Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

“Розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку”

Виконав: студент 4курсу, гр.2КІТС–20б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Господар І. М.
(прізвище та ініціали)

Керівник: к.т.н., доц., доцент каф. МБІС
Шиян А.А
(прізвище та ініціали)
«6» червня 2024 р.

Рецензент: к.т.н., доц., доцент каф. ОТ
Черняк О. І.
(прізвище та ініціали)
«6» червня 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
“6” червня 2024р.

Вінниця ВНТУ – 2024

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

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

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

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

_____ **д.т.н., проф. Яремчук Ю.Є.**
“ 22 ” березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Господарю Івану Миколайовичу
(прізвище, ім`я, по–батькові)

1. Тема роботи ”Розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку”
Керівник роботи: к. ф. –м. н., доц. Шиян Анатолій Антонович
(прізвище, ім`я, по–батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “ 11 ” березня 2024року № 80
2. Термін подання студентом роботи 10.06.2024
3. Вихідні дані до роботи:
Метою роботи є розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку для підвищення рівня безпеки та ефективності використання ресурсів в інформаційних системах організацій
4. Зміст текстової частини:
У першому розділі дипломної роботи здійснюється аналіз сучасного стану проблеми захисту та керування ресурсами в інформаційних системах, досліджуються існуючі підходи та рішення щодо рольового управління доступом та контролю мережевого трафіку, виявляються їх переваги та недоліки. У другому розділі розробляється підхід до управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, проектується архітектура програми та описується взаємодія з базами даних. Здійснюється вибір та обґрунтування використовуваних технологій, мов програмування та інструментів розробки. У

третьому розділі дипломної роботи виконується програмна реалі
розробленого підходу, створюється програмний комплекс для захист
управління розподілом ресурсів. Надаються інструкції користувачу щодо роб
програмним комплексом. У висновках підбиваються підсумки проведеної ро
аналізуються отримані результати та окреслюються перспективи подаль
розвитку та вдосконалення розробленої програми захисту та управ
розподілом ресурсів.

5. Перелік ілюстративного матеріалу:
У першому розділі бакалаврської дипломної роботи наявно 7 рисунка, у дру
розділі – 6 рисунків, у третьому розділі – 7 рисунків.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Прим
		початок	закінчення	
1	Визначення напрямку бакалаврської роботи, формулювання теми	24.03.2024	27.03.2024	
2	Аналіз предметної області обраної теми	28.03.2024	02.04.2024	
3	Розробка алгоритму роботи	04.04.2024	19.04.2024	
4	Написання бакалаврської роботи на основі розробленої теми	19.04.2024	04.05.2024	
5	Передзахист бакалаврської дипломної роботи	06.05.2024	24.05.2024	
6	Виправлення, уточнення, корегування бакалаврської дипломної роботи	27.05.2024	08.06.2024	
7	Захист бакалаврської дипломної роботи	17.06.2024	17.06.2024	

Студент _____ Господар І. М.
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Робота складається з 65 сторінки формату А4, в якій є 17 рисунків та перелік посилань, що містить 43 найменування.

Бакалаврська дипломна робота присвячена розробці програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. В роботі проведено аналіз існуючих підходів, методів та реалізацій захисту й управління розподілом ресурсів, виявлено їх переваги та недоліки.

На основі аналізу визначено необхідність розробки нової програми, яка б усувала недоліки існуючих рішень та забезпечувала ефективне, гнучке й адаптивне управління ресурсами. Запропоновано програму, що базується на рольовому контролі доступу (RBAC) та інтегрує методи управління розподілом ресурсів і обмеження трафіку.

Описано архітектуру, компоненти та деталі реалізації програми. Результати тестування демонструють її здатність забезпечувати високий рівень безпеки, оптимальне використання ресурсів та адаптивність до змінних умов функціонування.

Робота має теоретичне та практичне значення для фахівців у галузі інформаційної безпеки, системних адміністраторів та розробників програмного забезпечення. Результати дослідження можуть бути використані для подальшого вдосконалення методів та інструментів захисту й управління розподілом ресурсів в інформаційних системах.

Ключові слова: безпека, управління ресурсами, рольовий доступ, обмеження трафіку, аналітична база, програмне забезпечення.

ABSTRACT

The work consists of 65 pages of A4 format, which includes 17 figures and a list of references containing 43 items.

The bachelor's thesis is devoted to the development of a program for protecting and managing resource allocation with role–based access and traffic restrictions. The work analyzes existing approaches, methods and implementations of resource allocation protection and management, identifies their advantages and disadvantages.

Based on the analysis, the need to develop a new program that would eliminate the shortcomings of existing solutions and provide efficient, flexible and adaptive resource management is determined. A program based on role–based access control (RBAC) and integrating methods of managing resource allocation and traffic restriction is proposed.

The architecture, components, and implementation details of the program are described. The test results demonstrate its ability to provide a high level of security, optimal resource utilization, and adaptability to changing operating conditions.

The work is of theoretical and practical importance for information security specialists, system administrators and software developers. The results of the study can be used to further improve methods and tools for protecting and managing resource allocation in information systems.

Keywords: security, resource management, role–based access, traffic restriction, analytical base, software.

ЗМІСТ

ВСТУП..... 8
1. АНАЛІЗ ТЕОРЕТИЧНИХ ОСНОВ ЗАХИСТУ ТА УПРАВЛІННЯ РЕСУРСАМИ З РОЛЬОВИМ ДОСТУПОМ І УРАХУВАННЯМ ОБМЕЖЕННЯ ТРАФІКУ..... 11
1.1 Аналіз відомих підходів до реалізації захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку..... 11
1.2 Аналіз існуючих методів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку..... 15
1.3 Аналіз існуючих реалізацій методів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку..... 19
1.4 Висновки та постановка задач 26
2. РОЗРОБКА ПРОГРАМИ ЗАХИСТУ ДЛЯ УПРАВЛІННЯ РОЗПОДІЛОМ РЕСУРСІВ З РОЛЬОВИМ ДОСТУПОМ І УРАХУВАННЯМ ОБМЕЖЕННЯ ТРАФІКУ 28
2.1 Розробка підходу управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку 28
2.2 Проєктування архітектури розроблюваної програми захисту та опис взаємодії із базами даних 35
2.3 Обґрунтування вибору мови програмування, інструментів розробки та бази даних..... 44
2.4 Висновки до розділу 2..... 47
3 ПРАКТИЧНА РЕАЛІЗІЯ ПРОГРАМИ ЗАХИСТУ ТА УПРАВЛІННЯ РОЗПОДІЛОМ РЕСУРСІВ З РОЛЬОВИМ ДОСТУПОМ І УРАХУВАННЯМ ОБМЕЖЕННЯ ТРАФІКУ..... 49
3.1 Опис реалізація програмної логіки компонентів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку 49
3.2 Тестування розробленої програми та надання інструкцій користувачу ... 54
3.3 Висновки до розділу 3 61

ВИСНОВКИ 63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 65
Додаток А. Технічне завдання 70
Додаток Б. Лістинг коду модулів розробленого ПЗ 74
Додаток В. Ілюстративний матеріал 86
Додаток Г. Протокол перевірки на антиплагіат 92
Показники звіту подібності Unicheck 92

ВСТУП

Актуальність. У сучасному цифровому світі, де інформаційні технології стрімко розвиваються, а обсяги даних, що обробляються та передаються, постійно зростають, питання інформаційної безпеки та ефективного управління ресурсами набуває все більшої важливості. Організації різних галузей стикаються з викликами захисту своїх інформаційних активів від несанкціонованого доступу, забезпечення конфіденційності та цілісності даних, а також оптимального розподілу обчислювальних ресурсів та мережевого трафіку. Неефективне управління доступом та відсутність контролю за використанням ресурсів можуть призвести до серйозних наслідків, таких як витік конфіденційної інформації, порушення роботи систем, фінансові втрати та репутаційні ризики.

Ситуація ускладнюється постійним зростанням кіберзагроз, появою нових методів атак та збільшенням кількості вразливостей у програмному забезпеченні. Зловмисники активно шукають шляхи отримання несанкціонованого доступу до інформаційних систем, використовуючи слабкі місця в механізмах аутентифікації та авторизації, а також експлуатуючи недоліки в управлінні мережевим трафіком. Тому розробка ефективних програмних рішень для захисту та управління розподілом ресурсів з урахуванням рольового доступу та обмеження трафіку є критично важливим завданням.

Однак, існуючі рішення мають певні обмеження та недоліки. Зокрема, традиційні підходи до управління доступом, такі як дискреційний та мандатний контроль доступу, не завжди забезпечують достатню гнучкість та масштабованість в умовах складних організаційних структур та динамічних змін у ролях користувачів. Крім того, більшість існуючих систем контролю трафіку орієнтовані на статичні правила та не враховують контекстну інформацію, що може призвести до неоптимального використання мережевих ресурсів та ускладнити виявлення аномалій.

Таким чином, розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку є актуальним завданням, що дозволить підвищити рівень безпеки інформаційних систем, забезпечити ефективне використання ресурсів та мінімізувати ризики, пов'язані з несанкціонованим доступом та зловживанням привілеями.

Мета роботи полягає у розробці програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку для підвищення рівня безпеки та ефективності використання ресурсів в інформаційних системах організацій.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати сучасний стан проблеми захисту та керування ресурсами в інформаційних системах;

– дослідити існуючі підходи та рішення щодо рольового управління доступом та контролю мережевого трафіку, виявити їх переваги та недоліки;

– розробити підхід до управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, спроектувати архітектуру програмного комплексу та описати взаємодію з базами даних;

– реалізувати програмний комплекс для захисту та управління розподілом ресурсів відповідно до розробленого підходу та архітектури;

– розробити інструкції користувачу щодо роботи з програмним комплексом.

Об'єктом дослідження є процес захисту та управління розподілом ресурсів в інформаційних системах організацій.

Предметом дослідження є методи та засоби рольового управління доступом та контролю мережевого трафіку для забезпечення безпеки та ефективного використання ресурсів.

Новизна роботи полягає у розробці комплексного підходу до управління розподілом ресурсів, який поєднує рольове управління доступом з урахуванням обмеження трафіку. Запропонований підхід відрізняється від існуючих рішень тим, що забезпечує гнучкість та масштабованість рольового доступу, динамічне

управління мережевим трафіком на основі пріоритетів та контекстної інформації, а також можливість моніторингу та аналізу використання ресурсів для виявлення аномалій та потенційних загроз безпеці. Розроблена архітектура програмного комплексу та реалізовані алгоритми дозволяють ефективно вирішувати завдання захисту та управління ресурсами з урахуванням специфіки організаційних процесів та вимог безпеки.

Практична цінність роботи полягає у можливості застосування розробленої програми в реальних інформаційних системах організацій для підвищення рівня захищеності даних, запобігання несанкціонованому доступу, оптимізації розподілу ресурсів та контролю мережевого трафіку. Впровадження такого програмного комплексу дозволить мінімізувати ризики інформаційної безпеки, підвищити ефективність роботи систем та забезпечити відповідність вимогам регуляторних стандартів.

1. АНАЛІЗ ТЕОРЕТИЧНИХ ОСНОВ ЗАХИСТУ ТА УПРАВЛІННЯ РЕСУРСАМИ З РОЛЬОВИМ ДОСТУПОМ І УРАХУВАННЯМ ОБМЕЖЕННЯ ТРАФІКУ

1.1 Аналіз відомих підходів до реалізації захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

У сучасному світі інформаційних технологій захист та управління розподілом ресурсів є критично важливими аспектами забезпечення безпеки та ефективності роботи систем. Особливо актуальним це питання стає в контексті рольового доступу та врахування обмежень трафіку. Адже саме ці механізми дозволяють контролювати, хто і в якому обсязі може отримувати доступ до певних ресурсів, а також запобігати перевантаженню мережі та оптимізувати її роботу.

Існує декілька підходів до реалізації захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Кожен з них має свої особливості, переваги та недоліки, і може бути більш або менш ефективним залежно від конкретних потреб і умов використання. Тому для вибору оптимального рішення необхідно детально розглянути та проаналізувати кожен з цих підходів.

Одним з найбільш поширених і ефективних підходів є рольовий контроль доступу (Role–Based Access Control, RBAC). Він базується на призначенні користувачам певних ролей, які визначають їх права доступу до ресурсів системи. Кожна роль має набір дозволів, які чітко регламентують, які дії користувач може виконувати з тими чи іншими ресурсами. Такий підхід дозволяє гнучко керувати доступом до ресурсів і значно спрощує адміністрування, особливо у великих системах з великою кількістю користувачів і ресурсів [1].

Ще одним важливим аспектом є управління розподілом ресурсів (Resource Allocation Management). Цей підхід передбачає розподіл наявних ресурсів між користувачами або групами користувачів відповідно до їх потреб і пріоритетів. Це дозволяє оптимізувати використання ресурсів, запобігти їх перевантаженню і

забезпечити більш ефективну роботу системи в цілому. Особливо актуальним управління розподілом ресурсів стає в умовах обмеженості ресурсів, коли необхідно забезпечити їх найбільш раціональне використання [2].

Варто також згадати про такий підхід, як обмеження трафіку (Traffic Shaping). Він використовується для контролю та обмеження мережевого трафіку з метою запобігання перевантаженню мережі та забезпечення її стабільної роботи. Цей підхід дозволяє встановлювати ліміти на пропускну здатність для окремих користувачів або груп користувачів, пріоритезувати трафік залежно від його типу і важливості, а також динамічно перерозподіляти ресурси мережі залежно від поточних потреб.

Цікавим і перспективним є підхід, який передбачає інтеграцію рольового контролю доступу RBAC та управління розподілом ресурсів. Таке поєднання дозволяє призначати ролі користувачам і на основі цих ролей не тільки керувати їх доступом до ресурсів, а й розподіляти ці ресурси між ними. Це дає можливість створювати більш гнучкі та ефективні системи управління доступом і розподілом ресурсів, які враховують як права користувачів, так і їх реальні потреби в ресурсах [3].

Ще одним підходом, який заслуговує на увагу, є використання політик безпеки (Security Policies). Цей підхід базується на визначенні чітких правил і обмежень щодо доступу до ресурсів і їх використання. Такі політики можуть включати в себе правила рольового контролю доступу RBAC, управління розподілом ресурсів, обмеження трафіку тощо. Перевагою використання політик безпеки є можливість централізовано встановлювати і змінювати ці правила, що значно спрощує управління безпекою і доступом до ресурсів.

Кожен з розглянутих підходів має свої особливості і може бути більш або менш ефективним залежно від конкретних потреб і умов використання. Тому при виборі підходу до реалізації захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку необхідно враховувати специфіку конкретної системи, її масштаб, критичність ресурсів, які потребують захисту, а також наявні технічні та фінансові можливості [4].

Рольовий контроль доступу RBAC є одним з найбільш поширених і ефективних підходів, який дозволяє гнучко керувати доступом до ресурсів і спрощує адміністрування системи. Управління розподілом ресурсів дозволяє оптимізувати їх використання і запобігти перевантаженню. Обмеження трафіку допомагає забезпечити стабільну роботу мережі та запобігти її перевантаженню.

Інтеграція рольового контролю доступу та управління розподілом ресурсів дає можливість створювати більш гнучкі та ефективні системи, які враховують як права користувачів, так і їх реальні потреби в ресурсах. Використання політик безпеки дозволяє централізовано встановлювати і змінювати правила доступу до ресурсів і їх використання [5].

Таким чином, вибір оптимального підходу до реалізації захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку залежить від конкретних потреб і умов використання системи. При цьому необхідно враховувати її масштаб, критичність ресурсів, які потребують захисту, а також наявні технічні та фінансові можливості. Тільки комплексний аналіз всіх цих факторів дозволить вибрати найбільш ефективне рішення для забезпечення надійного захисту і оптимального управління розподілом ресурсів.

Кожен з розглянутих підходів має свої особливості і може бути більш або менш ефективним залежно від конкретних потреб і умов використання. Тому при виборі підходу до реалізації захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку необхідно враховувати специфіку конкретної системи, її масштаб, критичність ресурсів, які потребують захисту, а також наявні технічні та фінансові можливості [6].

Для порівняння характеристик відомих підходів до реалізації захисту та управління розподілом ресурсів, розглянемо їх основні недоліки.

Рольовий контроль доступу (RBAC), незважаючи на свою ефективність і гнучкість, може бути складним в реалізації та адмініструванні, особливо в системах з великою кількістю ролей і дозволів. Крім того, цей підхід не враховує динамічні зміни в потребах користувачів і може бути недостатньо гнучким в умовах швидких змін [7].

Управління розподілом ресурсів, хоча і дозволяє оптимізувати їх використання, може бути недостатньо ефективним в умовах непередбачуваних змін в потребах користувачів або при появі нових типів ресурсів, які не були враховані при початковому розподілі.

Обмеження трафіку, незважаючи на свою ефективність в запобіганні перевантаженню мережі, може негативно впливати на продуктивність системи і призводити до затримок в обробці запитів користувачів [8].

Інтеграція рольового контролю доступу та управління розподілом ресурсів, хоча і дозволяє створювати більш гнучкі та ефективні системи, може бути складною в реалізації та потребувати значних обчислювальних ресурсів.

Використання політик безпеки, незважаючи на можливість централізованого управління правилами доступу, може бути недостатньо гнучким в умовах швидких змін і потребувати частого оновлення політик.

Підсумовуючи, можна сказати, що існуючі підходи до реалізації захисту та управління розподілом ресурсів мають ряд недоліків, таких як складність реалізації та адміністрування, недостатня гнучкість в умовах динамічних змін, негативний вплив на продуктивність системи, потреба в значних обчислювальних ресурсах і частому оновленні політик безпеки [9].

Порівняння характеристик відомих підходів показало, що вони мають ряд недоліків, таких як складність реалізації та адміністрування, недостатня гнучкість в умовах динамічних змін, негативний вплив на продуктивність системи, потреба в значних обчислювальних ресурсах і частому оновленні політик безпеки.

Тому зважаючи на специфіку теми роботи та розробки програми захисту і управління розподілом ресурсів з рольовим доступом з урахуванням обмеження трафіку в подальшому буде використовуватися саме рольовий контроль доступу RBAC [10]. Цей вибір обумовлений тим, що RBAC є одним з найбільш ефективних і гнучких підходів до управління доступом до ресурсів, який дозволяє чітко визначати права користувачів на основі їх ролей в системі і спрощує адміністрування за рахунок централізованого управління ролями та дозволами.

Використання RBAC в поєднанні з механізмами управління розподілом ресурсів та обмеження трафіку дозволить створити ефективну і надійну систему захисту та управління ресурсами, яка буде враховувати як права користувачів, так і реальні потреби в ресурсах та обмеження мережі. Такий комплексний підхід дозволить забезпечити високий рівень безпеки і оптимальне використання ресурсів системи.

1.2 Аналіз існуючих методів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

Персоналізуючі програми управління доступом PSAC) орієнтовані на адаптацію прав доступу до потреб кожного окремого користувача. Вони дозволяють створювати детальні профілі користувачів, які включають інформацію про їхні ролі, рівні доступу, історію дій у системі, а також інші персональні дані. Це забезпечує більш точний контроль доступу до ресурсів і дозволяє ефективніше реагувати на загрози безпеці.

Захист та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку є важливими завданнями в сучасних інформаційних системах. Для вирішення цих завдань існує кілька методів, кожен з яких має свої особливості, переваги та недоліки. Розуміння цих методів є ключовим для вибору найбільш ефективного підходу до забезпечення безпеки та управління ресурсами в конкретній системі [11].

Одним з поширених методів є дискреційний контроль доступу (Discretionary Access Control, DAC). Цей метод базується на принципі, що власники ресурсів мають повний контроль над тим, хто може отримувати доступ до їхніх ресурсів і які дії можуть виконувати з ними. DAC дозволяє власникам ресурсів самостійно визначати політики доступу, що робить цей метод досить гнучким. Однак, він також має і певні недоліки, такі як складність управління в великих системах і потенційна вразливість до атак, пов'язаних з помилками конфігурації або зловживанням привілеями [12].

Іншим важливим методом є мандатний контроль доступу (Mandatory Access Control, MAC). На відміну від DAC, де контроль доступу визначається власниками ресурсів, в MAC політики доступу встановлюються централізовано на основі призначених рівнів безпеки. Кожному користувачу і ресурсу присвоюється певний рівень безпеки, і доступ надається тільки тоді, коли рівень безпеки користувача відповідає рівню безпеки ресурсу. MAC забезпечує більш строгий і надійний контроль доступу, особливо в системах з високими вимогами до безпеки, таких як військові або державні установи. Однак, він також може бути менш гнучким і більш складним в управлінні, ніж DAC [13].

Ще одним сучасним методом є атрибутний контроль доступу (Attribute–Based Access Control, ABAC). Цей метод базується на оцінці різних атрибутів користувачів, ресурсів і середовища для прийняття рішень про доступ. Атрибути можуть включати в себе такі характеристики, як роль користувача, місцезнаходження, час доступу, тип пристрою тощо. ABAC дозволяє створювати більш гнучкі та динамічні політики доступу в порівнянні з традиційними методами, такими як RBAC (рольовий контроль доступу). Це особливо корисно в системах з великою кількістю користувачів і ресурсів, де статичні ролі можуть бути недостатньо ефективними. Однак, реалізація ABAC може бути більш складною і потребувати більших обчислювальних ресурсів [14].

Крім методів контролю доступу, важливу роль в управлінні розподілом ресурсів відіграють також методи керування пропускною здатністю (Bandwidth Management). Ці методи дозволяють контролювати та обмежувати пропускну здатність мережі для окремих користувачів або груп користувачів. Це допомагає запобігти перевантаженню мережі та забезпечити справедливий розподіл ресурсів між користувачами. Методи керування пропускною здатністю можуть включати в себе такі техніки, як обмеження швидкості передачі даних, встановлення квот на використання трафіку, пріоритезація трафіку на основі типу додатків або користувачів тощо.

Пріоритезація трафіку (Traffic Prioritization) є ще одним важливим методом управління розподілом ресурсів. Цей метод дозволяє призначати різні пріоритети

різним типам трафіку в мережі. Наприклад, критично важливий трафік, такий як голосові дзвінки або відеоконференції, може отримувати вищий пріоритет, ніж менш важливий трафік, такий як завантаження файлів або перегляд веб–сторінок. Це допомагає забезпечити якість обслуговування (Quality of Service, QoS) для критичних додатків навіть в умовах обмеженої пропускної здатності мережі. Методи пріоритезації трафіку можуть базуватися на різних критеріях, таких як тип протоколу, IP–адреса джерела або призначення, порт тощо [15].

Кожен з розглянутих методів має свої переваги і недоліки, і вибір найбільш підходящого методу залежить від специфічних вимог і умов конкретної системи. Наприклад, в системах з високими вимогами до безпеки, таких як військові або фінансові установи, може бути доцільним використання мандатного контролю доступу MAC. В той же час, в системах з великою кількістю користувачів і ресурсів, де потрібна більша гнучкість, може бути ефективнішим використання атрибутного контролю доступу ABAC.

Важливо також розуміти, що ці методи не є взаємовиключними і можуть використовуватися в поєднанні для досягнення оптимального результату. Наприклад, дискреційний контроль доступу DAC часто використовується в поєднанні з рольовим контролем доступу RBAC для забезпечення більш гнучкого і в той же час надійного управління доступом. Методи керування пропускною здатністю та пріоритезації трафіку також можуть використовуватися спільно для забезпечення ефективного розподілу мережевих ресурсів [16].

Іншим важливим аспектом є те, що ефективність використання цих методів значною мірою залежить від правильності їх налаштування та управління. Навіть найбільш досконалі методи контролю доступу та управління ресурсами можуть бути неефективними або навіть небезпечними, якщо вони неправильно налаштовані або якщо ними неправильно керують. Тому важливо приділяти належну увагу процесам конфігурації, моніторингу та аудиту цих систем.

Крім того, важливо враховувати, що з розвитком технологій і появою нових загроз безпеці, методи захисту та управління розподілом ресурсів також повинні постійно розвиватися і адаптуватися. Це означає, що організації повинні бути

готові до регулярного перегляду та оновлення своїх політик безпеки та механізмів управління ресурсами відповідно до змін в технологічному ландшафті та ризиках безпеки.

Важливо розуміти, що ці методи можуть і повинні використовуватися в поєднанні для досягнення оптимального результату. Крім того, ефективність їх використання значною мірою залежить від правильності налаштування та управління [17].

В умовах постійного розвитку технологій і появи нових загроз безпеці, методи захисту та управління розподілом ресурсів також повинні постійно розвиватися і адаптуватися. Це вимагає від організацій регулярного перегляду та оновлення своїх політик безпеки та механізмів управління ресурсами.

Таким чином, розуміння і правильне застосування різних методів захисту та управління розподілом ресурсів є ключовим для забезпечення безпеки, ефективності та надійності сучасних інформаційних систем.

Проаналізувавши існуючі методи захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку в рамках підрозділу 1.2, можна зробити висновок, що кожен з цих методів має свої сильні і слабкі сторони, і вибір найбільш підходящого методу залежить від специфічних вимог і умов конкретної системи [18].

Дискреційний контроль доступу (DAC) є гнучким, але може бути складним в управлінні і потенційно вразливим. Мандатний контроль доступу (MAC) забезпечує високий рівень безпеки, але може бути менш гнучким. Атрибутний контроль доступу (ABAC) дозволяє створювати динамічні політики доступу, але може потребувати більших обчислювальних ресурсів [18].

Методи керування пропускною здатністю та пріоритезації трафіку є важливими інструментами для забезпечення ефективного розподілу мережевих ресурсів і підтримки якості обслуговування критичних додатків.

Враховуючи переваги та недоліки кожного з методів, а також специфіку теми дослідження, в подальшому буде використовуватися саме метод рольового контролю доступу (Role–Based Access Control, RBAC). Цей вибір обумовлений тим,

що RBAC дозволяє ефективно керувати доступом до ресурсів на основі ролей користувачів, що забезпечує високий рівень безпеки та гнучкості системи. Крім того, RBAC добре масштабується і може бути легко інтегрований з іншими методами, такими як управління пропускною здатністю та пріоритезація трафіку, для досягнення оптимального результату [19].

Використання RBAC в поєднанні з ретельним налаштуванням, моніторингом та регулярним оновленням політик безпеки дозволить створити надійну та ефективну систему захисту та управління розподілом ресурсів, яка відповідатиме сучасним викликам у сфері інформаційної безпеки.

1.3 Аналіз існуючих реалізацій методів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

У сучасному цифровому світі, де інформація є найціннішим активом, програми управління ресурсами з рольовим доступом RBAC відіграють ключову роль у забезпеченні безпеки та ефективності роботи організацій. RBAC дозволяє точно визначити, хто має доступ до яких ресурсів та які дії вони можуть виконувати, що є критично важливим для захисту конфіденційних даних та запобігання несанкціонованому доступу.

З розвитком технологій та збільшенням кількості хмарних сервісів, вибір відповідної програми RBAC стає все більш складним завданням. На ринку представлено безліч рішень, як локальних (on–premises), так і хмарних, кожне з яких має свої переваги та недоліки [20].

У цьому дослідженні ми проведемо детальний аналіз існуючих реалізацій методів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, розглянемо їх функціональність, переваги та недоліки, а також надамо рекомендації щодо вибору оптимального рішення залежно від потреб та особливостей конкретної організації.

Порівняння реалізацій методів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

Microsoft Active Directory (AD) є найпоширенішим рішенням для управління ідентифікацією та доступом в корпоративному середовищі. Він пропонує централізоване сховище для об'єктів (користувачів, комп'ютерів, груп), що дозволяє легко керувати доступом до ресурсів. AD підтримує різні протоколи аутентифікації, включаючи Kerberos та LDAP. Він також пропонує інструменти для управління груповими політиками, що дозволяє застосовувати налаштування до цілих груп користувачів або комп'ютерів (рис. 1.1) [21].

Рисунок 1.1 – Логотип програмного забезпечення “Active Directory”

Переваги:

- зріла та перевірена технологія;
- широка функціональність;
- високий рівень безпеки;
- централізоване управління;
- інтеграція з іншими продуктами Microsoft.

Недоліки:

- висока вартість впровадження та підтримки;
- складність налаштування для великих організацій;

– вимагає спеціалізованих знань.

OpenLDAP – це відкрита реалізація протоколу LDAP (Lightweight Directory Access Protocol), який часто використовується для централізованого зберігання і управління інформацією про користувачів і їх права доступу. OpenLDAP підтримує RBAC і може інтегруватися з різними системами автентифікації та авторизації. Він також підтримує реплікацію, що забезпечує високу доступність даних (рис. 1.2) [22].

Рис. 1.2 Логотип програмного забезпечення “OpenLDAP”

Переваги:

– безкоштовно та з відкритим кодом;
– широкі можливості налаштування;
– висока продуктивність;
– підтримка різних платформ;
– активна спільнота розробників.

Недоліки:

– вимагає технічних знань для впровадження та підтримки;
– менш зручний інтерфейс ніж у комерційних аналогів;
– обмежена підтримка.

FreeIPA є відкритим рішенням, побудованим на базі OpenLDAP. Воно спрощує управління ідентифікацією, аутентифікацією та авторизацією, надаючи зручний

веб–інтерфейс. FreeIPA інтегрується з Kerberos та підтримує автоматизоване управління сертифікатами (рис. 1.3) [23].

Рис. 1.3 Логотип програмного забезпечення “freeIPA”

Переваги:

- безкоштовно та з відкритим кодом;
- спрощене управління ідентифікацією;
- зручний веб–інтерфейс;
- інтеграція з іншими системами Linux;
- автоматизоване управління сертифікатами.

Недоліки:

- менш поширений ніж Active Directory;
- вимагає деяких технічних знань;
- обмежена документація.

Okta є хмарним рішенням для управління ідентифікацією та доступом. Воно пропонує широкий спектр функцій, включаючи єдиний вхід (SSO), управління життєвим циклом користувачів, адаптивну аутентифікацію та інтеграцію з багатьма хмарними сервісами. Okta також підтримує багатофакторну аутентифікацію та моніторинг безпеки в реальному часі (рис. 1.4) [24].

Рис 1.4 Логотип програмного забезпечення “okta”

Переваги:

- простота використання;
- масштабованість;
- високий рівень безпеки;
- швидке впровадження;
- постійна підтримка від Okta.

Недоліки:

- залежність від інтернет–з'єднання;
- вартість може бути високою для великих організацій;
- обмежений контроль над інфраструктурою.

Auth0 є хмарним рішенням для управління ідентифікацією та доступом. Воно підтримує різні протоколи аутентифікації, включаючи SAML, OAuth та OpenID Connect. Auth0 також пропонує детальну аналітику та дозволяє кастомізувати інтерфейс користувача (рис. 1.5) [25].

Рис 1.5 Логотип програмного забезпечення “Auth0”

Переваги:

- гнучкість налаштування;
- швидке впровадження;
- масштабованість;
- підтримка різних сценаріїв аутентифікації;
- детальна аналітика.

Недоліки:

- залежність від інтернет–з'єднання;
- вартість може бути високою для великих організацій;
- обмежений контроль над інфраструктурою.

JumpCloud є хмарним рішенням, яке поєднує в собі управління ідентифікацією та пристроями. Воно пропонує хмарний LDAP, багатофакторну аутентифікацію, інтеграцію з Active Directory, управління паролями та віддалений доступ до пристроїв. JumpCloud підтримує різні платформи, включаючи Windows, macOS та Linux (рис. 1.6) [26].

Рис 1.6 Логотип програмного забезпечення “jumpcloud”

Переваги:

- комплексне рішення для управління ідентифікацією та пристроями;
- масштабованість;
- кросплатформеність;
- централізоване управління;
- інтеграція з Active Directory.

Недоліки:

- залежність від інтернет–з'єднання;
- вартість може бути високою для великих організацій;
- обмежений контроль над інфраструктурою.

Загалом для малих і середніх організацій можуть підійти хмарні рішення або FreeIPA. Великі організації частіше використовують Active Directory [27]. OpenLDAP та FreeIPA – безкоштовні рішення. Хмарні сервіси мають різні тарифні плани. Active Directory та OpenLDAP вимагають кваліфікованих фахівців для впровадження та підтримки. Хмарні рішення простіші у використанні. Якщо у вас вже є інфраструктура Microsoft, то Active Directory може бути логічним вибором. Якщо ви використовуєте різні платформи, то OpenLDAP або хмарні рішення можуть бути кращими. Якщо безпека є пріоритетом, то варто звернути увагу на рішення з підтримкою багатофакторної аутентифікації та моніторингу безпеки [28].

Аналіз існуючих реалізацій методів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку показав, що сучасні рішення забезпечують високий рівень безпеки та функціональності, але мають певні обмеження, зокрема, у сфері вартості, складності використання та інтеграції з існуючими системами.

Вибір оптимальної реалізації залежить від специфічних потреб і особливостей конкретної організації, таких як розмір, бюджет, наявні технічні ресурси та інфраструктура, а також вимоги до безпеки [29].

Незважаючи на широкий вибір існуючих рішень, кожне з них має свої недоліки, які можуть обмежувати їх ефективність та застосовність у певних ситуаціях. Серед основних недоліків можна виділити високу вартість впровадження та підтримки, складність налаштування та використання, необхідність спеціалізованих знань, обмежену інтеграцію з системами інших виробників та потенційний вплив на продуктивність мережі [30].

Враховуючи ці недоліки, постає питання про розробку нової програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, яка б могла усунути або мінімізувати існуючі проблеми та

забезпечити більш ефективне, гнучке та адаптивне рішення для організацій різного розміру та з різними потребами.

Саме тому, в рамках даної роботи, буде проведено розробку нової програми захисту та управління розподілом ресурсів, яка зможе запропонувати альтернативу існуючим рішенням та забезпечити високий рівень безпеки, функціональності та зручності використання при оптимальних витратах ресурсів та з урахуванням специфічних потреб організацій.

1.4 Висновки та постановка задач

У першому розділі роботи було проаналізовано існуючі підходи, методи та реалізації захисту й управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Розглянуто переваги та недоліки кожного з підходів, а також фактори, які слід враховувати при виборі оптимального рішення.

На основі проведеного аналізу виявлено, що існуючі рішення мають певні недоліки, такі як висока вартість, складність використання та обмежена інтеграція з іншими системами. Це зумовлює необхідність розробки нової програми захисту та управління розподілом ресурсів, яка б могла усунути або мінімізувати ці недоліки та забезпечити більш ефективне, гнучке та адаптивне рішення.

Зроблено висновок, що рольовий контроль доступу RBAC є найбільш перспективним підходом для подальшого дослідження в рамках даної роботи. RBAC дозволяє ефективно керувати доступом до ресурсів на основі ролей користувачів, забезпечуючи високий рівень безпеки та гнучкості системи. Крім того, RBAC добре масштабується і може бути легко інтегрований з іншими методами, такими як управління пропускною здатністю та пріоритезація трафіку.

У процесі проведення аналізу теоретичного матеріалу, було поставлено такі подальші завдання:

- розробити підхід до захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку на основі RBAC;

– спроєктувати архітектуру та реалізувати програмний засіб, що автоматизує запропонований підхід;

– протестувати та оцінити ефективність розробленої програми в умовах, наближених до реальних.

Виконання усіх поставлених завдань дозволить досягнути головної мети бакалаврської дипломної роботи – створення ефективної та надійної системи захисту та управління розподілом ресурсів, яка забезпечить високий рівень безпеки, оптимальне використання ресурсів та адаптивність до змінних умов функціонування.

2. РОЗРОБКА ПРОГРАМИ ЗАХИСТУ ДЛЯ УПРАВЛІННЯ РОЗПОДІЛОМ РЕСУРСІВ З РОЛЬОВИМ ДОСТУПОМ І УРАХУВАННЯМ ОБМЕЖЕННЯ ТРАФІКУ

2.1 Розробка підходу управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

Проведений у першому розділі аналіз сучасного стану проблеми захисту та керування ресурсами в інформаційних системах виявив низку недоліків та обмежень існуючих програмних рішень. Серед найбільш суттєвих з них варто відзначити недостатню гнучкість реалізації рольового управління доступом, відсутність ефективних механізмів керування мережевим трафіком та обмежені можливості моніторингу використання ресурсів. Ці недоліки призводять до неоптимального використання обчислювальних потужностей та мережевих каналів, ускладнюють адміністрування систем та потенційно загрожують безпеці чи доступності інформаційних сервісів [31].

З метою подолання зазначених обмежень у даній роботі пропонується розробити комплексне програмне рішення, яке поєднуватиме функції управління доступом на основі ролей, моніторингу та оптимізації використання ресурсів, а також забезпечення мережевої безпеки шляхом контролю та фільтрації трафіку. Таке рішення базуватиметься на низці концептуальних та архітектурних удосконалень, що включають:

– реалізацію розширеної моделі ролей, яка підтримуватиме ієрархію ролей та дозволить гнучко описувати права доступу відповідно до функціональних обов'язків та сфер відповідальності користувачів в організації.

– розробку механізмів динамічного управління пропускною здатністю мережі на основі пріоритетів користувачів та типів трафіку, а також забезпечення додаткової фільтрації та інспекції трафіку для виявлення потенційних загроз безпеці.

– впровадження компонентів збору деталізованої статистики щодо завантаження ресурсів, їх агрегації та інтелектуального аналізу. Це дозволить виявляти неефективні чи підозрілі патерни використання ресурсів та генерувати сповіщення для адміністраторів.

– створення зручного та уніфікованого інтерфейсу керування з можливістю автоматизації рутинних операцій за допомогою скриптів та шаблонів конфігурації.

Ключовим аспектом забезпечення безпеки у розроблюваному програмі є реалізація ефективної системи управління доступом [32]. Для цього буде використано рольову модель доступу (Role–Based Access Control – RBAC), яка дозволяє гнучко налаштовувати права та привілеї користувачів відповідно до їх функціональних обов'язків.

Алгоритм управління доступом на основі ролей включає описуєтья так [33]:

– ідентифікація користувача – користувач проходить процес автентифікації, надаючи унікальні облікові дані (ім'я користувача та пароль). Окрім того, програма перевіряє достовірність наданих обліковик даних шляхом звірки з інформацією, збереженою у захищеній базі даних;

– визначення ролі користувача – після успішної автентифікації, програма визначає роль користувача на основі його облікового запису. Кожній ролі у системі відповідає певний набір дозволів та обмежень щодо доступу до файлів, систем та функціональних можливостей;

– перевірка прав доступу – перед наданням доступу до запитуваного ресурсу, програма перевіряє, чи має користувач достатні права відповідно до його ролі. Для кожного файлу, системи чи дії визначено конкретні ролі, яким дозволено або заборонено виконувати ці операції;

– надання або відмова в доступі – якщо права доступу користувача підтверджено, то йому надається доступ до запитуваного ресурсу. У разі, якщо права доступу недостатні, користувачу буде відмовлено в доступі з відповідним повідомленням;

– ведення журналу подій – всі спроби доступу, як успішні, так і неуспішні, фіксуються у захищеному журналі подій. Журнал аудиту забезпечує можливість розслідування інцидентів безпеки та аналізу дій користувачів.

Таким чином, рольова модель доступу дозволяє гнучко регулювати права користувачів, забезпечуючи надійний захист інформаційних ресурсів організації від несанкціонованого доступу та зловживання привілеями (рис. 2.1).

Рисунок 2.1 – Схема алгоритму управління доступом на основі ролей

Крок 1. Початок.

Крок 2. Ідентифікація користувача. Користувач проходить процес автентифікації, надаючи свої унікальні облікові дані (ім'я користувача та пароль).

Крок 3. Визначення ролі користувача. Програма визначає роль користувача на основі його облікового запису.

Крок 4. Перевірка прав доступу. Програма перевіряє, чи має користувач достатні права доступу відповідно до його ролі.

Крок 5. Чи права доступу достатні? Перевіряється, чи користувач має необхідні права для доступу до запитуваного ресурсу.

Крок 6. Надання доступу. Якщо права доступу підтверджено, користувачу надається доступ до ресурсу.

Крок 7. Відмова в доступі. Якщо права доступу недостатні, користувачу відмовляється в доступі.

Крок 8. Ведення журналу подій. Всі події, пов'язані з доступом, фіксуються у захищеному журналі аудиту.

Крок 9. Кінець.

Алгоритм управління доступом на основі ролей реалізує гнучку систему контролю доступу до файлів, систем та функціональних можливостей програмного комплексу. Він дозволяє ідентифікувати користувача, визначити його роль у системі та надавати або відмовляти в доступі відповідно до встановлених політик безпеки. Це забезпечує надійний захист від несанкціонованого доступу та зловживання привілеями користувачів [34].

Окрім управління доступом, важливим аспектом забезпечення безпеки у розроблюваній програмі є контроль та обмеження мережевого трафіку. Це дозволяє запобігати витокам конфіденційних даних, захищати від мережевих атак та оптимізувати використання пропускної здатності каналів зв'язку.

Алгоритм обмеження мережевого трафіку включає наступні етапи:

– моніторинг мережевого трафіку – програма постійно здійснює моніторинг вхідного та вихідного мережевого трафіку користувачів. Збираються дані про

обсяги переданих/отриманих даних, кількість з'єднань, використання пропускної здатності тощо;

– класифікація трафіку – зібрані дані про трафік аналізуються та класифікуються за різними ознаками, такими як: тип трафіку (HTTP, FTP, SSH), джерело та адресат трафіку, обсяг переданих даних, час та інтенсивність трафіку;

– порівняння з політиками безпеки – класифікований трафік порівнюється з встановленими політиками безпеки, які визначають допустимі межі використання мережевих ресурсів;

– політики можуть регулювати обмеження на типи трафіку, квоти на обсяги передачі даних, обмеження на кількість одночасних з'єднань тощо;

– застосування обмежень – якщо виявлено, що трафік користувача виходить за встановлені політиками межі, програма застосовує відповідні обмеження. Це може включати тимчасове або постійне обмеження пропускної здатності, блокування певних типів трафіку, відключення доступу до ресурсів тощо;

– ведення журналу подій – всі події, пов'язані з моніторингом та обмеженням трафіку, фіксуються у захищеному журналі. Журнал аудиту дозволяє аналізувати тенденції, виявляти аномалії та розслідувати інциденти безпеки, пов'язані з мережевою активністю [35];

У свою чергу, алгоритм обмеження мережевого трафіку відіграє важливу роль у запобіганні витокам конфіденційних даних та захисті від мережевих атак. Він здійснює постійний моніторинг та класифікацію мережевого трафіку, порівнюючи його з політиками безпеки. У разі виявлення трафіку, що виходить за допустимі межі, алгоритм застосовує відповідні обмеження, такі як обмеження пропускної здатності чи блокування певних типів трафіку. Це дозволяє ефективно контролювати та оптимізувати використання мережевих ресурсів.

Завдяки реалізації даного алгоритму, програмний комплекс здатний ефективно контролювати та обмежувати мережевий трафік користувачів відповідно до встановлених політик безпеки, захищаючи організацію від потенційних загроз, пов'язаних із мережевою активністю (рис. 2.2).

Крок 1. Початок.

Крок 2. Моніторинг мережевого трафіку. Програма здійснює постійний моніторинг вхідного та вихідного мережевого трафіку користувачів.

Крок 3. Класифікація трафіку. Зібрані дані про трафік аналізуються та класифікуються за різними ознаками, такими як тип, обсяг, джерело та адресат.

Крок 4. Порівняння з політиками безпеки. Класифікований трафік порівнюється з встановленими політиками безпеки, які визначають допустимі межі використання мережевих ресурсів.

Крок 5. Чи трафік відповідає політикам? Перевіряється, чи трафік користувача знаходиться в допустимих межах, визначених політиками безпеки.

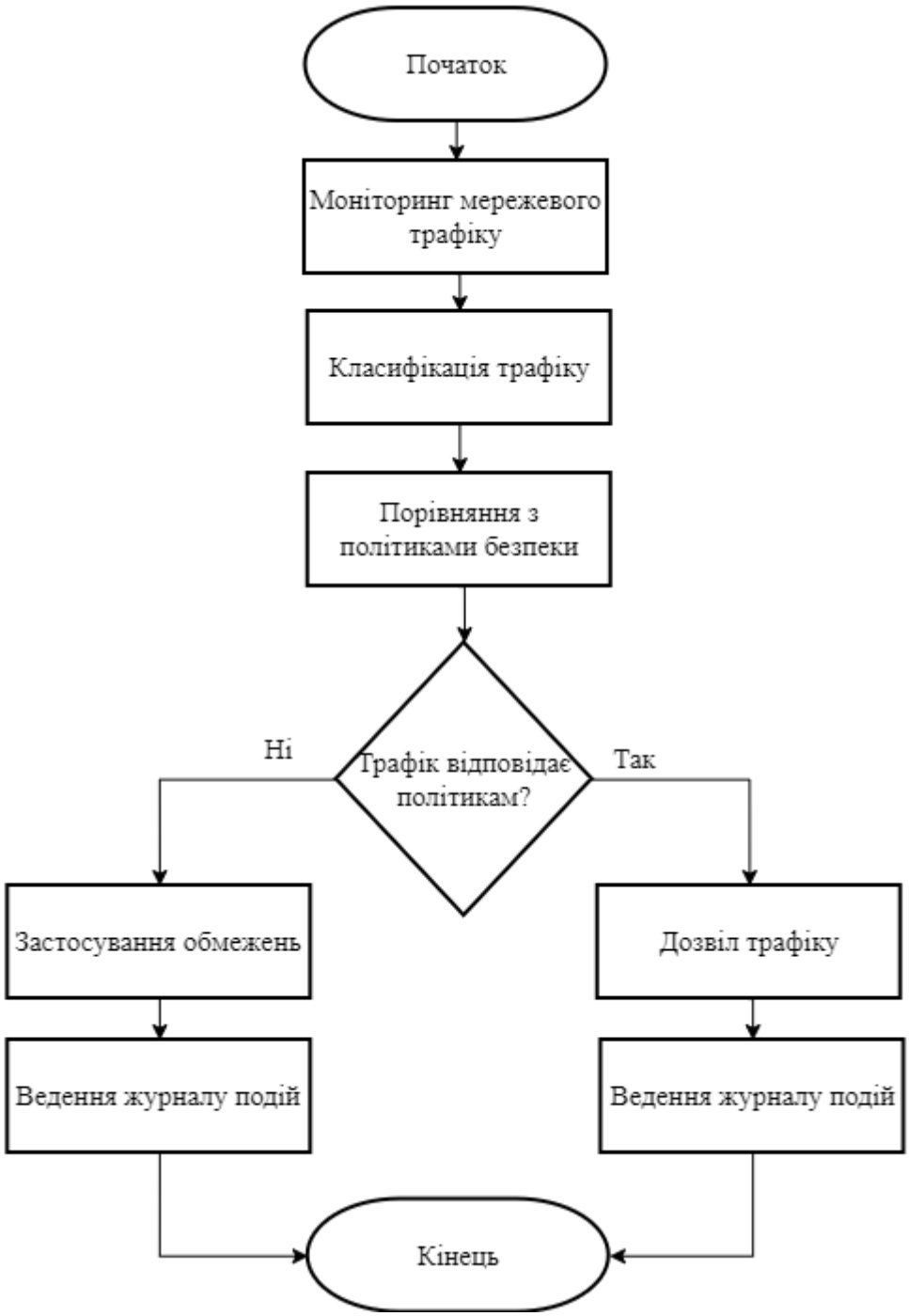


Рисунок 2.2 – Схема алгоритму обмеження мережевого трафіку

Крок 6. Дозвіл трафіку. Якщо трафік відповідає політикам, він дозволяється до передачі.

Крок 7. Застосування обмежень. Якщо трафік виходить за встановлені межі, програма застосовує відповідні обмеження, такі як обмеження пропускної здатності, блокування певних типів трафіку тощо.

Крок 8. Ведення журналу подій. Всі події, пов'язані з моніторингом та обмеженням трафіку, фіксуються у захищеному журналі аудиту.

Крок 9. Кінець.

Таким чином, ці два алгоритми є ключовими компонентами розроблюваної програми захисту та управління розподілом ресурсів та обмеження трафіку. Вони забезпечують надійний контроль доступу до інформаційних активів організації та захист від мережевих загроз, відіграючи важливу роль у комплексному вирішенні завдань забезпечення інформаційної безпеки.

2.2 Проєктування архітектури розроблюваної програми захисту та опис взаємодії із базами даних

Основним режимом роботи програми є циклічна обробка запитів від користувачів системи на доступ до захищених інформаційних ресурсів. Для кожного отриманого запиту послідовно виконуються наступні кроки. Спочатку виконується ідентифікація користувача, який надіслав запит, за допомогою наданих ним автентифікаційних даних (наприклад, логіну та пароля) [36]. На основі встановленої ідентичності користувача з бази даних вибираються призначені йому ролі та пов'язані з ними права доступу. Далі, спираючись на модель розмежування доступу, що міститься в пам'яті, перевіряється наявність у користувача права на доступ до запитаного ресурсу у відповідності до поточних політик безпеки. У випадку, коли доступ дозволено, додатково перевіряється, чи не перевищує запит встановлених для даного користувача або його групи лімітів на використання ресурсу – наприклад, обмежень на об'єм трафіку, кількість запитів у одиницю часу, тощо. Якщо всі перевірки завершуються успішно, програма авторизує запит та надає користувачеві доступ до ресурсу. У протилежному випадку генерується відмова у доступі з поясненням причини. В обох випадках інформація про запит та його параметри зберігається у базі даних для подальшого аналізу та формування статистичних звітів [37].

Паралельно з обробкою запитів користувачів програмний комплекс надає уповноваженим адміністраторам безпеки можливість виконувати керування політиками доступу та моніторинг функціонування системи через спеціальний інтерфейс. Зокрема, адміністратори мають змогу переглядати поточну статистику використання та завантаженості окремих ресурсів, додавати та видаляти користувачів, змінювати їх ролі та права доступу, а також встановлювати нові політики розмежування доступу та ліміти на використання ресурсів. Додатково передбачена можливість перегляду детальних журналів подій (логів), що генеруються компонентами системи. Всі дії адміністраторів із керування відображаються в єдиній базі даних, що забезпечує уникнення конфліктів з поточними політиками доступу та синхронізацію всіх компонентів програмного комплексу [38].

На етапі проектування архітектури програми захисту і керування ресурсами було розроблено узагальнений алгоритм його роботи. Укрупнену структуру даного алгоритму представлено у вигляді блок–схеми (рис. 2.3).

Крок 1. Початок.

Крок 2. Ініціалізація конфігураційних параметрів. Відбувається ініціалізація початкових параметрів та налаштувань програмного комплексу, необхідних для його подальшої роботи.

Крок 3. Встановлення з'єднання з базою даних. Здійснюється встановлення безпечного з'єднання з базою даних, де зберігаються конфігураційні налаштування, облікові дані користувачів та інша необхідна інформація.

Крок 4. Ініціалізація внутрішніх структур даних. Виконується ініціалізація внутрішніх структур даних програми, необхідних для подальшого функціонування системи.

Крок 5. Ідентифікація користувача. Відбувається ідентифікація користувача, який намагається отримати доступ до ресурсів програмного комплексу.

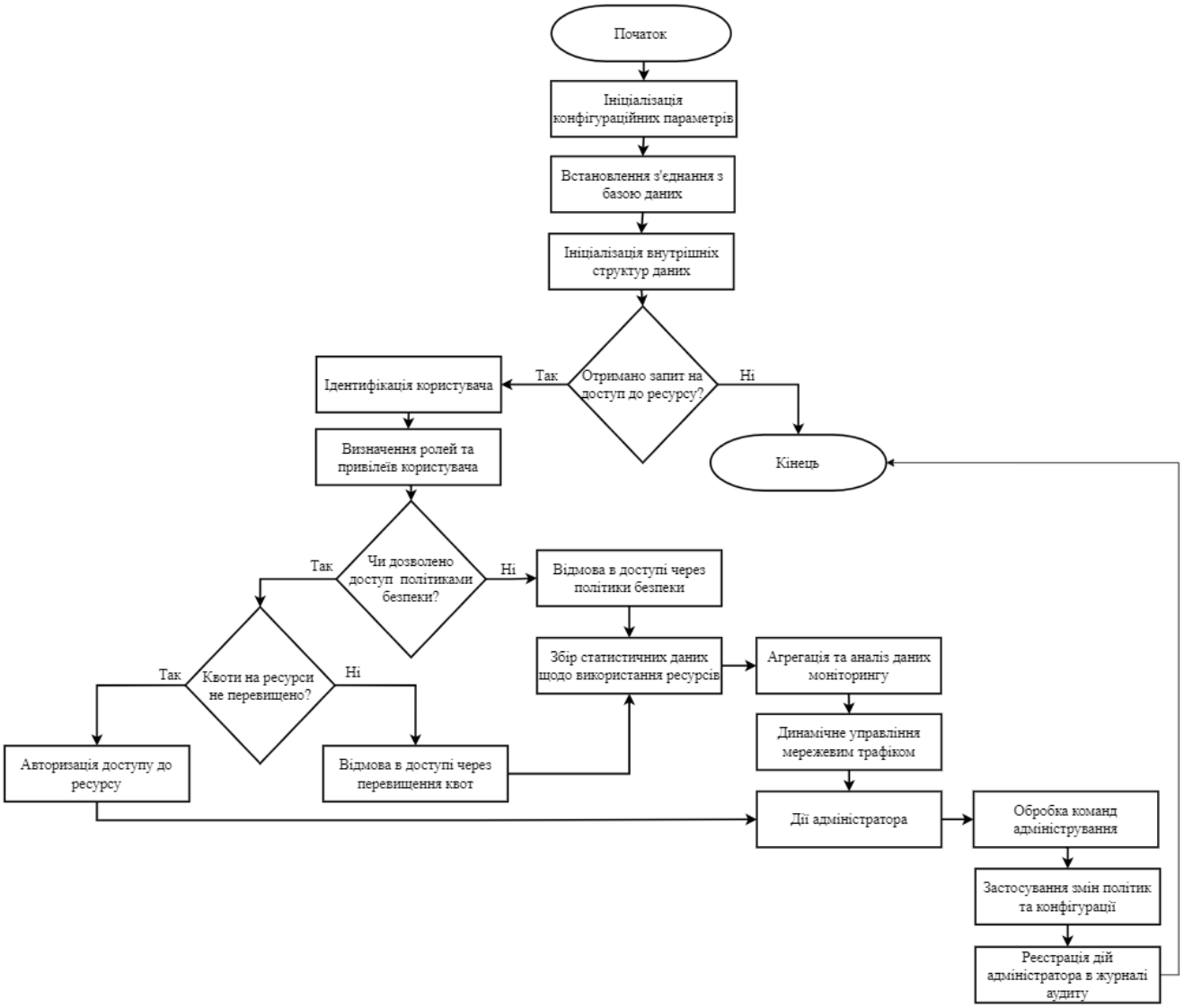


Рисунок 2.3 – Блок схема алгоритму функціонування архітектури розроблюваного ПЗ для захисту та управління розподілом ролей і урахуванням обмеження трафіку

Крок 6. Визначення ролей та привілеїв користувача. На основі ідентифікаційних даних, програма визначає ролі та привілеї користувача відповідно до встановлених політик безпеки.

Крок 7. Чи дозволено доступ користувача до ресурсу? Програма перевіряє, чи наданий доступ користувача до запитуваного ресурсу відповідно до його ролі та привілеїв.

Крок 8. Авторизація доступу до ресурсу. Якщо доступ дозволено, то програма авторизує користувача для взаємодії з ресурсом.

Крок 9. Вихід через заборону доступу. Якщо доступ не дозволено, то користувачу відмовляється в доступі до ресурсу.

Крок 10. Отримано запит на доступ до ресурсу. Програма отримує запит від користувача на доступ до певного ресурсу.

Крок 11. Чи крито на ресурси не перевищено? Програма перевіряє, чи не перевищено встановлених квот на використання ресурсів для даного користувача.

Крок 12. Вихід через перевищення квот. Якщо квоти на ресурси перевищено, то доступ користувачу не надається.

Крок 13. Вихід через перевищення квот на ресурси. Програма надає користувачу доступ до запитуваного ресурсу.

Крок 14. Збір статистичних даних щодо використання ресурсів. Програма збирає статистичні дані про використання ресурсів, включаючи мережевий трафік, завантаження систем тощо.

Крок 15. Агрегація та аналіз даних моніторингу. Зібрані статистичні дані агрегуються та аналізуються для виявлення потенційних загроз та аномалій.

Крок 16. Динамічне управління меседжевим трафіком. На основі результатів аналізу, програма динамічно управляє меседжевим трафіком, застосовуючи необхідні заходи контролю.

Крок 17. Для адміністратора. Результати моніторингу та аналізу доступні адміністратору безпеки для перегляду та подальшого управління.

Крок 18. Обробка команд адміністрування. Адміністратор може виконувати налаштування політик безпеки, конфігурацій системи та інші дії адміністрування.

Крок 19. Застосування змін політик та конфігурації. Внесені адміністратором зміни застосовуються та оновлюються в програмному комплексі.

Крок 20. Кінець.

Отже, після старту виконання програма виконує ініціалізацію всіх своїх компонентів, встановлює з'єднання з базою даних та завантажує в оперативну пам'ять попередньо задані політики розмежування доступу. Цей етап виконується одноразово під час розгортання програмного комплексу або його перезапуску [39].

Основним режимом роботи програми є циклічна обробка запитів від користувачів системи на доступ до захищених інформаційних ресурсів. Для кожного отриманого запиту послідовно виконуються кроки ідентифікації користувача, перевірки його прав доступу, авторизації, моніторингу використання ресурсів та, за необхідності, застосування обмежень (рис 2.4).

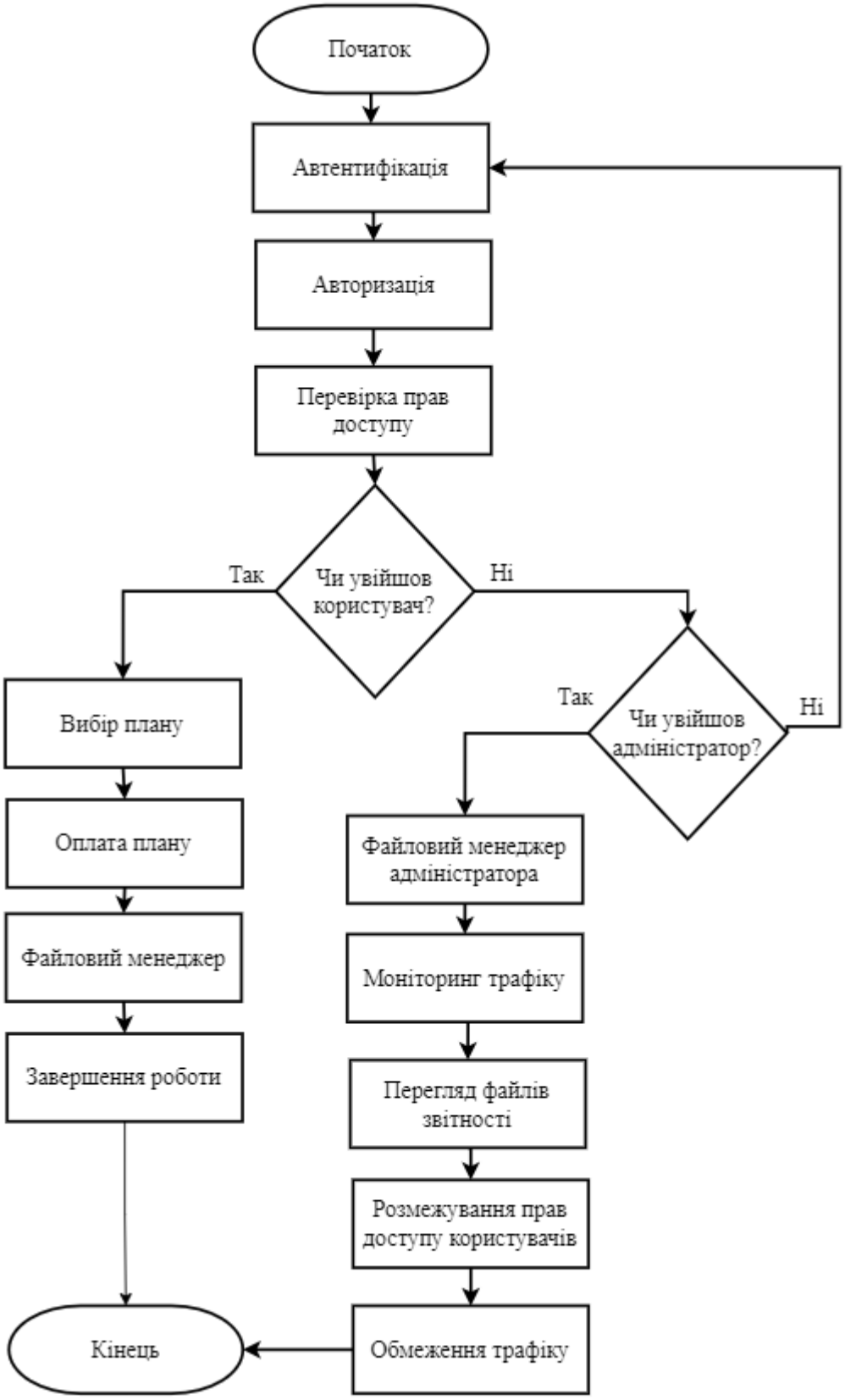


Рисунок 2.4 – Блок схема взаємодії користувача з розробленою програмою

Крок 1. Початок.

Крок 2. Автентифікація. Користувач проходить процес автентифікації, надаючи свої облікові дані (наприклад, ім'я користувача та пароль).

Крок 3. Авторизація. Після успішної автентифікації, програма перевіряє права доступу користувача та визначає його рівень повноважень.

Крок 4. Перевірка прав доступу. Програма перевіряє, чи має користувач достатні права для виконання запитуваної дії.

Крок 5. Чи дозволено користувачу? На основі перевірки прав доступу, програма визначає, чи може користувач виконати запитувану дію.

Крок 6. Вибір плану. Якщо доступ дозволено, користувач обирає бажаний план доступу, що визначає його можливості в системі.

Крок 7. Оплата плану. Користувач здійснює оплату обраного плану доступу через захищене з'єднання.

Крок 8. Файловий менеджер. Після успішної оплати, користувач отримує доступ до файлового менеджера програми.

Крок 9. Чи дозволено адміністратору? Перевіряється, чи має адміністратор дозвіл на виконання певних дій.

Крок 10. Файловий менеджер адміністратора. Якщо доступ адміністратора дозволено, він може виконувати дії у файловому менеджері.

Крок 11. Моніторинг трафіку. Програма здійснює моніторинг мережевого трафіку користувачів.

Крок 12. Перегляд файлів звітності. Адміністратор переглядає файли, що містять статистичну звітність про використання ресурсів.

Крок 13. Розмежування прав доступу користувачів. На основі результатів моніторингу, адміністратор може налаштовувати права доступу користувачів.

Крок 14. Обмеження трафіку. Програма застосовує обмеження на мережевий трафік користувачів відповідно до налаштувань адміністратора.

Крок 15. Кінець.

Представлена блок–схема ілюструє основні етапи взаємодії користувачів з програмним комплексом, включаючи процеси автентифікації, авторизації,

управління доступом до ресурсів, моніторингу активності та налаштування політик безпеки адміністраторами. Такий комплексний підхід забезпечує надійний захист інформаційних активів організації від несанкціонованого доступу та зловживань.

Забезпечення надійного контролю доступу до інформаційних ресурсів та ефективне управління розподілом обчислювальних потужностей є ключовими завданнями під час розробки сучасних програмних систем, що функціонують в умовах зростаючих вимог до інформаційної безпеки. Представлена блок–схема ілюструє комплексний підхід до проєктування архітектури розробленого програмного забезпечення, який дозволяє реалізувати багаторівневий механізм управління доступом користувачів, моніторинг та обмеження використання системних ресурсів відповідно до встановлених політик безпеки [40].

На основі визначених функціональних вимог та архітектурних рішень, для забезпечення надійного зберігання та ефективного управління даними у розробленому програмному комплексі було спроєктовано реляційну базу даних. Рисунок 2.5 відображає UML–діаграму, що наочно демонструє основні сутності бази даних та взаємозв'язки між ними.

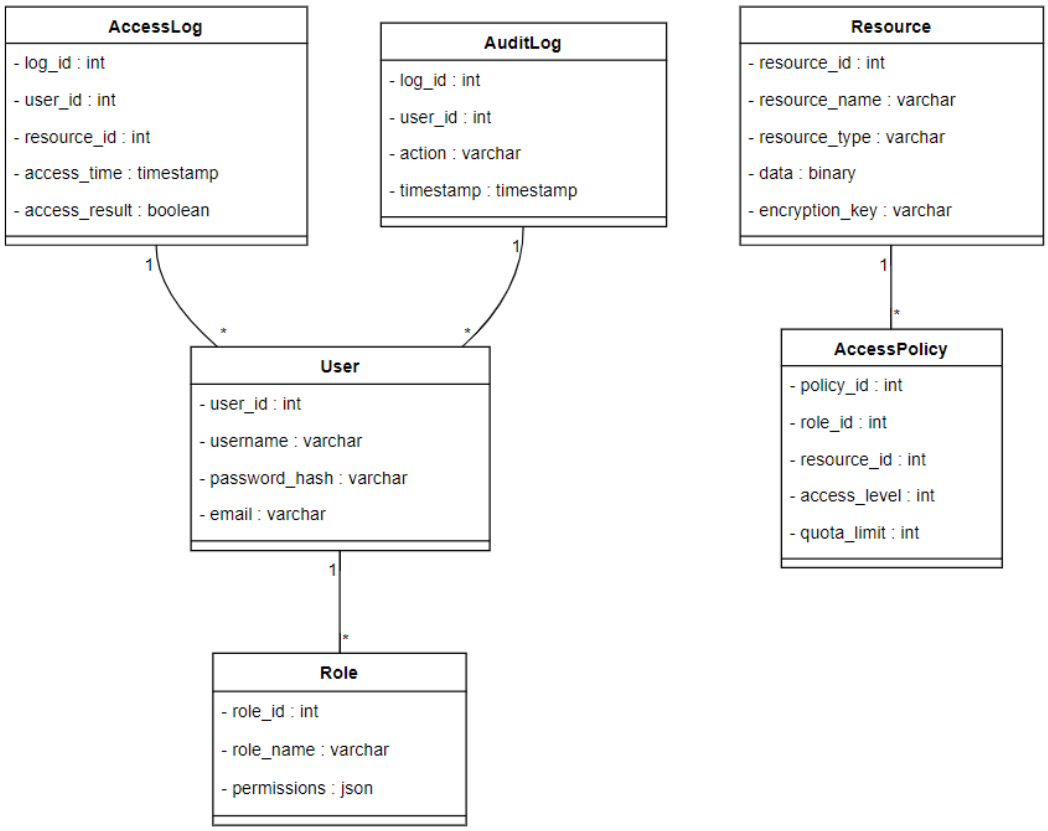


Рисунок 2.5 – UML–діаграма бази даних використаної у розробленому ПЗ

Опис сутностей:

– User – Містить облікові записи користувачів, включаючи ідентифікаційні дані, паролі (у вигляді хешів) та електронні адреси.

– Role – Визначає ролі користувачів у системі та пов'язані з ними права доступу (permissions).

– AccessLog – Зберігає записи про спроби доступу користувачів до ресурсів, включаючи час, результат та ідентифікатор користувача.

– Resource – Містить інформацію про захищені ресурси, такі як файли, дані або системи, включаючи тип, вміст та ключі шифрування.

– AccessPolicy – Визначає політики безпеки, що регулюють рівні доступу та ліміти використання ресурсів для кожної ролі.

– AuditLog – Фіксує всі дії адміністраторів у системі, такі як зміна політик, додавання/видалення користувачів тощо.

Для забезпечення надійного захисту даних, що зберігаються у базі SQLAlchemy, будуть застосовані такі заходи:

– шифрування даних – паролі користувачів будуть збережені у вигляді криптографічних хешів, використовуючи сучасні надійні алгоритми, такі як Bcrypt;

– вміст ресурсів (файлів, даних) буде зашифровано перед збереженням у базі даних, використовуючи надійні криптографічні алгоритми, як–от AES із достатньою довжиною ключа, а ключі шифрування будуть зберігатися у спеціальному, надійно захищеному сховищі AWS Key Management Service;

– розмежування доступу – права доступу користувачів до ресурсів будуть визначатися на основі їхніх ролей, відповідно до налаштованих політик безпеки (сутність AccessPolicy). Адміністратори матимуть можливість гнучко налаштовувати та оновлювати політики доступу для забезпечення актуальності прав користувачів;

– резервне копіювання та відновлення – регулярно виконуватиметься повне резервне копіювання всієї бази даних для забезпечення можливості відновлення у

разі аварій або втрати даних. Процедури резервного копіювання та відновлення будуть ретельно протестовані для гарантії відновлення даних у разі необхідності.

2.3 Обґрунтування вибору мови програмування, інструментів розробки та бази даних

Реалізація програмного забезпечення, що втілює алгоритми управління доступом та контролю використання ресурсів, вимагає ретельного підходу до вибору мови програмування, інструментів розробки та системи управління базами даних (СУБД). Цей вибір має ґрунтуватися на здатності обраних технологій ефективно імплементувати ключові функціональні вимоги до системи, забезпечуючи при цьому високу продуктивність, безпеку, масштабованість та зручність розробки.

Для визначення найбільш підходящої мови програмування було проведено порівняльний аналіз кількох популярних мов, що зазвичай використовуються для розробки програмних систем із подібним функціоналом, а саме: Python, Java, C++ та C#. Ці мови були обрані з огляду на їх широке застосування в галузі інформаційної безпеки, наявність розвинених бібліотек та інструментів, а також можливість ефективної реалізації алгоритмів управління доступом та контролю використання ресурсів. Порівняння мов програмування наведено таблиці 2.1 [41].

Таблиця 2.1 – Порівняння мов програмування за основними критеріями

Критерій	Python	Java	C++
Простота синтаксису	+	–	–
Швидкість виконання	–	+	+
Підтримка ООП	+	+	+
Наявність бібліотек для безпеки	+	+	+
Переносність	+	+	+
Швидкість розробки	+	–	–
Кросплатформеність	+	+	+

Проаналізувавши характеристики зазначених мов програмування було визначено, що мова Python найбільш оптимально відповідає вимогам розробки

програмного забезпечення для реалізації алгоритмів управління доступом та контролю використання ресурсів. Python вирізняється простотою синтаксису, наявністю широкого спектру бібліотек для забезпечення безпеки, а також високою швидкістю розробки, що є важливим аспектом для виконання поставленого завдання.

Для реалізації графічного інтерфейсу користувача (GUI) програмного комплексу було вирішено застосувати веб–технології, а саме HTML та CSS. Це дозволить створити зручний, кросплатформений та адаптивний веб–інтерфейс, який доповнюватиме основну функціональність, реалізовану на мові Python.

При виборі середовища розробки були розглянуті найпопулярніші інтегровані середовища (IDE), які підтримують розробку додатків на мові Python, а також містять необхідні інструменти для створення веб–інтерфейсів: Visual Studio Code, PyCharm, Spyder та Jupyter Notebook. Порівняння середовищ програмування наведено таблиці 2.2 [42].

Таблиця 2.2 – Порівняння середовища програмування за основними критеріями.

Критерій	Visual Studio Code	PyCharm	Spyder	Jupyter Notebook
Підтримка Python	+	+	+	+
Підтримка HTML/CSS	+	+	–	–
Зручність інтерфейсу	+	+	+	–
Налаштованість середовища	+	+	–	+
Інтеграція з системами контролю версій	+	+	–	+
Наявність debugger	+	+	+	–
Підтримка віртуальних середовищ	+	+	+	+

Аналізуючи критерії зазначених середовищ розробки, було визначено, що найбільш оптимальним вибором для розробки програмного комплексу є середовище Visual Studio Code.

Visual Studio Code, розроблене компанією Microsoft, є сучасним, кросплатформеним та високо налаштовуваним IDE, яке надає всі необхідні інструменти для ефективної розробки на мові Python, а також для створення веб–інтерфейсів на основі HTML та CSS. Воно має інтуїтивно зрозумілий інтерфейс, широкі можливості налаштування, підтримує систему контролю версій Git, має вбудований потужний дебагер та забезпечує підтримку віртуальних середовищ. Важливим фактором є те, що Visual Studio Code є безкоштовним та відкритим програмним забезпеченням.

Для ефективного зберігання та управління даними, необхідними для функціонування програмного комплексу, що реалізує алгоритми управління доступом та контролю використання ресурсів, потрібно обрати відповідну систему управління базами даних (СУБД). Вибір СУБД має ґрунтуватися на здатності забезпечити надійне та структуроване збереження інформації, а також можливість ефективної взаємодії з іншими компонентами програмного забезпечення, тому було проаналізовано популярні системи управління базами даних: SQLAlchemy, MySQL, SQLite та MongoDB. Порівняння відомих СУБД наведено таблиці 2.3 [43].

Таблиця 2.3 – Порівняння СУБД за основними критеріями.

Критерій	SQLAlchemy	MySQL	SQLite	MongoDB
Підтримка SQL	+	+	+	–
Масштабованість	+	+	–	+
Надійність	+	+	+	+
Швидкодія	+	+	+	+
Безпека даних	+	+	+	+
Кросплатформеність	+	+	+	+
Підтримка ORM	+	+	+	+

Порівняння характеристик зазначених СУБД показало, що реляційна база даних SQLAlchemy є найбільш оптимальним вибором для розроблюваного програмного комплексу.

SQLAlchemy – потужна, масштабована та надійна реляційна СУБД, яка забезпечує повну підтримку SQL, високу швидкодію та надійний захист даних. Вона є кросплатформеною, безкоштовною та має широку підтримку в розробницькому середовищі, що дозволяє легко інтегрувати її з обраними технологіями розробки.

Отже, враховуючи вимоги до надійності, безпеки та масштабованості, а також зручність інтеграції з обраними технологіями розробки, система управління базами даних SQLAlchemy була обрана як найбільш оптимальне рішення для забезпечення ефективного збереження та управління даними у розроблюваному програмному комплексі.

Таким чином, враховуючи функціональні вимоги до системи, характеристики розглянутих технологій та їх здатність забезпечити ефективну реалізацію алгоритмів управління доступом та контролю використання ресурсів, для розробки програмного комплексу було обрано мову програмування Python, середовище розробки Visual Studio Code, веб–технології HTML та CSS для створення графічного інтерфейсу користувача, а також систему управління базами даних SQLAlchemy. Такий вибір технологій дозволить створити високопродуктивне, безпечне та масштабоване програмне забезпечення, що відповідає поставленим вимогам та забезпечує зручність розробки та користувацького досвіду.

2.4 Висновки до розділу 2

У другому розділі було розглянуто процес розробки програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Розроблено підхід до управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Запропонований підхід базується на реалізації розширеної моделі ролей, динамічному управлінні пропускною здатністю мережі, моніторингу та аналізі використання ресурсів, а також створенні зручного інтерфейсу керування. Ключовими алгоритмами, що забезпечують

функціонування системи, є алгоритм управління доступом на основі ролей та алгоритм обмеження мережевого трафіку.

Спроєктовано архітектуру розроблюваної програми захисту та описано взаємодію з базами даних. Розроблений узагальнений алгоритм функціонування ПЗ, який включає ініціалізацію компонентів, обробку запитів користувачів, ідентифікацію та авторизацію, моніторинг використання ресурсів та застосування обмежень.

Обґрунтовано вибір мови програмування, інструментів розробки та бази даних. Для реалізації програмного комплексу було обрано мову програмування Python, яка найбільш оптимально відповідає вимогам розробки, забезпечуючи простоту синтаксису, наявність широкого спектру бібліотек для безпеки та високу швидкість розробки. Для створення графічного інтерфейсу користувача вирішено застосувати веб–технології HTML та CSS. Як середовище розробки обрано Visual Studio Code, яке надає всі необхідні інструменти для ефективної розробки на мові Python та створення веб–інтерфейсів. Для зберігання та управління даними обрано реляційну базу даних SQLAlchemy, яка забезпечує надійність та зручність.

Таким чином, у другому розділі було розроблено комплексний підхід до управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, спроєктовано архітектуру програмного забезпечення та обґрунтовано вибір технологій розробки. Отримані результати створюють міцну основу для подальшої реалізації та тестування розробленого програмного комплексу, що дозволить ефективно вирішувати завдання забезпечення інформаційної безпеки та оптимального використання ресурсів в організаціях.

3 ПРАКТИЧНА РЕАЛІЗІЯ ПРОГРАМИ ЗАХИСТУ ТА УПРАВЛІННЯ РОЗПОДІЛОМ РЕСУРСІВ З РОЛЬОВИМ ДОСТУПОМ І УРАХУВАННЯМ ОБМЕЖЕННЯ ТРАФІКУ

3.1 Опис реалізація програмної логіки компонентів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

На основі розроблених у попередньому розділі блок–схем алгоритмів роботи ПЗ було описано реалізацію програмної логіки компонентів захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Для реалізації використано обрані мови програмування, інструменти розробки та систему управління базами даних.

Наступний фрагмент коду відповідає за ініціалізацію конфігураційних параметрів та встановлення з'єднання з базою даних:

```
import psycopg2
import yaml
import pyshark
def init_config():
   with open('config.yaml', 'r') as file:
      config = yaml.safe_load(file)
   return config
```

Цей фрагмент коду відповідає за встановлення з'єднання з базою даних PostgreSQL на основі конфігураційних параметрів:

```
def connect_to_db(config):
   conn = psycopg2.connect(
      host=config['db_host'],
      port=config['db_port'],
      database=config['db_name'],
      user=config['db_user'],
      password=config['db_password']
   )
   return conn
config = init_config()
db_conn = connect_to_db(config)
```

Цей фрагмент коду відповідає за аутентифікацію користувача на основі наданих імені користувача та пароля:

```
def authenticate_user(username, password):
    cur = db_conn.cursor()
    cur.execute("SELECT * FROM users WHERE username = %s", (username,))
    user = cur.fetchone()
    cur.close()
    if user and user['password'] == password:
        return user
    return None
```

Далі наведено фрагмент коду, який відповідає за визначення ролей користувача і привілеїв:

```
def get_user_roles(user_id):
    cur = db_conn.cursor()
    cur.execute("SELECT role_id FROM user_roles WHERE user_id = %s", (user_id,))
    role_ids = [row[0] for row in cur.fetchall()]
    cur.close()
    return role_ids
def get_role_permissions(role_ids):
    cur = db_conn.cursor()
    cur.execute("SELECT permission FROM role_permissions WHERE role_id IN %s", (tuple(role_ids),))
    permissions = [row[0] for row in cur.fetchall()]
    cur.close()
    return permissions
user = authenticate_user(username, password)
if user:
    user_roles = get_user_roles(user['id'])
    user_permissions = get_role_permissions(user_roles)
else:
    print("Authentication failed")
    exit(1)
```

Фрагмент коду, що наведено далі відповідає за перевірку прав доступу користувача до запитуваного ресурсу:

```
def check_access(user_permissions, resource):
    if resource in user_permissions:
        return True
    return False
if check_access(user_permissions, requested_resource):
    print("Access granted")
    access_resource(requested_resource)
else:
    print("Access denied")
```

Цей фрагмент коду відповідає за моніторинг мережевого трафіку в режимі реального часу за допомогою бібліотеки `pyshark` та класифікацію трафіку на основі довірених IP–адрес та протоколу:

```
def monitor_network_traffic():
    capture = pyshark.LiveCapture(interface='eth0')
    for packet in capture.sniff_continuously():
        source_ip = packet['ip'].src
        destination_ip = packet['ip'].dst
        protocol = packet.transport_layer
        traffic_class = classify_traffic(source_ip, destination_ip, protocol)
        store_traffic_data(traffic_class, packet)
def classify_traffic(source_ip, destination_ip, protocol):
    if source_ip in trusted_ips and destination_ip in trusted_ips:
        return "trusted"
    elif protocol == "HTTP" or protocol == "HTTPS":
        return "web"
    else:
        return "unknown"
```

Наступний фрагмент коду відповідає за збереження класифікованих даних про трафік у базі даних:

```
def store_traffic_data(traffic_class, packet):
    cur = db_conn.cursor()
    cur.execute("INSERT INTO traffic_data (class, source_ip, destination_ip, protocol, timestamp) VALUES (%s, %s, %s, %s, %s)",
                (traffic_class, packet['ip'].src, packet['ip'].dst, packet.transport_layer, packet.sniff_time))
    db_conn.commit()
    cur.close()
```

Цей фрагмент коду відповідає за порівняння класу трафіку та ідентифікатора користувача з політиками безпеки, збереженими в базі даних:

```
def compare_with_security_policies(traffic_class, user_id):
    cur = db_conn.cursor()
    cur.execute("SELECT * FROM security_policies WHERE traffic_class = %s AND user_id = %s", (traffic_class, user_id))
    policy = cur.fetchone()
    cur.close()
    if policy:
        return policy['bandwidth_limit'], policy['connection_limit']
    return None, None
```

Наведений фрагмент коду відповідає за застосування обмежень трафіку для користувача на основі обмежень пропускної здатності та кількості з'єднань:

```
def apply_traffic_restrictions(user_id, traffic_class, bandwidth_limit, connection_limit):
    cur = db_conn.cursor()
    cur.execute("UPDATE user_traffic SET bandwidth_used = bandwidth_used + %s, connections_used = connections_used + 1 WHERE user_id = %s",
                (packet.length, user_id))
    db_conn.commit()
    cur.close()
```

```
    if bandwidth_limit is not None and connection_limit is not None:
        cur = db_conn.cursor()
        cur.execute("SELECT * FROM user_traffic WHERE user_id = %s", (user_id,))
        user_traffic = cur.fetchone()
        cur.close()
        if user_traffic['bandwidth_used'] > bandwidth_limit or user_traffic['connections_used'] > connection_limit:
            block_traffic(user_id)
bandwidth_limit, connection_limit = compare_with_security_policies(traffic_class, user_id)
if bandwidth_limit and connection_limit:
    apply_traffic_restrictions(user_id, traffic_class, bandwidth_limit, connection_limit)
```

Цей фрагмент коду відповідає за обробку команд адміністрування на основі отриманої команди та параметрів:

```
def handle_admin_command(command, params):
    if command == 'update_security_policy':
        update_security_policy(params['policy_id'], params['bandwidth_limit'], params['connection_limit'])
    elif command == 'add_user':
        add_user(params['username'], params['password'], params['role'])
    elif command == 'remove_user':
        remove_user(params['user_id'])
```

Наступний фрагмент коду відповідає за збір даних про використання ресурсів:

```
def collect_resource_usage_stats():
    cpu_usage = psutil.cpu_percent()
    memory_usage = psutil.virtual_memory().percent
    storage_usage = psutil.disk_usage('/').percent
    store_resource_usage_stats(cpu_usage, memory_usage, storage_usage)
```

Цей фрагмент коду відповідає за збереження інформацію. про раніше використаних даних, у базі даних:

```
def store_resource_usage_stats(cpu_usage, memory_usage, storage_usage):
    cur = db_conn.cursor()
    cur.execute("INSERT INTO resource_usage (cpu_usage, memory_usage, storage_usage, timestamp) VALUES
(%s, %s, %s, %s)",
                (cpu_usage, memory_usage, storage_usage, datetime.now()))
    db_conn.commit()
    cur.close()
```

Наступний код відповідає за динамічне управління мережевим трафіком на основі результатів аналізу:

```
def identify_traffic_patterns(analyzed_data):
    traffic_patterns = {}
    for timestamp, data in analyzed_data.items():
        if data['avg_cpu_usage'] > 80 or data['avg_memory_usage'] > 80:
            traffic_patterns[timestamp] = 'high_usage'
        else:
```

```
            traffic_patterns[timestamp] = 'normal_usage'
    return traffic_patterns
def detect_anomalies(analyzed_data):
    anomalies = []
    for timestamp, data in analyzed_data.items():
        if data['avg_cpu_usage'] > 95 or data['avg_memory_usage'] > 95:
            anomalies.append(timestamp)
    return anomalies
def apply_traffic_management_rules(traffic_patterns, anomalies):
    for timestamp, pattern in traffic_patterns.items():
        if pattern == 'high_usage':
            limit_bandwidth(timestamp)
        else:
            restore_bandwidth(timestamp)
    for anomaly_timestamp in anomalies:
        block_traffic(anomaly_timestamp)
dynamic_traffic_management(analyzed_data)
```

Наступний код відповідає за динамічне управління мережевим трафіком на основі результатів аналізу:

```
def get_admin_command():
    command = input("Enter admin command: ")
    params = {}
    if command == 'update_security_policy':
        params['policy_id'] = input("Enter policy ID: ")
        params['bandwidth_limit'] = input("Enter bandwidth limit: ")
        params['connection_limit'] = input("Enter connection limit: ")
    elif command == 'add_user':
        params['username'] = input("Enter username: ")
        params['password'] = input("Enter password: ")
        params['role'] = input("Enter user role: ")
    elif command == 'remove_user':
        params['user_id'] = input("Enter user ID: ")
    return command, params
```

Наступний код відповідає за динамічне управління мережевим трафіком на основі результатів аналізу:

```
def get_report_files():
    report_files = ['report1.txt', 'report2.txt', 'report3.txt']
    return report_files
def display_report_files(report_files):
    print("Report Files:")
    for file in report_files:
        print(file)
def save_report(report):
    print("Report saved successfully.")
```

Загалом, представлений код є програмною реалізації спроектованих вище блок–схем алгоритмів, що лежать у основі ефективного функціонування

розроблюваного ПЗ для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Описано фрагменти коду модулів управління доступом та контролю використання ресурсів з рольовим доступом і урахуванням обмеження трафіку.

Загалом, код демонструє різні аспекти функціональності, такі як аутентифікація користувачів, перевірка прав доступу, моніторинг мережевого трафіку, застосування політик безпеки, збір та аналіз статистичних даних, динамічне управління трафіком, обробка команд адміністрування та генерація звітів.

3.2 Тестування розробленої програми та надання інструкцій користувачу

Тестування розробленої програми є невід'ємною частиною процесу розробки, що дозволяє перевірити коректність функціонування системи та виявити потенційні помилки чи недоліки. Важливим аспектом тестування є перевірка роботи користувацького інтерфейсу, який забезпечує взаємодію між користувачем та системою. На рисунку 3.1 ілюстровано приклад екрану входу користувача до системи.

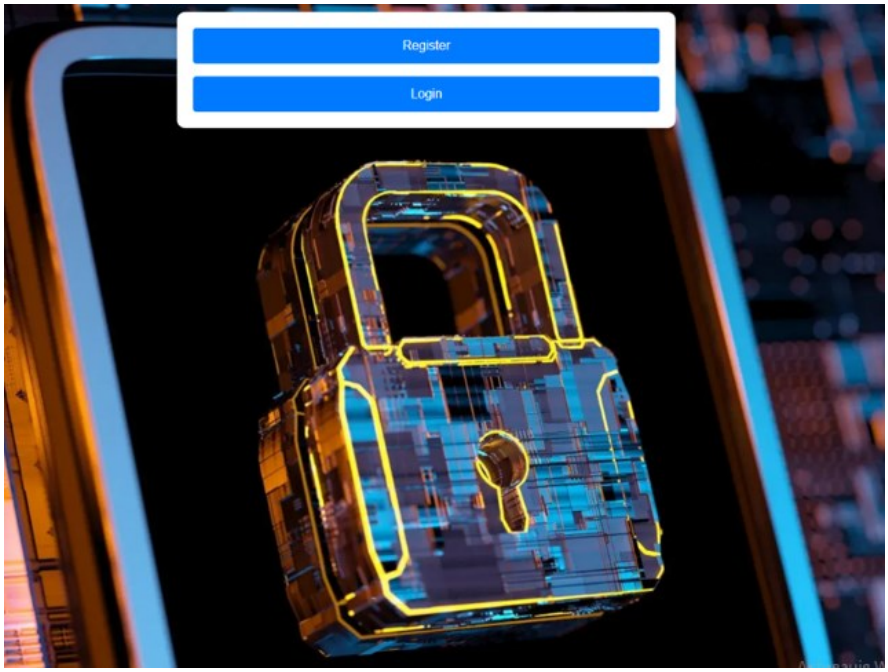


Рисунок 3.1 – Екран входу користувача до системи

Зображення демонструє приклад інтерфейсу входу користувача до системи на фоні мальовничого пейзажу. На передньому плані розташоване вікно входу, яке містить поля для введення імені користувача та пароля. Наявність полів для автентифікації користувача є стандартною практикою в системах з рольовим доступом, що дозволяє ідентифікувати користувача та надати йому відповідні права та привілеї в системі. Використання графічних елементів, таких як фонове зображення, надає інтерфейсу привабливого та дружнього вигляду, що покращує досвід взаємодії користувача з системою.

Однією з найважливіших функцій розробленого програмного забезпечення є можливість реєстрації нових користувачів у системі. Процес реєстрації дозволяє користувачам створювати персональні облікові записи, що надає їм доступ до відповідних ресурсів та функціональних можливостей системи згідно з їхніми ролями та привілеями. На рисунку 3.2 ілюстровано форму реєстрації нового користувача в системі.

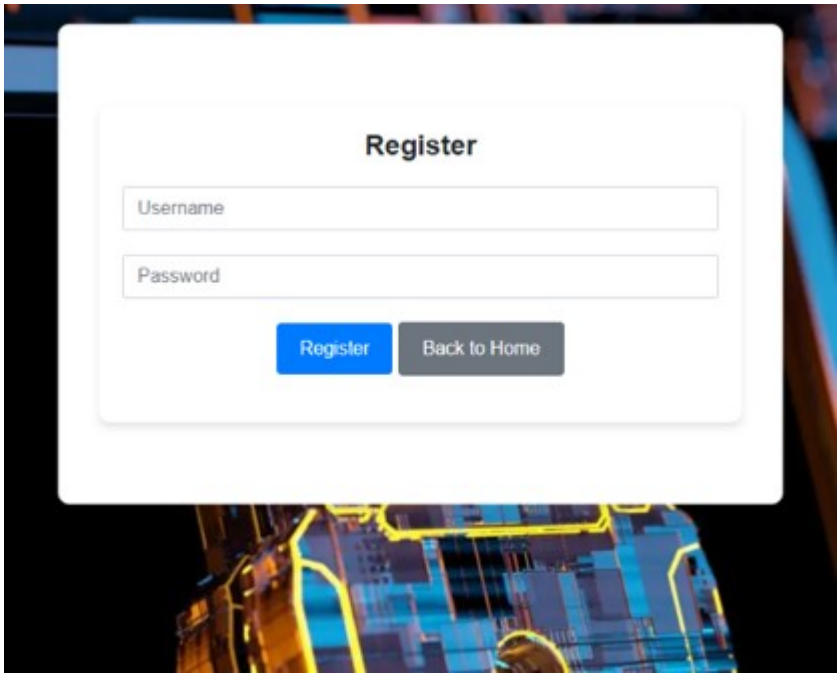


Рисунок 3.2 – Форма реєстрації нового користувача

Зображення демонструє форму реєстрації нового користувача, яка містить поля для введення необхідної інформації, такої як ім'я користувача та пароль. Наявність

цих полів є обов'язковою для створення нового облікового запису в системі. Кнопки "Register" та "Back to Home" дозволяють користувачу підтвердити реєстрацію або повернутися на головну сторінку системи.

Після реєстрації користувач може авторизуватися в системі, використовуючи свої облікові дані. Процес авторизації є важливим етапом для отримання доступу до персоналізованих функцій та ресурсів системи. На рисунку 3.3 представлено форму авторизації користувача.

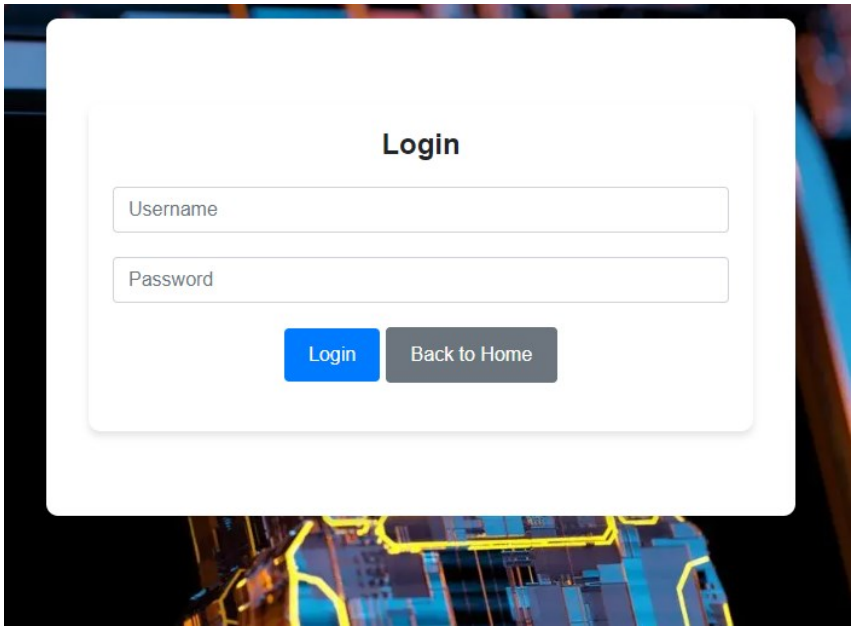


Рисунок 3.3 – Форма авторизації користувача

На зображенні показано форму авторизації користувача, яка містить поля для введення імені користувача та пароля. Користувач повинен ввести свої облікові дані, щоб отримати доступ до системи. Кнопка "Login" дозволяє підтвердити введені дані та виконати вхід, тоді як кнопка "Back to Home" дає можливість повернутися на головну сторінку без авторизації.

Після успішної авторизації користувач потрапляє на сторінку свого профілю, де може переглянути та редагувати персональні дані. рисунку 3.4 представлено приклад сторінки профілю користувача.

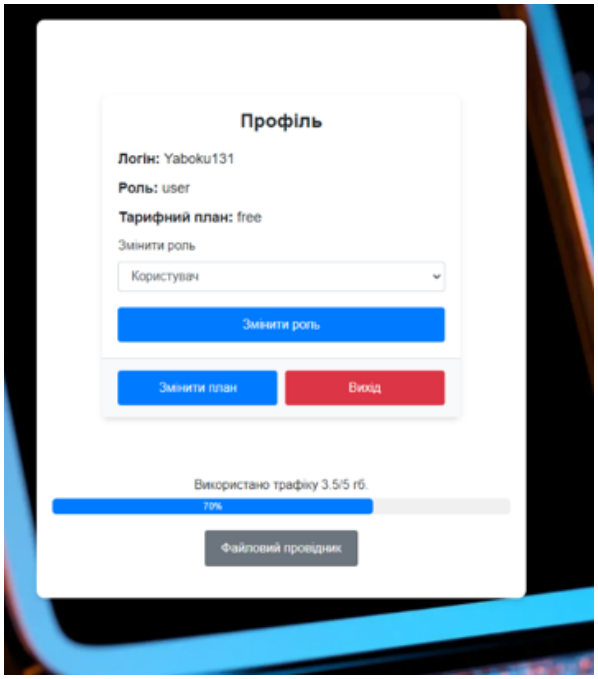


Рисунок 3.4 – Сторінка профілю користувача розробленої програми

На зображенні показано сторінку профілю користувача, яка містить поля з його персональними даними, такими як ім'я та роль у системі. Користувач має можливість редагувати свій профіль, натиснувши кнопку "Змінити дані". Також на сторінці присутня інформація про поточний тарифний план користувача та кнопка "Змінити тариф" для управління підпискою.

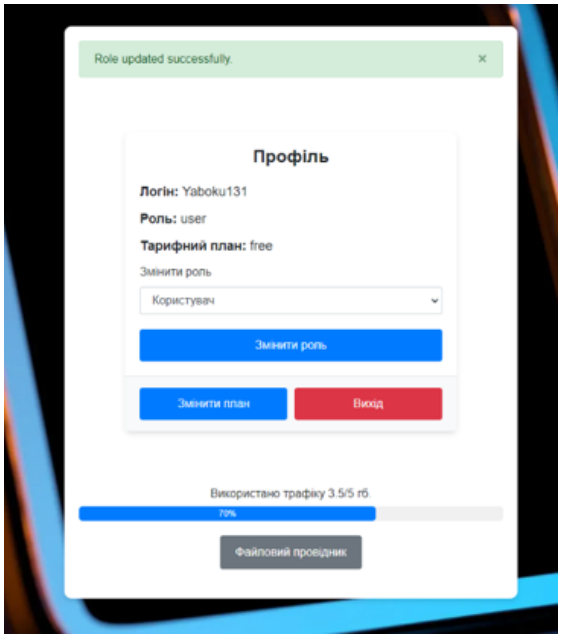


Рис 3.5 – Форма зміни ролі

Для забезпечення гнучкості та адаптивності системи до потреб користувачів, розроблене програмне забезпечення передбачає можливість вибору та зміни тарифного плану. На рисунку 3.6 зображено форму вибору тарифного плану користувача.

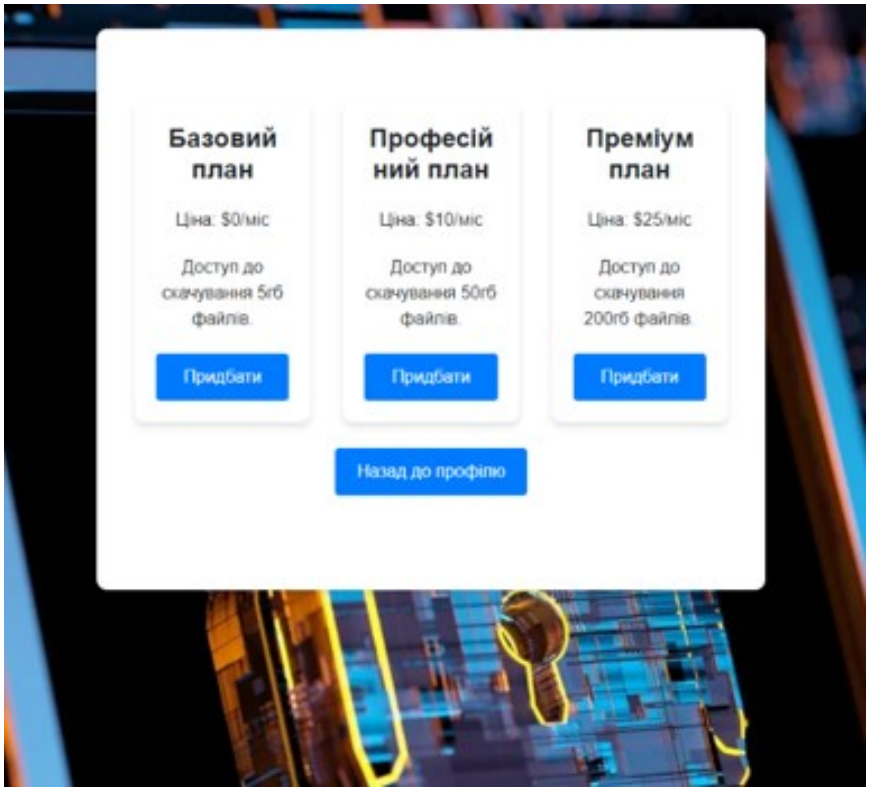


Рисунок 3.6 – Форма вибору тарифного плану користувача

Зображення ілюструє форму вибору тарифного плану, яка надає користувачу можливість обрати між двома доступними опціями – "Звичайний тариф" та "Розширений тариф". Користувач може переглянути деталі кожного тарифного плану, такі як ціна та доступні функції, що допомагає прийняти зважене рішення щодо вибору відповідного плану. Кнопка "Змінити тариф" дозволяє підтвердити вибір та застосувати обраний тарифний план до облікового запису користувача.

Розроблене програмне забезпечення надає користувачам зручний інструмент для управління файлами та ресурсами системи. Файловий менеджер дозволяє переглядати, завантажувати та організовувати файли, забезпечуючи ефективну роботу з даними та їх обмін між користувачами. На рисунку 3.7 представлено інтерфейс файлового менеджера користувача.

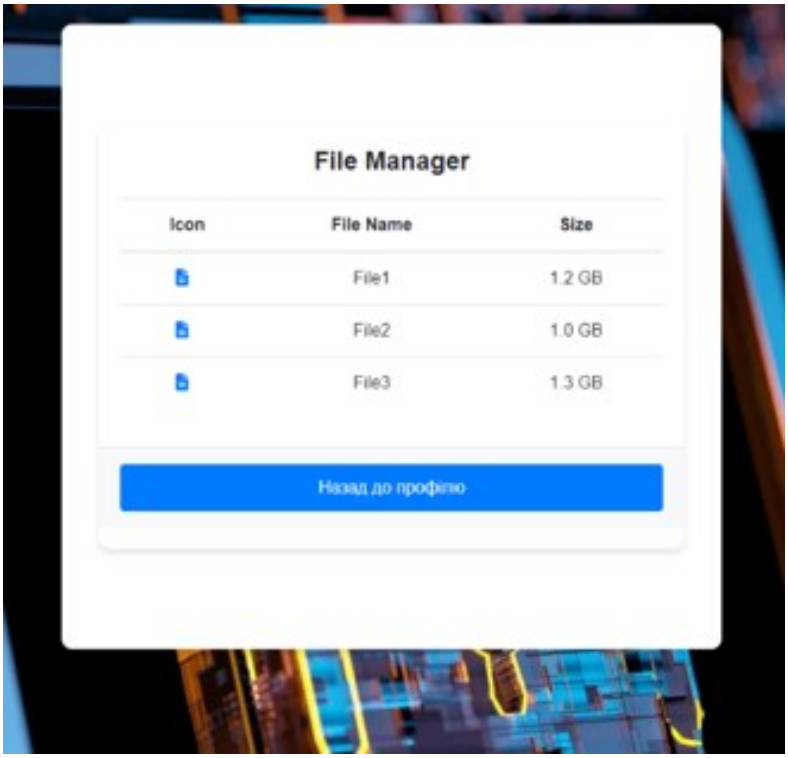


Рисунок 3.7 – Сторінка файлового провіднику

Зображення демонструє файловий менеджер користувача, який відображає список доступних файлів та їх основні характеристики, такі як ім'я файлу, розмір та дату модифікації. Користувач має можливість переглядати файли, сортувати їх за різними критеріями та виконувати дії з ними, наприклад, завантажувати файли на своє локальне сховище. Кнопка "Назад до профілю" дозволяє повернутися до сторінки профілю користувача.

Файловий менеджер є важливим компонентом розробленого програмного забезпечення, оскільки він забезпечує централізоване управління файлами та ресурсами, дозволяючи користувачам ефективно організовувати та обмінюватися даними в рамках системи. Ця функціональність сприяє підвищенню продуктивності роботи користувачів та спрощує процес управління файлами в контексті рольового доступу та обмеження трафіку.

Сторінку профілю користувача системи показано на рисунку 3.8. Сторінка містить інформацію про користувача, таку як логін (Yaboku131), роль (admin) та поточний тарифний план (free). Якщо у користувача є відповідні права згідно

розробленої програми та рольового доступу, тоді він має можливість змінити свою роль на адміністраторську, натиснувши кнопку "Змінити роль".

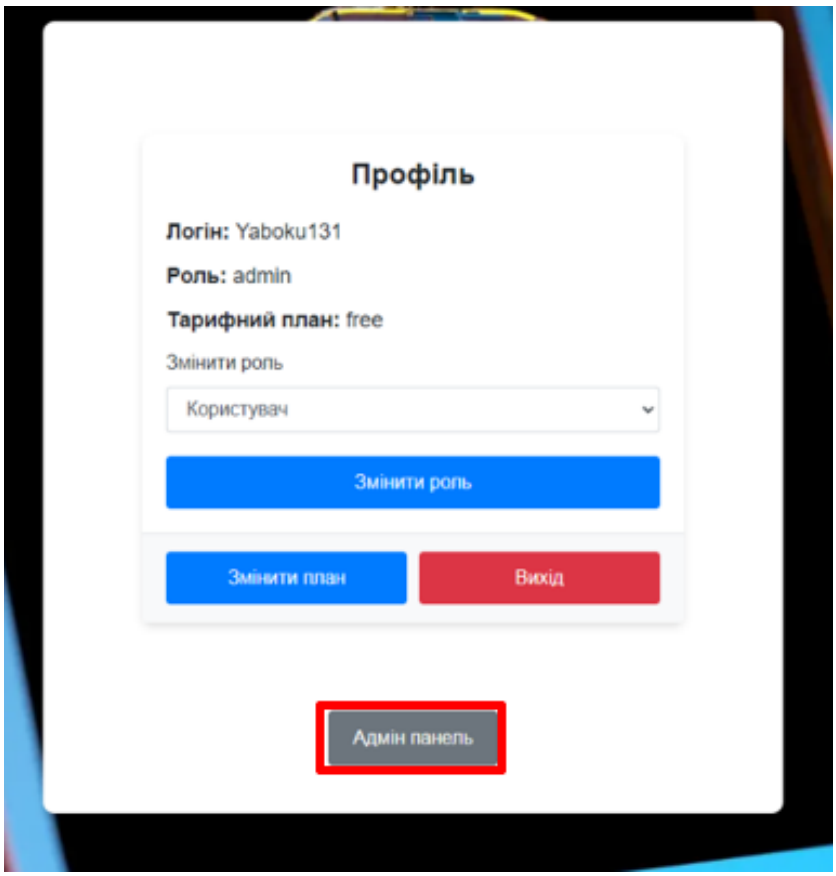


Рисунок 3.8 – Навігація користувача до адмін панелі

Також присутні кнопки "Змінити план" та "Вихід" для редагування тарифного плану та виходу з облікового запису відповідно. Червона кнопка "Адмін панель" дозволяє адміністратору перейти до панелі адміністрування системи.

Рисунок 3.9 демонструє адмін панель системи, яка надає адміністратору доступ до інформації про користувачів та їх використання трафіку. У таблиці представлено список користувачів з даними про їх логін, використаний трафік, тарифний план та дії, які можна виконати (завантажити). Адміністратор може аналізувати цю інформацію, щоб виявляти підозрілу активність, наприклад, надмірне використання трафіку, та вживати відповідних заходів, а саме обмежувати трафік чи встановлювати відповідні квоти на використання трафіку. Кнопка "Назад до профілю" дозволяє повернутися до сторінки профілю адміністратора.

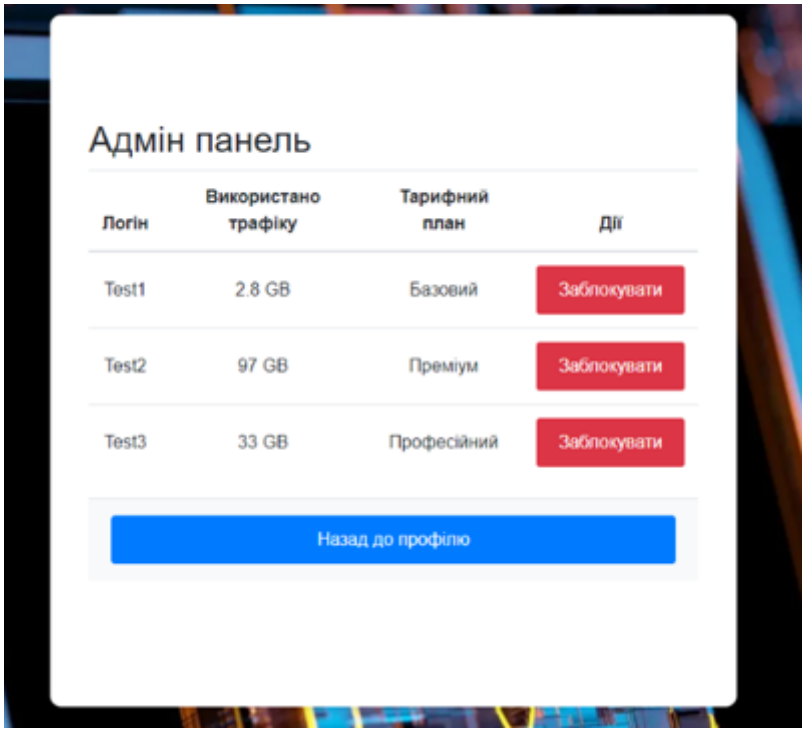


Рисунок 3.9 – Форма адмін панелі програми захисту

Функціонал адмін панелі відіграє важливу роль у забезпеченні захисту та управління розподілом ресурсів системи. Адміністратор має можливість відстежувати та контролювати використання трафіку користувачами, запобігаючи потенційним зловживанням та оптимізуючи розподіл ресурсів відповідно до встановлених обмежень та ролей користувачів.

Таким чином, представлені елементи користувацького інтерфейсу демонструють ключові функціональні можливості розробленого програмного забезпечення, такі як реєстрація нових користувачів, авторизація в системі та управління персональними даними. Ці функції забезпечують безпечний доступ до системи, персоналізацію досвіду користувача та можливість керування обліковим записом відповідно до індивідуальних потреб.

3.3 Висновки до розділу 3

У третьому розділі було проведено тестування розробленого програмного забезпечення для управління розподілом ресурсів з рольовим доступом і

урахуванням обмеження трафіку, а також надано інструкції користувачу щодо роботи з системою.

Процес тестування охоплював перевірку основних функціональних можливостей системи, таких як реєстрація та авторизація користувачів, управління персональними даними, вибір тарифного плану та робота з файловим менеджером. Результати тестування підтвердили коректність роботи програмного забезпечення та його відповідність висунутим вимогам.

Інструкції користувачу були представлені у вигляді покрокових описів та ілюстрацій ключових елементів інтерфейсу системи. Вони охоплювали процеси реєстрації, авторизації, редагування профілю, вибору тарифного плану та використання файлового менеджера. Надані інструкції забезпечують користувачів необхідною інформацією для ефективної взаємодії з розробленим програмним забезпеченням.

Таким чином, тестування та інструкції користувачу, представлені в даному розділі, демонструють функціональність та зручність використання розробленої системи управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Вони підтверджують готовність програмного забезпечення до практичного застосування та його здатність вирішувати поставлені завдання з управління ресурсами та забезпечення безпеки в організаціях.

ВИСНОВКИ

У даній бакалаврській дипломній роботі було розглянуто актуальну проблему захисту та управління розподілом ресурсів в інформаційних системах організацій. У ході дослідження проаналізовано сучасний стан проблеми, розроблено програмний комплекс для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, а також проведено його тестування та оцінку ефективності.

У першому розділі було проаналізовано теоретичні основи захисту та управління ресурсами в інформаційних системах. Розглянуто базові поняття інформаційної безпеки, рольового управління доступом та контролю мережевого трафіку. Проведено аналіз існуючих підходів та рішень у цій галузі, виявлено їх переваги та недоліки. На основі проведеного аналізу сформульовано завдання розробки програмного комплексу для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку.

У другому розділі запропоновано підхід до управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Розроблено архітектуру програмного комплексу, яка включає компоненти для реалізації рольового управління доступом, контролю мережевого трафіку, моніторингу та аналізу використання ресурсів. Описано взаємодію між компонентами системи та базами даних. Обґрунтовано вибір технологій, мов програмування та інструментів розробки для реалізації програмного комплексу.

Третій розділ присвячено практичній реалізації розробленого підходу та архітектури програмного комплексу. Створено програмні модулі для рольового управління доступом, контролю мережевого трафіку, моніторингу та аналізу використання ресурсів. Реалізовано алгоритми динамічного управління пропускною здатністю мережі на основі пріоритетів користувачів та типів трафіку, а також механізми фільтрації та інспекції трафіку для виявлення потенційних загроз безпеці. Розроблено зручний та інтуїтивно зрозумілий інтерфейс користувача для адміністрування системи та управління політиками безпеки.

Проведено тестування розробленого програмного комплексу на відповідність функціональним вимогам та вимогам безпеки. Перевірено коректність роботи компонентів системи, включаючи рольове управління доступом, контроль мережевого трафіку, моніторинг та аналіз використання ресурсів. Оцінено ефективність розробленого підходу щодо забезпечення безпеки та оптимального розподілу ресурсів в інформаційних системах організацій. Підготовлено інструкції користувачу для роботи з програмним комплексом.

Результатом роботи стала успішна розробка програмного комплексу для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Запропонований підхід забезпечує гнучкість та масштабованість рольового доступу, динамічне управління мережевим трафіком на основі пріоритетів та контекстної інформації, а також можливість моніторингу та аналізу використання ресурсів для виявлення аномалій та потенційних загроз безпеці.

Опираючись на проведені дослідження та отримані результати, можна стверджувати, що в даній бакалаврській дипломній роботі досягнуто поставленої мети та виконано всі завдання. Розроблений програмний комплекс для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку дозволяє підвищити рівень безпеки інформаційних систем, забезпечити ефективне використання ресурсів та мінімізувати ризики, пов'язані з несанкціонованим доступом та зловживанням привілеями.

Практична цінність роботи полягає у можливості впровадження розробленого програмного комплексу в реальні інформаційні системи організацій різних галузей. Це дозволить підвищити рівень захищеності даних, запобігти несанкціонованому доступу, оптимізувати розподіл ресурсів та контролювати мережевий трафік. Впровадження такого рішення сприятиме зменшенню ризиків інформаційної безпеки, підвищенню ефективності роботи систем та забезпеченню відповідності вимогам регуляторних стандартів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петровська Інна, Георгій Кучук. "Розподіл обчислювальних ресурсів у хмарних системах." Системи управління, навігації та зв'язку. Збірник наукових праць 2.68 (2022): 75–78.

2. Харченко, Ю. А. "Аналіз сучасних систем управління ресурсами підприємства." Коммунальное хозяйство городов 83 (2008): 103–110.

3. Чуб, О. І., М. В. Новожилова. "Оптимальний розподіл ресурсів при реалізації проектів реконструкції інженерних мереж в мультипроектному середовищі." *Вісник Національного технічного університету ХПІ. Сер.: Технічний прогрес та ефективність виробництва* 21 (2013): 58–64.

4. Бульба С., Давидов В., Кучук Г. "Метод розподілу ресурсів між композитними застосунками." Системи управління, навігації та зв'язку. Збірник наукових праць 4.50 (2018): 99–104.

5. Прус, Р. "Оптимізація розподілу ресурсів захисту інформації в динамічному режимі." Безпека інформації 1 (2012): 26–32.

6. Крилов, С. О. "Розробка системи аналізу мережевого трафіку." (2019).

7. Глущевський, В. В. "Моделювання потокових процесів розподілу ресурсів і продукції на мережі бізнес–процесів підприємства." Вісник Київського національного університету технологій та дизайну. Серія: Економічні науки 2 (2015): 139–148.

8. Квашук, В. П., Ю. П. Рак, В. В. Бондаренко. "Механізми управління розподілом ресурсів у проектах розвитку складних соціально–економічних систем." Управління розвитком складних систем 15 (2013): 25–29.

9. Циганок, В. В. "Проблема розподілу ресурсів як розширення можливостей систем підтримки прийняття рішень." Реєстрація, зберігання і обробка даних (2010).

10. Теленик, Сергій Федорович, et al. "Метод розподілу ресурсів між проектами." (2008).

11. Олійник, Олег Вікторович. "Організаційно–правові засади захисту інформаційних ресурсів України." К.: Інститут законодавства Верховної Ради України (2006).

12. Галата, Лілія Павлівна, Богдан Ярославович Корнієнко, Владислав Валерійович Заболотний. "Математична модель протидії загрозам у системі захисту критичних інформаційних ресурсів." Наукоємні технології 3 (2019): 300–306.

13. Корнієнко, Б. Я., Л. П. Галата. "Побудова та тестування імітаційного полігону захисту критичних інформаційних ресурсів." Наукоємні технології 4 (2017): 316–322.

14. Корнієнко, Б. Я., Л. П. Галата. "Дослідження імітаційного полігону захисту критичних інформаційних ресурсів методом IRISK." Моделювання та інформаційні технології 83 (2018): 34–42.

15. Франчук, В. М. "Захист інформаційних ресурсів." Захист інформаційних ресурсів/Василь Франчук.–К.: Редакції газет природничо–математичного циклу (2012).

16. Левченко, Євген Григорович. "Оптимізація розподілу ресурсів між об'єктами захисту інформації." Ukrainian Information Security Research Journal 9.1 (32) (2007): 33–39.

17. Саваневич, В. Е., et al. "Комплексна модель маршрутизації та обмеження трафіку в телекомунікаційних мережах військового призначення." (2010).

18. De Souza, Allan M., et al. "Traffic management systems: A classification, review, challenges, future perspectives." International Journal of Distributed Sensor Networks 13.4 (2017): 1550147716683612.

19. Лемешко, О. В., et al. "Математична модель швидкої перемаршрутизації з балансуванням навантаження та диференційованого обмеження трафіка в мережах SD–WAN." (2020).

20. Yang, Q. I𝍐, Haris N. Koutsopoulos. "A microscopic traffic simulator for evaluation of dynamic traffic management systems." Transportation Research Part C: Emerging Technologies 4.3 (1996): 113–129.

21. Єременко, Олександра Сергіївна, Марина Олександрівна Євдокименко, Анастасія Сергіївна Шаповалова. "Підвищення відмовостійкості мереж засобами швидкої перемаршрутизації з балансуванням навантаження та профілюванням трафіка." (2019).

22. Lepak, David P., et al. "A conceptual review of human resource management systems in strategic human resource management research." Research in personnel human resources management (2006): 217–271.

23. Ігнатов, В. О., М. М. Гузій, О. А. Ладигіна. "Статистична оптимізація обслуговування трафіка в гетерогенних інфокомунікаційних мережах." Problems of Informatization Management 1.49: 37–40.

24. Коломійцев, Олексій, et al. "Спосіб оптимізації розміщення фрагментів розподіленої бази даних у вузлах мережі хмарної структури за критерієм мінімуму ціни трафіку на основі рангового підходу до рішення задачі цілочисельного лінійного програмування з булевими змінними." Scientific Collection «InterConf» 173 (2023): 165–172.

25. Олійник Ю. О., Мельниченкр П. І. "Огляд методів захисту Web–застосунків від вразливостей типу CSRF (міжсайтова підробка запитів)." Problems of Informatization and Management 3.71: 28–34.

26. Мінгальова, Ю. І. "Новітні криптографічні методи захисту інформації." Матеріали II Всеукраїнської науково–практичної інтернет–конференції, присвяченої 95–річчю Херсонського державного університету (2012): 373–378.

27. Georgiadis, Leonidas, et al. "Efficient network QoS provisioning based on per node traffic shaping." IEEE/ACM transactions on networking 4.4 (1996): 482–501.

28. Saeed, Ahmed, et al. "Carousel: Scalable traffic shaping at end hosts." Proceedings of the Conference of the ACM Special Interest Group on Data Communication. 2017.

29. Chiasserini, C–F., and Ramesh R. Rao. "Improving battery performance by using traffic shaping techniques." IEEE Journal on Selected Areas in Communications 19.7 (2001): 1385–1394.

30. Nair, Suresh K., and David C. Novak. "A traffic shaping model for optimizing network operations." European Journal of Operational Research 180.3 (2007): 1358–1380.

31. Boon, Corine, Deanne N. Den Hartog, and David P. Lepak. "A systematic review of human resource management systems and their measurement." Journal of management 45.6 (2019): 2498–2537.

32. Fidler, Markus, Volker Sander, and Wojciech Klimala. "Traffic shaping in aggregate–based networks: implementation and analysis." Computer Communications 28.3 (2005): 274–286.

33. Georgiadis, Leonidas, et al. "The effect of traffic shaping in efficiently providing end–to–end performance guarantees." Telecommunication Systems 5 (1996): 71–83.

34. Marin, Gerald A. "Network security basics." IEEE security & privacy 3.6 (2005): 68–72.

35. Kaeo, Merike. Designing network security. Cisco Press, 2004.

36. Shiravi, Hadi, Ali Shiravi, and Ali A. Ghorbani. "A survey of visualization systems for network security." IEEE Transactions on visualization and computer graphics 18.8 (2011): 1313–1329.

37. Kizza, Joseph Migga, Wheeler Kizza, and Wheeler. Guide to computer network security. Vol. 8. Berlin: Springer, 2013.

38. Douligeris, Christos, and Dimitrios N. Serpanos. "Network security: current status and future directions." (2007).

39. Chen, Xiangqian, et al. "Sensor network security: a survey." IEEE Communications surveys & tutorials 11.2 (2009): 52–73.

40. Gadgil, Madhav, and Fikret Berkes. "Traditional resource management systems." Resource management and Optimization 8.3–4 (1991): 127–141.

41. Prechelt, Lutz. "An empirical comparison of c, c++, java, perl, python, rexx and tcl." IEEE Computer 33.10 (2000): 23–29.

42. Prechelt, Lutz. "Are scripting languages any good? a validation of perl, python, rexx, and tcl against c, c++, and java." Advances in Computers 58 (2002).

43. Панасюк, О. І. "Аналітичне дослідження і порівняння СУБД MongoDB та Redis." Електромеханічні, інформаційні системи та нанотехнології. Київський національний університет технологій та дизайну, 2023.

Додаток А. Технічне завдання

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

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
" 22 " березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:

Розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

08–72.БДР.007.00.065.ТЗ

Керівник бакалаврської дипломної роботи
к.т.н., доц., доц. каф. МБІС
_____ Шиян А. А.
"____" _____

Вінниця – 2024 р.

1. Найменування та область застосування

Програмний комплекс для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку. Область застосування: забезпечення безпеки, контролю доступу та ефективного використання ресурсів в інформаційних системах організацій.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробити програмний комплекс для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, спрямований на підвищення рівня безпеки та ефективності використання ресурсів в інформаційних системах.

3.2 Призначення: розроблений програмний комплекс забезпечує захист інформаційних ресурсів шляхом реалізації рольового доступу, контролю та обмеження мережевого трафіку, а також моніторингу та управління використанням ресурсів, запобігаючи несанкціонованому доступу, зловживанню привілеями та неоптимальному використанню обчислювальних потужностей.

4. Джерела розробки

4.1. Bulba, S., V. Davydov, and H. Kuchuk. "Метод розподілу ресурсів між композитними застосунками." Системи управління, навігації та зв’язку. Збірник наукових праць 4.50 (2018): 99–104.

4.2. Прус, Р. "Оптимізація розподілу ресурсів захисту інформації в динамічному режимі." Безпека інформації 1 (2012): 26–32.

4.3. Крилов, С. О. "Розробка системи аналізу мережевого трафіку." (2019).

4.4.Глущевський, В. В. "Моделювання потокових процесів розподілу ресурсів і продукції на мережі бізнес–процесів підприємства." Вісник Київського національного університету технологій та дизайну. Серія: Економічні науки 2

4.5. Циганок, В. В. "Проблема розподілу ресурсів як розширення можливостей систем підтримки прийняття рішень." Реєстрація, зберігання і обробка даних (2010).

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

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

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

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

5.2.2 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Intel Core i3–5200U 2.40 GHz і подібні до них;

– оперативна пам’ять – не менше 2Gb;

– середовище функціонування – операційна система сімейство Windows;

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

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.2

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко–економічні показники

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

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

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми	24.03.2024	27.03.2024
2.	Аналіз предметної області обраної теми	28.03.2024	02.04.2024
3.	Розробка алгоритму роботи	04.04.2024	19.04.2024
4.	Написання бакалаврської роботи на основі розробленої теми	19.04.2024	04.05.2024
5.	Передзахист бакалаврської дипломної роботи	06.05.2024	24.05.2024
6.	Виправлення, уточнення, корегування бакалаврської дипломної роботи	27.05.2024	08.06.2024
7.	Захист бакалаврської дипломної роботи	17.06.2024	17.06.2024

10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

– ПЗ до бакалаврської дипломної роботи;

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

Технічне завдання до виконання прийняв _____ Господара І. М.

Додаток Б. Лістинг коду модулів розробленого ПЗ

```
import psycopg2
import yaml
import pyshark
def init_config():
    with open('config.yaml', 'r') as file:
        config = yaml.safe_load(file)
    return config
pythonCopy codedef connect_to_db(config):
    conn = psycopg2.connect(
        host=config['db_host'],
        port=config['db_port'],
        database=config['db_name'],
        user=config['db_user'],
        password=config['db_password']
    )
    return conn
config = init_config()
db_conn = connect_to_db(config)
pythonCopy codedef authenticate_user(username, password):
    cur = db_conn.cursor()
    cur.execute("SELECT * FROM users WHERE username = %s", (username,))
    user = cur.fetchone()
    cur.close()
    if user and user['password'] == password:
        return user
    return None
pythonCopy codedef get_user_roles(user_id):
    cur = db_conn.cursor()
    cur.execute("SELECT role_id FROM user_roles WHERE user_id = %s", (user_id,))
    role_ids = [row[0] for row in cur.fetchall()]
    cur.close()
    return role_ids
def get_role_permissions(role_ids):
    cur = db_conn.cursor()
    cur.execute("SELECT permission FROM role_permissions WHERE role_id IN %s", (tuple(role_ids),))
    permissions = [row[0] for row in cur.fetchall()]
    cur.close()
    return permissions
user = authenticate_user(username, password)
if user:
    user_roles = get_user_roles(user['id'])
    user_permissions = get_role_permissions(user_roles)
else:
    print("Authentication failed")
    exit(1)
pythonCopy codedef check_access(user_permissions, resource):
    if resource in user_permissions:
        return True
    return False
if check_access(user_permissions, requested_resource):
    print("Access granted")
    access_resource(requested_resource)
else:
    print("Access denied")
```

```
pythonCopy codedef monitor_network_traffic():
    capture = pyshark.LiveCapture(interface='eth0')
    for packet in capture.sniff_continuously():
        source_ip = packet['ip'].src
        destination_ip = packet['ip'].dst
        protocol = packet.transport_layer
        traffic_class = classify_traffic(source_ip, destination_ip, protocol)
        store_traffic_data(traffic_class, packet)
def classify_traffic(source_ip, destination_ip, protocol):
    if source_ip in trusted_ips and destination_ip in trusted_ips:
        return "trusted"
    elif protocol == "HTTP" or protocol == "HTTPS":
        return "web"
    else:
        return "unknown"
pythonCopy codedef store_traffic_data(traffic_class, packet):
    cur = db_conn.cursor()
    cur.execute("INSERT INTO traffic_data (class, source_ip, destination_ip, protocol, timestamp) VALUES
(%s, %s, %s, %s, %s)",
                (traffic_class, packet['ip'].src, packet['ip'].dst, packet.transport_layer, packet.sniff_time))
    db_conn.commit()
    cur.close()
pythonCopy codedef compare_with_security_policies(traffic_class, user_id):
    cur = db_conn.cursor()
    cur.execute("SELECT * FROM security_policies WHERE traffic_class = %s AND user_id = %s", (traffic_class, user_id))
    policy = cur.fetchone()
    cur.close()
    if policy:
        return policy['bandwidth_limit'], policy['connection_limit']
    return None, None
pythonCopy codedef apply_traffic_restrictions(user_id, traffic_class, bandwidth_limit, connection_limit):
    cur = db_conn.cursor()
    cur.execute("UPDATE user_traffic SET bandwidth_used = bandwidth_used + %s, connections_used =
connections_used + 1 WHERE user_id = %s",
                (packet.length, user_id))
    db_conn.commit()
    cur.close()
    if bandwidth_limit is not None and connection_limit is not None:
        cur = db_conn.cursor()
        cur.execute("SELECT * FROM user_traffic WHERE user_id = %s", (user_id,))
        user_traffic = cur.fetchone()
        cur.close()
        if user_traffic['bandwidth_used'] > bandwidth_limit or user_traffic['connections_used'] > connection_limit:
            block_traffic(user_id)
bandwidth_limit, connection_limit = compare_with_security_policies(traffic_class, user_id)
if bandwidth_limit and connection_limit:
    apply_traffic_restrictions(user_id, traffic_class, bandwidth_limit, connection_limit)
pythonCopy codedef handle_admin_command(command, params):
    if command == 'update_security_policy':
        update_security_policy(params['policy_id'], params['bandwidth_limit'], params['connection_limit'])
    elif command == 'add_user':
        add_user(params['username'], params['password'], params['role'])
    elif command == 'remove_user':
        remove_user(params['user_id'])
def update_security_policy(policy_id, bandwidth_limit, connection_limit):
    cur = db_conn.cursor()
    cur.execute("UPDATE security_policies SET bandwidth_limit = %s, connection_limit = %s WHERE id = %s",
                (bandwidth_limit, connection_limit, policy_id))
```

```
    db_conn.commit()
    cur.close()
pythonCopy codedef add_user(username, password, role):
    cur = db_conn.cursor()
    cur.execute("INSERT INTO users (username, password, role) VALUES (%s, %s, %s)", (username, password, role))
    db_conn.commit()
    cur.close()
def remove_user(user_id):
    cur = db_conn.cursor()
    cur.execute("DELETE FROM users WHERE id = %s", (user_id,))
    db_conn.commit()
    cur.close()
command, params = get_admin_command()
handle_admin_command(command, params)
apply_policy_changes()
pythonCopy codedef collect_resource_usage_stats():
    cpu_usage = psutil.cpu_percent()
    memory_usage = psutil.virtual_memory().percent
    storage_usage = psutil.disk_usage('/').percent
    store_resource_usage_stats(cpu_usage, memory_usage, storage_usage)
def store_resource_usage_stats(cpu_usage, memory_usage, storage_usage):
    cur = db_conn.cursor()
    cur.execute("INSERT INTO resource_usage (cpu_usage, memory_usage, storage_usage, timestamp) VALUES
(%s, %s, %s, %s)",
                (cpu_usage, memory_usage, storage_usage, datetime.now()))
    db_conn.commit()
    cur.close()
pythonCopy codedef aggregate_and_analyze_data():
    cur = db_conn.cursor()
    cur.execute("SELECT * FROM resource_usage ORDER BY timestamp DESC LIMIT 100")
    stats_data = cur.fetchall()
    cur.close()
    aggregated_data = aggregate_data(stats_data)
    analyzed_data = analyze_data(aggregated_data)
    generate_reports(analyzed_data)
def aggregate_data(stats_data):
    aggregated_data = {}
    for data in stats_data:
        timestamp = data['timestamp'].strftime('%Y–%m–%d %H:%M')
        if timestamp not in aggregated_data:
            aggregated_data[timestamp] = {
                'cpu_usage': [],
                'memory_usage': [],
                'storage_usage': []
            }
        aggregated_data[timestamp]['cpu_usage'].append(data['cpu_usage'])
        aggregated_data[timestamp]['memory_usage'].append(data['memory_usage'])
        aggregated_data[timestamp]['storage_usage'].append(data['storage_usage'])
    return aggregated_data
def analyze_data(aggregated_data):
    analyzed_data = {}
    for timestamp, data in aggregated_data.items():
        analyzed_data[timestamp] = {
            'avg_cpu_usage': sum(data['cpu_usage']) / len(data['cpu_usage']),
            'avg_memory_usage': sum(data['memory_usage']) / len(data['memory_usage']),
            'avg_storage_usage': sum(data['storage_usage']) / len(data['storage_usage'])
        }
    return analyzed_data
```

```
        def generate_reports(analyzed_data):
            report = "Resource Usage Report\n\n"
            for timestamp, data in analyzed_data.items():
                report += f"Timestamp: {timestamp}\n"
                report += f"Average CPU Usage: {data['avg_cpu_usage']}%\n"
                report += f"Average Memory Usage: {data['avg_memory_usage']}%\n"
                report += f"Average Storage Usage: {data['avg_storage_usage']}%\n\n"
            save_report(report)
        collect_resource_usage_stats()
        aggregate_and_analyze_data()
        pythonCopy codedef dynamic_traffic_management(analyzed_data):
            traffic_patterns = identify_traffic_patterns(analyzed_data)
            anomalies = detect_anomalies(analyzed_data)
            apply_traffic_management_rules(traffic_patterns, anomalies)
        def identify_traffic_patterns(analyzed_data):
            traffic_patterns = {}
            for timestamp, data in analyzed_data.items():
                if data['avg_cpu_usage'] > 80 or data['avg_memory_usage'] > 80:
                    traffic_patterns[timestamp] = 'high_usage'
                else:
                    traffic_patterns[timestamp] = 'normal_usage'
            return traffic_patterns
        def detect_anomalies(analyzed_data):
            anomalies = []
            for timestamp, data in analyzed_data.items():
                if data['avg_cpu_usage'] > 95 or data['avg_memory_usage'] > 95:
                    anomalies.append(timestamp)
            return anomalies
        def apply_traffic_management_rules(traffic_patterns, anomalies):
            for timestamp, pattern in traffic_patterns.items():
                if pattern == 'high_usage':
                    limit_bandwidth(timestamp)
                else:
                    restore_bandwidth(timestamp)
            for anomaly_timestamp in anomalies:
                block_traffic(anomaly_timestamp)
        dynamic_traffic_management(analyzed_data)
        pythonCopy codedef view_report_files():
            report_files = get_report_files()
            display_report_files(report_files)
        def update_user_access_rights(user_id, new_permissions):
            cur = db_conn.cursor()
            cur.execute("UPDATE user_roles SET permissions = %s WHERE user_id = %s", (new_permissions, user_id))
            db_conn.commit()
            cur.close()
        view_report_files()
        user_id = get_user_id()
        new_permissions = get_new_permissions()
update_user_access_rights(user_id, new_permission.
Файл base.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My App</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
```

```
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
  <style>
    body {
        background-image: url('https://www.dhs.gov/sites/default/files/styles/webp_original_size/public/images/st/Our-
Work/20_0112_st_cybersecurity1.jpg.webp?itok=nhx5lfq4');
      background-size: cover;
    }
  </style>
</head>
<body class="{% block body_class %}{% endblock %}">
  <div class="container">
    <div class="row justify-content-center align-items-center h-100">
      <div class="col-md-6">
        <div class="card">
          <div class="card-body">
            <nav>
              {% if request.path == url_for('index') %}
                <a href="{{ url_for('register') }}" class="btn btn-primary btn-block mb-3">Register</a>
                <a href="{{ url_for('login') }}" class="btn btn-primary btn-block">Login</a>
              {% endif %}
            </nav>
            <div class="content">
              {% with messages = get_flashed_messages(with_categories=true) %}
                {% if messages %}
                  {% for category, message in messages %}
                    <div class="alert alert-{{ category }} alert-dismissible fade show">
                      {{ message }}
                      <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                      </button>
                    </div>
                  {% endfor %}
                {% endif %}
              {% endwith %}
              {% block content %}{% endblock %}
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>

Файл File_manager.html
{% extends 'base.html' %}

{% block body_class %}
file-manager-page
{% endblock %}

{% block content %}
<div class="container py-4">
  <div class="row justify-content-center">
    <div class="col-md-12">
```

```
        <div class="card">
          <div class="card-body">
            <h1 class="card-title text-center mb-8">File Manager</h1>
            <table class="table">
              <thead>
                <tr>
                  <th scope="col">Icon</th>
                  <th scope="col">File Name</th>
                  <th scope="col">Size</th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <td><i class="fas fa-file-alt"></i></td>
                  <td>File1</td>
                  <td>1.2 GB</td>
                </tr>
                <tr>
                  <td><i class="fas fa-file-alt"></i></td>
                  <td>File2</td>
                  <td>1.0 GB</td>
                </tr>
                <tr>
                  <td><i class="fas fa-file-alt"></i></td>
                  <td>File3</td>
                  <td>1.3 GB</td>
                </tr>
              </tbody>
            </table>
          </div>
          <div class="card-footer text-center">
            <a href="{{ url_for('profile') }}" class="btn btn-primary">Назад до профілю</a>
          </div>
        </div>
      </div>
    </div>
</div>
{% endblock %}


Файл Admin_dashboard.html
          {% extends 'base.html' %}

{% block body_class %}admin-dashboard-page{% endblock %}

{% block content %}
<div class="container py-4">
  <h2>Адмін панель</h2>
  <table class="table">
    <thead>
      <tr>
        <th scope="col">Логін</th>
        <th scope="col">Використано трафіку</th>
        <th scope="col">Тарифний план</th>
        <th scope="col">Дії</th>
      </tr>
    </thead>
    <tbody>
```

```
        {% for user in users_stats %}
        <tr>
          <td>{{ user.username }}</td>
          <td>{{ user.used_traffic }}</td>
          <td>{{ user.tariff_plan }}</td>
          <td><button class="btn btn-danger btn-sm">Заблокувати</button></td>
        </tr>
        {% endfor %}

      </tbody>
    </table>
    <div class="card-footer text-center">
      <a href="{{ url_for('profile') }}" class="btn btn-primary">Назад до профілю</a>
    </div>
</div>
{% endblock %}
```

Файл Plans.html

```
{% extends 'base.html' %}

{% block body_class %}
plans-page
{% endblock %}

{% block content %}
  <div class="container-fluid py-5">
    <div class="row">
      <div class="col-md-4">
        <div class="card mb-4">
          <div class="card-body">
            <h5 class="card-title text-center">Базовий план</h5>
            <p class="text-center">Ціна: $0/міс</p>
            <p class="text-center">Доступ до скачування 5гб файлів.</p>
            <a href="#" class="btn btn-primary btn-block" disabled>Придбати</a>
          </div>
        </div>
      </div>
      <div class="col-md-4">
        <div class="card mb-4">
          <div class="card-body">
            <h5 class="card-title text-center">Професійний план</h5>
            <p class="text-center">Ціна: $10/міс</p>
            <p class="text-center">Доступ до скачування 50гб файлів.</p>
            <a href="#" class="btn btn-primary btn-block" disabled>Придбати</a>
          </div>
        </div>
      </div>
      <div class="col-md-4">
        <div class="card mb-4">
          <div class="card-body">
            <h5 class="card-title text-center">Преміум план</h5>
            <p class="text-center">Ціна: $25/міс</p>
            <p class="text-center">Доступ до скачування 200гб файлів.</p>
            <a href="#" class="btn btn-primary btn-block" disabled>Придбати</a>
          </div>
        </div>
      </div>
```

```
      </div>
      <div class="row justify-content-center">
        <div class="col-md-6">
          <div class="text-center">
            <a href="{{ url_for('profile') }}" class="btn btn-primary">Назад до профілю</a>
          </div>
        </div>
      </div>
    </div>
{% endblock %}
```

Файл Profile.html

```
{% extends 'base.html' %}

{% block body_class %}
profile-page
{% endblock %}

{% block content %}
<div class="container py-4">
  <div class="row justify-content-center">
    <div class="col-md-10">
      <div class="card">
        <div class="card-body">
          <h1 class="card-title text-center mb-8">Профіль</h1>
          <div class="profile-info">
            <p><strong>Логін:</strong> {{ user.username }}</p>
            <p><strong>Роль:</strong> {{ user.role }}</p>
            <p><strong>Тарифний план:</strong> {{ user.tariff_plan }}</p>
          </div>
          <form action="{{ url_for('change_role') }}" method="post">
            <div class="form-group">
              <label for="role">Змінити роль</label>
              <select class="form-control" id="role" name="role">
                <option value="user">Користувач</option>
                <option value="admin">Адмін</option>
              </select>
            </div>
            <input type="hidden" name="user_id" value="{{ user.id }}">
            <button type="submit" class="btn btn-primary btn-block">Змінити роль</button>
          </form>
        </div>
        <div class="card-footer d-flex justify-content-between">
          <a href="{{ url_for('choose_plan') }}" class="btn btn-primary">Змінити план</a>
          <a href="{{ url_for('logout') }}" class="btn btn-danger">Вихід</a>
        </div>
      </div>
    </div>
  </div>
</div>

<div class="text-center mt-4">
  {% if user.role == 'admin' %}
    <a href="{{ url_for('admin_dashboard') }}" class="btn btn-secondary">Адмін панель</a>
  {% else %}
  <div class="progress-container">
    <div class="progress-label">Використано трафіку 3.5/5 гб.</div>
```

```
      <div class="progress">
        <div class="progress-bar" style="width: 70%;">70%</div>
      </div>
    </div>
      <a href="{{ url_for('file_manager') }}" class="btn btn-secondary">Файловий провідник</a>
    {% endif %}
</div>

{% endblock %}
```

Файл Register.html

```
{% extends 'base.html' %}

{% block body_class %}
register-page
{% endblock %}

{% block content %}

<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-12">
      <div class="card">
        <div class="card-body">
          <h1 class="card-title text-center mb-">Register</h1>
          <form action="{{ url_for('register') }}" method="POST">
            <div class="form-group">
              <input type="text" class="form-control" name="username" placeholder="Username" required>
            </div>
            <div class="form-group">
              <input type="password" class="form-control" name="password" placeholder="Password" required>
            </div>
            <div class="text-center">
              <button type="submit" class="btn btn-primary">Register</button>
              <a href="{{ url_for('index') }}" class="btn btn-secondary">Back to Home</a>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

Файл Login.html

```
{% extends 'base.html' %}

{% block body_class %}
login-page
{% endblock %}

{% block content %}
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-12">
      <div class="card">
        <div class="card-body">
```

```
            <h1 class="card-title text-center mb-">Логін</h1>
            <form action="{{ url_for('login') }}" method="POST">
              <div class="form-group">
                <input type="text" class="form-control" name="username" placeholder="Username" required>
              </div>
              <div class="form-group">
                <input type="password" class="form-control" name="password" placeholder="Password" required>
              </div>
              <div class="text-center">
                <button type="submit" class="btn btn-primary">Login</button>
                <a href="{{ url_for('index') }}" class="btn btn-secondary">Back to Home</a>
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
</div>
{% endblock %}
```

Файл Style.css

```
body {
  font-family: Arial, sans-serif;
}

.container {
  max-width: 1300px;
  margin: 50px auto;
}

.card {
  border: none;
  border-radius: 10px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  background-color: #ffffff;
}

.card-body {
  padding: 20px;
}

.card-title {
  font-size: 24px;
  font-weight: bold;
}

.table tbody + tbody {
  border-top: 2px solid #dee2e6;
}

.table th,
.table td {
  padding: 0.75rem;
  vertical-align: middle;
  border-top: 1px solid #dee2e6;
  text-align: center; /* Вирівнювання тексту по центру */
}
```

```
.profile-info p {
  font-size: 18px;
  margin-bottom: 10px;
}

.form-group {
  margin-bottom: 20px;
}

.btn {
  padding: 10px 20px;
  font-size: 16px;
}

.btn-primary {
  background-color: #007bff;
  border: none;
}

.btn-primary:hover {
  background-color: #0056b3;
}

.btn-danger {
  background-color: #dc3545;
  border: none;
}

.btn-danger:hover {
  background-color: #c82333;
}

.card-footer {
  display: flex;
  justify-content: space-between;
  padding: 15px;
  background-color: #f8f9fa;
  border-top: 1px solid #dee2e6;
}

.card-footer .btn {
  flex: 1;
  margin: 0 5px;
}

.progress-container {
  margin-bottom: 20px;
  display: flex;
  flex-direction: column;
  align-items: center;
}

.progress-label {
  margin-bottom: 5px;
}

.progress {
```

```
    width: 100%;
    height: 20px;
    background-color: #f0f0f0;
    border-radius: 5px;
    overflow: hidden;
}

.progress-bar {
    height: 100%;
    line-height: 20px;
    color: #fff;
    background-color: #007bff;
    border-radius: 5px;
    transition: width 0.5s ease;
}

.text-center {
    margin-bottom: 20px;
}

/* Background styles */
body.profile-page {
        background-image:    url('https://www.dhs.gov/sites/default/files/styles/webp_original_size/public/images/st/Our-
Work/20_0112_st_cybersecurity1.jpg.webp?itok=nhx5lfq4');
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center;
}

.fa-file-alt {
    margin-right: 10px;
    color: #007bff;
}
```

Додаток В. Ілюстративний матеріал

Розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку

Виконав ст. гр. 2КІТС – 20б Господар І.М.
Керівник к.ф – м. н., каф. МБІС Шиян А.А

Актуальність: У сучасному цифровому світі, де обсяги даних та кіберзагрози стрімко зростають, питання інформаційної безпеки та ефективного управління ресурсами набувають критичної важливості.

Мета роботи: розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку для підвищення рівня безпеки та ефективності використання ресурсів в інформаційних системах організацій.

Завдання:

- дослідити існуючі підходи та рішення щодо рольового управління доступом та контролю мережевого трафіку, виявити їх переваги та недоліки;
- розробити алгоритм до управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, спроектувати архітектуру програмного комплексу та описати взаємодію з базами даних;
- реалізувати програмний комплекс для захисту та управління розподілом ресурсів відповідно до розробленого підходу та архітектури;

Active Directory

- **Переваги:**
- зріла та перевірена технологія;
- широка функціональність;
- високий рівень безпеки;
- централізоване управління;
- інтеграція з іншими продуктами Microsoft.
- **Недоліки:**
- висока вартість впровадження та підтримки;
- складність налаштування для великих організацій;
- вимагає спеціалізованих знань.

OpenLDAP

- **Переваги:**
- безкоштовно та з відкритим кодом;
- широкі можливості налаштування;
- висока продуктивність;
- підтримка різних платформ;
- активна спільнота розробників.
- **Недоліки:**
- вимагає технічних знань для впровадження та підтримки;
- менш зручний інтерфейс ніж у комерційних аналогів;
- обмежена підтримка.

Jumpcloud

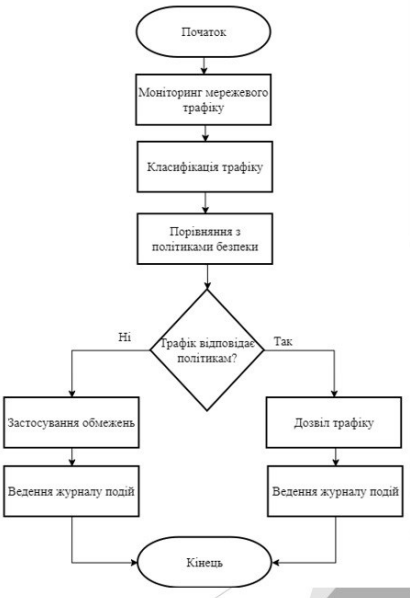
- **Переваги:**

- масштабованість;
- кросплатформеність;
- централізоване управління;
- інтеграція з Active Directory.

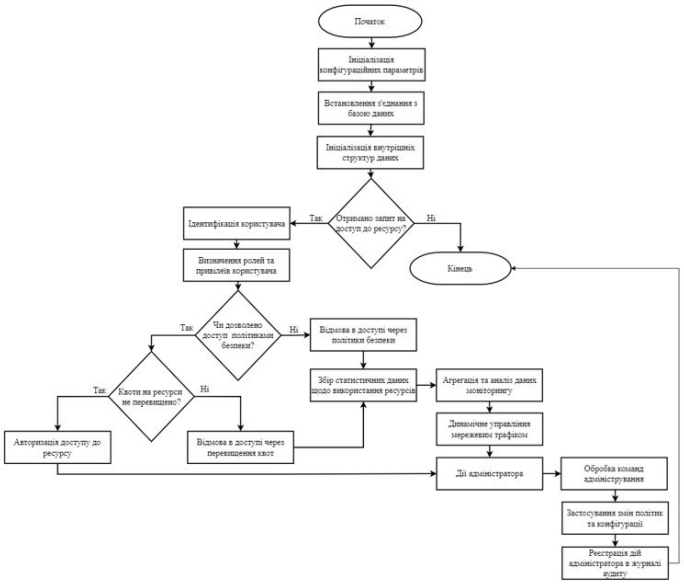
- **Недоліки:**

- залежність від інтернет–з'єднання;
- вартість може бути високою для великих організацій;
- обмежений контроль над інфраструктурою.

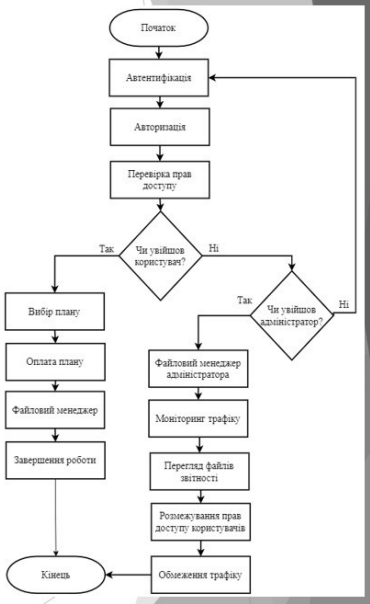
Блок схема управління доступом на основі ролей



Блок схема обмеження мережевого трафіку

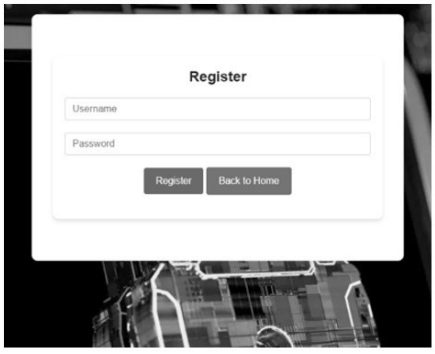


Блок схема алгоритму функціонування архітектури розроблюваного

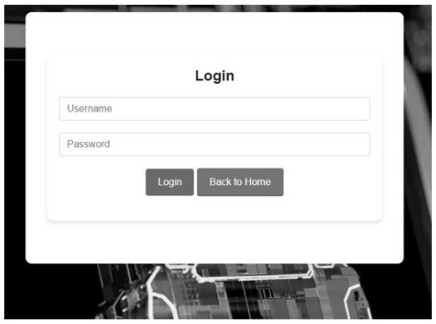


Блок схема взаємодії користувача з розробленою програмою

Екран входу користувача до системи

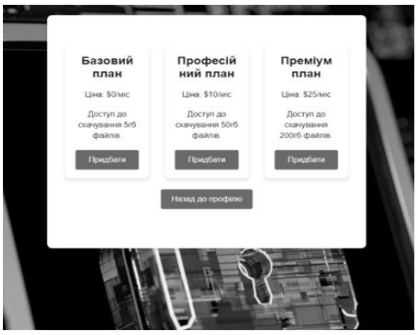


Форма реєстрації нового користувача

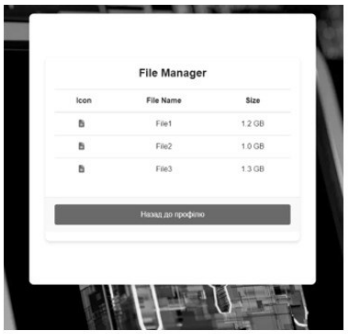


Форма авторизації користувача

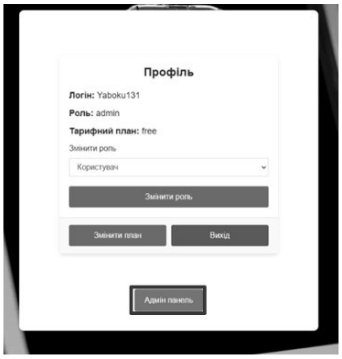
Сторінка профілю користувача розробленої програми



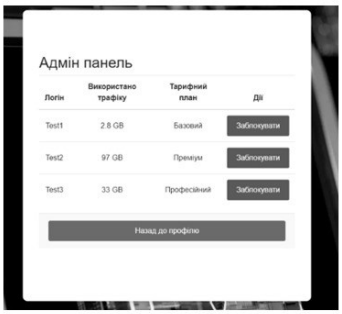
Форма вибору тарифного плану користувача



Сторінка файлового провіднику



Навігація користувача до адмін панелі



Форма адмін панелі програми захисту

РЕЗУЛЬТАТИ РОБОТИ

У даній бакалаврській дипломній роботі було розглянуто актуальну проблему захисту та управління розподілом ресурсів в інформаційних системах організацій. У ході дослідження проаналізовано сучасний стан проблеми, розроблено програмний комплекс для захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку, а також проведено його тестування та оцінку ефективності.

Дякую за увагу

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ
ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програми захисту та управління розподілом ресурсів з рольовим доступом і урахуванням обмеження трафіку
Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 98 % Схожість 2 %

Аналіз звіту подібності (відмітити потрібне):
1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Господар І.М.
(підпис) (прізвище, ініціали)

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