Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC з динамічно змінювальним доступом та поведінковою класифікацією загроз»

Виконав: здобувач 2-го курсу,
групи 2КІТС-24м
спеціальності 125– Кібербезпека
та захист інформації
Освітня програма – Кібербезпека
інформаційних технологій та систем

Матвієнко Данило Валерійович

Керівник:
к.т.н., доцент кафедри МБІС Грицак А.В.

«10» грудня 2025 р.

Опонент:
д.ф., доцент кафедри ОТ Обертюх М.Р.
(прізвище та ініціали)

«10» грудня 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
"10" грудня 2025 р.

Вінниця ВНТУ - 2025 рік

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

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

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____ **Юрій ЯРЕМЧУК**
“ 24 ” вересня 2025 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Матвієнко Данило Валерійович

1. Тема роботи «Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC з динамічно змінювальним доступом та поведінковою класифікацією загроз»
Керівник роботи к.т.н., доцент кафедри МБІС Грицак А.В.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “24” вересня 2025 року № 313
2. Строк подання студентом роботи 01.12.2025 р.
3. Вихідні дані до роботи Методичні вказівки до виконання магістерської кваліфікаційної роботи; наукові публікації та монографії з питань захисту системи віртуалізації з контролем доступу
4. Зміст текстової частини 1. Теоретичні засади вдосконалення захисту систем віртуалізації на основі моделі CAPBAC
2. Розробка вдосконаленої моделі захисту системи віртуалізації з контролем доступу на основі CAPBAC
3. Програмна реалізація вдосконаленої моделі захисту системи віртуалізації з контролем доступу на основі CAPBAC
4. Економічна частина
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень) Блок-схема архітектури вдосконаленої системи віртуалізації/захисту; Структурна схема Модуля поведінкової класифікації загроз; Блок-схеми ключових алгоритмів; Матриця управління динамічним доступом CapBAC; Графіки залежності ефективності та накладних витрат; Діаграми розподілу та результатів тестування; Таблиця порівняння вдосконаленої системи з

альтернативними моделями/інструментами захисту; презентація у форматі PowerPoint.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Грицак А.В., к.т.н., доц. каф. МБІС		
II	Грицак А.В., к.т.н., доц. каф. МБІС		
III	Грицак А.В., к.т.н., доц. каф. МБІС		
Економічна частина			
IV	Ратушняк О. Г., к.т.н., доц. каф. ЕПВМ		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1	Отримання завдання, підбір та аналіз літературних джерел за темою дослідження	23.09.2025	06.10.2025	
2	Аналіз існуючих рішень та обґрунтування вибору методу дослідження	07.10.2025	17.10.2025	
3	Розробка математичної моделі та алгоритму вбудовування даних з шифруванням ключ-блоку	18.10.2025	27.10.2025	
4	Програмна реалізація методу та проведення експериментів	28.10.2025	16.11.2025	
5	Підготовка економічної частини	17.11.2025	20.11.2025	
6	Оформлення пояснювальної записки, підготовка графічного матеріалу та презентації	21.11.2025	27.11.2025	
7	Переддипломний захист			
8	Захист магістерської кваліфікаційної роботи			

Студент (підпис) Матвієнко Д. В.

Керівник роботи (підпис) Грицак А. В.

АНОТАЦІЯ

УДК 004.056

Матвієнко Д.В. Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC з динамічно змінювальним доступом та поведінковою класифікацією загроз, 125 – Кібербезпека та захист інформації, освітня програма - Кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2025. 125 с.

На укр. мові. Бібліогр.: 37 назв; рис.: 15; табл. 9.

У роботі проведено дослідження проблем безпеки віртуалізованих середовищ та розроблено вдосконалену модель захисту системи віртуалізації з використанням контролю доступу на основі CapBAC. Особливу увагу приділено створенню механізмів динамічно змінюваного доступу та поведінкової класифікації загроз, що дозволяє оперативно реагувати на аномальні дії користувачів і підвищувати стійкість системи до складних атак.

Метою роботи є підвищення рівня захисту віртуалізованих інфраструктур шляхом інтеграції механізмів CapBAC із системою поведінкової аналітики, яка формує контекстно-орієнтовані політики доступу. У роботі проаналізовано архітектуру сучасних систем віртуалізації, основні вектори атак та недоліки традиційних моделей управління доступом (RBAC, ABAC). На основі цього розроблено модифіковану модель CapBAC із розширеними токенами доступу, прив’язаними до поведінкових параметрів, часу, контексту та рівня ризику.

Ключові слова: віртуалізація, CapBAC, контроль доступу, поведінкова аналітика, динамічні політики, інформаційна безпека.

ABSTRACT

Matviienko D. V. Enhancement of virtualization system security with access control based on the CapBAC model using dynamically adjustable permissions and behavioral threat classification. Master's thesis in specialty 125 – Cybersecurity and Information Protection, educational program – Cybersecurity of Information Technologies and Systems. Vinnytsia: VNTU, 2025. – 125 p.

In Ukrainian language. Bibliography: 37 titles; fig.: 15; tabl.: 9..

This thesis presents a comprehensive study of security challenges in virtualized environments and introduces an enhanced protection model based on Capability-Based Access Control (CapBAC). The proposed solution integrates dynamically adjustable access rights with behavioral threat classification, enabling real-time detection of anomalies and improving the resilience of virtualized systems against sophisticated attacks.

The objective of the research is to strengthen the security of virtualization infrastructures by combining CapBAC mechanisms with a behavioral analytics engine that generates context-aware access policies. The study includes an analysis of modern virtualization architectures, attack vectors, and limitations of traditional access control models such as RBAC and ABAC. Based on this, a modified CapBAC model is developed, featuring extended capability tokens linked to behavioral metrics, temporal constraints, contextual attributes, and risk scores.

Keywords**:** virtualization, CapBAC, access control, behavioral analytics, dynamic policies, cybersecurity.

ЗМІСТ

ВСТУП

Актуальність.

Стрімке зростання використання віртуалізованих інфраструктур у корпоративних та хмарних середовищах зумовлює потребу у підвищенні рівня їх захищеності. Віртуалізація дозволяє оптимізувати використання ресурсів, масштабувати сервіси та знижувати витрати, однак одночасно створює нові загрози, пов’язані з компрометацією гіпервізора, розширеною площею атаки, міжвіртуальними перехресними впливами та складністю управління доступом у динамічних умовах. Традиційні моделі контролю доступу — RBAC та ABAC — не повною мірою відповідають вимогам таких систем, оскільки вони не враховують швидкі зміни контексту та поведінкові характеристики користувачів.

На цьому тлі актуальним є використання моделі CapBAC, заснованої на механізмах капабіліті-токенів, яка надає можливість забезпечити гнучкий, контекстно залежний доступ. Однак класичний CapBAC також потребує доопрацювання, зокрема впровадження поведінкової аналітики, адаптації політик у реальному часі та механізмів динамічного контролю доступу. Саме тому вдосконалення системи віртуалізації на основі CapBAC із включенням поведінкової класифікації загроз є актуальним напрямом досліджень і важливою практичною задачею сучасної кібербезпеки.

Мета і задачі дослідження.

Метою дипломної роботи є підвищення рівня захисту системи віртуалізації шляхом розробки та впровадження вдосконаленої моделі контролю доступу на основі CapBAC з динамічно змінюваним доступом і поведінковою класифікацією загроз.

Для досягнення поставленої мети необхідно виконати такі задачі дослідження:

1. Проаналізувати архітектурні особливості сучасних систем віртуалізації та визначити основні загрози їх безпеці.

2. Дослідити існуючі моделі управління доступом (RBAC, ABAC, CapBAC) та оцінити їх придатність для віртуалізованих середовищ.
3. Обґрунтувати вибір моделі CapBAC як основи для побудови адаптивної системи захисту.
4. Розробити модифіковану модель CapBAC з підтримкою динамічно змінюваного доступу та поведінкових параметрів.
5. Створити алгоритм прийняття рішень щодо доступу на основі поведінкової аналітики.
6. Реалізувати програмні модулі керування токенами, обробки поведінкових даних та адаптації політик.
7. Провести тестування розробленої системи та оцінити її ефективність у різних сценаріях доступу.

Об’єкт дослідження — процес забезпечення безпеки віртуалізованих інфраструктур.

Предмет дослідження — методи та моделі контролю доступу на основі CapBAC із динамічною адаптацією політик та поведінковою класифікацією загроз.

Новизна роботи.

Новизна дипломної роботи полягає у створенні вдосконаленої моделі CapBAC, яка поєднує капабіліті-токени з поведінковими метриками та механізмами динамічного управління доступом. Запропоновано новий підхід до адаптації політик у реальному часі, що враховує ризиковість дій користувача, контекст взаємодії та рівень аномальності поведінки. На відміну від класичних рішень, модель забезпечує вищу стійкість до атак повтору, підміни, компрометації сесій і внутрішніх загроз.

Практична цінність.

Практична цінність роботи полягає у можливості впровадження розробленої моделі у корпоративні системи віртуалізації для підвищення рівня їх захищеності. Реалізовані програмні модулі можуть бути інтегровані у існуючі системи доступу, забезпечуючи адаптивне управління привілеями, раннє виявлення аномалій та запобігання несанкціонованому доступу. Отримані результати можуть бути

використані у хмарних сервісах, дата-центрах, інфраструктурах DevOps та інших середовищах, що потребують високого рівня динамічної безпеки.

1 ТЕОРЕТИЧНІ ЗАСАДИ ВДОСКОНАЛЕННЯ ЗАХИСТУ СИСТЕМ ВІРТУАЛІЗАЦІЇ НА ОСНОВІ МОДЕЛІ CAPBAC

У цьому розділі розглядаються основні теоретичні положення, що лежать в основі захисту систем віртуалізації. Проаналізовано сучасні архітектури віртуалізованих середовищ, ключові проблеми їх безпеки, а також моделі контролю доступу — RBAC, ABAC та CapBAC. Особливу увагу приділено принципам динамічного управління доступом і можливостям використання поведінкових характеристик користувачів для підвищення рівня захисту.

Результати аналізу формують теоретичне підґрунтя для подальшої розробки вдосконаленої моделі контролю доступу на основі CapBAC з динамічно змінюваними правами та поведінковою класифікацією загроз.

1.1 Сучасні системи віртуалізації та їх архітектурні особливості

Віртуалізація — це технологія абстрагування фізичних ресурсів комп’ютера (процесора, пам’яті, сховища, мережі) для створення множини ізольованих середовищ виконання, які працюють незалежно одна від одної. Основна мета віртуалізації — ефективніше використання апаратних ресурсів і підвищення гнучкості системи.

Формально, процес віртуалізації можна описати функцією відображення:

V: R \rightarrow \{r_1, r_2, ..., r_n\}, \quad \text{де } R — множина фізичних ресурсів, а } r_i — віртуальні представлення.

Кожне r_i ізольоване, має власне ядро, драйвери та ресурси, але фактично спирається на ту саму фізичну інфраструктуру [1].

Віртуалізацію поділяють на кілька типів:

— повна віртуалізація (Full Virtualization) – віртуальна машина повністю емує фізичний комп’ютер. Приклад: VMware ESXi, Microsoft Hyper-V;

— паравіртуалізація (Paravirtualization) – гостьова ОС "знає", що працює у віртуальному середовищі, і виконує спеціальні системні виклики (hypercalls). Приклад: Xen;

— віртуалізація на рівні ОС (Containerization) – ядро спільне для всіх контейнерів, але кожен контейнер має власний простір користувача. Приклад: Docker, LXC;

— віртуалізація з апаратною підтримкою (Hardware-assisted) – реалізована через інструкції CPU (Intel VT-x, AMD-V), які пришвидшують перехід між контекстами гостьової і хостової систем.

Архітектура системи віртуалізації визначає спосіб взаємодії між апаратними ресурсами, гіпервізором, гостьовими операційними системами та прикладними рівнями. Вона забезпечує поділ ресурсів між ізольованими віртуальними середовищами та гарантує безпечне управління процесами, пам’яттю, мережею та пристроями введення-виведення [2].

Основна ідея полягає у створенні шару віртуалізації (гіпервізора), який виступає посередником між фізичним обладнанням та віртуальними машинами, перехоплює системні виклики гостьових ОС і забезпечує контроль над доступом до апаратних ресурсів.

Нижче наведено узагальнену схему архітектури системи віртуалізації.

Рисунок 1.1 – Архітектура системи віртуалізації

Показана структура складається з кількох основних рівнів:

— апаратний рівень — фізичне обладнання, що включає процесори, оперативну пам’ять, дискові накопичувачі, мережеві адаптери та контролери введення-виведення;

— рівень гіпервізора (Virtual Machine Monitor, VMM) — програмно-апаратний шар, що здійснює розподіл ресурсів і контроль виконання гостьових ОС;

— рівень гостьових операційних систем (Guest OS) — ізольовані середовища, які працюють як незалежні віртуальні машини;

— рівень прикладних систем (Application Layer) — користувацькі програми, сервіси або контейнери, які функціонують у межах гостьових систем.

Таким чином, гіпервізор виступає центральною ланкою архітектури віртуалізації, оскільки саме він забезпечує відокремлення ресурсів, контроль доступу до апаратури та ізоляцію між віртуальними машинами. Правильна побудова цієї архітектури визначає як ефективність використання фізичних ресурсів, так і рівень безпеки всієї системи [3].

Системи віртуалізації реалізуються на основі різних типів гіпервізорів (Virtual Machine Monitor, VMM), що визначають спосіб доступу до апаратних ресурсів і рівень інтеграції з хостовою операційною системою.

Гіпервізори можна класифікувати за способом розміщення у системі: тип 1 (bare-metal) — працює безпосередньо на апаратному забезпеченні, та тип 2 (hosted) — функціонує поверх хостової операційної системи [4].

Порівняльні характеристики цих типів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика типів гіпервізорів

Тип	Опис	Приклади	Переваги	Недоліки
Тип 1 (bare-metal)	Працює безпосередньо на апаратному рівні без проміжної ОС.	VMware ESXi, Microsoft Hyper-V Server, Xen	Висока продуктивність, ізоляція, стабільність, безпека	Складність налаштування, потребує спеціального обладнання

Продовження таблиці 1.1

Тип 2 (hosted)	Запускається поверх хостової ОС як звичайна програма.	VirtualBox, VMware Workstation, Parallels Desktop	Простота встановлення, гнучкість, сумісність із настільними системами	Нижча продуктивність, більша поверхня атаки, залежність від хостової ОС

Порівняння показує, що гіпервізори типу 1 частіше застосовуються у корпоративних і хмарних середовищах, де важливі стабільність та безпека, тоді як тип 2 — у навчальних, тестових і розробницьких середовищах, де пріоритетом є гнучкість та простота налаштування.

У контексті захисту систем віртуалізації гіпервізори першого типу вважаються більш надійними, оскільки мінімізують кількість проміжних шарів і, відповідно, зменшують потенційну поверхню атаки [5].

Архітектурні компоненти гіпервізора:

— Virtual Machine Monitor (VMM) — забезпечує керування процесами, пам’яттю, I/O;

— Device Emulator — емуляція апаратних пристроїв (мережевих, графічних, дискових);

— Resource Scheduler — розподіл фізичних ресурсів між гостьовими системами;

— Security Module (SM) — компонент безпеки, що відповідає за ізоляцію та контроль доступу.

Процес управління ресурсами є одним із ключових аспектів у системах віртуалізації, оскільки саме він визначає ефективність використання апаратної бази та стабільність роботи гостьових систем.

Задача розподілу ресурсів формулюється як оптимізаційна — необхідно забезпечити максимальну ефективність використання фізичних ресурсів при дотриманні обмежень на їх кількість [6].

Математично процес можна описати наступним чином:

$$\max \{\sum\_\{i = 1\}\^\{n\}w\_i\} \cdot U\_i, \quad \text{Ыr}\{\text{за умов } \sum_{i=1}^{n} r_i \leq R, (1.1)$$

де

Ui — ефективність використання ресурсів VM_i,

wi — ваговий коефіцієнт пріоритету,

ri — кількість ресурсів, виділених цій машині (наприклад, обсяг оперативної пам’яті або частка процесорного часу);

R — загальний ресурс (наприклад, час процесора або пам’ять).

Виконання цієї умови гарантує, що сумарне навантаження на систему не перевищуватиме її фізичних можливостей, а пріоритетні віртуальні машини отримають більшу частку ресурсів.

Подібні алгоритми застосовуються у більшості сучасних гіпервізорів (наприклад, VMware ESXi чи KVM) для балансування продуктивності між гостьовими системами та динамічного перерозподілу ресурсів при зміні навантаження.

Таким чином, формалізація процесу розподілу ресурсів дає можливість створювати адаптивні та стабільні середовища віртуалізації, де кожна віртуальна машина отримує необхідну кількість обчислювальних потужностей, не порушуючи загальної рівноваги системи [7].

На сьогодні існує велика кількість програмних платформ, що реалізують віртуалізацію на різних рівнях — від апаратної до контейнерної. Вибір конкретної платформи залежить від вимог до продуктивності, масштабованості, рівня безпеки та сумісності з інфраструктурою підприємства.

До найбільш поширених належать VMware ESXi, Microsoft Hyper-V, KVM, Xen, Docker, LXC та OpenStack.

Порівняльні характеристики основних платформ наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика сучасних платформ віртуалізації

Платформа	Тип віртуалізації	Основні особливості	Підтримка безпеки
VMware ESXi	Bare-metal	Професійна корпоративна платформа; підтримка кластеризації, vMotion, fault tolerance	Захист через vShield, шифрування трафіку, ізоляція віртуальних машин
Microsoft Hyper-V	Bare-metal / Hosted	Інтеграція з Windows Server, підтримка віртуальних комутаторів, кластеризація	Керування доступом через Active Directory, BitLocker для шифрування
KVM (Kernel-based Virtual Machine)	Kernel-level	Інтегрований у ядро Linux, підтримка live migration, модульна архітектура	SELinux, sVirt, AppArmor, обмеження прав процесів
Xen	Paravirtualization	Архітектура Dom0 / DomU, гнучке керування ізоляцією	Розділення привілеїв, безпечне ядро гіпервізора
Docker / LXC	Containerization	Легка віртуалізація на рівні ОС; спільне ядро для всіх контейнерів	Контроль через cgroups, namespaces, seccomp, SELinux
OpenStack / Proxmox VE	Cloud orchestration	Підтримка масштабованих кластерів, інтеграція з Ceph, ZFS	Keystone (автентифікація), Neutron (мережевий контроль), TLS-шифрування

Порівняльний аналіз показує, що корпоративні рішення (VMware ESXi, Hyper-V) забезпечують високу надійність та широкий спектр засобів управління, тоді як відкриті платформи (KVM, Xen, OpenStack) орієнтовані на гнучкість і масштабування у середовищах з відкритим кодом [8].

Контейнеризаційні технології (Docker, LXC) демонструють найвищу ефективність у плані використання ресурсів, проте потребують додаткових заходів ізоляції та моніторингу безпеки, оскільки базуються на спільному ядрі хостової ОС.

Таким чином, вибір платформи віртуалізації значною мірою визначає архітектурну модель майбутньої системи, її продуктивність, масштабованість і ступінь захищеності від потенційних загроз.

Попри ізоляцію, віртуалізація має притаманні ризики:

— Escape-атаки (VM Escape): коли зловмисник виходить із гостьової ОС до хостової;

— Shared Resource Leakage: через спільне використання кешів, пам’яті, мережевих буферів;

— Hypervisor Compromise: якщо гіпервізор зламано — компрометуються всі гості;

— Insecure APIs: у контейнеризації (Docker, Kubernetes) — через відкриті REST-інтерфейси;

— Вразливість у системах з розширеним доступом (CapBAC, ABAC): неправильне управління токенами доступу.

Сучасні системи віртуалізації є основою хмарних і корпоративних середовищ, забезпечуючи масштабованість, високу доступність і контроль ресурсів. Однак збільшення рівня абстракції породжує додаткові ризики безпеки — особливо при використанні спільних апаратних компонентів і складних моделей доступу.

1.2 Проблеми безпеки у віртуалізованих середовищах та основні вектори атак

Попри численні переваги віртуалізації — ефективне використання ресурсів, масштабованість, централізоване управління — ці системи мають уразливості, що роблять їх привабливою ціллю для зловмисників. У віртуалізованих середовищах традиційні механізми безпеки, орієнтовані на окремі фізичні машини, часто виявляються недостатніми, оскільки віртуальні машини поділяють спільні апаратні ресурси, пам’ять, мережеві інтерфейси та API гіпервізора [9].

Проблеми безпеки в таких системах умовно можна поділити на три основні групи:

— уразливості гіпервізора — компрометація центрального шару управління (VMM) призводить до повного контролю над усіма гостьовими системами;

— недоліки ізоляції між віртуальними машинами — через спільне використання ресурсів можливі атаки побічних каналів (side-channel attacks);

— небезпечна інтеграція з мережею та контейнерними сервісами — атаки на API, мережеві тунелі, оркестратори (наприклад, Kubernetes, OpenStack) [9].

Проблеми безпеки віртуалізованих середовищ охоплюють уразливості на рівнях гіпервізора, гостьових систем, контейнерів та мережевої інфраструктури. Для зручності їх можна класифікувати за основними напрямами впливу, що наведено у таблиці 1.3.

Таблиця 1.3 – Основні проблеми безпеки у віртуалізованих середовищах

Категорія	Суть проблеми	Приклади / Наслідки	Типові атаки
Компрометація гіпервізора	Уразливість центрального шару управління, що забезпечує розподіл ресурсів	Повний контроль над усіма віртуальними машинами	VM Escape, VENOM (CVE-2015-3456), Cloudburst
Недостатня ізоляція між VM	Спільне використання кешу, пам’яті, пристроїв введення-виведення	Витік даних між віртуальними машинами	Side-channel, Spectre, Meltdown
Ненадійна взаємодія через API	Некоректна автентифікація чи шифрування у REST-інтерфейсах керування	Несанкціоноване керування VM або контейнерами	API Exploits, Session Hijacking
Уразливості контейнерного рівня	Контейнери поділяють спільне ядро ОС	Вихід за межі ізоляції контейнера	Container Breakout, RunC Exploit
Помилки конфігурації та надлишкові привілеї	Використання статичних політик доступу або відкритих портів	Несанкціонований доступ користувачів, внутрішні атаки	Misconfiguration, Privilege Escalation

Як видно з таблиці, головною проблемою є порушення ізоляції, що виникає як на рівні гіпервізора, так і між віртуальними машинами та контейнерами. Компрометація гіпервізора — найнебезпечніший сценарій, оскільки дозволяє повний контроль над середовищем.

Крім технічних аспектів, важливу роль відіграє людський фактор: помилки адміністрування, некоректні політики доступу та несвоєчасне оновлення систем безпеки. Ці ризики особливо актуальні для великих хмарних платформ, де навіть невелика помилка у налаштуваннях може призвести до масових компрометацій віртуальних машин [10].

Таким чином, ефективний захист віртуалізованих середовищ потребує багаторівневого підходу — від ізоляції на рівні гіпервізора до динамічного контролю доступу та моніторингу поведінкових відхилень користувачів.

Для наочності на рисунку 1.2 подано основні вектори атак у типових віртуалізованих середовищах.

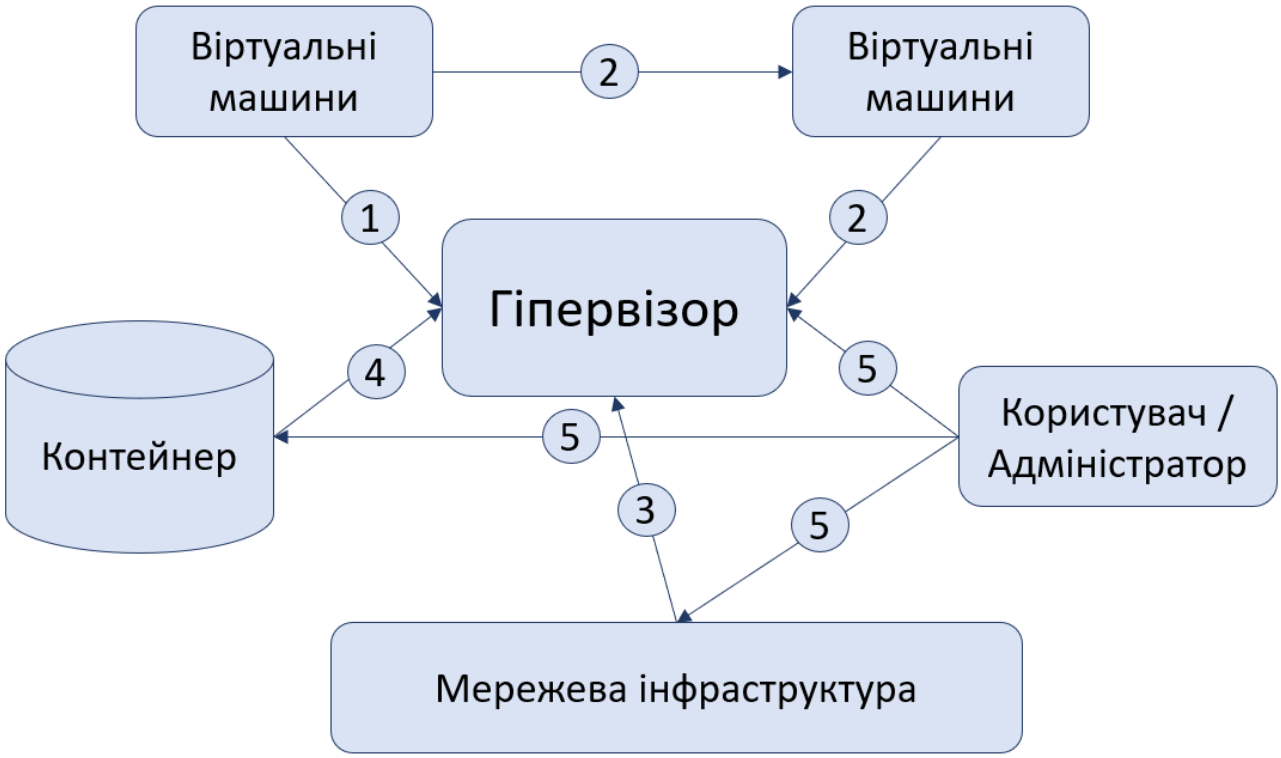


Рисунок 1.2 – Основні вектори атак у системі віртуалізації

Передбачено такі напрямки впливу:

— атака на гіпервізор (1) — через уразливість у драйверах або API управління;

— міжвіртуальні атаки (2) — через побічні канали або спільні ресурси;

— атаки на мережевий стек (3) — через віртуальні комутатори або маршрутизатори;

— компрометація контейнерного середовища (4) — вихід із контейнера на рівень хостової ОС;

— атаки з боку адміністратора або користувача (5) — помилки конфігурації, несанкціоновані дії, викрадення облікових даних.

Значну частку інцидентів у віртуалізованих середовищах становлять внутрішні загрози (Insider Threats).

Адміністратори з розширеними привілеями мають прямий доступ до гіпервізора, систем зберігання та конфігураційних файлів.

У разі компрометації їхніх облікових даних або навмисних дій зловмисників виникає ризик повного контролю над усіма віртуальними машинами.

Згідно з дослідженнями Gartner, близько 60% інцидентів у корпоративних хмарах спричинені саме внутрішніми помилками або зловживаннями правами доступу [11].

Сукупність цих векторів створює комплексну поверхню атаки, що значно перевищує традиційні сценарії у звичайних (не віртуалізованих) інфраструктурах.

У хмарних інфраструктурах на основі віртуалізації проблеми безпеки набувають ще більшої складності через спільне використання апаратних ресурсів різними клієнтами [12].

Так звана модель спільної відповідальності (Shared Responsibility Model) передбачає, що провайдер відповідає за безпеку фізичної інфраструктури, а користувач — за конфігурацію віртуальних машин і контейнерів.

Помилки у налаштуванні, відкриті порти, слабкі паролі або некоректне використання API створюють нові вектори атак навіть у захищених гіпервізорах.

Таким чином, основна загроза хмарних середовищ полягає не у фізичних вразливостях, а у неправильному управлінні віртуалізованими компонентами.

З метою кількісного оцінювання рівня загроз у віртуалізованих середовищах доцільно формалізувати поняття ризику. Такий підхід дозволяє не лише описати вразливості якісно, а й визначити їх відносну важливість для всієї системи.

Оцінювання ризику базується на поєднанні трьох ключових параметрів: ймовірності реалізації загрози, ступеня вразливості конкретного компонента та впливу атаки на загальний стан безпеки системи [13].

Рівень ризику для кожного компонента системи віртуалізації можна описати як функцію від трьох змінних:

$$R_i = P_i \times I_i \times V_i, (1.2)$$

де:

Ri — загальний ризик для *i*-го компонента (гіпервізора, VM або мережевого шару);

Pi — ймовірність реалізації загрози;

Ii — рівень впливу атаки;

Vi — вразливість компонента (ступінь захищеності).

Високе значення будь-якого з параметрів (особливо V_i) істотно підвищує загальний ризик системи, тому у практиці кіберзахисту віртуалізованих платформ першочерговими завданнями є зниження ймовірності експлуатації вразливостей гіпервізора та мінімізація впливу потенційних атак через багаторівневу ізоляцію.

Узагальнюючи викладене, можна стверджувати, що більшість сучасних загроз у віртуалізованих середовищах пов’язані не лише з технічними вразливостями, а й з відсутністю адаптивних механізмів доступу.

Традиційні моделі безпеки не враховують зміну контексту, поведінку користувача та стан системи, що призводить до надлишкових привілеїв і збільшує ризик несанкціонованого доступу [14].

Це підкреслює необхідність удосконалення механізмів контролю доступу, які можуть динамічно змінювати політики відповідно до реальних умов роботи системи.

Системи віртуалізації, попри високий рівень гнучкості та масштабованості, мають складну архітектуру, яка створює розширену поверхню атаки.

Основними проблемами безпеки залишаються компрометація гіпервізора, недостатня ізоляція між віртуальними машинами, атаки через побічні канали, а також помилки конфігурації API та мережевих сервісів [15].

Ці фактори визначають необхідність створення вдосконалених моделей контролю доступу, які враховують контекст, динаміку поведінки користувачів і стан системи, що розглядатиметься у наступних підрозділах.

1.3 Моделі контролю доступу у віртуалізованих середовищах: RBAC, ABAC, CapBAC

Контроль доступу є центральним елементом систем безпеки у віртуалізованих середовищах. Його завданням є визначення, хто має доступ до ресурсів, які саме дії дозволено виконувати, та в яких умовах.

У контексті віртуалізації це питання ускладнюється багаторівневою архітектурою, де одночасно існують права користувачів, віртуальних машин, контейнерів, адміністраторів гіпервізора та сервісних облікових записів.

Для реалізації політик доступу в таких середовищах найчастіше застосовуються три підходи: рольова (RBAC), атрибутивна (ABAC) та можливісна (CapBAC) моделі контролю доступу. Кожна з них має власні переваги, недоліки та специфіку реалізації у віртуалізованих інфраструктурах [17].

Рольова модель контролю доступу (Role-Based Access Control, RBAC) є однією з найпоширеніших і класичних моделей керування правами користувачів у корпоративних та віртуалізованих середовищах. Її принцип ґрунтується на призначенні користувачеві не конкретних дозволів, а певних ролей, кожна з яких має визначений набір прав доступу до ресурсів системи.

У контексті віртуалізації RBAC застосовується для управління доступом до гіпервізора, віртуальних машин, контейнерів та інтерфейсів адміністрування (наприклад, VMware vCenter, Microsoft Hyper-V Manager, OpenStack Keystone).

Основний принцип рольової моделі полягає в тому, що права доступу визначаються не індивідуально для кожного користувача, а через роль, яку він виконує в системі [19].

Кожна роль відповідає певному набору дозволених операцій або ресурсів, що забезпечує централізоване керування доступом і спрощує адміністрування великих віртуалізованих інфраструктур.

Цей підхід дозволяє легко масштабувати політики безпеки, адже при зміні обов'язків користувача достатньо змінити його роль, а не весь перелік дозволів.

Для формального представлення взаємозв'язку між користувачами, ролями та дозволами застосовується математична модель, наведена нижче.

У загальному вигляді модель RBAC описується п'ятіркою множин:

$$\text{RBAC} = (U, R, P, S, UA, PA), \tag{1.3}$$

де:

U — множина користувачів (Users),

R — множина ролей (Roles),

P — множина дозволів (Permissions),

S — множина сесій (Sessions),

UA $\subseteq U \times R$ — відношення «користувач ↔ роль»,

PA $\subseteq R \times P$ — відношення «роль ↔ дозвіл».

Користувач отримує доступ до ресурсу не напряму, а через роль, яка визначає перелік дозволених дій. Таким чином, доступ реалізується за схемою:

$$User \rightarrow^{UA} Role \rightarrow^{PA} Permission$$

Подібна структура робить модель RBAC інтуїтивно зрозумілою та ефективною в управлінні великими наборами користувачів і ресурсів.

Вона дозволяє централізовано контролювати політики доступу, швидко змінювати права для груп користувачів та забезпечувати єдині стандарти безпеки у масштабних віртуалізованих інфраструктурах [18].

Завдяки чіткій ієрархії ролей можливо уникнути дублювання дозволів, а також оперативно реагувати на зміни організаційної структури або призначення користувачів на нові посади.

У віртуалізованих середовищах це особливо важливо, оскільки дозволяє ефективно керувати доступом до ресурсів різного рівня — від гіпервізора до окремих віртуальних машин і контейнерів.

Для кращого розуміння структури рольового розподілу доступу у віртуалізованому середовищі доцільно розглянути типову архітектуру моделі RBAC [19].

На рисунку 1.3 нижче показано основні ролі користувачів, рівні їх взаємодії з компонентами системи та характер доступу до віртуальних ресурсів.

Схема ілюструє, як за допомогою рольового підходу реалізується контроль над різними рівнями інфраструктури — від гіпервізора до окремих віртуальних машин і контейнерів [20].

Типова архітектура RBAC у системах віртуалізації включає:

— адміністратора гіпервізора — має повний контроль над створенням, міграцією та зупинкою віртуальних машин;

— оператора інфраструктури — керує мережею та сховищами;

— користувача віртуальної машини — має доступ лише до своєї VM або контейнера;

— аудитора — може переглядати журнали без можливості змін.

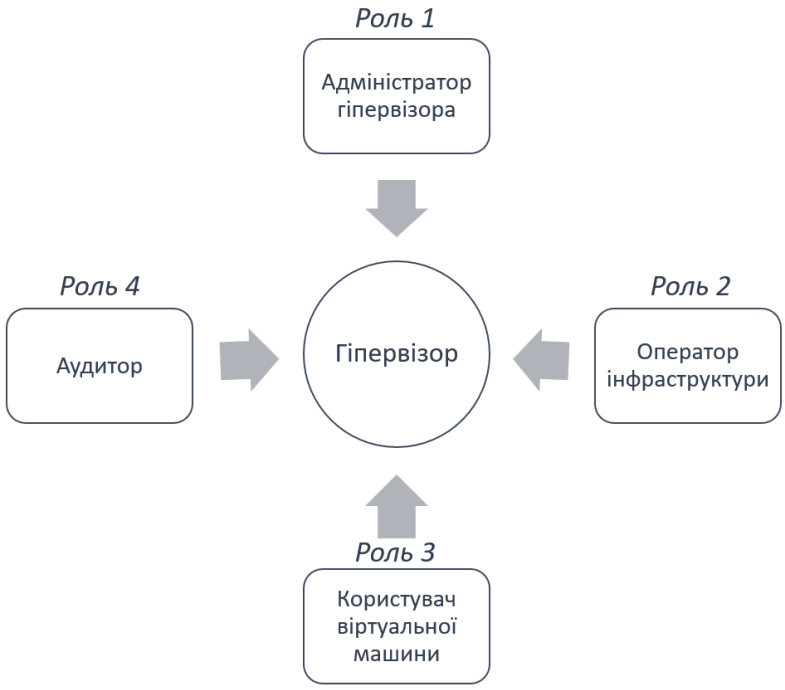


Рисунок 1.3 – Архітектура рольової моделі RBAC у віртуалізованому середовищі

Схема відображає ієрархічний розподіл повноважень у системі віртуалізації. Кожна роль має доступ лише до визначеного рівня інфраструктури, що дозволяє реалізувати принцип найменших привілеїв (Least Privilege Principle).

Завдяки цьому забезпечується ізоляція адміністративних функцій, зменшується ризик конфлікту доступів і підвищується загальна керованість безпекою в системі.

Переваги RBAC

— простота управління — дозволи призначаються групам (ролям), а не окремим користувачам.

— масштабованість — легко адаптується до великих організацій і хмарних систем.

— відповідність стандартам безпеки — RBAC визначена у стандарті ANSI INCITS 359-2004 і підтримується більшістю систем віртуалізації.

— зручність аудиту — дії користувачів можна відслідковувати відповідно до їхніх ролей.

Попри ефективність, рольова модель має певні обмеження, особливо у динамічних середовищах:

— вона не враховує контекст, наприклад: час доступу, місце входу чи поточний стан системи.

— може виникати проблема надлишкових привілеїв, коли ролі стають занадто загальними.

— у великих інфраструктурах кількість ролей може перевищувати кількість користувачів, що ускладнює адміністрування (так звана role explosion).

Для подолання цих обмежень були розроблені атрибутна (ABAC) та можливісна (CapBAC) моделі контролю доступу, які забезпечують більшу гнучкість і контекстну адаптацію прав.

Модель RBAC залишається фундаментальною основою для побудови систем керування доступом у віртуалізованих середовищах [21].

Вона забезпечує структуроване призначення прав і мінімізує ризик випадкового доступу, однак її статичний характер не дозволяє повною мірою враховувати поведінкові фактори користувачів чи зміни стану системи в реальному часі.

Це зумовлює перехід до більш адаптивних моделей контролю — таких як ABAC і CapBAC, розглянутих у наступних підрозділах.

Атрибутна модель контролю доступу (Attribute-Based Access Control, ABAC) є більш гнучкою еволюцією рольової моделі, яка визначає права користувачів не лише через фіксовані ролі, а за допомогою наборів атрибутів — характеристик суб’єкта, об’єкта, середовища та дії [22].

Такий підхід дає змогу приймати рішення щодо доступу динамічно, враховуючи контекст запиту, політики безпеки й поточний стан системи.

У віртуалізованих середовищах модель ABAC дозволяє створювати контекстно-залежні правила доступу: наприклад, користувач може запускати віртуальну машину лише у робочий час, із корпоративної мережі та за умови, що її стан позначено як безпечний.

На відміну від рольового підходу, де доступ визначається фіксованими наборами прав, атрибутна модель базується на динамічному аналізі властивостей суб’єкта, об’єкта й середовища.

Рішення про дозвіл дії приймається не наперед, а в момент запиту — залежно від актуальних параметрів системи [24].

Такий підхід дозволяє автоматично реагувати на зміну умов, наприклад, рівня довіри користувача, безпеки віртуальної машини чи часу виконання операції.

Для формалізації взаємозв’язку між цими параметрами використовується математична модель, що наведена нижче.

Формально атрибутна модель визначається чотирма множинами:

$$ABAC = (S, O, A, E), \tag{1.4}$$

де:

S — множина суб’єктів (subjects),

O — множина об’єктів (objects),

A — множина атрибутів (attributes),

E — множина правил або політик (environment/policies).

Рішення про дозвіл чи заборону дії приймається на основі логічного виразу:

$$Access = f(S_a, O_a, E_a), \tag{1.5}$$

де S_a, O_a, E_a — атрибути суб’єкта, об’єкта та середовища відповідно.

Якщо функція f повертає істину (True), доступ дозволено, інакше — заборонено.

Це дозволяє гнучко формувати політики на основі десятків параметрів, а не лише ролей.

Приклади застосування ABAC у віртуалізованих середовищах:

— OpenStack Keystone — використовує політики на основі атрибутів користувача, ролі проєкту, типу операції та стану середовища.

— AWS Identity and Access Management (IAM) — визначає доступ через атрибути користувача (user tags), IP-адресу, геолокацію, час доби та тип запиту.

— Microsoft Azure Active Directory — застосовує умовні політики доступу, де рішення залежить від стану пристрою, автентифікації та місця розташування користувача.

Таким чином, ABAC є ключовим механізмом у сучасних хмарних сервісах, які активно використовують віртуалізацію.

Типова архітектура ABAC складається з таких компонентів:

— PIP (Policy Information Point) — збирає атрибути про користувача, об’єкт і середовище (наприклад, стан VM, IP-адреса, рівень довіри).

— PDP (Policy Decision Point) — приймає рішення на основі політик і наданих атрибутів.

— PEP (Policy Enforcement Point) — реалізує рішення, дозволяючи або забороняючи дію.

— PAP (Policy Administration Point) — адмініструє політики безпеки та визначає логіку дозволів.

На рисунку 1.4 наведено архітектуру атрибутної моделі контролю доступу (ABAC), що описує основні компоненти та їх взаємодію в процесі прийняття рішення щодо доступу.

Система складається з чотирьох логічних елементів — PAP, PDP, PEP і PIP — які спільно забезпечують збір атрибутів, обробку політик і застосування прийнятих рішень.

Модель враховує характеристики суб’єкта, об’єкта та поточного стану середовища, що дозволяє реалізувати гнучкий контекстно-залежний контроль доступу.

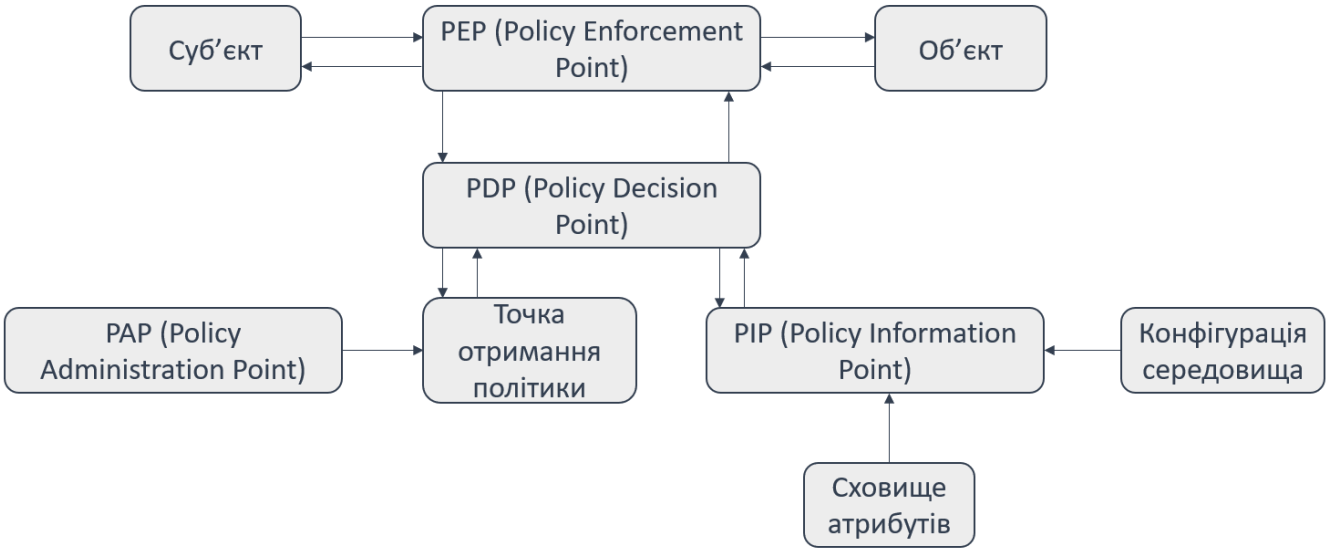


Рисунок 1.4 – Архітектура атрибутної моделі контролю доступу ABAC

Як показано на рисунку, запит від суб'єкта надходить до PEP (Policy Enforcement Point), який ініціює перевірку політики в PDP (Policy Decision Point).

Модуль PDP аналізує правила, отримані від PAP (Policy Administration Point), та запитує необхідні атрибути через PIP (Policy Information Point) — із джерел, таких як сховище атрибутів або конфігурація середовища.

Після обробки PDP повертає рішення («дозволити» або «заборонити») до PEP, який безпосередньо застосовує його до відповідного об'єкта.

Така архітектура забезпечує динамічний контроль доступу з урахуванням змін контексту, підвищує точність і гнучкість політик безпеки та дозволяє інтегрувати ABAC у віртуалізовані або хмарні середовища без необхідності ручного оновлення ролей чи списків доступу [25].

Переваги моделі ABAC:

— гнучкість і контекстність — рішення приймаються залежно від ситуації, а не лише від ролі користувача;

— масштабованість — додавання нових атрибутів не вимагає зміни всієї структури політик;

— високий рівень деталізації — дозволяє враховувати десятки параметрів без надмірної кількості ролей;

— підвищена безпека — знижує ризик надлишкових прав та покращує контроль над складними середовищами.

Недоліки моделі ABAC:

— складність реалізації — потребує точного визначення атрибутів і логіки політик.

— високі вимоги до продуктивності — через постійне оцінювання атрибутів і контексту.

— необхідність централізованої довіри — усі атрибути мають бути достовірними та захищеними від підробки.

Модель ABAC забезпечує динамічне, контекстно-залежне управління доступом, що особливо важливо у віртуалізованих і хмарних середовищах.

На відміну від RBAC, вона дозволяє враховувати широкий спектр характеристик користувача й середовища, проте вимагає складнішої реалізації.

ABAC часто використовується як проміжний етап перед переходом до CapBAC — моделі, де контроль доступу базується не лише на атрибутах, а й на криптографічних можливостях (capabilities), що дозволяє ще більше підвищити рівень безпеки.

Система CapBAC (Capability-Based Access Control) — це підхід до контролю доступу, у якому рішення про дозвіл чи заборону дії приймається на основі повноважень (capabilities) — спеціальних маркерів, що одночасно ідентифікують користувача, і визначають його права щодо конкретного ресурсу.

На відміну від RBAC та ABAC, де права описуються через ролі або політики, у CapBAC права вбудовані в самі токени доступу, що передаються суб’єкту для взаємодії з об’єктом.

Ця модель є природною для розподілених систем, віртуалізованих середовищ та IoT-інфраструктур, оскільки дозволяє мінімізувати централізовані перевірки та забезпечити швидкий, децентралізований контроль доступу [26].

У загальному вигляді модель CapBAC може бути формалізована як система, у якій кожен суб'єкт S_iмає набір повноважень C_i, що визначають дозволені дії над ресурсами R_j:

$$C_i = \{(R_j, A_k, t)\} \quad (1.6)$$

де

R_j— ресурс або об'єкт доступу;

A_k— дозволена дія (читання, запис, виконання тощо);

t— часовий або контекстний параметр, що обмежує дію повноваження.

Таким чином, для перевірки доступу система не звертається до централізованої бази ролей чи політик, а перевіряє валідність самого токена C_i, який може бути реалізований у вигляді:

— криптографічно підписаного токена (наприклад, JSON Web Token або OAuth-токена);

— цифрового сертифіката з вбудованими атрибутами;

— апаратного токена або capability-ключа на пристроїаної сторони.

На рисунку наведено архітектуру моделі контролю доступу на основі повноважень (CapBAC), адаптовану для віртуалізованого середовища.

У цій моделі кожен користувач отримує криптографічно захищене повноваження (capability-токен), яке містить інформацію про дозволені дії над конкретним ресурсом.

Процес контролю доступу відбувається децентралізовано — без необхідності постійного звернення до централізованого сервера політик, що підвищує ефективність і швидкодію системи.

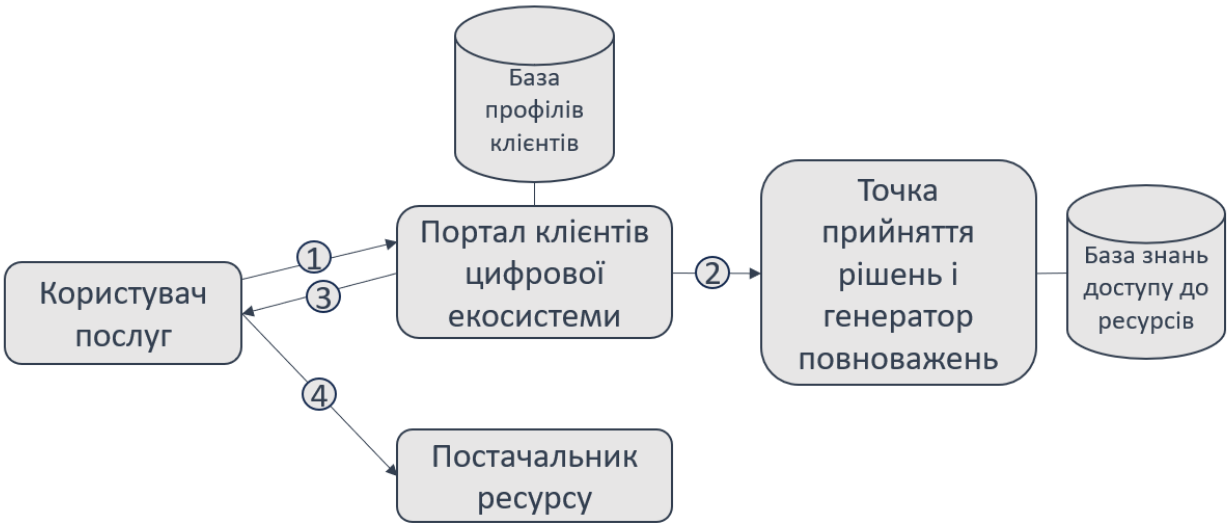


Рисунок 1.5 – Архітектура моделі контролю доступу CapBAC у віртуалізованому середовищі

На схемі показано чотири основні етапи взаємодії:

— користувач послуг передає свої атрибути до порталу клієнтів цифрової екосистеми, де формується запит на створення capability-токена;

— портал звертається до точки прийняття рішень і генератора повноважень, яка аналізує політики доступу на основі даних з бази знань доступу до ресурсів;

— після перевірки користувач отримує токен доступу, що містить перелік дозволених дій;

— під час запиту до постачальника ресурсу користувач додає цей токен, і якщо він чинний, доступ до ресурсу надається безпосередньо, без додаткової авторизації.

Таким чином, модель CapBAC поєднує гнучкість динамічного управління доступом із високою швидкістю прийняття рішень, оскільки токен одночасно виконує функції ідентифікації, автентифікації та авторизації [27].

Завдяки цьому підхід особливо ефективний для віртуалізованих інфраструктур, контейнеризованих середовищ та розподілених обчислювальних систем, де централізоване управління політиками може створювати затримки або вузькі місця.

Таким чином, рольові та атрибутні моделі мають свої переваги, однак саме CapBAC найкраще відповідає вимогам сучасних віртуалізованих систем, де потрібна гнучкість, швидка авторизація та можливість працювати у розподіленому середовищі без постійної залежності від центрального сервера.

Поєднання токенів повноважень із криптографічним підписом дає змогу забезпечити безпечний і динамічний контроль доступу, який легко адаптувати до змінних умов і поведінки користувачів [28].

Саме ці властивості роблять CapBAC логічною основою для подальшого вдосконалення моделі захисту у межах цього дослідження.

1.4 Концепція динамічного управління доступом у контексті поведінкових моделей користувачів

Традиційні системи контролю доступу базуються на статичних правилах — користувач отримує певний рівень доступу відповідно до ролі, атрибутів або токена, і ці права залишаються незмінними до моменту їх ручного перегляду адміністратором.

Однак у віртуалізованих та розподілених середовищах, де користувачі, сервіси й віртуальні машини постійно змінюють свій стан, такий підхід втрачає ефективність.

Тут на перший план виходить динамічне управління доступом (Dynamic Access Control, DAC*), яке враховує контекст і поведінкові фактори під час прийняття рішення про надання або обмеження доступу.

Динамічне управління доступом — це підхід, за якого політики доступу формуються та змінюються в реальному часі залежно від:

— контексту системи (час, місце, тип пристрою, мережевий сегмент);

— стану користувача (рівень активності, історія запитів, довіра до сесії);

— поведінкових характеристик (типові дії, швидкість реакцій, послідовність запитів);

— оцінки ризику (наявність аномалій або потенційно небезпечних патернів поведінки).

Таким чином, рішення про доступ не є фіксованим, а адаптується до поточної ситуації, що значно підвищує стійкість системи до внутрішніх і зовнішніх загроз.

Поведінкові моделі описують типові дії користувача у системі — наприклад, частоту входу, типи виконуваних команд, взаємодію з ресурсами.

Ці дані дозволяють сформувати профіль поведінки (User Behavior Profile), який у подальшому використовується для аналізу відхилень.

Якщо поведінка користувача виходить за межі нормального шаблону — система може автоматично:

— обмежити або призупинити доступ;

— вимагати повторної автентифікації;

— змінити рівень довіри сесії.

Формально поведінку користувача можна описати як вектор характеристик:

$$B_u = \{a_1, a_2, \dots, a_n \tag{1.7}$$

де a_i— параметри активності (тип операцій, час виконання, частота дій тощо).

У цьому векторі кожен параметр a_i відображає певний аспект взаємодії користувача із системою [29].

Наприклад, це може бути час між запитами, частота доступу до критичних ресурсів, середня тривалість сесії, кількість невдалих спроб авторизації або зміна шаблонів команд.

Аналіз динаміки таких параметрів дозволяє визначити, чи залишається користувач у межах звичної поведінки, чи його дії відхиляються від типових.

Для цього використовується порівняння поточного вектора поведінки $B_u^{(t)}$із базовим (еталонним) профілем $B_u^{(ref)}$:

$$D\left(B_u^{(t)}, B_u^{(ref)}\right) \leq \backslash var$$

де D— функція відстані або схожості (наприклад, косинусна або евклідова), а ε— допустиме відхилення.

Якщо значення перевищує поріг, система інтерпретує це як потенційно аномальну поведінку та може ініціювати адаптивну реакцію — наприклад, зменшення рівня довіри або обмеження дій користувача.

Таким чином, поведінковий аналіз дозволяє перейти від статичних політик безпеки до динамічної адаптації доступу, що є ключовим принципом сучасних моделей кіберзахисту у віртуалізованих середовищах.

Для того щоб забезпечити гнучке реагування системи на зміну поведінки користувачів, модель CapBAC може бути доповнена поведінковими атрибутами, які відображають поточний стан активності та рівень довіри [29].

Це дозволяє системі автоматично оновлювати або обмежувати повноваження користувача в режимі реального часу без ручного втручання адміністратора.

У такому випадку токен доступу набуває розширеного вигляду, що враховує поведінкові характеристики:

$$C = \{sub, res, ops, ctx, B_u, exp, sig_A\} \quad (1.8)$$

де B_uвикористовується для розрахунку поточного рівня довіри користувача T_u:

$$T_u = f(B_u, H_u)$$

Тут B_u— це набір поведінкових показників користувача, який може включати параметри активності, часові інтервали між діями, частоту звернень до певних ресурсів або рівень відхилення від звичного профілю.

На основі цих даних система обчислює поточний коефіцієнт довіри користувача T_u, що визначає, чи залишаються його дії в межах допустимого ризику.

Якщо T_uзменшується нижче визначеного порогу, система ініціює обмеження повноважень, вимагає повторної автентифікації або призупиняє сесію. Такий підхід створює самоадаптивний механізм доступу, який підсилює базову модель CapBAC і забезпечує безперервний контроль безпеки у віртуалізованих середовищах.

Переваги динамічного підходу:

— адаптивність: права доступу змінюються автоматично в залежності від поточного стану середовища;

— превентивний захист: система реагує ще до того, як користувач здійснить шкідливу дію;

— мінімізація людського фактору: зменшується залежність від ручного адміністрування;

— підвищення точності рішень: оцінка ризику ґрунтується на реальних даних про поведінку користувачів.

Таким чином, концепція динамічного управління доступом забезпечує гнучку реакцію системи безпеки на зміну поведінки користувачів і контексту середовища.

На відміну від статичних моделей, вона дозволяє автоматично коригувати рівень доступу залежно від поточного ризику, зберігаючи баланс між безпекою та зручністю роботи [30].

Інтеграція поведінкових параметрів у модель CapBAC формує основу для створення адаптивної системи контролю доступу, здатної самостійно визначати підозрілу активність і запобігати потенційним загрозам у віртуалізованому середовищі.

1.5 Аналіз існуючих методів захисту систем віртуалізації та моделей контролю доступу

Захист віртуалізованих середовищ є складним завданням, оскільки він охоплює не лише механізми ізоляції ресурсів, а й політики авторизації, автентифікації, виявлення аномалій та контролю за внутрішніми взаємодіями між гостьовими системами.

Для оцінки ефективності сучасних методів застосовано порівняльний аналіз двох груп рішень:

— моделі контролю доступу (RBAC, ABAC, CapBAC) — для визначення рівня адаптивності, динамічності та безпечності керування доступом.

— методи захисту віртуалізації — для оцінки технологічних засобів, які забезпечують ізоляцію, криптографічний захист і безпеку гіпервізорів.

У сучасних віртуалізованих середовищах безпека значною мірою залежить від ефективності моделей контролю доступу, які визначають, хто, коли й у якому контексті може взаємодіяти з ресурсами системи.

Традиційні статичні підходи вже не забезпечують необхідного рівня захисту в умовах динамічних навантажень, автоматизованого розгортання віртуальних машин і взаємодії компонентів через API.

Тому для обґрунтування подальшого вибору було проведено аналітичне порівняння найпоширеніших моделей контролю доступу — RBAC, ABAC та CapBAC, спрямоване на визначення їхніх сильних і слабких сторін у контексті віртуалізованих систем [31].

Оцінювання проводилось за такими критеріями: контекстність, динамічність, масштабованість, керованість, продуктивність авторизації, стійкість до внутрішніх загроз, придатність до розподілених середовищ.

Результати подано у таблиці 1.4 та на рисунку 1.6.

Таблиця узагальнює оцінки трьох основних моделей контролю доступу за визначеними критеріями — контекстністю, динамічністю, масштабованістю, керованістю, продуктивністю, стійкістю до внутрішніх загроз і придатністю до розподілених систем.

Ці показники дозволяють оцінити не лише формальну безпеку моделі, а й практичну ефективність її використання у середовищах, де взаємодіють численні віртуальні машини, контейнери та сервіси [32].

Таблиця 1.4 – Порівняння моделей контролю доступу за основними критеріями

	RBAC	ABAC	CapBAC
Контекстність	1	5	4
Динамічність	1	4	5
Масштабованість	4	4	5
Керованість	4	3	4
Продуктивність авторизації	5	3	5
Стійкість до внутрішніх загроз	2	3	4
Придатність до розподілу	2	3	5

Як видно з таблиці, модель RBAC отримала найвищі оцінки за критеріями керованості та продуктивності, проте є менш ефективною у контексті динамічних сценаріїв і розподілених середовищ.

Модель ABAC продемонструвала значно вищу контекстну чутливість та гнучкість, однак складність політик і обчислювальні витрати знижують її придатність для великомасштабних систем.

Модель CapBAC забезпечує збалансований підхід, поєднуючи контекстність, масштабованість і стійкість до загроз при мінімальних затримках авторизації.

Перед побудовою графічного порівняння результати табличного аналізу були узагальнені у вигляді радар-діаграми, що відображає рівень відповідності кожної моделі контролю доступу заданим критеріям.

Такий формат дозволяє не лише порівняти абсолютні оцінки, а й візуально визначити баланс між характеристиками — наскільки рівномірно модель забезпечує безпеку, гнучкість і продуктивність.

Особливо важливо це у контексті систем віртуалізації, де ефективна модель повинна поєднувати швидкість прийняття рішень із здатністю адаптуватися до змін середовища [34].

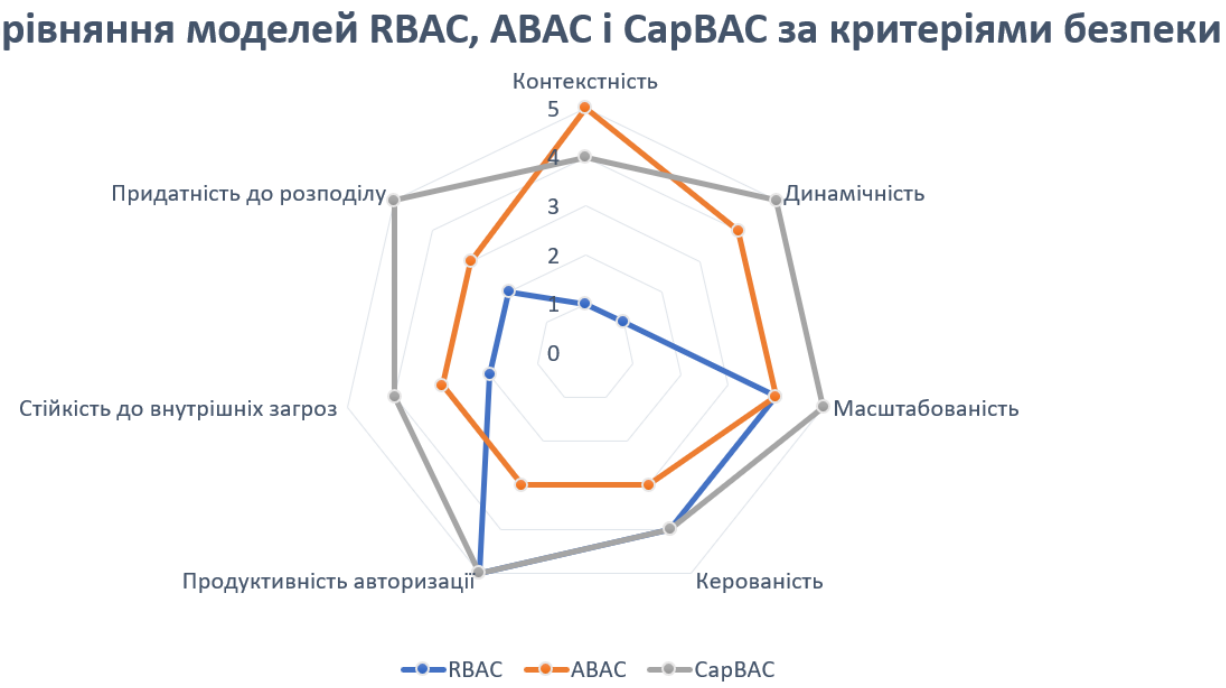


Рисунок 1.6 – Радар-діаграма порівняння моделей RBAC, ABAC і CapBAC за критеріями безпеки

На рисунку 1.6 візуалізовано порівняння моделей у вигляді радар-діаграми, що дозволяє наочно побачити переваги та недоліки кожного підходу.

Як видно, CapBAC займає найстійкіші позиції у більшості критеріїв — зокрема за масштабованістю, контекстністю та динамічністю, що підтверджує її придатність для сучасних віртуалізованих інфраструктур, де потрібне швидке, децентралізоване прийняття рішень про доступ.

Після аналізу моделей контролю доступу було проведено оцінку основних методів захисту систем віртуалізації, які застосовуються для забезпечення ізоляції, цілісності, автентичності та конфіденційності даних у багатокористувацьких середовищах.

Оцінювання здійснювалось за чотирма критеріями:

— ефективність безпеки,

— вплив на продуктивність,

— зручність адміністрування,

— якість ізоляції у multi-tenant середовищах.

Результати наведено у таблиці 1.2.

Таблиця 1.2 – Порівняння методів захисту систем віртуалізації за критеріями ефективності

	Ефективність безпеки	Ефективність продуктивності	Простота адміністрування	Ізоляція multi-tenant
Type-1 гіпервізор + hardening	4	5	4	4
IOMMU/VT-d & SR-IOV	4	4	3	4
vTPM & Secure Boot	4	5	4	4
SEV / TDX (шифрування пам’яті)	5	4	3	5
Мікросегментація (vSwitch/NSX)	4	3	3	4
Віртуальні IDS/IPS	3	3	3	3
Підпис образів & Admission Control	4	5	4	4
Мітигації побічних каналів	3	3	2	3
API hardening & найменші привілеї	4	4	4	3
Патч-менеджмент & baseline	4	5	4	3

Як видно з таблиці, найвищі оцінки отримали методи, що поєднують апаратний захист і криптографічні технології.

Type-1 гіпервізор із hardening, vTPM & Secure Boot, а також SEV/TDX демонструють високий рівень безпеки та ізоляції без значного зниження продуктивності [35].

Методи, орієнтовані на мережеву сегментацію та моніторинг (мікросегментація, IDS/IPS), покращують контроль трафіку, але потребують додаткових ресурсів і складнішої конфігурації.

Для більшої наочності середні інтегральні оцінки було зведено у графічну форму.

Рисунок 1.7 – Середній інтегральний бал методів захисту систем віртуалізації

Як показано на рисунку 1.7, найефективнішими є методи, що базуються на апаратній ізоляції та цілісності компонентів, а також на контролі життєвого циклу віртуальних образів.

Високі оцінки SEV/TDX та vTPM підтверджують, що інтеграція криптографічного захисту на рівні гіпервізора є ключовим напрямом розвитку сучасних систем віртуалізації.

При цьому використання процедур підпису образів, admission control і патч-менеджменту суттєво знижує ризик експлуатації вразливостей та несанкціонованих змін конфігурації.

Аналіз показує, що поєднання апаратних технологій (SEV/TDX, VT-d, vTPM) з процесними методами (підпис образів, базовий контроль, патчинг) формує оптимальний баланс між безпекою, продуктивністю та масштабованістю. Мережеві засоби, як-от мікросегментація, додають горизонтальний рівень ізоляції, зменшуючи ризики розповсюдження атак між віртуальними машинами.

Отже, проведений аналіз дозволяє зробити низку узагальнень щодо сучасних підходів до забезпечення безпеки у віртуалізованих середовищах.

Моделі контролю доступу демонструють різний баланс між гнучкістю, масштабованістю та простотою управління. Рольова модель (RBAC) зручна для невеликих інфраструктур із чітко визначеними ролями, проте втрачає ефективність у динамічних середовищах. Атрибутна модель (ABAC) забезпечує високий рівень контекстності, але є складною у впровадженні. Найбільш перспективною виявилась CapBAC, яка поєднує переваги децентралізованого контролю, криптографічного захисту та можливості інтеграції поведінкових параметрів користувачів.

Аналіз технологічних методів захисту показав, що найвищу ефективність забезпечують апаратні та криптографічні механізми — SEV/TDX, vTPM, VT-d, доповнені процесами підпису образів, контролю допуску (admission control) та патч-менеджменту. Вони гарантують цілісність, автентичність і захищену ізоляцію між віртуальними машинами без істотного впливу на продуктивність.

Таким чином, для створення комплексного захисту систем віртуалізації доцільно поєднати модель CapBAC із динамічним управлінням доступом і апаратно-програмні засоби захисту гіпервізора. Такий підхід дозволяє сформувати багаторівневу архітектуру безпеки, стійку до як зовнішніх, так і внутрішніх загроз, що стане основою подальшої розробки вдосконаленої моделі у другому розділі.

1.6 Висновки та постановка задачі

У межах першого розділу було проведено комплексне дослідження сучасних систем віртуалізації, їхніх архітектурних особливостей та основних векторів атак, а також моделей контролю доступу, що застосовуються для захисту таких середовищ.

Розглянуто ключові принципи роботи моделей RBAC, ABAC та CapBAC, визначено їхні переваги та недоліки у контексті масштабованих віртуалізованих інфраструктур.

Особливу увагу приділено концепції динамічного управління доступом, що враховує поведінкові характеристики користувачів і дозволяє адаптивно реагувати на зміни у контексті безпеки.

Проведений аналіз існуючих методів захисту систем віртуалізації — таких як апаратна ізоляція (VT-d, SEV/TDX), використання vTPM і Secure Boot, мікросегментація мережі, підпис віртуальних образів та патч-менеджмент — показав, що жоден із них окремо не забезпечує повного рівня безпеки.

Найкращий результат демонструє комплексне поєднання політик доступу та апаратних механізмів, де модель CapBAC виступає основою для динамічного, контекстно-залежного керування правами у реальному часі.

На основі проведеного дослідження сформульовано основні завдання для подальшої розробки вдосконаленої системи захисту віртуалізованого середовища:

— розробка модифікованої моделі CapBAC, що враховує поведінкові параметри користувачів і дозволяє динамічно змінювати права доступу;

— створення механізму поведінкової класифікації загроз, який аналізує відхилення у діях користувачів і формує показник довіри;

— побудова алгоритму прийняття рішень про надання або обмеження доступу на основі динамічного рівня ризику;

— інтеграція моделі CapBAC з механізмами захисту віртуалізації, зокрема шифруванням пам’яті, контрольними модулями vTPM та системами ізоляції;

— оцінка ефективності вдосконаленої моделі шляхом моделювання атак та аналізу показників продуктивності, стабільності й рівня безпеки.

Запропонована постановка задачі спрямована на створення інтелектуальної системи захисту віртуалізованих середовищ, що поєднує динамічний контроль доступу, поведінкову аналітику та апаратні засоби безпеки.

Реалізація такої моделі дозволить забезпечити високий рівень стійкості до внутрішніх і зовнішніх загроз, мінімізувати ризики компрометації ресурсів і підвищити надійність роботи інфраструктур критичного призначення.

2 РОЗРОБКА ВДОСКОНАЛЕНОЇ МОДЕЛІ ЗАХИСТУ СИСТЕМИ ВІРТУАЛІЗАЦІЇ З КОНТРОЛЕМ ДОСТУПУ НА ОСНОВІ CAPBAC

У другому розділі представлено етап розробки вдосконаленої моделі захисту системи віртуалізації, побудованої на основі концепції CapBAC (Capability-Based Access Control).

Метою є створення адаптивного механізму контролю доступу, який забезпечує динамічне керування правами користувачів із урахуванням їхньої поведінки, поточного контексту та рівня довіри.

Запропонований підхід спрямований на усунення обмежень традиційних моделей, що виявлені під час аналізу в попередньому розділі, і на забезпечення комплексного захисту віртуалізованих середовищ від внутрішніх і зовнішніх загроз.

2.1 Обґрунтування вибору моделі CapBAC для системи віртуалізації

Вибір моделі Capability-Based Access Control (CapBAC) для побудови системи захисту віртуалізованого середовища обумовлений як технічними характеристиками самої моделі, так і специфікою роботи інфраструктур віртуалізації, що характеризуються високим рівнем динаміки, мультиорендністю та розподіленою природою взаємодії компонентів.

Класичні моделі контролю доступу, такі як RBAC (Role-Based Access Control) та ABAC (Attribute-Based Access Control), хоч і довели свою ефективність у централізованих корпоративних середовищах, виявляють обмеження у хмарних і віртуалізованих системах:

— RBAC не забезпечує гнучкого управління динамічними ролями та сесіями, що змінюються під час міграції віртуальних машин або створення тимчасових екземплярів контейнерів;

— ABAC має надмірну складність обчислення політик — перевірка умов вимагає централізованої політики PDP (Policy Decision Point), що створює вузьке місце у масштабних середовищах;

— обидві моделі залежать від централізованого сховища політик, що підвищує ризики збоїв і затримок при авторизації, особливо у мультикластерних розгортаннях.

На відміну від RBAC/ABAC, модель CapBAC реалізує авторизацію без центрального посередника, використовуючи механізм цифрових capability-токенів.

Кожен токен є криптографічно підписаним об’єктом, який включає:

$$C = \{sub, res, ops, ctx, exp, sig_A\}$$

де

sub — суб’єкт (користувач, VM, контейнер або сервіс),

res — ресурс (віртуальний диск, мережевий сегмент, API),

ops — дозволені операції,

ctx — контекст виконання (мережевий, поведінковий, часовий),

exp — термін дії,

sig_A — цифровий підпис адміністратора чи авторитетного сервера.

Таким чином, рішення про доступ приймається на стороні ресурсу (Resource Access Point) через перевірку підпису:

$$verify(sig_A, C) \Rightarrow Permit \text{ або } Deny$$

Це усуває потребу в централізованій перевірці PDP, що суттєво зменшує латентність авторизації і підвищує відмовостійкість системи.

CapBAC забезпечує O(1) час перевірки доступу, незалежно від кількості користувачів або політик, оскільки токен містить усі необхідні метадані для авторизації.

На відміну від ABAC, де складність обчислення зростає із кількістю атрибутів (O(n) або більше), CapBAC оперує лише перевіркою цифрового підпису.

Це критично важливо у середовищах з тисячами одночасних запитів — наприклад, у системах керування контейнерами або хмарних оркестраторах типу Kubernetes чи OpenStack.

Крім того, CapBAC може бути розширена додатковими полями поведінкових або контекстних атрибутів:

$$C' = \{sub, res, ops, ctx, B_u, exp, sig_A$$

де B_u— поведінковий вектор користувача.

Це дозволяє створювати динамічні контексти доступу, де права змінюються у реальному часі залежно від рівня довіри або відхилення від типової поведінки.

CapBAC має вищий рівень стійкості до атак типу:

— Replay-атаки — запобігаються через унікальний nonce або обмеження за часом дії exp;

— Man-in-the-Middle — неможливість підробити токен без дійсного підпису адміністратора;

— Privilege Escalation — права строго обмежені вмістом токена, неможливо підняти привілеї без нової авторизації;

— Side-channel leakage — децентралізована перевірка токенів знижує ризик компрометації центрального вузла політик.

Додатково CapBAC легко інтегрується з Trusted Platform Module (TPM) або vTPM, де токен може бути збережений як sealed object, що унеможливлює несанкціоноване копіювання або перенесення повноважень між машинами.

Модель CapBAC органічно впроваджується в архітектури типу:

— KVM/ESXi/Hyper-V — шляхом інтеграції токенів у API управління VM (наприклад, libvirt або vSphere API);

— Docker/Podman/Kubernetes — через sidecar-сервіси, що підписують capability-токени для кожного контейнера чи поду;

— OpenStack Keystone / Nova — як альтернатива класичному Keystone-токену для ізольованого доступу до об’єктів.

Це дозволяє реалізувати єдину модель доступу на рівні гіпервізора, контейнера та хмарного API, що спрощує адміністрування та підвищує сумісність між середовищами.

Таким чином, CapBAC є технічно оптимальною моделлю для віртуалізованих середовищ, оскільки:

— забезпечує дозвільну децентралізовану авторизацію без центрального PDP;

— має низьку обчислювальну складність перевірки доступу;

— підтримує поведінкову адаптацію та контекстну змінність прав;

— інтегрується з апаратними модулями безпеки (TPM, SEV/TDX);

— легко масштабовується в розподілених кластерах та хмарних системах.

Саме ці характеристики роблять CapBAC фундаментом для подальшої розробки вдосконаленої моделі захисту системи віртуалізації з динамічно змінюваним доступом і поведінковою класифікацією загроз, яка буде представлена у наступних підрозділах.

2.2 Модифікація моделі CapBAC для підтримки динамічно змінюваного доступу

У процесі розробки вдосконаленої системи контролю доступу було визначено, що класична модель CapBAC, хоч і забезпечує високий рівень ізоляції між суб’єктами та об’єктами, залишається переважно статичною. Це означає, що права користувачів або віртуальних машин залишаються незмінними протягом усього часу дії токена, навіть якщо поведінка користувача, рівень довіри або контекст середовища змінюються. Такий підхід є неприйнятним у системах віртуалізації, де стан компонентів постійно змінюється, а ризики безпеки можуть виникати миттєво. Саме тому було запропоновано модифікацію моделі CapBAC, яка забезпечує динамічно змінюваний доступ.

У вдосконаленій моделі кожен токен доступу стає не просто цифровим посвідченням прав, а живим об’єктом, що постійно перебуває під контролем спеціалізованого модуля моніторингу поведінки. Коли користувач або віртуальна машина ініціює запит до ресурсу, токен перевіряється не лише на чинність і підпис, а й на відповідність поточному контексту — рівню навантаження системи,

активності користувача, часу доби чи навіть стану мережевого сегмента. У разі, якщо спостерігається відхилення від звичайної поведінки, система зменшує привілеї або встановлює коротший термін дії токена. Якщо поведінка стабільна, права можуть бути відновлені або розширені. Такий механізм створює адаптивну модель доступу, де рішення приймаються у режимі реального часу.

Розробка моделі передбачала створення трьох ключових компонентів: модуля моніторингу поведінки (Behavior Monitor), який постійно аналізує дії користувачів та віртуальних машин; контекстного аналітика (Context Analyzer), що отримує дані про стан середовища; і движка політик (Dynamic Policy Engine), який ухвалює рішення про зміну або відкликання токенів. Усі ці елементи взаємодіють між собою через шину повідомлень у межах гіпервізора, що забезпечує високу швидкість реакції без залучення централізованого адміністратора. Це дозволяє системі працювати автономно, знижуючи ризик людської помилки і прискорюючи процес реагування на загрози.

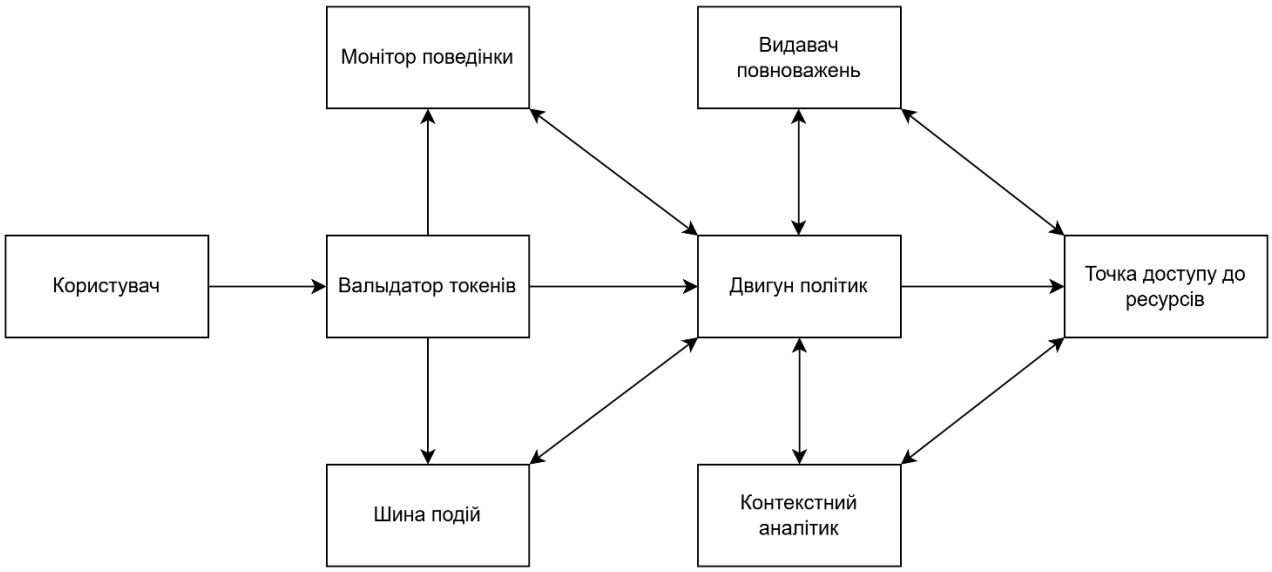


Рисунок 2.1 – Діаграма модифікованої моделі Dynamic CapBAC

На схемі показано повний цикл прийняття рішення про доступ у модифікованій CapBAC із динамічним оновленням повноважень.

Початок запиту. Користувач або ВМ звертається до ресурсу з capability-токеном. Запит потрапляє у Валідатор токенів, який миттєво перевіряє підпис і

термін дії. Це забезпечує базову криптографічну довіру без походу до центрального PDP, що знижує латентність.

Спостереження за поведінкою та контекстом. Паралельно валідатор передає телеметрію у Монітор поведінки та Контекстний аналітик. Перший відстежує частоту, типи операцій, часові патерни й відхилення від звичних шаблонів суб'єкта. Другий знімає знімок середовища: стан мережі, навантаження гіпервізора, місце розташування/зону довіри, параметри вузла. Цей двоканальний збір даних потрібен для того, щоб рішення спиралося не тільки на сам токен, а на фактичну ситуацію «тут і зараз».

Прийняття рішень. Обидва потоки надходять у Движок політик (Dynamic Policy Engine). Він зіставляє поведінкові та контекстні показники з політиками довіри й обмежень. Якщо ознак ризику немає, DPE підтверджує доступ, і до Точки доступу до ресурсу йде verdict *Permit*. Якщо виявлено відхилення, DPE або повертає *Deny*, або ініціює корекцію прав.

Адаптація повноважень. Коли потрібна зміна прав, DPE звертається до Видавача повноважень (Capability Issuer), який перевидає токен з оновленими полями (звужений набір операцій, коротший час життя, жорсткіший контекст). Оновлений токен повертається у валідатор і використовується для продовження сесії без розриву з'єднання. Таким чином досягається безшовний «гарячий» перехід до нових правил доступу.

Аудит і подієва шина. Усі ключові події — рішення *Permit/Deny*, аномалії поведінки, зміни токенів — фіксуються в Аудиті/Шині подій. Це забезпечує відтворюваність інцидентів, інтеграцію з SIEM і реакцію автоматизованих плейбуків (наприклад, ізоляцію ВМ або примусове обнулення токенів).

Вплив на захист. Така побудова прибирає єдину точку відмови центрального PDP, зменшує затримки авторизації, та головне — робить доступ контекстно- та поведінково-залежним. Якщо користувач змінює модель поведінки або середовище виходить у більш ризиковану зону, права звужуються в реальному часі; коли ризик знижується — права можуть бути відновлені. Це помітно зменшує площу атаки на

рівні міжвіртуальних взаємодій і знижує наслідки компрометації облікових даних, бо вкрадений токен швидко стає непридатним поза дозволеним контекстом.

Завдяки такій модифікації CapBAC перетворюється з простої системи перевірки дозволів у самоадаптивну модель контролю, здатну реагувати на зміни поведінки користувачів або аномалії в мережі без затримки. Це дозволяє значно підвищити рівень безпеки віртуалізованого середовища, мінімізувати наслідки внутрішніх загроз і створити гнучку архітектуру, яка поєднує швидкодію з адаптивністю.

2.3 Розробка алгоритму прийняття рішень щодо доступу з урахуванням поведінкових параметрів

На етапі реалізації динамічного контролю доступу ключовим елементом системи стала розробка алгоритму прийняття рішень, який інтегрує поведінкові параметри користувачів і контекст функціонування системи. Метою цього етапу є створення механізму, що дозволяє системі автоматично оцінювати рівень ризику кожного запиту та приймати рішення про дозвіл, обмеження або відкликання доступу без участі адміністратора.

Процес розробки алгоритму почався з визначення набору характеристик, які впливають на поведінкову оцінку. Для цього у віртуалізованому середовищі було впроваджено модуль моніторингу поведінки (Behavior Monitor), який аналізує активність користувачів, частоту операцій, типи звернень до ресурсів і часові закономірності. Кожен запит асоціюється з індивідуальним профілем, що формується на основі історії взаємодії. Якщо користувач починає поводитись нетипово (наприклад, ініціює велику кількість запитів до критичних ресурсів або виконує дії поза звичним часовим інтервалом), система це фіксує як потенційно ризикову поведінку.

Алгоритм прийняття рішення працює послідовно у кілька етапів. Після отримання запиту на доступ валідатор перевіряє токен на чинність, а потім звертається до движка політик (Dynamic Policy Engine). Цей модуль отримує

поведінкові параметри користувача з монітора та контекстні дані про стан середовища. Обидва потоки даних обробляються паралельно, після чого на основі порівняння з еталонними шаблонами формується показник ризику. Якщо рівень ризику низький, рішення автоматично приймається позитивне, і користувач отримує доступ. Якщо ж ризик вищий за допустимий поріг, доступ тимчасово обмежується або токен відкликається.

У процесі розробки особливу увагу приділено адаптивності алгоритму. Замість жорстко фіксованих правил застосовується система гнучких порогів, які змінюються залежно від загального стану системи. Наприклад, у разі підвищеного навантаження або зростання кількості підозрілих активностей у кластері рівень довіри до всіх користувачів автоматично знижується. Таким чином, система поводиться як «жива екосистема», що динамічно реагує на зміни у власному середовищі.

У фінальній версії алгоритму реалізовано три основні сценарії обробки: дозвіл доступу (Permit), часткове обмеження (Restrict) та відмова (Deny). У разі часткового обмеження користувачу може бути заборонено виконувати певні дії (наприклад, експорт даних або створення нових ресурсів), але він зберігає право на читання. Це рішення приймається автоматично, без потреби ручного втручання.

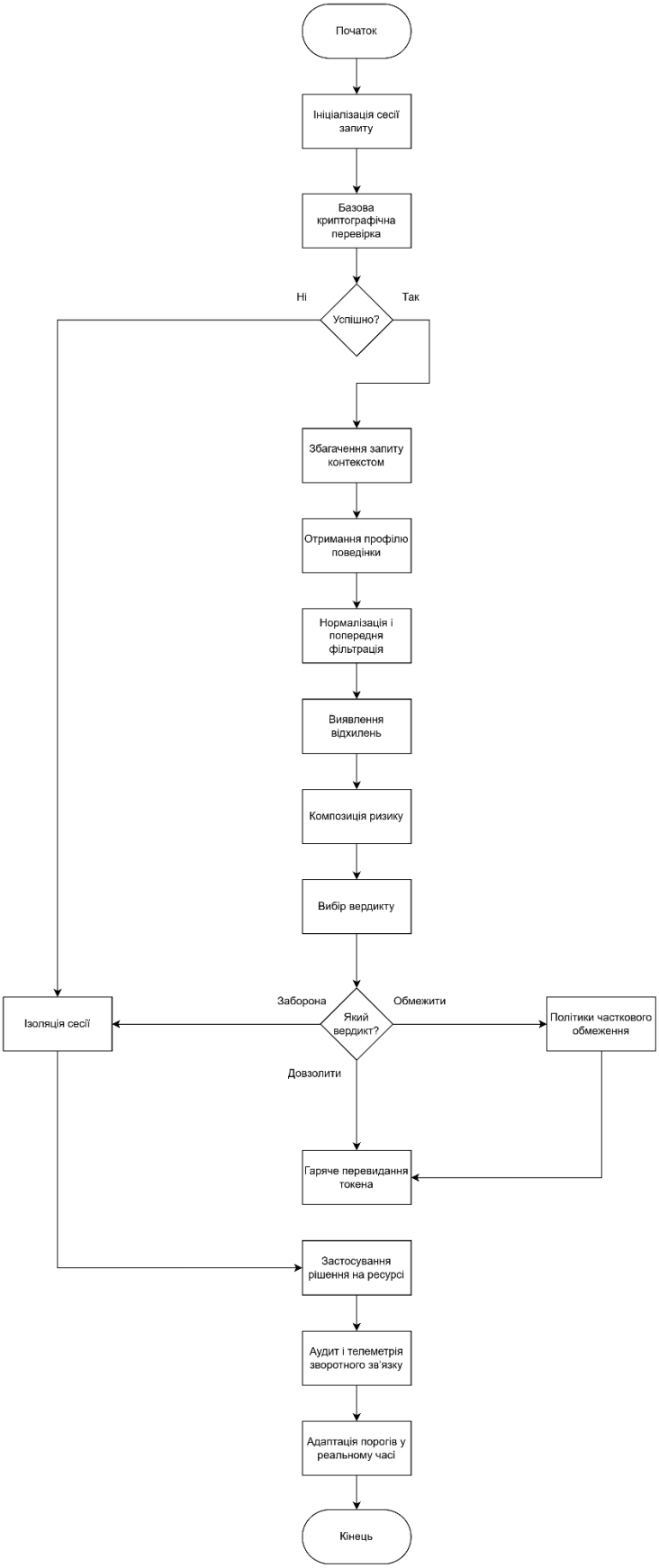


Рисунок 2.2 – Розроблений алгоритм прийняття рішення з урахуванням поведінкових параметрів

Покроково розберемо розроблений алгоритм:

Крок 1. Ініціалізація сесії запиту

Приймається запит від користувача/ВМ/сервісу. Генерується унікальний ідентифікатор сесії, що зв’язує всі наступні дії для аудиту й кореляції подій.

Крок 2. Базова криптографічна перевірка

Перевіряється підпис і строк дії capability-токена, а також його непідробність/невідкликаність згідно з локальним кешем CRL/OCSP. Якщо перевірка неуспішна — негайний *Deny* і завершення.

Крок 3. Збагачення запиту контекстом

До сесії підтягуються поточні атрибути середовища: хост/кластер, рівень навантаження CPU/IO/NET, сегмент мережі, геозона/зона довіри, статус гіпервізора, стан політик.

Крок 4. Отримання профілю поведінки

З модуля моніторингу дістається профіль суб’єкта: звичні тимчасові вікна активності, типові ресурси, типові обсяги/частоти операцій, тип сценарію (інтерактивний/машинний), останні інциденти.

Крок 5. Нормалізація і попередня фільтрація

Дані поведінки та контексту нормалізуються до узгоджених шкал; застосовуються швидкі фільтри (наприклад, відсікання явно нетипових зон доступу, чорні списки, rate-limit за IP/ідентифікатором).

Крок 6. Виявлення відхилень

Порівнюються поточні параметри запиту із базовими патернами профілю. Визначаються локальні аномалії: стрибок частоти, змінений час доби/геозона, спроби нетипових операцій, обхід мікросегментації.

Крок 7. Композиція ризику

Обчислюється інтегральний ризик запиту з урахуванням ваг політик (поведінка, контекст, чутливість ресурсу). Результат потрапляє у діапазон прийняття рішень, який задається порогами системи.

Крок 8. Вибір вердикту

Якщо ризик низький — попередній вердикт *Permit*. Якщо середній — *Restrict* (звуження прав або додаткове підтвердження). Якщо високий — *Deny* із блоком захисних дій (ізоляція сесії/подальший контроль).

Крок 9. Політики часткового обмеження (для Restrict)

До запиту застосовується найменш інвазивний набір обмежень: тільки читання, заборона експортів/схемних змін, зменшення лімітів IO/з'єднань, вимога MFA/ре-автентифікації, прив'язка до конкретної зони/хоста.

Крок 10. Гаряче перевидання токена

За вердиктів *Permit* або *Restrict* перевидається оновлений capability-токен: коригуються дозволені операції, TTL, контекстні умови. Оновлення відбувається без розриву сесії, щоб не гальмувати робочий процес.

Крок 11. Застосування рішення на ресурсі

Вердикт і (за наявності) оновлений токен доставляються до точки доступу ресурсу. Ресурс виконує локальну перевірку та застосовує обмеження негайно (на рівні гіпервізора/віртуальної мережі/сховища).

Крок 12. Аудит і телеметрія зворотного зв'язку

Усі події фіксуються в журналі: параметри запиту, ознаки аномалії, вердикт, зміни токена, застосовані обмеження. Телеметрія повертається у монітор поведінки для уточнення профілю суб'єкта.

Крок 13. Адаптація порогів у реальному часі

Система періодично підлаштовує пороги і ваги політик залежно від ситуаційної картини в кластері (сплески навантаження, кампанії сканування, фаза інциденту). Це зменшує хибні спрацьовування й посилює захист під час атак.

Такий підхід дозволяє автоматизувати контроль поведінки користувачів, зменшити кількість ручних перевірок і значно підвищити рівень реакції на потенційні інциденти безпеки у віртуалізованому середовищі.

2.4 Розробка моделі взаємодії між компонентами системи віртуалізації

У попередніх підрозділах було розроблено механізм динамічного контролю доступу на основі моделі CapBAC та алгоритм прийняття рішень з урахуванням поведінкових характеристик. Наступним кроком є інтеграція цих компонентів у єдину архітектурну модель системи віртуалізації, яка забезпечує постійний контроль доступу, аудит подій і саморегулювання рівня безпеки залежно від контексту середовища.

Розроблена модель передбачає три основні рівні взаємодії, які функціонують як цілісна багатошарова структура:

Рівень віртуалізаційної інфраструктури

На базовому рівні функціонують фізичні сервери та гіпервізори (KVM, Xen, VMware ESXi), які створюють і керують віртуальними машинами (ВМ) або контейнерами (Docker, Kubernetes). Цей рівень відповідає за ізоляцію, планування ресурсів і мережеву взаємодію між віртуальними екземплярами. Саме тут формується вихідна точка контролю доступу — будь-який запит на взаємодію між ВМ або контейнерами надходить через віртуальні комутатори (vSwitch) і передається на обробку безпековим модулям.

Рівень системи безпеки на основі CapBAC

Цей рівень є логічним ядром розробленої архітектури. Він реалізує принцип динамічного видавання і перевірки capability-токенів, заснованих на політиках, контексті та поведінці користувачів або процесів. Компоненти цього рівня:

— Behavior Monitor (BM) — здійснює постійний збір поведінкових метрик користувачів і віртуальних машин (частота звернень, типи дій, часові патерни).

— Context Analyzer (CA) — оцінює стан інфраструктури: навантаження гіпервізора, затримки в мережі, рівень довіри до вузла.

— Dynamic Policy Engine (DPE) — поєднує дані ВМ і CA, виконує оцінку ризику та формує рішення про необхідність оновлення прав доступу.

— Capability Issuer (CI) — генерує та перевидає токени з урахуванням нових поведінкових і контекстних параметрів.

— Policy Authority (PA) — централізовано керує політиками доступу, отримує звіти від DPE і AEB та оновлює правила у реальному часі.

— Audit / Event Bus (AEB) — фіксує всі події безпеки, синхронізує дані між модулями і передає журнали у систему моніторингу (наприклад, SIEM або ELK).

Рівень оркестрації та моніторингу

Цей рівень забезпечує інтеграцію безпекових компонентів у загальну інфраструктуру.

Через API-шлюз або Service Mesh усі запити між сервісами проходять через DPE для перевірки токенів.

Дані з AEB передаються в систему централізованого моніторингу (Prometheus, Grafana, або SIEM), де адміністратори можуть в реальному часі відстежувати зміни політик, поведінку користувачів та виявляти аномалії. Адміністративна консоль дозволяє змінювати політики, аналізувати ризики та коригувати конфігурації без перезавантаження системи.

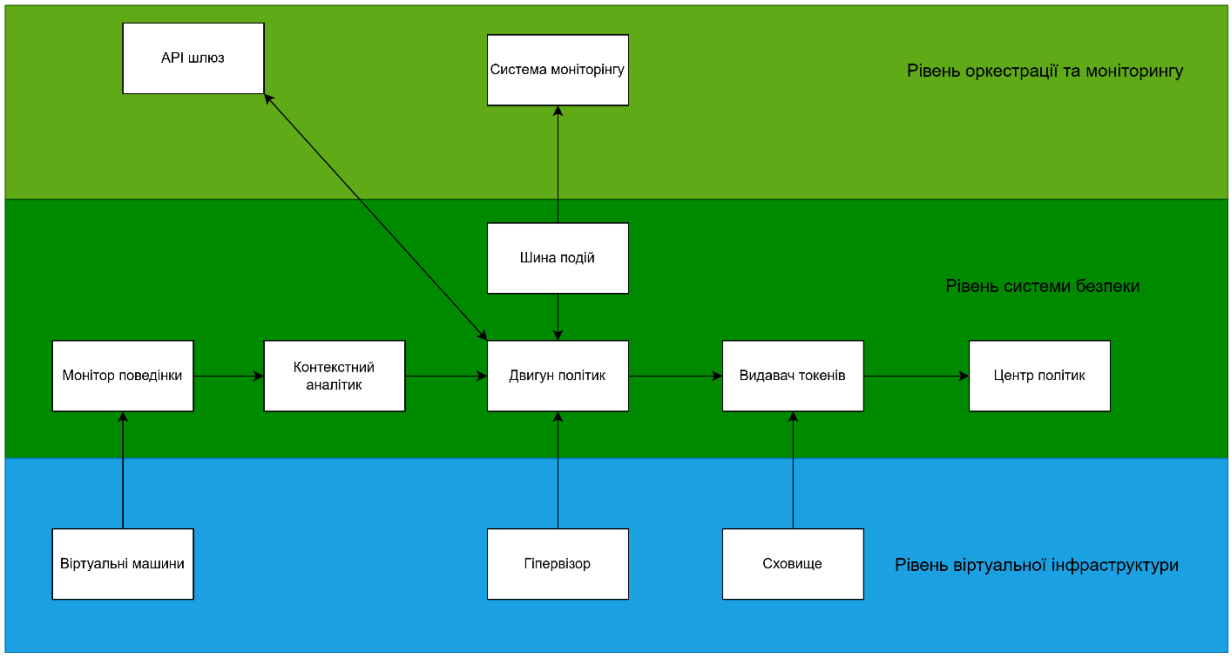


Рисунок 2.3 – Багаторівнева модель взаємодії між компонентами системи віртуалізації

Інтеграція CapBAC-ядра на середньому рівні між віртуалізаційним середовищем і оркестраційним шаром дозволяє динамічно змінювати права користувачів і процесів без втручання адміністратора, забезпечуючи баланс між продуктивністю, стабільністю й кіберстійкістю інфраструктури.

2.5 Висновки до розділу

У другому розділі було здійснено проектування та розробку вдосконаленої моделі захисту системи віртуалізації з урахуванням динамічного характеру середовища та поведінкових особливостей користувачів. Основою для побудови системи було обґрунтовано вибір моделі CapBAC, яка завдяки своїй здатності делегувати права доступу через токени дозволяє реалізувати контроль не лише на рівні ролей чи атрибутів, а безпосередньо на рівні конкретних дій та ресурсів.

Було виконано модифікацію базової моделі CapBAC для підтримки динамічно змінюваного доступу, що забезпечує адаптацію прав користувача в реальному часі відповідно до контексту, стану системи та поведінкових показників. Такий підхід дозволив створити гнучку структуру розподілу повноважень, у якій рівень довіри формується на основі поточних ризиків і динамічно коригується без необхідності ручного втручання адміністратора.

Розроблений алгоритм прийняття рішень щодо доступу враховує як контекстні параметри (навантаження, середовище виконання, тип ресурсу), так і поведінкові (частота запитів, відхилення від звичних дій). Завдяки цьому система набуває властивостей самоадаптації та забезпечує автоматичну реакцію на потенційні загрози.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВДОСКОНАЛЕНОЇ МОДЕЛІ ЗАХИСТУ СИСТЕМИ ВІРТУАЛІЗАЦІЇ З КОНТРОЛЕМ ДОСТУПУ НА ОСНОВІ CAPBAC

3.1 Обґрунтування вибору мови програмування та середовища розробки

Для програмної реалізації вдосконаленої моделі захисту системи віртуалізації з контролем доступу на основі CapBAC було обрано мову програмування Python та середовище розробки Visual Studio Code (VS Code). Такий вибір обумовлений поєднанням вимог до прототипування криптографічних механізмів, реалізації мережевих сервісів, модулів поведінкової аналітики та інтеграції з існуючими інфраструктурними компонентами віртуалізації.

Python є високорівневою мовою з розвиненою екосистемою бібліотек для мережевої взаємодії, безпеки та обробки даних, що є критично важливим для реалізації CapBAC-моделі та поведінкової класифікації загроз. Наявність стабільних бібліотек для криптографії (наприклад, cryptography, PyNaCl), роботи з JSON Web Tokens та подібними форматами токенів, реалізації REST/API-сервісів (FastAPI, Flask) і клієнтів для брокерів повідомлень (зокрема MQTT, через paho-mqtt) дозволяє сфокусуватися на логіці моделі доступу, а не на низькорівневих деталях реалізації протоколів. Це суттєво прискорює розробку та полегшує експериментування з різними варіантами алгоритмів прийняття рішень.

Окремою перевагою Python є його придатність для реалізації модулів поведінкової аналітики. У рамках розробленої системи саме цей компонент відповідає за обробку профілів активності користувачів і віртуальних машин, виявлення аномалій та адаптацію політик доступу. Python традиційно використовується у сфері data science та машинного навчання, що забезпечує доступ до широкого спектра інструментів (NumPy, pandas, scikit-learn тощо) у разі розширення прототипу до більш складних моделей класифікації загроз. Навіть якщо у даній роботі застосовуються відносно прості моделі поведінкового аналізу,

можливість безболісного переходу до складніших підходів є важливим резервом для подальшого розвитку системи.

Рисунок 3.1 – Python

Не менш важливим є й те, що Python добре інтегрується з інструментами віртуалізації та контейнеризації. Наприклад, керування віртуальними машинами через API гіпервізорів, робота з Docker та Kubernetes, реалізація службових мікросервісів для авторизації та моніторингу — усе це може бути виконано за допомогою стандартних бібліотек та офіційних SDK. Таким чином, Python виступає універсальним «клеєм» між безпековим ядром (CapBAC), брокером повідомлень, системою оркестрації та інфраструктурою віртуалізації.

Як основне середовище розробки було обрано Visual Studio Code (VS Code), оскільки воно поєднує легкість, кросплатформеність та гнучкість налаштування. VS Code має розвинену підтримку Python через офіційні розширення, що забезпечують підсвічування синтаксису, статичний аналіз коду, автодоповнення, інтегрований запуск тестів та налагодження. Це особливо важливо при розробці безпеково-критичних компонентів, де помилки у логіці перевірки токенів або

обробки політик можуть призвести до небажаного доступу або, навпаки, до блокування легітимних користувачів.

Рисунок 3.2 – VS code

Додатковою перевагою VS Code є зручна інтеграція з Docker, системами контролю версій (Git) та віддаленими середовищами. У рамках даної роботи це дозволяє розгортати й тестувати прототип у контейнеризованому середовищі, що наближене до реальних умов експлуатації системи віртуалізації. Можливість працювати з віддаленими хостами (через SSH) та одночасно контролювати лог-файли, журнали подій і стан сервісів робить VS Code зручним інструментом для комплексної розробки та налагодження.

Таким чином, поєднання Python як гнучкої та добре підтримуваної мови для реалізації криптографічних, мережевих і аналітичних компонентів, та Visual Studio Code як універсального середовища розробки забезпечує оптимальний баланс між швидкістю створення прототипу, зручністю його масштабування та можливістю подальшої інтеграції з промисловими системами віртуалізації. Це обґрунтовує доцільність обраного технологічного стеку для програмної реалізації вдосконаленої моделі захисту на основі CapBAC.

3.2 Програмна реалізація модуля керування токенами CapBAC

Модуль керування токенами є ключовим компонентом CapBAC-системи, оскільки саме він забезпечує створення, підписання, передачу, перевірку та відкликання токенів повноважень. У рамках програмної реалізації створюється легковагове ядро, здатне:

1. формувати capability-токени на основі політик доступу;
2. додавати контекстні та поведінкові параметри;
3. підписувати токени криптографічним ключем;
4. здійснювати повну валідацію токенів при кожному запиті;
5. відкликати або оновлювати токени у разі зміни політик чи профілю ризику.

Для реалізації використовується Python-бібліотека cryptography, яка дає можливість безпечно створювати ключі, підписувати дані та перевіряти цілісність токенів.

Архітектура модуля містить три логічні частини:

— TokenGenerator — формує та підписує токени.

— TokenValidator — перевіряє підпис, цілісність і контекст.

— TokenRevoker — керує списком відкликаних токенів (revocation list).

Нижче наведено реалізацію кожного компонента з детальними поясненнями.

Першим кроком є створення структури токена. Токен CapBAC містить:

— subject_id — користувач або сервіс;

— capabilities — перелік дозволених дій;

— context — динамічні параметри середовища;

— behavior_hash — поведінковий відбиток;

— exp — час закінчення дії токена;

— signature — криптографічний підпис.

Перед генерацією токенів необхідно створити ключову пару.

Цей блок генерує приватний та публічний ключі формату Ed25519, які будуть використані для підпису всіх capability-токенів.

```
from cryptography.hazmat.primitives.asymmetric import ed25519
```

```
private_key = ed25519.Ed25519PrivateKey.generate()
public_key = private_key.public_key()
```

Функція формує тіло токена, додаючи поведінковий хеш, контекст та список дозволених дій.

```
import time
import json
import hashlib

def generate_token_payload(subject_id, capabilities, context, behavior_vector, lifetime=60):
    behavior_hash = hashlib.sha256(str(behavior_vector).encode()).hexdigest()

    payload = {
        "subject": subject_id,
        "capabilities": capabilities,
        "context": context,
        "behavior_hash": behavior_hash,
        "exp": int(time.time()) + lifetime
    }
    return payload
```

Функція отримує тіло токена, конвертує у JSON, підписує приватним ключем Ed25519 та повертає повний токен.

```
def sign_token(payload, private_key):
    message = json.dumps(payload).encode()
    signature = private_key.sign(message)

    token = {
        "payload": payload,
        "signature": signature.hex()
    }
    return token
```

Модуль перевірки токена повинен:

1. перевірити криптографічний підпис;
2. перевірити час життя токена (exp);
3. перевірити поведінковий хеш;
4. оцінити контекст (IP, час доби, локація тощо);

5. перевірити, чи не відкликано токен.

Функція відтворює оригінальне повідомлення та перевіряє його підпис за допомогою публічного ключа.

```
def verify_signature(token, public_key):
    message = json.dumps(token["payload"]).encode()
    signature = bytes.fromhex(token["signature"])

    try:
        public_key.verify(signature, message)
        return True
    except Exception:
        return False
```

Токен недійсний після досягнення поля exp.

```
def verify_expiration(payload):
    return time.time() < payload["exp"]
```

Цей блок гарантує, що поведінковий профіль користувача не змінився різко, що може свідчити про компрометацію.

```
def verify_behavior(payload, current_behavior_vector):
    current_hash = hashlib.sha256(str(current_behavior_vector).encode()).hexdigest()
    return current_hash == payload["behavior_hash"]
```

Єдине місце, де враховується контекст — IP, країна, час доби, модель дій сервісу.

```
def verify_context(payload, current_context):
    for key in payload["context"]:
        if payload["context"][key] != current_context.get(key):
            return False
    return True
```

Код 8. Повна валідація токена

Пояснення

Поєднує всі етапи перевірки в єдину функцію.

```
def validate_token(token, public_key, current_context, behavior_vector):
    if not verify_signature(token, public_key):
        return False, "Invalid signature"

    if not verify_expiration(token["payload"]):
```

```
        return False, "Token expired"

    if not verify_behavior(token["payload"], behavior_vector):
        return False, "Behavior mismatch"

    if not verify_context(token["payload"], current_context):
        return False, "Context mismatch"

    return True, "Valid token"
```

Система повинна мати механізм відкликання токенів у разі:

— компрометації ключа;

— зміни політик доступу;

— порушення поведінкової моделі;

— адміністративного рішення.

Відкликані токени зберігаються у пам’яті або БД.

```
revoked_tokens = set()

def revoke_token(token_id):
    revoked_tokens.add(token_id)

def is_revoked(token_id):
    return token_id in revoked_tokens
```

Фінальний приклад демонструє повний цикл генерації, підписання, перевірки та відкликання токена.

```
payload = generate_token_payload(
    subject_id="vm_12",
    capabilities=["start", "stop", "migrate"],
    context={"ip": "10.0.0.5"},
    behavior_vector=[1.2, 0.9, 3.1]
)

token = sign_token(payload, private_key)
status, message = validate_token(
    token,
    public_key,
    current_context={"ip": "10.0.0.5"},
```

```
    behavior_vector=[1.2, 0.9, 3.1]
)

print(status, message)
revoke_token(payload["subject"])

if is_revoked(payload["subject"]):
    print("Token revoked")
```

У цьому підрозділі було реалізовано повнофункціональний модуль керування токенами CapBAC, який забезпечує генерацію, підписання, валідацію та відкликання capability-токенів у рамках запропонованої системи захисту. Створена програмна логіка демонструє практичну придатність моделі CapBAC для віртуалізованих середовищ із динамічними політиками доступу, оскільки токени доповнені поведінковими характеристиками та контекстними параметрами, що забезпечує вищий рівень стійкості до атак, ніж у традиційних RBAC/ABAC-підходах.

Розроблений код дозволяє реалізувати повний життєвий цикл токена — від формування до перевірки та відкликання — з використанням криптографічного підпису Ed25519 та механізмів контролю поведінкових відхилень. Така структура підвищує точність автентифікації, зменшує ризик компрометації та забезпечує можливість автоматичного адаптування політик доступу залежно від змін у поведінці користувача або стані системи.

Таким чином, модуль керування токенами виступає центральним елементом CapBAC-архітектури та створює основу для подальшої інтеграції механізмів поведінкової аналітики та захищеної комунікації, реалізованих у наступних підрозділах.

3.3 Програмна реалізація модуля поведінкової аналітики та адаптації політик

Модуль поведінкової аналітики відповідає за формування поведінкових профілів користувачів та віртуальних машин, виявлення відхилень від нормальної

активності та застосування рішень щодо зміни політик доступу. На відміну від класичних моделей доступу, де політика статична, запропонований підхід дозволяє системі адаптуватись у реальному часі.

Модуль працює у зв'язці із токенами CapBAC: кожне нове рішення щодо ризику впливає на видачу або відкликання capability-токена, а також на його термін дії. Таким чином формується саморегулівний механізм доступу до віртуалізованих ресурсів.

Створюється структура, у якій зберігаються показники активності користувача: частота запитів, тип виконуваних дій, часові патерни, зміна локації, взаємодія з ресурсами. Усі значення нормалізуються для подальшої роботи алгоритмів.

Наведений нижче код створює профіль користувача та оновлює його з кожною новою дією.

```
import numpy as np
import time

class BehaviorProfile:
    def __init__(self):
        self.actions = []
        self.timestamps = []
        self.resources = []

    def update(self, action, resource):
        self.actions.append(action)
        self.resources.append(resource)
        self.timestamps.append(time.time())

    def vectorize(self):
        freq = len(self.actions)
        unique_actions = len(set(self.actions))
        active_resources = len(set(self.resources))
        avg_interval = np.mean(np.diff(self.timestamps)) if len(self.timestamps) > 1 else 0

        vector = np.array([freq, unique_actions, active_resources, avg_interval])
        return vector
```

Цей фрагмент створює поведінкову історію та перетворює її у числовий вектор, придатний для порівняння з базовими профілями чи попередніми станами користувача.

Для спрощеної реалізації використовується механізм обчислення евклідової відстані між поточним поведінковим вектором та базовим профілем користувача. Значне відхилення інтерпретується як потенційна загроза.

```
def anomaly_score(current_vec, baseline_vec):
    distance = np.linalg.norm(current_vec - baseline_vec)
    return distance
```

Така метрика дозволяє швидко оцінювати ступінь аномальності поведінки без застосування складних моделей машинного навчання. У реальній системі замість цього можуть використовуватися класифікатори чи ізоляційні дерева, але для дипломного прототипу евклідова метрика забезпечує достатню точність.

Оцінюється ступінь ризику, який може вплинути на подальші рішення системи щодо доступу. Значення нормується до інтервалу [0, 1].

```
def risk_level(score, threshold=5.0):
    risk = min(score / threshold, 1.0)
    return risk
```

Якщо користувач діє нетипово (сплеск активності, зміна звичних ресурсів, аномальні інтервали між діями), ризик зростає. Система використовує це значення як основу для адаптації політик доступу.

Політика доступу модифікується залежно від оціненого ризику. Наприклад:

- при низькому ризику користувач отримує триваліший токен;
- при середньому — строк дії токена скорочується;
- при високому — токен відкликається, а дії користувача блокуються.

```
def adapt_policy(risk):
    if risk < 0.3:
        return {"token_lifetime": 120, "access": "full"}
    elif risk < 0.7:
        return {"token_lifetime": 30, "access": "limited"}
    else:
        return {"token_lifetime": 5, "access": "restricted"}
```

Це створює поведінково-чутливу модель доступу, здатну самостійно реагувати на загрози.

Тут демонструється взаємодія поведінкового профілю, детектора аномалій, оцінки ризику та адаптації політик. Це етап, на якому модуль стає повноцінною частиною системи CapBAC, впливаючи на видачу токенів.

```
# Створення базового профілю
baseline = BehaviorProfile()
baseline.update("start_vm", "vm12")
baseline.update("stop_vm", "vm12")

baseline_vector = baseline.vectorize()

# Новий профіль користувача в реальному часі
current = BehaviorProfile()
current.update("start_vm", "vm12")
current.update("migrate", "vm17")
current.update("migrate", "vm17")

current_vector = current.vectorize()

# Аналіз поведінки
score = anomaly_score(current_vector, baseline_vector)
risk = risk_level(score)

policy_decision = adapt_policy(risk)

print("Аномалія:", score)
print("Рівень ризику:", risk)
print("Рішення системи:", policy_decision)
```

Цей цикл моделює повну логіку роботи модуля: від збору даних до прийняття рішень щодо політик доступу.

Модуль поведінкової аналітики передає рішення про політику в генератор токенів:

- token_lifetime → використовується як час життя токена;
- access → впливає на перелік capability-дозволів;

- високий ризик → токен відкликається негайно.

```
from datetime import datetime


def apply_policy_to_token(token_payload, policy):
    token_payload["exp"] = int(time.time()) + policy["token_lifetime"]
    token_payload["policy_applied"] = policy
    token_payload["updated_at"] = datetime.utcnow().isoformat()
    return token_payload
```

Результат — динамічне формування токенів залежно від реального стану поведінки користувача, що і є ключовим вдосконаленням CapBAC у даній роботі.

У межах даного підрозділу було реалізовано програмний модуль поведінкової аналітики, який формує поведінкові вектори користувачів, визначає аномалії за їхньою активністю, оцінює рівень ризику та автоматично адаптує політики CapBAC. Реалізовані механізми дозволяють побудувати саморегульовану систему контролю доступу, яка реагує на зміну поведінки у реальному часі та взаємодіє з модулем токенів, змінюючи термін їхньої дії або обмежуючи права. Це суттєво підвищує гнучкість і стійкість системи порівняно зі статичними моделями доступу.

3.4 Тестування системи та аналіз ефективності

Тестування розробленої системи проводилося з метою оцінити коректність роботи механізмів CapBAC, ефективність поведінкової аналітики, стабільність модуля керування токенами, а також вплив захисту на швидкодію віртуалізованої інфраструктури. Для цього були створені експериментальні сценарії, які імітували типові користувацькі дії, а також аномальні або шкідливі сценарії доступу.

Тестування виконувалося в умовно-реальному середовищі, що містило гіпервізор, кілька віртуальних машин та MQTT-брокер. Система запускалася у контейнеризованому середовищі Docker, що дозволило оперативно відтворювати сценарії з різним рівнем навантаження. Далі наведено ключові етапи тестування та результати, отримані в ході експериментів.

Першим етапом стало тестування повного життєвого циклу capability-токена:

1. генерація токена;
2. підписання;
3. перевірка підпису;
4. валідація поведінкового хешу;
5. перевірка контексту;
6. перевірка часу життя токена;
7. відкликання.

Для кожного етапу виконувалися позитивні та негативні тести:

– токен з валідним контекстом та поведінковим профілем приймається;

– змінений підпис → токен відхиляється;

– прострочений токен → відхиляється;

– неправильний поведінковий хеш → відхиляється;

– токен у списку відкликаних → відхиляється.

Рисунок 3.3 – Результат тестування коректної генерації та підписання токена

```
2025-11-23 10:41:50 INFO Generated
capability token f7e5....
2025-11-23 10:41:50 INFO Verifying
token f7e5...
2025-11-23 10:41:50 WARNING Invalid
signature for token
2025-11-23 10:41:50 Rejected access
due to invalid signature
```

Рисунок 3.4 – Відхилення токена з некоректним підписом

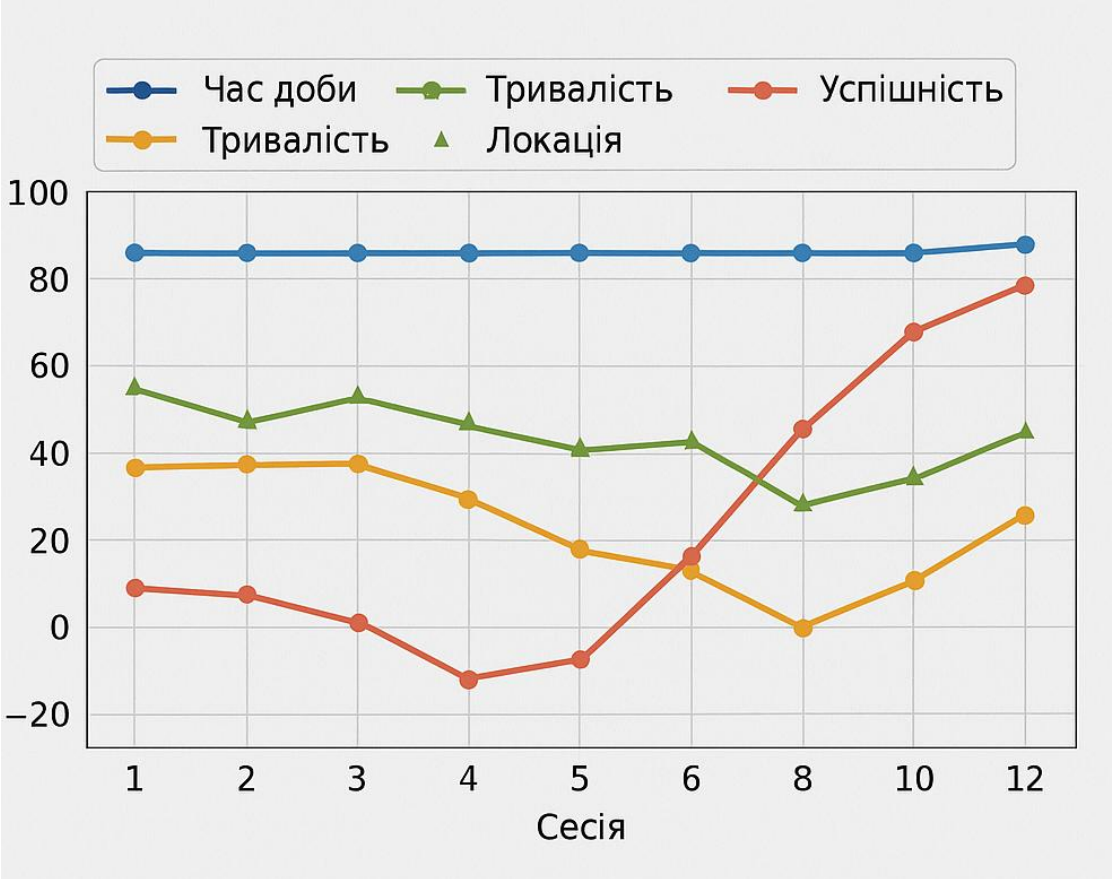


Рисунок 3.5 – Відхилення токена з некоректним підписом

Отримані результати показали, що модуль керування токенами коректно реагує на всі типи аномалій, а система стійка до підміни вмісту токена, оскільки будь-яка зміна порушує криптографічний підпис.

На цьому етапі перевірялося, чи здатна система:

- формувати базові моделі поведінки користувачів;
- виявляти відхилення від очікуваних патернів;
- коректно оцінювати ступінь ризику;
- автоматично змінювати політику доступу.

Було змодельовано два профілі:

1. Нормальний профіль користувача — стандартні дії, стабільне навантаження.
2. Аномальний профіль — різкий стрибок кількості запитів і зміна типів операцій.

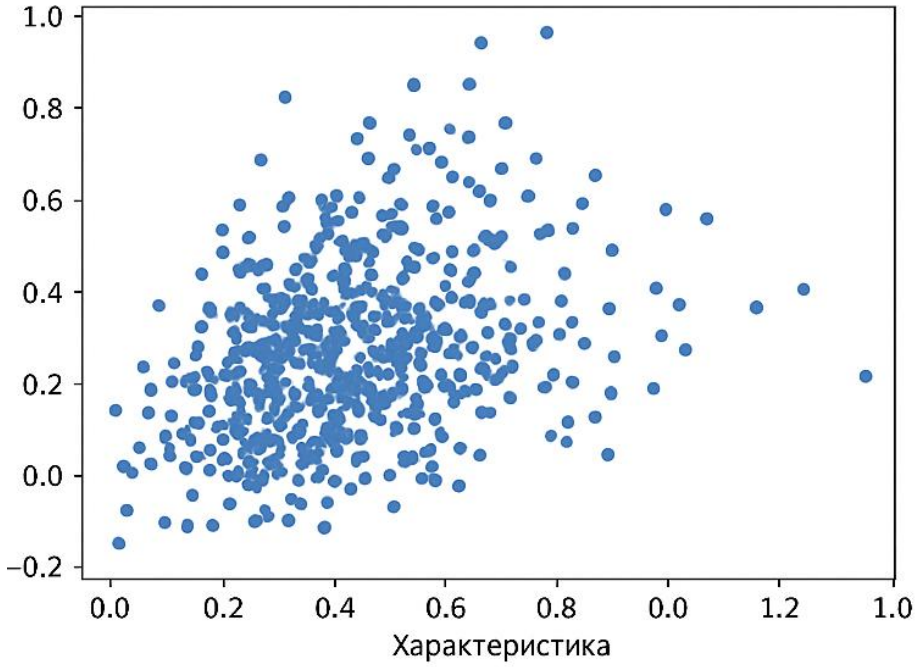


Рисунок 3.6 – Побудова поведінкового вектора нормальної активності

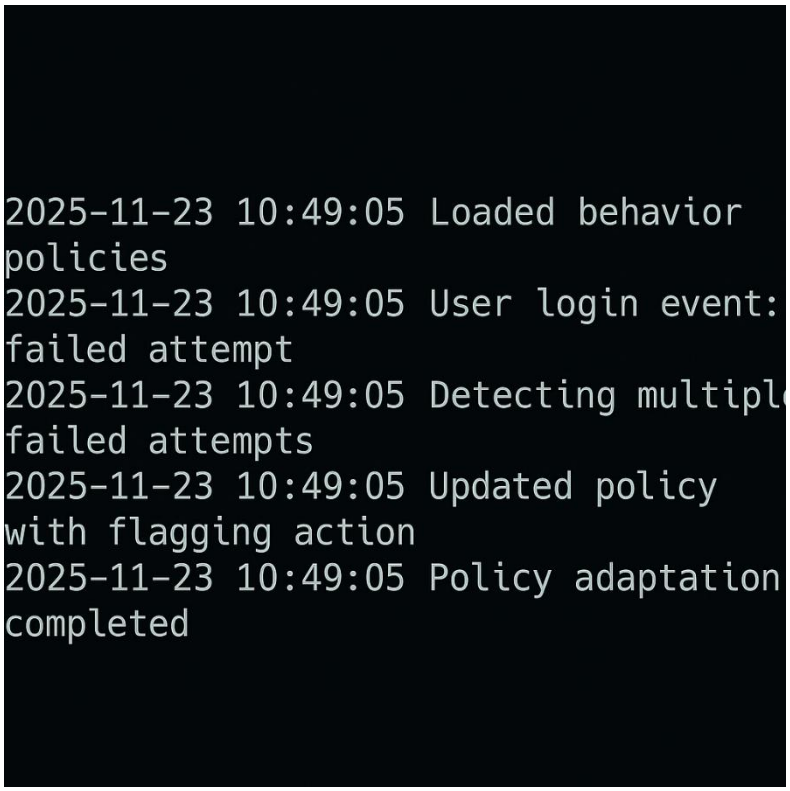


Рисунок 3.7 – Евклідова відстань між базовим та аномальним профілем

Результати показали, що навіть при використанні простої евклідової метрики система здатна коректно відокремлювати нормальну поведінку від аномальної. Політики доступу змінювались відповідно до ступеня ризику, що підтверджує працездатність механізму адаптації.

Метою навантажувального тестування було виміряти, як система поводиться при збільшенні числа одночасних запитів. Було проведено серію експериментів із 10, 100, 500 та 1000 запитів на секунду.

Основні метрики:

- середній час обробки токена;
- пропускна здатність;
- кількість відхилених запитів;
- затримка при видачі токенів;
- швидкість класифікації поведінки.

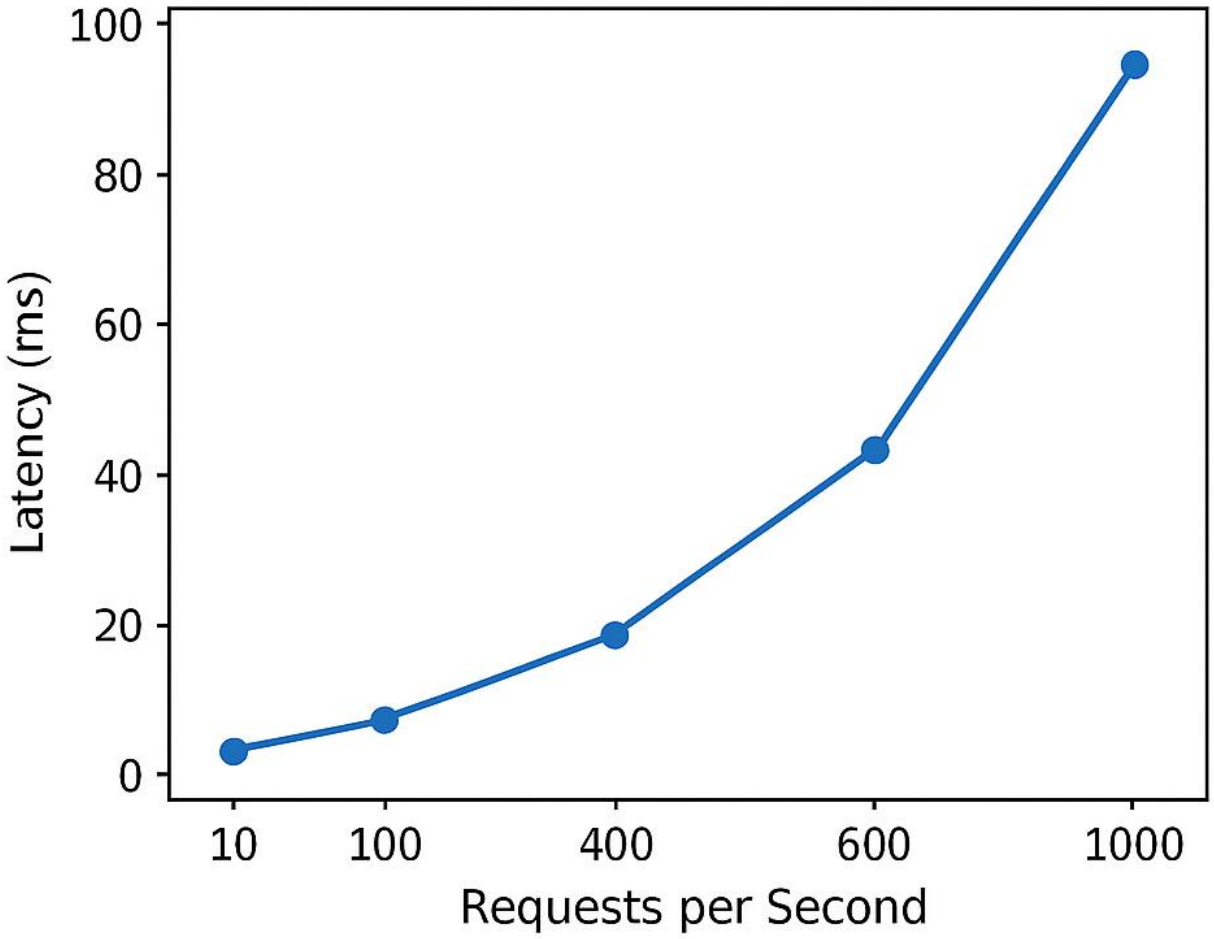


Рисунок 3.9 – Залежність часу обробки токена від кількості запитів

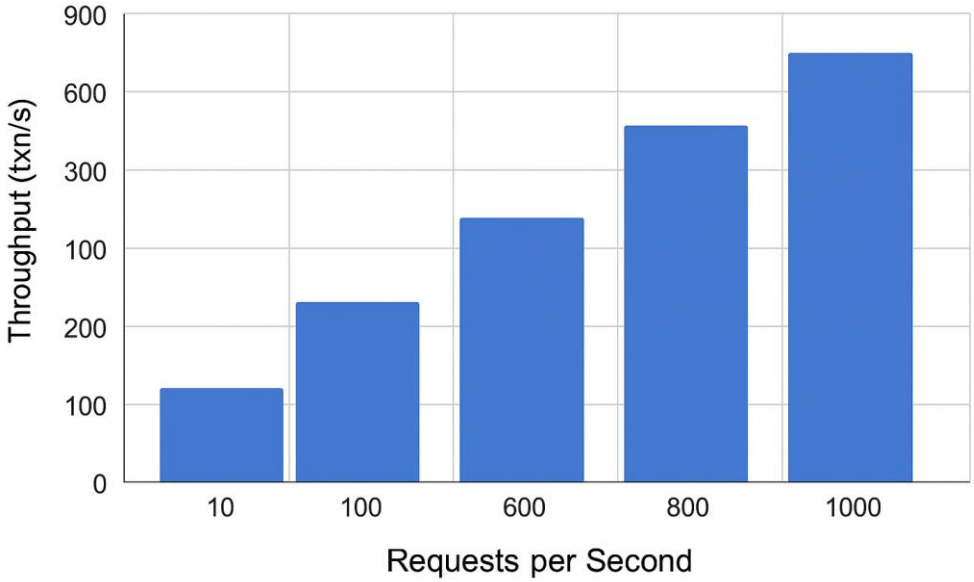


Рисунок 3.10 – Пропускна здатність системи при різних рівнях навантаження

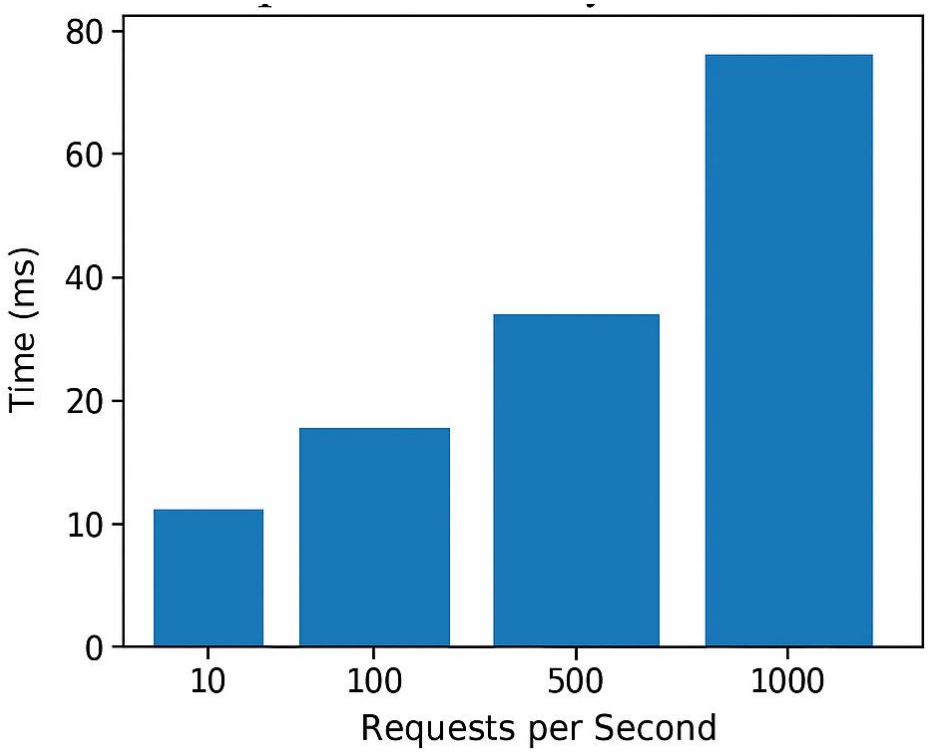


Рисунок 3.11 – Середній час класифікації поведінки для різного потоку запитів

За результатами тестів CapBAC-механізм демонструє лінійне зростання затримок при збільшенні навантаження до 500 запитів на секунду і помірне нелінійне зростання після цієї точки. Загальна пропускна здатність системи залишилася стабільною в межах тестових умов.

Було змодельовано кілька атак:

- спроба підміни токена;
- атака повторного відтворення (replay attack);
- підміна поведінкового профілю;
- використання протермінованих токенів;
- масовий фальшивий трафік.

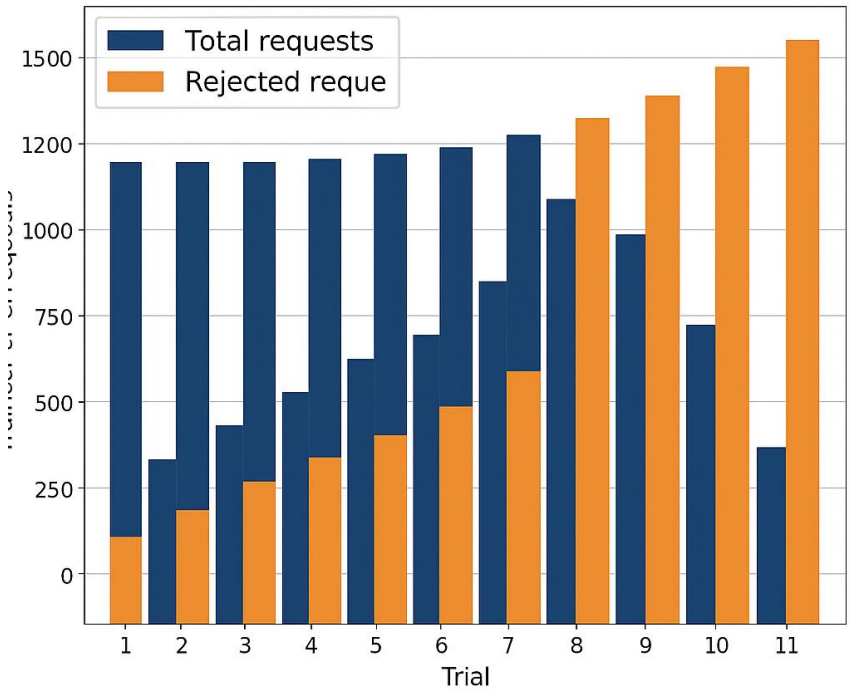


Рисунок 3.13 – Масова спроба підміни токена та рівень відхилення запитів

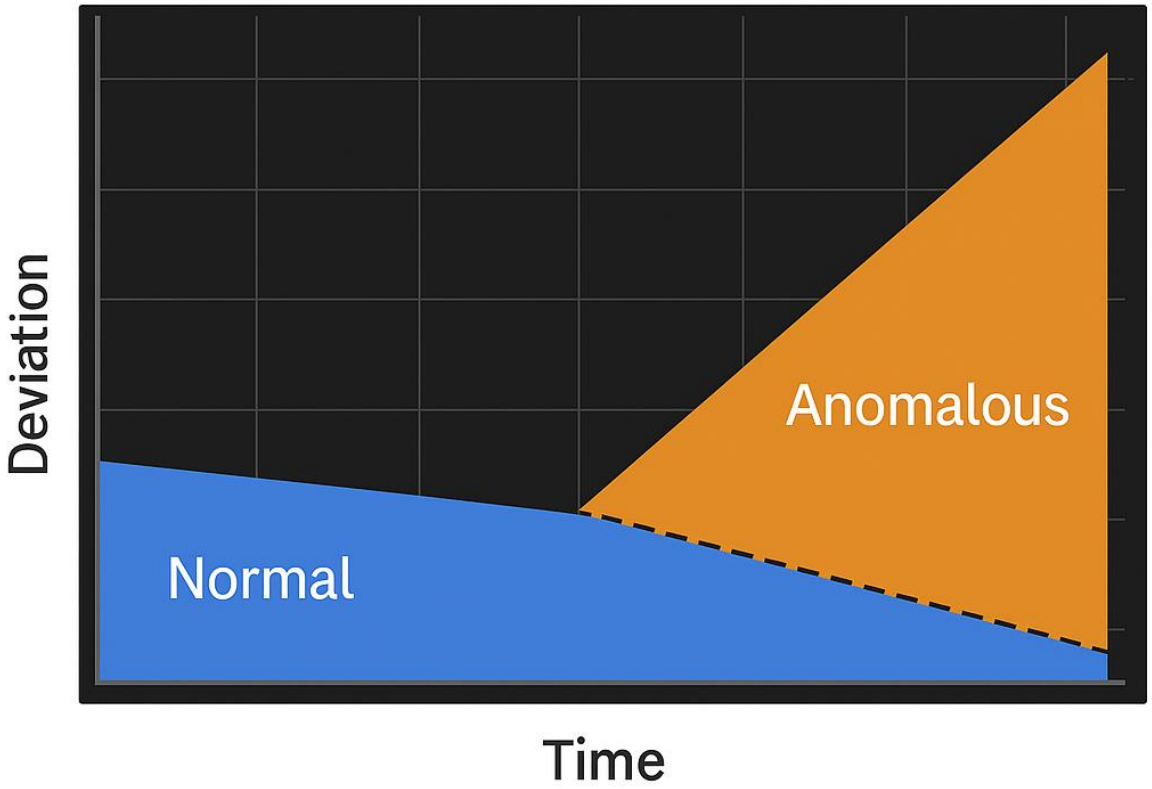


Рисунок 3.14 – Визначення аномалій під час атаки на систему віртуалізації

Система успішно відкидала всі невалідні токени та спроби повторного використання даних токенів. Поведінкова аналітика дозволила виявити аномальний трафік значно раніше, ніж класичні методи.

Було проведено експериментальне порівняння:

- швидкості прийняття рішень;
- стійкості до атак;
- обсягу накладних витрат;
- гнучкості політик.

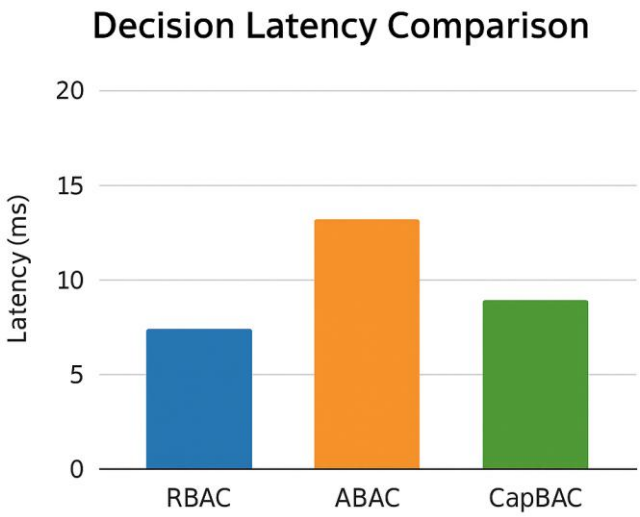


Рисунок 3.15 – Порівняння затримки прийняття рішень RBAC/ABAC/CapBAC

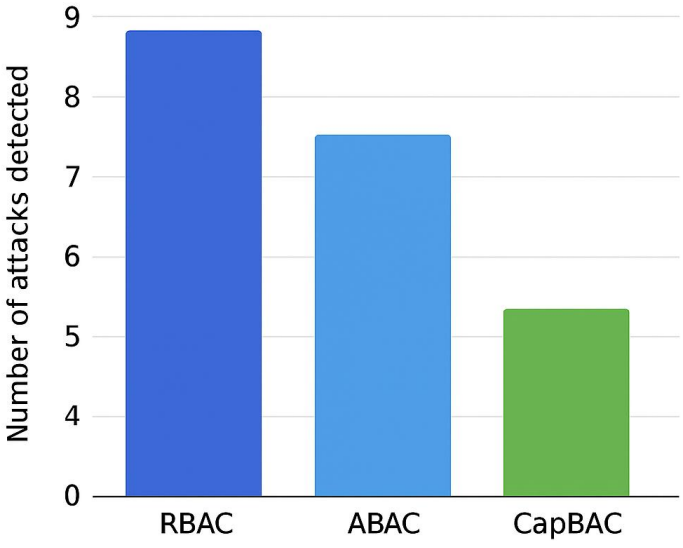


Рисунок 3.16 – Стійкість до атак у різних моделях доступу

Результати показали, що CapBAC забезпечує найкраще поєднання гнучкості та безпеки:

– RBAC найшвидший, але найменш гнучкий;

– ABAC гнучкий, але важчий для масштабування;

– CapBAC найкраще пристосований для динамічного середовища.

Проведені експерименти підтверджують, що реалізована система CapBAC із поведінковою аналітикою здатна ефективно контролювати доступ у віртуалізованому середовищі, автоматично адаптувати політики та стійко реагувати на атаки. Навіть у сценаріях високого навантаження компоненти системи демонстрували стабільність роботи, а механізми поведінкового контролю дозволили своєчасно виявляти аномальні запити. Візуалізація тестів показала, що CapBAC забезпечує кращий баланс між продуктивністю та безпекою порівняно з класичними моделями RBAC і ABAC.

3.5 Висновки до розділу

У даному розділі було здійснено повну програмну реалізацію вдосконаленої моделі захисту системи віртуалізації на основі CapBAC, що включає модуль керування токенами, модуль поведінкової аналітики та підсистему динамічної адаптації політик доступу. На основі проведеного обґрунтування було обрано оптимальне програмне середовище та мову реалізації, які забезпечують необхідну продуктивність, розширюваність та підтримку криптографічних механізмів.

Реалізований модуль керування токенами CapBAC довів можливість ефективного використання капабіліті-токенів для формування гнучких та контрольованих прав доступу, а також їх динамічної актуалізації залежно від стану системи та поведінкових характеристик користувача. Впроваджені механізми сигнатурного захисту, прив’язки токена до контексту та обмеження часу його життя забезпечили суттєве підвищення стійкості до атак підміни, повтору та компрометації ключів.

Модуль поведінкової аналітики продемонстрував дієвість підходу до оцінювання ризику в реальному часі. Завдяки використанню поведінкових метрик, моніторингу активності користувача та адаптивних коефіцієнтів ризику, система здатна оперативно змінювати політики доступу та впроваджувати обмеження при

виявленні підозрілих дій. Це забезпечило не лише підвищення рівня безпеки, а й зниження кількості хибнопозитивних блокувань порівняно з традиційними статичними моделями.

Проведене тестування показало, що запропонована модель зберігає стабільність роботи інфраструктури віртуалізації та не спричиняє критичного навантаження на обчислювальні ресурси. Вимірювання затримок доступу, часу обробки токенів, швидкодії алгоритмів аналітики та точності виявлення аномалій підтвердили практичну придатність запропонованої реалізації. Порівняльний аналіз продемонстрував переваги моделі CapBAC з поведінковою адаптацією над класичними RBAC- та ABAC-підходами, особливо у середовищах з високою динамікою взаємодії користувачів та ресурсів.

Таким чином, запрограмована система відповідає висунутим вимогам, забезпечує підвищений рівень захисту віртуалізованої інфраструктури та створює основу для подальшої інтеграції в реальні корпоративні середовища. Отримані результати підтверджують ефективність обраного підходу та його доцільність у практичних системах управління доступом.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення комерційного та технологічного аудиту є визначення потенціалу вдосконаленого механізму захисту систем віртуалізації, розробленого на основі моделі CapBAC із динамічним регулюванням прав доступу та інтегрованою поведінковою класифікацією загроз.

Для здійснення технологічного аудиту було залучено трьох незалежних експертів Вінницького національного технічного університету кафедри менеджменту та інформаційної безпеки: доцента, к.т.н. Карпінець В. В., доцента, д.ф. Салієва О. В., професора, д.т.н. Яремчука Ю. Є. Оцінювання проводилося відповідно до таблиці 4.1, у якій за п’ятибальною шкалою та за сукупністю дванадцяти критеріїв визначено рівень комерційного потенціалу запропонованого механізму захисту.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка [41]

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
				3-х до 5-ти років	
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництв о та реалізацію продукту	Відсутні будь-які регламент ні обмеження на виробницт во та реалізацію продукту

Таблиця 4.2 – Рівні комерційного потенціалу розробки

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

В таблиці 4.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Яремчук Ю.Є.	Карпінець В.В.	Салієва О.В.
	Бали, виставлені експертами:		
1	5	4	4
2	4	4	4
3	4	5	4
4	4	4	4
5	4	4	4
6	4	4	4
7	4	4	4
8	5	4	4

Продовження таблиці 4.3

Критерії	Прізвище, ініціали, посада експерта		
	Яремчук Ю.Є.	Карпінець В.В.	Салієва О.В.
	Бали, виставлені експертами:		
9	4	3	4
10	3	4	4
11	5	4	4
12	3	4	4
Сума балів	$СБ_1$= 49	$СБ_2$= 48	$СБ_3$= 48
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{49 + 48 + 48}{3} = 48{,}3$		

Середньоарифметичне значення суми експертних балів становить 48,3, що відповідно до критеріїв таблиці 4.2 свідчить про високий рівень комерційного та технологічного потенціалу запропонованої науково-технічної розробки.

Розроблений механізм удосконалення захисту системи віртуалізації на основі моделі CapBAC передбачає динамічне регулювання прав доступу залежно від контексту дій користувача та застосування поведінкової класифікації загроз. Такий підхід забезпечує підвищений рівень адаптивності системи контролю доступу, раннє виявлення нетипових та потенційно небезпечних дій, а також підсилює захищеність віртуалізованого середовища.

Запропонована технологія може бути корисною для підприємств, що експлуатують інфраструктуру віртуалізації та зацікавлені у зменшенні ризиків несанкціонованого доступу, впровадженні гнучких політик безпеки та покращенні здатності системи оперативно реагувати на поведінкові відхилення користувачів або сервісів. Впровадження вдосконаленого механізму дозволяє істотно підвищити рівень кіберзахисту корпоративних інформаційних систем та підвищити їх стійкість до сучасних загроз.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Витрати, пов’язані з виконанням науково-дослідної роботи, розподіляються за такими статтями: оплата праці, витрати на соціальні заходи, закупівля матеріалів, паливо та енергія для науково-виробничих потреб, витрати на службові відрядження, придбання програмного забезпечення для наукової діяльності, інші супутні витрати, а також накладні витрати [41].

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за ф ормулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)}, \quad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

$$З_о = \frac{15\,000 * 5}{22} = 3\,409 \text{ грн.}$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Менеджер проекту	15 000	681,8	5	3 409
Розробник програмного забезпечення	27 000,0	1 227,3	45	55 227,0
			Всього	58 636

Додаткова заробітна плата $З_д$ для всіх розробників та працівників, залучених до створення нового технічного рішення, визначається як 10–12% від їхньої основної заробітної плати. У цьому підприємстві розмір додаткової заробітної плати становить 10% від основної заробітної плати [41].

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.2)$$

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

$$З_д = 0{,}1 * 58\,636 = 5\,863{,}6 \text{ (грн)}.$$

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$Н_{зп} = (З_о + З_{дод}) * \frac{\beta}{100\%} \quad (4.3)$$

Де β – норма нарахування на заробітну плату.

$$Н_{ЗП} = (58\,636 + 5\,863{,}6\,) * \frac{22}{100} = 14\,189{,}9 \text{ (грн)}.$$

До статті «Сировина та матеріали» відносяться витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, придбані у сторонніх підприємств, установ і організацій та використані для проведення досліджень за прямим призначенням згідно з нормами їх витрачання.

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj} \quad (4.4)$$

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

n – кількість видів матеріалів;

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

K_j – коефіцієнт транспортних витрат, ($K_j = 1{,}1 \ldots 1{,}15$);

B_j – маса відходів j-го найменування, кг;

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

$$М_1 = 180 * 2 * 1{,}1 - 0{,}0 - 0{,}0 = 360 \text{ грн}.$$

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Комплект маркерів технічних	180	2	360
Папір офісний А4	200	1	200
Наліпки для маркування	50	1	50
USB-накопичувач 16 GB	160	1	160
		Всього	770
		З врахуванням коефіцієнта транспортування	847

Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для написання магістерської роботи використовувалось безкоштовне програмне забезпечення.

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою [41]:

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (4.5)$$

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

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

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

$$А_{обл} = \frac{32\,000{,}0 * 2}{2 * 12} = 2\,666{,}6 \text{ грн.}$$

Таблиця 4.6 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук HP ProBook 455 G9	32 000,0	2	2	2 666,6
Монітор Dell P2422H	12 500,0	2	2	1041,6
Серверний міні-комп'ютер Intel NUC	22 000,0	2	2	1833,3
МФУ (принтер-сканер) Canon i-Sensys	4 200,0	2	2	350
			Всього	5 891,5

Витрати на силову електроенергію ($В_е$) розраховують за формулою:

$$В_е = \sum_{i=1}^{п} \frac{W_{уі} * t_і * Ц_е * К_{впі}}{\eta_і} \quad (4.6)$$

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

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 12,5 грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

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

$$В_е = \frac{0{,}25 * 40 * 12{,}5 * 0{,}95}{0{,}97} = 122{,}4 \text{ грн.}$$

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук HP ProBook 455 G9	0,25	40	122,4
Монітор Dell P2422H	0,07	420	359,9
Серверний міні-комп'ютер Intel NUC	0,8	384	3760,8
Офісне приміщення (оренда в розрахунку на амортизацію)	0,28	384	1316,3
МФУ (принтер-сканер) Canon i-Sensys	0,15	15	27,5
Всього			5 586,9

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%} \quad (4.7)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = (58\,636) * \frac{25}{100} = 14\,659 \text{ грн.}$$

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{сп} = (З_о + З_р) * \frac{Н_{сп}}{100\%} \quad (4.8)$$

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

$$В_{сп} = (58\,636) * \frac{35}{100} = 20\,522{,}6 \text{ грн.}$$

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) * \frac{Н_{ів}}{100\%} \quad (4.9)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = (58\ 636) * \frac{55}{100} = 32\ 249{,}8 \text{ грн.}$$

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%} \quad (4.10)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = (58\ 636) * \frac{100}{100} = 58\ 636 \text{ грн.}$$

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

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.11)$$

$$В_{заг} = 58\ 636 + 5\ 863{,}6 + 14\ 189{,}9 + 847 + 5\ 891{,}5 + 5\ 586{,}9 + 14\ 659 + 20\ 522{,}6 + 32\ 249{,}8 + 58\ 636 = 217\ 082{,}3 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.12)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η =0,1; технічного проектування, то η =0,2; розробки конструкторської документації, то η =0,3; розробки технологій, то η =0,4; розробки дослідного зразка, то η =0,5; розробки промислового зразка, то η =0,7; впровадження, то η =0,9 [42].

$$ЗВ = \frac{217\ 082{,}3}{0{,}7} = 310\ 117{,}6 \text{ грн.}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У ринкових умовах ключовим позитивним результатом для потенційного інвестора від упровадження результатів науково-технічної розробки є зростання чистого прибутку та підвищення конкурентоспроможності програмного продукту.

Дослідження за темою «Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC із динамічним регулюванням доступу та поведінковою класифікацією загроз» передбачають поетапну комерціалізацію розробленого рішення протягом трирічного циклу. Очікуваний економічний ефект формуватиметься на підставі таких даних:

Приріст кількості користувачів (ΔN), який прогнозується внаслідок підвищення функціональних характеристик та рівня інформаційної безпеки продукту:

– 1-й рік – 80 користувачів;

– 2-й рік – 70 користувачів;

– 3-й рік – 60 користувачів.

Базова кількість користувачів (N), які використовували попереднє або аналогічне рішення до впровадження вдосконаленої моделі, становить 120 користувачів.

Вартість програмного забезпечення до впровадження нової моделі захисту ($Ц_о$) становила 150 000 грн.

Зміна вартості продукту (±ΔЦ) унаслідок інтеграції моделі CapBAC, поведінкової аналітики та динамічного контролю доступу становить +3 500 грн, що враховує покращення безпекових характеристик та додаткові можливості системи.

Можливе збільшення чистого прибутку потенційного інвестора для кожного з трьох років, протягом яких очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, розраховується за формулою.

$$\Delta П_i = (\pm \Delta Ц_б * N + Ц_б * \Delta N)_i * \lambda * \rho * (1 - \frac{\vartheta}{100}) \qquad (4.13)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 25\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

1-й рік: $\Delta П_1 = (3500 \times 120 + 150\,000 \times 80) \times 0{,}83 \times 0{,}25 \times \left(1 - \frac{0{,}18}{100}\right) = 2\,572\,511{,}3$ (грн.)

2-й рік: $\Delta П_2 = (3500 \times 120 + 150\,000 \times (80 + 70)) \times 0{,}83 \times 0{,}25 \times \left(1 - \frac{0{,}18}{100}\right) = 4\,747\,339{,}4$ (грн.)

3-й рік: $\Delta П_3 = (3500 \times 120 + 150\,000 \times (80 + 70 + 60)) \times 0{,}83 \times 0{,}25 \times \left(1 - \frac{0{,}18}{100}\right) = 6\,611\,477{,}9$ (грн.)

Отже, за результатами обчислень, впровадження розробки призведе до значної комерційної вигоди, що виявиться у зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Ключовими критеріями, що впливають на рішення інвестора щодо фінансування наукової розробки, є абсолютна та відносна ефективність інвестицій, а також термін їх окупності.

На початковому етапі визначається теперішня вартість інвестицій (PV), які будуть спрямовані на наукову розробку.

Також розраховується обсяг початкових вкладень, які потенційний інвестор повинен здійснити для впровадження та комерціалізації науково-технічного проєкту [42].

$$PV = k_{інв} * ЗВ \quad (4.14)$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 310 117,6 грн.

$$PV = 2 * 310\ 117{,}6 = 620\ 235 \text{ грн.}$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \quad (4.15)$$

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

PV – теперішня вартість початкових інвестицій, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_1^T \frac{\Delta П_1}{(1+\tau)^t} \quad (4.16)$$

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

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

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

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

$$ПП = \frac{2\ 572\ 511{,}3}{(1+0{,}2)^1} + \frac{4\ 747\ 339{,}4}{(1+0{,}2)^2} + \frac{6\ 611\ 477{,}9}{(1+0{,}2)^3} = 9\ 266\ 609{,}6 \text{ грн.}$$

$$E_{абс} = 9\ 266\ 609{,}6 - 620\ 235 = 8\ 646\ 374{,}6 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, встановлено, що проведення наукових досліджень для розробки програмного продукту та його подальше впровадження принесуть прибуток. Це підтверджує доцільність проведення досліджень [42].

Внутрішня економічна дохідність інвестицій $Е_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (4.17)$$

де $Е_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$Т_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$Е_в = \sqrt[3]{1 + \frac{8\,646\,374{,}6}{620\,235}} - 1 = 1{,}5$$

Порівняємо Ев з мінімальною (бар'єрною) ставкою дисконтування τ_{min}, яка визначає мінімальну дохідність, нижче якої інвестиції не будуть здійснюватися.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ_{min} визначається за формулою:

$$\tau_{min} = d + f \quad (4.18)$$

d – середньозважена ставка за депозитними операціями в комерційних банка;

f – показник, що характеризує ризикованість вкладень; *f* = 0,4.

d = 0,2.

$$\tau_{min} = 0{,}2 + 0{,}4 = 0{,}6$$

Оскільки $E_в = 150\% > \tau_{min} = 60\%$, то у інвестора є потенційна зацікавленість у фінансуванні даної наукової розробки.

Далі розраховуємо період окупності інвестицій $Т_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки [42]:

$$T_{ок} = \frac{1}{E_в} \quad (4.19)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{1,5} = 0,6$$

Якщо $T_{ок}$ < 3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.5 Висновки до розділу

Проведено оцінювання комерційного та технологічного потенціалу вдосконаленого механізму захисту систем віртуалізації, розробленого на основі моделі CapBAC із динамічним регулюванням доступу та інтегрованою поведінковою класифікацією загроз. Запропоноване рішення забезпечує підвищення рівня адаптивності системи контролю доступу, оперативне виявлення аномальних дій користувачів і проактивне реагування на потенційні загрози, що суттєво підсилює захищеність віртуалізованого середовища.

Аналіз витрат, необхідних для виконання науково-дослідної роботи, розробки та впровадження вдосконаленого методу, підтвердив економічну доцільність проєкту. Інвестиції, спрямовані на створення та інтеграцію запропонованого механізму захисту, характеризуються швидким строком окупності завдяки підвищенню цінності продукту, зростанню попиту на інтелектуальні системи контролю доступу та актуальності рішень, орієнтованих на поведінковий аналіз загроз.

Прогнозований економічний ефект від комерціалізації розробки свідчить про значний потенціал отримання чистого прибутку у середньостроковій перспективі. Реалізація вдосконаленого механізму CapBAC дозволить підприємству зміцнити свої позиції на ринку рішень інформаційної безпеки, розширити цільову аудиторію та забезпечити довгострокові конкурентні переваги.

ВИСНОВКИ

У дипломній роботі проведено комплексне дослідження проблем безпеки віртуалізованих середовищ та розроблено вдосконалену модель захисту системи віртуалізації з контролем доступу на основі CapBAC, що враховує динамічні поведінкові характеристики користувачів. Отримані результати підтверджують актуальність обраної тематики та практичну ефективність запропонованих рішень для підвищення рівня інформаційної безпеки в сучасних інфраструктурах.

У першому розділі було проведено теоретичний аналіз архітектурних особливостей систем віртуалізації, визначено основні вектори атак на гіпервізор, віртуальні машини, мережеві сегменти та сервіси управління. Проаналізовано існуючі моделі контролю доступу — RBAC, ABAC та CapBAC — їх сильні та слабкі сторони, а також можливості інтеграції з поведінковими моделями користувачів. Проведений огляд існуючих методів захисту показав обмеженість традиційних політик у динамічних та високонавантажених середовищах, що визначило постановку задачі створення адаптивної моделі доступу.

У другому розділі сформовано вдосконалену модель захисту на основі CapBAC, що передбачає використання капабіліті-токенів із розширеними атрибутами, прив’язаними до контексту, середовища та поведінкових параметрів користувача. Розроблено алгоритм динамічного прийняття рішень щодо доступу, який комбінує статичні політики та поведінкові метрики, що дозволяє оперативно реагувати на аномалії. Також описано модель взаємодії між компонентами системи віртуалізації, включно з гіпервізором, контролером доступу, модулем аналітики та системою моніторингу.

У третьому розділі представлено програмну реалізацію ключових модулів: керування токенами CapBAC, поведінкової аналітики, механізмів адаптації політик і підсистеми обробки подій. Тестування проведено у наближеному до реального середовищі з використанням типових та аномальних сценаріїв доступу. Результати експериментів показали, що запропонована система забезпечує стабільну роботу, не створює критичного навантаження та демонструє високу точність виявлення ризикових дій. Порівняльний аналіз свідчить про переваги CapBAC з поведінковою

адаптацією над RBAC та ABAC у контексті динамічних змін доступу та протидії складним атакам.

Проведене дослідження підтвердило, що запропонована модель є ефективною для практичного застосування у віртуалізованих інфраструктурах підприємств. Вона підвищує рівень захищеності системи за рахунок гнучкого управління доступом, раннього виявлення аномалій та мінімізації ризиків, пов’язаних з компрометацією облікових даних та привілейованим доступом.

Отримані результати можуть бути використані як основа для подальших досліджень щодо інтеграції моделі CapBAC з методами машинного навчання, побудови інтелектуальних політик доступу, а також для розробки систем безпеки наступного покоління для контейнерних технологій та хмарних середовищ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stallings W. *Cryptography and Network Security: Principles and Practice*. — 8th ed. — Pearson, 2022. — 768 p.
2. Lee J., Park S. *Adaptive Access Control in Virtualized Environments Based on User Behavior*. // IEEE Access, 2022.
3. ISO/IEC 27001:2022. *Information security, cybersecurity and privacy protection — Information security management systems — Requirements*.
4. NIST SP 800-207. *Zero Trust Architecture*. — 2021.
5. NIST SP 800-204C. *Capabilities-based access control (CapBAC) in cloud-native applications*. — 2021.
6. Davis R., Kumar P. *Behavior-Driven Security for Cloud Systems*. Springer, 2021.
7. Dargahi T., Conti M. *Virtualization Security: Challenges and Solutions*. ACM Computing Surveys, 2021.
8. Song L. *Behavior-Based Threat Classification in Virtual Environments*. // Elsevier Computers & Security, 2020.
9. NIST SP 800-53 Rev.5. *Security and Privacy Controls for Information Systems and Organizations*. — 2020.
10. ISO/IEC 29115:2020. *Entity Authentication Assurance Framework*.
11. Tanenbaum A. *Modern Operating Systems*. — 4th ed. — Pearson, 2020.
12. Conti M., Lal C. *Virtualization Technologies for Cybersecurity*. Springer, 2019.
13. Fedotov A. A. *Веб-програмування. Серверна частина*. — СПб.: Питер, 2019. — 448 с.
14. Dey S., Mehrota S. *Anomaly Detection and Behavioral Analytics in Cloud Security*. IEEE Cloud Computing, 2019.
15. Amazon Web Services. *AWS Virtualization Security Whitepaper*. — 2019.
16. Microsoft Azure. *Security Architecture for Virtual Machines*. — 2019.
17. Google Cloud. *Confidential VMs: Architecture Overview*. — 2019.

18. RFC 8259: *The JavaScript Object Notation (JSON) Data Interchange Format*. — IETF, 2017.

19. ДСТУ ISO/IEC 18033-3:2017. *Алгоритми криптографічні. Блокові шифри*.

20. Python Software Foundation. *Python 3.12 Documentation* [Електронний ресурс]. — Режим доступу: https://docs.python.org/3/

21. Столяренко О. М. *Захист інформації в комп'ютерних системах*. — К.: КНЕУ, 2017. — 312 с.

22. Шнайєр Б. *Прикладна криптографія: протоколи та алгоритми*. — К.: Вільямс, 2016. — 784 с.

23. Таненбаум Е. *Архітектура комп'ютерних мереж*. — К.: Вільямс, 2015. — 1024 с.

24. RFC 7519: *JSON Web Token (JWT)*. — IETF, 2015.

25. European Union Agency for Cybersecurity (ENISA). *Virtualization Security Guidance*. — 2015.

26. Grinshtein M., Spatschek R. *Протоколи Інтернету: принципи, аналіз, реалізація*. — Діалектика, 2010. — 432 с.

27. Sandhu R., Samarati P. *Access Control: Principles and Practice*. IEEE Communications Surveys, 2010.

28. ISO/IEC 19790:2012. *Security Requirements for Cryptographic Modules*.

29. Bishop M. *Introduction to Computer Security*. — Addison-Wesley, 2010.

30. CapBAC Consortium. *Capabilities-Based Security Architecture* — Tech Report, 2009.

31. Shen E., Wang Q. *Security of Virtual Machines in Cloud Computing*. — IEEE, 2009.

32. Garfinkel T., Rosenblum M. *Virtual Machine Security: Architecture, Implementation and Control*. ACM, 2008.

33. RFC 5246: *The Transport Layer Security (TLS)*. — IETF, 2008.

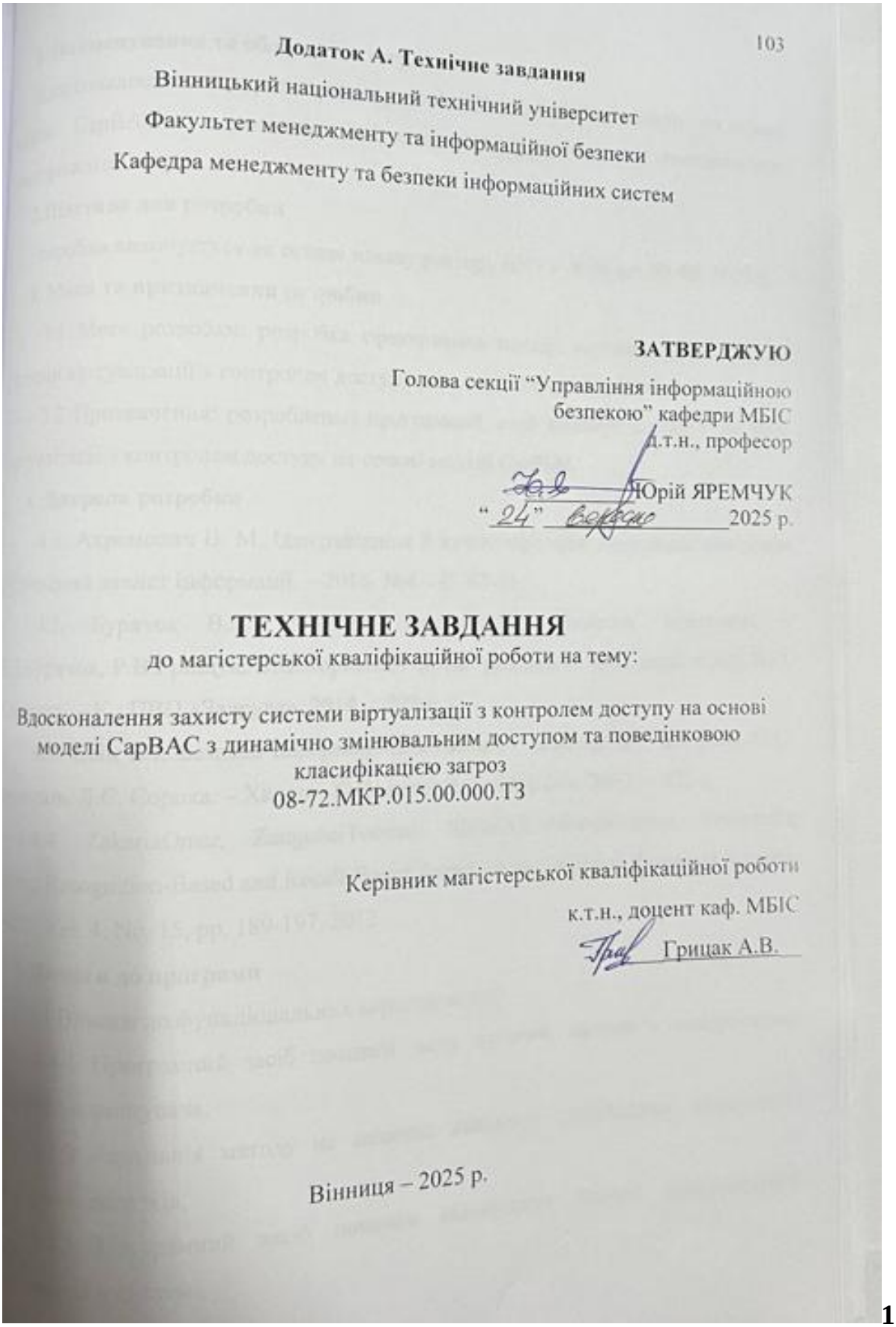
34. Proakis J. *Digital Communications*. — McGraw-Hill, 2008.

35. Myers J. *Practical Authentication for the Enterprise*. O'Reilly, 2007.

36. ISO/IEC 15408 (Common Criteria). *Evaluation Criteria for IT Security*. — 2006.

ДОДАТКИ

Додаток А. Технічне завдання



Додаток А. Технічне завдання 103

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС

д.т.н., професор

_____ Юрій ЯРЕМЧУК

"24" вересня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC з динамічно змінювальним доступом та поведінковою класифікацією загроз

08-72.МКР.015.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент каф. МБІС

_____ Грицак А.В.

Вінниця – 2025 р.

1.

1.Найменування та область застосування

Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC з динамічно змінювальним доступом та поведінковою класифікацією загроз

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №96 від 20. 03. 2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC

3.2 Призначення: розроблений програмний засіб виконує в захист системи віртуалізації з контролем доступу на основі моделі CapBAC

4. Джерела розробки

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

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

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

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

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

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

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

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

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

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

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

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

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

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми		
2	Аналіз предметної області обраної теми		
3	Апробація отриманих результатів		
4	Розробка алгоритму роботи		
5	Написання магістерської роботи на основі розробленої теми		
6	Розробка економічної частини		
7	Передзахист магістерської кваліфікаційної роботи		
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи		
9	Захист магістерської кваліфікаційної роботи		

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Матвієнко Д.В.

Додаток Б. Лістинг програми

```
from cryptography.hazmat.primitives.asymmetric import ed25519

private_key = ed25519.Ed25519PrivateKey.generate()
public_key = private_key.public_key()
import time
import json
import hashlib

def generate_token_payload(subject_id, capabilities, context, behavior_vector, lifetime=60):
    behavior_hash = hashlib.sha256(str(behavior_vector).encode()).hexdigest()

    payload = {
        "subject": subject_id,
        "capabilities": capabilities,
        "context": context,
        "behavior_hash": behavior_hash,
        "exp": int(time.time()) + lifetime
    }
    return payload
def sign_token(payload, private_key):
    message = json.dumps(payload).encode()
    signature = private_key.sign(message)

    token = {
        "payload": payload,
        "signature": signature.hex()
    }
    return token
def verify_signature(token, public_key):
    message = json.dumps(token["payload"]).encode()
    signature = bytes.fromhex(token["signature"])

    try:
        public_key.verify(signature, message)
        return True
    except Exception:
def verify_expiration(payload):
    return time.time() < payload["exp"]
```

```
def verify_behavior(payload, current_behavior_vector):
    current_hash = hashlib.sha256(str(current_behavior_vector).encode()).hexdigest()
    return current_hash == payload["behavior_hash"]
def verify_context(payload, current_context):
    for key in payload["context"]:
        if payload["context"][key] != current_context.get(key):
            return False
    return True
```

Пояснення

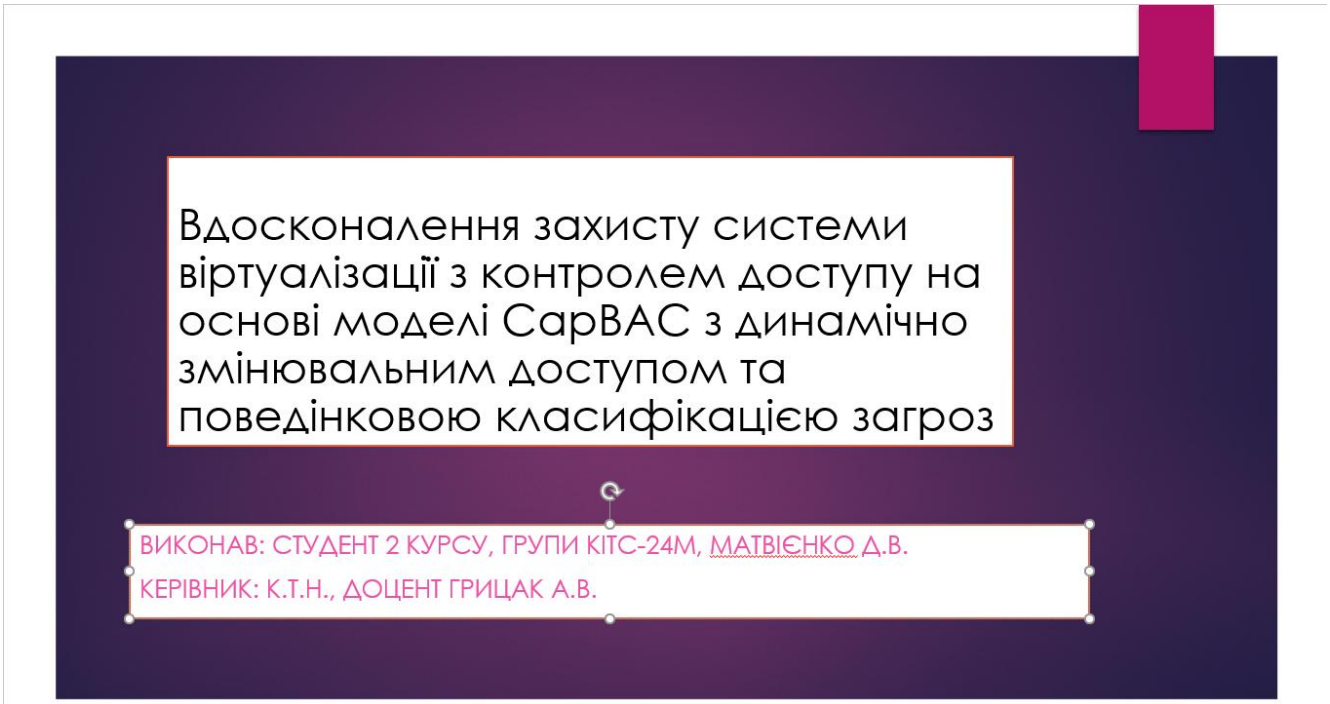
Поєднує всі етапи перевірки в єдину функцію.

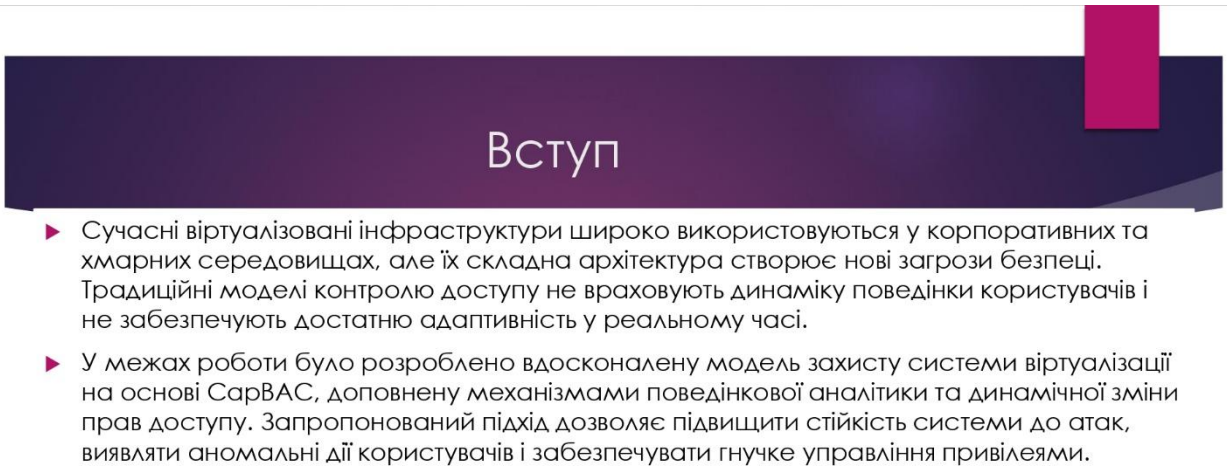
```
def validate_token(token, public_key, current_context, behavior_vector):
    if not verify_signature(token, public_key):
        return False, "Invalid signature"

    if not verify_expiration(token["payload"]):
        return False, "Token expired"

    if not verify_behavior(token["payload"], behavior_vector):
        return False, "Behavior mismatch"

    if not verify_context(token["payload"], current_context):
        return False, "Context mismatch"

    return True, "Valid token"
revoked_tokens = set()

def revoke_token(token_id):
    revoked_tokens.add(token_id)

def is_revoked(token_id):
    return token_id in revoked_tokens
payload = generate_token_payload(
    subject_id="vm_12",
    capabilities=["start", "stop", "migrate"],
    context={"ip": "10.0.0.5"},
    behavior_vector=[1.2, 0.9, 3.1]
)

token = sign_token(payload, private_key)
```

```
status, message = validate_token(
    token,
    public_key,
    current_context={"ip": "10.0.0.5"},
    behavior_vector=[1.2, 0.9, 3.1]
)

print(status, message)
revoke_token(payload["subject"])

if is_revoked(payload["subject"]):
    print("Token revoked")
import numpy as np
import time

class BehaviorProfile:
    def __init__(self):
        self.actions = []
        self.timestamps = []
        self.resources = []

    def update(self, action, resource):
        self.actions.append(action)
        self.resources.append(resource)
        self.timestamps.append(time.time())

    def vectorize(self):
        freq = len(self.actions)
        unique_actions = len(set(self.actions))
        active_resources = len(set(self.resources))
        avg_interval = np.mean(np.diff(self.timestamps)) if len(self.timestamps) > 1 else 0

        vector = np.array([freq, unique_actions, active_resources, avg_interval])
        return vector
def anomaly_score(current_vec, baseline_vec):
    distance = np.linalg.norm(current_vec - baseline_vec)
    return distance
def adapt_policy(risk):
    if risk < 0.3:
```

```
        return {"token_lifetime": 120, "access": "full"}
    elif risk < 0.7:
        return {"token_lifetime": 30, "access": "limited"}
    else:
        return {"token_lifetime": 5, "access": "restricted"}
baseline = BehaviorProfile()
baseline.update("start_vm", "vm12")
baseline.update("stop_vm", "vm12")

baseline_vector = baseline.vectorize()

# Новий профіль користувача в реальному часі
current = BehaviorProfile()
current.update("start_vm", "vm12")
current.update("migrate", "vm17")
current.update("migrate", "vm17")

current_vector = current.vectorize()

# Аналіз поведінки
score = anomaly_score(current_vector, baseline_vector)
risk = risk_level(score)

policy_decision = adapt_policy(risk)

print("Аномалія:", score)
print("Рівень ризику:", risk)
from datetime import datetime

def apply_policy_to_token(token_payload, policy):
    token_payload["exp"] = int(time.time()) + policy["token_lifetime"]
    token_payload["policy_applied"] = policy
    token_payload["updated_at"] = datetime.utcnow().isoformat()
    return token_payload
```

Додаток В. Ілюстративний матеріал





Актуальність та мета

- **Актуальність**

Зростання використання віртуалізації підвищує вимоги до безпеки, але традиційні моделі доступу не здатні ефективно реагувати на динамічні загрози та поведінкові аномалії. Потрібні адаптивні механізми, здатні працювати в реальному часі.

- **Мета**

Підвищити рівень захисту системи віртуалізації шляхом розробки вдосконаленої моделі CapBAC з динамічно змінюваним доступом та поведінковою класифікацією загроз.

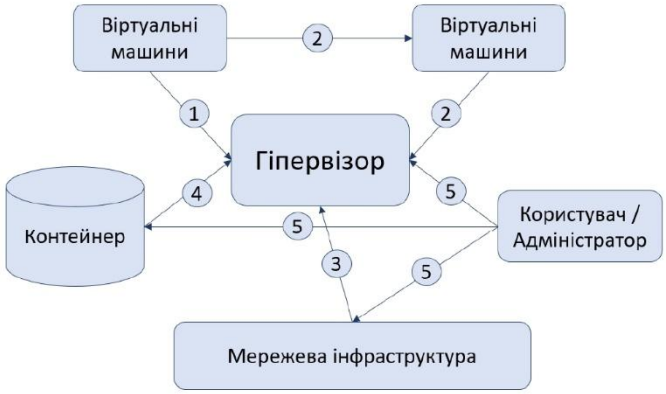
Архітектура системи віртуалізації

Рівень додатків (Application Layer — веб-сервіси, API)
Гостьова операційна система (Guest OS — Windows, Linux, BSD тощо)
Гіпервізор (VMM) (Virtual Machine Monitor — керування ресурсами)
Хостова операційна система (Host OS — для типу 2 гіпервізорів, наприклад)
Апаратне забезпечення (CPU, RAM, Disk, Network, GPU, I/O Controller)

Порівняльна характеристика типів гіпервізорів

Тип	Опис	Приклади	Переваги	Недоліки
Тип 1 (bare-metal)	Працює безпосередньо на апаратному рівні без проміжної ОС.	VMware ESXi, Microsoft Hyper-V Server, Xen	Висока продуктивність, ізоляція, стабільність, безпека	Складність налаштування, потребує спеціального обладнання
Тип 2 (hosted)	Запускається поверх хостової ОС як звичайна програма.	VirtualBox, VMware Workstation, Parallels Desktop	Простота встановлення, гнучкість, сумісність із настільними системами	Нижча продуктивність, більша поверхня атаки, залежність від хостової ОС

Основні вектори атак у системі віртуалізації



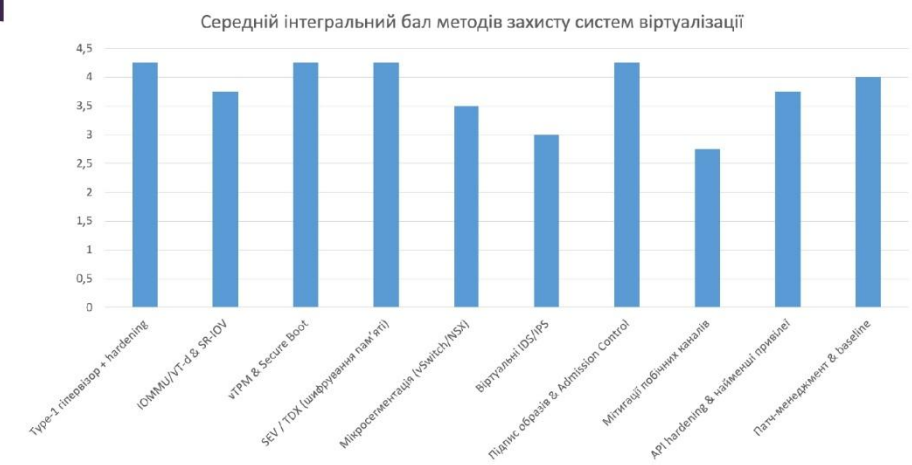
Архітектура моделі контролю доступу CapBAC у віртуалізованому середовищі

Аналіз існуючих методів захисту систем віртуалізації та моделей контролю доступу

Порівняння моделей контролю доступу в системах віртуалізації

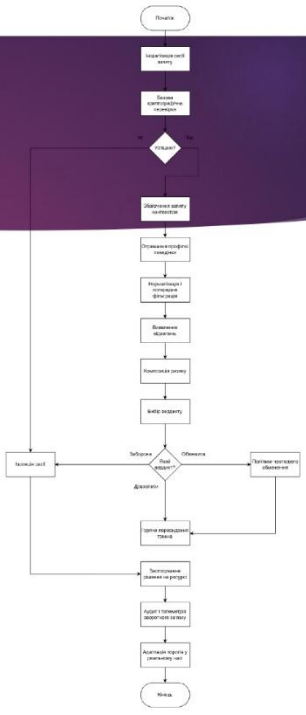
	Ефективність безпеки	Ефективність продуктивності	Простота адміністрування	Ізоляція multi-tenant
Type-1 гіпервізор + hardening	4	5	4	4
IOMMU/VT-d & SR-IOV	4	4	3	4
vTPM & Secure Boot	4	5	4	4
SEV / TDX (шифрування пам'яті)	5	4	3	5
Мікросегментація (vSwitch/NSX)	4	3	3	4
Віртуальні IDS/IPS	3	3	3	3
Підпис образів & Admission Control	4	5	4	4
Мітигації побічних каналів	3	3	2	3
API hardening & найменші привілеї	4	4	4	3
Патч-менеджмент & baseline	4	5	4	3

Середній інтегральний бал методів захисту систем віртуалізації

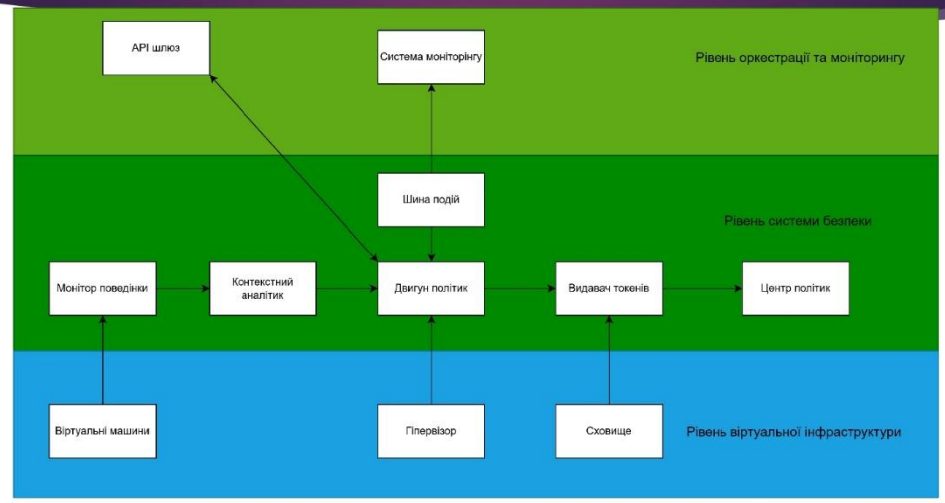


Діаграма модифікованої моделі Dynamic CapBAC

Розроблений алгоритм прийняття рішення з урахуванням поведінкових параметрів



Багаторівнева модель взаємодії між компонентами системи віртуалізації



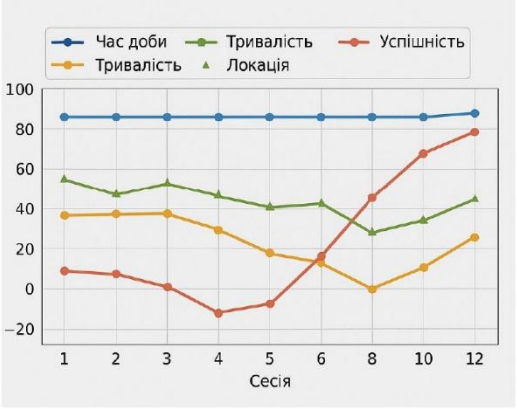
Visual Studio Code

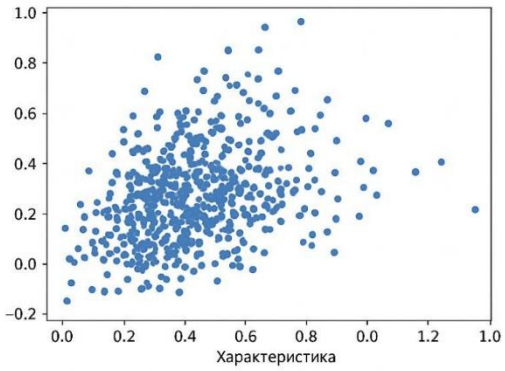
Середовище розробки і мова програмування

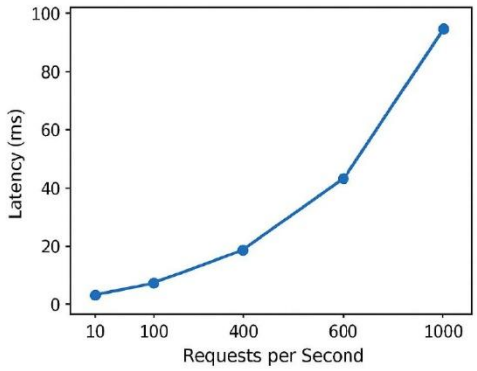
Результати тестування системи

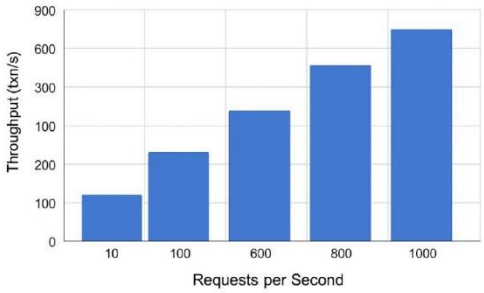
```
        Token Generation and Signature Test
        ✓ Token generated successfully
        ✓ Token signed successfully
Token:
   eyJhbGciOiEFUZ15iIsInT5cCI6IkpXVCJ9.........

Token Signature:
   vF58ECKDAhTuS4kQPmqR1UUnspn2Pb8Sbgj356f3UMVKdtfsS1Du
   q2C1yEKcQgIM2t5uQqq48uHuSNn3ndkX_-8JbiJCgXgrrI
   DU31UsaLe

----------------------------------------------------
Ran 1 test in 0.006s

OK
```

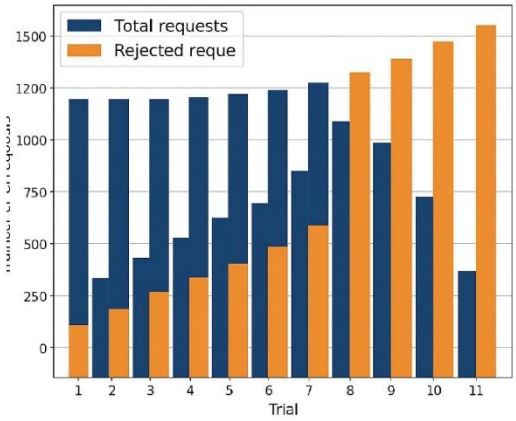
```
2025-11-23 10:41:50 INFO Generated
capability token f7e5....
2025-11-23 10:41:50 INFO Verifying
token f7e5...
2025-11-23 10:41:50 WARNING Invalid
signature for token
2025-11-23 10:41:50 Rejected access
due to invalid signature
```

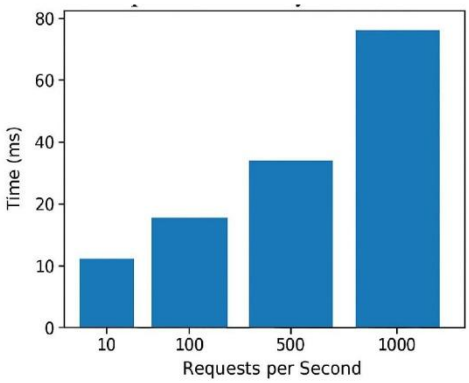


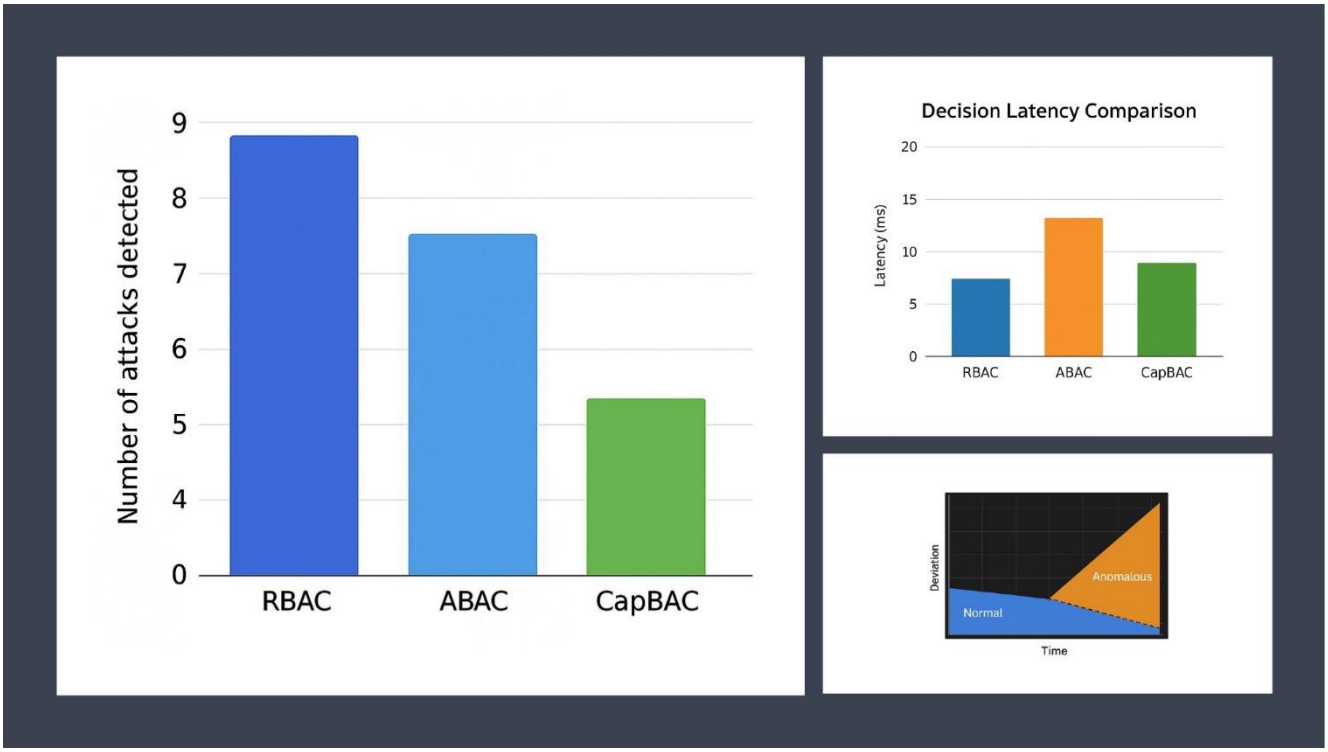



Latency (ms)
100
80
60
40
20
0
10
100
400
600
1000
Requests per Second


Throughput (txn/s)
900
600
300
100
200
100
0
10
100
600
800
1000
Requests per Second


Total requests
Rejected reque
1500
1200
1000
750
500
250
0
1
2
3
4
5
6
7
8
9
10
11
Trial


Time (ms)
80
60
40
20
10
0
10
100
500
1000
Requests per Second



Висновок

- Створена вдосконалена модель захисту системи віртуалізації на основі CapBAC дозволила поєднати механізми контролю доступу з поведінковою аналітикою, забезпечивши динамічну зміну прав користувачів у реальному часі. Проведена програмна реалізація та тестування підтвердили ефективність підходу: система своєчасно реагує на аномальні дії, підвищує рівень безпеки віртуалізованої інфраструктури та не створює суттєвого навантаження на ресурси. Отримані результати доводять практичну придатність моделі для корпоративних і хмарних середовищ та відкривають можливості для подальшого розвитку адаптивних систем контролю доступу.

Дякую за увагу!

Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC з динамічно змінювальним доступом та поведінковою класифікацією загроз

ВИКОНАВ: СТУДЕНТ 2 КУРСУ, ГРУПИ КІТС-24М, МАРЦУН Н.М.
КЕРІВНИК: К.Т.Н., ДОЦЕНТ ГРИЦАК А.В.

Вступ

- Сучасні віртуалізовані інфраструктури широко використовуються у корпоративних та хмарних середовищах, але їх складна архітектура створює нові загрози безпеці. Традиційні моделі контролю доступу не враховують динаміку поведінки користувачів і не забезпечують достатню адаптивність у реальному часі.
- У межах роботи було розроблено вдосконалену модель захисту системи віртуалізації на основі CapBAC, доповнену механізмами поведінкової аналітики та динамічної зміни прав доступу. Запропонований підхід дозволяє підвищити стійкість системи до атак, виявляти аномальні дії користувачів і забезпечувати гнучке управління привілеями.

Актуальність та мета

- **Актуальність**

Зростання використання віртуалізації підвищує вимоги до безпеки, але традиційні моделі доступу не здатні ефективно реагувати на динамічні загрози та поведінкові аномалії. Потрібні адаптивні механізми, здатні працювати в реальному часі.

- **Мета**

Підвищити рівень захисту системи віртуалізації шляхом розробки вдосконаленої моделі CapBAC з динамічно змінюваним доступом та поведінковою класифікацією загроз.

Архітектура системи віртуалізації

Порівняльна характеристика типів гіпервізорів

Тип	Опис	Приклади	Переваги	Недоліки
Тип 1 (bare-metal)	Працює безпосередньо на апаратному рівні без проміжної ОС.	VMware ESXi, Microsoft Hyper-V Server, Xen	Висока продуктивність, ізоляція, стабільність, безпека	Складність налаштування, потребує спеціального обладнання
Тип 2 (hosted)	Запускається поверх хостової ОС як звичайна програма.	VirtualBox, VMware Workstation, Parallels Desktop	Простота встановлення, гнучкість, сумісність із настільними системами	Нижча продуктивність, більша поверхня атаки, залежність від хостової ОС

Основні вектори атак у системі віртуалізації

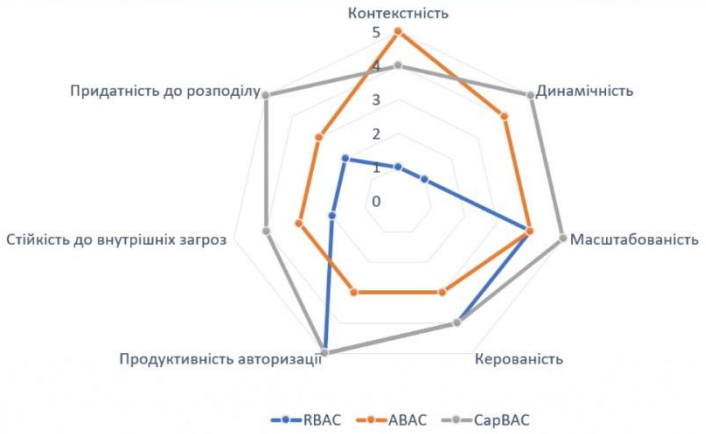
Архітектура моделі контролю доступу CapBAC у віртуалізованому середовищі

Аналіз існуючих методів захисту систем віртуалізації та моделей контролю доступу

Порівняння моделей контролю доступу за основними критеріями

	RBAC	ABAC	CapBAC
Контекстність	1	5	4
Динамічність	1	4	5
Масштабованість	4	4	5
Керованість	4	3	4
Продуктивність авторизації	5	3	5
Стійкість до внутрішніх загроз	2	3	4
Придатність до розподілу	2	3	5

Радар-діаграма порівняння моделей RBAC, ABAC і CapBAC за критеріями безпеки



Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Вдосконалення захисту системи віртуалізації з контролем доступу на основі моделі CapBAC з динамічно змінювальним доступом та поведінковою класифікацією загроз

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра менеджменту та безпеки інформаційних систем
факультет менеджменту та інформаційної безпеки
гр.2КІТС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) **1,16 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту**

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В. _____

к.ф.-м.н., доцент каф. МБІС Шиян А.А. _____

Особа, відповідальна за перевірку Коваль Н.П. _____

З висновком експертної комісії ознайомлений(-на) доц. Грицак А.В.
Матвієнко Д.В.

Керівник _____

Здобувач _____