НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»
УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський
“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека
на тему: Метод виявлення зв’язків між сутностями в криміналістичному аналізі джерел даних

Виконав (-ла): студент (-ка) 2 курсу, групи ФБ-71мп
(шифр групи)

Карбовський Платон Юрійович
(прізвище, ім’я, по батькові)

Науковий керівник к.т.н., доц. Барановський Олексій Миколайович ___
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент к.т.н.,с.н.с. Погребняк В.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (спеціалізація) – 125 Кібербезпека («Системи і технології кібербезпеки)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Карбовському Платону Юрійовичу

1. Тема дисертації: Метод виявлення зв'язків між сутностями в криміналістичному аналізі джерел даних
науковий керівник дисертації к.т.н., доц. Барановський Олексій Миколайович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «15» листопада 2018 р. № 4171-с

2. Термін подання студентом дисертації 12.12.2018 р.

3. Об'єкт дослідження _____
_____

4. Вихідні дані _____
_____
_____
_____

5. Перелік завдань, які потрібно розробити _____
_____
_____
_____
_____
_____

6. Орієнтовний перелік ілюстративного матеріалу _____
_____
_____

7. Орієнтовний перелік публікацій _____
_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

Студент _____ (підпис) _____ (ініціали, прізвище)

Науковий керівник дисертації _____ (підпис) _____ (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Робота обсягом 104 сторінки містить 18 ілюстрацій, 22 таблиці та 9 літературних посилань.

Метою дослідження є побудова і реалізація моделі розмежування доступу в розподілених системах кешування даних за для підвищення безпеки таких систем.

Об’єктом дослідження є процес розмежування доступу в розподіленій системі кешування даних, обробки та доступу до інформації.

Предметом дослідження є методи розмежування доступу в інформаційних системах.

Результати роботи викладені у вигляді описання моделі доступу в розподілених системах кешування даних та екземплярів програмного коду реалізації такої моделі.

Результати роботи можуть бути використані для створення програмного забезпечення та підвищення рівня безпеки ресурсів, що використовують розподілені системи кешування даних.

РОЗПОДІЛЕНА СИСТЕМА,КЕШУВАННЯ ДАНИХ,РОЗМЕЖУВАННЯ ДОСТУПУ, КЕРУВАННЯ ДОСТУПОМ

ABSTRACT

The work includes 104 pages, 18 figures, 22 tables and 9 literary references.

The goal of this qualification is to develop a distributed cache system access control model.

The object of research is a process of access control in distributed cache system environment.

The subject of research is information systems access control methods.

The results of the work is presented as an distributed cache system access control model description and corresponding programming code implementation.

The results of the work can be used to build a security software for the purpose of using by different project, having distributed cache systems in their architecture design.

DISTRIBUTED SYSTEM, DATA CACHING, ACCESS CONTROL, ACCESS MANAGEMENT

РЕФЕРАТ

Работа объемом 104 страниц содержит 18 иллюстраций, 22 таблицы и 9 литературных источников.

Целью данного исследования является построение и реализация модели разграничения доступа в распределенных системах кэширования данных с целью повышения безопасности таких систем.

Объектом исследования является процесс разграничения доступа в распределенной системе кэширования данных, обработки и доступа к информации.

Предметом исследования являются методы разграничения доступа в информационных системах.

Результаты работы изложены в виде описания модели доступа в распределенной системе кэширования данных, а также экземпляров программного кода реализации такой модели.

Результаты работы могут быть использованы для создания программного обеспечения для повышения уровня безопасности ресурсов, использующих распределенные системы кэширования данных.

РАСПРЕДЕЛЕННАЯ СИСТЕМА, КЭШИРОВАНИЯ ДАННЫХ, РАЗГРАНИЧЕНИЕ ДОСТУПА, УПРАВЛЕНИЕ ДОСТУПОМ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACID – Atomicity, Consistency, Isolation, Durability;

DML – Data Manipulation Language;

DDL – Data Definition Language;

JAR – Java Archive;

API – Application programming interface;

REST – Representational State Transfer;

СУБД – Система управління базами даних;

КСЗ – Комплекс засобів захисту;

RBAC - Role based access control

ВСТУП

Сьогодні ми можемо спостерігати бурхливий розвиток інформаційних технологій. Обчислювальна техніка стає застарілою лише за декілька років, а її технічні характеристики ростуть експотенційно. З розвитком цих технологій все більше різних компаній використовують інформаційні технології для автоматизації їх бізнесу. Ці зміни суттєво збільшують обсяги важливої інформації, що зберігаються у центрах обробки даних та хмарних сховищах , що призведе до збільшення навантаження на такі сховища. Це формулює тенденцію використовувати розподілені системи кешування даних все більше і більше. Багато з таких існуючих систем мають проблеми с розмежуванням доступу. У даній роботі буде розглянуто використання різних моделей розмежування доступу для таких систем, їх переваги та недоліки, а також створено такий механізм розмежування.

Актуальність роботи. Незважаючи на те, що з кожним роком створюється все більше та більше нових систем кешування даних, вони зачасту не враховують деталі реалізації методів розмежування доступу - в створених засобах зачасту використовують примітивні підходи.

Мета дослідження. Метою даного дослідження є побудова і імплементація моделі розмежування доступу в розподілених системах кешування даних за для підвищення безпеки таких систем.

Завданням дослідження є аналіз розподілених систем кешування даних та систем розмежування доступу;

побудова моделей розмежування доступу в розподілених системах кешування даних;

розробка програмного рішення для реалізації моделі розмежування доступу в таких системах.

Об'єкт дослідження. Об'єктом дослідження в даній роботі є процесс розмежування доступу в розподіленій системі кешування даних, обробки та доступу до інформації.

Предмет дослідження. Предметом дослідження є моделі та методи розмежування доступу в інформаційних системах.

Методи дослідження. Для створення моделі розмежування доступу в розподілених системах кешування даних у цій роботі будуть використовуватися існуючі методи розмежування доступу в інформаційних системах. Їх використання описано в контексті об'єктів та суб'єктів розподіленої системи кешування даних.

Наукова новизна одержаних результатів. Наукова новизна дослідження є модель розмежування доступу в розподілених системах кешування даних.

Практичне значення отриманих результатів. У даній роботі розроблено програмне рішення для реалізації моделі розмежування доступу в розподілених системах кешування даних таких як Apache Ignite і Hazelcast. Така реалізація вирішить задача несанкціонованого доступу до розподілених систем кешування даних, що в свою чергу підвищить рівень безпеки системи, де використовується такий розподілений кеш.

1 РОЗПОДІЛЕНІ СИСТЕМИ КЕШУВАННЯ ДАНИХ ТА ЇХ ВИКОРИСТАННЯ

Кешування дає змогу збільшувати продуктивність веб-додатків за рахунок використання збережених раніше даних, на кшталт відповідей на мережеві запити або результатів обчислень. Завдяки кешу, при черговому зверненні клієнта за одними і тими ж даними, сервер може обслуговувати запити швидше. Кешування – ефективний архітектурний патерн, так як більшість програм часто звертаються до одних і тих же даних і інструкцій. Ця технологія присутня на всіх рівнях обчислювальних систем. Кеші є у процесорів, жорстких дисків, серверів, браузерів. У контексті цієї роботи розглядаються в основному кеші на рівні серверів та браузерів, що потребує розгляду можливостей деяких існуючих систем кешування даних.

1.1 Існуючі системи кешування даних

1.1.1 Apache Ignite

Apache ignite – це розподілена база даних, для кешування та обробки даних. Може забезпечувати транзакціиність, обробку великих об’ємів даних в режимі реального часу та виконувати різні аналітичні навантаження[3].

Основні можливості розподіленого кешу:

1) Стійке, надійне сховище:

Оперативна пам’ять в Apache Ignite використовується не просто як розподілений кеш, а як повністю функціональний рівень зберігання. Якщо зберігання на диску вимкнено, тоді Ignite може діяти як розподілена в пам’яті бази даних або сітки даних в пам’яті. Якщо зберігання включено, тоді Ignite стає розподіленою, горизонтально масштабованою базою даних, яка гарантує повну узгодженість даних і є стійкою до повних збоїв кластера.

Ignite Native Persistence – це розподілене, ACID і SQL-сумісне дискове сховище, яке просто інтегрується з Ignite's Durable Memory як необов’язковий рівень диска.

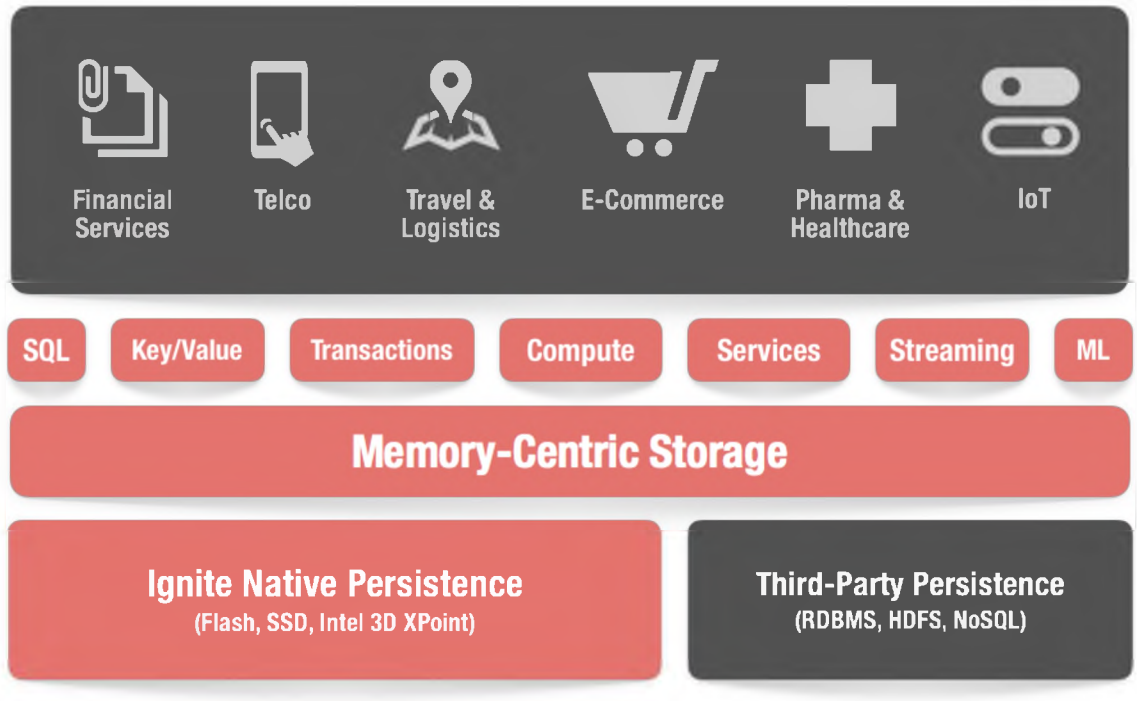

Рисунок 1.1 – Основні можливості розподіленого кешу Apache Ignite

2) Консолідація з ACID:

Дані, що зберігаються в Ignite, сумісні з ACID як у пам'яті, так і на диску, що робить Ignite строго послідовною системою. Транзакції можуть виконуватись через мережу і можуть охоплювати кілька серверів.

3) Підтримка SQL:

Ignite забезпечує повну підтримку SQL, DDL та DML, що дає змогу користувачам взаємодіяти з Ignite за допомогою чистого SQL без написання будь-якого коду. Це означає, що користувачі можуть створювати таблиці та індекси, а також вставляти, оновлювати та запитувати дані, використовуючи лише SQL. Маючи таку повну підтримку SQL, Ignite створює унікальну розподілену базу даних SQL.

1.1.2 Hazelcast

Hazelcast IMDG представляє собою розподілений кеш із відкритим вихідним кодом на основі Java. Розподілений кеш Hazelcast отримав назву від назви компанії, що розробляє цей продукт[8].

Hazelcast IMDG працює як обчислювальна платформа, яка керує даними в операційній пам'яті і організовує обробку в паралельному виконанні для досягнення найбільшої швидкості та легкості масштабування.

Hazelcast має відкритий вихідний код. У додаток до розповсюдження даних в пам'яті, Hazelcast надає зручний набір API для доступу до даних у вашому кластері для максимальної швидкості обробки. Оскільки Hazelcast постачається у вигляді компактної бібліотеки (JAR), і він не має зовнішніх залежностей, відмінних від Java, він легко підключається до вашого програмного рішення і надає розподілені структури даних та розподілені обчислювальні утиліти.

Hazelcast має високу масштабованість та доступність. Розподілені додатки можуть використовувати Hazelcast для розподіленого кешування, синхронізації, кластеризації, обробки повідомленнями тощо. Hazelcast реалізується на Java і має клієнтів для Java, C / C ++, .NET, REST, Python, Go і Node.js. Також Hazelcast підтримує протокол Memcached. Що ще важливіше, Hazelcast спрощує розподілені розрахунки та пропонує розподілені реалізації багатьох дружніх для розробників інтерфейсів з Java, таких як Map, Queue, ExecutorService, Lock і JCache. Наприклад, інтерфейс Map забезпечує зберігання значень ключа в пам'яті, що дає безліч переваг NoSQL з точки зору зручності розробників і продуктивності розробників. Hazelcast підключається до Hibernate і може бути легко використаний з будь-якою існуючою системою баз даних. Повна архітектура системи представлена на рисунку 1.1.2.

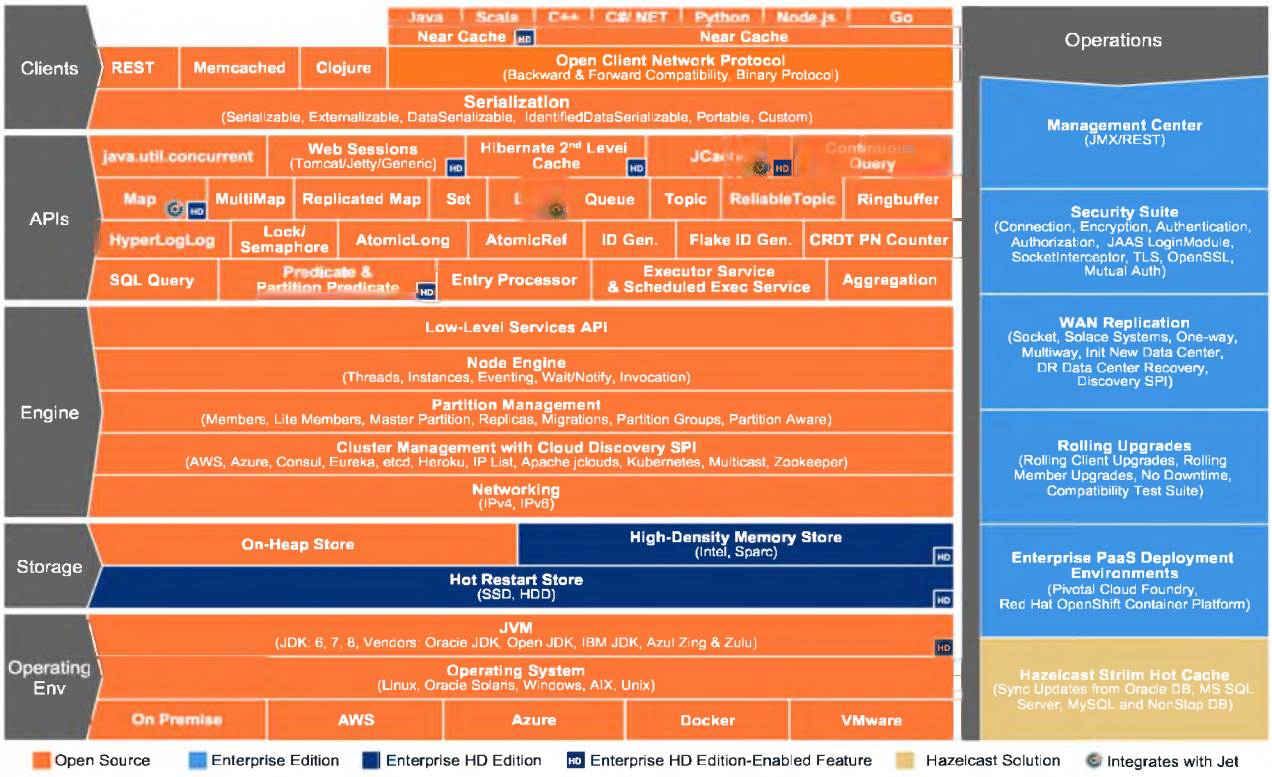

Рисунок 1.2 – Основні можливості розподіленого кешу Hazelcast

1.2 Використання систем кешування даних

На цей час системи кешування даних використовуються в багатьох проектах в різних типах інформаційної і не інформаційної діяльності. В загалі, використання таких механізмів обумовлюється великим навантаженням на системи, що потребують швидкої обробки інформації та найшвидшої її передачі кінцевому споживачу. Веб-додатки не можуть миттєво реагувати на дії користувача, що, зокрема, пов'язано з діями, які потребують обміну даними з серверами цих додатків, з необхідністю виконання деяких обчислень перед відправкою відповіді. У час, необхідний для передачі даних від сервера клієнту, входить і проміжок, необхідний для:

– пошуку необхідних даних на диску;

– мережевої затримки;

– обробки черг запитів;

– регулювання балансування мереж.

Якщо врахувати, що все це може відбуватися на безлічі комп'ютерів, що знаходяться між клієнтом і сервером, то можна говорити про те, що всі ці

затримки здатні серйозно збільшити час, необхідний для приходу запиту на сервер і отримання клієнтом відповіді. Правильно налаштована система кешування здатна значно поліпшити загальну продуктивність сервера. Кеші скорочують затримки, що неминуче виникають при передачі даних по мережі та допомагають економити мережевий трафік. В результаті, зменшується час, необхідний для того, щоб браузер вивів запитану у сервера інформацію.

1.3 Об'єкти та суб'єкти розмежування доступу в розподілених системах кешування даних

Оскільки об'єкти та суб'єкти в певних системах є конкретними, розглянемо і опишемо їх більше детально с точки зору розподілених систем кешування даних.

Суб'єкт інформаційного доступу – це, сутність що має/не має права доступу до інформації в розподіленій системі кешування даних, має/не має права створювати користувачів системи, надавати/забирати доступ у тих чи інших користувачів[2].

Основними суб'єктами інформаційного доступу є:

1) Користувачі системи.
2) Адміністратори системи.
3) Власники інформації в системі.

Всі зазначені суб'єкти доступу можна представити у виді користувача системи з деякими доступними йому привілеями.

Ключовим об'єктом в розподілених системах кешування даних та в загалому інформаційних системах виступає інформація. В розподілених системах кешування даних основними методами та структурами доступу і зберігання інформації є:

1. Розподілений кеш – аналог таблиці в базі даних.
2. Елемент запису кеша – аналог записів в таблиці в базі даних.

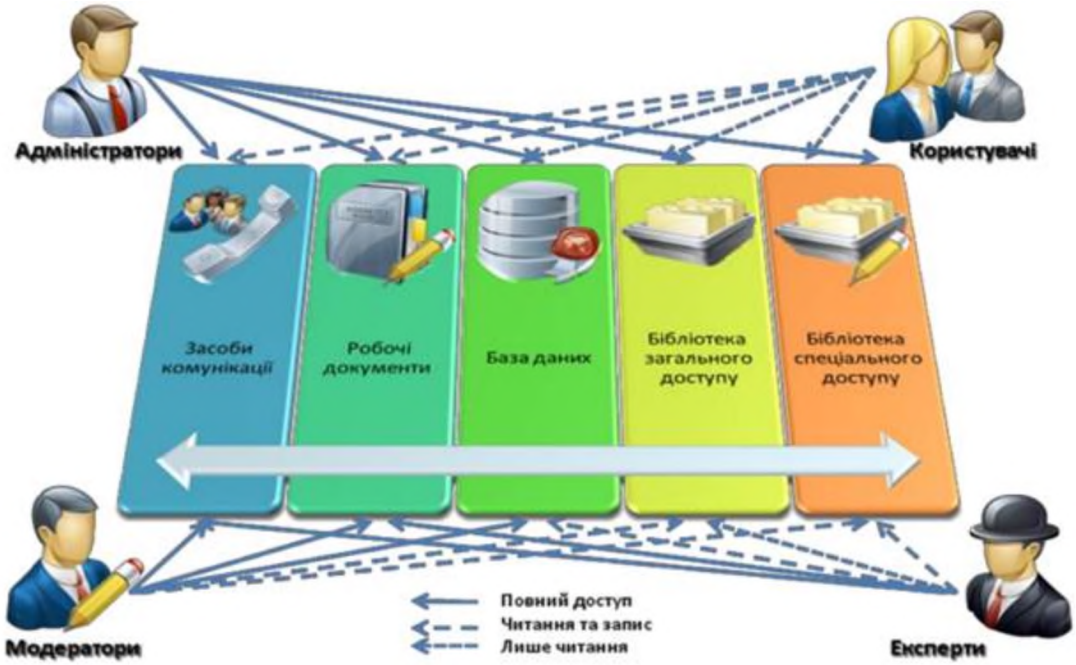

Рисунок 1.3 – Основні суб’єкти інформаційного доступу

Розмежування доступу в розподілених системах кешування даних описує правильне керування доступом та взаємодії об’єктів та суб’єктів доступу в інформаційній системі. Це включає такі дії як:

1) Правильна передача прав взаємодії об’єкта та суб’єкта в розподіленій інформаційній системі кешування даних.

2) Використання та надання правильних грифів секретності для об’єктів в розподіленій інформаційній системі кешування даних.

3) Використання та надання правильних грифів секретності для суб’єктів в розподіленій інформаційній системі кешування даних.

4) Правильна передача активів інформаційної системи кешування даних від одного суб’єкта до іншого.

5) Чітка процедура організації процесу передачі прав та активів одного суб’єкта іншому.

6) Створення та описання чіткого процесу надання доступу та передачі прав суб’єктам розподіленої інформаційної системи кешування даних. Для створення простого механізму комунікації та організації такого процесу, потрібно створити механізми на рівні розподіленої системи кешування даних, для швидкого їх використання в великих організаціях та створення

чіткого і зрозумілого підходу використання ресурсів розподіленої інформаційної системи кешування даних.

1.4 Проблеми розмежування доступу в системах кешування даних

Оскільки проблеми розмежування доступу є конкретними і практичними для всіх систем де використовується розмежування доступу між користувачами, тому розглянемо їх більш детально. Проблеми розмежування доступу виступають при будь-яких взаємодіях об'єктів інформаційної системи, і є найбільш суттєвим елементом проектування такої системи розмежування доступу. Відповідно до встановлених в інформаційній системі ієрархічних ролей – адміністраторів безпеки, адміністраторів ОС, проблеми розмежування доступу включають таке:

1. Неавторизований доступ до даних, програмними засобами або , іншим ресурсам інформаційної системи.

2. Проблеми створення резервних і архівних копій системи кешування даних.

3. Проблеми унікальних ідентифікаторів та паролів доступу в розподіленій інформаційній системі кешування даних

4. Проблеми підходу авторизації та аутентифікації в розподіленій інформаційній системі кешування даних

5. Проблеми оновлення атрибутів доступу в розподілених інформаційній системі кешування даних.

Загальні проблеми розмежування доступу мають бути конкретизовані на рівні вибору необхідних функціональних послуг захисту та впровадженні організаційними методами та вибором стратегії використання існуючих механізмів розмежування системи,що включає:

– обмеження доступу;

– розмежування доступу;

– розмежування повноважень;

– контроль і облік доступу.

У розподілених інформаційних системах кешування даних має бути реалізовано адміністративне керування доступом. Тільки адміністратори розподіленої інформаційної системи кешування даних мають право додавати і видаляти (або надавати такі права) користувачів та об'єкти. Задача обмеження доступу – протидія загрозі випадкового чи навмисного доступу сторонніх осіб на територію розміщення розподіленої інформаційної системи кешування даних та безпосередньо до її ресурсів. Обмеження доступу в розподіленій інформаційній системи кешування даних полягає в існуванні фізично або програмно замкнутої перешкоди навколо об'єкта захисту та організації контрольованого доступу осіб, зав'язаних на об'єкт захисту по своїм функціональних обов'язкам. Доступ у середу розподіленої інформаційної системи кешування, де розташовані об'єкти, контролюється за допомогою комплексу організаційно-програмних заходів, які визначені в розподіленій інформаційній системі кешування. Крім того, в середу інформаційної системи встановлюються такі обмеження на роботу користувачів:

– обмеження періоду часу, в ході якого користувач може входити в мережу;

– визначення адрес робочих станцій, з якими дозволено входити в мережу;

– обмеження кількості робочих станцій, з яких одночасно можна входити в мережу;

– обмеження кількості спроб входу в мережу з неправильним паролем.

Розмежування доступу полягає в організації доступу до інформації користувачів відповідно до їхніх функціональних особливостей і повноважень, тобто:

– визначення правил доступу до закритих ресурсів;

– визначення категорій користувачів згідно повноважень та обов'язків;

– встановлення повноважень доступу.

Задача розмежування доступу: скорочення кількості користувачів, що не мають відношення до певної категорії інформації при виконанні своїх функцій, тобто захист інформації від порушника серед допущеного до неї персоналу. Усі

користувачі можуть мати допуск до інформації з найвищим грифом. Але вони повинні мати обмеження по доступу до певних інформаційних ресурсів розподіленої інформаційної системи кешування даних в залежності від їх посадових обов'язків.

Розподіл повноважень доступу користувачів до даних і ресурсів розподіленої інформаційної системи кешування даних виконується на основі принципу, згідно з яким користувач одержує лише ті повноваження, які у мінімальному обсязі потрібні йому для виконання своїх обов'язків.

Доцільно виділити наступні категорії користувачів, що мають різні сукупності повноважень по доступу:

– користувачі, які мають доступ до інформації з грифом "таємно";

– користувачі, які мають доступ до інформації з грифом "конфіденційно";

– користувачі, які мають доступ до інформації з грифом "нетаємно".

Контроль і облік доступу до ресурсів робочих станцій та мережевих ресурсів розподіленої інформаційної системи кешування даних реалізується за допомогою комплексу організаційно-програмних заходів та технічних засобів системи захисту інформації розподіленої системи кешування даних.

Ідентифікація/автентифікація використовуються для підтвердження дійсності суб'єкта, забезпечення його роботи в системі, і визначення законності прав суб'єкта на об'єкт або на певні дії з ним. У процесі ідентифікації елементи системи розпізнаються за допомогою заздалегідь визначеного ідентифікатора (кожен суб'єкт чи об'єкт системи однозначно ідентифікується). В процесі автентифікації, яка обов'язково здійснюється перед дозволом на доступ, перевіряється дійсність ідентифікації елементу системи, а також перевіряються цілісність та авторство даних при їхньому збереженні або передачі для запобігання несанкціонованої модифікації. Подальші взаємодії з системою можливі тільки після успішної ідентифікації/автентифікації. Інформація щодо ідентифікації/автентифікації зберігається таким чином, що тільки адміністратор системи захисту інформації має до неї доступ.

Висновки до розділу 1

У цьому розділі розглянуто основні існуючі системи кешування даних, їх можливий набір функціональностей, переваги та недоліки. Описані основні принципи використання таких механізмів доступу, цілі і задачі в яких вони використовуються.

Також, було розглянуто основні об’єкти та суб’єкти інформаційного доступу в розподілених системах кешування даних, іх роль у розмежуванні доступу в таких системах.

Визначено, що суб’єктами доступу в розподіленій інформаційній системі можуть бути адміністратори, прямі користувачі системи та технічні облікові користувачі для інтеграції з іншими системами. Об’єктами доступу в таких системах виступають самі Кеші(аналоги таблиць в базах даних) а також записи в кешах (аналоги записів в таблицях баз даних).

В останньому пункті були розглянуті основні проблеми розмежування доступу в існуючих системах кешування даних та основні ключові організаційні дії які зачасту використовують в системах де потребується таке розмежування

2 АНАЛІЗ ФУНКЦІОНАЛЬНОСТІ СИСТЕМИ РОЗМЕЖУВАННЯ ДОСТУПОМ

2.1 Існуючі методи розмежування доступу

Розмежування доступу – сукупність процедур, що реалізують перевірку запитів на доступ і оцінку на підставі правил розмежування доступу. Правила розмежування доступу – частина політики безпеки, що регламентує правила доступу користувачів і процесів до пасивних об’єктів.

При розгляді взаємодії двох об’єктів комп’ютерної системи, що виступають як приймальники або джерела інформації, слід виділити пасивний об’єкт, над яким виконується операція, і активний об’єкт, який виконує або ініціює цю операцію.

Коли користувачі або процеси намагаються одержати доступ до пасивних об'єктів, механізми, що реалізують керування доступом, на підставі політики безпеки і перевірки атрибутів доступу можуть “прийняти рішення” про легальність запиту. Використовуючи набір атрибутів доступу відповідно до прийнятої політики безпеки, можна реалізувати потрібне керування доступом.

2.1.1 Дискреційні методи розмежування доступу

Модель, що заснована на дискреційних методах характеризується розмежуванням доступу між названими суб’єктами та об’єктами. Суб’єкт з певним правом доступу може передавати це право будь-якому іншому суб'єкту. Для кожної пари (суб’єкт – об’єкт) має бути задано явне і недвозначне перерахування допустимих типів доступу (читання, писання тощо), які є санкціонованими для цього суб’єкта до відповідного ресурсу (об’єкта). Кожен об’єкт системи має прив’язаний до нього суб’єкт, що називається власником. При цьому власник встановлює права доступу до об’єкта[9].

Система має єдиний виділений привілейований суб’єкт, який уповноважений встановлювати права власності для всіх інших суб’єктів системи. Можливо і змішані варіанти побудови, коли одночасно в системі присутні як власники, які встановлюють права доступу до своїх об’єктів, так і привілейовані субє’кти, що мають можливість змінити права для будь-якого

об’єкта, або зміни його власника. Такий змішаний варіант реалізується в більшості операційних систем.

Дискреційне керування доступом є основною реалізацією політики розмежованого доступу до ресурсів при обробці конфіденційних відомостей відовідно до вимог до системи захисту інформації (рисунок 2.1).

Рисунок 2.1 – Організація доступу в інформаційних системах з використанням дискреційних методів

Дискреційна політика безпеки передбачає, що права доступу суб’єктів до кожного окремого об’єкта системи можуть бути довільним чином обмежені на основі деякого зовнішнього по відношенню до системи правила. Також дискреційна політика безпеки вимагає ідентифікованості всіх суб’єктів та об’єктів системи.

Основний елемент дискреційного розмежування доступу є матриця доступу. Матриця доступу – матриця D розміром $|S|$ на $|O|$, рядки якої відповідають суб’єктам, а стовпчики – об’єктам. Кожний елемент матриці доступу $D[s,o]$ R визначає права доступу суб’єкта s до об’єкта o, де R– множина прав доступу.

Суб’єкти s є активними сутностями, здебільшого це користувачі або процеси. Об’єкти o є пасивними сутностями, що потребують захисту. Це можуть бути, наприклад, файли, записи баз даних, сегменти оперативної пам’яті. У деяких операціях доступу суб’єкти можуть виступати як пасивні

сутності, до яких здійснюють доступ інші суб'єкти, тому множини S та O знаходяться у відповідності S до O.

У матриці доступу D кожен рядок відповідає певному суб'єктові s, а кожен стовпчик – об'єктові o (рис. 2.2). Елементом матриці $D[s,o]$ є множина прав доступу, або повноважень суб'єкта s стосовно об'єкта o. Ці права, власне, і визначають, що може робити суб'єкт з об'єктом.

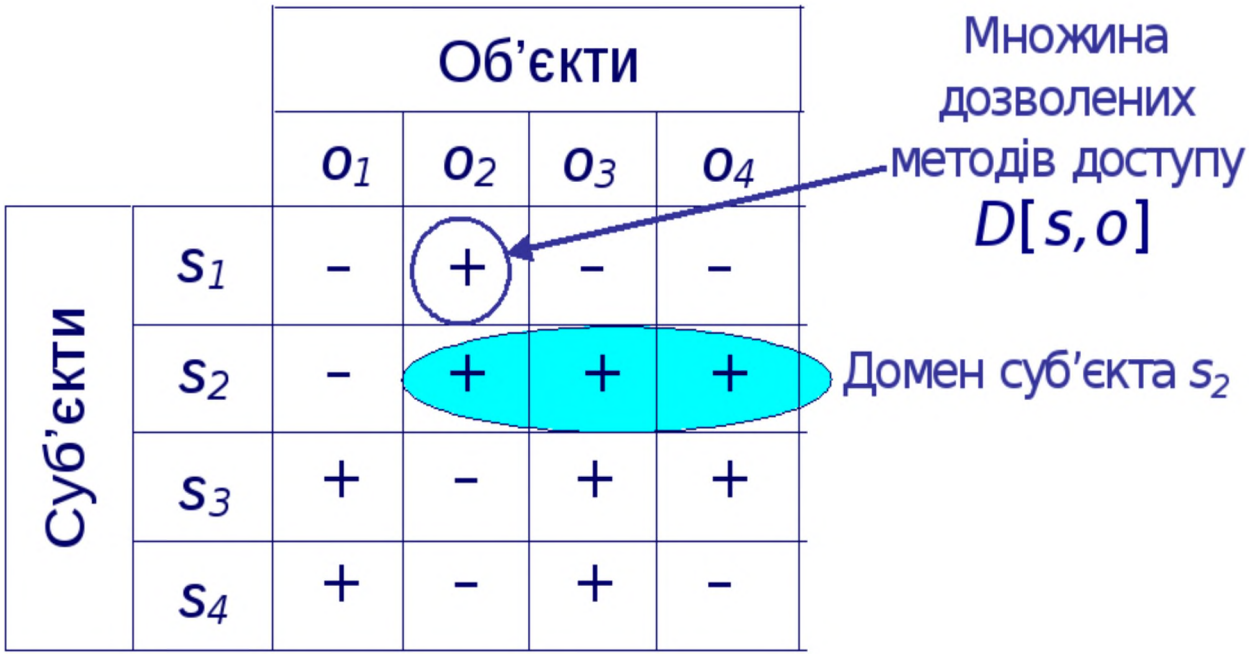

Рисунок 2.2 – Матриця доступу в дискреційній моделі

Підмножина об'єктів, до яких визначений суб'єкт має певні права доступу, називається доменом цього суб'єкта.

Матриця доступу дуже розріджена і неефективна з точки зору використання пам'яті. Тому, замість неї у реальних системах використовуються списки доступу та списки повноважень. Список доступу асоціюється з кожним захищеним об'єктом в системі і містить в собі ідентифікатори різних суб'єктів разом з їхніми правами доступу до цього об'єкту (список доступу, таким чином, описує стовпчик матриці доступу). На відміну від списку доступу, список повноважень асоціюється з кожним суб'єктом в системі і містить в собі ідентифікатори об'єктів разом з повноваженнями цього суб'єкта стосовно цих

об’єктів. Список повноважень, таким чином, відповідає рядкові матриці доступу.

При використанні матричної моделі безпеки політика безпеки інформації містить не лише саму матрицю доступу, яка описує правила розмежування доступу, але й обмеження, що накладаються на спосіб модифікації матриці доступу. Так, у випадку довірчого керування доступом всі права на зміну прав доступу до об’єкта надаються (довіряються) суб’єктові, що є власником цього об’єкта. Тобто, якщо список прав доступу суб’єкта *s* до об’єкта *o* містить право власника, то суб’єкт *s* отримує повний контроль над стовпчиком матриці доступу, що відповідає *o*. У випадку адміністративного керування доступом система захисту визначає можливість доступу суб’єктів до об’єктів, базуючись на мітках або атрибутах доступу, які може встановлювати або змінювати лише спеціально призначений адміністратор.

Перевагою дискреційної політики безпеки є проста реалізація системи розмежування доступу і, як наслідок, її широка розповсюдженість на практиці. Разом з цим ця політика вважається недосконалою із-за низки суттєвих недоліків.

Недоліками цієї політики є статичність правил розмежування доступу, які не враховують динаміку змін стану комп’ютерної системи. Також, у випадку використання дискреційної політики безпеки, при доступі суб’єкта до об’єкта кожного разу необхідно визначати права доступу та аналізувати їх вплив на безпеку системи, що знижує її прозорість. У загальному випадку для систем дискреційного політики задача перевірки безпеки є алгоритмічно нерозв’язною. Доведення того факту, що система, у якій реалізовано дискреційну політику, є захищеною у заданому стані, має бути проведено для кожної конкретної системи і для кожного стану цієї системи.

2.1.2 Мандатні методи розмежування доступом

Для реалізації цього принципу кожному суб'єкту і об'єкту повинні зіставлятися класифікаційні мітки, що відображають місце даного суб’єкта (об’єкта) у відповідній ієрархії. За допомогою цих міток суб’єктам і об’єктам

призначаються класифікаційні рівні (рівні уразливості, категорії секретності тощо), які є комбінаціями ієрархічних і неієрархічних категорій. Ці мітки мають служити основою мандатного принципу розмежування доступу. Комплекс засобів захисту (КСЗ) при введенні нових даних в систему має запитувати і отримувати від санкціонованого користувача класифікаційні мітки цих даних. При санкціонованому занесенні в список користувачів нового суб'єкта має здійснюватися присвоєння йому класифікаційних міток.

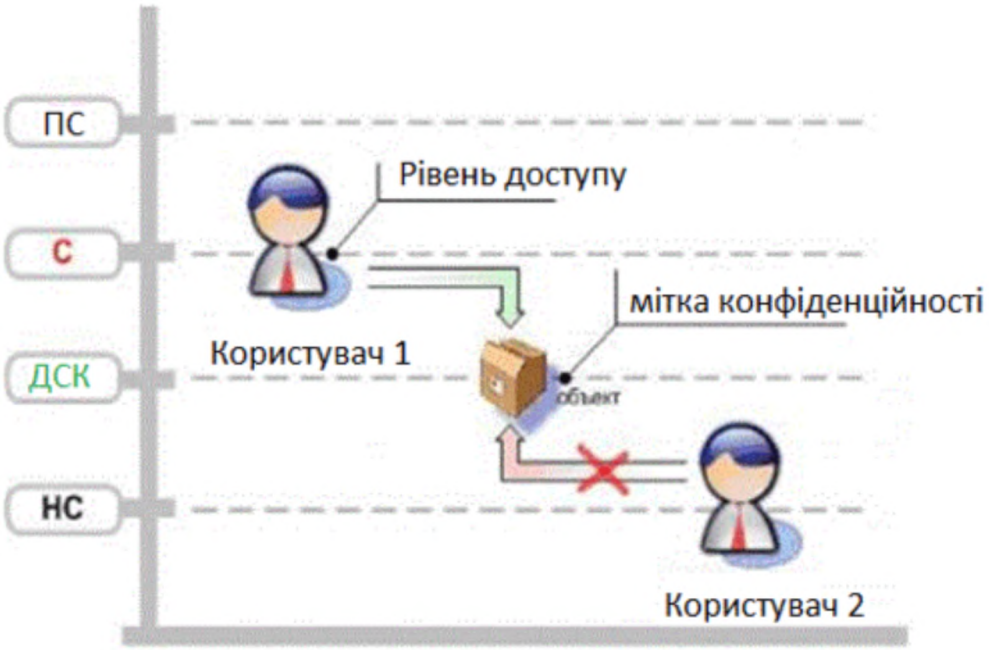

Рисунок 2.3 – Приклад реалізації мандатної моделі доступу

Зовнішні класифікаційні мітки (суб'єктів, об'єктів) повинні точно відповідати внутрішнім мітках (всередині КСЗ). КСЗ має реалізовувати мандатний принцип контролю доступу стосовно до всіх об'єктів при явному і прихованому доступі з боку будь-якого із суб'єктів. Суб'єкт може читати об'єкт, тільки якщо ієрархічна класифікація суб'єктом не менше, ніж ієрархічна класифікація об'єкта, і якщо не ієрархічні категорії суб'єкта включають в себе всі ієрархічні категорії об'єкта.

Суб'єкт здійснює запис в об'єкт, тільки якщо класифікаційний рівень суб'єкта максимум, ніж класифікаційний рівень об'єкта, і всі ієрархічні категорії суб'єкта включаються в неієрархічні категорії об'єкта.

Реалізація мандатних правил розмежування доступу має передбачати можливості супроводу зміни класифікаційних рівнів суб'єктів і об'єктів спеціально виділеними суб'єктами. Повинен бути реалізований механізм доступу, що здійснює перехоплення всіх звернень суб'єктів до об'єктів, а також розмежування доступу відповідно до заданого принципом розмежування доступу. При цьому рішення про санкціонованості запиту на доступ має прийматися тільки при одночасному його вирішенні мандатних правил розмежування доступу. Таким чином, має контролюватися не тільки одиночний акт доступу, а й потоки інформації.

Мандатна політика безпеки передбачає існування таких умов:

– визначеності решітки конфіденційності інформації;

– надання кожному об'єкту системи рівня конфіденційності, який визначає цінність інформації, яка містися в ньому;

– задоволення вимоги ідентифікованості всіх суб'єктів та об'єктів системи.

Головне завдання мандатної політики безпеки – запобігання витоку інформації від об'єктів з високим рівнем доступу до об'єктів з низьким рівнем доступу.

Найбільш розповсюдженим описом мандатної політики безпеки є модель Белла – ЛаПадула[5]. Ця модель гарантує, що суб'єкт може ознайомитись з інформацією лише тоді, коли має на це достатні повноваження, і будь-який суб'єкт (крім адміністратора, якому надано повноваження встановлювати рівні конфіденційності об'єктів) ніяким чином не може здійснити перенесення даних з об'єкта з вищим рівнем конфіденційності у об'єкт з більш низьким рівнем конфіденційності. Отже, це – модель конфіденційності.

Перевагами мандатної політики безпеки є те, що її правила більш прозорі та зрозумілі у порівнянні з правилами дискреційної політики. Системи, що побудовані на основі цієї політики безпеки також більш надійні у порівнянні із системами, які побудовані на дискреційній політиці безпеки.

У загальному випадку для систем мандатної політики задача перевірки безпеки є алгоритмічно розв'язною і безпека систем мандатної політики є доведеною.

Недоліками мандатної політики безпеки є значні вимоги до обчислювальних ресурсів та складність у практичної реалізації.

2.1.3 Рольовий метод розмежування доступом

Основною ідеєю управління доступом на основі ролей є ідея про зв'язування дозволів доступу до ролей, які призначаються кожному користувачеві. Ця ідея виникла одночасно з появою багатокористувацьких систем. Однак до недавнього часу дослідники мало звертали увагу на цей принцип.

Рольове розмежування доступу являє собою розвиток політики дискреційного розмежування доступу, при цьому права доступу суб'єктів системи на об'єкти групуються з урахуванням специфіки їх застосування, утворюючи ролі.

Таке розмежування доступу є складовою багатьох сучасних комп'ютерних систем. Як правило, цей підхід застосовується в системах захисту ситем управління базами даних (СУБД), а окремі елементи реалізуються в мережевих операційних системах.

Завдання ролей дає змогу визначити більш чіткі і зрозумілі для користувачів комп'ютерної системи правила розмежування доступу. При цьому такий підхід часто використовується в системах, для користувачів яким чітко визначені їх посадові повноваження та обов'язки.

Роль є сукупністю прав доступу на об'єкти комп'ютерної системи, однак рольове розмежування аж ніяк не є окремим випадком дискреційного розмежування, так як її правила визначають порядок надання прав доступу суб'єктам комп'ютерної системи в залежності від сесії його роботи і від наявних (або відсутніх) у нього ролей в кожен момент часу, що є характерним для систем мандатного розмежування доступу. З іншого боку, правила

рольового розмежування доступу є більш гнучкими, ніж при мандатному підході до розмежування.

Якщо підвести підсумок, то у кожної з перерахованих систем є свої переваги, проте ключовим є те, що жодна з описаних моделеї не стоїть на місці, а динамічно розвивається. Прихильники є у кожної з них, однак, об'єктивно подивившись на речі, важко віддати перевагу якійсь одній системі. Вони просто різні і призначені для різних цілей.

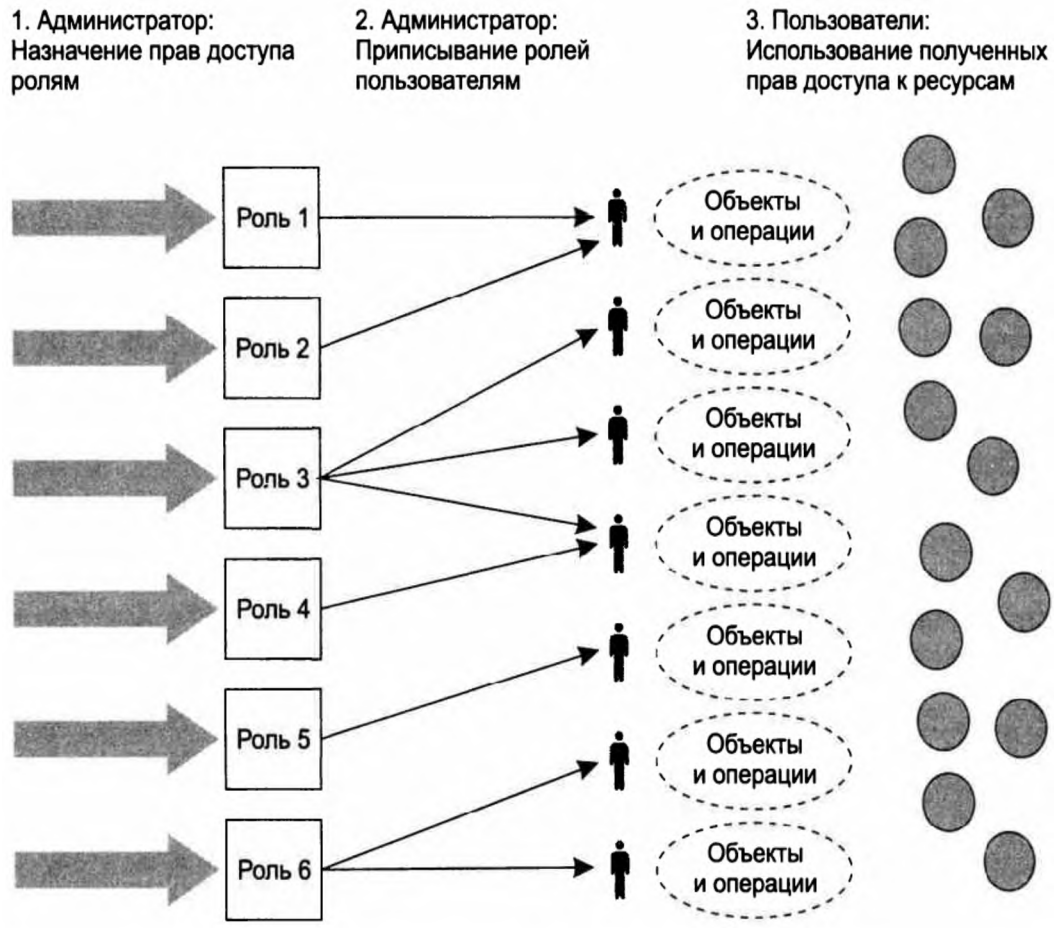

Рисунок 2.4 – Рольове розмежування доступом

Формування ролей покликане визначити чіткі і зрозумілі для користувачів комп'ютерної системи правила розмежування доступу. Рольове розмежування доступу дає змогу реалізувати гнучкі та динамічно змінні в процесі функціонування комп'ютерної системи правила розмежування доступу. Таке розмежування доступу є складовою багатьох сучасних комп'ютерних систем. Як правило, цей підхід застосовується в системах захисту СУБД, а окремі елементи реалізуються в мережевих операційних системах. Рольовий підхід

часто використовується в системах, для користувачів для яких чітко визначено коло їх посадових повноважень і обов'язків. Незважаючи на те, що роль є сукупністю прав доступу на об'єкти комп'ютерної системи, рольове керування доступом аж ніяк не є окремим випадком вибіркового управління доступом, так як його правила визначають порядок надання доступу суб'єктам комп'ютерної системи в залежності від наявних (або відсутніх) у нього ролей в кожен момент часу, що є характерним для систем мандатного керування доступом. З іншого боку, правила рольового розмежування доступу є більш гнучкими, ніж при мандатному підході до розмежування. Так як привілеї не призначаються користувачам безпосередньо і отримуються ними тільки через свою роль (або ролі), управління індивідуальними правами користувача по суті зводиться до призначення йому ролей. Це спрощує такі операції, як додавання користувача або зміна підрозділу користувачем.

Для визначення моделі RBAC визначаються такі умови:

1) S = Суб'єкт = Людина або автоматизований агент (множина користувачів);

2) R = Роль = Робоча функція або назва, яка визначається на рівні авторизації (множина ролей);

3) P = Дозволи = Затвердження режиму доступу до ресурсу (множина прав доступу на об'єкти системи);

4) SE = Сесія = Відповідність між S, R та / або P

5) SA = Призначення суб'єкта

6) PA: R → 2p — функція, що визначає для кожної ролі множину прав доступу; при цьому для кожного p ∈ P існує r ∈ R така, що p ∈ PA (r);

7) RH = Частково впорядкована ієрархія ролей. RH може бути ще записана так:

– один суб'єкт може мати кілька ролей;

– одну роль можуть мати декілька суб'єктів;

– одна роль може мати кілька дозволів;

– один дозвіл може належати кільком ролям.

Ролі призначаються суб’єктам, внаслідок чого суб’єкти отримують ті чи інші дозволи через ролі. Рольова модель вимагає саме такого призначення, а не прямого призначення дозволів суб’єктам, інакше це призводить до складно контрольованих відносин між суб’єктами і дозволами. На можливість успадкування дозволів від протилежних ролей накладається обмежувальна норма, яка дає змогу досягти належного поділу режимів. Наприклад, одній і тій же особі може бути не дозволено створити обліковий запис для когось, а потім авторизуватися під цим обліковим записом.

Використовуючи нотацію теорії множин:

1) PA <= P x R, при цьому дозволи призначаються зв’язкам ролей у відношені “багато до багатьох”.

2) SA <= S x A, при цьому суб'єкти призначаються зв'язкам ролей і суб'єктів у відношені “багато до багатьох”.

3) RH <= R x R

Позначення: x ≥ y означає, що x успадковує дозволи y. Суб’єкт може мати множину одночасних сесій з різними дозволами. Технологія керування доступом на основі ролей досить гнучка і сильна, щоб змоделювати як вибіркове керування доступом, так і мандатне керування доступом. Ролі створюються всередині організації для різних робочих функцій. Певним ролям присвоюються повноваження для виконання тих чи інших операцій. Штатним співробітникам (або іншим користувачам системи) призначаються фіксовані ролі, через які вони отримують відповідні привілеї для виконання фіксованих системних функцій. Рольова модель може давати привілеї на складні операції зі складовими даним, а також на атомарні операції з низькорівневими об’єктами даних. Наприклад, список контролю доступу може надати або позбавити права запису у якомусь системному файлу, але він не може обмежити те, яким чином цей файл може бути змінений. Система, заснована на Рольовій моделі, дає змогу створити таку операцію як відкриття “кредиту” у фінансовому додатку або заповнення запису “тест на рівень цукру в крові” в медичному додатку.

В організаціях з різнорідною IT-інфраструктурою, що містять десятки і сотні систем і додатків, допомагає використання ієрархії ролей і успадкування привілеїв. Без цього використання рольової моделі стає вкрай заплутаним. Для великих систем з сотнями ролей, тисячами користувачів і мільйонами дозволів, управління ролями, користувачами, дозволами і їх взаємозв'язками є складним завданням, яке неможливо здійснити малою групою адміністраторів безпеки. Привабливою можливістю є використання самої рольової моделі для сприяння децентралізованому управлінню ролями.

Рольова модель широко використовується для управління призначеними для користувача привілеями в межах єдиної системи або єдиного додатку.

2.2 Задача розмежування доступу в монолітних системах кешування даних

Отже, розглянувши основні методи розмежування доступу в інформаційних та не інформаційних системах, перейдемо к формулюванню проблеми розмежування доступу в монолітній системі кешування даних.

Розглянемо доступ суб'єкта до об'єкту інформаційної системи кешування даних.

У ролі суб'єкта виступає користувач системи, що хоче отримати інформацію з об'єкта кеша. В ролі об'єктів в монолітній системі кешування даних виступають кеш так запис у цьому кешу.

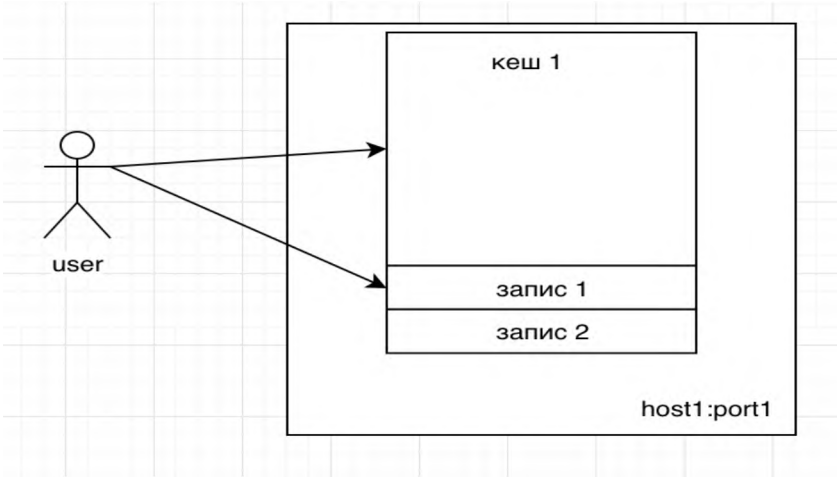

Рисунок 2.5 - Обєкти та субєкти доступу в монолітній системі кешування даних

Типовий хід дій під час роботи користувача з системою кешування даних без аутентифікації та розмежування доступу:

1) Суб’єкт формулює запит до системи кешування даних(put, get, update)
2) Система приймає запит та обробляє його, повертає результат суб’єкту

Типовий хід дій під час роботи користувача з системою кешування даних та з аутентифікацією включаючи розмежування доступу:

1) Суб’єкт формулює запит до системи кешування даних(put, get, update)
2) Система перевіряє рівень доступу суб’єкта та його права для доступу до вибраного об’єкту системи кешування даних
3) Якщо рівень та права суб’єкту дозволяють йому зробити дію над об’єктом - система приймає запит та обробляє його, повертає результат суб’єкту
4) Якщо рівень та права суб’єкту не дозволяють йому зробити дію над об’єктом - система повертає помилку або блокує дії суб’єкта

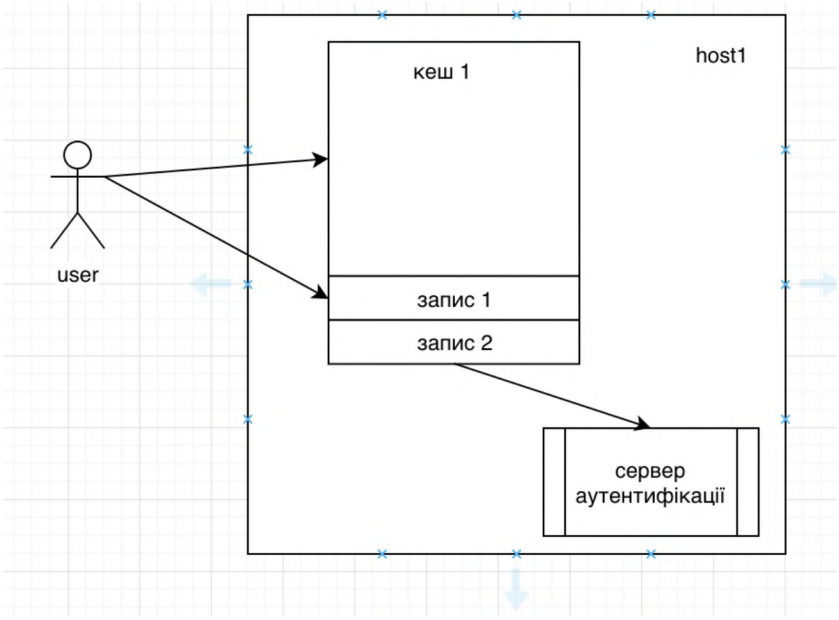

Рисунок 2.6 – Об’єкти та суб’єкти доступу в монолітній системі кешування даних використовуючи сервер аутентифікації.

Як можемо побачити, з’являється еще один етап взаємодії системи. Також з’являється ще один учасник - сервер аутентифікації. В контексті монолітної системи, це може бути механізм самої системи кешування даних, а також, в цій

ролі може виступати зовнішня система. В кейсі зовнішньої системи, система кешування даних може представляти механізм інтеграції для взаємодії з будь-яким сервером зовнішньої авторизації та надання прав.

Під час появи процесу розмежування доступу в системі, з'являється потреба в реалізації механізмів аутентифікації та авторизації, це обумовлює використання механізмів серверу аутентифікації та авторизації. Даний механізм може бути вузьким місцем в архітектурі системи в якій використовується система кешування даних.

Наприклад, якщо система аутентифікації буде зовнішньою, то у випадку її відмови, суб'єкт не зможе в повній мірі взаємодіяти з системою кешування даних.

Якщо система кешування даних буде мати свій вмонтований механізм доступу для аутентифікації та авторизації, то в випадку інтеграції з іншими системами, виникне питання синхронізації механіхмів серверу аутентифікації. Тому в таких випадках має значення можливості використання внутрішнього серверу аутентифікації, для випадку коли взаємодіє з іншими серверами аутентифікації не потрібна. В загалі, такий кейс зустрічається не часто (в реальному житті в великих і складних системах де потребується кешування, досить багато сервісів з якими приходиться взаємодіяти). В таких випадках потрібно використовувати можливість інтеграції з іншими серверами аутентифікації(якщо такі потрібні).

Також, може виникнути проблема під час якої монолітний сервер кешу не буде відповідати, при цьому клієнт не зможе отримати йому потрібні дані.

Часто, в складних системах використовують один зовнішній сервер аутентифікації та надання прав, тому, для систем кешування даних, що підтримують розмежування доступу, дуже важливо мати такий механізм інтеграції.

2.3 Задача розмежування доступу в розподілених системах кешування даних

Отже, розглянувши основні методи розмежування доступу в інформаційних та не інформаційних системах, а також розглянувши задачу

розмежування доступу в монолітній системі кешування даних перейдемо к формулюванню проблеми розмежування доступу в розподілених системі кешування даних.

Розглянемо доступ суб'єкта до об'єкту інформаційної системи кешування даних. В ролі суб'єкта, також виступає користувач системи, що хоче отримати інформацію з об'єкта кеша. В ролі обєктів в розподілених системі кешування даних виступають кеш так запис у цьому кешу.

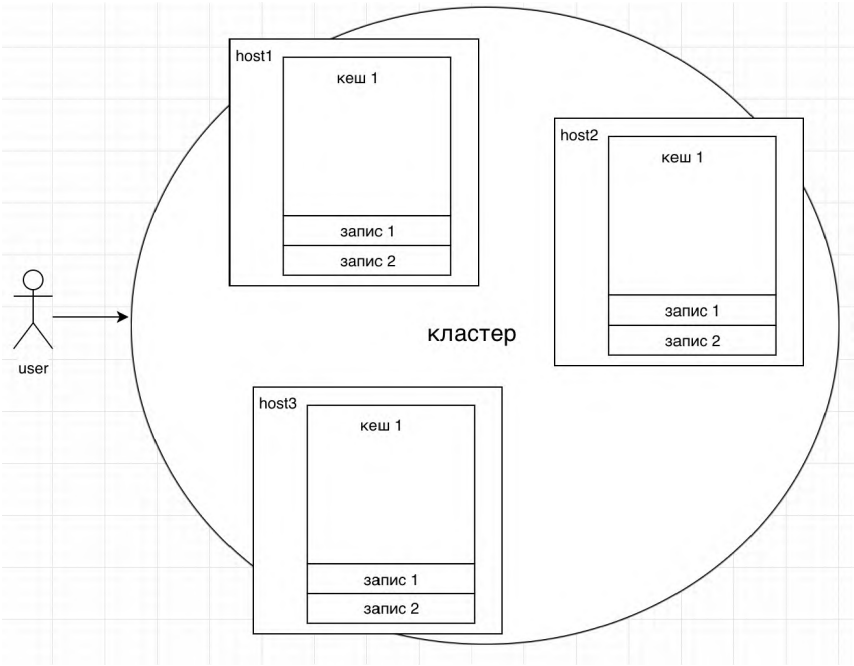

Рисунок 2.7 - Обєкти та субєкти доступу в розподіленій системі кешування даних

Типовий хід дій під час роботи користувача з системою кешування даних без аутентифікації та розмежування доступу:

1) Суб'єкт формулює запит до системи кешування даних(put, get, update), часто це один із хостів розподіленої системи;
2) Система приймає запит та обробляє його, обирає хост, на якій знаходиться потрібна інформація та передає запит на неї. Вибір потрібного хоста вирішується геш-функцією від ключа запиту. Цей механізм пошуку по шардам кеша можна замінити на свій;
3) Цільовий хост, на якому знаходяться потрібні дані обробляє запит та повертає дані на той хост з якого цей запит був виконаний;
4) Цільовий хост повертає дані кінцевому користувачеві.

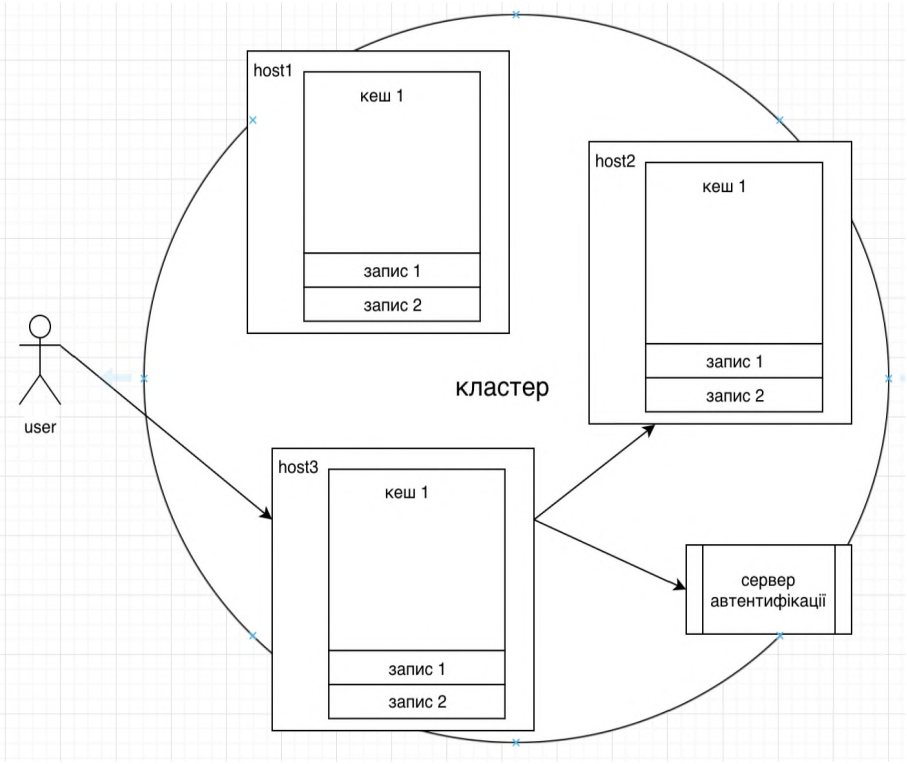

Рисунок 2.8 - Обєкти та субєкти доступу в розподіленій системі кешування даних використовуючи сервер автентифікації

Типовий хід дій під час роботи користувача з системою кешування даних з сервером аутентифікації та розмежування доступу:

1) Суб'єкт формулює запит до системи кешування даних(put, get, update), зачасту це один із хостів розподіленої системи;
2) Система перевіряє рівень доступу суб'єкта та його права для доступу до вибраного об'єкту системи кешування даних використовуючи сервер аутентифікації;

3) Якщо рівень та права суб'єкту не дозволяють йому зробити дію над об'єктом - система повертає помилку або блокує дії суб'єкта;

4) Система приймає запит та обробляє його, обирає хост, на якій знаходиться потрібна інформація та передає запит на неї.Часто вибір потрібного хоста вибирається геш-функцією від ключа запиту. Цей механізм пошуку по шардам кеша можна замінити на свій;

5) Цільовий хост, на якому знаходяться потрібні дані, обробляє запит, та повертає дані на той хост з якого цей запит був виконаний;

6) Цільовий хост повертає дані кінцевому користувачеві.

У випадку розподіленої системи, розміщення серверу аутентифікації в системі кешування даних не має великого значення - система розподілена та механізм зберігання даних для виконання доступу не дуже логічний. Система кешування даних може використовувати ці дані, та зберігає їх в своїх системних кешах для швидкого доступу та моментальному виконанню перевірки прав.

В такому підході можуть бути також деякі проблеми, а саме:

1) В випадку коли сервер аутентифікації не відповідає, система не зможе повернути дані з кешу кінцевому користувачу. Дане питання можна будет вирішити завдяки використанню реплікованого серверу автентифікації;
2) В випадку відвалу/відключення ноди з кластеру, у клієнта не буде можливості отримати дані що лежать на цій ноді. Дану проблему можна вирішити завдяки використання реплік для розподілених даних по кластеру. В випадку, коли число реплік дорівнює 3, кластер зможе вижити та не загубити дані в випадку коли 2 ноди вийдуть з кластеру.

Також, такий варіант побудови механізму кеша кращий за монолітний, так як в випадку відключення однієї ноди, кластер кешу буде і далі працювати в штатному режимі для користувача та виконувати всі запити.

Отже, проаналізувавши монолітний та розподілений метод розмежування доступу в системі кешування даних, було показано, що підхід реалізації розмежування не залежить від кількості хостів в кластері кешу. Тому розмежування доступу може бути реалізоване в контексті систем кешування даних використовуючи різні механізми авторизації в системі. Для систем кешування даних, не важливо, монолітна вона чи ні, схема реалізації розмежування будет ідентичною. Для цього у 3 розділі будуть розглянуто існуючі механізми розмежування доступу в контексті розподілених систем кешування даних, та буде підібраний найкращий варіант для кеш-таблиці рівня розмежування та рівня запису у ній.

2.4 Існуючі механізми розмежування та керування доступом в розподілених та монолітних системах кешування даних

На даний момент, як такого прямої реалізації розмежування доступу в розподілених системах кешування даних немає. Зачасту, провайдери систем дають досить примітивний механізм що просто не дозволяє взагалі виконати запит до системи. Ніякого розмежування на рівні кешів(аналогу таблиць в бд або кеш таблиць) та на рівні записів в кешах немає. Програмісти часто використовують саморобні рішення, що потребують більше часу та вимагають додаткового тестування. Таких підхід призводить до помилок, що в деяких сферах діяльності неприпустимо. Наприклад, банківська сфера. У випадку помилки, користувач системи може отримати доступ до даних іншого користувача. Така помилка не може бути припустимою, та в деяких випадках може призвести до суворого покарання судом.

Нижче, на малюнку зображена типова архітектура системи, що використовує кеш та систему аутентифікації:

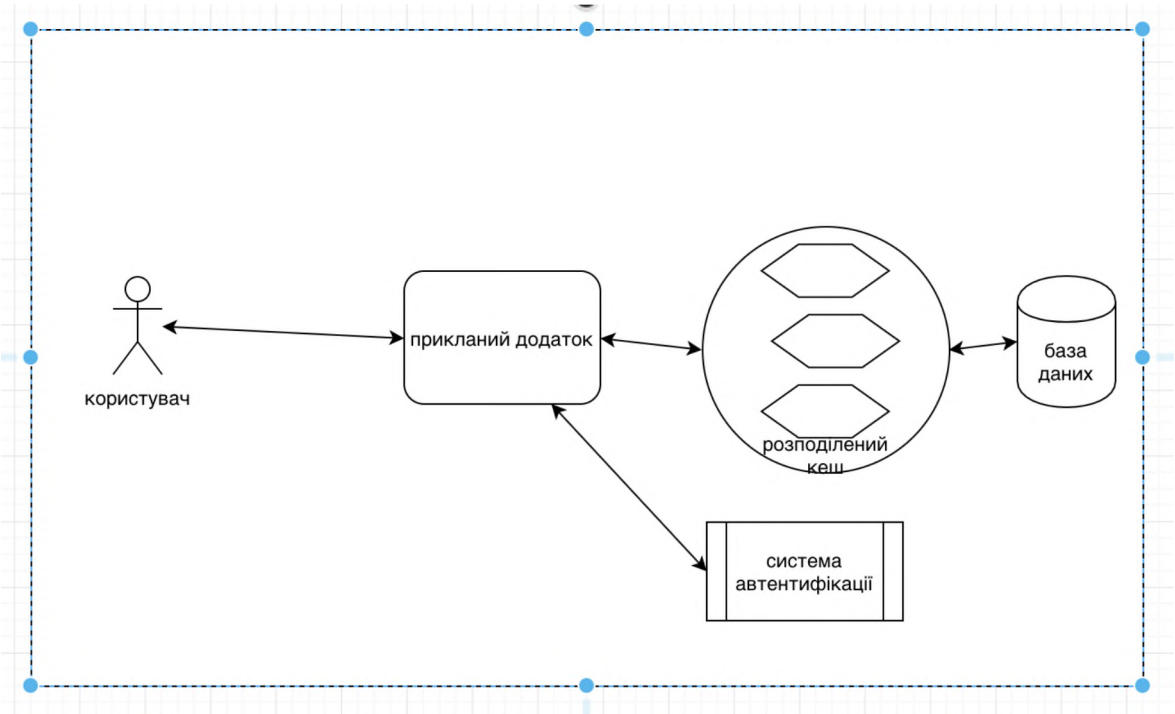

Рисунок 2.9 - Типова архітектура системи, що використовує кеш та систему аутентифікації

Як видно з малюнку, вся логіка реалізації механізму розмежування доступом викона на прикладному рівні, де вже і виконується саме розмежування доступу.

Також, у випадку коли розподілений кеш ніяк не захищений зовні програмними засобами, у неревелантної особи може з'явитися можливість підключитися до кластеру для доступу потрібної йому інформації.

Висновки до розділу 2

Отже, у цьому розділі було розглянуто основні методи розмежування доступу в інформаційних системах, їх переваги та недоліки. Використовуючи ці методи, було поставлено задачу розмежування доступу для монолітної та розподіленої системи кешування даних. З проаналізованих дій, було виявлено, що механізм реалізації розподілення доступу для монолітної системи кешування даних ідентичний та може бути перевикористаним.

Проаналізувавши монолітний та розподілені варіанти розмежування доступу, було виявлено, що використання розподіленого варіанту є важчим, але в свою чергу допомагає уникнути проблем роботи кластеру під час відказу одного сервера с кластеру.

Також, було проаналізовано існуючі підходи реалізації такого розмежування та автентифікації. Було виявлено, що як такої практики використання існуючих механізмів немає - на цей час вони являються не гнучкими і не адаптивними під різні бізнес задачі.

У наступному розділі буде розглянуто більш розумні підходи розмежування доступу в системах кешування даних, що можуть бути використані для різних прикладних кейсів задач.

3 ПОБУДОВА МОДЕЛІ РОЗМЕЖУВАННЯ ДОСТУПУ В РОЗПОДІЛЕНИХ СИСТЕМАХ КЕШУВАННЯ ДАНИХ

3.1 Розробка моделі дискреційного методу

Дискреційний контроль доступу управляє доступом суб'єктів до об'єктів базуючись на інформації суб'єкта і списку доступу об'єкта, що містить набір суб'єктів (або груп суб'єктів) і асоційованих з ними типів доступу (наприклад читання, запис). При запиті доступу до об'єкта, система шукає ідентифікатор суб'єкта в списку прав доступу об'єкта і надає йлму доступ якщо суб'єкт присутній в списку і дозволений тип доступу включає необхідний тип прав. Інакше доступ не надається.

3.1.1 Розмежування доступу суб’єктів та об’єктів розподіленої системи кешування даних використовуючи дискреційний метод

Описуючи механізм доступу суб’єктів до обєктів в розподіленій системі кешування даних, потрібно розглянути:

1) підхід доступу користувачів системи до кешів з інформацією
2) підхід доступу користувачів системи до записів а цих кешах
3) підхід передачі та надання доступу від одного суб’єкта до іншого

Під час роботи з кешом, для дискреційної моделі можна виділити основні операції що дозволяюся суб’єкту над кешом, а саме:

1) запис/оновлення
2) читання
3) перевірка на існування
4) створення кешу
5) надавання/відкликання прав на взаємодію з кешом

Для роботи на рівні записів в кеші, для дискреційної моделі можна виділити основні операції, а саме:

1) перевірка на існування
2) запис/оновлення
3) читання

4) надавання/відкликання прав на взаємодію з записом

Отже, проаналізуємо детально підхіл доступу користувачів системи до кешів з інформацією, схема дії користувача системи, що вже має доступ до читання/ писання деякого кеша:

1) користувач виконує запрос в розподілену систему кешування даних безпосередньо або через іншу систему, що використовує кеш для швидкого доступу критично важливої інформації;
2) під час запиту, система використовує та звану “матрицю доступу”, в якій зберігається вся інформація про розмежування;
3) якщо копія для швидкого доступу до ресурсів серверу автентифікації не присутня в сервері в якому виконується запит, система виконує один додатковий запит в сервер автентифікації для отримання інформації доступу. Наступного разу, для даного користувача дані серверу автентифікації беруться з локального серверу(а саме з системного кешу), що пришвидшує доступ до ресурсу;
4) якщо привілегій достатньо - успішно повертається потрібна інформація.

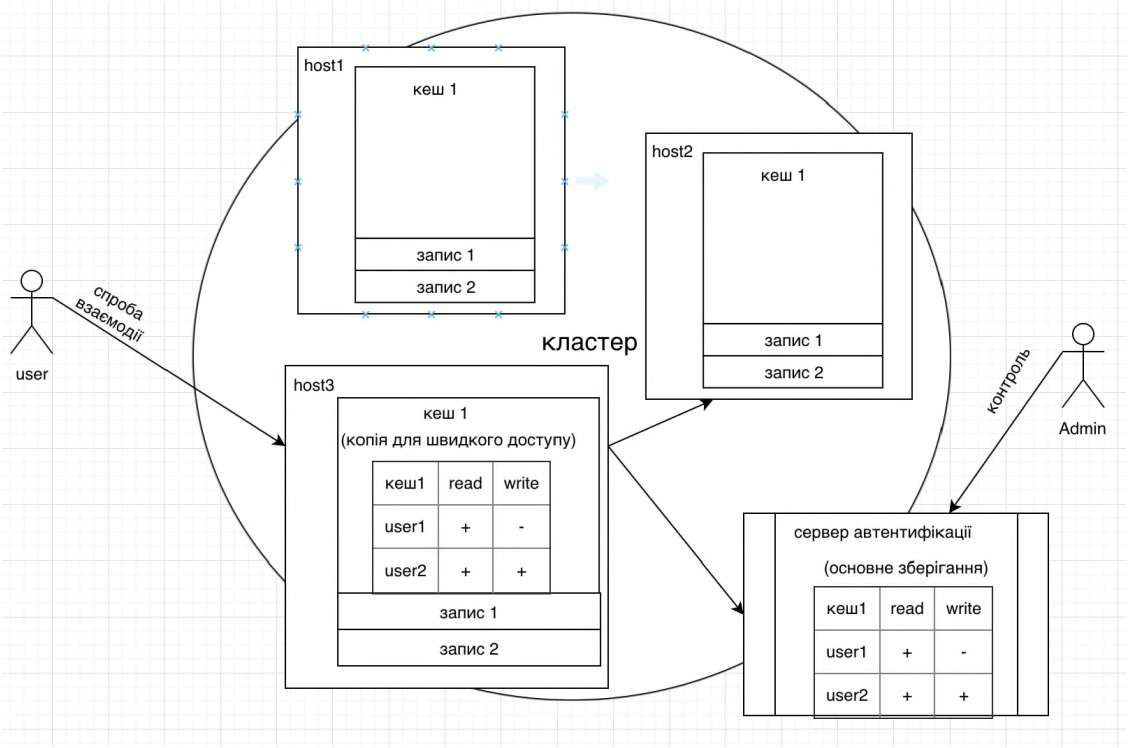

Рисунок 3.1 - Дискреційних підхід організації доступу в розподіленій системі кешування даних на рівні кешей.

Передача прав доступу в такому підході виглядає досить просто. Користувач, що має право надавати доступ на деяку операцію над кешом, просто записує в потрібну колонку матриці доступу до потрібного користувача значання, що буде дозволяти йому виконувати привегілейовану операцію.

Тепер, розглянувши підхід доступу користувачів системи до кешів з інформацією, розглянемо доступ на рівні записів до цих кешів, схема дії користувача системи, що вже має доступ до читання/писання деякого кеша:

1) користувач виконує запрос в розподілену систему кешування даних безпосередньо або через іншу систему, що використовує кеш для швидкого доступу критично важливої інформації;
2) під час запиту до запису в кеші, система використовує використовує так звану “матрицю доступу”, в якій зберігається вся інформація про розмежування. Ця матриця доступу являється досить малою для рівня одного запису, та може бути збережена разом із самим записом в кеші;
3) якщо привілегій достатньо - успішно повертається потрібна інформація.

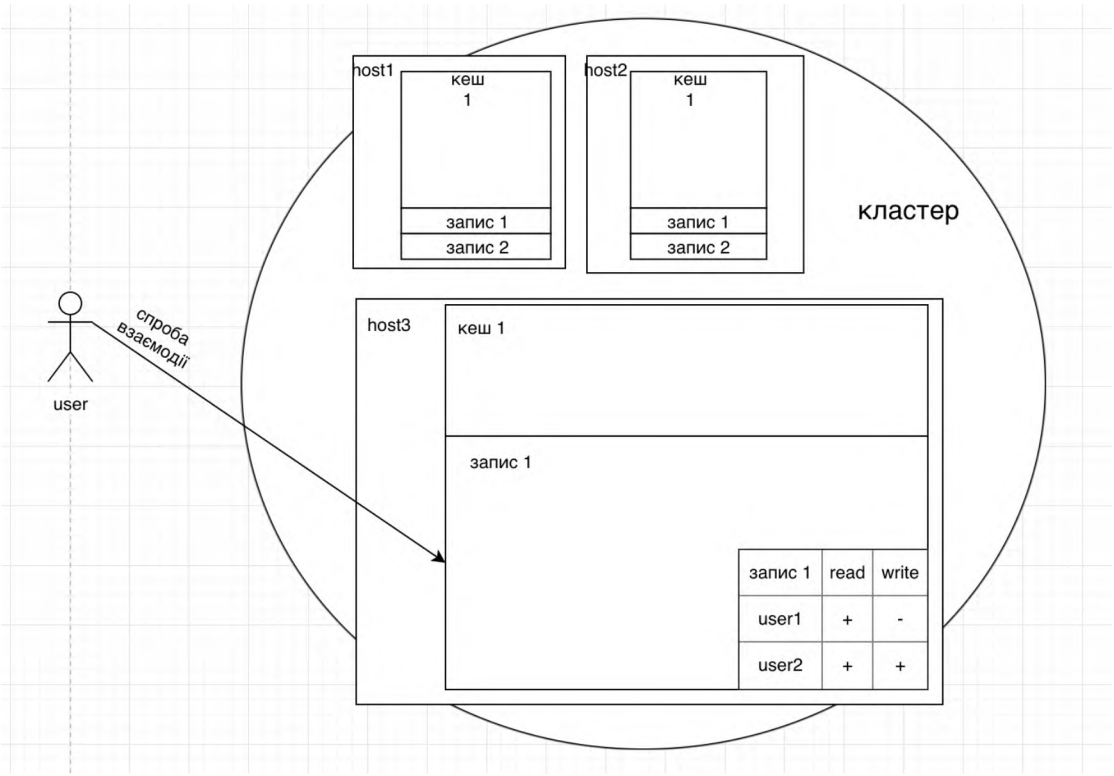

Рисунок 3.2 - Дискреційних підхід організації доступу в розподіленій системі кешування даних на рівні записів в кешах.

Передача прав доступу в такому підході розмежування доступу для записів в кеші виглядає досить просто. Користувач, що має право надавати доступ на деяку операцію над кешом, просто записує в потрібну колонку матриці доступу до потрібного користувача значання, що буде дозволяти йому виконувати привегілейовану йому операцію.

3.1.2 Переваги та недоліки дискреційного методу

Як структура даних, не обов'язково використовувати "матрицю" для зберігання такої інформації. Кількість кешів та користувачів буде зростати, тому досить важко буде зберігати всю інформацію про доступ, так як вона буде не гнучка та буде потребувати багато дискового місця. Також, якщо в системі з'явиться нова дія на яку можна буде виділити доступ, матриця доступу буде потребувати змін, тому дуже важливо зробити її гнучкою.

Також, дуже важливим моментом під час перевірки автентифікації та доступу користувача є кешування його мета-інформації по автентифікації. Такі дії знизять навантаження на сервер автентифікації (що зачасту є не масштабованим сервісом) та збільшать швидкість доступу до ресурсу кеша. Одним мінусом такого підходу є те, що для зберігання копій даних розмежування доступу в системах з великою кількістю користувачів буде потребувати забагато пам'яті для зберігання такої інформації. Також, дуже важливо час від часу виконувати синхронізацію таких ресурсів де зберігається інформація по розмежуванню доступ, так як ці джерела інформації час від часу оновлюються.

Під час використання розмежування доступу на рівні записів в розподіленій системі кешування даних, в такому випадку зберігання інформації в зовнішній системі не буде мати значення. По-перше, зчитування одного запису буде займати багато часу.

По-друге, зберігання такої збиткової інформації в зовнішній системі не завжди має сенс, життєвий цикл одного запису в кеші набагато менший за час існування самого кешу, тому зачасту таке зберігання не потрібне.

Для розмежування доступу на рівні записів в кеші, найкращим варіантом буде зберігання інформації доступу до запису біля самого запису.

3.2 Розробка моделі мандатного методу

У цій моделі важливо розрізняти поняття користувач і суб'єкт. Рівні безпеки призначаються суб'єктам. А користувачі можуть виступати від імені суб'єкта в той чи інший момент. При цьому в різних ситуаціях один користувач може виступати від імені різних суб'єктів. При цьому важливо, щоб в кожен конкретний момент, користувач виступав від імені тільки одного суб'єкта. Це забезпечує неможливість передачі інформації від високого рівня до більш низького.

3.2.1 Розмежування доступу суб’єктів та об’єктів розподіленої системи кешування даних використовуючи мандатний метод

Описуючи механізм доступу суб’єктів до обєктів в розподіленій системі кешування даних використовуючи мандатний метод, також потрібно розглянути:

4) підхід доступу користувачів системи до кешів з інформацією
5) підхід доступу користувачів системи до записів а цих кешах
6) підхід передачі та надання доступу від одного суб’єкта до іншого

Під час роботи з кешом, для мандатної моделі можна виділити основні рівні довіри що дозволяються або не дозволяються суб’єкту над кешом, а саме:

1) секретні кеші
2) закриті кеші
3) публічні кеші

Отже, проаналізуємо детально підхіл доступу користувачів системи до кешів з інформацією, схема дії користувача системи, що вже має доступ до читання/ писання деякого кеша:

1) користувач виконує запрос в розподілену систему кешування даних безпосередньо або через іншу систему, що використовує кеш для швидкого доступу критично важливої інформації
2) під час запиту, система використовує так звані рівні доступу, для перевірки на правильність доступу суб’єкта до об’єкта

3) як і в дискреційній моделі якщо копія для швидкого доступу до ресурсів серверу автентифікації не присутня в сервері в якому виконується запит, система виконує один додатковий запит через сервер автентифікації за для отримання інформації доступу. Наступного разу, для даного користувача дані серверу автентифікації беруться з локального серверу коммунікації, що пришвидшує доступ до ресурсу
4) якщо рівень доступу достатній для суб'єкта - успішно повертається потрібна інформація

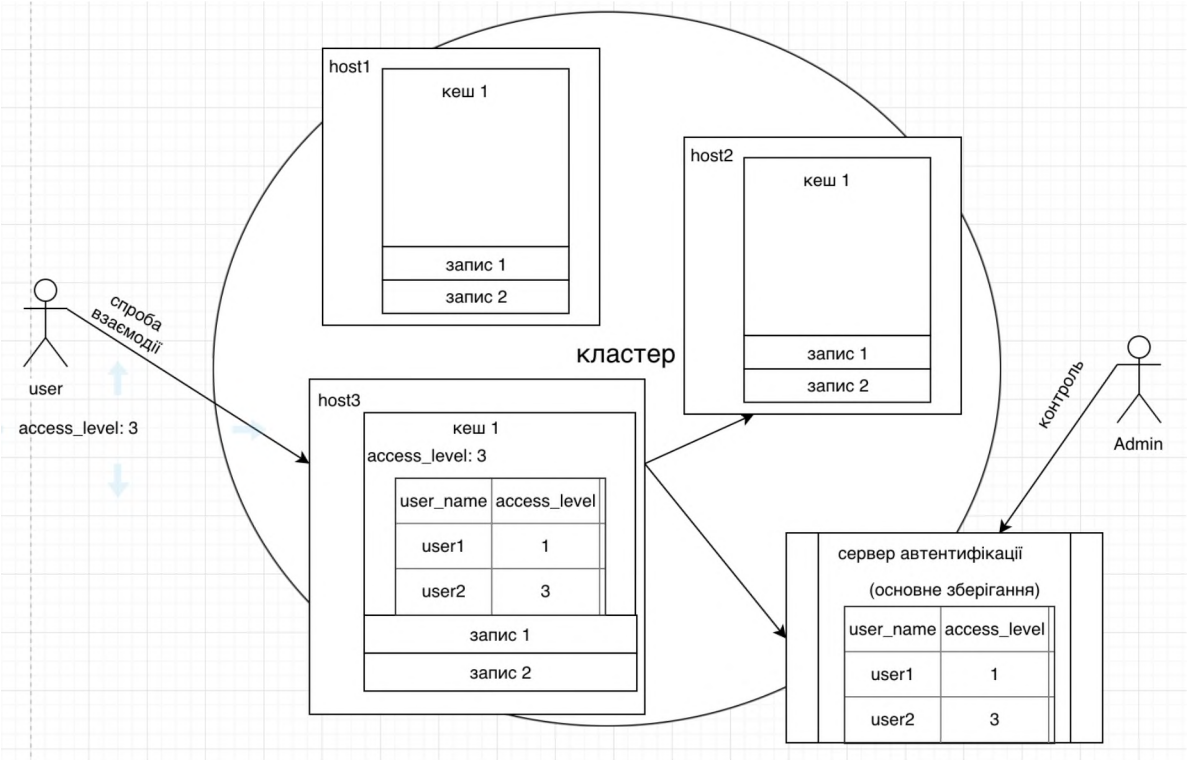

Рисунок 3.3 - Мандатний підхід організації доступу в розподіленій системі кешування даних на рівні кешей.

Для роботи на рівні записів в кеші, для мандатної моделі можна виділити основні рівні довіри що дозволяються або не дозволяються суб'єкту над записом в кеші, а саме:

1) секретні записи в кеші
2) закриті записи в кеші
3) публічні записи в кеші

Тепер, розглянувши підхід доступу користувачів системи до кешів з інформацією, розглянемо доступ на рівні записів до цих кешів, схема дії користувача системи, що вже має доступ до читання/писання деякого кеша:

1) користувач виконує запрос в розподілену систему кешування даних безпосередньо або через іншу систему, що використовує кеш для швидкого доступу критично важливої інформації
2) під час запиту, система використовує використовує та звані рівні доступу, для перевірки на правильність доступу суб'єкта до об'єкта. Ця інформація доступу являється досить малою для рівня одного запису, та може бути збережена разом із самим записом в кеші(По факту, потрібно зберігати лише один запис про рівень доступу разом з ключом та значенням в записі кеша).
3) якщо привілегій достатньо - успішно повертається потрібна інформація

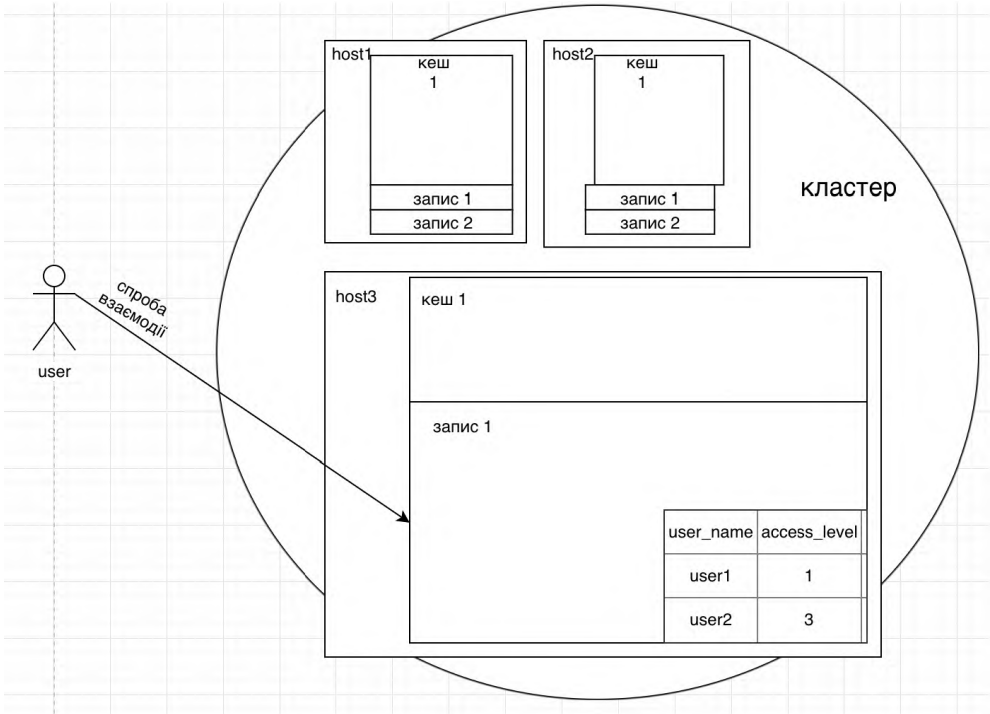

Рисунок 3.4 - Мандатний підхід організації доступу в розподіленій системі кешування даних на рівні записів в кешах.

Передача прав доступу в такому підході розмежування доступу для записів в кеші виглядає не досить простою. Для виконання передачі прав об'єкта що

знаходиться на вищому рівні, суб'єкту потрібно отримати доступ за допомогою підвищення рівня його дій. Це не завжди являється правильним шляхом надання прав, так як деякі інші об'єкти доступу що знаходяться на рівні вище, не завжди також потрібно бути дозволяти відкривати, в такому випадку, система буде потребувати змінення кількості рівні доступу в системі, що може зламати всю ієрархію послідовностей розмежування доступу в системі.

3.2.2 Переваги та недоліки мандатного методу

Проаналізувавши мандатний метод, можна зробити висновки що від погано підходить під архітектуру розподіленої системи кешування даних, так як кейсів в якому він може бути використаний, дуже мало в існуючих інформаційних системам.

Цей метод може бути використаний як допоміжний в системі, але точно не як основний як для розподілення на рівні кешу, так і на розмежування на рівні записів в кеші.

Основними перевагами такого підходу є невелика кількість пам'яті та даних для його використання, а також простота в реалізації механізму розмежування. До недоліків можна віднести не гнучкість підходу розмежування, так як постійна динамічна інформація в таких розподілених кешах може потребувати постійного оновлення рівнів безпеки та розмежування в системі. Цю проблему можна вирішити за допомогою динамічного виділення рівнів доступу до кешей, але це буде потребувати більше пам'яті для зберігання такої інформації а також можливість використання зовнішнього сервісу авторизації та автентифікації.

3.3 Розробка моделі рольового методу

Основна ідея рольової моделі контролю за доступом (Role-Based Access Control - RBAC) заснована на максимальному наближенні логіки роботи системи до реального поділу функцій персоналу в організації. Рольовий метод управління доступом контролює доступ користувачів до інформації на основі типів їх активностей в системі. Застосування даного методу передбачає визначення ролей в системі. Поняття роль можна визначити як сукупність дій і

обов'язків, пов'язаних з певним видом діяльності. Таким чином, замість того, щоб вказувати всі типи доступу для кожного користувача до кожного об'єкту, досить вказати тип доступу до об'єктів для ролі. А користувачам, в свою чергу, вказати їх ролі. Користувач, "виконує" роль, має доступ певний для ролі.

3.3.1 Розмежування доступу суб'єктів та об'єктів розподіленої системи кешування даних використовуючи рольовий метод

Описуючи механізм доступу суб'єктів до обєктів в розподіленій системі кешування даних для рольової моделі, також, потрібно розглянути:

1) підхід доступу користувачів системи до кешів з інформацією
2) підхід доступу користувачів системи до записів а цих кешах
3) підхід передачі та надання доступу від одного суб'єкта до іншого

Під час роботи з кешом, для рольової моделі можна виділити основні операції що можуть бути представлені в виді ієрархічної структури, де дії над об'єктами можна групувати у інші ролі для кращого менеджменту дозволених дій що дозволяться суб'єкту над кешом, а саме:

1) роль читання з кешу;
2) роль для запису в кеш;
3) роль перевірки запису в кеші;
4) роль що дозволяє створювати кеші;
5) роль що дозволяє надавати ролі іншим суб'єктам системи (адміністратор);
6) роль адмін системи (включає в собі всі попередні).

Отже, проаналізуємо детально підхіл доступу користувачів системи до кешів з інформацією, схема дії користувача системи, що вже має доступ до читання/ писання деякого кеша:

1) користувач виконує запрос в розподілену систему кешування даних безпосередньо або через іншу систему, що використовує кеш для швидкого доступу критично важливої інформації;
2) під час запиту, система використовує використовує ролі з зовнішньої або внутрішньої системи, в якій зберігається вся інформація про розмежування;

3) якщо копія для швидкого доступу до ресурсів серверу автентифікації не присутня в сервері де виконується запит, система виконує один додатковий запит з серверу автентифікації для отримання інформації доступу. Наступного разу, для даного користувача дані серверу автентифікації беруться з локального серверу коммунікації, що пришвидшує доступ до ресурсу;
4) якщо привілегій достатньо - успішно повертається потрібна інформація.

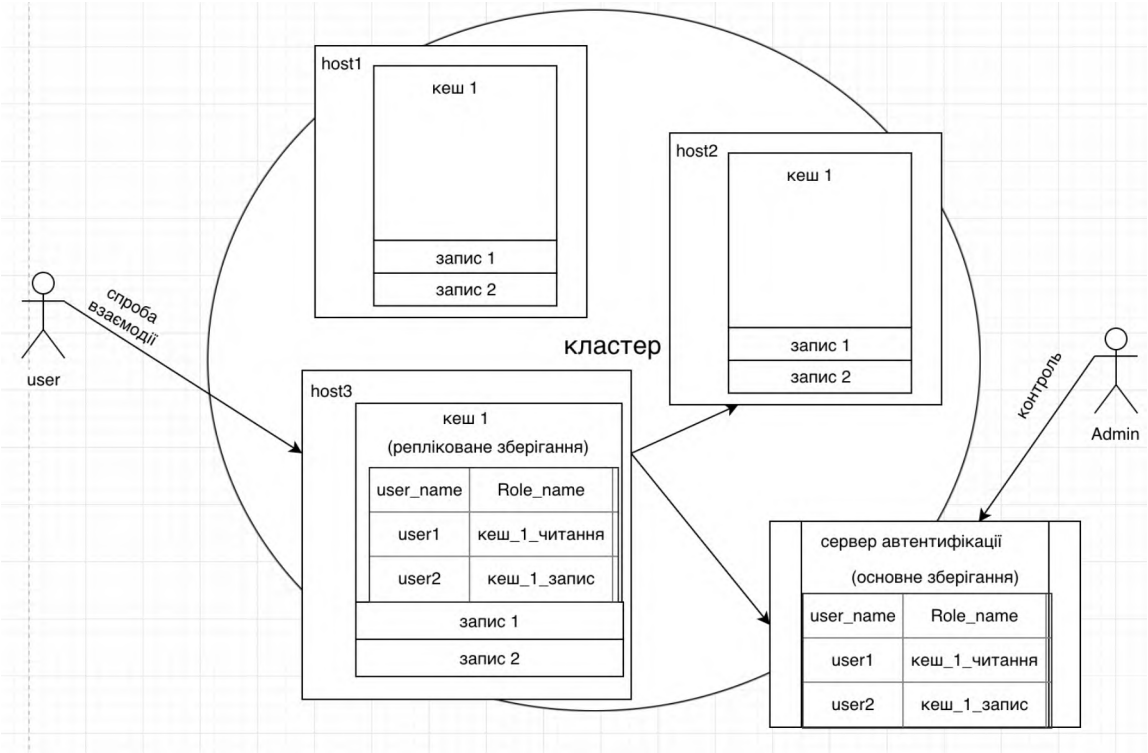

Рисунок 3.5 - рольовий підхід організації доступу в розподіленій системі кешування даних на рівні доступу кешів.

Передача прав доступу в такому підході виглядає досить просто. Користувач, що має право надавати доступ на деяку операцію над кешом, просто записує в потрібну колонку таблиці ролей доступу до потрібного користувача значання, що буде дозволяти йому виконувати привегілейовану йому операцію(наприклад роль що буде йому дозволяти читати з кеша).

Тепер, розглянувши підхід доступу користувачів системи до кешів з інформацією, розглянемо доступ на рівні записів до цих кешів, схема дії користувача системи, що вже має доступ до читання/писання деякого кеша:

1) користувач виконує запрос в розподілену систему кешування даних безпосередньо або через іншу систему, що використовує кеш для швидкого доступу критично важливої інформації;
2) під час запиту, система використовує використовує ролі з зовнішньої або внутрішньої системи, в якій зберігається вся інформація про розмежування , ця інформація являється досить малою для рівня одного запису, та може бути збережена разом із самим записом в кеші;
3) якщо привілегій достатньо - успішно повертається потрібна інформація.

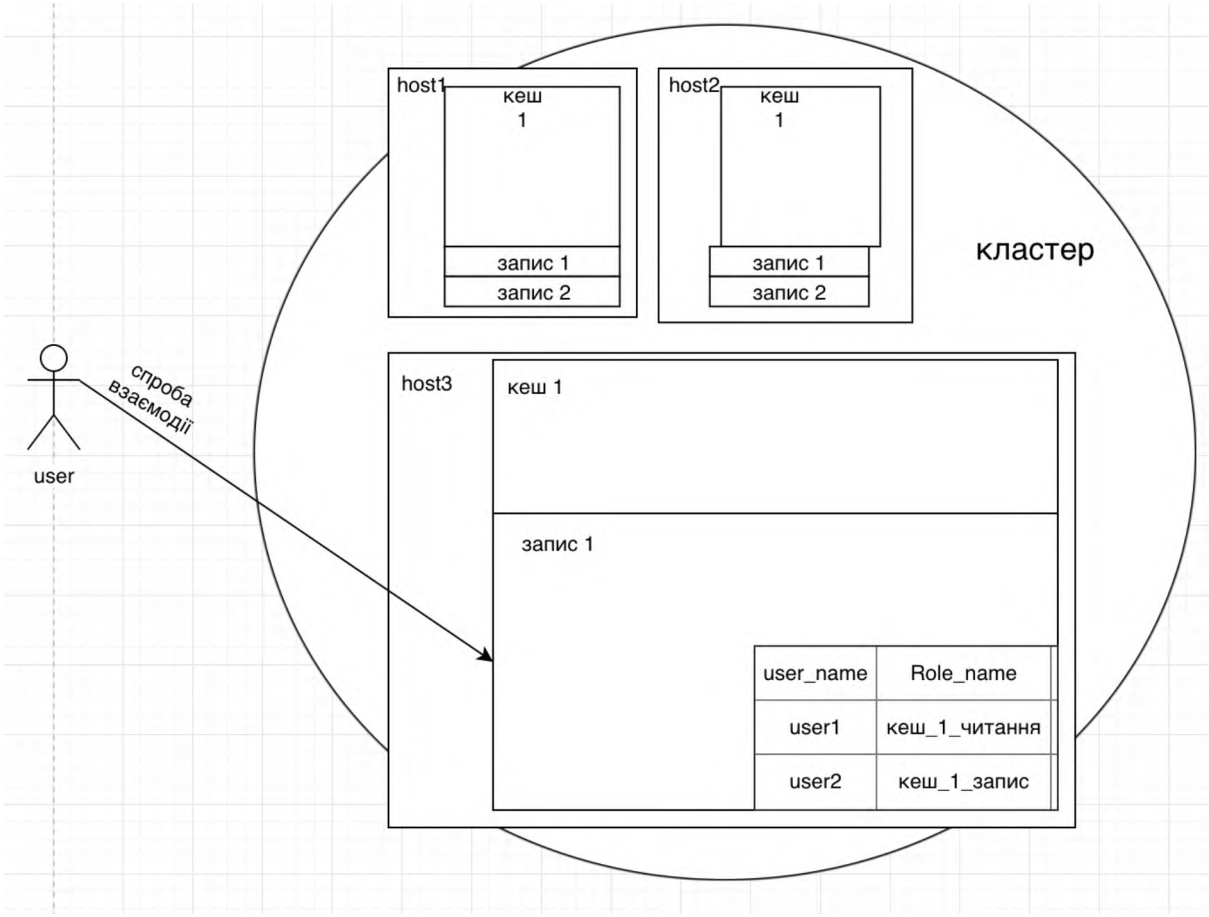

Рисунок 3.6 - рольовий підхід організації доступу в розподіленій системі кешування даних на рівні доступу до записів до кешів.

Передача прав доступу в такому підході виглядає досить просто. Користувач, що має право надавати доступ на деяку операцію над кешом, просто записує в потрібну колонку таблиці ролей доступу для конкретного запису ролі значення імені користувача що має право використовувати цей запис, це будет дозволяти йому виконувати привегілейовану операцію(наприклад роль що буде йому дозволяти читати з кеша).

3.3.2 Переваги та недоліки рольового методу

Основними перевагами рольової моделі управління доступом є:

У класичних моделях розмежування доступу, права на виконання певних операцій над об'єктом прописуються для кожного користувача або групи користувачів. У рольової моделі поділ понять роль і користувач дозволяє розбити задачу на дві частини: визначення ролі користувача і визначення прав доступу до об'єкта для ролі. Такий підхід сильно спрощує процес адміністрування, оскільки при зміні області відповідальності користувача, досить прибрати у нього старі ролі і призначити інші відповідні його нових обов'язків. У разі коли права доступу визначаються безпосередньо між користувачами і об'єктами, ця ж процедура потребує масу зусиль з перепризначення нових прав користувача.

Систему ролей можна налаштувати таким чином, щоб вона набагато ближче відображала реальні бізнес процеси за допомогою побудови ієрархії ролей. Кожна роль поряд зі своїми власними привілеями може успадковувати привілеї інших ролей. Такий підхід також істотно спрощує адміністрування системи.

Рольова модель дозволяє користувачеві реєструватися в системі з наймен- шою роллю дозволяє йому виконувати необхідні завдання. Користувачам мають безліч ролей, не завжди потрібні всі їхні привілеї для виконання конкретного завдання. Принцип найменшої привілегії дуже важливий для забезпечення достовірності даних в системі. Він вимагає, щоб користувачі давали тільки ті з дозволених йому привілеїв, які йому потрібні для виконання конкретного завдання. Для цього потрібно з'ясувати цілі завдання, набір привілеїв необхідних для її виконання і обмеження привілегії користувача цим набором. Заборона привілеїв користувача не вимагаються для виконання поточного за- вдання дозволяє уникнути можливості обійти політику безпеки системи. Ще одним важливим принципом в системі управління доступом є поділ обов'язків. Досить поширені ситуації, в яких ряд певних дій не може виконуватися однією людиною, щоб уникнути шахрайств. Прикладом цього можуть служити операції по створенню платежу і його підтвердження. Очевидно що ці операції

не можуть виконуватися одною і тою же людиною. Система рольового управління доступом допомагає вирішити цю задачу з максимальною простотою.

3.4 Реалізація розмежування доступу розподілених систем кешування даних

Розглянемо основні інтерфейси коду розмежування доступу для розподілених систем кешування даних:

Список, що визначає основні операції які можуть бути виконані в розподіленій системі кешування даних(використовуються для кешу та запису у кеші):

```
public enum CacheAction {
   WRITE,UPDATE,READ,DELETE
}
```

Класс що визначає та описує мандатну політику доступу до об’єкту:

```
public class LabelEntryImpl implements LabelEntry {
     private String labelName;
   public LabelEntryImpl( String labelName) {
       this.labelName = labelName;
   }
   public String getLabelName() {
       return labelName;
   }
}
```

Класс що визначає та описує дикреційну політику доступу до об’єкту:

```
public class MatrixEntryImpl<I> implements MatrixEntry<I> {
   private final List<Permission<I>> permissions;

   public MatrixEntryImpl( List<Permission<I>> permissions) {
       this.permissions = Collections.unmodifiableList(permissions);
   }
   @Override
   public List<Permission<I>> getPermissions() {
       return permissions;
   }
}
```

Класс що визначає та описує рольову політику доступу до об’єкту:

```
public class RoleEntryImpl implements RoleEntry {
   private Role role;
   public RoleEntryImpl( Role role) {
       this.role = role;
   }
   @Override
   public Role getRole() {
       return role;
   }
}
```

Інтерфейс що визначає та описує привілегії суб'єкта в мандатній політиці доступу до об'єкту:

```
public interface MandatoryBasedPrincipal extends Principal {
   List<LabelEntry> getLabels();
}
```

Інтерфейс що визначає та описує привілегії суб'єкта в рольовій політиці доступу до об'єкту:

```
public interface RoleBasedPrincipal extends Principal {
   List<RoleEntry> getRoles();
}
```

Інтерфейс що визначає та описує привілегії суб'єкта в дискреційній політиці доступу до об'єкту:

```
public interface DiscretionalBasedPrincipal< I> extends Principal {
   List<MatrixEntry<I>> getMatrix();
}
```

Значення даних що зберігаються в записі кешу, визначається наступним інтерфейсом. Конкретна реалізація цього інтерфейсу буде визначати яку політику доступу потрібно буде використовувати під час роботи з записом в кеші:

```
public interface CacheValueEntry<V, I>  {
    V getValue();
    boolean isSecured();
     void validateSecurityParams(I identifier, CacheAction cacheAction) throws
SecurityException;
}
```

Також, реалізація дискреційного методу для конкретного запису в кеші визначається так:

```
public final class MatrixCacheValue<V,I> implements CacheValueEntry<V,I> {
   private final V value;
   private final MatrixEntry<I> matrixEntry;
   public MatrixCacheValue(V value, List<Permission<I>> permissions) {
       this.value = value;
       this.matrixEntry =new MatrixEntryImpl<>(permissions);
   }
   @Override
   public V getValue() {
       return value;
   }
   @Override
   public boolean isSecured() {
       return !matrixEntry.getPermissions().isEmpty();
   }
   @Override
    public void validateSecurityParams(I identifier, CacheAction cacheAction)
throws SecurityException {
       matrixEntry.getPermissions().stream()
                                                    .filter(iPermission    ->
cacheAction.equals(iPermission.getCacheAction()))
                                                    .filter(iPermission    ->
iPermission.getIdentifier().equals(identifier))
                    .findAny().orElseThrow(()-> new SecurityException("access
denied"));
   }
}
```

Якщо механізм розмежування доступу до запису не потребується, можна використати реалізацію що не потребує розмежування:

```
public class SimpleCacheValueEntry<V,I> implements CacheValueEntry<V,I> {
   private final V value;
   public SimpleCacheValueEntry(V value) {
       this.value = value;
   }
   @Override
   public V getValue() {
       return value;
   }
   @Override
   public boolean isSecured() {
       return false;
   }
   @Override
    public void validateSecurityParams(I identifier, CacheAction cacheAction) throws
SecurityException {
       //nothing to validate
   }
}
```

Основний інтерфейс інтеграції кеша з зовнішньою системою зберігання інформації розмежування доступу для системи кешування даних:

```
public abstract class AbstractAuthenticator<I,PC extends Principal>
        implements Authenticator<I> {
    private final Credentials<I> credentials;
    private final CacheAuthMetadataSource<PC, I> cacheAuthMetadataSource;
    private final AuthValidator<PC> authValidator;
    public AbstractAuthenticator(Credentials<I> credentials,
                                                    CacheAuthMetadataSource<PC, I>
cacheAuthMetadataSource,
                                 AuthValidator<PC> authValidator) {
        this.credentials = credentials;
        this.cacheAuthMetadataSource = cacheAuthMetadataSource;
        this.authValidator = authValidator;
    }
    @Override
      public boolean isCachePermissionsValid(String cacheName, CacheAction
cacheAction) {
                                    PC     cachePrincipalForCredentials     =
cacheAuthMetadataSource.getCredentialsPrincipalForCache(credentials,cacheName)
;
                                 return     authValidator.validate(cacheAction,
cachePrincipalForCredentials);
    }
    @Override
    public I getIdentifier() {
        return credentials.getIdentifier();
    }
}
```

SecureCacheProxy - Основна сутність, що реалізує проксування роботи кеша та додає логіку роботи з розмежуванням доступу дозволяючи чи не дозволяючи доступ до системи кешу. Тут використовується інтерфейс `Authenticator` що визначає доступ деякого користувача і дозволяє чи не дозволяє йому виконувати операції на рівні кешу.

Може бути використана з будь якою реалізацією кешу, що підтримує стандартний java інтерфейс javax.cache.Cache:

```
public class SecureCacheProxy<K,V,I> {
   private final Cache<K, CacheValueEntry<V,I>> igniteCache;
   private final Authenticator<I> cacheAuthenticator;
   public SecureCacheProxy(Cache<K, CacheValueEntry<V,I>> igniteCache,
                           Authenticator cacheAuthenticator) {
       this.igniteCache = igniteCache;
       this.cacheAuthenticator = cacheAuthenticator;
   }
   public void put(K key, CacheValueEntry<V,I> value) {
           if(cacheAuthenticator.isCachePermissionsValid(igniteCache.getName(),
CacheAction.WRITE)){
            if(value.isSecured()) {

value.validateSecurityParams(cacheAuthenticator.getIdentifier(),
CacheAction.WRITE);
            }
                igniteCache.put(key,value);
       }
       else throw new SecurityException("access denied");
   }
   public V get(K key) {
           if(cacheAuthenticator.isCachePermissionsValid(igniteCache.getName(),
CacheAction.READ)){
            System.out.println(igniteCache.get(key));
            CacheValueEntry<V,I> cacheValueEntry = igniteCache.get(key);
            if(cacheValueEntry.isSecured()) {

cacheValueEntry.validateSecurityParams(cacheAuthenticator.getIdentifier(),
CacheAction.READ);
            }
            return cacheValueEntry.getValue();
       }
       else throw new SecurityException("access denied");
   }
   public String getName() {
           if(cacheAuthenticator.isCachePermissionsValid(igniteCache.getName(),
CacheAction.READ)) {
            return igniteCache.getName();
       }
       else throw new SecurityException("access denied");
   }
}
```

Визначивши всі основні інтерфейси так, можемо використати їх для виконання розмежування доступу в розподіленій системі кешування даних, виконаємо наступні дії:

1) Старт розподіленої системи кешування даних[1]:

```
Ignite ignite = Ignition.start("ignite-server-local.xml");
```

2) Створення кешу:

```
IgniteCache<String,CacheValueEntry<String, String>> cache =
ignite.getOrCreateCache("CACHE");
```

3)Проксування кешу для використання розмежування доступу, відкриття сессії взаємодії з кешом:

```
SecureCacheProxy<String, String, String> secureCacheProxy =
                        new SecureCacheProxy(cache, new DummyAuthenticator(new
UsernamePasswordCredentials("user", "user")));
```

4) Додавання записів до кешу використовуючи різні стратегії доступу до запису в кеші:

```
secureCacheProxy.put("1",new SimpleCacheValueEntry<>("1"));
secureCacheProxy.put("2",new MatrixCacheValue<>("1", Arrays.asList(new
Permission("user", CacheAction.WRITE))));
```

5) Успішна спроба дістати значення з кешу, так як запис не розмежується ніякою політикою доступу:

```
System.out.println(secureCacheProxy.get("1"));
```

6) Безуспішна спроба дістати значення з кешу, так як запис розмежується дискреційною політикою доступу(під час запису до кешу, користувач надав собі права лише на запис до цього кешу, не на читання):

```
System.out.println(secureCacheProxy.get("2"));
```

Висновки до розділу 3

Отже, у цьому розділі було побудовано модель розмежування доступу для розподілених систем кешування даних використовуючи методи розмежування доступу в інформаційних системах, розглянуто переваги та недоліки кожного із них.

Також було виконано реалізацію механізму розмежування доступу на основі запропонованих методів та моделей розмежування доступу. Як результат, було створено механізм, що дозволяє інтеграцію з будь-якою системою зберігання прав для суб'єктів доступу. Система має можливість використання будь-якої політики доступу користувачів до кешів, так дає можливість контролювати доступ до записів в кешах використовуючи дискреційну модель доступу, а також паралельно, для інших записів, використовувати інший підхід.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Розробка багато функціональної системи кешування даних з встроєною системою розмежування доступу. Аудит, консалтинг корпоративних клієнтів та розробка рішень на основному продукті платформи.	1. Компанії середнього та великих масштабів(банки, телеком компанії, сільськогосподарскі системи контролю)	Мінімізація фінансових та репутаційних втрат, забезпечення інформаційної безпеки як складової безпеки бізнесу. Забезпечення захищеності інформаційної структури.Розробка програмного забезпечення надвисокого рівня з мінімальним використанням ресурсу
Розслідування інцидентів, аудит консалтинг в сфері інформаційної безпеки.	2. Органи державної влади України (public sector)	Отримання єдиного інструмента керування безпекою та комплексного бачення стану захищеності.
Постачання і впровадження засобів розподіленного кешування даних.	3. Open Source	Отримання користування системи в “бета” варіанті та можливість використання загальних механізмів в тестовому режимі

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

1	*2*	*3*		*4*	*5*	*6*
№ п/п	*Техніко - економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*		*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Apache Ignite*			
1.	економічні	Витрати на розробку рішення, закупку ліцензій, розміщення на хостингу, навчання спеціалістів та маркетинг – 500,000 $	Витрати на розробку рішення, закупку ліцензій, розміщення з використанням власних обчислювальних потужностей, зарплати програмістів та маркетинг – 30 00,000 $	Недостатній рівень витратна маркетинг	Схожість функціонал у рішення	Значно дешевша реалізація проекту. Використання більш нових технологій з продвинутими методами обробки даних

Продовження таблиці 4.2

1	2	3		5	6	7
2.	технічні	Використання продвинутого комплексу розмежування доступу	Використання різних мов програмування для охоплення більшої аудиторії	Складність у налаштуванні програмного забезпечення	немає	Більша кількість різних сфер використання комплексу для забезпечення інформаційної безпеки
3.	надійності	Використання надійних механіхмів розмежування доступу та механізмів розподілених систем	Використання механізмів розподілених систем	Наявність затримок, що пов’язані з використанням розмежування доступу	Можливість неполадок, які можуть бути пов’язані з використанням sql	Досягнення вищого рівня відмовостійкості та безперервності роботи рішення і надання послуг

Продовження таблиці 4.2

4.	технологічні	Використання платформ, що надають інформацію про загрози, індикатори компрометації, оновлення та виправлення для програмного забезпечення, загальний стан інформаційної безпеки	Відсутність таких компонентів	немає	Більші видатки на експлуатацію рішення	Отримання більшої кількості інформації для підвищення ефективності роботи SOC та якості надаваних послуг
5.	ергономічні	Можливість налаштування розмежування доступу через програмний комплекс рішення під будь-які потреби користувача	Налаштування розмежування доступу через програмний комплекс обмежено	Додаткова складність при налаштуванні комплексу	немає	Отримання консолі для рішення, яка буде налаштована відповідно до потреб та побажань користувача
6.	естетичні	Інтерфейс взаємодії та архітектура програмного коду	Простий механізм масштабування рішення	немає	немає	Можливість налаштування під конкретні потреби користувача.

Продовження таблиці 4.2

1	2	3		4	5	6
7.	органолептичні	Швидкість роботи рішення для користувача базується на політиці конфігурації системи розмежування доступу та конфігурації параметрів кешів	Швидкість роботи рішення базується переважно на швидкості конфігурації параметрів кешів	Віддаленість серверів автентифікації від користувачів, що може спричинити затримки в роботі системи розмежування доступу	Якщо користувач отримує доступ до кешу з відаленого серверу, це також може спричинити затримки	Надання доступу до системи для користувачів з будь-якої точки світу; однаковий рівень затримки, що не залежить від місця розташування користувача.

Продовження таблиці 4.2

8.	транспортабельності	Досить швидке перенесення рішення на нові сервери завдяки використанню плагінної архітектури	Перенесення серверів і даних може зайняти досить тривалий проміжок часу	Значно менший рівень контролю за процесом «переїзду» рішення	немає	Низький рівень downtime для надання послуг при перенесенні даних та програмного забезпечення
9.	екологічності	Мінімізація негативного впливу на довкілля з використанням передових технологій кондиціонування та енергозбереження	немає	немає		Відсутність витрат на системи електрожи влення та кондиціонування

Кінець таблиці 4.2

10.	безпеки	Відсутність витрат для можливості безпечного використання сервісу	Витрати на побудову, експлуатацію та підписку за масштабоване рішення	немає	Реалізація серверних потужностей для рішення у відповідності до прийнятих стандартів.	За безпеку використання серверного парку та інженерних комунікацій несе відповідальність компанія, що надає послуги для розміщення рішення у «хмарі».

4.2 Технологічний аудит ідеї проекту

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Моніторинг безпеки інформаційних систем клієнтів	Використання програмного забезпечення що входить в пакет підписки	Потрібно придбати ліцензію на софт та виконувати інтеграці/	Так, ця технологія є доступною

Кінець таблиці 4.3

2	Впровадження системи плагінної системи розмежування доступу, аутсорс	Розгортання та налаштування плагінного комплексу розмежування доступу на базі системи кешування даних	Потрібно реалізовувати самостійно	Так, ця технологія є доступною
3	Розслідування інцидентів, аудит та консалтинг в сфері розподілених систем кешування даних	Використання платформи для розслідування інцидентів IBM Resilient та побудова системи для роботи call-центру	Потрібно придбати ліцензії IBM та телефони	Так, ця технологія є доступною
4	Постачання і впровадження засобів захисту мережевої інфраструктури системи кешування даних	Використання систем для захисту мережевої інфраструктури Cisco FirePower, Cisco DNA та інші рішення	Заключення договорів на постачання та розширену підтримку обладнання та програмного забезпечення з компанією Cisco	Так, ця технологія є доступною
Обрана технологія реалізації ідеї проекту: так як для реалізації ідеї проекту, всі технології є наявними та доступними, тому обираються всі вище описані технології.				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап-проекту

1	*2*	*3*
№ п/п	*Показники стану ринку хмарних SOC*	*Характеристика*
1	Кількість головних гравців, од	Три (CEO - головний по коммунікація,CTO - головний по розробки програмного рішення, CMO-маркетинг директор)
2	Загальний обсяг продаж, грн/ум.од	1 млн ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Потреба у кадрах із високим рівнем компетентності у сфері інформаційної безпеки та розробки програмного забезпечення

Кінець таблиці 4.4.

5	Специфічні вимоги до стандартизації та сертифікації	Проведення експертиз та сертифікацій щодо стандартизації рішення для використання у державних установах
6	Середня норма рентабельності в галузі, %	Не менше 160

За попередніми оцінками, ринок розподілених систем кешування даних є дуже привабливим для входження, але має ряд обмежень та вимог до сертифікації.

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*

Кінець таблиці 4.5.

1	Розробка високонагружених систем роботи з критичними даними Моніторинг безпеки інформаційних систем	- Компанії середнього і великого бізнесу (medium and large enterprises - MLE) в Україні і за кордоном Провайдери (ISP) - Органи державної влади України (public sector)	Для кожної з трьох категорій існують окремі цінності пропозиції. Пропозиція відрізняється для кожної цільової групи. Клієнти надають перевагу open source. Надто малий розмір/ обмежений бюджет у клієнтів для розгортання повноцінної інфраструктури захисту. Вимоги регулятора (Держспецзв'язку, НБУ тощо)/ Compliance (PCI DSS, ISO 27001). Для клієнтів, таких як державні корпорації та органи влади, закупівлі можуть проводитись у формі відкритих торгів.	- забезпечення надійної та ефективної роботи сервісів, підтримка у режимі 24/7. - Використання сертифікованого програмного забезпечення, швидке усунення неполадок, проведення консультацій та навчання користувачів.

Таблиця 4.6 - Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Цінова конкуренція	Коливання цін на послуги конкурентів	Пошук шляхів зниження вартості послуг, створення нової пропозиції продукту
2	Зниження доходів потенційних споживачів	Зниження купівельної спроможності клієнтів	Вимушене зменшення обсягів виробництва
3	Збільшення розміру податків	Відтік коштів із сфери виробництва до бюджету	Пошук шляхів мінімізації податків
4	Рівень інфляції	Знецінювання коштів, ріст різниці в курсах валют	Отримання довгострокового кредиту

Таблиця 4.7 - Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Поява нових технологій та високоефективного обладнання	Розширення спектру надаваних послуг для клієнтів, збільшення кількості клієнтів, підвищення ефективності роботи сервісів, зменшення витрат на роботу сервісів	Швидке впровадження нових технологій завдяки хмарним сервісам, підвищення вартості послуг, створення нових сервісів для клієнтів, проведення маркетингової кампанії

Кінець таблиці 4.7

2	Стабілізація політичного та економічного становища в державі	Зменшення рівня інфляції, збільшення кількості клієнтів	Спроби лобіювання інтересів компанії в держаних установах
3	Збільшення кількості кібер-атак, діяльності зловмисного програмного коду тощо	Збільшення попиту на послуги компанії	Залучення нових клієнтів, можливе підвищення вартості послуг
4	Залучення нових відомих компаній у якості клієнтів чи постачальників	Збільшення впливу бренду Cloud SOC на ринку послуг розробки програмного забезпечення, зменшення цін на рекламу та маркетинг	Розробка та впровадження нових засобів та програмних комплексів для розширення спектру послуг у сфері розробки систем кешування даних та кібербезпеки

Таблиця 4.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*

Продовження таблиці 4.8

1. олігополістична конкуренція	Динаміка цін, яка майже не залежить від рівню попиту на продукцію; конкуренція зміщуються в площину реклами, рівня якості продукції та індивідуалізації	Вдосконалення рішення для підвищення якості надаваних послуг для клієнтів
2. За рівнем конкурентної боротьби - національний	Надання послуг для різних груп та типів клієнтів в Україні та за її межами	Застосування хмарних технологій для підвищення конкурентоспроможності, застосування нових технологій та підходів для вдосконалення рішення
3. За галузевою ознакою - міжгалузева	Рішення може використовуватися користувачами з різних галузей програмного забезпечення в бізнесі що потребує автоматизацію рішень	Розширення сфери надання послуг, додання нових функцій сервісу та послуг для клієнтів

Кінець таблиці 4.8

4. Конкуренція за видами товарів: - товарно-видова	Рішення, що використовується для задоволення потреб клієнтів, але істотно відрізняються від рішень конкурентів на користь якості сервісів	Надання послуг із розслідування інцидентів, аутсорсу та консалтингу в сфері інформаційної безпеки та розробки програмного забезпечення
5. За характером конкурентних переваг - цінова	Цінова	Надання ширшого спектру послуг, порівняно із конкурентами
6. За інтенсивністю - марочна	Сукупність характеристик та властивостей рішення	Підвищення якості роботи сервісів для клієнтів

Таблиця 4.9 - Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари - замінники*
Складові аналізу	- Apache Ignite - Hazelcast	Розмір капіталовкладень; доступ до ресурсів у конкурентів; наявність патентів та товарних знаків.	Значення розмірів поставок для постачальників. Диференціація витрат.	Розміри закупівель державних підприємств та органів влади; рівень чутливості до зміни ціни; контроль якості	Ціна товару та змінні витрати
Висновки:	Наявна досить інтенсивна конкурентна боротьба з боку прямих конкурентів	Є можливості входу на ринок, але існує чимало серйозних конкурентів, що вже працюють на цьому ринку.	Постачальники диктують умови роботи на ринку; компанія, яка здійснює більшу кількість продажів, отримує привілеї та більші розміри знижок на товари та послуги	Клієнти диктують умови на ринку; при організації закупівель товарів та послуг, відчутний рівень відношення до вартості рішення та його якості.	Обмежень на ринку через товари замінники немає.

Даний проект має принципові можливості для роботи на ринку з огляду на конкурентну ситуацію. Серед сильних сторін, можна виділити використання потужного плагінного сервісу розмежування доступу, надання широкого спектру послуг в сфері зберігання великих даних, що не має аналогів в Україні, незалежність від цін за обслуговування серверних потужностей (електроенергія, інженерні комунікації тощо), швидке впровадження нових технологій; стабілізація політичного та економічного стану, що буде сприяти збільшенню рівня попиту на послуги та лобіювання інтересів компанії на рівні держави; залучення великих відомих компаній в якості постачальників чи клієнтів.

Таблиця 4.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1	Ціна та змінні витрати	Ціноутворення, яке не є однаковим на послуги для різних типів клієнтів (це обумовлено рівнем видатків за використання хмарних технологій для розміщення потужностей SOC).
2	Розміри закупівель замовників	розподілений кеш, який реалізований у хмарному середовищі, здатний легко масштабуватись у відповідності до потреб замовників послуг; це значно підвищує рентабельність проекту, порівняно з конкурентами, у яких потужності не розміщені «хмарі».

Продовження таблиці 4.10

3	Доступ до ресурсів у конкурентів	Так як компанії-конкуренти вже працюють на ринку, вони мають клієнтів та встановлений шлях продажів та маркетингу, тому вони мають більше ресурсів як матеріальних, так і інформаційних
4	Гнучкість цін на послуги	На відміну від конкурентів, при застосуванні хмарних технологій, є можливість зменшення цін на послуги для користувачів
5	Рівень концентрації послуг	Спектр послуг, які можуть бути надані клієнтам є значно ширшим, в порівнянні з конкурентами

Кінець таблиці 4.10

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін « Sys Cache Store»

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з Sys Cache Store*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Ціна та змінні витрати	15			+				
2	Розміри закупівель замовників	16	+						
3	Доступ до ресурсів у конкурентів	8							+
4	Гнучкість цін на послуги	12		+					
5	Рівень концентрації послуг	17		+					

Таблиця 4.12 - SWOT-аналіз стартап-проекту

Сильні сторони:багатофункіональний сервіс зберігання великих обємів даних та плагінний підхід організації розмежування достуау; надання широкого спектру послуг, що не має аналогів в Україні;використання хмарних технологій з метою швидкого масштабування	*Слабкі сторони*: низький рівень маркетингу; негативне ставлення до хмарних технологій з боку держави; можливі затримки у роботі сервісів, що пов'язані із віддаленим розташування сервісів від клієнтів
Можливості: швидке впровадження нових технологій; стабілізація політичного та економічного стану, що буде сприяти збільшенню рівня попиту на послуги та лобіювання інтересів компанії на рівні держави; залучення великих відомих компаній в якості постачальників чи клієнтів	*Загрози*: цінова конкуренція; збільшення розміру податків; зниження доходів потенційних споживачів; збільшення рівня інфляції

Таблиця 4.13 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Концентрація на одному чи двох типах клієнтів	Висока ймовірність отримання ресурсів	6 місяців

Кінець таблиці 4.13

2	Побудова рішення, використовуючи компоненти однієї компанії - постачальника	Висока ймовірність отримання ресурсів та залучення підтримкою вендора	1 рік
3	Побудова гібридного сервісу	Мала ймовірність отримання ресурсів	3 роки

З означених альтернатив ринкового впровадження даного стартап-проекту було вирішено обрати побудову рішення з концентрацією на одному чи двох типах клієнтів при цьому строки реалізації становитимуть приблизно 6 місяців.

4.4 Розроблення ринкової стратегії проекту

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

1	*2*	*3*	*4*	*5*	*5*
№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*

Продовження таблиці 4.14

1	Державні корпорації та органи державної влади	Клієнти потребують продукт такого типу та готові ним користуватись	Високий попит, пов'язаний із необхідністю послуг у сфері зберігання критичних даних в великих обємах	Невисока конкуренція в сегменті, як а пов'язана із вимогами щодо стандартизації та сертифікації	Є складність входу, яка пов'язана із негативним ставленням до хмарних сервісів
3	Компанії середнього і великого бізнесу (medium and large enterprises - MLE) в Україні і за кордоном	Клієнти потребують продукт такого типу та готові ним користуватись	Високий попит, пов'язаний із необхідністю послуг у сфері зберігання критичних даних в великих обємах	Дуже високий рівень конкуренції	Є складність, пов'язана з ціновою конкуренцією на послуги

Кінець таблиці 4.14

4	Поодинокі користувачі (фізичні особи)	Споживачі зацікавлені продуктом та готові ним користуватись	низький попит, пов'язаний із високою вартістю послуг, використання open source версії	Практично відсутня конкуренція	Легкий вхід, але низька рентабельність проекту
Які цільові групи обрано: державні корпорації та органи державної влади, провайдери ISP та компанії середнього і великого бізнесу.					

На підставі ринкової стратегії обрано використання стратегії диференційованого маркетингу.

Таблиця 4.15 - Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
1	Флангова атака	Стратегія диференційованого маркетингу	Сильні сторони та можливості рішення	Стратегія диференціації

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
1	Ні	Так	Так. Використання розподіленого сервісу кешування даних, Datagrid	Стратегія виклику лідера

Таблиця 4.17 - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту*

Кінець таблиці 4.17

1	Отримання комплексних послуг з «одних» рук. Зниження видатків з бюджету на забезпечення захисту. Отримання знань від спеціалістів щодо інформаційної безпеки та розподілених систем кешування даних. Отримання єдиного інструмента керування безпекою та комплексного бачення стану захищеності в системі зберігання даних. Мінімізація фінансових та репутаційних втрат, забезпечення інформаційної безпеки як складової безпеки бізнесу через зберігання та розмежуавння доступу до даних. Забезпечення захищеності інформаційної структури.	Стратегія диференціації	Сильні сторони та можливості рішення	Моніторинг безпеки інформаційних систем клієнтів Sys Cache Store. Розслідування інцидентів, розробка програмного забезпечення використовуючи систему, консалтинг та підтримка

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару

1	*2*	*3*	*4*
№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
1	Зберігання великих обємів критичної інформації що часто використовуєтьс я	Забезпечення захищеності інформаційної структури	Отримання комплексних послуг з «одних» рук. Зниження видатків з бюджету на забезпечення захисту. Отримання знань від спеціалістів щодо розмежування доступу використовуючи систему
2	Розмежування доступу до інформації в системі	Високий рівень відмовостійкос ті та безперервності роботи рішення і надання послуг	Кваліфіковані та сертифіковані кадри для виконання поставлених задач (розслідування інцидентів, консультації, розробка, консалтинг тощо)
3	Отримання єдиного сервісу керування даними та розмежуванням доступу	Розподілена система кешування даних Sys Cache Store	програмне рішення, надання широкого спектру послуг, що не має аналогів в Україні; гнучкість цін на послуги

Таблиця 4.19 - Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
І. Товар за задумом	Моніторинг безпеки інформаційних систем клієнтів Sys Cache Store. Розслідування інцидентів, розробка програмного забезпечення використовуючи систему, консалтинг та підтримка		
ІІ. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Використання хмарних сервісів для розміщення рішення	М	Вр/Тх/Тл
	2. Використання плагінної системи розмежування доступом	М	Тх/Тл/Е
	3. Використання платформ, що надають інформацію про загрози, індикатори компрометації, оновлення та виправлення для програмного забезпечення, загальний стан інформаційної безпеки	Нм	Вр/Тл/Е
	4. Можливість налаштування системи розмежування доступу та методи і підходи зберігання даних у кешах	М	Тх/Е/Ор
	5. Швидкість роботи рішення для користувача	Нм	Тл/Е/Ор

Кінець таблиці 4.19

	6. Надання хорошого інтерфейсу взаємодії с системою кешування даних та системою розмежуваннчя даних	M	Тл/Е
	Якість: Забезпечення захищеності інформаційної структури. Виконання вимог побудови надійного сховища даних/Compliance (PCI DSS, ISO 27001)		
	Надання користувачу електронної ліцензії (ключа доступу) до хмарного сервісу (якщо побудова системи відбувається в хмарному середовищі)		
	Марка: Sys Cache Store		
III. Товар із підкріпленням	До продажу: для стимулювання попиту на продукт можна розробити програму надання лімітованого Open Source продукту		
	Після продажу: Розробка та проведення рекламної кампанії		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності			

Таблиця 4.20 - Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
1	20000 ум.од	650 ум.од	Високий або середній	Ціна підписки на сервіс моніторингу/ розмежування доступу– 350-700 ум.од.; ціна послуг розробки – 1000-30000 ум.од. за місяць роботи

Таблиця 4.21 - Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*

Кінець таблиці 4.21

1	прийняття рішення про необхідність систем розподіленого кешування та керування даних, вибір джерела задоволення своїх потреб для забезпечення безпеки інформаційних систем і укладення угоди	пристосування збутової мережі до запитів споживачів; пошук перспективних засобів просування товарів; розробка та вдосконалення маркетингової політики; вибір посередників	Дворівневий канал збуту (із залученням посередника та дистриб'ютора)	Залучення компаній посередників та партнерів для формування системи збуту

Таблиця 4.22 - Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*

Кінець таблиці 4.22.

1	Дослідження властивостей та якостей рішення, можливість тестування продукту, прийняття рішення про необхідність використання продукту для задоволення своїх потреб	Канали інтегрованих маркетингових комунікацій	Використання сильних сторін рішення: розподілений сервіс кешування даних; надання широкого спектру послуг, що не має аналогів в Україні; незалежність від цін за обслуговування серверних потужностей; швидке впровадження нових технологій	повідомлення, пов'язані з особистою вигодою аудиторії, що показують, як товар може задовольнят и потреби покупця;	реклама, що демонструє якість товару, його економічність, цінність або можливості експлуатації

Висновки до розділу 4

Результатом проведеного аналізу та оцінки ризиків, було виявлено, що даний проект має можливість для ринкової комерціалізації. Для представленого рішення є високий рівень попиту, що пов'язаний із станом росту крупних проектів, де використання розподілених систем кешування даних з використанням плагіного підходу розмежування доступу допоможе пришвидшити роботу з даними та буде гарантувати їх безпеку. Для цього стартап-проекту є непогані перспективи входження в ринок, але наявні певні бар'єри, подолання яких підвищують конкурентоспроможність представленого рішення. Щодо альтернативи впровадження стартап-проекту та його ринкової реалізації, доцільно обрати механізми для побудови рішення з використанням компонентів однієї компанії постачальника. Я вважаю, що є доцільною подальша робота над імплементацією даного проекту.

ВИСНОВКИ

В роботі було виконано аналіз розподілених систем кешування даних та систем розмежування доступу, побудову моделей розмежування доступу в розподілених системах кешування даних та розроблено програмне рішення для реалізації моделі розмежування доступу в таких системах.

Як результат, проаналізувавши монолітний та розподілені варіанти розмежування доступу, було виявлено, що використання розподіленого варіанту є важчим, але в свою чергу він допомагає уникнути проблеми роботи кластеру під час відказу одного сервера з топології. Було проаналізовано існуючі підходи реалізації такого розмежування та автентифікації. Виявлено, що практики використання існуючих механізмів немає - на цей час вони не являються гнучкими і адаптивними під бізнес задачі.

Було побудовано модель розмежування доступу для розподілених систем кешування даних використовуючи методи розмежування доступу в інформаційних системах. Виявленно, що для розмежування доступу на рівні кешів, найкраще підходить рольова модель, так як ієрархічний підхід доступу до кешів безперечно вписується до вимог використання таких ресурсів. Розмежування доступу на рівні записів в кешах гарним чином можна описати через дискреційну модель. Обмежена кількість операцій та не велика множина суб'єктів в матриці доступу дискреційної моделі, що потребує запис, дає можливість чудово зберігати цю інформацію разом із самим записом, як наслідок, кеш не буде страждати від проблем зі швидкістю роботи процесів запису та читання. Побудова мандатної моделі, в такій системі не виглядає перспективною, а саме, з точки зору її використання на практиці, імплементації та експлуатації.

Результат роботи - механізм, що дозволяє інтеграцію з будь-якою системою зберігання прав для суб'єктів доступу. Система має можливість використання будь-якої політики доступу користувачів до кешів, дає можливість контролювати доступ до записів в кешах використовуючи дискреційну модель доступу, а також паралельно, для інших записів, не використовувати взагалі.

Такий підхід являється зручним, так як для різних записів в одному кеші, ми можемо визначати різні політики доступу, що підтверджує гнучкість та зручність даного інтерфейсу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1 High Performance in-memory computing with Apache Ignite.

2 П. Н. ДЕВЯНИН. МОДЕЛИ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ / П. Н. ДЕВЯНИН. – Москва, 2005.

3 Apache Ignite Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://apacheignite.readme.io/docs.

4 Модели безопасности компьютерных систем [Електронний ресурс] – Режим доступу до ресурсу: http://itsec.ru/articles2/Oborandteh/modeli-be- zopasnosti-komputernih-setei.

5 Модель_Бела_—_ЛаПадули [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%91%D0%B5%D0%BB%D0%B0_%E2%80%94_%D0%9B%D0%B0%D0%9F%D0%B0%D0%B4%D1%83%D0%BB%D0%B8.

6 Модель_Біби [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%91%D1%96%D0%B1%D0%B8.

7 Обзор и сравнение существующих методов управления доступом [Електронний ресурс] – Режим доступу до ресурсу: http://www.ict.nsc.ru/ws/YM2003/6312/.

8 Hazelcast [Електронний ресурс] – Режим доступу до ресурсу: https://ru.bmstu.wiki/Hazelcast.

9 Правила розмежування доступу [Електронний ресурс] – Режим доступу до ресурсу: https://helpiks.org/6-26934.html.

10 Керування доступом на основі ролей [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BE%D0%

BC_%D0%BD%D0%B0_%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%96_%D1%80%D0%BE%D0%BB%D0%B5%D0%B9.

ДОДАТКИ

ДОДАТОК А

Програмна реалізація методів розмежування доступу

```
public class SecureCacheProxy<K,V,I> {
   private final Cache<K, CacheValueEntry<V,I>> igniteCache;
   private final Authenticator<I> cacheAuthenticator;
   public SecureCacheProxy(Cache<K, CacheValueEntry<V,I>> igniteCache,
                           Authenticator cacheAuthenticator) {
      this.igniteCache = igniteCache;
      this.cacheAuthenticator = cacheAuthenticator;
   }
   public void put(K key, CacheValueEntry<V,I> value) {
      if(cacheAuthenticator.isCachePermissionsValid(igniteCache.getName(),
CacheAction.WRITE)){
         if(value.isSecured()) {
            value.validateSecurityParams(cacheAuthenticator.getIdentifier(),
CacheAction.WRITE);
         }
            igniteCache.put(key,value);
      }
      else throw new SecurityException("access denied");
   }
   public V get(K key) {
      if(cacheAuthenticator.isCachePermissionsValid(igniteCache.getName(),
CacheAction.READ)){
         System.out.println(igniteCache.get(key));
         CacheValueEntry<V,I> cacheValueEntry = igniteCache.get(key);
         if(cacheValueEntry.isSecured()) {

cacheValueEntry.validateSecurityParams(cacheAuthenticator.getIdentifier(),
CacheAction.READ);
         }
```

```
            return cacheValueEntry.getValue();
        }
        else throw new SecurityException("access denied");
    }
    public String getName() {
        if(cacheAuthenticator.isCachePermissionsValid(igmiteCache.getName(),
CacheAction.READ)) {
            return igniteCache.getName();
        }
        else throw new SecurityException("access denied");
    }
}

public interface Credentials<I> {
    I getIdentifier();
}

public class UsernamePasswordCredentials implements Credentials<String> {

    private final String username;

    private final String password;

    public UsernamePasswordCredentials(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }
```

```
    public String getPassword() {
        return password;
    }
    @Override
    public String getIdentifier() {
        return username;
    }
}
public class Permission<I> {

    private final I identifier;

    private final CacheAction cacheAction;

    public Permission(I identifier, CacheAction cacheAction) {
        this.identifier = identifier;
        this.cacheAction = cacheAction;
    }

    public I getIdentifier() {
        return identifier;
    }

    public CacheAction getCacheAction() {
        return cacheAction;
    }
}public class Role {

    private final String roleName;

    private final List<Role> dependingRoles;
```

```
    public Role(String roleName) {
        this.roleName = roleName;
        this.dependingRoles = Collections.unmodifiableList(new ArrayList<>());
    }


    public Role(String roleName, List<Role> dependingRoles) {
        this.roleName = roleName;
        this.dependingRoles = Collections.unmodifiableList(dependingRoles);
    }

    public List<Role> getDependingRoles() {
        return dependingRoles;
    }

    public String getRoleName() {
        return roleName;
    }

    boolean isLeafRole(){
        return dependingRoles.size() == 0;
    }

}
public class LabelEntryImpl implements LabelEntry {

    private String labelName;

    public LabelEntryImpl( String labelName) {
        this.labelName = labelName;
```

```
    }

    public String getLabelName() {
        return labelName;
    }
}
public class MatrixEntryImpl<I> implements MatrixEntry<I> {
    private final List<Permission<I>> permissions;

    public MatrixEntryImpl( List<Permission<I>> permissions) {
        this.permissions = Collections.unmodifiableList(permissions);
    }
    @Override
    public List<Permission<I>> getPermissions() {
        return permissions;
    }
}
public class RoleEntryImpl implements RoleEntry {
    private Role role;
    public RoleEntryImpl( Role role) {
        this.role = role;
    }
    @Override
    public Role getRole() {
        return role;
    }
}
public interface CacheValueEntry<V, I>  {
     V getValue();
     boolean isSecured();
```

```
    void validateSecurityParams(I identifier, CacheAction cacheAction) throws
SecurityException;
}
public final class MatrixCacheValue<V,I> implements CacheValueEntry<V,I> {
   private final V value;
   private final MatrixEntry<I> matrixEntry;
   public MatrixCacheValue(V value, List<Permission<I>> permissions) {
      this.value = value;
      this.matrixEntry =new MatrixEntryImpl<>(permissions);
   }
   @Override
   public V getValue() {
      return value;
   }
   @Override
   public boolean isSecured() {
      return !matrixEntry.getPermissions().isEmpty();
   }
   @Override
   public void validateSecurityParams(I identifier, CacheAction cacheAction) throws
SecurityException {
      matrixEntry.getPermissions().stream()
            .filter(iPermission -> cacheAction.equals(iPermission.getCacheAction()))
            .filter(iPermission -> iPermission.getIdentifier().equals(identifier))
            .findAny().orElseThrow(()-> new SecurityException("access denied"));
   }
}
public class SimpleCacheValueEntry<V,I> implements CacheValueEntry<V,I> {
   private final V value;
   public SimpleCacheValueEntry(V value) {
      this.value = value;
```

```
    }
    @Override
    public V getValue() {
        return value;
    }
    @Override
    public boolean isSecured() {
        return false;
    }
    @Override
    public void validateSecurityParams(I identifier, CacheAction cacheAction) throws
SecurityException {
        //nothing to validate
    }
}
public enum CacheAction {
    WRITE,UPDATE,READ,DELETE
}
public interface DiscretionalBasedPrincipal< I> extends Principal {
    List<MatrixEntry<I>> getMatrix();
}
public interface MandatoryBasedPrincipal extends Principal {
    List<LabelEntry> getLabels();
}
public interface Principal  {
}
public interface RoleBasedPrincipal extends Principal {
    List<RoleEntry> getRoles();
}
public abstract class AbstractAuthenticator<I,PC extends Principal>
        implements Authenticator<I> {
```

```
    private final Credentials<I> credentials;
    private final CacheAuthMetadataSource<PC, I> cacheAuthMetadataSource;
    private final AuthValidator<PC> authValidator;
    public AbstractAuthenticator(Credentials<I> credentials,
                                 CacheAuthMetadataSource<PC, I> cacheAuthMetadataSource,
                                 AuthValidator<PC> authValidator) {
        this.credentials = credentials;
        this.cacheAuthMetadataSource = cacheAuthMetadataSource;
        this.authValidator = authValidator;
    }
    @Override
    public boolean isCachePermissionsValid(String cacheName, CacheAction
cacheAction) {
        PC cachePrincipalForCredentials =
cacheAuthMetadataSource.getCredentialsPrincipalForCache(credentials,cacheName)
;
        return authValidator.validate(cacheAction, cachePrincipalForCredentials);
    }
    @Override
    public I getIdentifier() {
        return credentials.getIdentifier();
    }
}
public interface Authenticator<I> {
    boolean isCachePermissionsValid(String cacheName, CacheAction cacheAction);

    I getIdentifier();

}
public class DummyAuthenticator<I> extends AbstractAuthenticator<I,
RoleBasedPrincipal> {
```

```
    public DummyAuthenticator(Credentials<I> credentials) {
        super(credentials, new DummyRoleBasedCacheAuthMetadataSource<>(), new
RoleBasedAuthValidator());
    }
}
//used for integration with external source
public interface CacheAuthMetadataSource<P extends Principal, I> {

    P getCredentialsPrincipalForCache(Credentials<I> credentials, String
cacheName);
}
public class DummyRoleBasedCacheAuthMetadataSource<I> implements
CacheAuthMetadataSource<RoleBasedPrincipal, I> {

    private static Map<String, RoleBasedPrincipal> map = new HashMap<>();

    static {
        map.put("user"+ "_"+"CACHE",
                ()->Arrays.asList( new RoleEntryImpl( new Role("ROLE_READ")),new
RoleEntryImpl( new Role("ROLE_WRITE"))  ));
    }
    @Override
    public RoleBasedPrincipal getCredentialsPrincipalForCache(Credentials<I>
credentials, String cacheName) {
        I identifier = credentials.getIdentifier();
        //by credentials identifier and cache name get principal
        return map.get(identifier+"_"+cacheName);
    }

}
public interface AuthValidator<PC extends Principal> {
```

```
    boolean validate(CacheAction cacheAction, PC principal);
}

public class RoleBasedAuthValidator implements
AuthValidator<RoleBasedPrincipal> {
    @Override
    public boolean validate(CacheAction cacheAction, RoleBasedPrincipal principal)
{
        return principal.getRoles().stream()
                .anyMatch(val ->
val.getRole().getRoleName().contains(cacheAction.toString()));
    }
}
public class Main {
    public static void main(String[] args) throws IgniteException {

        Ignite ignite = Ignition.start("ignite-server-local.xml");

        IgniteCache<String,CacheValueEntry<String, String>> cache =
ignite.getOrCreateCache("CACHE");
        SecureCacheProxy<String, String, String> secureCacheProxy =
                new SecureCacheProxy(cache, new DummyAuthenticator(new
UsernamePasswordCredentials("user", "user")));
        secureCacheProxy.put("1",new SimpleCacheValueEntry<>("1"));
        secureCacheProxy.put("2",new MatrixCacheValue<>("1", Arrays.asList(new
Permission("user", CacheAction.WRITE))));
        System.out.println(secureCacheProxy.getName());
        System.out.println(secureCacheProxy.get("1"));
        System.out.println(secureCacheProxy.get("2"));
    }
}
```