ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ВІДДАЛЕНОГО ДОСТУПУ ДО СЕРВЕРНОЇ ІНФРАСТРУКТУРИ НА БАЗІ TAILSCALE»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми «Комп'ютерна інженерія»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Ярослав ЛУЗАН*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-41
Ярослав ЛУЗАН
Ім'я, ПРІЗВИЩЕ

Керівник: Ярослав ТОРОШАНКО
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерна інженерія

Освітньо-професійна програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ Наталія ЛАЩЕВСЬКА
«_____» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лузан Ярослав Дмитрович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Програмно-апаратний комплекс віддаленого доступу до серверної інфраструктури на базі Tailscale»
керівник кваліфікаційної роботи: Ярослав ТОРОШАНКО, кандидат технічних наук, доцент, старший науковий співробітник ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026р. №51
2. Строк подання кваліфікаційної роботи«14» травня 2026р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, програмно-визначених мережі, Zero Trust, архітектура та принципи функціонування Tailscale і Mesh-мереж, протоколи захищеної взаємодії, механізми NAT Traversal, технології віртуалізації та контейнеризації (Proxmox, Docker), апаратні характеристики, обмеження серверних вузлів, методи організації віддаленого доступу в умовах ізольованих мереж, верифікація та порівняльний аналіз рішень.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Теоретичні основи програмно-визначених мереж та контейнеризації
    2. Архітектурне проектування та розгортання системи на базі неттопа
    3. Оптимізація, безпека та економічне обґрунтування
5. Перелік ілюстративного матеріалу: *презентація*
    1. Мета, об'єкт, предмет дослідження
    2. Актуальність роботи
    3. Порівняння підходів
    4. Загальна архітектура рішення
    5. Апаратна реалізація
    6. Розгортання середовища
    7. Інтеграція Tailscale
    8. Робота в умовах Double NAT
    9. Демонстрація роботи системи

10. Проблеми та їх вирішення
11. Економічна ефективність
12. Висновки
13. Публікації та апробація роботи

6. Дата видачі завдання «20» лютого 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	22.02-20.03.2026	Виконано
2	Обробка матеріалу	20.03-30.03.2026	Виконано
3	Розробка теоретичної частини	31.03-10.04.2026	Виконано
4	Архітектурне проектування та розгортання системи на базі неттопа	11.04-20.04.2026	Виконано
5	Висновки за результатами аналізу	21.04-26.04.2026	Виконано
6	Оформлення пояснювальної записки	26.04-28.04.2026	Виконано
7	Розробка презентації	29.04-01.05.2026	Виконано
8	Передзахист	08.05-11.05.2026	Виконано
9	Здача в деканат	08.05-14.05.2026	Виконано

Здобувач(ка) вищої освіти _____ Ярослав ЛУЗАН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Ярослав ТОРОШАНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 83 стор., 39 рис., 7 табл., 32 джерел.

Мета роботи – спрощення та підвищення безпеки віддаленого доступу до серверної інфраструктури в ізольованих мережах за рахунок застосування принципів Zero Trust та технологій Tailscale.

Об'єкт дослідження – процеси організації віддаленого доступу до серверних ресурсів у мережах з обмеженою або ізольованою інфраструктурою.

Предмет дослідження – засоби та технології побудови програмно-апаратних комплексів віддаленого доступу з використанням Tailscale, контейнеризації Docker та гіпервізорів.

Короткий зміст роботи: Розглянуто еволюцію підходів до віддаленого доступу від класичних VPN до моделі Zero Trust, принципи програмно-визначених мереж та архітектуру Tailscale, включаючи механізми NAT Traversal і P2P-з'єднання. Проаналізовано обмеження традиційних VPN у складних мережевих умовах і визначено переваги підходу Zero Trust, а також узагальнено роль контейнеризації та компактних апаратних платформ.

Особливу увагу приділено розгортанні системи на базі неттопа з використанням віртуалізації та контейнерів, інтегрованих у Tailscale. Реалізовано доступ до ізольованих сервісів без відкриття портів, підтверджено обхід подвійного NAT, а також оцінено ефективність і можливості масштабування рішення.

КЛЮЧОВІ СЛОВА: TAILSCALE, ZERO TRUST, VPN, ПРОГРАМНО-ВИЗНАЧЕНІ МЕРЕЖІ, REMOTE ACCESS, DOCKER, PROXMOX, NAT TRAVERSAL, SUBNET ROUTER, MAGICDNS, SSH, МІКРОСЕРВІСИ.

ЗМІСТ

ВСТУП 8
1. ТЕОРЕТИЧНІ ОСНОВИ ПРОГРАМНО-ВИЗНАЧЕНИХ МЕРЕЖ ТА КОНТЕЙНЕРИЗАЦІЇ 9
1.1 Еволюція парадигми віддаленого доступу 9
1.2 Технологічний стек Tailscale та протокол взаємодії 15
1.3 Мікросервісна архітектура та технології контейнеризації 21
1.4 Апаратне забезпечення для мікросервісних вузлів (Edge Computing) 27
1.5 Безпека безпарольного доступу до апаратної інфраструктури. 33
2. АРХІТЕКТУРНЕ ПРОЕКТУВАННЯ ТА РОЗГОРТАННЯ СИСТЕМИ НА БАЗІ НЕТТОПА 38
2.1 Архітектура програмно-апаратного комплексу та топологія мережі 38
2.2 Покрокова побудова та конфігурація серверного вузла 46
2.3 Алгоритми функціонування Mesh-мережі в умовах ізоляції 56
2.4 Практична реалізація та верифікація доступу 65
3. ОПТИМІЗАЦІЯ, БЕЗПЕКА ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ 71
3.1 Методи оптимізації та вирішення апаратних конфліктів (Troubleshooting) 71
3.2 Порівняльний аналіз архітектур: Zero Trust проти класичних рішень 78
3.3 Техніко-економічне обґрунтування та перспективи масштабування 82
ВИСНОВКИ 90
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 91
Додаток А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 93

ВСТУП

Сучасні підходи до організації віддаленого доступу зазнали суттєвої трансформації у зв’язку з ускладненням мережевих середовищ, зростанням кількості розподілених систем та підвищеними вимогами до безпеки, що призвело до поступового відходу від класичних периметральних моделей захисту на користь концепції Zero Trust, де кожен запит розглядається як потенційно недовірений незалежно від місця його походження, а традиційні VPN-рішення демонструють обмежену ефективність у середовищах із багаторівневим NAT, відсутністю публічних IP-адрес та складною топологією мереж.

Розвиток програмно-визначених мереж та Mesh-топологій забезпечив новий рівень гнучкості у побудові інфраструктур, де взаємодія між вузлами здійснюється напряму з використанням криптографічно захищених тунелів, а технологічний стек Tailscale реалізує таку модель через поєднання координаційних серверів, механізмів автентифікації на основі SSO та автоматизованого встановлення P2P-з’єднань, що дозволяє абстрагуватися від фізичної мережевої інфраструктури та забезпечити стабільний зв’язок навіть у складних умовах ізоляції.

Розвиток мереж популяризував мікросервіси та Docker-контейнеризацію. Використання неттопів у поєднанні з віртуалізацією дозволяє створювати енергоефективні, ізольовані серверні вузли, здатні стабільно працювати 24/7 навіть за обмежених ресурсів.

Практична частина присвячена розробці комплексу віддаленого доступу на базі неттопа та Mesh-мережі Tailscale. Проєкт передбачає налаштування захищеної інфраструктури без відкритих портів (через подвійний NAT) з використанням Subnet Router і MagicDNS, а також експериментальну перевірку ефективності такої архітектури.

1. ТЕОРЕТИЧНІ ОСНОВИ ПРОГРАМНО-ВИЗНАЧЕНИХ МЕРЕЖ ТА КОНТЕЙНЕРИЗАЦІЇ

1.1 Еволюція парадигми віддаленого доступу

Розвиток інформаційних технологій, що спостерігається протягом останніх років, призвів до суттєвої трансформації підходів до організації віддаленого доступу до ресурсів, оскільки традиційні моделі більше не відповідають вимогам сучасних розподілених середовищ. Якщо раніше основним принципом побудови мережевої безпеки залишалася концепція чітко визначеного периметра, в межах якого всі внутрішні ресурси вважалися довіреними, то сьогодні такий підхід поступово втрачає актуальність через сукупність факторів, пов’язаних із широким впровадженням хмарних технологій, мобільних пристроїв, розподілених обчислювальних систем і мікросервісної архітектури, що розвиваються швидкими темпами та змінюють загальну структуру IT-інфраструктури.

Традиційна модель безпеки будувалася на припущенні, що внутрішній периметр мережі є безпечним середовищем, тоді як зовнішній простір розглядався як потенційно небезпечний, при цьому доступ до внутрішніх ресурсів контролювався за допомогою міжмережевих екранів, VPN-тунелів і систем керування доступом, що формували своєрідний бар’єр між внутрішньою та зовнішньою частинами інфраструктури. Така модель була ефективною у часи, коли ресурси знаходилися в межах однієї локальної мережі, а кількість користувачів і пристроїв була обмеженою, однак зі зростанням кількості точок доступу, збільшенням числа користувачів і появою віддалених сервісів вона перестала забезпечувати належний рівень захисту, оскільки межі периметра стали розмитими, а контроль доступу — складнішим і менш передбачуваним (див. рис. 1.1).

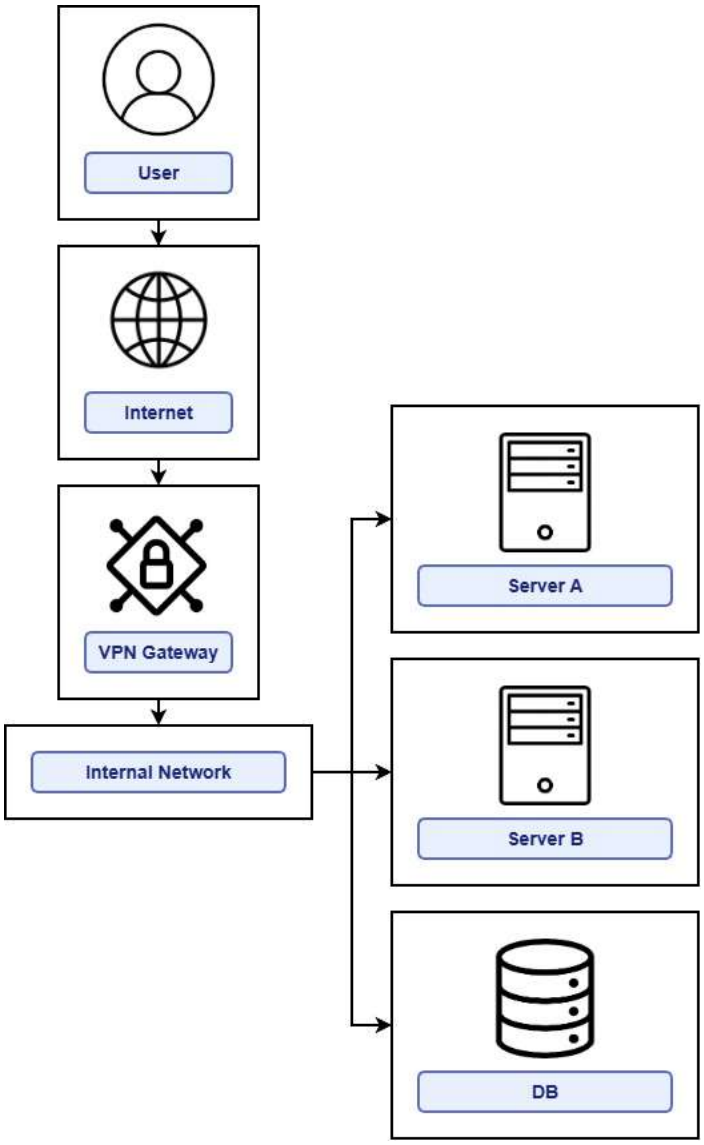


Рисунок 1.1 – Класична модель VPN-доступу (Perimeter Security)

У результаті відбувся перехід до нової парадигми, в якій довіра не надається жодному елементу системи за замовчуванням, а кожен запит розглядається як потенційно небезпечний незалежно від того, звідки він надходить, що лягло в основу концепції Zero Trust, яка передбачає постійну перевірку автентичності користувача, його прав доступу та контексту взаємодії з ресурсами, включаючи параметри пристрою, місце розташування та інші фактори, що можуть впливати на рівень довіри. Такий підхід дозволяє значно підвищити рівень безпеки, оскільки доступ до ресурсів надається не на основі місцезнаходження в мережі, а на основі підтвердженої ідентичності та відповідності встановленим політикам.

Обмеження класичних VPN-рішень у сучасних інфраструктурах. Віртуальні приватні мережі протягом тривалого часу залишалися основним інструментом забезпечення безпечного віддаленого доступу до корпоративних ресурсів, оскільки дозволяли створювати зашифровані тунелі між клієнтом і сервером, забезпечуючи конфіденційність передавання даних і захист від несанкціонованого доступу. Незважаючи на це, в умовах сучасних розподілених інфраструктур їх використання супроводжується рядом суттєвих обмежень, які знижують ефективність такого підходу та ускладнюють його застосування.

Однією з ключових проблем є модель повного доступу до мережі після успішної автентифікації, відповідно до якої користувач, підключившись до VPN, отримує доступ до значної частини внутрішньої інфраструктури, навіть якщо фактично йому потрібен лише один конкретний сервіс, що створює потенційні ризики горизонтального переміщення в межах мережі у випадку компрометації облікових даних. Така ситуація може призвести до поширення загроз між різними сегментами інфраструктури, оскільки відсутній достатній рівень ізоляції між сервісами [1].

Іншим важливим обмеженням є централізована архітектура VPN-серверів, яка формує єдину точку відмови, оскільки у випадку перевантаження або виходу з ладу сервера доступ до ресурсів стає недоступним для всіх користувачів одночасно, що негативно впливає на відмовостійкість системи та її масштабованість. Масштабування таких рішень потребує додаткових апаратних ресурсів, складного налаштування та постійного адміністрування, що підвищує витрати та ускладнює підтримку інфраструктури [1].

Додатковим фактором є обмежена гнучкість VPN у середовищах із динамічною інфраструктурою, зокрема в контейнеризованих і хмарних системах, де сервіси можуть створюватися і знищуватися в автоматичному режимі, що характерно для мікросервісної архітектури, у той час як VPN-рішення не забезпечують необхідного рівня адаптивності до таких змін [1-2].

Окрему увагу слід приділити складності керування доступами, оскільки адміністрування VPN передбачає ручне налаштування політик доступу,

маршрутизації та криптографічних ключів, що підвищує ймовірність помилок конфігурації та створює додаткові ризики безпеки (див. табл. 1.1).

Таблиця 1.1 – Порівняння класичних VPN та підходу Zero Trust

Критерій	Класичний VPN	Zero Trust
Модель довіри	Довіра після автентифікації	Відсутність довіри за замовчуванням
Тип доступу	Повний доступ до мережі	Доступ лише до конкретних ресурсів
Залежність від інфраструктури	Висока (сервер, порти, маршрутизація)	Низька (оверлей-мережа)
Потреба у публічній IP	Так	Ні
Масштабованість	Обмежена, складна	Висока, гнучка
Безпека	Ризик lateral movement	Ізоляція та контроль доступу
Складність налаштування	Висока	Помірна/нижча

У результаті сукупність зазначених недоліків призводить до поступового зниження ефективності класичних VPN-рішень у сучасних розподілених середовищах, що обумовлює необхідність переходу до більш гнучких і безпечних підходів організації доступу.

Концепція Zero Trust Network Access (ZTNA) та її базові принципи. Zero Trust Network Access є сучасною концепцією організації доступу до мережевих ресурсів, що базується на принципі «ніколи не довіряти, завжди перевіряти», який суттєво відрізняється від традиційних підходів, де довіра надається користувачу після успішного входу до мережі, тоді як у межах ZTNA передбачається постійна перевірка кожного запиту, кожної спроби підключення та будь-якої взаємодії незалежно від джерела походження. Основна ідея такого підходу полягає у наданні доступу не до всієї мережі, а виключно до конкретного ресурсу, який необхідний користувачу для виконання визначеної задачі, що досягається шляхом застосування політик доступу, які враховують ідентичність користувача, стан пристрою, геолокацію та інші контекстні параметри, здатні впливати на рівень довіри до запиту [3].

Такий підхід дозволяє реалізувати принцип мінімально необхідних привілеїв, відповідно до якого користувач отримує лише ті права, які безпосередньо необхідні для виконання його функцій, що зменшує ризик несанкціонованого доступу до інших частин системи, при цьому перевірка доступу не обмежується лише етапом первинної автентифікації, а здійснюється протягом усього сеансу взаємодії, що формує механізм безперервної автентифікації. Додатково реалізується мікросегментація мережі, яка передбачає її поділ на ізольовані сегменти з метою обмеження можливого поширення загроз у разі компрометації одного з елементів, а контроль доступу здійснюється через централізовані системи керування ідентичністю, такі як SSO, що дозволяє ефективно управляти правами користувачів та їх взаємодією з ресурсами (див. рис. 1.2) [3].

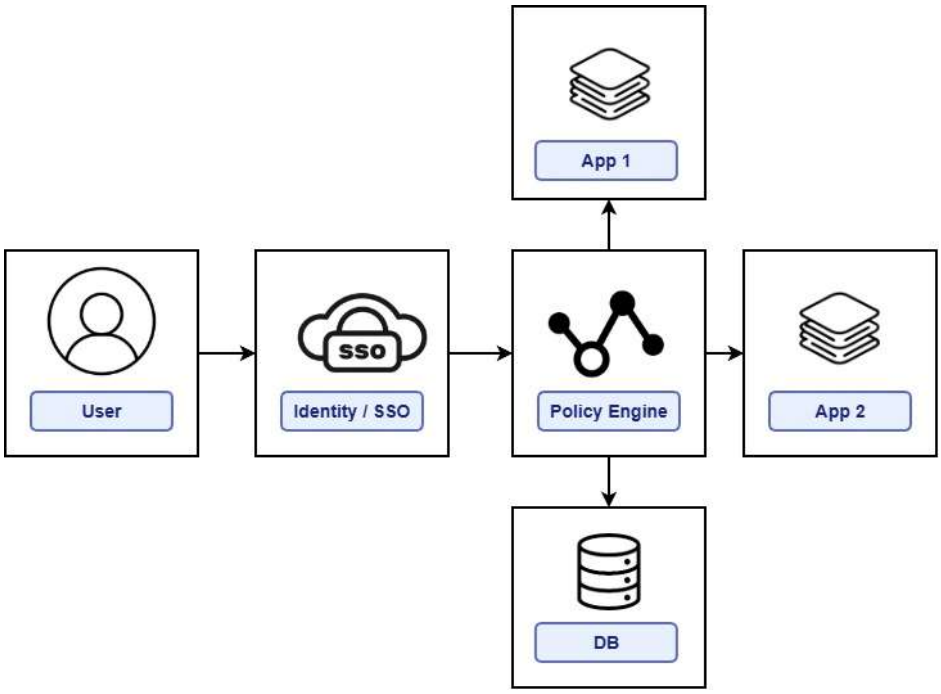


Рисунок 1.2 – Концепція Zero Trust Network Access (ZTNA)

Захист переданих даних забезпечується шляхом шифрування трафіку незалежно від каналу передачі, що унеможливлює його перехоплення або модифікацію, при цьому сама концепція добре інтегрується з сучасними технологіями, зокрема хмарними сервісами, контейнеризацією та програмно-

визначеними мережами, що дозволяє досягти високого рівня гнучкості та масштабованості інфраструктури. Додатковою особливістю є відсутність необхідності відкривати порти на мережевих пристроях або публічно експонувати сервіси, оскільки доступ здійснюється через захищені канали, що значно зменшує площу атаки та мінімізує потенційні вектори проникнення в систему [3-4].

Програмно-визначені мережі (SDN) як основа для сучасних Mesh-топологій. Програмно-визначені мережі є однією з базових технологій, що забезпечують реалізацію сучасних підходів до побудови мережевої інфраструктури, зокрема у контексті Zero Trust та Mesh-архітектур, оскільки дозволяють змінити традиційну модель керування мережею за рахунок відокремлення площини керування від площини передачі даних. У класичних мережах функції керування та маршрутизації реалізуються безпосередньо на мережевих пристроях, таких як маршрутизатори та комутатори, що ускладнює централізоване управління та обмежує гнучкість системи, тоді як у SDN логіка керування переноситься на окремий програмний контролер, який визначає правила обробки трафіку та маршрутизації (див. рис. 1.3) [5].

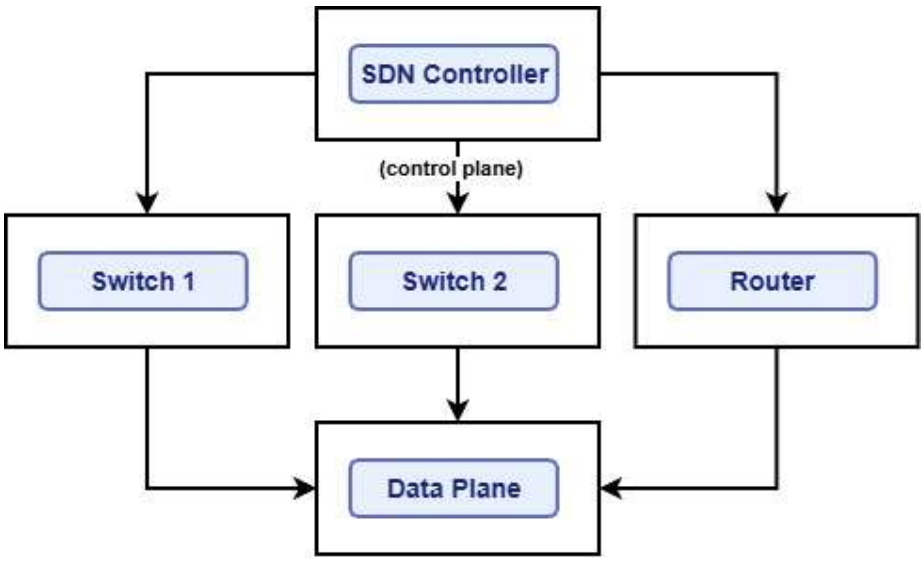


Рисунок 1.3 – Архітектура програмно-визначеної мережі (SDN)

Такий підхід дозволяє централізовано керувати мережею, швидко змінювати конфігурацію без фізичного втручання в інфраструктуру, автоматизувати процеси

розгортання та масштабування, а також реалізовувати складні політики безпеки, які можуть динамічно адаптуватися до змін у середовищі. Одним із важливих напрямів застосування SDN є побудова Mesh-мереж, у яких вузли взаємодіють безпосередньо один з одним за принципом peer-to-peer, що усуває необхідність у використанні центрального вузла для передачі даних і підвищує загальну відмовостійкість системи.

У такій топології забезпечується більш ефективне використання мережевих ресурсів, а продуктивність зростає за рахунок скорочення кількості проміжних ланок у передачі трафіку, при цьому особливої ефективності досягається у поєднанні з технологіями обходу NAT, які дозволяють встановлювати прямі з'єднання між пристроями навіть у випадках використання «сірих» IP-адрес, що є типовою ситуацією для сучасних мереж [5-6].

Додатковою перевагою програмно-визначених мереж є можливість оптимізації використання апаратних ресурсів, оскільки програмне керування дозволяє більш точно розподіляти навантаження між процесором, пам'яттю та мережевими інтерфейсами, що особливо важливо для компактних пристроїв, таких як неттопи або одноплатні комп'ютери, які використовуються у складі edge-інфраструктури. У результаті поєднання SDN, Zero Trust та Mesh-топологій формує основу для побудови гнучких, масштабованих і безпечних мережевих систем, здатних ефективно функціонувати у сучасних розподілених середовищах із мікросервісною архітектурою.

1.2 Технологічний стек Tailscale та протокол взаємодії

В умовах розвитку мережевих технологій дедалі більшого поширення набувають рішення, що поєднують простоту розгортання, високий рівень безпеки та мінімальні вимоги до інфраструктури, що зумовлено зростанням кількості розподілених систем, мобільних пристроїв і необхідністю забезпечення

захищеного доступу без складних конфігурацій. Одним із таких рішень є Tailscale, який реалізує підхід програмно-визначеної мережі та базується на концепції Zero Trust, поєднуючи централізоване управління через Control Plane із децентралізованою передачею даних у межах Data Plane, що дозволяє досягти балансу між зручністю адміністрування та високою продуктивністю мережевої взаємодії.

На відміну від класичних VPN-рішень, де весь трафік проходить через центральний сервер, у такій моделі передача даних відбувається безпосередньо між вузлами мережі, що суттєво зменшує затримки, знижує навантаження на інфраструктуру та усуває необхідність масштабування центральних вузлів під зростаючий обсяг трафіку, при цьому система автоматично забезпечує генерацію криптографічних ключів, їх регулярну ротацію та інтеграцію з механізмами керування ідентичністю, що дозволяє реалізувати високий рівень захисту без значних адміністративних витрат. Додатково використовуються механізми обходу NAT, які забезпечують встановлення прямих з’єднань навіть у складних мережевих умовах, що робить такий підхід ефективним для побудови захищених мереж у розподілених середовищах [32].

Архітектура координаційних серверів (Control Plane) та вузлів (Data Plane). Архітектурна модель Tailscale базується на чіткому розмежуванні функцій між площиною керування та площиною передачі даних, що є характерною ознакою програмно-визначених мереж і дозволяє досягти високого рівня гнучкості та масштабованості. Control Plane виконує функції автентифікації користувачів і пристроїв, зберігання інформації про вузли мережі, включаючи їх:

— IP-адреси;

— криптографічні ключі;

— статус підключення.

Все це забезпечує розповсюдження політик доступу та координацію процесу встановлення з’єднань між вузлами, при цьому не бере участі у передачі користувацького трафіку, що зменшує навантаження на центральні компоненти системи та усуває ризик виникнення вузьких місць [7].

Data Plane, у свою чергу, відповідає за безпосередню передачу даних між вузлами мережі, кожен із яких може представляти собою [8]:

— фізичний комп’ютер;

— віртуальну машину;

— контейнер;

— окремий сервіс.

При цьому на кожному вузлі функціонує спеціалізований агент, що встановлює зашифровані з’єднання з іншими учасниками мережі, виконує маршрутизацію пакетів і забезпечує взаємодію з мережевим стеком операційної системи. У результаті формується Mesh-мережа, у якій кожен вузол може напряму взаємодіяти з іншими без необхідності проходження через центральний сервер, що підвищує відмовостійкість системи, оскільки навіть у випадку недоступності Control Plane вже встановлені з’єднання продовжують функціонувати без переривань (див. рис. 1.4) [8].

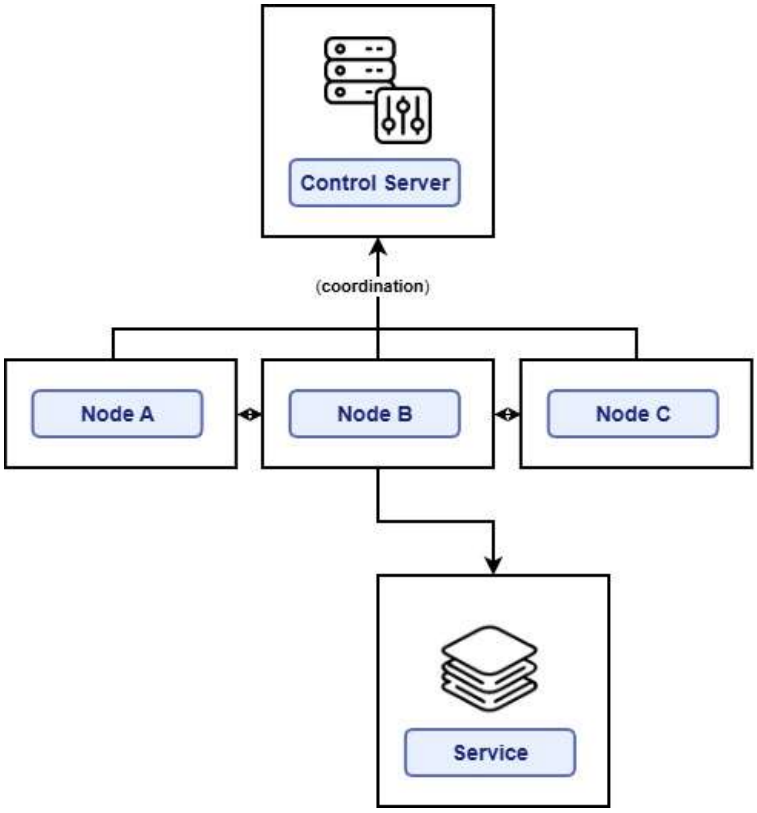


Рисунок 1.4 – Загальна архітектура Tailscale

З інженерної точки зору важливою особливістю є використання віртуального мережевого інтерфейсу типу TUN, який інтегрується безпосередньо в ядро операційної системи та дозволяє обробляти мережевий трафік на низькому рівні, забезпечуючи прозорість для прикладних програм і сумісність із існуючими мережевими сервісами без необхідності їх модифікації, що значно спрощує інтеграцію у вже наявну інфраструктуру та дозволяє використовувати стандартні інструменти адміністрування без додаткових налаштувань.

Механізми забезпечення Identity Access Management (IAM) через SSO. Однією з характерних особливостей сучасних мережевих рішень є інтеграція з системами керування ідентичністю, що дозволяє змістити акцент із мережевих параметрів доступу на перевірену ідентичність користувача, у зв'язку з чим у Tailscale реалізовано підхід, за якого доступ до ресурсів базується не на IP-адресах або статичних ключах, а на облікових записах користувачів, підтверджених через зовнішні провайдери ідентичності, такі як:

— Google;

— Microsoft;

— GitHub.

Така інтеграція забезпечує централізоване управління обліковими записами, дозволяє використовувати багатофакторну автентифікацію та зменшує кількість паролів, які необхідно зберігати або запам'ятовувати, що у підсумку підвищує загальний рівень безпеки системи та спрощує адміністрування доступів [9].

Після успішної автентифікації користувач отримує доступ до мережевих ресурсів відповідно до визначених політик, які формалізуються у вигляді списків контролю доступу (ACL) і дозволяють гнучко регулювати взаємодію між користувачами, пристроями та сервісами, визначаючи, які ресурси доступні конкретному користувачу або групі користувачів, при цьому можливе обмеження доступу до окремих сервісів лише для певних ролей або дозволення підключення виключно з визначених пристроїв. Такий підхід реалізує принцип мінімально необхідних привілеїв, відповідно до якого кожен користувач отримує лише ті права, що необхідні для виконання його задач, що значно зменшує ризик

несанкціонованого доступу та обмежує потенційне поширення загроз у межах інфраструктури [10].

Додатковим аспектом є врахування стану пристрою під час прийняття рішення про надання доступу, що дозволяє реалізувати так звану перевірку "device posture", у межах якої доступ може бути наданий лише за умови відповідності пристрою встановленим політикам безпеки, наприклад наявності актуальних оновлень операційної системи або відсутності відомих вразливостей, що підвищує рівень захищеності навіть у випадках використання скомпрометованих або ненадійних пристроїв (див. рис. 1.5).

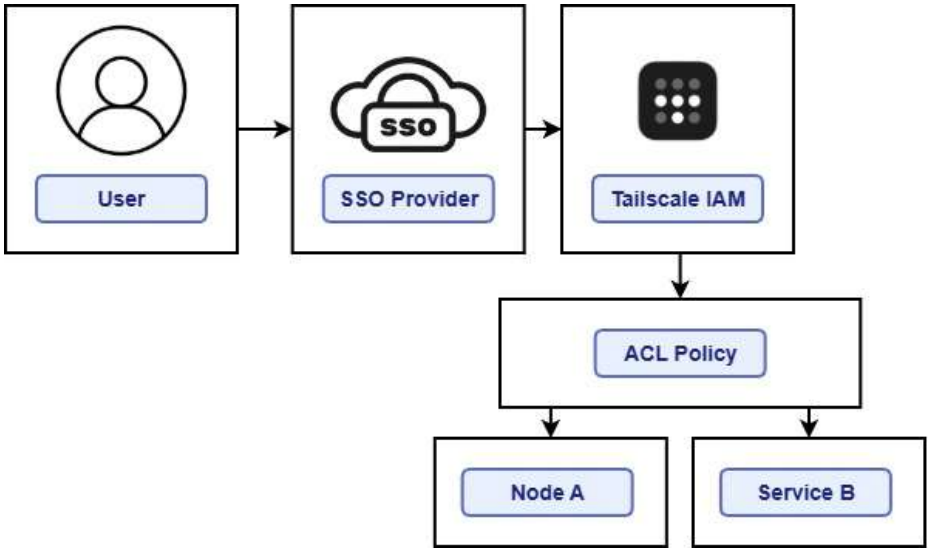


Рисунок 1.5 – Модель IAM у Tailscale

У результаті застосування IAM через SSO формує модель доступу, у якій ідентичність користувача стає основним фактором авторизації, а не мережеве розташування або технічні параметри підключення, що повністю узгоджується з принципами Zero Trust і дозволяє створювати більш гнучкі та контрольовані системи доступу до ресурсів.

Принципи побудови криптографічних тунелів P2P. Функціонування мережевої взаємодії у Tailscale базується на створенні захищених каналів зв’язку між вузлами, що реалізується через використання сучасних криптографічних протоколів, серед яких ключове місце займає WireGuard, який забезпечує високий

рівень безпеки та продуктивності за рахунок застосування ефективних алгоритмів шифрування та оптимізованої реалізації. Основна ідея полягає у встановленні прямих "peer-to-peer" з'єднань між вузлами, для чого використовуються пари криптографічних ключів — публічні та приватні, які застосовуються для автентифікації та шифрування трафіку, при цьому обмін публічними ключами координується через Control Plane, після чого вузли отримують можливість встановлювати пряме з'єднання без участі центрального сервера у передачі даних [11].

Процес встановлення тунелю включає:

1. ініціалізацію вузла з отриманням конфігурації;
2. обмін криптографічними ключами між учасниками взаємодії;
3. визначення оптимального маршруту з урахуванням мережевих умов і наявності NAT.

Після чого формується зашифрований канал, через який здійснюється передача даних, що забезпечує конфіденційність і цілісність інформації. Однією з основних технічних складностей у таких з'єднаннях є наявність механізму трансляції мережевих адрес, який ускладнює встановлення прямого зв'язку між вузлами, тому використовуються допоміжні технології, зокрема STUN для визначення зовнішніх параметрів підключення та DERP-сервери як ретранслятори у випадках, коли пряме з'єднання встановити неможливо, при цьому навіть у режимі ретрансляції трафік залишається зашифрованим і недоступним для сторонніх осіб [11-12].

З інженерної точки зору важливою характеристикою є низький рівень накладних витрат протоколу WireGuard, що досягається завдяки використанню сучасних криптографічних алгоритмів і компактної реалізації, яка забезпечує високу швидкість передачі даних навіть на малопотужних пристроях, а автоматична ротація ключів дозволяє підтримувати актуальний рівень безпеки без необхідності ручного втручання, що особливо важливо у великих мережах із значною кількістю вузлів. У результаті поєднання "peer-to-peer" тунелів із сучасними криптографічними механізмами дозволяє створювати ефективну, захищену та масштабовану мережеву інфраструктуру, що відповідає вимогам

сучасних розподілених систем і забезпечує стабільну роботу навіть у складних мережевих умовах.

1.3 Мікросервісна архітектура та технології контейнеризації

В умовах розвитку інформаційних систем дедалі частіше застосовується підхід, що базується на мікросервісній архітектурі, яка передбачає поділ складного програмного продукту на незалежні, слабо пов’язані між собою компоненти, кожен із яких реалізує окрему функціональність, має власну логіку обробки даних і може оновлюватися або масштабуватися незалежно від інших частин системи. Така організація дозволяє:

1. підвищити гнучкість інфраструктури;
2. спростити процеси розробки та супроводу;
3. забезпечити більшу відмовостійкість.

Все це тому, що вихід з ладу одного сервісу не призводить до повної зупинки всієї системи, а впливає лише на обмежену частину функціоналу.

З інженерної точки зору мікросервісна модель дозволяє більш ефективно використовувати апаратні ресурси, оскільки кожен сервіс може бути розгорнутий із урахуванням конкретних вимог до процесорного часу, пам’яті та мережевих ресурсів, що є особливо актуальним для середовищ з обмеженими можливостями, таких як компактні сервери, неттопи або віртуальні машини. Реалізація такого підходу безпосередньо пов’язана з використанням технологій контейнеризації, які забезпечують ізоляцію окремих сервісів у межах операційної системи, зберігаючи при цьому їхню портативність і незалежність.

Контейнери представляють собою ізольовані середовища виконання, у яких розміщується окремий сервіс разом із усіма необхідними залежностями, що дозволяє гарантувати однакову поведінку додатка незалежно від середовища розгортання, при цьому використання контейнерів не потребує створення

повноцінної віртуальної машини, оскільки вони працюють на основі спільного ядра операційної системи, що значно зменшує накладні витрати та підвищує ефективність використання ресурсів. Одним із найбільш поширених інструментів контейнеризації є Docker, який надає засоби для створення, запуску та керування контейнерами, дозволяючи автоматизувати процеси розгортання і спростити адміністрування системи.

У контексті сучасних мережевих інфраструктур контейнеризація дозволяє створювати ізольовані середовища для кожного сервісу, контролювати їхню взаємодію та оптимізувати використання ресурсів, що сприяє підвищенню ефективності роботи системи та спрощує реалізацію складних архітектурних рішень.

Порівняльний аналіз віртуальних машин та Docker-контейнерів. Віртуалізація традиційно є основою побудови сучасних IT-інфраструктур, оскільки дозволяє ізолювати середовища виконання та забезпечувати незалежність різних сервісів, при цьому основними підходами до реалізації ізоляції є використання віртуальних машин або контейнерів, кожен із яких має свої особливості, переваги та обмеження. Віртуальні машини забезпечують повну ізоляцію за рахунок емуляції апаратного забезпечення, що дозволяє запускати окремі операційні системи на одному фізичному хості, однак такий підхід потребує значних апаратних ресурсів, включаючи [13]:

— процесорний час;

— оперативну пам’ять;

— дисковий простір.

Все що обмежує кількість віртуальних середовищ, які можуть бути розгорнуті на одному пристрої.

Контейнери, на відміну від віртуальних машин, використовують спільне ядро операційної системи хоста та ізолюють лише користувацький простір, що дозволяє [14]:

1. значно зменшити накладні витрати;
2. прискорити процес запуску;

3. підвищити щільність розміщення сервісів.

При чому один фізичний пристрій може обслуговувати значно більшу кількість контейнерів порівняно з кількістю віртуальних машин. Запуск контейнерів відбувається за лічені секунди, тоді як ініціалізація віртуальної машини може тривати значно довше через необхідність завантаження повноцінної операційної системи.

Контейнери забезпечують більшу гнучкість у розгортанні та масштабуванні, оскільки можуть легко переноситися між різними середовищами без необхідності змін у конфігурації, що робить їх зручним інструментом для побудови динамічних систем, однак рівень ізоляції у контейнерів є нижчим порівняно з віртуальними машинами через використання спільного ядра, що створює додаткові вимоги до безпеки, включаючи обмеження привілеїв, застосування механізмів контролю доступу та використання додаткових інструментів захисту [13-14].

З інженерної точки зору доцільним є використання гібридного підходу, за якого віртуальні машини застосовуються для базової ізоляції середовища, а контейнеризація використовується всередині таких середовищ для розгортання окремих сервісів, що дозволяє поєднати високий рівень безпеки з ефективним використанням ресурсів і гнучкістю управління інфраструктурою.

У підсумку застосування контейнерів дозволяє максимально ефективно використовувати наявні апаратні ресурси, що є критично важливим для компактних обчислювальних платформ, де обмеження за ресурсами вимагають оптимізації кожного компонента системи, при цьому правильне поєднання різних рівнів віртуалізації дозволяє досягти балансу між продуктивністю, безпекою та масштабованістю.

Мережева ізоляція в Docker (Bridge, Host, Macvlan). У середовищі контейнеризації важливим аспектом функціонування є організація мережевої взаємодії між окремими контейнерами, оскільки кожен сервіс має обмінюватися даними як із іншими компонентами системи, так і з зовнішніми ресурсами, при цьому необхідно забезпечити баланс між ізоляцією, продуктивністю та зручністю адміністрування, що досягається завдяки використанню різних режимів мережі, які

підтримуються Docker і дозволяють гнучко налаштовувати архітектуру взаємодії залежно від вимог конкретного сценарію.

Найбільш поширеним режимом є Bridge-мережа, яка використовується за замовчуванням і передбачає створення віртуального мережевого інтерфейсу, через який контейнери взаємодіють між собою в межах ізольованого середовища, при цьому доступ до таких контейнерів із зовнішньої мережі здійснюється через механізм пробросу портів, що дозволяє чітко контролювати, які саме сервіси доступні зовні, а які залишаються внутрішніми, така модель забезпечує достатній рівень ізоляції та безпеки, зберігаючи при цьому простоту налаштування і універсальність використання, що робить її придатною для більшості типових задач, пов’язаних із розгортанням веб-сервісів, API або допоміжних компонентів системи [15].

Інший підхід реалізується через Host-режим, у якому контейнер використовує мережевий стек операційної системи без додаткового рівня віртуалізації, що означає прямий доступ до мережевих інтерфейсів хоста і відсутність необхідності у пробросі портів, у такому випадку досягається максимальна продуктивність передачі даних та мінімізуються накладні витрати, пов’язані з мережевою ізоляцією, однак одночасно знижується рівень безпеки, оскільки контейнер фактично стає частиною мережевого середовища хоста, що може призводити до конфліктів портів або ускладнювати контроль доступу, тому використання такого режиму доцільне лише у випадках, коли пріоритетом є швидкодія та мінімальні затримки, наприклад для високонавантажених сервісів або систем моніторингу [15].

Більш гнучким і водночас складнішим у реалізації є режим Macvlan, який дозволяє кожному контейнеру отримувати власну MAC-адресу та IP-адресу у межах фізичної мережі, унаслідок чого контейнер виглядає як окремий повноцінний пристрій, що підключений до локальної мережі, та може безпосередньо взаємодіяти з іншими вузлами без використання механізмів пробросу портів, такий підхід забезпечує високий рівень інтеграції контейнерів у мережеву інфраструктуру та дозволяє використовувати їх як самостійні мережеві

елементи, однак потребує більш складної конфігурації та врахування особливостей маршрутизації, що може ускладнювати впровадження у порівнянні з базовими режимами (див. рис. 1.6).

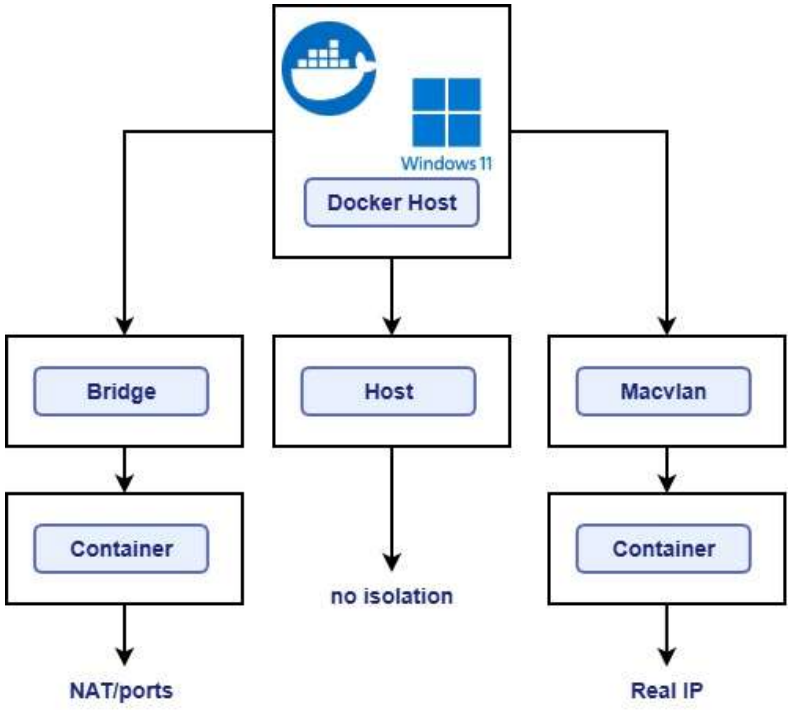


Рисунок 1.6 – Мережеві режими Docker (Bridge, Host, Macvlan)

Вибір конкретного режиму мережевої взаємодії визначається вимогами до безпеки, продуктивності та архітектури системи, оскільки для внутрішніх сервісів зазвичай достатньо використання Bridge-мережі, яка забезпечує необхідний рівень ізоляції та керованості, тоді як для задач, що потребують максимальної продуктивності або тісної інтеграції з фізичною мережею, можуть застосовуватися Host або Macvlan залежно від специфіки реалізації та вимог до мережевої взаємодії.

Керування життєвим циклом контейнерів (Docker Compose). У середовищах, де використовується значна кількість контейнерів, виникає потреба у централізованому керуванні їхнім життєвим циклом, оскільки ручне адміністрування кожного окремого контейнера стає неефективним і призводить до зростання складності підтримки системи, у такому випадку застосовуються інструменти автоматизації, серед яких одним із найбільш поширених є Docker

Compose, що дозволяє описати конфігурацію всієї системи у вигляді декларативного файлу, у якому визначаються [16]:

— сервіси;
— їхні параметри;
— мережеві налаштування;
— залежності сховища даних.

Завдяки використанню такого підходу стає можливим швидке розгортання складних багатокомпонентних систем, які можуть включати:

— веб-сервери;
— проксі-рішення;
— сервіси доступу;
— системи моніторингу;
— інші компоненти.

При цьому всі вони об’єднуються в єдину мережеву структуру та запускаються за допомогою однієї команди, що значно спрощує процес розгортання та зменшує ймовірність помилок конфігурації, додатковою перевагою є можливість відтворення середовища на різних пристроях без необхідності повторного налаштування, оскільки конфігураційний файл може бути перенесений та використаний повторно.

Керування життєвим циклом контейнерів включає [16]:

— запуск;
— зупинку;
— перезапуск;
— оновлення сервісів.

Все це дозволяє підтримувати стабільність роботи системи у режимі безперервної експлуатації, особливо у сценаріях, де сервіси працюють у режимі 24/7 та потребують високої доступності, при цьому Docker Compose виконує функцію базового інструменту оркестрації, який забезпечує достатній рівень керованості без необхідності використання складніших платформ, таких як Kubernetes, що робить

його оптимальним вибором для невеликих та середніх інфраструктур (див. рис. 1.7).

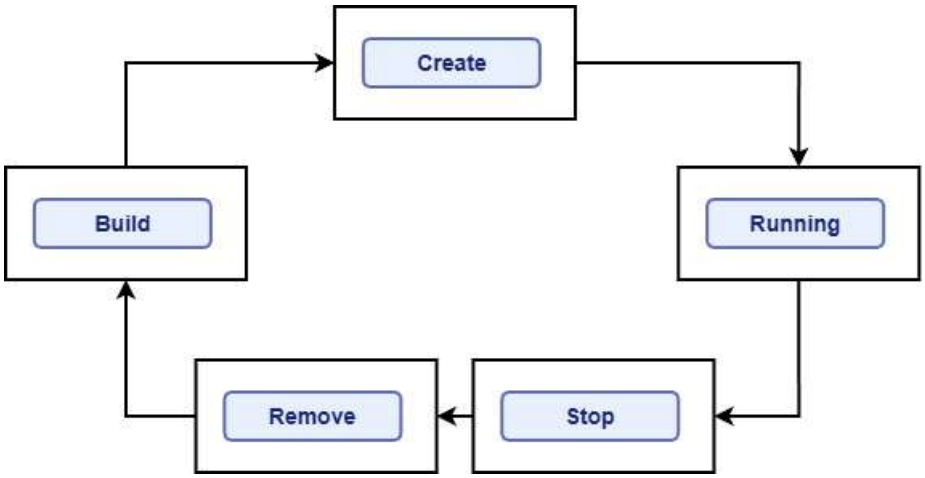


Рисунок 1.7 – Життєвий цикл контейнера Docker

Поєднання мікросервісної архітектури, контейнеризації та інструментів керування життєвим циклом дозволяє сформувати ефективну основу для побудови сучасних програмно-апаратних комплексів, у межах яких забезпечується ізоляція сервісів, оптимізація використання ресурсів та гнучке управління інфраструктурою, що особливо актуально для розподілених систем і середовищ із обмеженими апаратними можливостями.

1.4 Апаратне забезпечення для мікросервісних вузлів (Edge Computing)

В умовах розвитку розподілених інформаційних систем дедалі більшого поширення набуває концепція edge computing, яка передбачає перенесення обчислювальних процесів ближче до джерела виникнення даних, що дозволяє:

1. зменшити затримки передачі;
2. знизити навантаження на мережеву інфраструктуру;
3. підвищити загальну стійкість системи до відмов.

При чому обробка не залежить виключно від централізованих хмарних ресурсів і може виконуватися локально навіть у разі обмеженого або нестабільного з’єднання із зовнішніми сервісами, при цьому такий підхід дозволяє ефективно використовувати компактні та енергоефективні пристрої як локальні вузли для розгортання мікросервісів, мережевих сервісів і систем контролю доступу.

З інженерної точки зору застосування edge-підходу створює можливість побудови гнучкої інфраструктури, у якій обчислювальні ресурси розподіляються між різними вузлами, що можуть виконувати широкий спектр задач, включаючи маршрутизацію трафіку, обробку запитів, організацію захищених з’єднань, розгортання веб-додатків та підтримку сервісів управління, при цьому важливим аспектом залишається забезпечення балансу між продуктивністю, енергоспоживанням та надійністю функціонування, оскільки обмежені апаратні ресурси таких вузлів потребують оптимального використання та ефективного розподілу навантаження.

Особливої актуальності такий підхід набуває у середовищах малого бізнесу, навчальних лабораторій або домашніх серверних рішень, де відсутня потреба у використанні повноцінної серверної інфраструктури, а натомість доцільним є застосування більш доступних і компактних альтернатив, що дозволяють забезпечити необхідний рівень функціональності без значних фінансових витрат, при цьому edge-вузли можуть інтегруватися у загальну мережу, формуючи розподілену систему, яка здатна масштабуватися шляхом додавання нових пристроїв без суттєвого ускладнення архітектури.

Використання неттопів (на прикладі архітектури x86_64) як локальних серверів. У цьому контексті компактні персональні комп’ютери, відомі як неттопи, що базуються переважно на архітектурі x86_64, розглядаються як ефективна платформа для реалізації локальних серверів, оскільки вони поєднують у собі достатню обчислювальну потужність, сумісність із сучасним програмним забезпеченням і відносно низьке енергоспоживання, що дозволяє використовувати їх у режимі безперервної роботи без значного впливу на енергетичні витрати [17].

Архітектура x86_64 забезпечує повну сумісність із більшістю сучасних операційних систем, включаючи різні дистрибутиви Linux і Windows, а також підтримку широкого спектра програмного забезпечення, такого як системи контейнеризації, інструменти віртуалізації та мережеві сервіси, що дозволяє використовувати такі пристрої як універсальну платформу для розгортання мікросервісних середовищ без необхідності адаптації програмного коду або використання специфічних рішень.

З технічної точки зору неттопи можуть бути оснащені багатоядерними процесорами із підтримкою багатопотоковості, достатнім обсягом оперативної пам’яті, що зазвичай варіюється у межах від 8 до 32 гігабайтів, швидкими накопичувачами типу SSD або NVMe, а також інтегрованими мережевими інтерфейсами, включаючи Ethernet та безпровідні модулі, що створює передумови для стабільної роботи декількох контейнеризованих сервісів одночасно та дозволяє ефективно розподіляти навантаження між різними компонентами системи (див. рис. 1.8).



Рисунок 1.8 – Зовнішній вигляд неттопа Lenovo ThinkCenre M920q
як серверного вузла

У практичному застосуванні один такий пристрій може виконувати одночасно декілька ролей, включаючи функції:

1. реверс-проксі сервера;
2. вузла мережевої взаємодії;
3. платформи для хостингу веб-додатків;
4. середовища для запуску допоміжних сервісів.

Все це дозволяє значно спростити архітектуру системи та зменшити кількість необхідного обладнання, при цьому компактні розміри та низький рівень шуму роблять такі рішення придатними для використання у звичайних офісних або домашніх умовах без потреби у спеціалізованих серверних приміщеннях.

Додатковою перевагою є можливість об’єднання декількох неттопів у єдину мережу, що дозволяє реалізувати базові механізми масштабування та балансування навантаження, наближаючи таку інфраструктуру до принципів кластеризації, при цьому навіть у простих конфігураціях досягається підвищення відмовостійкості, оскільки у разі виходу з ладу одного вузла інші можуть продовжувати виконання частини функцій, забезпечуючи безперервність роботи системи та підвищуючи її надійність у довгостроковій перспективі.

Енергоефективність та ресурсна ємність компактних ПК у режимі 24/7. Функціонування edge-вузлів у режимі безперервної роботи 24/7 формує підвищені вимоги до енергоефективності, стабільності та надійності апаратного забезпечення, оскільки тривала експлуатація без перерви безпосередньо впливає як на загальні витрати електроенергії, так і на довговічність компонентів, при цьому компактні персональні комп’ютери демонструють значно нижчий рівень енергоспоживання порівняно з класичними серверними системами, що зазвичай споживають суттєво більше електроенергії залежно від конфігурації та навантаження, тоді як компактні рішення можуть працювати у значно економнішому діапазоні, що робить їх доцільними для постійної експлуатації у невеликих інфраструктурах.

Досягнення високого рівня енергоефективності забезпечується за рахунок використання енергоощадних процесорів, які часто базуються на мобільних архітектурах із оптимізованим енергоспоживанням, інтегрованих графічних підсистем, що не потребують додаткових дискретних компонентів, а також ефективних систем живлення із високим коефіцієнтом корисної дії, що дозволяє мінімізувати втрати енергії під час роботи, у поєднанні з продуманими системами охолодження, які забезпечують стабільний температурний режим навіть за тривалого навантаження, що є критично важливим фактором для запобігання перегріву та деградації компонентів [18].

Обмежені габарити компактних пристроїв накладають певні обмеження на можливості тепловідведення, унаслідок чого особлива увага приділяється підтримці оптимального температурного режиму експлуатації, що досягається як за рахунок апаратних рішень, так і шляхом програмного контролю навантаження, який дозволяє уникати перевищення допустимих значень використання процесора та пам’яті, забезпечуючи стабільну роботу системи без зниження продуктивності або аварійних зупинок.

З точки зору ресурсної ємності сучасні компактні ПК здатні забезпечити достатній рівень продуктивності для більшості задач, характерних для мікросервісної архітектури, при цьому використання контейнеризації дозволяє більш ефективно розподіляти доступні ресурси між окремими сервісами, оскільки кожен контейнер використовує лише необхідну частину обчислювальної потужності та оперативної пам’яті, залишаючи інші ресурси доступними для паралельного виконання додаткових задач, що дозволяє уникнути перевитрат та забезпечити більш раціональне використання апаратної платформи.

Такий підхід дозволяє досягти балансу між енергоефективністю та продуктивністю, що робить компактні пристрої оптимальним вибором для реалізації edge-середовищ, особливо у випадках, коли критичним фактором є зниження експлуатаційних витрат і спрощення процесу розгортання інфраструктури без втрати функціональних можливостей.

Вимоги до апаратної віртуалізації та мережевих інтерфейсів. Ефективна реалізація мікросервісної архітектури на базі контейнеризації також значною мірою залежить від апаратних можливостей пристрою, серед яких важливе місце займає підтримка технологій апаратної віртуалізації, що реалізується за допомогою сучасних інструкцій процесора, які дозволяють виконувати ізольовані середовища з мінімальними накладними витратами, при цьому навіть за умови використання контейнерів, які не потребують повноцінної віртуалізації, наявність такої підтримки відкриває можливість створення гібридних середовищ, у яких поєднуються віртуальні машини та контейнери, забезпечуючи додатковий рівень ізоляції та гнучкість у побудові інфраструктури.

Сучасні процесори також підтримують розширення для ефективного управління пам’яттю, що дозволяє підвищити продуктивність віртуалізованих середовищ і зменшити накладні витрати під час виконання ізольованих процесів, що є важливим фактором у системах із високим рівнем навантаження або обмеженими ресурсами [19].

Не менш важливим аспектом є характеристики мережевих інтерфейсів, оскільки саме вони визначають можливості пристрою щодо обробки мережевого трафіку, при цьому значення мають такі параметри, як:

1. пропускна здатність;
2. підтримка апаратного прискорення обробки пакетів;
3. сумісність із драйверами операційної системи;
4. можливість роботи з віртуальними мережевими інтерфейсами.

Вони ж використовуються для організації захищених каналів зв’язку та реалізації програмно-визначених мереж.

Підтримка віртуальних інтерфейсів типу TUN або TAP дозволяє здійснювати обробку мережевих пакетів на рівні ядра операційної системи, забезпечуючи гнучкість у налаштуванні маршрутизації та створенні ізольованих каналів передачі даних, що є необхідним для реалізації сучасних мережевих рішень, включаючи VPN та mesh-архітектури, при цьому додатково враховується можливість налаштування параметрів маршрутизації, таких як IP forwarding, що дозволяє використовувати пристрій у ролі маршрутизатора або шлюзу між різними сегментами мережі [19].

У середовищах, де пристрої функціонують за умов мережевої трансляції адрес, виникає потреба у використанні механізмів обходу NAT та забезпеченні стабільного з’єднання між вузлами, що накладає додаткові вимоги до якості реалізації мережевих інтерфейсів і їхньої здатності підтримувати сучасні протоколи взаємодії, унаслідок чого апаратні характеристики безпосередньо впливають на ефективність роботи всієї мікросервісної інфраструктури.

Узгоджене поєднання енергоефективних апаратних платформ, підтримки сучасних технологій віртуалізації та розвинених мережевих можливостей формує

основу для побудови стабільних, продуктивних і безпечних програмно-апаратних комплексів, що відповідають вимогам сучасних розподілених систем і дозволяють реалізувати гнучкі сценарії розгортання сервісів у межах edge-інфраструктури.

1.5 Безпека безпарольного доступу до апаратної інфраструктури

В умовах розвитку інформаційних систем питання безпеки доступу до апаратної інфраструктури набуває особливої актуальності у зв'язку зі зростанням кількості розподілених вузлів, контейнеризованих сервісів і віддалених користувачів, що взаємодіють із системою через мережу, при цьому традиційні підходи до автентифікації, які тривалий час використовувалися як базові механізми захисту, поступово втрачають ефективність у середовищах, де доступ до ресурсів здійснюється у режимі 24/7 та потребує високого рівня гнучкості, масштабованості та централізованого контролю, унаслідок чого виникає необхідність переходу до більш сучасних моделей забезпечення безпеки, орієнтованих не лише на перевірку факту доступу, але й на контекст його використання.

Вразливості класичної автентифікації (RSA-ключі та паролі). Класичні методи автентифікації, що базуються на використанні паролів або криптографічних ключів, залишаються поширеними у багатьох системах, однак їх застосування супроводжується низкою суттєвих обмежень і потенційних вразливостей, які можуть бути використані для несанкціонованого доступу до ресурсів, при цьому найбільш вразливим елементом такої моделі є людський фактор, оскільки використання слабких або повторюваних паролів, їх зберігання у небезпечних місцях або передача через незахищені канали створює передумови для реалізації атак типу brute force або фішингу, що може призвести до компрометації облікових даних і отримання повного доступу до системи.

Навіть за умови використання складних паролів ризик компрометації залишається актуальним, оскільки централізоване управління великою кількістю

облікових записів ускладнюється, а необхідність регулярного оновлення та контролю політик доступу створює додаткове навантаження на адміністрування, унаслідок чого зростає ймовірність помилок конфігурації, що також може бути використано як вектор атаки (див. рис. 1.9).

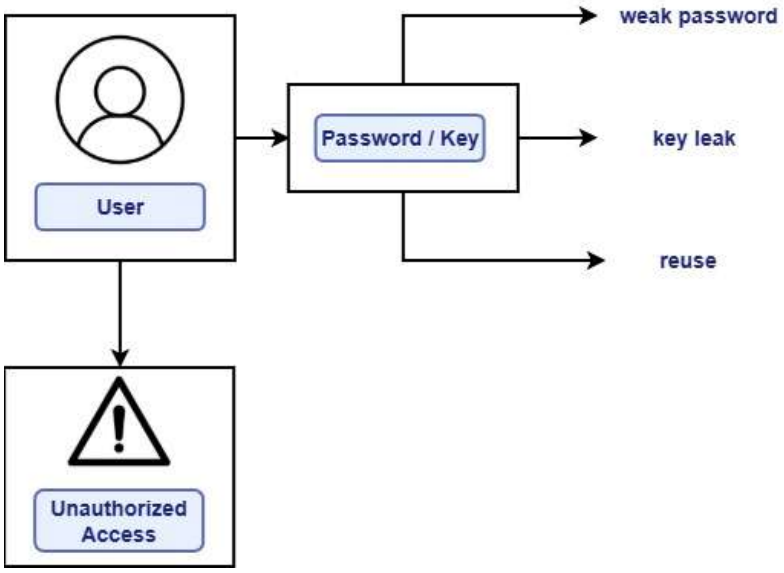


Рисунок 1.9 – Вразливості класичної SSH-автентифікації

Альтернативою паролям традиційно виступає автентифікація на основі криптографічних ключів, зокрема SSH із використанням RSA, яка дозволяє уникнути передачі пароля мережею та підвищує рівень безпеки за рахунок застосування асиметричного шифрування, однак і цей підхід має низку обмежень, пов’язаних із необхідністю зберігання приватних ключів на клієнтських пристроях, що створює ризик їх компрометації у разі втрати або несанкціонованого доступу до пристрою, при цьому управління ключами у великих інфраструктурах стає складним завданням через відсутність централізованого контролю, складність їх ротації та відкликання [20].

Особливістю такого підходу є прив’язка доступу до конкретного пристрою, а не до ідентичності користувача, що означає можливість отримання доступу будь-якою особою, яка має відповідний ключ, незалежно від того, чи є вона авторизованим користувачем системи, унаслідок чого у разі передачі ключа або

компрометації пристрою система не має вбудованих механізмів додаткової перевірки, що суттєво знижує рівень безпеки в умовах сучасних розподілених середовищ (див. рис. 1.10).

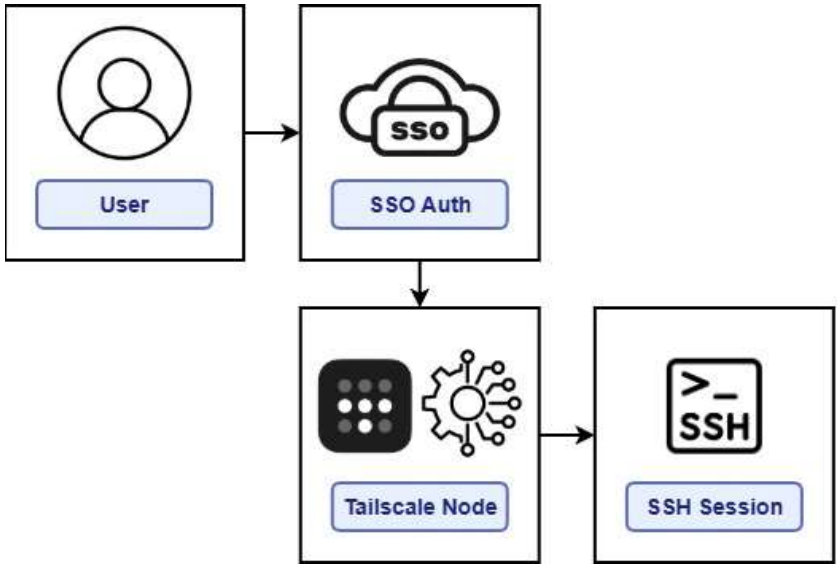


Рисунок 1.10 – Принцип роботи Tailscale SSH

Зростання масштабів інфраструктури, кількості вузлів та користувачів призводить до ускладнення управління доступом, що формує потребу:

1. впровадженні централізованих механізмів контролю;
2. автоматичної ротації облікових даних;
3. можливості оперативного відкликання доступу.

Все це є складним для реалізації в рамках класичних підходів, унаслідок чого виникає необхідність переходу до моделей, які базуються на ідентичності користувача як основному факторі автентифікації.

У відповідь на ці виклики формується підхід, що відповідає концепції Zero Trust, у межах якого доступ до ресурсів надається не на основі мережевого розташування або наявності ключа, а з урахуванням ідентичності користувача, контексту підключення та політик безпеки, що дозволяє значно підвищити рівень захисту інфраструктури, при цьому одним із прикладів реалізації такого підходу є механізми безпарольного доступу, які використовують інтеграцію з системами

керування ідентичністю та забезпечують автентифікацію на основі централізованих облікових записів.

Такий підхід дозволяє усунути залежність від статичних паролів і ключів, забезпечити централізоване управління доступом, впровадити багатофакторну автентифікацію та автоматизувати процеси контролю безпеки, що особливо важливо для систем із великою кількістю вузлів та користувачів, де ручне адміністрування стає неефективним і підвищує ризик помилок, унаслідок чого перехід до безпарольних механізмів автентифікації розглядається як один із напрямів підвищення безпеки сучасних програмно-апаратних комплексів.

Концепція Tailscale SSH – авторизація на рівні ідентичності користувача, а не пристрою. Сучасний підхід до організації безпечного віддаленого доступу до інфраструктури поступово зміщується у напрямку використання механізмів, де основним елементом контролю виступає не пристрій і не статичні облікові дані, а безпосередньо ідентичність користувача, що логічно продовжує розвиток концепції Zero Trust і знаходить своє практичне відображення у технології Tailscale SSH, яка інтегрує мережеву взаємодію, криптографічний захист та централізоване управління доступом у єдину систему без використання традиційних паролів або SSH-ключів, при цьому сам механізм доступу базується на автентифікації через зовнішніх провайдерів ідентичності, таких як Google, Microsoft або інші сервіси, що підтримують SSO, у результаті чого кожна спроба підключення до ресурсу супроводжується перевіркою облікового запису користувача, його приналежності до відповідних груп та відповідності заданим політикам доступу.

У такій моделі доступ більше не прив’язується до конкретного пристрою або файлу ключа, а визначається виключно через цифрову ідентичність користувача, що дозволяє централізовано управляти правами доступу, швидко змінювати або відкликати їх у разі необхідності, а також застосовувати додаткові механізми безпеки, включаючи багатофакторну автентифікацію, перевірку стану пристрою та контексту підключення, завдяки чому значно зменшується ризик несанкціонованого доступу навіть у випадку компрометації окремого вузла або мережевого сегмента [21].

Процес встановлення з’єднання у Tailscale SSH відбувається у декілька логічно пов’язаних етапів, де спочатку користувач проходить автентифікацію через SSO-провайдера, після чого його пристрій приєднується до приватної мережі типу Tailnet і отримує відповідні параметри доступу, далі при ініціації SSH-з’єднання система перевіряє політики доступу, визначені у вигляді ACL, і лише у випадку відповідності встановлюється захищений канал зв’язку, який функціонує без використання традиційних паролів або статичних ключів, а всі дані передаються через зашифровані тунелі з використанням сучасних криптографічних алгоритмів, що забезпечує конфіденційність і цілісність інформації навіть у незахищених мережах.

Додатковою особливістю такого підходу є повна відсутність необхідності відкриття портів на маршрутизаторах або виконання складних налаштувань мережевої маршрутизації, оскільки взаємодія між вузлами відбувається через захищене середовище з автоматичним встановленням з’єднань, що суттєво зменшує площину атаки та виключає можливість прямого сканування сервісів із зовнішнього середовища, а також мінімізує ризики, пов’язані з помилками конфігурації.

Важливим аспектом є також підтримка механізмів журналювання та аудиту, які дозволяють фіксувати всі спроби підключення, виконані дії та зміну політик доступу, що забезпечує прозорість функціонування системи та можливість оперативного реагування на інциденти безпеки, при цьому централізоване управління доступом значно спрощує адміністрування, оскільки відсутня необхідність у генерації, розповсюдженні та зберіганні криптографічних ключів на клієнтських пристроях, що традиційно було одним із найскладніших і найбільш вразливих елементів класичних SSH-рішень.

У підсумку такий підхід формує якісно новий рівень організації доступу до апаратної інфраструктури, де поєднуються:

1. принципи zero trust;
2. централізоване управління ідентичностями;
3. автоматизовані механізми шифрування і встановлення з’єднань.

2. АРХІТЕКТУРНЕ ПРОЕКТУВАННЯ ТА РОЗГОРТАННЯ СИСТЕМИ НА БАЗІ НЕТТОПА

2.1 Архітектура програмно-апаратного комплексу та топологія мережі

Проєктування програмно-апаратного комплексу віддаленого доступу до сервісної інфраструктури передбачає поєднання:

1. апаратних ресурсів компактного обчислювального вузла;
2. віртуалізованого середовища;
3. програмно визначеної мережі.

Остання ж забезпечує безпечну взаємодію між усіма компонентами системи, причому архітектура формується з урахуванням обмежених ресурсів, потенційних апаратних несправностей і необхідності стабільного доступу в умовах ізольованих або складних мережевих середовищ, включаючи сценарії з подвійним NAT, де класичні підходи на основі статичних IP-адрес або пробросу портів демонструють нестабільну або обмежену роботу, тому основною концепцією стає побудова багаторівневої системи, в якій фізичний вузол виступає базовим середовищем, гіпервізор відповідає за ізоляцію та управління ресурсами, віртуальні машини забезпечують виконання сервісів, а мережевий рівень реалізується через Zero Trust-підхід із використанням Tailscale, що дозволяє забезпечити доступ навіть у випадках повної мережевої ізоляції (див. рис. 2.1).

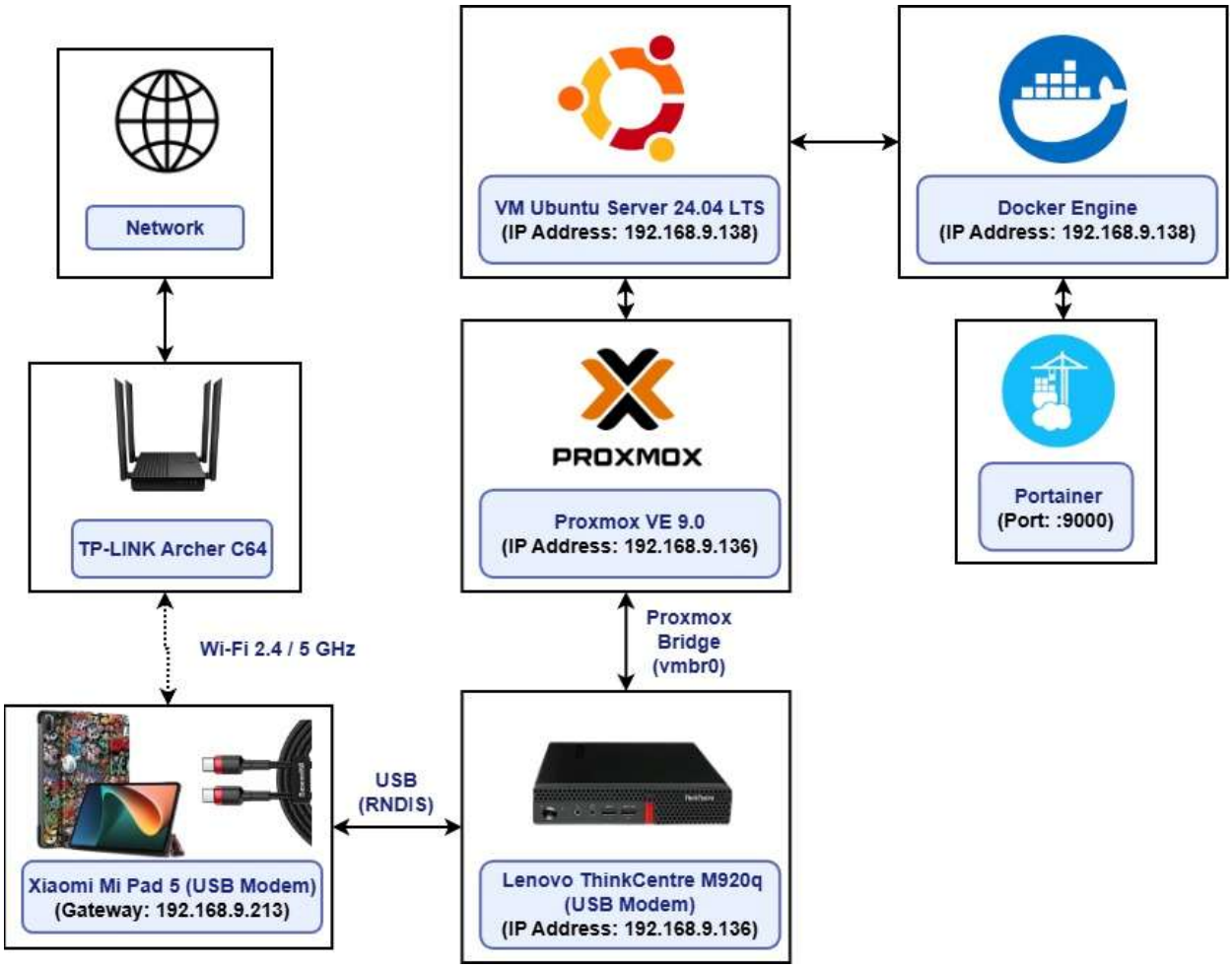


Рисунок 2.1 – Архітектура програмно-апаратного комплексу віддаленого доступу до серверної інфраструктури на базі Tailscale

Апаратна специфікація вузла. У якості базового серверного вузла використовується неттоп Lenovo ThinkCentre M920q, який належить до класу компактних корпоративних обчислювальних пристроїв і має конфігурацію:

1. процесор Intel Core i5-8400T.
2. Samsung 2x16ГБ (32ГБ) оперативної пам’яті стандарту DDR4;
3. Samsung PM981 обсягом 512 ГБ (NVME-накопичувач).

При цьому вибір такого форм-фактору обумовлений сукупністю технічних і економічних факторів, серед яких ключовими є:

— компактність;

— низьке енергоспоживання;

— розміщення в обмежених умовах (без необхідності серверної стійки);

— достатній рівень продуктивності (для виконання задач віртуалізації).

Сама наявність 32 ГБ оперативної пам’яті дозволяють одночасно запускати декілька віртуальних машин, що критично для ізоляції сервісів і тестових середовищ, тоді як використання NVMe-накопичувача забезпечує високу швидкість доступу до даних і позитивно впливає на загальну продуктивність системи (див. рис. 2.2, табл. 2.1).



Рисунок 2.2 – Апаратна конфігурація серверного вузла
на базі Lenovo ThinkCentre M920q

Разом із цим у процесі практичного використання було виявлено низку апаратних обмежень, які безпосередньо вплинули на архітектурні рішення, зокрема проблема із вбудованою мережею картою на базі контролера Intel I219-LM, яка некоректно визначається або працює нестабільно в середовищі гіпервізора, що призводить до відсутності повноцінного мережевого інтерфейсу, подібна ситуація

є характерною для певних ревізій компактних пристроїв, включаючи моделі цієї серії, і створює необхідність застосування альтернативних підходів до організації мережевого підключення, серед яких можливими варіантами виступають використання зовнішніх USB-адаптерів або встановлення додаткової мережевої карти через PCIe riser, однак у межах даної реалізації було обрано інший підхід, що дозволяє не лише обійти апаратні обмеження, а й створити ускладнене середовище для тестування мережевих технологій.

Таблиця 2.1 – Апаратна конфігурація серверного вузла

Компонент	Модель	Характеристики
CPU	Intel Core i5-8400T	6 ядер, до 3.3 ГГц
RAM	Samsung DDR4	32 ГБ (2×16 ГБ)
SSD	Samsung PM981	512 ГБ NVMe
NIC	Intel I219-LM	Gigabit Ethernet
Платформа	Lenovo ThinkCentre M920q	x86_64, компактний форм-фактор
Додатково	USB Ethernet / PCIe NIC	Альтернативний інтерфейс

Програмний стек гіпервізора. Програмний стек гіпервізора у даній архітектурі формується навколо використання платформи Proxmox VE, яка встановлюється безпосередньо на фізичний вузол і виступає центральним середовищем керування всією інфраструктурою, при цьому обрана версія належить до актуальної гілки 9.1.1, що забезпечує сумісність із сучасними компонентами та підтримку необхідних механізмів віртуалізації і мережевої ізоляції, а сам процес розгортання починається з базової інсталяції гіпервізора, після чого формується середовище, у якому створюються віртуальні машини, розподіляються ресурси та забезпечується контроль доступу до них, де важливим елементом автоматизації виступає механізм Cloud-init, що дозволяє створити універсальний шаблон інстансу на базі Ubuntu Server версії 24.04 LTS із попередньо визначеними параметрами, такими як користувачі, SSH-ключі, мережеві налаштування та базові конфігурації системи, і такий підхід дозволяє уникнути необхідності ручного налаштування кожної нової віртуальної машини, оскільки після створення базового образу виконується його конвертація у формат, придатний для використання в середовищі гіпервізора, далі інтегрується Cloud-init, після чого формується

шаблон, з якого надалі здійснюється клонування інстансів із автоматичним застосуванням конфігурації при першому запуску, що значно скорочує час розгортання та зменшує ризик помилок конфігурації, аналогічно підходам, які застосовуються у хмарних інфраструктурах [22-23].

У межах створеної віртуальної машини розгортається контейнерне середовище на базі Docker, яке використовується для запуску прикладних сервісів, зокрема сервісу керування контейнерами Portainer, що виконує роль тестового ресурсу доступу, при цьому важливою особливістю архітектури є те, що Tailscale не розгортається як контейнер усередині віртуальної машини, хоча такий підхід є типовим для багатьох реалізацій, а інтегрується безпосередньо на рівні гіпервізора, що дозволяє уникнути додаткового рівня залежностей та спростити мережеву взаємодію між компонентами, а також забезпечити більш стабільний доступ до самого вузла навіть у випадках, коли внутрішні мережеві інтерфейси працюють нестабільно або мають апаратні обмеження.

Топологія ізольованого середовища. Топологія ізольованого середовища характеризується використанням нетипової мережевої архітектури, побудованої на основі подвійної трансляції мережевих адрес, де в якості тимчасового мережевого шлюзу застосовується планшет Xiaomi Mi Pad 5, який працює у режимі USB-модему та одночасно підключений до основного маршрутизатора через Wi-Fi, у результаті чого формується окрема підмережа з власною адресацією, наприклад 192.168.9.0/24, у якій сам планшет виступає як шлюз із адресою типу 192.168.9.213, тоді як фізичний вузол отримує IP-адресу 192.168.9.136, і подальше розширення цієї підмережі відбувається через механізм Linux Bridge у середовищі гіпервізора, що дозволяє підключити до неї віртуальні машини, для яких через Cloud-init задається статична адресація, наприклад 192.168.9.138 для інстансу Ubuntu Server, у межах якого функціонує контейнерне середовище та розгорнутий сервіс.

У результаті формується багаторівнева ієрархія мережі, у якій глобальна мережа Інтернет підключається до домашнього маршрутизатора з адресою 192.168.0.1, далі планшет, підключений через Wi-Fi, створює власну ізольовану підмережу та виконує роль проміжного NAT-рівня, після чого через USB-з’єднання

типу RNDIS передає мережевий доступ до неттопа, де вже формується віртуальне середовище з власною внутрішньою адресацією, а далі в межах цього середовища функціонують віртуальні машини та контейнеризовані сервіси, доступ до яких можливий лише зсередини цієї ізольованої структури, при цьому навіть за наявності фізичного підключення до основного маршрутизатора відсутня можливість прямого доступу до вузлів підмережі 192.168.9.0/24, оскільки пакети не можуть пройти через подвійний NAT без спеціалізованих механізмів маршрутизації або тунелювання (див. рис. 2.3).



Рисунок 2.3 – Ізоляція середовища через подвійним NAT

Така топологія створює експериментальне середовище підвищеної складності, у якому традиційні підходи, такі як порт-форвардинг або прямий доступ до сервісів, демонструють обмежену ефективність або взагалі не працюють, і саме використання подібної схеми дозволяє перевірити здатність сучасних програмно-визначених мереж забезпечувати стабільний доступ до ресурсів навіть у випадках, коли інфраструктура повністю ізольована кількома рівнями трансляції адрес, що наближує умови тестування до реальних сценаріїв використання мобільних або обмежених мережевих середовищ.

Гібридна інтеграція Tailscale. З урахуванням описаної архітектури ізольованого середовища та наявності подвійного NAT природним етапом розвитку системи стає інтеграція програмно-визначеної мережі на основі Tailscale, що забезпечує стабільну взаємодію між клієнтами та серверним вузлом незалежно від фізичної топології та обмежень транспортного середовища, причому на відміну від типових підходів, де Tailscale розгортається всередині контейнера або окремої віртуальної машини, у даній реалізації обрано гібридну модель інтеграції на рівні

хост-системи, що дозволяє безпосередньо включити сам гіпервізор у mesh-мережу та зробити його повноцінним учасником Tailnet із власною адресацією та політиками доступу, завдяки чому формується більш прямий та передбачуваний шлях доступу до керуючого вузла інфраструктури.

Інтеграція на рівні хоста логічно випливає з апаратних обмежень, зокрема нестабільної роботи вбудованого мережевого інтерфейсу, оскільки у випадку розгортання Tailscale лише всередині віртуальних машин або контейнерів виникає залежність від коректності функціонування внутрішніх мережевих мостів та маршрутизації, тоді як встановлення безпосередньо на Proxmox дозволяє отримати незалежний канал керування навіть у випадках часткової деградації мережевої підсистеми, а також мінімізує кількість проміжних рівнів обробки трафіку, що позитивно впливає на затримки та загальну стабільність з'єднання.

Процес інтеграції починається з інсталяції клієнта Tailscale через офіційний репозиторій із попереднім очищенням конфігурації пакетних джерел від enterprise-компонентів, які можуть викликати конфлікти залежностей у середовищі Proxmox, після чого виконується авторизація вузла через зовнішнього провайдера ідентифікації, наприклад Google SSO, у результаті чого фізичний сервер додається до Tailnet як окремий пристрій із власним унікальним ідентифікатором та отримує внутрішню адресу у діапазоні 100.x.x.x, яка використовується для взаємодії з іншими вузлами мережі незалежно від їхнього фактичного розташування.

Після успішної авторизації формується логічна мережа, у якій клієнтські пристрої, такі як персональні комп'ютери або мобільні пристрої, підключаються до Tailnet через відповідний клієнт і отримують можливість адресувати серверний вузол безпосередньо за його Tailscale IP або доменним ім'ям, визначеним через механізм MagicDNS, причому сам нettоп виконує не лише роль кінцевого вузла, але й виступає як точка доступу до ізольованої підмережі, що знаходиться за ним.

У цьому контексті реалізується механізм IP forwarding та анонсування маршрутів, який дозволяє транслювати інформацію про доступність підмережі 192.168.9.0/24 у межах Tailnet, внаслідок чого клієнти, підключені до mesh-мережі, отримують можливість взаємодіяти не лише з самим гіпервізором, але й з усіма

віртуальними машинами, що функціонують у межах ізольованого середовища, включаючи Ubuntu Server із розгорнутим Docker Engine та сервісом Portainer, доступ до якого здійснюється через порт 9000 [22-23].

Формується багаторівнева схема доступу, у якій зовнішній клієнт підключається до Tailnet, отримує внутрішню адресу, після чого встановлює захищене з'єднання з гіпервізором, який у свою чергу маршрутизує трафік до відповідної віртуальної машини у підмережі 192.168.9.0/24, причому весь цей процес відбувається без необхідності відкриття портів на маршрутизаторі або планшеті, що функціонує як USB-модем, а також без налаштування класичних механізмів port forwarding (див. рис. 2.4).



Рисунок 2.4 – Інтеграція Tailscale у локальну мережу

Особливістю такої архітектури стає здатність системи функціонувати в умовах подвійного NAT, де класичні VPN-рішення часто стикаються з проблемами встановлення з’єднання або потребують додаткових конфігурацій, тоді як Tailscale використовує механізми NAT traversal, включаючи UDP hole punching та fallback через DERP-сервери, що забезпечує встановлення стабільного з’єднання навіть у випадках повністю ізольованих мережевих середовищ, у яких відсутній прямий вхідний доступ до вузлів [21].

Додатково реалізується централізована модель контролю доступу на основі ACL, що дозволяє визначати політики взаємодії між вузлами Tailnet, обмежуючи доступ лише до конкретних сервісів або адрес, наприклад надання доступу виключно до веб-інтерфейсу Portainer без можливості взаємодії з іншими ресурсами підмережі, що значно підвищує рівень безпеки та відповідає концепції Zero Trust, у межах якої кожен запит до ресурсу проходить перевірку незалежно від розташування клієнта.

Таким чином формується гібридна модель мережевої інтеграції, у якій поєднується фізична ізоляція інфраструктури, віртуалізоване середовище виконання та програмно-визначена mesh-мережа, що забезпечує прозорий, безпечний та стабільний доступ до серверних ресурсів навіть у складних умовах мережевих обмежень, причому архітектура залишається масштабованою та адаптивною до змін, що дозволяє використовувати її як у навчальних сценаріях, так і в реальних інфраструктурних рішеннях із підвищеними вимогами до надійності та безпеки.

2.2 Покрокова побудова та конфігурація серверного вузла

Покрокова побудова та конфігурація серверного вузла характеризується як послідовна реалізація програмно-апаратного комплексу віддаленого доступу, у межах якої організація інфраструктури виконується з урахуванням:

1. реальних апаратних обмежень;
2. нестабільності стандартних мережевих інтерфейсів;
3. специфіки ізольованого середовища функціонування.

Все це потребує застосування адаптивних підходів до побудови мережевої взаємодії та розгортання сервісів, при цьому загальна логіка формування системи передбачає поступовий перехід від базового рівня мережевого доступу до створення віртуалізованого середовища, подальшої контейнеризації прикладних компонентів і завершальної інтеграції у захищену mesh-мережу, здатну забезпечити стабільний доступ навіть за наявності багаторівневої ізоляції.

Налаштування мережевого шлюзу та обхід апаратних обмежень. Перший етап розгортання пов'язаний із вирішенням критичної проблеми відсутності стабільного мережевого інтерфейсу на рівні гіпервізора, що виникає через некоректну роботу вбудованого мережевого контролера, унаслідок чого пряме підключення до фізичної мережі стає неможливим, а застосування традиційних підходів у вигляді встановлення додаткових PCI-мережевих карт або використання USB Ethernet-адаптерів хоча і залишається технічно допустимим, у даній реалізації замінюється альтернативним рішенням із використанням планшета як тимчасового мережевого шлюзу, що дозволяє не лише обійти апаратні обмеження, але й створити контрольоване середовище для тестування поведінки системи в умовах подвійного NAT [22].

Планшет, зокрема Xiaomi Mi Pad 5, підключається до неттопа через USB-інтерфейс та функціонує у режимі USB-модему з використанням протоколу RNDIS, у результаті чого автоматично формується окрема підмережа з власною адресацією, у межах якої сам пристрій виступає одночасно як DHCP-сервер і шлюз за замовчуванням, забезпечуючи розподіл IP-адрес та маршрутизацію трафіку, причому доступ до глобальної мережі здійснюється через Wi-Fi-з'єднання планшета з основним маршрутизатором, що формує додатковий рівень трансляції адрес і створює характерну ієрархію, де зовнішній інтернет трафік проходить через домашній маршрутизатор, далі через безпровідне з'єднання потрапляє на планшет,

після чого передається до неттопа через USB-інтерфейс, утворюючи другий рівень NAT (див. рис. 2.5).

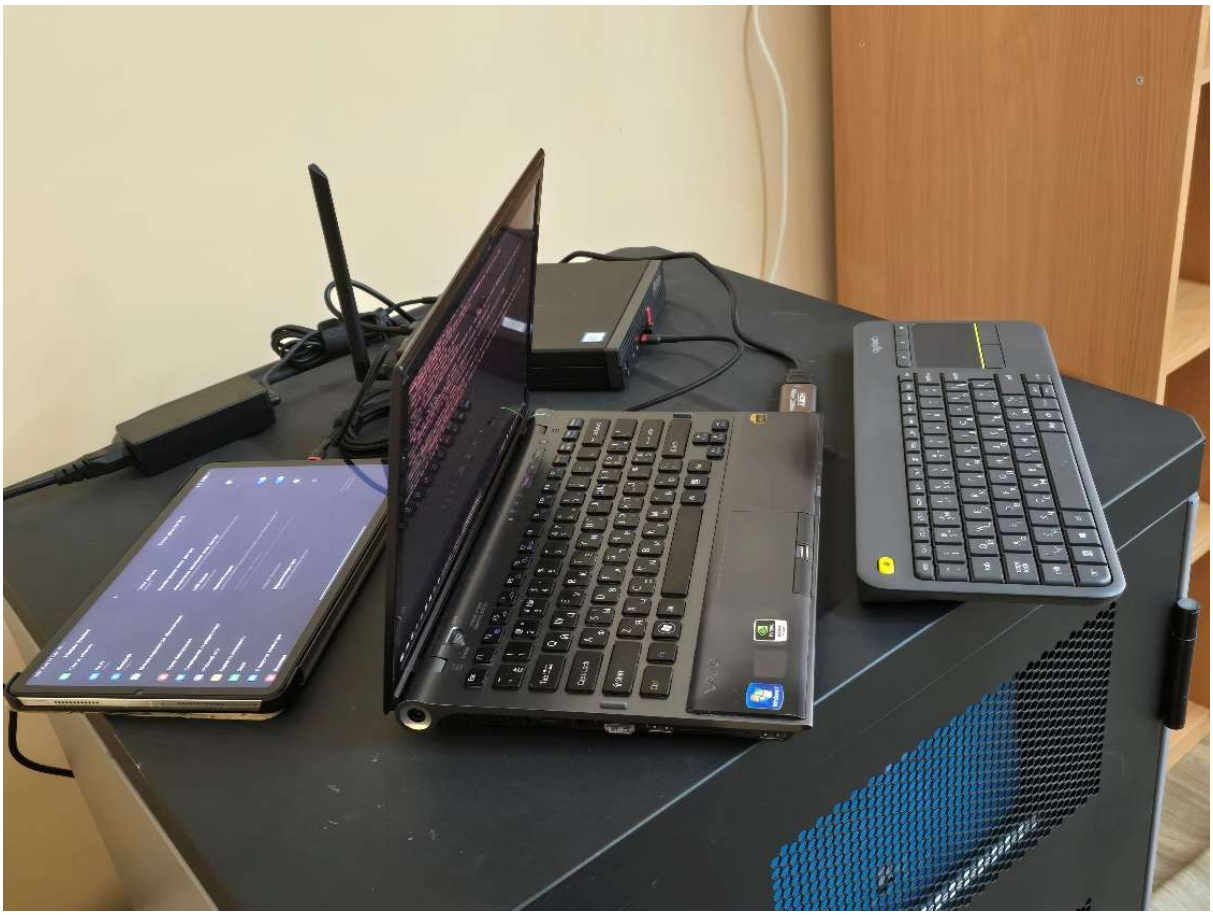
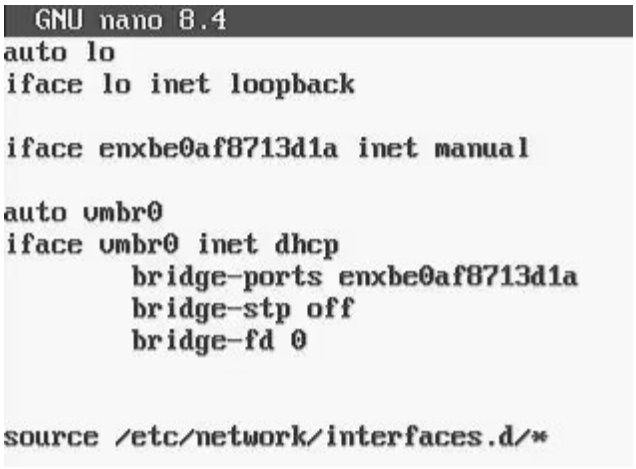


Рисунок 2.5 – Підключення через USB-модем

У сформованій конфігурації підмережа 192.168.9.0/24 ізольована від основної локальної мережі, а неттоп отримує IP-адресу, наприклад 192.168.9.136, тоді як планшет виступає шлюзом із адресою на кшталт 192.168.9.213, що унеможливлює прямий доступ до вузлів цієї підмережі з інших пристроїв у мережі маршрутизатора, створюючи експериментальне середовище подвійного NAT, у якому класичні методи доступу, такі як пряме підключення або port forwarding, стають непридатними без додаткових налаштувань (див. рис. 2.6).




```
root@phy-home-srv-pve-tiny-01:/etc/network# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
10: enxbe0af8713d1a: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master vmbr0 state UNKNOWN group default qlen 1000
    link/ether be:0a:f8:71:3d:1a brd ff:ff:ff:ff:ff:ff
11: vmbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether be:0a:f8:71:3d:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.9.136/24 brd 192.168.9.255 scope global dynamic vmbr0
       valid_lft 3583sec preferred_lft 3583sec
    inet6 fe80::bc0a:f8ff:fe71:3d1a/64 scope link proto kernel_ll
       valid_lft forever preferred_lft forever
root@phy-home-srv-pve-tiny-01:/etc/network# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=59 time=23.7 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=59 time=22.3 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=59 time=27.8 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=59 time=22.1 ms
^C
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 22.125/23.973/27.804/2.290 ms
root@phy-home-srv-pve-tiny-01:/etc/network# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=63 time=22.6 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=63 time=3.56 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=63 time=21.6 ms
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 3.557/15.904/22.599/8.741 ms
root@phy-home-srv-pve-tiny-01:/etc/network# _
```

Рисунок 2.6 – Налаштування мережевого інтерфейсу

Подальша конфігурація передбачає налаштування мережевої підсистеми гіпервізора шляхом створення Linux Bridge, що виступає як програмний комутатор для об’єднання фізичних та віртуальних інтерфейсів у єдину логічну структуру, у межах якої формується міст із типовою назвою "vmbr0", до якого прив’язується інтерфейс USB-модему, отриманий від планшета, що дозволяє передавати мережевий трафік між хост-системою та віртуальними машинами без необхідності додаткових рівнів трансляції [22].

Усі віртуальні машини підключаються до цього мосту як до основного мережевого інтерфейсу, що забезпечує отримання IP-адрес у межах підмережі або

використання попередньо визначеної статичної адресації, а сама модель взаємодії передбачає прозору маршрутизацію трафіку між гіпервізором та гостьовими системами, що дозволяє централізовано керувати мережевими параметрами, контролювати ізоляцію та забезпечувати передбачуваність поведінки всієї інфраструктури навіть у складних умовах функціонування.

Розгортання віртуального середовища. Після організації мережевого шлюзу та забезпечення доступу до інтернету через USB-модем наступним етапом стає формування віртуального середовища, яке базується на використанні Ubuntu Server як універсальної платформи для розгортання контейнеризованих сервісів та мережевих компонентів, причому замість ручного встановлення операційної системи для кожної окремої віртуальної машини застосовується підхід із використанням попередньо підготовленого Cloud-образу, що дозволяє автоматизувати процес розгортання та забезпечити відтворюваність конфігурації (див. табл. 2.2).

Таблиця 2.2 – Параметри віртуального середовища та контейнерів

Тип	ОС/Платформа	CPU/RAM	Мережевий режим
Гіпервізор	Proxmox VE 9.1.1	Host	Bridge
VM	Ubuntu Server 24.04	2 vCPU / 8 ГБ	Bridge/NAT
Контейнер	Docker	Спільні ресурси	Bridge
Сервіс	Portainer	Мінімальні	Bridge
Сервіс	Tailscale (host)	Host	Overlay
Шаблон	Cloud-init	-	-

Процес починається із завантаження Cloud-образу Ubuntu Server 24.04 LTS із довгостроковою підтримкою, після чого на його основі створюється базова віртуальна машина у середовищі гіпервізора, до якої підключається механізм Cloud-init, що забезпечує можливість попереднього визначення параметрів системи, включаючи створення користувачів, налаштування мережевих інтерфейсів, додавання SSH-ключів та виконання початкових сценаріїв конфігурації, у результаті чого формується універсальний шаблон, придатний для подальшого масштабування [22-23].

Сформований інстанс перетворюється у шаблон, який використовується для клонування нових віртуальних машин без необхідності повторного проходження етапів інсталяції, що значно скорочує час розгортання та мінімізує ризики помилок, пов’язаних із ручною конфігурацією, причому ключовою особливістю такого підходу стає використання Cloud-init для автоматичного призначення параметрів під час першого запуску кожного клонованого екземпляра (див. рис. 2.7) [23].

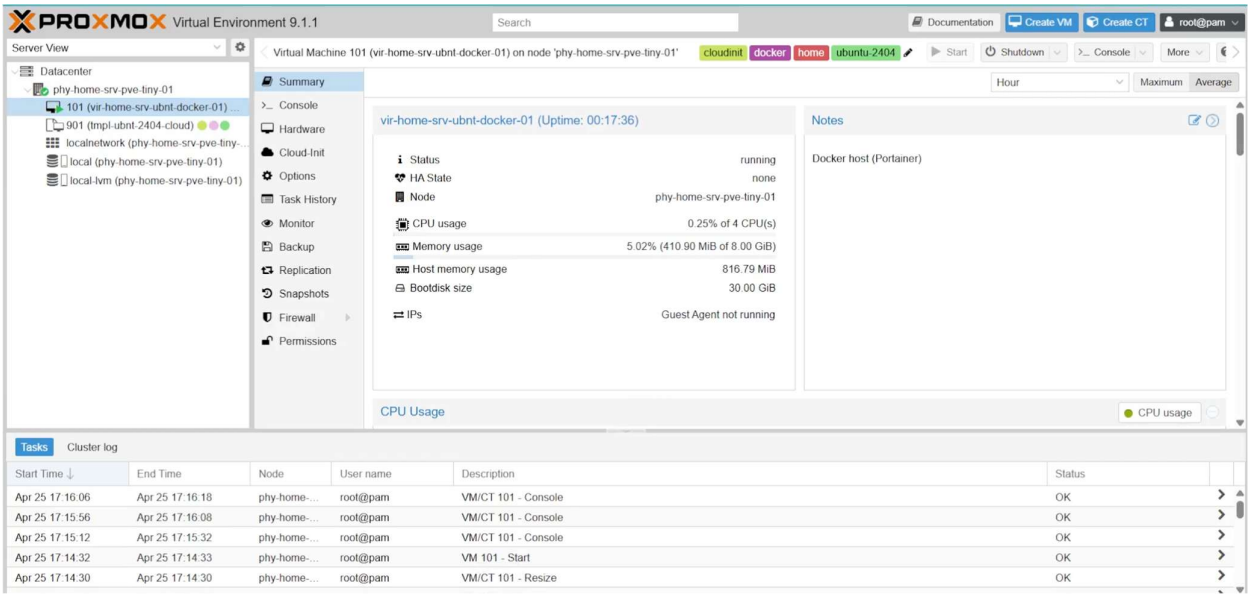


Рисунок 2.7 – Створення віртуальної машини

У межах даної архітектури особливу увагу приділено налаштуванню статичної IP-адресації, оскільки в умовах ізольованої підмережі 192.168.9.0/24 стабільність адрес є критичною для подальшої маршрутизації та інтеграції з mesh-мережею, у зв’язку з чим для віртуальних машин задається фіксована адреса, наприклад 192.168.9.138, що дозволяє уникнути змін адресації при перезапусках і забезпечує передбачуваність мережевої взаємодії між компонентами системи.

Додатково в межах Cloud-init інтегрується механізм автентифікації на основі SSH-ключів, які генеруються заздалегідь, наприклад у середовищі Windows, та додаються до конфігурації віртуальної машини, у результаті чого забезпечується можливість безпарольного доступу до системи, що підвищує рівень безпеки та відповідає сучасним практикам адміністрування серверної інфраструктури.

Впровадження Tailscale на рівні гіпервізора. Наступним логічним етапом стає інтеграція всієї системи у mesh-мережу шляхом встановлення Tailscale безпосередньо на гіпервізор Proxmox, що дозволяє забезпечити доступ до всієї інфраструктури незалежно від стану окремих віртуальних машин та особливостей їхньої конфігурації, а також створює єдину точку входу для віддаленого адміністрування.

Процес впровадження Tailscale включає оновлення системи та налаштування пакетного менеджера з додаванням офіційного репозиторію, після чого виконується встановлення клієнта та його ініціалізація, у результаті чого генерується посилання для авторизації, що відкривається у браузері та дозволяє виконати вхід через зовнішнього провайдера ідентифікації, зокрема Google, унаслідок чого вузол додається до Tailnet і стає доступним для інших пристроїв у межах цієї мережі (див. рис. 2.8).

```
root@phy-home-srv-pve-tiny-01:/etc/network# apt install -y tailscale
Installing:
  tailscale

Installing dependencies:
  tailscale-archive-keyring

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 126
  Download size: 36.6 MB
  Space needed: 71.9 MB / 91.5 GB available

Get:1 https://pkgs.tailscale.com/stable/debian trixie/main amd64 tailscale amd64 1.96.4 [36.6 MB]
Get:2 https://pkgs.tailscale.com/stable/debian trixie/main all tailscale-archive-keyring all 1.35.181 [3,082 B]
Fetched 36.6 MB in 2s (20.9 MB/s)
Selecting previously unselected package tailscale.
(Reading database ... 52049 files and directories currently installed.)
Preparing to unpack .../tailscale_1.96.4_amd64.deb ...
Unpacking tailscale (1.96.4) ...
Selecting previously unselected package tailscale-archive-keyring.
Preparing to unpack .../tailscale-archive-keyring_1.35.181_all.deb ...
Unpacking tailscale-archive-keyring (1.35.181) ...
Setting up tailscale-archive-keyring (1.35.181) ...
Setting up tailscale (1.96.4) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/tailscaled.service' → '/usr/lib/systemd/system/tailscaled.service'.
root@phy-home-srv-pve-tiny-01:/etc/network#
```

Рисунок 2.8 – Встановлення Tailscale

Після успішної авторизації гіпервізор отримує внутрішню IP-адресу у діапазоні 100.x.x.x, яка використовується для прямої взаємодії з іншими клієнтами Tailscale, однак для повноцінного доступу до внутрішньої підмережі цього недостатньо, оскільки необхідно забезпечити маршрутизацію трафіку від клієнтів Tailnet до ізольованої мережі, що знаходиться за гіпервізором (див. рис. 2.9).

```
root@phy-home-srv-pve-tiny-01:/etc/network# systemctl status tailscaled.service
* tailscaled.service - Tailscale node agent
     Loaded: loaded (/usr/lib/systemd/system/tailscaled.service; enabled; preset: enabled)
     Active: active (running) since Sat 2026-04-25 14:22:04 EEST; 32s ago
 Invocation: e4133330250541bc8a952a93f85b69ad
       Docs: https://tailscale.com/docs/
   Main PID: 20324 (tailscaled)
     Status: "Needs login: "
      Tasks: 12 (limit: 38132)
     Memory: 59.6M (peak: 60.1M)
        CPU: 936ms
     CGroup: /system.slice/tailscaled.service
             └─20324 /usr/sbin/tailscaled --state=/var/lib/tailscale/tailscaled.state --socket=/run/tailscale/tailscaled.sock --port=41641

Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: Switching ipn state NoState -> NeedsLogin (WantRunning=false, nm=false)
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: health(warnable=wantrunning-false): error: Tailscale is stopped.
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: blockEngineUpdates(true)
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: wgengine: Reconfig: configuring userspace WireGuard config (with 0/0 peers)
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: wgengine: Reconfig: configuring router
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: wgengine: Reconfig: user dialer
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: wgengine: Reconfig: configuring DNS
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: dns: Set: {DefaultResolvers:[] Routes:{} SearchDomains:[] Hosts:0}
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: dns: Resolvercfg: {Routes:{} Hosts:0 LocalDomains:[]}
Apr 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[20324]: dns: OSCfg: {}
root@phy-home-srv-pve-tiny-01:/etc/network# tailscale up

To authenticate, visit:

        https://login.tailscale.com/a/e27224c014719
```

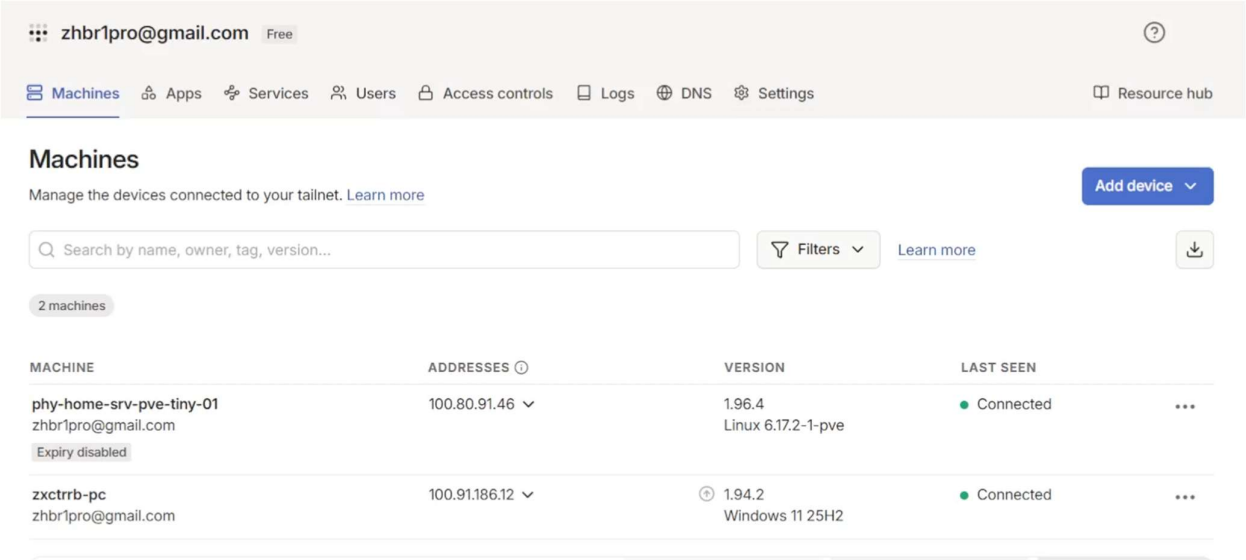


Рисунок 2.9 – Авторизація вузла в tailnet

Для реалізації такого сценарію виконується налаштування IP forwarding у системі шляхом зміни параметрів ядра, створення відповідного конфігураційного файлу та застосування змін, після чого Tailscale запускається із параметром анонсування маршрутів, що дозволяє оголосити підмережу 192.168.9.0/24 доступною для інших вузлів Tailnet, у результаті чого клієнти, підключені до mesh-мережі, отримують можливість звертатися до ресурсів ізольованого середовища так, ніби вони знаходяться в одній локальній мережі.

Після активації відповідного маршруту у веб-інтерфейсі Tailscale формується повноцінна схема доступу, у межах якої зовнішні клієнти можуть взаємодіяти як із

самим гіпервізором, так і з віртуальними машинами та контейнеризованими сервісами, включаючи Portainer, причому весь процес відбувається без необхідності відкриття портів на маршрутизаторі або планшеті, що виступає як проміжний мережевий вузол.

Контейнеризація прикладних сервісів. Завершальним етапом побудови серверного вузла виступає розгортання прикладних сервісів у межах попередньо сформованого віртуального середовища, де як базовий інструмент контейнеризації використовується Docker, що встановлюється безпосередньо всередині віртуальної машини на базі Ubuntu Server 24.04 LTS, створеної із застосуванням Cloud-init, завдяки чому забезпечується уніфіковане, ізольоване та відтворюване середовище виконання сервісів без впливу на інші компоненти системи.

Після інсталяції Docker Engine формується конфігураційний опис сервісів за допомогою Docker Compose, який дозволяє декларативно задати параметри розгортання контейнерів, включаючи:

— мережеві налаштування;
— змінні середовища;
— політики перезапуску;
— прив’язку портів.

У результаті чого спрощується процес управління сервісами та забезпечується можливість швидкого повторного розгортання інфраструктури у випадку змін або відмов.

У якості тестового прикладного сервісу обрано Portainer, що реалізує веб-інтерфейс для керування контейнерним середовищем і дозволяє здійснювати моніторинг, розгортання та адміністрування контейнерів через зручний графічний інтерфейс, при цьому сам сервіс розгортається у контейнері та публікується на порту 9000 всередині віртуальної машини, яка має статичну IP-адресу 192.168.9.138 у межах ізольованої підмережі (див. рис. 2.10).

```
docker@vir-home-srv-ubnt-docker-01:~/portainer$ docker compose up -d
[+] up 13/13
 ✔ Image portainer/portainer-ce:lts Pulled                                                                                    6.1s
 ✔ Network docker                   Created                                                                                   0.0s
 ✔ Container portainer              Started                                                                                   0.4s
docker@vir-home-srv-ubnt-docker-01:~/portainer$ docker ps
CONTAINER ID   IMAGE                          COMMAND        CREATED         STATUS         PORTS
                                NAMES
bf94de6da16d   portainer/portainer-ce:lts   "/portainer"   5 seconds ago   Up 4 seconds   0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp, 0.0.0.0:9443->9443/tcp,
 [::]:9443->9443/tcp, 9000/tcp   portainer
docker@vir-home-srv-ubnt-docker-01:~/portainer$
```

Рисунок 2.10 – Розгортання Docker-контейнера

Доступ до Portainer обмежується виключно внутрішнім мережевим середовищем, оскільки жодні порти не відкриваються на рівні зовнішнього маршрутизатора або проміжного вузла у вигляді планшета, що функціонує як USB-модем, унаслідок чого сервіс залишається недоступним із зовнішньої мережі через класичні механізми підключення, а взаємодія з ним можлива лише в межах локальної підмережі або через інтегровану mesh-мережу (див. рис. 2.11).

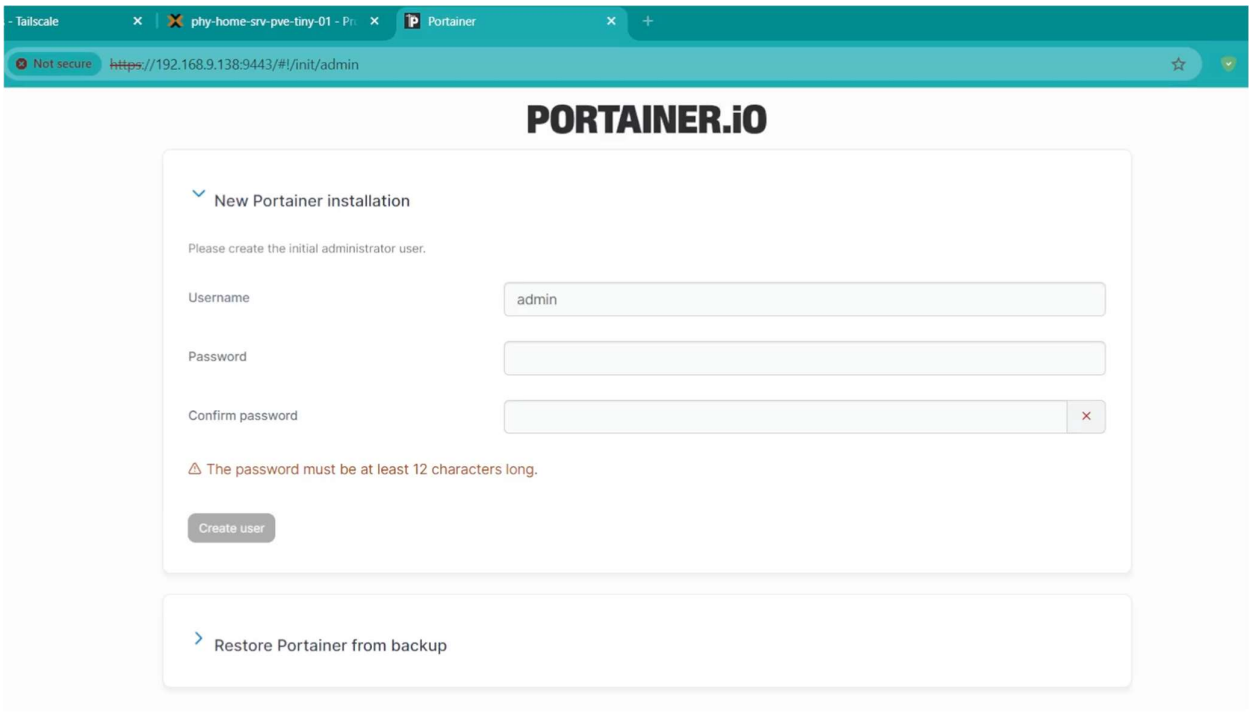


Рисунок 2.11 – Інтерфейс Portainer після старту контейнера

Перевірка доступності сервісу на початковому етапі підтверджує, що у відсутності інтеграції з Tailscale звернення до веб-інтерфейсу за адресою 192.168.9.138:9000 неможливе поза межами ізольованої мережі, що демонструє ефективність створеної ізоляції та підтверджує відсутність небажаного доступу

ззовні, водночас після підключення клієнтського пристрою до Tailnet з’являється можливість встановлення захищеного з’єднання з серверним вузлом і подальшого доступу до сервісу через внутрішню IP-адресу або з використанням механізму доменних імен.

Застосування MagicDNS дозволяє звертатися до сервісу не лише за IP-адресою, але й за символьним ім’ям вузла, що спрощує адміністрування та підвищує зручність використання системи, оскільки усувається необхідність запам’ятовування адрес, які можуть змінюватися у межах Tailnet, а взаємодія з ресурсами набуває більш прозорого та передбачуваного характеру.

Додатково виконується перевірка доступу до серверного вузла через SSH, що реалізується із використанням ключової автентифікації, попередньо налаштованої через Cloud-init, у поєднанні з механізмами безпеки Tailscale, у результаті чого забезпечується можливість підключення без використання паролів, що підвищує рівень захисту та відповідає сучасним вимогам до адміністрування інфраструктури.

У підсумку формується завершена модель функціонування прикладного рівня, у межах якої контейнеризований сервіс розгортається у віртуальному середовищі, ізольованому від зовнішньої мережі, та стає доступним виключно через захищений канал mesh-мережі, що дозволяє демонструвати можливість організації віддаленого доступу до внутрішніх ресурсів без відкриття портів, без складної конфігурації маршрутизації та без залежності від фізичних обмежень мережевої інфраструктури.

2.3 Алгоритми функціонування Mesh-мережі в умовах ізоляції

Функціонування mesh-мережі в умовах ізольованого середовища визначається необхідністю забезпечення стабільної, передбачуваної та безпечної взаємодії між вузлами за відсутності прямої мережевої доступності, що характерно для багаторівневих NAT-конфігурацій, сформованих унаслідок використання

мобільного шлюзу у вигляді USB-модему, де планшет створює окрему підмережу 192.168.9.0/24, ізольовану від основної локальної інфраструктури, у результаті чого всі внутрішні вузли, включаючи гіпервізор та віртуальні машини, стають недоступними для класичних механізмів мережевої взаємодії, що автоматично виключає можливість використання традиційних підходів, таких як статична маршрутизація, проброс портів або пряме встановлення з'єднань через TCP/IP.

У такій конфігурації особливого значення набувають алгоритмічні механізми побудови overlay-мережі, де логічна адресація та маршрутизація відокремлюються від фізичної топології, а доступ до ресурсів визначається не їхнім розташуванням у мережі, а політиками автентифікації та авторизації, реалізованими через централізовану систему керування, що дозволяє сформувати стійку до змін середовища архітектуру доступу, здатну функціонувати навіть у випадках повної ізоляції на рівні L3.

Окремий акцент у даній реалізації зміщується на три взаємопов'язані алгоритмічні складові, серед яких:

1. механізм обходу NAT;
2. конфігурація вузла як маршрутизатора підмережі;
3. система іменування і доступу.

Все це разом формують завершену модель функціонування mesh-мережі у складних умовах.

Реалізація NAT Traversal. Проблематика трансляції мережевих адрес у сучасних мережах набуває особливої складності у випадках багаторівневого NAT, де кожен рівень трансляції обмежує можливість ініціації вхідних з'єднань, а в умовах подвійного NAT, сформованого поєднанням домашнього маршрутизатора та USB-модему планшета, формується повністю ізольоване середовище, в якому вузол із адресою 192.168.9.136 може встановлювати лише вихідні з'єднання, тоді як будь-які спроби доступу до нього ззовні блокуються на рівні мережевої інфраструктури.

Така ситуація унеможливлює використання класичних VPN-рішень без додаткових налаштувань, оскільки відсутня можливість відкриття портів або

використання статичної IP-адресації, а також ускладнюється встановлення тунелів у режимі server-client, де сервер має бути доступним із зовнішньої мережі, тому реалізація NAT traversal базується на принципово іншому підході, що передбачає ініціацію з'єднання з обох сторін та використання проміжного координаційного сервера [25].

Алгоритм встановлення з'єднання у такому випадку має послідовну структуру, у якій гіпервізор, що знаходиться за подвійним NAT, ініціює вихідне з'єднання до координаційного сервера, одночасно клієнтський пристрій також встановлює з'єднання з тим самим сервером, після чого відбувається обмін інформацією про публічні IP-адреси, порти та типи NAT, що дозволяє визначити можливість встановлення прямого P2P-з'єднання.

Наступним етапом стає спроба реалізації UDP hole punching, у процесі якої обидва вузли одночасно відправляють пакети один одному через визначені порти, що дозволяє обійти обмеження NAT і сформувати прямий канал передачі даних, а у випадку, коли така спроба не є успішною через суворі правила трансляції або симетричний NAT, використовується резервний механізм у вигляді relay-серверів, які виступають як проміжна ланка для передачі зашифрованого трафіку [25].

Після встановлення каналу зв'язку відбувається формування захищеного тунелю на основі сучасного криптографічного протоколу, що забезпечує шифрування всього переданого трафіку, автентифікацію вузлів та цілісність даних, у результаті чого формується стабільне з'єднання, незалежне від фізичної мережевої топології (див. рис. 2.12).

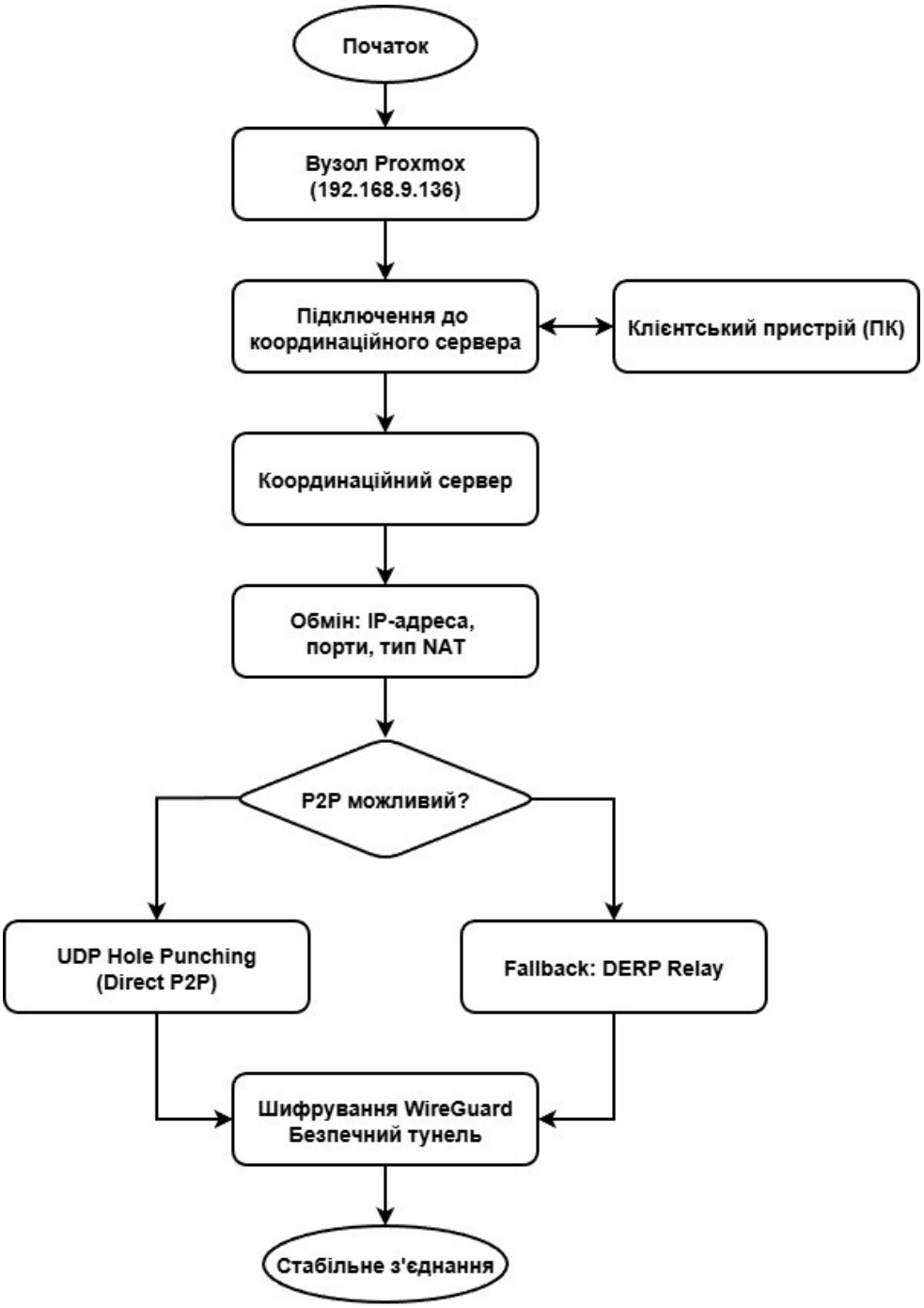


Рисунок 2.12 – Алгоритм NAT Traversal

Такий підхід дозволяє забезпечити доступ до вузла, що знаходиться у повністю ізольованій підмережі, навіть у випадку відсутності будь-яких відкритих портів або можливості конфігурації мережевого обладнання, що є критично важливим для сценаріїв із використанням мобільного інтернету або обмежених інфраструктур (див. табл. 2.3).

Таблиця 2.3 – Порівняння механізмів NAT Traversal

Метод	Умови застосування	Затримка	Залежність від мережі	Ефективність при Double NAT
STUN	Визначення зовнішньої адреси	Низька	Середня	Обмежена
P2P (UDP hole punching)	Підтримка NAT	Дуже низька	Висока	Висока (якщо працює)
DERP (relay)	Будь-які умови	Вища	Низька	Гарантована
Комбінований (Tailscale)	Автоматичний вибір	Оптимальна	Мінімальна	Максимальна

Конфігурація Subnet Router. Для забезпечення доступу не лише до окремого вузла, а й до всієї ізольованої підмережі використовується режим subnet router, у межах якого компактний серверний вузол на базі неттопа Lenovo ThinkCentre M920q виступає як шлюз між віртуальною мережею tailnet з адресацією діапазону 100.x.x.x та локальною ізольованою підмережею 192.168.9.0/24, сформованою через використання планшета Xiaomi Mi Pad 5 у режимі USB-модему, що фактично створює розрив між фізичною топологією та логічною доступністю ресурсів і потребує впровадження механізму маршрутизації на рівні гіпервізора.

Початковий етап конфігурації передбачає активацію пересилання пакетів між мережевими інтерфейсами на рівні операційної системи, що реалізується шляхом створення конфігураційного файлу "/etc/sysctl.d/99-tailscale.conf" із параметрами "net.ipv4.ip_forward=1" та "net.ipv6.conf.all.forwarding=1", після чого застосування змін виконується через відповідну команду системного рівня, що дозволяє ядру обробляти транзитний трафік і фактично перетворює вузол на маршрутизатор (див. рис. 2.13).

```
root@phy-home-srv-pve-tiny-01:/# echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/99-tailscale.conf
root@phy-home-srv-pve-tiny-01:/# echo "net.ipv6.conf.all.forwarding=1" >> /etc/sysctl.d/99-tailscale.conf
root@phy-home-srv-pve-tiny-01:/# cat /etc/sysctl.d/99-tailscale.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
root@phy-home-srv-pve-tiny-01:/# _
```

Рисунок 2.13 – Конфігурація IP forwarding

Наступний етап передбачає анонсування маршрутів до ізольованої підмережі, що виконується під час запуску сервісу із зазначенням параметра advertise-routes для діапазону 192.168.9.0/24, у результаті чого інформація про доступність цієї мережі поширюється серед усіх вузлів tailnet, а після активації відповідного маршруту у централізованій системі керування формується логічний канал доступу до внутрішньої інфраструктури через даний вузол.

Подальша робота механізму описується послідовністю обробки мережевого запиту, у межах якої клієнтський пристрій із адресою діапазону 100.x.x.x виконує перевірку таблиці маршрутів і визначає наявність шляху до підмережі 192.168.9.0/24, після чого пакет передається через захищений тунель до гіпервізора, який у ролі subnet router виконує перевірку активності IP forwarding і, у разі позитивного результату, здійснює маршрутизацію трафіку у локальну мережу, де відбувається доставка до цільового вузла [26].

У межах такої архітектури стають доступними як сам гіпервізор із адресою 192.168.9.136, так і віртуальне середовище, зокрема віртуальна машина з розгорнутим контейнеризованим сервісом керування контейнерами за адресою 192.168.9.138, причому доступ до них забезпечується без прямого підключення до ізольованої мережі та без необхідності зміни конфігурації фізичної мережевої інфраструктури, що формує логічну модель взаємодії, у якій клієнт передає трафік через overlay-мережу до subnet router, а той виступає посередником між двома сегментами (див. рис. 2.14) [22; 26].

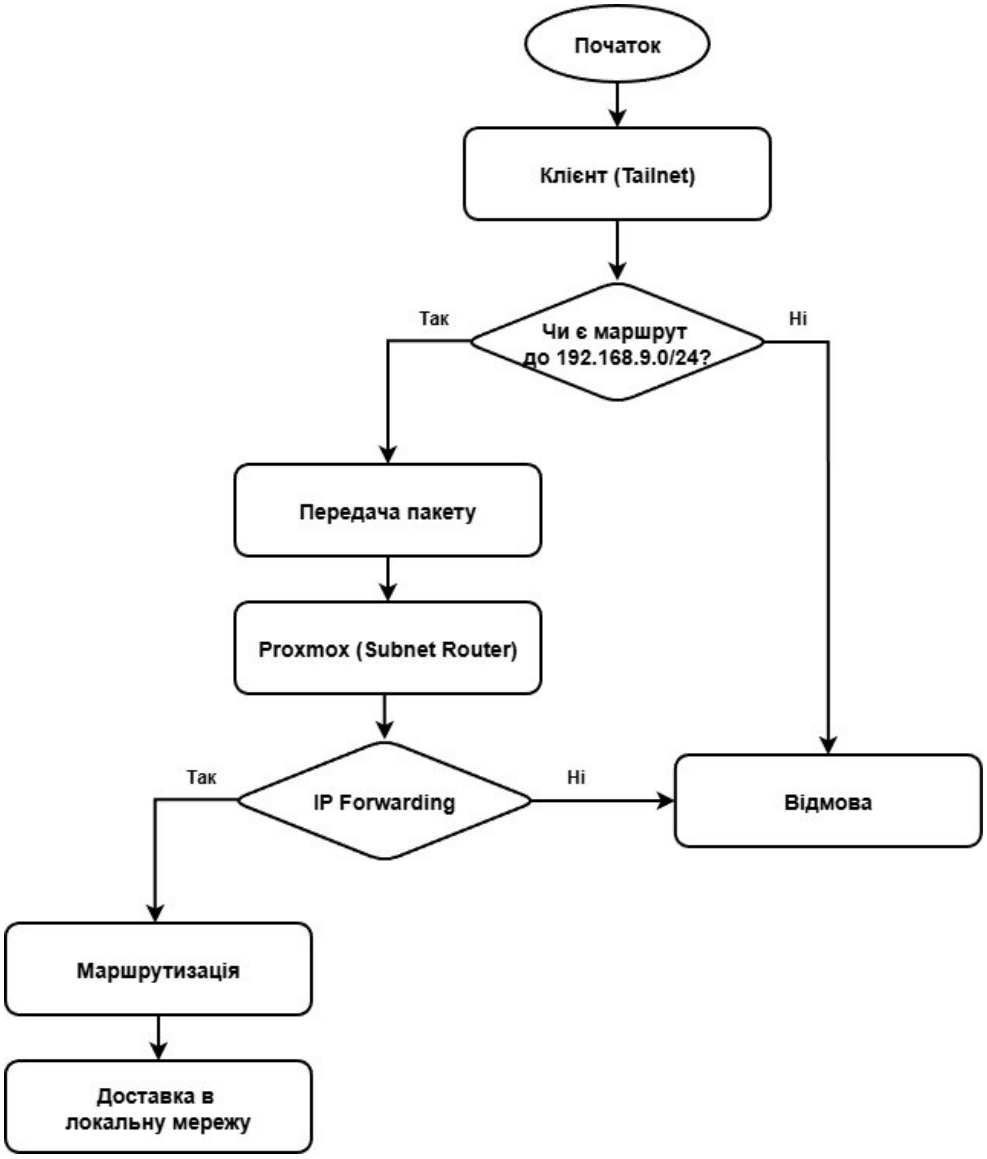


Рисунок 2.14 – Алгоритм роботи Subnet Router

Узагальнення такої моделі дозволяє описати її як послідовність переходів між мережевими рівнями, де клієнтський вузол взаємодіє з tailnet, далі трафік передається до гіпервізора як точки входу у локальну підмережу, після чого здійснюється доставка до внутрішніх сервісів, включаючи системні компоненти та контейнеризовані застосунки, що забезпечує повноцінний доступ до ізольованої інфраструктури без фізичної присутності у ній.

Робота MagicDNS та Tailscale SSH. Забезпечення зручного доступу до вузлів у межах mesh-мережі реалізується через використання механізму іменування, що дозволяє звертатися до ресурсів не за IP-адресами, а за унікальними

символьними ідентифікаторами, які автоматично призначаються під час реєстрації вузла у мережі, у результаті чого формується узгоджений простір імен, де кожен пристрій отримує власне DNS-ім’я, наприклад "phy-home-srv-pve-tiny-01", яке відповідає конкретній IP-адресі у діапазоні 100.x.x.x [27].

Процес встановлення з’єднання у такому випадку включає введення імені вузла замість IP-адреси, після чого виконується DNS lookup у межах tailnet, у результаті якого визначається відповідна адреса, далі ініціюється встановлення захищеного з’єднання через тунель, що дозволяє абстрагуватися від фізичної мережевої топології та спростити процес адміністрування [27].

Паралельно з цим реалізується механізм віддаленого доступу до командного рядка через SSH без використання паролів і без необхідності ручного налаштування ключів або відкриття стандартного порту 22, оскільки автентифікація делегується зовнішньому провайдеру ідентифікації, а доступ визначається політиками контролю доступу, що зберігаються у централізованій системі [20-21].

Алгоритм роботи такого підходу включає ініціацію запиту на підключення, перевірку автентичності користувача через систему єдиного входу, аналіз політик доступу на основі ACL, після чого у разі відповідності встановлюється захищене з’єднання, що надає повний доступ до консолі вузла без введення додаткових облікових даних, а у випадку невідповідності доступ блокується на рівні політик.

Практична перевірка такого механізму демонструє можливість миттєвого підключення до серверного вузла за допомогою команди, що використовує символьне ім’я замість IP-адреси, у результаті чого забезпечується швидкий і безпечний доступ до системи, незалежно від її фізичного розташування та мережевих обмежень (див. рис. 2.15).

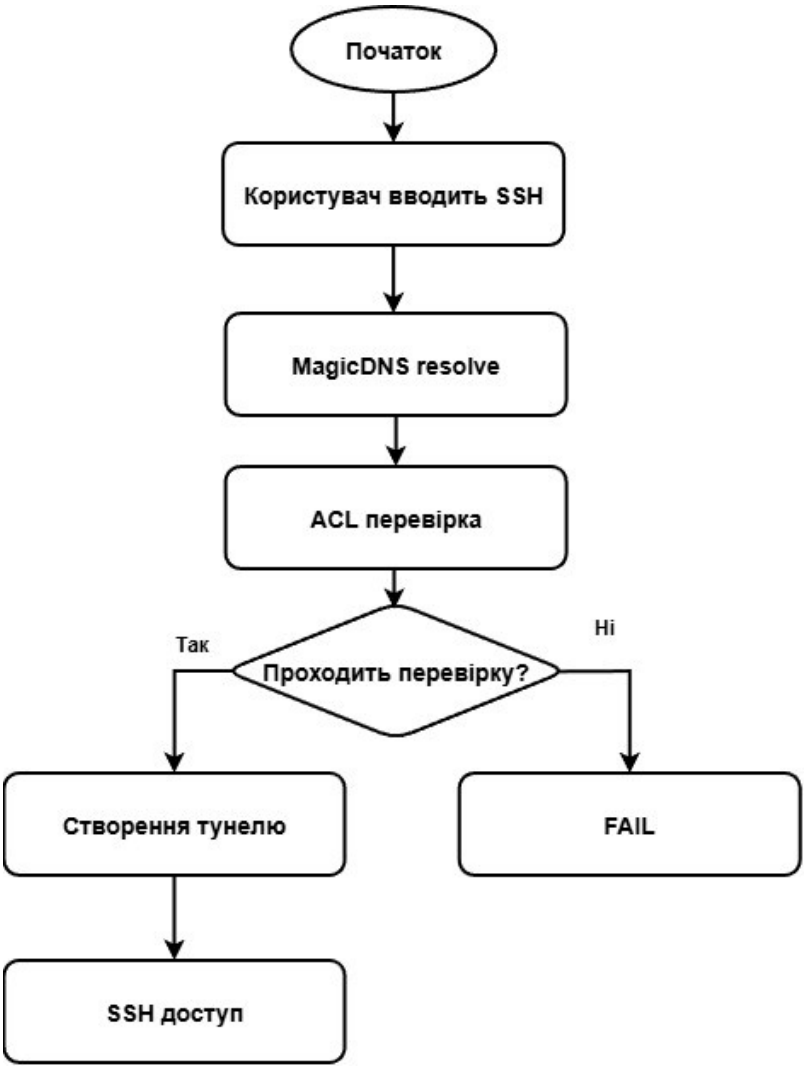


Рисунок 2.15 – Алгоритм підключення через Tailscale SSH

Узагальнення функціонування механізмів іменування та віддаленого доступу дозволяє сформувати модель, у якій IP-адресація стає внутрішньою технічною деталлю, а взаємодія з інфраструктурою відбувається через логічні ідентифікатори та централізовані політики, що забезпечує стабільність з’єднання, відсутність необхідності відкриття портів, підвищений рівень безпеки та мінімізацію ручного налаштування, що особливо ефективно проявляється в умовах ізольованих мереж із подвійним NAT, де традиційні підходи до організації доступу виявляються неефективними або повністю непридатними.

2.4 Практична реалізація та верифікація доступу

Подальший етап дослідження зосереджується на перевірці працездатності сформованого програмно-апаратного комплексу в умовах ізольованого мережевого середовища, де основною метою виступає експериментальне підтвердження можливості доступу до серверних ресурсів без використання механізмів пробросу портів або статичних публічних IP-адрес, що особливо актуально для сценаріїв із багаторівневим NAT та мобільними каналами зв'язку, а сама верифікація реалізується через послідовний аналіз доступності вузлів до та після активації віртуальної mesh-мережі, перевірку доступу до ізольованого веб-сервісу та тестування безпечного безпарольного адміністративного доступу [31-32].

Демонстрація «пробиття» Double NAT. Початкові умови експерименту формуються у конфігурації, де серверний вузол на базі компактного неттопа підключений до мережі через USB-модем у вигляді планшета, який створює окрему підмережу 192.168.9.0/24, при цьому сам сервер отримує адресу 192.168.9.136, а віртуальна машина з розгорнутим середовищем Ubuntu Server 24.04 LTS та контейнеризованим сервісом керування має адресу 192.168.9.138, тоді як основна домашня мережа функціонує у діапазоні 192.168.0.0/24 із шлюзом 192.168.0.1, що формує чітко розділені сегменти без прямої маршрутизації між ними, а використання планшета як проміжного вузла створює другий рівень трансляції адрес, який повністю блокує вхідні з'єднання.

У такій конфігурації виконується первинна перевірка доступності з клієнтського комп'ютера, підключеного до основної мережі, де здійснюється спроба звернення до серверного вузла та віртуальної машини за допомогою ICMP-запитів, однак результати демонструють повну недоступність обох адрес, що проявляється у втраті пакетів та відсутності відповіді, аналогічно HTTP-запити до веб-інтерфейсу також не встановлюють з'єднання, а причиною виступає відсутність маршрутів між сегментами та неможливість використання порт-

форвардингу в умовах подвійного NAT, сформованого як на рівні маршрутизатора, так і на рівні мобільного пристрою (див. рис. 2.16).

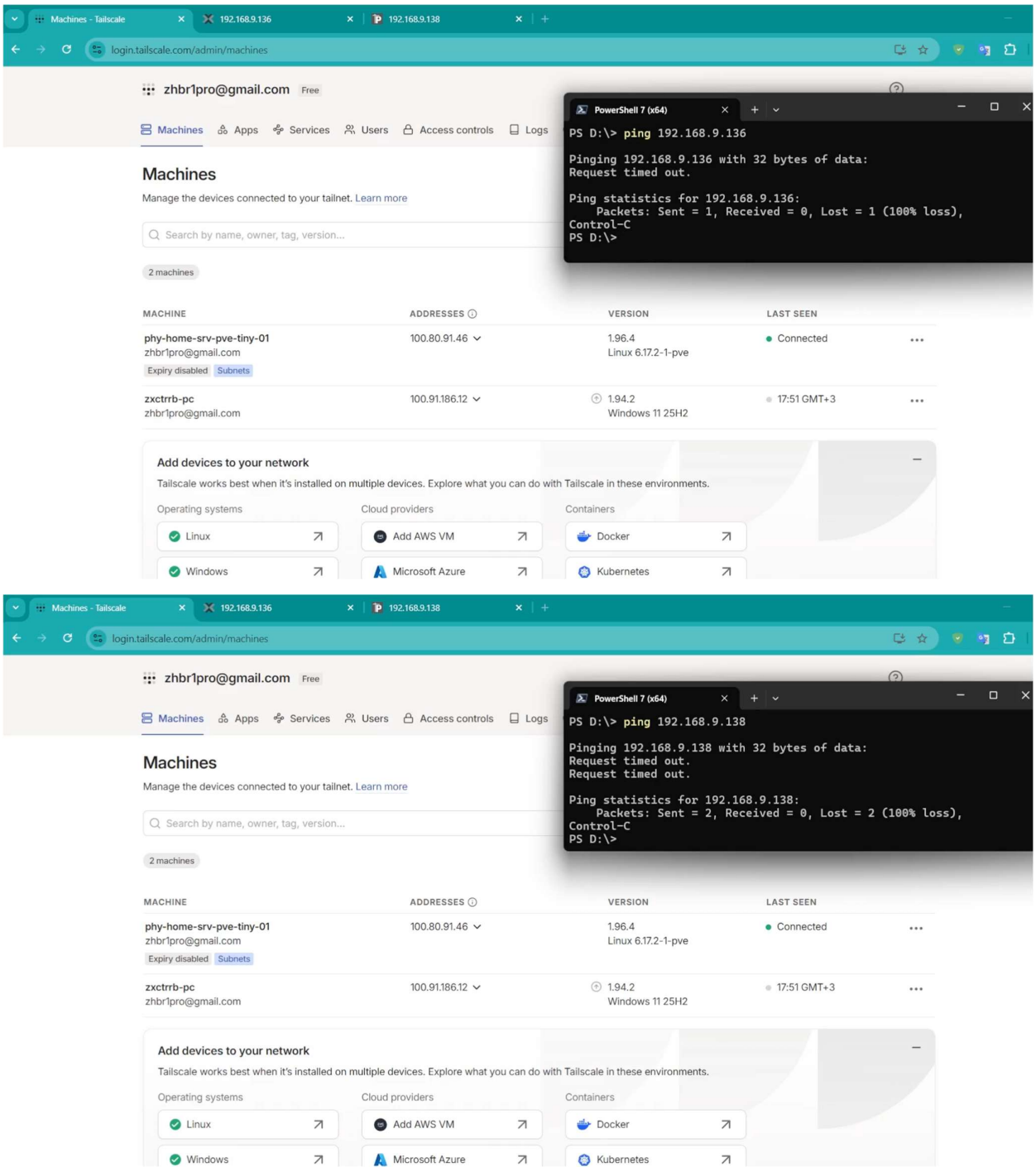


Рисунок 2.16 – Відсутність доступу без Tailscale (Ping test)

Подальший етап включає активацію програмно визначеної мережі, після встановлення відповідного клієнта на серверному вузлі та клієнтському пристрої,

виконання авторизації через зовнішнього провайдера ідентифікації та налаштування маршрутизації підмережі через механізм subnet router, у результаті чого формується логічна overlay-мережа з адресним простором 100.x.x.x, яка функціонує незалежно від фізичної топології та обмежень NAT.

Повторна перевірка доступності демонструє кардинально інший результат, де ICMP-запити до адрес 192.168.9.136 та 192.168.9.138 успішно проходять, а веб-інтерфейс сервісу стає доступним, що пояснюється зміною маршруту передачі трафіку, оскільки запити більше не спрямовуються через фізичну мережеву інфраструктуру з обмеженнями NAT, а передаються через захищений тунель overlay-мережі до вузла, який виступає маршрутизатором підмережі та забезпечує доставку пакетів у внутрішній сегмент, таким чином експериментально підтверджується можливість обходу подвійного NAT без внесення змін у конфігурацію маршрутизатора або відкриття портів (див. рис. 2.17).

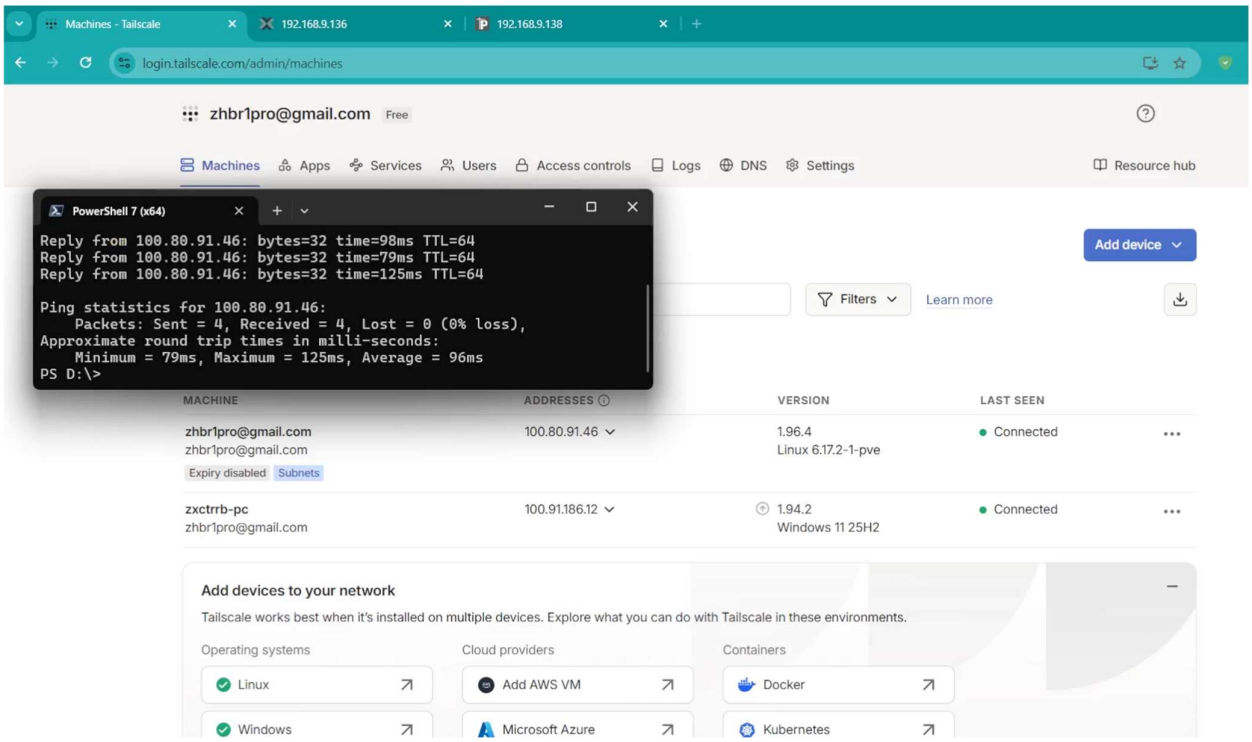


Рисунок 2.17 – Доступ після активації Tailscale (Ping test)

Веб-інтерфейс Portainer через Tailscale. Передумовою для реалізації доступу виступає розгортання у віртуальній машині контейнеризованого сервісу керування,

який функціонує на базі середовища Ubuntu Server 24.04 LTS та має внутрішню IP-адресу 192.168.9.138, при цьому веб-інтерфейс доступний через порт 9000 виключно в межах ізольованої підмережі, сформованої через USB-модем, а сам порт не відкривається ні на рівні планшета, ні на рівні основного маршрутизатора, що створює повністю закритий контур доступу без будь-якого експонування у зовнішню мережу [24].

У таких умовах спроба звернення до сервісу через публічну адресу або навіть через локальну мережу маршрутизатора не дає результату, оскільки подвійний NAT повністю блокує вхідні з’єднання, а відсутність правил перенаправлення портів унеможливлює встановлення TCP-сесії, що підтверджує ізольований характер середовища та відсутність класичних шляхів доступу до внутрішніх ресурсів.

Після підключення клієнтського пристрою до віртуальної mesh-мережі формується альтернативний канал взаємодії, в межах якого вузол отримує IP-адресу з діапазону 100.x.x.x, що дозволяє звертатися до сервісу безпосередньо через цю адресу із зазначенням відповідного порту, у результаті чого веб-інтерфейс відкривається стабільно та без затримок, а передача даних здійснюється через зашифрований тунель, незалежний від фізичної топології мережі.

Додатковий рівень зручності забезпечується використанням механізму іменування вузлів, де замість числової адресації застосовується доменне ім’я, яке автоматично резолвиться у відповідну IP-адресу віртуальної мережі, що дозволяє звертатися до сервісу у форматі зрозумілого ідентифікатора із вказанням порту доступу, а сам процес резолюції відбувається всередині мережі без використання зовнішніх DNS-серверів (див. рис. 2.18).

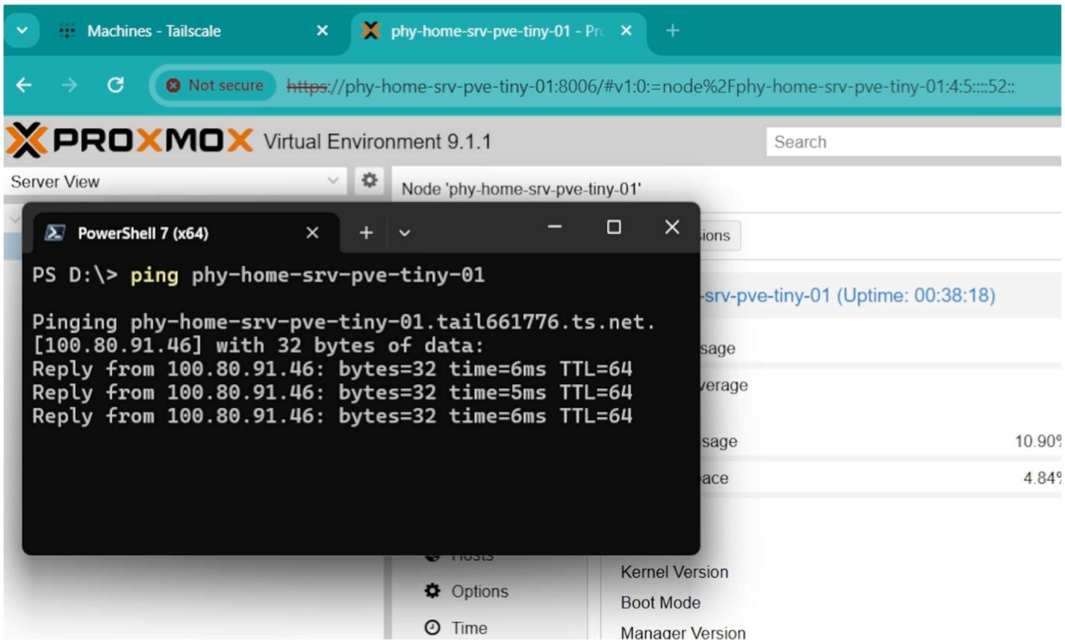


Рисунок 2.18 – Доступ через MagicDNS (браузер)

Архітектура доступу у даному випадку формується як багаторівнева послідовність взаємодії, де браузер клієнтського пристрою ініціює запит до віртуальної мережі, після чого трафік передається через захищений тунель до серверного вузла, далі маршрутизується до гіпервізора і передається у віртуальне середовище, де контейнерний рушій забезпечує обробку запиту та повернення відповіді від веб-сервісу, що створює логічно цілісний ланцюг без необхідності взаємодії із зовнішніми мережевими компонентами.

Інженерна перевага такого підходу проявляється у відсутності відкритих портів, відсутності необхідності налаштування NAT-правил та обмеженні доступу виключно до авторизованих учасників віртуальної мережі, що дозволяє реалізувати модель доступу до внутрішнього сервісу без його публічного розміщення, зберігаючи при цьому повний функціонал керування контейнеризованим середовищем та відповідність принципам ізольованої інфраструктури.

Безпарольне адміністрування. Подальша реалізація передбачає організацію адміністративного доступу до серверного вузла без використання традиційних механізмів парольної автентифікації, де початкові параметри користувача та відкритого ключа задаються ще на етапі ініціалізації віртуальної машини через

механізм автоматичної конфігурації, що дозволяє одразу забезпечити доступ до системи на основі криптографічної ідентифікації.

У поєднанні з можливостями віртуальної мережі формується сценарій, у якому підключення до серверного вузла здійснюється через захищений тунель, а запит на встановлення SSH-сесії проходить послідовну перевірку, що включає ідентифікацію користувача, аналіз політик доступу та підтвердження прав на підключення, після чого у разі позитивного результату автоматично встановлюється зашифроване з'єднання без необхідності введення пароля.

Такий підхід дозволяє уникнути відкриття стандартного порту 22 у зовнішній мережі та виключає необхідність ручного розповсюдження або керування ключами доступу, оскільки всі механізми автентифікації централізуються та інтегруються з системою ідентифікації, що значно знижує ризики несанкціонованого доступу, включаючи атаки типу brute force, та спрощує процес адміністрування інфраструктури.

Практична перевірка демонструє миттєве встановлення SSH-з'єднання із сервером без запиту пароля, що підтверджує коректність налаштування як локальних параметрів системи, так і політик доступу в межах віртуальної мережі, а сам процес підключення відбувається прозоро для користувача, забезпечуючи одночасно високий рівень безпеки та зручності використання (див. рис. 2.19).

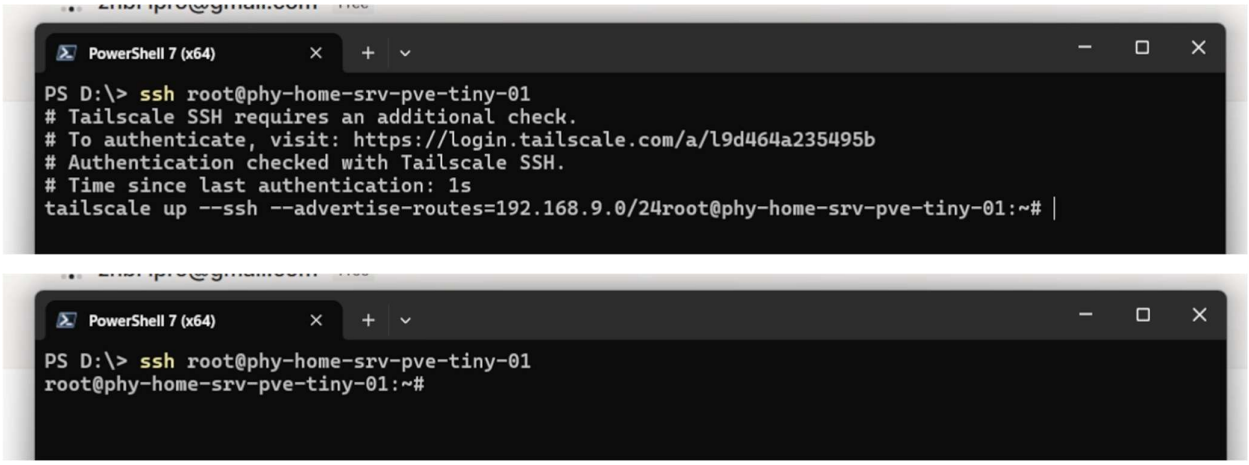


Рисунок 2.19 – SSH-підключення без пароля через термінал

3. ОПТИМІЗАЦІЯ, БЕЗПЕКА ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

3.1 Методи оптимізації та вирішення апаратних конфліктів (Troubleshooting)

У процесі реалізації програмно-апаратного комплексу на базі компактного серверного вузла було виявлено низку критичних обмежень, пов’язаних як із особливостями апаратної платформи, так і з нюансами її інтеграції у середовище віртуалізації, що особливо проявляється при використанні неттопів корпоративного класу у ролі серверних систем, де апаратна сумісність не завжди відповідає вимогам стабільної роботи під керуванням гіпервізора, а проведений аналіз дозволив не лише зафіксувати проблемні аспекти, але й сформувати практичні підходи до їх усунення, які у подальшому стали основою для побудови стабільної та масштабованої інфраструктури з використанням програмно визначеної мережі.

Проблематика вбудованих мережевих карт Intel. У ході розгортання гіпервізора у середовищі віртуалізації на компактному обчислювальному вузлі було виявлено некоректну роботу вбудованого мережевого адаптера на базі контролерів Intel, де незважаючи на фізичну присутність пристрою та його коректне визначення на рівні BIOS, у середовищі операційної системи виникають проблеми з ініціалізацією драйверів, що проявляється у відсутності повноцінного підняття мережевого інтерфейсу після завантаження, при цьому типовий сценарій полягає у тому, що BIOS коректно ідентифікує мережевий контролер, ядро операційної системи частково його ініціалізує, однак на рівні гіпервізора інтерфейс або не з’являється взагалі, або працює нестабільно [28].

Основними причинами такої поведінки виступає часткова або некоректна підтримка конкретних ревізій мережевих контролерів драйверами ядра операційної системи, що особливо характерно для серій, орієнтованих на корпоративні

десктопні рішення, де апаратна реалізація може відрізнятися від класичних серверних адаптерів, у результаті чого виникає несумісність, яка ускладнює використання стандартних мережевих можливостей системи (див. рис. 3.1).

```
phy-home-srv-pve-tiny-01 login: root
Password:
Linux phy-home-srv-pve-tiny-01 6.17.2-1-pve #1 SMP PREEMPT_DYNAMIC PMX 6.17.2-1 (2025-10-21T11:55Z) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@phy-home-srv-pve-tiny-01:~# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: vmbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 6e:6b:a5:2f:ae:8d brd ff:ff:ff:ff:ff:ff
root@phy-home-srv-pve-tiny-01:~# ip -c a^C
root@phy-home-srv-pve-tiny-01:~# modprobe e1000e
root@phy-home-srv-pve-tiny-01:~# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: vmbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 6e:6b:a5:2f:ae:8d brd ff:ff:ff:ff:ff:ff
root@phy-home-srv-pve-tiny-01:~# dmesg | grep -i e1000e
[    1.869236] e1000e: Intel(R) PRO/1000 Network Driver
[    1.869240] e1000e: Copyright(c) 1999 - 2015 Intel Corporation.
[    1.869536] e1000e 0000:00:1f.6: Interrupt Throttling Rate (ints/sec) set to dynamic conservative mode
[    1.942392] e1000e 0000:00:1f.6: The NVM Checksum Is Not Valid
[    1.975903] e1000e 0000:00:1f.6: probe with driver e1000e failed with error -5
root@phy-home-srv-pve-tiny-01:~# _
```

Рисунок 3.1 – Проблеми ініціалізації мережевих адаптерів Intel

Практичний вплив цієї проблеми на початковому етапі розгортання проявляється у:

1. повній відсутності доступу до мережі;
2. неможливості підключення до веб-інтерфейсу гіпервізора;
3. необхідності використання альтернативних каналів доступу.

Все це фактично блокує стандартний процес налаштування інфраструктури та створює залежність від обхідних рішень.

Одним із перших підходів до вирішення стає використання зовнішніх мережевих інтерфейсів, зокрема USB Ethernet-адаптерів або мобільних пристроїв у режимі USB-модему, де підключення до мережі здійснюється через зовнішній вузол, який забезпечує передачу трафіку до серверного пристрою, при цьому використання планшета як USB-модему дозволяє не лише отримати доступ до мережі, але й сформувати ізольовану підмережу, що додатково використовується

як тестове середовище для перевірки роботи мережевих технологій у складних умовах (див. рис. 3.2).

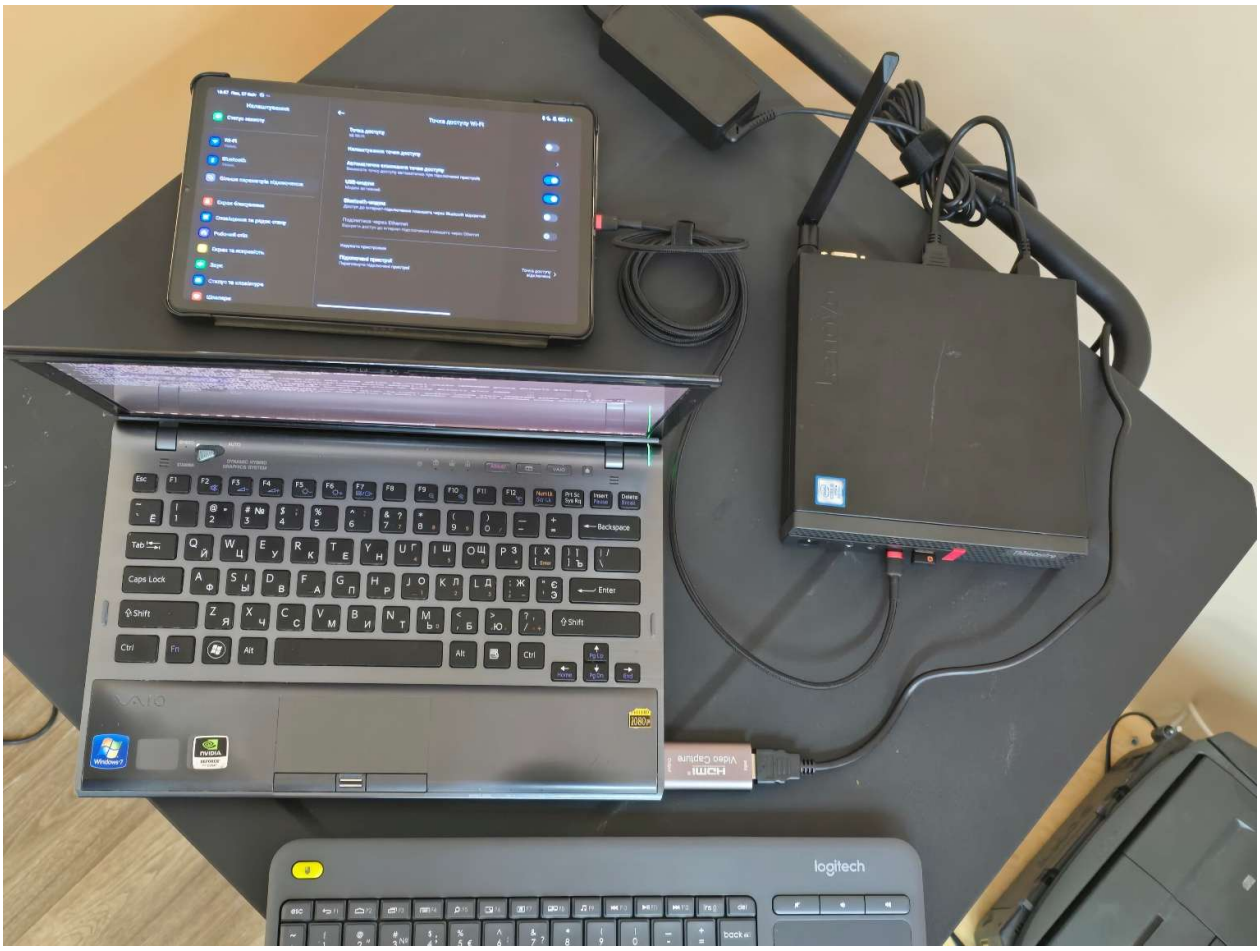


Рисунок 3.2 – Використання USB Ethernet адаптера або USB-модема

Перевагою такого підходу виступає повна незалежність від вбудованого мережевого адаптера та сумісність із операційною системою, однак водночас присутні обмеження у вигляді потенційної нестабільності з’єднання, обмеженої пропускної здатності та тимчасового характеру рішення, що не відповідає вимогам довготривалої експлуатації серверної інфраструктури.

Більш оптимальним варіантом виступає використання дискретних мережевих адаптерів із підключенням через розширювальні інтерфейси, зокрема із застосуванням PCI riser, що дозволяє інтегрувати повноцінну мережеву карту серверного класу у компактний корпус, забезпечуючи стабільну роботу, повну

сумісність із драйверами операційної системи та прогнозовану продуктивність, у результаті чого формується надійна мережева основа для подальшого розгортання віртуалізованого середовища (див. рис. 3.3).

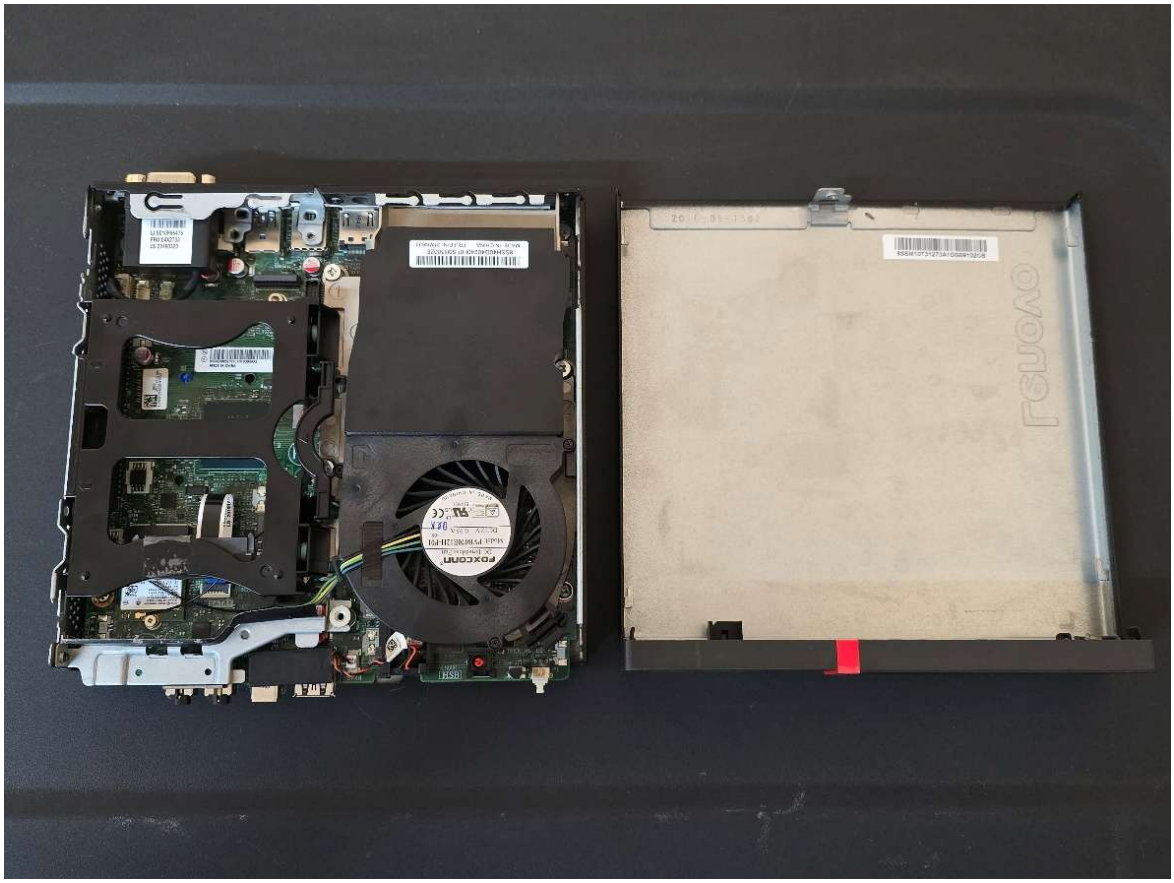


Рисунок 3.3 – Можливість використання повноцінного PCIe NIC

Обмеження технології Intel vPro/AMT. Додатковим аспектом стала перевірка можливостей технології Intel vPro/AMT, яка позиціонується як інструмент віддаленого керування з доступом до BIOS, IP-KVM та повного контролю над системою, однак у реальних умовах експлуатації було встановлено, що на платформі Lenovo ThinkCentre M920q функціональність vPro присутня лише частково та обмежена базовими можливостями.

Фактичні обмеження проявляються у відсутності повноцінного KVM-доступу, наявності лише базових функцій керування та неможливості реалізації повного віддаленого контролю системи, що обумовлено версією AMT, відсутністю

необхідних ліцензій або обмеженнями самої апаратної платформи, у результаті чого навіть при використанні спеціалізованих інструментів, таких як MeshCommander, доступ залишається обмеженим і не відповідає очікуваному рівню enterprise-рішень (див. рис. 3.4) [29].

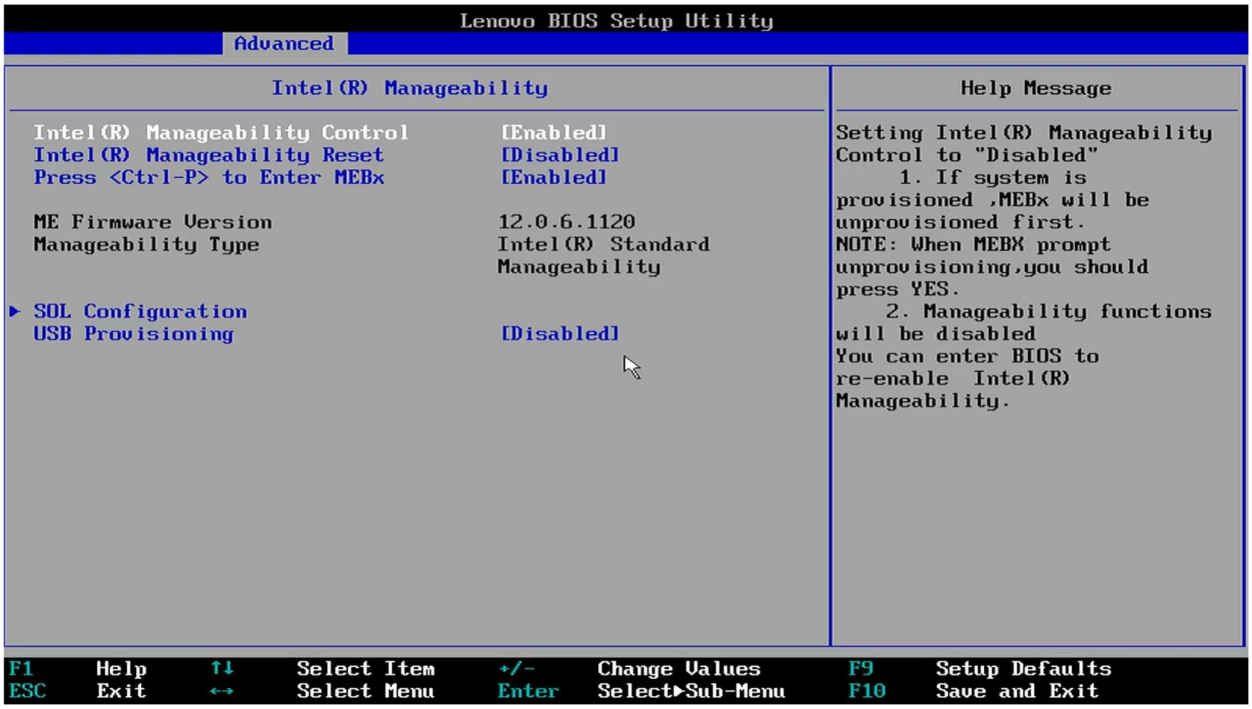


Рисунок 3.4 – Обмеження Intel vPro/AMT через невідповідність редакцій

Альтернативним підходом стало використання карти захоплення відеосигналу для перегляду зображення з HDMI-виходу та безпровідної клавіатури Logitech K400 Plus із вбудованим тачпадом, що працює через USB-приймач на частоті 2.4 ГГц, що дозволяє організувати повноцінний контроль над системою під час налаштування та відладки, забезпечуючи незалежність від обмежень vPro, хоча потребує фізичної присутності обладнання та додаткових витрат (див. рис. 3.5).

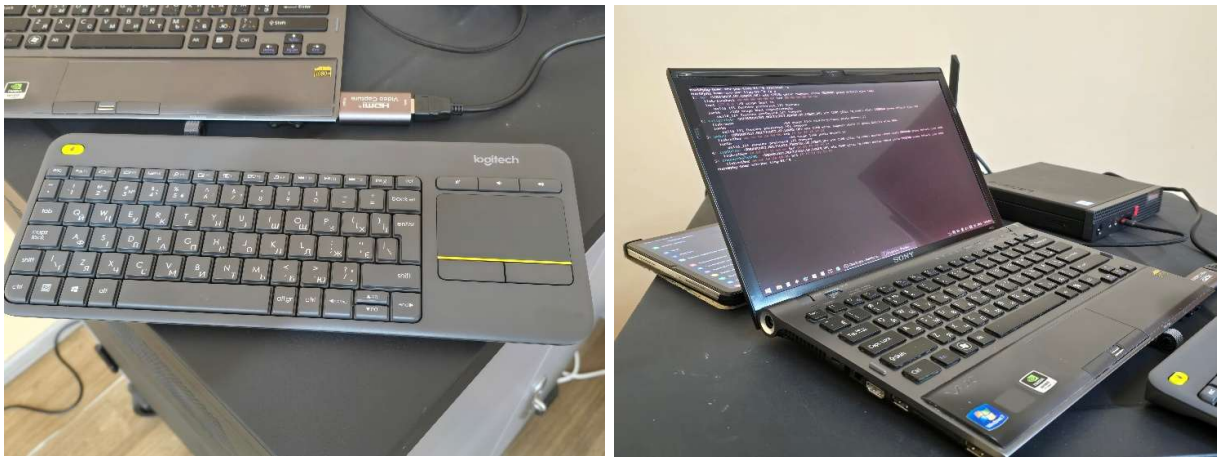


Рисунок 3.5 – Альтернативна схема керування (USB HDMI capture та клавіатура)

У результаті встановлено, що в бюджетних або відновлених корпоративних системах технологія vPro не гарантує повноцінного віддаленого доступу, тому необхідно передбачати альтернативні методи керування інфраструктурою.

Корекція BIOS та ідентифікаторів. У процесі експлуатації також було виявлено проблеми, пов’язані з некоректною ідентифікацією пристрою, зокрема:

— невідповідністю серійного номера;

— змінами після перепрошивки BIOS;

— нестабільністю MAC-адреси.

Все це може негативно впливати на мережеву стабільність, системи моніторингу та механізми ліцензування.

Причинами таких конфліктів виступають попередні модифікації пристрою, заміна компонентів або скидання параметрів без належної синхронізації, у результаті чого формується ланцюг залежностей, де некоректні дані BIOS впливають на мережеві інтерфейси, що в свою чергу відображається на стабільності мережі та ідентифікації вузла.

Методи вирішення включають перепрошивку BIOS за допомогою офіційних утиліт або відновлювальних образів, відновлення серійних номерів через OEM-інструменти, а також фіксацію MAC-адрес на рівні операційної системи або DHCP-сервера, що дозволяє частково стабілізувати роботу мережі (див. рис. 3.6).

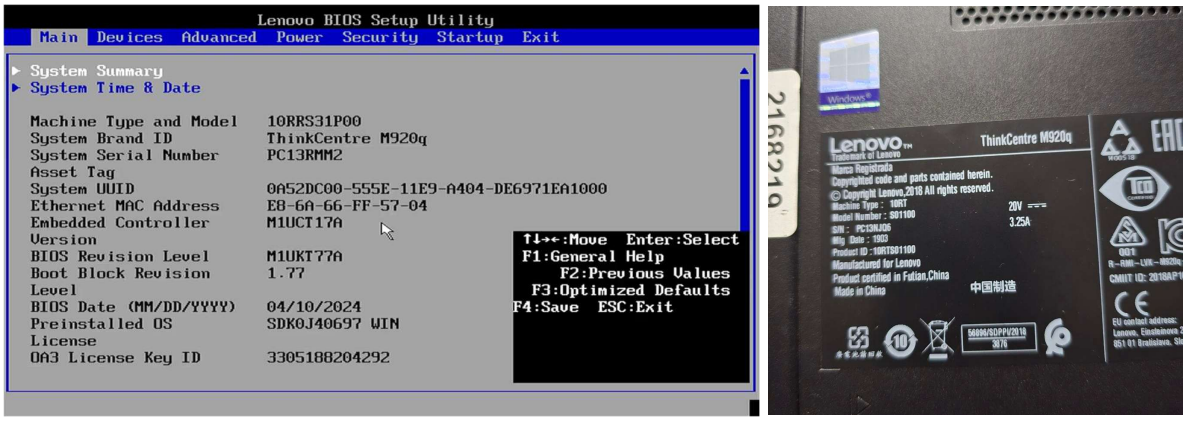


Рисунок 3.6 – Вплив BIOS на мережеву ідентифікацію

Додатково застосовано підхід абстрагування від апаратних ідентифікаторів шляхом використання програмно-визначеної мережі, у межах якої ідентифікація вузлів здійснюється незалежно від фізичних параметрів обладнання, що дозволяє мінімізувати вплив апаратних проблем та забезпечити стабільну мережеву взаємодію навіть у випадках нестандартної конфігурації.

Таблиця 3.1 – Виявлені апаратні проблеми та методи їх вирішення

Проблема	Причина	Прояв	Метод вирішення	Ефективність
NIC Intel	Несумісність драйверів	Відсутність мережі	USB Ethernet / PCIe NIC	Висока
vPro/AMT	Обмежена реалізація	Немає KVM	Альтернативні засоби (HDMI capture)	Середня
BIOS/ID	Некоректні дані	Нестабільність мережі	Перепрошивка, фіксація MAC	Висока
Ізоляція мережі	Double NAT	Відсутність доступу	Tailscale	Висока
USB-модем	Тимчасове рішення	Нестабільність	Замінити NIC	Середня

У підсумку встановлено (див. табл. 3.1), що основними джерелами проблем при побудові серверного вузла на базі неттопа є нестабільність вбудованих мережевих інтерфейсів, обмежена функціональність корпоративних технологій віддаленого керування та проблеми апаратної ідентифікації, при цьому значна частина цих обмежень може бути компенсована програмними рішеннями, що

дозволяє забезпечити стабільну роботу системи навіть у складних та нестандартних умовах експлуатації.

3.2 Порівняльний аналіз архітектур: Zero Trust проти класичних рішень

У межах дослідження доцільним є проведення системного аналізу підходу Zero Trust Networking, реалізованого за допомогою Tailscale, у порівнянні з традиційними VPN-рішеннями, такими як OpenVPN або WireGuard, з урахуванням реальних умов експлуатації, що включають нестандартну мережеву топологію, обмеження апаратного забезпечення та наявність багаторівневого NAT, що істотно впливає на можливість застосування класичних підходів до організації віддаленого доступу.

Переваги над класичним VPN. Класичні VPN-рішення базуються на моделі периметрової безпеки, у межах якої вся внутрішня мережа вважається довіреною, а доступ до неї організовується через централізований шлюз, де клієнт встановлює з’єднання з VPN-сервером, після чого отримує доступ до ресурсів внутрішньої мережі, що формує залежність від єдиної точки входу та створює жорстку централізовану архітектуру.

На відміну від цього підхід Zero Trust реалізує мережу у вигляді mesh-топології, де кожен вузол виступає рівноправним учасником взаємодії, а підключення відбувається безпосередньо до віртуальної мережі, в якій кожен пристрій, незалежно від ролі, може встановлювати захищені з’єднання з іншими вузлами без необхідності проходження через централізований сервер, що усуває залежність від єдиної точки входу та змінює сам принцип організації доступу.

У традиційних VPN-рішеннях часто виникає потреба у наявності публічної IP-адреси на сервері або використанні механізмів порт-форвардингу, що ускладнює розгортання у середовищах із NAT, особливо у випадках багаторівневої трансляції

адрес, де додатково застосовуються допоміжні сервіси на кшталт динамічного DNS, тоді як у розглянутій конфігурації використовується подвійний NAT, сформований поєднанням домашнього маршрутизатора та USB-модему планшета, що створює ізольовану підмережу з адресним простором 192.168.9.0/24 та повною відсутністю прямого доступу до серверного вузла, унаслідок чого будь-які спроби звернення до адрес на кшталт 192.168.9.136 або 192.168.9.138 завершуються безрезультатно.

У таких умовах класичні VPN-рішення без суттєвих змін мережевої архітектури або складних додаткових налаштувань не здатні забезпечити стабільне з’єднання, оскільки механізми порт-форвардингу є недоступними або технічно нереалізовними, зокрема через обмеження мобільних пристроїв на базі Android, які не дозволяють гнучко керувати маршрутизацією або NAT-правилами, тоді як Tailscale автоматично визначає тип мережевого оточення, застосовує механізми обходу NAT, включаючи UDP hole punching, та у разі необхідності використовує relay-сервери, що дозволяє встановлювати з’єднання навіть у складних ізольованих середовищах без зміни існуючої інфраструктури.

Важливою відмінністю є відсутність необхідності відкриття портів у Zero Trust-підході, тоді як класичні VPN потребують явного відкриття портів на маршрутизаторі, що створює додаткові ризики безпеки та вимагає доступу до мережевого обладнання, у той час як у розглянутій архітектурі доступ до маршрутизатора обмежений, а сама мережа ізольована, що робить використання порт-форвардингу неможливим як з технічної, так і з практичної точки зору.

Додатковим аспектом є автоматизація криптографічних процесів та управління доступом, оскільки у класичних VPN-рішеннях необхідно вручну генерувати ключі, налаштовувати конфігурації вузлів та підтримувати їх актуальність, а у випадку OpenVPN також впроваджувати інфраструктуру відкритих ключів із сертифікатами та політиками доступу, тоді як у Tailscale інтеграція з провайдерами ідентифікації забезпечує централізовану автентифікацію, автоматичну генерацію та ротацію ключів без необхідності

ручного втручання, що значно знижує складність адміністрування та ймовірність помилок.

З точки зору відмовостійкості класичні VPN-рішення мають єдину точку відмови у вигляді VPN-сервера, недоступність якого призводить до повної втрати доступу до мережі, тоді як у mesh-архітектурі вузли можуть взаємодіяти безпосередньо між собою, а координаційний сервер використовується лише для встановлення з’єднання і не бере участі у передачі трафіку, що забезпечує вищий рівень стійкості до відмов та кращу масштабованість.

Ізоляція сервісів. У межах традиційних моделей віддаленого доступу користувач після підключення до VPN зазвичай отримує доступ до всієї підмережі, що створює суттєві обмеження з точки зору безпеки, оскільки відсутня гранулярна сегментація доступу і виникає ризик компрометації всієї інфраструктури, адже за наявності доступу до одного вузла відкривається можливість взаємодії з іншими ресурсами, такими як сервери, бази даних або мережеве обладнання, незалежно від їх критичності чи ролі в системі.

У подібній моделі ситуація виглядає таким чином, що клієнт, підключений до VPN, може взаємодіяти одночасно з декількома вузлами, наприклад із сервером за однією IP-адресою, базою даних за іншою та маршрутизатором, при цьому контроль доступу на рівні окремих сервісів або портів часто реалізується обмежено або взагалі відсутній, що ускладнює забезпечення принципу мінімальних привілеїв та підвищує площину потенційної атаки.

У розглянутій реалізації застосовано підхід ізольованого доступу до сервісів, де контейнер із веб-інтерфейсом Portainer розгорнутий у межах віртуальної машини з IP-адресою 192.168.9.138 та доступний через порт 9000, при цьому за допомогою механізмів контролю доступу формується політика, що дозволяє конкретному користувачу взаємодіяти виключно з цим сервісом без можливості доступу до гіпервізора, який розташований за іншою адресою, або до інших контейнерів і внутрішніх служб.

Архітектура доступу передбачає наявність проміжного рівня перевірки політик, у межах якого кожен запит проходить перевірку відповідності визначеним

правилам, у результаті чого доступ надається лише у випадку відповідності заданим умовам, а у протилежному випадку запит відхиляється, що забезпечує чітке розмежування прав і контроль взаємодії між вузлами.

Такий підхід дозволяє мінімізувати площину атаки, обмежуючи доступ лише необхідними сервісами, виключаючи можливість горизонтального переміщення між вузлами мережі та забезпечуючи гнучкість у керуванні доступом, оскільки зміна політик виконується централізовано і не потребує модифікації мережевої топології або фізичної інфраструктури, що значно спрощує адміністрування та підвищує адаптивність системи до змін.

Використання моделі Zero Trust, яка базується на принципі відсутності довіри за замовчуванням, дозволяє забезпечити контроль доступу на рівні окремих сервісів, а не всієї мережі, що є суттєвою перевагою порівняно з класичними VPN-рішеннями, де довіра надається після встановлення з’єднання без детального розмежування прав [30].

Узагальнення результатів аналізу демонструє, що в умовах подвійного NAT, ізольованих підмереж та обмеженого доступу до мережевого обладнання, а також при використанні нестандартних каналів підключення, класичні підходи втрачають ефективність або стають непридатними без значних змін інфраструктури, тоді як підхід Zero Trust забезпечує прозорий і контрольований доступ до окремих сервісів без необхідності втручання у мережеву конфігурацію, зберігає працездатність у складних умовах та підвищує загальний рівень безпеки завдяки ізоляції та централізованому контролю доступу, що робить таку архітектуру доцільною для використання у лабораторних середовищах, малому бізнесі, мобільних рішеннях та експериментальних мережевих інфраструктурах.

3.3 Техніко-економічне обґрунтування та перспективи масштабування

Узагальнення результатів реалізації програмно-апаратного комплексу передбачає оцінку економічної доцільності впровадження та визначення потенціалу подальшого масштабування, при цьому аналіз базується на практичному розгортанні системи на базі неттопа Lenovo ThinkCentre M920q із використанням Proxmox VE, Docker та Tailscale в умовах обмеженої мережевої інфраструктури, що характеризується відсутністю публічної IP-адреси, наявністю подвійного NAT та фактично ізольованим мережевим сегментом [31-32].

Розрахунок вартості (CAPEX). Ключова ідея полягає у використанні відновленого корпоративного обладнання як альтернативи дорогим серверним рішенням, де в ролі базового вузла застосовується компактний неттоп Lenovo ThinkCentre M920q, що дозволяє суттєво знизити капітальні витрати без критичної втрати функціональності для поставлених задач.

Структура витрат формується з урахуванням фактичної конфігурації, де сам неттоп у відновленому стані оцінюється орієнтовно в межах 4–5 тисяч гривень залежно від комплектації, при цьому базова конфігурація зазвичай потребує дооснащення, зокрема оперативною пам’яттю обсягом 32 ГБ DDR4, вартість якої на поточний момент становить близько 10000 гривень, тоді як раніше аналогічні модулі могли коштувати значно дешевше (4000 грн до подорожчення), що відображає динаміку ринку комплектуючих.

Накопичувач формату NVMe обсягом 512 ГБ, наприклад серії Samsung PM981, у розглянутому випадку вже входить до складу неттопа, тому не потребує додаткових витрат (вартість на момент купівлі була в межах 1000 грн, наразі вже 2000 грн), що додатково підвищує економічну ефективність рішення, тоді як питання мережевого підключення вирішується або за рахунок USB Ethernet адаптера, вартість якого становить близько 500 гривень, або шляхом встановлення повноцінної мережевої карти через PCIe райзер, вартість якого коливається в межах

500-1000 гривень, що дозволяє сформувати більш стабільну мережеву конфігурацію.

До допоміжних витрат належать периферійні пристрої, зокрема безпровідна клавіатура з тачпадом Logitech K400 Plus, кабелі та карта захоплення відеосигналу, що сумарно додають приблизно 1800 гривень до загального бюджету, у результаті чого загальна вартість системи формується на рівні близько 20 900 гривень (тоді як на момент купівлі – близько 13 900 гривень), із можливими коливаннями в діапазоні приблизно від 13 до 21 тисяч гривень залежно від конкретної конфігурації та доступності компонентів (див. табл. 3.2).

Таблиця 3.2 – Структура капітальних витрат (CAPEX)

Компонент	Вартість після подорожчення (грн)	Вартість до подорожчення (грн)
Неттоп Lenovo ThinkCentre M920q	5500	5500
Samsung 2x16GB DDR4 3200MHz SODIMM (M471A2K43DB1-CWE)	10000	4000
SSD Samsung PM981a 512Gb NVMe M.2	2000	1000
Lenovo USB-C to Ethernet (RJ45) Gigabit (тимчасове рішення)	500	500
PCIe riser зі Intel Ethernet Server Adapter I350-T4 (NIC) (постійне рішення)	1000	1000
Wireless keyboard Logitech Touch K400 Plus UA	1400	1400
USB 3.0 HDMI capture (1080P)	500	500
Разом	~20900	~13900

Порівняння з класичними серверними рішеннями демонструє суттєву різницю у вартості, оскільки навіть базові сервери корпоративного рівня можуть коштувати від 30 000 гривень і більше без урахування повної комплектації, що робить використання неттопа значно більш доступним варіантом для розгортання інфраструктури (див. рис. 3.7).

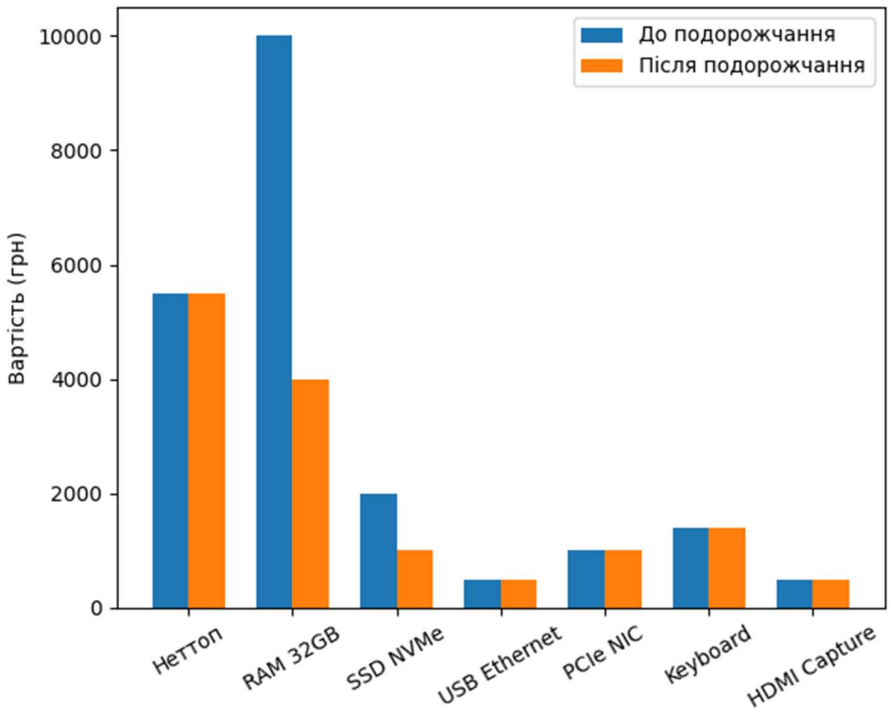


Рисунок 3.7 – Порівняння вартості обраного неттопа
та класичного серверу (CAPEX)

Додатковою перевагою виступає енергоспоживання, яке для неттопа знаходиться в межах приблизно 35–65 Вт під навантаженням і може знижуватися до 10–20 Вт у простої, тоді як серверні рішення зазвичай споживають від 200 до 500 Вт у робочому режимі, що прямо впливає на експлуатаційні витрати та доцільність використання у довгостроковій перспективі (див. рис. 3.8).

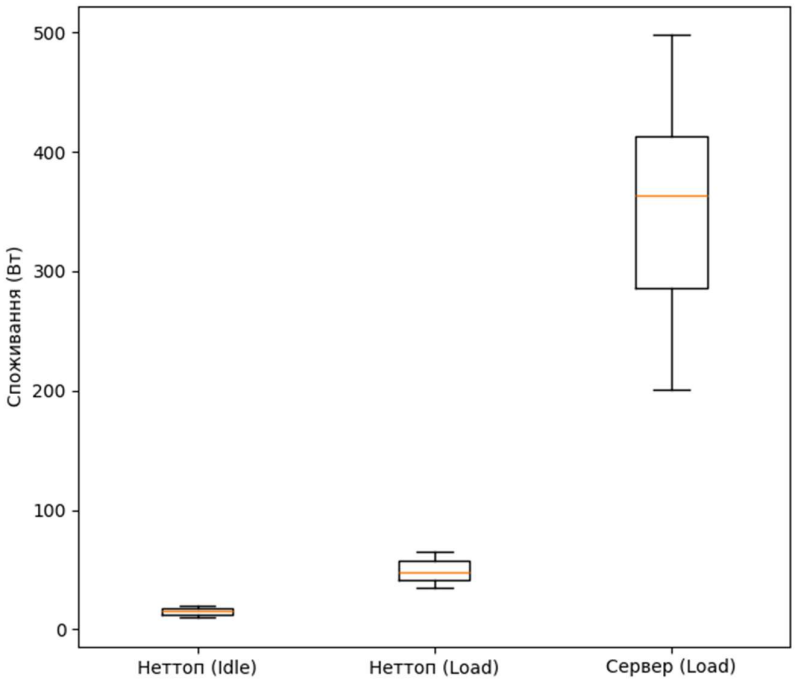


Рисунок 3.8 – Графік енергоспоживання (OPEX)

Форм-фактор неттопа забезпечує компактність, низький рівень шуму та можливість розміщення у будь-якому середовищі без потреби у спеціалізованій серверній інфраструктурі, тоді як класичні сервери потребують стійок, мають значно більші габарити та створюють підвищений рівень шуму, що обмежує їх використання у побутових або навчальних умовах.

Разом із цим враховуються виявлені апаратні обмеження, зокрема проблеми сумісності вбудованих мережевих адаптерів із середовищем Linux та Proxmox VE, що може призводити до відсутності мережевого інтерфейсу або нестабільної роботи, у зв’язку з чим у бюджет обов’язково включається використання альтернативних мережевих рішень, таких як USB Ethernet адаптери або дискретні мережеві карти через PCIe райзер, що може також відкривати можливість використання високошвидкісних інтерфейсів, включаючи 10 Гбіт/с.

У підсумку встановлюється, що використання Lenovo ThinkCentre M920q дозволяє знизити капітальні витрати у декілька разів у порівнянні з традиційними серверними рішеннями, забезпечує швидке розгортання інфраструктури та є доцільним для навчальних лабораторій, стартапів, тестових середовищ і домашніх

серверних сценаріїв, при цьому зберігається можливість подальшого масштабування за рахунок модернізації окремих компонентів та розширення функціональності системи.

Операційна ефективність (OPEX). Операційна складова витрат формується переважно за рахунок використання безкоштовного та відкритого програмного забезпечення, де гіпервізор Proxmox Virtual Environment застосовується без придбання enterprise-підписки, серверна операційна система Ubuntu Server 24.04 LTS розповсюджується безкоштовно, а мережевий рівень реалізовано за допомогою Tailscale у межах безоплатного тарифу, що підтримує значну кількість пристроїв, у результаті чого витрати на ліцензування та підписки фактично відсутні та дорівнюють нулю, а модель експлуатації зводиться до мінімально необхідного рівня адміністрування з помірною складністю налаштування і відносно низькими вимогами до обслуговування.

Енергоспоживання виступає основною статтею операційних витрат, де середнє навантаження неттопа перебуває у діапазоні приблизно 40–60 Вт, а за умов безперервної роботи протягом року із середнім значенням близько 50 Вт формується річне споживання на рівні приблизно 438 кВт·год, що при тарифі близько 4,32 грн за кВт·год дає орієнтовні витрати на електроенергію на рівні близько 1890 грн на рік (див. табл. 3.3).

Таблиця 3.3 – Порівняння енергоспоживання та експлуатаційних витрат (OPEX)

Параметр	Неттоп	Сервер
Споживання (Вт)	40–60	200–500
Річне споживання (кВт·год)	~438	~1750+
Вартість/рік (грн)	~1800–2000	~7000+
Рівень шуму	Низький	Високий
Обслуговування	Просте	Складне
Габарити	Компактні	Великі

При цьому за сценаріїв часткового навантаження або простою фактичні витрати можуть бути істотно нижчими, що значно вигідніше порівняно з класичними серверними рішеннями з високим енергоспоживанням (див. рис. 3.9).

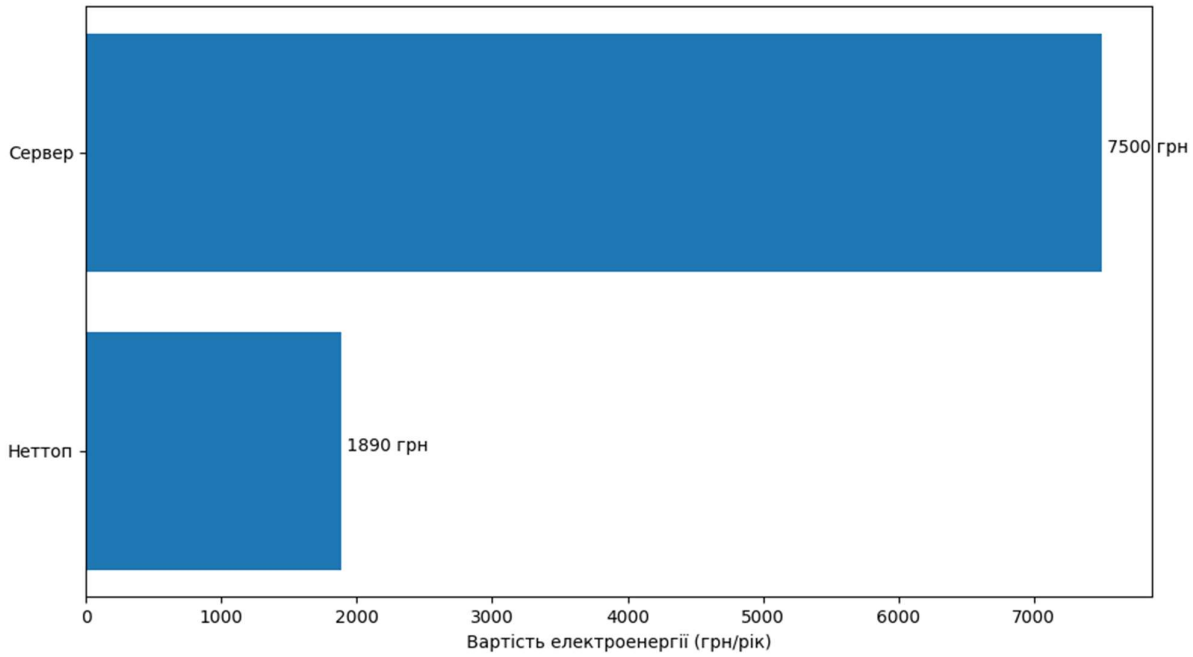


Рисунок 3.9 – Річні витрати електроенергії

Додатковим фактором підвищення ефективності виступає автоматизація мережевих процесів, де Tailscale самостійно формує тунелі, виконує маршрутизацію та знижує потребу у ручному налаштуванні VPN або мережевої інфраструктури, а відсутність необхідності у публічній IP-адресі, доступі до маршрутизатора чи складних правилах NAT суттєво спрощує експлуатацію, у той час як використання контейнеризації через Docker разом із шаблонами Cloud-init забезпечує швидке відновлення сервісів та спрощене розгортання нових компонентів системи, що особливо актуально для лабораторних середовищ, навчальних стендів та невеликих команд розробки.

Узагальнення показує, що експлуатаційні витрати залишаються мінімальними, система не потребує складної підтримки, характеризується високою автономністю та забезпечує стабільну роботу навіть за умов обмежених ресурсів, що робить таку архітектуру економічно доцільною у довгостроковій перспективі.

Масштабування та гнучкість. Питання масштабування та гнучкості розглядається через призму переходу від централізованої моделі до горизонтально розподіленої архітектури, де замість одного потужного сервера застосовується

підхід використання декількох малих вузлів, кожен з яких виконує функції мікросервера, що дозволяє поступово нарощувати обчислювальні ресурси без необхідності повної перебудови інфраструктури, а наступним логічним кроком розвитку такої системи виступає інтеграція легковагових оркестраційних платформ на кшталт K3s, де формується кластер із керуючого вузла та робочих вузлів із розподіленням контейнеризованих сервісів.

Переваги горизонтального масштабування проявляються у підвищенні відмовостійкості, оскільки вихід з ладу окремого вузла не призводить до зупинки всієї системи, а також у можливості балансування навантаження між вузлами та гнучкому розгортанні нових сервісів, де додавання нового вузла зводиться до його підключення до мережі tailnet без складних процедур конфігурації, у той час як інтеграція з Tailscale забезпечує єдиний адресний простір у діапазоні 100.x.x.x незалежно від фізичного розташування пристроїв, що усуває необхідність у традиційних VPN-шлюзах (див. рис. 3.10).

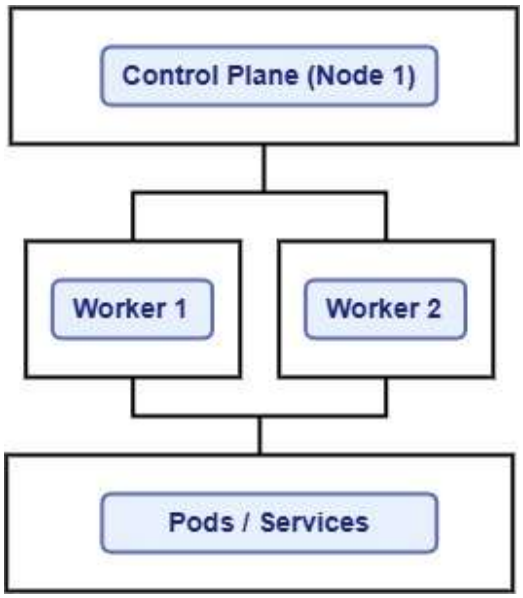


Рисунок 3.10 – Горизонтальне масштабування через кластер вузлів

Розширення інфраструктури можливе шляхом інтеграції локальних вузлів із віддаленими VPS або хмарними ресурсами, що формує гібридну архітектуру типу Homelab зі Cloud, де всі компоненти об’єднані в єдину логічну мережу з прозорою

маршрутизацією та централізованою автентифікацією, а така модель дозволяє змінювати конфігурацію системи без перебудови фізичної мережі, переносити контейнери між вузлами та масштабувати ресурси відповідно до поточних потреб.

У підсумку реалізоване рішення демонструє високу економічну ефективність завдяки низьким показникам капітальних та операційних витрат, поєднує гнучкість, масштабованість і незалежність від складної мережевої інфраструктури, а використання компактного апаратного вузла разом із гіпервізором, контейнеризацією та програмно-визначеною мережею формує універсальну платформу, здатну еволюціонувати у повноцінну розподілену хмарну систему для практичного застосування в різних умовах.

ВИСНОВКИ

Розроблена архітектура програмно-апаратного комплексу продемонструвала можливість побудови функціональної серверної інфраструктури на базі компактного неттопа з використанням гіпервізора, віртуальних машин та контейнеризованих сервісів, де ключовим елементом стала гібридна інтеграція Tailscale, що забезпечила мережеву взаємодію між вузлами незалежно від фізичної топології та обмежень середовища, а застосування ізольованої підмережі у поєднанні з програмно-визначеною мережею дозволило сформувати стабільну та керовану інфраструктуру.

Покрокова конфігурація серверного вузла підтвердила ефективність використання альтернативних підходів до організації мережевого доступу, зокрема застосування USB-модему як шлюзу у випадку апаратних обмежень, впровадження віртуального середовища на основі Cloud-init для автоматизації розгортання та використання контейнеризації для ізоляції прикладних сервісів, що забезпечило гнучкість управління, стабільність адресації та спрощення адміністрування.

Реалізовані алгоритми функціонування Mesh-мережі підтвердили здатність системи працювати в умовах подвійного NAT, де механізми NAT Traversal, Subnet Router та MagicDNS забезпечили прозору маршрутизацію трафіку, доступ до внутрішніх ресурсів та зручну ідентифікацію вузлів, а практична перевірка засвідчила можливість «пробиття» ізольованого середовища без відкриття портів.

Аналіз процесів оптимізації та експлуатації показав, що навіть за наявності апаратних обмежень і проблем сумісності можлива побудова стабільної системи за рахунок програмних рішень, а використання підходу Zero Trust забезпечує вищий рівень безпеки порівняно з класичними VPN за рахунок ізоляції доступу та відсутності залежності від мережевого периметра, при цьому низькі капітальні та операційні витрати у поєднанні з можливістю горизонтального масштабування формують економічно доцільну та гнучку платформу, придатну для використання в лабораторних, навчальних і практичних середовищах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a VPN? Cloudflare. URL: https://www.cloudflare.com/learning/network-layer/what-is-a-vpn/ (дата звернення: 25.02.2026).
2. VPN limitations in modern networks. Cisco. URL: https://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-clients/what-is-vpn.html (дата звернення: 28.02.2026).
3. Zero Trust Architecture. NIST. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf (дата звернення: 03.03.2026).
4. What is Zero Trust? Microsoft Security. URL: https://www.microsoft.com/en-us/security/business/zero-trust (дата звернення: 04.03.2026).
5. Software-Defined Networking (SDN). Open Networking Foundation. URL: https://opennetworking.org/sdn-definition/ (дата звернення: 05.03.2026).
6. Mesh Networking Overview. IBM. URL: https://www.ibm.com/topics/mesh-networking (дата звернення: 06.03.2026).
7. How Tailscale Works. Tailscale Docs. URL: https://tailscale.com/kb/1151/how-tailscale-works/ (дата звернення: 09.03.2026).
8. Control Plane vs Data Plane. Cloudflare. URL: https://www.cloudflare.com/learning/network-layer/what-is-control-plane/ (дата звернення: 11.03.2026).
9. What is SSO? Okta. URL: https://www.okta.com/identity-101/what-is-single-sign-on/ (дата звернення: 13.03.2026).
10. Identity and Access Management (IAM). AWS. URL: https://aws.amazon.com/iam/ (дата звернення: 14.03.2026).
11. WireGuard Protocol Overview. WireGuard. URL: https://www.wireguard.com/protocol/ (дата звернення: 17.03.2026).
12. Peer-to-Peer Networking Explained. TechTarget. URL: https://www.techtarget.com/searchnetworking/definition/peer-to-peer (дата звернення: 18.03.2026).
13. Virtualization vs Containerization. Scale Computing. URL: https://www.scalecomputing.com/resources/whats-the-difference-between-virtualization-vs-containerization (дата звернення: 19.03.2026).
14. Docker Overview. Docker Docs. URL: https://docs.docker.com/get-started/overview/ (дата звернення: 21.03.2026).
15. Docker Networking. Docker Docs. URL: https://docs.docker.com/network/ (дата звернення: 24.03.2026).
16. Docker Compose Documentation. Docker Docs. URL: https://docs.docker.com/compose/ (дата звернення: 26.03.2026).
17. Edge Computing Explained. Red Hat. URL: https://www.redhat.com/en/topics/edge-computing (дата звернення: 28.03.2026).

18. Energy Efficiency in Computing. Intel. URL: https://www.intel.com/content/www/us/en/it-management/energy-efficiency.html (дата звернення: 01.04.2026).
19. Hardware Virtualization Basics. VMware. URL: https://www.vmware.com/topics/glossary/content/hardware-virtualization (дата звернення: 03.04.2026).
20. SSH Security Best Practices. SSH.com. URL: https://www.ssh.com/academy/ssh/security (дата звернення: 04.04.2026).
21. Tailscale SSH Documentation. Tailscale Docs. URL: https://tailscale.com/kb/1193/tailscale-ssh/ (дата звернення: 06.04.2026).
22. Proxmox VE Documentation. Proxmox. URL: https://www.proxmox.com/en/proxmox-ve (дата звернення: 07.04.2026).
23. Ubuntu Server Guide. Canonical. URL: https://ubuntu.com/server/docs (дата звернення: 09.04.2026).
24. Portainer Documentation. Portainer. URL: https://docs.portainer.io/ (дата звернення: 16.04.2026).
25. NAT Traversal Techniques. IETF. URL: https://datatracker.ietf.org/doc/html/rfc5128 (дата звернення: 17.04.2026).
26. Subnet Routing in Tailscale. Tailscale Docs. URL: https://tailscale.com/kb/1019/subnets/ (дата звернення: 20.04.2026).
27. MagicDNS Overview. Tailscale Docs. URL: https://tailscale.com/kb/1081/magicdns/ (дата звернення: 24.04.2026).
28. Intel Ethernet Issues Linux. Intel Community. URL: https://community.intel.com/ (дата звернення: 28.04.2026).
29. Intel vPro and AMT Overview. Intel. URL: https://www.intel.com/content/www/us/en/architecture-and-technology/vpro/what-is-vpro.html (дата звернення: 29.04.2026).
30. Zero Trust vs VPN Comparison. Palo Alto Networks. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-zero-trust (дата звернення: 02.05.2026).
31. Лузан Я.Д., Торошанко Я.І. Програмно-апаратний комплекс віддаленого доступу до серверної інфраструктури на базі Tailscale. «Сучасний стан та перспективи розвитку IoT» : VII Науково-техн. конф., м. Київ, 20 квітня 2026 р. С.104-106.
32. Лузан Я.Д., Торошанко Я.І. Програмно-визначена mesh-мережа для захищеного доступу до контейнеризованих сервісів на базі Tailscale. «Застосування програмного забезпечення в ІКТ» : VII Всеукр. науково-техн. конф., м. Київ, 23 квітня 2026 р. (подано до публікації).

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерної інженерії



КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

НА ТЕМУ: «ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ВІДДАЛЕНОГО ДОСТУПУ ДО СЕРВЕРНОЇ ІНФРАСТРУКТУРИ НА БАЗІ TAILSCALE»

Виконав студент групи КІД-41
Лузан Ярослав Дмитрович
Керівник кваліфікаційної роботи:
Торошанко Ярослав Іванович,
канд. техн. наук, доц., с.н.с.

Київ – 2026

Мета, об’єкт, предмет дослідження

Мета роботи – спрощення та підвищення безпеки віддаленого доступу до серверної інфраструктури в ізольованих мережах за рахунок застосування принципів Zero Trust та технологій Tailscale

Об’єкт дослідження – процеси організації віддаленого доступу до серверних ресурсів у мережах з обмеженою або ізольованою інфраструктурою

Предмет дослідження – засоби та технології побудови програмно-апаратних комплексів віддаленого доступу з використанням Tailscale, контейнеризації Docker та гіпервізорів

Актуальність роботи

Розглянуто еволюцію підходів до віддаленого доступу від класичних VPN до моделі Zero Trust, принципи програмно-визначених мереж та архітектуру Tailscale, включаючи механізми NAT Traversal і P2P-з'єднання. Проаналізовано обмеження традиційних VPN у складних мережевих умовах і визначено переваги підходу Zero Trust, а також узагальнено роль контейнеризації та компактних апаратних платформ.

Особливу увагу приділено розгортанні системи на базі неттопа з використанням віртуалізації та контейнерів, інтегрованих у Tailscale. Реалізовано доступ до ізольованих сервісів без відкриття портів, підтверджено обхід подвійного NAT, а також оцінено ефективність і можливості масштабування рішення.

Порівняння підходів

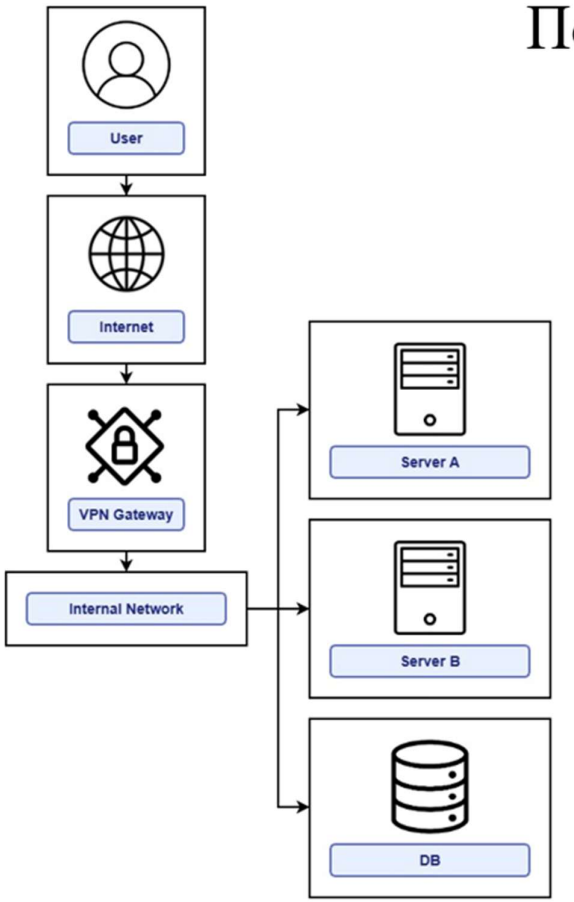


Рис. 1.1 – Класична модель VPN-доступу (Perimeter Security)

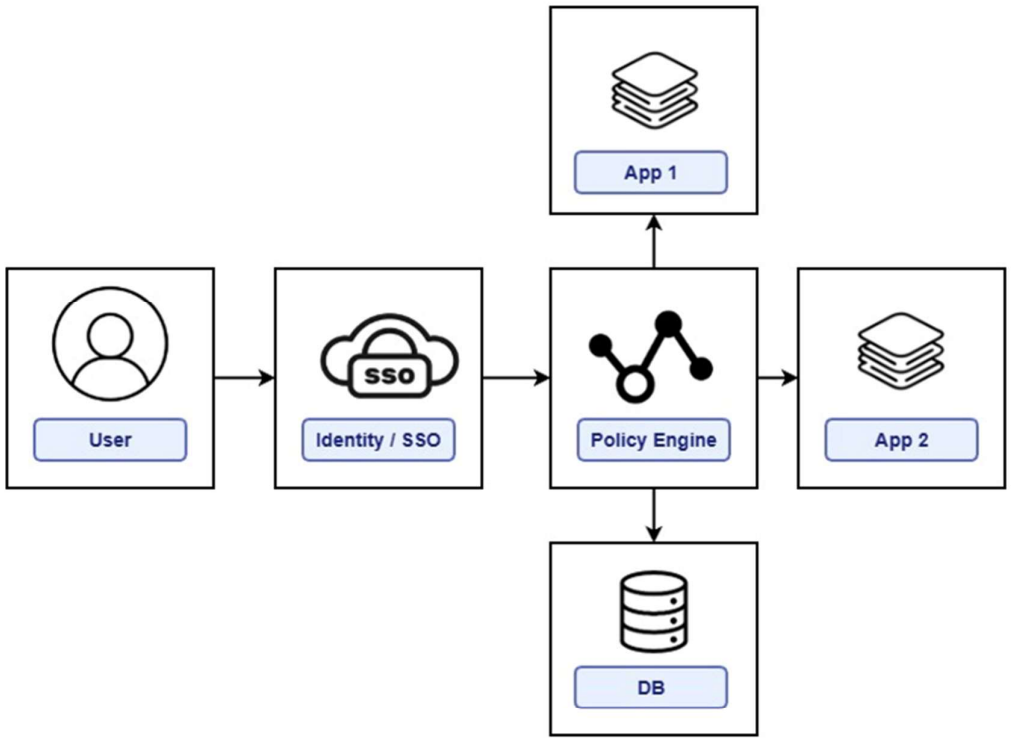


Рис. 1.2 – Концепція Zero Trust Network Access (ZTNA)

Загальна архітектура рішення

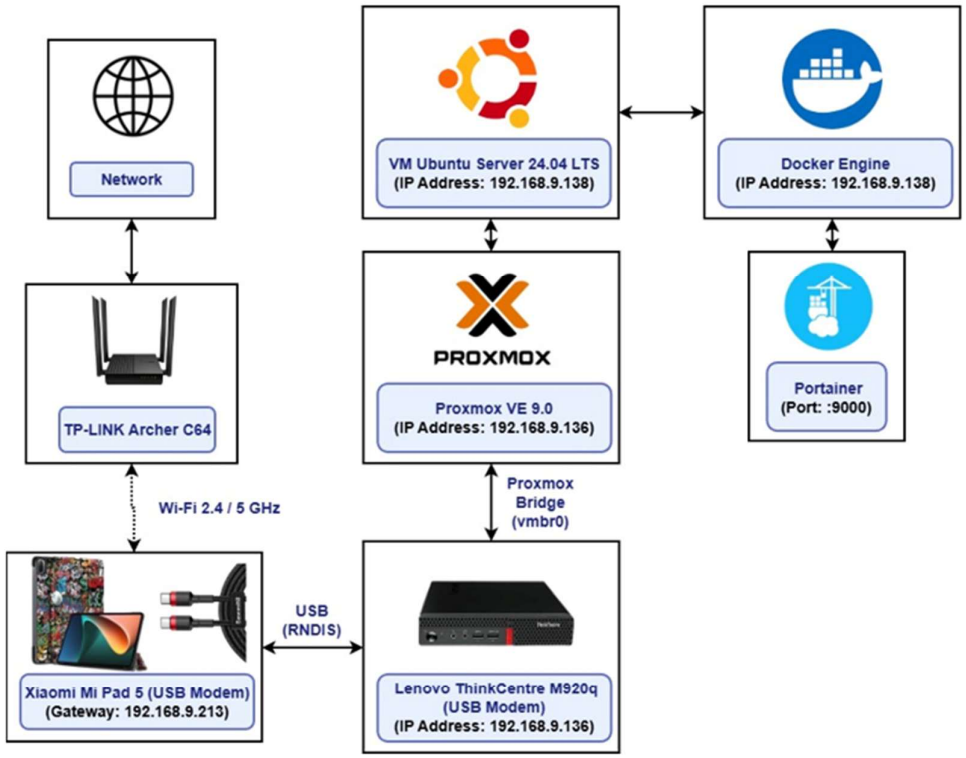


Рис. 2.1 – Архітектура програмно-апаратного комплексу віддаленого доступу до серверної інфраструктури на базі Tailscale

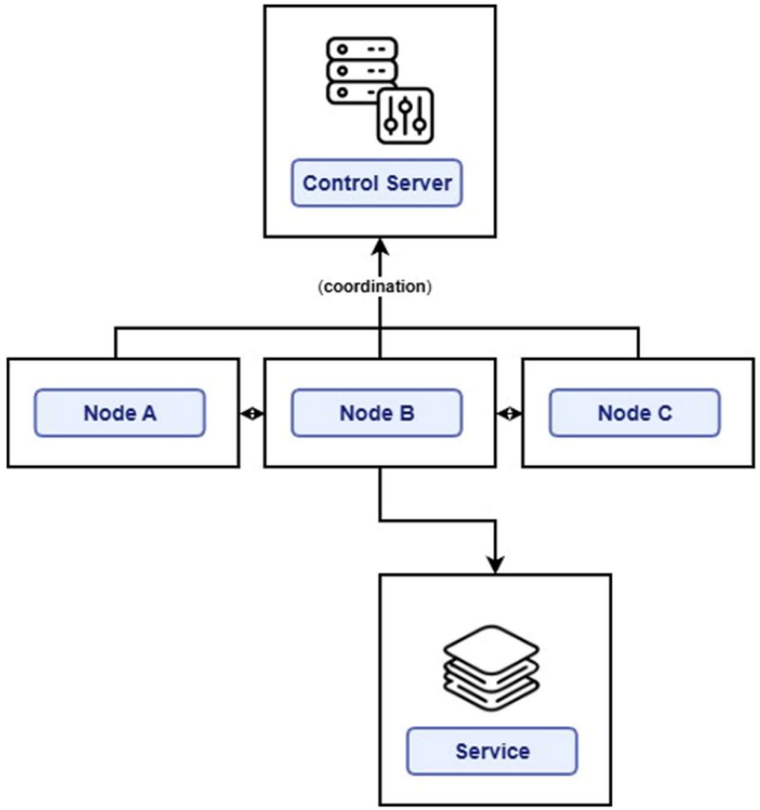


Рис. 1.4 – Загальна архітектура Tailscale

Апаратна реалізація



Рис. 2.2 – Апаратна конфігурація серверного вузла
на базі Lenovo ThinkCentre M920q

Табл. 2.1 – Апаратна конфігурація серверного вузла

Компонент	Модель	Характеристики
CPU	Intel Core i5-8400T	6 ядер, до 3.3 ГГц
RAM	Samsung DDR4	32 ГБ (2×16 ГБ)
SSD	Samsung PM981	512 ГБ NVMe
NIC	Intel I219-LM	Gigabit Ethernet
Платформа	Lenovo ThinkCentre M920q	x86_64, компактний форм-фактор
Додатково	USB Ethernet / PCIe NIC	Альтернативний інтерфейс

Розгортання середовища

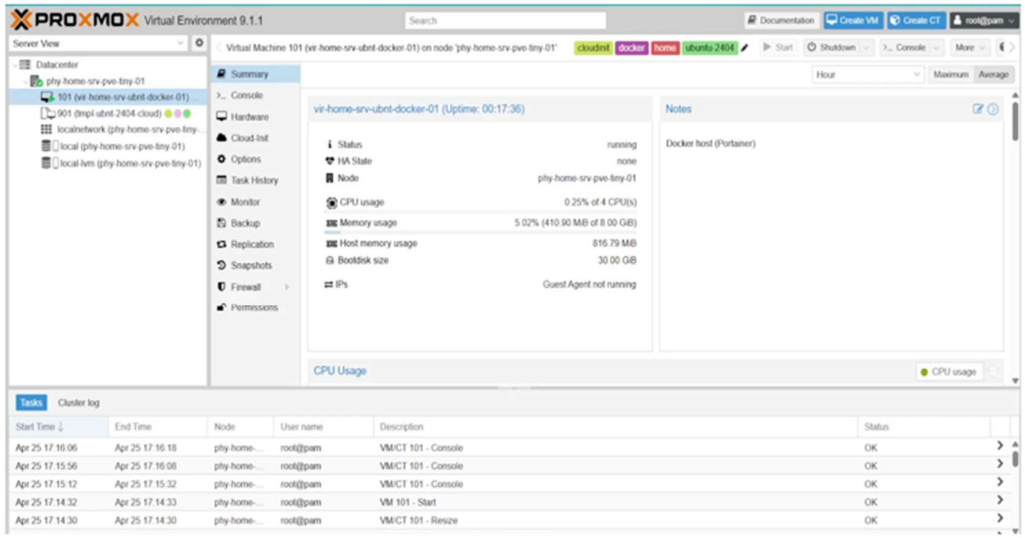


Рис. 2.7 – Створення віртуальної машини

```
docker@vir-home-srv-ubnt-docker-01:~/portainer$ docker compose up -d
[+] up 13/13
 ✓ Image portainer/portainer-ce:lts Pulled                                    6.1s
 ✓ Network docker                   Created                                   0.0s
 ✓ Container portainer              Started                                   0.4s
docker@vir-home-srv-ubnt-docker-01:~/portainer$ docker ps
CONTAINER ID   IMAGE                          COMMAND        CREATED         STATUS         PORTS
                                  NAMES
bf94de6da16d   portainer/portainer-ce:lts   "/portainer"   5 seconds ago   Up 4 seconds   0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp, 0.0.0.0:9443->9443/tcp,
 [::]:9443->9443/tcp, 9000/tcp   portainer
docker@vir-home-srv-ubnt-docker-01:~/portainer$
```

Рис. 2.10 – Розгортання Docker-контейнера

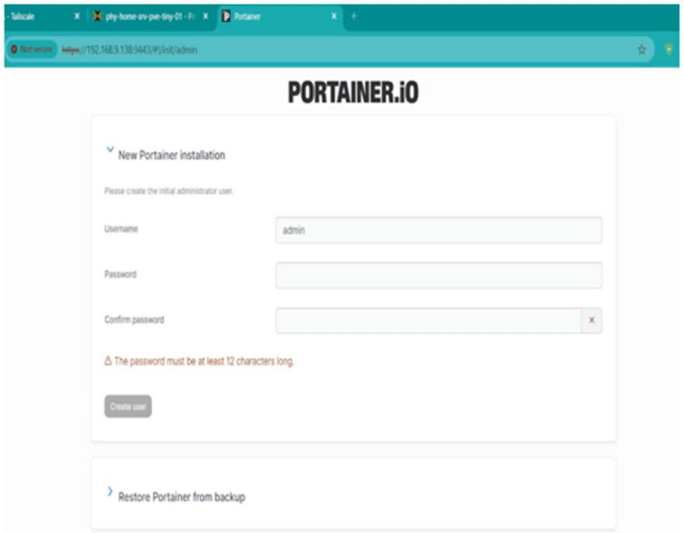


Рис. 2.11 – Інтерфейс Portainer після старту контейнера

Інтеграція Tailscale

```
root@phy-home-srv-pve-tiny-01:/etc/network# apt install -y tailscale
Installing:
  tailscale

Installing dependencies:
  tailscale-archive-keyring

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 126
  Download size: 36.6 MB
  Space needed: 71.9 MB / 91.5 GB available

Get:1 https://pkgs.tailscale.com/stable/debian trixie/main amd64 tailscale amd64 1.96.4 [36.6 MB]
Get:2 https://pkgs.tailscale.com/stable/debian trixie/main all tailscale-archive-keyring all 1.35.181 [3,082 B]
Fetched 36.6 MB in 2s (20.9 MB/s)
Selecting previously unselected package tailscale.
(Reading database ... 52049 files and directories currently installed.)
Preparing to unpack .../tailscale_1.96.4_amd64.deb ...
Unpacking tailscale (1.96.4) ...
Selecting previously unselected package tailscale-archive-keyring.
Preparing to unpack .../tailscale-archive-keyring_1.35.181_all.deb ...
Unpacking tailscale-archive-keyring (1.35.181) ...
Setting up tailscale-archive-keyring (1.35.181) ...
Setting up tailscale (1.96.4) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/tailscaled.service' → '/usr/lib/systemd/system/tailscaled.service'.
root@phy-home-srv-pve-tiny-01:/etc/network#
```

Рис. 2.8 – Встановлення Tailscale



Рис. 2.4 – Інтеграція Tailscale у локальну мережу

```
oot@phy-home-srv-pve-tiny-01:/etc/network# systemctl status tailscaled.service
 tailscaled.service - Tailscale node agent
     Loaded: loaded (/usr/lib/systemd/system/tailscaled.service; enabled; preset: enabled)
     Active: active (running) since Sat 2026-04-25 14:22:04 EEST; 32s ago
Invocation: e413333025054 1bc8a952a93f85b69ad
       Docs: https://tailscale.com/docs/
   Main PID: 28324 (tailscaled)
     Status: "Needs login: "
      Tasks: 12 (limit: 38132)
     Memory: 59.6M (peak: 60.1M)
        CPU: 936ms
     CGroup: /system.slice/tailscaled.service
             └─28324 /usr/sbin/tailscaled --state=/var/lib/tailscale/tailscaled.state --socket=/run/tailscale/tailscaled.sock --port=41641

пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: Switching ipn state NoState -> NeedsLogin (WantRunning=false, nm=false)
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: health(warnable=warntunning-failure): error: Tailscale is stopped.
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: blockEngineUpdates(true)
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: wgengine: Reconfig: configuring userspace WireGuard config (with 0/0 peers)
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: wgengine: Reconfig: configuring router
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: wgengine: Reconfig: user dialer
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: wgengine: Reconfig: configuring DNS
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: dns: Set: {DefaultResolvers:[] Routes:{} SearchDomains:[] Hosts:0}
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: dns: Resolvercfg: {Routes:{} Hosts:0 LocalDomains:[]}
пт 25 14:22:04 phy-home-srv-pve-tiny-01 tailscaled[28324]: dns: OScfg: {}
oot@phy-home-srv-pve-tiny-01:/etc/network# tailscale up

) authenticate, visit:

        https://login.tailscale.com/a/e27224c014719
```

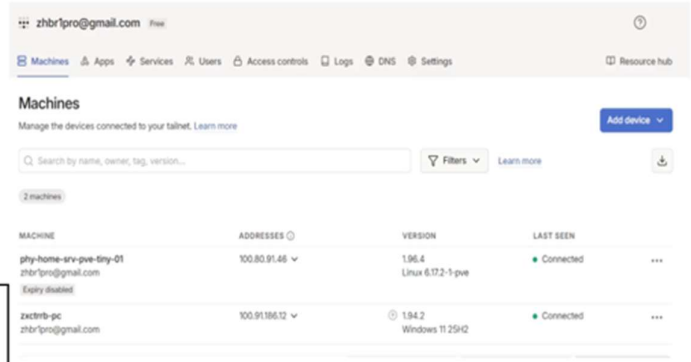


Рис. 2.9 – Авторизація вузла в tailnet

Робота в умовах Double NAT

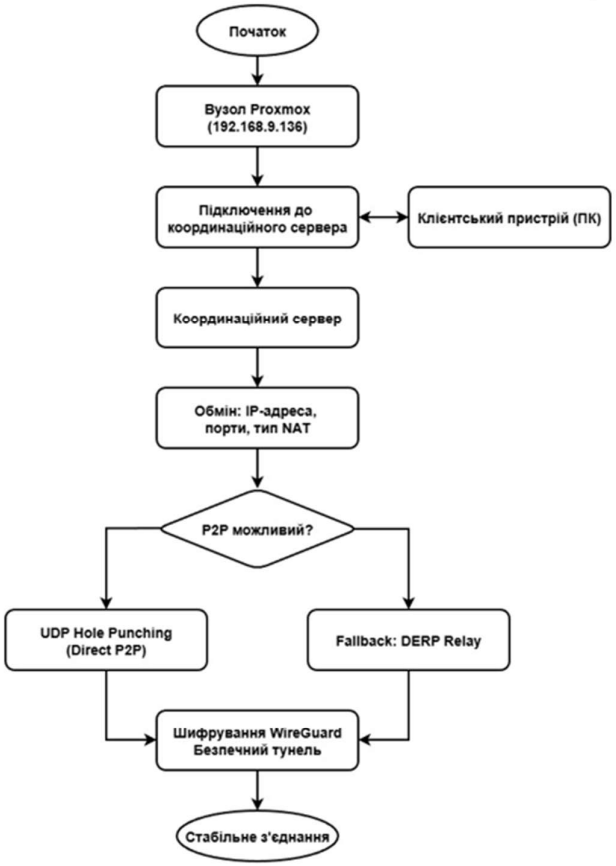


Рис. 2.12 – Алгоритм NAT Traversal

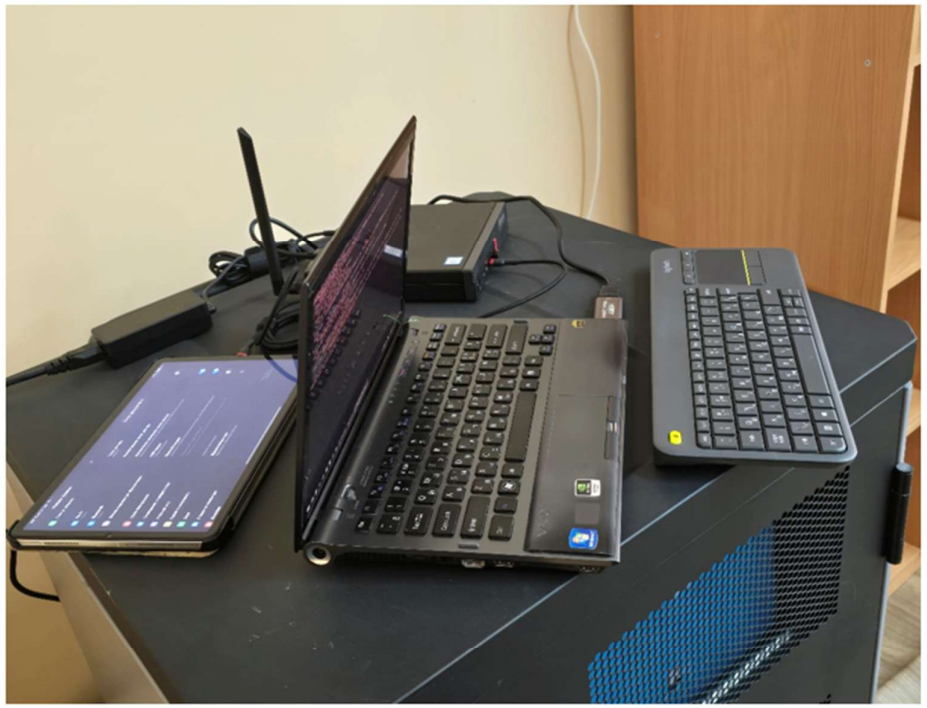


Рис. 2.5 – Підключення через USB-модем

Демонстрація роботи системи

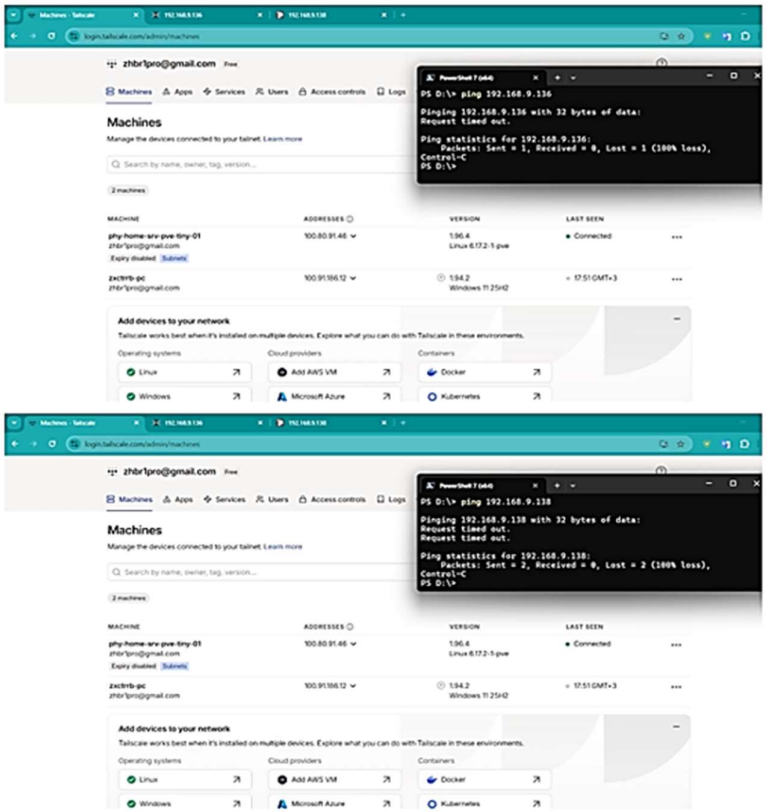


Рис. 2.16 – Відсутність доступу без Tailscale (Ping test)

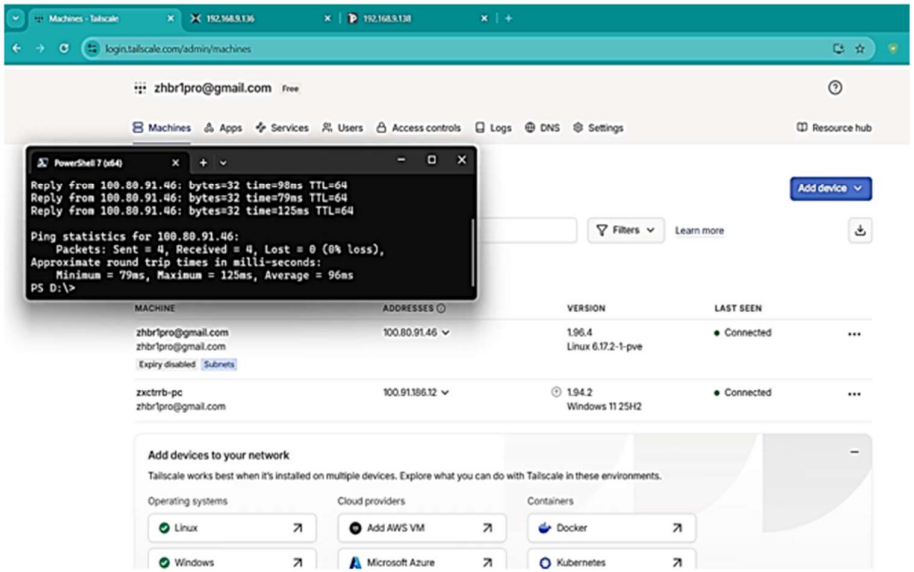


Рис. 2.17 – Доступ після активації Tailscale (Ping test)

Демонстрація роботи системи

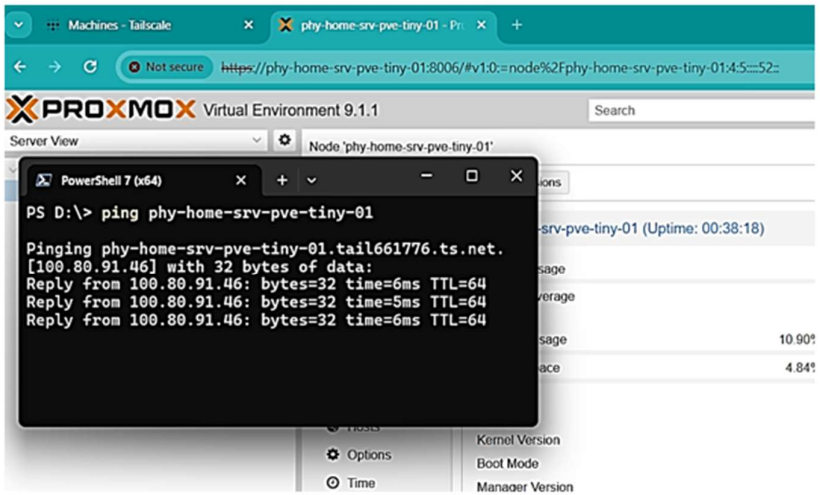


Рис. 2.18 – Доступ через MagicDNS (браузер)

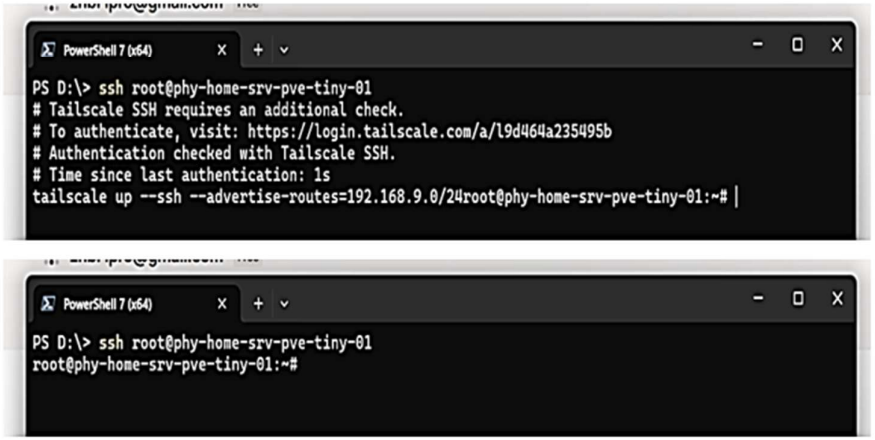


Рис. 2.19 – SSH-підключення без пароля через термінал

Проблеми та їх вирішення

```
phy-home-srv-pve-tiny-01 login: root
Password:
Linux phy-home-srv-pve-tiny-01 6.17.2-1-pve #1 SMP PREEMPT_DYNAMIC PMX 6.17.2-1 (2025-10-21T11:55Z) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@phy-home-srv-pve-tiny-01:~# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: vmbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 6e:6b:a5:2f:ae:8d brd ff:ff:ff:ff:ff:ff
root@phy-home-srv-pve-tiny-01:~# ip -c a^C
root@phy-home-srv-pve-tiny-01:~# modprobe e1000e
root@phy-home-srv-pve-tiny-01:~# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: vmbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 6e:6b:a5:2f:ae:8d brd ff:ff:ff:ff:ff:ff
root@phy-home-srv-pve-tiny-01:~# dmesg | grep -i e1000e
[    1.869236] e1000e: Intel(R) PRO/1000 Network Driver
[    1.869240] e1000e: Copyright(c) 1999 - 2015 Intel Corporation.
[    1.869536] e1000e 0000:00:1f.6: Interrupt Throttling Rate (ints/sec) set to dynamic conservative mode
[    1.942392] e1000e 0000:00:1f.6: The NVM Checksum Is Not Valid
[    1.975903] e1000e 0000:00:1f.6: probe with driver e1000e failed with error -5
root@phy-home-srv-pve-tiny-01:~# _
```

Рис. 3.1 – Проблеми ініціалізації мережевих адаптерів Intel

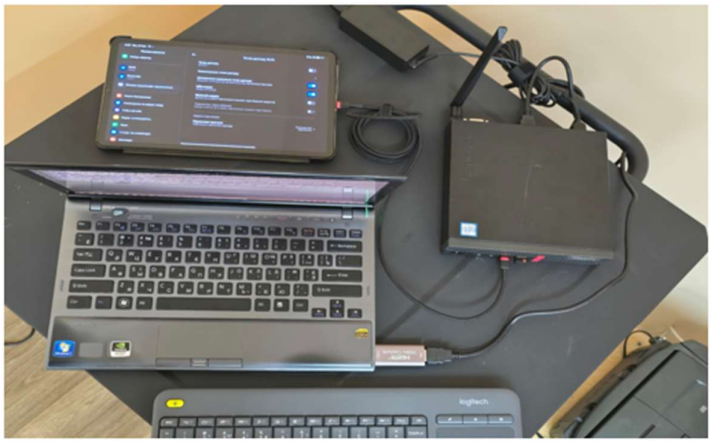


Рис. 3.2 – Використання USB Ethernet адаптера або USB-модема

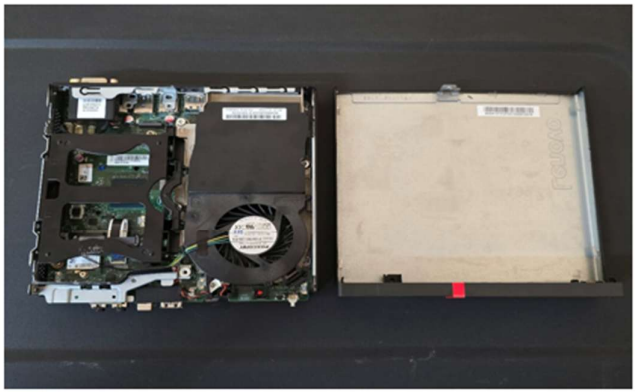


Рис. 3.3 – Можливість використання повноцінного PCIe NIC

Економічна ефективність

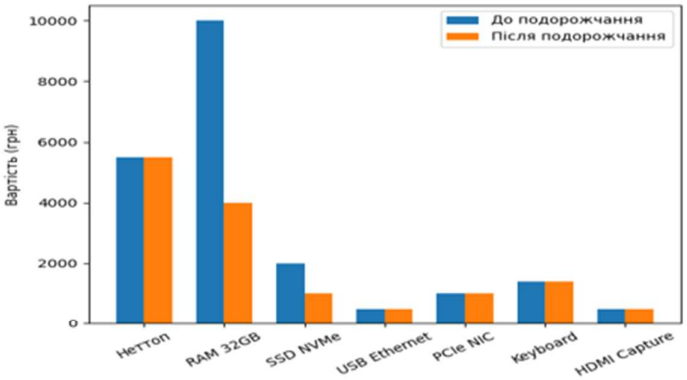


Рис. 3.7 – Порівняння вартості обраного неттопа та класичного серверу (CAPEX)

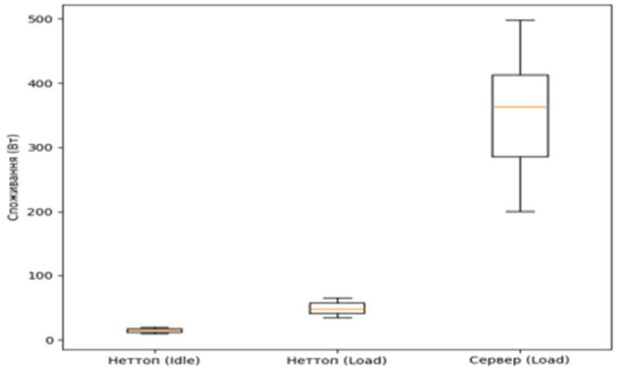


Рис. 3.8 – Графік енергоспоживання (OPEX)

Табл. 3.2 – Структура капітальних витрат (CAPEX)

Компонент	Вартість після подорожчення (грн)	Вартість до подорожчення (грн)
Неттоп Lenovo ThinkCentre M920q	5500	5500
Samsung 2x16GB DDR4 3200MHz SODIMM (M471A2K43DB1-CWE)	10000	4000
SSD Samsung PM981a 512Gb NVMe M.2	2000	1000
Lenovo USB-C to Ethernet (RJ45) Gigabit (тимчасове рішення)	500	500
PCIe riser зі Intel Ethernet Server Adapter I350-T4 (NIC) (постійне рішення)	1000	1000
Wireless keyboard Logitech Touch K400 Plus UA	1400	1400
USB 3.0 HDMI capture (1080P)	500	500
Разом	~20900	~13900

Табл. 3.3 – Порівняння енергоспоживання та експлуатаційних витрат (OPEX)

Параметр	Неттоп	Сервер
Споживання (Вт)	40–60	200–500
Річне споживання (кВт·год)	~438	~1750+
Вартість/рік (грн)	~1800–2000	~7000+
Рівень шуму	Низький	Високий
Обслуговування	Просте	Складне
Габарити	Компактні	Великі

Висновки

Розроблений програмно-апаратний комплекс підтвердив можливість побудови функціональної серверної інфраструктури на базі компактного неттопа з використанням гіпервізора, віртуальних машин, контейнеризації та інтеграції Tailscale. Практична реалізація показала ефективність альтернативних рішень, зокрема застосування USB-модему як шлюзу, автоматизованого розгортання через Cloud-init і контейнерної ізоляції сервісів, що забезпечило стабільність роботи, гнучкість адміністрування та незалежність від фізичних мережевих обмежень.

Практична перевірка підтвердила працездатність системи в умовах подвійного NAT завдяки механізмам NAT Traversal, Subnet Router і MagicDNS, які забезпечили прозорий доступ до ресурсів без відкриття портів. Проведений аналіз засвідчив, що поєднання підходу Zero Trust, низьких капітальних та операційних витрат, а також можливості масштабування формує безпечну, економічно доцільну й перспективну платформу для лабораторного, навчального та практичного використання.

Результати впровадження

1. Лузан Я.Д., Торошанко Я.І. Програмно-апаратний комплекс віддаленого доступу до серверної інфраструктури на базі Tailscale. «Сучасний стан та перспективи розвитку IoT» : VII Науково-техн. конф., м. Київ, 20 квітня 2026 р. С.104-106.
2. Лузан Я.Д., Торошанко Я.І. Програмно-визначена mesh-мережа для захищеного доступу до контейнеризованих сервісів на базі Tailscale. «Застосування програмного забезпечення в ІКТ» : VII Всеукр. науково-техн. конф., м. Київ, 23 квітня 2026 р. (подано до публікації).