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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу»

Виконав: ст. 2-го курсу, групи КІТС-23м
спеціальності 125– Кібербезпека та захист інформації
Освітня програма – Кібербезпека інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Аллахвердієв О.Е. огли
(прізвище та ініціали)

Керівник: д.т.н., доц. каф. МБІС
Грицак А.В.
(прізвище та ініціали)
«____» _____ 2024 р.

Опонент: к.т.н., доцент, каф. ОТ
Богомолов С. В.
(прізвище та ініціали)
«____» _____ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
"09" грудня 2024 р.

Вінниця ВНТУ – 2024 рік

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

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформаці
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____ **Юрій ЯРЕМЧУК**
“ 20 ” вересня 2024 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Аллахвердієв Олександр Еміль огли
(прізвище, ім`я, по-батькові)

1. Тема роботи:

«Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу»
Керівник роботи_: д.т.н., доц. каф. МБІС МБІС Грицак А.В.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “17” вересня 2024 року № 310
2. Строк подання студентом роботи_за тиждень до захисту.__
3. Вихідні дані до роботи:
Стандарти, електронні джерела, підручники та наукові статті по темі, які стосуються теми магістерської кваліфікаційної роботи.
4. Зміст текстової частини:
Робота складається з трьох розділів. У першому розділі розглядаються теоретичні засади створення системи управління доступом на основі принципів ризик-орієнтованих та атрибутивних політик.. У другому розділі представлено розробку захищеної системи, зокрема особливості створення та впровадження принципів адаптивного управління доступом, розробку алгоритмів контролю доступу та динамічного моніторингу аномалій. У третьому розділі реалізовано програмну частину системи, що включає обґрунтування вибору мови програмування, розробку модулів керування базою даних, контролю доступу, моніторингу аномалій та графічного інтерфейсу.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
У дугому розділі магістерської кваліфікаційної роботи наведено 3 рисунків, у третьому розділі – 10 рисунків

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Грицак А.В. д.т.н., доц. каф. МБІС		
II	Грицак А.В. д.т.н., доц. каф. МБІС		
III	Грицак А.В.. д.т.н., доц. каф. МБІС		
Економічна частина			
IV	Бурєннікова Н. В. д.е.н., професор каф. ЕПВМ		

7. Дата видачі завдання 20 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1.	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024	
2.	Аналіз предметної області обраної теми	01.10.2024	15.10.2024	
3.	Розробка роботи	16.10.2024	26.10.2024	
4.	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024	
5.	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024	
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024	
7.	Захист магістерської кваліфікаційної роботи	11.12.2024	17.12.2024	

Студент _____ Грицак А.В.
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

УДК 004.56.5

Аллахвердієв О.Е. огли Магістерська кваліфікаційна робота зі спеціальності 125 – «Кібербезпека та захист інформації», освітня програма «Кібербезпека інформаційних технологій та систем». Вінниця: ВНТУ, 2024. – 126 с.

Укр. мовою. Бібліогр.: 41 назв; рис.: 10; табл. 9.

Ключові слова: кібезбезпека, управління доступом

Магістерська робота присвячена дослідженню, розробці та впровадженню захищеної системи управління доступом, яка базується на вдосконалених адаптивних алгоритмах контролю доступу та динамічного моніторингу аномалій. Робота охоплює теоретичні аспекти управління доступом, включаючи ризик-орієнтовані та атрибутивні політики, аналіз сучасних загроз і вимог до безпеки інформаційних систем.

У роботі запропоновано алгоритми адаптивного управління доступом, які інтегрують оцінку ризиків, аналіз контексту доступу та моніторинг поведінки користувачів. На основі цих алгоритмів реалізовано систему, яка дозволяє динамічно змінювати рівні доступу залежно від поточних умов і виявляти потенційні загрози в режимі реального часу.

Результати роботи включають розробку модулів керування базою даних, контролю доступу, моніторингу аномалій і графічного інтерфейсу для взаємодії з користувачами та адміністраторами. Система була протестована на відповідність функціональним та безпековим вимогам, що підтвердило її ефективність і стабільність у реальних умовах.

Робота має практичну цінність і може бути застосована для забезпечення захисту інформаційних ресурсів у різних галузях, таких як корпоративні мережі, хмарні платформи та IoT-середовища. Впровадження запропонованих рішень сприяє підвищенню рівня інформаційної безпеки та адаптивності систем управління доступом до сучасних викликів кіберзагроз.

ABSTRACT

UDC 004.56.5

Allakhverdiyev O.E. ogly Master's qualification work in specialty 125 - “Cybersecurity and Information Protection”, educational program “Cybersecurity of Information Technologies and Systems”. Vinnytsia: VNTU, 2024. - 126 p.

In Ukrainian. Bibliography: 41 titles; Figures: 10; Table 9.

Keywords: cybersecurity, access control

The master's thesis is devoted to the research, development and implementation of a secure access control system based on advanced adaptive access control algorithms and dynamic anomaly monitoring. The work covers the theoretical aspects of access control, including risk-based and attribute-based policies, analysis of modern threats and requirements for the security of information systems.

The paper proposes adaptive access control algorithms that integrate risk assessment, access context analysis, and user behavior monitoring. Based on these algorithms, a system has been implemented that allows to dynamically change access levels depending on current conditions and detect potential threats in real time.

The results of the work include the development of modules for database management, access control, anomaly monitoring, and a graphical interface for interaction with users and administrators. The system was tested for compliance with functional and security requirements, which confirmed its effectiveness and stability in real-world conditions.

The work is of practical value and can be used to ensure the protection of information resources in various industries, such as corporate networks, cloud platforms, and IoT environments. Implementation of the proposed solutions contributes to increasing the level of information security and adaptability of access control systems to modern cyber threats.

ЗМІСТ

ВСТУП

Актуальність. Сучасні інформаційні системи стають дедалі складнішими та важливішими для забезпечення роботи організацій, що призводить до зростання кількості кібератак, спрямованих на несанкціонований доступ до даних. Традиційні методи управління доступом, які базуються на статичних правилах, часто виявляються недостатньо ефективними в умовах динамічних загроз. Зокрема, фіксовані політики доступу не враховують змінні фактори, такі як місцезнаходження користувача, поведінкові патерни або рівень ризику доступу до ресурсу. Це обґрунтовує необхідність розробки адаптивних систем управління доступом, які дозволяють динамічно оцінювати контекст запиту та виявляти аномалії в реальному часі. Розробка таких систем є актуальною проблемою, що має значний вплив на забезпечення інформаційної безпеки.

Мета і задачі дослідження. Метою роботи є розробка захищеної системи управління доступом, яка базується на вдосконалених адаптивних алгоритмах контролю доступу та динамічного моніторингу аномалій. Ця система повинна забезпечувати високий рівень безпеки та гнучкості в управлінні доступом, адаптуючись до змінних умов і ризиків.

Задачами дослідження є:

- провести аналіз сучасних загроз і вимог до систем управління доступом.
- дослідити та обґрунтувати використання ризик-орієнтованих та атрибутивних політик доступу.
- розробити концептуальну модель системи управління доступом із включенням алгоритмів оцінки ризику та моніторингу аномалій.
- реалізувати програмну частину системи, включаючи модулі контролю доступу, моніторингу аномалій, керування базою даних і графічний інтерфейс.
- провести тестування розробленої системи для перевірки її функціональності, стабільності та відповідності вимогам безпеки.

Об’єкт дослідження. Об'єктом дослідження є процеси управління доступом до інформаційних ресурсів у сучасних системах із підвищеними вимогами до безпеки.

Предмет дослідження. Предметом дослідження є методи та алгоритми адаптивного управління доступом, засновані на оцінці ризиків, атрибутивних політиках доступу та динамічному моніторингу аномалій.

Новизна роботи. Новизна роботи полягає у розробці вдосконалених алгоритмів управління доступом, які поєднують ризик-орієнтовані підходи з атрибутивними політиками. Запропоновані алгоритми дозволяють динамічно адаптувати рівень доступу залежно від контексту запиту та виявляти потенційні загрози у режимі реального часу. Крім того, розроблена система інтегрує механізми оцінки ризиків та аналізу поведінкових аномалій, що забезпечує її ефективність і надійність.

Практична цінність. Практична цінність роботи полягає в можливості впровадження розробленої системи в організаціях для підвищення рівня захищеності інформаційних ресурсів. Використання адаптивного управління доступом дозволяє знизити ризик несанкціонованого доступу, забезпечити гнучке управління правами користувачів та своєчасно реагувати на потенційні загрози. Розроблена система може бути застосована в різних середовищах, включаючи корпоративні мережі, хмарні платформи та IoT-системи, що робить її універсальним рішенням для забезпечення інформаційної безпеки.

1 ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ ДОСТУПОМ НА ОСНОВІ ПРИНЦИПІВ РИЗИК-ОРІЄНТОВАНИХ ТА АТРИБУТИВНИХ ПОЛІТИК ДОСТУПУ

У даному розділі розглядаються теоретичні основи створення системи управління доступом, яка базується на принципах ризик-орієнтованих та атрибутивних політик. Описано важливість захисту інформаційних систем і актуальність впровадження адаптивного управління доступом. Здійснено аналіз основних принципів ризик-орієнтованих і атрибутивних політик доступу, що формують основу для гнучкого управління привілеями користувачів. Представлено огляд сучасних загроз і вимог до систем управління доступом, які обґрунтовують необхідність динамічного підходу до безпеки. Також проведено аналіз методів динамічного контролю доступу з урахуванням рівня ризику, що дозволяє визначити ключові підходи до розробки системи. Завершується розділ висновками та постановкою задачі, які визначають напрямки подальшої роботи.

1.1 Важливість захисту інформаційних систем та актуальність адаптивного управління доступом

Захист інформаційних систем є ключовою складовою в управлінні сучасними організаціями. Він спрямований на забезпечення безпеки даних, що обробляються, зберігаються та передаються в рамках внутрішніх і зовнішніх систем [1]. Через зростання кількості кіберзагроз, таких як фішинг-атаки, зловмисні програми, витоки даних і цілеспрямовані атаки на інформаційну інфраструктуру, потреба в надійних і ефективних засобах захисту інформаційних систем зростає щоденно. Адаптивне управління доступом вважається одним із перспективних напрямків, що дозволяє підвищити рівень захищеності інформаційних ресурсів завдяки гнучкому і динамічному підходу до контролю доступу [2].

Важливість захисту інформаційних систем

– збереження конфіденційності та цілісності інформації: Інформаційні системи зберігають велику кількість конфіденційних даних, таких як персональна

інформація користувачів, фінансова інформація, інтелектуальна власність та інші корпоративні дані. Збереження конфіденційності та цілісності цих даних є критично важливим для забезпечення довіри з боку користувачів і партнерів. Захист інформації запобігає несанкціонованому доступу та маніпуляціям з боку зловмисників, що можуть призвести до великих збитків;

– підтримка безперервності бізнес-процесів: Витік або втрата важливої інформації може мати катастрофічні наслідки для бізнесу, такі як зупинка виробничих процесів, репутаційні збитки та фінансові втрати. Безперервність діяльності організації значною мірою залежить від її здатності захистити свої інформаційні активи та швидко реагувати на інциденти, пов’язані з порушенням безпеки. Ефективні заходи захисту інформаційних систем сприяють зменшенню ризиків, що пов'язані з інцидентами безпеки, і підвищують стійкість організації до зовнішніх і внутрішніх загроз;

– відповідність законодавчим вимогам: Багато країн мають закони та нормативні акти, що вимагають захисту особистих даних і обмежують їхнє несанкціоноване використання. Відповідність вимогам таких нормативів, як GDPR, HIPAA, CCPA, і багатьох інших регуляцій, вимагає надійних методів захисту інформаційних систем та управління доступом. Недотримання таких вимог може призвести до значних штрафів і судових позовів, а також до втрати довіри з боку клієнтів;

– захист від зловмисних дій: Сучасні кіберзлочинці використовують різноманітні методи, від фішингу і соціальної інженерії до високотехнологічних атак, для отримання несанкціонованого доступу до інформаційних систем. Захист інформаційних систем, особливо критично важливих даних, вимагає застосування адаптивних підходів, що дозволяють враховувати постійно змінні умови і ризики.

Адаптивне управління доступом (АМД) є сучасним підходом до безпеки інформаційних систем, що динамічно адаптує правила доступу до даних, враховуючи поточні умови, контекст і ризики. Його актуальність пояснюється

необхідністю ефективного захисту у світі, де кіберзагрози швидко змінюються, а бізнес-процеси вимагають зручності та безперервності доступу до інформації.

Гнучкість управління доступом:

Адаптивне управління доступом забезпечує гнучкість у політиках безпеки, дозволяючи системі автоматично змінювати доступ користувачів відповідно до поточного контексту. Наприклад:

– геолокація: система може обмежити доступ для користувача, якщо він заходить з незвичної географічної локації;

– час доби: доступ може бути дозволений тільки у певний час, наприклад, в робочі години, що зменшує ризик зовнішніх атак вночі;

– тип пристрою: система може адаптувати доступ на основі характеристик пристрою, з якого виконується вхід, наприклад, обмежити доступ, якщо пристрій не авторизований або небезпечний.

Ця гнучкість знижує потребу у жорстких обмеженнях для всіх користувачів, натомість концентруючи зусилля на тих випадках, які вимагають підвищеного рівня захисту [2].

Контекстуальний контроль доступу:

Контекстуальний контроль доступу полягає у врахуванні додаткових параметрів для оцінки ризиків. Такий підхід дозволяє адаптивним системам виявляти незвичну поведінку, яка може вказувати на спробу несанкціонованого доступу:

– історія активності: аналізуючи звичну поведінку користувача, система може виявляти аномальні дії. Наприклад, якщо користувач зазвичай входить у систему з певної IP-адреси, а новий вхід відбувається з іншої країни, це викликає підозру;

– аналіз взаємодії: фіксуються незвичні зміни в поведінці, наприклад, спроби доступу до нових ресурсів або високий рівень активності за короткий час.

Завдяки контекстуальним параметрам, система здатна приймати рішення на основі ширшого обсягу інформації, що підвищує рівень безпеки.

Інтеграція принципів ризик-орієнтованого доступу:

Ризик-орієнтований доступ оцінює загрозу, враховуючи як поточний контекст, так і потенційні наслідки для системи. Це допомагає приймати рішення про надання доступу, аналізуючи ризик у режимі реального часу:

– динамічна оцінка ризику: для кожної сесії або запиту на доступ система оцінює рівень ризику на основі атрибутів, таких як рівень привілеїв користувача, чутливість запитуваних даних, характер поточної активності;

– диференційований рівень доступу: за високого ризику система може дозволити тільки частковий доступ або потребувати додаткової аутентифікації (наприклад, двофакторної), а при низькому ризику — дозволити доступ без додаткових заходів.

Цей підхід забезпечує баланс між безпекою і зручністю користування, дозволяючи користувачам виконувати свої завдання, мінімізуючи ризики [3].

Адаптивність до нових кіберзагроз:

Здатність швидко адаптуватися до нових кіберзагроз є ключовою перевагою адаптивного управління доступом:

– інтеграція з системами моніторингу: адаптивні системи можуть отримувати дані від систем моніторингу (SIEM, IDS) для виявлення нових загроз, включаючи шкідливе програмне забезпечення або специфічні патерни атак;

– автоматичне навчання: з використанням технологій машинного навчання система може постійно вдосконалювати свої механізми виявлення аномалій, відзначаючи нові патерни поведінки і налаштовуючи алгоритми відповідно до змін у профілях загроз.

Таким чином, адаптивне управління доступом підвищує стійкість систем до сучасних та майбутніх загроз, що забезпечує гнучкість і швидкість реакції.

Зниження людського фактора:

Адаптивні системи мінімізують потребу у ручному втручанні, автоматизуючи процес прийняття рішень щодо доступу:

– автоматична обробка запитів: система самостійно ухвалює рішення щодо доступу залежно від ризиків і контексту, що знижує ймовірність людських помилок;

– мінімізація втручання адміністраторів: адаптивне управління зменшує навантаження на команди безпеки, дозволяючи їм зосередитися на більш критичних аспектах безпеки. Це особливо важливо у великих організаціях, де обробка численних запитів на доступ може бути значним навантаженням.

Зниження людського фактора не тільки підвищує ефективність, але й знижує ймовірність ненавмисних помилок, що можуть бути використані зловмисниками.

Покращення користувацького досвіду:

Адаптивне управління доступом значно підвищує зручність користування інформаційними системами, забезпечуючи при цьому високий рівень захисту:

– мінімізація переривань: коли система визначає, що рівень ризику низький, користувачі можуть отримати доступ без додаткових перевірок, що знижує їхній стрес та покращує взаємодію із системою;

– персоналізований доступ: для користувачів, які постійно працюють у захищеній мережі або з фіксованого місця, адаптивне управління знижує кількість запитів на аутентифікацію, тим самим підвищуючи ефективність їхньої роботи.

Це дозволяє організаціям забезпечувати захист без значних втрат продуктивності, що особливо актуально для компаній з великими командами та віддаленими співробітниками [4].

Адаптивне управління доступом стає невід’ємним компонентом сучасних інформаційних систем, оскільки воно дозволяє динамічно контролювати доступ на основі актуальної ситуації та ризиків. Такий підхід є важливим елементом забезпечення безпеки, гнучкості та зручності користування, знижуючи ймовірність несанкціонованих дій, підвищуючи ефективність управління та забезпечуючи актуальність захисту у світі постійно змінюваних кіберзагроз.

1.2 Основні принципи ризик-орієнтованих та атрибутивних політик доступу

Ризик-орієнтовані та атрибутивні політики доступу є сучасними підходами до управління доступом, які дозволяють більш ефективно забезпечувати безпеку інформаційних систем, орієнтуючись на оцінку ризиків і атрибути користувача, пристроїв та умов доступу. Вони надають можливість динамічно змінювати правила доступу в залежності від контексту, що особливо важливо для складних інфраструктур з високим рівнем загроз.

Ризик-орієнтовані політики доступу (Risk-Based Access Control, RBAC):

Ризик-орієнтовані політики доступу застосовують підхід, заснований на оцінці ризиків, де рішення про надання або обмеження доступу приймаються на основі рівня ризику, пов'язаного з конкретною ситуацією. Основні принципи ризик-орієнтованого управління доступом включають:

Оцінка рівня ризику в реальному часі.

Ризик-орієнтований підхід передбачає, що система аналізує різні фактори для оцінки ризику при кожному запиті на доступ:

– локація: доступ з нової або підозрілої географічної локації може вважатися ризикованим, що спричиняє додаткові заходи безпеки;

– тип пристрою: якщо доступ запитується з невідомого або неавторизованого пристрою, рівень ризику підвищується;

– тип даних або ресурсу: доступ до критично важливих даних потребує жорсткішого контролю порівняно з загальнодоступною інформацією;

– історія користувача: аналіз попередніх дій користувача може виявити аномальну поведінку, яка підвищує рівень ризику.

Динамічне коригування рівня доступу.

На відміну від статичних систем, ризик-орієнтовані політики забезпечують динамічний контроль доступу. Залежно від рівня ризику система може:

– заборонити доступ при високому ризику;

– потребувати додаткової аутентифікації (наприклад, двофакторної) для підвищення рівня захисту;

– знизити рівень доступу (наприклад, дозволити лише читання замість повного доступу) для зменшення можливих збитків у разі порушення безпеки.

Інтеграція з системами моніторингу.

Ризик-орієнтовані політики часто інтегруються з системами моніторингу та аналізу (такими як SIEM або IDS), щоб відстежувати підозрілі активності в реальному часі. Це дозволяє:

– виявляти аномалії та потенційні загрози ще на етапі запиту доступу;

– аналізувати лог-файли та активність користувачів для коригування ризикових моделей і вдосконалення алгоритмів оцінки.

Врахування ймовірності та впливу ризику [6].

У ризик-орієнтованому підході приймається до уваги не тільки ймовірність загрози, але й потенційний вплив такої загрози на систему:

– ймовірність визначається на основі історичних даних, типових патернів поведінки та можливих векторів атак;

– вплив — це ймовірна шкода у разі успішного несанкціонованого доступу, що може варіюватися від втрати конфіденційних даних до порушення операційної діяльності.

Баланс між зручністю та безпекою.

Важливою метою ризик-орієнтованих політик є забезпечення безпеки без надмірного обтяження користувача. Наприклад:

– користувачі, які виконують дії з низьким рівнем ризику, можуть отримати доступ без додаткових обмежень;

– додаткові заходи (такі як багатофакторна аутентифікація) застосовуються лише в ситуаціях підвищеного ризику.

Атрибутивні політики доступу орієнтовані на атрибути користувача, ресурсів та середовища для визначення прав доступу. На відміну від традиційного підходу з ролями (Role-Based Access Control, RBAC), де доступ ґрунтується на ролях,

атрибутивний підхід забезпечує значно вищу гнучкість. Основні принципи атрибутивних політик доступу включають:

Атрибути користувача, об'єкта та умов доступу.

У політиках ABAC для прийняття рішень використовуються атрибути, які характеризують:

Користувача (ідентифікатор, посада, повноваження, рівень безпеки, групи, до яких він належить).

– об'єкт (тип даних або ресурсів, до яких здійснюється доступ, рівень їх чутливості);

– контекст або умови доступу (місцезнаходження, тип пристрою, час, тип мережі тощо);

Гнучкість і можливість деталізованого управління доступом.

Використання атрибутів дозволяє створювати політики, що враховують різні умови:

– доступ можна обмежити до певних атрибутів, наприклад, дозволяти доступ до певних ресурсів лише з авторизованих пристроїв або у визначений робочий час;

– політики можуть бути специфічними для окремих ресурсів, ролей або об'єктів, що дозволяє деталізувати управління доступом, адаптуючи його до потреб кожного користувача чи типу завдань.

Централізоване управління політиками.

Політики доступу на основі атрибутів визначаються централізовано, що дозволяє змінювати їх без необхідності модифікації кожного ресурсу:

– центральне управління дозволяє швидко оновлювати політики доступу, що особливо корисно у великих організаціях, де управління доступом є складним і постійно змінюється;

– централізоване управління забезпечує більшу прозорість та контроль за виконанням політик, полегшуючи аудит і моніторинг доступу.

Підтримка динамічних політик доступу [7].

Атрибутивні політики можуть динамічно змінюватися залежно від поточного контексту, що підвищує захист в умовах швидкозмінного середовища:

– автоматичне блокування або надання доступу на основі поточного контексту (наприклад, якщо користувач заходить з незвичного місця або підозрілого пристрою);

– врахування зовнішніх подій: система може змінювати політику доступу залежно від інцидентів безпеки або змін у мережевому середовищі.

Висока масштабованість.

На відміну від RBAC, який може потребувати створення великої кількості ролей у великих організаціях, атрибутивний підхід є більш масштабованим:

– необхідність в управлінні ролями мінімізується, оскільки рішення про доступ приймаються на основі конкретних атрибутів, а не фіксованих ролей;

– легкість додавання нових атрибутів або змінення умов доступу дозволяє системі адаптуватися до росту організації та зміни її структури.

Аудит та відповідність регуляторним вимогам.

Завдяки точному управлінню атрибутами і контекстом, ABAC дозволяє легше забезпечити відповідність вимогам регуляторів:

– політики доступу можна налаштувати так, щоб відповідати вимогам конфіденційності, наприклад, GDPR чи HIPAA, обмежуючи доступ до конфіденційних даних лише для певних груп користувачів у визначених умовах;

– ABAC забезпечує детальний аудит дій користувачів та точний контроль за виконанням політик, що спрощує процес звітності і перевірки відповідності.

Взаємодія ризик-орієнтованих та атрибутивних політик.

Ризик-орієнтовані та атрибутивні політики можуть взаємодіяти, створюючи гібридний підхід, де:

– рівень ризику використовується як один із атрибутів у політиках ABAC, що дозволяє деталізувати управління доступом на основі поточного рівня загроз;

– динамічна оцінка ризиків інтегрується в атрибутивний підхід, що дозволяє враховувати контекст і виявляти ризики ще до прийняття рішення про доступ.

Цей підхід дозволяє використовувати як поточні атрибути користувача і контексту, так і оцінку рівня ризику для забезпечення високого рівня безпеки і гнучкості доступу в умовах сучасних загроз.

Ризик-орієнтовані та атрибутивні політики доступу представляють собою передові підходи до управління доступом, які поєднують динамічність, гнучкість і високу ефективність у протидії сучасним загрозам. Врахування ризику в режимі реального часу та використання атрибутів як основи для прийняття рішень щодо доступу дозволяють забезпечити збалансованість між безпекою і зручністю використання системи.

Ризик-орієнтовані політики надають можливість адаптувати доступ відповідно до поточних умов, а атрибутивні політики забезпечують деталізований контроль доступу, враховуючи різноманітні параметри користувача, об'єкта та середовища. Обидва підходи мають високий рівень масштабованості і дозволяють ефективно управляти доступом у великих організаціях з високою кількістю користувачів та різними типами ресурсів [9].

Сучасна інформаційна безпека вимагає інтеграції ризик-орієнтованого та атрибутивного підходів, що дозволяє посилити захист і забезпечити швидке реагування на зміни у профілі загроз.

1.3 Огляд сучасних загроз та вимог до систем управління доступом

З розвитком інформаційних технологій та збільшенням обсягу і критичності даних, які зберігаються і обробляються в цифровому форматі, сучасні інформаційні системи постійно стикаються з новими загрозами безпеці. Системи управління доступом є однією з основних ліній захисту організаційних ресурсів і повинні відповідати актуальним вимогам безпеки, щоб ефективно протидіяти сучасним загрозам [10].

Фішинг є однією з найпоширеніших атак на інформаційні системи, коли зловмисники використовують підроблені повідомлення (наприклад, електронні листи, повідомлення в месенджерах) для того, щоб змусити користувача надати

конфіденційну інформацію, наприклад, логіни і паролі. Ці атаки можуть бути цілеспрямованими, що значно підвищує їхню ефективність.

– цільовий фішинг (спірфішинг): орієнтований на конкретних користувачів або організації для збору конкретних даних, що робить атаку більш персоналізованою і успішною;

– вимоги до систем управління доступом: Системи повинні передбачати багатофакторну аутентифікацію (MFA), щоб навіть при компрометації паролів зловмисники не змогли отримати доступ.

Шкідливе програмне забезпечення, включаючи віруси, трояни, програми-вимагачі (ransomware) та шпигунське ПЗ, здатне проникати в інформаційні системи, викрадаючи дані або блокуючи їхній доступ [11].

– програми-вимагачі: блокують дані користувача, вимагаючи викуп за їхнє розблокування. Це серйозно загрожує не лише доступу до даних, а й може порушити роботу всієї організації;

– вимоги до систем управління доступом: Необхідно забезпечити обмеження доступу для процесів та програм, щоб мінімізувати можливість поширення шкідливого ПЗ. Також важливо впроваджувати політики найменших привілеїв (Least Privilege).

Внутрішні загрози, що виникають, коли співробітники або особи, що мають легальний доступ, навмисно або ненавмисно зловживають своїми привілеями для отримання або витоку конфіденційної інформації.

– приклади: Недобросовісні співробітники, що використовують свій доступ до критично важливих систем для копіювання або видалення даних;

– вимоги до систем управління доступом: Системи повинні підтримувати детальний моніторинг та аудит активності користувачів, а також мати можливість відстежувати підозрілі дії в режимі реального часу.

Соціальна інженерія — це маніпуляції, спрямовані на отримання конфіденційної інформації від користувачів шляхом психологічного впливу.

Наприклад, зловмисники можуть видавати себе за працівників технічної підтримки, щоб отримати доступ до облікових даних користувачів.

– вимоги до систем управління доступом: Системи повинні забезпечувати методи додаткової перевірки ідентичності користувача, такі як контекстуальна аутентифікація, щоб знизити ймовірність успіху соціальних атак.

Використання незахищених пристроїв або відкритих мереж (наприклад, Wi-Fi у громадських місцях) створює додатковий ризик, коли зловмисники можуть перехоплювати інформацію під час її передачі.

– вимоги до систем управління доступом: Необхідне впровадження політик доступу на основі атрибутів, які враховують контекст доступу (пристрій, мережу), і вимагають додаткової аутентифікації при доступі з потенційно небезпечних умов.

Зловмисники можуть використовувати вразливості у системах для підвищення своїх прав доступу, що дозволяє їм отримати контроль над більш критичними даними та ресурсами.

– вимоги до систем управління доступом: Системи мають реалізовувати політику мінімальних привілеїв, а також регулярно проводити перевірки і знижувати права доступу для користувачів та облікових записів.

Сучасні системи управління доступом (СУД) повинні забезпечувати комплексний захист інформаційних ресурсів організації, враховуючи швидко змінювані загрози та високі вимоги до конфіденційності й доступності даних. Враховуючи складність сучасного кіберсередовища, такі системи мають відповідати ряду ключових вимог для забезпечення гнучкого, динамічного та багаторівневого контролю доступу.

Одним із ключових засобів захисту є багатофакторна аутентифікація (MFA), яка залучає кілька рівнів перевірки ідентичності для підвищення надійності процесу аутентифікації [11].

Типи факторів: MFA зазвичай використовує різні категорії факторів, зокрема:

– що знає користувач (наприклад, пароль чи PIN-код);

– що належить користувачу (одноразовий пароль (OTP) на мобільний пристрій або фізичний токен);

– хто є користувачем (біометричні дані, такі як відбитки пальців або розпізнавання обличчя). Адаптивність MFA: При незвичній активності чи спробах доступу з нетипового місця (наприклад, іншої країни), система автоматично додає вимогу ще одного фактора аутентифікації для підвищення безпеки. Захист від атак: використання MFA суттєво зменшує ризик компрометації облікових записів під час фішингових атак або у разі витоку паролів, оскільки пароль сам по собі не забезпечує повного доступу без додаткового підтвердження [12].

Політика найменших привілеїв обмежує доступ користувачів до мінімально необхідних ресурсів і функцій для виконання їхніх обов'язків:

– принцип обмеженого доступу: кожному користувачу надається лише той доступ, який необхідний для виконання його задач, що дозволяє знизити ризик внутрішніх загроз та випадкового витоку інформації;

– динамічний контроль доступу: доступ має надаватися та відкликатися автоматично при зміні обов'язків користувача. Це запобігає ситуаціям, коли користувачі залишаються з надмірними правами після завершення проєктів або переходу на інші посади;

– тимчасовий доступ: у разі потреби надання доступу до додаткових ресурсів він має бути тимчасовим з автоматичним завершенням після завершення роботи.

Адаптивне управління доступом на основі ризику дозволяє динамічно регулювати рівень доступу на основі оцінки ризиків:

– оцінка ризику в реальному часі: система аналізує різні фактори, такі як локація, тип пристрою, час доби та поведінка користувача, для оцінки ризику кожного запиту на доступ;

– динамічні політики: у разі підвищеного ризику система може обмежувати доступ або вимагати додаткову аутентифікацію. Наприклад, якщо користувач намагається отримати доступ з незвичного місця, система може знизити рівень доступу або вимагати додатковий фактор перевірки;

– інтеграція з системами моніторингу: оцінка ризику може інтегруватися з іншими системами, такими як SIEM або IDS, для миттєвого реагування на виявлені загрози.

ABAC дозволяє встановлювати політики доступу на основі атрибутів користувачів, пристроїв та умов доступу, що забезпечує вищу гнучкість порівняно з традиційними рольовими політиками:

– атрибути користувачів: доступ може базуватися на характеристиках користувача, наприклад, його посаді, рівні повноважень, відділі, геолокації;

– атрибути об'єкта та середовища: політика може враховувати властивості ресурсу (наприклад, конфіденційність даних) або контексту (наприклад, час доби, тип мережі);

– можливість детального контролю доступу: завдяки ABAC можна створювати детальні політики, які контролюють доступ до конкретних даних в залежності від контексту, що особливо корисно у великих організаціях з різноманітними процесами.

Контекстуальна аутентифікація — це динамічна аутентифікація, яка враховує контекст доступу (наприклад, місцезнаходження, пристрій, поведінку) і додає додаткові рівні захисту:

– аналіз контексту: система перевіряє, чи відповідає поточна сесія стандартній поведінці користувача. Наприклад, якщо користувач зазвичай працює з офісу, а вхід здійснюється з іншої країни, система вимагає додаткової перевірки;

– виявлення аномалій: якщо система виявляє аномальну поведінку, вона може вимагати додаткової аутентифікації або заблокувати доступ;

– гнучкість та підвищена безпека: контекстуальна аутентифікація забезпечує доступ у звичних умовах без зайвих перевірок, але підвищує рівень захисту, коли поведінка користувача здається підозрілою.

Системи управління доступом повинні мати потужні функції моніторингу для відстеження всіх дій користувачів та виявлення підозрілих активностей:

– журналювання активностей: система повинна зберігати повну історію дій користувачів для проведення аудиту та розслідувань у разі інциденту;

– виявлення аномалій: автоматичний аналіз журналів та виявлення підозрілої активності допомагає виявляти можливі інсайдерські загрози або зломи;

– регулярні перевірки: аудит дозволяє виявити та скоригувати політики доступу, які можуть не відповідати потребам безпеки, а також відстежувати відповідність регуляторним вимогам [13].

Системи повинні легко адаптуватися до змін у середовищі та нових загроз, включаючи можливість швидких оновлень політик доступу:

– оперативність оновлень: у разі виявлення нових загроз або вразливостей система повинна дозволяти швидке внесення змін без значних затримок;

– гнучкість налаштувань: адміністратори повинні мати змогу легко змінювати параметри доступу в залежності від зміни умов бізнесу або загроз;

– масштабованість: система повинна бути здатною обробляти великий обсяг запитів і пристосовуватися до збільшення кількості користувачів та ресурсів.

Щоб швидко реагувати на загрози, система повинна мати засоби сповіщення та автоматичні механізми реагування:

– сповіщення про підозрілу активність: при виявленні аномальної поведінки або порушень система повинна негайно сповіщати адміністратора або автоматично обмежувати доступ;

– автоматичні дії: система може автоматично блокувати доступ або активувати додаткову аутентифікацію при виявленні потенційно небезпечних дій;

– інтеграція з іншими засобами кібербезпеки: система управління доступом повинна взаємодіяти з іншими інструментами, такими як системи виявлення вторгнень (IDS), SIEM, для забезпечення комплексного захисту.

Для комплексного захисту СУД повинна інтегруватися з іншими рішеннями кібербезпеки, що дозволить забезпечити єдиний рівень захисту:

– інтеграція з SIEM-системами: збирання та аналіз подій доступу дозволяє швидко ідентифікувати підозрілу активність;

– підтримка DLP (Data Loss Prevention): інтеграція з DLP дозволяє додатково контролювати витік даних та обмежувати доступ на рівні конкретних типів інформації;

– спільне використання даних: забезпечення передачі даних між різними системами безпеки для підвищення ефективності виявлення загроз.

Системи управління доступом повинні відповідати сучасним регуляторним вимогам і стандартам у галузі інформаційної безпеки:

– дотримання законодавства: сучасні регуляції, такі як GDPR, HIPAA, PCI-DSS, накладають жорсткі вимоги на безпеку даних і їх обробку;

– автоматичний аудит та звітність: система повинна забезпечувати інструменти для автоматичного збору даних для аудиту та формування звітів про виконання політик доступу;

– шифрування даних та захист конфіденційності: захист даних є обов’язковою вимогою для дотримання багатьох стандартів, що забезпечує збереження конфіденційної інформації від зовнішніх та внутрішніх загроз [14].

Сучасні системи управління доступом повинні бути багатофункціональними, динамічними та гнучкими, щоб відповідати вимогам кібербезпеки та захищати критичні дані в умовах постійно змінюваного кіберсередовища. Вони повинні забезпечувати багатофакторну аутентифікацію, динамічне управління доступом на основі ризику та атрибутів, моніторинг і аудит дій, автоматичні реакції на загрози та відповідність регуляторним вимогам. Інтеграція цих вимог створює надійну систему захисту, яка може адаптуватися до нових викликів і забезпечувати безперервний захист ресурсів організації.

1.4 Аналіз методів динамічного контролю доступу з урахуванням рівня ризику

Динамічний контроль доступу з урахуванням рівня ризику (Risk-Based Access Control, RBAC) є підходом, що дозволяє адаптивно змінювати рівень доступу на основі оцінки ризиків у реальному часі. Це особливо важливо в умовах

сучасних кіберзагроз, коли необхідно швидко адаптуватися до різних загроз і умов. Ризик-орієнтований підхід дозволяє враховувати змінні фактори і автоматично приймати рішення щодо доступу відповідно до поточної ситуації.

Основні принципи динамічного контролю доступу з урахуванням рівня ризику:

– контекстуальність: Рівень доступу надається з урахуванням контексту, в якому відбувається запит доступу. Такий підхід аналізує різні фактори, такі як місцезнаходження користувача, тип пристрою, час доби, попередню активність та інші параметри, що можуть вказувати на ризикову поведінку;

– оцінка ризику в реальному часі: Система проводить оцінку ризику кожного запиту на доступ у режимі реального часу. На основі отриманих даних система визначає рівень ризику і автоматично коригує рівень доступу. Це дозволяє оперативно реагувати на потенційно небезпечні ситуації і забезпечувати захист інформаційних ресурсів;

– адаптивність: У разі підвищення рівня ризику система автоматично змінює політику доступу. Наприклад, при підозрілій активності (незвичайному місці доступу або високій кількості запитів на доступ до конфіденційних даних) система може тимчасово обмежити доступ або запросити додаткову аутентифікацію;

– інтеграція з іншими системами моніторингу: Динамічний контроль доступу часто інтегрується з системами моніторингу, такими як SIEM (системи управління інформацією про безпеку та події) або IDS (системи виявлення вторгнень), що дозволяє автоматично виявляти підозрілі активності та адаптувати політики доступу відповідно до зміни ризиків.

Модель адаптивного контролю доступу на основі ризику.

Ця модель базується на динамічній оцінці ризиків і включає такі етапи:

– збір даних: система збирає інформацію про кожен запит на доступ, включаючи дані про користувача, пристрій, геолокацію, час і інші параметри;

– аналіз поведінки: оцінюється поведінка користувача в порівнянні з його історичними даними для виявлення аномальних дій;

– оцінка ризику: на основі зібраних даних визначається рівень ризику, використовуючи матрицю оцінки ризиків або машинне навчання;

– коригування доступу: система динамічно змінює рівень доступу. При низькому ризику доступ надається без додаткових заходів, при середньому - може вимагатися додаткова перевірка, а при високому ризику доступ блокується.

Політики на основі атрибутів та контексту.

У цьому підході динамічний контроль доступу базується на атрибутах користувача і контексту доступу:

– атрибути користувача: система враховує посаду, роль, права доступу та інші характеристики користувача, які можуть впливати на рівень ризику;

– контекст доступу: аналізується місце знаходження користувача, тип пристрою, з якого здійснюється доступ, час доби, тип мережі (приватна чи публічна), а також попередня поведінка користувача;

– реагування на зміни: якщо атрибути або контекст змінюються, система автоматично коригує політику доступу. Наприклад, користувач, що здійснює доступ з корпоративного пристрою під час робочого часу, може отримати доступ без додаткових перевірок, тоді як доступ з особистого пристрою поза робочим часом потребуватиме додаткової аутентифікації [16].

Завдяки штучному інтелекту та машинному навчанню динамічний контроль доступу може бути ще більш точним і ефективним:

– аналіз великих обсягів даних: AI-системи можуть аналізувати великі обсяги історичних даних для виявлення патернів поведінки користувачів;

– прогнозування ризику: за допомогою алгоритмів машинного навчання система прогнозує рівень ризику на основі поведінки користувача, його взаємодій та даних про поточні кіберзагрози;

– автоматичне коригування політик: система може автоматично змінювати політику доступу на основі навчання та аналізу, що дозволяє реагувати на нові загрози ще до їхнього активного виявлення.

Модель динамічної багатофакторної аутентифікації (MFA) на основі ризику.

Ця модель базується на динамічному підборі рівня аутентифікації, вимагаючи додаткових підтверджень при підвищеному рівні ризику:

– звичайний вхід: якщо система визначає низький рівень ризику, користувач може увійти лише за допомогою основного пароля;

– підвищений рівень аутентифікації: при визначенні середнього або високого рівня ризику система вимагає додаткові фактори, такі як одноразовий код (OTP) або біометричні дані;

– адаптивність: на основі різних факторів (наприклад, пристрою або геолокації) рівень аутентифікації коригується, щоб забезпечити баланс між безпекою та зручністю користувачів.

Поведінковий аналіз та відстеження аномалій.

Цей підхід полягає у виявленні аномальної активності, яка може свідчити про несанкціоновані спроби доступу:

– порівняння з типовою поведінкою: система створює профіль звичайної поведінки користувача (час доступу, типи запитуваних ресурсів, локація) і виявляє будь-які відхилення;

– інтеграція з SIEM/IDS: система використовує дані з інших засобів безпеки для аналізу подій та виявлення аномалій у поведінці;

– автоматична реакція на аномалії: при виявленні аномалії система автоматично вживає заходів, таких як тимчасове блокування доступу або вимога додаткової перевірки.

Переваги динамічного контролю доступу з урахуванням рівня ризику:

– зменшення кількості загроз, пов'язаних з компрометацією доступу: Динамічний контроль доступу дозволяє оперативно реагувати на спроби несанкціонованого доступу, що знижує ймовірність компрометації облікових записів;

– гнучкість і адаптивність: Цей підхід забезпечує гнучкий контроль доступу, що дозволяє швидко адаптуватися до змін у середовищі та оперативно реагувати на нові кіберзагрози;

– збалансованість між безпекою та зручністю: Динамічний контроль доступу дозволяє мінімізувати втручання у випадках низького ризику, підвищуючи зручність для користувачів, і водночас посилює захист у разі підозрілих дій;

– підвищена безпека критичних даних: Завдяки ризик-орієнтованому підходу забезпечується високий рівень захисту для критично важливих даних, до яких застосовуються жорсткіші вимоги.

Недоліки та виклики впровадження:

– складність налаштування та адміністрування: Впровадження ризик-орієнтованого підходу вимагає детального налаштування політик, збору й аналізу даних, що може бути складним завданням для великих організацій;

– потреба в постійному моніторингу та корекції: Для забезпечення ефективності динамічного контролю доступу потрібно регулярно коригувати політики, вивчаючи нові загрози і змінюючи параметри безпеки відповідно до змін у поведінці користувачів та нових типів атак;

– ресурси для обробки даних у реальному часі: Динамічна оцінка ризику потребує значних ресурсів для аналізу великих обсягів даних у реальному часі, що може вимагати використання сучасного апаратного забезпечення або технологій хмарних обчислень [19].

Динамічний контроль доступу з урахуванням рівня ризику використовує кілька підходів, що дозволяють адаптувати політики доступу до поточних умов і ризиків. Кожен з цих методів має власні сильні та слабкі сторони, що впливають на його ефективність та зручність використання. Для наочності ключові особливості, переваги та недоліки кожного методу зібрані в таблиці 1.1, що допоможе порівняти їх і вибрати найвідповідніший варіант для конкретних умов і вимог безпеки:

Таблиця 1.1 – Порівняння методів динамічного контролю доступу з урахуванням рівня ризику

Метод контролю доступу	Опис	Переваги	Недоліки
Адаптивний контроль доступу на основі ризику	Динамічно оцінює ризики та коригує рівень доступу на основі поведінки користувача та параметрів запиту.	Гнучкий контроль, знижує ризики компрометації доступу, адаптивність до умов.	Складність налаштування, потребує постійного моніторингу і корекції.
Політики на основі атрибутів та контексту	Враховує атрибути користувача, ресурсів і контекст доступу (час, місце, пристрій) для визначення доступу.	Детальний контроль, можливість враховувати різні умови доступу, підвищена безпека.	Потребує розширених налаштувань та моніторингу для обліку зміни контексту.
Використання AI та машинного навчання	Аналізує великі обсяги даних для прогнозування ризиків та автоматичного коригування політик доступу.	Прогнозування загроз, постійне самонавчання, виявлення нових патернів.	Високі вимоги до ресурсів, можливість помилкових спрацювань при аномальних діях.
Динамічна багатофакторна аутентифікація (MFA)	Автоматично вимагає додаткові фактори аутентифікації при підвищеному ризику для збереження безпеки.	Баланс безпеки і зручності, адаптується до рівня ризику, посилює захист.	Додаткове навантаження на користувачів при частих запитах на MFA.
Поведінковий аналіз та відстеження аномалій	Виявляє аномалії на основі відхилення від звичайної поведінки користувача та автоматично коригує доступ.	Ефективний захист від інсайдерських загроз, автоматична реакція на аномалії.	Необхідність високоякісних даних для аналізу, ресурсоємність та налаштування SIEM/IDS.

Таблиця 1.1 «Порівняння методів динамічного контролю доступу з урахуванням рівня ризику» надає стислий огляд ключових методів управління доступом, які використовують оцінку ризику.

Методи динамічного контролю доступу з урахуванням рівня ризику є потужним інструментом для захисту інформаційних систем в умовах сучасних загроз. Використовуючи адаптивний підхід, ці методи забезпечують гнучкість, безпеку та зручність для користувачів, що особливо важливо у великих та складних інфраструктурах.

1.5 Висновки та постановка задачі

В рамках цієї роботи було проведено дослідження теоретичних основ підвищення захищеності системи управління доступом на основі вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій. Проведене дослідження дозволило оцінити важливі аспекти безпеки, пов’язані з динамічним управлінням доступом, та виділити фактори, що впливають на ефективність протидії загрозам, орієнтуючись на аналіз поведінки користувачів і контексту доступу [20].

На основі отриманих результатів та проведеного аналізу сформульовано такі завдання для подальшої розробки:

– розробка алгоритму адаптивного управління доступом з урахуванням ризику та контексту;

– проектування динамічної багатофакторної аутентифікації на основі рівня ризику;

– створення схеми поведінкового аналізу для виявлення аномалій у системі доступу;

– практична реалізація та тестування системи з функцією моніторингу аномалій для забезпечення безперервного контролю та захисту від сучасних загроз.

2 РОЗРОБКА ЗАХИЩЕНОЇ СИСТЕМИ УПРАВЛІННЯ ДОСТУПОМ ВДОСКОНАЛЕНИМИ АДАПТИВНИМИ АЛГОРИТМАМИ КОНТРОЛЮ ДОСТУПУ ТА ДИНАМІЧНОГО МОНІТОРИНГУ АНОМАЛІЙ НА ОСНОВІ ПРИНЦИПІВ РИЗИК-ОРІЄНТОВАНИХ ТА АТРИБУТИВНИХ ПОЛІТИК ДОСТУПУ

У даному розділі представлено розробку захищеної системи управління доступом, яка базується на вдосконалених адаптивних алгоритмах контролю доступу та динамічного моніторингу аномалій. У розділі описуються особливості створення захищеності системи, зокрема використання адаптивних підходів, які дозволяють динамічно змінювати рівень доступу залежно від ризиків і контексту. Розглядаються аспекти впровадження принципів ризик-орієнтованих та атрибутивних політик доступу, які забезпечують гнучкість і точність управління правами користувачів. Особлива увага приділяється розробці алгоритмів контролю доступу, що інтегрують оцінку ризиків і поведінкові особливості користувачів. Завершується розділ висновками, які підсумовують отримані результати та формують основу для подальшої реалізації системи.

2.1 Особливості створення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу

Підвищення захищеності системи управління доступом є критично важливим у сучасному кіберсередовищі, де загрози постійно змінюються і стають все складнішими. Одним із найбільш ефективних способів забезпечення безпеки є застосування вдосконалених адаптивних алгоритмів контролю доступу, що здатні автоматично змінювати свої політики відповідно до умов, у яких здійснюється запит на доступ. У цьому підрозділі розглянуто ключові аспекти, що визначають особливості створення системи захищеності з використанням адаптивних алгоритмів контролю доступу [21].

Основні особливості адаптивних алгоритмів контролю доступу

- динамічний підхід до контролю доступу

Адаптивні алгоритми контролю доступу забезпечують динамічний підхід, який дозволяє автоматично змінювати політики доступу в реальному часі, враховуючи поточний рівень ризику. Цей підхід передбачає, що доступ до ресурсів коригується на основі аналізу атрибутів користувача, пристрою, місцезнаходження, часу запиту та інших факторів, що формують контекст.

Важливість динамічності полягає в тому, що дозволяє забезпечити високий рівень безпеки без жорсткого обмеження доступу, застосовуючи додаткові заходи тільки при підвищеному ризику.

- контекстуальність у прийнятті рішень щодо доступу

Адаптивні алгоритми базуються на контекстуальному підході, який враховує численні умови доступу для підвищення точності рішень. Наприклад, доступ до системи може залежати від часу доби, IP-адреси, попередніх дій користувача, типу пристрою, з якого здійснюється запит, та рівня довіри до мережі (публічна чи корпоративна).

Перевага контекстуальності полягає в тому, що вона дозволяє системі виявляти підозрілу поведінку ще на етапі запиту доступу та відповідно адаптуватися до змін у поведінці користувачів.

- оцінка ризику в режимі реального часу

Однією з ключових характеристик адаптивних алгоритмів є здатність оцінювати ризики при кожному запиті доступу. Для цього використовуються попередньо визначені моделі ризику, які враховують різні фактори. Наприклад, при спробі входу з нового місця або пристрою рівень ризику підвищується, що автоматично активує додаткові перевірки.

Реальна оцінка ризику дозволяє знизити кількість компрометацій даних, запобігаючи несанкціонованому доступу на основі аномальних параметрів.

- інтеграція з іншими системами захисту

Ефективні адаптивні алгоритми контролю доступу інтегруються з іншими системами безпеки, такими як системи моніторингу безпеки (SIEM), аналітика поведінки користувачів (UEBA) та системи виявлення вторгнень (IDS). Це

дозволяє отримувати додаткові дані для оцінки ризиків та відстеження аномальної активності, що є критичним для своєчасного реагування на загрози.

Переваги інтеграції полягають у забезпеченні більш повного і гнучкого контролю доступу, оскільки інформація з інших систем дозволяє краще оцінювати ситуацію та адаптувати політики відповідно до рівня загроз.

Використання атрибутивних та ризик-орієнтованих політик

- атрибутивний підхід до контролю доступу

Використання атрибутивного підходу в адаптивних алгоритмах дозволяє враховувати різні атрибути користувача, такі як роль, посада, права доступу, а також параметри запиту (геолокація, тип мережі, пристрій). Ці атрибути використовуються для динамічного прийняття рішень, залежно від чутливості даних, до яких здійснюється доступ [22].

Атрибутивний підхід надає можливість більш гнучкого налаштування політик доступу, знижуючи ризики при доступі до критично важливих даних.

- ризик-орієнтований підхід до прийняття рішень

Ризик-орієнтований підхід забезпечує, щоб кожен запит на доступ оцінювався з точки зору рівня ризику, пов’язаного з даними або операцією. При низькому ризику система може надати повний доступ, при середньому — вимагати додаткової перевірки (наприклад, багатофакторної аутентифікації), а при високому — відмовити у доступі.

Переваги ризик-орієнтованого підходу полягають у можливості забезпечення більшого рівня захисту в умовах постійно змінюваного кіберсередовища, дозволяючи автоматично коригувати політику доступу на основі аналізу ризиків.

Використання адаптивних алгоритмів для мінімізації людського фактору

- автоматизація рішень щодо доступу

Адаптивні алгоритми забезпечують високий рівень автоматизації у процесі прийняття рішень, що знижує залежність від людського фактору та ризик помилок. Система автоматично аналізує параметри доступу і приймає рішення, базуючись на встановлених політиках та поточних умовах.

Автоматизація дозволяє уникнути суб'єктивності та зменшує навантаження на команду безпеки, яка може зосередитись на критично важливих аспектах захисту.

- зниження необхідності ручного налаштування політик

Завдяки адаптивним алгоритмам контролю доступу система самостійно адаптує політики залежно від поточних загроз, що дозволяє скоротити кількість ручного втручання. Це знижує ймовірність помилок та забезпечує більш надійний захист у режимі реального часу.

Переваги автоматичного налаштування політик включають зниження витрат на адміністрування та забезпечення оперативного оновлення політик відповідно до нових загроз.

Особливості створення захищеної системи управління доступом з використанням вдосконалених адаптивних алгоритмів контролю доступу базуються на принципах динамічності, контекстуальності та ризик-орієнтованого підходу. Ці алгоритми забезпечують високу гнучкість, оскільки дозволяють адаптувати політики доступу до умов у реальному часі та знижувати ризики, враховуючи контекст та поведінку користувачів. Інтеграція з іншими системами безпеки, такими як SIEM та UEBA, підвищує ефективність виявлення аномалій та реагування на загрози, що особливо важливо для забезпечення безпеки в умовах постійно змінюваних загроз.

2.2 Особливості впровадження принципів ризик-орієнтованих та атрибутивних політик доступу

Для забезпечення захищеності системи управління доступом у сучасних умовах важливо враховувати не лише стандартні механізми контролю, але й динамічні параметри, такі як рівень ризику та атрибути доступу. Ризик-орієнтовані та атрибутивні політики доступу дозволяють значно підвищити рівень безпеки, забезпечуючи при цьому гнучкість та адаптивність системи управління доступом.

Основні принципи ризик-орієнтованих політик доступу

- динамічна оцінка ризику в режимі реального часу

Ризик-орієнтовані політики доступу передбачають постійну оцінку рівня ризику кожного запиту на доступ. Для цього аналізуються різні фактори, включаючи місце розташування користувача, пристрій, час, попередню поведінку та чутливість даних.

Оцінка ризику у реальному часі дозволяє автоматично змінювати рівень доступу залежно від умов: у випадках низького ризику користувачеві може бути наданий повний доступ, тоді як у випадках високого ризику — доступ може бути обмежений або заблокований.

- гнучка адаптація політик доступу

Завдяки ризик-орієнтованому підходу система може автоматично адаптувати свої політики, підлаштовуючись під поточний рівень загроз. Наприклад, якщо користувач входить до системи з незвичного місця або нового пристрою, система може активувати додаткові перевірки або обмежити доступ.

Перевага адаптивності полягає в тому, що система може швидко реагувати на нові загрози без необхідності ручного втручання, що підвищує швидкість і ефективність захисту.

- інтеграція з системами управління безпекою

Ризик-орієнтовані політики доступу інтегруються з іншими системами безпеки, такими як SIEM, для збору та аналізу інформації про події безпеки. Це дозволяє більш точно оцінювати ризики, використовуючи реальні дані про підозрілу активність і поведінку користувачів.

Інтеграція з SIEM забезпечує системі доступу додаткову інформацію, що дозволяє краще реагувати на потенційні загрози, адаптуючи політики доступу відповідно до отриманих даних [23].

- контекстуальний підхід до прийняття рішень

Ризик-орієнтовані політики доступу використовують контекстуальні дані для ухвалення рішень про доступ. Враховуються фактори, які можуть впливати на рівень ризику, наприклад, географічне розташування, пристрій, час доби та мережа. При незвичній активності система застосовує більш суворі політики доступу.

Контекстуальність дозволяє системі ідентифікувати підозрілі дії ще на етапі запиту доступу та відповідно адаптувати політики доступу залежно від ситуації.

Основні принципи атрибутивних політик доступу

- використання атрибутів для гнучкого управління доступом

Атрибутивні політики доступу дозволяють детальніше контролювати доступ, враховуючи конкретні атрибути користувача, такі як посада, відділ, роль, права доступу, а також атрибути пристрою (тип, рівень захисту) та мережі (публічна чи корпоративна).

Гнучкість атрибутивного підходу дозволяє налаштовувати доступ відповідно до унікальних характеристик користувачів і умов, що особливо корисно у великих організаціях з різними групами користувачів та даних [23].

- контекстуальна адаптація політик доступу

Атрибутивні політики можуть автоматично адаптуватися на основі контексту, в якому здійснюється запит доступу. Наприклад, доступ до певних ресурсів може надаватися тільки користувачам з певним рівнем привілеїв у робочий час або з корпоративної мережі.

Контекстуальність атрибутивного підходу дозволяє забезпечити більшу деталізацію у контролі доступу, знижуючи ризики за рахунок врахування додаткових умов, що робить систему більш захищеною.

- автоматизація на основі атрибутів

Система може автоматично коригувати політики доступу, використовуючи атрибути в режимі реального часу. Наприклад, зміни в атрибутах користувача (переведення на іншу посаду, зміна рівня доступу) автоматично враховуються в політиках доступу.

Автоматичне оновлення політик на основі атрибутів знижує потребу в ручному адмініструванні та забезпечує актуальність політик відповідно до реальної ситуації в організації.

- реалізація політик найменших привілеїв

Атрибутивні політики сприяють реалізації принципу найменших привілеїв, надаючи користувачам лише мінімально необхідний доступ відповідно до їхніх атрибутів і поточного контексту.

Найменші привілеї забезпечують мінімізацію ризиків, пов’язаних із надмірним доступом, що знижує ймовірність несанкціонованих дій або компрометації даних.

Виклики впровадження ризик-орієнтованих та атрибутивних політик доступу

- складність налаштування та адміністрування

Впровадження ризик-орієнтованих і атрибутивних політик вимагає налаштування великої кількості параметрів та постійного моніторингу. Це може ускладнити процес для великих організацій з численними групами користувачів і різними рівнями доступу.

Необхідність у комплексному налаштуванні може вимагати додаткових ресурсів і часу, особливо на етапі впровадження.

- потреба у високоякісних даних для аналізу

Для ефективного функціонування системи управління доступом необхідні точні дані про поведінку користувачів, атрибути, контекст і рівні ризику. Неповні або некоректні дані можуть призвести до неправильного прийняття рішень щодо доступу.

Якість даних має вирішальне значення для коректної роботи ризик-орієнтованих та атрибутивних політик доступу, тому необхідно забезпечити безперервний процес збору і валідації даних.

- інтеграція з існуючими системами безпеки

Для досягнення максимального ефекту ризик-орієнтовані та атрибутивні політики потребують інтеграції з іншими засобами безпеки, такими як SIEM, IDS, та аналітика поведінки користувачів (UEBA). Це може бути складним завданням через різні вимоги систем і можливі конфлікти політик.

Складність інтеграції з існуючими системами безпеки може призвести до додаткових витрат і потреби в налаштуванні сумісності різних інструментів.

- забезпечення збалансованості між безпекою і зручністю

Іноді суворі політики можуть знижувати зручність для користувачів, що може призвести до зниження продуктивності та створення перешкод у роботі. Для ефективного впровадження політик важливо забезпечити баланс між безпекою та зручністю.

Баланс між безпекою та зручністю є важливим завданням, оскільки суворі політики можуть негативно вплинути на користувацький досвід і ефективність робочих процесів.

Впровадження ризик-орієнтованих та атрибутивних політик доступу дозволяє значно підвищити рівень захищеності інформаційних систем, забезпечуючи динамічний і гнучкий контроль доступу. Завдяки динамічній оцінці ризиків та врахуванню атрибутів і контексту доступу система може оперативно адаптувати політики до умов у реальному часі, знижуючи ймовірність несанкціонованого доступу та компрометації даних. Проте впровадження таких політик вимагає ретельного налаштування, інтеграції з іншими системами безпеки та забезпечення балансу між безпекою і зручністю для користувачів [24].

2.3 Розробка алгоритму контролю доступу

Розробка алгоритму контролю доступу для захищеної інформаційної системи на основі ризик-орієнтованих і атрибутивних політик дозволяє забезпечити надійний захист даних, гнучкість у керуванні доступом і адаптацію до мінливих умов. Основою такого підходу є динамічна оцінка ризиків і використання атрибутів користувачів та контексту доступу, що забезпечує прийняття рішень у реальному часі, підвищуючи безпеку і точність контролю доступу.

Ключові етапи розробки алгоритму контролю доступу

1. Збір та аналіз атрибутів доступу

Атрибути користувача: Алгоритм збирає та обробляє різні атрибути користувачів, такі як їхня роль, рівень привілеїв, посада, історія активності та

рівень авторизації. Ці атрибути допомагають визначити базові права користувача та характер запитуваного доступу.

Атрибути контексту доступу: До контекстуальних атрибутів належать місце доступу, пристрій, з якого здійснюється запит (персональний або корпоративний), час доби, а також мережа (публічна або захищена), через яку користувач підключається до системи.

Атрибути ресурсів: Алгоритм також враховує тип та чутливість ресурсів, до яких запитано доступ. Це дозволяє адаптувати політики доступу залежно від того, чи є дані конфіденційними або критичними для системи.

2. Оцінка рівня ризику в режимі реального часу

Моделювання ризиків: На основі отриманих атрибутів алгоритм використовує матрицю ризиків або попередньо налаштовані моделі для оцінки рівня загрози. Моделювання ризиків може включати шкали від низького до високого рівня, де кожен рівень визначає необхідні заходи для забезпечення безпеки.

Визначення ризикових сценаріїв: Алгоритм враховує можливі ризикові сценарії, такі як незвична активність користувача, вхід з незнайомого пристрою чи геолокації. Це дозволяє точніше ідентифікувати підозрілі дії ще на етапі запиту доступу.

Реальна оцінка ризику: Після аналізу атрибутів алгоритм класифікує ризик для кожного запиту як низький, середній або високий. Такий підхід дозволяє автоматично визначити рівень перевірок, необхідних для надання доступу.

3. Прийняття рішень щодо доступу на основі оцінки ризику та атрибутів

Модель прийняття рішень: Використовуючи результати оцінки ризику та атрибути доступу, алгоритм приймає рішення про надання або відмову в доступі. Наприклад:

Низький ризик: доступ надається без додаткових перевірок.

Середній ризик: доступ дозволяється після проходження багатофакторної аутентифікації.

Високий ризик: доступ обмежується або повністю блокується для захисту системи.

Гнучкість у прийнятті рішень: У разі зміни умов доступу або поведінки користувача алгоритм автоматично адаптує рішення, змінюючи рівень доступу або вимагаючи додаткових перевірок.

4. Застосування адаптивних політик доступу

Атрибутивні політики доступу: На основі атрибутів користувача та контексту запиту алгоритм вибирає відповідні політики. Наприклад, для працівників з високим рівнем привілеїв дозволяється доступ лише з корпоративних пристроїв у робочий час.

Ризик-орієнтовані політики: Алгоритм автоматично коригує політики доступу відповідно до рівня ризику. Наприклад, при спробі доступу з незвичного місця алгоритм може вимагати додаткову аутентифікацію або відмовити в доступі до деяких чутливих ресурсів.

5. Виявлення та реагування на аномалії

Моніторинг активності користувачів: Алгоритм постійно відстежує дії користувачів, порівнюючи їх з типовими паттернами поведінки. Це дозволяє оперативно виявляти аномалії та підозрілу активність.

Виявлення аномалій та автоматична реакція: У разі виявлення аномальної активності алгоритм автоматично змінює політику доступу або надсилає сповіщення адміністраторам системи. Це може включати тимчасове обмеження доступу або активацію додаткових заходів безпеки.

Інтеграція з іншими системами моніторингу: Алгоритм може бути інтегрований із SIEM-системами або системами виявлення вторгнень (IDS), що дозволяє ефективніше відстежувати загрози та реагувати на них у реальному часі.

6. Інтеграція з іншими системами кібербезпеки

Взаємодія з SIEM: Інтеграція з системами управління інформацією про безпеку та події (SIEM) дозволяє алгоритму використовувати додаткові дані для оцінки ризиків. Це покращує точність виявлення аномалій та адаптивність політик доступу.

Обмін інформацією про загрози: Алгоритм підтримує обмін даними з іншими системами кібербезпеки, що дозволяє своєчасно виявляти нові загрози, адаптувати політики доступу та коригувати алгоритм в реальному часі.

Інтеграція з DLP та IDS: Використання систем попередження витоків даних (DLP) та виявлення вторгнень (IDS) дозволяє забезпечити більш повний захист інформаційних ресурсів, своєчасно реагуючи на спроби несанкціонованого доступу.

Структура алгоритму контролю доступу

Алгоритм контролю доступу розділяється на кілька основних блоків:

- збір даних та аналіз атрибутів: система збирає дані про користувача, його атрибути та контекст доступу, а також інформацію про поточні загрози.
- оцінка ризику: на основі зібраних даних алгоритм оцінює рівень ризику та визначає рівень доступу.
- прийняття рішень щодо доступу: на основі оцінки ризику і атрибутів система ухвалює рішення щодо доступу.
- моніторинг та реагування на аномалії: система відстежує поведінку користувачів для виявлення аномалій та адаптує політики доступу у разі виникнення підозрілої активності.
- інтеграція з іншими системами: алгоритм взаємодіє з SIEM, IDS, DLP та іншими системами для підвищення ефективності контролю доступу.

Розробимо алгоритм на основі даної інформації (рис.2.1)

Розберемо покроково даний алгоритм:

Крок 1: Ініціація запиту на доступ

Користувач здійснює запит на доступ до певного ресурсу або системи. Запит містить ідентифікатор користувача, інформацію про запитуваний ресурс і час запиту.

Крок 2: Збір базових атрибутів користувача

Алгоритм автоматично збирає основні атрибути користувача, такі як його роль, посада, рівень привілеїв і групи, до яких він належить. Ці дані використовуються як базові для визначення рівня доступу.

Крок 3: Аналіз атрибутів пристрою

Алгоритм визначає, з якого пристрою здійснюється запит, враховуючи тип пристрою (наприклад, персональний чи корпоративний), його операційну систему, рівень безпеки і наявність оновлень безпеки.

Крок 4: Перевірка атрибутів мережі

Система перевіряє, чи підключений користувач до корпоративної або публічної мережі. Це важливо для оцінки безпеки з’єднання та ризику, пов’язаного з підключенням через публічні мережі.

Крок 5: Збір контекстуальних даних

Алгоритм враховує додаткові параметри контексту, такі як місце розташування користувача, час доби та попередня активність у системі. Наприклад, доступ у робочий час з офісу зазвичай є менш ризикованим, ніж пізно вночі з публічної мережі.

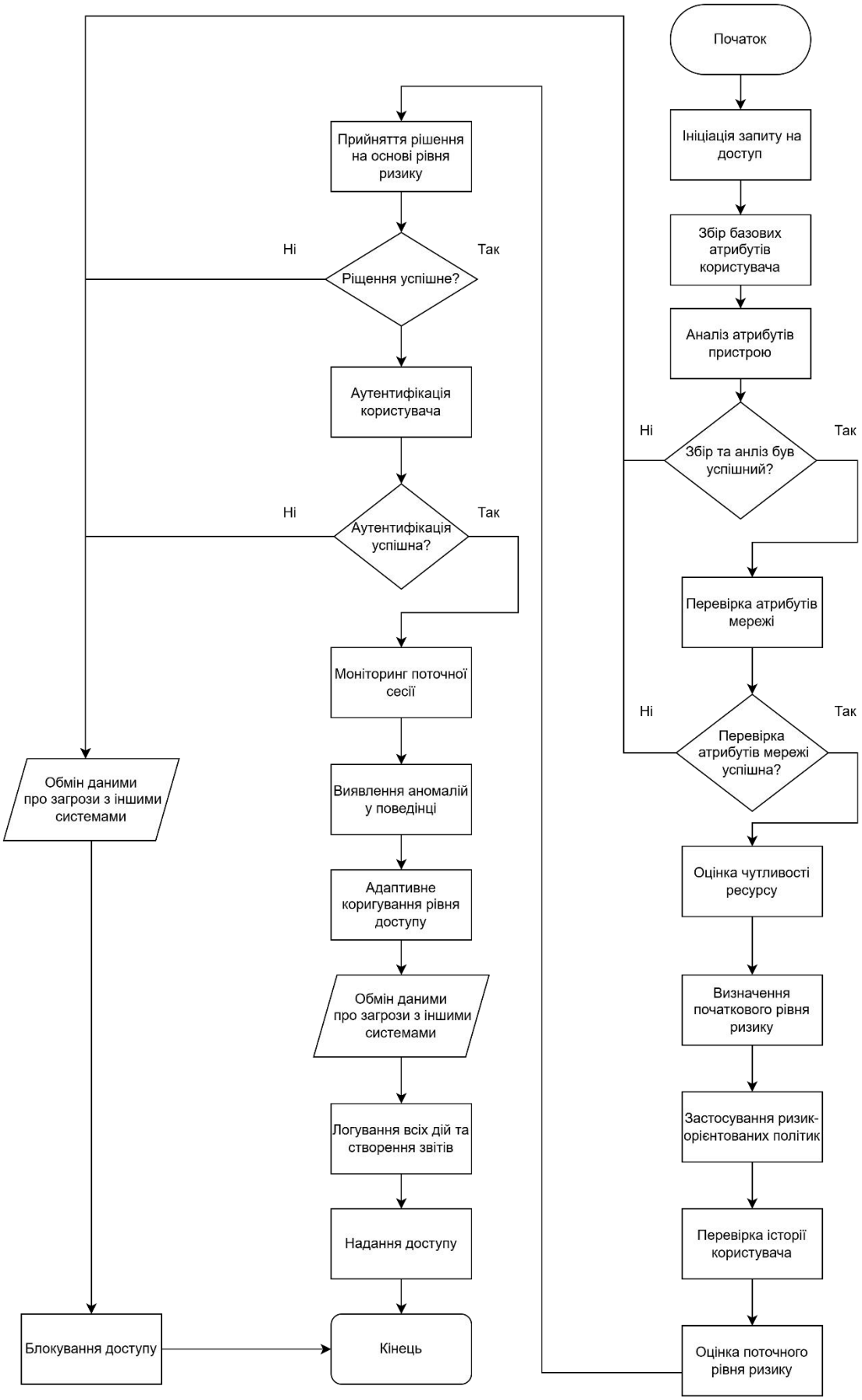


Рисунок 2.1 -Алгоритм

Крок 6: Оцінка чутливості ресурсу

Алгоритм аналізує, до якого ресурсу здійснюється доступ, враховуючи його чутливість. Наприклад, доступ до конфіденційних даних вимагає суворіших заходів безпеки, ніж доступ до загальнодоступної інформації.

Крок 7: Визначення початкового рівня ризику

На основі зібраних атрибутів і контексту система визначає початковий рівень ризику. Це оцінка базується на правилах і політиках безпеки, де кожен атрибут має свою вагу в розрахунку рівня ризику.

Крок 8: Застосування ризик-орієнтованих політик

Якщо початковий рівень ризику середній або високий, алгоритм застосовує відповідні ризик-орієнтовані політики. Наприклад, для середнього ризику може вимагатися багатофакторна аутентифікація (MFA), а для високого — обмеження доступу або повна відмова.

Крок 9: Перевірка історії користувача

Алгоритм аналізує попередню активність користувача для виявлення аномалій. Якщо поточний запит суттєво відрізняється від звичної поведінки, рівень ризику підвищується.

Крок 10: Оцінка поточного рівня ризику

Після аналізу історії та застосування ризик-орієнтованих політик алгоритм повторно оцінює рівень ризику, оновлюючи його відповідно до нових даних. Це може змінити рівень ризику на вищий або нижчий.

Крок 11: Прийняття рішення на основі рівня ризику

Алгоритм приймає рішення про доступ, враховуючи поточний рівень ризику:

Низький ризик: доступ дозволяється без додаткових перевірок.

Середній ризик: система запитує багатофакторну аутентифікацію для підтвердження доступу.

Високий ризик: доступ обмежується або блокується.

Крок 12: Аутентифікація користувача (якщо потрібно)

Якщо для підтвердження запиту потрібна багатофакторна аутентифікація, користувач отримує запит на додаткову перевірку (наприклад, OTP, біометричні дані). Лише після проходження цієї аутентифікації користувач отримає доступ.

Крок 13: Моніторинг поточної сесії

Алгоритм продовжує моніторинг активності користувача після отримання доступу. Якщо виявлено підозрілі дії, такі як нетиповий обсяг завантаження даних або запити на незвичні ресурси, алгоритм оновлює рівень ризику та може змінити політику доступу.

Крок 14: Виявлення аномалій у поведінці

Алгоритм порівнює поточну активність користувача з типовим профілем поведінки. Виявлення аномалій (наприклад, запити на доступ до незвичних ресурсів) активує додаткові заходи безпеки, такі як тимчасове обмеження доступу або сповіщення адміністратора.

Крок 15: Адаптивне коригування рівня доступу

Якщо виявлено ризиковану поведінку, алгоритм адаптує рівень доступу користувача в реальному часі. Це може включати зниження рівня доступу, обмеження доступу до певних ресурсів або тимчасове блокування.

Крок 16: Інтеграція з SIEM для додаткового аналізу

Дані про активність користувача передаються до SIEM (системи управління інформацією про безпеку та події), яка здійснює поглиблений аналіз загроз. Це дозволяє отримати розширену інформацію про можливі загрози та підвищує точність алгоритму.

Крок 17: Оновлення політик доступу

На основі даних про ризики та загрози алгоритм автоматично оновлює політики доступу для забезпечення відповідності новим загрозам. Це дозволяє адаптуватися до нових умов і ризиків без необхідності втручання адміністратора.

Крок 18: Обмін даними про загрози з іншими системами

Алгоритм обмінюється даними про виявлені загрози з іншими системами кібербезпеки, такими як IDS (системи виявлення вторгнень) або DLP (системи запобігання витоку даних). Це підвищує рівень захисту всієї системи.

Крок 19: Логування всіх дій та створення звітів

Алгоритм реєструє всі дії користувача, включаючи спроби доступу, проходження аутентифікації та виявлені аномалії. Ці дані використовуються для створення звітів про активність і для подальшого аудиту системи безпеки.

Цей алгоритм забезпечує багаторівневий підхід до управління доступом, враховуючи динамічні умови, оцінку ризиків і атрибути користувачів, що дозволяє швидко адаптуватися до загроз та гарантувати безпеку інформаційної системи.

2.5 Висновки до розділу.

У другому розділі було детально розглянуто процес розробки захищеної системи управління доступом, що використовує вдосконалені адаптивні алгоритми контролю доступу та динамічний моніторинг аномалій. Основні елементи цієї системи, такі як оцінка ризиків у реальному часі, атрибутивні та ризик-орієнтовані політики доступу, а також алгоритм виявлення аномальної активності, дозволяють ефективно контролювати доступ та забезпечують високий рівень захищеності інформаційних ресурсів.

Розроблені адаптивні алгоритми контролю доступу враховують численні атрибути користувача, контекст доступу і чутливість запитуваних даних, що дозволяє гнучко регулювати політики безпеки залежно від поточного рівня ризику. Такий підхід значно підвищує стійкість системи до несанкціонованого доступу, зловживання привілеями та інсайдерських загроз.

Алгоритм динамічного моніторингу аномалій, що використовує методи поведінкового аналізу та машинного навчання, виявляє відхилення від типових патернів активності користувачів, оперативно оцінюючи рівень ризику кожної аномалії. Це дозволяє вчасно реагувати на підозрілу активність і запобігати можливим порушенням безпеки. Інтеграція алгоритму з іншими системами безпеки (такими як SIEM та IDS) дозволяє отримувати додаткові дані про загрози, що покращує точність і швидкість реагування на потенційні загрози.

Таким чином, розроблена система управління доступом забезпечує надійний захист за рахунок гнучкого, адаптивного підходу до контролю доступу та динамічного виявлення загроз, що відповідає сучасним вимогам кібербезпеки. Подальші дослідження можуть бути спрямовані на оптимізацію алгоритмів та вдосконалення методів оцінки ризиків для ще більш ефективного управління доступом і виявлення загроз.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАХИЩЕНОЇ СИСТЕМИ УПРАВЛІННЯ ДОСТУПОМ ВДОСКОНАЛЕНИМИ АДАПТИВНИМИ АЛГОРИТМАМИ КОНТРОЛЮ ДОСТУПУ ТА ДИНАМІЧНОГО МОНІТОРИНГУ АНОМАЛІЙ НА ОСНОВІ ПРИНЦИПІВ РИЗИК-ОРІЄНТОВАНИХ ТА АТРИБУТИВНИХ ПОЛІТИК ДОСТУПУ

У даному розділі представлено програмну реалізацію захищеної системи управління доступом, що базується на вдосконалених адаптивних алгоритмах контролю доступу та динамічного моніторингу аномалій. Реалізація включає вибір мови програмування, моделювання бази даних, створення модулів для управління доступом, моніторингу аномалій та інтерактивного графічного інтерфейсу. Кожен компонент системи розроблено з урахуванням принципів ризик-орієнтованого та атрибутивного управління доступом.

Особлива увага приділена обґрунтуванню вибору інструментів і технологій, які забезпечують ефективність та масштабованість системи. Описано процес інтеграції компонентів, їхню взаємодію та впровадження адаптивних алгоритмів для оцінки ризиків у реальному часі. Завершальним етапом є тестування системи, що підтверджує її коректність, надійність та відповідність поставленим вимогам.

У цьому розділі поетапно розглядаються ключові аспекти програмної реалізації, які забезпечують створення функціональної та безпечної системи управління доступом.

3.1 Обґрунтування вибору мови програмування

Для реалізації захищеної системи управління доступом було обрано екосистему Node.js, а саме використання таких фреймворків, як NestJS для створення бекенду та Next.js для фронтенду. Обґрунтування цього вибору базується на ряді технічних, функціональних та практичних переваг, які забезпечують ефективну реалізацію системи [28].

Для реалізації системи управління доступом було обрано Node.js як основну платформу, разом із NestJS для бекенду та Next.js для фронтенду. Такий вибір

зумовлений рядом технічних, функціональних і практичних переваг, які забезпечують ефективність, безпеку та гнучкість розробленого рішення.

Node.js є популярною платформою для створення високопродуктивних систем, що обробляють численні запити одночасно завдяки неблокуючій моделі вводу/виводу. Її асинхронна природа дозволяє зменшити затримки під час виконання завдань і забезпечити високу продуктивність навіть за умови інтенсивного використання. Окрім того, платформа підтримує роботу з TypeScript, що дозволяє впроваджувати статичну типізацію, зменшуючи кількість помилок і підвищуючи стабільність системи. Екосистема Node.js містить безліч бібліотек і фреймворків, доступних через npm, що значно пришвидшує процес розробки.

Node.js — це середовище виконання JavaScript, яке дозволяє запускати код не в браузері, а на сервері. Воно побудоване на основі V8, високопродуктивного рушія JavaScript від Google, який забезпечує швидке виконання коду. Основна перевага Node.js полягає у його асинхронній, неблокуючій моделі вводу/виводу, яка дозволяє обробляти багато одночасних запитів із мінімальним використанням ресурсів. Ця властивість робить Node.js ідеальним вибором для розробки систем реального часу, таких як чати, онлайн-ігри, системи управління доступом або аналітичні платформи [29].

Node.js має величезну екосистему пакетів і модулів, доступних через менеджер npm (Node Package Manager). Це дозволяє розробникам легко інтегрувати популярні бібліотеки та фреймворки, такі як Express, Koa або NestJS, які значно спрощують створення веб-серверів, API та мікросервісів. Завдяки цьому Node.js забезпечує високу швидкість розробки й дозволяє зосередитися на бізнес-логіці додатку.

Ще однією важливою перевагою Node.js є підтримка єдиної мови програмування — JavaScript, що використовується як для фронтенду, так і для бекенду. Це значно знижує складність розробки, оскільки команди розробників можуть використовувати ті самі інструменти, синтаксис і підходи для обох частин додатку. Крім того, розробники можуть легко переносити функціонал між різними

частинами системи, наприклад, виконувати обчислення як на клієнтській стороні, так і на серверній.

Node.js також підтримує TypeScript — супермножину JavaScript, яка додає статичну типізацію. Використання TypeScript у проектах на Node.js дозволяє знижувати кількість помилок у коді, підвищує його зрозумілість і стабільність, а також полегшує підтримку великих додатків.

Середовище Node.js часто використовується для створення RESTful API та GraphQL API. Завдяки своїй продуктивності та зручності, Node.js забезпечує швидку обробку запитів і відповідає навіть на великих обсягах даних. Інтеграція з базами даних, такими як MongoDB, PostgreSQL чи MySQL, реалізується за допомогою численних бібліотек, що доступні в екосистемі. Наприклад, Sequelize та TypeORM дозволяють працювати з реляційними базами даних, тоді як Mongoose забезпечує зручний інструментарій для роботи з MongoDB.

JavaScript — це динамічна, інтерпретована мова програмування, яка спочатку була створена для роботи у веб-браузерах. Вона дозволяє створювати інтерактивні інтерфейси, обробляти події користувача та працювати з DOM (Document Object Model). Однак завдяки появі Node.js JavaScript отримав можливість виконуватись поза браузером і використовуватись для серверних завдань. Це зробило мову універсальною для повноцінного розробницького циклу [30].

JavaScript підтримує кілька парадигм програмування, включаючи імперативне, об'єктно-орієнтоване та функціональне програмування. Це робить його гнучким і дозволяє розробникам використовувати різні підходи залежно від потреб проекту. Важливою особливістю JavaScript є його асинхронна природа, яка дозволяє ефективно працювати з великою кількістю запитів і обробляти події в реальному часі.

JavaScript має багату екосистему фреймворків і бібліотек, таких як React, Angular, Vue.js, що значно полегшує створення фронтенд-додатків. Завдяки такій потужній базі інструментів розробка веб-додатків стає швидшою, ефективнішою та більш стандартизованою.

Node.js та JavaScript разом створюють універсальну платформу, яка дозволяє розробникам працювати як над клієнтськими, так і над серверними частинами додатків. Це зменшує витрати на розробку, спрощує підтримку додатків і робить їх легкими для масштабування. Завдяки цим властивостям, комбінація Node.js і JavaScript є ідеальним вибором для сучасних веб-додатків і систем.

NestJS був обраний для реалізації серверної частини завдяки його модульній архітектурі, яка дозволяє легко організовувати та підтримувати великий кодовий базис. Цей фреймворк підтримує використання ORM, таких як TypeORM і Prisma, що спрощує роботу з базами даних. Він також надає вбудовані інструменти безпеки, включно з підтримкою JWT для аутентифікації, захистом від атак XSS і CSRF, а також інтеграцією middleware для обробки запитів. Додатковою перевагою NestJS є підтримка автоматизованого тестування, що забезпечує високу якість коду й стабільність системи. Його масштабованість дозволяє легко додавати нові функціональні модулі, такі як моніторинг аномалій чи управління політиками доступу [31].

Next.js було обрано для реалізації фронтенд-частини завдяки його можливостям швидкого створення веб-додатків із використанням серверного рендерингу (SSR) та статичного генератора сторінок (SSG). Ці технології підвищують продуктивність і забезпечують SEO-оптимізацію, що важливо для додатків, які потребують публічного інтерфейсу. Next.js дозволяє ефективно інтегруватися з бекендом через REST або GraphQL API, забезпечуючи швидку передачу даних між компонентами. Крім того, його функціонал дозволяє легко інтегрувати сучасні методи аутентифікації, такі як Auth0, або створити власну реалізацію на основі JWT. Висока продуктивність Next.js та його підтримка сучасних інструментів розробки значно пришвидшують процес створення зручного інтерфейсу для кінцевих користувачів.

Комбінація NestJS і Next.js у межах екосистеми Node.js забезпечує універсальність розробки. Завдяки єдиній мові програмування, TypeScript, яка використовується і для серверної, і для клієнтської частин, зменшується складність проекту, прискорюється впровадження змін і покращується комунікація між

членами команди. Крім того, це дозволяє повторно використовувати частини коду, спрощуючи інтеграцію між компонентами системи. NestJS та Next.js разом забезпечують гнучкість і масштабованість, що дозволяє розробляти складні рішення з мінімальними витратами ресурсів.

Next.js є сучасним фреймворком для розробки веб-додатків, створеним на основі React. Його ключовою особливістю є підтримка серверного рендерингу (Server-Side Rendering, SSR) та статичної генерації сторінок (Static Site Generation, SSG), що забезпечує високу продуктивність і поліпшення користувацького досвіду. Це робить Next.js ідеальним вибором для фронтенд-реалізації системи управління доступом, яка повинна бути швидкою, адаптивною та SEO-оптимізованою.

Однією з головних переваг Next.js є його здатність забезпечувати гібридний підхід до рендерингу. Залежно від вимог до конкретних сторінок додатку, можна обирати між SSR і SSG. Наприклад, у системі управління доступом сторінки з динамічно змінюваним вмістом, такими як панелі керування або звіти про моніторинг аномалій, можуть рендеритися на сервері, щоб забезпечити актуальність даних. З іншого боку, статичні сторінки, наприклад довідкова інформація чи політики доступу, можуть бути згенеровані наперед для зменшення навантаження на сервер і прискорення відгуку.

Next.js також підтримує API-роутинг, що дозволяє створювати серверні функції прямо у фронтенд-додатку. Це зручно для реалізації невеликих бекенд-функцій, таких як перевірка аутентифікаційних токенів або виконання простих запитів до бази даних. У системі управління доступом це може бути використано, наприклад, для обробки запитів на багатофакторну аутентифікацію (MFA) або валідацію атрибутів користувача без необхідності звертатися до окремого серверного додатка [33].

Фреймворк також забезпечує ефективну інтеграцію з бекендом через REST або GraphQL API. У нашій системі Next.js використовуватиметься для передачі даних між фронтендом і бекендом, реалізованим на NestJS. Це дозволить реалізувати динамічний інтерфейс користувача, який відображатиме актуальні дані про політики доступу, ризики та аномалії в реальному часі. Завдяки підтримці

TypeScript інтеграція є ще більш зручною та безпечною, оскільки статична типізація забезпечує точність взаємодії між різними компонентами системи.

Ще однією важливою перевагою Next.js є його підтримка оптимізації продуктивності. Фреймворк забезпечує автоматичне розділення коду (code splitting), що дозволяє завантажувати лише ті частини додатку, які необхідні в конкретний момент. Це значно знижує час завантаження сторінок і покращує швидкість роботи системи. Крім того, Next.js має вбудовані механізми кешування та попереднього завантаження сторінок (prefetching), які сприяють швидкому переходу між сторінками.

З точки зору безпеки Next.js також пропонує низку важливих можливостей. Він дозволяє легко інтегрувати сучасні методи аутентифікації, такі як Auth0, Okta або власні реалізації на основі JWT. Крім того, вбудовані механізми для захисту від атак XSS (міжсайтовий скриптинг) та CSRF (міжсайтові запити підробки) забезпечують безпеку користувацького інтерфейсу.

SEO-оптимізація є ще одним важливим аспектом Next.js. Серверний рендеринг забезпечує високу видимість сторінок у пошукових системах, що особливо важливо для корпоративних систем, які мають загальнодоступні інтерфейси або довідкову інформацію. Крім того, підтримка метатегів і мікророзмітки дозволяє точно налаштувати відображення контенту в пошукових системах.

Next.js має багату екосистему інструментів, які прискорюють розробку. Він інтегрується з такими популярними бібліотеками, як Tailwind CSS, Material-UI або Chakra UI, що дозволяє створювати естетичний і функціональний інтерфейс користувача. Крім того, фреймворк підтримує інструменти для тестування, такі як Jest та Cypress, що дозволяє забезпечити високу якість коду та стабільність додатку.

Отже, вибір Next.js як фронтенд-фреймворка для системи управління доступом обґрунтований його гнучкістю, продуктивністю, інтеграційними можливостями, безпекою та SEO-оптимізацією. Він дозволяє створити сучасний, швидкий і зручний інтерфейс користувача, який відповідає високим вимогам до функціональності та безпеки в умовах сучасних кіберзагроз [34].

Вибір цих інструментів забезпечує високу продуктивність, зручність розробки, надійність і безпеку системи управління доступом. Така технологічна основа дозволяє не лише відповідати сучасним вимогам кібербезпеки, але й створювати гнучкі, масштабовані та адаптивні системи, що здатні ефективно функціонувати в умовах постійно змінюваного цифрового середовища.

3.2 Реалізаця модуля керуванн базою даних

Модуль керування базою даних є одним із ключових компонентів системи управління доступом. Його основне завдання полягає в забезпеченні надійного зберігання даних про користувачів, їхні атрибути, політики доступу, а також логів активності та виявлених аномалій. Для реалізації цього модуля була обрана база даних PostgreSQL через її високу продуктивність, підтримку реляційної моделі даних та можливість працювати із складними запитами. Як ORM (об’єктно-реляційний мапер) використано TypeORM, що спрощує взаємодію з базою даних у проєктах на NestJS.

Архітектура модуля

Модуль керування базою даних в системі побудовано за принципами модульності та інкапсуляції. Він складається з таких компонентів:

Схеми бази даних, яка визначає структуру таблиць, зв’язків між ними та обмежень.

Моделей даних, які відповідають об'єктам у програмному коді та дозволяють ORM автоматично генерувати SQL-запити.

Сервісів для роботи з базою даних, що містять бізнес-логіку доступу до даних і забезпечують централізований контроль операцій CRUD (створення, зчитування, оновлення та видалення).

Схема бази даних

База даних містить кілька основних таблиць, які зберігають інформацію про користувачів, політики доступу, логіни, дії та аномалії:

- Users: зберігає інформацію про користувачів, включаючи ідентифікатори, імена, ролі, паролі (у хешованому вигляді) та інші атрибути.
- Policies: містить дані про політики доступу, включаючи правила, пов’язані з рівнями привілеїв.
- Logs: використовується для збереження історії дій користувачів та виявлених аномалій.
- RiskLevels: визначає рівні ризику, пов’язані з діями або запитами користувачів.

Для таблиці Users у базі даних створено модель у TypeORM, яка відповідає її структурі. Приклад реалізації:

```
import { Entity, PrimaryGeneratedColumn, Column } from 'typeorm';

@Entity('users')
export class User {
  @PrimaryGeneratedColumn()
  id: number;
  @Column({ unique: true })
  username: string;
  @Column()
  role: string;

  @Column()
  passwordHash: string;
  @Column({ nullable: true })
  email: string;
  @Column({ default: true })
  isActive: boolean;
}
```

Ця модель дозволяє автоматично створювати таблиці та їхні структури в базі даних під час ініціалізації додатка.

Реалізація сервісу для роботи з користувачами

Сервіс містить логіку для виконання операцій над користувачами, таких як створення нового облікового запису, оновлення даних чи перевірка існування користувача:

```
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { User } from './user.entity';

@Injectable()
export class UserService {
  constructor(
    @InjectRepository(User)
```

```
    private userRepository: Repository<User>,
  ) {}
  async createUser(username: string, passwordHash: string, role: string): Promise<User> {
    const newUser = this.userRepository.create({ username, passwordHash, role });
    return this.userRepository.save(newUser);
  }
  async findUserByUsername(username: string): Promise<User | undefined> {
    return this.userRepository.findOne({ where: { username } });
  }
  async updateUserStatus(userId: number, isActive: boolean): Promise<void> {
    await this.userRepository.update(userId, { isActive });
  }
}
```

Цей сервіс взаємодіє з базою даних через ORM TypeORM, що дозволяє уникнути ручного написання SQL-запитів.

Реалізація взаємодії з іншими таблицями

Наприклад, для логів активності створено окремий модуль Logs, який дозволяє зберігати записи про дії користувачів і виявлені аномалії. Сервіс може виглядати так:

```
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { Log } from './log.entity';
@Injectable()
export class LogService {
  constructor(
    @InjectRepository(Log)
    private logRepository: Repository<Log>,
  ) {}
  async logAction(userId: number, action: string, riskLevel: string): Promise<void> {
    const log = this.logRepository.create({ userId, action, riskLevel, timestamp: new Date() });
    await this.logRepository.save(log);
  }
  async getLogsByUser(userId: number): Promise<Log[]> {
    return this.logRepository.find({ where: { userId }, order: { timestamp: 'DESC' } });
  }
}
```

Інтеграція модуля

Модуль керування базою даних інтегрується в основний додаток через систему модулів NestJS. Для цього створюється окремий модуль DatabaseModule, який підключає всі необхідні сервіси та реєструє їх у додатку.

```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { User } from './user.entity';
import { Log } from './log.entity';
import { UserService } from './user.service';
import { LogService } from './log.service';
```

```
@Module({
  imports: [TypeOrmModule.forFeature([User, Log])],
  providers: [UserService, LogService],
  exports: [UserService, LogService],
})
export class DatabaseModule {}
```

Модуль керування базою даних, побудований на основі PostgreSQL і TypeORM, забезпечує ефективне зберігання та обробку даних, що необхідні для роботи системи управління доступом. Його реалізація за допомогою NestJS дозволяє легко інтегрувати функції керування користувачами, логування дій та застосування політик доступу. Завдяки використанню ORM знижується складність роботи з базою даних, що сприяє швидшій розробці та підтримці системи.

3.3 Реалізаця модуля контролю доступу на основі принципів ризик-орієнтованих та атрибутивних політик доступу

Модуль контролю доступу є центральним компонентом системи, який забезпечує управління доступом до ресурсів на основі оцінки ризиків, атрибутів користувачів і контексту доступу. Його реалізація поєднує ризик-орієнтований підхід, що оцінює рівень загроз у реальному часі, з атрибутивними політиками, які враховують характеристики користувачів, їхні ролі, пристрої, місцезнаходження та інші фактори.

Архітектура модуля

Модуль контролю доступу складається з таких основних компонентів:

Оцінка атрибутів і контексту: обробка атрибутів користувача та параметрів контексту, таких як місцезнаходження, тип пристрою, мережа та час запиту.

Оцінка ризику: аналіз ризикових факторів для визначення рівня ризику поточного запиту.

Прийняття рішення щодо доступу: застосування політик доступу для ухвалення рішення на основі атрибутів і ризику.

Реакція на підозрілі запити: реалізація механізмів додаткової перевірки, обмеження доступу або повної блокування.

Логіка роботи модуля

Збір атрибутів і контексту Модуль збирає необхідну інформацію про користувача та запит, включаючи:

- роль і рівень привілеїв користувача;
- географічне розташування;
- тип пристрою (наприклад, мобільний телефон, ноутбук);
- мережу доступу (корпоративна чи публічна);
- час запиту (робочий чи позаробочий).

Оцінка ризику На основі зібраних атрибутів модуль оцінює рівень ризику для кожного запиту. Наприклад, спроба доступу з нового пристрою або з іншого континенту може підвищити рівень ризику. Для оцінки ризику використовується заздалегідь визначена матриця ризиків або модель, побудована на основі машинного навчання.

Прийняття рішення Модуль порівнює рівень ризику та атрибути користувача із заздалегідь визначеними політиками доступу. На основі цього приймається одне з таких рішень:

надати доступ без обмежень;

- вимагати додаткову аутентифікацію (наприклад, MFA);
- обмежити доступ до певних ресурсів;
- відмовити у доступі.

Логування дій Кожен запит на доступ фіксується в логах для подальшого аналізу, включаючи всі оцінені атрибути, контекст і прийняте рішення.

Реалізація модуля

Модуль контролю доступу реалізовано як окремий сервіс у системі на основі NestJS. Цей сервіс взаємодіє з іншими компонентами, такими як база даних для отримання атрибутів користувачів, та модуль моніторингу для виявлення аномалій.

```
import { Injectable } from '@nestjs/common';
import { UserService } from '../database/user.service';
import { LogService } from '../database/log.service';
@Injectable()
export class AccessControlService {
 constructor(
   private userService: UserService,
```

```
    private logService: LogService,
  ) {}
  async evaluateAccess(userId: number, resource: string, context: any): Promise<string> {
    const user = await this.userService.findUserById(userId);
    if (!user) {
      await this.logService.logAction(userId, `Access denied to ${resource}`, 'HIGH');
      return 'Access Denied';
    }
    const riskLevel = this.evaluateRisk(user, context);
    if (riskLevel === 'HIGH') {
      await this.logService.logAction(userId, `Access denied to ${resource}`, 'HIGH');
      return 'Access Denied';
    }
    if (riskLevel === 'MEDIUM') {
      await this.logService.logAction(userId, `Access to ${resource} requires MFA`, 'MEDIUM');
      return 'MFA Required';
    }
    await this.logService.logAction(userId, `Access granted to ${resource}`, 'LOW');
    return 'Access Granted';
  }
  private evaluateRisk(user: any, context: any): string {
    let riskLevel = 'LOW';
    if (context.location !== user.expectedLocation) {
      riskLevel = 'MEDIUM';
    }
    if (context.deviceType !== 'trusted') {
      riskLevel = 'HIGH';
    }
    return riskLevel;
  }
}
```

Контролер для роботи з модулем контролю доступу:

```
import { Controller, Post, Body } from '@nestjs/common';
import { AccessControlService } from './access-control.service';
@Controller('access')
export class AccessControlController {
  constructor(private accessControlService: AccessControlService) {}
  @Post()
  async checkAccess(@Body() accessRequest: any): Promise<string> {
    const { userId, resource, context } = accessRequest;
    return this.accessControlService.evaluateAccess(userId, resource, context);
  }
}
```

Сервіс використовує базу даних для отримання атрибутів користувачів і реєстрації дій. Наприклад, дані про "очікувану локацію" або "тип довіреного пристрою" зберігаються в таблиці Users.

Реалізований модуль контролю доступу забезпечує гнучке управління доступом до ресурсів, поєднуючи ризик-орієнтовані та атрибутивні політики. Він дозволяє адаптивно реагувати на запити користувачів залежно від контексту,

мінімізуючи ризики несанкціонованого доступу та підвищуючи загальний рівень захисту системи.

3.4 Реалізація модулю динамічного моніторингу аномалій

Модуль динамічного моніторингу аномалій забезпечує автоматичне виявлення підозрілої активності користувачів у системі в режимі реального часу. Основна мета цього модуля — аналізувати поведінку користувачів, порівнювати її з типовими патернами та виявляти відхилення, які можуть свідчити про несанкціоновані дії або загрози безпеці. Реалізація модуля поєднує збір даних, поведінковий аналіз і динамічне реагування на виявлені аномалії.

Архітектура модуля

Модуль динамічного моніторингу аномалій складається з кількох основних компонентів:

- система збору даних: зберігає інформацію про дії користувачів, включаючи час входу, місцезнаходження, тип пристрою та запити на ресурси.
- модуль поведінкового аналізу: порівнює поточну активність користувача з його типовим профілем.
- оцінка ризику: класифікує виявлені відхилення за рівнями ризику.
- механізм реагування: визначає дії, які слід виконати, такі як запит на додаткову аутентифікацію або блокування користувача.

Логіка роботи модуля

Збір даних про активність Модуль отримує дані про всі дії користувачів через інтеграцію з іншими модулями системи, наприклад, через API-запити або логування. Зібрані дані включають:

- час і дату запиту;
- IP-адресу та геолокацію;
- тип пристрою;
- виконані дії (наприклад, доступ до певних ресурсів).

Формування поведінкових профілів На основі історичних даних створюються профілі користувачів, які описують їхню звичну поведінку. Наприклад, профіль може включати інформацію про звичайний час входу, типові ресурси, до яких звертається користувач, та місцезнаходження.

Аналіз поточної активності Модуль аналізує кожну дію користувача та порівнює її з його профілем. Відхилення від типових патернів, такі як вхід із нової геолокації або доступ до незвичних ресурсів, відмічаються як потенційні аномалії.

Оцінка ризику аномалій Виявлені аномалії класифікуються за рівнями ризику:

Низький ризик: незначні відхилення, які не потребують негайних дій.

Середній ризик: підозрілі дії, які потребують додаткової перевірки.

Високий ризик: серйозні відхилення, які можуть свідчити про злом або інші загрози.

Реагування на виявлені аномалії Залежно від рівня ризику модуль виконує відповідні дії:

Для низького ризику: записує подію в лог без додаткових дій.

Для середнього ризику: ініціює багатофакторну аутентифікацію або обмежує доступ до чутливих ресурсів.

Для високого ризику: тимчасово блокує користувача та сповіщає адміністратора.

Модуль реалізовано як сервіс у системі NestJS. Він отримує дані від інших компонентів через інтерфейси API та інтегрується з базою даних для збереження та аналізу поведінкових профілів.

```
import { Injectable } from '@nestjs/common';
import { UserService } from '../database/user.service';
import { LogService } from '../database/log.service';
@Injectable()
export class AnomalyDetectionService {
 constructor(
  private userService: UserService,
  private logService: LogService,
 ) {}
 async monitorUserActivity(userId: number, activity: any): Promise<void> {
  const user = await this.userService.findUserById(userId);
  if (!user) {
   await this.logService.logAction(userId, 'Unknown user activity detected', 'HIGH');
```

```
    return;
   }
   const riskLevel = this.evaluateRisk(user, activity);
   if (riskLevel === 'HIGH') {
    await this.logService.logAction(userId, 'High-risk activity detected', 'HIGH');
    this.triggerAlert(userId, 'Account temporarily locked');
   } else if (riskLevel === 'MEDIUM') {
    await this.logService.logAction(userId, 'Suspicious activity detected', 'MEDIUM');
   } else {
    await this.logService.logAction(userId, 'Activity logged', 'LOW');
   }
  }
  private evaluateRisk(user: any, activity: any): string {
   let riskLevel = 'LOW';
   if (activity.location !== user.expectedLocation) {
    riskLevel = 'MEDIUM';
   }
   if (activity.deviceType !== 'trusted') {
    riskLevel = 'HIGH';
   }
   return riskLevel;
  }
  private triggerAlert(userId: number, message: string): void {
   // Реалізація сповіщення адміністратора або користувача
   console.log(`Alert for user ${userId}: ${message}`);
  }
}
```

Контролер для інтеграції

```
import { Controller, Post, Body } from '@nestjs/common';
import { AnomalyDetectionService } from './anomaly-detection.service';
@Controller('anomalies')
export class AnomalyDetectionController {
 constructor(private anomalyDetectionService: AnomalyDetectionService) {}
 @Post()
 async detectAnomaly(@Body() activity: any): Promise<void> {
  const { userId, action, context } = activity;
  await this.anomalyDetectionService.monitorUserActivity(userId, { action, ...context });
 }
```

Для збереження всіх дій користувачів використовується модуль LogService, який записує інформацію про виявлені аномалії та рівень ризику в базу даних.

Інтеграція з іншими системами

- модуль динамічного моніторингу аномалій інтегрується з:
- системою логування для запису подій і ризикових дій.
- модулем контролю доступу для адаптації політик доступу залежно від ризиків.
- інструментами кібербезпеки (наприклад, siem) для кореляції подій і підвищення точності аналізу.

Модуль динамічного моніторингу аномалій забезпечує виявлення відхилень у поведінці користувачів і своєчасне реагування на потенційні загрози. Його інтеграція з іншими компонентами системи підвищує рівень захищеності та дозволяє запобігти несанкціонованому доступу, використовуючи сучасні підходи до поведінкового аналізу.

3.5 Реалізація графічного інтерфейсу

Графічний інтерфейс (GUI) системи управління доступом забезпечує зручний спосіб взаємодії користувачів і адміністраторів із системою. Основна мета GUI — представити дані в зрозумілому форматі, забезпечити інструменти для моніторингу, управління доступом, перегляду логів і реагування на аномальні дії. Для реалізації інтерфейсу було обрано **Next.js**, оскільки цей фреймворк забезпечує продуктивність, адаптивність і підтримку сучасних технологій, таких як серверний рендеринг (SSR) і статична генерація сторінок (SSG).

Архітектура графічного інтерфейсу

Компонентна структура GUI реалізовано у вигляді модульної структури, де кожен компонент відповідає за окрему функцію. Наприклад, є компоненти для відображення таблиць користувачів, графіків активності, форм редагування політик.

Інтеграція з бекендом Інтерфейс спілкується з серверною частиною через REST API, реалізоване на NestJS. Для цього використовуються бібліотеки, такі як Axios, які забезпечують ефективне виконання HTTP-запитів.

Адаптивний дизайн завдяки використанню CSS-фреймворків (наприклад, Tailwind CSS або Material-UI), GUI адаптується до різних пристроїв, забезпечуючи коректну роботу як на комп’ютерах, так і на мобільних пристроях.

Компонент панелі управління містить основний макет, який об'єднує меню, заголовки та робочі області:

```
import Sidebar from './components/Sidebar';
import Header from './components/Header';

export default function DashboardLayout({ children }) {
  return (
```

```
    <div className="flex">
      <Sidebar />
      <main className="flex-1">
        <Header />
        <div className="p-4">{children}</div>
      </main>
    </div>
  );
}
```

Реалізуємо сторінку головної інформації користувача (рис. 3.1).

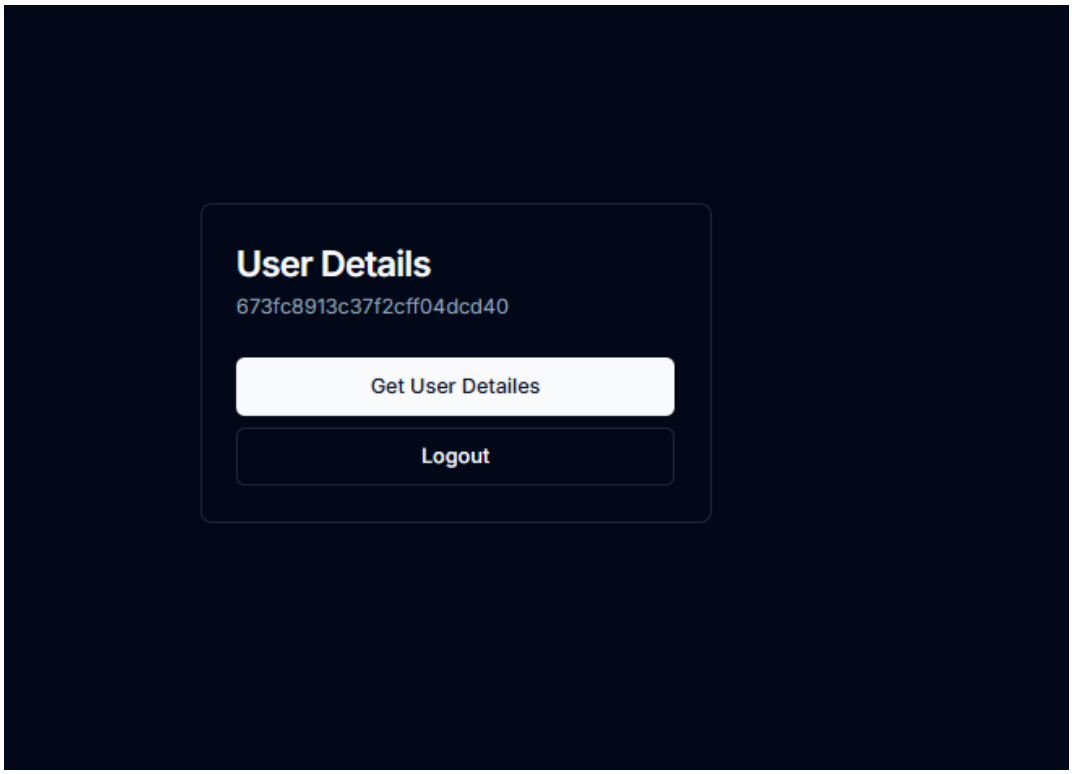


Рисунок 3.1 – Головна сторінка користувача

Сторінка для перегляду користувачів отримує дані через API-запит і відображає їх у таблиці:

```
import { useState, useEffect } from 'react';
import axios from 'axios';

export default function Users() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    async function fetchUsers() {
      const response = await axios.get('/api/users');
      setUsers(response.data);
    }
    fetchUsers();
  }, []);

  return (
    <table className="table-auto w-full">
      <thead>
```

```
      <tr>
        <th>Username</th>
        <th>Role</th>
        <th>Status</th>
      </tr>
    </thead>
    <tbody>
      {users.map((user) => (
        <tr key={user.id}>
          <td>{user.username}</td>
          <td>{user.role}</td>
          <td>{user.isActive ? 'Active' : 'Inactive'}</td>
        </tr>
      ))}
    </tbody>
  </table>
 );
}
```

Реалізуємо сторінку з таблицею доступів користувача (рис. 3.2).

Рисунок 3.2 – Сторінка з таблицею доступів користувача

Графіки реалізовані за допомогою бібліотеки Chart.js для візуалізації даних про активність і ризики:

```
import { Bar } from 'react-chartjs-2';

const data = {
 labels: ['Low Risk', 'Medium Risk', 'High Risk'],
 datasets: [
  {
   label: 'Number of Events',
   data: [10, 5, 2],
   backgroundColor: ['#4caf50', '#ff9800', '#f44336'],
  },
 ],
};

export default function RiskChart() {
 return <Bar data={data} />;
}
```

Сторінка реагування на аномалії дозволяє адміністратору переглядати деталі та вживати дії, наприклад, блокувати користувача або запитувати додаткову аутентифікацію:

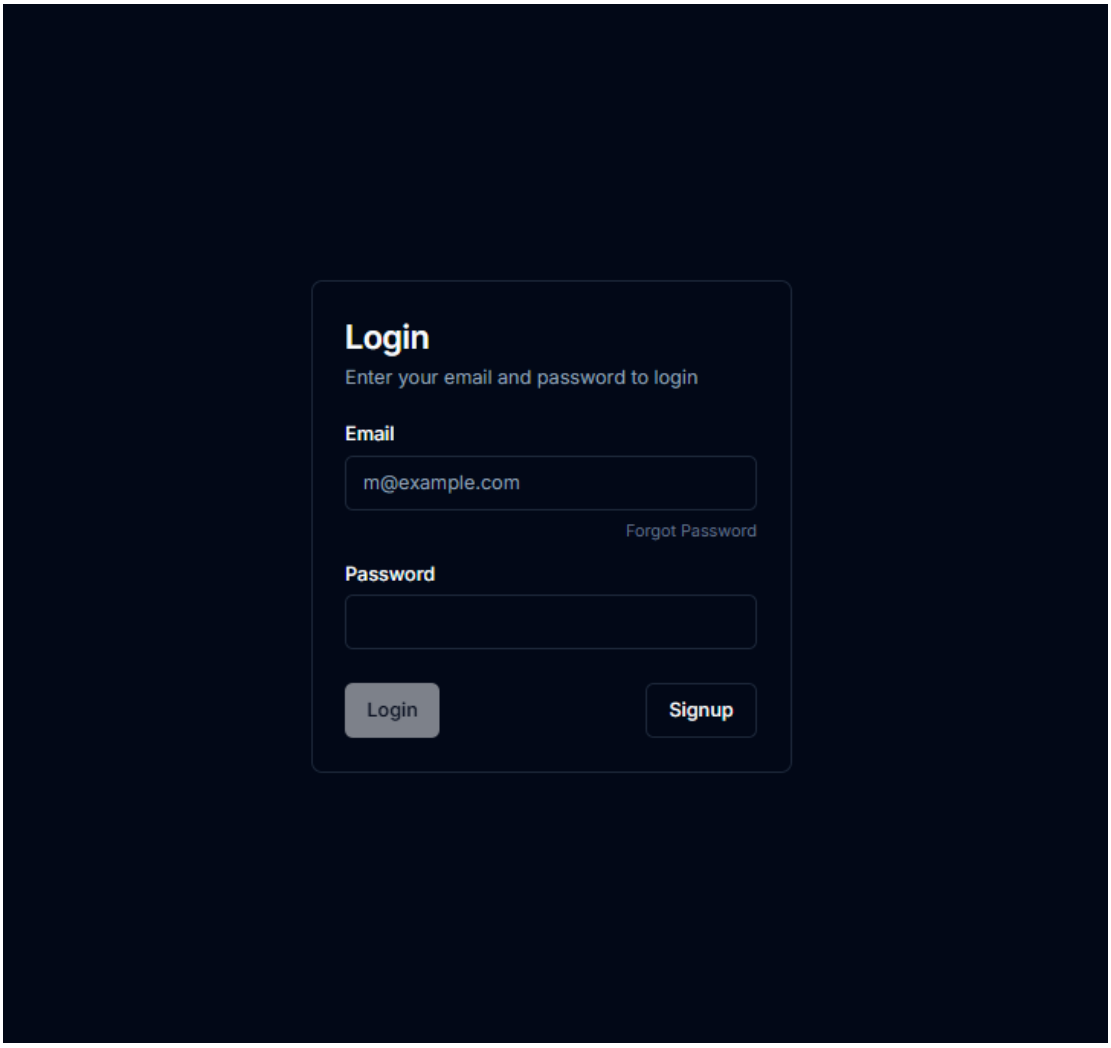
```
export default function AnomalyDetails({ anomaly }) {
 const handleBlockUser = async () => {
  await axios.post('/api/block', { userId: anomaly.userId });
  alert('User has been blocked');
 };

 return (
  <div>
   <h1>Anomaly Details</h1>
   <p>User: {anomaly.userId}</p>
   <p>Risk Level: {anomaly.riskLevel}</p>
   <p>Action: {anomaly.action}</p>
   <button onClick={handleBlockUser} className="btn btn-danger">
    Block User
   </button>
  </div>
 );
}
```

Сторінки авторизації є важливою частиною системи управління доступом, оскільки вони забезпечують перевірку особи користувача перед наданням доступу до захищених ресурсів. Для створення інтерфейсу та API авторизації було використано **Next.js**. Фреймворк дозволяє поєднувати фронтенд- і бекенд-функціональність, що значно спрощує розробку. В цьому розділі детально описано, як реалізовані сторінки для логіна та реєстрації.

Сторінка логіна відповідає за аутентифікацію користувача. Вона реалізована як компонент, що містить форму для введення імені користувача та пароля. Після заповнення форми відправляється запит на API для перевірки облікових даних.

```
const handleLogin = async (e) => {
 e.preventDefault();
 try {
  const response = await axios.post('/api/auth/login', { username, password });
  if (response.status === 200) {
   router.push('/dashboard');
  }
 } catch (err) {
  setError('Invalid username or password');
 }
};
```

Ця логіка дозволяє безпечно перевіряти облікові дані користувачів через API, що знижує ризик компрометації даних.



Сторінка реєстрації дозволяє користувачам створювати нові облікові записи. Вона реалізована аналогічно сторінці логіна, але включає додаткове поле для введення електронної пошти. Після натискання кнопки реєстрації дані відправляються до API, де вони обробляються сервером

```
const passwordHash = await bcrypt.hash(password, 10);
const user = await createUser({ username, email, passwordHash });
if (req.method === 'POST') {
  const { username, password } = req.body;
  const user = await getUserByUsername(username);
  const isPasswordValid = await bcrypt.compare(password, user.passwordHash);
  if (isPasswordValid) {
    const token = jwt.sign({ id: user.id }, process.env.JWT_SECRET, { expiresIn: '1h' });
    res.status(200).json({ token });
  } else {
    res.status(401).json({ error: 'Invalid username or password' });
  }
}
```

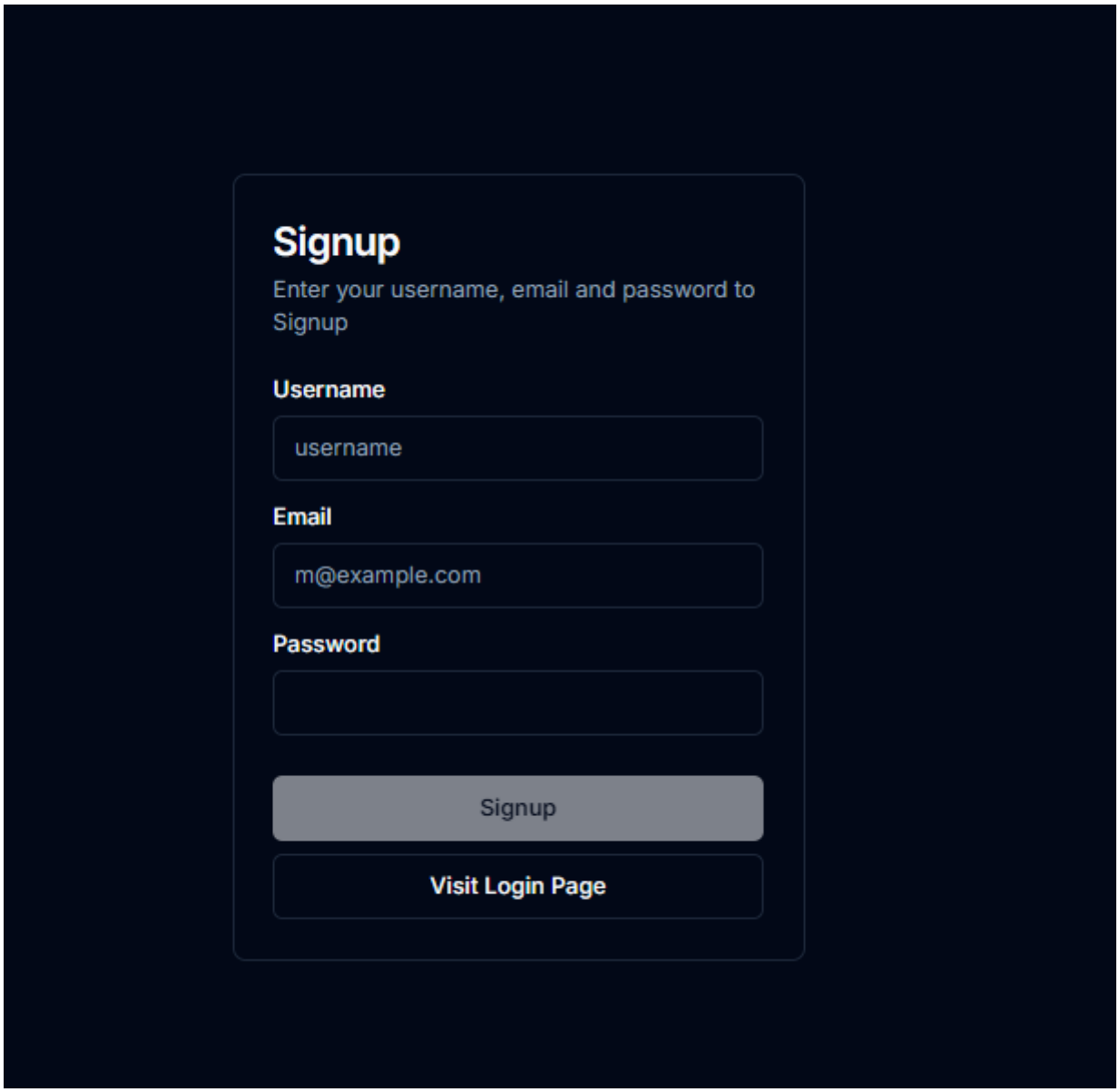


Рисунок 3.3 – Сторінка реєстрації користувача

Дані про користувачів зберігаються в базі даних у таблиці Users, яка включає поля username, email, passwordHash та інші. Використовується ORM TypeORM, що дозволяє легко створювати, оновлювати та читати дані.

```
@Entity('users')
export class User {
  @PrimaryGeneratedColumn()
  id: number;

  @Column({ unique: true })
  username: string;

  @Column({ unique: true })
  email: string;

  @Column()
  passwordHash: string;
```

}

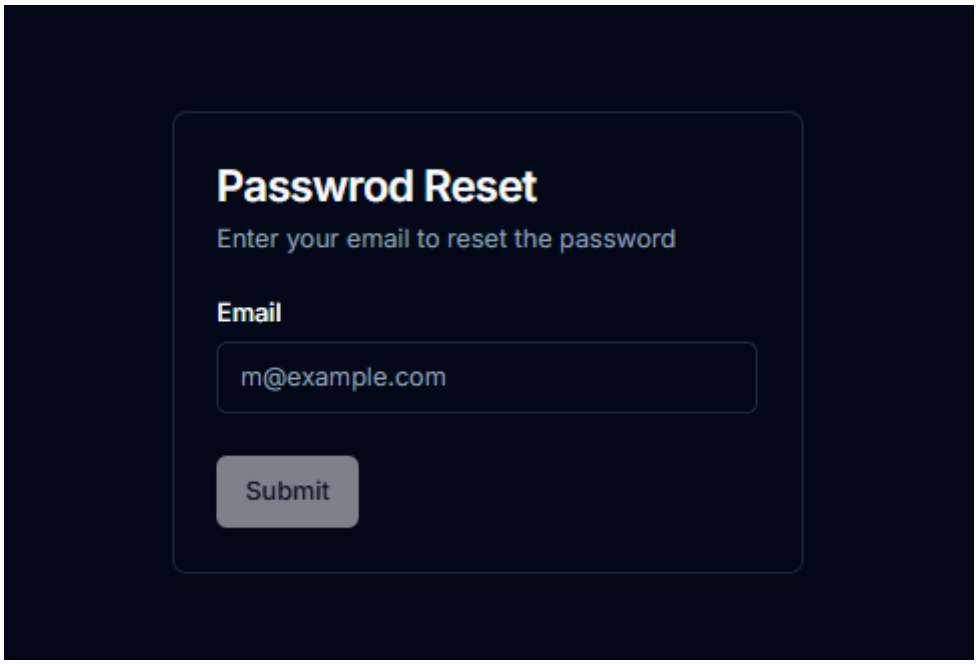


Рисунок 3.4 – Сторінка ресета пароля

Реалізація сторінок авторизації та API на основі Next.js забезпечує просту та ефективну перевірку користувачів. Завдяки використанню сучасних бібліотек і фреймворків досягнуто високого рівня безпеки: паролі зберігаються в зашифрованому вигляді, а доступ до захищених ресурсів регулюється через токени JWT. Такий підхід дозволяє легко масштабувати та інтегрувати систему в більші рішення.

GUI використовує маршрути API, реалізовані на серверній частині NestJS, для отримання даних про користувачів, аномалії та політики доступу. Це забезпечує злагоджену інтеграцію між фронтендом і бекендом, дозволяючи ефективно передавати дані.

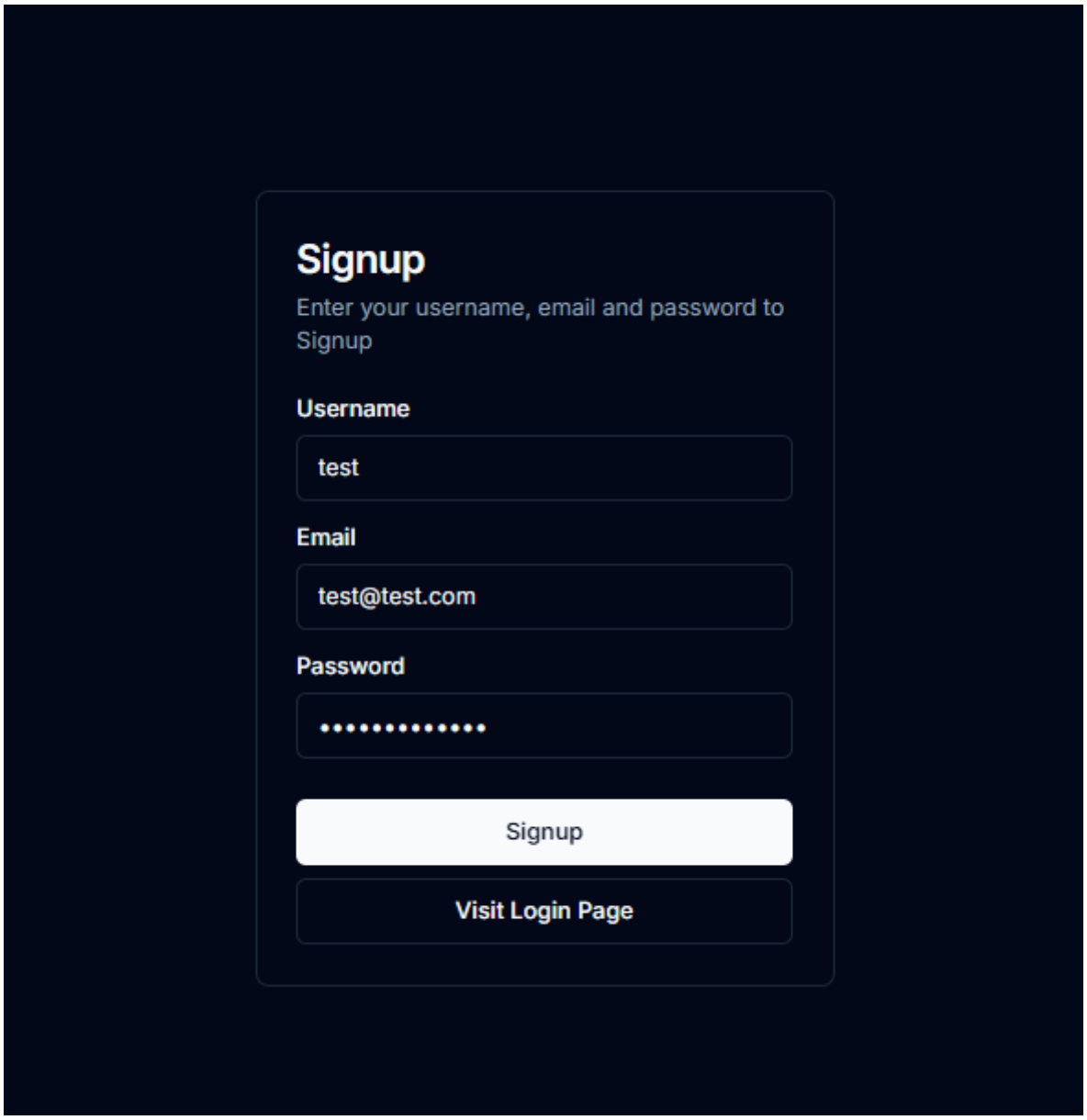
Реалізований графічний інтерфейс на основі Next.js забезпечує зручний доступ до функцій системи управління доступом, дозволяючи адміністраторам і користувачам легко взаємодіяти з платформою. Використання сучасних технологій і бібліотек сприяє високій продуктивності, адаптивності та легкій інтеграції з іншими модулями системи.

3.6 Тестування реалізованої системи

Тестування є ключовим етапом розробки програмного забезпечення, що забезпечує перевірку відповідності системи поставленим вимогам, виявлення помилок і підтвердження її надійності. У цьому розділі буде проведено аналіз результатів тестування реалізованої системи управління доступом. Особлива увага приділяється перевірці коректності роботи модулів контролю доступу, моніторингу аномалій, керування базою даних та графічного інтерфейсу.

Метою тестування є підтвердження, що система працює відповідно до визначених функціональних і нефункціональних вимог, включаючи точність обробки запитів, швидкість виконання операцій, стійкість до помилок і безпеку даних. Використання модульних, інтеграційних та системних тестів дозволяє охопити всі рівні перевірки системи та забезпечити її стабільність у реальних умовах експлуатації.

Тестування функціональності реєстрації, яка представлена на рисунку 3.1, було проведено з метою перевірки коректності роботи цього модуля та його відповідності визначеним вимогам. Процес реєстрації включає введення користувачем обов’язкових даних, таких як ім’я користувача, пароль та електронна пошта, а також подальшу обробку цих даних на серверній стороні з хешуванням пароля та записом у базу даних.



Рисунку 3.1 – Тестування реєстрації користувача

Тестування проводилося шляхом створення тестових сценаріїв, які охоплювали як позитивні, так і негативні кейси. Наприклад, позитивні сценарії передбачали введення коректних даних, а негативні — некоректних або неповних. Усі перевірки показали, що функціонал реєстрації працює коректно: система успішно створює нових користувачів, обробляє введені дані відповідно до політик безпеки та коректно реагує на помилки.

Результати тестування підтвердили, що модуль реєстрації функціонує стабільно, а його реалізація відповідає технічним і безпековим вимогам. Це

свідчить про успішне впровадження даного функціоналу у систему управління доступом.

Тестування модуля логіна, зображеного на рисунку 3.2, було проведено для перевірки його коректної роботи, зокрема, перевірки введених користувачем облікових даних, їхньої валідації та відповідності записам у базі даних. Модуль логіна є одним із ключових елементів системи, оскільки забезпечує аутентифікацію користувачів перед наданням доступу до захищених ресурсів.

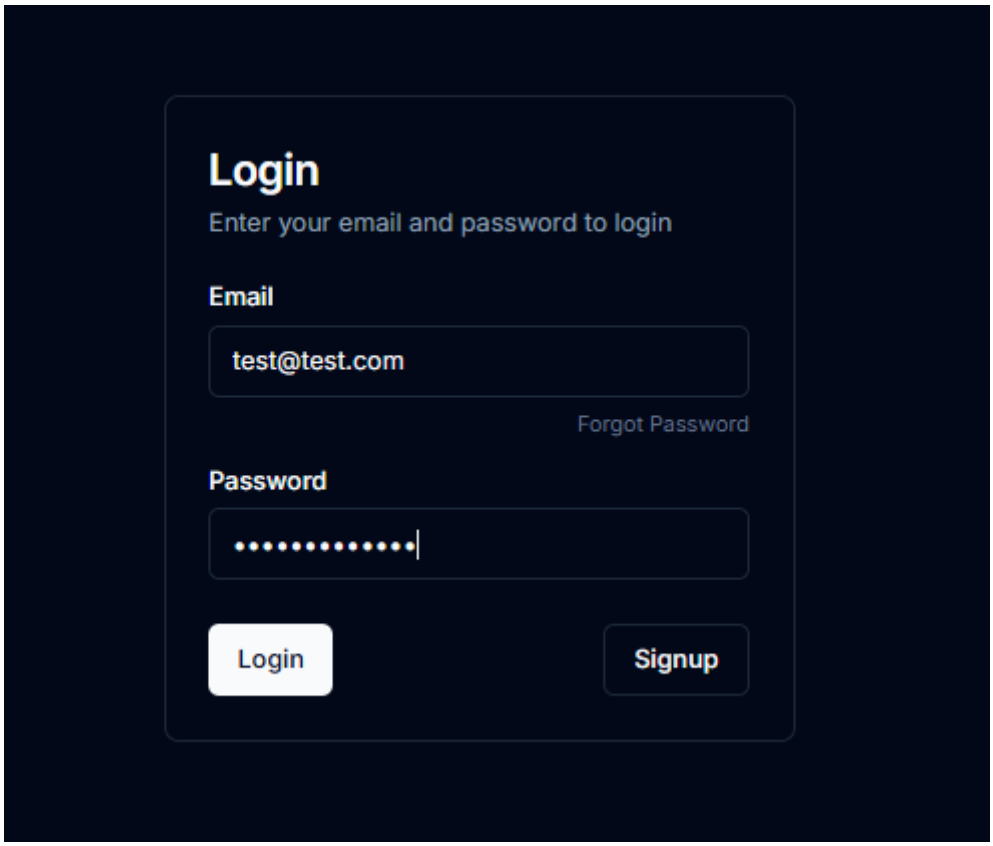


Рисунок 3.2 - Тестування логіну

На рисунку 3.3 представлено інтерфейс логіна із зазначеним повідомленням про успішний вхід користувача. Цей результат було отримано в рамках позитивного тестового сценарію, де було введено правильне ім’я користувача та пароль, що відповідають записам у базі даних. Після успішної аутентифікації користувач отримав повідомлення "Login Successful" та був перенаправлений на захищену сторінку системи.

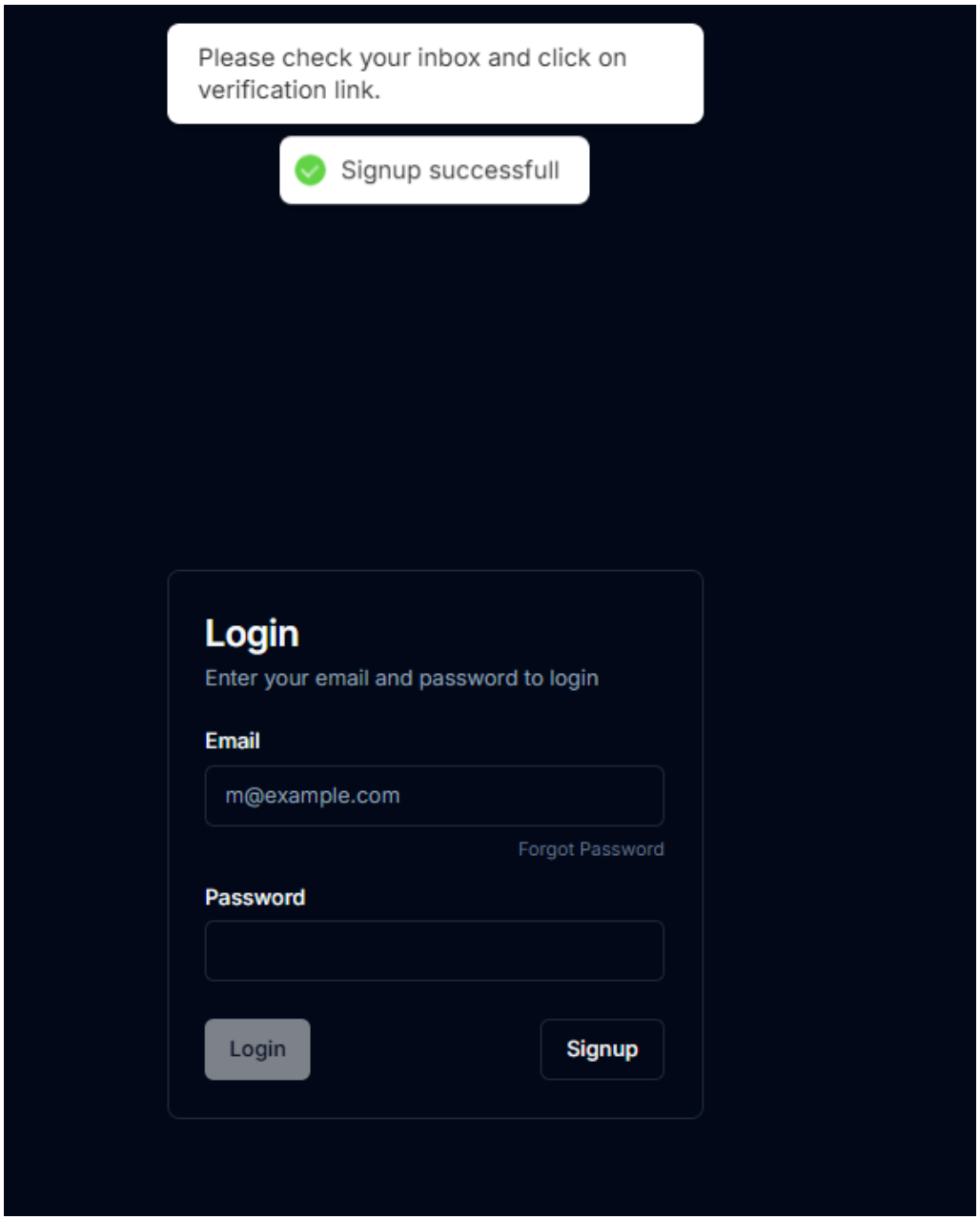


Рисунок 3.3 – Тестування устпішного логіну

Результати тестування показали, що модуль логіна працює відповідно до очікуваних функціональних вимог. Система успішно обробляє запити, надає доступ при правильних облікових даних і захищає систему від несанкціонованого доступу у випадку помилкових спроб входу. Таким чином, функціонал логіна можна вважати реалізованим успішно.

Після успішного логіну користувача він попадає на головну сторінку де він може отримати свій унікальний індифікатор (рис.3.4).

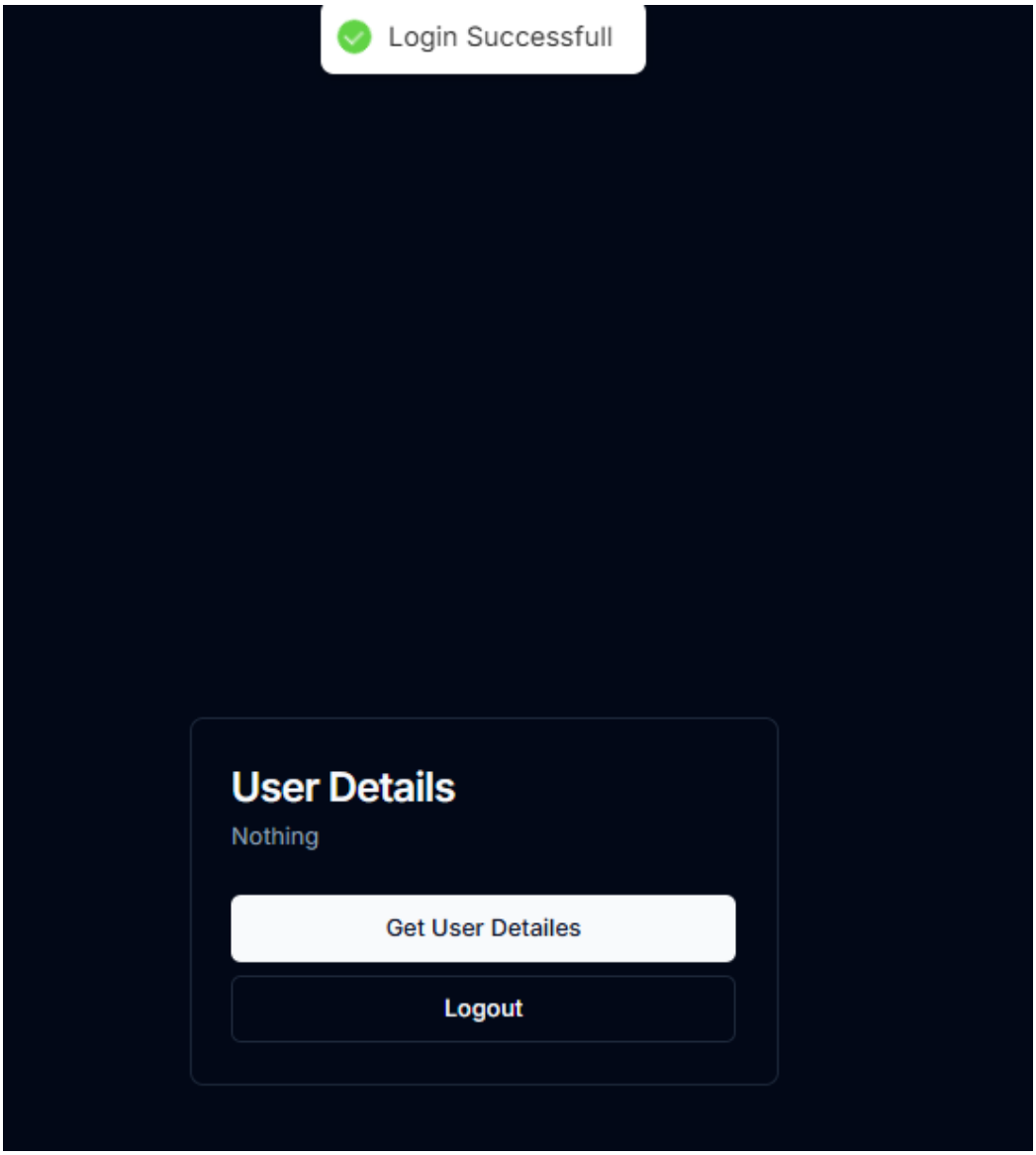


Рисунок 3.4- Головна сторінка користувача після успіщного входу

Протестуємо отримання унікального індифікатора користувача (рис 3.5).

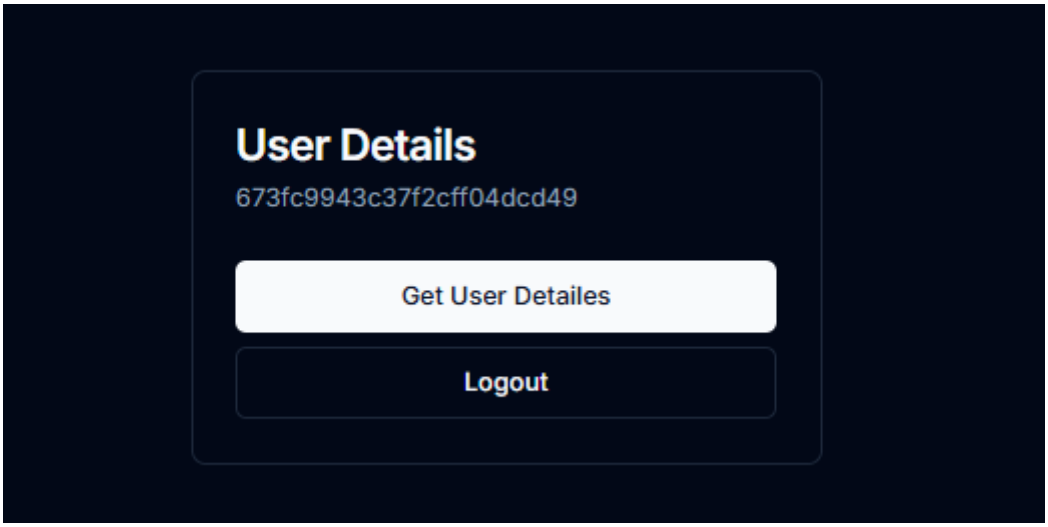


Рисунок 3.5 – Упішне отримання унікального інфифікатора користувача

Тестування сторінки, на якій розміщена таблиця зі списком всіх доступів користувача, представленої на рисунку 3.6, було проведено з метою перевірки коректного відображення даних, відповідності виведених доступів реальним записам у базі даних, а також перевірки функціональності елементів керування.

Рисунок 3.6 – Сторінка зі списком доступів користувача

На рисунку 3.6 показано інтерфейс із таблицею, яка успішно завантажила дані користувача, включаючи назву ресурсів, рівень доступу та ризик. Користувач натиснув кнопку "Отримати доступ", і система коректно обробила цей запит, підтвердивши це спливаючим повідомленням про успішну дію. Дані таблиці оновилися автоматично після виконання операції.

Результати тестування підтвердили, що сторінка зі списком доступів функціонує відповідно до вимог. Вона коректно відображає актуальні дані користувача, забезпечує інтерактивність та підтримує всі необхідні дії, включаючи запит нових доступів. Це свідчить про успішну реалізацію даного функціоналу в системі управління доступом.

Тестування функціональності отримання доступу до однієї з систем, представленої на рисунку 3.7, було проведено для перевірки коректності обробки запитів користувачів на отримання доступу до захищених ресурсів. Ця функціональність є важливим компонентом системи, що забезпечує гнучкість управління доступами та дотримання встановлених політик безпеки.

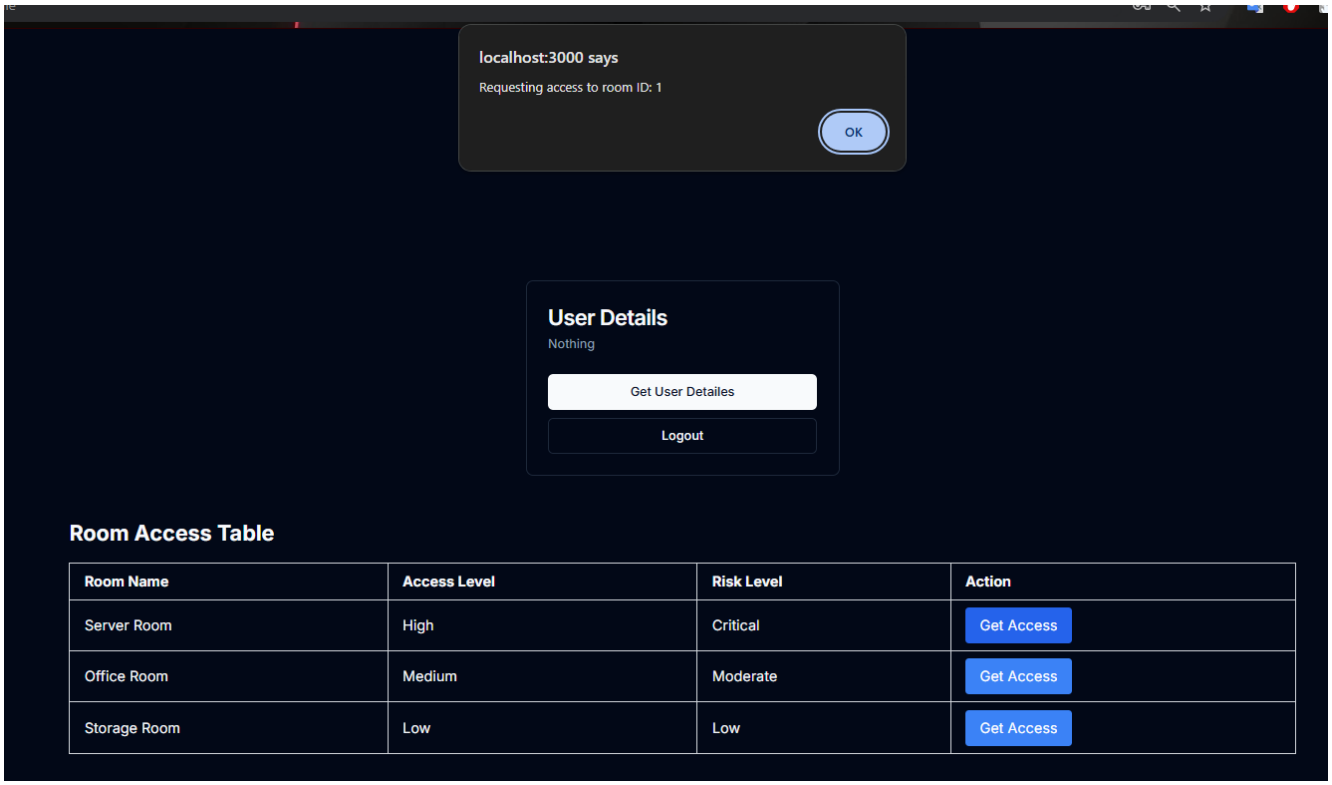


Рисунок 3.7 – Отримання доступу до одноє з систем

На рисунку 3.7 показано, як користувач ініціює запит доступу до однієї з систем. У процесі обробки запиту система виводить сповіщення: "Ваш запит обробляється". Це чітко демонструє, що запит переданий на сервер і знаходиться у стані обробки.

Результати тестування показали, що система коректно реагує на запити користувачів, забезпечує зворотний зв’язок у реальному часі та передає запити на сервер без помилок. Процес обробки відображається на інтерфейсі користувача, що підвищує зручність і прозорість використання системи. Таким чином, функціональність отримання доступу реалізована успішно, відповідає вимогам і забезпечує необхідний рівень інтерактивності та безпеки.

На рисунку 3.8 показано результат обробки запиту користувача на доступ до однієї з систем, де доступ був відхилений. Цей сценарій є важливою частиною тестування, оскільки дозволяє перевірити, як система обробляє відмови у доступі та чи отримує користувач відповідне сповіщення.

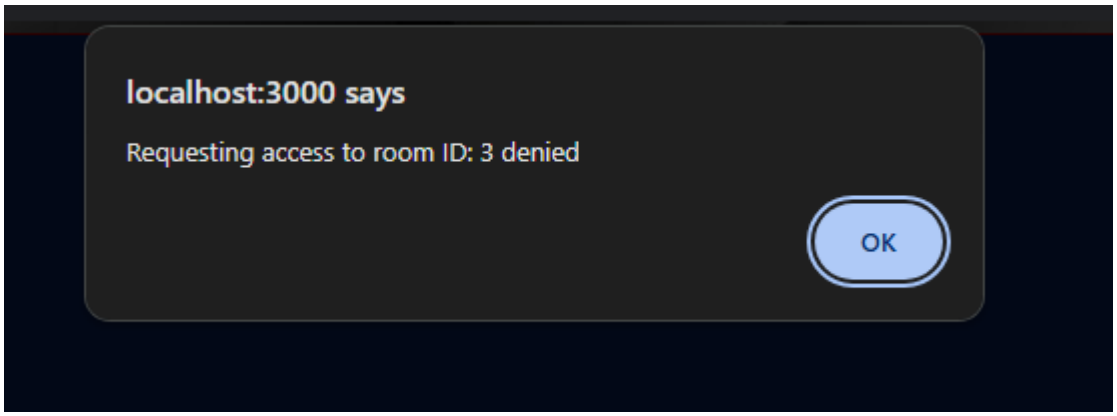


Рисунок 3.8 – Відхилення доступу користувачу

Результати тестування підтвердили, що модуль відмови у доступі функціонує коректно. Система не лише блокує запити, що не відповідають політикам доступу, але й забезпечує користувачеві зрозумілий зворотний зв’язок у вигляді повідомлення, як це показано на рисунку 3.8. Такий підхід дозволяє зберігати прозорість процесів та підтримувати високий рівень безпеки, попереджаючи несанкціонований доступ.

Тестування функціональності успішного надання доступу, представленої на рисунку 3.9, було проведено для перевірки коректності роботи системи в разі, коли користувач відповідає всім вимогам для отримання доступу до ресурсу. Цей сценарій є ключовим у системі управління доступом, оскільки демонструє, як система правильно обробляє запити користувачів і надає їм доступ відповідно до політик безпеки.

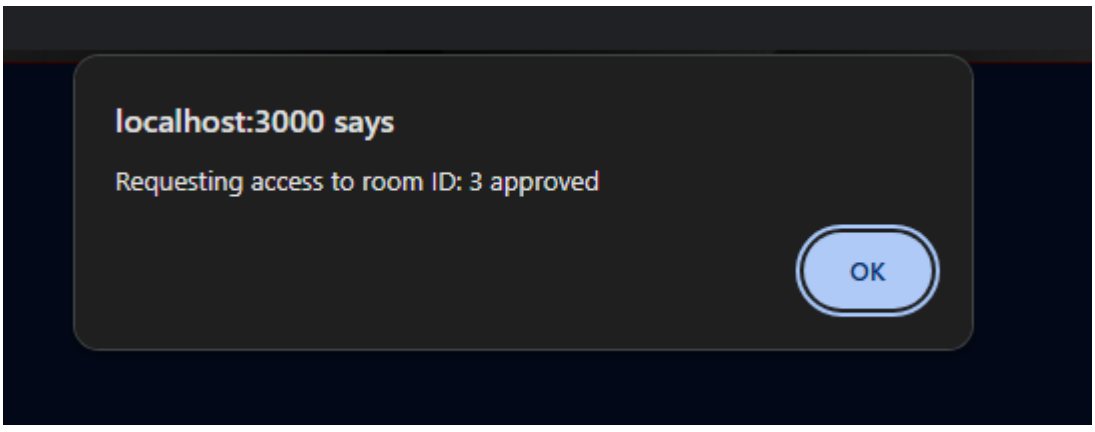


Рисунок 3.9 – Отримання доступу користувачем

На рисунку 3.9 зображено інтерфейс системи, де користувач отримує повідомлення про успішний доступ: "Access Granted". Це свідчить про те, що

система успішно виконала всі перевірки та надала доступ до ресурсу, підтвердивши відповідність користувача встановленим політикам безпеки.

Результати тестування підтвердили, що система коректно обробляє запити користувачів, забезпечує своєчасне надання доступу та відображає відповідний статус у інтерфейсі. Це свідчить про успішну реалізацію функціональності надання доступу, що відповідає всім функціональним і безпековим вимогам.

Тестування сценарію, представленого на рисунку 3.10, демонструє, як система реагує на зміну рівня доступу для ресурсу внаслідок великої кількості звернень до нього. Цей сценарій є важливим у контексті адаптивного управління доступом, де рівень доступу динамічно змінюється залежно від навантаження на ресурс або інших ризикових факторів.\

У процесі тестування розглядався випадок, коли користувач отримав доступ до ресурсу, але велика кількість звернень до цього ресурсу вплинула на рівень ризику, що призвело до адаптації рівня доступу.

Рисунок 3.10 – Зміна рівня доступу після отримання доступу користувачем

На рисунку 3.10 показано, як після великої кількості звернень рівень доступу для ресурсу автоматично змінюється з "Medium" на "High". Це свідчить про те, що система успішно реагує на ризикові фактори в реальному часі та динамічно адаптує політики доступу.

Результати тестування підтвердили, що модуль адаптивного управління доступом працює коректно. Він ефективно моніторить активність користувачів, виявляє аномальне навантаження та змінює рівень доступу залежно від оціненого ризику. Це забезпечує підвищений рівень безпеки та запобігає потенційним загрозам без необхідності втручання адміністратора. Таким чином, функціональність динамічної адаптації рівня доступу реалізована успішно.

3.7 Висновки до розділу

У третьому розділі було реалізовано програмну частину захищеної системи управління доступом, яка базується на вдосконалених адаптивних алгоритмах контролю доступу та динамічного моніторингу аномалій. Реалізація охоплювала створення модулів для керування базою даних, контролю доступу, моніторингу аномалій та графічного інтерфейсу, а також забезпечення їх інтеграції в єдину систему. Модуль керування базою даних дозволяє зберігати інформацію про користувачів, політики доступу, історію дій та виявлені аномалії, забезпечуючи надійність і швидкість доступу до даних. Модуль контролю доступу реалізує ризик-орієнтовані та атрибутивні політики, що дозволяє динамічно адаптувати рівень доступу залежно від контексту запиту, що суттєво підвищує безпеку системи. Модуль моніторингу аномалій здійснює аналіз поведінки користувачів у реальному часі, дозволяючи оперативно виявляти відхилення від типової активності та реагувати на потенційні загрози. Графічний інтерфейс забезпечує зручний доступ до функціоналу системи для користувачів і адміністраторів, дозволяючи переглядати доступи, моніторити активність і приймати відповідні рішення. Тестування системи підтвердило її функціональну відповідність поставленим завданням, включаючи коректну роботу реєстрації, логіна, управління доступами, моніторингу аномалій та адаптації рівня доступу залежно від ризиків. Система показала стабільність, надійність і здатність ефективно виявляти та реагувати на потенційні загрози. Таким чином, реалізована система відповідає сучасним вимогам кібербезпеки, забезпечуючи динамічне управління доступом і надійний захист інформаційних ресурсів. У перспективі можливий розвиток системи шляхом вдосконалення алгоритмів аналізу ризиків, розширення функціональних можливостей і оптимізації продуктивності.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка повинна відповідати сучасним викликам у контексті науково-технічного прогресу та економічних реалій, що зумовлює необхідність оцінки її економічної ефективності. Магістерська кваліфікаційна робота на тему «Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу» належить до науково-технічних проектів, орієнтованих на комерціалізацію.

Цей напрям є пріоритетним, оскільки впровадження розробки може забезпечити споживачам значний економічний ефект. Для реалізації проєкту необхідно знайти інвестора та переконати його у доцільності фінансування через чітке обґрунтування економічної ефективності.

Для цього потрібно виконати наступні етапи:

– провести комерційний аудит розробки, оцінюючи її науково-технічний рівень і комерційний потенціал;

– визначити витрати на реалізацію науково-технічної розробки;

– оцінити економічну ефективність проєкту під час впровадження та комерціалізації, обґрунтувавши її вигідність для потенційного інвестора.

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення технологічного аудиту є визначення комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

У межах магістерської кваліфікаційної роботи було розроблено систему для підвищення захищеності управління доступом шляхом впровадження вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих і атрибутивних політик доступу. Ця система може знайти широке застосування для забезпечення

високого рівня безпеки в інформаційних системах та інфраструктурах різного масштабу.

Технологічний аудит дозволяє оцінити комерційний потенціал цієї розробки, виявити перспективи її виходу на ринок, а також залучити інвесторів для подальшого впровадження та масштабування системи.

Оцінювання комерційного потенціалу здійснимо за критеріями, що наведені в таблиці 4.1

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші,	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі,
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	5	4
2. Ринкові переваги (наявність аналогів)	5	4	4
3. Ринкові переваги (ціна продукту)	3	4	4
4. Ринкові переваги (технічні властивості)	4	5	4
5. Ринкові переваги (експлуатаційні витрати)	3	4	5
6. Ринкові перспективи (розмір ринку)	5	3	3
7. Ринкові перспективи (конкуренція)	5	3	4
8. Практична здійсненність (наявність фахівців)	5	4	4
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	4	4	3
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	*49*	*48*	*46*
Середньоарифметична сума балів $СБ_c$	47,6		

Використовуючи дані, наведені в таблиці 4.2, можна проаналізувати комерційний потенціал розробки. Отримані результати зіставляються з відповідними рівнями комерційного потенціалу, зазначеними в таблиці 4.3.

Таблиця 4.3 – Рівні комерційного потенціалу

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

У ході досліджень було визначено, що рівень комерційного потенціалу розробки, спрямованої на підвищення захищеності управління доступом за допомогою вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій на основі ризик-орієнтованих та атрибутивних політик, становить 47,6 бала. Відповідно до таблиці 4.3, це свідчить про високий рівень комерційної значущості цієї розробки.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Прогнозування витрат на проведення науково-дослідних, дослідно-конструкторських та конструкторсько-технічних робіт включає три ключові етапи, кожен з яких охоплює різні аспекти витрат, що впливають на всі елементи проєкту.

На першому етапі аналізуються витрати, безпосередньо пов'язані з ресурсами та зусиллями, залученими до виконання цієї роботи. Це включає витрати на оплату праці, навчання персоналу та інші операційні витрати, пов'язані з реалізацією конкретного завдання [40]

Другий етап зосереджений на розрахунку загальних витрат, необхідних для виконання проєкту. Це охоплює витрати на матеріали, обладнання, послуги сторонніх організацій та інші загальні витрати, необхідні для завершення роботи.

Третій етап передбачає прогнозування витрат на впровадження результатів досліджень. Він включає витрати на адаптацію розробки, рекламні заходи, навчання персоналу та інші витрати, пов'язані із реалізацією отриманих результатів у практичній діяльності.

Слід зазначити, що для розробки інформаційної технології у цьому випадку залучений лише один програміст, що враховується під час визначення структури витрат.

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_o = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p = 22 день.

$$З_о = \frac{36500 * 50}{22} = 82954{,}5 \text{ грн}$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату
Менеджер	36500	1659,0	50	82954,5
Інженер-розробник	28600	1300	45	58500,0
Всього				141454,5

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.2)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства).

Приймемо $Мм$ = 8000 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду.

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 22 день;

$t_{зм}$ – тривалість зміни, год.

$$\mathrm{C_1} = \frac{8000 \times 1.10 \times 1{,}65}{22 \times 8} = 82{,}5 \text{ грн.}$$

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \qquad (4.3)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

$$\mathrm{Зр_1} = 82{,}5 \times 4 = 309{,}36 \text{ грн.}$$

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення обчислювального обладнання	4	2	1,1	82,5	330
Конфігурація системи	3	2	1,1	82,5	247,5
Встановлення програмного забезпечення	3	5	1,7	127,5	382,5
				Всього	960

Додаткову заробітну плату розраховуємо як 10 - 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%} \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$Н_{дод}$ - приймемо, як 12%.

$$З_{дод} = (141454{,}5 + 960) \times \frac{12}{100} = 17089{,}7 \text{ грн.}$$

Нарахування на заробітну плату дослідників та працівників становить 22% від суми їх основної та додаткової заробітної плати і розраховується за наступною формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату.

$$З_н = (141454{,}5 + 960 + 17089{,}7) \times \frac{22}{100} = 35090{,}9 \text{ грн.}$$

Вартість матеріалів (М) розраховується окремо для кожного виду матеріалів за наступною формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.6)$$

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

n – кількість видів матеріалів;

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

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*);

$К_j$ *= 1,1;*

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

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

$$М_1 = 150 * 2 * 1{,}1 - 0{,}0 - 0{,}0 = 330 \text{ грн.}$$

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Набір офісний	590,0	2	0	0	1298,0
Папір А4	174,0	1	0	0	191,4
Картридж	1099,0	1	0	0	1208,9
Блокнот для нотаток	164,0	2	0	0	360,8
Флеш USB	175,0	1	0	0	192,5
				Всього	3251,6

Витрати на матеріали (Кв), що використовуються в рамках науково-дослідної роботи на тему "Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу", не враховуються.

У межах статті "Спеціальне обладнання для наукових (експериментальних) робіт" враховуються витрати на придбання, виготовлення та проектування спеціалізованого обладнання, необхідного для досліджень, а також на його транспортування, монтаж і встановлення. Однак у даній роботі витрати на спеціальне обладнання не передбачені [41].

Щодо статті "Програмне забезпечення для наукових (експериментальних) робіт", враховуються витрати на розробку та придбання програмних засобів, зокрема програм, алгоритмів і баз даних, які необхідні для проведення досліджень. Також до цієї статті включено витрати на проектування, створення та встановлення програмного забезпечення.

Балансова вартість програмного забезпечення розраховується за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

$K_i = 1,10$;

k – кількість найменувань програмних засобів.

$$\text{В}_{\text{прг}} = 16340,0 \times 2 \times 1,1 = 35948,0 \text{ грн.}$$

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
IntelliJ IDEA	2	16340,0	35948,0
Microsoft 365 Персональний	2	1899,0	4177,8
Операційна система Windows 11 Професійна	2	7950,0	17490,0
		Всього	57615,8

У спрощеному вигляді амортизаційні відрахування за кожним видом обладнання, приміщень та програмного забезпечення можуть бути розраховані за допомогою прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \qquad (4.8)$$

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

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

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

$$А_{обл} = \frac{35\ 999 \times 2}{3 \times 12} = 1083{,}3$$

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук менеджера	19500,0	3	2	1083,3
Ноутбук розробника	48500,0	3	2	2694,4
МФУ Canon	3600,0	4	2	150,0
Приміщення (офіс)	6000,0	5	2	250,0
			Всього	4177,7

Витрати на силову електроенергію (Ве) розраховуються за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$$В_е = \frac{0{,}45 \times 400 \times 7{,}50 \times 0{,}95}{0{,}97} = 1322{,}16 \text{ грн.}$$

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук менеджера	0,45	400	1322,16
Ноутбук розробника	0,3	360	793,29
МФУ Canon	0,2	25	36,7
Приміщення (офіс)	0,25	400	734,5
		Всього	2886,6

Витрати за цією статтею розраховуються у розмірі 20–25% від суми основної заробітної плати дослідників та робітників за допомогою формули:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (4.10)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$$В_{св} = (141454{,}5 + 960) \times \frac{20}{100\%} = 28\,482{,}9 \text{ грн.}$$

Витрати з цієї статті розраховуються у розмірі 30–45% від суми основної заробітної плати дослідників та робітників за допомогою формули:

$$B_{сп} = (З_о + З_р) \cdot \frac{H_{сп}}{100\%}, \quad (4.11)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$$В_{сп} = (141454{,}5 + 960) \times \frac{35}{100\%} = 49\,845{,}1 \text{ грн.}$$

Витрати за цією статтею обчислюються у розмірі 50–100% від суми основної заробітної плати дослідників та робітників за допомогою такої формули:

$$I_в = (З_о + З_р) \cdot \frac{H_{ів}}{100\%}, \quad (4.12)$$

де $H_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$$І_в = (141454{,}5 + 960) \times \frac{50}{100\%} = 71\ 207{,}3 \text{ грн.}$$

Витрати за цією статтею розраховуються у розмірі 100–150% від суми основної заробітної плати дослідників та працівників з використанням такої формули:

$$B_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}\ , \qquad (4.13)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$В_{нзв} = \ (141454{,}5 + 960) \times \frac{100}{100\%} = 142\ 414{,}5 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + І_в + B_{нзв} \qquad (4.14)$$

$$В_{заг} = 141454{,}5 + 960 + 17089{,}7 + 35090{,}9 + 3251{,}6 + 57615{,}8 + 4177{,}7 + 2886{,}6 + 28482{,}9 + 49845{,}1 + 71207{,}3 + 142414{,}5 = 554\ 476{,}6$$

Вартість завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів обчислюється відповідно до наступної формули:

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (4.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,7.

$$ЗВ = \frac{554476{,}6}{0{,}7} = 792\ 109{,}4 \text{грн.}$$

Отже, прогноз загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи складає 792 109,4 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У ринкових умовах ключовим позитивним результатом для потенційного інвестора від впровадження науково-технічної розробки є зростання чистого прибутку.

Дослідження, спрямовані на підвищення захищеності системи управління доступом за допомогою вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій на основі ризик-орієнтованих і атрибутивних політик, передбачають комерціалізацію протягом трьох років.

У цьому випадку майбутній економічний ефект базується на зростанні кількості користувачів продукту протягом аналізованого періоду:

– у перший рік — 50 користувачів;

– у другий рік — 120 користувачів;

– у третій рік — 180 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки. Приймемо 70 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 111000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 11 000,00 грн.

Для кожного з випадків потенційне збільшення чистого прибутку у потенційного інвестора ΔПі в роки очікуваного позитивного результату від можливого впровадження та комерціалізації науково-технічної розробки розраховується за відповідною формулою:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

$$\Delta П_1 = (11\,000{,}00 \times 70 + 111000{,}00 \times 50) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right)$$

$$= 1\,570\,847{,}38 \text{ грн.}$$

$$\Delta П_2 = \left(11\,000{,}00 \times 70 + 111000{,}00 \times (50 + 120)\right) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right)$$

$$= 4\,881\,557{,}35 \text{ грн.}$$

$$\Delta П_3 = \left(11\,000{,}00 \times 70 + 111000{,}00 \times (50 + 120 + 180)\right) \times 0{,}83 \times 0{,}3$$

$$\times \left(1 - \frac{0{,}18}{100}\right) = 9\,847\,622{,}32 \text{ грн.}$$

Для кожного з випадків потенційне збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ в роки очікуваного позитивного результату від можливого впровадження та комерціалізації науково-технічної розробки розраховується за відповідною формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,2;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$\text{ПП} = \frac{1570847{,}38}{(1+0{,}2)^1} + \frac{4881557{,}35}{(1+0{,}2)^2} + \frac{9847622{,}32}{(1+0{,}2)^3} = 10397865{,}37 \text{ грн.}$$

Отже, згідно з розрахунками, комерційна користь від упровадження розробки буде значною, що підтверджує прогнози, і проявиться у зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Основними критеріями, які визначають доцільність інвестування у наукову розробку з боку потенційного інвестора, є абсолютна і відносна ефективність вкладених коштів, а також термін їх окупності. Першим етапом цього процесу є обчислення теперішньої вартості інвестицій (PV), що будуть спрямовані на реалізацію наукової розробки.

Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ, \tag{4.18}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =3;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 792 109,4 грн.

$$\text{PV} = 3 * 792\,109{,}4 = 2\,376\,328{,}2 \text{ грн.}$$

Таким чином, чистий приведений дохід (NPV) або абсолютний економічний ефект ($Е_{абс}$) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки буде таким:

$$E_{абс} = ПП - PV \quad (4.19)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 10397865,37 грн.

PV – теперішня вартість початкових інвестицій, 2 376 328,2 грн.

$$\text{Е}_{\text{абс}} = 10397865{,}37 - 2\,376\,328{,}2 = 8\,021\,537{,}17 \text{ грн.}$$

Внутрішня економічна дохідність ($\text{Е}_{\text{в}}$) інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, обчислюється за допомогою такої формули:

$$E_{в} = \sqrt[T_{ж}]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 8 021 537,17 грн.

PV – теперішня вартість початкових інвестицій, 2 376 328,2 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$\text{Е}_{\text{в}} = \sqrt[3]{1 + \frac{8\,021\,537{,}17}{2\,376\,328{,}2}} - 1 = 0{,}6$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій (мін τ) визначається згідно такою формулою:

$$\tau_{мін} = d + f, \quad (4.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,2.

$$\tau_{min} = 0{,}2 + 0{,}2 = 0{,}4$$

Оскільки $E_{в}$ = 60% > τ_{min} = 40%, це свідчить про те, що внутрішня економічна дохідність інвестицій, які можуть бути вкладені потенційним інвестором у

впровадження та комерціалізацію науково-технічної розробки, перевищує мінімальну внутрішню дохідність.

Далі обчислюємо період окупності інвестицій ($Т_{ок}$ або DPP, Discounted Payback Period), які потенційний інвестор може вкласти у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.22}$$

$$Т_{ок} = \frac{1}{0{,}6} = 1{,}7 \text{ року}.$$

З огляду на те, що період окупності інвестицій у реалізацію наукового проєкту становить менше трьох років, можна дійти висновку, що фінансування цієї нової розробки є виправданим.

4.5 Висновки до розділу

Згідно з проведеними дослідженнями, рівень комерційного потенціалу розробки на тему «Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі ризик-орієнтованих і атрибутивних політик доступу» становить 47,6 бала, що свідчить про високу комерційну значущість цієї розробки.

Термін окупності проекту становить 1,7 року, що значно коротше за стандартний трирічний період. Це підтверджує привабливість розробки для комерціалізації та може заохотити потенційних інвесторів до фінансування її впровадження і виходу на ринок.

Отже, можна зробити висновок про доцільність проведення науково-дослідних робіт за цією темою, оскільки вони мають високий потенціал для комерційного успіху.

ВИСНОВКИ

У результаті виконаної роботи було досліджено, розроблено та протестовано захищену систему управління доступом на основі вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій. Система поєднує сучасні підходи до кібербезпеки, такі як ризик-орієнтовані та атрибутивні політики доступу, що дозволяє підвищити рівень захисту інформаційних ресурсів.

У першому розділі було виконано аналіз сучасних загроз та вимог до систем управління доступом. Було розглянуто основні принципи ризик-орієнтованих і атрибутивних політик доступу, які формують базу для адаптивного управління доступом залежно від контексту. Здійснено аналіз методів динамічного контролю доступу з урахуванням рівня ризику, що дозволило виявити ключові підходи до вдосконалення безпеки систем. Огляд сучасних загроз показав, що традиційні системи управління доступом часто не можуть ефективно реагувати на динамічні ризики, що обґрунтовує необхідність впровадження адаптивних методів контролю.

У другому розділі було розроблено концептуальну модель системи, яка включає алгоритми контролю доступу та динамічного моніторингу аномалій. Було створено алгоритми для оцінки атрибутів користувачів, контексту доступу та рівнів ризику. Особливу увагу приділено механізмам адаптивної зміни рівня доступу в реальному часі залежно від динамічних умов. Це дозволило закласти основу для створення гнучкої системи управління доступом, яка здатна реагувати на потенційні загрози в процесі їх виникнення.

У третьому розділі було реалізовано програмну частину системи. Розроблено модулі для керування базою даних, контролю доступу, моніторингу аномалій та графічного інтерфейсу. Впроваджено механізми динамічного оцінювання ризиків і адаптивного реагування на виявлені загрози. Графічний інтерфейс забезпечив зручний доступ до функціональності системи для адміністраторів і користувачів. Тестування системи підтвердило її коректність, стабільність і відповідність функціональним та безпековим вимогам. Зокрема, система успішно виявляє аномалії, реагує на ризики та динамічно адаптує рівень доступу залежно від контексту.

Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу

ВИКОНАВ: СТ. 2 КУРСУ, ГРУПИ КІТС-23М
АЛЛАХВЕРДІЄВ ОЛЕКСАНДР ЕМІЛЬ ОГЛИ
КЕРІВНИК: Д.Т.Н., ДОЦ. КАФ. МБІС ГРИЦАК А.В.

ВСТУП

Сучасні інформаційні системи стають дедалі складнішими та важливішими для забезпечення роботи організацій, що призводить до зростання кількості кібератак, спрямованих на несанкціонований доступ до даних. Традиційні методи управління доступом, які базуються на статичних правилах, часто виявляються недостатньо ефективними в умовах динамічних загроз. Зокрема, фіксовані політики доступу не враховують змінні фактори, такі як місцезнаходження користувача, поведінкові патерни або рівень ризику доступу до ресурсу. Це обґрунтовує необхідність розробки адаптивних систем управління доступом, які дозволяють динамічно оцінювати контекст запиту та виявляти аномалії в реальному часі.

Актуальність та мета

Актуальність роботи полягає в необхідності підвищення рівня безпеки інформаційних систем у сучасних умовах зростання кількості кіберзагроз. Традиційні підходи до управління доступом стають недостатньо ефективними через їхню статичність і нездатність адаптуватися до динамічних умов. Впровадження вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій дозволяє створити гнучкі та ефективні системи управління доступом, які враховують рівень ризику і контекст запитів, що особливо важливо для захисту критично важливих ресурсів у корпоративних, хмарних і IoT-середовищах.

Мета і задачі дослідження. Метою роботи є розробка захищеної системи управління доступом, яка базується на вдосконалених адаптивних алгоритмах контролю доступу та динамічного моніторингу аномалій. Ця система повинна забезпечувати високий рівень безпеки та гнучкості в управлінні доступом, адаптуючись до змінних умов і ризиків.

Важливість захисту інформаційних систем та актуальність адаптивного управління доступом

Захист інформаційних систем є критично важливим у сучасному світі, де дані виступають стратегічним ресурсом для бізнесу, урядових структур, медицини та інших сфер. Зростання кількості кіберзагроз, таких як фішинг, зловмисне програмне забезпечення та інсайдерські атаки, підсилює необхідність у надійних системах безпеки. Крім того, розширення IT-інфраструктури за рахунок хмарних технологій, мобільних пристроїв і IoT створює нові точки доступу, що потребують ефективного управління. Традиційні методи контролю доступу, такі як рольовий доступ, не враховують контексту запиту та динамічних змін, що робить їх уразливими. Адаптивне управління доступом дозволяє динамічно змінювати рівень доступу залежно від ризиків, враховуючи місцезнаходження, час, тип пристрою і поведінку користувача. Воно інтегрує передові технології, такі як штучний інтелект і поведінковий аналіз, для швидкого виявлення загроз і підвищення безпеки. Цей підхід забезпечує баланс між зручністю для користувачів і захистом ресурсів, реагуючи на загрози в реальному часі, що робить його актуальним у сучасних умовах кібербезпеки

Основні принципи ризик-орієнтованих та атрибутивних політик доступу

Основні принципи ризик-орієнтованих та атрибутивних політик доступу базуються на сучасному підході до управління доступом, що враховує змінні фактори ризику та характеристики користувачів і ресурсів. Ці принципи забезпечують гнучкість і точність у визначенні прав доступу, підвищуючи рівень безпеки системи.

Ризик-орієнтовані політики доступу спрямовані на оцінку та управління ризиками, пов'язаними з кожним запитом. Основним принципом є врахування динамічних параметрів, таких як місцезнаходження, час доступу, тип пристрою, IP-адреса та інші контекстуальні фактори. Наприклад, якщо користувач намагається отримати доступ до системи з незвичного місця чи пристрою, політика може вимагати додаткову автентифікацію або обмежити рівень доступу. Такий підхід дозволяє оперативно реагувати на зміну умов і мінімізувати ризики, знижуючи ймовірність компрометації системи.

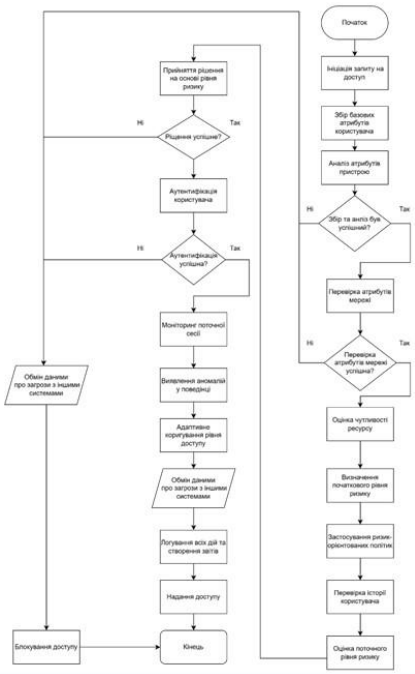
Атрибутивні політики доступу ґрунтуються на використанні набору атрибутів для прийняття рішень про доступ. Ці атрибути можуть стосуватися як користувача (роль, посада, рівень доступу, відділ), так і ресурсу (тип, рівень конфіденційності) або контексту (час, місце, мережа). Наприклад, доступ до конфіденційного документа може бути дозволений лише в робочий час із корпоративної мережі. Цей підхід забезпечує деталізований контроль і враховує різноманітні умови доступу.

Спільним принципом обох підходів є динамічність і адаптивність. Системи, які використовують ці політики, можуть змінювати права доступу залежно від поточних обставин, що дозволяє знижувати ризики компрометації даних. Це особливо важливо в умовах сучасних кіберзагроз, коли статичні політики стають недостатньо ефективними. Ще одним важливим принципом є інтеграція з іншими елементами системи безпеки, такими як моніторинг аномалій чи поведінковий аналіз, що дозволяє створити багаторівневу систему захисту.

Таким чином, ризик-орієнтовані та атрибутивні політики доступу є ключовими компонентами сучасних систем управління доступом, які забезпечують як безпеку, так і гнучкість у регулюванні прав доступу залежно від ризиків і атрибутів користувачів. Їхнє впровадження дозволяє знизити ймовірність загроз і забезпечити ефективне управління доступом у складних інформаційних середовищах.

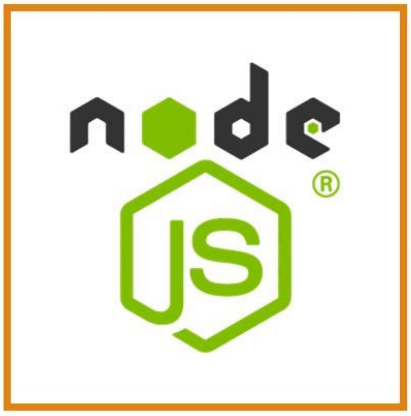
Порівняння методів динамічного контролю доступу з урахуванням рівня ризику

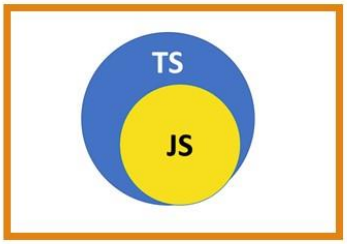
Метод контролю доступу	Опис	Переваги	Недоліки
Адаптивний контроль доступу на основі ризику	Динамічно оцінює ризики та коригує рівень доступу на основі поведінки користувача та параметрів запиту.	Гнучкий контроль, знижує ризики компрометації доступу, адаптивність до умов.	Складність налаштування, потребує постійного моніторингу і корекцій.
Політики на основі атрибутів та контексту	Враховує атрибути користувача, ресурсів і контекст доступу (час, місце, пристрій) для визначення доступу.	Детальний контроль, можливість враховувати різні умови доступу, підвищена безпека.	Потребує розширених налаштувань та моніторингу для обліку зміни контексту.
Використання AI та машинного навчання	Аналізує великі обсяги даних для прогнозування ризиків та автоматичного коригування політик доступу.	Прогнозування загроз, постійне самонавчання, виявлення нових патернів.	Високі вимоги до ресурсів, можливість помилкових спрацювань при аномальних діях.
Динамічна багатофакторна аутентифікація (MFA)	Автоматично вимагає додаткові фактори аутентифікації при підвищеному ризику для збереження безпеки.	Баланс безпеки і зручності, адаптується до рівня ризику, посилює захист.	Додаткове навантаження на користувачів при частих запитах на MFA.
Поведінковий аналіз та відстеження аномалій	Виявляє аномалії на основі відхилення від звичайної поведінки користувача та автоматично коригує доступ.	Ефективний захист від інсайдерських загроз, автоматична реакція на аномалії.	Необхідність високоякісних даних для аналізу, ресурсоємність та налаштування SIEM/IDS.

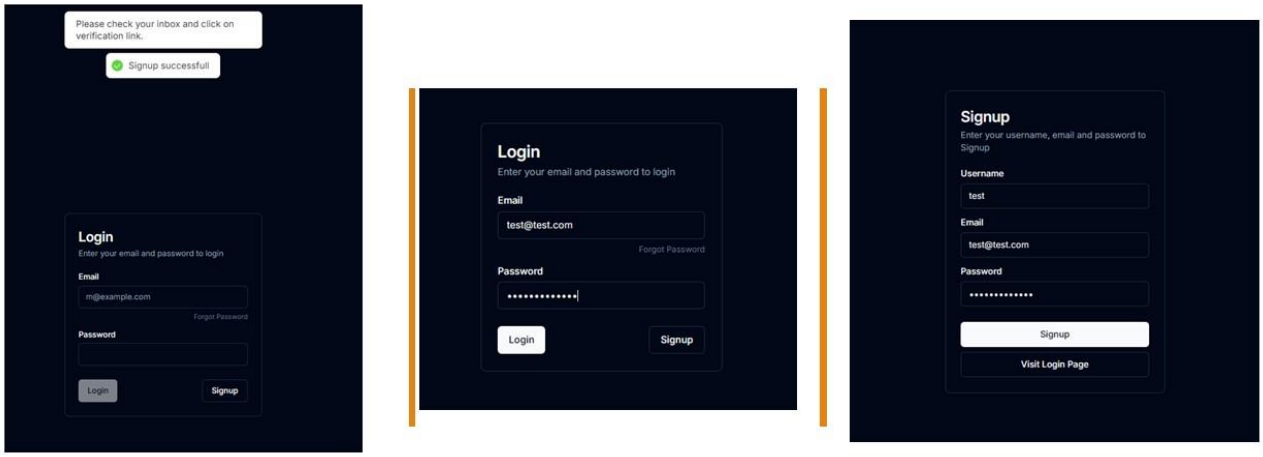


Алгоритму контролю доступу

Використані технології







Тестування системи

Тестування системи

Тестування системи

Висновок

У результаті виконаної роботи було досліджено, розроблено та протестовано захищену систему управління доступом на основі вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій. Система поєднує сучасні підходи до кібербезпеки, такі як ризик-орієнтовані та атрибутивні політики доступу, що дозволяє підвищити рівень захисту інформаційних ресурсів.

Загалом виконана робота продемонструвала, що розроблена система управління доступом відповідає сучасним вимогам кібербезпеки та дозволяє ефективно забезпечувати захист інформаційних ресурсів. Вона поєднує адаптивність, гнучкість і високу продуктивність, що робить її перспективною для впровадження у різних інформаційних середовищах.

ДЯКУЮ ЗА УВАГУ!

Загалом виконана робота продемонструвала, що розроблена система управління доступом відповідає сучасним вимогам кібербезпеки та дозволяє ефективно забезпечувати захист інформаційних ресурсів. Вона поєднує адаптивність, гнучкість і високу продуктивність, що робить її перспективною для впровадження у різних інформаційних середовищах. У подальшому можливий розвиток системи шляхом удосконалення алгоритмів аналізу ризиків, інтеграції з іншими платформами кібербезпеки та оптимізації продуктивності для роботи у великих масштабах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tanenbaum A. S., Wetherall D. J. Computer Networks. 5th ed. Pearson, 2010. 960 p.

2. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Pearson, 2020. 768 p.

3. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 20th Anniversary Ed. Wiley, 2015. 928 p.

4. NIST Special Publication 800-63B. Digital Identity Guidelines: Authentication and Lifecycle Management. Gaithersburg, 2017. URL: https://doi.org/10.6028/NIST.SP.800-63b.

5. OWASP Foundation. OWASP Top Ten Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/.

6. Kim D., Solomon M. Fundamentals of Information Systems Security. 4th ed. Jones & Bartlett Learning, 2021. 542 p.

7. Shostack A. Threat Modeling: Designing for Security. Wiley, 2014. 624 p.

8. Коваленко С. В., Клименко В. М. Захист інформації в комп'ютерних системах. Харків : Вид-во ХНУРЕ, 2018. 272 с.

9. Zettler J. Mastering Modern Web Penetration Testing. Packt Publishing, 2016. 244 p.

10. RFC 6749: The OAuth 2.0 Authorization Framework. Internet Engineering Task Force, 2012. URL: https://datatracker.ietf.org/doc/html/rfc6749.

11. RFC 7252: The Constrained Application Protocol (CoAP). Internet Engineering Task Force, 2014. URL: https://datatracker.ietf.org/doc/html/rfc7252.

12. АНАЛІЗ МЕТОДІВ ДИНАМІЧНОГО КОНТРОЛЮ ДОСТУПУ З УРАХУВАННЯМ РІВНЯ РИЗИКУ URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/author/submission/22835

13. AWS Security Best Practices. Amazon Web Services Documentation. URL: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/.

14. What is Node.js? Node.js Documentation. URL: https://nodejs.org/en/docs/.

15. Mastering NestJS: A Progressive Node.js Framework. URL: https://nestjs.com/.

16. JavaScript Guide. Mozilla Developer Network (MDN). URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide.

17. Using Next.js. Next.js Documentation. URL: https://nextjs.org/docs/getting-started.

18. Kovacs A. Practical Web Development. Packt Publishing, 2016. 472 p.

19. Ватаманюк І. Т., Пічурін І. В. Сучасні методи захисту інформації: монографія. Київ : НТУУ "КПІ", 2020. 196 с.

20. How MFA Works: Benefits and Implementation. Microsoft Documentation. URL: https://learn.microsoft.com/en-us/azure/active-directory/authentication/multi-factor-authentication.

21. Machine Learning for Cybersecurity. Springer International Publishing, 2021. 340 p.

22. NIST Special Publication 800-207. Zero Trust Architecture. Gaithersburg, 2020. URL: https://doi.org/10.6028/NIST.SP.800-207.

23. Eberle W. Dynamic Anomaly Detection: Algorithms and Applications. Springer, 2020. 280 p.

24. Войцеховський Д. В., Руденко В. О. Сучасні підходи до управління доступом у розподілених системах. Київ : КНУ ім. Т. Шевченка, 2018. 188 с.

25. ISO/IEC 27001:2013. Information Security Management Systems. Geneva: International Organization for Standardization, 2013. URL: https://www.iso.org/isoiec-27001-information-security.html.

26. Hostinger Tutorials. What is CoAP? Understanding the Constrained Application Protocol. URL: https://www.hostinger.com/tutorials/what-is-coap.

27. Демиденко С. О. Системи моніторингу виявлення аномалій: монографія. Харків : Видавництво ХАІ, 2020. 212 с.

28. RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. Internet Engineering Task Force, 2008. URL: https://datatracker.ietf.org/doc/html/rfc5246.

29. RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3. Internet Engineering Task Force, 2018. URL: https://datatracker.ietf.org/doc/html/rfc8446.

30. Protecting Cloud Data: Security Recommendations. Google Cloud Documentation. URL: https://cloud.google.com/security.

31. Кавецький В. В., Лесько О. Й. Ефективність інформаційних систем безпеки: методичні рекомендації. Вінниця : ВНТУ, 2019. 64 с.

32. What is MFA? Benefits and Use Cases. Duo Security Tutorials. URL: https://duo.com/docs/mfa.

33. Practical Next.js Security Patterns. Snyk Blog. URL: https://snyk.io/blog/nextjs-security-best-practices.

34. Що таке NestJS: огляд фреймворку. DevUA Blog. URL: https://devua.com/nestjs-overview.

35. Як працює Node.js: основи продуктивності. JavaScript Tutorials. URL: https://javascript.info/nodejs-basics.

36. Ван Дж. Інтеграція CoAP в IoT-середовища: підходи та виклики. IoT Digest, 2021. URL: https://iotdigest.com/coap-integration.

37. OWASP API Security Top Ten. OWASP Foundation. URL: https://owasp.org/www-project-api-security/.

38. Security by Design: Building Secure Systems. Addison-Wesley, 2021. 256 p.

39. Економічне обґрунтування впровадження систем кібербезпеки / Уклад. : В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2020. 52 с.

40. Методичні вказівки до виконання курсових робіт із захисту інформації. Харків : Вид-во ХНУРЕ, 2019. 45 с.

41. Вендров А. Б. Управління безпекою інформаційних ресурсів: навчальний посібник. Київ : НТУУ "КПІ", 2022. 128 с.

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

43. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с

ДОДАТКИ

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

"20" вересня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу

08-72.МКР.001.00.132.ТЗ

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент Грицак А.В.

_____

Вінниця – 2024 р.

1. Найменування та область застосування

Програмний засіб підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №310 від 17. 09. 2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу захисту від атак

3.2 Призначення: розроблений програмний засіб виконує захист від атак

4. Джерела розробки

4.1. Ахрамович В. М. Ідентифікація й аутентифікація, керування доступом // Сучасний захист інформації. – 2016. №4.– С. 47-51.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

4.4. ZakariaOmar, ZangooeiToomaj, MohdAfiziMohdShukran. Enhancing Mixing Recognition-Based and Recall-Based Approach in Graphical Password Scheme. IJACT, Vol. 4, No. 15, pp. 189-197, 2012.

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 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– 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 3.4

7. Вимоги до технічного захисту інформації

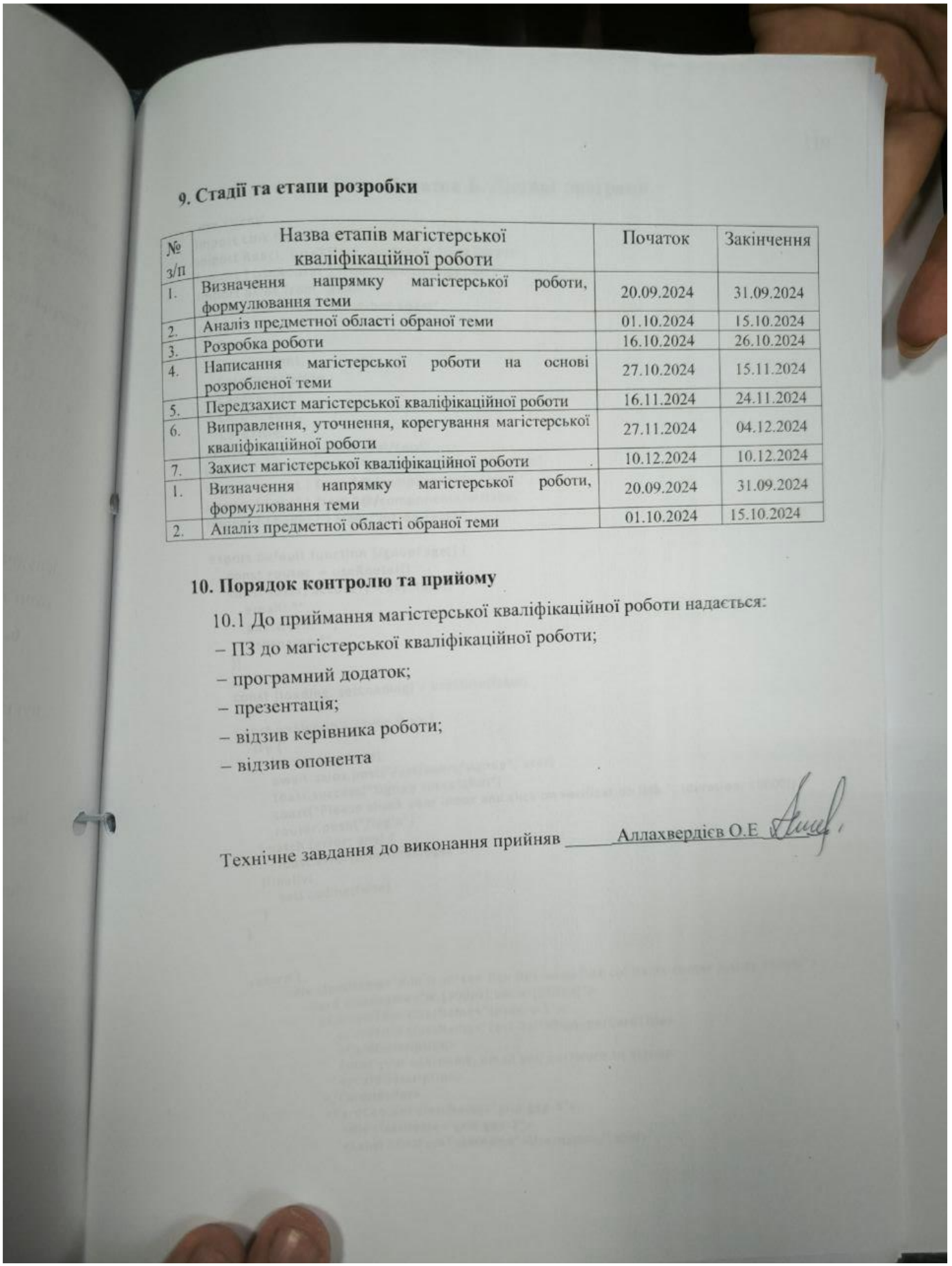
7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

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 магістерської кваліфікаційної роботи	Початок	Закінчення
1.	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024
2.	Аналіз предметної області обраної теми	01.10.2024	15.10.2024
3.	Розробка роботи	16.10.2024	26.10.2024
4.	Написання магістерської роботи на основі розробленої теми	27.10.2024	15.11.2024
5.	Передзахист магістерської кваліфікаційної роботи	16.11.2024	24.11.2024
6.	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2024	04.12.2024
7.	Захист магістерської кваліфікаційної роботи	10.12.2024	10.12.2024
1.	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	31.09.2024
2.	Аналіз предметної області обраної теми	01.10.2024	15.10.2024

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Аллахвердієв О.Е _____

Додаток Б. Лістинг програми

```
'use client'
import Link from "next/link"
import React, {useEffect, useState} from "react"
import { useRouter } from "next/navigation"
import axios from "axios"
import toast from "react-hot-toast"
import {
   Card,
   CardContent,
   CardDescription,
   CardFooter,
   CardHeader,
   CardTitle,
 } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { Loader2 } from "lucide-react"

export default function SignupPage() {
   const router  = useRouter()
   const [user, setUser] = useState({
      email: "",
      password: "",
      username:"",
   })

   const [loading, setLoading] = useState(false)

   const onSignup = async () => {
      try {
         setLoading(true)
         await axios.post("/api/users/signup", user)
         toast.success("Signup successfull")
         toast("Please check your inbox and click on verification link.", {duration: 10000})
         router.push("/login")
      } catch (error: any) {
         toast.error(error.message)
      }finally{
         setLoading(false)
      }
   }


   return (
         <div className="min-h-screen flex flex-wrap flex-col items-center justify-center">
            <Card className="w-[300px] sm:w-[350px]">
               <CardHeader className="space-y-1">
                  <CardTitle className="text-2xl">Signup</CardTitle>
                  <CardDescription>
                  Enter your username, email and password to Signup
                  </CardDescription>
               </CardHeader>
               <CardContent className="grid gap-4">
                  <div className="grid gap-2">
                  <Label htmlFor="username">Username</Label>
```

```
                        <Input
                          id="username"
                          type="text"
                          placeholder="username"
                          value={user.username}
                          onChange={(e) => setUser({...user, username:e.target.value})}
                        />
                      </div>
                      <div className="grid gap-2">
                      <Label htmlFor="email">Email</Label>
                        <Input
                          id="email"
                          type="email"
                          placeholder="m@example.com"
                          value={user.email}
                          onChange={(e) => setUser({...user, email:e.target.value})}
                        />

                      </div>
                      <div className="grid gap-2">
                      <Label htmlFor="password">Password</Label>
                        <Input
                          id="password"
                          type="password"
                          value={user.password}
                          onChange={(e) => setUser({...user, password:e.target.value})}
                        />
                      </div>
                    </CardContent>
                    <CardFooter className="flex flex-col space-y-2">
                                <Button className="w-full" disabled={user.username.length>0 &&
user.email.length >0 && user.password.length >0 ? false : true} onClick={onSignup}>
                          {loading ? <Loader2 className="mr-2 h-4 w-4 animate-spin" /> : "Signup"}
                        </Button>
                                  <Link className="w-full" href='/login'><Button className="w-full"
variant='outline'>Visit Login Page</Button></Link>
                      </CardFooter>
                  </Card>
                  </div>
          )
        }
        import { connect } from "@/dbConfig/dbConfig";
        import User from "@/models/userModel";
        import { NextRequest, NextResponse } from "next/server";
        import bcryptjs from "bcryptjs";
        import { sendEmail } from "@/helpers/mailer";

        connect();

        export async function POST(request: NextRequest) {
         try {
          const reqBody = await request.json();
          const { username, email, password } = reqBody;
          console.log(reqBody);

          //check if user already exists
          const user = await User.findOne({ email });
          if (user) {
```

```
      return NextResponse.json(
        { error: "User Already Exists" },
        { status: 400 }
      );
    }
    //Hash the password
    const salt = await bcryptjs.genSalt(10);
    const hashedPassword = await bcryptjs.hash(password, salt);
    //Create a new user
    const newUser = new User({
      username,
      email,
      password: hashedPassword,
    });
    //Save the new user to the database
    const savedUser = await newUser.save();
    console.log(savedUser);

    //send verification email

    // await sendEmail({email, emailType: "VERIFY", userId: savedUser._id})

    //Returning the response
    return NextResponse.json({
      message: "User created successfully",
      success: true,
      savedUser,
    });
  } catch (error: any) {
    return NextResponse.json({ error: error.message }, { status: 500 });
  }
}

import { getDataFromToken } from '@/helpers/getDataFromToken'
import { NextRequest, NextResponse } from 'next/server'
import User from "@/models/userModel"
import {connect} from "@/dbConfig/dbConfig"

connect()

export async function GET(request:NextRequest) {
    try {
        const userId = await getDataFromToken(request)
        const user = await User.findOne({_id: userId}).select("-password")
        return NextResponse.json({
            message: "User found",
            data: user,
        })
    } catch (error:any) {
        return NextResponse.json({ error: error.message}, {status: 400})

    }
}
import { connect } from '@/dbConfig/dbConfig'
import User from '@/models/userModel'
import { NextRequest, NextResponse } from 'next/server'
import bcryptjs from "bcryptjs"
import jwt from "jsonwebtoken"
```

```
connect()

export async function POST(request:NextRequest){
   try {
      const reqBody = await request.json()
      const {email, password} = reqBody
      console.log(reqBody)

      //Check if user is exists
      const user = await User.findOne({email})
      if(!user){
         return NextResponse.json({error: "User not found"}, {status: 400})
      }

      //Check if password is correct
      const validPassword = await bcryptjs.compare(password, user.password)
      if(!validPassword){
         return NextResponse.json({error: "Invalid Password"}, {status: 400})
      }

      //Create token data
      const tokenData = {
         id: user._id,
         username: user.username,
         email: user.email,
      }

      //Create token
      const token = await jwt.sign(tokenData, process.env.TOKEN_SECRET!, {expiresIn: "1d"})

      //Sent token to user cookies
      const response = NextResponse.json({
         message: "Login successful",
         success: true,
      })
      response.cookies.set("token", token, {
         httpOnly: true,
      })
      return response

   } catch (error: any) {
      return NextResponse.json({error: error.message}, {status: 500})
   }
}
import { NextResponse } from "next/server";

export async function GET() {
   try {
      const response = NextResponse.json({
         message: "Logout Successfull",
         success: true,
      })
      response.cookies.set("token", "", { httpOnly: true, expires: new Date(0)})
      return response
   } catch (error) {
      return NextResponse.json({ error: error.message}, {status: 500})
   }
```

```
}
'use client'
import { useState, useEffect } from "react"
import axios from "axios"
import { toast } from "react-hot-toast"
import { useRouter } from "next/navigation"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import {
   Card,
   CardContent,
   CardDescription,
   CardHeader,
   CardTitle,
 } from "@/components/ui/card"
import {
   HoverCard,
   HoverCardContent,
   HoverCardTrigger,
 } from "@/components/ui/hover-card"
import { Loader2 } from "lucide-react"

export default function PasswordReset() {
   const router = useRouter()
   const [token, setToken] = useState("")
   const [password, setPassword] = useState("")
   const [loading, setLoading] = useState(false)

   const changePassword = async () => {
      try {
         setLoading(true)
         const response = await axios.post('/api/users/passwordreset/', {token, password})
         toast.success("Password Changed Successfully", response.data.message)
         setPassword("")
         router.push("/login")
      } catch (error:any) {
         toast.error(error.message)
      }finally{
         setLoading(false)
      }
   }

   useEffect(() => {
      const urlToken = window.location.search.split("=")[1]
      setToken(urlToken || "")
   }, [])

   return (
      <div className="flex flex-col items-center justify-center min-h-screen py-2 container">
         <HoverCard>
            <HoverCardTrigger>
            <Button variant="link">Token</Button>
            </HoverCardTrigger>
            <HoverCardContent className="w-max">
            {token ? `${token}` : "no token"}
            </HoverCardContent>
         </HoverCard>
         <Card className="w-[300px] sm:w-[350px]">
```

```
                <CardHeader className="space-y-1">
                  <CardTitle className="text-2xl">Password Reset</CardTitle>
                  <CardDescription>
                  Please Enter Your New Password
                  </CardDescription>
                </CardHeader>
                <CardContent className="grid gap-4">
                <div className="grid gap-2">
                    <Input type="password" placeholder="password" value={password} id="password"
onChange={(e) => setPassword(e.target.value)} />
                <Button disabled={password.length >0 ? false : true} onClick={changePassword}>
                {loading ? <Loader2 className="mr-2 h-4 w-4 animate-spin" /> : "Submit"}
                </Button>
                </div>
                </CardContent>
                </Card>
              </div>
            )
          }
          'use client'
          import axios from "axios"
          import Link from "next/link"
          import { useState, useEffect } from "react"
          import { toast } from "react-hot-toast"
          import {Button} from '@/components/ui/button'
          import { useRouter } from "next/navigation"

          export default function VerifyEmailPage() {

            const router = useRouter()

            const [token, setToken] = useState("")
            const [verified, setVerified] = useState(false)
            const [error, setError] = useState(false)

            const verifyUserEmail = async () => {
              try {
                await axios.post('/api/users/verifyemail', {token})
                setVerified(true)
                toast.success("Your account has been verified")
              } catch (error: any) {
                setError(true)
                toast.error(error.message)
              }
            }

            useEffect(() => {
              const urlToken = window.location.search.split("=")[1]
              setToken(urlToken || "")
            }, [])

            useEffect(() => {
              if(token.length > 0){
                verifyUserEmail()
              }
            }, [token])

            useEffect(() => {
```

```
            const timer = setTimeout(() => {
              router.push("/login")
            }, 5000);
            return () => clearTimeout(timer);
          }, []);

          return (
            <div className="flex flex-col items-center justify-center min-h-screen py-4">
              <h1 className="text-4xl">Email Verification</h1>
              <h2 className="p-2 bg-orange-500 text-black">{token ? `${token}` : "no token"}</h2>
              {verified && (
                <div>
                                                    <Link     className="py-2"     href={"/login"}><Button
variant='outline'>Login</Button></Link>
                </div>
              )}
              {error && (
                <div>
                  <h2 className="text-2xl bg-red-500 text-black">Error</h2>
                </div>
              )}
            </div>
          )
        }
        @tailwind base;
        @tailwind components;
        @tailwind utilities;

        @layer base {
          :root {
            --background: 0 0% 100%;
            --foreground: 222.2 84% 4.9%;

            --card: 0 0% 100%;
            --card-foreground: 222.2 84% 4.9%;

            --popover: 0 0% 100%;
            --popover-foreground: 222.2 84% 4.9%;

            --primary: 222.2 47.4% 11.2%;
            --primary-foreground: 210 40% 98%;

            --secondary: 210 40% 96.1%;
            --secondary-foreground: 222.2 47.4% 11.2%;

            --muted: 210 40% 96.1%;
            --muted-foreground: 215.4 16.3% 46.9%;

            --accent: 210 40% 96.1%;
            --accent-foreground: 222.2 47.4% 11.2%;

            --destructive: 0 84.2% 60.2%;
            --destructive-foreground: 210 40% 98%;

            --border: 214.3 31.8% 91.4%;
            --input: 214.3 31.8% 91.4%;
            --ring: 222.2 84% 4.9%;
```

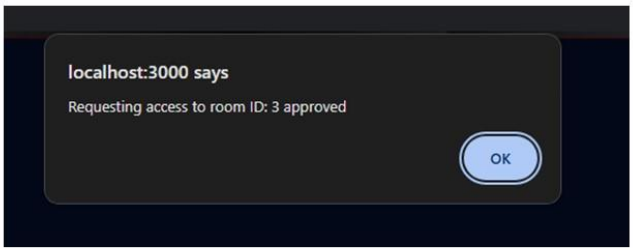
```
    --radius: 0.5rem;
  }

  .dark {
    --background: 222.2 84% 4.9%;
    --foreground: 210 40% 98%;

    --card: 222.2 84% 4.9%;
    --card-foreground: 210 40% 98%;

    --popover: 222.2 84% 4.9%;
    --popover-foreground: 210 40% 98%;

    --primary: 210 40% 98%;
    --primary-foreground: 222.2 47.4% 11.2%;

    --secondary: 217.2 32.6% 17.5%;
    --secondary-foreground: 210 40% 98%;

    --muted: 217.2 32.6% 17.5%;
    --muted-foreground: 215 20.2% 65.1%;

    --accent: 217.2 32.6% 17.5%;
    --accent-foreground: 210 40% 98%;

    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 210 40% 98%;

    --border: 217.2 32.6% 17.5%;
    --input: 217.2 32.6% 17.5%;
    --ring: hsl(212.7,26.8%,83.9);
  }
}

@layer base {
  * {
    @apply border-border;
  }
  body {
    @apply bg-background text-foreground;
  }
}
```

Додаток В. Ілюстративний матеріал

Висновок

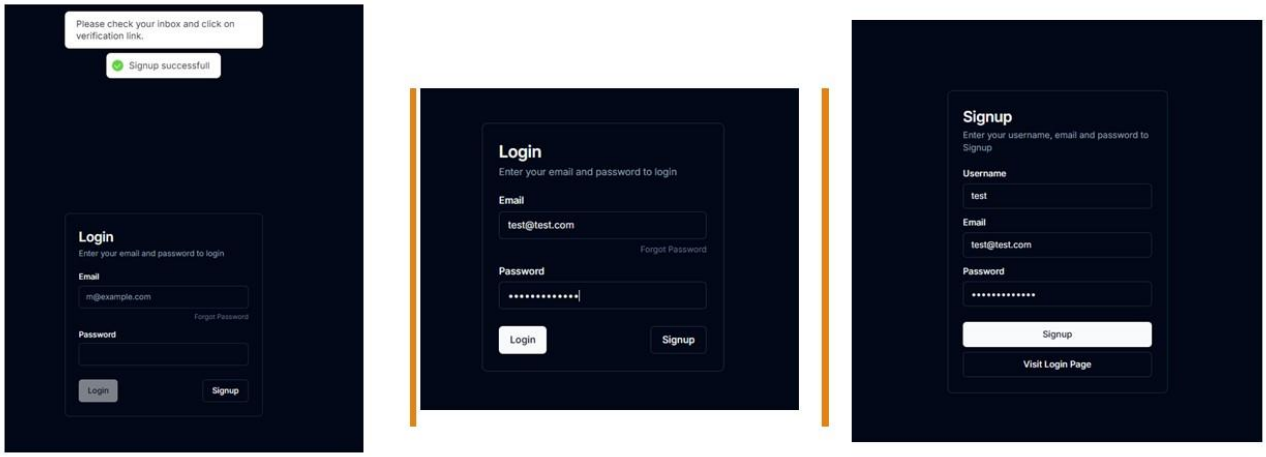
У результаті виконаної роботи було досліджено, розроблено та протестовано захищену систему управління доступом на основі вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій. Система поєднує сучасні підходи до кібербезпеки, такі як ризик-орієнтовані та атрибутивні політики доступу, що дозволяє підвищити рівень захисту інформаційних ресурсів.

Загалом виконана робота продемонструвала, що розроблена система управління доступом відповідає сучасним вимогам кібербезпеки та дозволяє ефективно забезпечувати захист інформаційних ресурсів. Вона поєднує адаптивність, гнучкість і високу продуктивність, що робить її перспективною для впровадження у різних інформаційних середовищах.



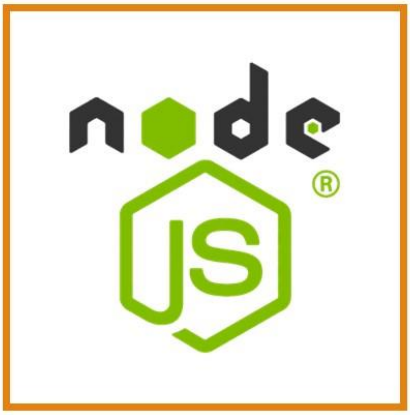
Тестування системи

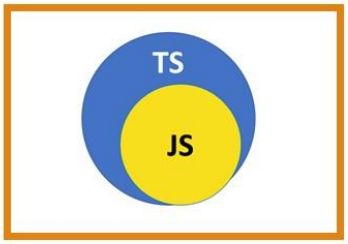
Тестування системи

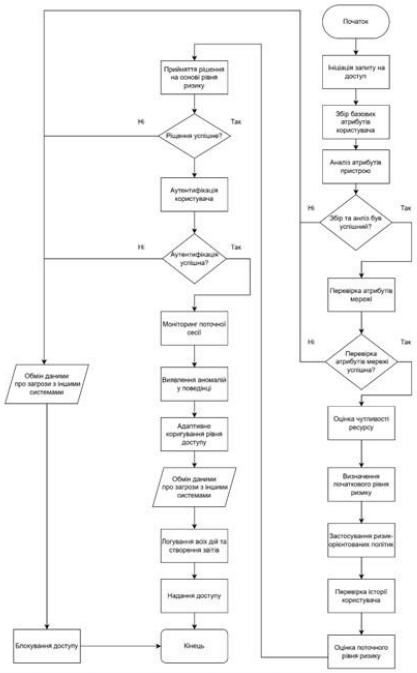


Тестування системи

Використані технології







Алгоритму контролю доступу

Порівняння методів динамічного контролю доступу з урахуванням рівня ризику

Метод контролю доступу	Опис	Переваги	Недоліки
Адаптивний контроль доступу на основі ризику	Динамічно оцінює ризики та коригує рівень доступу на основі поведінки користувача та параметрів запиту.	Гнучкий контроль, знижує ризики компрометації доступу, адаптивність до умов.	Складність налаштування, потребує постійного моніторингу і корекції.
Політики на основі атрибутів та контексту	Враховує атрибути користувача, ресурсів і контекст доступу (час, місце, пристрій) для визначення доступу.	Детальний контроль, можливість враховувати різні умови доступу, підвищена безпека.	Потребує розширених налаштувань та моніторингу для обліку зміни контексту.
Використання AI та машинного навчання	Аналізує великі обсяги даних для прогнозування ризиків та автоматичного коригування політик доступу.	Прогнозування загроз, постійне самонавчання, виявлення нових патернів.	Високі вимоги до ресурсів, можливість помилкових спрацювань при аномальних діях.
Динамічна багатофакторна аутентифікація (MFA)	Автоматично вимагає додаткові фактори аутентифікації при підвищеному ризику для збереження безпеки.	Баланс безпеки і зручності, адаптується до рівня ризику, посилює захист.	Додаткове навантаження на користувачів при частих запитах на MFA.
Поведінковий аналіз та відстеження аномалій	Виявляє аномалії на основі відхилення від звичайної поведінки користувача та автоматично коригує доступ.	Ефективний захист від інсайдерських загроз, автоматична реакція на аномалії.	Необхідність високоякісних даних для аналізу, ресурсоємність та налаштування SIEM/IDS.

Основні принципи ризик-орієнтованих та атрибутивних політик доступу

Основні принципи ризик-орієнтованих та атрибутивних політик доступу базуються на сучасному підході до управління доступом, що враховує змінні фактори ризику та характеристики користувачів і ресурсів. Ці принципи забезпечують гнучкість і точність у визначенні прав доступу, підвищуючи рівень безпеки системи.

Ризик-орієнтовані політики доступу спрямовані на оцінку та управління ризиками, пов'язаними з кожним запитом. Основним принципом є врахування динамічних параметрів, таких як місцезнаходження, час доступу, тип пристрою, IP-адреса та інші контекстуальні фактори. Наприклад, якщо користувач намагається отримати доступ до системи з незвичного місця чи пристрою, політика може вимагати додаткову автентифікацію або обмежити рівень доступу. Такий підхід дозволяє оперативно реагувати на зміну умов і мінімізувати ризики, знижуючи ймовірність компрометації системи.

Атрибутивні політики доступу ґрунтуються на використанні набору атрибутів для прийняття рішень про доступ. Ці атрибути можуть стосуватися як користувача (роль, посада, рівень доступу, відділ), так і ресурсу (тип, рівень конфіденційності) або контексту (час, місце, мережа). Наприклад, доступ до конфіденційного документа може бути дозволений лише в робочий час із корпоративної мережі. Цей підхід забезпечує деталізований контроль і враховує різноманітні умови доступу.

Спільним принципом обох підходів є динамічність і адаптивність. Системи, які використовують ці політики, можуть змінювати права доступу залежно від поточних обставин, що дозволяє знижувати ризики компрометації даних. Це особливо важливо в умовах сучасних кіберзагроз, коли статичні політики стають недостатньо ефективними. Ще одним важливим принципом є інтеграція з іншими елементами системи безпеки, такими як моніторинг аномалій чи поведінковий аналіз, що дозволяє створити багаторівневу систему захисту.

Таким чином, ризик-орієнтовані та атрибутивні політики доступу є ключовими компонентами сучасних систем управління доступом, які забезпечують як безпеку, так і гнучкість у регулюванні прав доступу залежно від ризиків і атрибутів користувачів. Їхнє впровадження дозволяє знизити ймовірність загроз і забезпечити ефективне управління доступом у складних інформаційних середовищах.

Важливість захисту інформаційних систем та актуальність адаптивного управління доступом

Захист інформаційних систем є критично важливим у сучасному світі, де дані виступають стратегічним ресурсом для бізнесу, урядових структур, медицини та інших сфер. Зростання кількості кіберзагроз, таких як фішинг, зловмисне програмне забезпечення та інсайдерські атаки, підсилює необхідність у надійних системах безпеки. Крім того, розширення IT-інфраструктури за рахунок хмарних технологій, мобільних пристроїв і IoT створює нові точки доступу, що потребують ефективного управління. Традиційні методи контролю доступу, такі як рольовий доступ, не враховують контексту запиту та динамічних змін, що робить їх уразливими. Адаптивне управління доступом дозволяє динамічно змінювати рівень доступу залежно від ризиків, враховуючи місцезнаходження, час, тип пристрою і поведінку користувача. Воно інтегрує передові технології, такі як штучний інтелект і поведінковий аналіз, для швидкого виявлення загроз і підвищення безпеки. Цей підхід забезпечує баланс між зручністю для користувачів і захистом ресурсів, реагуючи на загрози в реальному часі, що робить його актуальним у сучасних умовах кібербезпеки

Актуальність та мета

Актуальність роботи полягає в необхідності підвищення рівня безпеки інформаційних систем у сучасних умовах зростання кількості кіберзагроз. Традиційні підходи до управління доступом стають недостатньо ефективними через їхню статичність і нездатність адаптуватися до динамічних умов. Впровадження вдосконалених адаптивних алгоритмів контролю доступу та динамічного моніторингу аномалій дозволяє створити гнучкі та ефективні системи управління доступом, які враховують рівень ризику і контекст запитів, що особливо важливо для захисту критично важливих ресурсів у корпоративних, хмарних і IoT-середовищах.

Мета і задачі дослідження. Метою роботи є розробка захищеної системи управління доступом, яка базується на вдосконалених адаптивних алгоритмах контролю доступу та динамічного моніторингу аномалій. Ця система повинна забезпечувати високий рівень безпеки та гнучкості в управлінні доступом, адаптуючись до змінних умов і ризиків.

ВСТУП

Сучасні інформаційні системи стають дедалі складнішими та важливішими для забезпечення роботи організацій, що призводить до зростання кількості кібератак, спрямованих на несанкціонований доступ до даних. Традиційні методи управління доступом, які базуються на статичних правилах, часто виявляються недостатньо ефективними в умовах динамічних загроз. Зокрема, фіксовані політики доступу не враховують змінні фактори, такі як місцезнаходження користувача, поведінкові патерни або рівень ризику доступу до ресурсу. Це обґрунтовує необхідність розробки адаптивних систем управління доступом, які дозволяють динамічно оцінювати контекст запиту та виявляти аномалії в реальному часі.

Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу

ВИКОНАВ: СТ. 2 КУРСУ, ГРУПИ КІТС-23М
АЛЛАХВЕРДІЄВ ОЛЕКСАНДР ЕМІЛЬ ОГЛИ
КЕРІВНИК: Д.Т.Н., ДОЦ. КАФ. МБІС ГРИЦАК А.В.

ДЯКУЮ ЗА УВАГУ!

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Підвищення захищеності системи управління доступом вдосконаленими адаптивними алгоритмами контролю доступу та динамічного моніторингу аномалій на основі принципів ризик-орієнтованих та атрибутивних політик доступу

Тип роботи: магістерська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки .
Гр. КІТС-23м

Керівник доцент Грицак А.В.

Показники звіту подібності

Turnitin	
Оригінальність	96%
Загальна схожість	4%

Аналіз звіту подібності (відмітити потрібне)

- ☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Аллахвердієв О.Е.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

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

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

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)