Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему: «Проєкт підсистеми кластеризованої багатофакторної автентифікації користувачів для доменної інфраструктури Windows Server»

Виконав: студент 4-го курсу групи КІ-22-1
спеціальності 123 «Комп'ютерна інженерія»
_____ Артем ДАВИДЕНКО
Керівник: Старший викладач кафедри КІ та КБ
_____ Ігор ФАЛЬКОВСЬКИЙ
Рецензент: Доцент кафедри КІ та КБ, к.т.н.
_____ Олексій ШЕЛУХА

Житомир – 2026

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
Андрій Єфіменко
2 лютого 2026 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу бакалавра

Студент: Давиденко Артем Анатолійович

Тема роботи: «Проєкт підсистеми кластеризованої багатофакторної автентифікації користувачів для доменної інфраструктури Windows Server»

затверджена Наказом університету від 2 лютого 2026 р. №57/с

Термін здачі студентом закінченої роботи 8 червня 2026 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

- процес забезпечення безпеки автентифікації користувачів у корпоративних інформаційних системах на базі Microsoft Active Directory;
- технології багатофакторної автентифікації, механізми інтеграції PrivacyIDEA з Microsoft Active Directory, а також засоби побудови кластерів;

Обладнання та програмне забезпечення:

1. Oracle VirtualBox 7.2.2
2. Pacemaker 3.0.1
3. Windows Server 2022
4. Ubuntu Server 20.04 LTS

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

Студент _____ Артем ДАВИДЕНКО
Керівник _____ Ігор ФАЛЬКОВСЬКИЙ

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання	Примітка
1	Вибір теми випускної кваліфікаційної роботи та аналіз проблематики багатофакторної автентифікації	02.02.2026	Виконано
2	Розробка та затвердження структури пояснювальної записки	06.02.2026	Виконано
3	Аналіз сучасних методів багатофакторної автентифікації та типів MFA-атак	16.02.2026	Виконано
4	Дослідження архітектури та принципів роботи PrivacyIDEA	23.02.2026	Виконано
5	Аналіз альтернативних MFA-рішень та обґрунтування вибору PrivacyIDEA	02.03.2026	Виконано
6	Дослідження принципів кластеризації Linux-систем та технологій HA-кластерів	09.03.2026	Виконано
7	Проєктування тестового віртуального середовища та схеми адресації	13.03.2026	Виконано
8	Розгортання середовища VirtualBox та встановлення Windows Server 2022	20.03.2026	Виконано
9	Налаштування Active Directory Domain Services, DHCP та доменної структури	27.03.2026	Виконано
10	Встановлення та базове налаштування серверів Ubuntu Server 20.04 LTS	04.04.2026	Виконано
11	Встановлення та первинна конфігурація PrivacyIDEA; Інтеграція PrivacyIDEA з Active Directory через LDAP Resolver та Realm	10.04.2026	Виконано
12	Налаштування PostgreSQL та взаємодії між серверами MFA	24.04.2026	Виконано
13	Реалізація HA-кластеризації PrivacyIDEA з використанням Pacemaker і Corosync	06.05.2026	Виконано
14	Налаштування політик MFA, токенів та тестових користувачів	18.05.2026	Виконано
15	Проведення тестування багатофакторної автентифікації та відмовостійкості кластера	25.05.2026	Виконано
16	Аналіз результатів тестування та аудит роботи системи	29.05.2026	Виконано

Студент _____ Артем ДАВИДЕНКО

Керівник _____ Ігор ФАЛЬКОВСЬКИЙ

АНОТАЦІЯ

Анотація кваліфікаційної роботи бакалавра з теми «Проєкт підсистеми кластеризованої багатофакторної автентифікації користувачів для доменної інфраструктури Windows Server» містить відомості про розробку та тестування системи багатофакторної автентифікації для доменної інфраструктури Microsoft Active Directory. Обсяг пояснювальної записки становить 90 сторінок, які містять 122 рисунки, 3 таблиці та 5 додатків. Список використаних джерел включає 14 найменувань та розміщений на 2 сторінках.

Мета роботи: розробити та протестувати систему багатофакторної автентифікації для середовища Windows Server з використанням open source рішення PrivacyIDEA.

У ході виконання роботи було проаналізовано сучасні методи багатофакторної автентифікації, основні типи MFA-атак та проблеми однофакторної автентифікації. Проведено дослідження архітектури та принципів роботи PrivacyIDEA, а також виконано порівняльний аналіз open source і комерційних MFA-рішень. Окрему увагу приділено технологіям кластеризації Linux-систем та механізмам забезпечення високої доступності сервісів.

У практичній частині роботи було розгорнуто тестове віртуальне середовище на базі Windows Server 2022, Ubuntu Server 20.04 LTS та PostgreSQL. Реалізовано доменну інфраструктуру Active Directory, виконано інтеграцію PrivacyIDEA з LDAP-каталогом та налаштовано систему MFA для користувачів домену. Також проведено тестування працездатності та відмовостійкості системи.

Ключові слова: БАГАТОФАКТОРНА АВТЕНТИФІКАЦІЯ, PRIVACYIDEA, WINDOWS SERVER 2022, ACTIVE DIRECTORY, OPEN SOURCE, LDAP, POSTGRESQL, PACEMAKER, COROSYNC, НА-КЛАСТЕР, ІНФОРМАЦІЙНА БЕЗПЕКА, MFA.

ANNOTATION

The annotation of the bachelor's qualification work on the topic “Design of a Clustered Multi-Factor User Authentication Subsystem for Windows Server Domain Infrastructure” provides information on the development and testing of a multi-factor authentication system for the Microsoft Active Directory domain infrastructure. The explanatory note consists of 90 pages, containing 122 figures, 3 tables, and 5 appendices. The list of references includes 14 titles and spans 2 pages.

The purpose of the work is to develop and test a multi-factor authentication system for the Windows Server environment using the open-source solution PrivacyIDEA.

During the course of the work, modern methods of multi-factor authentication, the main types of MFA attacks, and the problems of single-factor authentication were analyzed. A study of the architecture and operating principles of PrivacyIDEA was conducted, and a comparison of open-source and commercial MFA solutions was performed. Special attention was paid to Linux system clustering technologies and mechanisms for ensuring high service availability.

In the practical part of the work, a test virtual environment was deployed based on Windows Server 2022, Ubuntu Server 20.04 LTS, and PostgreSQL. An Active Directory domain infrastructure was implemented, PrivacyIDEA was integrated with the LDAP directory, and the MFA system was configured for domain users. Performance and fault tolerance testing of the system was also conducted.

Keywords: MULTIFACTOR AUTHENTICATION, PRIVACYIDEA, WINDOWS SERVER 2022, ACTIVE DIRECTORY, OPEN SOURCE SOFTWARE, LDAP, POSTGRESQL, PACEMAKER, COROSYNC, HA CLUSTER, INFORMATION SECURITY, MFA.

ЗМІСТ

ВСТУП.....8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ БАГАТОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ ТА КЛАСТЕРИЗАЦІЇ В КОРПОРАТИВНИХ СЕРЕДОВИЩАХ 10
1.1. Основні поняття та принципи багатофакторної автентифікації 10
1.2. Теоретичні основи роботи системи PrivacyIDEA 16
1.3. Аналіз альтернативних MFA-рішень та порівняльна характеристика.....23
1.4. Кластеризація в Linux 26
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ПІДГОТОВКА ТЕСТОВОГО СЕРЕДОВИЩА 33
2.1. Опис віртуального середовища та вибір програмного забезпечення.....33
2.2. Розробка схеми адресації 35
2.3. Базові налаштування серверів.....36
РОЗДІЛ 3. ВПРОВАДЖЕННЯ КЛАСТЕРИЗОВАНОЇ БАГАТОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ.....40
3.1. Підготовка серверного середовища та встановлення PrivacyIDEA 40
3.2. Інтеграція PrivacyIDEA з Active Directory.....41
3.3. Захист каналів взаємодії.....44
3.4. Налаштування багатофакторної автентифікації.....52
3.5. Налаштування кластеризації.....57
РОЗДІЛ 4. ТЕСТУВАННЯ ПРАЦЕЗДАТНОСТІ СИСТЕМИ.....68
4.1. Тестування працездатності кластера.....68
4.2. Тестування працездатності багатофакторної автентифікації.....72
ВИСНОВКИ.....77
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....78
ДОДАТКИ.....80

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AD – Active Directory

CSR – Certificate Signing Request

FOSS – Free and open-source software

HA – High Availability

HPC – High-performance computing

IP – Internet Protocol

LDAP – Lightweight Directory Access Protocol

MFA – Multi-Factor Authentication

NAT – Network Address Translation

SCIM – System for Cross-domain Identity Management

SCP – Secure Copy

SSH – Secure Shell

STONITH – Shoot The Other Node in The Head

TCP – Transmission Control Protocol

TLS – Transport Layer Security

U2F – Universal 2nd Factor

ВСТУП

Актуальність теми. Сучасні інформаційні системи та корпоративні мережі дедалі частіше стають об’єктами кібератак, спрямованих на компрометацію облікових записів користувачів і отримання несанкціонованого доступу до критичних ресурсів. Використання лише парольної автентифікації вже не забезпечує належного рівня захисту, оскільки паролі можуть бути викрадені внаслідок фішингових атак, витоків даних або соціальної інженерії. У зв’язку з цим виникає потреба у впровадженні багатофакторної автентифікації як одного з ключових механізмів підвищення рівня інформаційної безпеки.

Використання open source рішення PrivacyIDEA у поєднанні з Microsoft Active Directory дозволяє реалізувати централізовану систему MFA для доменної інфраструктури без необхідності використання дорогих комерційних продуктів. Додатково застосування технологій кластеризації забезпечує підвищення відмовостійкості та доступності сервісів автентифікації. Саме тому дослідження та впровадження системи багатофакторної автентифікації в середовищі Windows Server є актуальним і практично значущим завданням.

Метою кваліфікаційної роботи є розробка та впровадження системи багатофакторної автентифікації в середовищі Windows Server з використанням open source рішення PrivacyIDEA та технологій кластеризації для забезпечення підвищеного рівня інформаційної безпеки й відмовостійкості сервісів автентифікації.

Визначена мета обумовлює наступні завдання:

- провести аналіз сучасних загроз для систем автентифікації та методів їх нейтралізації;
- дослідити принципи роботи багатофакторної автентифікації та системи PrivacyIDEA;
- виконати порівняльний аналіз open source та комерційних MFA-рішень;
- дослідити принципи побудови кластерів високої доступності в Linux-системах;
- розробити тестове віртуальне середовище для впровадження MFA;

- реалізувати інтеграцію PrivacyIDEA з Microsoft Active Directory;
- налаштувати систему багатофакторної автентифікації для користувачів домену;
- виконати тестування працездатності та відмовостійкості реалізованої системи.

Об'єктом дослідження є процес забезпечення безпеки автентифікації користувачів у корпоративних інформаційних системах на базі Microsoft Active Directory.

Предметом дослідження є технології багатофакторної автентифікації, механізми інтеграції PrivacyIDEA з Microsoft Active Directory, а також засоби побудови кластерів високої доступності в Linux-системах для забезпечення безперервної роботи сервісів автентифікації.

У процесі виконання кваліфікаційної роботи було розроблено проєкт системи багатофакторної автентифікації для середовища Windows Server із використанням open source рішення PrivacyIDEA та інтеграцією з Microsoft Active Directory. Реалізоване рішення забезпечує централізовану MFA-автентифікацію користувачів домену та підвищення відмовостійкості сервісів за рахунок використання HA-кластеризації.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ БАГАТОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ ТА КЛАСТЕРИЗАЦІЇ В КОРПОРАТИВНИХ СЕРЕДОВИЩАХ

1.1. Основні поняття та принципи багатофакторної автентифікації

Багатофакторна автентифікація (Multi-Factor Authentication) – це метод контролю доступу, який вимагає від користувача надання двох або більше незалежних факторів автентифікації з різних категорій для підтвердження своєї особистості та отримання доступу до облікового запису, програми або системи [9].

Вона є одним із ключових механізмів забезпечення безпеки сучасних інформаційних систем. На відміну від традиційної однофакторної автентифікації, яка базується переважно на перевірці пароля, багатофакторна істотно ускладнює доступ для зловмисників, навіть якщо один із факторів – зазвичай пароль – було скомпрометовано. Сьогодні багатофакторна автентифікація є галузевим стандартом у сфері кіберзахисту та рекомендована більшістю міжнародних організацій, що формують політику інформаційної безпеки [10].

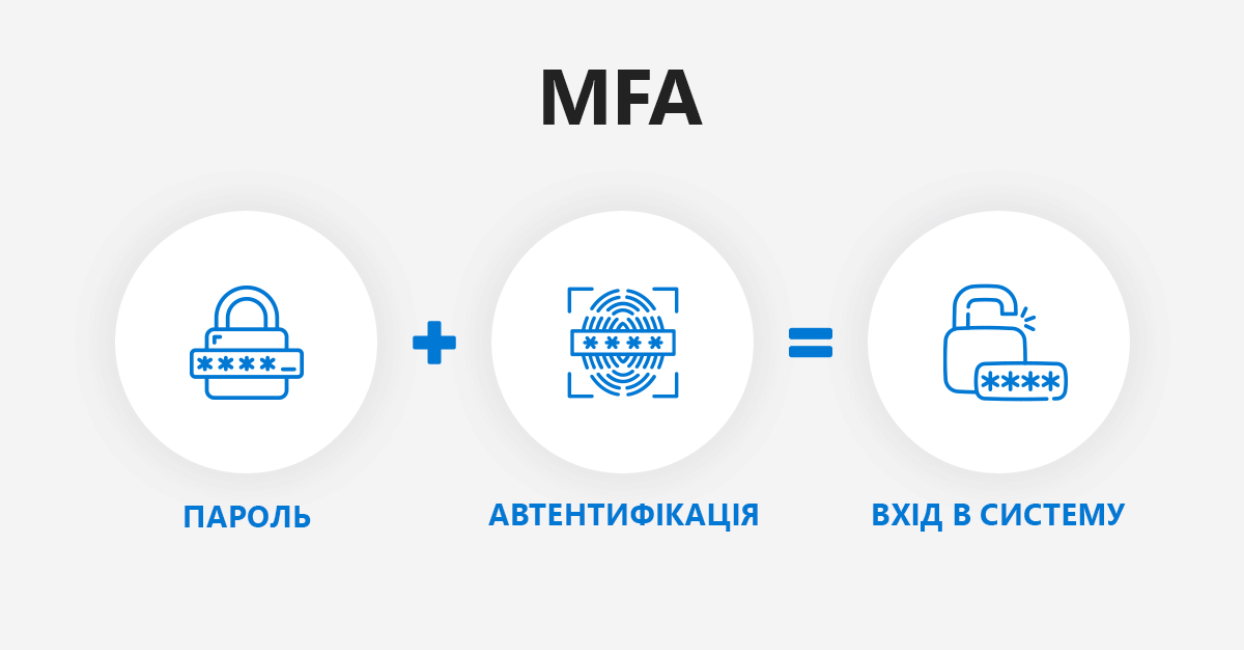


Рисунок 1.1 Базовий принцип багатофакторної автентифікації

Загалом виділяється три основні категорії факторів автентифікації:

- Фактор знання (Something you know) – пароль, PIN-код, відповідь на секретне запитання. Це найпоширеніший метод автентифікації, але й один із найбільш вразливих до фішингу та атак грубою силою;
- Фактор володіння (Something you have) – одноразовий код із мобільного додатка, TOTP-токен, смарт-картка, USB-ключ. Цей фактор забезпечує значно вищий рівень безпеки, оскільки для зламу потрібно фізично отримати пристрій;
- Фактор притаманності (Something you are) – відбиток пальця, розпізнавання обличчя, сітківка ока, голосовий аналіз. Біометричні дані важко підробити, тому вони вважаються одним із найбільш надійних видів факторів [9], [10].

Згідно зі статистичними даними, понад 80% компрометацій облікових записів пов'язані зі зламом або викраденням паролів. Це робить MFA не просто бажаним, а необхідним елементом сьогодення.

При розгляді багатофакторної автентифікації бажано вказати про основні ризики використання однофакторної. Однофакторна автентифікація, що базується на паролях, є найстарішим і найпоширенішим механізмом входу в інформаційні системи. Однак сучасні атаки дозволяють з високою ймовірністю обходити цей захист. Основні проблеми:

- Фішингові атаки – це кібершахрайство, де зловмисники маскуються під довірені організації через підроблені електронні листи, повідомлення чи сайти, щоб обманом виманити у користувачів конфіденційні дані: паролі, номери карток, фінансову інформацію [9];
- Брутфорс та словникові атаки – це метод зламу паролів шляхом систематичного перебору усіх можливих комбінацій. Словникова атака – це різновид брутфорсу, що використовує обмежений список найпопулярніших, часто вживаних або «слабких» паролів для швидшого підбору, і є ефективнішою проти недбалих користувачів, які використовують прості паролі;

• Атаки через повторно використані паролі – це тип комбінаційних атак, за яких зловмисник, отримавши доступ до автентифікаційних даних користувача на одному сервісі, намагається використати ці самі облікові дані для доступу до інших ресурсів. Оскільки значна кількість користувачів застосовує однакові або схожі паролі на різних платформах, компрометація одного ресурсу створює ризик ланцюгового проникнення до інших систем;

• Людський фактор – паролі можуть бути забуті, записані на папері, передані колегам – всі ці фактори знижують рівень безпеки.

Залежно від вимог до безпеки та особливостей інфраструктури, MFA може реалізовуватися різними способами. Розглянемо основні механізми реалізації більш детально.

Одноразові паролі (OTP) – це пароль, який є дійсним тільки для одного сеансу автентифікації. Його дія також може бути обмежена певним проміжком часу. Перевага такого паролю порівняно зі статичним полягає в тому, що його неможливо використовувати повторно. Для використання такого типу додаткової автентифікації може бути використана пошта, SMS або спеціалізований додаток. Додаток-автентифікатор (Authenticator application) – це мобільний додаток, що забезпечує додатковий рівень безпеки для ваших онлайн-акаунтів, генеруючи одноразові паролі для двофакторної автентифікації [2].

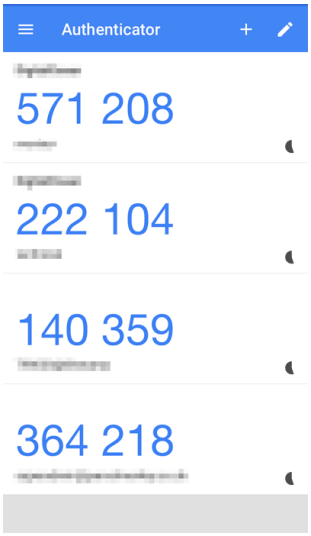


Рисунок 1.2 Приклад налаштованого додатку-автентифікатора

Апаратна автентифікація – це процес підтвердження особи користувача за допомогою фізичних пристроїв або унікальних біологічних характеристик.

• FIDO U2F – це відкритий стандарт двофакторної автентифікації від FIDO Alliance, який використовує апаратні ключі, як сильний, фішинг-стійкий другий фактор для безпечного входу [9];



Рисунок 1.3 Приклад пристрою FIDO U2F

• FIDO2/WebAuthn – це сучасні стандарти для безпечної автентифікації без паролів. FIDO2 – загальний набір специфікацій для усунення паролів, що складається з двох частин: WebAuthn для браузерів та сайтів, який дозволяє використовувати біометрію або апаратні ключі, та CTAP для зв'язку між пристроєм і автентифікатором [9];

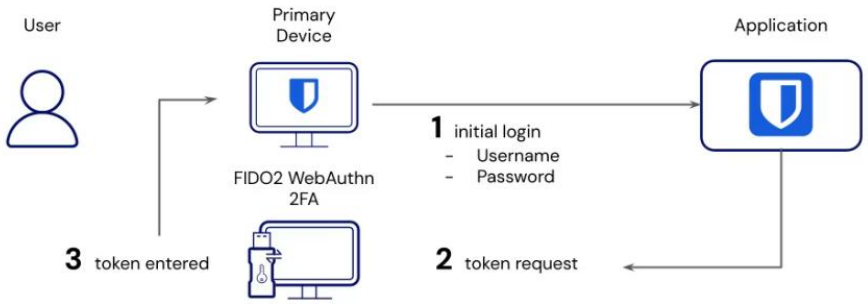


Рисунок 1.4 Принцип роботи FIDO2/WebAuthn

• PKI-смарт-картки – це пластикові картки з мікропроцесором, які використовуються для безпечного зберігання цифрових сертифікатів, приватних ключів та виконання криптографічних операцій, забезпечуючи надійну автентифікацію, підпис та шифрування в інфраструктурах відкритих ключів [5], [7].



Рисунок 1.5 Приклад PKI-смарт-картки

Push-автентифікація – це зручний метод автентифікації, коли замість ручного введення коду надходить сповіщення на смартфон із запитом підтвердити або відхилити вхід до сервісу в один клік, що робить процес швидшим та безпечнішим, ніж введення одноразових паролів.

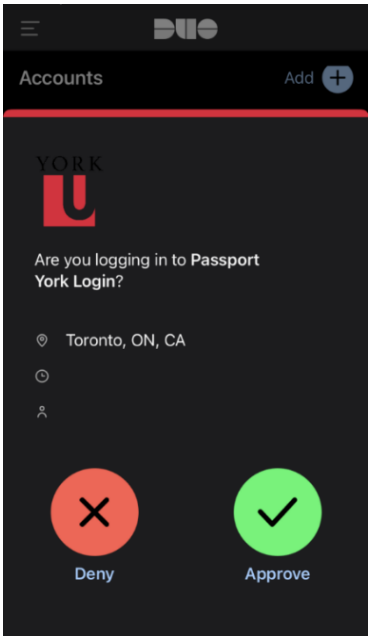


Рисунок 1.6 Приклад налаштованої push-автентифікації

Попри те, що багатофакторна автентифікація значно підвищує рівень безпеки, жодна її форма не є повністю невразливою. Зловмисники активно адаптують свої методи, спрямовані на обходження додаткових факторів захисту, використовуючи як соціальну інженерію, так і технічні рішення. Розглянемо найбільш поширені та успішні типи атак.

- Атаки соціальної інженерії на MFA – це тип атак, в якому зловмисники маніпулюють користувачем з метою викрасти логін/пароль або змусити жертву підтвердити шахрайський запит на вхід;
- Атака втоми MFA (MFA Fatigue Attack) – це метод, коли хакери переповнюють користувача багатьма запитами на авторизацію (push-повідомленнями), щоб він випадково або навмисно схвалив вхід, намагаючись зупинити спам;
- Телефонний фішинг (Voice phishing, vishing) – вид шахрайства, коли зловмисник телефонує жертві, видаючи себе за довірену особу, щоб обманом вмовити людину продиктувати одноразовий код, схвалити push-запит, встановити вірусну програму під виглядом корисного додатку чи перейти на фішингову сторінку.
- Перехоплення SMS (SS7 атаки) – це тип кібератак, які використовують критичні вразливості в застарілому протоколі Signaling System No. 7, що є основою для обміну службовою інформацією між телекомунікаційними мережами по всьому світу.
- SIM-Swapping – це вид кібератаки, при якому шахраї викрадають мобільний номер телефону жертви, переносячи його на власну SIM-картку.

Отже, незважаючи на те, що MFA є стандартизованим в сучасному світі, він все ще не гарантує повної безпеки облікових даних. Існує безліч кіберзагроз, окрім описаних вище. Багато з цих атак спрямовані не на прямі недоліки MFA, а на людські фактори, що підкреслює необхідність поєднання MFA з додатковими механізмами безпеки [9], [10].

1.2. Теоретичні основи роботи системи PrivacyIDEA

PrivacyIDEA – це відкритий сервер багатофакторної автентифікації, створений для того, щоб зробити MFA гнучким, масштабованим і повністю контрольованим всередині організації. Він не замінює існуючі каталоги користувачів, а працює поверх них, додаючи шар керування токенами, правилами та політиками доступу [1].

Основна концепція полягає в тому, що PrivacyIDEA виступає як централізований сервіс: він зберігає й керує токенами, приймає запити на автентифікацію через REST API та застосовує налаштовані політики, які визначають, чи буде автентифікація дозволена в конкретному випадку.

Архітектурно PrivacyIDEA це – веб-сервіс оснований на мові програмування Python з фреймворком Flask та SQL-бази даних для зберігання інформації про користувачів, токени, політики, що робить її гнучкою для інтеграції з існуючими системами без потреби повної перебудови інфраструктури [1].

Архітектура складається з декількох логічних шарів:

- Веб-інтерфейс (WebUI) – Це фронтенд, через який адміністратори та користувачі взаємодіють із системою: адміністратори керують реєстрацією, токенами та політиками; Користувачі можуть самостійно керувати своїми токенами. Він побудований за сучасними принципами та може включати дашборд, налаштування політик, журналювання та перегляд статистики [1].

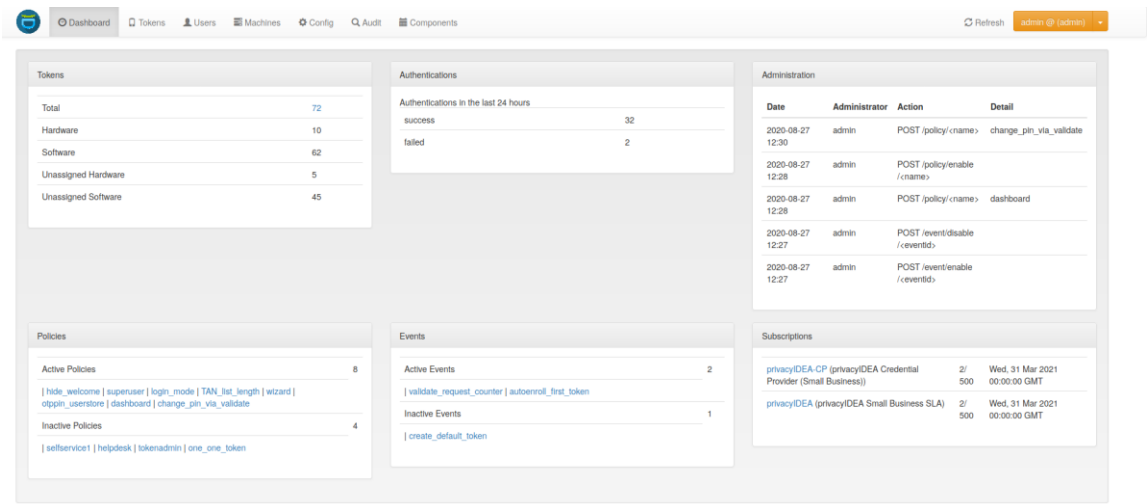


Рисунок 1.7 Вигляд dashboard вікна

• REST API шар – це основний спосіб взаємодії зовнішніх додатків та сервісів з PrivacyIDEA. Вся функціональність, від автентифікації до управління токенами, доступна через API [1].

API підтримує авторизацію через JWT-токени та дозволяє: отримувати інформацію про токени, виконувати адміністрування сервера, реєструвати нові пристрої та валідувати фактори автентифікації. Він є критично важливою частиною архітектури, оскільки саме він дозволяє інтеграцію PrivacyIDEA з іншими сервісами за допомогою спеціальних плагінів та модулів.

• Базова логіка – де відбувається весь основний функціонал: політики, реєстрація та управління токенами, валідатори. Базова логіка читає правила політик і приймає рішення: дозволити автентифікацію, відмовити або запросити додатковий фактор [1].

• Доступ до даних (Data Access Layer) – Цей шар відповідає за збереження, шифрування та доступ до даних: всі токени, журнали подій та конфігурації зберігаються в SQL-сховищі. SQLAlchemy застосовується для об'єктно-реляційного відображення (ORM), що спрощує модульне тестування і гнучкість роботи з даними. У базі інформація про токени шифрується за допомогою AES, що підсилює захист секретних даних [1].

Оскільки API організований у великі групи ендпоінтів з відповідним призначенням, де кожна група – це окрема частина функціоналу системи, розглянемо основні з них.

Ендпоінти автентифікації (Authentication endpoints) – відповідають за отримання JWT-токена для доступу до інших частин API. Після успішної автентифікації подальші запити повинні містити заголовок з отриманим токеном, що забезпечує контроль доступу до ресурсів та операцій всередині системи.

Ендпоінти валідації (Validate endpoints) – відповідають за безпосередню перевірку MFA-значень. Вони обробляють введені користувачем одноразові паролі OTP/HOTP/TOTP, ініціалізують механізм challenge-response для push, або WebAuthn-токенів, а також підтримують офлайн-токени, що працюють без мережі. Кожен запит до цих ендпоінтів передається до Core Logic, де застосовуються

політики безпеки, перевіряються права доступу та ведеться аудит подій. Таким чином, ендпоінти валідації забезпечують захищений та контрольований процес перевірки другого фактора [1], [2].

Системні ендпоінти (System endpoints) – призначені для керування конфігурацією сервера та його загальними параметрами. Вони дозволяють додавати або змінювати налаштування, встановлювати параметри за замовчуванням для нових реєстрацій, управляти глобальними принципами роботи системи. Доступ до цих ендпоінтів потребує адміністративних прав, оскільки зміни конфігурації можуть впливати на стабільність та безпеку всієї системи [1].

Resolver та Realm endpoints – є ключовими для отримання та обробки зовнішніх даних про користувачів. Resolver endpoints використовують механізми доступу до джерел користувачів, таких як LDAP [8], Active Directory [4], SQL або SCIM, забезпечуючи централізоване підключення до корпоративних каталогів. Realm endpoints дозволяють створювати, переглядати, редагувати та видаляти реалми – логічні групи користувачів, об'єднані за певними джерелами. Для кожного реалму можна застосовувати окремі політики безпеки, що робить систему гнучкою та масштабованою для великих організацій із розділеною структурою [1].

Користувацькі та токен ендпоінти (Token, User endpoints) – відповідають за повний цикл керування MFA-токенами. Вони дозволяють реєструвати нові токени, активувати або деактивувати вже існуючі, оновлювати конфігурацію токенів, а також видаляти токени. Крім того, через ці ендпоінти можна отримати список користувачів, здійснювати пошук по реалмам та переглядати дані про токени відповідних користувачів, що забезпечує централізоване управління життєвим циклом кожного токена.

Ендпоінти політики (Policy endpoints) – дозволяють створювати та налаштовувати правила, які визначають, як саме відбувається багатофакторна автентифікація, які токени допустимі для яких дій та хто має право виконувати ті чи інші операції. Політики можуть застосовуватися до конкретних груп користувачів, реалмів або клієнтських додатків. Вони забезпечують гнучкість

системи та дозволяють точно налаштовувати MFA під конкретні потреби організації.

Ендпоінти журналювання (Audit endpoints) – забезпечують ведення аудиту та детальне логування всіх подій. Через ці ендпоінти можна отримувати записи логів, фільтрувати їх за типами подій – успішні та неуспішні автентифікації, зміни політик, операції з токенами та дії адміністраторів – а також статистику подій. Це ключовий механізм для аналізу безпеки, моніторингу та розслідування інцидентів.

PrivacyIDEA підтримує широкий спектр типів токенів, що дозволяє підлаштуватися під абсолютно різні сценарії використання. Основні типи наведені в таблиці 1.1 [1], [2], [9].

Таблиця 1.1

Основні типи токенів PrivacyIDEA

Тип токена	Опис	Механізм роботи	Приклади використання
TOTP	Одноразові паролі, що генеруються на основі часу	Користувач або апаратний/мобільний токен генерує пароль, дійсний протягом короткого часу (зазвичай 30 сек)	Google Authenticator, Authy
HOTP	Одноразові паролі, що генеруються на основі лічильника	Пароль генерується на основі лічильника, що збільшується з кожним використанням	Деякі апаратні токени YubiKey, RSA SecurID
Push-токени	Підтвердження через мобільний додаток	Сервер надсилає push-повідомлення на смартфон, користувач підтверджує або відхиляє запит	PrivacyIDEA Authenticator
WebAuthn / FIDO2	Криптографічна автентифікація без пароля або як другий фактор	Використання ключів або біометрії для створення криптографічного підпису	Біометричні сенсори на ноутбуках, YubiKey, FIDO2 ключі

Таблиця 1.1

Продовження

Тип токена	Опис	Механізм роботи	Приклади використання
U2F	Апаратний ключ для другого фактора	Підключення через USB/NFC/Bluetooth, генерує криптографічний підпис	YubiKey, Solo-Keys
SMS-токени	Одноразові паролі, надіслані SMS	Сервер генерує OTP та надсилає його на номер телефону користувача	Додатковий фактор для банків або VPN
Email-токени	Одноразові паролі, надіслані електронною поштою	Сервер генерує OTP і надсилає на email	Додатковий фактор для корпоративних сервісів
X.509 / PKI сертифікати	Криптографічна автентифікація	Використання сертифікатів для підтвердження особи	Доступ до серверів, VPN, SSH
SSH ключі	Автентифікація через публічно-приватний ключ	Генерація криптографічного підпису для входу на сервер	Віддалені сервери Linux
Offline токени	Генерація OTP без підключення до мережі	Користувач генерує пароль локально, сервер перевіряє відповідно до алгоритму	Апаратні токени, мобільні додатки

Архітектура безпеки PrivacyIDEA реалізована згідно з принципом ешелонованого захисту, де критичні дані відокремлені від логіки їх обробки. Фундаментальним елементом системи є захист секретних ключів, які використовуються для генерації одноразових паролів. Оскільки ці дані необхідні системі для верифікації, вони не можуть бути хешовані безповоротно, тому застосовується симетричне шифрування за стандартом AES-256. Критично важливою особливістю є розділення місця зберігання зашифрованих даних та ключів шифрування. Зашифровані секрети токенів, паролі до зовнішніх каталогів

та ключі API зберігаються у базі даних, тоді як майстер-ключ шифрування розміщується окремо у конфігураційних файлах або в окремому захищеному сховищі для підвищеного рівня безпеки. Такий підхід гарантує, що компрометація дампа бази даних не призведе до витоку автентифікаційних секретів.

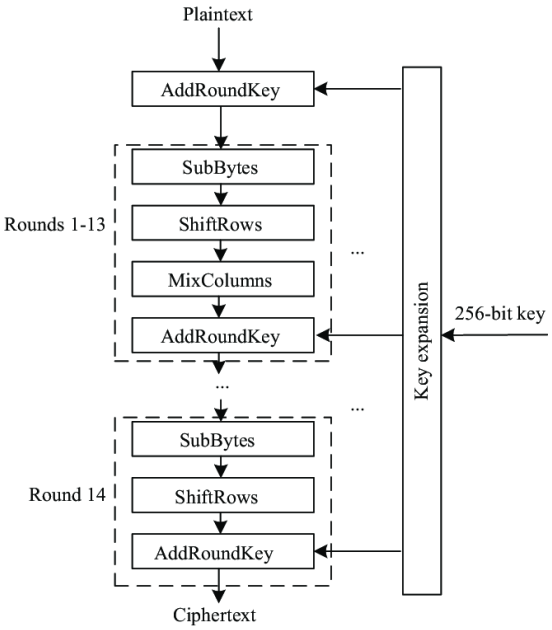


Рисунок 1.8 Загальна схема AES-256

Для захисту транспортного рівня застосовується протокол TLS версії 1.2 або 1.3, залежно від конфігурації веб-сервера, який використовується як веб-інтерфейс для PrivacyIDEA. Це забезпечує шифрування переданих даних, автентифікацію сервера за допомогою X.509 сертифікатів та захист від атак типу Man-in-the-Middle [3], [7].

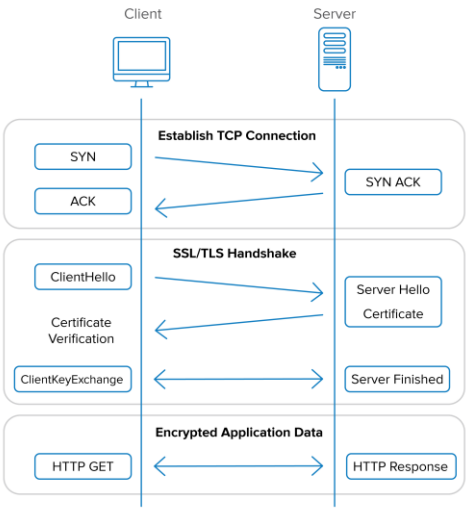


Рисунок 1.9 Принцип роботи TLS

На прикладному рівні реалізовано контроль сесій та обмеження кількості спроб автентифікації. PrivacyIDEA дозволяє налаштовувати політики, які визначають максимальну кількість помилкових спроб введення OTP, часові затримки між спробами та поведінку системи у випадку перевищення лімітів. Це є ефективним захистом від атак грубою силою (brute-force) та автоматизованих скриптів підбору одноразових паролів [2], [9].

Не менш важливим є питання ведення аудиту подій. PrivacyIDEA містить розвинену систему журналювання, яка є критично важливою для аналізу безпеки та відповідності вимогам аудиту. Усі значущі події фіксуються в аудит-логах, включно зі спробами автентифікації, операціями з токенами, змінами політик та адміністративними діями.

Журнали можуть зберігатися у локальній базі даних або передаватися до зовнішніх систем логування та моніторингу. Окрему увагу приділено збору статистики використання токенів. Система дозволяє аналізувати частоту використання конкретних типів токенів, кількість активних і заблокованих токенів, а також загальну динаміку автентифікаційних подій. Такі дані можуть бути використані для виявлення аномалій, наприклад, різкого зростання кількості невдалих спроб входу, що може свідчити про атаку.

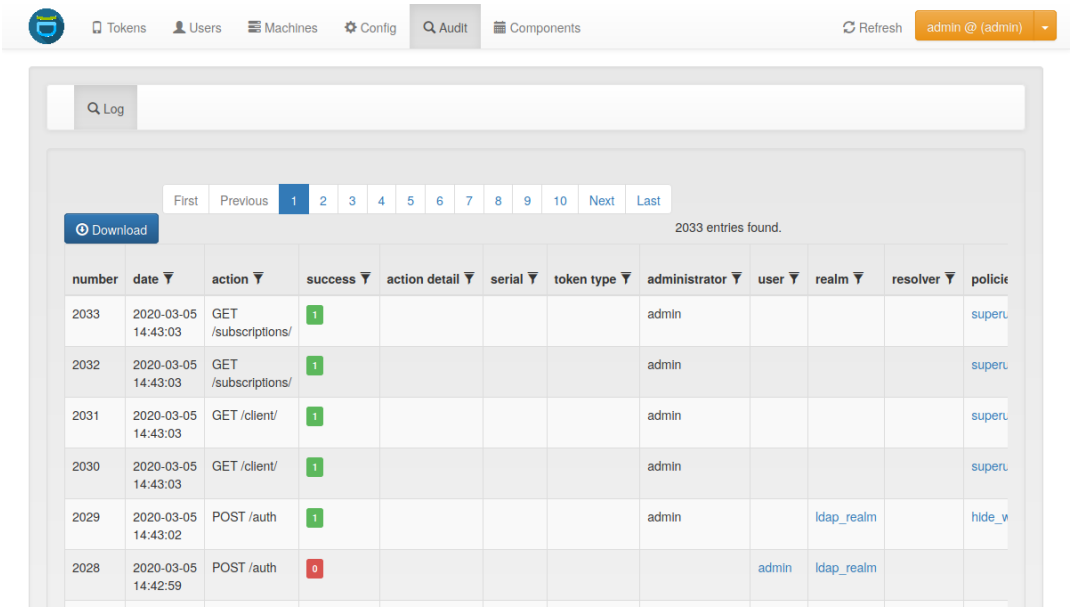


Рисунок 1.10 Вигляд сторінки аудиту

Таким чином, PrivacyIDEA може розглядатися як надійна основа для побудови сучасної системи автентифікації. Його функціонал та гнучкість забезпечують можливість тонкого налаштування політик доступу, управління користувачами та токенами, а завдяки модульній архітектурі, використанню стандартних веб-технологій, REST API та підтримці широкого спектра токенів, система добре масштабується й адаптується до різних сценаріїв [1].

1.3. Аналіз альтернативних MFA-рішень та порівняльна характеристика

Вибір програмного забезпечення для реалізації багатофакторної автентифікації потребує порівняльного аналізу наявних рішень з урахуванням функціональних можливостей, архітектурних особливостей та умов експлуатації. У зв'язку з цим було розглянуто альтернативні MFA-рішення та наведено їх порівняльну характеристику [9].

Microsoft Azure MFA – це хмарний сервіс багатофакторної автентифікації, що входить до складу Microsoft Entra ID. Рішення орієнтоване на централізоване керування ідентичностями в екосистемі Microsoft та широко використовується в корпоративних середовищах.

Azure MFA базується на хмарній інфраструктурі та передбачає передачу запитів автентифікації до сервісів Azure через захищені канали зв'язку з використанням TLS. Підтримується широкий спектр методів MFA: push-повідомлення через Microsoft Authenticator, TOTP, SMS, голосові виклики, а також апаратні ключі стандарту FIDO2 [2], [9].

Його основною перевагою є глибока інтеграція з Windows Server, Active Directory, Azure AD, Microsoft 365 та іншими сервісами Microsoft. Однак залежність від хмарної інфраструктури та умови ліцензування обмежують контроль над внутрішніми механізмами роботи, що робить це рішення менш привабливим для організацій, які вимагають повного on-premise розгортання та прозорості механізмів безпеки.

Розглядаючи інші комерційні рішення слід відзначити Cisco Duo, який також широко застосовується в гетерогенних корпоративних середовищах.

Cisco Duo – це комерційна хмарна платформа MFA, орієнтована на швидке впровадження та простоту використання. Вона активно застосовується для захисту VPN-доступу, віддаленого робочого столу, веб-додатків та корпоративних сервісів.

Архітектура Duo передбачає використання агентів, які встановлюються на сервери або шлюзи доступу. Вони передають запити автентифікації до хмарного сервісу Cisco Duo, де виконується перевірка другого фактора. Підтримуються push-повідомлення, TOTP, SMS, голосові дзвінки та апаратні токени.

Сильним боком Duo є зручність адміністрування та добре продуманий веб-інтерфейс. Водночас, як і у випадку з Azure MFA, рішення залежне від хмарної інфраструктури, що обмежує контроль над даними автентифікації та ускладнює використання в середовищах із підвищеними вимогами до ізоляції.

Серед open source рішень було розглянуто LinOTP.

LinOTP – це відкритий сервер багатофакторної автентифікації, орієнтований на локальне розгортання та повний контроль над даними користувачів. Він активно застосовується для захисту VPN-доступу, веб-додатків, серверів та корпоративних сервісів.

За структурою він схожий на PrivacyIDEA, але є менш модульним та більш орієнтованим саме на середні корпоративні середовища. Передбачає центральний сервер MFA, який інтегрується з корпоративними каталогами користувачів та забезпечує управління токенами й політиками автентифікації. Сервер обробляє запити автентифікації через REST API або спеціальні агенти, підтримуючи TOTP, HOTP, push-повідомлення через мобільний застосунок LinOTP Authenticator, SMS/Email-токени та апаратні токени [2], [9].

Під час аналізу було складено порівняльну таблицю підтримки тих чи інших технологій та ключових функціональних аспектів MFA-рішень [1], [9].

Таблиця 1.2

Порівняльний аналіз MFA-рішень

Критерій	Azure MFA	Cisco Duo Security	LinOTP	PrivacyIDEA
Тип рішення	Комерційне, хмарне	Комерційне, хмарне/гібридне	FOSS, локальне	FOSS, локальне
Локальне розгортання	Ні	Ні	Так	Так
Open Source	Ні	Ні	Так	Так
Підтримка TLS/HTTPS	Так	Так	Так	Так
Інтеграція з Active Directory	Так	Так	Так	Так
LDAP/AD/SQL каталоги	Так	Так	Так	Так
REST API	Так	Так	Так	Так
TOTP (OTP)	Так	Так	Так	Так
HOTP (OTP)	Так	Так	Так	Так
Push-сповіщення	Так	Так	Так	Так
SMS-токени	Так	Так	Так	Так
Email-токени	Так	Так	Так	Так
WebAuthn / FIDO2	Так	Так(обмежено)	Ні(обмежено)	Так
U2F (апаратні ключі)	Так	Так	Так	Так
Апаратні токени (HOTP)	Так	Так	Так	Так
Апаратні USB/Bluetooth ключі (FIDO2)	Так	Так	Ні	Так
Інтеграція з VPN (RADIUS/NPS)	Так	Так	Так	Так
SSO / IAM функції	Так	Так(обмежено)	Ні	Ні
Гнучкі політики доступу	Так	Так	Ні(базові)	Так
Аудит / логування	Так	Так	Так	Так

Таблиця 1.2

Продовження

Критерій	Azure MFA	Cisco Duo Security	LinOTP	PrivacyIDEA
Масштабованість для великих організацій	Так	Так	Так	Так
Залежність від постачальника хмари	Висока	Висока	Ні	Ні
Вартість впровадження	Висока	Висока	Низька	Низька
Контроль над даними	Обмежений	Обмежений	Повний	Повний

Враховуючи проведений аналіз, було обрано некомерційне рішення PrivacyIDEA. На відміну від LinOTP у тому ж сегменті, PrivacyIDEA має більш модульну архітектуру, гнучкішу систему політик та кращу можливість інтеграції через REST API. Ці фактори роблять PrivacyIDEA оптимальним вибором для впровадження MFA у досліджуваному середовищі.

1.4. Кластеризація в Linux

Кластеризація в Linux – це спосіб об'єднання декількох серверів у єдину систему, яка працює як один логічний комплекс. Такий підхід використовується для забезпечення високої доступності сервісів, масштабування навантаження, підвищення продуктивності та мінімізації простоїв у роботі систем. У кластері кожен сервер, який називається вузлом, взаємодіє з іншими вузлами через мережу та бере участь у виконанні спільних завдань. Якщо один із вузлів виходить з ладу, інші вузли можуть автоматично взяти на себе його функції, завдяки чому робота сервісів не припиняється. Саме тому кластеризація широко використовується у датацентрах, банківських системах, телекомунікаціях, хмарних платформах, системах віртуалізації та інших критично важливих інфраструктурах, де навіть короткочасний простій є небажаним або неприпустимим.

У Linux існує декілька напрямків кластеризації, які відрізняються своїм призначенням. Найпоширенішими є кластери високої доступності (High Availability, HA), кластери балансування навантаження та HPC-кластери, орієнтовані на високопродуктивні обчислення [14].

HA-кластери створюються для забезпечення безперервної роботи сервісів. У такій конфігурації один вузол може бути активним, а інший – резервним. У разі відмови активного вузла кластер автоматично переносить сервіси на резервний сервер. Цей механізм називається failover. Основна ідея полягає в тому, щоб користувачі або прикладні системи не помітили збою. Для цього кластер постійно контролює стан вузлів та сервісів, а також виконує автоматичний перезапуск ресурсів у разі помилок [11].

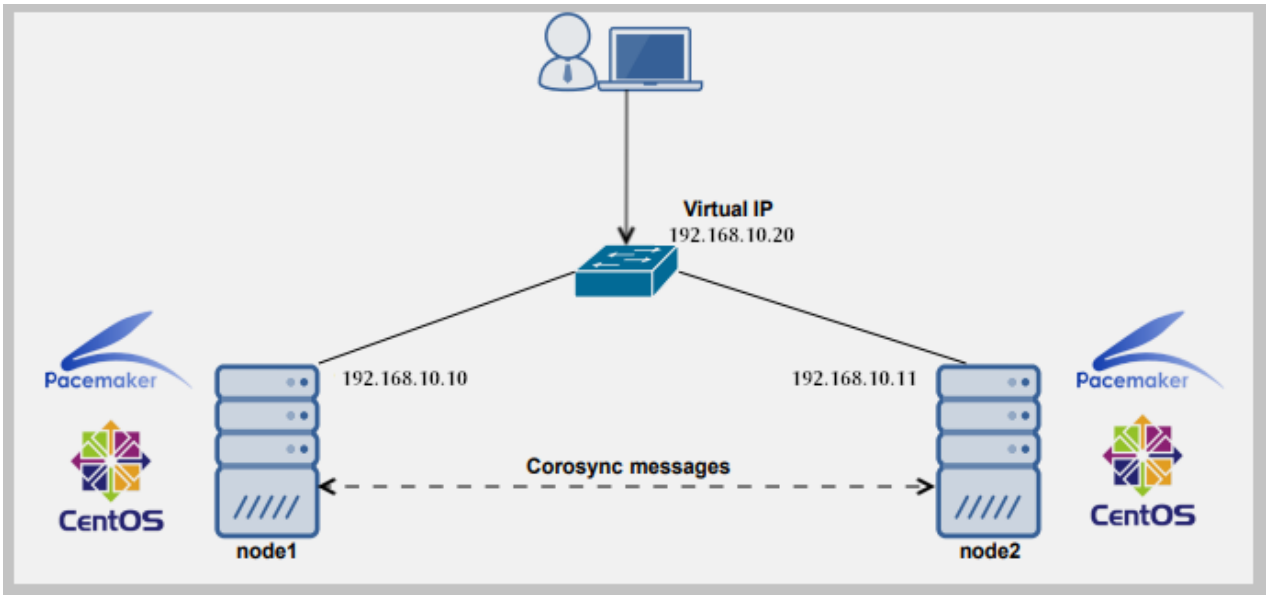


Рисунок 1.11 Схема реалізації HA-кластера

Кластери балансування навантаження включають декілька серверів, які одночасно обробляють запити користувачів, а спеціальні механізми рівномірно розподіляють навантаження між вузлами. Такий підхід дозволяє суттєво підвищити продуктивність системи та уникнути перевантаження окремих серверів. Подібні рішення використовуються у веб-хостингу, CDN-мережах, великих веб-додатках і хмарних сервісах. У Linux для цього часто використовуються HAProxy, Nginx або Linux Virtual Server (LVS) [14].

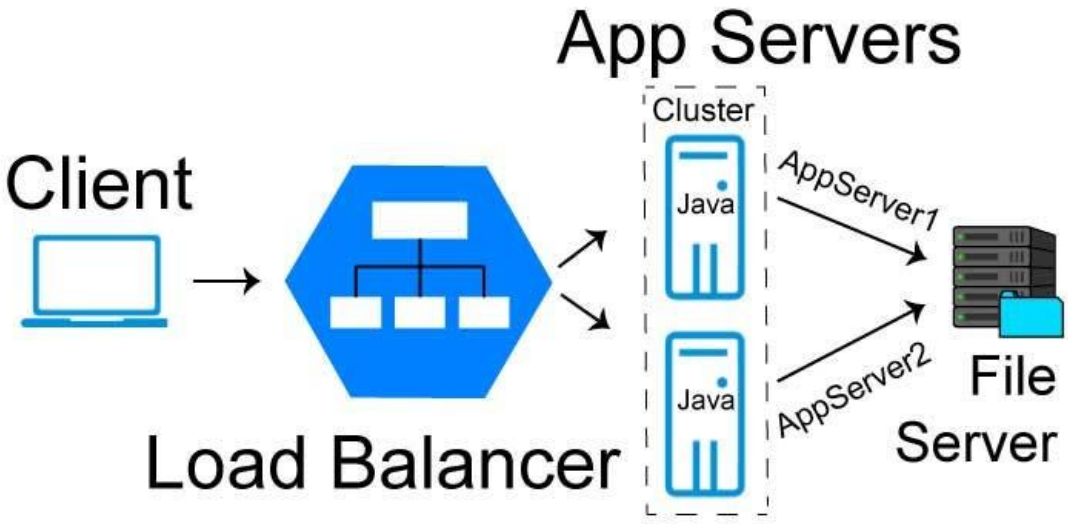


Рисунок 1.12 Принцип роботи кластера балансування навантаження

HPC-кластери (High Performance Computing), у свою чергу, орієнтовані не на відмовостійкість, а на максимальну продуктивність. У таких системах велика кількість вузлів одночасно виконує частини однієї складної задачі. HPC-кластери активно застосовуються у наукових дослідженнях, моделюванні фізичних процесів, криптографії, машинному навчанні та аналізі великих обсягів даних. На відміну від HA-кластерів, тут основний акцент робиться саме на обчислювальній потужності та паралельному виконанні задач [14].

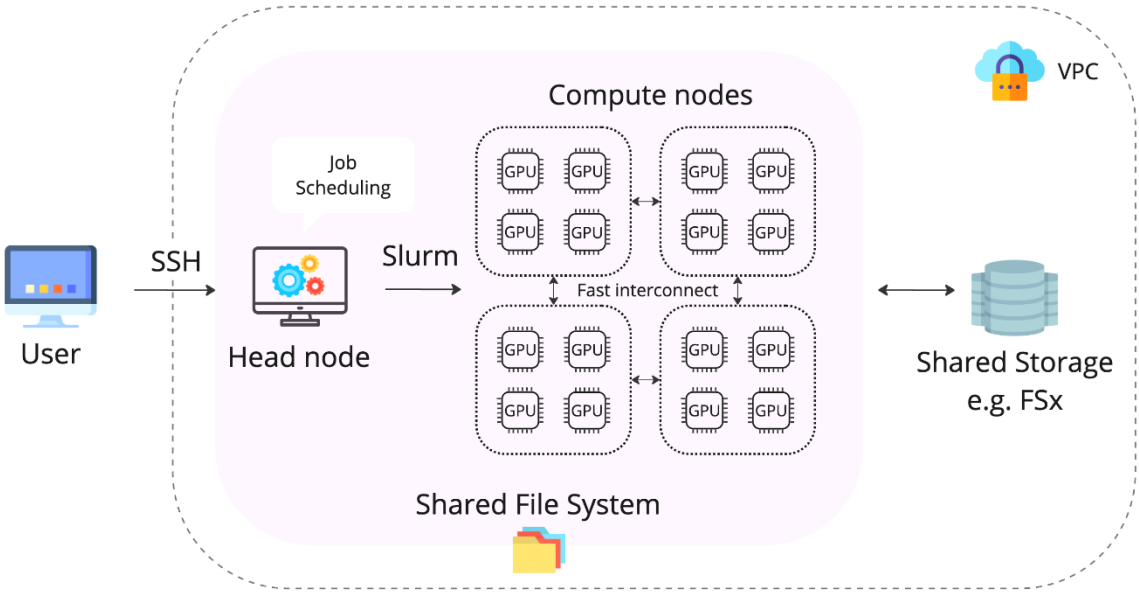


Рисунок 1.13 Принцип роботи HPC-кластера

Архітектура Linux-кластера складається з декількох взаємопов'язаних компонентів. Основою є вузли кластера – фізичні сервери або віртуальні машини, між якими відбувається обмін повідомленнями через спеціальну мережу. Для організації комунікації між вузлами часто використовується Corosync. Його завдання полягає у передачі heartbeat-повідомлень, визначенні членства вузлів у кластері та контролі кворуму. Heartbeat використовується для перевірки доступності вузлів: якщо сервер перестає відповідати, система вважає його несправним [11].

Над Corosync зазвичай працює Pacemaker, що є менеджером ресурсів кластера. Він приймає рішення про запуск, зупинку або перенесення сервісів між вузлами. Pacemaker аналізує стан ресурсів, визначає пріоритети та реагує на різноманітні події, наприклад на втрату вузла або збій сервісу. Якщо сервер перестає відповідати, Pacemaker може автоматично перенести ресурси на інший вузол. Для управління сервісами використовуються так звані Resource Agents – спеціальні скрипти або модулі, які відповідають за запуск, зупинку та моніторинг конкретних ресурсів, наприклад баз даних, IP-адрес, файлових систем або веб-серверів [11].

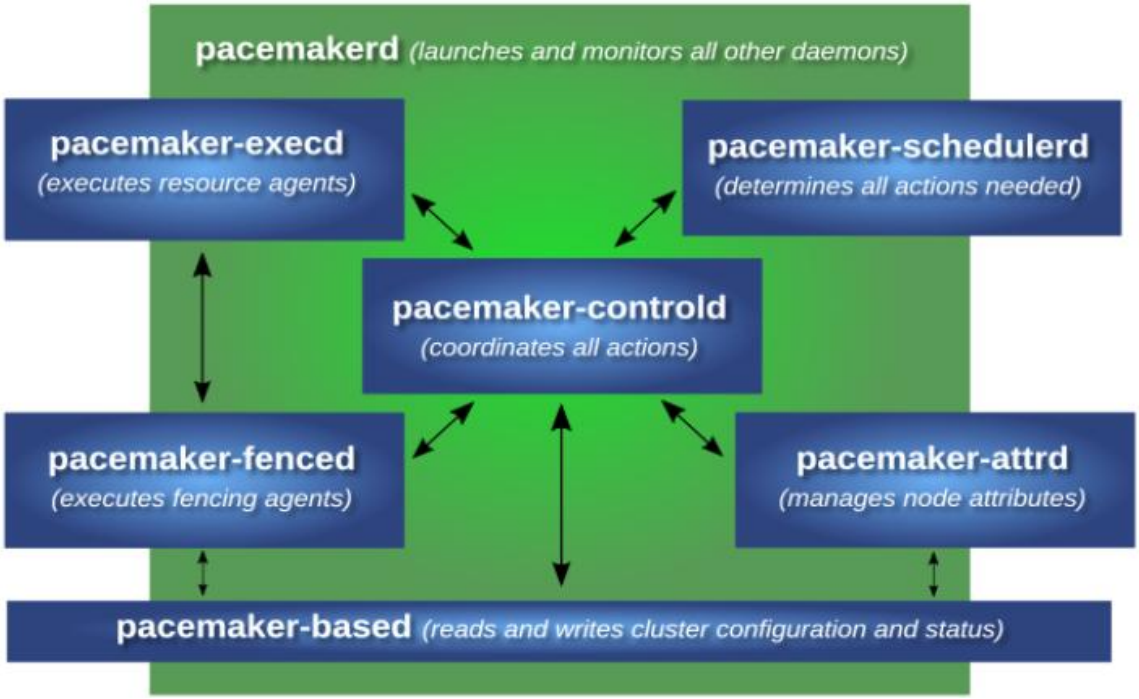


Рисунок 1.14 Архітектура Pacemaker

Кворум є однією з найважливіших концепцій у Linux-кластерах. Його завдання полягає у запобіганні split-brain ситуаціям. Split-brain – це стан, коли один кластер або розподілена система розділяється на дві частини, які не можуть обмінюватися повідомленнями, але кожна частина вважає себе «головною» і продовжує працювати самостійно. У результаті декілька серверів можуть одночасно працювати з одними й тими самими даними, що часто призводить до пошкодження файлових систем або баз даних. Щоб уникнути цього, кластер повинен мати більшість голосів, тобто кворум. Якщо достатня кількість вузлів недоступна, система може автоматично зупинити сервіси для захисту даних.

Для додаткового захисту використовується fencing або STONITH (Shoot The Other Node In The Head). Це механізм примусового відключення несправного вузла від спільних ресурсів. Наприклад, сервер може бути автоматично перезавантажений через IPMI або повністю відключений від живлення. Основна мета fencing – гарантувати, що несправний вузол більше не має доступу до спільних даних. У Pacemaker fencing вважається критично важливим елементом безпечної роботи кластера, оскільки без нього неможливо гарантувати відсутність split-brain [12].

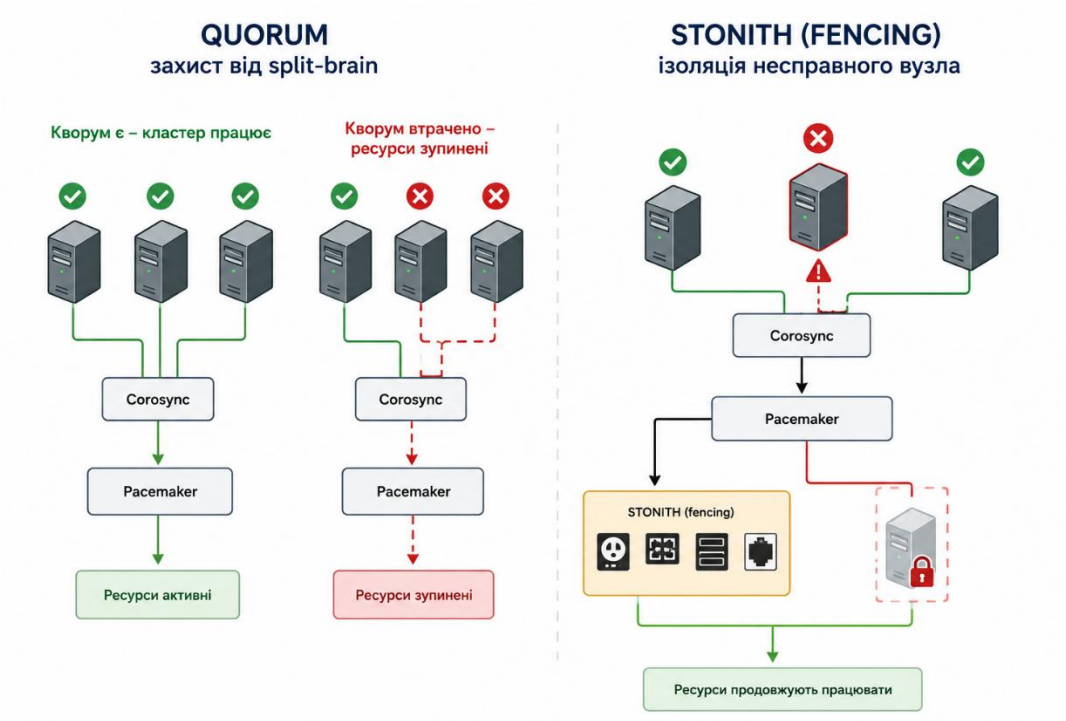


Рисунок 1.15 Принцип роботи Quorum та Fencing

У Linux-кластерах можуть використовуватися різні схеми резервування. Найпростішою є active/passive, де один вузол працює постійно, а інший перебуває в резерві. Існує також схема active/active, у якій декілька вузлів одночасно обробляють навантаження. У разі відмови одного сервера його завдання автоматично розподіляються між іншими вузлами. Для великих інфраструктур можуть застосовуватися складніші моделі, наприклад N+1 або N-to-N, де резервування відбувається одразу для декількох серверів [14].

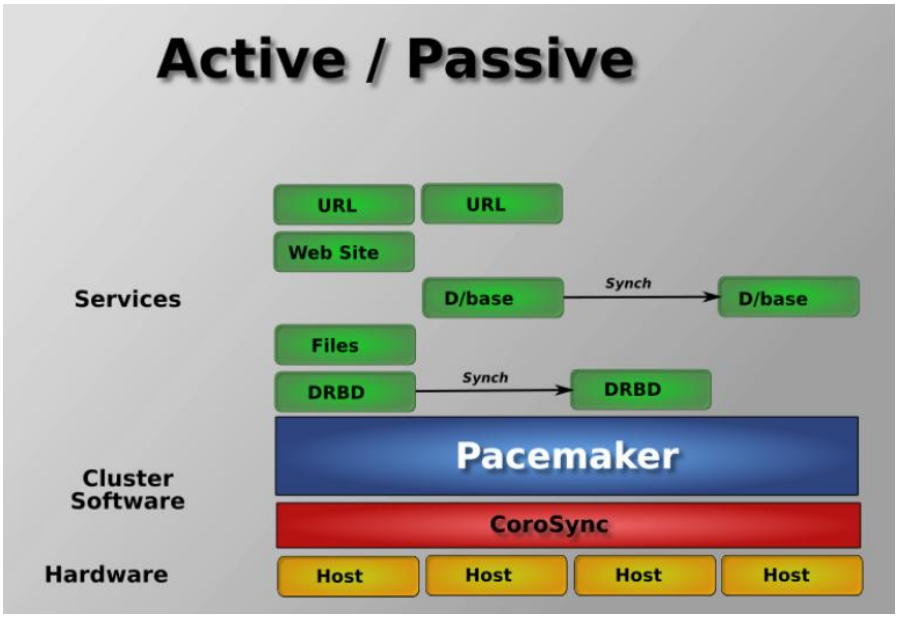


Рисунок 1.16 Схема роботи active/passive

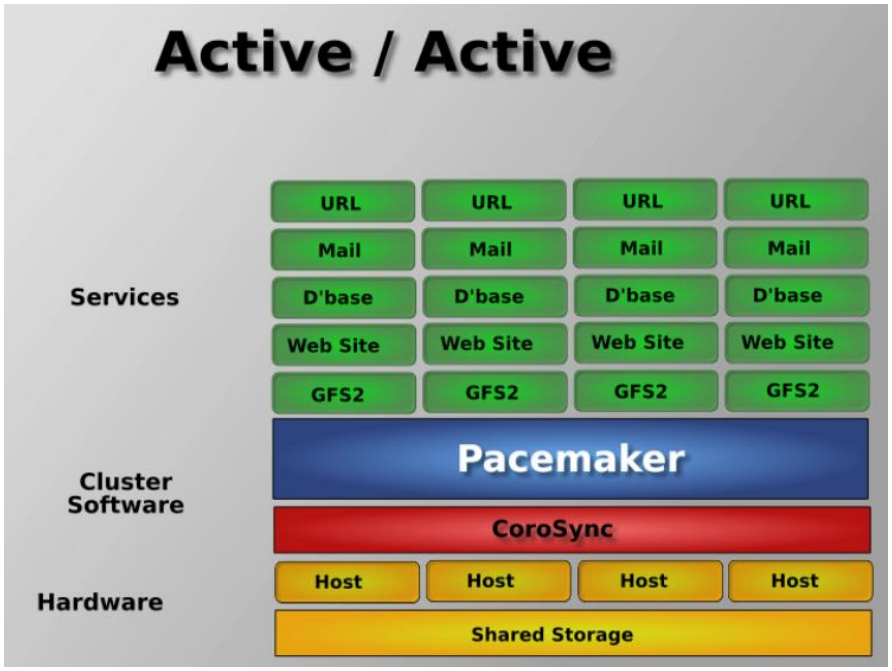


Рисунок 1.17 Схема роботи active/active

Попри значні переваги, Linux-кластери мають і певні недоліки. Побудова кластерної інфраструктури потребує ретельного планування та правильної конфігурації мережі, fencing-механізмів і засобів моніторингу. Помилки конфігурації можуть призвести до нестабільної роботи або втрати даних. Крім того, адміністрування кластерів є значно складнішим, ніж підтримка окремих серверів, особливо коли мова йде про великі HA-системи або розподілені сховища даних.

Таким чином, кластеризація в Linux є одним із ключових механізмів забезпечення відмовостійкості та масштабованості сучасної IT-інфраструктури. Найпоширенішим рішенням для побудови HA-кластерів є використання Pacemaker і Corosync, які забезпечують контроль вузлів, управління ресурсами, quorum і fencing. Завдяки цьому Linux-кластери дозволяють забезпечити безперервну роботу сервісів, мінімізувати простої та підвищити надійність інформаційних систем.

Висновки до першого розділу

У першому розділі було розглянуто теоретичні основи багатофакторної автентифікації як одного з ключових механізмів підвищення рівня безпеки доступу в сучасних інформаційних системах. Проаналізовано основні категорії факторів автентифікації та проблеми однофакторної автентифікації.

Окрему увагу приділено типовим загрозам і атакам на системи MFA, зокрема фішингу, атакам на push-сповіщення, соціальній інженерії та зловживанню людським фактором. З'ясовано, що MFA є ефективним механізмом захисту, але не гарантує абсолютної безпеки без застосування додаткових заходів.

Детально розглянуто архітектуру та принципи роботи PrivacyIDEA. Проаналізовано його основні компоненти: веб-інтерфейс, REST API, базову логіку, механізми доступу до даних, систему політик та підтримувані типи токенів. Також проведено порівняльний аналіз комерційних та open source MFA-рішень. Виявлено переваги й недоліки кожного з них та обрано оптимальне рішення для практичної реалізації. Було приділено увагу кластеризації в Linux-системах, розглянуто основні поняття, варіації, принципи роботи різних типів кластеризації та описано переваги й недоліки використання кластерів.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ПІДГОТОВКА ТЕСТОВОГО СЕРЕДОВИЩА

2.1. Опис віртуального середовища та вибір програмного забезпечення

Для практичної реалізації було обрано платформу віртуалізації Oracle VirtualBox, яка дозволяє створювати декілька взаємопов'язаних віртуальних машин у межах єдиної інфраструктури. Для мережевої взаємодії використовуватиметься NAT Network, що забезпечує внутрішню комунікацію між кінцевими вузлами мережі.

У межах розробленої мережі передбачено мінімальну кількість вузлів для впровадження багатофакторної автентифікації, оскільки середовище є тестовим та не потребує детальної інфраструктури.

Центральним елементом є Windows Server 2022, який виконує роль контролера домену. На сервері розгорнуто доменну структуру Active Directory, що використовується для централізованого керування користувачами, групами, політиками безпеки та мережевими ресурсами. Використання контролера домену дозволяє реалізувати єдину систему автентифікації, спростити адміністрування мережі та забезпечити централізований контроль доступу до ресурсів.

Вибір Windows Server 2022 обумовлений його широким поширенням у корпоративних інфраструктурах, високою стабільністю та підтримкою сучасних механізмів безпеки. Ця серверна операційна система підтримує роботу з віртуалізацією, кластеризацією, резервуванням служб та централізованим адмініструванням, що робить її доцільною для побудови тестового або виробничого середовища. Версія Long-Term Servicing Channel (LTSC) забезпечує тривалий цикл підтримки та оновлень без необхідності частих змін версії системи, що особливо важливо для стабільної роботи серверної інфраструктури.

Також було розгорнуто сервери автентифікації на базі PrivacyIDEA, які відповідають за реалізацію багатофакторної автентифікації, управління токенами, політиками доступу та процесами валідації користувачів. У межах середовища розгорнуто два однакові сервери автентифікації, які функціонують у кластері високої доступності. Такий підхід дозволяє забезпечити безперервність роботи

системи автентифікації навіть у випадку відмови одного з вузлів, а також мінімізувати ризик недоступності сервісу MFA для користувачів.

Окремо розгорнуто сервер бази даних UbuntuDB, який використовується для централізованого зберігання даних системи багатофакторної автентифікації. На даному сервері розгорнуто систему керування базами даних PostgreSQL, до якої підключені обидва сервери PrivacyIDEA. Хоча для PrivacyIDEA використовується MySQL як стандартне рішення, у даному середовищі було обрано саме PostgreSQL. Такий вибір пояснюється високою надійністю PostgreSQL, підтримкою складних механізмів забезпечення цілісності даних, стабільною роботою під навантаженням та розширеними можливостями масштабування. PostgreSQL також відомий кращою відповідністю стандартам SQL, ефективною системою транзакцій та високим рівнем захисту даних, що є важливим для сервісів автентифікації та систем інформаційної безпеки.

Сервери автентифікації та бази даних побудовані на базі Ubuntu Server 20.04 LTS, що обумовлено стабільністю даної серверної операційної системи, тривалим циклом підтримки Long-Term Support та широким використанням у серверних Linux-інфраструктурах.

Також було розгорнуто дві робочі станції на базі Windows 10, які підключені до домену та використовуються для тестування виконаних налаштувань.

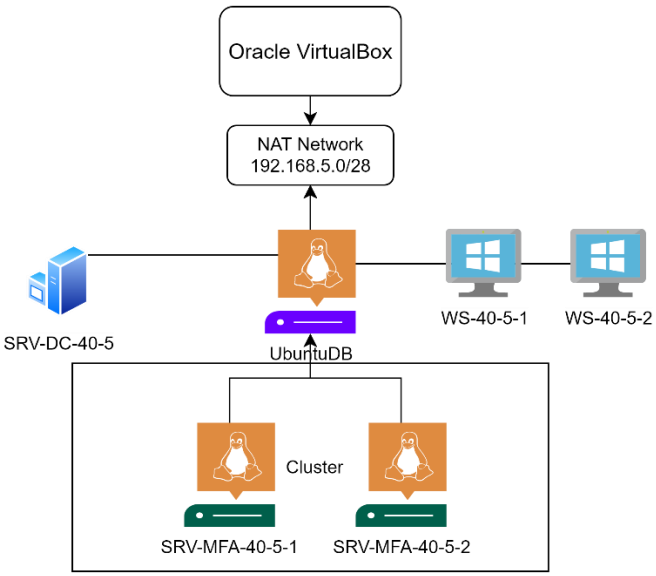


Рисунок 2.1 Схематичне зображення побудованої мережі

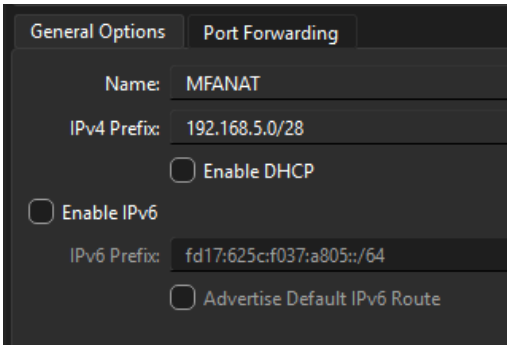


Рисунок 2.2 Створена NAT Network в VirtualBox

2.2. Розробка схеми адресації

Схема адресації була розроблена довільно, щоб забезпечити потреби підключення кінцевих вузлів. На всіх серверах було налаштовано статичну адресу, а робочі станції зі свого боку налаштовані на отримання IP-адреси через DHCP.

Таблиця 2.1

Схема адресації пристроїв

Мережа / Пристрій	Інтерфейс / Мережний адаптер / Шлюз	IP-адреса	Маска	Префікс
Мережа A	-	192.168.5.0	255.255.255.240	/28
R-40-5-1 (NAT Network VirtualBox)	Eth1/0	192.168.5.1	255.255.255.240	/28
SW-40-5-1(NAT Network VirtualBox)	Інтерфейс Vlan 1	192.168.5.2	255.255.255.240	/28
Сервер SRV-DC-40-5	Мережний адаптер	192.168.5.3	255.255.255.240	/28
	Шлюз за замовчуванням	192.168.5.1	-	-
Сервер SRV-MFA-40-5	Мережний адаптер	192.168.5.4	255.255.255.240	/28
	Шлюз за замовчуванням	192.168.5.1	-	-
Сервер SRV-HA-MFA-40-5	Мережний адаптер	192.168.5.5	255.255.255.240	/28
	Шлюз за замовчуванням	192.168.5.1	-	-

Таблиця 2.1

Продовження

Мережа / Пристрій	Інтерфейс / Мережний адаптер / Шлюз	IP-адреса	Маска	Префікс
Сервер Ubuntu-DB	Мережний адаптер	192.168.5.6	255.255.255.240	/28
	Шлюз за замовчуванням	192.168.5.1	-	-
WS-40-5-1	Мережний адаптер	DHCP		
	Шлюз за замовчуванням			
WS-40-5-2	Мережний адаптер	DHCP		
	Шлюз за замовчуванням			

2.3. Базові налаштування серверів

Для коректної роботи доменної інфраструктури та подальшої інтеграції з PrivacyIDEA необхідно виконати базові налаштування контролера домену. Для початку було встановлено IP-адресу згідно з таблицею адресації (п. 2.2).

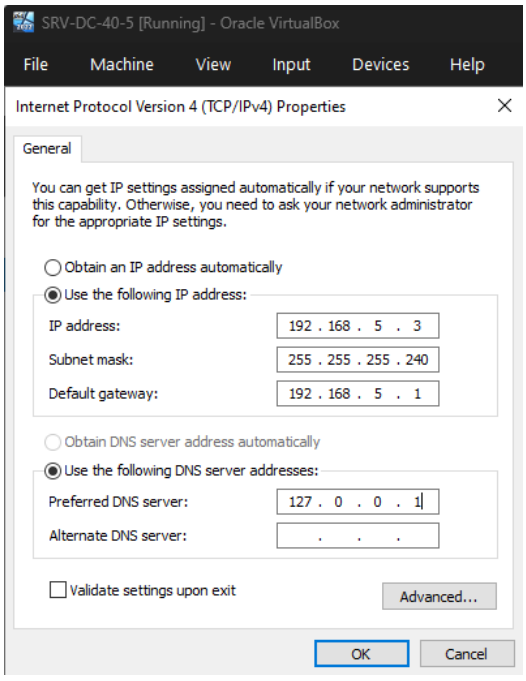


Рисунок 2.3 Налаштування IP-адресації контролера домену

Далі змінимо ім’я та встановимо роль AD DS виконуючи відповідні команди в оснащенні PowerShell.

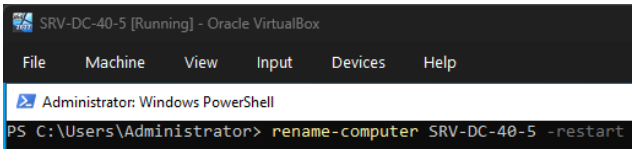


Рисунок 2.4 Зміна імені контролера домену

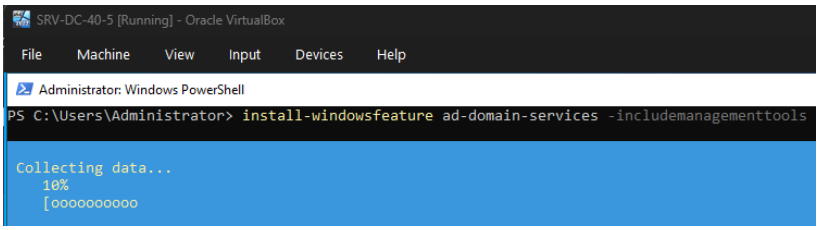


Рисунок 2.5 Встановлення ролі AD DS(Active Directory Domain Services)

Для створення домену було використано ім’я org.net, яке функціонує виключно в ізольованому віртуальному середовищі та не має взаємодії з публічною мережевою інфраструктурою.

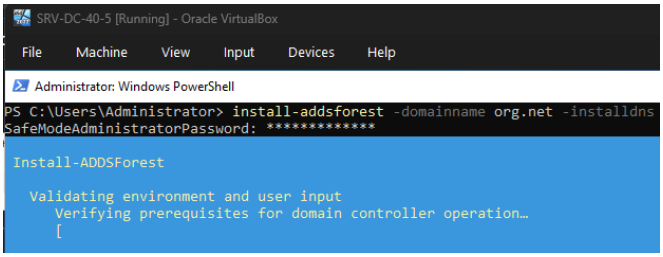


Рисунок 2.6 Ініціалізація домену

Також було встановлено та налаштовано DHCP на контролері домену.

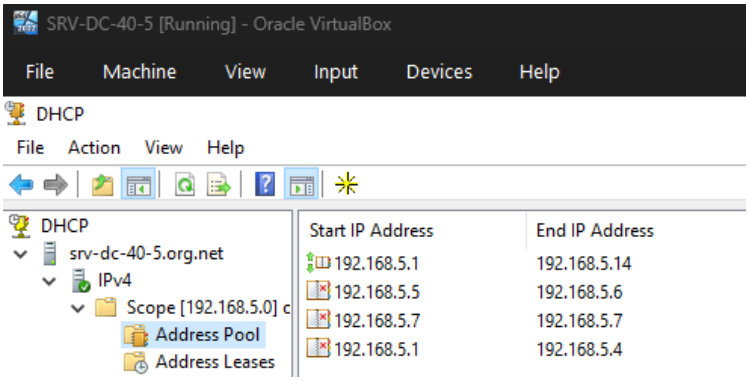


Рисунок 2.7 Налаштований DHCP Scope

Було створено базову структуру організаційних одиниць із розподілом користувачів за департаментами, що дозволяє відокремити їх від стандартного контейнера Users та забезпечує зручніше управління обліковими записами.

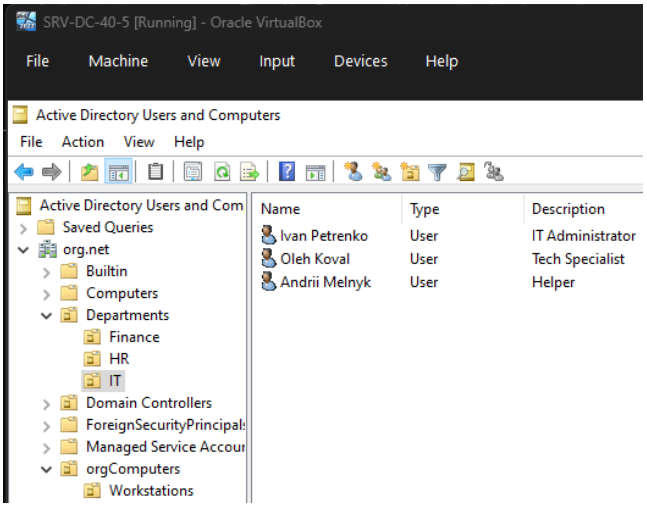


Рисунок 2.8 Побудована структура домену

На Linux-серверах також було проведено базове налаштування IP-адресації для зв'язку з усіма вузлами мережі, а також виконано port forwarding для зручнішої роботи в терміналах.

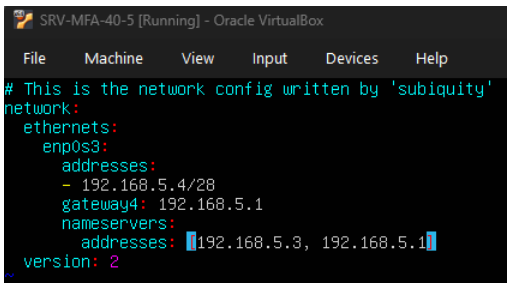


Рисунок 2.9 Налаштування мережевого конфігураційного файлу SRV-MFA-40-5

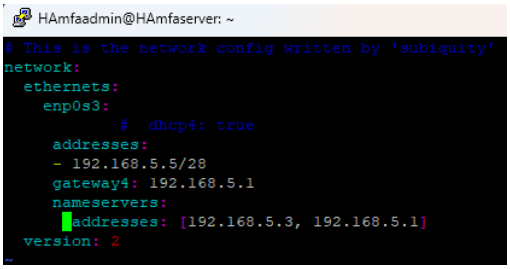


Рисунок 2.10 Налаштування мережевого конфігураційного файлу SRV-HA-MFA-40-5

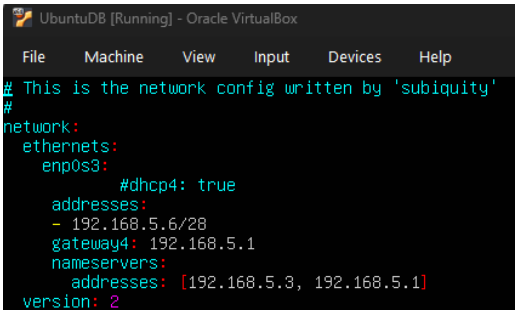


Рисунок 2.11 Налаштування мережевого конфігураційного файлу UbuntuDB

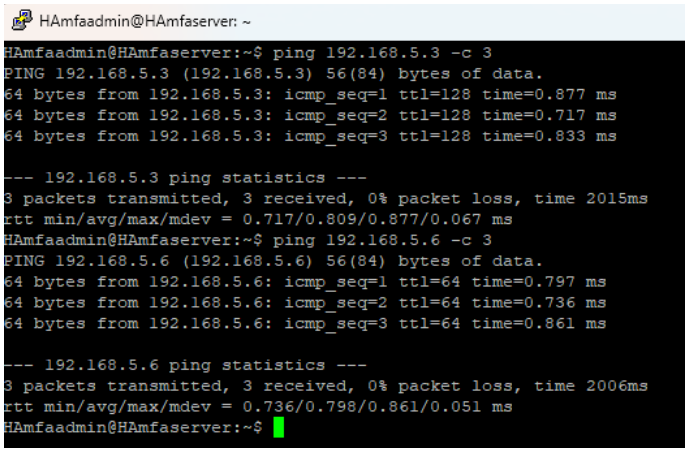


Рисунок 2.12 Перевірка зв’язку між вузлами мережі

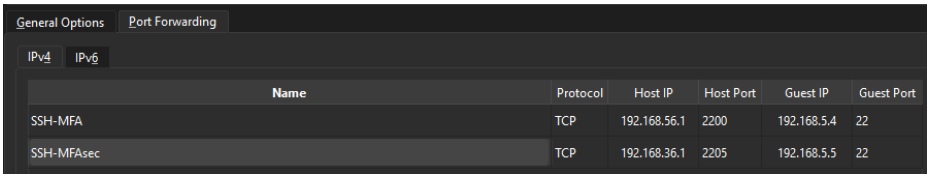


Рисунок 2.13 Виконання прокидання портів

Висновки до другого розділу

У другому розділі було виконано проєктування та підготовку тестового середовища для впровадження багатофакторної автентифікації. Сформовано логічну структуру мережі в ізольованому віртуальному середовищі VirtualBox, що включає контролер домену Windows Server, сервери PrivacyIDEA, сервер бази даних та робочі станції, які були об’єднані в одну мережу.

Особливу увагу приділено підготовці контролера домену та розробці схеми адресації. Було створено тестовий домен та базову ієрархію організаційних одиниць із розподілом користувачів за департаментами, що сформувало впорядковану структуру. Також було проведено базові налаштування мережевих адаптерів серверів на базі операційної системи Linux.

Таким чином сформовано готове середовище для подальшого впровадження та налаштування PrivacyIDEA, яке дозволяє коректно змоделювати взаємодію між користувачами домену та сервером автентифікації.

РОЗДІЛ 3. ВПРОВАДЖЕННЯ КЛАСТЕРИЗОВАНОЇ БАГАТОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ

3.1. Підготовка серверного середовища та встановлення PrivacyIDEA

Після проведення базових налаштувань IP-адресації першим та одним із найважливіших кроків є оновлення списків доступних пакетів та програмного забезпечення, вже встановленого в системі. Оскільки для подальшого коректного налаштування роботи кластера нам потрібні два сервери, всі налаштування та процес встановлення на SRV-HA-MFA-40-5 будуть ідентичними. Для прикладу будуть показані покрокові дії, що виконувалися на сервері SRV-MFA-40-5.

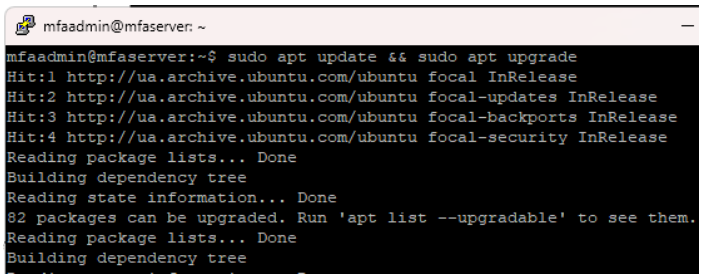


Рисунок 3.1 Виконання sudo apt update та upgrade

Наразі середовище повністю підготовлене до встановлення PrivacyIDEA. Спочатку потрібно завантажити ключ цифрового підпису та додати його до системи, після чого слід додати репозиторій для відповідної версії та виконати встановлення.

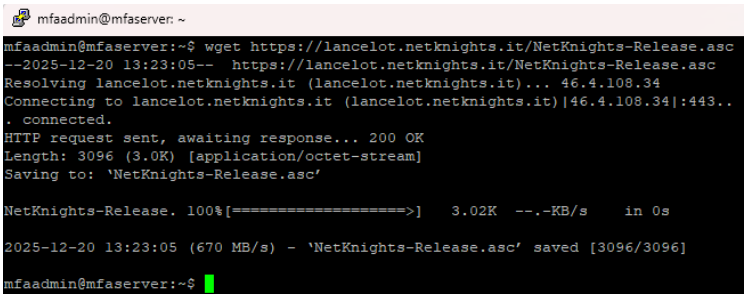


Рисунок 3.2 Завантаження ключа підпису

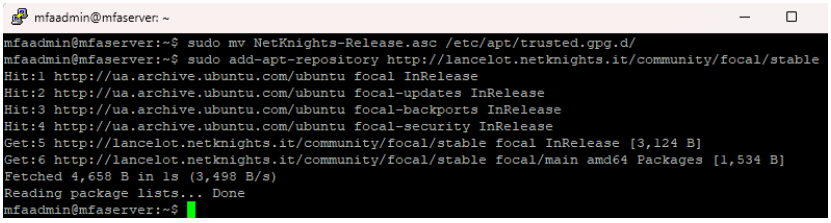


Рисунок 3.3 Додавання ключа та репозиторія в систему

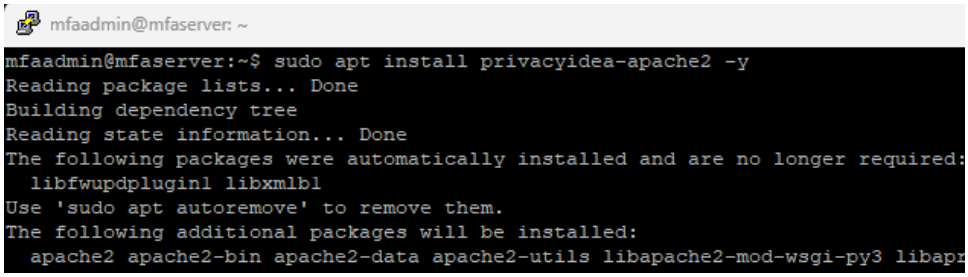


Рисунок 3.4 Встановлення PrivacyIDEA

Після успішного встановлення PrivacyIDEA з усіма потрібними залежностями необхідно створити обліковий запис адміністратора для доступу до веб-інтерфейсу сервера. Також треба пройти процес авторизації за щойно створеним користувачем.

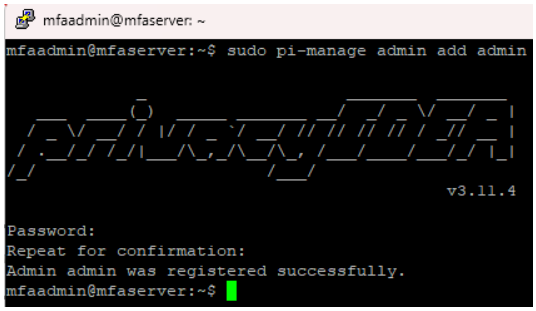


Рисунок 3.5 Створення облікового запису адміністратора

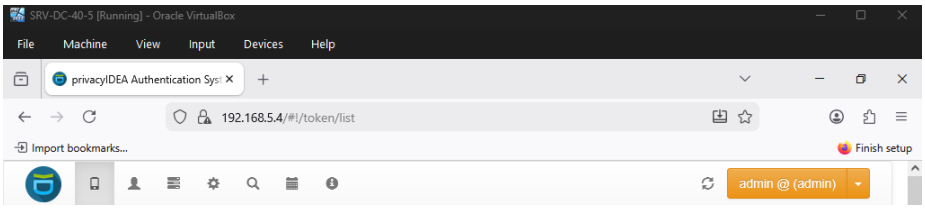


Рисунок 3.6 Успішний вхід за створеним користувачем

3.2. Інтеграція PrivacyIDEA з Active Directory

З метою уникнення використання облікового запису адміністратора Windows Server створимо службового користувача для взаємодії між PrivacyIDEA та AD. Цей користувач буде використовуватися лише для отримання інформації про наявних користувачів у доменній структурі за допомогою протоколу LDAP, тому задля більшої безпеки також заборонимо йому вхід у систему.

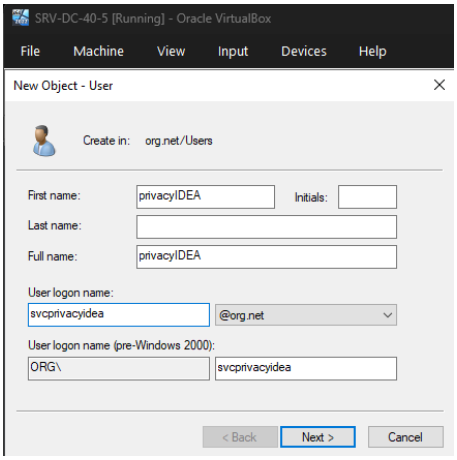


Рисунок 3.7 Створення службового облікового запису

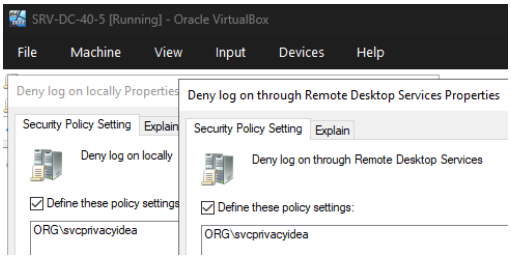


Рисунок 3.8 Заборона входу для службового користувача

Для інтеграції з Active Directory у WebUI системи PrivacyIDEA потрібно створити LDAP-resolver, який у подальшому буде прив'язаний до відповідного realm'у. Підключення до каталогу здійснюється з використанням щойно створеного службового облікового запису.

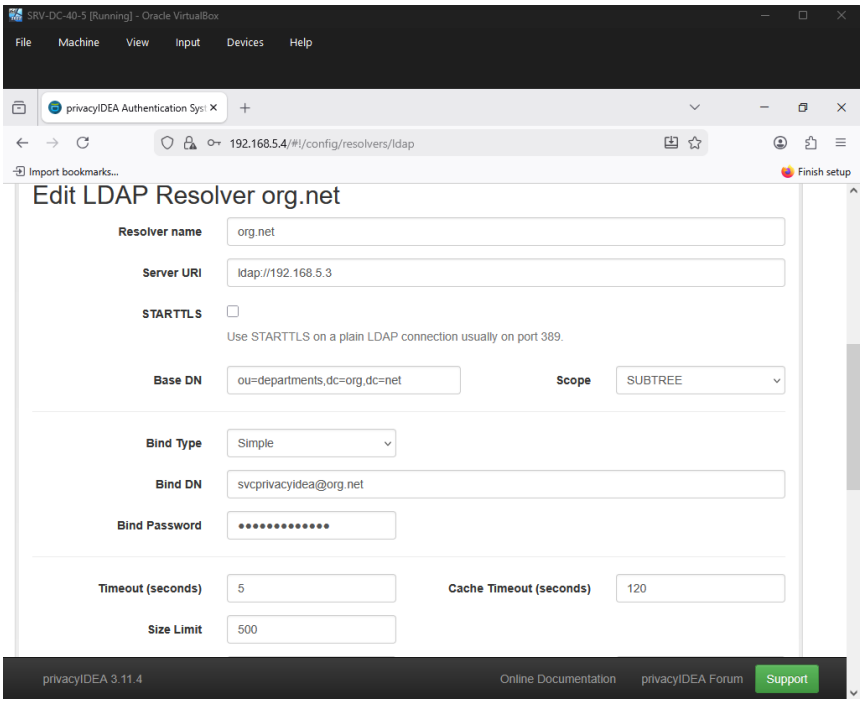


Рисунок 3.9 Створення ldap resolver

Під час налаштування обираємо попередньо визначений пресет для Active Directory та проводимо тестування коректності з'єднання та отримання даних.

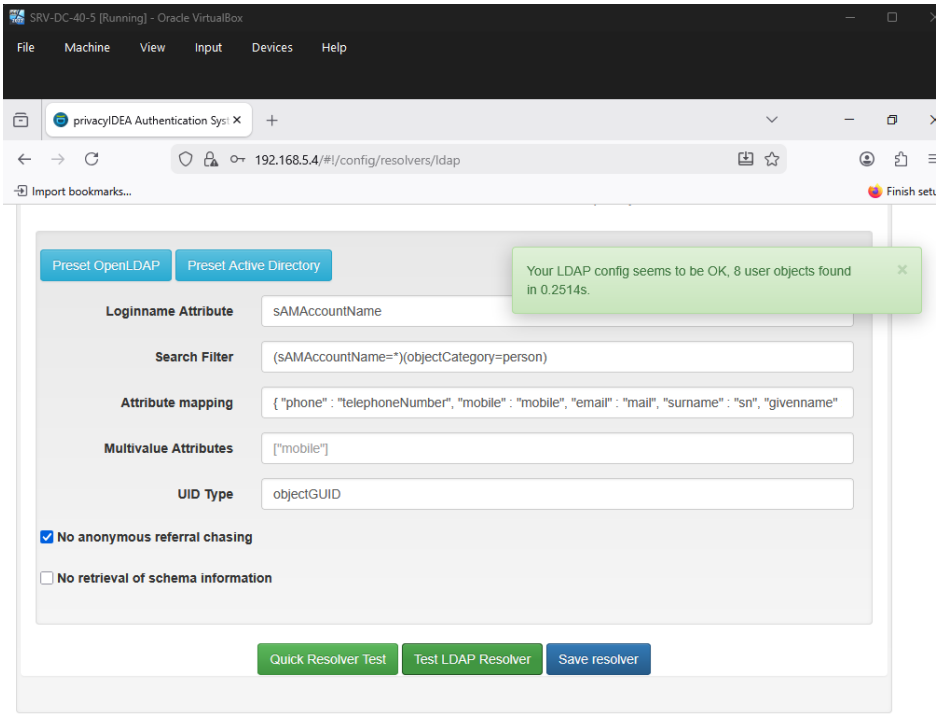


Рисунок 3.10 Тестування роботи ldap resolver

Наступним етапом потрібно створити realm та прив'язати щойно створений resolver до нього. Завдяки цьому ми отримаємо логічну групу користувачів з назвою users.org.net, до якої в подальшому можна буде налаштовувати окремі політики та правила.

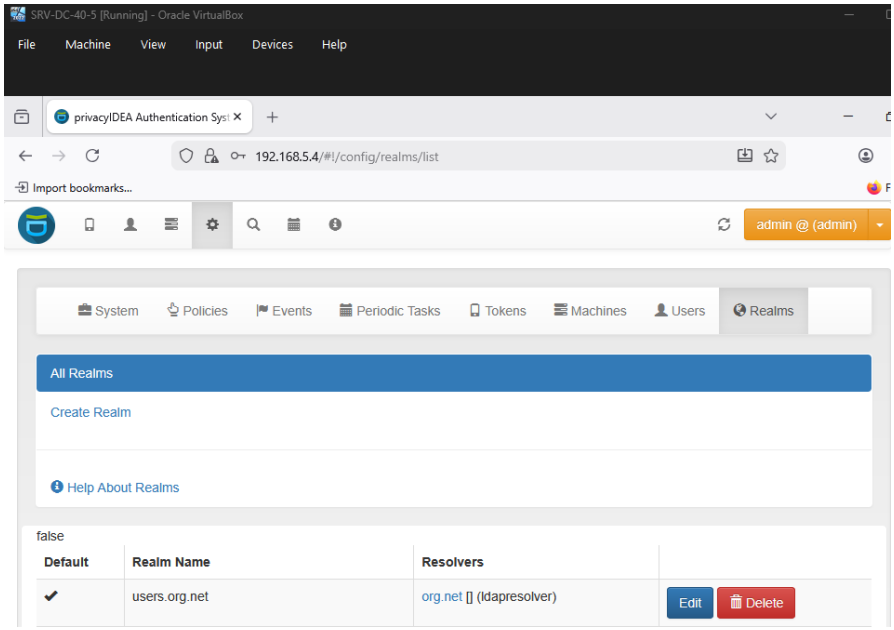


Рисунок 3.11 Прив'язка до realm

В результаті виконаних дій система отримує користувачів з організаційної одиниці Departments та записує їх у realm users.org.net.

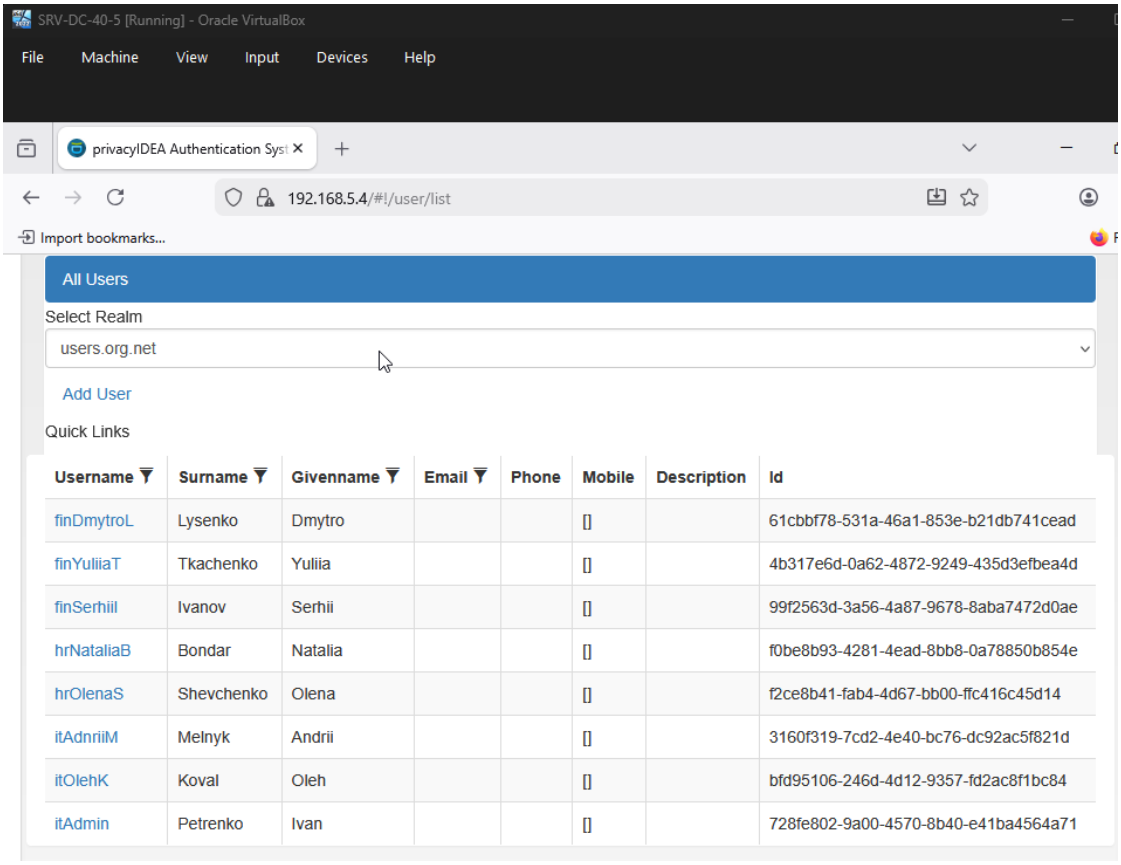


Рисунок 3.12 Перевірка отримання даних

На даному етапі система функціонує коректно, однак обмін даними з Active Directory здійснюється через незахищений протокол LDAP, а доступ до веб-інтерфейсу PrivacyIDEA відбувається через HTTP. У подальшому буде впроваджено самопідписані сертифікати з метою захисту мережевих з'єднань.

3.3. Захист каналів взаємодії

З метою усунення ризиків, пов'язаних із передаванням облікових даних та службової інформації у відкритому вигляді, на наступному етапі реалізується захист мережевої взаємодії між компонентами системи. Основна увага приділяється шифруванню з'єднань між PrivacyIDEA та Active Directory, а також захисту доступу до веб-інтерфейсу системи.

На контролері домену розгортаємо Active Directory Certificate Services та створюємо сертифікат за шаблоном Domain Controller Authentication.

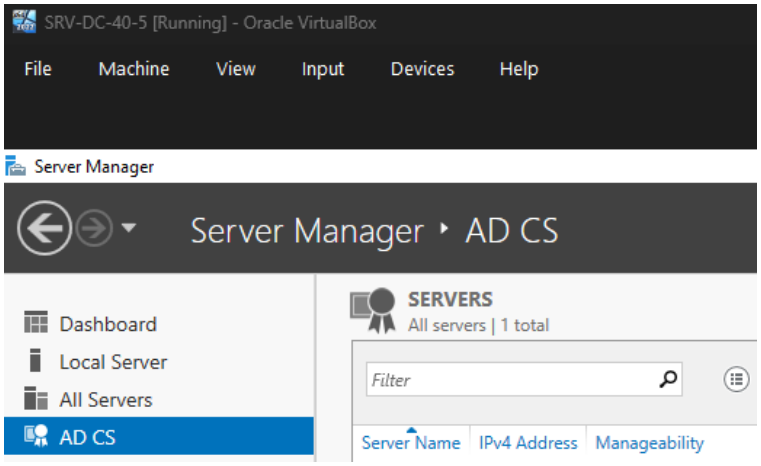


Рисунок 3.13 Встановлений Active Directory Certificate Services

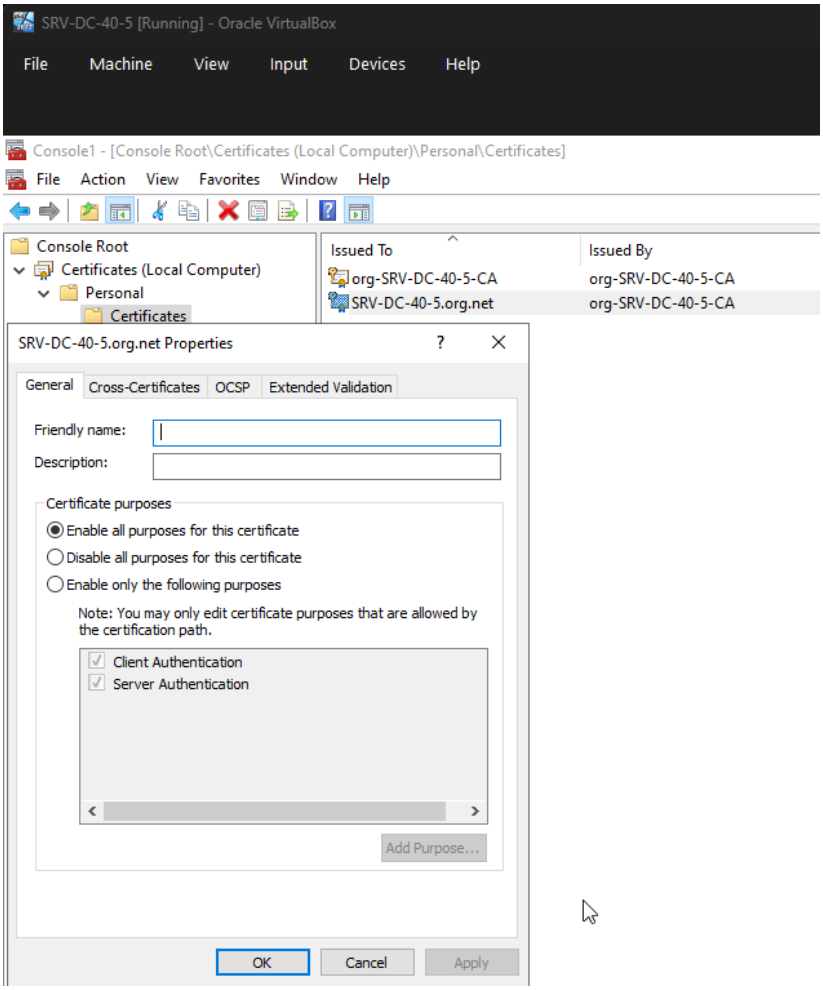


Рисунок 3.14 Створений сертифікат

Тепер треба експортувати даний сертифікат та перенести його на Linux-сервер, на якому знаходиться наш PrivacyIDEA. Для цього використаємо вбудовану функцію експорту, а для безпечної передачі сертифіката між серверами – scp.

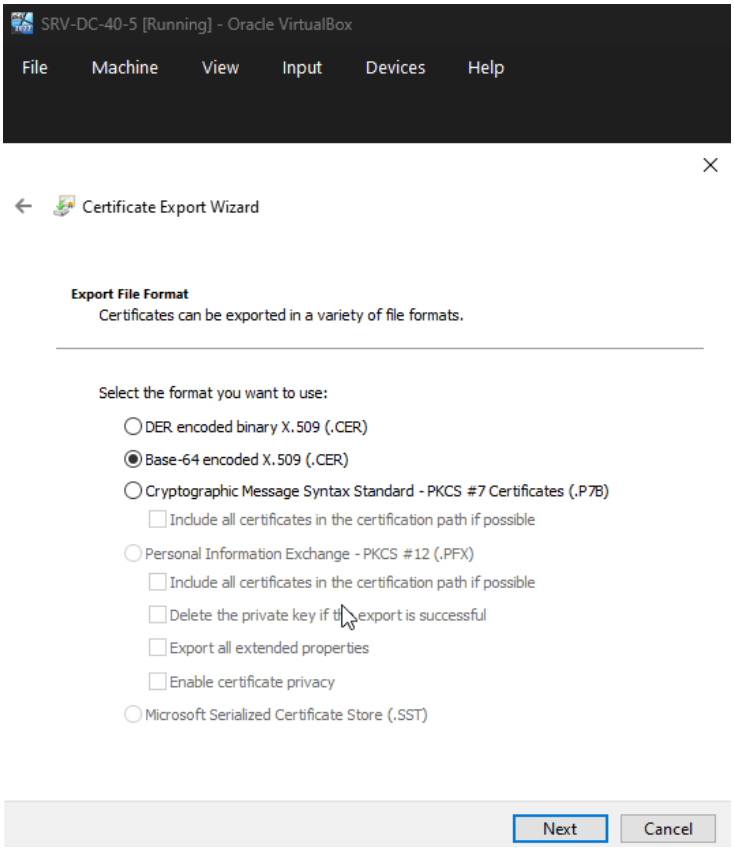


Рисунок 3.15 Експортування сертифікату

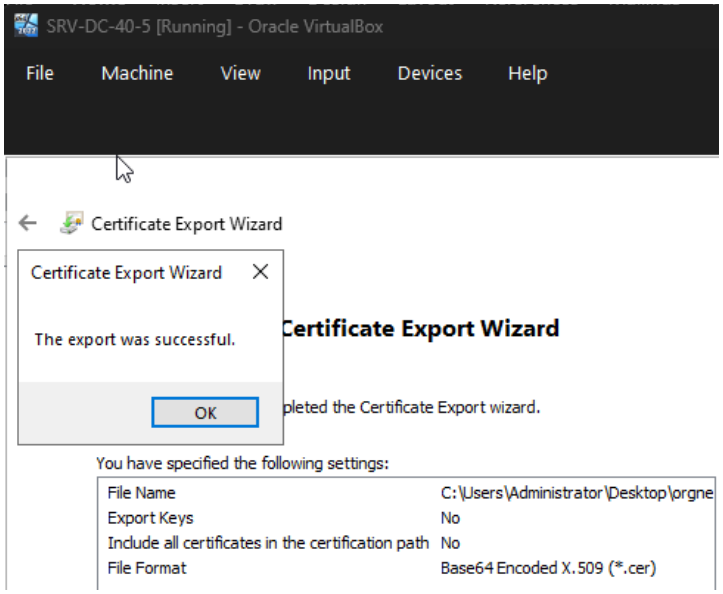


Рисунок 3.16 Успішно експортований сертифікат

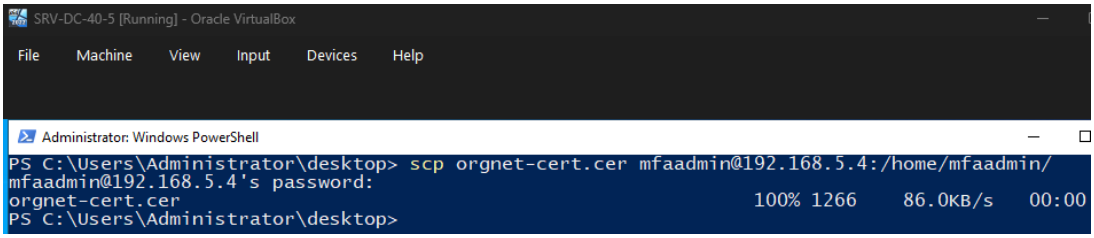


Рисунок 3.17 Передача сертифікату за допомогою scp

Після успішної передачі сертифіката на Linux-сервері скопіюємо його в стандартну директорію довірених сертифікатів та виконаємо оновлення даного сховища.

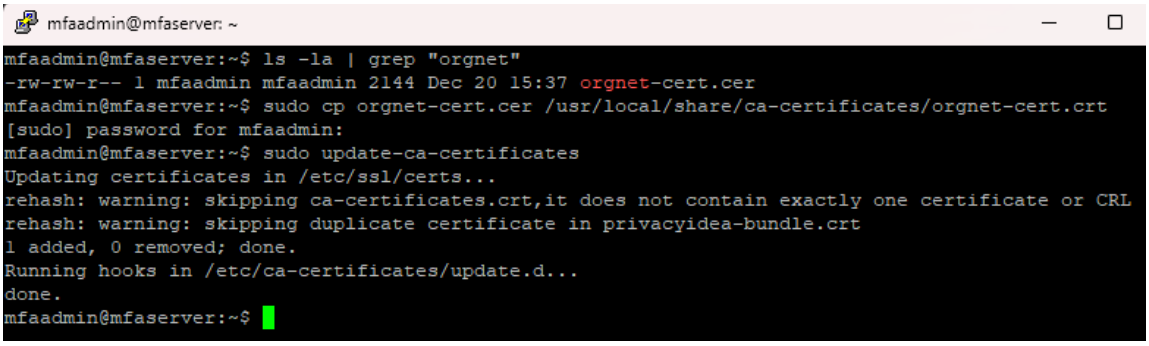


Рисунок 3.18 Додавання сертифікату до списку довірених на SRV-MFA-40-5

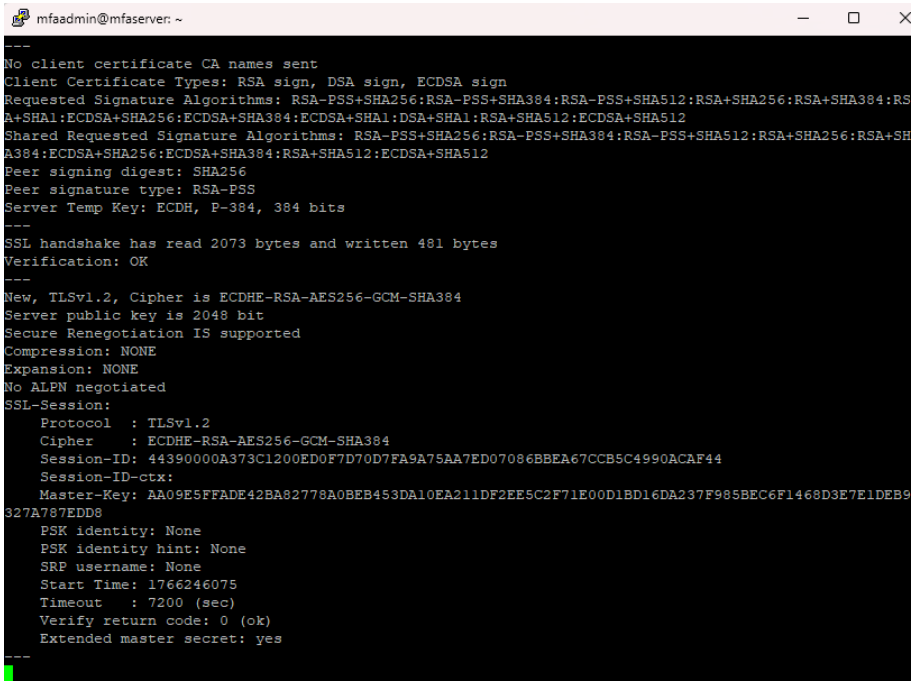


Рисунок 3.19 Перевірка роботи встановленого сертифікату за допомогою OpenSSL

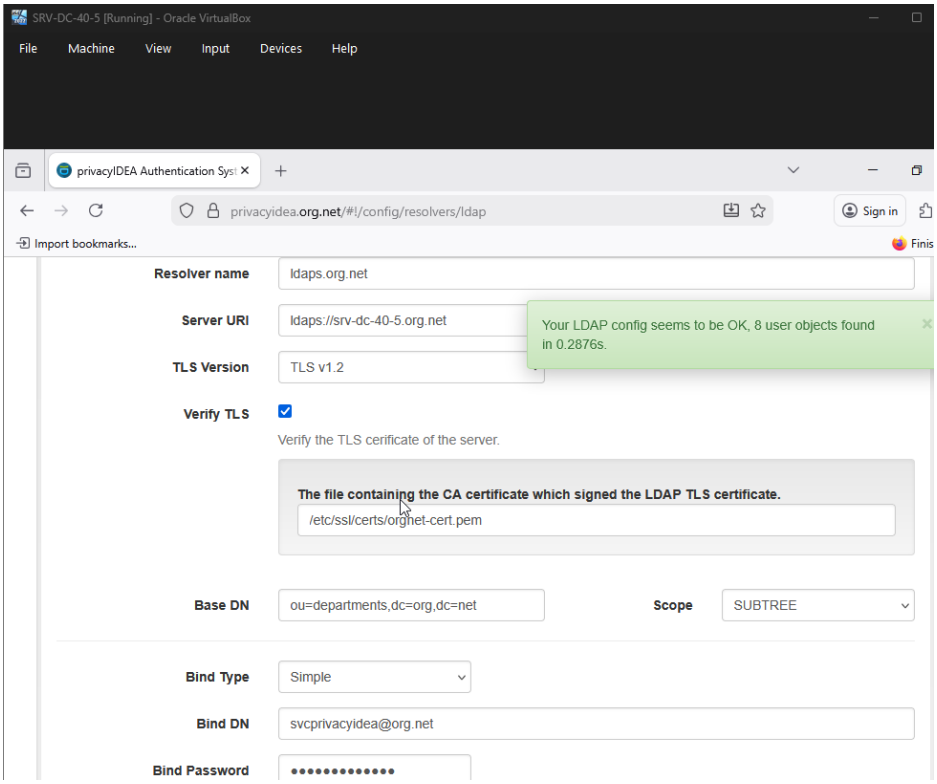


Рисунок 3.20 Перевірка роботи протоколу LDAPS

Наступним етапом згенеруємо самопідписаний сертифікат для роботи з HTTPS. Для цього на Linux-сервері спочатку створимо приватний ключ та CSR-запит. Далі підпишемо цей CSR власним ключем, щоб отримати готовий сертифікат. Після цього відредагуємо конфігурацію Apache, вказавши шляхи до нового ключа та сертифіката, і перезапустимо службу.

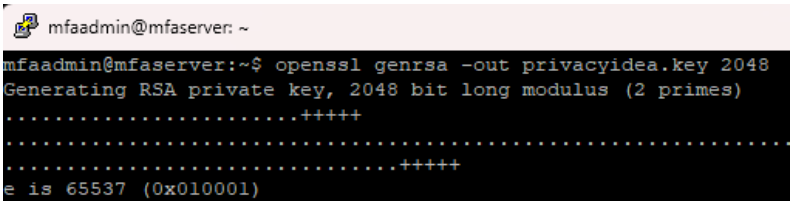


Рисунок 3.21 Генерація ключа

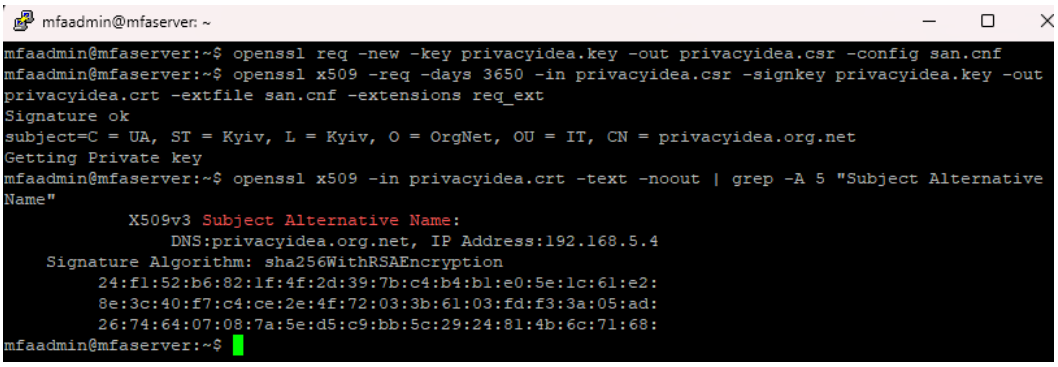


Рисунок 3.22 Створення самопідписаного сертифікату

Для зручності конфігурації Apache перенесемо ключ та сертифікат у кореневу директорію PrivacyIDEA.

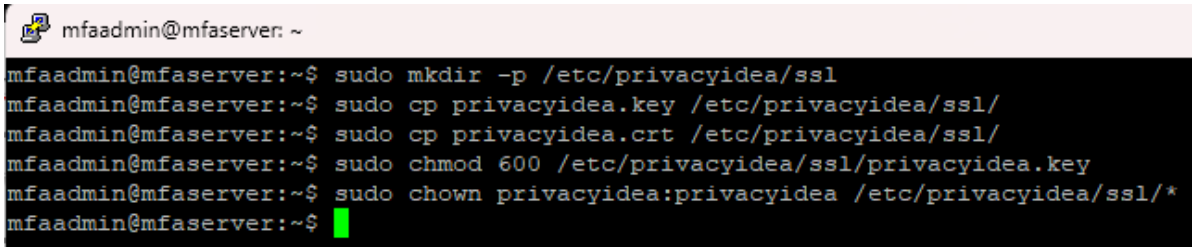


Рисунок 3.23 Створення директорій та переміщення сертифікату з ключем

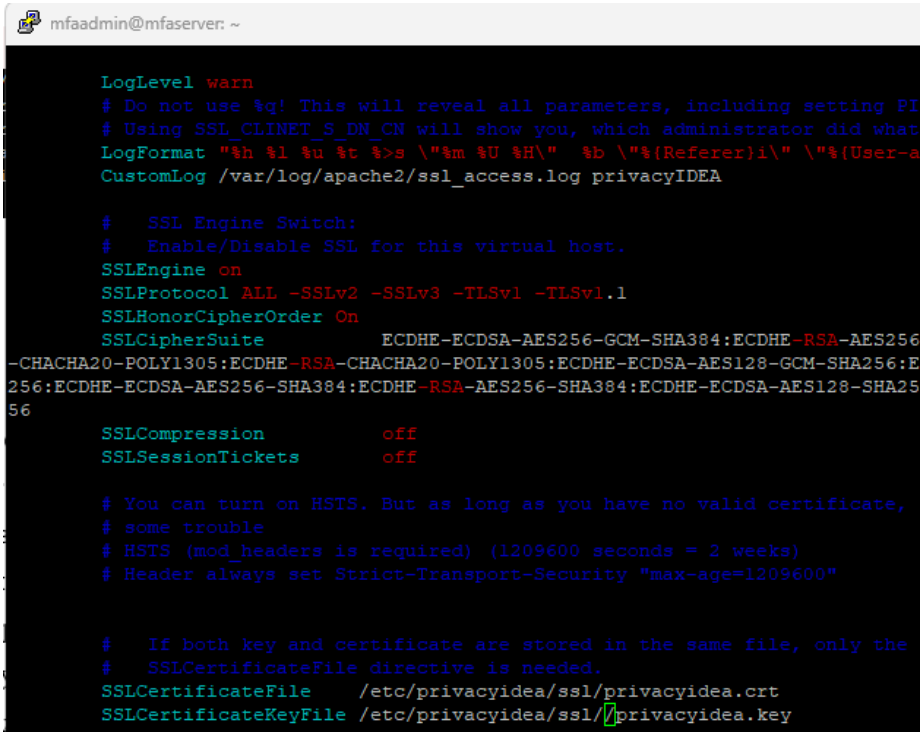


Рисунок 3.24 Редагування конфігураційного файлу Apache

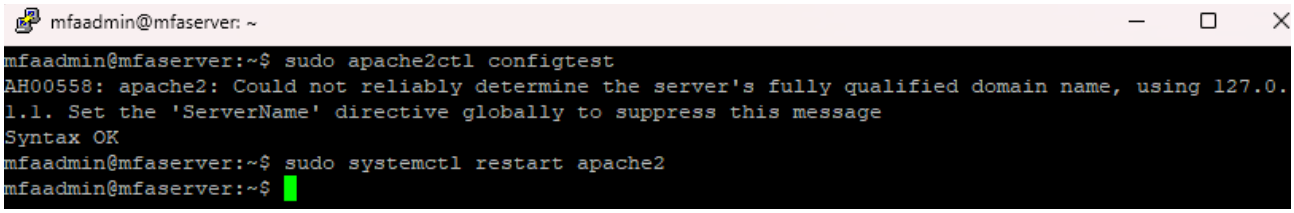


Рисунок 3.25 Перевірка конфігурації та перезапуск служби

Останнім етапом імпортуємо даний сертифікат у Windows Server, щоб усі доменні користувачі довіряли йому, та перевіряємо роботу HTTPS. Копіюємо на Windows Server, використовуючи scp, та за допомогою certmgr.msc імпортуємо в систему.

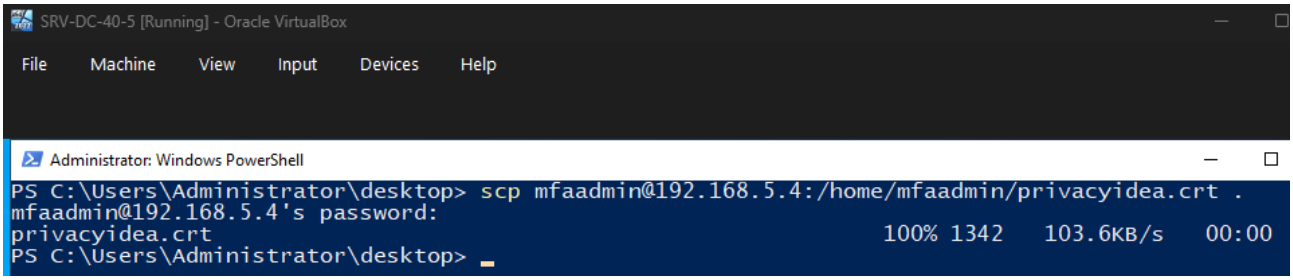


Рисунок 3.26 Копіювання сертифікату на контролер домену

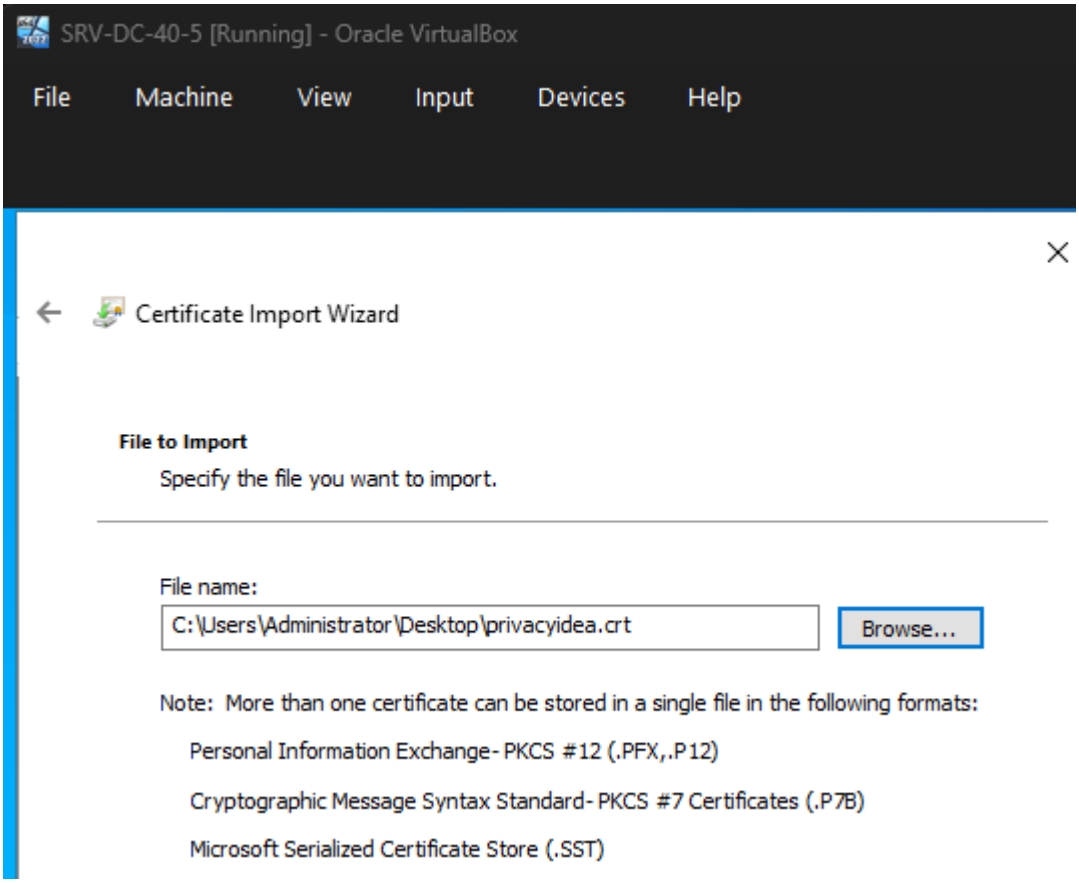


Рисунок 3.27 Імпортування сертифікату

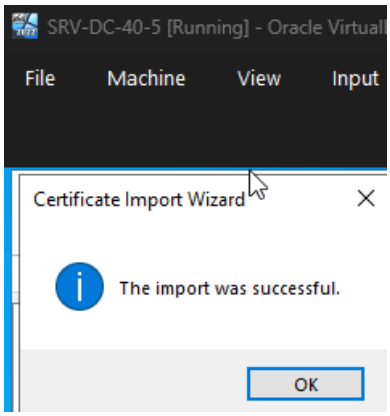


Рисунок 3.28 Результат успішного імпортування

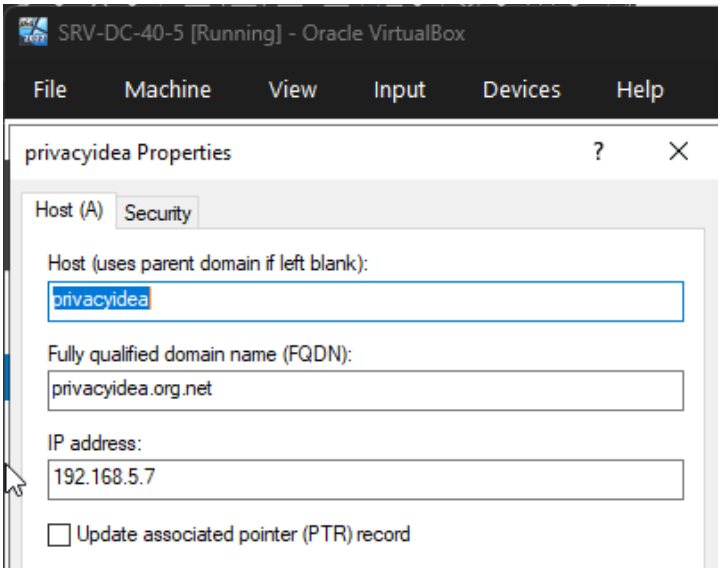


Рисунок 3.29 Створення DNS запису

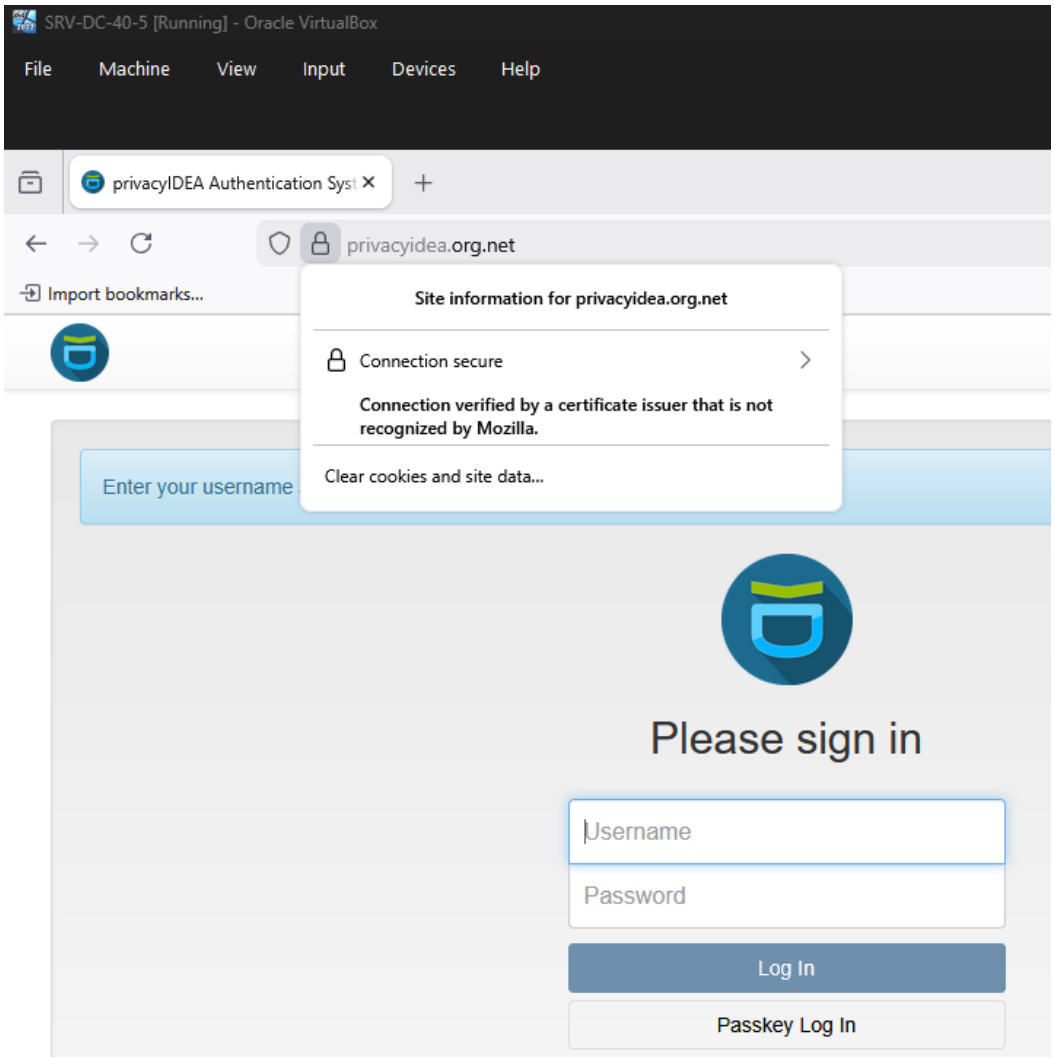


Рисунок 3.30 Перевірка роботи сертифікату

3.4. Налаштування багатофакторної автентифікації

Для коректної роботи автентифікації та валідації токенів потрібно створити базові налаштування політик. Для цього перейдемо до веб-інтерфейсу системи, далі – конфігурація та політики.

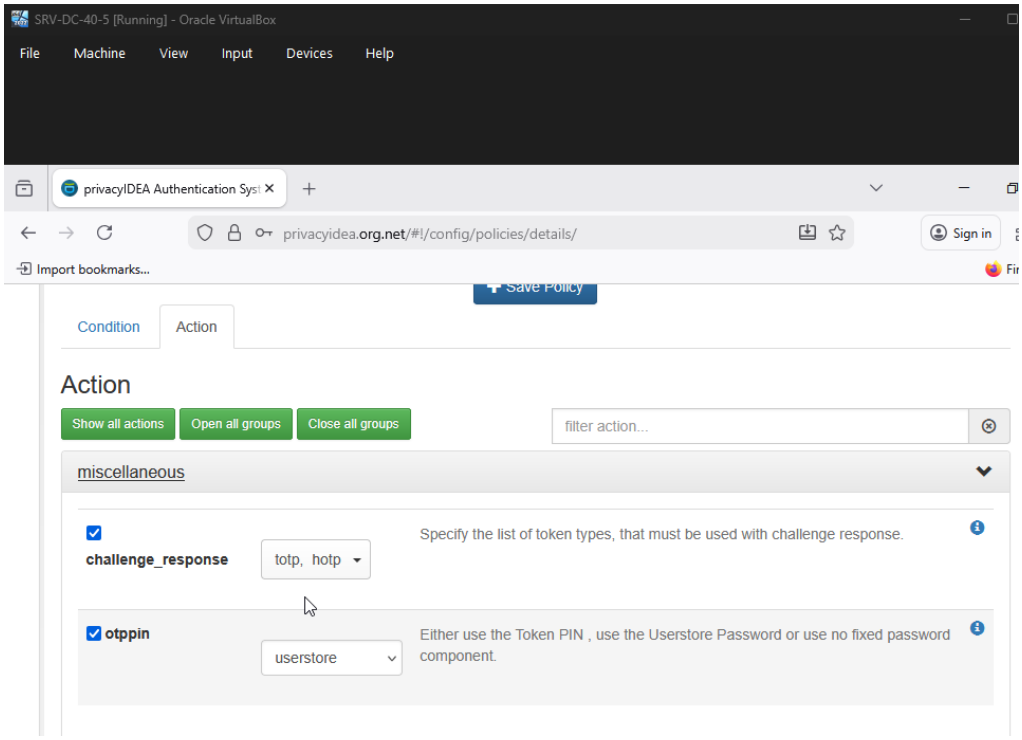


Рисунок 3.31 Створення політики валідації для кінцевих вузлів

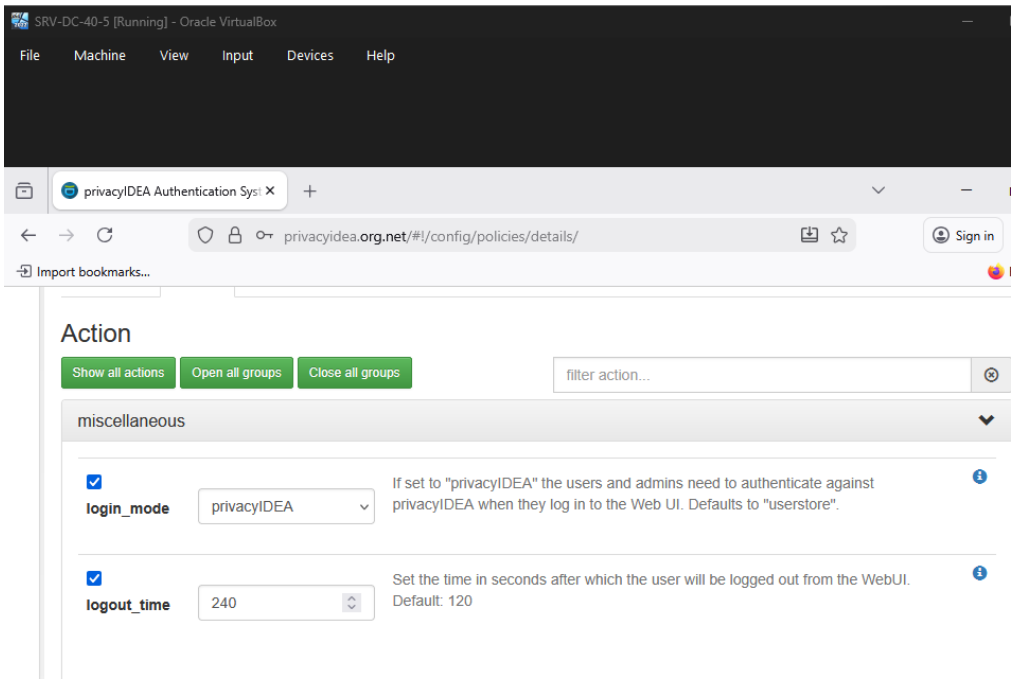


Рисунок 3.32 Створення політики автентифікації до WebUI

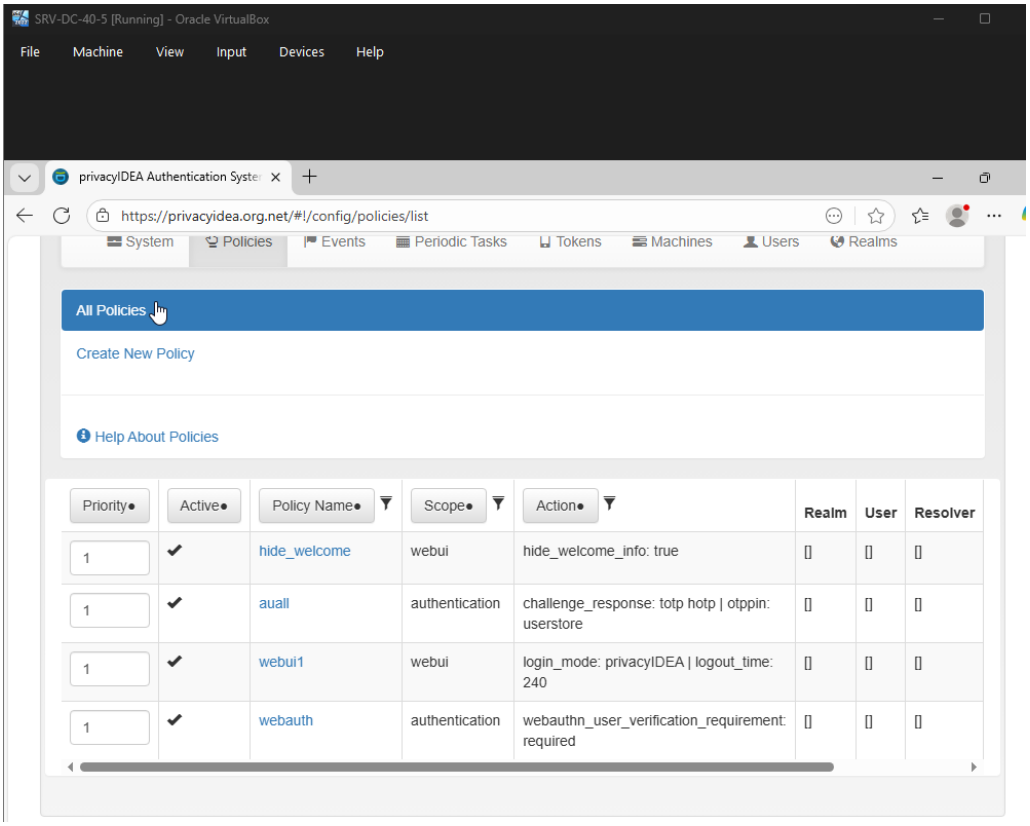


Рисунок 3.33 Успішно створені основні політики

Перед створенням токенів на кінцеві вузли встановлюємо PrivacyIDEA Credential Provider, за допомогою якого буде здійснюватися зв'язок між клієнтом та PrivacyIDEA сервером та проходити процес валідації користувача.

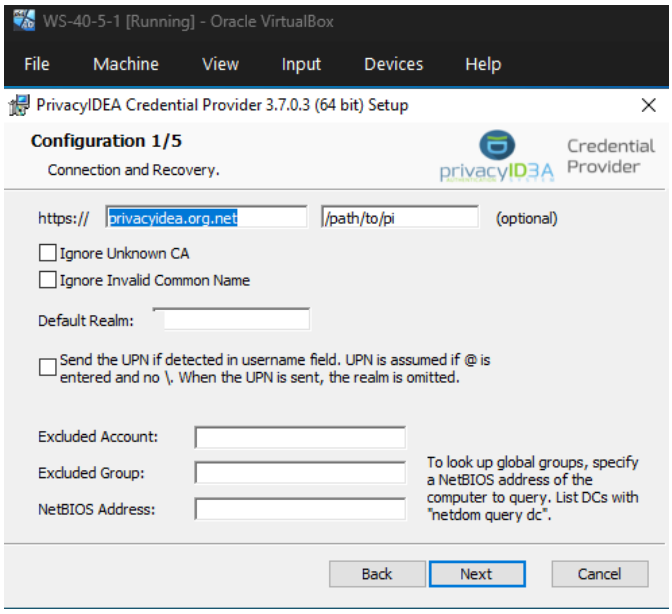


Рисунок 3.34 Процес встановлення Credential Provider

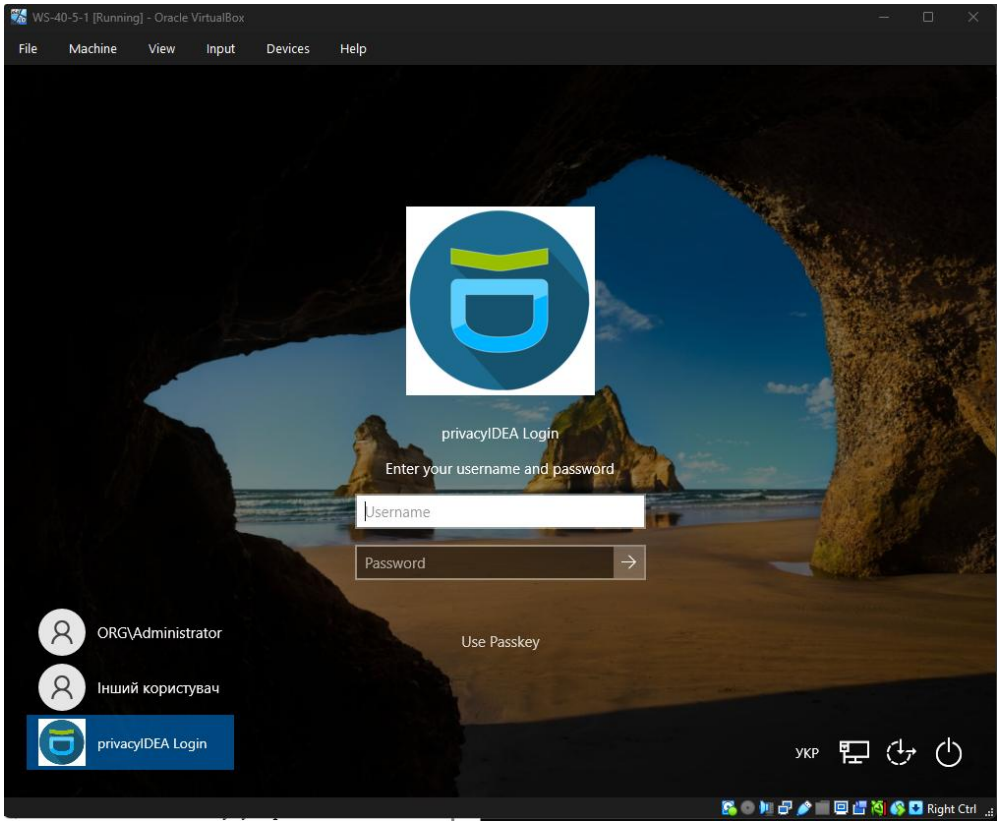


Рисунок 3.35 Успішно встановлений Credential Provider

Оскільки PrivacyIDEA Credential Provider – це лише додатковий вид автентифікації облікових записів, який не порушує роботу стандартних засобів Windows, ми все ще можемо обійти його та використати звичайний вхід. Щоб запобігти цьому, на контролері домену в груповій політиці створимо виключення для стандартного Password Credential Provider.

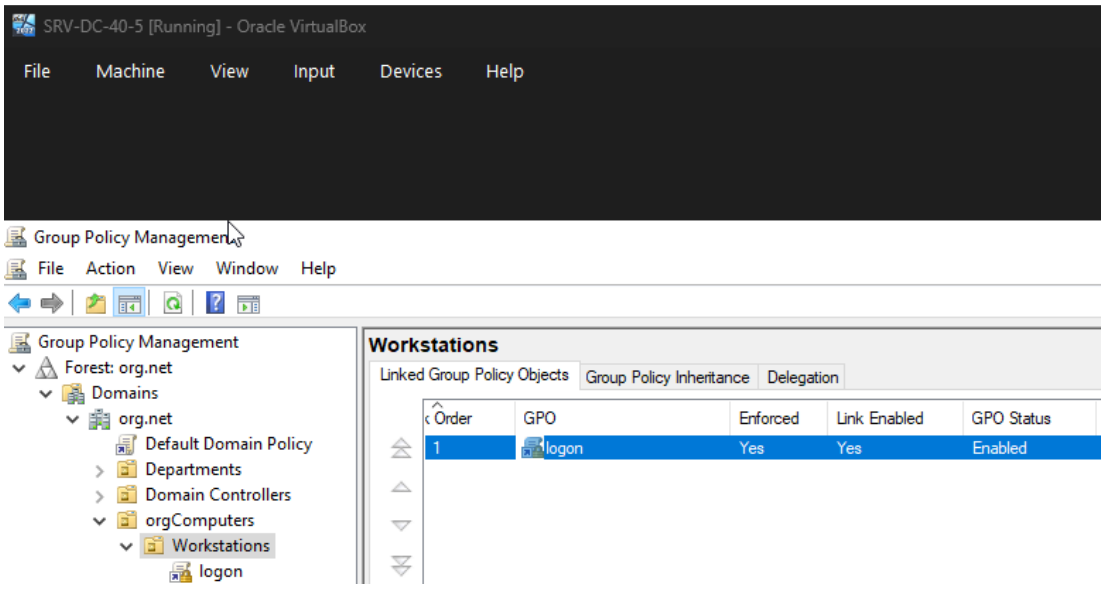


Рисунок 3.36 Створена нова GPO для OU Workstations

У редакторі реєстру Windows переходимо за шляхом: Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\Credential Providers. І знаходимо CLSID, що призначено для Password Credential Provider.

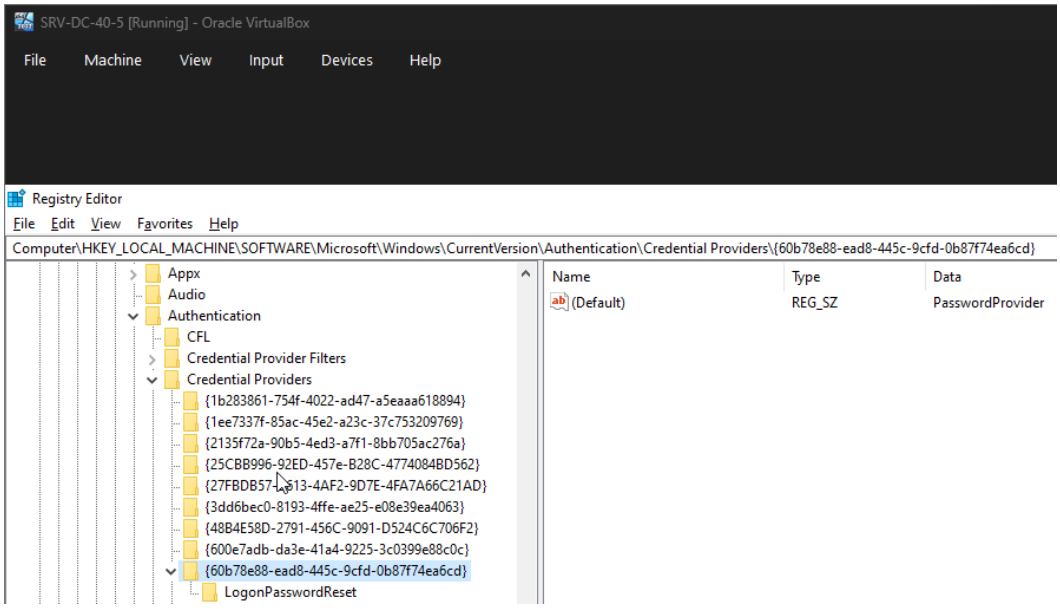


Рисунок 3.37 Гілка реєстру з CLSID парольного провайдеру

Використовуючи групові політики та правило Exclude Credential Providers, виключаємо стандартний парольний провайдер Windows, зберігаємо зміни та примусово оновлюємо політики.

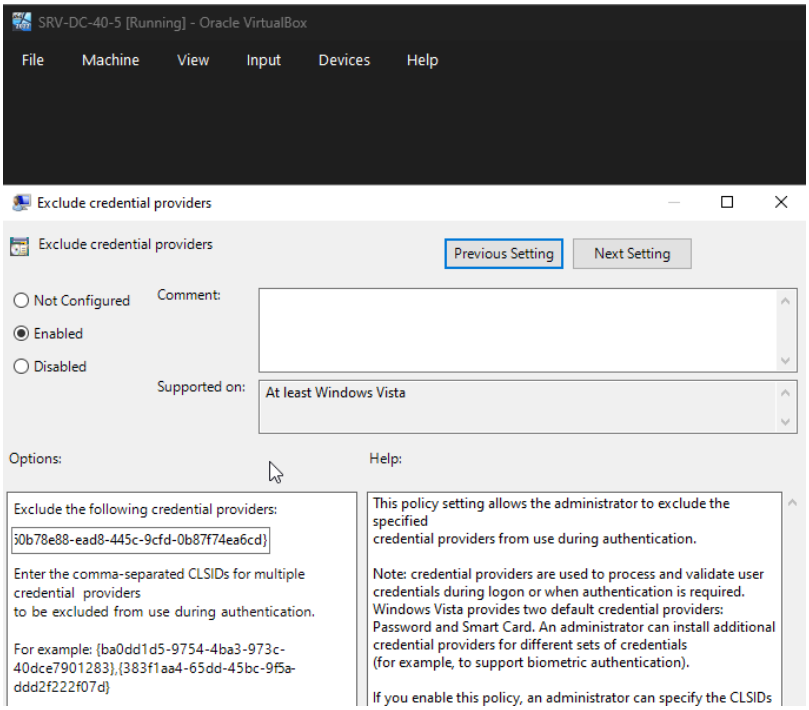


Рисунок 3.38 Виключення Password Credential Provider

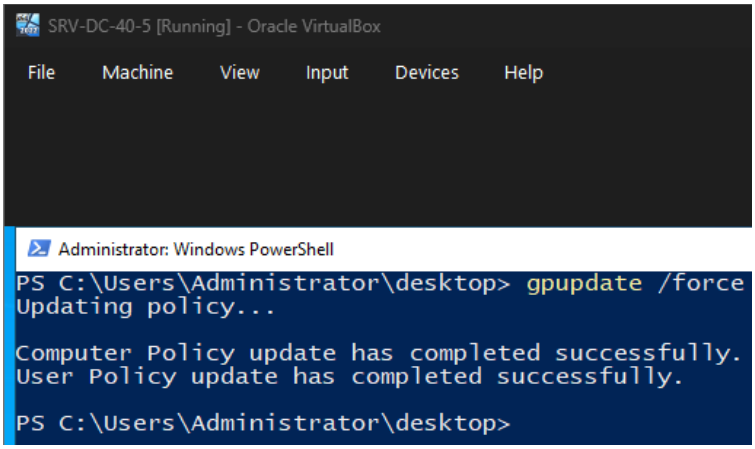


Рисунок 3.39 Примусове оновлення групових політик

Після виконання дій перевіряємо вхід у систему на робочій станції. Єдиною точкою входу тепер залишається PrivacyIDEA Credential Provider, що унеможливлює авторизацію без проходження двофакторної перевірки. Використовуючи веб-інтерфейс, створимо TOTP токен для доменного користувача, відсканувавши згенерований QR-код за допомогою мобільного додатка PrivacyIDEA Authenticator, побачимо код для автентифікації.

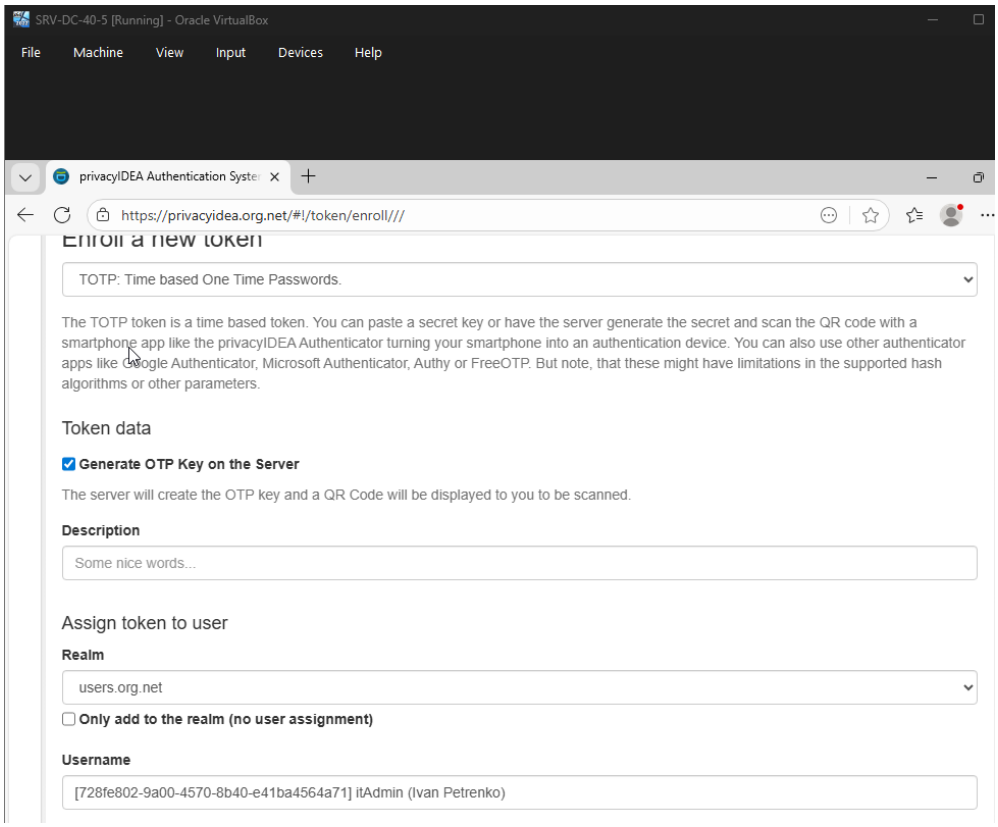


Рисунок 3.40 Генерація токену для облікового запису itAdmin

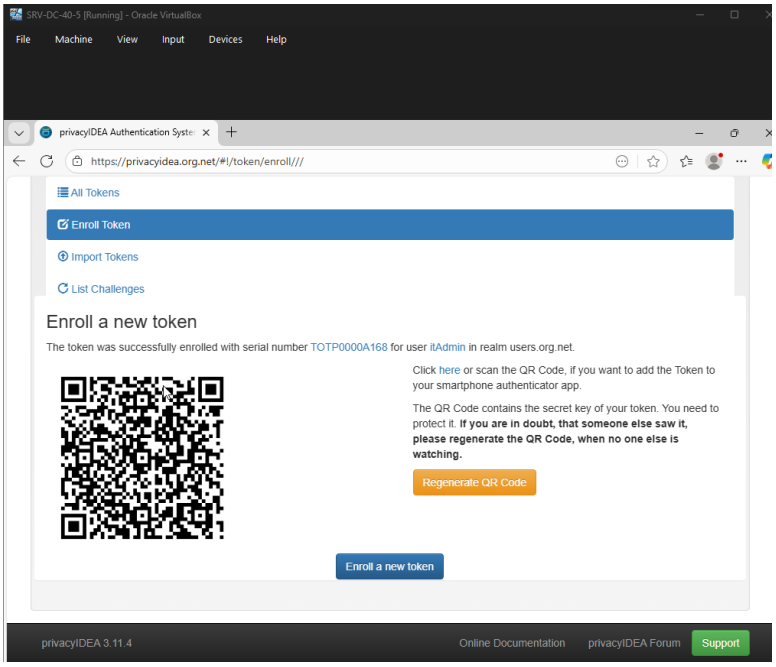


Рисунок 3.41 Згенерований QR-код



Рисунок 3.42 Створений токен у веб-інтерфейсі

3.5. Налаштування кластеризації

Для реалізації кластеризації Linux машин було використано менеджер ресурсів Pacemaker, для комунікації між вузлами – corosync, також для легшого конфігурування було використано pcs. Перед початком роботи потрібно завантажити відповідні пакети з усіма залежностями.

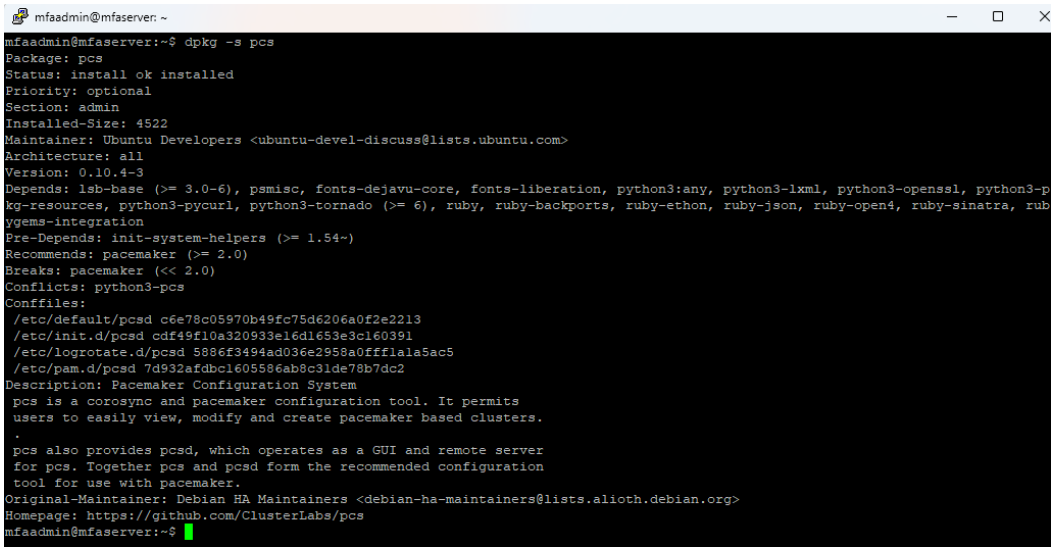


Рисунок 3.43 Встановлений пакет pcs

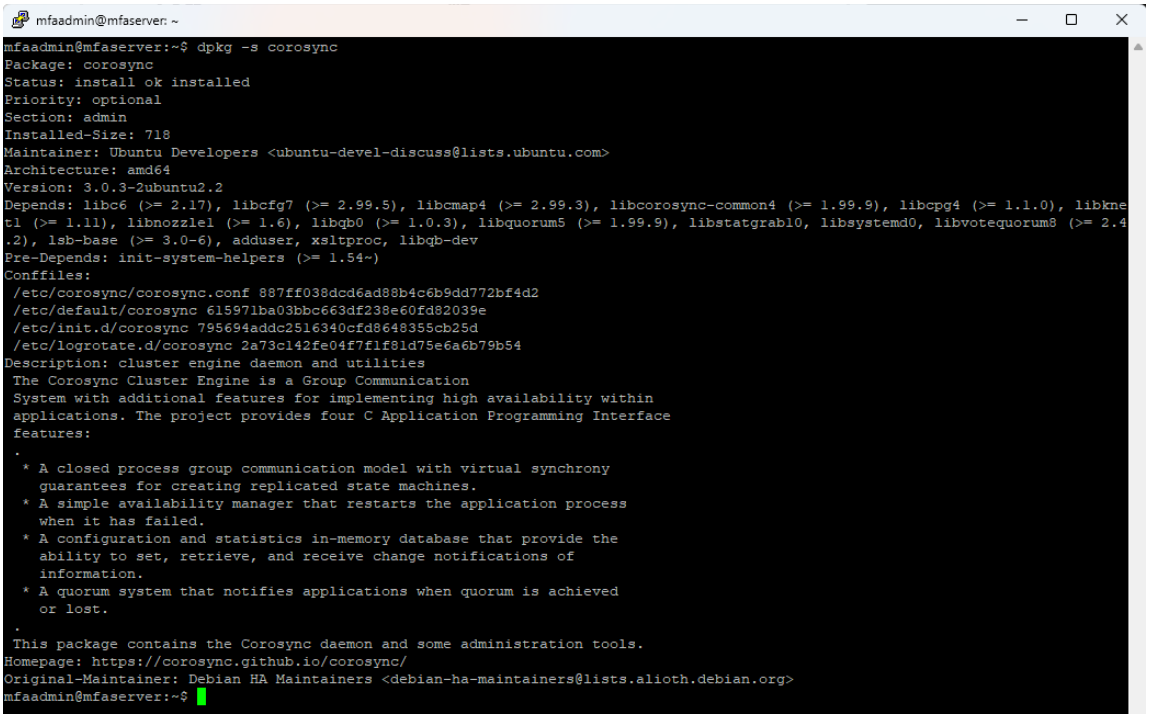


Рисунок 3.44 Встановлений пакет corosync

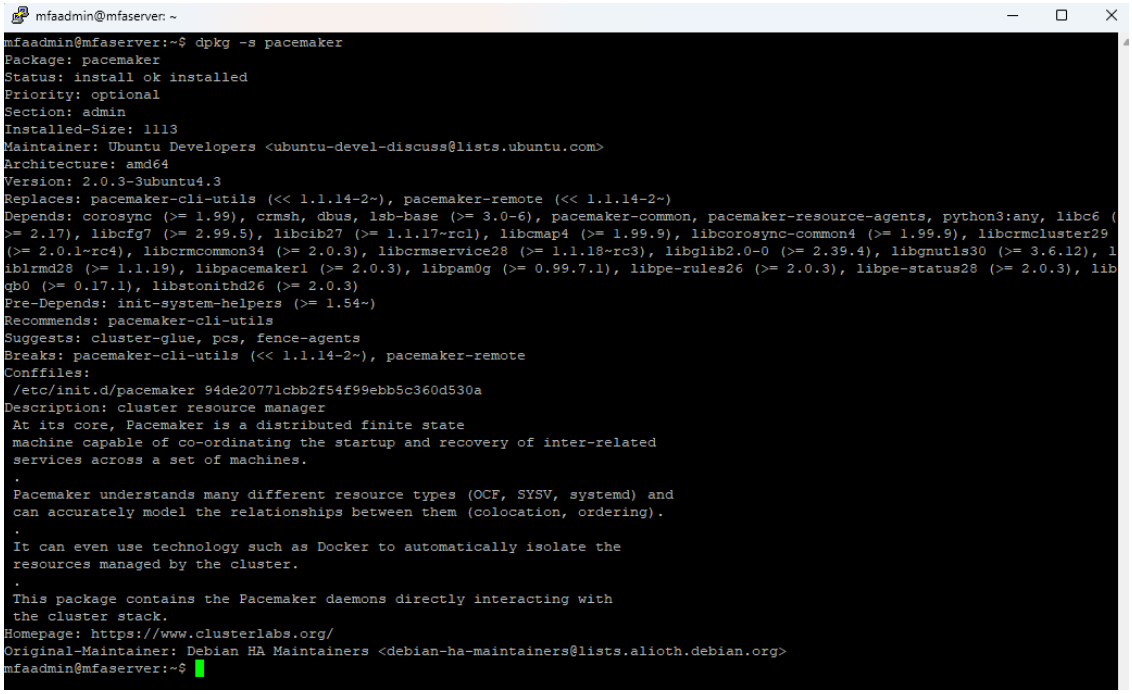


Рисунок 3.45 Встановлений пакет pacemaker

Увімкнемо службу pcsd та додамо її до автостарту, щоб вона вмикалася під час завантаження системи. Також для користувача hacluster на обох наших майбутніх нодах потрібно встановити однаковий пароль та налаштувати конфігураційний файл /etc/hosts.

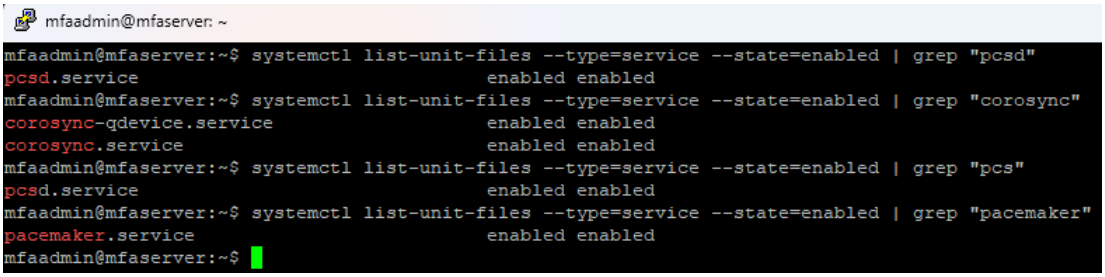


Рисунок 3.46 Перевірка увімкнення автозапуску для основних сервісів

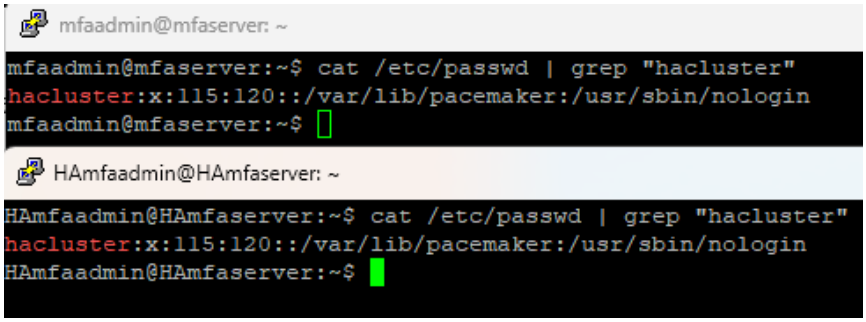


Рисунок 3.47 Налаштовані користувачі на серверах

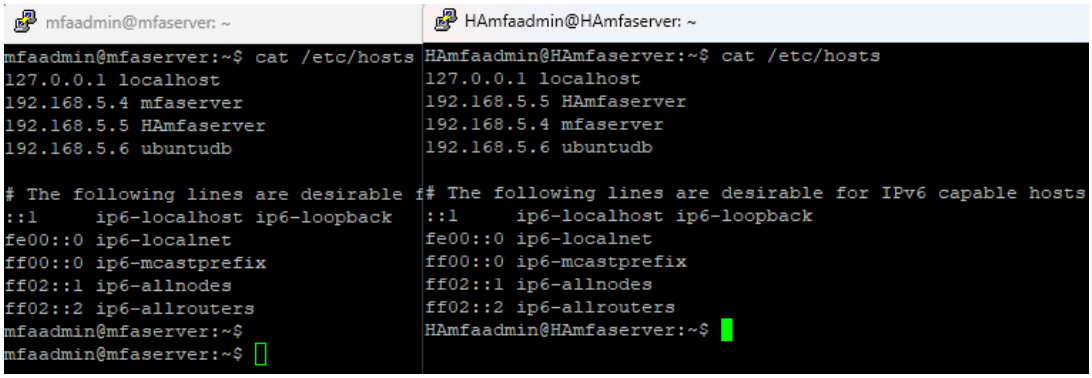


Рисунок 3.48 Вивід /etc/hosts

Наступним кроком буде створення самого кластера та відповідних ресурсів. Використовуючи команди, наведені нижче, ми авторизуємо наші хости, створимо кластер із назвою privacyidea_ha, а також створимо ресурси: віртуальну IP-адресу(VIP) для нодів та веб-сервіс Apache2. Саме цими ресурсами і буде керувати Pacemaker, контролюючи їх розташування між нодами.

sudo pcs host auth mfaadmin HAmfaadmin -u hacluster -p zt,fysqlfey

sudo pcs cluster setup privacyidea_ha mfaadmin HAmfaadmin

sudo pcs cluster start --all

sudo pcs cluster enable –all

Наразі STONITH буде вимкнено, а quorum policy встановлено в параметр ignore. У подальшому обидві ці функції будуть налаштовані.

```
sudo pcs property set stonith-enabled=false
sudo pcs property set no-quorum-policy=ignore
sudo pcs resource create vip ocf:heartbeat:IPaddr2 ip=192.168.5.7 cidr_netmask=28 op monitor interval=10s
sudo pcs resource create websvc systemd:apache2 op monitor interval=10s timeout=30s
sudo pcs resource group add pi_group vip websvc
sudo pcs constraint location pi_group prefers mfaadmin=100
```

За замовчуванням кожен із серверів PrivacyIDEA має свою локальну базу даних, але тепер вони знаходяться в кластері, і постає питання спільного доступу до ресурсів, оскільки при втраті одного сервера інший також повинен мати інформацію про новостворені токени користувачів. Для того щоб вирішити це питання, обидва сервери були підключені до зовнішньої бази даних PostgreSQL.



Рисунок 3.49 Створена база даних на UbuntuDB

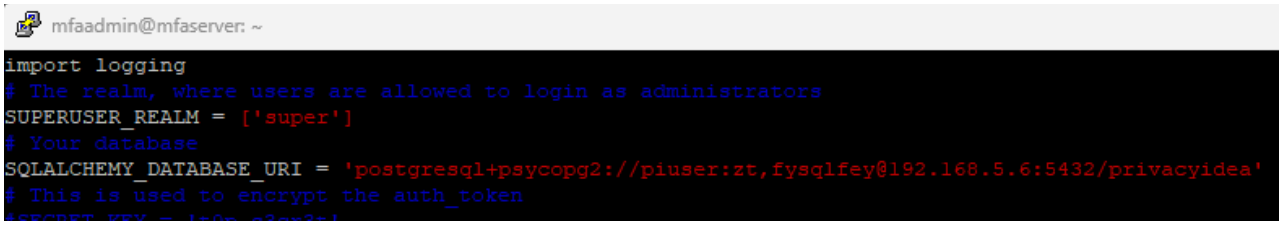


Рисунок 3.50 Підключення PrivacyIDEA до бази даних

Після виконаних налаштувань на обох серверах та застосування змін командою pi-manage db upgrade база даних для роботи системи оновлюється.

Наступним етапом буде конфігурація quorum девайса(witness). Використаємо пакет corosync-qnetd, також на кластерних нодах потрібно встановити corosync-qdevice для синхронізації процесів.

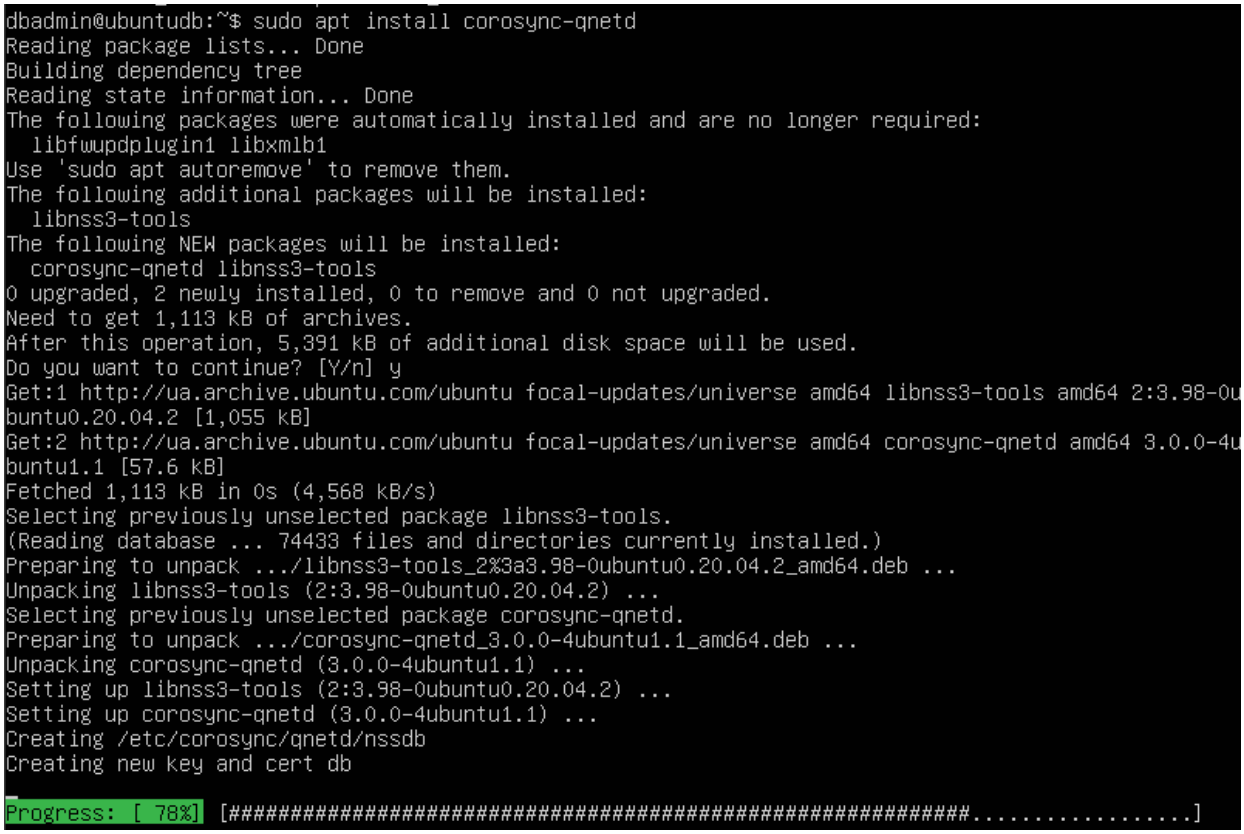


Рисунок 3.51 Встановлення corosync-qnetd

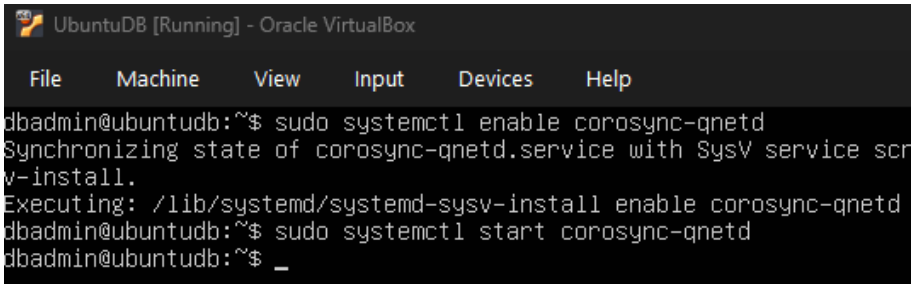


Рисунок 3.52 Додавання corosync-qnetd до списку автозапуску

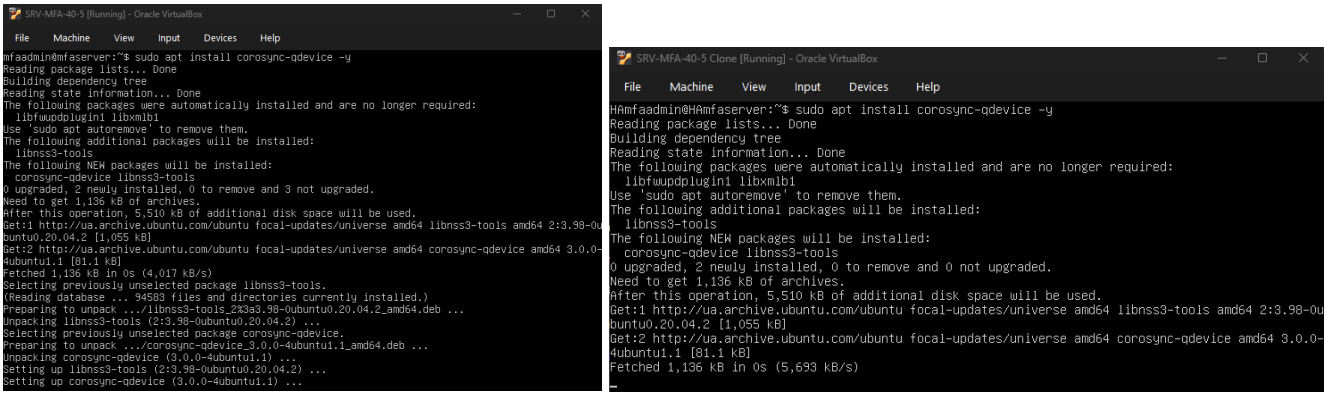


Рисунок 3.53 Встановлення corosync-qdevice на кластер нодах

На сервері свідка також потрібно встановити pcsd, додати його до автозапуску та налаштувати користувача hacluster згідно з налаштуваннями на серверах PrivacyIDEA.

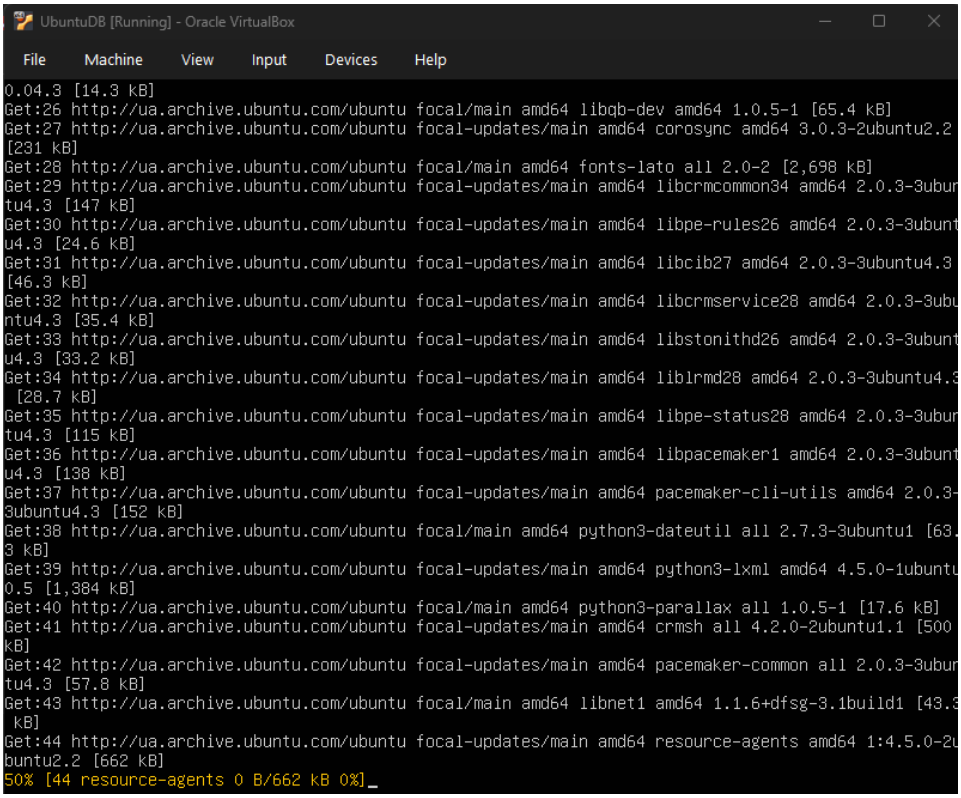


Рисунок 3.54 Встановлення pcsd

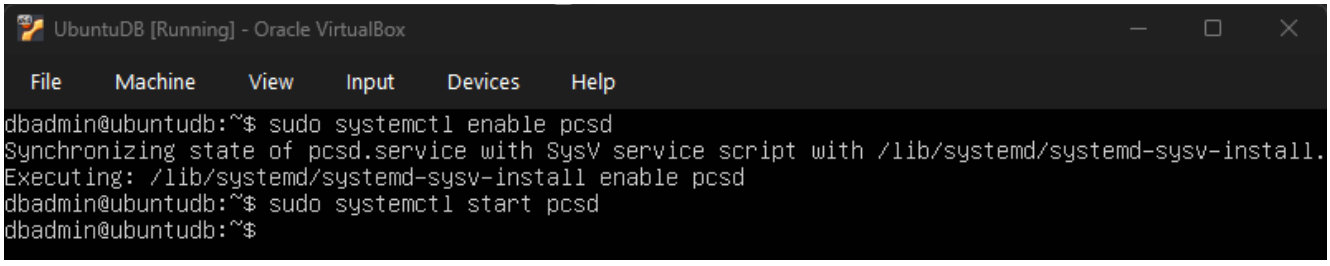


Рисунок 3.55 Додавання pcsd до автозапуску

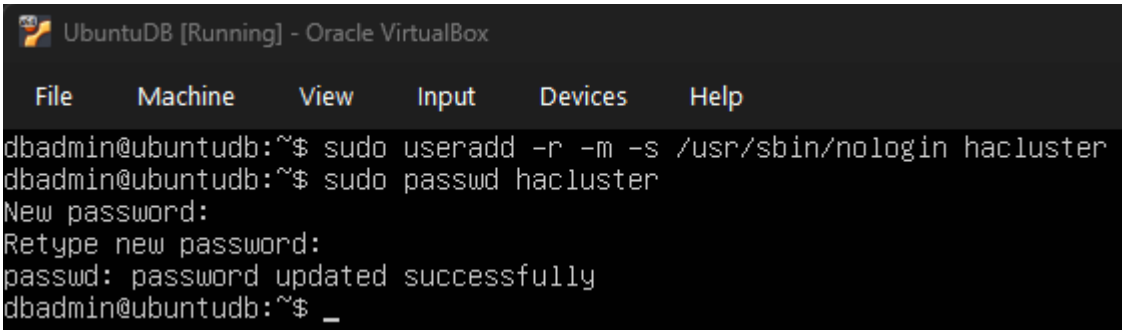


Рисунок 3.56 Налаштування користувача hacluster

На одному з нодів авторизуємо нашого свідка та додаємо його до системи, після чого змінюємо параметр no-quorum-policy на stop. У цій конфігурації, якщо доступу до кворуму не буде або його буде втрачено, і він не віддасть свій голос, усі ресурси будуть зупинені для уникнення ситуації split-brain.

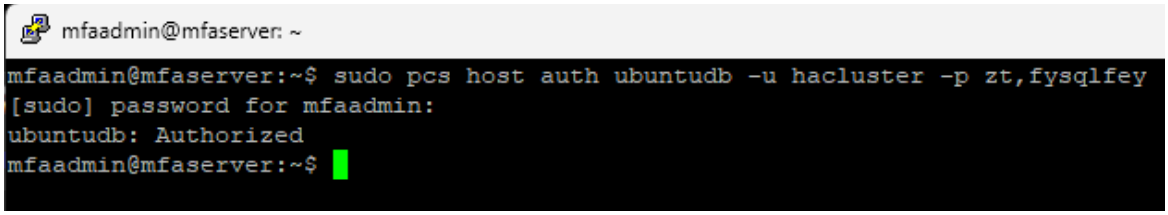


Рисунок 3.57 Авторизація хосту свідка

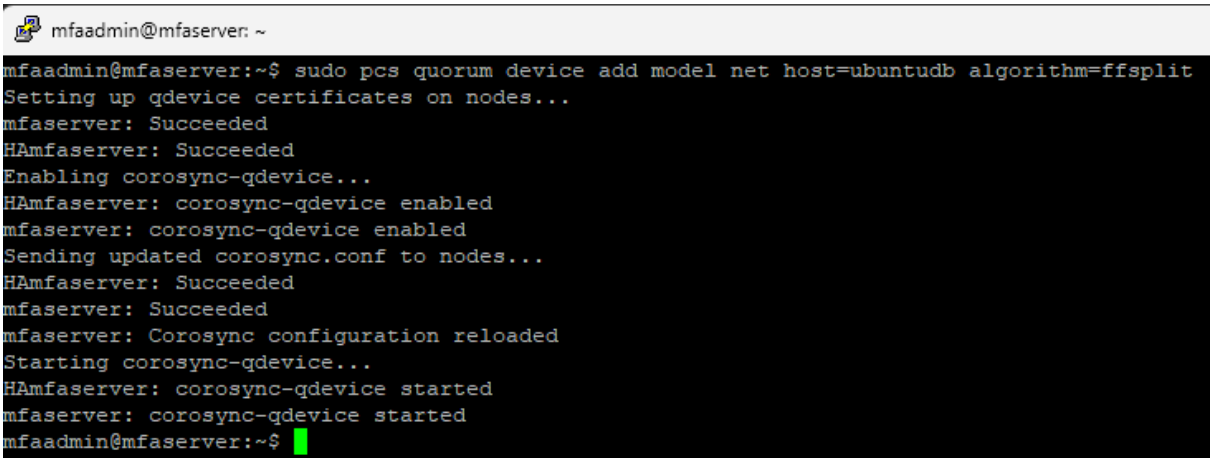


Рисунок 3.58 Додавання кворум девайсу до кластера

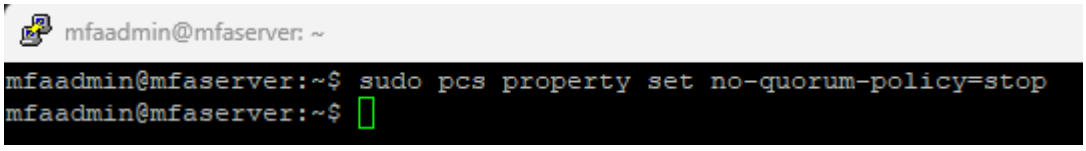


Рисунок 3.59 Налаштування параметру no-quorum-policy

Для реалізації STONITH було використано fence_vbox. fence_vbox – це фенсінг-агент, за допомогою якого віртуальні машини підключаються по SSH до хоста, на якому запущено VirtualBox, та виконують команди від імені гіпервізора.

Першим кроком буде перевірка зв'язку між віртуальними машинами та хост-машиною, для цього виконаємо ping до інтерфейсу VirtualBox Host-Only Ethernet Adapter.

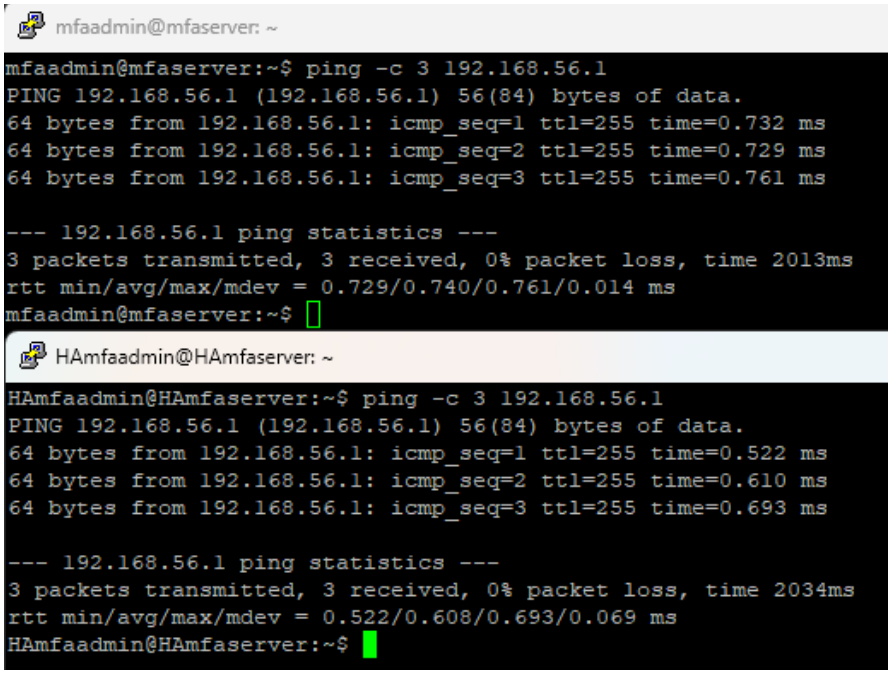


Рисунок 3.60 Перевірка зв'язку

Після перевірки успішно встановленого зв'язку на хост-машині потрібно розгорнути SSH-сервер, для цього буде використано OpenSSH. Також потрібно створити відповідне правило для Windows Firewall та на нодах встановити fence-agents.

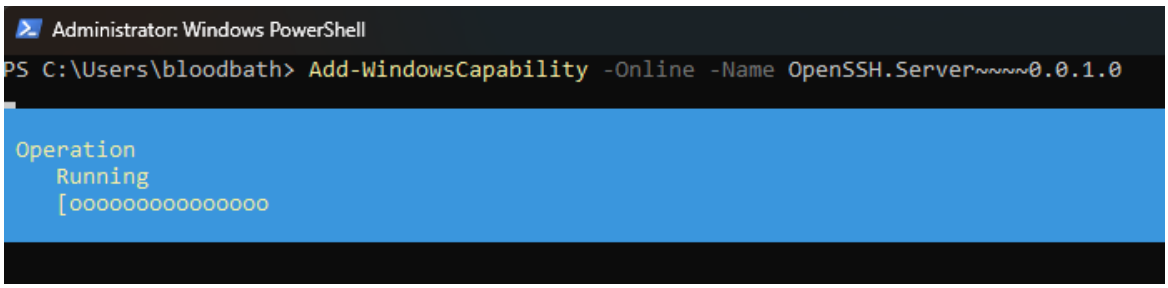


Рисунок 3.61 Встановлення OpenSSH Server

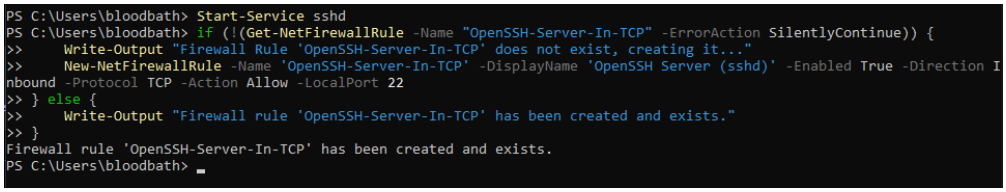


Рисунок 3.62 Створення правила для Windows Firewall

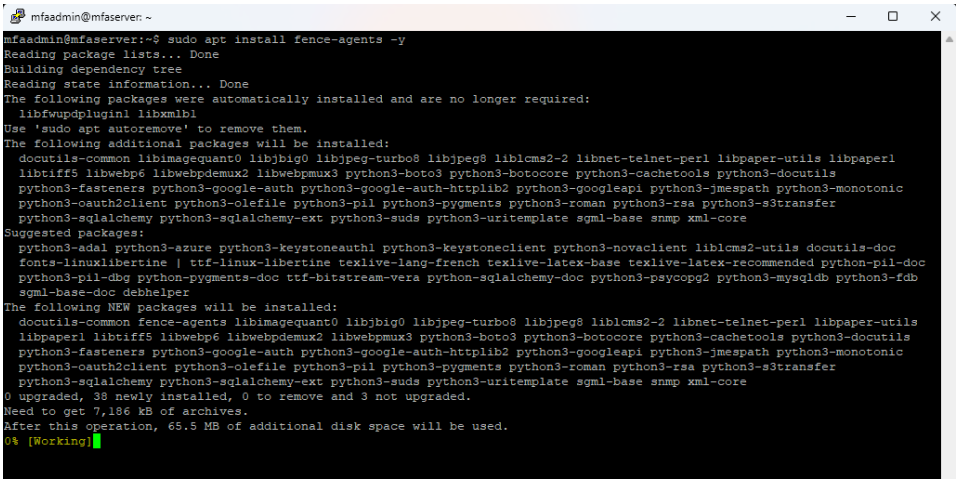


Рисунок 3.63 Встановлення fence-agents на SRV-MFA-40-5

Наступним кроком створимо користувачів на серверах, які будуть використовуватися для підключення до хост-машини, а також SSH ключі для уникнення парольного доступу.

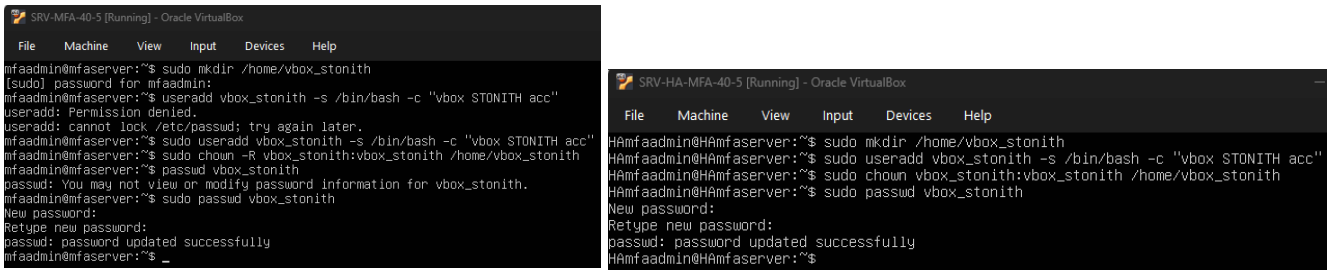


Рисунок 3.64 Створення облікового запису vbox_stonith

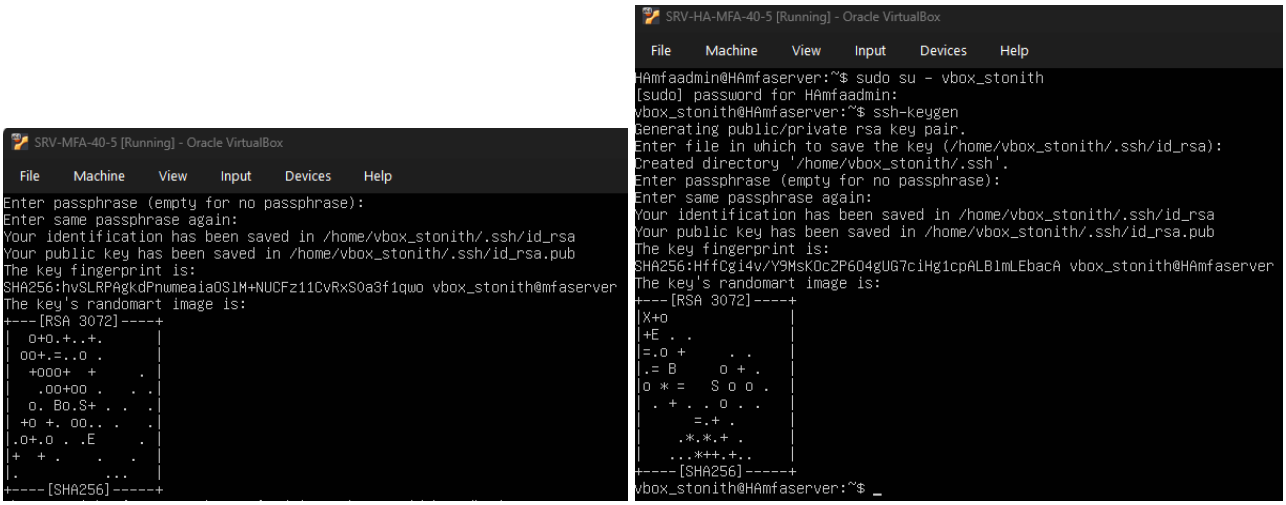


Рисунок 3.65 Генерація SSH ключів

Копіюємо вміст згенерованих ключів на хост-машину у файл authorized_keys та перевіряємо з'єднання однієї з нодів.



Рисунок 3.66 Скопійовані ключі на хост-машині

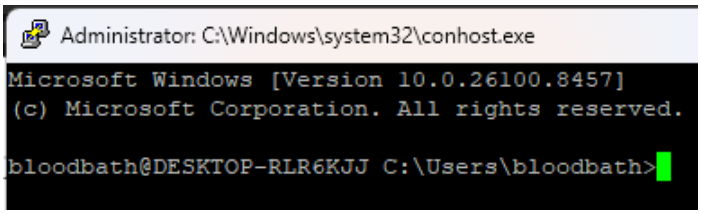


Рисунок 3.67 Виконання успішного SSH з'єднання

Останнім кроком буде додавання fencing-девайсів у кластер. Забороняємо ресурсам виконуватися на відповідних нодах, щоб уникнути «стрілянини» по самих собі, також виконуємо налаштування параметрів timeout та action.

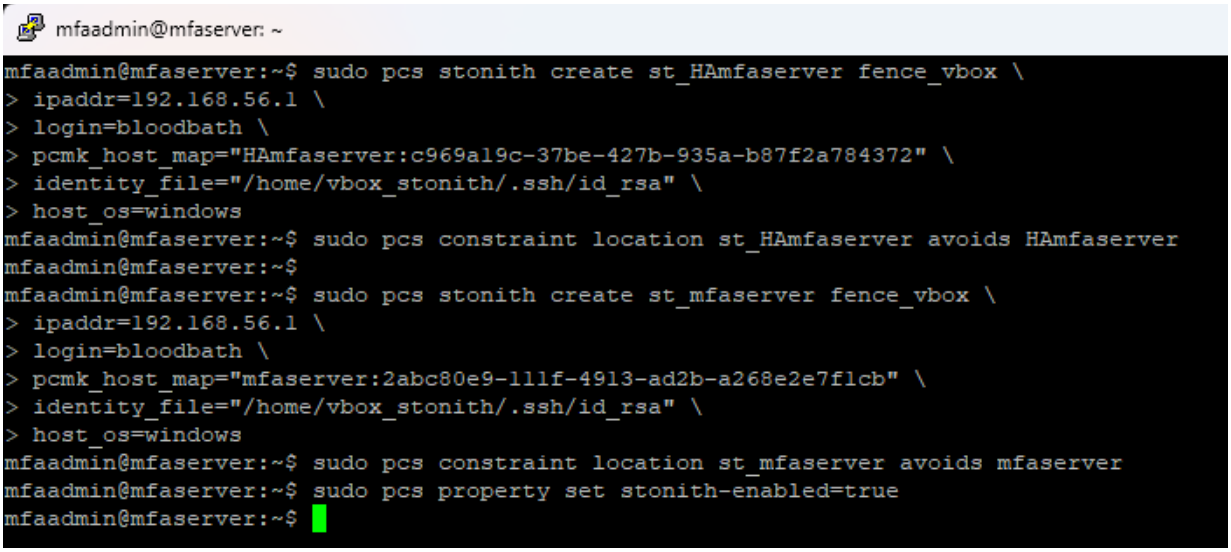


Рисунок 3.68 Додавання fencing-девайсів

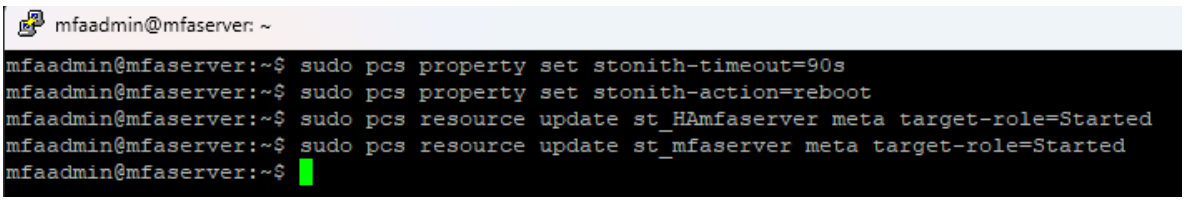


Рисунок 3.69 Налаштування параметрів timeout та action

Висновки до третього розділу

У третьому розділі було реалізовано впровадження та налаштування системи багатофакторної автентифікації PrivacyIDEA у тестовому середовищі. На початковому етапі виконано підготовку серверного середовища та встановлення сервісу PrivacyIDEA з урахуванням вимог до безпеки, надійності та подальшої масштабованості.

Для інтеграції PrivacyIDEA з Active Directory було створено окремого службового користувача з обмеженими правами доступу, що дозволило відмовитися від використання облікового запису адміністратора. Налаштовано взаємодію через LDAP-resolver та realm, у результаті чого забезпечено коректне отримання облікових записів користувачів з відповідних організаційних одиниць Active Directory.

Особливу увагу приділено захисту каналів зв'язку. Впроваджено захищене з'єднання з Active Directory за протоколом LDAPS з повною верифікацією сертифіката доменного ЦС, а також реалізовано доступ до веб-інтерфейсу PrivacyIDEA виключно по HTTPS з використанням самопідписаного сертифіката. Це дозволило усунути передачу конфіденційних даних у відкритому вигляді та забезпечити їхню цілісність і конфіденційність.

Завершальним етапом виконано налаштування механізмів багатофакторної автентифікації, включно з політиками автентифікації та контролем кількості невдалих спроб входу. Також було побудовано відмовостійкий кластер із двох серверів PrivacyIDEA із дотриманням ключових вимог безпеки та високої доступності. Налаштування quorum device і механізму STONITH дозволило забезпечити коректну роботу кластера навіть у разі відмови одного з вузлів, що підвищує надійність та стабільність системи багатофакторної автентифікації.

РОЗДІЛ 4. ТЕСТУВАННЯ ПРАЦЕЗДАТНОСТІ СИСТЕМИ

4.1. Тестування працездатності кластера

Для початку перевіримо робочий стан кластера. Це включає перевірку доступності вузлів, стану кворуму та стану налаштованих ресурсів.

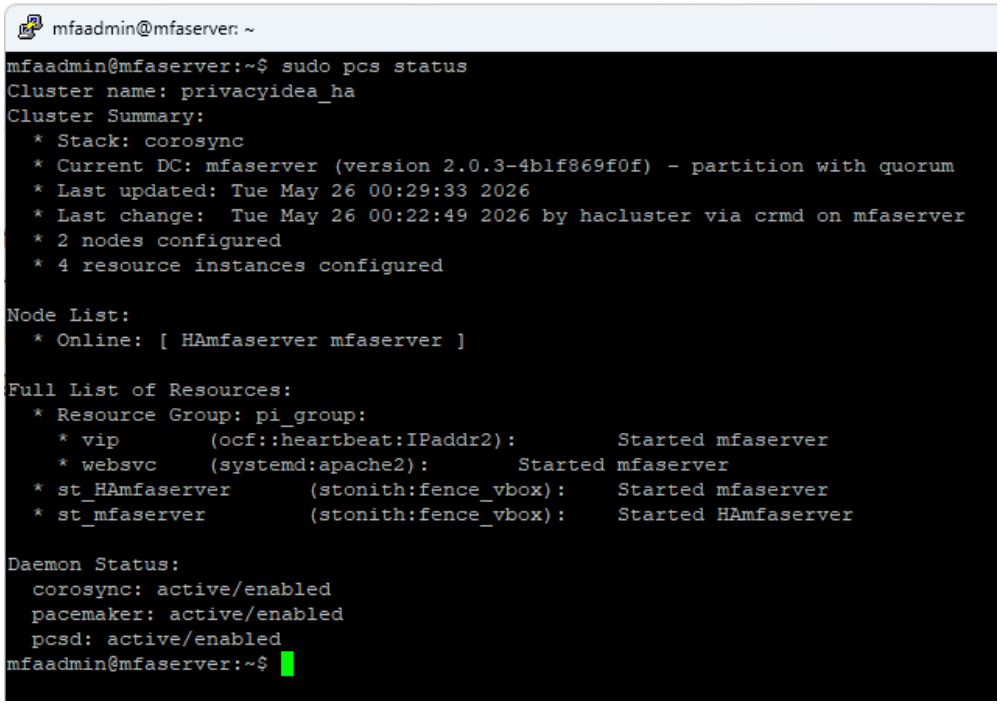


Рисунок 4.1 Виконання команди pcs status на SRV-MFA-40-5

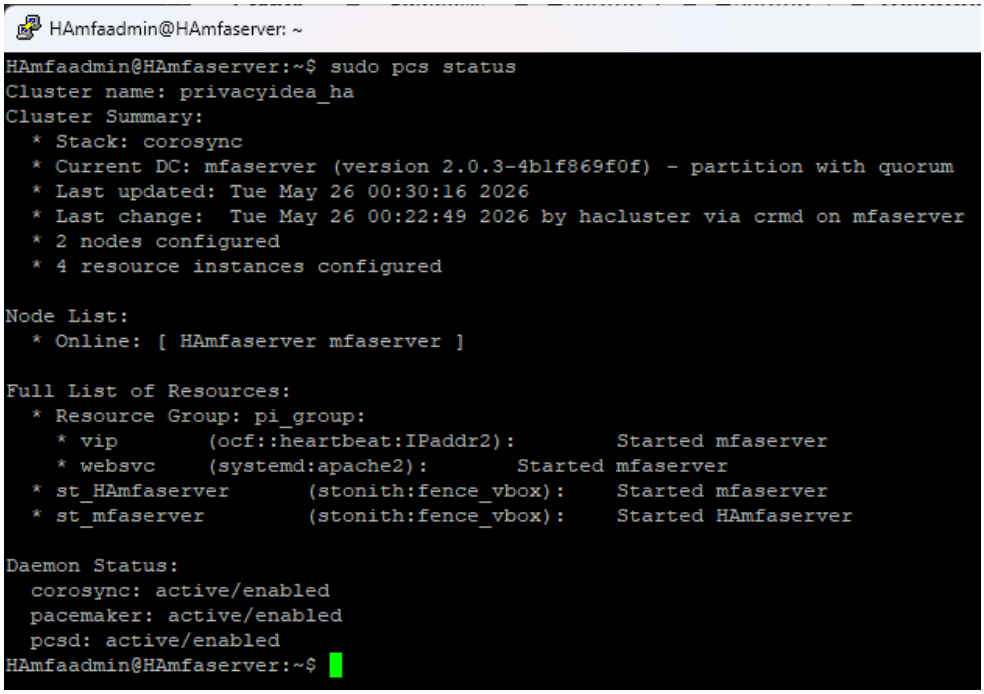


Рисунок 4.2 Виконання команди pcs status на SRV-HA-MFA-40-5

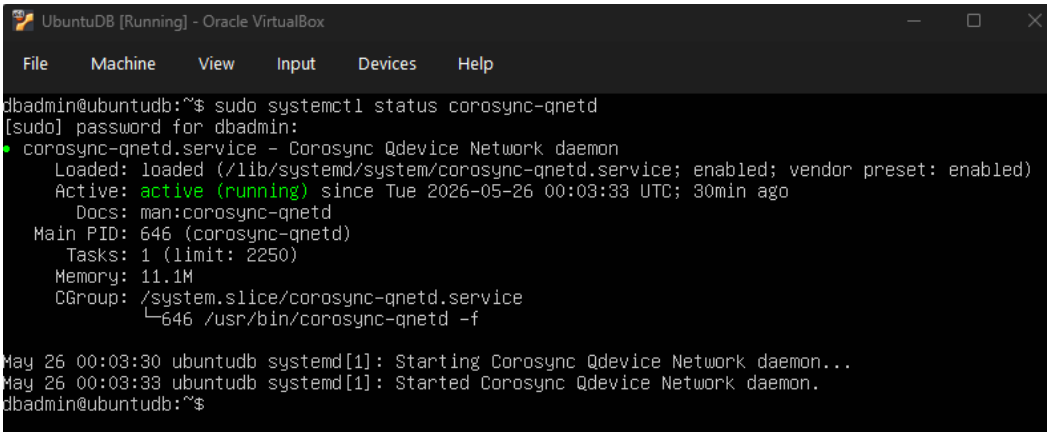


Рисунок 4.3 Перевірка статусу служби corosync-qnetd на UbuntuDB

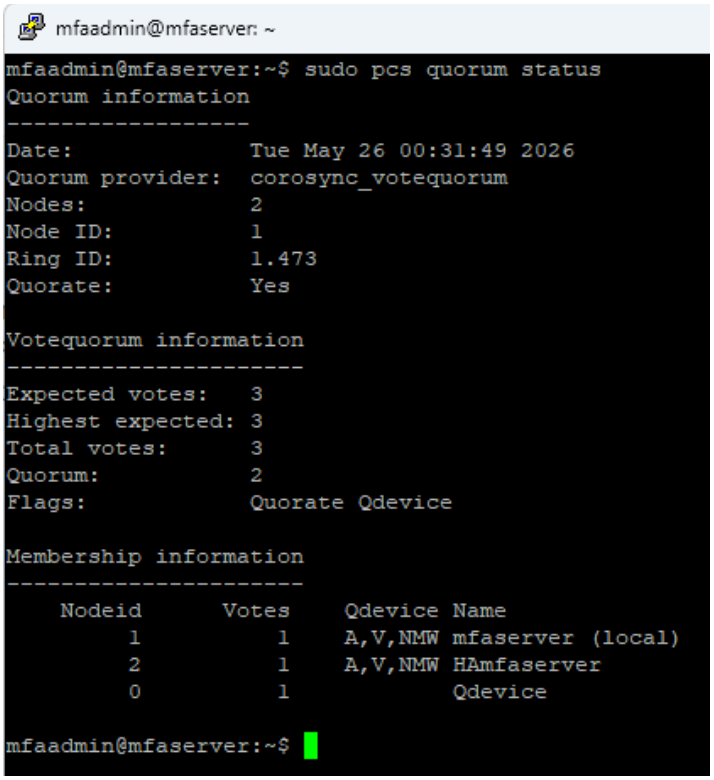


Рисунок 4.4 Перевірка роботи quorum на SRV-MFA-40-5

Для перевірки працездатності високої доступності проводимо тестування переходу ресурсів на резервний вузол. На одному з вузлів кластера навмисно вимкнемо службу pacemaker та перевіримо переміщення ресурсів на резервний сервер.

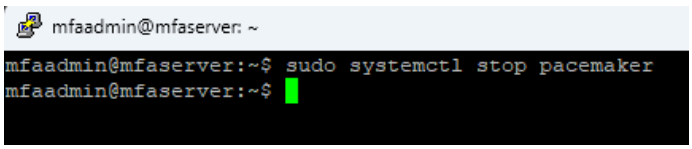


Рисунок 4.5 Зупинка роботи pacemaker на SRV-MFA-40-5

В результаті зупинки Pacemaker всі ресурси, які були запущені на ноді, переходять на резервний сервер, та користувачі не побачать проблем у роботі системи автентифікації.

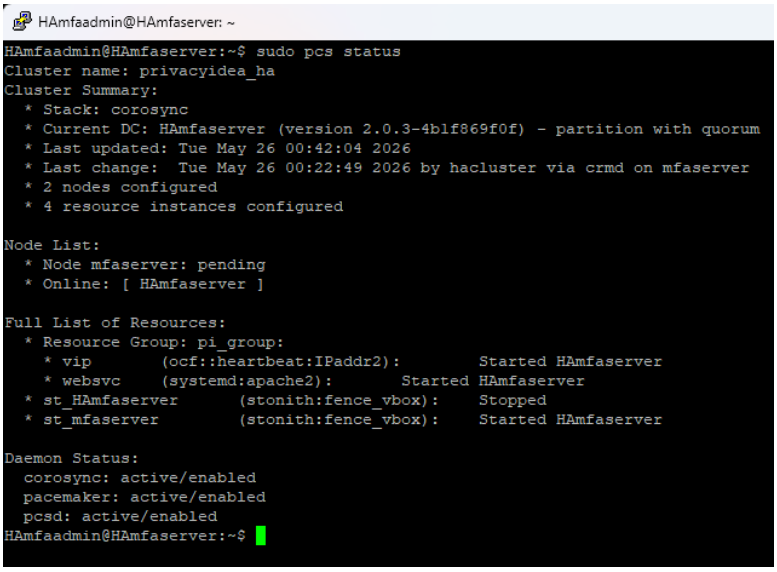


Рисунок 4.6 Успішна міграція ресурсів на резервний сервер

У даному сценарії не буде застосовано STONITH, оскільки ручне відключення служби імітує повне вимкнення SRV-MFA-40-5. Після повернення служби до робочого стану всі ресурси знову перейдуть на основний сервер, через те що він має вищий пріоритет.

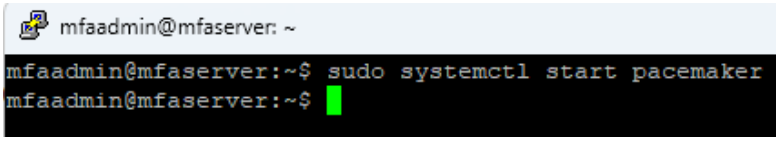


Рисунок 4.7 Запуск роботи pacemaker на SRV-MFA-40-5

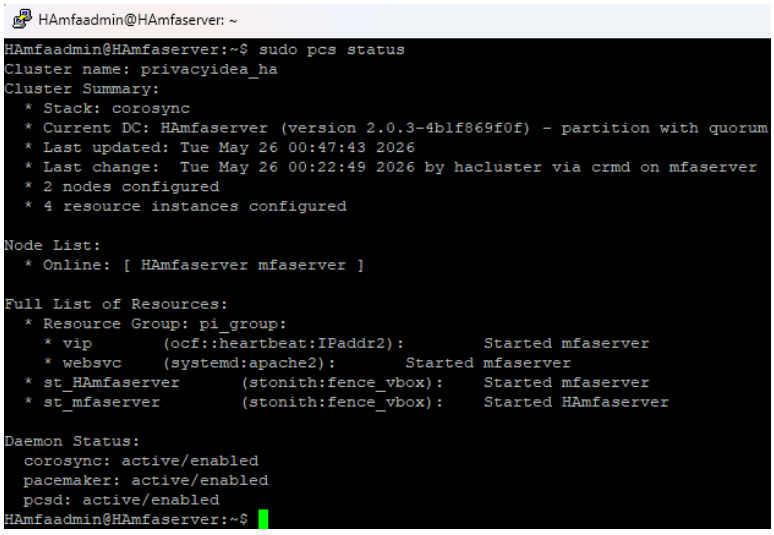


Рисунок 4.8 Успішна міграція ресурсів на основний сервер

Щоб перевірити саму роботу кворуму, нам доведеться зупинити сервер SRV-MFA-40-5. У даній ситуації в нас буде лише 1 активна нода, тому при перевірці pcs quorum status очікуємо Total Votes: 2.

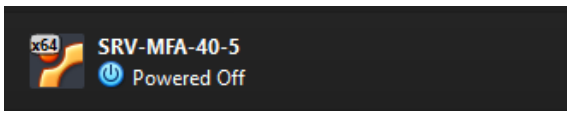


Рисунок 4.9 Зупинений сервер SRV-MFA-40-5

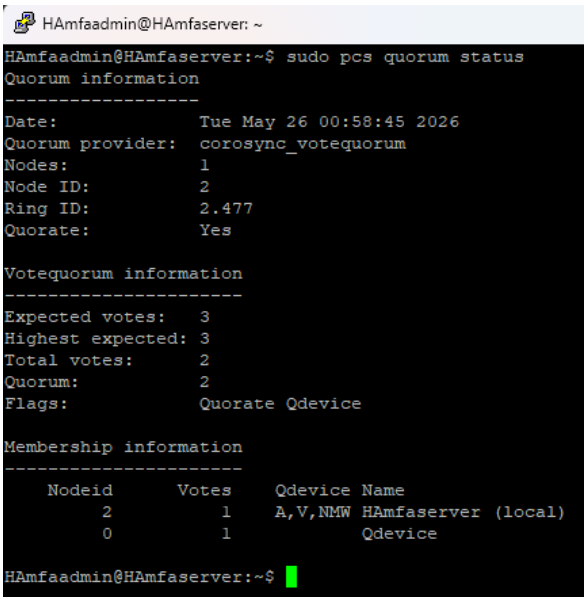


Рисунок 4.10 Перевірка роботи кворуму

Також важливо перевірити налаштовану політику no-quorum-policy=stop. Якщо сервери втратять зв’язок зі свідком та між собою, всі ресурси повинні бути зупинені, щоб уникнути ситуації split-brain.

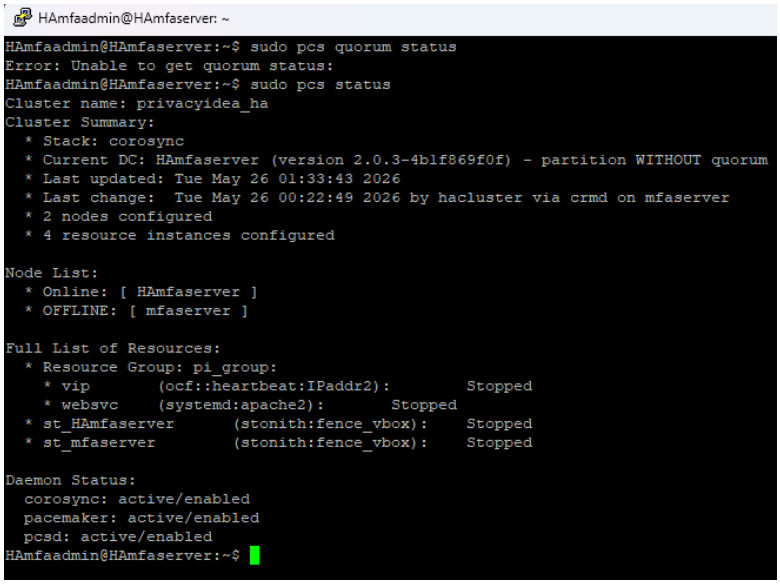


Рисунок 4.11 Зупинені ресурси після втрати зв’язку з свідком

Останнім кроком буде перевірка роботи STONITH. Для цього «вб'ємо» процес corosync на одному з нодів, щоб ініціювати раптову втрату зв'язку між нодами. В результаті цієї перевірки віртуальна машина, зв'язок з якою було втрачено, повинна перезапуститися та під'єднатися назад до кластера.

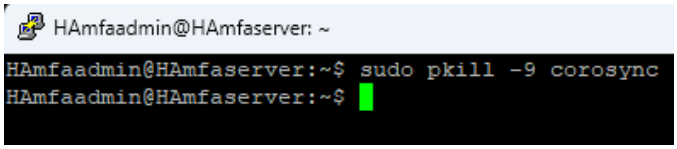


Рисунок 4.12 Завершення процесу corosync

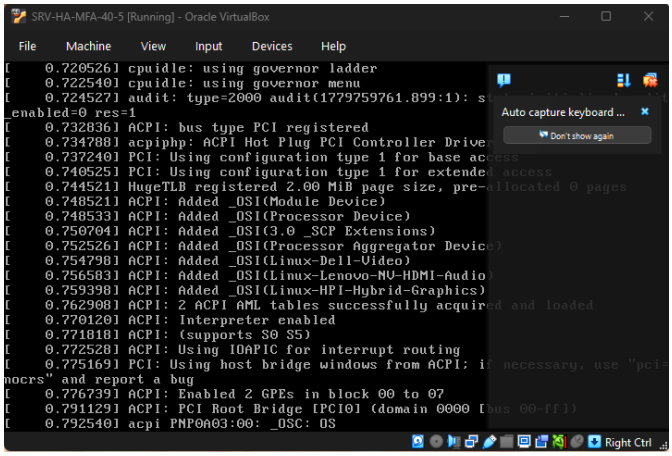


Рисунок 4.13 Успішний перезапуск віртуальної машини

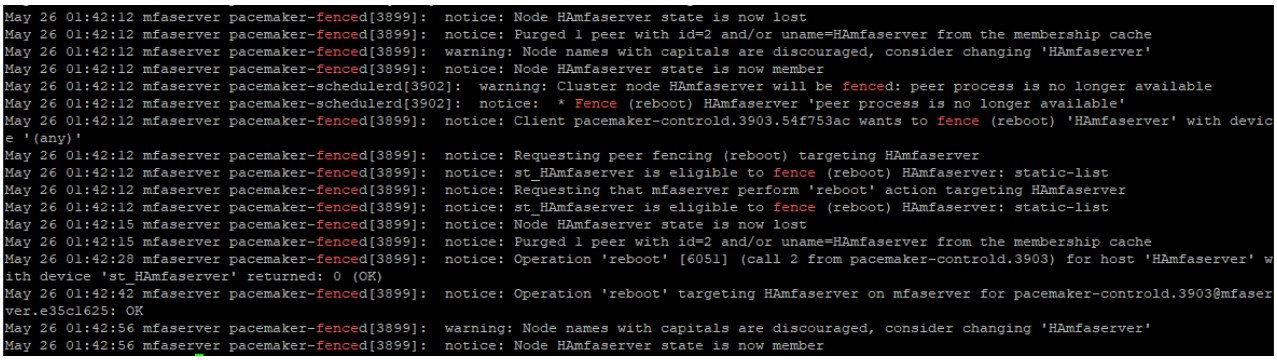


Рисунок 4.14 Перевірка логів про виконання STONITH

4.2. Тестування працездатності багатофакторної автентифікації

Виконаємо успішну автентифікацію для тестування роботи системи та перевірки логування подій.

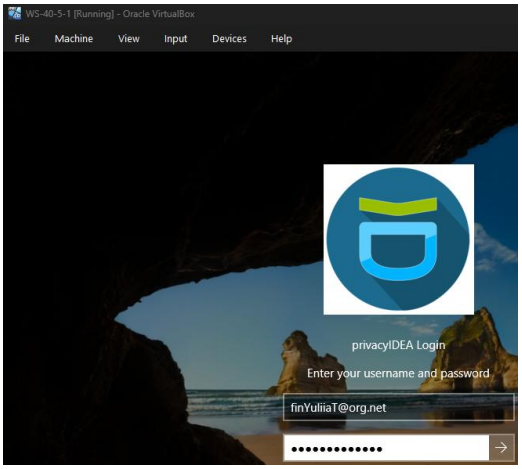


Рисунок 4.15 Виконання даних облікового запису

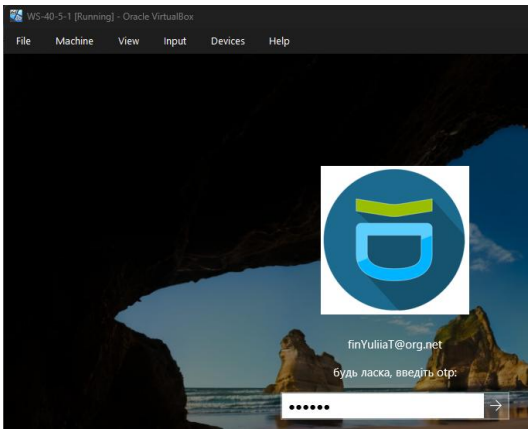


Рисунок 4.16 Введення TOTP коду

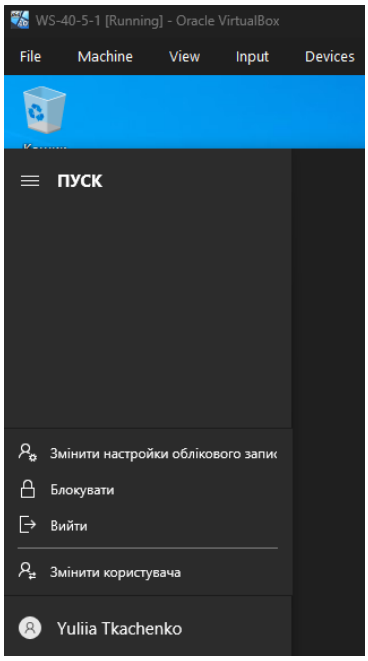


Рисунок 4.17 Успішний вхід

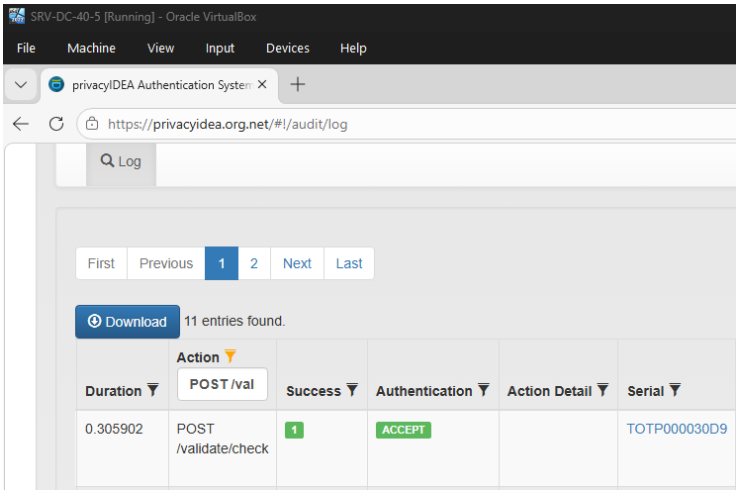


Рисунок 4.18 Перевірка логів про успішний вхід

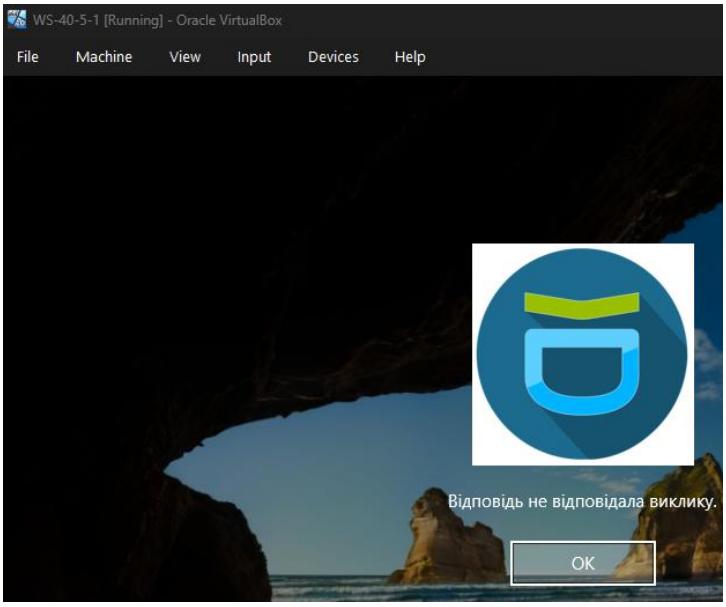


Рисунок 4.19 Невірно введений TOTP код

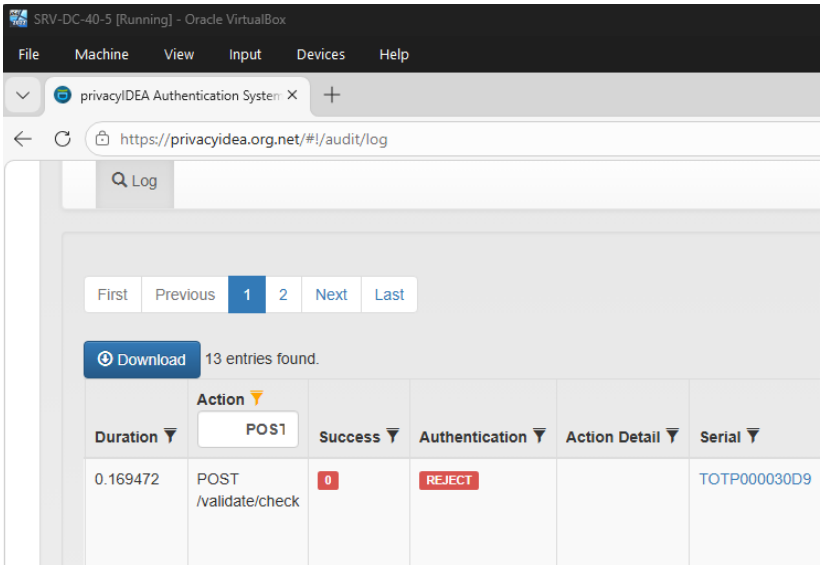


Рисунок 4.20 Перевірка логів про відхилений запит на вхід

Після трьох неправильних спроб введення коду токен перестає приймати навіть правильні коди та не може бути використаним, поки адміністратор не анулює fail counter. Також було додатково створено обробник подій, який спрацьовує, коли fail counter = 3, та переводить токен у стан disabled.

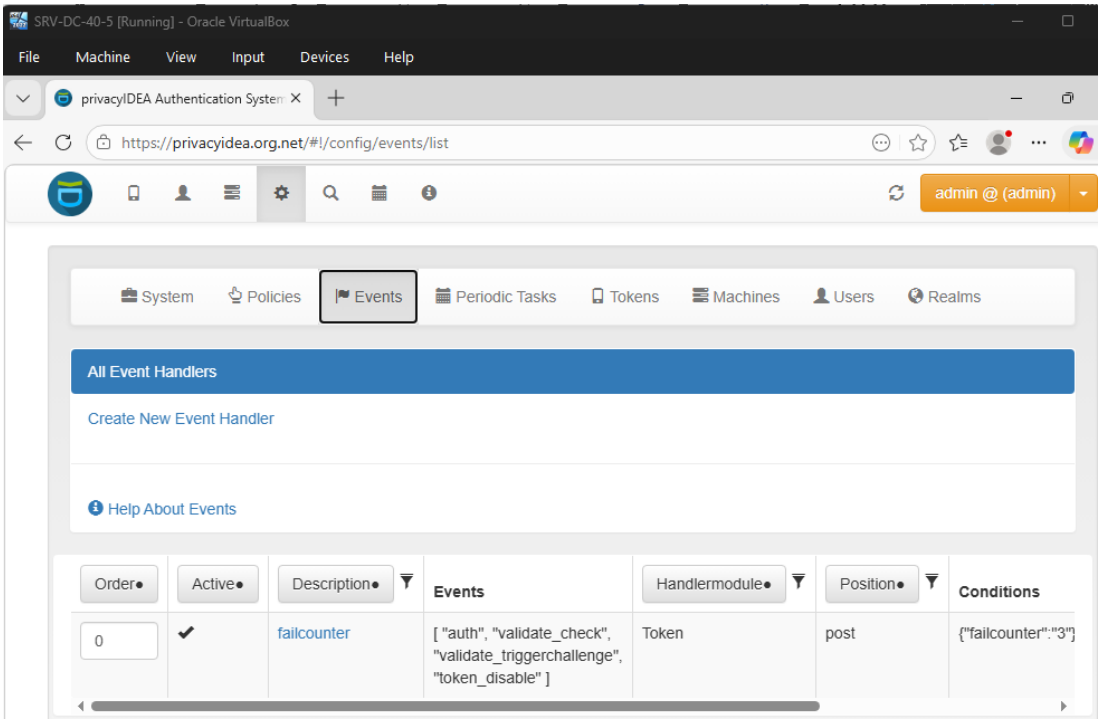


Рисунок 4.21 Створена подія

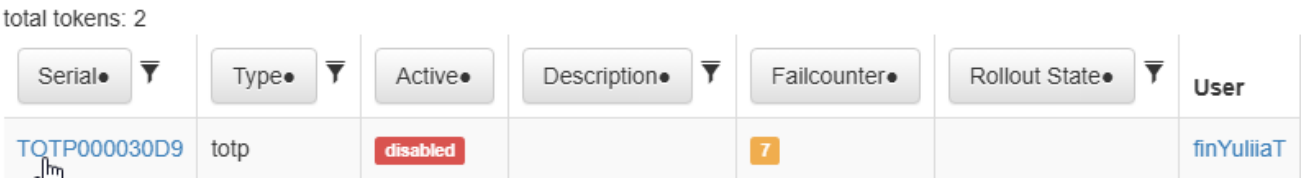


Рисунок 4.22 Перевірка роботи події

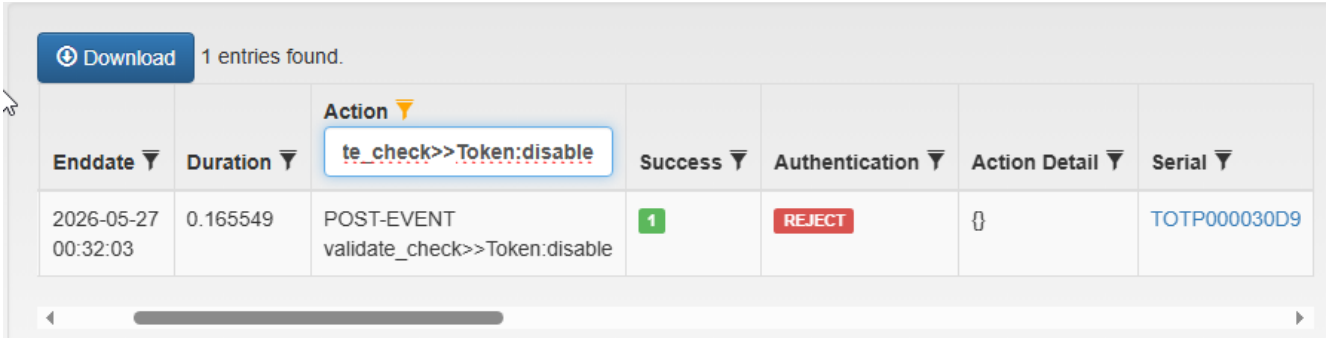


Рисунок 4.23 Перевірка логів про виконання події

Висновки до четвертого розділу

У четвертому розділі було проведено практичне тестування реалізованої системи багатофакторної автентифікації та відмовостійкого кластерного середовища. Під час тестування підтверджено коректну роботу MFA-автентифікації, успішну інтеграцію PrivacyIDEA з Active Directory, а також правильність роботи механізмів синхронізації та перевірки облікових записів користувачів.

Окрему увагу було приділено перевірці працездатності HA-кластера, побудованого на базі open source рішень Pacemaker та Corosync. У ході тестування підтверджено коректну роботу механізмів кворуму, автоматичного перенесення ресурсів між вузлами, функціонування QDevice та забезпечення безперервності роботи сервісу MFA у випадку відмови одного з серверів.

Результати тестування підтвердили можливість побудови ефективної та відмовостійкої системи багатофакторної автентифікації в середовищі Windows Server із використанням open source програмних рішень, що відповідає вимогам інформаційної безпеки та забезпечення безперервності сервісів автентифікації.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто основної мети – досліджено теоретичні та практичні аспекти впровадження кластеризованої багатофакторної автентифікації в корпоративних середовищах. Розглянуто основні поняття та принципи MFA, кластеризації та теоретичні основи функціонування системи PrivacyIDEA, а також виконано аналіз альтернативних рішень багатофакторної автентифікації з їх порівняльною характеристикою. Це дозволило обґрунтувати вибір PrivacyIDEA як гнучкого та функціонального рішення для інтеграції з корпоративною інфраструктурою.

Було виконано проєктування та підготовку тестового середовища, описано віртуальну інфраструктуру, розроблено схему IP-адресації та здійснено базове налаштування серверної інфраструктури. Підготовлене середовище забезпечило необхідні умови для подальшої інтеграції системи MFA та кластера високої доступності.

У межах практичної частини було безпосередньо впроваджено багатофакторну автентифікацію. Виконано встановлення та початкове налаштування системи PrivacyIDEA, інтеграцію з Active Directory із використанням службового облікового запису, а також налаштування resolver'ів і realm'ів. Окрему увагу приділено захисту каналів взаємодії шляхом впровадження захищених з'єднань за протоколами LDAPS та HTTPS із використанням самопідписаних сертифікатів, а також особливостям налаштування кластера на базі операційної системи Linux.

У процесі тестування було перевірено коректність роботи механізмів багатофакторної автентифікації на основі одноразових паролів TOTP, обробку успішних і неуспішних спроб входу, роботу лічильників помилок та автоматичне блокування токенів відповідно до налаштованих політик безпеки. Також протестовано ключові механізми функціонування кластера, зокрема керування ресурсами, забезпечення кворуму та роботу STONITH. Отримані результати підтверджують, що реалізована система є функціональною, безпечною та придатною для адаптації до потреб реального корпоративного середовища.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

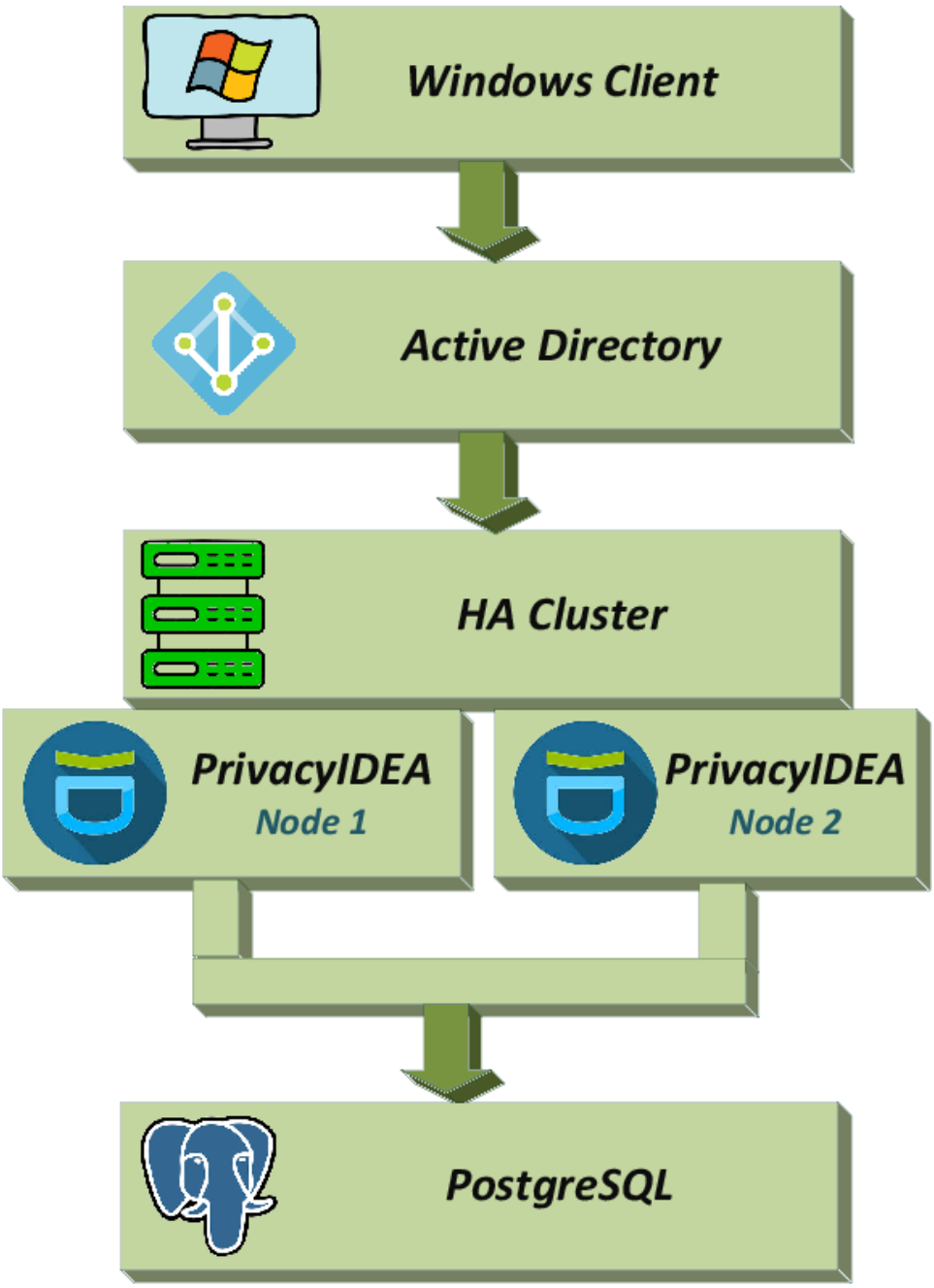
1. privacyIDEA Documentation [Електронний ресурс] - Режим доступу: https://privacyidea.readthedocs.io/en/latest/
2. Time-Based One-Time Password Algorithm (TOTP) [Електронний ресурс] - Режим доступу: https://datatracker.ietf.org/doc/html/rfc6238
3. The Transport Layer Security (TLS) Protocol Version 1.2 [Електронний ресурс] - Режим доступу: https://datatracker.ietf.org/doc/html/rfc5246
4. Active Directory Domain Services Overview [Електронний ресурс] - Режим доступу: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview
5. Active Directory Certificate Services Overview [Електронний ресурс] - Режим доступу: https://learn.microsoft.com/en-us/windows-server/identity/ad-cs/active-directory-certificate-services-overview
6. Credential providers in Windows [Електронний ресурс] - Режим доступу: https://learn.microsoft.com/en-us/windows/win32/secauthn/credential-providers-in-windows
7. Ubuntu Security and certificates [Електронний ресурс] - Режим доступу: https://documentation.ubuntu.com/server/explanation/security/certificates/index.html
8. LDAP Authentication Methods and Security Mechanisms [Електронний ресурс] - Режим доступу: https://datatracker.ietf.org/doc/html/rfc4513
9. Multifactor Authentication Cheat Sheet [Електронний ресурс] - Режим доступу:https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html
10. Digital Identity Guidelines: Authentication and Lifecycle Management [Електронний ресурс] - Режим доступу: https://pages.nist.gov/800-63-3/sp800-63b.html
11. Pacemaker Explained [Електронний ресурс] - Режим доступу: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/pdf/Pacemaker_Explained.pdf

12. Configuring and managing high availability clusters [Електронний ресурс] - Режим доступу: https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/configuring_and_managing_high_availability_clusters/index
13. OpenSSH for Windows [Електронний ресурс] - Режим доступу: https://learn.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse?tabs=powershell&pivots=windows-11
14. Linux Clustering Concepts [Електронний ресурс] - Режим доступу: https://docs.rackspace.com/docs/clustering-concepts

ДОДАТКИ

Додаток А

Схема взаємодії компонентів системи MFA



Додаток Б

Адаптований код fence_vbox

Лістинг програми:

```
#!/usr/bin/python3 -tt

# The Following Agent Has Been Tested On:
#
# VirtualBox 5.0.4 x64 on openSUSE 13.2
#

import sys
import re
import time
import atexit
sys.path.append("/usr/share/fence")

from fencing import *
from fencing import fail_usage

def _invoke(conn, options, *cmd):
    prefix = options["--sudo-path"] + " " if "--use-sudo" in options else ""
    full_cmd = prefix + options["--vboxmanage-path"] + " " + " ".join(cmd)

    conn.send_eol(full_cmd)

    timeout = 15 if any(x in cmd for x in ["startvm", "controlvm"]) else int(op-
tions.get("--shell-timeout", 8))

    try:
        conn.log_expect(options["--command-prompt"], timeout)
    except:
        pass

    raw = ""
    if hasattr(conn, 'before') and conn.before:
        raw += str(conn.before)
    if hasattr(conn, 'after') and conn.after:
        raw += str(conn.after)

    raw = re.sub(r'\x1b\][^\x07]*\x07', '', raw)
    raw = re.sub(r'\x1b\[[0-9;]*[a-zA-Z]', '', raw)

    return raw

def get_outlets_status(conn, options):
    result = {}
    output = _invoke(conn, options, "list", "vms")

    for line in output.splitlines():
```

```
        line = line.strip()
        if not line or '"' not in line or '{' not in line:
            continue

        match = re.search(r'"([^"]+)"\s*\{([0-9a-fA-F-]+)\}', line)
        if match:
            vm_name = match.group(1).strip()
            vm_uuid = match.group(2).strip()
            result[vm_name] = (vm_uuid, "off")

    return result

def get_power_status(conn, options):
    outlets = get_outlets_status(conn, options)
    plug = options["--plug"].strip()
    plug_lower = plug.lower()

    for name, (uuid, _) in outlets.items():
        if plug_lower == name.strip().lower() or plug_lower == uuid.lower():
            try:
                out = _invoke(conn, options, "showvminfo", f'"{plug}"', "--machiner-
eadable")
                if 'VMState="running"' in out or 'VMState=running' in out:
                    return "on"
            except:
                pass
            return "off"

    if "--missing-as-off" in options:
        return "off"

    fail_usage(f"Failed: VM '{plug}' not found!")

def set_power_status(conn, options):
    plug = options["--plug"]
    action = options["--action"]

    print(f"DEBUG: Action '{action}' on '{plug}'")

    if action == "on":
        _invoke(conn, options, "startvm", f'"{plug}"')
        time.sleep(4)

    elif action == "off":
        _invoke(conn, options, "controlvm", f'"{plug}"', "poweroff")
        time.sleep(3)

    elif action == "reboot":
        _invoke(conn, options, "controlvm", f'"{plug}"', "reset")
        time.sleep(3)
```

```
    else:
        fail_usage(f"Unknown action: {action}")

def define_new_opts():
    all_opt["vboxmanage_path"] = {
        "getopt" : ":",
        "longopt" : "vboxmanage-path",
        "help" : "--vboxmanage-path=[path]  Path to VBoxManage on the host",
        "required" : "0",
        "shortdesc" : "Path to VBoxManage on the host",
        "default" : "VBoxManage",
        "order" : 200
    }
    all_opt["host_os"] = {
        "getopt" : ":",
        "longopt" : "host-os",
        "help" : "--host-os=[os]  Operating system of the host",
        "required" : "0",
        "shortdesc" : "Operating system of the host",
        "choices" : ["linux", "macos", "windows"],
        "default" : "windows",
        "order" : 200
    }

def main():
    device_opt = ["ipaddr", "login", "passwd", "cmd_prompt", "secure", "port",
"sudo",
    "missing_as_off", "vboxmanage_path", "host_os"]
    define_new_opts()

    atexit.register(atexit_handler)

    all_opt["secure"]["default"] = "1"

    all_opt["cmd_prompt"]["default"] = [r"\[EXPECT\]#\ "]
    all_opt["ssh_options"]["default"] = "-t '/bin/bash -c \"" + r"PS1=\\[EX-
PECT\\]#\  " + "/bin/bash --noprofile --norc\"'"

    opt = process_input(device_opt)

    if "--host-os" not in opt or opt["--host-os"] is None:
        opt["--host-os"] = "windows"

    opt["logout_string"] = "quit"
    if "--host-os" in opt and "--vboxmanage-path" not in opt:
        if opt["--host-os"] == "linux":
            opt["--vboxmanage-path"] = "VBoxManage"
        elif opt["--host-os"] == "macos":
            opt["--vboxmanage-path"] = "/Applications/VirtualBox.app/Contents/Ma-
cOS/VBoxManage"
            opt["logout_string"] = "exit"
```

```
        elif opt["--host-os"] == "windows":
            opt["--vboxmanage-path"] = '"C:\\Program Files\\Oracle\\VirtualBox\\VBox-
Manage.exe"'
            opt["--command-prompt"] = [r".*>"]
            opt["--ssh-options"] = ""

    options = check_input(device_opt, opt)
    options["eol"] = "\r\n"

    docs = {}
    docs["shortdesc"] = "Fence agent for VirtualBox"
    docs["longdesc"] = "fence_vbox is an I/O Fencing agent \
which can be used with the virtual machines managed by VirtualBox. \
It logs via ssh to a dom0 where it runs VBoxManage to do all of \
the work. \
\n.P\n\
By default, vbox needs to log in as a user that is a member of the \
vboxusers group. Also, you must allow ssh login in your sshd_config."
    docs["vendorurl"] = "https://www.virtualbox.org/"
    show_docs(options, docs)

    # Operate the fencing device
    conn = fence_login(options)
    result = fence_action(conn, options, set_power_status, get_power_status, get_out-
lets_status)
    fence_logout(conn, opt["logout_string"])
    sys.exit(result)

if __name__ == "__main__":
    main()
```

Додаток В

Конфігураційний файл PrivacyIDEA

```
import logging
# The realm, where users are allowed to login as administrators
SUPERUSER_REALM = ['super']
# Your database
SQLALCHEMY_DATABASE_URI = 'postgresql+psycopg2://pi-
user:zt,fysqlfey@192.168.5.6:5432/privacyidea'
# This is used to encrypt the auth_token
#SECRET_KEY = 't0p s3cr3t'
# This is used to encrypt the admin passwords
#PI_PEPPER = "Never know..."
# This is used to encrypt the token data and token passwords
PI_ENCFILE = '/etc/privacyidea/enckey'
# This is used to sign the audit log
# This is the dummy base class
#PI_AUDIT_MODULE = 'privacyidea.lib.auditmodules.base'
# This is the default
#PI_AUDIT_MODULE = 'privacyidea.lib.auditmodules.sqlaudit'
# This is used to sign the audit log
PI_AUDIT_KEY_PRIVATE = '/etc/privacyidea/private.pem'
PI_AUDIT_KEY_PUBLIC = '/etc/privacyidea/public.pem'
PI_AUDIT_SQL_TRUNCATE = True
# The Class for managing the SQL connection pool
PI_ENGINE_REGISTRY_CLASS = "shared"
PI_AUDIT_POOL_SIZE = 20
PI_LOGFILE = '/var/log/privacyidea/privacyidea.log'
PI_LOGLEVEL = logging.INFO
PI_PEPPER = 'yeN3CSxSeqxl2qRBQh34ucsZ'
SECRET_KEY = '6UkUzAgrAxmDyf0xvLQz1tXN'
#SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://pi:tempPass12345@localhost/pi?char-
set=utf8'
```

Додаток Г

Конфігураційний файл Pacemaker

```
Cluster Name: privacyidea_ha
Corosync Nodes:
 mfaserver HAmfaserver
Pacemaker Nodes:
 HAmfaserver mfaserver

Resources:
 Group: pi_group
  Resource: vip (class=ocf provider=heartbeat type=IPaddr2)
   Attributes: cidr_netmask=28 ip=192.168.5.7
   Operations: monitor interval=10s (vip-monitor-interval-10s)
               start interval=0s timeout=20s (vip-start-interval-0s)
               stop interval=0s timeout=20s (vip-stop-interval-0s)
  Resource: websvc (class=systemd type=apache2)
   Operations: monitor interval=10s timeout=30s (websvc-monitor-interval-10s)
               start interval=0s timeout=100 (websvc-start-interval-0s)
               stop interval=0s timeout=100 (websvc-stop-interval-0s)

Stonith Devices:
 Resource: st_HAmfaserver (class=stonith type=fence_vbox)
  Attributes: identity_file=/home/vbox_stonith/.ssh/id_rsa ip=192.168.56.1
pcmk_host_list=HAmfaserver pcmk_reboot_action=reboot plug=c969a19c-37be-427b-935a-
b87f2a784372 username=bloodbath
  Meta Attrs: target-role=Started
  Operations: monitor interval=60s (st_HAmfaserver-monitor-interval-60s)
 Resource: st_mfaserver (class=stonith type=fence_vbox)
  Attributes: identity_file=/home/vbox_stonith/.ssh/id_rsa ip=192.168.56.1
pcmk_host_list=mfaserver plug=2abc80e9-111f-4913-ad2b-a268e2e7f1cb username=bloodbath
  Meta Attrs: target-role=Started
  Operations: monitor interval=60s (st_mfaserver-monitor-interval-60s)
Fencing Levels:

Location Constraints:
  Resource: pi_group
    Enabled on:
      Node: HAmfaserver (score:50) (id:location-pi_group-HAmfaserver-50)
      Node: mfaserver (score:100) (id:location-pi_group-mfaserver-100)
  Resource: st_HAmfaserver
    Disabled on:
      Node: HAmfaserver (score:-INFINITY) (id:location-st_HAmfaserver-HAmfaserver--
INFINITY)
  Resource: st_mfaserver
    Disabled on:
      Node: mfaserver (score:-INFINITY) (id:location-st_mfaserver-mfaserver--INFIN-
ITY)
Ordering Constraints:
Colocation Constraints:
```

```
Ticket Constraints:

Alerts:
 No alerts defined

Resources Defaults:
 No defaults set
Operations Defaults:
 No defaults set

Cluster Properties:
 cluster-infrastructure: corosync
 cluster-name: privacyidea_ha
 dc-version: 2.0.3-4b1f869f0f
 have-watchdog: false
 last-lrm-refresh: 1779754969
 no-quorum-policy: stop
 startup-fencing: true
 stonith-action: reboot
 stonith-enabled: true
 stonith-timeout: 60s

Quorum:
  Options:
  Device:
    votes: 1
    Model: net
      algorithm: ffsplit
      host: ubuntudb
```

Конфігураційний файл Corosync

```
totem {
    version: 2
    cluster_name: privacyidea_ha
    transport: knet
    crypto_cipher: aes256
    crypto_hash: sha256
}

nodelist {
    node {
        ring0_addr: mfaserver
        name: mfaserver
        nodeid: 1
    }

    node {
        ring0_addr: hamfaserver
        name: HAmfaserver
```

```
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum

    device {
        model: net
        votes: 1

        net {
            algorithm: ffsplit
            host: ubuntudb
        }
    }
}

logging {
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    to_syslog: yes
    timestamp: on
}
```

Додаток Д

Створення LDAP-resolver та Realm

```
sudo pi-manage resolver create ldapresolver \
    --resolver ldaps.org.net \
    --server ldaps://srv-dc-40-5.org.net \
    --basedn "ou=departments,dc=org,dc=net" \
    --binddn "svcprivacyidea@org.net" \
    --bindpw "zt,fysqlfey@0" \
    --loginnameattribute "sAMAccountName" \
    --searchfilter "(&(objectCategory=person)(sAMAccountName=*))" \
    --tls-ca-file "/etc/ssl/certs/orgnet-cert.pem" \
    --noreferrals

sudo pi-manage realm create users.org.net ldaps.org.net
```