мережі від кібератак та забезпечити безперебійну роботу підприємства. Дипломна робота

також має на меті дослідити поточний стан кібербезпеки на підприємстві та виявити проблемні аспекти, що можуть стати джерелом загроз для мережі. Результатом дослідження повинен бути комплексний план заходів, які необхідно реалізувати для підвищення рівня кібербезпеки підприємства.

Для досягнення поставленої мети в роботі вирішені такі **завдання:**

1. Аналіз потенційних кіберзагроз та уразливостей корпоративної мережі підприємства.
2. Вибір та побудова ефективної моделі мережі.
3. Розробка методів та інструментів захисту корпоративної мережі підприємства від кібератак, включаючи методи шифрування даних, захист від шкідливих програм та виявлення підроблених даних.
4. Розробка програмного забезпечення для уникнення загрози кібератаки.
5. Експериментальна перевірка ефективності розроблених методів та інструментів захисту корпоративної мережі підприємства.
6. Розробка рекомендацій щодо покращення кібербезпеки корпоративної мережі підприємства на майбутнє.

Результати дослідження дипломної роботи можуть бути корисні для підприємств, які мають велику корпоративну мережу та стикаються з проблемами кібербезпеки. Для цих підприємств важливо мати відповідну стратегію забезпечення кібербезпеки та ефективні методи та інструменти захисту мережі. Дипломна робота допоможе таким підприємствам зрозуміти, які кроки потрібно зробити для підвищення кібербезпеки та які інструменти можна використовувати для захисту корпоративної мережі від кібератак.

Об’єкт дослідження – створення комп’ютерних мультисервісних корпоративних мереж.

Дослідження охоплює аналіз різних аспектів мережі, включаючи архітектуру, протоколи зв'язку, застосування технологій безпеки, заходи забезпечення захисту мережі, процеси моніторингу та аудиту, а також реакцію на інциденти безпеки.

Дипломна робота зосереджується на вивченні конкретної мультисервісної корпоративної мережі підприємства, але її результати можуть бути застосовані для розуміння загальних принципів та методів захисту мереж від кібератак та інших загроз кібербезпеки.

Предмет дослідження – розроблення мультисервісної кіберстійкої корпоративної мережі підприємства, з урахуванням різних аспектів захисту, таких як програмний захист: забезпечення методів шифрування, захист від DDOS атак, використання міжсерверної аутентифікації, використання файрволів, включаючи людські ресурси, процеси, технології та структури.

Методи досліджень. Аналіз документів та літературних джерел, експертні оцінки, тестування та аудит мережі, моделювання.

Інформаційною базою досліджень є:

Науково-технічна література з тематики мережних технологій, безпеки комп'ютерних систем, інформаційної безпеки та ін.

Законодавчі акти, що регулюють питання інформаційної безпеки на підприємстві та в ІТ-сфері загалом.

Документація від виробників мережного обладнання та програмного забезпечення.

Стандарти та рекомендації організацій з мережних технологій, які встановлюють вимоги до безпеки та ефективності мережі.

Результати досліджень та проекти з розробки та впровадження подібних мереж в інших організаціях.

Експертні оцінки та поради фахівців з мережних технологій та інформаційної безпеки.

Дані та статистика з практики застосування сучасних мережних технологій та систем безпеки на підприємствах різних галузей.

Практичне значення отриманих результатів. Отримані результати дослідження можуть мати практичне значення, зокрема, щодо вдосконалення технічної інфраструктури, забезпечення безпеки та захисту інформації,

зменшення витрат на технічне обслуговування та ремонт мережі, а також забезпечення стійкості мережі під час збоїв та аварій.

Апробація отриманих результатів. Результати роботи доповідались та отримали позитивну оцінку на II Всеукраїнській науково-практичній інтернет-конференції молодих учених та студентів «Електромеханічні та інформаційні системи» (м. Київ, КНУТД, 20 квітня 2023 р.)

Структура та обсяг роботи. Дипломна робота бакалавра складається зі вступу, 3 розділів та висновків по них, загальних висновків, списку використаних джерел та додатків. Основний текст роботи викладений на 51 сторінках, містить 22 рисунка, список джерел з 22 найменувань. Загальний обсяг роботи, враховуючи додаток, складає 64 аркуша.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ КІБЕРСТІЙКИХ МУЛЬТИСЕРВІСНИХ КОРПОРАТИВНИХ МЕРЕЖ ПІДПРИЄМСТВА

1.1. Структура кіберстійкої мультисервісної мережі підприємства

Кіберстійка мультисервісна мережа підприємства - це інформаційна мережа, яка забезпечує зв'язок між різними комп'ютерними пристроями та системами, що використовуються в діяльності підприємства. Ця мережа має бути надійно захищена від зовнішніх кіберзагроз, таких як хакерські атаки та віруси, а також мати високий рівень доступності та швидкості передачі даних.

Основна мета кіберстійки - забезпечити швидкий та безперебійний обмін даними та інформацією між різними підрозділами підприємства, що забезпечує підвищення продуктивності та ефективності роботи, а також скорочення часу на вирішення проблем.

Крім того, кіберстійка мультисервісна мережа підприємства дозволяє забезпечити надійний захист конфіденційної інформації, зокрема за допомогою шифрування та аутентифікації даних, що зберігаються та передаються по мережі.

Основні складові кіберстійки мультисервісної мережі підприємства включають в себе різноманітні мережеві пристрої, такі як маршрутизатори, комутатори, файрволи, сервери, системи зберігання даних, безперебійні джерела живлення та інші компоненти. Крім того, кіберстійка також може включати в себе різноманітні програмні засоби, такі як операційні системи, системи віртуалізації, бази даних, програми для забезпечення безпеки та інші.

У сучасному бізнес-середовищі кіберстійка мультисервісна мережа підприємства є необхідністю для ефективної та безпечної роботи підприємства.

Кіберстійка мережа може бути розгорнута в різних сферах, таких як військова, фінансова, медична тощо. У таких сферах захист даних є особливо важливим, і тому кіберстійка мережа може стати надійним засобом захисту інформації.

Окрім технічних заходів, важливою частиною кіберстійкої мережі є людський фактор. Зокрема, користувачі мережі повинні мати належне розуміння ризиків тих дій в мережі, знати, як захистити свої дані, використовувати паролі складної структури та відмінні від інших, уникати підозрілих посилань та вкладень, використовувати антивірусне програмне забезпечення та оновлювати його регулярно, бути пильними щодо підозрілих дій у мережі.

Крім того, кіберстійка мережа повинна бути готова до реагування на можливі загрози та швидкої відповіді на їхні прояви. Це може включати автоматизовану систему виявлення загроз, плани екстреного реагування та резервне копіювання даних для відновлення в разі атаки або невідомих помилок.

Кіберстійка мережа також може бути використана для підвищення ефективності та продуктивності роботи в компанії. Вона дозволяє віддалену роботу, обмін даними, розподілену обробку даних, що знижує вартість і збільшує швидкість роботи компанії. Крім того, кіберстійка мережа дозволяє забезпечити доступ до даних з будь-якої точки світу, що є важливим для глобальної бізнес-інфраструктури.

У сучасному світі, де кіберзлочинність стає все більшою загрозою, розробка та впровадження кіберстійких мереж є важливим завданням. Забезпечення високого рівня захисту інформації та ресурсів мережі є необхідністю для забезпечення безпеки користувачів та компаній, а також для забезпечення стабільності та продуктивності комп'ютерних систем у сучасному світі [1].

1.2. Огляд топологій існуючих комп’ютерних мереж

Комп'ютерна мережа - це з'єднання двох або більше комп'ютерів, що дозволяє їм обмінюватися даними, ресурсами та послугами. Ці мережі можуть бути збудовані на базі різноманітних технологій, таких як Ethernet, Wi-Fi, Bluetooth, та інших.

Комп'ютерні мережі можуть бути класифіковані за різними ознаками, такими як розмір (локальна, міська, глобальна), топологія (зірка, шина, дерево, кільце, меш), протоколи комунікації (TCP/IP, HTTP, FTP, SMTP, і т. д.), та інші.

Комп'ютерні мережі мають безліч застосувань, таких як спільне використання джерел, друк, обмін файлами, доступ до Інтернету, відео конференції, онлайн-ігри, та багато іншого. Також вони забезпечують можливість зберігання та обробки даних в централізованому вигляді, що робить їх корисними для бізнесу та наукових досліджень.

Топологія "Загальна шина" (англ. Bus topology) - це тип комп'ютерної мережі, в якій кожен комп'ютер підключений до спільної шини (кабелю), до якої також підключені інші комп'ютери та пристрої (рис. 1.1).

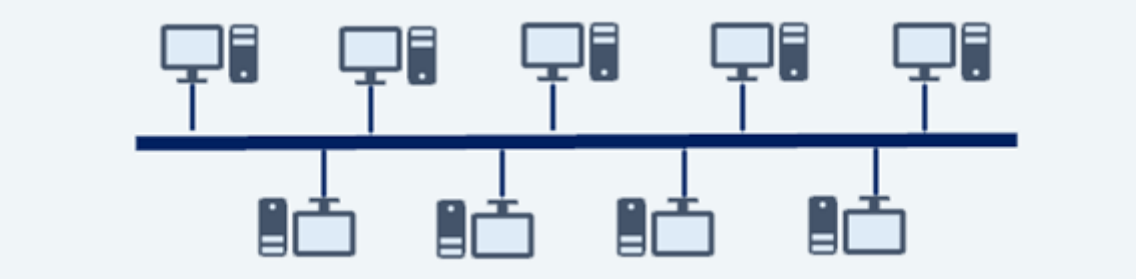


Рисунок 1.1 – Топологія «загальна шина»

У цій топології дані передаються з одного пристрою до іншого через спільний канал передачі даних, що може призвести до затримок та колізій даних, особливо в тому випадку, якщо багато комп'ютерів використовують цю шину одночасно.

Однак топологія "Загальна шина" має переваги, зокрема простоту налаштування та дешевизну, оскільки не потребує додаткового обладнання, такого як концентратори чи комутатори. Також вона може бути легко розширена шляхом додавання нових пристроїв до шини.

Загальна шина використовується в деяких локальних мережах, таких як Ethernet, але частіше використовуються топології "Зірка" та "Дерево", які мають більш високу надійність та швидкість передачі даних.

Топологія «кільце» (англ. Ring topology) - це тип комп'ютерної мережі, в якій кожен комп'ютер підключений до наступного та попереднього комп'ютерів в коловому зв'язку (рис. 1.2).

Рисунок 1.2 – Топологія «кільце»

У цій топології дані передаються по колу в одному напрямку, і кожен комп'ютер може отримувати та відправляти дані. Для передачі даних використовуються технології, такі як Token Ring, в яких передача даних відбувається тільки тоді, коли мережевий пристрій отримує контрольний токен.

Однією з переваг топології «кільце» є те, що вона забезпечує рівномірний розподіл навантаження між комп'ютерами та пристроями мережі, що робить її ефективною для передачі великих об'ємів даних. Також вона забезпечує високу надійність, оскільки в разі відмови одного комп'ютера мережа все ще може працювати, але з меншою швидкістю.

Однак топологія «кільце» має свої недоліки, зокрема високу вартість та складність ремонту та підтримки, а також затримки при передачі даних, особливо, якщо в мережі багато комп'ютерів. Зараз ця топологія майже не використовується в нових мережевих рішеннях, проте вона може використовуватися у застарілих мережах.

Топологія «зірка» (англ. Star topology) - це тип комп'ютерної мережі, в якій кожен комп'ютер підключений до центрального пристрою, такого як комутатор або концентратор (рис. 1.3).

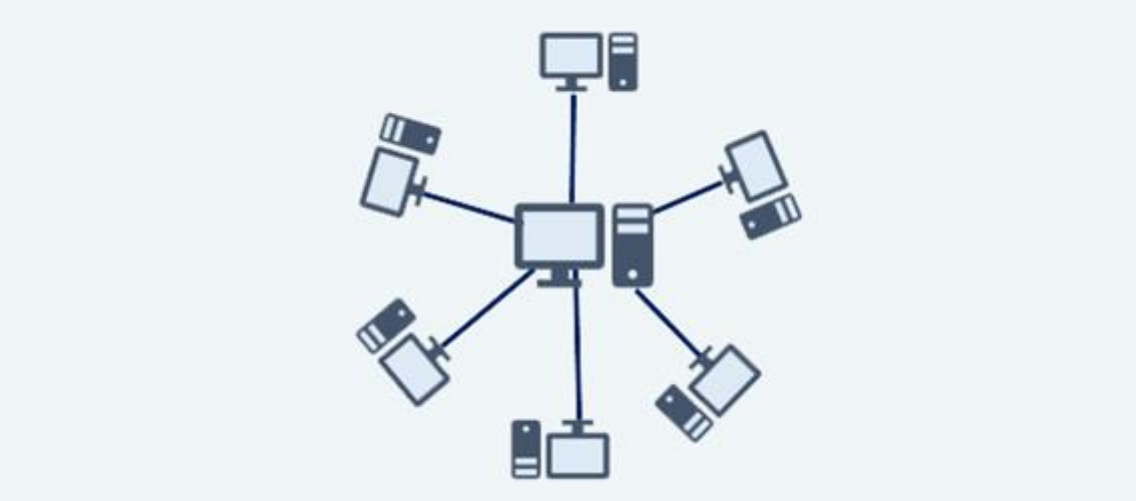


Рисунок 1.3 – Топологія «зірка»

У цій топології дані передаються від кожного комп'ютера до центрального пристрою, а потім розсилаються до призначеного комп'ютера. Це забезпечує високу швидкість передачі даних та зменшує вплив можливих затримок та колізій. Крім того, якщо один з комп'ютерів відмовить, то решта мережі все ще може працювати.

Топологія «зірка» має декілька переваг. Перш за все, вона дуже проста у налаштуванні та підтримці. Крім того, вона забезпечує легке підключення нових комп'ютерів та пристроїв до мережі, оскільки їх потрібно просто підключити до центрального пристрою.

Однак топологія «зірка» має свої недоліки. Перш за все, центральний пристрій є одиничним пунктом відмови, і якщо він вийде з ладу, то все з'єднання мережі буде зупинено. Крім того, вона вимагає більшої кількості кабелів, що може збільшувати вартість мережі, особливо якщо кожен комп'ютер має свій власний кабель до центрального пристрою.

Топологія «зірка» є однією з найбільш поширених топологій у сучасних мережевих рішеннях, особливо у локальних мережах Ethernet.

Якщо топологія складається з двох або більше різних топологій, вона називається гібридною топологією (рис. 1.4).

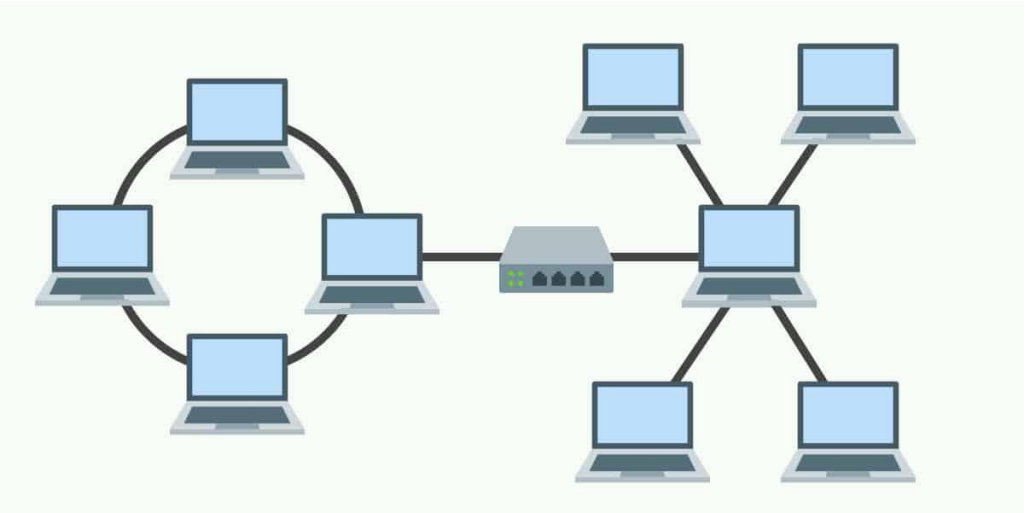


Рисунок 1.4 – Гібридна топологія

Гібридні топології є найбільш часто зустрічаються на великих підприємствах де окремі відділи мають мережеві топології, що відрізняються від інших топологій в організації. З'єднання цих топологій разом призведе до гібридної топології. Як наслідок, можливості та вразливості залежать від типів топології, які пов'язані між собою [2].

1.3. Огляд мережевих файрволів

Мережевий файрвол - це система безпеки мережі, яка контролює трафік, що проходить через неї, та приймає рішення про те, який трафік дозволено проходити, а який блокується (рис. 1.5).

Види файрволів:

1. **Файрвол мережного рівня**, також відомий як мережевий файрвол, це програмний або апаратний засіб, який використовується для захисту мережі від несанкціонованого доступу і зловмисних дій.

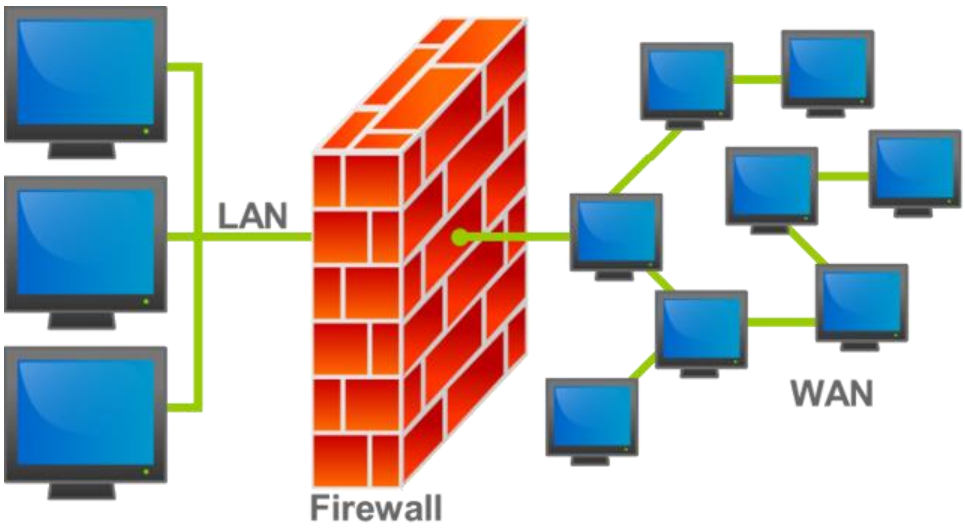


Рисунок 1.5 – Мережевий файрвол

Цей тип файрвола контролює рух трафіку на мережевому рівні OSI, де дані передаються в пакетах між різними мережевими пристроями. Файрвол мережного рівня зазвичай працює з даними на рівні мережевого протоколу, такі як IP-адреси та порти.

Основні функції файрвола мережного рівня полягають у блокуванні небезпечного трафіку, фільтрації трафіку на основі IP-адрес, портів та протоколів, контролю доступу до мережі та іншій різноманітній функціональності.

Основна перевага мережевих файрволів полягає в їхній здатності захищати мережу від зловмисних дій, таких як віруси, хакерські атаки та інші загрози, що можуть виникати в процесі роботи мережі. Файрволи мережного рівня також забезпечують більш високий рівень безпеки, ніж файрволи на рівні додатків, оскільки вони можуть блокувати трафік на ранніх етапах обміну даними в мережі.

2. **Файрвол прикладного рівня**, також відомий як додатковий або програмний файрвол, це засіб захисту, який працює на рівні додатків мережі, таких як веб-сайти, електронна пошта і т.д.

Основна функція файрвола прикладного рівня полягає у блокуванні небезпечного трафіку на основі типу додатку, протоколу та інших характеристик. Файрвол прикладного рівня може блокувати конкретні протоколи та порти, які використовуються для передачі небезпечного трафіку, такого як віруси, шкідливі програми, атаки з використанням вразливостей в програмах та інше.

До переваг файрвола прикладного рівня належить те, що він здатний виявляти та блокувати певні типи трафіку, які можуть бути пропущені через файрвол мережного рівня. Це дозволяє забезпечити вищий рівень безпеки та захисту мережі від небезпечного трафіку, який може бути складним для виявлення на рівні мережевого протоколу.

Однак, файрвол прикладного рівня може бути менш ефективним у виявленні та блокуванні складних атак, які використовують багатошаровий підхід та шифрування для приховування своєї активності. Крім того, файрвол прикладного рівня може спричинити певні проблеми з продуктивністю та швидкістю мережі, оскільки він працює на рівні додатків та може потребувати додаткових ресурсів для обробки трафіку.

3. **Файрвол рівня з'єднання**, також відомий як файрвол рівня сесій, це засіб захисту, який працює на рівні з'єднань мережі. Основна функція файрвола рівня з'єднання полягає у контролі доступу до мережевих ресурсів на основі стану з'єднання та інших параметрів.

Файрвол рівня з'єднання працює на рівні TCP/IP, а не на рівні додатків, як файрвол рівня прикладного. Він використовує інформацію про стан з'єднання, таку як IP-адреси і порти, щоб визначити, чи дозволено з'єднання до мережевого ресурсу.

Один з головних переваг файрвола рівня з'єднання полягає у тому, що він здатний виявляти та блокувати небезпечний трафік, який може пройти через файрволи більш низьких рівнів. Файрвол рівня з'єднання також забезпечує більшу точність у відборі трафіку, що дозволяє забезпечити більш високий рівень безпеки мережі.

Однак, файрвол рівня з'єднання може мати певні недоліки, такі як обмежена продуктивність та швидкість мережі. Також, він може бути менш ефективним у виявленні та блокуванні складних атак, які використовують багатошаровий підхід та шифрування для приховування своєї активності [3].

Висновки до розділу 1

У цьому розділі було розглянуто та проаналізовано яким чином можна забезпечити кіберстійкі корпоративні мережі підприємства. Зокрема було розглянуто що таке кіберстійка мультисервісна корпоративна мережа підпримєства, огляд комп’ютерних систем та поняття та види фаєрволів.

Завдяки стрімкому розвитку технологій та інформаційного простору, стає все актуальнішою потреба у забезпеченні кібербезпеки в комп’ютерних мережах, зокрема в мультисервісних мережах підприємств.

Огляд комп'ютерних мереж, зокрема різних типів топологій, є важливим етапом в забезпеченні безпеки мережі, оскільки він дозволяє визначити спосіб підключення компонентів мережі та відповідні виклики для їх захисту.

Вибір гібридної топології є досить хорошим результатом, адже масштабованість гібридних налаштувань робить їх добре пристосованими до великих мереж що дає змогу розширяти мережу, а це є однією з найважливіших переваг для створення мережі підприємства.

Огляд мережевих файрволів вказує на їх важливість для забезпечення кібербезпеки мережі. Кожен тип файрволів має свої особливості та можливості щодо захисту мережі від зовнішніх загроз. Застосування мережевих файрволів дозволяє ефективно контролювати доступ до мережевих ресурсів та виявляти та блокувати небезпечний трафік.

Отже, усі ці аспекти є важливими для забезпечення кібербезпеки мультисервісної мережі підприємства, оскільки вони дозволяють виявляти та блокувати загрози, контролювати доступ до мережевих ресурсів та забезпечувати безпеку даних. Забезпечення кібербезпеки є важливою задачею для будь-якої компанії, і використання заходів безпеки, таких як мережеві файрволи, є важливою складовою її успіху та стійкості.

РОЗДІЛ 2. ВИБІР ІНСТРУМЕНТІВ ДЛЯ КІБЕРСТІЙКОЇ МУЛЬТИСЕРВІСНОЇ КОРПОРАТИВНОЇ МЕРЕЖІ ПІДПРИЄМСТВА

2.1 Етапи побудови кіберстійкої мультисервісної корпоративної мережі підприємства

Побудова кіберстійкої мультисервісної корпоративної мережі включає в себе комплекс завдань які необхідно вирішити для досягнення цієї цілі.

Основні етапи такої побудови можуть включати наступне:

1. Аналіз потреб та вимог до мережі: необхідно ретельно проаналізувати потреби та вимоги підприємства до мережі, включаючи кількість користувачів, типи сервісів та додатків, обсяги даних, рівень безпеки тощо.
2. Розробка архітектури мережі: після проведення аналізу необхідно розробити архітектуру мережі, яка відповідає потребам підприємства. Це може включати в себе вибір типу мережі, топологію, розташування активного та пасивного обладнання, мережеві протоколи тощо.
3. Вибір обладнання та програмного забезпечення: після розробки архітектури мережі необхідно вибрати обладнання та програмне забезпечення, які відповідають вимогам підприємства та архітектурі мережі.
4. Розгортання мережі: після вибору обладнання та програмного забезпечення необхідно провести розгортання мережі. Це може включати в себе встановлення та налаштування активного та пасивного обладнання, встановлення та налаштування програмного забезпечення, створення мережевих з'єднань тощо.
5. Налаштування безпеки: після розгортання мережі необхідно провести налаштування безпеки мережі, включаючи встановлення файрволів, аутентифікацію та авторизацію користувачів, контроль доступу до ресурсів мережі тощо.
6. Тестування та налагодження: після налаштування безпеки необхідно провести тестування та налагодження мережі. Це дозволяє перевірити

працездатність та ефективність мережі, а також виявити та виправити можливі проблеми.

7. Підтримка та адміністрування: після успішного розгортання та тестування мережі необхідно забезпечити її підтримку та адміністрування. Це може включати в себе моніторинг мережі, усунення проблем, налаштування нових сервісів та додатків, а також регулярну оновлення програмного забезпечення та обладнання.

8. Розвиток та оновлення: мережа підприємства повинна бути постійно розвиватись та оновлюватись з метою забезпечення максимальної ефективності та безпеки. Це може включати в себе встановлення нового обладнання, розширення мережі, оновлення програмного забезпечення тощо.

Усі ці етапи є важливими для побудови кіберстійкої мультисервісної корпоративної мережі підприємства та допоможуть забезпечити її ефективність, безпеку та стабільність [4].

2.2 Вибір та аналіз доступних технологій та програмного забезпечення

Для вирішення поставленої задачі розробки кіберстійкої мультисервісної корпоративної мережі підприємства будуть використані наступні програми:

1. Cisco Packet Tracer - це програмне забезпечення, що використовується для моделювання мережевих схем та протоколів. Воно дозволяє студентам та професіоналам в області мережевих технологій вивчати та відтворювати різні сценарії роботи мережі без необхідності реального обладнання.

Cisco Packet Tracer дозволяє користувачам створювати та моделювати мережі, встановлювати на них мережеві пристрої, такі як маршрутизатори, комутатори та файрволи, налаштовувати їх та тестувати їх роботу. Він також містить бібліотеку пристроїв та технологій, які використовуються в сучасних мережах, таких як IPv6, OSPF, EIGRP, BGP, VLAN, VPN та багато інших.

Програмне забезпечення Cisco Packet Tracer підтримується компанією Cisco та доступне для використання на різних платформах, включаючи Windows, macOS та Linux. Це дозволяє користувачам легко створювати, налаштовувати та тестувати мережеві конфігурації, що робить його корисним інструментом для вивчення мережевих технологій.

2. IntelliJ IDEA - це потужний інструмент для розробки програмного забезпечення на мові Java та інших мовах, що дозволяє зручно та ефективно працювати з кодом та підвищувати продуктивність розробки.

IntelliJ IDEA має високу популярність серед програмістів, що працюють з Java, оскільки надає зручний інтерфейс, широкий функціонал та підтримку різних інструментів.

Деякі з функцій IntelliJ IDEA включають:

- Підтримка великих проектів: IntelliJ IDEA має високу продуктивність та може працювати з дуже великими проектами, що робить його популярним серед команд розробників.
- Розумний редактор коду: IntelliJ IDEA має розумний редактор коду, який дозволяє швидко та ефективно писати код, надаючи автодоповнення, перевірку синтаксису та інші корисні функції.
- Підтримка Git: IntelliJ IDEA підтримує роботу з системою контролю версій Git, що дозволяє зручно працювати з кодом у команді.
- Підтримка тестування: IntelliJ IDEA має вбудовану підтримку для тестування коду, що дозволяє швидко запускати тести та аналізувати результати.
- Підтримка фреймворків: IntelliJ IDEA має вбудовану підтримку для різних фреймворків, таких як Spring, Hibernate, Struts та інших.
- Підтримка мови Kotlin: IntelliJ IDEA розроблено компанією JetBrains, яка є розробником мови Kotlin, тому підтримка Kotlin в IntelliJ IDEA є дуже доброю.

2.3 Комутатори

Комутатор (англ. switch) - це мережевий пристрій, який використовується для підключення різних пристроїв до мережі і передачі даних між ними. Комутатор забезпечує з'єднання кількох пристроїв, таких як комп'ютери, принтери, сервери, маршрутизатори тощо, у локальній мережі (LAN).

Комутатори використовуються в локальних мережах з метою підвищення швидкості передачі даних та забезпечення безпеки мережі. Кожен порт комутатора забезпечує ізольоване з'єднання між двома пристроями, що дозволяє передавати дані від одного пристрою до іншого без втрати швидкості.

Основні функції комутаторів:

1. Фільтрація даних: комутатор фільтрує пакети даних, пересилаючи їх тільки на порти, до яких приєднані необхідні пристрої.
2. Підтримка VLAN: комутатори можуть розділяти мережу на різні віртуальні LAN (VLAN), що дозволяє забезпечити більшу безпеку та ефективність мережі.
3. Керування трафіком: комутатори можуть керувати трафіком мережі, що дозволяє підвищити продуктивність та безпеку мережі.
4. Підтримка QoS: комутатори можуть забезпечувати якість обслуговування (QoS) для різних типів трафіку, що дозволяє підвищити продуктивність та ефективність мережі.
5. Підтримка додаткових функцій: комутатори можуть мати додаткові функції, такі як підтримка Power over Ethernet (PoE), що дозволяє живити підключені пристрої через кабель Ethernet, без потреби в окремому джерелі живлення.

Загалом, комутатори є важливим елементом інфраструктури мережі, який дозволяє забезпечувати швидку та безпечну передачу даних між різними пристроями у локальній мережі. Комутатори можуть бути використані в різних варіаціях, від простих малогабаритних пристроїв для домашнього використання до потужних комутаторів для підприємств з великим обсягом даних (рис. 2.3).

Рисунок 2.3 – Комутатори Cisco 2960-24TT

Характеристики комутатора Cisco 2960-24TT

1. Кількість портів: Комутатор Cisco 2960-24TT має 24 порти.
2. Швидкість портів: Кожен порт комутатора підтримує швидкість 10/100 Мбіт/с.
3. Управління мережею: Комутатор підтримує протоколи управління мережею, такі як SNMP, CLI, Telnet, SSH і веб-інтерфейси.
4. Підтримка VLAN: Комутатор підтримує віртуальні локальні мережі (VLAN) і може розділити комутатор на незалежні сегменти мережі.
5. QoS (Quality of Service): Комутатор підтримує функцію QoS, яка дозволяє приділяти пріоритет трафіку залежно від важливості та вимог до пропускної здатності.

Один з важливих аспектів роботи комутатора - це його механізм комутації. Комутація відбувається в двох основних режимах - cut-through та store-and-forward. У режимі cut-through, комутатор пересилає пакети даних безпосередньо після отримання пакета, що дозволяє знизити затримки в мережі, але може призвести до пошкодження даних. У режимі store-and-forward, комутатор збирає пакети даних перед пересиланням, перевіряючи їх правильність та цілісність, що

забезпечує більшу надійність мережі, але може збільшити затримки в передачі даних.

Крім того, комутатори можуть мати різні кількості портів - від кількох до кількох сотень, в залежності від призначення та потреб користувача. Також комутатори можуть мати підтримку різних протоколів - наприклад, Ethernet, Token Ring, FDDI та інших.

Крім базової функціональності, такої як пересилання даних між пристроями, деякі комутатори можуть мати додаткові функції, такі як Quality of Service (QoS), Virtual LANs (VLANs), Spanning Tree Protocol (STP) та інші. STP дозволяє забезпечити безпеку та надійність мережі, зменшуючи ризик петель в мережі та забезпечуючи автоматичне відновлення мережі в разі виникнення проблем.

У загальному, використання комутаторів в мережевих інфраструктурах дозволяє підвищити швидкість та продуктивність передачі даних, забезпечити безпеку мережі, та оптимізувати роботу мережевого середовища [5].

2.4 Маршрутизатори

Маршрутизатор - це мережевий пристрій, який дозволяє підключати різні мережі і керувати трафіком між ними. Він працює на третьому (Network) рівні моделі OSI і забезпечує передачу даних між різними мережами, знаходячи оптимальний шлях для кожного пакету даних (рис. 2.4).

Рисунок 2.4 – Маршрутизатор Cisco 2901

У маршрутизатора є кілька інтерфейсів, які дозволяють йому підключатися до різних мереж. Наприклад, для підключення до Інтернету використовуються WAN-інтерфейси, такі як ADSL, кабельний модем, супутниковий зв'язок тощо.

Характеристики маршрутизатора Cisco 2901:

1. Швидкість маршрутизації: Маршрутизатор Cisco 2901 здатен обробляти до 75 мільйонів пакетів в секунду (Mpps).
2. Швидкість WAN-портів: Має вбудований порт WAN з швидкістю до 75 Mbps.
3. Швидкість LAN-портів: Має 2 вбудованих порта Gigabit Ethernet (10/100/1000 Mbps) для підключення до локальної мережі.
4. Модульність: Маршрутизатор Cisco 2901 має розширювальні слоти для встановлення додаткових модулів, таких як модулі комутації, модулі безпеки, модулі голосової комутації тощо.
5. Підтримка VPN: Підтримує віртуальні приватні мережі (VPN), включаючи IPsec VPN та SSL VPN для безпечного з'єднання з віддаленими мережами або користувачами.
6. QoS (Quality of Service): Підтримка функції QoS дозволяє приділяти пріоритети різним типам трафіку для забезпечення оптимальної якості обслуговування.
7. Вбудований апаратний мережевий прискорювач: Має вбудований апаратний прискорювач для покращення продуктивності маршрутизації та обробки пакетів.
8. Підтримка голосу: Має можливість підключення голосових пристроїв та підтримує голосові послуги, такі як IP-телефонія.
9. Безпека: Забезпечує різні функції безпеки, включаючи захист від атак, мережевий моніторинг, шифрування трафіку тощо.

Для підключення до локальної мережі, наприклад, комп'ютерів в офісі, використовуються LAN-інтерфейси, такі як Ethernet або Wi-Fi.

Маршрутизатори зазвичай використовують протоколи маршрутизації для визначення оптимального шляху для пакетів даних. Ці протоколи дозволяють маршрутизаторам обмінюватися інформацією про доступні маршрути та визначати найкоротший шлях для кожного пакету даних. Популярні протоколи маршрутизації включають OSPF, RIP та BGP.

Крім базової функціональності, маршрутизатори можуть мати додаткові функції, такі як Quality of Service (QoS), Network Address Translation (NAT) та інші. QoS дозволяє пріоритизувати різні види трафіку, щоб забезпечити високу якість обслуговування для бізнес-застосувань, тоді як NAT дозволяє перетворювати IP-адреси для забезпечення безпеки та конфіденційності мережі.

Маршрутизатори використовуються в різних мережевих середовищах, від невеликих домашніх мереж до великих корпоративних мереж з сотнями або навіть тисячами комп'ютерів. Великі корпорації можуть використовувати кластери маршрутизаторів для забезпечення високої доступності та надійності мережі.

Одним з прикладів використання маршрутизаторів є розподілений офіс. Розподілений офіс - це мережева архітектура, в якій працівники знаходяться в різних місцях, але з'єднані в одну логічну мережу. Маршрутизатори дозволяють з'єднувати різні офіси між собою та забезпечувати доступ до спільних ресурсів, таких як файли та друкери. Крім того, маршрутизатори можуть бути використані для підключення до хмарних сервісів та інтернет-провайдерів.

Загалом, маршрутизатори грають важливу роль в будь-якій мережі, забезпечуючи надійність, безпеку та ефективність передачі даних. Інтернет є найбільшою мережею маршрутизаторів у світі, яка забезпечує зв'язок між мільярдами пристроїв та користувачів з усього світу.

Маршрутизатори також важливі для забезпечення безпеки мережі. Вони можуть бути налаштовані для фільтрації та блокування трафіку, який містить шкідливі програми або шкідливі IP-адреси. Крім того, маршрутизатори можуть

бути використані для налаштування віртуальних приватних мереж (VPN), які забезпечують безпечний доступ до мережевих ресурсів через Інтернет [6].

Висновки до розділу 2

Комутатори та маршрутизатори - це основні елементи мережі, які забезпечують передачу даних та маршрутизацію мережевого трафіку. Комутатори використовуються для підключення пристроїв в мережі та передачі даних внутрішньо мережі, а маршрутизатори - для маршрутизації даних між різними мережами.

Однак, при використанні комутаторів та маршрутизаторів, необхідно дотримуватися заходів кібербезпеки, оскільки ці пристрої можуть бути скомпрометовані зловмисниками та використані для атак на мережу. Для забезпечення кібербезпеки використовуються захисні механізми, такі як брандмауери, VPN-з'єднання та регулярне оновлення програмного забезпечення.

Крім того, в програмуванні на мові Java також необхідно дотримуватися заходів кібербезпеки, оскільки програми можуть містити вразливості, які можуть бути використані зловмисниками для атак на систему. Для забезпечення кібербезпеки на Java використовуються безпечні функції та бібліотеки, які забезпечують безпеку від більшості атак, а також заходи для запобігання SQL-ін'єкціям та XSS-атакам.

Узагальнюючи, кібербезпека є важливою складовою будь-якої мережі та програмного забезпечення. Дотримання заходів кібербезпеки може допомогти запобігти атакам та забезпечити безпеку та захист даних в мережах та програмах.

РОЗДІЛ 3. РОЗРОБКА КІБЕРСТІЙКОЇ МУЛЬТИСЕРВІСНОЇ КОРПОРАТИВНОЇ МЕРЕЖІ ПІДПРИЄМСТВА

3.1 Побудова моделі мережі підприємства

Побудова моделі мережі для підприємства є складним процесом, який вимагає розуміння топології мережі, вибору обладнання, налаштування мережевої адресації та інших параметрів. У цьому процесі було використовувано програмне забезпечення для побудови моделі мережі Cisco Packet Tracer (рис. 3.1 – 3.4).

Основні кроки побудови моделі мережі для підприємства:

1. Визначення топології мережі: вибір типу топології залежить від розміру мережі, вимог до продуктивності та надійності, і може бути зірковою, деревоподібною, комбінованою топологією.
2. Вибір обладнання: вибір обладнання залежить від потреб мережі в ресурсах та функціональних можливостях. Обладнання може включати маршрутизатори, комутатори, файрволи, сервери та інше обладнання.
3. Налаштування мережевої адресації: визначення IP-адреси для кожного пристрою в мережі, встановлення маски мережі та шлюзу за замовчуванням.
4. Конфігурування обладнання: налаштування маршрутизаторів, комутаторів та іншого обладнання, включаючи встановлення паролів, налаштування VLAN, VPN, маршрутизації, безпеки та інше.
5. Перевірка мережі: виконання тестів, щоб переконатися в працездатності мережі та виявлення та виправлення помилок.

Після цих кроків, мережу можна розгорнути на практиці, налаштувавши обладнання відповідно до розробленої моделі та провівши необхідні тести на працездатність [7].

Після встановлення та налаштування мережі, необхідно забезпечувати постійне її функціонування та підтримку. Це означає, що потрібно відстежувати стан мережі, виявляти та виправляти помилки, оновлювати програмне забезпечення та забезпечувати захист від загроз.

Для підтримки мережі можна використовувати спеціальні програми та інструменти, такі як моніторинг мережі, системи резервного копіювання, системи захисту від вірусів та інші. Також важливо проводити регулярні аудити мережі для виявлення потенційних загроз та вразливостей.

Усі ці кроки допоможуть забезпечити безперебійне та ефективне функціонування мережі, що є критично важливим для бізнесу та його успіху.

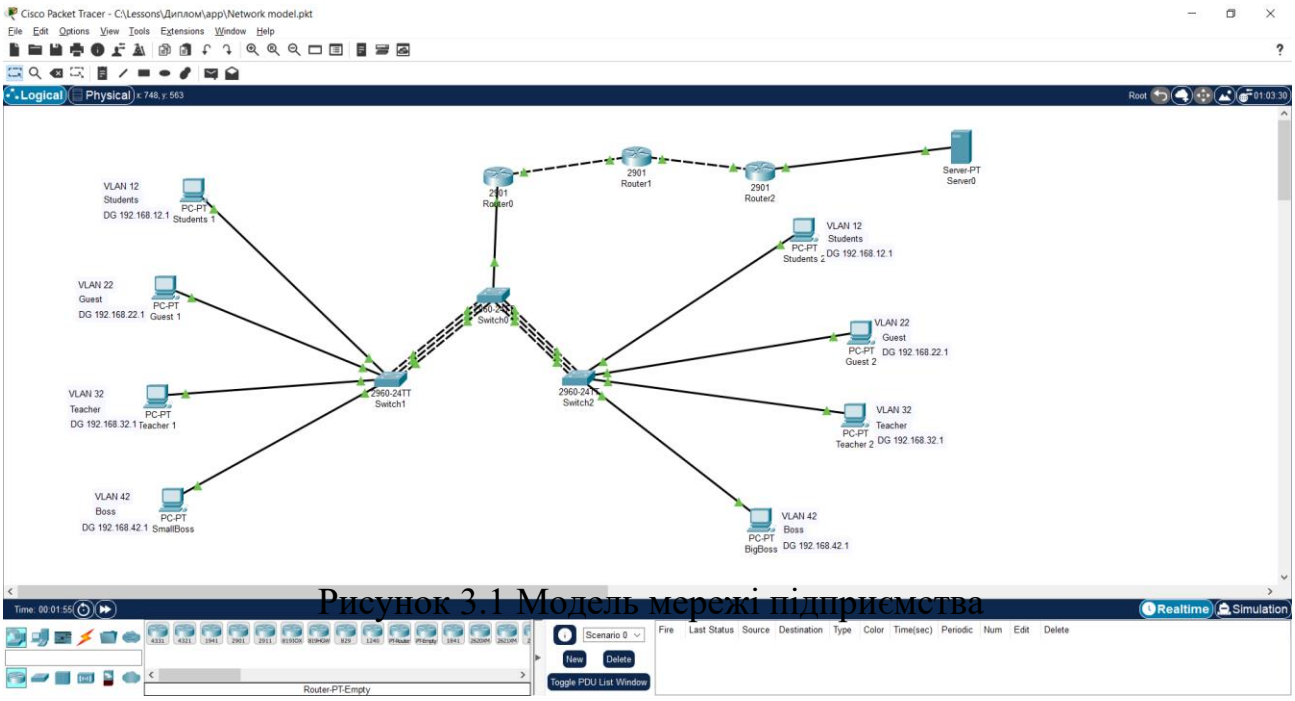


Рисунок 3.1 Модель мережі підприємства

Рисунок 3.1 – Побудова моделі мережі

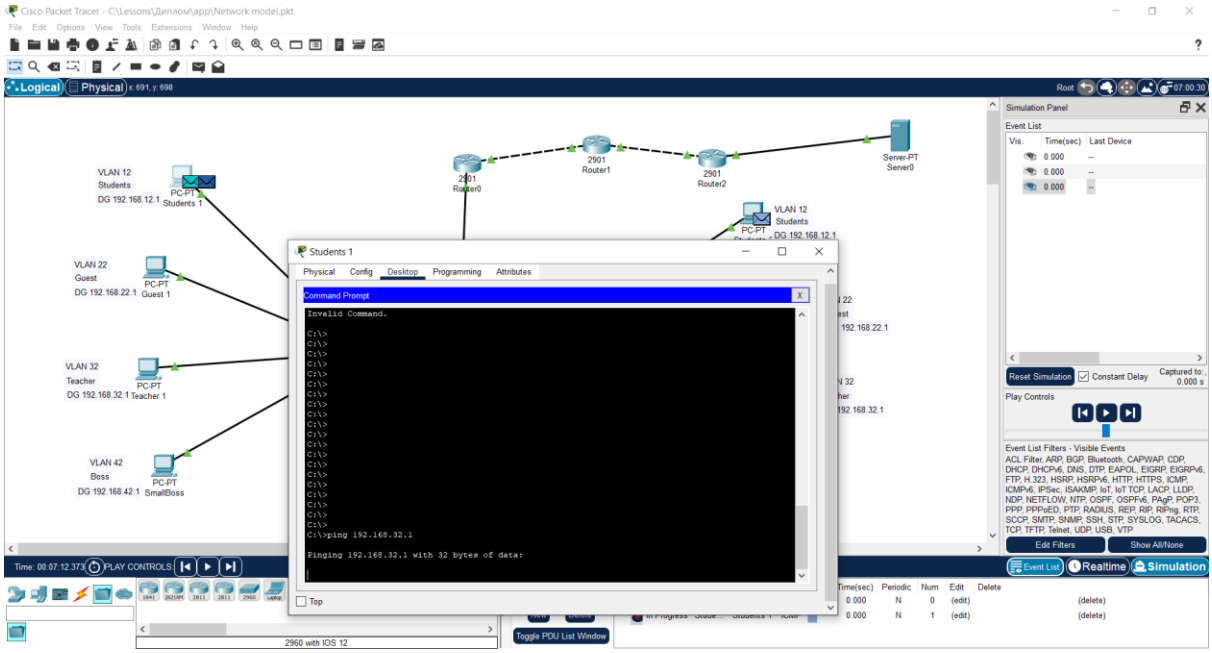


Рисунок 3.2 – Процес пінгування комп’ютера

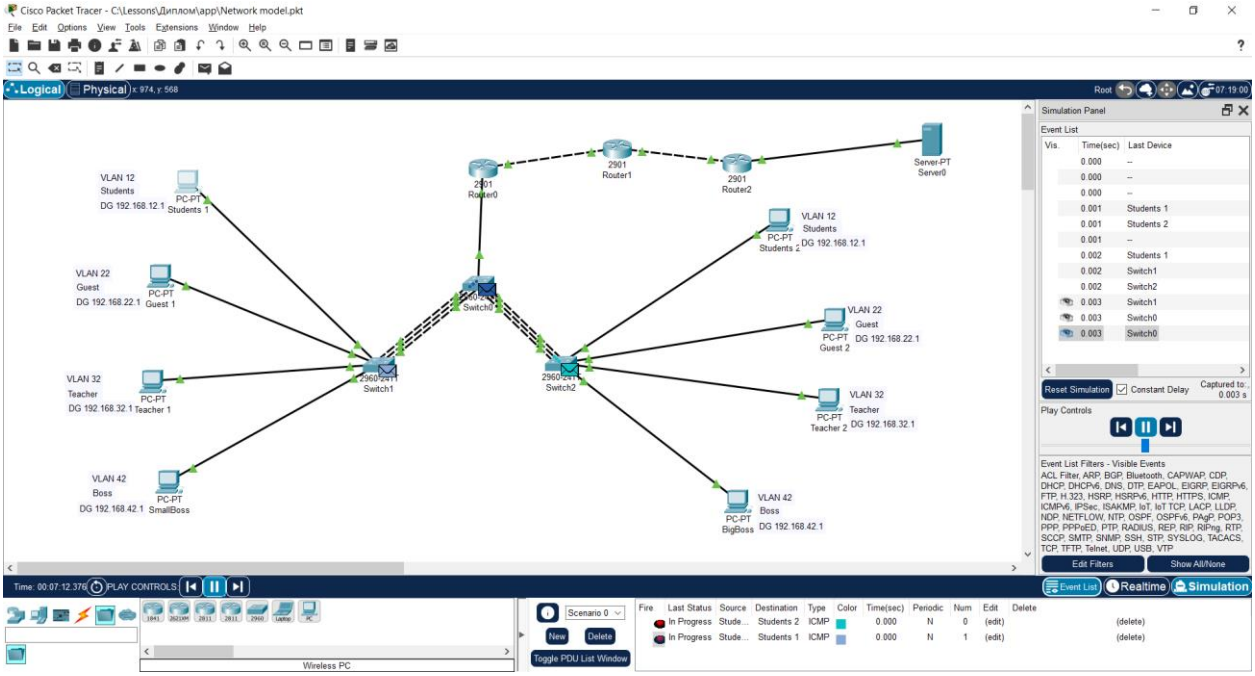


Рисунок 3.3 – Модель відправлення пакетів до іншого комп’ютера

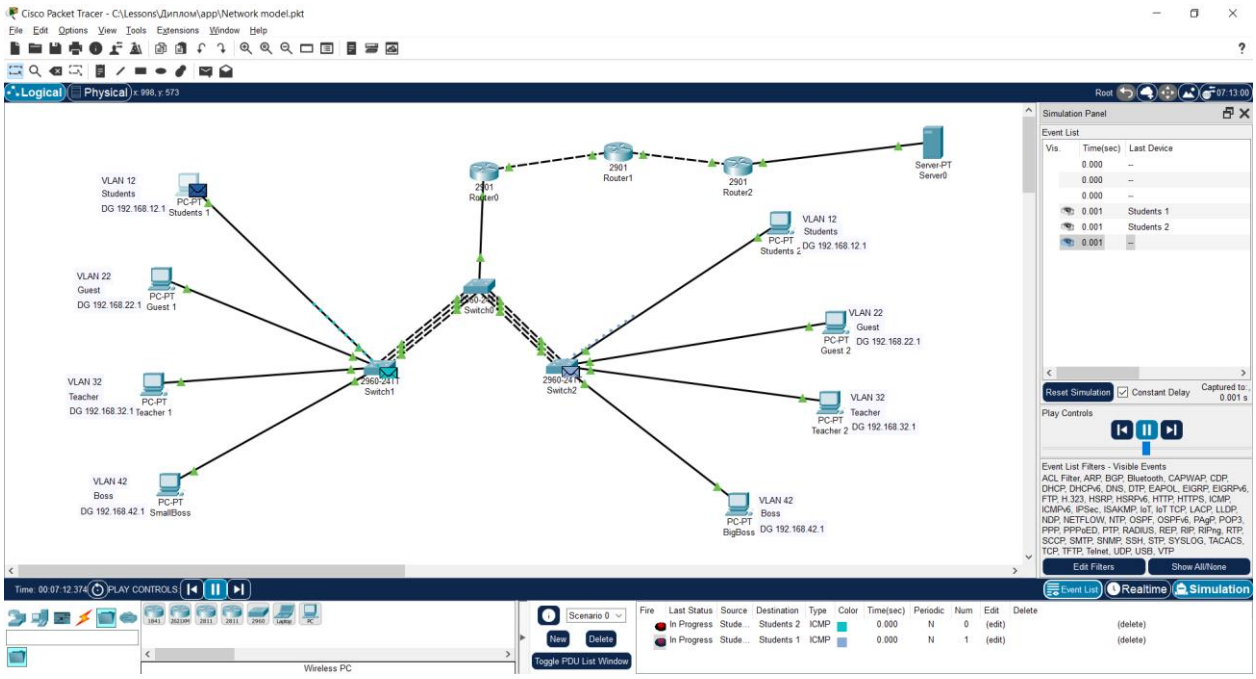


Рисунок 3.4 – Модель відправлення пакетів до іншого комп’ютера

3.2 Вибір та опис компонентів

3.2.1 Машрутизатор Cisco 2901

Cisco 2901 - це маршрутизатор серії ISR G2 від компанії Cisco. Він є елементом мережевого обладнання, що використовується для передачі даних між мережами. Маршрутизатор 2901 має високу продуктивність, забезпечує швидку передачу даних та має різноманітні можливості (рис. 3.2.1).

Рисунок 3.2.1 – Маршрутизатор 2901

У Cisco Packet Tracer можна побудувати модель мережі з маршрутизатором 2901. Для цього можна використовувати інтерфейс користувача програми та додавати мережеві пристрої та з'єднання між ними.

Для додавання маршрутизатора 2901 необхідно відкрити вкладку "Роутери" у бібліотеці пристроїв та перетягнути маршрутизатор 2901 на поле робочої області. Після цього можна виконати налаштування маршрутизатора за допомогою командного рядка або графічного інтерфейсу.

Маршрутизатор 2901 має чотири порти Gigabit Ethernet та два порти High-Speed WAN Interface Card (HWIC), які можуть бути використані для підключення до інших мереж та пристроїв. Також він має підтримку багатьох протоколів, таких як OSPF, BGP, EIGRP, RIP, IPv6 та інші.

Для додавання з'єднання до маршрутизатора 2901 необхідно відкрити вкладку "Кабелі" у бібліотеці пристроїв та перетягнути кабель на поле робочої області.

У Cisco Packet Tracer можна також використовувати різні інструменти для тестування та аналізу мережі, такі як Ping, Traceroute, Telnet та інші. Ці інструменти допомагають виявляти проблеми в мережі та швидко покращувати її роботу.

Для налаштування маршрутизатора 2901 можна використовувати графічний інтерфейс Cisco Configuration Professional (CCP) або командний рядок. За допомогою графічного інтерфейсу можна налаштувати мережеві інтерфейси, статичні та динамічні маршрути, фільтри пакетів, безпеку мережі та інші параметри.

Маршрутизатор 2901 підтримує багато функцій, які дозволяють забезпечити безпеку мережі та оптимізувати її роботу. Наприклад, можна налаштувати вхідні та вихідні списки контролю доступу (ACL), які дозволяють фільтрувати пакети на основі IP-адрес, номера порту, протоколу та інших параметрів. Також можна використовувати віртуальні приватні мережі (VPN), які забезпечують безпеку та конфіденційність передачі даних через мережу [8].

Крім того, маршрутизатор 2901 може бути використаний для підключення до хмарних служб та сервісів, таких як AWS, Azure, Google Cloud та інші. Для цього можна використовувати різні протоколи та сервіси, такі як VPN, MPLS, VRF та інші.

У загальному, маршрутизатор 2901 є потужним та надійним мережевим пристроєм, який може бути використаний для побудови різноманітних мереж, від малих офісів до великих корпоративних мереж. Завдяки широким можливостям та високій продуктивності, він дозволяє забезпечити ефективну та безпечну роботу мережі.

3.2.2 Комутатор Cisco 2960-24TT

Комутатор 2960-24TT від Cisco є потужним мережевим пристроєм, який може бути використаний для побудови мережі з низькою витратою енергії та високою продуктивністю. Цей комутатор має 24 порти Fast Ethernet та 2 порти Gigabit Ethernet, що дозволяє підключити до нього багато різних пристроїв, таких як комп'ютери, принтери, IP-телефони, маршрутизатори та інші (рис. 3.2.2).

Рисунок 3.2.2 – Комутатор Cisco 2960-24TT

Однією з головних переваг комутатора 2960-24TT є його підтримка технології Power over Ethernet (PoE), яка дозволяє живити підключені до

комутатора пристрої, такі як IP-телефони, точки доступу Wi-Fi та інші, через Ethernet-кабель. Також цей комутатор підтримує технологію Quality of Service (QoS), яка дозволяє регулювати пропускну здатність мережі та підтримувати якість обслуговування для різних типів трафіку.

У комутатора 2960-24TT є вбудовані функції безпеки, такі як список контролю доступу (ACL), який дозволяє фільтрувати пакети на основі різних параметрів, а також функція Private VLAN, яка дозволяє створювати віртуальні мережі з різним рівнем доступу до ресурсів мережі.

Комутатор 2960-24TT можна налаштувати за допомогою командного рядка або графічного інтерфейсу Cisco Network Assistant (CNA). Графічний інтерфейс CNA дозволяє налаштувати мережеві інтерфейси, VLAN, безпеку мережі та інші параметри комутатора. Командний рядок дозволяє виконувати більш розширені налаштування та налаштовувати комутатор через телнет або консольний порт.

Комутатор 2960-24TT підтримує також протоколи Spanning Tree Protocol (STP) та Rapid Spanning Tree Protocol (RSTP), які дозволяють уникнути петель у мережі та запобігти затримкам та втратам пакетів. Крім того, він підтримує протоколи динамічної маршрутизації, такі як Routing Information Protocol (RIP) та Open Shortest Path First (OSPF), які дозволяють комутатору автоматично визначати шлях до інших мереж у мережі.

Комутатор 2960-24TT підтримує різні типи VLAN, включаючи стандартні VLAN, VLAN згідно з протоколом 802.1Q та VLAN згідно з протоколом VLAN Trunking Protocol (VTP), які дозволяють розділити мережу на логічні сегменти з різними рівнями доступу до ресурсів мережі.

Комутатор 2960-24TT також має можливість налаштування портів з різними параметрами, такими як швидкість передачі даних, дуплексний режим та інші. Також можна використовувати комутатор як агрегаційний пристрій, що дозволяє підключати до нього інші комутатори та розподіляти мережевий трафік [9].

У загальному, комутатор 2960-24TT є потужним та надійним мережевим пристроєм, який може бути використаний для побудови різних типів мереж, від

невеликих офісних мереж до більших корпоративних мереж з високим рівнем продуктивності та безпеки.

3.2.3 Віртуальний комп’ютер PC-PT

PC-PT (PC Packet Tracer) - це віртуальний комп'ютер, який може бути використаний в Cisco Packet Tracer для симуляції поведінки реального комп'ютера в мережі. Він працює на базі операційної системи Windows і має можливості, схожі на звичайний персональний комп'ютер. За допомогою PC-PT можна перевірити роботу мережі, налаштувати мережеві підключення та налаштувати додаткові сервіси, які зазвичай використовуються в мережевому середовищі (рис. 3.2.3).

Рисунок 3.2.3 – Віртуальний комп’ютер PC-PT

PC-PT може бути доданий до мережі в Cisco Packet Tracer, перетягнувши відповідний символ з панелі інструментів на дошку симуляції. Після цього можна налаштувати параметри мережевих підключень, такі як IP-адресу, маску підмережі та шлюз за замовчуванням, щоб забезпечити правильну роботу комп'ютера в мережі [10].

У PC-PT можна встановлювати програмне забезпечення, що зазвичай використовується на персональних комп'ютерах, таке як браузери, офісні пакети, програми для спілкування тощо. Крім того, PC-PT може бути налаштований для підключення до мережі Інтернет або віртуальної часткової мережі (VPN).

За допомогою PC-PT можна також проводити діагностику мережі, виконувати тестування мережевих сервісів та перевіряти роботу мережевих пристроїв, таких як маршрутизатори, комутатори, файрволи тощо.

Узагалі, PC-PT дозволяє користувачам виконувати широкий спектр завдань, що пов'язані з налаштуванням та тестуванням мережевих сервісів в Cisco Packet Tracer.

3.2.4 Віртуальний комп’ютер Server-PT

Server-PT (Server Packet Tracer) - це віртуальний сервер, який може бути використаний в Cisco Packet Tracer для симуляції поведінки реального сервера в мережі. Він працює на базі операційної системи Windows Server і має можливості, схожі на звичайний сервер (рис. 3.2.4).

Рисунок 3.2.4 - Віртуальний комп’ютер PC-PT

Server-PT може бути доданий до мережі в Cisco Packet Tracer, перетягнувши відповідний символ з панелі інструментів на дошку симуляції. Після цього можна налаштувати параметри мережевих підключень, такі як IP-адресу, маску підмережі та шлюз за замовчуванням, щоб забезпечити правильну роботу сервера в мережі [11].

Server-PT має можливості, що дозволяють йому виконувати різні функції, такі як збереження та обробка даних, розподілення ресурсів, надання мережевих служб, які зазвичай використовуються в мережевому середовищі. Наприклад, Server-PT може бути налаштований для надання служби файлового сервера, друку

або веб-сервера. Також можна налаштувати Server-PT як DHCP-сервер, DNS-сервер або AD-сервер (Active Directory).

Server-PT може бути налаштований для забезпечення безпеки мережі, наприклад, для налаштування файрвола або VPN-сервера. Він також може використовуватися для моніторингу та діагностики мережі, або для збереження резервних копій даних.

Server-PT може бути підключений до інших мережевих пристроїв, таких як маршрутизатори або комутатори, для забезпечення доступу користувачів до мережевих ресурсів, а також для забезпечення взаємодії між різними мережевими службами.

Узагалі, Server-PT дозволяє користувачам виконувати широкий спектр завдань, що пов'язаний з налаштуванням та тестуванням різних мережевих служб в контрольованому середовищі Cisco Packet Tracer. Це дозволяє студентам, мережним адміністраторам та іншим зацікавленим особам отримати практичні знання та навички в роботі з серверами та мережевими службами, що є важливим у сучасному світі технологій.

Налаштування Server-PT досить просте та інтуїтивно зрозуміле. Можна встановити різні опції налаштування, такі як тип служби, яку необхідно надавати, мережеві налаштування, такі як IP-адреса та маска підмережі, а також параметри безпеки, такі як файрвол або VPN-з'єднання.

Після налаштування Server-PT можна почати тестування його функцій, щоб переконатися, що він працює правильно. Для цього можна використовувати інші пристрої в мережі, такі як ПК або ноутбуки, щоб підключитися до Server-PT та отримати доступ до наданих ним мережевих служб. Можна також використовувати різні інструменти в Cisco Packet Tracer, такі як Wireshark або Packet Tracer Analyzer, для аналізу мережевого трафіку та виявлення будь-яких помилок чи проблем у налаштуванні мережі.

Узагалі, Server-PT є корисним інструментом для вивчення та розуміння роботи серверів та мережевих служб, а також для практичного використання цих

знань в мережевих середовищах. Він дозволяє студентам та мережним адміністраторам відтворювати різні сценарії роботи серверів та мережевих служб та відстежувати їх роботу в безпечному та контрольованому середовищі [12].

3.3 Програмне забезпечення кіберзахисту мережі підприємства на Java

3.3.1 Використання міжсерверної аутентифікації на Java

Міжсерверна аутентифікація (англ. cross-server authentication) - це процес аутентифікації користувача, який дозволяє йому отримувати доступ до різних серверів без потреби повторного введення облікових даних. Цей процес використовує спільну базу даних користувачів, що знаходиться на центральному сервері.

У Java існують різні методи реалізації міжсерверної аутентифікації. Одним з найпоширеніших є використання технології Java Authentication and Authorization Service (JAAS).

JAAS - це фреймворк для аутентифікації та авторизації користувачів в Java. Він надає можливість використовувати різні механізми аутентифікації, включаючи базу даних, LDAP, Kerberos, SSL і багато інших.

Для реалізації міжсерверної аутентифікації з використанням JAAS потрібно:

1. Створити спільну базу даних користувачів на центральному сервері і налаштувати механізм аутентифікації.
2. Реалізувати JAAS-модуль для аутентифікації користувачів на локальному сервері. Для цього потрібно створити клас, який реалізує інтерфейс javax.security.auth.spi.LoginModule.
3. Підключити JAAS-модуль до програми, що запускається на локальному сервері. Для цього потрібно налаштувати конфігураційний файл login.conf, додавши в нього запис про JAAS-модуль.

4. Підключити програму на локальному сервері до центрального сервера, використовуючи механізм RMI (Remote Method Invocation).

5. Виконати аутентифікацію користувача на центральному сервері і передати його облікові дані на локальний сервер.

6. При повторному зверненні до іншого сервера, на локальному сервері буде проведена аутентифіція користувача на основі отриманих облікових даних від центрального сервера, що дозволить йому отримати доступ до потрібного ресурсу без додаткового введення облікових даних.

Основні кроки міжсерверної аутентифікації на Java з використанням JAAS:

1. Створення спільної бази даних користувачів на центральному сервері. Налаштування механізму аутентифікації на базі даних.

2. Створення JAAS-модуля на локальному сервері для аутентифікації користувачів. Реалізація класу, що імплементує інтерфейс javax.security.auth.spi.LoginModule.

3. Налаштування конфігураційного файлу login.conf на локальному сервері, додавання в нього запису про JAAS-модуль.

4. Підключення програми на локальному сервері до центрального сервера за допомогою механізму RMI.

5. Аутентифікація користувача на центральному сервері та передача його облікових даних на локальний сервер.

6. При повторному зверненні до іншого сервера, на локальному сервері проводиться аутентифікація користувача на основі отриманих облікових даних від центрального сервера.

Використання міжсерверної аутентифікації з використанням JAAS дозволяє підвищити рівень безпеки мережевих додатків та зменшити навантаження на користувачів, оскільки вони не повинні постійно вводити свої облікові дані для доступу до різних ресурсів на серверах [13].

Програмний код для міжсерверної аутентифікації на Java з використанням токенів JWT:

```
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import java.security.Key;
import java.util.Date;

public class JWTAuthentication {

private static final long EXPIRATION_TIME = 864_000_000; // 10 days in milliseconds
private static final Key SECRET_KEY = Keys.secretKeyFor(io.jsonwebtoken.SignatureAlgorithm.HS256);
    public static void main(String[] args) {
        String username = "user";
        // Create JWT token
        String jwt = Jwts.builder()
                .setSubject(username)
                .setExpiration(new Date(System.currentTimeMillis() + EXPIRATION_TIME))
                .signWith(SECRET_KEY)
                .compact();

        // Verify JWT token
        Claims claims = Jwts.parserBuilder()
                .setSigningKey(SECRET_KEY)
                .build()
                .parseClaimsJws(jwt)
```

```
                .getBody();

        System.out.println("Authentication successful! User: " + claims.getSubject());
        }
    }
```

3.3.2 Використання міжсерверної аутентифікації на Java

Ідентифікація та аутентифікація є ключовими аспектами багатьох програмних рішень. Нижче наведено декілька способів, які можна використовувати для реалізації ідентифікації та аутентифікації на Java [14]:

1. Використання бази даних користувачів і паролів: Створення таблиці бази даних для зберігання даних користувача, включаючи ім'я користувача та хеш пароля. Коли користувач входить до системи, введене ім'я користувача та пароль перевіряються на відповідність значенням, збереженим в базі даних.
2. Використання веб-служб: Створення веб-служб, які надають інтерфейс для аутентифікації користувачів. Зазвичай такі веб-служби використовують токени аутентифікації для перевірки того, що запит користувача є дійсним.
3. Використання бібліотек: Використання готових бібліотек, таких як Spring Security або Apache Shiro, для реалізації ідентифікації та аутентифікації. Ці бібліотеки надають високорівневі API для обробки аутентифікації користувачів та управління дозволами.
4. Використання стандартів і протоколів: Використання стандартів і протоколів, таких як OAuth або OpenID Connect, для реалізації ідентифікації та аутентифікації. Ці протоколи надають механізми для взаємодії з іншими системами для аутентифікації користувачів.
5. Використання мікросервісів: Використання мікросервісної архітектури, де кожен мікросервіс відповідає за певну функціональність, включаючи аутентифікацію та ідентифікацію. Кожен мікросервіс може мати свою власну базу

даних користувачів та паролів, або використовувати зовнішні сервіси для аутентифікації користувачів.

6. Використання двофакторної аутентифікації: Використання двофакторної аутентифікації, де користувач повинен надати не тільки пароль, але й додатковий код, зазвичай згенерований мобільним додатком або штучним інтелектом.

7. Використання шифрування: Використання шифрування для збереження конфіденційної інформації, такої як паролі, в базі даних [15].

Нижче наведено приклад коду на Java для створення об'єкту користувача з ім'ям та паролем, а також перевірки введеного пароля користувача зі збереженим паролем в базі даних:

```
public class User {
    private String username;
    private String passwordHash;

    public User(String username, String password) {
        this.username = username;
        this.passwordHash = hashPassword(password);
    }

    public boolean checkPassword(String password) {
        return passwordHash.equals(hashPassword(password));
    }

    private String hashPassword(String password) {
        // код для хешування пароля
    }
}
```

```
public class UserDatabase {
    private Map<String, User> users = new HashMap<>();

    public void addUser(User user) {
        users.put(user.getUsername(), user);
    }

    public User getUser(String username) {
        return users.get(username);
    }
}

public class Main {
    public static void main(String[] args) {
        UserDatabase database = new UserDatabase();
        database.addUser(new User("Alice", "pa$$w0rd"));

        String username = "Alice";
        String password = "pa$$w0rd";

        User user = database.getUser(username);
        if (user != null && user.checkPassword(password)) {
            System.out.println("Authentication successful!");
        } else {
            System.out.println("Authentication failed.");
        }
    }
}
```

У цьому прикладі створюється об'єкт користувача з ім'ям "Alice" та паролем "pa$$w0rd". Об'єкт користувача додається до бази даних користувачів. При введенні імені користувача та пароля, код перевіряє, чи існує користувач з введеним іменем, та перевіряє правильність [16].

3.3.3 Захист від DDOS атаки

Використання фільтрів сервлетів (Servlet Filters) для обмеження запитів від клієнтів за певний період часу.

Нижче наведено приклад коду, який дозволяє обмежити кількість запитів від клієнта до 10 за 1 хвилину:

```
public class RequestFilter implements Filter {
    private final Map<String, AtomicInteger> requests = new
ConcurrentHashMap<>();
    private final int MAX_REQUESTS = 10;
    private final int REQUEST_EXPIRATION_TIME = 60000; // 1 minute in
milliseconds

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
            throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) request;
        String clientIpAddress = getClientIpAddress(httpRequest);
        AtomicInteger requestCount = requests.get(clientIpAddress);
        if (requestCount == null) {
            requestCount = new AtomicInteger(0);
            requests.put(clientIpAddress, requestCount);
        }
        if (requestCount.incrementAndGet() > MAX_REQUESTS) {
            response.getWriter().write("Too many requests from the same IP
address.");
            response.setStatus(HttpStatus.TOO_MANY_REQUESTS.value());
            return;
        }
```

```
            new Timer().schedule(new TimerTask() {
                @Override
                public void run() {
                    requests.remove(clientIpAddress);
                }
            }, REQUEST_EXPIRATION_TIME);
            chain.doFilter(request, response);
        }

        private String getClientIpAddress(HttpServletRequest request) {
            String xForwardedForHeader = request.getHeader("X-Forwarded-For");
            if (xForwardedForHeader == null) {
                return request.getRemoteAddr();
            }
            return xForwardedForHeader.split(",")[0];
        }
    }
```

Використання кешування результатів обробки запитів (response caching) за допомогою бібліотеки Ehcache.

Використання кешування дозволяє зменшити кількість запитів до сервера та зменшити час відповіді [17]. Нижче наведено приклад коду для кешування результатів запитів за допомогою бібліотеки Ehcache:

```
    public class ResponseCacheFilter implements Filter {
        private CacheManager cacheManager;

        @Override
        public void init(FilterConfig filterConfig) throws ServletException {
            cacheManager = CacheManagerBuilder.newCacheManagerBuilder()
                    .withCache("responseCache",

CacheConfigurationBuilder.newCacheConfigurationBuilder(Object.class, Object.class,
                            ResourcePoolsBuilder.heap(100))

.withExpiry(ExpiryPolicyBuilder.timeToIdleExpiration(Duration.ofSeconds(30)))
                            .build())
                    .build(true);
        }
```

```
        @Override
        public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
                throws IOException, ServletException {
            String key = request.getParameter("key");
            Cache<Object, Object> responseCache =
cacheManager.getCache("responseCache", Object.class, Object.class);
            Object cachedResponse = responseCache.get(key);
            if (cachedResponse != null) {
                HttpServletResponse httpServletResponse = (HttpServletResponse)
response;
                httpServletResponse.getOutputStream().write((byte[]) cachedResponse);
                return;
            }
            CachedResponseWrapper wrapperResponse = new
CachedResponseWrapper((HttpServletResponse) response);
            chain.doFilter(request, wrapperResponse);
            byte[] content = wrapperResponse.getCachedContent();
            responseCache.put(key, content);
            response.getOutputStream().write(content);
        }

        @Override
        public void destroy() {
            cacheManager.close();
        }
    }

    public class CachedResponseWrapper extends HttpServletResponseWrapper {
        private ByteArrayOutputStream cachedOutputStream;
        private ServletOutputStream servletOutputStream;

        public CachedResponseWrapper(HttpServletResponse response) throws
IOException {
            super(response);
            cachedOutputStream = new ByteArrayOutputStream();
            servletOutputStream = new ServletOutputStream() {
                @Override
                public void write(int b) throws IOException {
```

```
                cachedOutputStream.write(b);
            }
        };
    }

    @Override
    public ServletOutputStream getOutputStream() throws IOException {
        return servletOutputStream;
    }

    public byte[] getCachedContent() {
        return cachedOutputStream.toByteArray();
    }
}
```

За допомогою WebSocket-з'єднання клієнт може відправляти повідомлення на сервер та отримувати відповіді без необхідності відправляти окремі запити на сервер для кожного повідомлення. Це може допомогти зменшити навантаження на сервер та покращити продуктивність додатка [18].

```
@ServerEndpoint("/websocket")
public class WebSocketServer {
    private static Set<Session> sessions = Collections.synchronizedSet(new
HashSet<>());

    @OnOpen
    public void onOpen(Session session) {
        sessions.add(session);
    }

    @OnMessage
    public void onMessage(String message, Session session) throws IOException {
        // обробка повідомлення від клієнта
        String response = processMessage(message);
        session.getBasicRemote().sendText(response);
    }

    @OnClose
    public void onClose(Session session) {
        sessions.remove(session);
```

```
    }

    private String processMessage(String message) {
        // обробка повідомлення
        return "Response to message: " + message;
    }
}
```

3.3.4 Шифрування та розшифрування даних за допомогою алгоритму AES

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.util.Arrays;
import java.util.Base64;

public class EncryptionUtils {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes(StandardCharsets.UTF_8);
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static String encrypt(String strToEncrypt, String secret) {
        try {
            setKey(secret);
```

```
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardC
harsets.UTF_8)));
        } catch (Exception e) {
            System.out.println("Error while encrypting: " + e.toString());
        }
        return null;
    }

    public static String decrypt(String strToDecrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return                                                    new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        } catch (Exception e) {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }
}
```

Приклад використання:

```
String secretKey = "mySecretKey";
String originalString = "Hello world";

String encryptedString = EncryptionUtils.encrypt(originalString, secretKey);
String decryptedString = EncryptionUtils.decrypt(encryptedString, secretKey);

System.out.println(originalString);
System.out.println(encryptedString);
System.out.println(decryptedString);
```

Цей код шифрує та розшифровує рядок за допомогою алгоритму AES з використанням ключа **mySecretKey**. Зашифрований рядок виводиться на екран, а потім розшифровується знову і виводиться на екран [19].

3.4 Демонстрація виконання програми

У ході розробки дипломного проекту, була розроблена система захисту підприємства від кібератак. У цьому розділі буде наведено програмний код, який реалізує захист від DDOS-атак, у вигляді обмеження кількості запитів на сервер з перевіркою на ip-адрес, та користувача, який здійснював запит. Також було реалізовано доступ до даних за допомогою авторизації, таким чином щоб до певних даних мали доступ визначені користувачі через їх «ролі» (рис. 3.4.1 – 3.4.7).

```java
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "Hello, World!";
    }

    @GetMapping("/user1")
    public String user1() {
        return "Hello, User 1!";
    }

    @GetMapping("/user2")
    public String user2() {
        return "Hello, User 2!";
    }
}
```

Рисунок 3.4.1 – Створення програмного коду HomeController

Анотація **@RestController** перед класом вказує, що цей клас є контролером, який повертає дані у вигляді JSON або XML.

Три методи-контролери містять анотацію **@GetMapping**, яка вказує, що методи будуть відповідати на запити HTTP GET на відповідні шляхи. Перший метод **home()** буде відповідати на запити на кореневий шлях / та повертатиме рядок "Hello, World!". Другий метод **user1()** буде відповідати на запити на шлях **/user1** та повертатиме рядок "Hello, User 1!". Третій метод **user2()** буде відповідати на запити на шлях **/user2** та повертатиме рядок "Hello, User 2!".

Таким чином, при запуску цього коду на сервері, веб-додаток буде доступний за URL-адресою /, **/user1** та **/user2**. Кожен з цих URL-адрес поверне рядок з відповідним текстом.

```java
package com.example.demo;
import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.atomic.AtomicInteger;
public class RateLimitFilter implements Filter {
    private final ConcurrentHashMap<String, AtomicInteger> ipCounter = new ConcurrentHashMap<>();
    private final ConcurrentHashMap<String, AtomicInteger> userCounter = new ConcurrentHashMap<>();
    private final int MAX_REQUESTS_PER_MINUTE = 5;
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        HttpServletRequest httpServletRequest = (HttpServletRequest) request;
        String clientIp = httpServletRequest.getRemoteAddr();
        String user = httpServletRequest.getRemoteUser();
        ipCounter.putIfAbsent(clientIp, new AtomicInteger( initialValue: 0)); userCounter.putIfAbsent(user, new AtomicInteger( initialValue: 0));
        if (ipCounter.get(clientIp).incrementAndGet() > MAX_REQUESTS_PER_MINUTE || userCounter.get(user).incrementAndGet() > MAX_REQUESTS_PER_MINUTE) {
            HttpServletResponse httpServletResponse = (HttpServletResponse) response;
            httpServletResponse.setStatus(429);
            httpServletResponse.getWriter().write( s: "Too many requests per minute");
            return;
        }
        chain.doFilter(request, response);
    }
    @Override
    public void init(FilterConfig filterConfig) {}
    @Override
    public void destroy() {}
}
```

Рисунок 3.4.2 - Створення програмного коду RateLimitFilter

Даний код представляє собою фільтр для обмеження швидкості запитів до веб-сервера. Конкретно, цей фільтр встановлює максимальну кількість запитів, які користувач або IP-адреса може відправити до веб-сервера за одну хвилину.

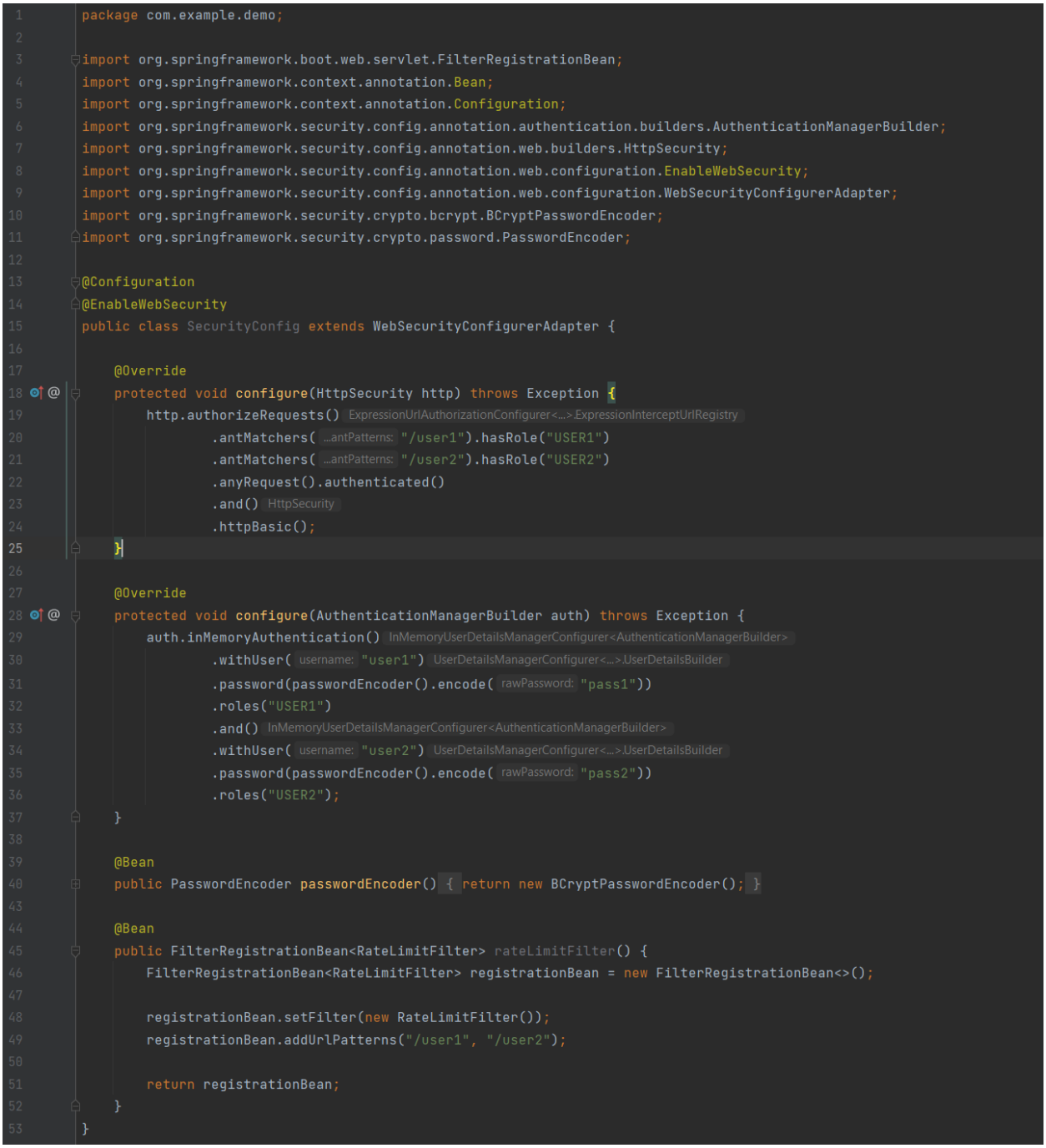
Клас **RateLimitFilter** реалізує інтерфейс **Filter**, тому він може бути використаний як фільтр для HTTP-запитів.

Фільтр зберігає лічильники запитів для кожного користувача та IP-адреси в ConcurrentHashMap, яка є потокобезпечною. Лічильники ініціалізуються зі значенням 0. Фільтр знаходить IP-адресу та користувача, що відправив запит, та

збільшує лічильники відповідного користувача та IP-адреси. Якщо кількість запитів перевищує максимальну допустиму кількість запитів на хвилину, фільтр повертає статус відповіді **429 Too Many Requests** та повідомлення "Too many requests per minute".

Якщо кількість запитів не перевищує максимальну допустиму кількість запитів, фільтр передає запит далі за ланцюжком фільтрів за допомогою методу **chain.doFilter()**.

Фільтр ініціалізується за допомогою методу **init()**, який виконується один раз при запуску фільтра, та очищається за допомогою методу **destroy()**, який виконується один раз при зупинці фільтра [20].



```java
package com.example.demo;

import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
                .antMatchers( ...antPatterns: "/user1").hasRole("USER1")
                .antMatchers( ...antPatterns: "/user2").hasRole("USER2")
                .anyRequest().authenticated()
                .and() HttpSecurity
                .httpBasic();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication() InMemoryUserDetailsManagerConfigurer<AuthenticationManagerBuilder>
                .withUser( username: "user1") UserDetailsManagerConfigurer<...>.UserDetailsBuilder
                .password(passwordEncoder().encode( rawPassword: "pass1"))
                .roles("USER1")
                .and() InMemoryUserDetailsManagerConfigurer<AuthenticationManagerBuilder>
                .withUser( username: "user2") UserDetailsManagerConfigurer<...>.UserDetailsBuilder
                .password(passwordEncoder().encode( rawPassword: "pass2"))
                .roles("USER2");
    }

    @Bean
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Bean
    public FilterRegistrationBean<RateLimitFilter> rateLimitFilter() {
        FilterRegistrationBean<RateLimitFilter> registrationBean = new FilterRegistrationBean<>();

        registrationBean.setFilter(new RateLimitFilter());
        registrationBean.addUrlPatterns("/user1", "/user2");

        return registrationBean;
    }
}
```

Рисунок 3.4.3 - Створення програмного коду SecurityConfig

Даний код відповідає за налаштування захисту доступу до сторінок веб-додатку та встановлення обмеження на кількість запитів, що можуть бути зроблені користувачем за одну хвилину.

Клас **SecurityConfig** відповідає за конфігурування налаштувань безпеки. Метод **configure(HttpSecurity http)** встановлює обмеження на доступ користувачів до сторінок в залежності від їх ролів, наприклад, доступ до **/user1** мають лише

користувачі з роллю **USER1**, а доступ до **/user2** мають лише користувачі з роллю **USER2**. Крім того, будь-який інший запит потребує автентифікації.

Метод **configure(AuthenticationManagerBuilder auth)** встановлює користувачів та їхні паролі для аутентифікації.

Клас **RateLimitFilter** відповідає за встановлення обмеження на кількість запитів, що можуть бути зроблені користувачем за одну хвилину. Він використовує ConcurrentHashMap для зберігання лічильників запитів користувачів та ініціалізує лічильники для кожного користувача та його IP-адреси. Якщо кількість запитів перевищує дозволену межу за одну хвилину, то фільтр повертає відповідь з кодом статусу 429 ("Too Many Requests") і повідомленням про занадто багато запитів за хвилину [21].

Метод **rateLimitFilter()** конфігурує **RateLimitFilter** і встановлює його для обробки запитів, які мають шлях **/user1** або **/user2**.

```
1  package com.example.demo;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class Application {
8
9      public static void main(String[] args) {
10         SpringApplication.run(Application.class, args);
11     }
12 }
```

Рисунок 3.4.4 - Створення програмного коду Application

Цей код відповідає за запуск Spring Boot додатку. Він містить основний метод main(), який викликає метод run() класу SpringApplication з параметрами, що вказують на головний клас додатку та аргументи командного рядка. Цей клас містить анотацію @SpringBootApplication, яка включає в себе декілька інших анотацій, необхідних для автоматичного конфігурування Spring додатку, таких як @Configuration, @EnableAutoConfiguration та @ComponentScan [22].

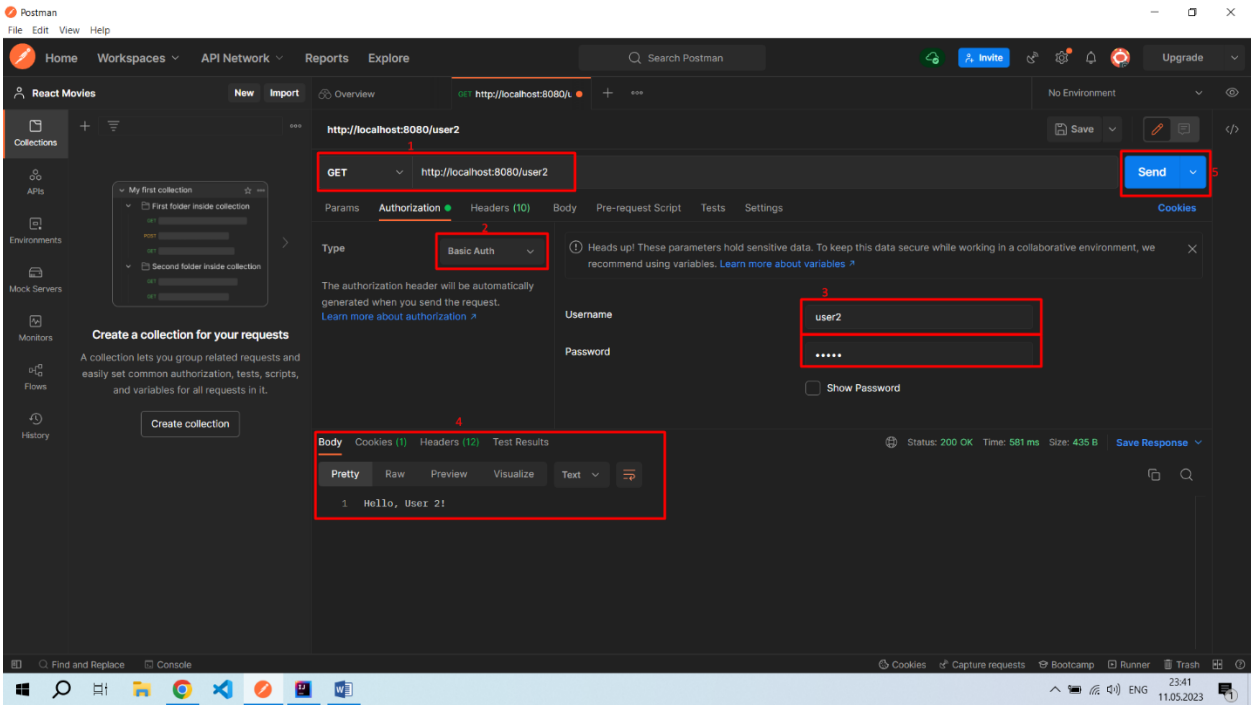


Рисунок 3.4.5 – Результат виконання програми

На даному рисунку зображено результат успішного виконання програми. На рисунку під підписом 1, зображено ввід url, по якій ми отримує доступ до даних.

На рисунку під підписом 2, зображено що ми використовуємо автентифікацію, за допомогою якої передається ім’я користувача і пароль (зображено на рисунку під підписом 3) у вигляді тексту через HTTP-запит. На рисунку під підписом 4 ми отримує відповідь від серверу з успішною авторизацією та доступом до даних.

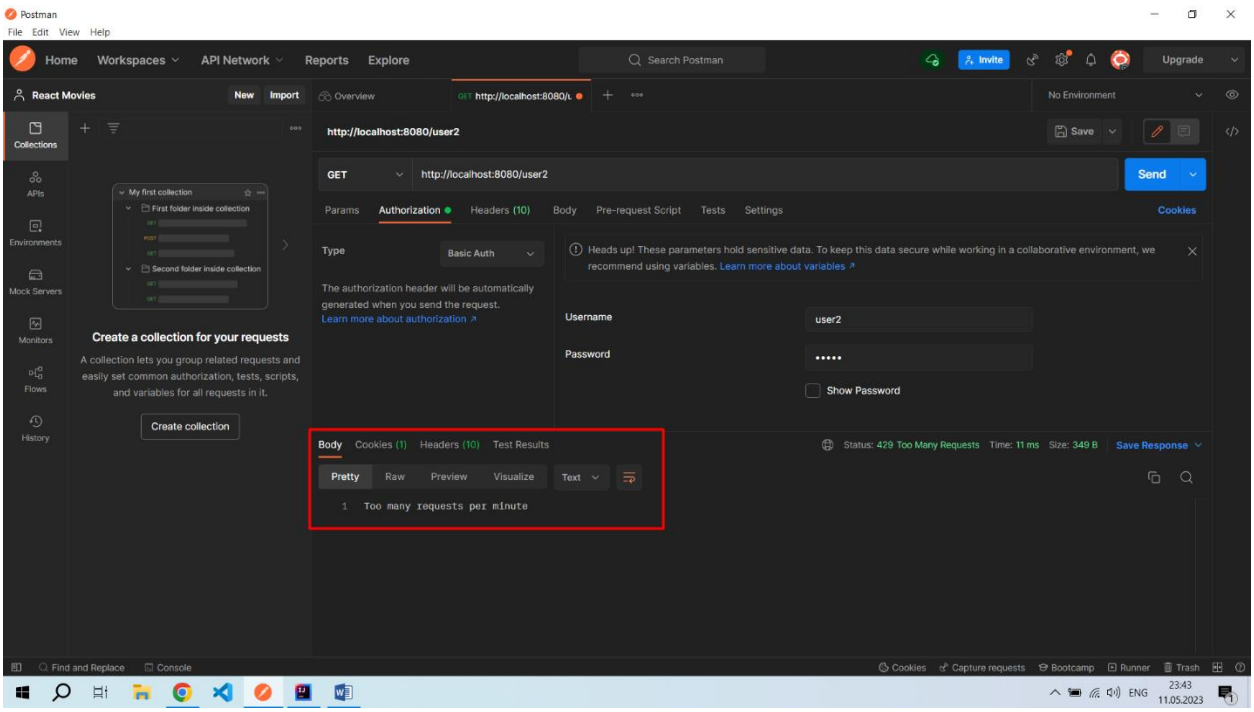


Рисунок 3.4.6 – Результат виконання програми зі статусом «Перевищено ліміт запитів за хвилину»

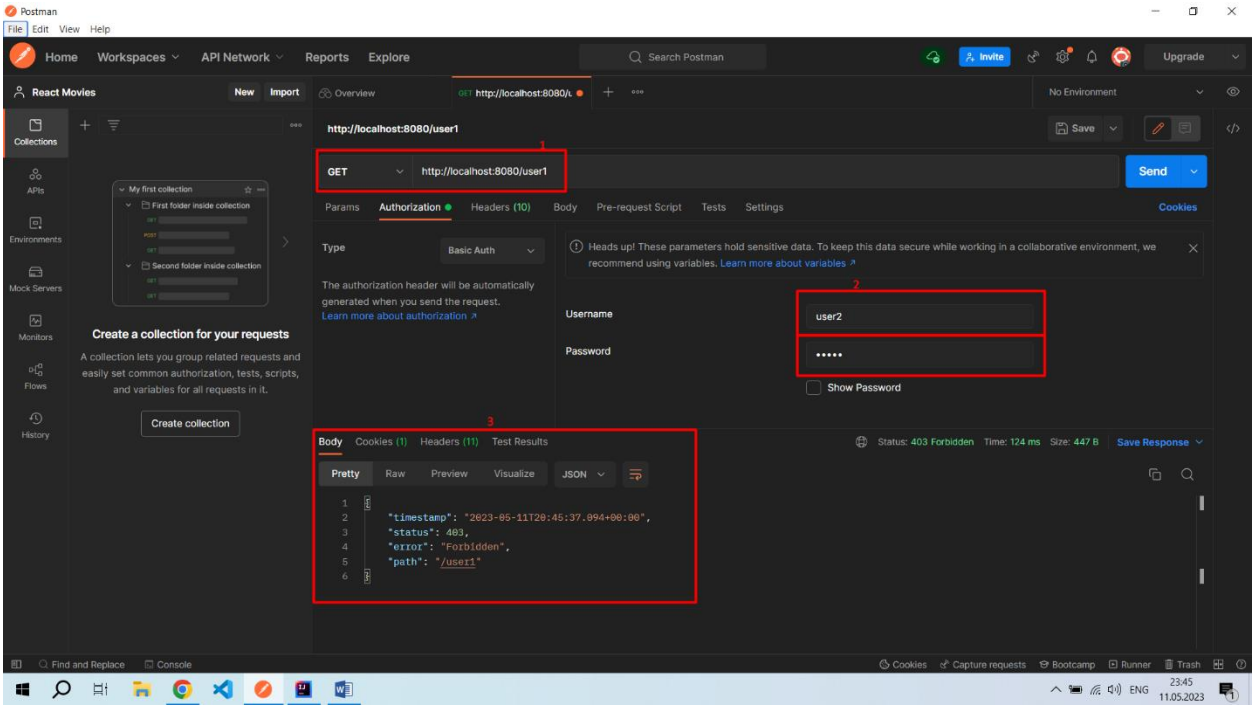


Рисунок 3.4.7 – Результат виконання програми зі статусом «Заборонено»

Висновки до розділу 3

Побудова моделі мережі підприємства є надзвичайно важливим етапом при створенні ефективної та безпечної інфраструктури. Для цього можна використовувати різноманітні інструменти та технології, такі як Cisco Packet Tracer та програмне забезпечення кіберзахисту мережі на Java.

Cisco Packet Tracer є програмою для моделювання мереж, яка дозволяє створювати, налаштовувати та відлагоджувати різноманітні мережеві пристрої та топології. Цей інструмент дозволяє досліджувати різні сценарії роботи мереж, перевіряти пропускну здатність та стійкість до відмов.

Однак, на даний момент, розробка імітаційних моделей мережі вже не є достатньою для забезпечення повної безпеки мережі. Тому, важливим етапом при побудові моделі мережі є використання програмного забезпечення кіберзахисту.

Програмне забезпечення кіберзахисту мережі на Java дозволяє забезпечити повну безпеку мережі та захистити її від різних видів кібератак. Це можливо завдяки використанню різноманітних методів захисту, таких як брандмауери, антивірусне програмне забезпечення, інструменти моніторингу мережі та інші.

Узагальнюючи, побудова моделі мережі підприємства за допомогою Cisco Packet Tracer є важливим етапом у створенні ефективної та безпечної інфраструктури. Однак, для забезпечення повної безпеки мережі необхідно використовувати різноманітні програмні засоби кіберзахисту, зокрема програмне забезпечення кіберзахисту мережі на Java.

Це дозволить забезпечити необхідний рівень безпеки мережі та захистити її від різних видів кіберзагроз. Належне поєднання цих двох інструментів дозволяє створити ефективну та безпечну мережеву інфраструктуру, яка забезпечить надійну роботу підприємства та захистить його від можливих кібернападів.

Програмне забезпечення кіберзахисту мережі на Java зазвичай має вбудовані механізми моніторингу та аналізу мережевого трафіку, які дозволяють розпізнавати аномальні підключення та зловмисні дії в мережі. Брандмауери та

інші інструменти забезпечують контроль над доступом до мережі та захист від можливих атак ззовні.

Також важливо враховувати особливості конкретної мережі та її функціональні потреби при виборі програмного забезпечення кіберзахисту. Наприклад, для мереж з високими вимогами до швидкості передачі даних можуть бути необхідні спеціальні інструменти для оптимізації мережевої пропускної здатності.

У підсумку, побудова моделі мережі підприємства та використання програмного забезпечення кіберзахисту на Java є надзвичайно важливим етапом у створенні ефективної та безпечної мережевої інфраструктури. Правильний вибір інструментів та їх належна настройка дозволяють забезпечити надійну роботу мережі та захистити її від можливих кіберзагроз.

ЗАГАЛЬНІ ВИСНОВКИ

Під час виконання дипломної роботи було створено кіберстійку мультисервісну корпоративну мережу підприємства. Дана мережа забезпечує високий рівень захисту від кіберзагроз, таких як хакерські атаки, віруси, фішинг тощо. Вона здатна швидко виявляти та реагувати на зловмисну діяльність і мінімізувати можливі шкоди. Кіберстійка мережа має важливе значення для захисту конфіденційної інформації та збереження надійності роботи систем..

В першому розділі було розглянуто важливі аспекти забезпечення кібербезпеки мультисервісної мережі підприємства, такі як огляд комп'ютерних мереж, вибір гібридної топології та застосування мережевих файрволів. Ці аспекти дозволяють виявляти та блокувати загрози, контролювати доступ до мережевих ресурсів та забезпечувати безпеку даних. Забезпечення кібербезпеки є важливою задачею для будь-якої компанії, і використання заходів безпеки є важливою складовою успіху та стійкості.

Окрім огляду комп'ютерних систем та вибору гібридної топології мережі, у цьому розділі також було розглянуто питання захисту від DDoS-атак, які є однією з найбільш поширених та небезпечних загроз для кібербезпеки. Для запобігання DDoS-атакам можуть використовуватись різноманітні методи, такі як виявлення та блокування шкідливого трафіку, розподілене збереження даних, а також використання веб-безпеки.

У другому розділі було розглянуто основні елементи мережі - комутатори та маршрутизатори, які забезпечують передачу даних та маршрутизацію мережевого трафіку. Наголошено на важливості дотримання заходів кібербезпеки при використанні цих пристроїв, оскільки вони можуть бути скомпрометовані зловмисниками та використані для атак на мережу. Вказано на необхідність застосування захисних механізмів, таких як брандмауери, VPN-з'єднання та регулярне оновлення програмного забезпечення для забезпечення кібербезпеки. Далі описано важливість дотримання заходів кібербезпеки в програмуванні на мові Java, оскільки програми можуть містити вразливості, які можуть бути

використані зловмисниками для атак на систему. Вказано на використання безпечних функцій та бібліотек для забезпечення безпеки від більшості атак, а також на заходи для запобігання SQL-ін'єкціям та XSS-атакам. Узагальнюючи, підкреслено, що кібербезпека є важливою складовою будь-якої мережі та програмного забезпечення, а дотримання заходів кібербезпеки може допомогти запобігти атакам та забезпечити безпеку та захист даних в мережах та програмах.

У третьому розділі було описано важливість побудови моделі мережі підприємства для створення ефективної та безпечної інфраструктури, яку можна забезпечити за допомогою різних інструментів та технологій. Один з таких інструментів - Cisco Packet Tracer, який дозволяє створювати та налаштовувати мережеві пристрої та топології. Однак, для забезпечення повної безпеки мережі потрібно використовувати програмне забезпечення кіберзахисту мережі на Java, яке дозволяє захистити мережу від кібератак. Використання програмного забезпечення кіберзахисту мережі на Java допомагає забезпечити безпеку мережі за допомогою різних методів захисту, таких як брандмауери та антивірусне програмне забезпечення. В цілому, поєднання Cisco Packet Tracer та програмного забезпечення кіберзахисту мережі на Java допомагає створити ефективну та безпечну мережеву інфраструктуру для підприємства, яка дозволяє забезпечити його надійну роботу та захист від кібернападів.

Отже, під час виконання дипломної роботи було розроблено ефективну кіберстійку мультисервісну корпоративну мережу підприємства. Результатом дослідження є створення кіберстійкої мережі з використанням методів захисту від DDOS-атак, призначенням ролей, аутентифікацією та авторизацією на мові програмування Java. Моделювання цієї мережі дозволить підприємствам забезпечити безпеку і захист від кіберзагроз, що є надзвичайно важливим для їхнього успішного функціонування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Computer Networking: A Top-Down Approach, 7th Edition by James F. Kurose and Keith W. Ross. Pearson, 2016. 864 с.
2. TCP/IP Illustrated, Volume 1: The Protocols, 2nd Edition by W. Richard Stevens and Gary R. Wright. Addison-Wesley Professional, 1994. 600 с.
3. "Firewalls: Jumpstart for Network and Systems Administrators" - Michael Rash (2005), 352 с.
4. CCNA Routing and Switching Complete Study Guide: Exam 100-105, Exam 200-105, Exam 200-125, 2nd Edition by Todd Lammle. Sybex, 2016. 1136 с.
5. "Switching Basics and Intermediate Routing CCNA 3 Labs and Study Guide" - Allan Johnson (2006), 576 с.
6. "Cisco OSPF Command and Configuration Handbook" - William R. Parkhurst (2002), 552 с.
7. Cybersecurity and Cyberwar: What Everyone Needs to Know, Revised and Expanded Edition by P.W. Singer and Allan Friedman. Oxford University Press, 2018. 336 с.
8. "CCNA Routing and Switching 200-125 Official Cert Guide Library" - Wendell Odom (2016), 1600 с.
9. "Cisco ASA: All-in-One Firewall, IPS, and VPN Adaptive Security Appliance" - Jazib Frahim, Omar Santos, Andrew Ossipov (2014), 1248 с.
10. "Cisco LAN Switching Fundamentals" - David Barnes, Basir Sakandar (2004), 576 с.
11. "Implementing Cisco IP Switched Networks (SWITCH) Foundation Learning Guide: Foundation learning for SWITCH 300-115" - Richard Froom, Erum Frahim, Balaji Sivasubramanian (2015), 560 с.
12. "Cisco IOS in a Nutshell" - James Boney (2009), 860 с.
13. Computer Security: Principles and Practice, 3rd Edition by William Stallings and Lawrie Brown. Pearson, 2014. 848 с.
14. H. Deitel and P. Deitel, Java How to Program, 11th Edition. Pearson, 2017. 1232 с.

15. J. Bloch, Effective Java, 3rd Edition. Addison-Wesley Professional, 2018. 416 с.
16. B. Eckel, Thinking in Java, 4th Edition. Prentice Hall, 2006. 1150 с.
17. C. Horstmann and G. Cornell, Core Java, Volume II--Advanced Features, 11th Edition. Prentice Hall, 2018. 1152 с.
18. J. Gosling, B. Joy, G. Steele, and G. Bracha, The Java Language Specification, Java SE 8 Edition. Addison-Wesley Professional, 2014. 792 с.
19. Antonakakis M., Perdisci R., Dagon D., Lee W. Enabling fine-grained multi-perspective view of the internet malicious activities. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp. 241-256. IEEE Computer Society, 2010. 16 с.
20. Khattak A. M., Arshad J., Manzoor U., Naeem M., Javaid N. An Overview of DDoS Attacks and Defense Mechanisms in Cloud Computing. Journal of Cloud Computing: Advances, Systems and Applications, vol. 9, no. 1, pp. 1-17, 2020. 17 с.
21. Mirkovic J., Dietrich D., Dittrich D., Reiher P. Internet Denial of Service: Attack and Defense Mechanisms. 1st Ed. Prentice Hall, 2004. 464 с.
22. Nazari M., Zolfaghari M. H. Secure programming for web application against DDOS attack in Java environment. International Journal of Computer Science and Network Security, vol. 14, no. 3, pp. 87-92, 2014. 6 с.

ДОДАТОК А

Кіберстійка мультисервісна корпоративна мережа підприємства

HomeController.java

```
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "Hello, World!";
    }

    @GetMapping("/user1")
    public String user1() {
        return "Hello, User 1!";
    }

    @GetMapping("/user2")
    public String user2() {
        return "Hello, User 2!";
    }
}
```

RateLimitFilter.java

```
package com.example.demo;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.atomic.AtomicInteger;

public class RateLimitFilter implements Filter {

    private final ConcurrentHashMap<String, AtomicInteger> ipCounter = new ConcurrentHashMap<>();
    private final ConcurrentHashMap<String, AtomicInteger> userCounter = new ConcurrentHashMap<>();
```

```
    private final int MAX_REQUESTS_PER_MINUTE = 5;

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        HttpServletRequest httpServletRequest = (HttpServletRequest) request;
        String clientIp = httpServletRequest.getRemoteAddr();
        String user = httpServletRequest.getRemoteUser();

        ipCounter.putIfAbsent(clientIp, new AtomicInteger(0));
        userCounter.putIfAbsent(user, new AtomicInteger(0));

        if (ipCounter.get(clientIp).incrementAndGet() > MAX_REQUESTS_PER_MINUTE ||
                userCounter.get(user).incrementAndGet() > MAX_REQUESTS_PER_MINUTE) {
            HttpServletResponse httpServletResponse = (HttpServletResponse) response;
            httpServletResponse.setStatus(429);
            httpServletResponse.getWriter().write("Too many requests per minute");
            return;
        }

        chain.doFilter(request, response);
    }

    @Override
    public void init(FilterConfig filterConfig) {
        // Initialize filter
    }

    @Override
    public void destroy() {
        // Cleanup filter
    }
}
```

SecurityConfig.java

```
package com.example.demo;

import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
```

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/user1").hasRole("USER1")
                .antMatchers("/user2").hasRole("USER2")
                .anyRequest().authenticated()
                .and()
                .httpBasic();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication()
                .withUser("user1")
                .password(passwordEncoder().encode("pass1"))
                .roles("USER1")
                .and()
                .withUser("user2")
                .password(passwordEncoder().encode("pass2"))
                .roles("USER2");
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public FilterRegistrationBean<RateLimitFilter> rateLimitFilter() {
        FilterRegistrationBean<RateLimitFilter> registrationBean = new FilterRegistrationBean<>();

        registrationBean.setFilter(new RateLimitFilter());
        registrationBean.addUrlPatterns("/user1", "/user2");

        return registrationBean;
    }
}
```

Application.java

```
package com.example.demo;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```