

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи магістра
на тему: «Методи та технології безпечного віддаленого
доступу користувачів корпоративних мереж»

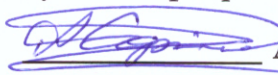
Виконав студент 2-го курсу групи КБм-24-1
спеціальності 125 «Кібербезпека та захист інформації»

 Владислав ФОМІН

Керівник: старший викладач кафедри КІ та КБ,

 Олександра ПОКОТИЛО

Рецензент: завідувач кафедри КІ та КБ, к.т.н., доцент

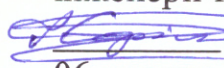
 Андрій ЄФІМЕНКО

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека та захист інформації»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій ЄФІМЕНКО
06 жовтня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу магістра

Студент: Фомін Владислав Васильович

Тема роботи: «Методи та технології безпечного віддаленого доступу користувачів корпоративних мереж»

затверджена Наказом університету від 06 жовтня 2025 р. №566/с

Термін здачі студентом закінченої роботи 16 грудня 2025 р.

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

1. Процеси організації та захисту віддаленого доступу в корпоративних мережах.
2. Методи та технології побудови захищеної інфраструктури віддаленого доступу з використанням VPN WireGuard, RDP, VNC (сесійного та контейнеризованого) та SIEM (Wazuh).

Обладнання та програмне забезпечення:

1. Програмне забезпечення емуляції мережі: GNS3.
2. Віртуальні маршрутизатори: MikroTik Cloud Hosted Router v7.x.
3. Віртуальні машини: Windows Server 2019, Ubuntu Server 24.04.
4. Технології та протоколи: WireGuard, RDP, VNC (TigerVNC), Docker, Docker Compose.
5. Система моніторингу: Wazuh (Indexer, Manager, Dashboard, Agent).
6. Інструменти тестування та керування: nmap, ping, SSH-клієнт, VNC Viewer, RDP Client.

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

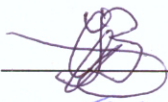
Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

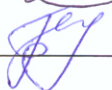
Студент  Владислав ФОМІН

Керівник  Олександра ПОКОТИЛО

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми	06.10.2025	
2	Постановка завдання та визначення мети і завдань дослідження	06.10.2025 – 08.10.2025	
3	Аналіз літературних джерел	09.10.2025 – 14.10.2025	
4	Порівняльний аналіз технологій	15.10.2025 – 17.10.2025	
5	Обґрунтування вибору технологій	20.10.2025 – 22.10.2025	
6	Проектування архітектури мережі та IP-адресації	23.10.2025 – 28.10.2025	
7	Проектування VPN-рішення	29.10.2025 – 31.10.2025	
8	Проектування рішень віддаленого доступу	03.10.2025 – 05.11.2025	
9	Проектування системи захисту	06.11.2025 – 07.11.2025	
10	Реалізація тестового стенду в GNS3	10.11.2025 – 14.11.2025	
11	Налаштування SIEM	17.11.2025 – 19.11.2025	
12	Проведення експериментального дослідження безпеки	20.11.2025 – 24.11.2025	
13	Проведення експериментального дослідження продуктивності	25.11.2025 – 28.11.2025	
14	Аналіз результатів дослідження, формулювання висновків	01.12.2025 – 03.12.2025	
15	Оформлення пояснювальної записки	04.12.2025 – 10.12.2025	
16	Оформлення презентації	11.12.2025 – 15.12.2025	
17	Захист	16.12.2025	

Студент  Владислав ФОМІН

Керівник  Олександра ПОКОТИЛО

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи на тему «Методи та технології безпечного віддаленого доступу користувачів корпоративних мереж» складається зі вступу, трьох розділів, висновку та списку використаної літератури.

Текстова частина викладена на 111 сторінках друкованого тексту. Пояснювальна записка має 99 сторінки та додаток на 12 сторінок. Список використаних джерел містить 17 найменувань і займає 2 сторінки. В роботі наведено 89 рисунків та 4 таблиці. Загальний обсяг роботи 111 сторінок.

Метою випускної кваліфікаційної роботи є дослідження методів безпечного віддаленого доступу та розробка захищеної інфраструктури корпоративної мережі.

У роботі проведено аналіз сучасних технологій VPN (WireGuard, OpenVPN) та протоколів віддаленого доступу (RDP, VNC). Спроектовано архітектуру мережі з використанням тунелювання WireGuard на маршрутизаторах MikroTik. Реалізовано гібридну систему доступу: RDP для серверів Windows та VNC сесійно та в ізольованих Docker-контейнерах для Linux. Впроваджено систему моніторингу подій безпеки (SIEM) на базі Wazuh. Проведено експериментальне дослідження, яке підтвердило ефективність захисту від атак (brute-force, сканування портів) та переваги контейнеризації для забезпечення безпеки користувачів та у використанні системних ресурсів.

Науково-практичні результати дослідження, проведеного в роботі були представлені та обговорені на VII Всеукраїнській науково-технічній конференції "Комп'ютерні технології: інновації, проблеми, рішення" та XV Міжнародній науково-технічній конференції «Інформаційно-комп'ютерні технології», що підтверджується публікацією тез доповідей:

1. Фомін В.В., Покотило О.А. Аналіз технологій VPN для використання у корпоративному середовищі // Комп'ютерні технології: інновації, проблеми, рішення: тези доп. VII Всеукр. наук.-техн. конф. (м. Житомир, 2025). Житомир, 2025. С. 117. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/01/povnyj-tekst-1.pdf>.

2. Фомін В.В., Покотило О.А. Аналіз технологій віддаленого доступу для використання у корпоративному середовищі // Інформаційно-комп'ютерні технології: тези доп. XV Міжнар. наук.-техн. конф. (м. Житомир, 2025). Житомир, 2025. С. 83.
URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/povnyj-tekst.pdf>.

Ключові слова: КОРПОРАТИВНА МЕРЕЖА, ВІДДАЛЕНИЙ ДОСТУП, WIREGUARD, RDP, VNC, DOCKER, WAZUH, SIEM, MIKROTIK.

ANNOTATION

The explanatory note to the qualification work on the topic “Methods and technologies for secure remote access for corporate network users” consists of an introduction, three chapters, a conclusion, and a list of references.

The text part is presented on 111 pages of printed text. The explanatory note has 95 pages and an appendix of 12 pages. The list of references contains 17 titles and occupies 2 pages. The work contains 89 figures and 4 tables. The total volume of the work is 111 pages.

The purpose of the final qualification work is to study methods of secure remote access and develop a secure corporate network infrastructure.

The work analyzes modern VPN technologies (WireGuard, OpenVPN) and remote access protocols (RDP, VNC). A network architecture was designed using WireGuard tunneling on MikroTik routers. A hybrid access system was implemented: RDP for Windows servers and VNC session-based and in isolated Docker containers for Linux. A security information and event management (SIEM) system based on Wazuh has been implemented. An experimental study has been conducted, confirming the effectiveness of protection against attacks (brute-force, port scanning) and the advantages of containerization for ensuring user safety and system resource utilization.

The scientific and practical results of the research conducted in this work were presented and discussed at the VII All-Ukrainian Scientific and Technical Conference “Computer Technologies: Innovations, Problems, Solutions” and the XV International Scientific and Technical Conference “Information and Computer Technologies,” as confirmed by the publication of the abstracts:

1. Fomin V.V., Pokotylo O.A. Analysis of VPN technologies for use in a corporate environment // Computer technologies: innovations, problems, solutions: abstracts of reports. VII All-Ukrainian Scientific and Technical Conference (Zhytomyr, 2025). Zhytomyr, 2025. P. 117. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/01/povnyj-tekst-1.pdf>.

2. Fomin V.V., Pokotylo O.A. Analysis of remote access technologies for use in a corporate environment // Information and computer technologies: abstracts of the XV

International Scientific and Technical Conference (Zhytomyr, 2025). Zhytomyr, 2025.
P. 83. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/povnyj-tekst.pdf>
(accessed: 25.10.2025).

Keywords: CORPORATE NETWORK, REMOTE ACCESS, WIREGUARD,
RDP, VNC, DOCKER, WAZUH, SIEM, MIKROTIK.

ЗМІСТ

ВСТУП.....	10
РОЗДІЛ 1. РОЗГЛЯД ОСОБЛИВОСТЕЙ КОРПОРАТИВНОЇ МЕРЕЖІ ТА ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ VPN ТА ВІДДАЛЕНОГО ДОСТУПУ У КОРПОРАТИВНОМУ СЕРЕДОВИЩІ.....	11
1.1. Захист інформації в корпоративних мережах.....	11
1.2. Віддалений доступ у корпоративному середовищі	14
1.3. Загальні відомості про VPN.....	19
1.4. Wireguard VPN. Особливості та використання.....	23
1.5. Методи захисту VPN та віддаленого доступу	27
РОЗДІЛ 2. ПРОЄКТУВАННЯ КОРПОРАТИВНОЇ МЕРЕЖІ, ВИБРАНОЇ VPN-ТЕХНОЛОГІЇ ТА МЕТОДІВ ВІДДАЛЕНОГО ДОСТУПУ	31
2.1. Визначення вимог до мережі та віддаленого доступу	31
2.2. Аналіз та обґрунтування вибору VPN-технології.....	31
2.3. Аналіз та обґрунтування вибору технологій віддаленого доступу.....	32
2.4. Архітектура корпоративної мережі	34
2.5. Проєктування VPN-рішення на базі WireGuard	36
2.6. Проєктування рішення віддаленого доступу	37
2.7. Додаткові заходи безпеки.....	38
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ЗПРОЄКТОВАНИХ РІШЕНЬ НА ТЕСТОВОМУ СТЕНДІ	40
3.1. Підготовка середовища емуляції.....	40
3.2. Налаштування VPN-з'єднання на маршрутизаторах MikroTik	42
3.4. Реалізація віддаленого доступу через VNC.....	58
3.5. Організація ізольованих VNC-середовищ у Docker	64
3.6. Реалізація системи моніторингу Wazuh.....	67
3.7. Дослідження роботи тестового стенду	74
3.8. Підсумки дослідження роботи тестового стенду	93
ВИСНОВКИ.....	96
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	97
ДОДАТКИ.....	99

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- AD DS – Служби доменів Active Directory (Active Directory Domain Services)
- CPU – Центральний процесор (Central Processing Unit)
- DNS – Система доменних імен (Domain Name System)
- GNS3 – Графічний мережевий симулятор-3 (Graphical Network Simulator-3)
- GUI – Графічний інтерфейс користувача (Graphical User Interface)
- ID – Ідентифікатор (Identifier)
- IP – Інтернет-протокол (Internet Protocol)
- IPv4/IPv6 – Інтернет-протокол версії 4 / версії 6
- LAN – Локальна мережа (Local Area Network)
- NAT – Трансляція мережевих адрес (Network Address Translation)
- OS – Операційна система (Operating System)
- PPA – Персональний архів пакунків (Personal Package Archive)
- RAM – Оперативна пам'ять (Random Access Memory)
- RDP – Протокол віддаленого робочого столу (Remote Desktop Protocol)
- RDS – Служби віддаленого робочого столу (Remote Desktop Services)
- SIEM – Управління інформаційною безпекою та подіями безпеки (Security Information and Event Management)
- SSH – Захищена оболонка (Secure Shell)
- TCP – Протокол керування передачею (Transmission Control Protocol)
- UDP – Протокол дейтаграм користувача (User Datagram Protocol)
- VM – Віртуальна машина (Virtual Machine)
- VNC – Віртуальні мережеві обчислення (Virtual Network Computing)
- VPN – Віртуальна приватна мережа (Virtual Private Network)
- WAN – Глобальна мережа (Wide Area Network)
- XML – Розширювана мова розмітки (Extensible Markup Language)

ВСТУП

Актуальність теми. В умовах масового переходу на дистанційну роботу та зростання кіберзагроз, забезпечення захищеного доступу до корпоративних ресурсів є критичним завданням. Класичні методи (прямий RDP, застарілі VPN) не завжди відповідають сучасним вимогам безпеки та продуктивності. Тому розробка комплексного рішення, що поєднує швидкісні VPN-тунелі, ізольовані середовища та активний моніторинг, є актуальною задачею.

Метою кваліфікаційної роботи є підвищення рівня захищеності та ефективності віддаленого доступу в корпоративних мережах шляхом розробки та впровадження комбінованої системи на базі технологій WireGuard, Docker та Wazuh.

Об'єктом дослідження є процеси організації та захисту віддаленого доступу користувачів до ресурсів корпоративної мережі.

Предметом дослідження є методи та технології побудови захищеної інфраструктури доступу з використанням VPN-тунелювання, контейнеризації сервісів та систем моніторингу подій безпеки.

Визначена мета обумовлює наступні завдання:

1. Проаналізувати сучасні технології VPN та протоколи віддаленого доступу, обґрунтувати вибір WireGuard, RDP та VNC.
2. Спроекувати архітектуру захищеної мережі з використанням маршрутизаторів MikroTik та ізольованих середовищ користувачів.
3. Реалізувати та налаштувати тестовий стенд.
4. Провести експериментальне дослідження безпеки (стійкість до атак) та продуктивності (споживання ресурсів) розробленого рішення.

У процесі виконання кваліфікаційної роботи було розроблено проєкт захищеної корпоративної мережі, що інтегрує високошвидкісний VPN WireGuard, реалізацію рішень віддаленого доступу RDP, VNC та Docker та систему виявлення вторгнень Wazuh, ефективність якого підтверджено практичними тестами.

РОЗДІЛ 1. РОЗГЛЯД ОСОБЛИВОСТЕЙ КОРПОРАТИВНОЇ МЕРЕЖІ ТА ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ VPN ТА ВІДДАЛЕНОГО ДОСТУПУ У КОРПОРАТИВНОМУ СЕРЕДОВИЩІ

1.1. Захист інформації в корпоративних мережах

Корпоративна мережа – це основа будь якого великого підприємства чи організації, її завдання – забезпечення з'єднання пристроїв, користувачів та корпоративних ресурсів на великих відстанях. В даному типі мереж приділяється особлива увага до безпеки, забезпечення масштабованості, надійності та ефективності, адже саме ця мережа повинна забезпечувати безперебійний та завжди доступний спосіб комунікації між користувачем та ресурсами корпорації.

На відміну від локальної мережі, яка призначена для підключення пристроїв у межах одного приміщення або будівлі, корпоративна мережа охоплює географічно розподілені об'єкти та інтегрує різні технологічні рівні, забезпечуючи централізовану взаємодію і захист даних. Ця масштабна мережа може з'єднувати окремі офіси та філіали, що знаходяться у різних містах чи навіть континентах, охоплюючи всі внутрішні мережі і пристрої.

Корпоративна мережа виступає в ролі глобальної мережі (WAN), що з'єднує локальні мережі офісів чи філіалів даної корпорації з головним офісом та між собою. Приклад того як може виглядати корпоративна мережа наведено на рисунку 1.1.

Забезпечення безпеки корпоративної мережі є критично важливим завданням для кожної компанії. Втрата даних або успішна кібератака може коштувати мільйони доларів і завдати серйозного удару по репутації. Для запобігання загрозам необхідно враховувати основні ризики, впроваджувати відповідні засоби захисту та підвищувати обізнаність співробітників про базові принципи кібербезпеки.

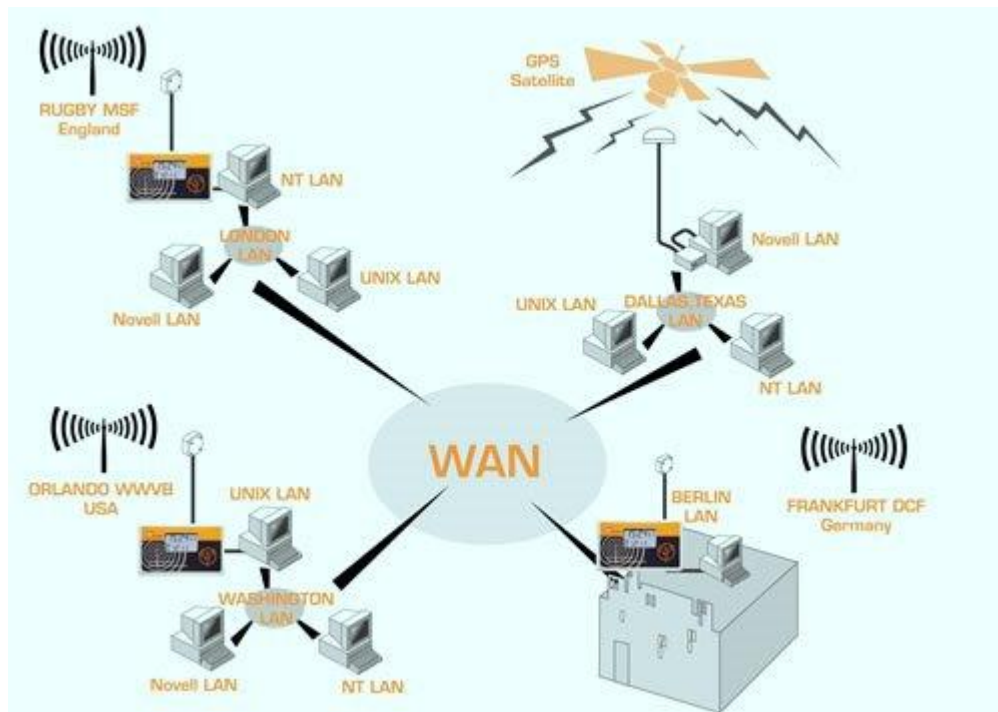


Рисунок 1.1 – Приклад вигляду корпоративної мережі.

Корпоративна мережа може стати ціллю для багатьох видів атак, серед яких виділяються [1]:

1. Атаки грубою силою (brute force). Підбір паролів до облікових записів за допомогою автоматизованих інструментів.
2. DDoS-атаки. Ці атаки перевантажують мережу або сервер великою кількістю запитів, що призводить до збоїв у роботі. Для їх запобігання необхідно впроваджувати спеціальні інструменти для виявлення і фільтрації шкідливого трафіку.
3. Шкідливе програмне забезпечення (malware). Віруси, трояни, програми-вимагачі та інші типи шкідливих програм можуть порушити роботу мережі або отримати доступ до конфіденційних даних. Для захисту потрібна комплексна стратегія, що включає антивірусні рішення, оновлення систем і регулярний моніторинг.
4. Фішингові атаки. Хакери обманом змушують користувачів виконувати небезпечні дії, наприклад, натискати на посилання чи відкривати заражені файли. Запобігти цьому допоможе навчання співробітників, використання фільтрів електронної пошти та встановлення брандмауерів.

Основні заходи безпеки у корпоративних мережах передбачають [1]:

1. Впровадження захисних систем. Використання брандмауерів, систем виявлення та запобігання вторгненням (IDS/IPS), фільтрів веб-контенту та контролю доступу до мережі (NAC) є базовими заходами. Ці інструменти допомагають виявляти загрози в реальному часі, обмежувати доступ і захищати важливі ресурси.

2. Сегментація мережі. Розподіл мережі на зони з окремими функціями дозволяє зменшити можливу шкоду від атаки. Хакеру доведеться витратити більше часу та ресурсів для проникнення в кожную зону. До того ж, сегментація дозволяє створити різні рівні безпеки для кожної категорії даних.

3. Використання VPN. Віртуальні приватні мережі створюють безпечний тунель для передачі даних. Шифрування даних і використання протоколів, таких як IPSec або L2TP, забезпечують додатковий рівень захисту. Проте VPN слід поєднувати з іншими методами для досягнення повноцінної безпеки.

4. Оптимізація аутентифікації та контролю доступу. Двофакторна аутентифікація, обмеження прав доступу, шифрування паролів і моніторинг дій користувачів мінімізують ризики як зовнішніх, так і внутрішніх загроз.

Ефективний захист корпоративної мережі потребує комплексного підходу, який охоплює як технічні, так і організаційні аспекти. Одним із ключових рішень для підвищення рівня безпеки є сегментація мережі. Розподіл інфраструктури на окремі зони дозволяє мінімізувати ризики та локалізувати потенційну шкоду в разі успішної атаки. Такий підхід також дає змогу встановлювати різні рівні безпеки для кожного сегмента, залежно від його функціонального призначення.

Ще одним важливим інструментом є використання VPN для створення безпечного каналу передачі даних між користувачами та корпоративною мережею. VPN забезпечує шифрування даних і захищає інформацію від перехоплення, що є особливо актуальним для віддаленого доступу. Однак варто зазначити, що VPN не є універсальним захистом. Для забезпечення повної

безпеки його необхідно поєднувати з іншими методами, такими як двофакторна аутентифікація, брандмауери та системи моніторингу мережі.

Поєднання сегментації мережі, впровадження VPN і додаткових засобів захисту створює багаторівневу систему безпеки, яка ефективно протидіє сучасним кіберзагрозам та забезпечує надійний захист даних компанії.

1.2. Віддалений доступ у корпоративному середовищі

Віддалений доступ дозволяє користувачеві підключатися до іншого комп'ютера або мережі, які знаходяться у віддаленому місці. Завдяки віддаленому доступу можна моніторити, обслуговувати та керувати пристроями, якщо вони підключені до однієї мережі. Це дозволяє усувати несправності без фізичної присутності біля пристрою, який має проблему.

Також, віддалений доступ дозволяє отримувати потрібні файли без пересилання їх електронною поштою чи іншими засобами. Ви можете визначати права доступу до файлів, а також організовувати користувачів у різні категорії, надаючи доступ одній групі та обмежуючи іншу.

Доступ до віддаленого робочого столу описує програмне забезпечення, яке дозволяє одному користувачеві взаємодіяти з робочим столом іншого комп'ютера. У процесі взаємодії віддалений користувач може бачити робочий стіл іншого пристрою на своєму екрані.

З'єднання через віддалений доступ дозволяє користувачам підключатися до приватної мережі з іншого місця. Для цього обидва користувачі мають бути підключені до тієї ж мережі.

Після встановлення підключення спеціальний протокол, керований інтерфейсами програмного забезпечення, дозволяє одному пристрою взаємодіяти з іншим. Цей протокол забезпечує можливість використовувати клавіатуру, трекпад, сенсорний екран чи мишу для управління віддаленим пристроєм.

Розуміння сильних сторін і особливостей протоколів віддаленого доступу є важливим для створення захищених та ефективних умов дистанційної роботи. Нижче описані найпоширеніші протоколи [2]:

1. Point-to-Point Protocol (PPP). Протокол, який використовується для встановлення прямого з'єднання між двома комп'ютерами. PPP широко застосовується у випадках, коли потрібна висока швидкість і надійна передача даних, наприклад у широкосмугових мережах.

2. Internet Protocol Security (IPsec). IPsec забезпечує конфіденційність і цілісність даних, шифруючи інформацію, що передається між комп'ютерами через IP-мережу.

3. Point-to-Point Tunneling Protocol (PPTP). Колись популярний протокол для створення VPN-з'єднань, PPTP втратив актуальність через виявлені вразливості. Рекомендується використовувати сучасніші та більш захищені альтернативи.

4. Remote Access Services (RAS). Сервіс, який дозволяє користувачам безпечно підключатися до корпоративної мережі з віддалених місць через комутовані з'єднання або VPN.

5. Remote Desktop Protocol (RDP). Власний протокол Microsoft, який надає користувачам графічний інтерфейс для роботи з робочим столом іншого комп'ютера через мережу.

6. Virtual Private Network (VPN). VPN створює зашифровані тунелі через публічні мережі, такі як інтернет, дозволяючи віддаленим користувачам безпечно підключатися до приватних мереж і ресурсів.

7. Virtual Network Computing (VNC). VNC забезпечує графічний інтерфейс для доступу до віддаленого робочого столу, дозволяючи користувачам переглядати та контролювати інший комп'ютер з будь-якого місця.

8. Пропріетарні протоколи. Деякі рішення для віддаленого доступу використовують унікальні протоколи, адаптовані до їх платформ. Такі протоколи включають сучасні заходи безпеки, наприклад AES-256 шифрування та обмін ключами SSL/TLS, що гарантує захищеність сеансів.

Три найпоширеніші підходи до віддаленого доступу – це інфраструктури віртуальних робочих столів (VDI), віртуальні приватні мережі (VPN) та протокол віддаленого робочого столу (RDP) [3].

Кожен із цих методів має свої особливі виклики у сфері безпеки та вимагає відповідних заходів захисту.

1. VPN: Найкраще підходить для організацій, які цінують прямий контроль над локальним обладнанням і потребують доступу до локальних ресурсів, або для індивідуальних користувачів, які цінують конфіденційність і безпеку.

2. VDI: Найкраще підходить для стандартизованого та масштабованого віддаленого доступу.

3. RDP: Більше не вважається безпечним чи надійним методом підключення.

На рисунку 1.2. наведено порівняння VPN, VDI та RDP.

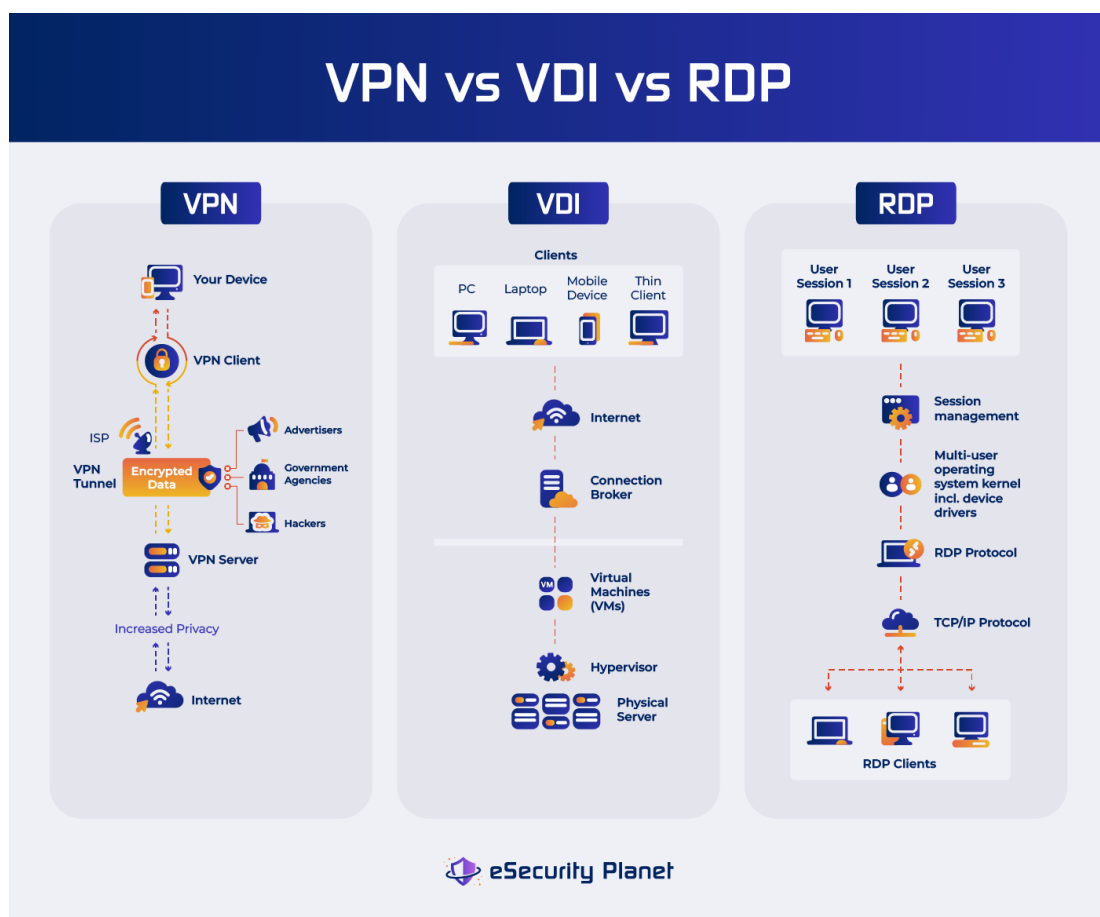


Рисунок 1.2 – Порівняння та принципи роботи VPN, VDI та RDP

VDI дозволяє підприємствам розробляти та керувати віртуалізованими середовищами робочих столів у рамках централізованої серверної архітектури або в хмарі. VDI надає користувачам можливість віддалено працювати зі своїми персональними робочими столами з різних пристроїв, включаючи ноутбуки, планшети та тонкі клієнти, тоді як основні обчислення та зберігання даних виконуються на сервері. Цей метод забезпечує гнучкий і безпечний досвід роботи з робочим столом, не обмежуючи користувачів одним фізичним пристроєм чи місцем. VDI ідеально підходить для організацій, які цінують адаптивність та стандартизоване середовище для віддаленої роботи.

Віртуальні робочі столи забезпечують покращені функції безпеки. Наприклад, зламати віртуальний робочий стіл не означає доступ до брандмауера чи мережі, а також не залишає фізичне обладнання, яке хакери можуть атакувати. Однак VDI зазвичай управляється сторонніми постачальниками, оскільки самостійне впровадження є дорогим і складним.

Налаштування VDI включає кілька ключових компонентів:

1. Сервер: Потужний сервер містить кілька віртуальних машин (VM), кожна з яких представляє індивідуальний робочий стіл користувача. Ці віртуальні машини ізольовані одна від одної для забезпечення конфіденційності та безпеки даних.

2. Гіпервізор: Це програмний рівень, який управляє створенням, розподілом і роботою віртуальних машин на сервері. Він гарантує оптимальне використання ресурсів і продуктивність кожної VM.

3. Віртуальний робочий стіл: Кожному користувачу надається віртуальний робочий стіл, що імітує стандартне середовище фізичного ПК із включенням операційної системи, додатків, налаштувань і даних.

4. Клієнтські пристрої: Користувачі отримують доступ до своїх робочих столів через віддалені протоколи доступу, використовуючи ноутбуки, мобільні пристрої тощо. Ці протоколи передають зміни екрана, натискання клавіш і рухи миші між клієнтом та віртуальним робочим столом.

5. Централізоване управління: ІТ-менеджери можуть швидко впроваджувати нові робочі станції, оновлювати програмне забезпечення та забезпечувати дотримання заходів безпеки.

RDP – це пропрієтарний протокол, розроблений компанією Microsoft, який дозволяє користувачам віддалено отримувати доступ і керувати комп'ютером або сервером з іншого місця, ніби вони фізично знаходяться там. RDP забезпечує плавний і звичний досвід користування, надаючи доступ до програм, файлів і ресурсів віддаленої системи.

Для організацій, які прагнуть прямого контролю над віддаленими системами, RDP є недорогим варіантом, але зараз він вважається небезпечним.

RDP базується на моделі клієнт-сервер:

1. Клієнт: Пристрій користувача, який ініціює підключення до віддаленої системи через RDP-клієнт. Клієнт передає дані про натискання клавіш, рухи миші та зміни екрана на сервер;

2. Сервер: Віддалений комп'ютер обробляє дії клієнта, оновлює екран і забезпечує користувачеві інтегрований досвід;

3. Інтерактивність користувача: Користувач може виконувати дії, працювати з файлами та додатками, ніби він фізично знаходиться за пристроєм.

Аналогом RDP у системах на базі Linux виступає VNC. Хоча SSH забезпечує доступ до терміналу, для графічного інтерфейсу робочого столу Ubuntu потрібні альтернативні протоколи, такі як VNC (віртуальні мережеві обчислення).

VNC передає повноцінне візуальне середовище робочого столу через віддалену сесію — це схоже на підключення фізичного монітора, миші та клавіатури. Цей підхід корисний для завдань, які потребують графічного інтерфейсу, таких як:

- запуск програм з графічним інтерфейсом;
- перегляд мультимедійного контенту;
- налаштування параметрів системи;
- надання підтримки користувачам;

- адміністративне управління мережею.

VNC дозволяє переглядати мультимедіа, користуватися додатками з графічним інтерфейсом та виконувати інші дії з робочим столом віддалено через Ubuntu. Для підвищення безпеки рекомендується поєднувати VNC із SSH-тунелями, щоб шифрувати трафік віддаленого робочого столу [4].

1.3. Загальні відомості про VPN

Віртуальна приватна мережа (VPN) є важливим інструментом для забезпечення безпеки інтернет-з'єднання та захисту конфіденційності. Для корпоративного середовища VPN виступає невід'ємною складовою кіберзахисту, особливо для співробітників, які працюють віддалено, часто подорожують або підключаються до Інтернету через публічні Wi-Fi-мережі.

На відміну від індивідуальних VPN-сервісів, корпоративні VPN пропонують розширені функції, включаючи посилену безпеку та авторизацію користувачів на основі ролей. Це дозволяє гнучко налаштовувати доступ до корпоративних даних, забезпечуючи безпеку навіть у великих і розподілених організаціях.

Однією з ключових функцій VPN є створення тунелю «точка-точка», який шифрує дані користувачів і приховує їх IP-адресу [5]. Завдяки цьому дії користувачів залишаються анонімними, а їх дані – захищеними навіть при використанні незахищених з'єднань. Крім того, VPN може створювати ілюзію перебування пристрою в іншій країні, що додає рівень захисту від географічно орієнтованих атак.

Для сучасних компаній VPN стає критично важливим рішенням у забезпеченні безпечного доступу співробітників до корпоративних даних із дому чи інших віддалених локацій. Однак, окрім підтримки віддалених співробітників, корпоративний VPN також захищає працівників, які знаходяться у філіях або регіональних офісах, запобігаючи можливим витокам даних та атакам.

Таким чином, корпоративний VPN не тільки захищає дані компанії від зовнішніх загроз, але й створює умови для безпечної та гнучкої роботи, що є особливо актуальним у сучасному світі зростаючої цифрової мобільності.

VPN-клієнти базуються на поширених протоколах. Ці протоколи застосовують стандарти шифрування до даних, що передаються через VPN-тунелі, і вони представлені в різних формах.

Слід уникати застарілих VPN-протоколів, таких як PPTP або L2TP/IPSec, через їхні обмеження у швидкості та безпеці. В свою чергу OpenVPN, SSTP, WireGuard і NordLynx мають свої переваги [6]:

1. OpenVPN – це надійний і швидкий протокол тунелювання, який забезпечує максимальну гнучкість завдяки своїй архітектурі з відкритим кодом.
2. SSTP (Secure Socket Tunneling Protocol) працює, захищаючи мережі на базі операційних систем Windows.
3. WireGuard – один із новітніх VPN-протоколів, який використовується передовими провайдерами, такими як NordLayer, для забезпечення безпеки їхніх сервісів. Цей протокол швидкий, безпечний і простий у використанні, що робить його новим стандартом.
4. NordLynx – це новий протокол, розроблений компанією NordLayer, який пропонує покращену швидкість і безпеку порівняно з іншими VPN-протоколами. Він поєднує швидкість WireGuard із додатковими шарами шифрування, забезпечуючи підвищену конфіденційність і анонімність.

У даному дослідженні зупинимося на 2 протоколах, а саме – WireGuard та OpenVPN. OpenVPN і WireGuard – два найпопулярніші VPN-протоколи, які широко використовуються для захисту інтернет-з'єднань [7].

OpenVPN, що з'явився у 2001 році, є відкритим програмним забезпеченням, яке дозволяє створювати безпечне з'єднання між двома точками (point-to-point) або між сайтами (site-to-site) через інтернет.

WireGuard, випущений у 2016 році, є новим відкритим VPN-протоколом, створеним із метою бути швидшим, простішим і більш безпечним, ніж IPsec і OpenVPN.

Обидва протоколи чудово підходять для шифрування інтернет-трафіку та приховування активності користувачів в мережі. OpenVPN може бути трохи простішим у налаштуванні, тоді як WireGuard забезпечує значно вищу швидкість роботи завдяки своїй сучасній архітектурі.

WireGuard – це відносно новий VPN-протокол, який набув популярності за останні роки. Багато користувачів переходять з OpenVPN на WireGuard завдяки низці переваг, які він пропонує.

Однією з основних переваг WireGuard є його швидкість і невелика ресурсозатратність. Він використовує сучасну криптографію, яка споживає менше ресурсів процесора, забезпечуючи вищу швидкість з'єднання. Це робить WireGuard ідеальним вибором для мобільних пристроїв із обмеженими обчислювальними потужностями. На відміну від цього, OpenVPN використовує застарілі криптографічні алгоритми, які потребують більше ресурсів процесора і працюють повільніше, особливо на мобільних пристроях.

Ще однією перевагою є простота налаштування та розгортання. Конфігурація OpenVPN може бути складною через велику кількість параметрів, які можуть заплутати користувача. WireGuard, навпаки, має мінімалістичну конфігурацію з кількома ключовими параметрами. Завдяки цій простоті VPN на основі WireGuard є більш надійними та менш схильними до технічних проблем.

Однією з основних цілей при створенні WireGuard було забезпечити його легку аудиторність, настільки, щоб один інженер міг виконати цю задачу за розумний проміжок часу. У порівнянні з довгим процесом і цілими командами, необхідними для аудиту протоколу, як OpenVPN, це була амбіційна мета.

WireGuard досягає цього завдяки надзвичайно малому обсягу — всього 4000 рядків коду [8]. Для порівняння, в OpenVPN лише відкритий код містить близько 70 000 рядків. Таким чином, WireGuard пропонує ефективне, просте у використанні та безпечне рішення для захисту мереж, особливо на мобільних і ресурсозатратних пристроях.

Протокол OpenVPN вважається стандартом за замовчуванням у сфері мереж з відкритим кодом [9]. Він підтримує всі основні операційні системи,

включаючи Windows, macOS, Linux, Android та iOS, а також менш поширені платформи, такі як FreeBSD, QNX, Solaris, Maemo, Windows Mobile і ChromeOS.

OpenVPN забезпечує високий рівень безпеки та є надзвичайно адаптивним для інтеграції зі стороннім програмним забезпеченням. Він використовує 256-бітне шифрування (кількість бітів може змінюватися за потреби) за допомогою бібліотеки OpenSSL, яка широко застосовується для захисту з'єднань у різних мережах.

OpenVPN підтримує два протоколи передачі даних: TCP та UDP. UDP є більш поширеним і рекомендованим варіантом завдяки своїй швидкості та ефективності. У випадках, коли використання UDP неможливе через мережеві обмеження, OpenVPN може переключитися на TCP. Однак TCP варто використовувати лише за крайньої необхідності, оскільки це може призвести до зниження продуктивності та появи так званого явища "TCP meltdown". OpenVPN використовує або UDP, або TCP для встановлення з'єднання між пристроями та серверами. Обидва протоколи досягають схожих цілей, але різними способами. Ось основні відмінності:

- TCP забезпечує більш безпечну та стабільну передачу даних завдяки функціям корекції помилок;
- UDP не забезпечує стабільної передачі даних і функцій корекції помилок, але працює швидше.

Однак більшість постачальників VPN за замовчуванням використовують OpenVPN через UDP [10].

Проект OpenVPN з відкритим кодом, також відомий як Community Edition (CE), створює захищені VPN-з'єднання через інтернет за допомогою власного протоколу безпеки, який базується на технологіях SSL/TLS [11]. Протокол підтримує розширення мереж на рівнях OSI 2 або 3 за допомогою SSL/TLS; гнучкі методи автентифікації клієнтів, зокрема сертифікати, смарт-карти та двофакторну автентифікацію; контроль доступу для окремих користувачів чи груп через політики фаєрволу, які застосовуються до віртуального інтерфейсу VPN [11].

Оскільки CE є проектом з відкритим кодом, він повністю безкоштовний для впровадження. Проте для його налаштування потрібні глибокі знання Linux та вміння працювати з інтерфейсом командного рядка.

1.4. Wireguard VPN. Особливості та використання

WireGuard – це сучасний протокол тунелювання VPN, який використовує UDP для передачі даних. Він використовується для віртуальних приватних мереж і передачі звичайного трафіку, адже зазвичай він є дуже швидким [12].

WireGuard має на меті бути таким же простим у налаштуванні та розгортанні, як і SSH. Підключення VPN здійснюється шляхом обміну простими відкритими ключами – аналогічно до обміну ключами SSH – і решта процесів прозоро обробляється WireGuard. Протокол навіть здатний перемикається між IP-адресами, подібно до Mosh. Немає потреби керувати з'єднаннями, турбуватися про стан, запускати демонів чи розбиратися у складних технічних деталях. WireGuard пропонує надзвичайно простий, але потужний інтерфейс.

WireGuard використовує передову криптографію, зокрема Noise Protocol Framework, Curve25519, ChaCha20, Poly1305, BLAKE2, SipHash24, HKDF, а також перевірені та надійні конструкції [13]. Протокол робить консервативний та обґрунтований вибір криптографічних алгоритмів, які були перевірені криптографами.

WireGuard створено з акцентом на простоту реалізації та використання. Він розроблений так, щоб його можна було легко реалізувати у кількох рядках коду, що спрощує аудит безпеки. На відміну від громіздких рішень, таких як Swan/IPsec чи OpenVPN/OpenSSL, аудит яких є складним завданням навіть для великих команд експертів з безпеки, WireGuard створений так, щоб бути повністю зрозумілим і перевіреним навіть однією особою.

Налаштування WireGuard на нашій стороні складається з наступних основних компонентів [14]:

1. Інстанси (Instances): У конфігурації WireGuard вони називаються «інтерфейсами» і описують, як налаштований віртуальний пристрій wgX на нашій стороні з точки зору адресації та криптографії.

2. Піри (Peers): Це клієнти, яким дозволено підключатися до нас. Вони описуються їхньою опціональною віддаленою адресою, включаючи мережі, яким дозволено передавати дані через тунель. Піри належать до одного або декількох інстансів.

WireGuard додає мережевий інтерфейс (або кілька), подібний до eth0 чи wlan0, з іменем, наприклад, wg0 (або wg1, wg2, wg3 тощо). Цей інтерфейс можна налаштовувати стандартними інструментами, такими як ifconfig(8) або ip-address(8), додавати та видаляти маршрути за допомогою route(8) чи ip-route(8), а також використовувати інші мережеві утиліти. Специфічні функції WireGuard налаштовуються за допомогою інструмента wg(8). Цей інтерфейс виконує роль тунельного інтерфейсу.

Коли інтерфейс відправляє пакет до вузла, відбувається наступне:

1. Визначення вузла: «Цей пакет призначений для 192.168.30.8. Хто це? Перевіримо... Це вузол ABCDEFGH. (Якщо жоден вузол не налаштований для цього пакета, його відхиляють.)»

2. Шифрування: «Зашифруємо весь IP-пакет, використовуючи публічний ключ вузла ABCDEFGH.»

3. Пошук кінцевої точки: «Яка кінцева точка вузла ABCDEFGH? Перевіримо... Це порт UDP 53133 на хості 216.58.211.110.»

4. Відправка: «Відправляємо зашифровані байти з кроку 2 через Інтернет до 216.58.211.110:53133 за допомогою UDP.»

Коли інтерфейс отримує пакет, виконується наступне:

1. Отримання пакета: «Я отримав пакет із порту UDP 7361 на хості 98.139.183.24. Давайте його дешифруємо!»

2. Дешифрування: «Пакет успішно дешифровано та автентифіковано для вузла LMNOPQRS. Запам'ятаємо, що остання точка підключення цього вузла – це 98.139.183.24:7361 через UDP.»

3. Перевірка адреси: «Розшифрований пакет надійшов від 192.168.43.89. Чи дозволено вузлу LMNOPQRS надсилати нам пакети з цієї адреси?»

4. Дія: Якщо так – приймаємо пакет на інтерфейсі. Якщо ні – відхиляємо.

У центрі роботи WireGuard лежить концепція Cryptokey Routing, яка асоціює публічні ключі з переліком дозволених IP-адрес тунелю [13]. Кожен мережевий інтерфейс має приватний ключ та список вузлів (peers), кожен із яких має свій публічний ключ. Публічні ключі є короткими й простими, використовуються для аутентифікації між вузлами та можуть передаватися будь-якими зручними способами, аналогічно до обміну SSH-ключами.

У контексті відправлення пакетів список дозволених IP-адрес працює як таблиця маршрутизації, а при отриманні пакетів — як список контролю доступу (ACL).

У серверній конфігурації, коли мережевий інтерфейс хоче надіслати пакет до пірінга (клієнта), він перевіряє IP-адресу призначення пакету та порівнює її зі списком дозволених IP-адрес для кожного пірінга, щоб визначити, кому його надіслати.

Наприклад, якщо мережевий інтерфейс отримує запит на відправку пакету з IP-адресою 10.10.10.230, він зашифрує його, використовуючи публічний ключ пірінга gN65BkIK..., а потім відправить його на останній відомий Інтернет-адрес пірінга. Після отримання пакету сервер здійснює перевірку, і якщо IP-адреса джерела відповідає дозволеному списку для пірінга, пакет передається далі; якщо ж IP не відповідає, він відкидається.

Приклад конфігурації сервера WireGuard:

[Interface]

PrivateKey = yAnz5TF+lXXJte14tji3zlMNq+hd2rYUIgJBgB3fBmk=

ListenPort = 51820

[Peer]

PublicKey = xTIBA5rboUvnH4htodjb6e697QjLERt1NAB4mZqp8Dg=
AllowedIPs = 10.192.122.3/32, 10.192.124.1/24

[Peer]

PublicKey = TrMvSoP4jYQlY6RIzBgbssQqY3vxI2Pi+y71lOWWXX0=
AllowedIPs = 10.192.122.4/32, 192.168.0.0/16

[Peer]

PublicKey = gN65BkIKyl eCE9pPlwdc8ROUtkHLF2PfAqYdyYBz6EA=
AllowedIPs = 10.10.10.230/32

У клієнтській конфігурації є тільки один пірінг (сервер), який може відправляти пакети до мережевого інтерфейсу з будь-якою IP-адресою джерела, оскільки в списку дозволених IP вказано 0.0.0.0/0 (це wildcard-адреса). Коли пакет отримується від пірінга, система розшифровує його і перевіряє підпис; якщо перевірка пройшла успішно, пакет приймається, незалежно від IP-адреси джерела. Клієнт при відправці пакету до сервера зашифровує його публічним ключем сервера і надсилає на останній відомий Інтернет-адрес сервера.

Приклад конфігурації клієнта WireGuard:

[Interface]

PrivateKey = gI6EdUSYvn8ugXOt8QQD6Yc+JyiZxIhp3GInSWRfWGE=
ListenPort = 21841

[Peer]

PublicKey = HIgo9xNzJMWLKASShiTqIybxZ0U3wGLiUeJlPKf8ykw=
Endpoint = 192.95.5.69:51820
AllowedIPs = 0.0.0.0/0

Конфігурація клієнта містить початкову точку доступу для єдиного пірінга (сервера), щоб клієнт знав, куди відправляти зашифровані дані до того, як отримає їх від сервера. Конфігурація сервера не містить початкових точок

доступу для своїх пірінгів (клієнтів), оскільки сервер визначає точку доступу пірінгів, аналізуючи, з якого джерела надійшли правильно автентифіковані дані. Якщо сервер змінює свою точку доступу та надсилає дані клієнтам, ці клієнти виявляють нову точку доступу сервера та оновлюють конфігурацію відповідно. Як клієнт, так і сервер відправляють зашифровані дані на останню відому IP-адресу, для якої успішно розшифровано дані. Таким чином, роумінг IP працює на обох кінцях.

WireGuard відправляє та отримує зашифровані пакети, використовуючи мережевий простір імен, в якому був спочатку створений інтерфейс WireGuard. Це означає, що можна створити інтерфейс WireGuard у основному мережевому просторі імен, який має доступ до Інтернету, а потім перемістити його в мережевий простір імен, що належить контейнеру Docker, як єдиний інтерфейс цього контейнера. Завдяки цьому, єдиним способом, яким контейнер може отримати доступ до мережі, це через безпечний зашифрований тунель WireGuard.

1.5. Методи захисту VPN та віддаленого доступу

Для забезпечення безпеки середовища, відповідності вимогам і операційної ефективності організація повинна ретельно контролювати сесії віддаленого доступу на предмет підозрілої активності, підтримувати оновлення пристроїв і додатків, а також відстежувати стан своїх систем.

Більшість мереж мають базові інструменти для моніторингу та керування віддаленим доступом. Наприклад, Windows Server включає Event Viewer для перегляду системних і прикладних журналів, а також сервіс Windows Remote Management (WinRM), який дозволяє адміністраторам керувати системами Windows за допомогою команд PowerShell віддалено. Проте ці інструменти потребують значних часових і трудових витрат з боку персоналу безпеки.

Щоб знизити операційні витрати та ефективно підтримувати віддалену роботу, багато організацій інвестують у сторонні рішення, які автоматизують

рутинні завдання. Популярні пропозиції, що підтримують середовища Windows, Mac та Linux, включають [15]:

1. Netwrix Auditor – допомагає забезпечити безпеку, відповідність вимогам і покращення операційної ефективності. Забезпечує повну видимість активності користувачів, виявляє підозрілу поведінку, надсилає сповіщення в реальному часі та автоматизує реагування на потенційні загрози. Також пропонує розширену звітність для відповідності вимогам і спрощення аудитів.

2. LogicMonitor – дає змогу командам проактивно виявляти та вирішувати проблеми, пов'язані з віддаленим доступом за допомогою таких функцій, як автоматизоване виявлення, контроль доступу, шифрування, сповіщення в реальному часі та налаштовувані інформаційні панелі.

3. SolarWinds – пропонує рішення для віддаленого моніторингу та керування, яке дозволяє командам ІТ виявляти нові пристрої в мережі та управляти ними централізовано. Можливості сповіщення та звітності в реальному часі забезпечують проактивний моніторинг стану пристроїв, продуктивності та безпеки.

Хоча віддалений доступ забезпечує численні переваги, важливо враховувати притаманні йому ризики, щоб захистити дані та підтримувати безпечне робоче середовище. Несанкціонований доступ, витік даних і зараження шкідливими програмами є потенційними ризиками, які організації повинні активно мінімізувати.

Поширені ризики [2]:

1. Несанкціонований доступ: Віддалений доступ збільшує поверхню атаки, тому необхідно реалізовувати надійні механізми автентифікації, такі як складні паролі, багатофакторна автентифікація (MFA) та контроль доступу, щоб запобігти проникненню несанкціонованих осіб.

2. Витік даних: Чутливі дані, які передаються під час віддалених сесій, можуть бути перехоплені або скомпрометовані, якщо вони недостатньо захищені. Протоколи шифрування, такі як VPN, та механізми захищеної передачі файлів є важливими для забезпечення конфіденційності даних.

3. Зараження шкідливими програмами: Віддалені пристрої можуть стати точками проникнення шкідливих програм, які поширюються мережею. Рішення для захисту кінцевих точок, такі як антивірусне програмне забезпечення та фаєрволи, є критично важливими для виявлення та запобігання зловмисним атакам.

Покращення безпеки та оптимізація переваг [2]:

1. Багатофакторна автентифікація (MFA): Додає додатковий рівень захисту, окрім паролів, вимагаючи від користувачів додаткової перевірки, наприклад, за допомогою відбитка пальця або одноразового коду, що суттєво знижує ризик несанкціонованого доступу.

2. Шифрування: Використання надійних протоколів шифрування, таких як VPN, гарантує, що дані, передані під час віддалених сесій, залишаються конфіденційними та захищеними.

3. Захист кінцевих пристроїв: Розгортання надійних рішень для захисту кінцевих точок, таких як антивірусне програмне забезпечення, фаєрволи та системи виявлення вторгнень, допомагає захистити віддалені пристрої від шкідливих програм та інших загроз.

4. Сегментація мережі: Ізоляція чутливих даних і критичних систем за допомогою сегментації мережі обмежує потенційний вплив порушення, запобігаючи боковому поширенню та локалізуючи загрози.

5. Регулярні аудити безпеки та оновлення: Проведення періодичних аудитів безпеки та підтримка програмного забезпечення й систем в актуальному стані допомагає виявляти вразливості та оперативно їх усувати, мінімізуючи ризик експлуатації.

6. Навчання співробітників: Освіта працівників про найкращі практики безпеки, такі як розпізнавання фішингових атак і використання надійних паролів, є ключовою для створення культури безпеки.

За умови використання захисту кінцевих точок, багатофакторної автентифікації, складних паролів і оновленого програмного забезпечення віддалений доступ може бути безпечним способом підключення двох пристроїв.

Проте важливо бути пильним щодо загроз, які можуть бути особливо небезпечними під час віддаленого підключення. Наприклад, троянський кінь може легко перенестися з одного пристрою на інший і залишитися непоміченим, оскільки він часто здається нешкідливим. Також важливо бути обережними з файлами, які передаються між пристроями, оскільки вони можуть містити віруси, шпигунські програми або інше шкідливе ПЗ.

Висновки до першого розділу: у першому розділі було проведено теоретичний аналіз особливостей корпоративних мереж та сучасних загроз їхній безпеці. Розглянуто ключові технології віддаленого доступу, зокрема VDI, RDP та VNC, а також детально проаналізовано принципи роботи та протоколи VPN. Особливу увагу приділено порівняльному аналізу протоколів OpenVPN та WireGuard.

РОЗДІЛ 2. ПРОЄКТУВАННЯ КОРПОРАТИВНОЇ МЕРЕЖІ, ВИБРАНОЇ VPN-ТЕХНОЛОГІЇ ТА МЕТОДІВ ВІДДАЛЕНОГО ДОСТУПУ

2.1. Визначення вимог до мережі та віддаленого доступу

Перед тим як збирати топологію, варто сформулювати кілька ключових вимог, на яких базується вся архітектура мережі.

Основна ідея – створити просту, але реалістичну корпоративну мережу, де є два логічні сегменти: центральний офіс і віддалений філіал. Вони мають взаємодіяти між собою через VPN, і користувачі з обох сторін повинні мати:

- безпечний доступ до ресурсів іншої підмережі, але без зайвих ручних налаштувань на клієнтських машинах;
- можливість виходу в Інтернет з обох сегментів;
- підтримку GUI-доступу через RDP (Windows) і VNC (Linux);
- простоту в розгортанні та масштабуванні, щоб у майбутньому можна було додати нові офіси або вузли без повної реконструкції мережі;
- з'єднання між користувачами відбуватиметься через Інтернет, без прямих підключень.

Враховуючи те, що мова йдеться про корпоративну мережу, потрібно приділити особливу увагу безпеці. Для цього: тунель має бути зашифрований, порти – закриті, доступ – лише через VPN. Тому VPN варто реалізовувати на рівні маршрутизаторів, що дозволить відразу ізолювати трафік і уникнути встановлення VPN на кожному окремому пристрої.

2.2. Аналіз та обґрунтування вибору VPN-технології

Проектування захищеної корпоративної мережі вимагає вибору ефективного та безпечного VPN-протоколу для з'єднання віддалених сегментів. У рамках даної роботи було проведено порівняльний аналіз між двома ключовими технологіями: OpenVPN та WireGuard.

Хоча OpenVPN є перевіреним стандартом з високою гнучкістю, його архітектура має суттєві недоліки у швидкодії та складності аудиту безпеки.

WireGuard, навпаки, спроектований з акцентом на простоту та високу продуктивність.

Для наочного порівняння ключових показників, розглянемо таблицю 2.1, складену на основі аналізу [16].

Таблиця 2.1

Порівняльна характеристика технологій VPN

	WireGuard	OpenVPN
Швидкодія (швидкість завантаження та вивантаження)	<i>Висока (280 Мбіт/с та 209 Мбіт/с)</i>	<i>Середня (по UDP 137 Мбіт/с, 143 Мбіт/с; по TCP 30 Мбіт/с, 35Мбіт/с)</i>
Шифрування	<i>Високе (ChaCha20)</i>	<i>Від низької до високої (підтримує AES, Blowfish, Camellia, ChaCha20)</i>
Безпека	<i>Висока</i>	<i>Висока</i>
Можливість аудиту відкритого вихідного коду	<i>Висока (близько 4 000 ліній коду)</i>	<i>Низька (близько 70 000 ліній коду)</i>
Сумісність	<i>Низька</i>	<i>Висока</i>

На основі даних, наведених у таблиці, для реалізації в даному проєкті було обрано WireGuard. Це рішення обґрунтоване пріоритетом високої продуктивності, що критично для комфортної роботи віддалених користувачів, та прозорістю безпеки завдяки мінімалістичній кодовій базі.

Незважаючи на нижчу сумісність порівняно з OpenVPN, для завдань проєкту його переваги у швидкості та простоті аудиту є більш вагомими.

2.3. Аналіз та обґрунтування вибору технологій віддаленого доступу

Проєкт передбачає організацію віддаленого доступу до гетерогенного середовища, що включає сервери на базі Windows та робочі станції Linux. Це зумовило необхідність аналізу та вибору оптимальних протоколів для кожної з платформ.

Було проведено аналіз двох основних протоколів: RDP (Remote Desktop Protocol) та VNC (Virtual Network Computing). RDP є нативним рішенням для Windows, оптимізованим для передачі графічних команд, що забезпечує високу продуктивність та вбудоване шифрування. VNC, у свою чергу, є

кросплатформеним стандартом, що передає безпосередньо зображення екрану, забезпечуючи універсальність, але поступаючись у продуктивності.

Порівняльна характеристика протоколів наведена у таблиці 2.2 [17].

Таблиця 2.2

Порівняльна характеристика технологій віддаленого доступу

	VNC	RDP
Призначення	<i>Віддалене керування комп'ютером через передачу зображення екрану</i>	<i>Віддалений доступ до робочого столу з інтеграцією у Windows</i>
Протокол	<i>Remote Framebuffer Protocol</i>	<i>Remote Desktop Protocol</i>
Сумісність	<i>Кросплатформна</i>	<i>Переважно Windows</i>
Принцип роботи	<i>Передача зображення екрану, введення з клавіатури та миші</i>	<i>Передача графічних команд для рендерингу на клієнтському пристрої</i>
Продуктивність	<i>Нижча, потребує високої пропускної здатності</i>	<i>Висока, оптимізована</i>
Безпека	<i>Вимагає додаткових налаштувань</i>	<i>Вбудоване шифрування</i>
Можливості користувача	<i>Дозволяє керувати віддаленим екраном без можливості зміни сесії</i>	<i>Підтримує вхід у систему як окремий користувач без втручання у локальну сесію</i>
Функціональність	<i>Базовий спільний доступ до екрану</i>	<i>Додаткові функції (файли, аудіо, USB)</i>
Використання ресурсів	<i>Високе навантаження на мережу та сервер</i>	<i>Оптимізоване кодування трафіку</i>
Кількість користувачів	<i>Декілька користувачів одночасно</i>	<i>Один на сесію, багатокористувацький доступ до серверів</i>
Сфера застосування	<i>Адміністрування серверів, технічна підтримка,</i>	<i>Корпоративні мережі, мультимедіа</i>

Аналіз показав, що жоден з протоколів не є універсальним для всіх завдань проекту. Тому було прийнято гібридне проєктне рішення.

RDP використовується для доступу до Windows Server. Це рішення обґрунтоване його високою продуктивністю, інтеграцією з ОС та вбудованими засобами безпеки.

VNC використовується для доступу до робочих середовищ Linux. Це зумовлено його кросплатформеністю. Для нівелювання недоліків безпеки варто приділити увагу налаштуванням додаткових засобів безпеки.

2.4. Архітектура корпоративної мережі

В якості середовища для емуляції мережі було обрано GNS3. Основна структура – це два незалежні сегменти з власними хостами та маршрутизаторами, з'єднані між собою тунелем WireGuard, який реалізовано на MikroTik маршрутизаторах. Такий підхід спростить конфігурацію хостів і дозволить централізовано керувати VPN-зв'язком.

Загалом є два сегменти – Центральний офіс та Віддалений вузол.

Центральний офіс складатиметься з:

1. Маршрутизатора MikroTik як VPN-шлюзу і точка виходу в Інтернет.
2. 2 машин з Ubuntu Server, як серверу з віддаленими робочими столами на основі Linux.
3. Windows Server 19, як серверу з віддаленими робочими столами на основі Windows.
4. Окремого серверу SIEM, для моніторингу стану сервісів та пристроїв мережі.

Віддалений вузол складатиметься з:

1. Маршрутизатора MikroTik як другого кінця VPN-тунелю і точка виходу в Інтернет.
2. Робочої станції на основі Linux.
3. Робочої станції на основі Windows.
4. Кілька VPCs для перевірки зв'язку.

Дану спроектовану схему наведено на рисунку 2.1.

Для цих мереж також розроблено параметри IP-адресації, які наведено у таблицях 2.3 та 2.4.

Таблиця 2.3

Параметри IP-адресації для центральної мережі

Пристрій / Мережа	IP-адреса
Центральна мережа	192.168.1.0/24
Ubuntu Server (Session-Based)	192.168.1.10/24

Продовження таблиці 2.3.

Ubuntu Server (Docker)	192.168.1.11/24
Windows Server	192.168.1.12/24
SIEM Server	192.168.1.21/24
R1	192.168.1.1/24
Тунель WireGuard (R1)	10.0.0.1/30

Таблиця 2.4

Параметри IP-адресації для віддаленої мережі

Пристрій / Мережа	IP-адреса
Віддалена мережа	192.168.2.0/24
Робоча станція з Ubuntu	192.168.2.10/24
Робоча станція з Windows	192.168.2.11/24
R2	192.168.2.1/24
Тунель WireGuard (R2)	10.0.0.2/30

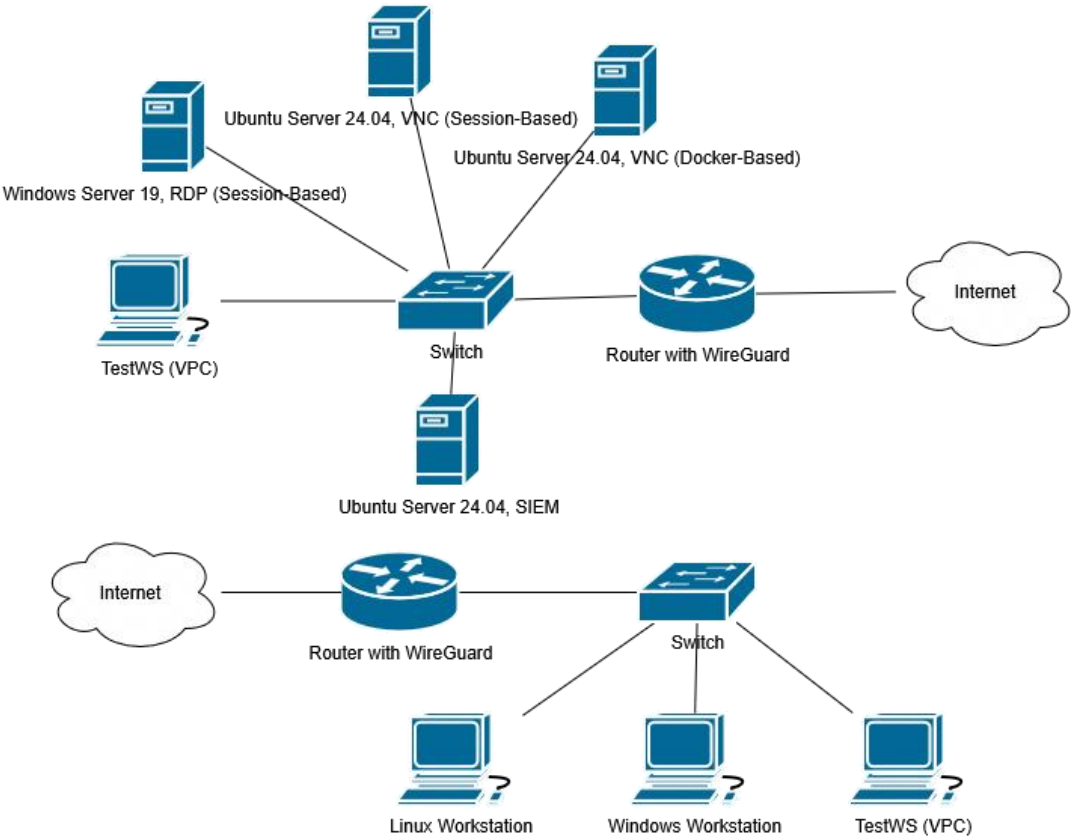


Рисунок 2.1 – Проєкт мережі

Окремим етапом потрібно буде виконати налаштування маршрутів і правил фаєрволу на маршрутизаторі, щоб трафік з обох підмереж успішно проходив через тунель.

На самих хостах налаштування простіше – їм необхідно лише задати шлюзом відповідний маршрутизатор і вони вже «бачитимуть» іншу сторону.

Для того щоб забезпечити вихід в Інтернет, на маршрутизаторах також необхідно налаштувати NAT, а на хостах встановити DNS-сервери.

2.5. Проектування VPN-рішення на базі WireGuard

На відміну від класичного підходу, де тунель WireGuard налаштовується прямо на кінцевих пристроях, у цьому проєкті VPN реалізовуватиметься безпосередньо на маршрутизаторах. Це дасть змогу повністю ізолювати тунель від користувачів – весь трафік між підмережами автоматично маршрутизуватиметься через VPN-шлюз, прозоро для хостів.

Принцип цієї реалізації такий, що кожен маршрутизатор виконує роль своєрідного шлюзу, що:

- піднімає інтерфейс WireGuard;
- встановлює однорангове з'єднання з іншим маршрутизатором;
- має маршрут для передавання трафіку між підмережами через цей тунель, обробляє NAT для виходу хостів в Інтернет;

У кожного з пристроїв буде статичний маршрут до віддаленої підмережі через інтерфейс WireGuard. Це означає, що будь-який трафік між вузлами йде не напряму через Інтернет, а через захищений тунель.

Цей підхід дозволяє винести всю логіку VPN за межі хостів. Це для підтримки, масштабування і безпеки – користувачам не потрібно нічого налаштовувати або знати про VPN. Їхній трафік автоматично потраплятиме в іншу мережу, як ніби вона є локальною.

2.6. Проектування рішення віддаленого доступу

Після налаштування тунелю WireGuard між сегментами мережі виникла необхідність організувати повноцінний доступ до робочих середовищ віддалених користувачів. Для віддаленого доступу, для майбутнього порівняння, використано дві різні технології: RDP – для серверної Windows-інфраструктури, та VNC – для графічного доступу до Linux-контейнерів, які запускаються через Docker.

На вузлі з Windows Server 2019 необхідно встановити служби віддаленого робочого столу. Під час тестової інсталяції довелося відмовитись від варіанту з віртуальними машинами – Virtual machine-based deployment давав помилки, ймовірно через обмеження в середовищі GNS3 або невідповідність апаратних вимог. Натомість, можливо використовувати Session-based desktop environment – він набагато простіший у реалізації й дозволяє запускати RDP-сеанси для кількох користувачів без складних налаштувань, однак він не передбачає ізоляцію користувачів.

Основні кроки налаштування виглядають так:

1. У Server Manager додати полі Remote Desktop Services, Session Host, Licensing.
2. Налаштувати групу Remote Desktop Users та додати туди потрібних користувачів.
3. RDP порт (3389) відкрити тільки у напрямку VPN-тунелю – ззовні він не повинен бути доступний.

У Linux-середовищі такий же функціонал можна реалізувати на базі VNC, окремо налаштовуючи службу VNC для кожного окремого користувача, однак такий підхід не забезпечує ізоляцію користувачів і всі вони «ділитимуть» одну машину, але є і інший спосіб.

Можна використати контейнери Docker. Принцип роботи такий, що у кожного користувача буде свій Docker-контейнер з встановленим Linux та

засобом віддаленого доступу VNC, а для збереження даних між сеансами кожен контейнер матиме свою змонтовану директорію.

Основні кроки налаштування такого підходу такі:

1. Встановлення Docker.
2. Створення образу з Linux, GUI та VNC, або використання уже існуючого з бази Docker.io.
3. Запуск контейнера з змонтованою директорією.
4. Для автоматизації процесу, щоб піднімати одразу усі контейнери користувачів, можна створити docker-compose.yml файл, де виконати усі попередні налаштування контейнерів. Зокрема:
 - a. Створення контейнеру з образу.
 - b. Відкриття портів.
 - c. Встановлення паролю для доступу VNC.
 - d. Створення користувача.
 - e. Монтування директорії.
5. Після чого налаштувати автозапуск контейнерів.

У результаті матимемо зручну реалізацію VNC-серверу, де можна запустити стільки VNC-сесій, скільки потрібно – просто варіюючи порти і назви контейнерів у docker-compose.yml файлі.

Таким чином, комбінуючи RDP для Windows і VNC для Linux ми можемо швидко й гнучко налаштовувати графічний віддалений доступ до будь-якої машини в корпоративній мережі. І найголовніше – усе це працює в межах VPN, без потреби відкривати небезпечні порти назовні.

2.7. Додаткові заходи безпеки

Крім базового функціоналу VPN та віддаленого доступу, необхідно звернути увагу на безпеку створеного середовища. Для підвищення безпеки можна виконати такі кроки:

1. SIEM/моніторинг:

Впровадити Wazuh або Graylog. Основна ідея – мати логування входів, спроб авторизації, запусків контейнерів, тунельних сесій. У реальному середовищі ніхто не працює без журналів.

2. Брандмауер:

Відкрити лише необхідні порти і лише для VPN, всі інші спроби підключення – блокувати.

Висновки до другого розділу: у другому розділі було спроектовано архітектуру корпоративної мережі, що складається з центрального офісу та віддаленого вузла. На основі аналізу, проведеного у попередньому розділі та додаткових порівняльних таблиць, було обґрунтовано вибір ключових технологій: протокол WireGuard для реалізації VPN-тунелю на рівні маршрутизаторів MikroTik та гібридний підхід до віддаленого доступу – RDP для Windows та VNC (з ізоляцією через Docker) для Linux. Розроблено детальний план IP-адресації та передбачено впровадження додаткових заходів безпеки, включаючи систему моніторингу SIEM (Wazuh).

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ЗПРОЄКТОВАНИХ РІШЕНЬ НА ТЕСТОВОМУ СТЕНДІ

3.1. Підготовка середовища емуляції

Почнемо з налаштування мережі у GNS3. Варто зазначити, що якщо на машині-хості з GNS3 відключений Hyper-V, необхідно додати новий параметр у конфігураційний файл `gns3_server.ini`. Редагування конфігураційного файлу для емуляції мережі показано на рисунку 3.1.

```
[Server]
path = D:\Program Files\GNS3\gns3server.EXE
ubridge_path = D:\Program Files\GNS3\ubridge\ubridge.EXE
host = localhost
port = 3080
images_path = C:\Users\Avyn\GNS3\images
projects_path = C:\Users\Avyn\GNS3\projects
appliances_path = C:\Users\Avyn\GNS3\appliances
additional_images_paths =
symbols_path = C:\Users\Avyn\GNS3\symbols
configs_path = C:\Users\Avyn\GNS3\configs
report_errors = True
auto_start = True
allow_console_from_anywhere = False
auth = True
user = admin
password = UDatmnC4QHLdd284j5HYjbW9MZvpJ7n0Mcu8mt3Ho3igGoJ1fcsGQDSGnoYTT7E1
protocol = http
console_start_port_range = 5000
console_end_port_range = 10000
udp_start_port_range = 10000
udp_end_port_range = 20000
hardware_virtualization_check = False
```

Рисунок 3.1 – Редагування конфігураційного файлу

Першим етапом налаштування мережі буде додавання маршрутизаторів на яких буде налаштовано WireGuard. Для цього було вирішено використати маршрутизатор від MikroTik. Для цього з їхнього сайту необхідно завантажити Cloud Hosted Router рішення. Для імпорту в GNS3 нам знадобиться саме варіант з “Raw disk image”. Процес завантаження образу маршрутизатора MikroTik для використання WireGuard відображено на рисунку 3.2.

Cloud Hosted Router				
	6.49.18 Long-Term	6.49.19 Stable	7.20.2 Stable	7.21beta3 Testing
Images	vmdk, vhd, vdi, ova, img			
Main package				
VHDX image				
VMDK image				
VDI image				
VirtualPC image				
OVA template				
Raw disk image	SHA256	SHA256	SHA256	SHA256
Extra packages				
The Dude client				
VDI image (arm64)				
Raw disk image (arm64)				
Changelog				

Рисунок 3.2 – Завантаження образу маршрутизатора від MikroTik

Для цієї роботи необхідно саме версія 7.x, тому що у попередніх 6.x версіях WireGuard відсутній.

Процес імпорту досить простий – розархівовуємо архів і в меню Qemu VMs натискаємо New, вписуємо ім'я і далі всі параметри за замовчуванням. На етапі вибору образу вказуємо шлях до розпакованого образу. На рисунку 3.3 наведено приклад імпорту образу маршрутизатора MikroTik в GNS3.

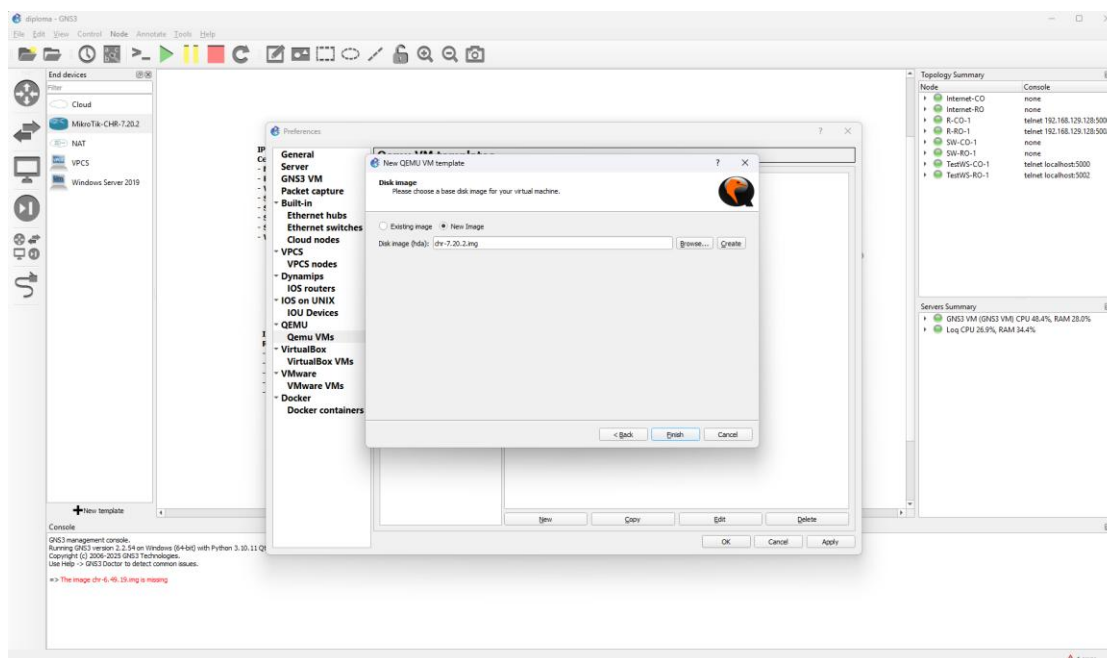


Рисунок 3.3 – Завантаження образу маршрутизатора від MikroTik

Загальна конфігурація тестового стенду, де обидва маршрутизатори підключені до Інтернету, представлена на рисунку 3.4.

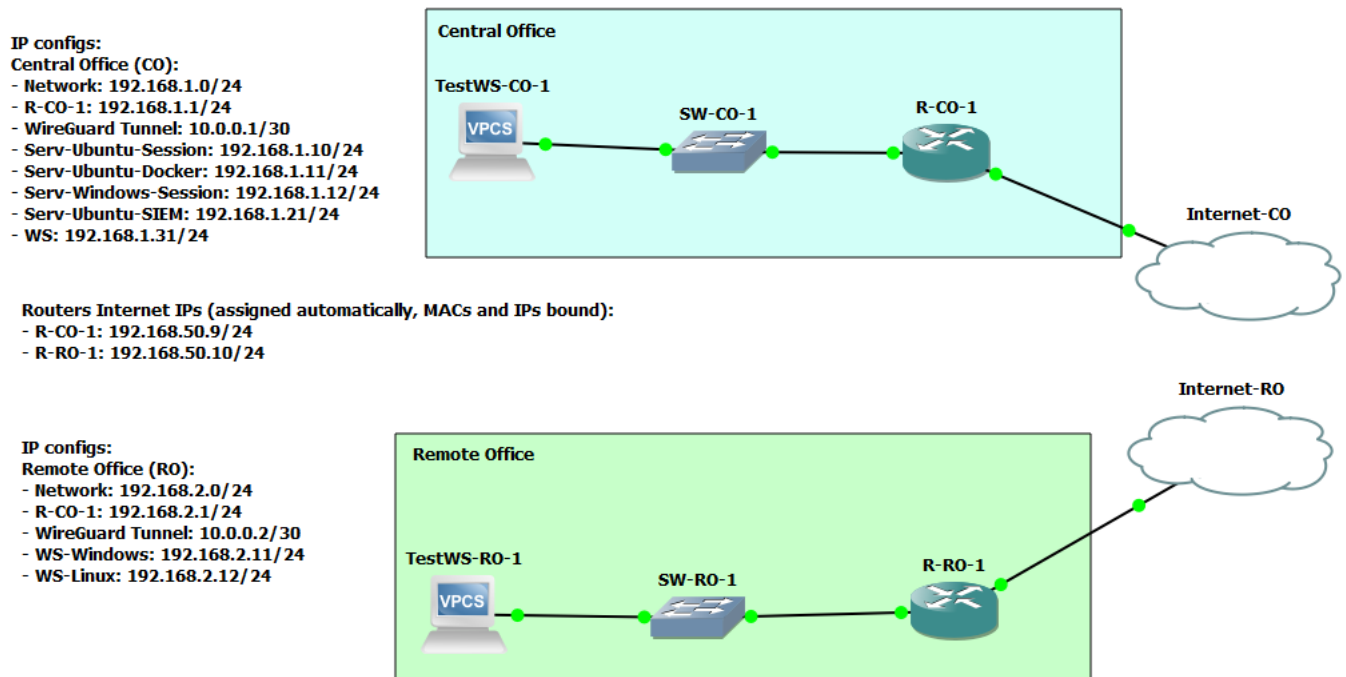


Рисунок 3.4 – Обидва маршрутизатори підключені до Інтернету

Обидва маршрутизатори підключаємо до Cloud, тому що зв'язок між ними реалізовуватиметься через Інтернет.

3.2. Налаштування VPN-з'єднання на маршрутизаторах MikroTik

Для початку змінюємо ім'я пристрою, створюємо новий WireGuard інтерфейс та прописуємо IP-адреси. Це виконується даними командами:

```
system identity set name=R1
in wireguard add listen-port=12321 name=wireguard1
ip addr add addr = 10.0.0.1/30 in=wireguard1
ip addr add addr=192.168.1.1/24 in=ether2
```

Приклад початкового налаштування WireGuard інтерфейсу та IP-адресації на маршрутизаторі R1 показано на рисунку 3.5.

```
[admin@MikroTik] > ip addr print
Flags: D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS NETWORK INTERFACE
0 D 192.168.50.9/24 192.168.50.0 ether1
[admin@MikroTik] >
[admin@MikroTik] > system identity set name=R1
[admin@R1] > ip addr add addr=192.168.1.1/24 in=ether2
[admin@R1] > in wireguard add listen-port=12321 name=wireguard1
[admin@R1] > ip addr add addr=10.0.0.1/30 in=wireguard1
[admin@R1] > in wireguard print
Flags: X - disabled; R - running
0 R name="wireguard1" mtu=1420 listen-port=12321
private-key="MCRQfCySMKn4gqlks1SLWj1D1cQWS+j4ThEzNoFOglM="
public-key="vmRQf1ASwBQqrn/HgqFPNrPz/+Tyh99m5bASkxgTwo="
[admin@R1] > []
```

Рисунок 3.5 – Приклад налаштування на R-CO-1

Це необхідно виконати на обох маршрутизаторах. Тепер необхідно налаштувати підключення по WireGuard між двома маршрутизаторами, для цього нам спочатку необхідно дізнатися їх публічні ключі за допомогою команди “in wireguard print”. Налаштування вузла з WireGuard виконується даними командами:

```
in wireguard peers add allowed-address=192.168.2.0/24 \
endpoint-address=192.168.50.10 \
endpoint-port=13231 \
interface=wireguard1 \
public-key="dZM7NaFIQoImSDa2S69Jq8PKC00ugRW+zClhBEqSdWY="
```

На рисунку 3.6 наведено приклад налаштування однорангового вузла WireGuard на R1 з публічним ключем віддаленого вузла R2.

```
[admin@R1] > in wireguard peers add allowed-address=192.168.2.0/24 \
... endpoint-address=192.168.50.10 \
... endpoint-port=12321 \
... interface=wireguard1 \
... public-key="dZM7NaFIQoImSDa2S69Jq8PKC00ugRW+zClhBEqSdWY="
[admin@R1] > []
```

Рисунок 3.6 – Приклад налаштування на R-CO-1

У полі публічного ключа необхідно зазначити ключ віддаленого вузла, у дана конфігурація з налаштування маршрутизатора R1, відповідно вводиться ключ з маршрутизатора R2.

Після чого необхідно встановити статичний маршрут та налаштувати брандмауер для можливості підключень між мережами.

Налаштування статичного маршруту:

```
ip route add dst-address=192.168.2.0/24 gateway=wireguard1
```

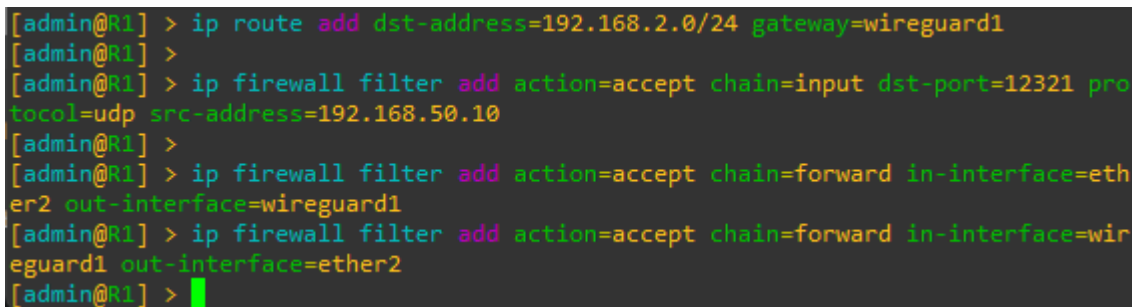
Правила фаєрволу, які пропускають трафік з локальної мережі до інтерфейсу WireGuard, та навпаки:

```
ip firewall filter add action=accept chain=input dst-port=12321 protocol=udp src-address=192.168.50.10
```

```
ip firewall filter add action=accept chain=forward in-interface=ether2 out-interface=wireguard1
```

```
ip firewall filter add action=accept chain=forward in-interface=wireguard1 out-interface=ether2
```

Приклад налаштування статичного маршруту та правил фаєрволу на R1, що дозволяють трафік через тунель, представлено на рисунку 3.7.

A screenshot of a terminal window showing the configuration of router R1. The commands are entered line by line, and the prompt is [admin@R1] >. The commands are: ip route add dst-address=192.168.2.0/24 gateway=wireguard1; ip firewall filter add action=accept chain=input dst-port=12321 protocol=udp src-address=192.168.50.10; ip firewall filter add action=accept chain=forward in-interface=ether2 out-interface=wireguard1; ip firewall filter add action=accept chain=forward in-interface=wireguard1 out-interface=ether2. The last line shows the prompt [admin@R1] > with a cursor.

```
[admin@R1] > ip route add dst-address=192.168.2.0/24 gateway=wireguard1
[admin@R1] >
[admin@R1] > ip firewall filter add action=accept chain=input dst-port=12321 protocol=udp src-address=192.168.50.10
[admin@R1] >
[admin@R1] > ip firewall filter add action=accept chain=forward in-interface=ether2 out-interface=wireguard1
[admin@R1] > ip firewall filter add action=accept chain=forward in-interface=wireguard1 out-interface=ether2
[admin@R1] >
```

Рисунок 3.7 – Приклад налаштування на R-CO-1

Аналогічна конфігурація і для маршрутизатора R2:

```
system identity set name=R2
```

```
ip addr add addr=192.168.2.1/24 in=ether2
```

```
in wireguard add listen-port=12321 name=wireguard1
```

```
ip addr add addr=10.0.0.2/30 in=wireguard1
```

```
in wireguard print
```

```
in wireguard peers add allowed-address=192.168.1.0/24 \
```

```
endpoint-address=192.168.50.9 \
```

```
endpoint-port=12321 \
```

```
interface=wireguard1 \
```

```
public-key="vmRQf1ASwBQqm/HgqFPNrPz/+Tyh99m5bASkxgTWo="
```

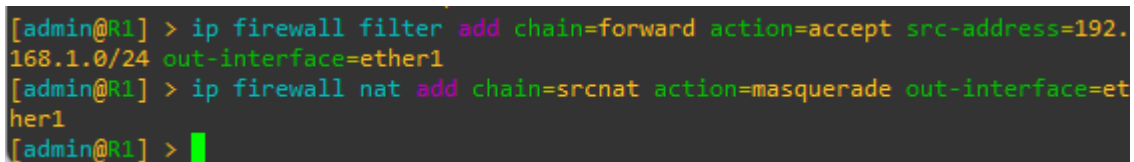
```
ip route add dst-address=192.168.1.0/24 gateway=wireguard1
```

```
ip firewall filter add action=accept chain=input dst-port=12321 protocol=udp src-  
address=192.168.50.9
```

```
ip firewall filter add action=accept chain=forward in-interface=ether2 out-interface=wireguard1
```

```
ip firewall filter add action=accept chain=forward in-interface=wireguard1 out-interface=ether2
```

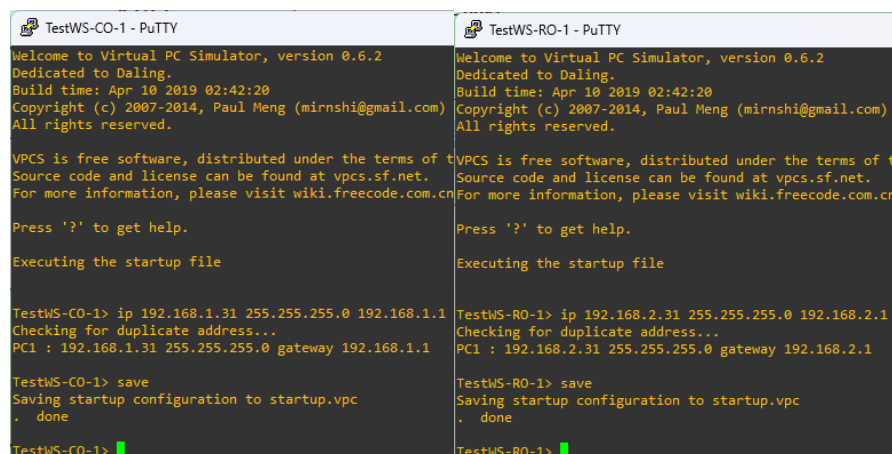
Приклад налаштування статичного маршруту та правил фаєрволу на R2, що дозволяють трафік через тунель, представлено на рисунку 3.8.



```
[admin@R1] > ip firewall filter add chain=forward action=accept src-address=192.  
168.1.0/24 out-interface=ether1  
[admin@R1] > ip firewall nat add chain=srcnat action=masquerade out-interface=et  
her1  
[admin@R1] >
```

Рисунок 3.8 – Приклад налаштування на R-CO-1

Результатом даних налаштувань є вдало встановлене з'єднання WireGuard та можливість обміну між пристроями віддалених підмереж. На рисунку 3.9 відображено налаштування IP-адрес тестових VPCs у локальній та віддаленій мережах. На рисунку 3.10 та 3.11 видно успішне підключення між мережами та деталі з'єднання WireGuard, що підтверджує його успішну роботу між маршрутизаторами R1 та R2.



TestWS-CO-1 - PuTTY	TestWS-RO-1 - PuTTY
Welcome to Virtual PC Simulator, version 0.6.2 Dedicated to Daling. Build time: Apr 10 2019 02:42:20 Copyright (c) 2007-2014, Paul Meng (mirnshi@gmail.com) All rights reserved. VPCS is free software, distributed under the terms of the Source code and license can be found at vpcs.sf.net. For more information, please visit wiki.freecode.com.cn Press '?' to get help. Executing the startup file TestWS-CO-1> ip 192.168.1.31 255.255.255.0 192.168.1.1 Checking for duplicate address... PC1 : 192.168.1.31 255.255.255.0 gateway 192.168.1.1 TestWS-CO-1> save Saving startup configuration to startup.vpc . done TestWS-CO-1>	Welcome to Virtual PC Simulator, version 0.6.2 Dedicated to Daling. Build time: Apr 10 2019 02:42:20 Copyright (c) 2007-2014, Paul Meng (mirnshi@gmail.com) All rights reserved. VPCS is free software, distributed under the terms of the Source code and license can be found at vpcs.sf.net. For more information, please visit wiki.freecode.com.cn Press '?' to get help. Executing the startup file TestWS-RO-1> ip 192.168.2.31 255.255.255.0 192.168.2.1 Checking for duplicate address... PC1 : 192.168.2.31 255.255.255.0 gateway 192.168.2.1 TestWS-RO-1> save Saving startup configuration to startup.vpc . done TestWS-RO-1>

Рисунок 3.9 – Налаштування тестових VPCs у обох мережах

TestWS-RO-1 - PuTTY	TestWS-CO-1 - PuTTY
Source code and license can be found at vpcs.sf.net . For more information, please visit wiki.freecode.com.cn . Press '?' to get help. Executing the startup file TestWS-RO-1> ip 192.168.2.31 255.255.255.0 192.168.2.1 Checking for duplicate address... PC1 : 192.168.2.31 255.255.255.0 gateway 192.168.2.1 TestWS-RO-1> save Saving startup configuration to startup.vpc . done TestWS-RO-1> ping 192.168.1.31 84 bytes from 192.168.1.31 icmp_seq=1 ttl=62 time=3.843 ms 84 bytes from 192.168.1.31 icmp_seq=2 ttl=62 time=3.095 ms 84 bytes from 192.168.1.31 icmp_seq=3 ttl=62 time=4.786 ms 84 bytes from 192.168.1.31 icmp_seq=4 ttl=62 time=2.812 ms 84 bytes from 192.168.1.31 icmp_seq=5 ttl=62 time=2.921 ms TestWS-RO-1> █	TestWS-CO-1> ping 192.168.2.31 192.168.2.31 icmp_seq=1 timeout 192.168.2.31 icmp_seq=2 timeout 192.168.2.31 icmp_seq=3 timeout 192.168.2.31 icmp_seq=4 timeout 192.168.2.31 icmp_seq=5 timeout TestWS-CO-1> ping 192.168.2.31 192.168.2.31 icmp_seq=1 timeout 192.168.2.31 icmp_seq=2 timeout 192.168.2.31 icmp_seq=3 timeout 192.168.2.31 icmp_seq=4 timeout 192.168.2.31 icmp_seq=5 timeout TestWS-CO-1> ping 192.168.2.31 84 bytes from 192.168.2.31 icmp_seq=1 ttl=62 time=15.052 ms 84 bytes from 192.168.2.31 icmp_seq=2 ttl=62 time=2.597 ms 84 bytes from 192.168.2.31 icmp_seq=3 ttl=62 time=6.261 ms 84 bytes from 192.168.2.31 icmp_seq=4 ttl=62 time=3.152 ms 84 bytes from 192.168.2.31 icmp_seq=5 ttl=62 time=4.155 ms TestWS-CO-1> █ TestWS-CO-1> █

Рисунок 3.10 – Зв'язок між пристроями встановлених

```
[admin@R1] > in wireguard peers print detail
Flags: X - disabled; D - dynamic
 0 interface=wireguard1 name="peer1"
  public-key="dZM7NaFIQoImSDa2S69Jq8PKC00ugRW+zClhBEqSdWY="
  private-key="" endpoint-address=192.168.50.10 endpoint-port=12321
  current-endpoint-address=192.168.50.10 current-endpoint-port=12321
  allowed-address=192.168.2.0/24 preshared-key="" client-endpoint=""
  rx=1404 tx=5.8KiB last-handshake=53s
[admin@R1] > █

[admin@R2] > in wireguard peers print detail
Flags: X - disabled; D - dynamic
 0 interface=wireguard1 name="peer1"
  public-key="vmRQf1ASwBQqrn/HgqFPNrPz+/+Tyh99m5bASkxgTWO="
  private-key="" endpoint-address=192.168.50.9 endpoint-port=12321
  current-endpoint-address=192.168.50.9 current-endpoint-port=12321
  allowed-address=192.168.1.0/24 preshared-key="" client-endpoint=""
  rx=1460 tx=1404 last-handshake=42s
[admin@R2] > █
```

Рисунок 3.11 – WireGuard успішно працює

Також варто забезпечити можливість виходу в Інтернет для всіх пристроїв підмережі, аби встановлювати оновлення, тощо. Для цього необхідно налаштувати NAT-masquarade, а також налаштувати роутер як DNS-сервер. Налаштування NAT-маскарадингу та DNS на маршрутизаторі R1 для забезпечення виходу в Інтернет показано на рисунку 3.12.

```

[admin@R1] > ip firewall nat add chain=srcnat action=masquerade out-interface=ether1
[admin@R1] > /ip dns set allow-remote-requests=yes
[admin@R1] > /ip dns set servers=8.8.8.8,1.1.1.1
[admin@R1] > ping google.com
  SEQ HOST                               SIZE TTL TIME          STATUS
    0 142.250.109.113                     56 113 24ms148us
    1 142.250.109.113                     56 113 24ms330us
sent=2 received=2 packet-loss=0% min-rtt=24ms148us avg-rtt=24ms239us
max-rtt=24ms330us
[admin@R1] >

```

Рисунок 3.12 – Приклад налаштування на R-CO-1

Окрему увагу потрібно приділити налаштуванням безпеки за допомогою фаєрволу. На рисунку 3.13 представлено приклад повного налаштування правил фаєрволу на маршрутизаторі R1.

```

[admin@R1] > /ip firewall filter add action=accept chain=input connection-state=established,related comment="Allow Established/Related"ed"
[admin@R1] > /ip firewall filter add action=accept chain=input protocol=udp in-interface=ether1 dst-port=12321 comment="Allow WireGuard UDP from WAN"AN"
[admin@R1] > /ip firewall filter add action=accept chain=input protocol=icmp in-interface=ether2 comment="Allow Ping TO Router from LAN"AN"
[admin@R1] > /ip firewall filter add action=accept chain=input protocol=udp in-interface=ether2 dst-port=53 comment="Allow DNS from LAN"AN"
[admin@R1] > /ip firewall filter add action=accept chain=input protocol=tcp in-interface=ether2 dst-port=53 comment="Allow DNS from LAN (TCP)"P"
[admin@R1] > /ip firewall filter add action=drop chain=input in-interface=ether1 comment="Drop ALL other WAN input"ut"
[admin@R1] > /ip firewall filter add action=drop chain=input comment="Drop ALL other input"ut"
[admin@R1] > /ip firewall filter add action=accept chain=forward connection-state=established,related comment="Allow Established/Related"ed"
[admin@R1] > /ip firewall filter add action=drop chain=forward in-interface=wireguard1 dst-address=192.168.1.1 comment="Drop access TO R1 from VPN"PN"
[admin@R1] > /ip firewall filter add action=accept chain=forward in-interface=ether2 out-interface=wireguard1 comment="Allow LAN TO VPN"PN"
[admin@R1] > /ip firewall filter add action=accept chain=forward in-interface=wireguard1 out-interface=ether2 comment="Allow VPN TO LAN"AN"
[admin@R1] > /ip firewall filter add action=accept chain=forward in-interface=ether2 out-interface=ether1 comment="Allow LAN TO WAN"AN"
[admin@R1] > /ip firewall filter add action=drop chain=forward connection-state=invalid comment="Drop Invalid Connections"ns"
[admin@R1] > /ip firewall filter add action=drop chain=forward comment="Drop ALL other forward"rd"
[admin@R1] >

```

Рисунок 3.13 – Приклад налаштування на R-CO-1

На маршрутизаторі реалізовано набір правил фаєрволу, що базується на принципі "заборонено все, що не дозволено явно", для забезпечення багаторівневого захисту. У ланцюжку input, який обробляє трафік, призначений для самого роутера, першочергово дозволяються вже встановлені та пов'язані з'єднання для підтримки існуючих сесій. Далі, явним чином дозволяється

вхідний UDP-трафік на порт WireGuard з зовнішнього WAN-інтерфейсу (ether1), а також ICMP-запити (ping) з локальної мережі (ether2) для діагностики. Одразу після цього спрацьовує правило, що блокує (drop) весь інший трафік, який надходить з WAN-інтерфейсу та призначений для роутера, ефективно закриваючи доступ до інтерфейсів керування з Інтернету.

Ланцюжок forward, відповідальний за транзитний трафік, також починається з дозволу встановлених та пов'язаних з'єднань. Наступним кроком є критично важливе правило, яке блокує (drop) будь-які спроби з VPN-тунелю (wireguard1) отримати доступ безпосередньо до IP-адреси роутера у локальній мережі (192.168.1.1), запобігаючи несанкціонованому доступу до керування з іншого сегмента VPN. Після цього явно дозволяється необхідний робочий трафік: обмін даними між локальною мережею (ether2) та VPN-тунелем (wireguard1) в обох напрямках, а також вихід трафіку з локальної мережі (ether2) в Інтернет (ether1). Завершується конфігурація правилом, що блокує "недійсні" пакети для підвищення стійкості до деяких видів атак. Такий підхід забезпечує захист як периметра мережі, так і внутрішніх сегментів, дозволяючи лише легітимний та очікуваний трафік.

Аналогічні налаштування і для другого маршрутизатора.

Далі варто ознайомитися з процесом налаштування серверів з віддаленим доступом.

3.3. Реалізація віддаленого доступу через RDP

Варто перейти до налаштування Windows Server 2019 та RDP серверу на ньому. Перша задача – налаштувати домен на сервері, тому що без нього встановлення Remote Desktop Services блокується. Встановлення ролі Active Directory Domain Services (AD DS) на Windows Server 2019 показано на рисунку 3.14. Успішне встановлення ролі AD DS підтверджено на рисунку 3.15.

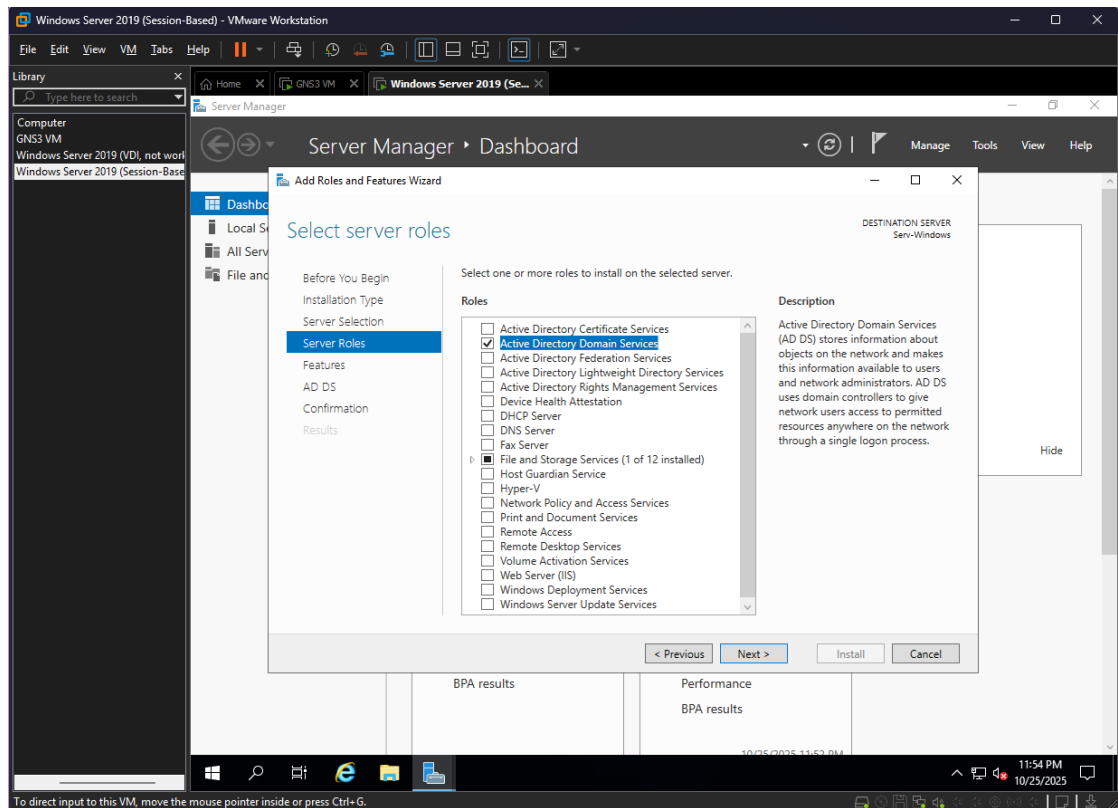


Рисунок 3.14 – Встановлення AD DS

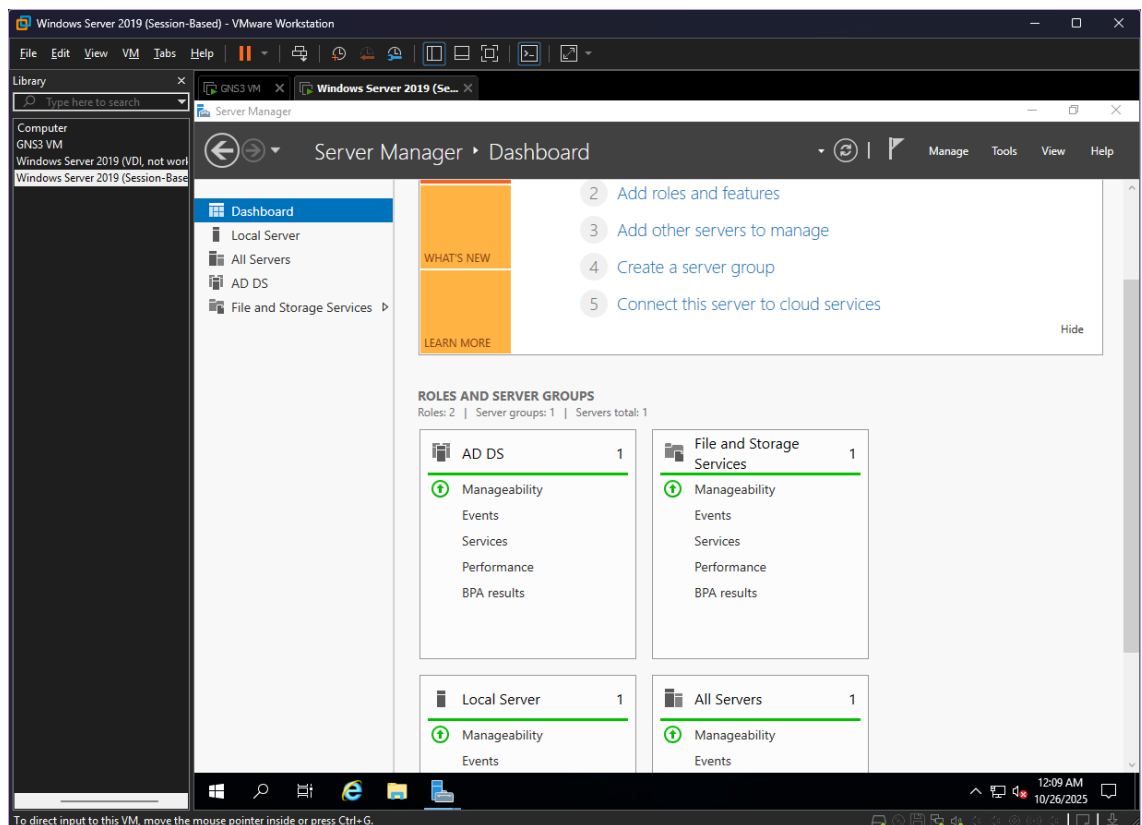


Рисунок 3.15 – Встановлення AD DS успішне

Коли в нас вже є сервер з налаштованим доменом, можна переходити до інсталяції RDS. Для цього у вікні Server Manager переходимо у “Manage” з випадаючого меню вибираємо “Add roles and Features”. Тут у пункті “Installation Type” необхідно обрати “Remote Desktop Services installation”. На рисунку 3.16 відображено вибір сценарію розгортання Remote Desktop Services (RDS)

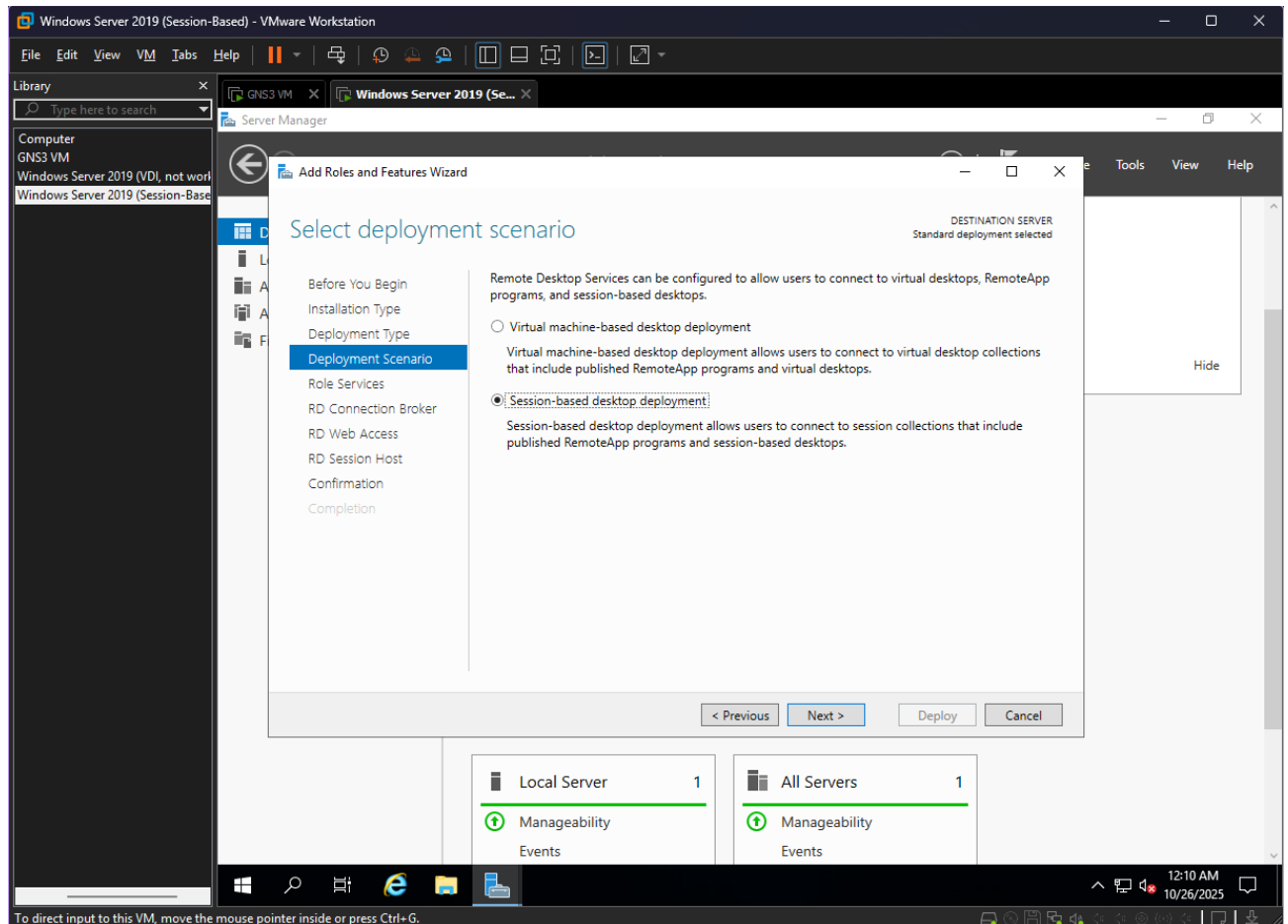


Рисунок 3.16 – Перегляд варіантів встановлення RDS

Далі є 2 опції для серверу – Session based desktop deployment та Virtual machine-based desktop deployment. Розгортання Virtual machine-based desktop deployment вимагає Hyper-V. Ввімкнення Hyper-V всередині віртуальної машини технічно можливе завдяки технологіям “вкладеної віртуалізації” (nested virtualization) за допомогою ввімкнення Intel VT-x/EPT або AMD-V/RVI в середовищі віртуалізації. На рисунку 3.17 показано, як увімкнути опцію "nested virtualization" у налаштуваннях віртуальної машини VMware.

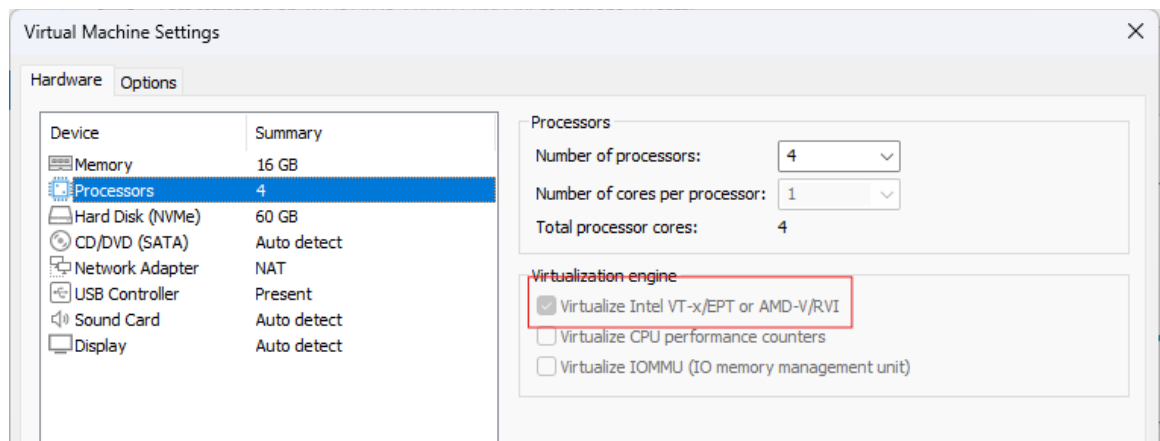


Рисунок 3.17 – Увімкнення «nested virtualization» у VMware

Однак після проведення тестової реалізації, у такому підході некоректно працює RD Virtualization host через який неможливо створити нову колекцію віртуальних столів, а отже реалізація VDI неможлива. Спроба встановлення сценарію Virtual machine-based desktop deployment для RDS показана на рисунку 3.18. Процес встановлення компонентів RDS при спробі розгортання Virtual machine-based desktop deployment відображено на рисунку 3.19. На рисунку 3.20 та 3.21 зафіксовано помилку при встановленні компонентів RDS, що призвела до відкочування змін.

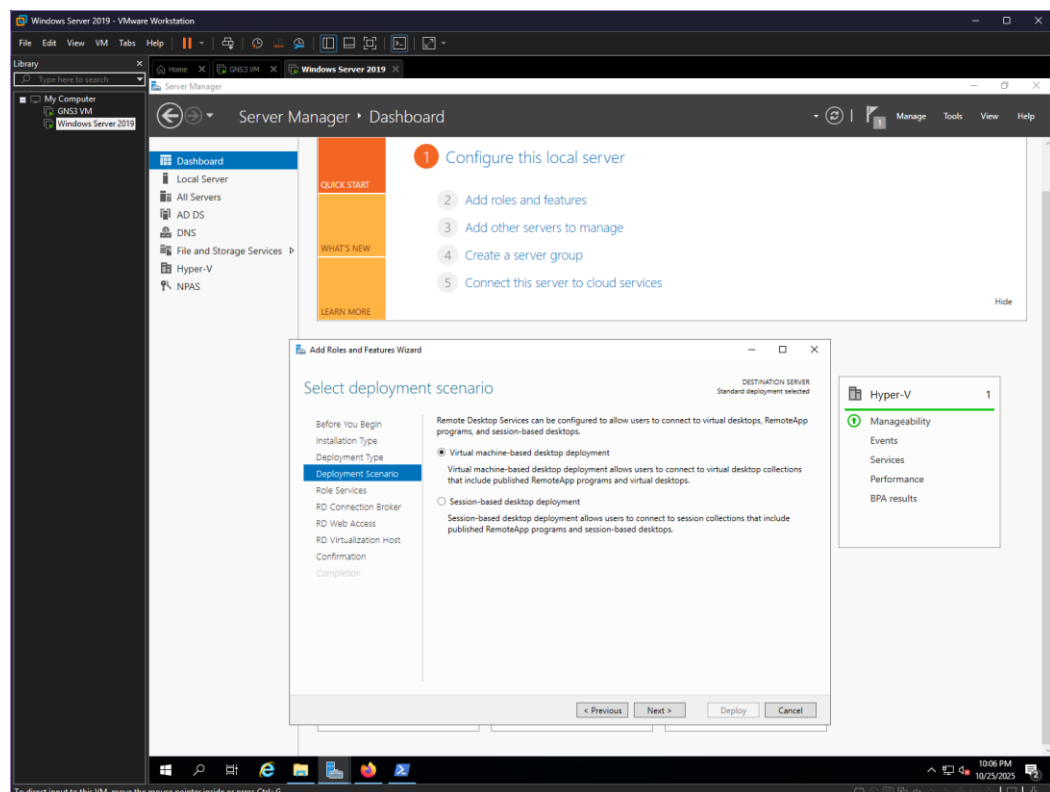


Рисунок 3.18 – Спроба встановлення Virtual machine based deployment

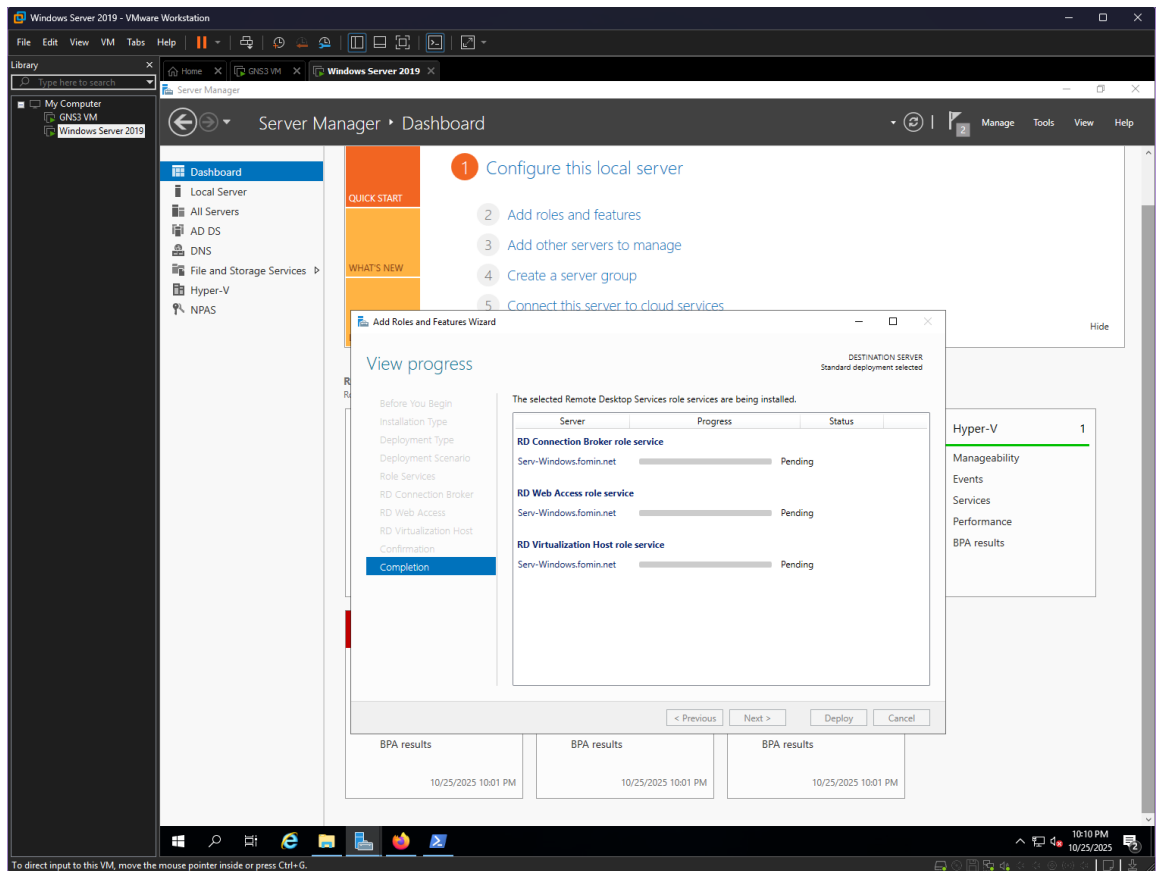


Рисунок 3.19 – Спроба встановлення Virtual machine based deployment

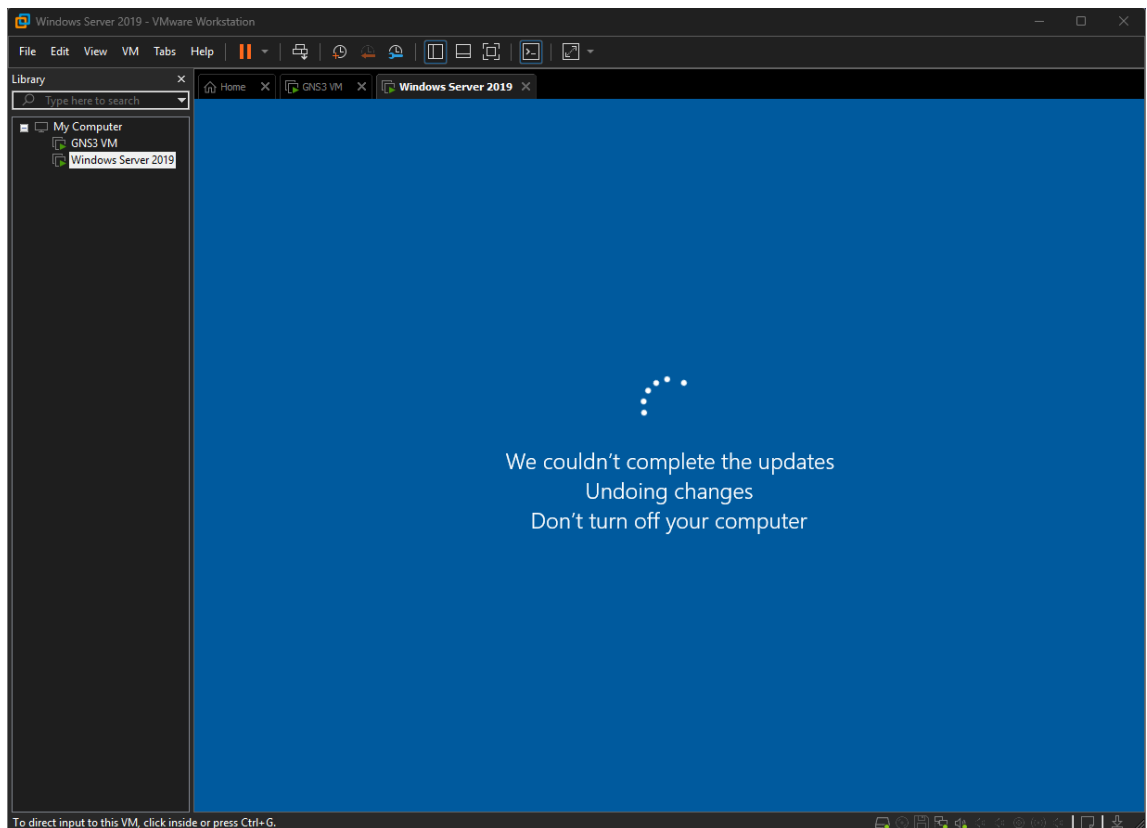


Рисунок 3.20 – Помилки при встановленні Virtual machine based deployment

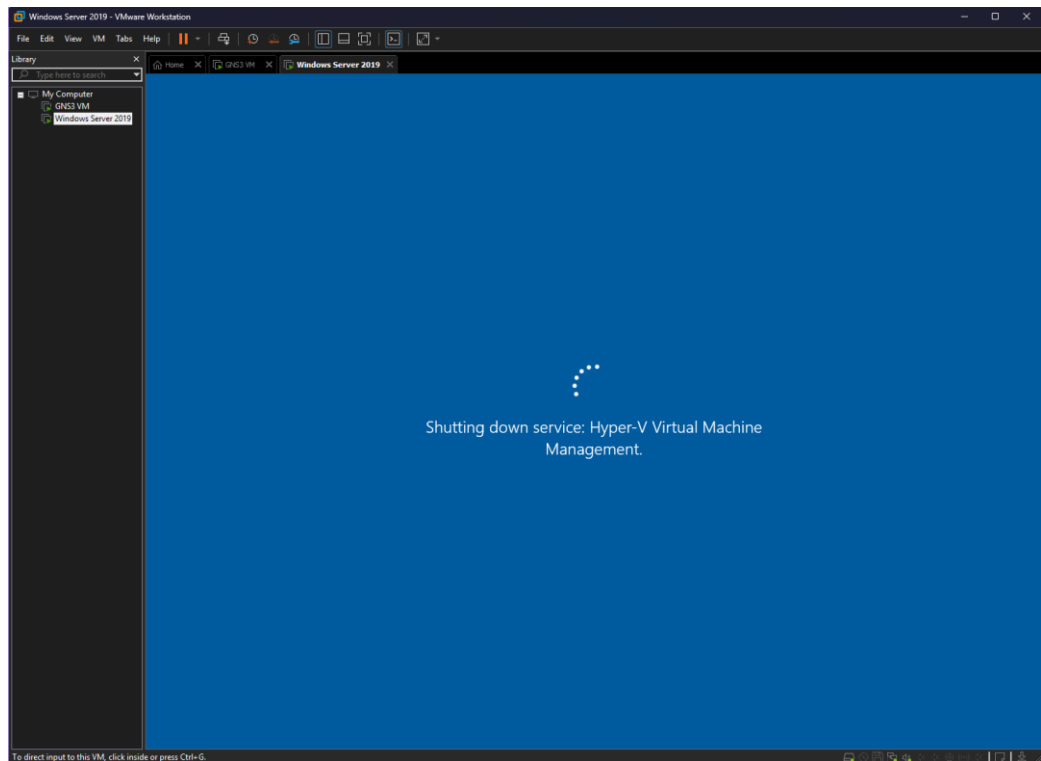


Рисунок 3.21 – Помилки при встановленні Virtual machine based deployment

Так як з налаштуванням іншого варіанту розгортання служб виникають технічні помилки – натомість прийдеться використовувати першу опцію – Session based desktop deployment. На рисунку 3.22 відображено підтвердження компонентів RDS для сценарію Session based desktop deployment.

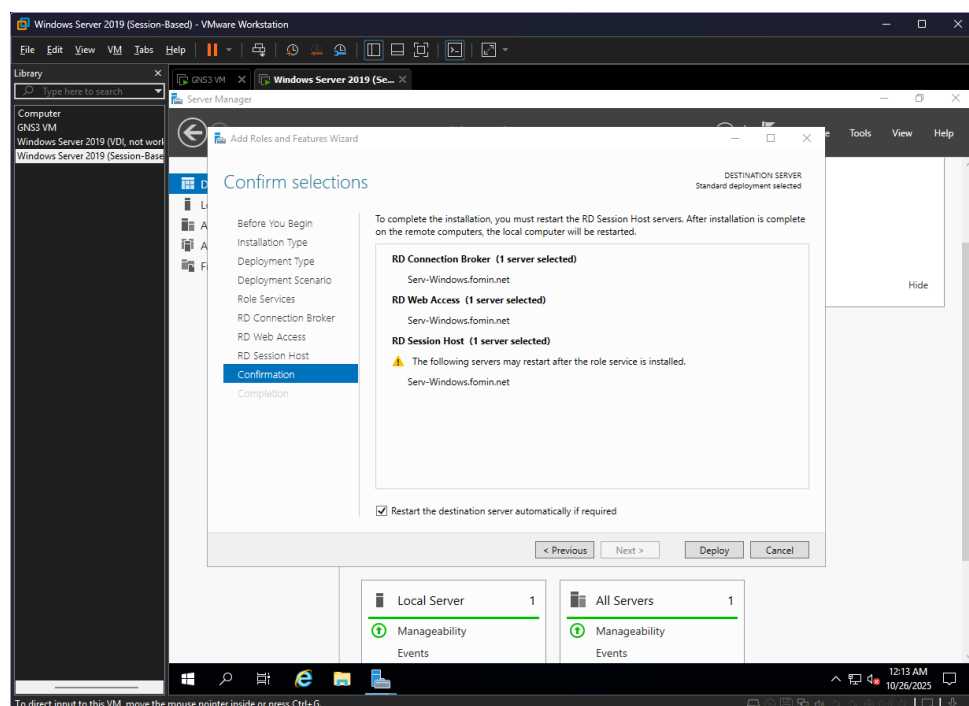


Рисунок 3.22 – Встановлення RDS

Успішне встановлення компонентів RDS для сценарію Session-Based відображено на рисунку 3.23.

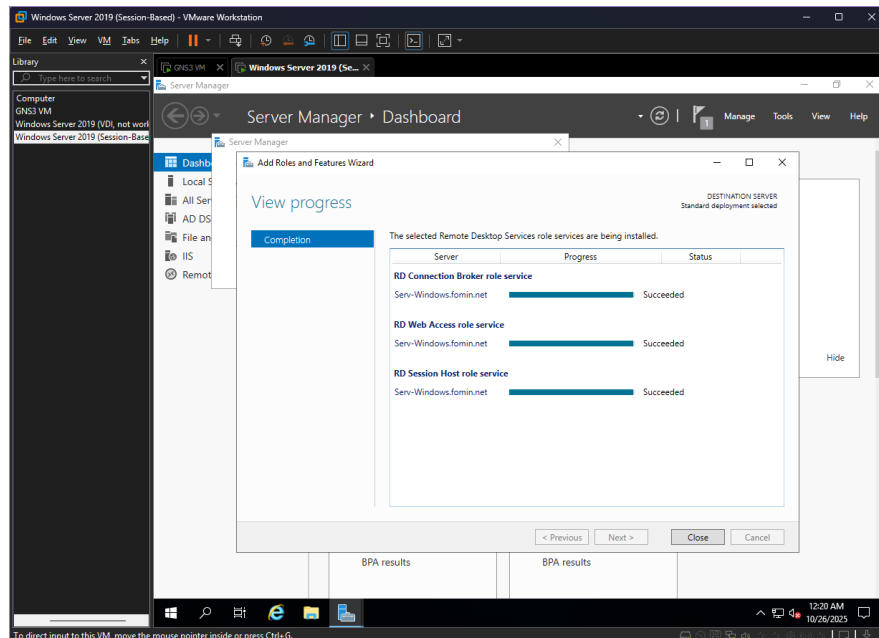


Рисунок 3.23 – Успішне встановлення RDS

Як можна побачити, RDS ще додатково встановлює необхідні служби для її роботи. Встановлення необхідно перейти до налаштування Remote Desktop services, а саме необхідно створити першу колекцію сесій. На рисунку 3.24 показано початок налаштування колекцій сесій для RDS.

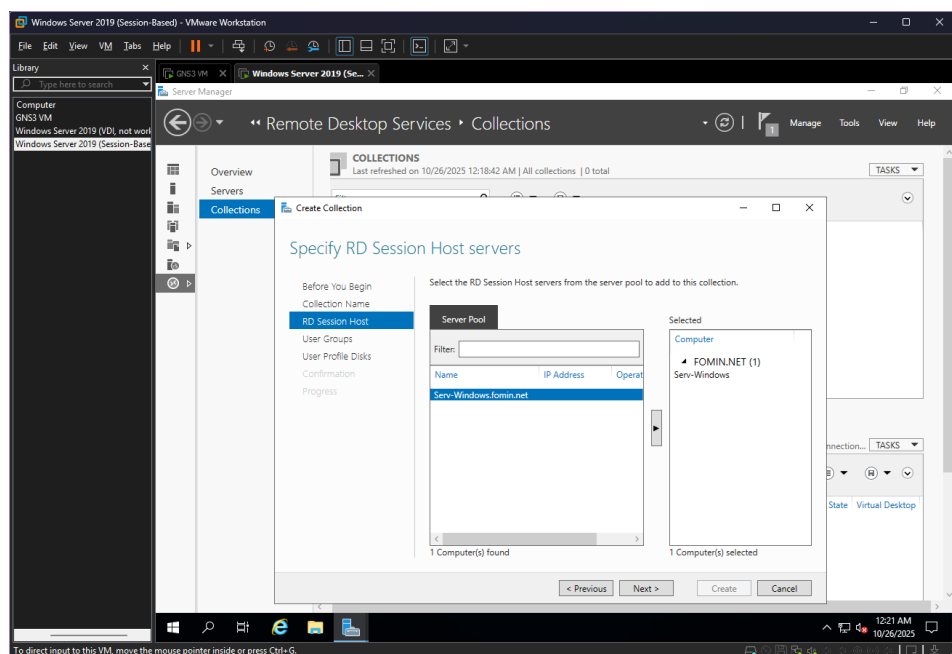


Рисунок 3.24 – Налаштування колекцій для RDS

Налаштування групи користувачів, яким дозволено доступ до колекції RDS, відображено на рисунку 3.25. А на рисунку 3.26 показано налаштування дисків профілів користувачів RDS. Фінальне підтвердження налаштувань колекції сесій RDS відображено на рисунку 3.27.

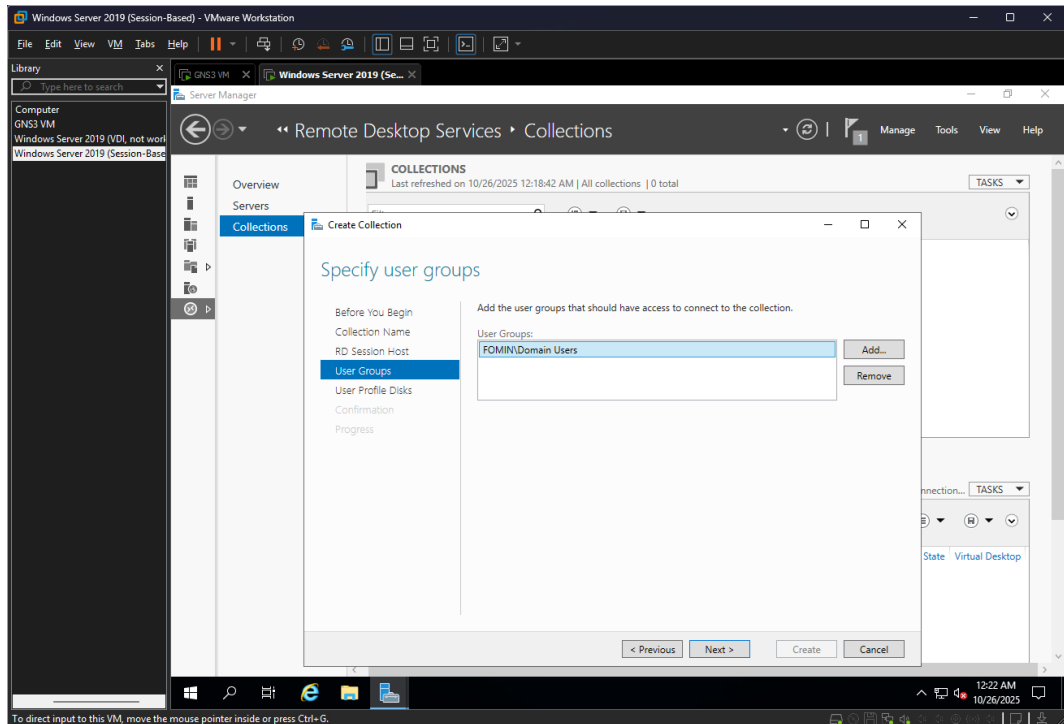


Рисунок 3.25 – Налаштування колекцій для RDS

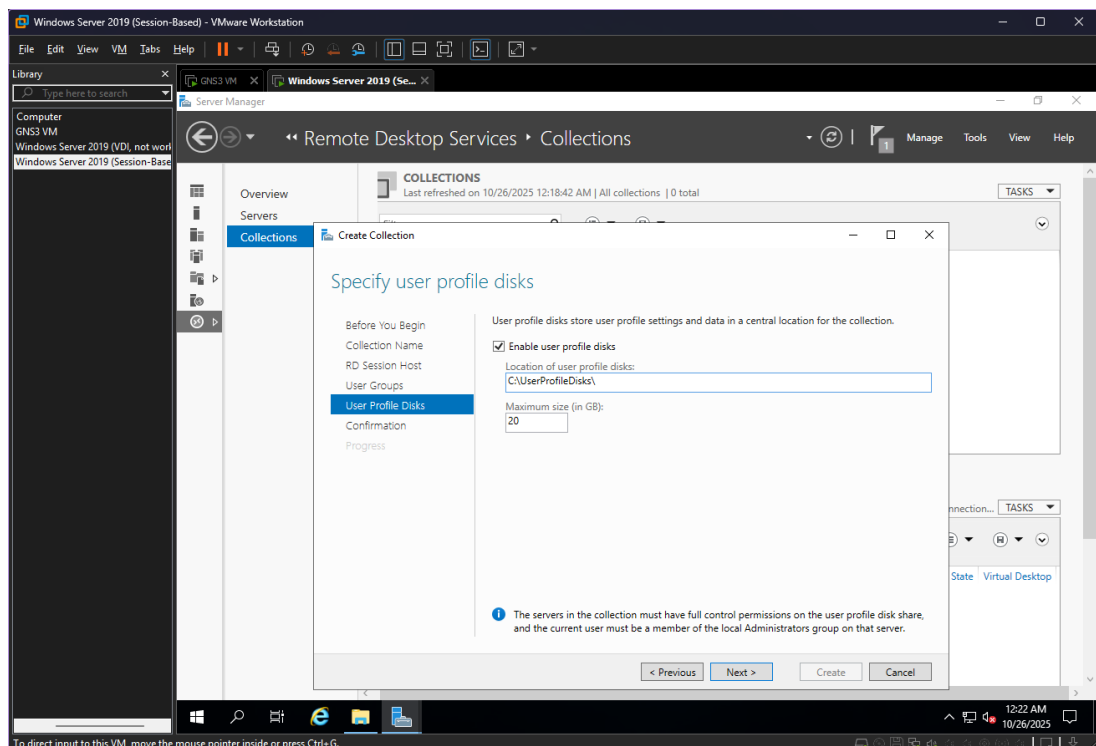


Рисунок 3.26 – Налаштування колекцій для RDS

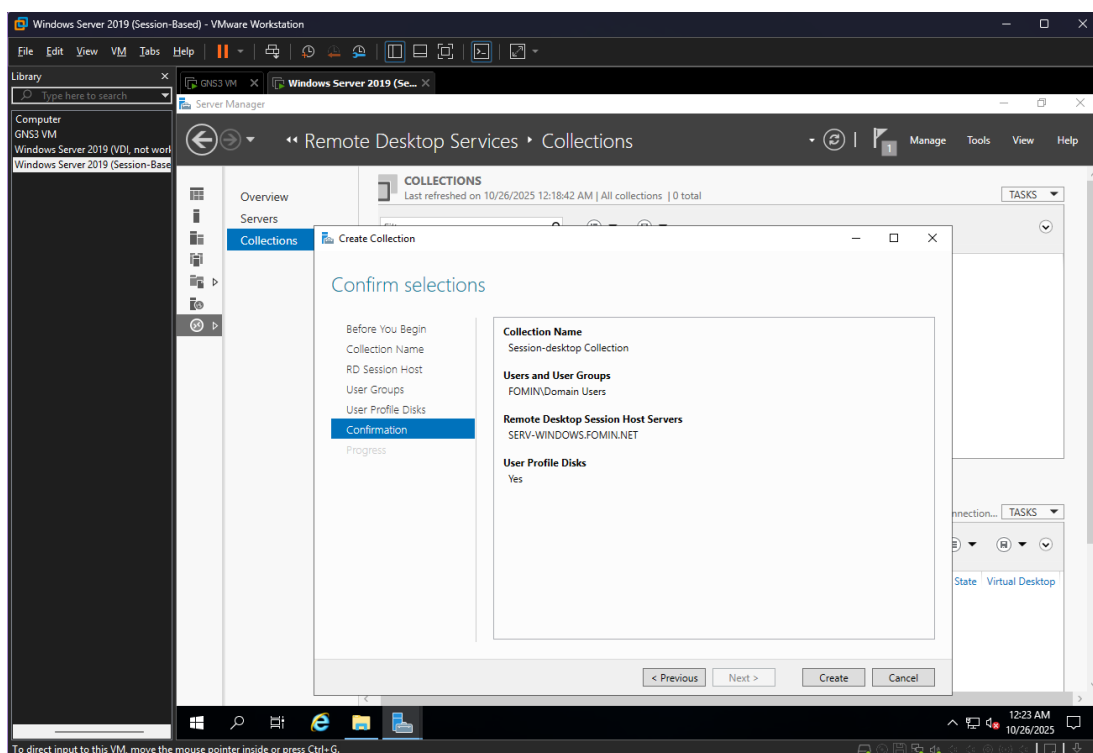


Рисунок 3.27 – Налаштування колекцій для RDS

Після цього для перевірки роботи RDP також варто створити тестових користувачів. Виконавши дані налаштування можна перейти до перевірки роботи віддаленого доступу. Створення тестових користувачів у Active Directory Users and Computers для перевірки RDP-доступу відображено на рисунку 3.28.

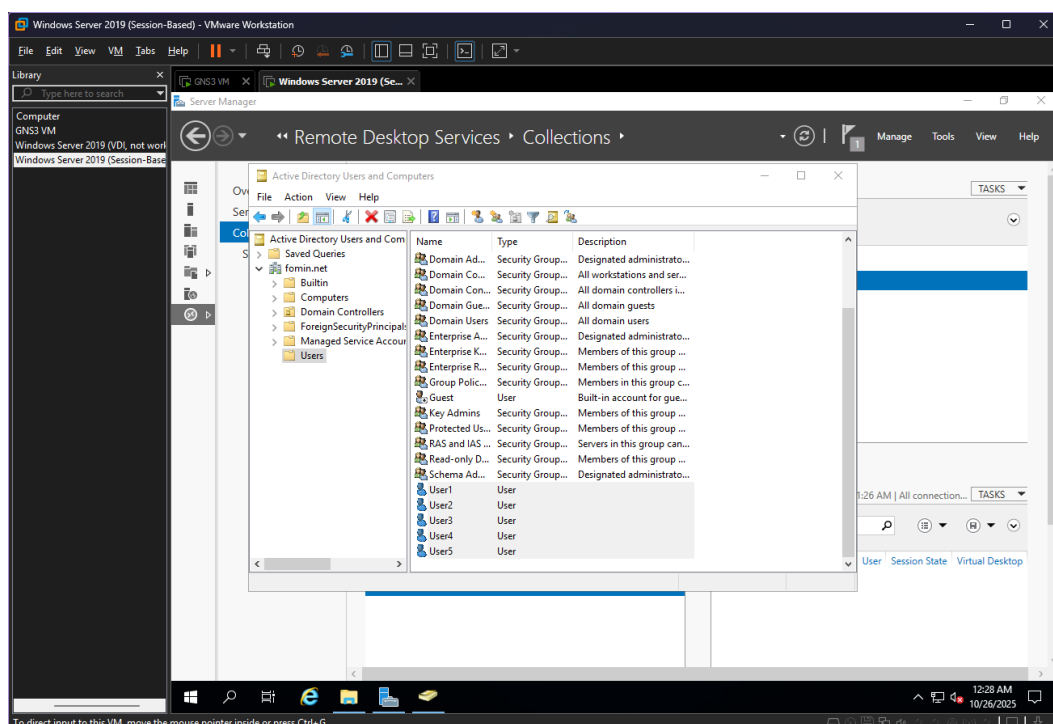


Рисунок 3.28 – Створення тестових користувачів

Так як на цьому етапі налаштування віртуальна машина ще хоститься на нашій робочій станції, а не в GNS3, перевірка підключень здійснюватиметься за допомогою утиліти з набору Sysinternals – Remote Desktop Connection Manager. На рисунку 3.29 продемонстровано успішну роботу декількох одночасних RDP-підключень до сервера.

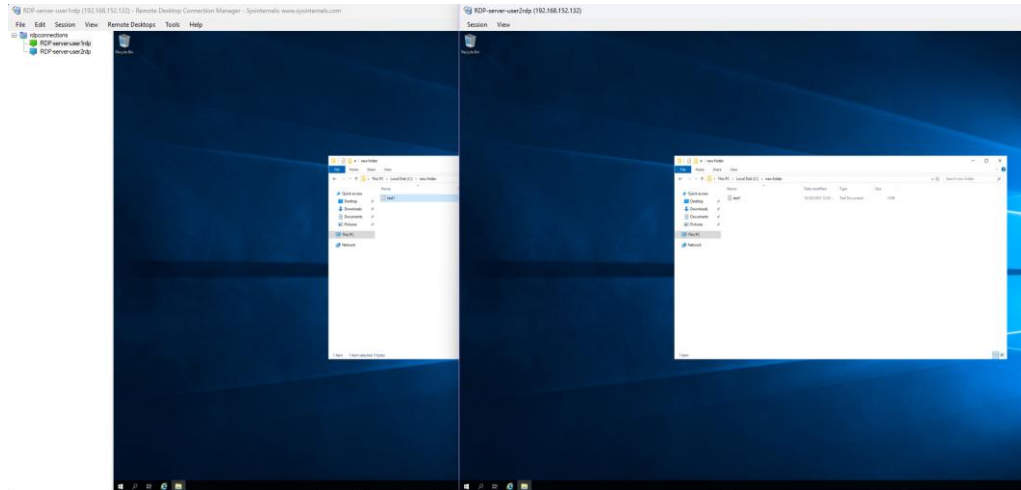


Рисунок 3.29 – Перевірка роботи RDP

Як можна побачити, підключення по RDP працює коректно і є можливість мати декілька одночасних підключень.

Після всіх налаштувань, необхідно змінити параметри IP-адресації на сервері та підключити його до мережі в GNS3. Зміна параметрів IP-адресації на Windows Server перед підключенням до мережі GNS3 показана на рисунку 3.30.

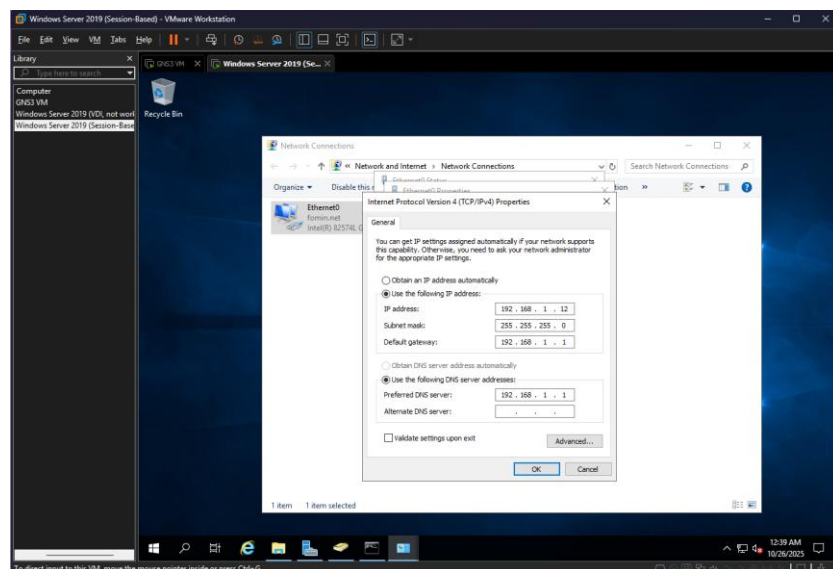


Рисунок 3.30 – Зміна параметрів IP-адресації

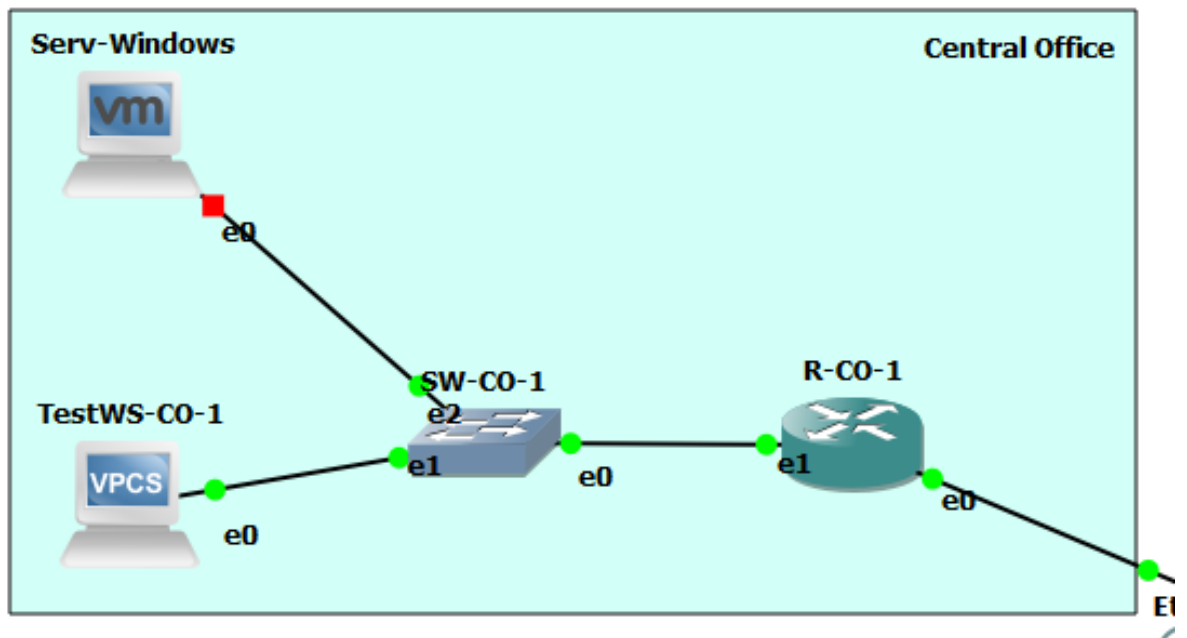


Рисунок 3.31 – Сервер з Windows додано до центрального офісу

3.4. Реалізація віддаленого доступу через VNC

Тепер можна переходити до реалізації VNC. Для початку встановимо TigerVNC на Ubuntu Server. Це буде “bare-minimum” інсталяція, за необхідності на сервері можна налагодити будь які інші необхідні пакунки.

Для VNC-серверу всі необхідні пакунки встановлюються командою “`sudo apt install tigervnc-standalone-server tigervnc-common`”. Для підключення до VNC-серверу необхідно встановити `tigervnc-viewer`.

Так як на сервері немає ніякого встановленого графічного інтерфейсу, варто ще встановити і його, наприклад `xfce`, встановлення якого здійснюється командою “`sudo apt install dbus-x11 xfce4 xfce4-goodies`”.

Також тепер варто створити користувачів командою “`sudo adduser user1`”. На рисунку 3.32 показано встановлення необхідних пакунків та створення користувачів на `Serv-Ubuntu-Session`. Встановлення паролів для доступу до VNC для користувачів `Serv-Ubuntu-Session` відображено на рисунку 3.33.

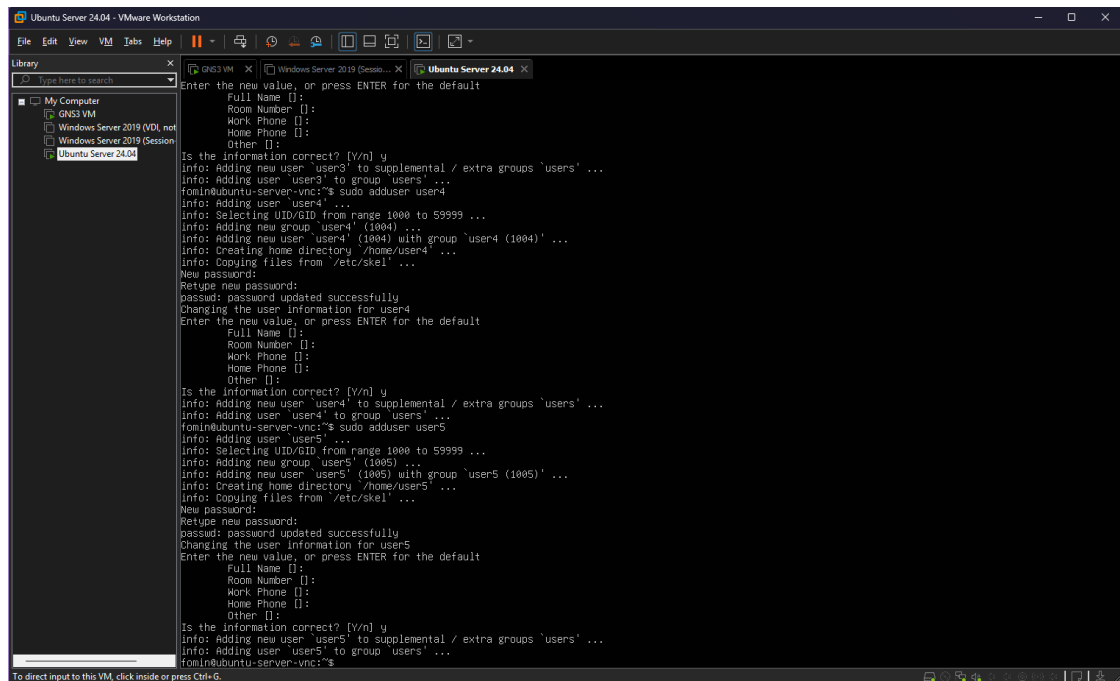


Рисунок 3.32 – Встановлення всіх пакунків та користувачів

Для всіх користувачів також необхідно встановити пароль для доступу до VNC. Робиться це командою “vncpasswd” виконаною з-під профілю користувача.

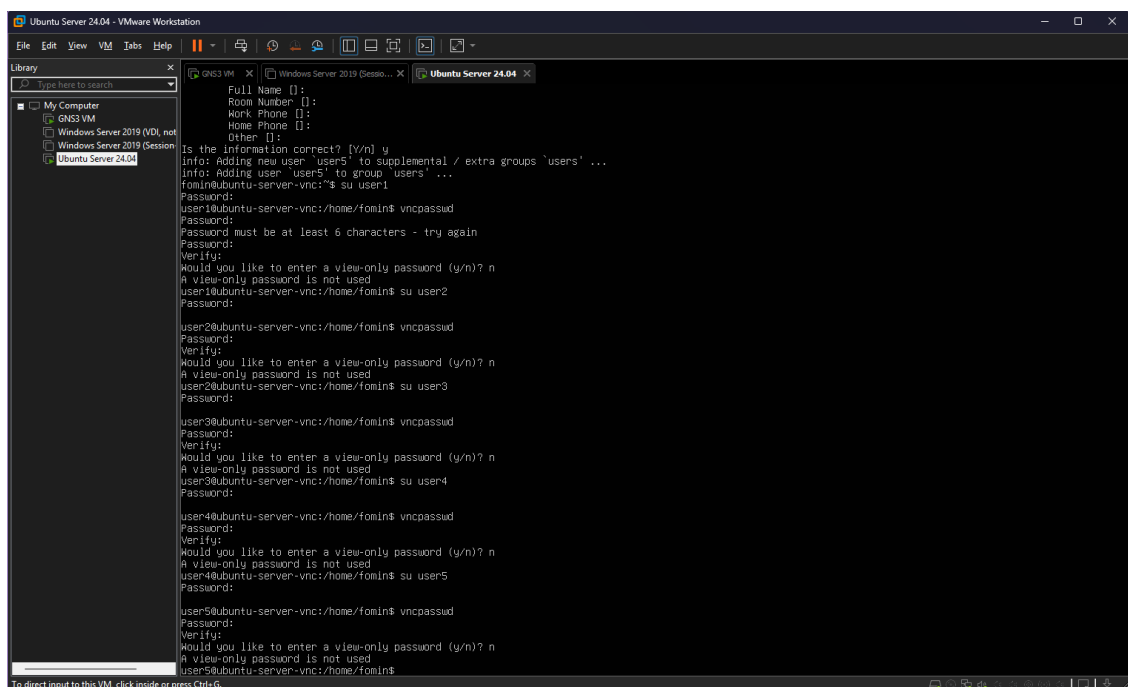


Рисунок 3.33 – Встановлення паролів VNC для користувачів

Тепер, кожному користувачу потрібно запустити його VNC-сервер. Для цього ще окремо треба налаштувати скрипт, який запускатиме наш графічний інтерфейс для користувача, коли він підключатиметься по VNC. Також

необхідно зробити даний файл виконуваним командою “chmod +x /home/user1/.vnc/xstartup”. Налаштування скрипта xstartup для запуску графічного інтерфейсу XFCE для VNC-сесії показано на рисунку 3.34.

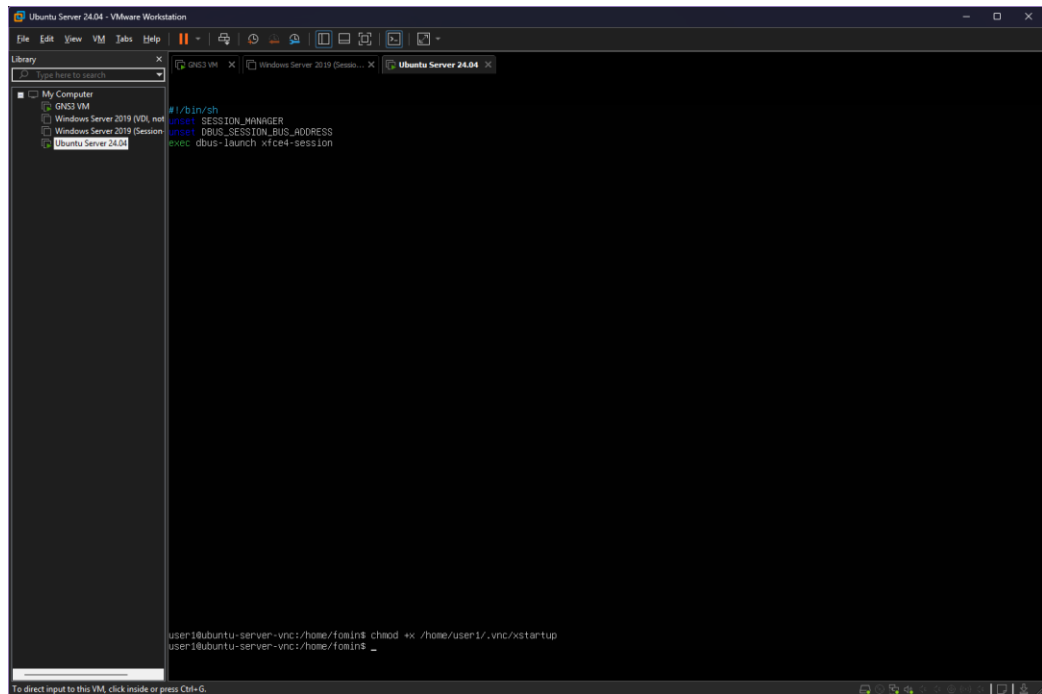


Рисунок 3.34 – Налаштування скрипта запуску графічного інтерфейсу

Запуск серверу виконується командою “vncserver :1 -localhost no”. Де “:1” – номер дисплею, а “-localhost no” дозволяє підключення не тільки з локальної мережі. Запуск VNC-серверу для першого користувача на Serv-Ubuntu-Session відображено на рисунку 3.35. Успішне підключення та робота графічного середовища через VNC на Serv-Ubuntu-Session продемонстровано на рисунку 3.36.

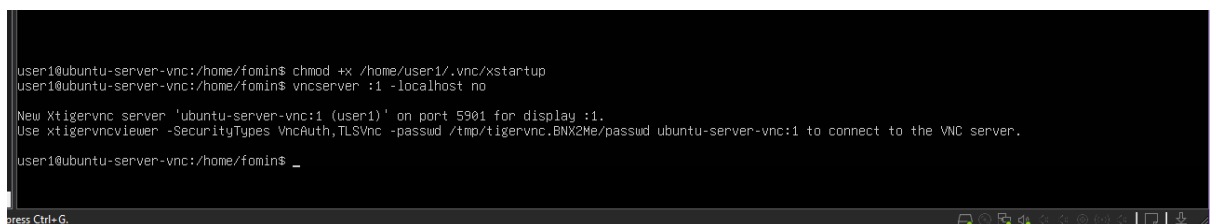


Рисунок 3.35 – Запуск VNC-серверу для користувача user1

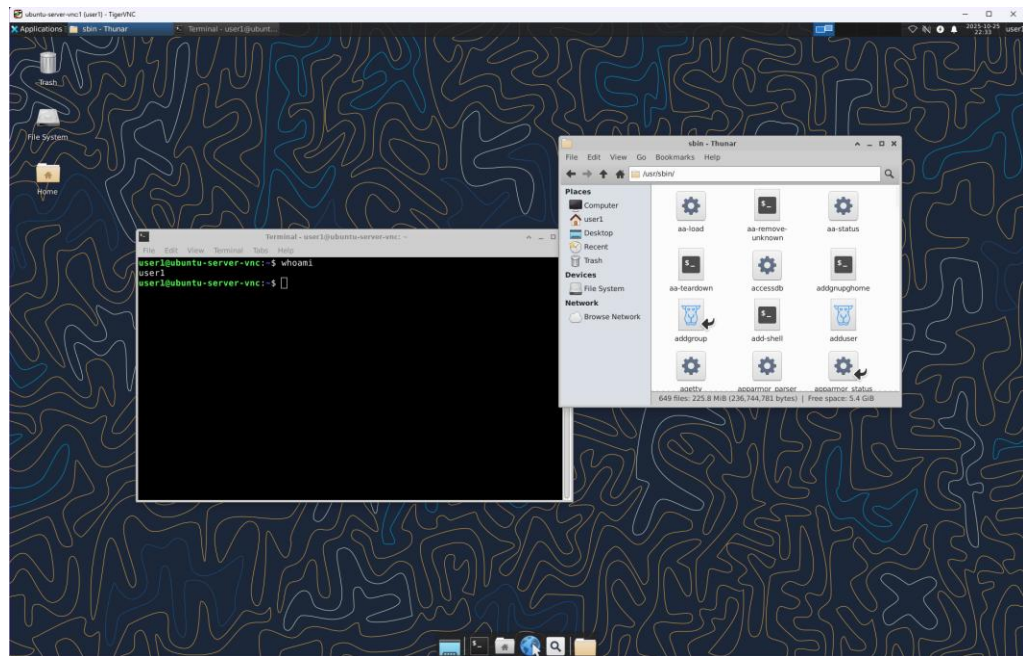


Рисунок 3.36 – VNC успішно працює

Тепер для повноцінної реалізації серверу віддаленого доступу необхідно автоматизувати запуск VNC-серверу для кожного користувача при кожному запуску системи. Для цього необхідно налаштувати systemd за допомогою шаблонів. Створення шаблону systemd-служби `vncserver@.service` для автоматичного запуску VNC-сервера відображено на рисунку 3.37.

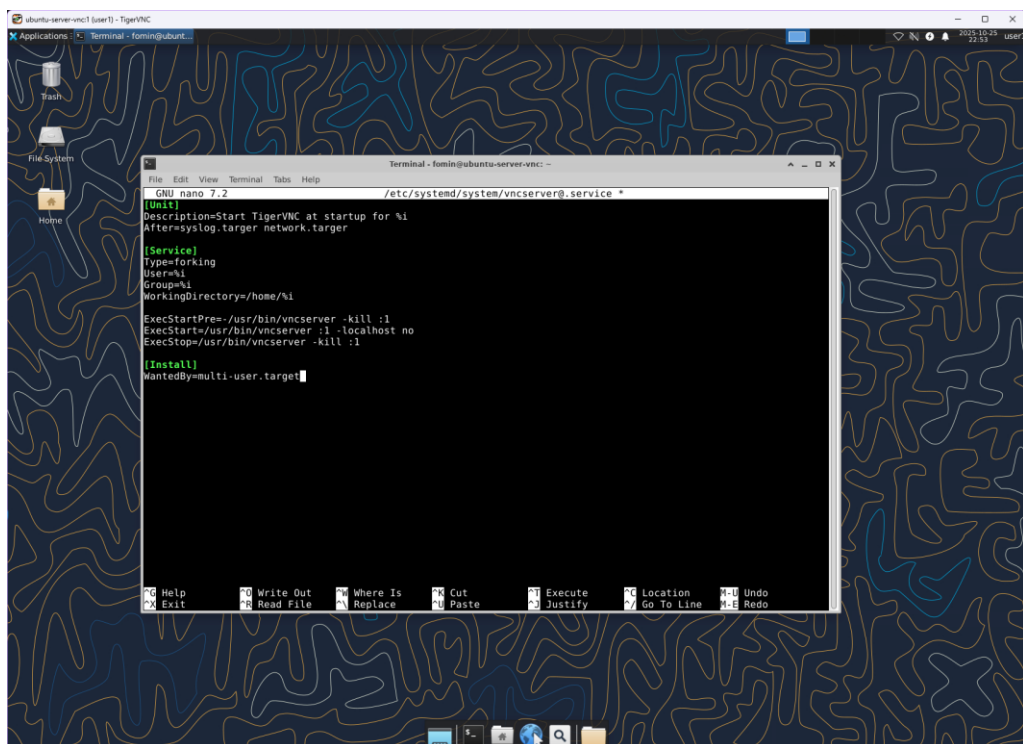


Рисунок 3.37 – Створення шаблону для systemd

Після цього необхідно перезапустити systemctl і ввімкнути створену службу. Увімкнення та перезавантаження systemd для активації VNC-сервісу показано на рисунку 3.38.

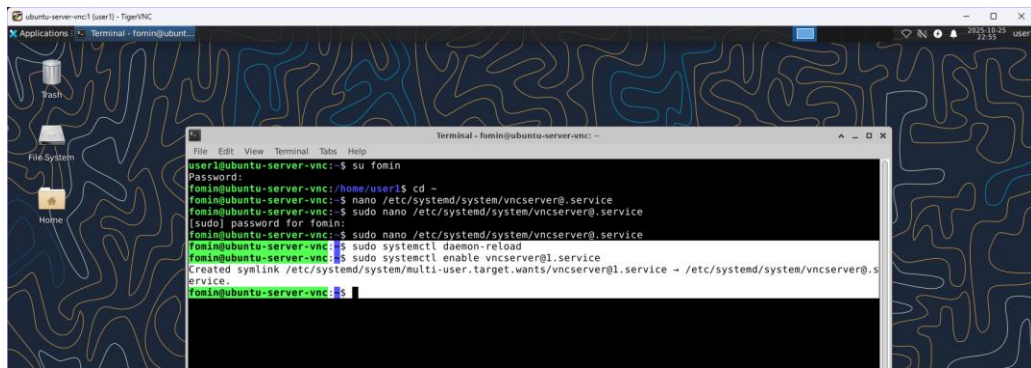


Рисунок 3.38 – Створення шаблону для systemd

Одна з переваг використання Linux – це те, як легко можна автоматизувати всі процеси за допомогою скриптів, тому далі створення служб для всіх користувачів здійснюватиметься скриптом full-setup.sh. Автоматизоване налаштування VNC-серверів та systemd-служб для всіх 5 користувачів за допомогою скрипта показано на рисунку 3.39. На рисунку 3.40 відображено успішний запуск браузера Firefox через VNC-підключення

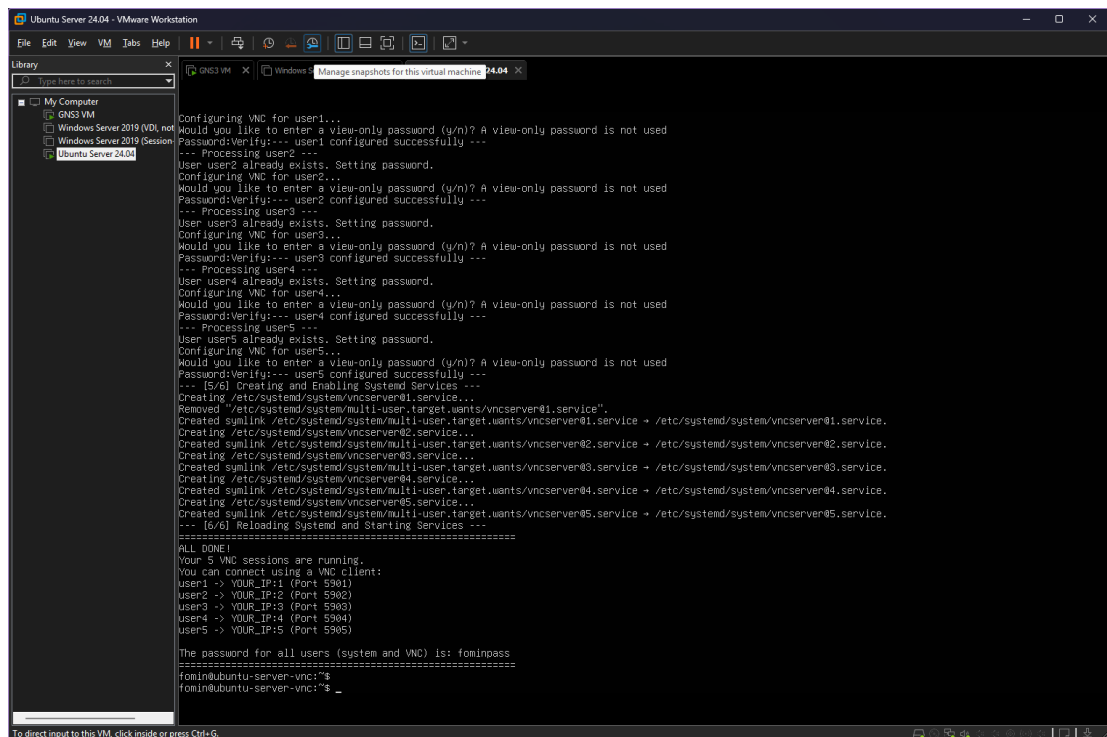


Рисунок 3.39 – Виконання решти налаштувань для решти користувачів

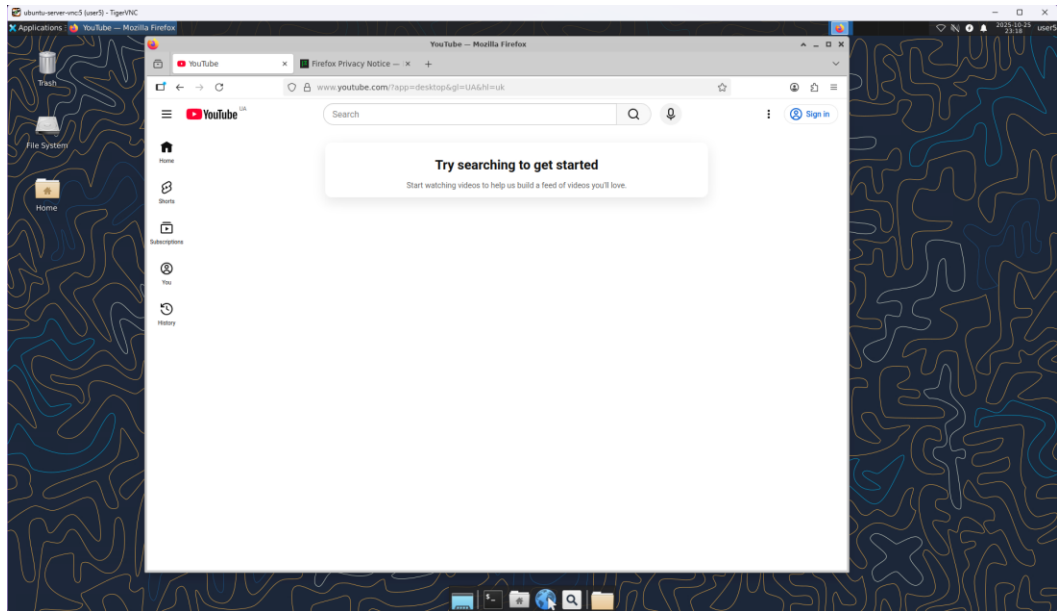


Рисунок 3.40 – Налаштування успішні

Тепер коли ми переконалися що все працює, змінимо IP конфігурацію і підключимо сервер до мережі. Налаштування IP-адресації для Serv-Ubuntu-Session перед підключенням до мережі GNS3 відображено на рисунку 3.41.

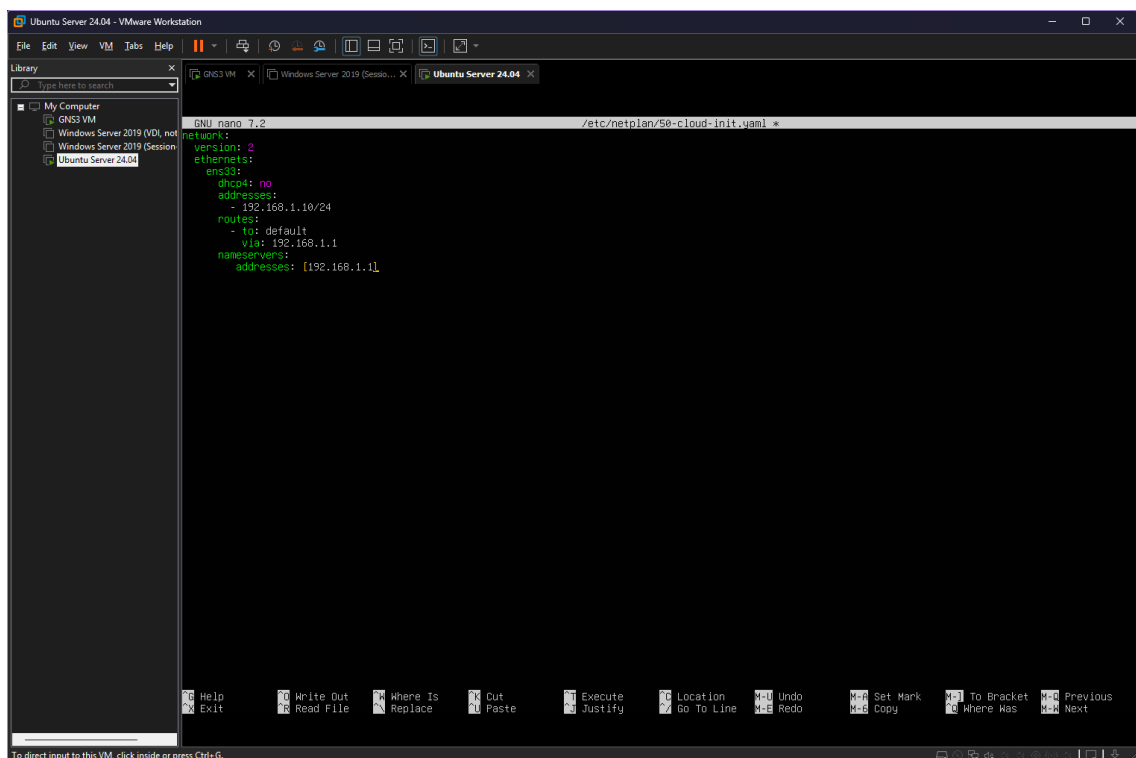


Рисунок 3.41 – Налаштування IP-адресації

Після всіх налаштувань варто нагадати, що у такого підходу такі ж мінуси як і у RDP з Session based desktop environment – кожен користувач “ділить” одну

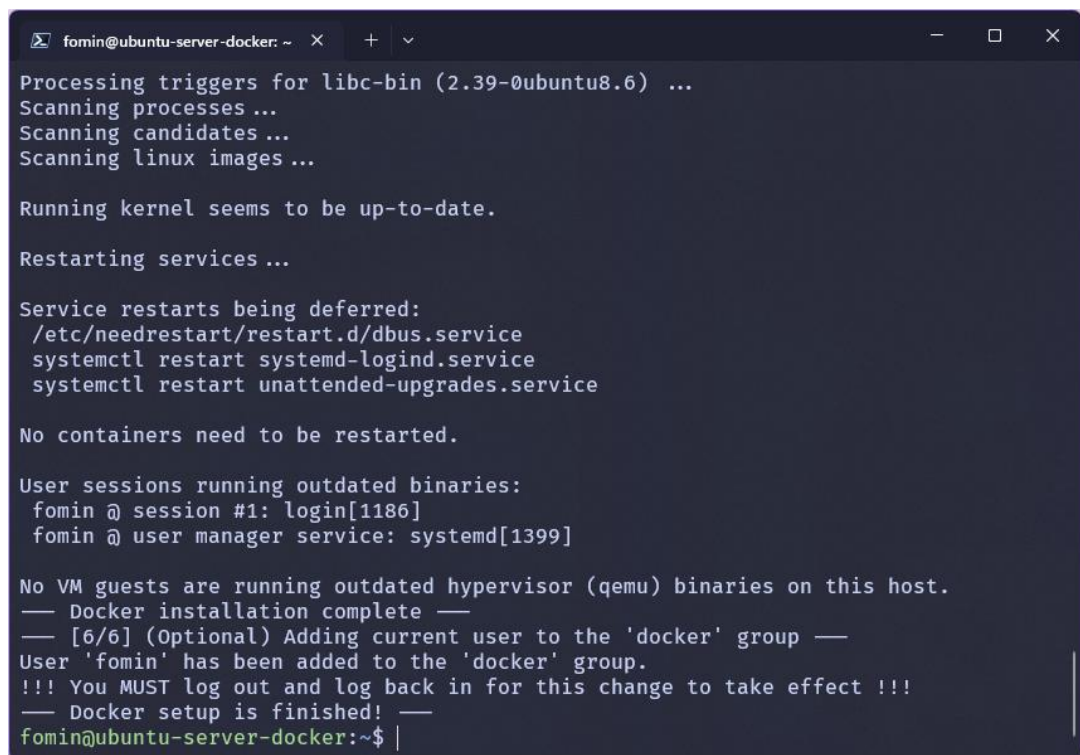
машину і не є ізольованим одним від одного. У випадку з RDP це вимушене рішення, адже це проблема віртуалізованого середовища, а у випадку з Linux можна здійснити іншу реалізацію VNC-серверу, яка більш подібна до VDI.

3.5. Організація ізольованих VNC-середовищ у Docker

Ця реалізація базується на використанні Docker-контейнерів на основі образу Linux з встановленим VNC-сервером. Для такої реалізації можна або створити свій або використати вже існуючий образ Linux з VNC та графічним інтерфейсом. Для тестування такої реалізації VNC-серверу, було створено власний Docker-образ на основі Ubuntu 24.04, який розгортатиме контейнери з встановленим графічним інтерфейсом та VNC-сервером.

Для початку необхідно встановити сам Docker та Docker compose. Його встановлення здійснюється таким набором команд. Для простішого та швидшого налаштування все встановлення здійснюватиметься скриптами.

Лістинг скрипту setup-docker.sh для встановлення Docker та Docker compose наведено в додатках. Успішне виконання скрипту встановлення Docker та додавання користувача до групи docker показано на рисунку 3.42.



```
fomin@ubuntu-server-docker: ~  
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...  
Scanning processes ...  
Scanning candidates ...  
Scanning linux images ...  
  
Running kernel seems to be up-to-date.  
  
Restarting services ...  
  
Service restarts being deferred:  
/etc/needrestart/restart.d/dbus.service  
systemctl restart systemd-logind.service  
systemctl restart unattended-upgrades.service  
  
No containers need to be restarted.  
  
User sessions running outdated binaries:  
fomin @ session #1: login[1186]  
fomin @ user manager service: systemd[1399]  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
— Docker installation complete —  
— [6/6] (Optional) Adding current user to the 'docker' group —  
User 'fomin' has been added to the 'docker' group.  
!!! You MUST log out and log back in for this change to take effect !!!  
— Docker setup is finished! —  
fomin@ubuntu-server-docker:~$
```

Рисунок 3.42 – Скрипт відпрацював успішно

Тепер коли Docker встановлено, перейдемо до створення необхідного образу. Для цього нам треба свій Dockerfile, лістинг у додатках.

У цьому Dockerfile описується, як зібрати наш образ. Він базується на ubuntu:24.04 і встановлює ті самі компоненти, що на іншому Ubuntu Server (XFCE, TigerVNC, non-Snap Firefox, dbus), а також supervisor та wazuh-agent.

Передбачено встановлення Wazuh agent для підтримки майбутнього моніторингу. Для стартових налаштувань образу також додано скрипт entrypoint.sh, лістинг у додатках.

Цей скрипт запускається один раз при старті контейнера. Його завдання – налаштувати паролі та IP-адреси, які ми передамо через docker-compose.yml, а потім запустити supervisor.

Supervisor – це програма, яка запускає та моніторить кілька процесів. Ми використовуємо її для запуску wazuh-agent (від root) та vncserver (від vncuser).

Останнім необхідним компонентом налаштування даної системи є створення docker-compose.yml файлу, лістинг у додатках.

Це фінальний файл, який збирає все до купи. Він запустить 5 сервісів (vnc-host1...vnc-host5), кожен з яких:

1. Збирається з нашого Dockerfile.
2. Використовує маппінг портів (5901 -> 5901, 5902 -> 5901, і т.д.).
3. Передає унікальний пароль та IP-адресу Wazuh-менеджера.

У результаті цей файл дозволяє легко та централізовано розгорнути два незалежні VNC-середовища, що базуються на раніше створеному образу з Dockerfile.

Запуск контейнерів здійснюватиметься командою “sudo docker compose up –build -d”. Успішний запуск Docker-контейнерів для ізольованих VNC-сесій відображено на рисунку 3.43. Успішне підключення до VNC-сесії в ізольованому Docker-контейнері продемонстровано на рисунку 3.44.

```
fomin@ubuntu-server-docker: ~$ docker build -t fomin-vnc-host4 .
=> writing image sha256:1d3aadb400c92c568b55fa472c086645fc7d37d9b01cec989cf6285cdb07630d 0.0s
=> naming to docker.io/library/fomin-vnc-host4 0.0s
=> [vnc-host2] exporting to image 0.0s
=> exporting layers 0.0s
=> writing image sha256:7b4728d262062364ae4e6b0642e137e4994b0342d02c71a78ea3e91b89d88fc0 0.0s
=> naming to docker.io/library/fomin-vnc-host2 0.0s
=> [vnc-host5] resolving provenance for metadata file 0.0s
=> [vnc-host3] resolving provenance for metadata file 0.0s
=> [vnc-host2] resolving provenance for metadata file 0.0s
=> [vnc-host4] resolving provenance for metadata file 0.0s
=> [vnc-host1] resolving provenance for metadata file 0.0s
[+] Running 11/11
✔ fomin-vnc-host4 Built 0.0s
✔ fomin-vnc-host5 Built 0.0s
✔ fomin-vnc-host1 Built 0.0s
✔ fomin-vnc-host2 Built 0.0s
✔ fomin-vnc-host3 Built 0.0s
✔ Network fomin_default Created 0.0s
✔ Container vnc-host1 Started 0.8s
✔ Container vnc-host4 Started 0.6s
✔ Container vnc-host3 Started 0.8s
✔ Container vnc-host5 Started 0.8s
✔ Container vnc-host2 Started 0.6s
fomin@ubuntu-server-docker:~$ docker logs vnc-host1
Setting VNC password ...
Would you like to enter a view-only password (y/n)? A view-only password is not used
Password:Verify:Configuring Wazuh Agent to connect to 192.168.1.20 ...
Ensuring correct permissions for /home/vncuser ...
Starting Supervisor ...
fomin@ubuntu-server-docker:~$ docker logs vnc-host1
Setting VNC password ...
Would you like to enter a view-only password (y/n)? A view-only password is not used
Password:Verify:Configuring Wazuh Agent to connect to 192.168.1.20 ...
Ensuring correct permissions for /home/vncuser ...
Starting Supervisor ...
2025-10-26 00:37:48,568 CRIT Supervisor is running as root. Privileges were not dropped because no user is specified in the config file. If you intend to run as root, you can set user=root in the config file to avoid this message.
2025-10-26 00:37:48,568 INFO Included extra file "/etc/supervisor/conf.d/supervisord.conf" during parsing
2025-10-26 00:37:48,572 INFO RPC interface 'supervisor' initialized
2025-10-26 00:37:48,572 CRIT Server 'unix_http_server' running without any HTTP authentication checking
2025-10-26 00:37:48,574 INFO supervisord started with pid 1
2025-10-26 00:37:49,574 INFO spawned: 'vnc' with pid 51
2025-10-26 00:37:49,580 INFO spawned: 'wazuh' with pid 52
fomin@ubuntu-server-docker:~$ docker logs vnc-host1
Setting VNC password ...
Would you like to enter a view-only password (y/n)? A view-only password is not used
Password:Verify:Configuring Wazuh Agent to connect to 192.168.1.20 ...
Ensuring correct permissions for /home/vncuser ...
```

Рисунок 3.43 – Запуск docker-контейнерів

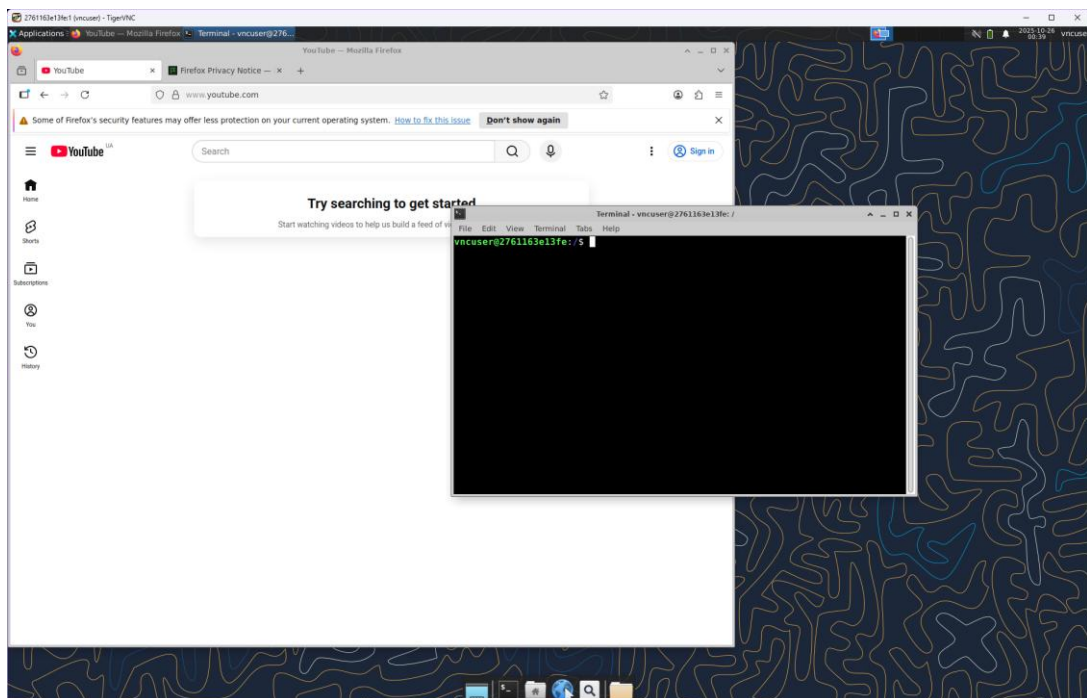
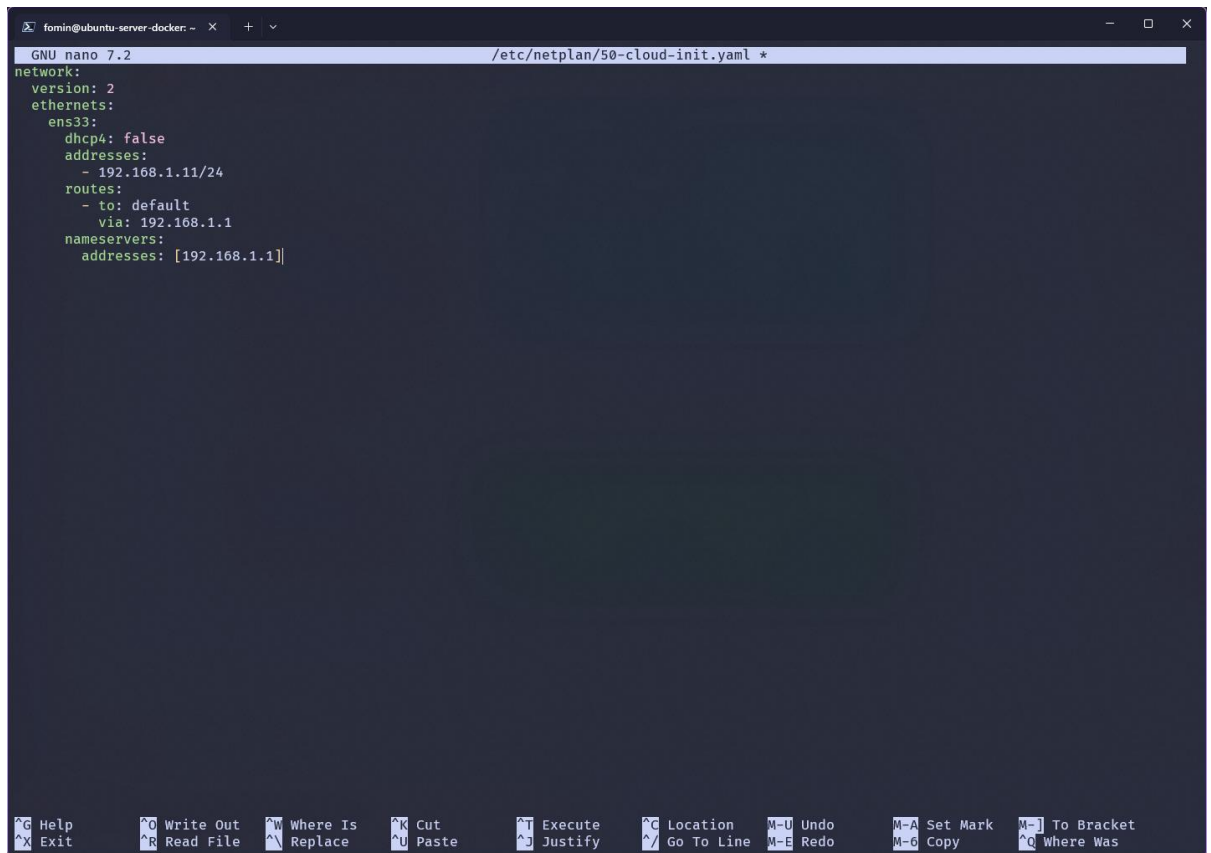


Рисунок 3.44 – Підключення до контейнера успішне

На цьому налаштування VDI на базі Docker завершено. Змінюємо IP-адресацію і переносимо сервер до мережі в GNS3. Налаштування IP-адресації для Serv-Ubuntu-Docker перед інтеграцією в мережу GNS3 відображено на рисунку 3.45.



The screenshot shows a terminal window titled 'fomin@ubuntu-server-docker: ~'. The user is editing the file '/etc/netplan/50-cloud-init.yaml' using the 'GNU nano 7.2' editor. The file contains the following YAML configuration:

```
network:
  version: 2
  ethernet:
    ens33:
      dhcp4: false
      addresses:
        - 192.168.1.11/24
      routes:
        - to: default
          via: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1]
```

The terminal window has a dark background and a light blue title bar. The bottom of the window shows a menu bar with various keyboard shortcuts for nano editor functions like Help, Exit, Write Out, Read File, etc.

Рисунок 3.45 – Налаштування IP-адресації для серверу з Docker

3.6. Реалізація системи моніторингу Wazuh

Наступним кроком буде реалізація серверу моніторингу та підключення наших пристроїв до нього. В якості серверу моніторингу використовуватиметься Ubuntu Server з встановленим Wazuh.

Для початку треба зазначити, що для Wazuh треба приділити більше вільного місця ніж зазвичай виділяється при встановленні Ubuntu Server: краще всього виділити хоча б близько 50 ГБ замість звичайних 20 ГБ.

Далі встановлення Wazuh здійснюється командою “curl -sO https://packages.wazuh.com/4.14/wazuh-install.sh && sudo bash ./wazuh-install.sh -a”. На рисунку 3.46. продемонстровано процес встановлення Wazuh, а на рисунку 3.47 показано панель Wazuh Dashboard після першого входу. Рисунок 3.48 відображає налаштування IP-адресації для SIEM-серверу перед його додаванням до GNS3.

```
fomin@ubuntu-siem: ~$ ./wazuh-install.sh
./wazuh-install.sh: line 1: "<?xml version='1.0' encoding='UTF-8'?"
fomin@ubuntu-siem:~$ curl -sO https://packages.wazuh.com/4.x/CzuH-install.sh 66 sudo bash ./wazuh-install.sh -a
fomin@ubuntu-siem:~$ curl -sO https://packages.wazuh.com/4.14/wazuh-install.sh 66 sudo bash ./wazuh-install.sh -a
26/10/2025 15:41:34 INFO: Starting Wazuh installation assistant. Wazuh version: 4.14.0
26/10/2025 15:41:34 INFO: Verbose logging redirected to /var/log/wazuh-install.log
26/10/2025 15:41:38 INFO: Verifying that your system meets the recommended minimum hardware requirements.
26/10/2025 15:41:38 INFO: Wazuh web interface port will be 443.
26/10/2025 15:41:42 INFO: --- Dependencies ---
26/10/2025 15:41:42 INFO: Installing apt-transport-https.
26/10/2025 15:41:47 INFO: Installing debhelper.
26/10/2025 15:42:01 INFO: Wazuh repository added.
26/10/2025 15:42:01 INFO: --- Configuration files ---
26/10/2025 15:42:01 INFO: Generating configuration files.
26/10/2025 15:42:01 INFO: Generating the root certificate.
26/10/2025 15:42:01 INFO: Generating Admin certificates.
26/10/2025 15:42:02 INFO: Generating Wazuh indexer certificates.
26/10/2025 15:42:02 INFO: Generating Filebeat certificates.
26/10/2025 15:42:02 INFO: Generating Wazuh dashboard certificates.
26/10/2025 15:42:02 INFO: Created wazuh-install-files.tar. It contains the Wazuh cluster key, certificates, and passwords necessary for installation.
26/10/2025 15:42:03 INFO: --- Wazuh indexer ---
26/10/2025 15:42:03 INFO: Starting Wazuh indexer installation.
26/10/2025 15:43:25 INFO: Wazuh indexer installation finished.
26/10/2025 15:43:25 INFO: Wazuh indexer post-install configuration finished.
26/10/2025 15:43:25 INFO: Starting service wazuh-indexer.
26/10/2025 15:43:36 INFO: wazuh-indexer service started.
26/10/2025 15:43:36 INFO: Initializing Wazuh indexer cluster security settings.
26/10/2025 15:43:39 INFO: Wazuh indexer cluster security configuration initialized.
26/10/2025 15:43:39 INFO: Wazuh indexer cluster initialized.
26/10/2025 15:43:39 INFO: --- Wazuh server ---
26/10/2025 15:43:39 INFO: Starting the Wazuh manager installation.
26/10/2025 15:44:58 INFO: Wazuh manager installation finished.
26/10/2025 15:44:58 INFO: Wazuh manager vulnerability detection configuration finished.
26/10/2025 15:44:58 INFO: Starting service wazuh-manager.
26/10/2025 15:45:12 INFO: wazuh-manager service started.
26/10/2025 15:45:12 INFO: Starting Filebeat installation.
26/10/2025 15:45:18 INFO: Filebeat installation finished.
26/10/2025 15:45:18 INFO: Filebeat post-install configuration finished.
26/10/2025 15:45:18 INFO: Starting service filebeat.
26/10/2025 15:45:19 INFO: filebeat service started.
26/10/2025 15:45:19 INFO: --- Wazuh dashboard ---
26/10/2025 15:45:19 INFO: Starting Wazuh dashboard installation.
26/10/2025 15:46:13 INFO: Wazuh dashboard installation finished.
26/10/2025 15:46:13 INFO: Wazuh dashboard post-install configuration finished.
26/10/2025 15:46:13 INFO: Starting service wazuh-dashboard.
26/10/2025 15:46:13 INFO: wazuh-dashboard service started.
26/10/2025 15:46:16 INFO: Updating the internal users.
26/10/2025 15:46:20 INFO: A backup of the internal users has been saved in the /etc/wazuh-indexer/internalusers-backup folder.
26/10/2025 15:46:30 INFO: The filebeat.yml file has been updated to use the Filebeat Keystore username and password.
26/10/2025 15:46:57 INFO: Initializing Wazuh dashboard web application.
26/10/2025 15:46:58 INFO: Wazuh dashboard web application initialized.
26/10/2025 15:46:58 INFO: --- Summary ---
26/10/2025 15:46:58 INFO: You can access the web interface https://cwazuh-dashboard-ip:443
User: admin
Password: EEEEDPrAancBECylqPBLMHI+SdgZzpCT
26/10/2025 15:46:58 INFO: Installation finished.
fomin@ubuntu-siem:~$
```

Рисунок 3.46 – Встановлення Wazuh

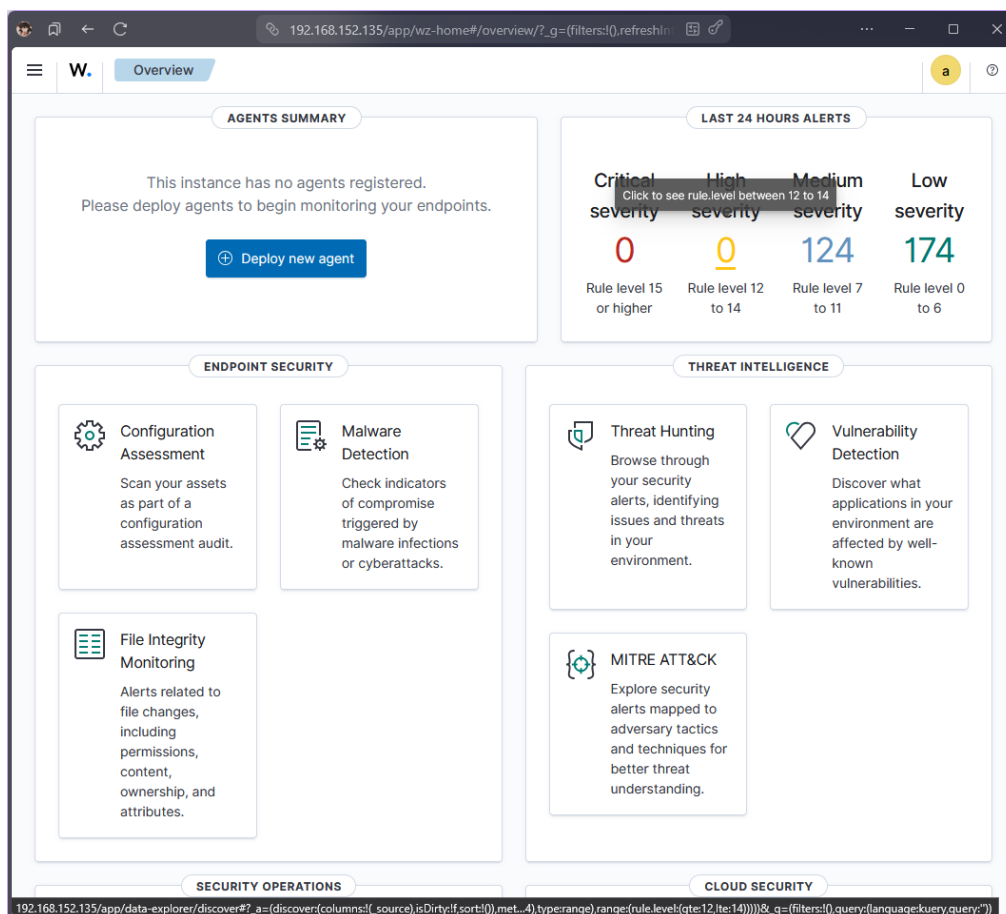


Рисунок 3.47 – Підключення до Wazuh Dashboard через браузер

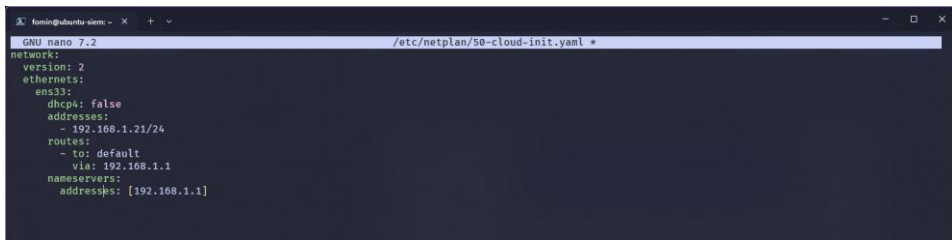


Рисунок 3.48 – Зміна параметрів IP-адресації для SIEM-серверу

На даному етапі встановлення Wazuh завершено. Для підключення клієнтів, на їх пристроях необхідно встановити Wazuh agent.

Для прикладу, ще на етапі створення та налаштування Docker-контейнерів, ми також встановили і налаштували Wazuh agent. І у самому Wazuh тепер можна побачити наші пристрої. На рисунку 3.49 показано відображення підключених клієнтів на Wazuh Dashboard.

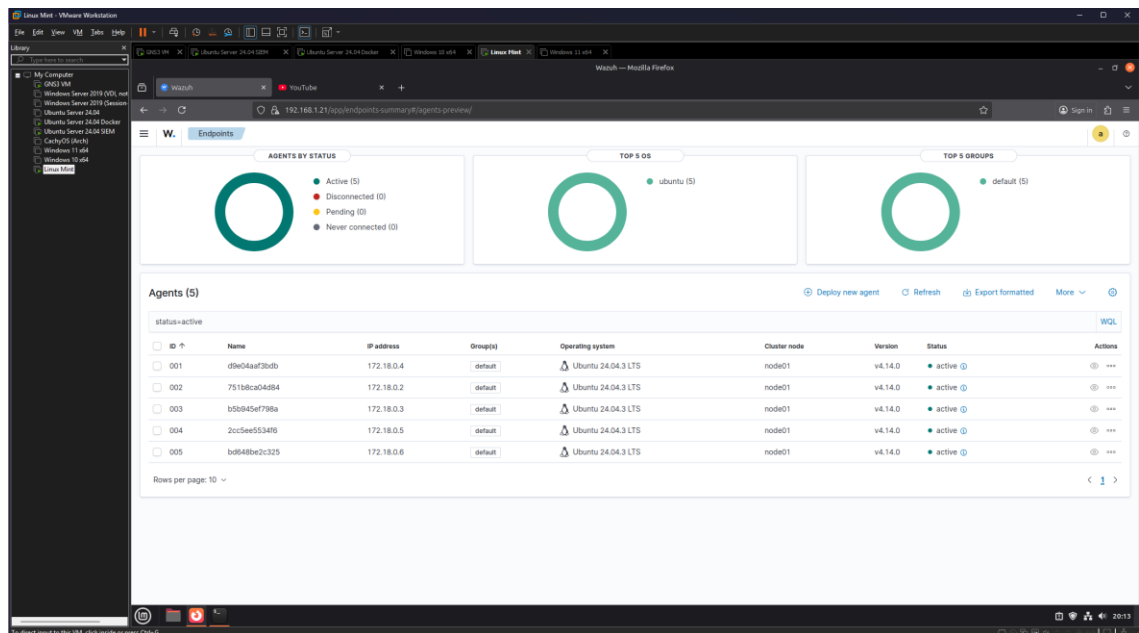


Рисунок 3.49 – Приклад відображення пристроїв у Wazuh

Встановимо ще Wazuh-agent на Serv-Ubuntu-Session. Встановлення виконується це в декілька етапів.

Спочатку додаємо репозиторій Wazuh командами:

```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --dearmor | sudo tee /usr/share/keyrings/wazuh.gpg > /dev/null
```

```
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee /etc/apt/sources.list.d/wazuh.list
```

sudo apt update

Після чого встановлюємо самого агента командою “WAZUH_MANAGER=”192.168.1.21” apt-get install wazuh-agent”. Встановлення Wazuh-agent на Serv-Ubuntu-Session за допомогою командного рядка показано на рисунку 3.50. На рисунку 3.51 відображено редагування конфігураційного файлу ossec.conf для налаштування Wazuh-agent на Serv-Ubuntu-Session. На рисунку 3.52 показано увімкнення та запуск Wazuh-agent як systemd-служби.

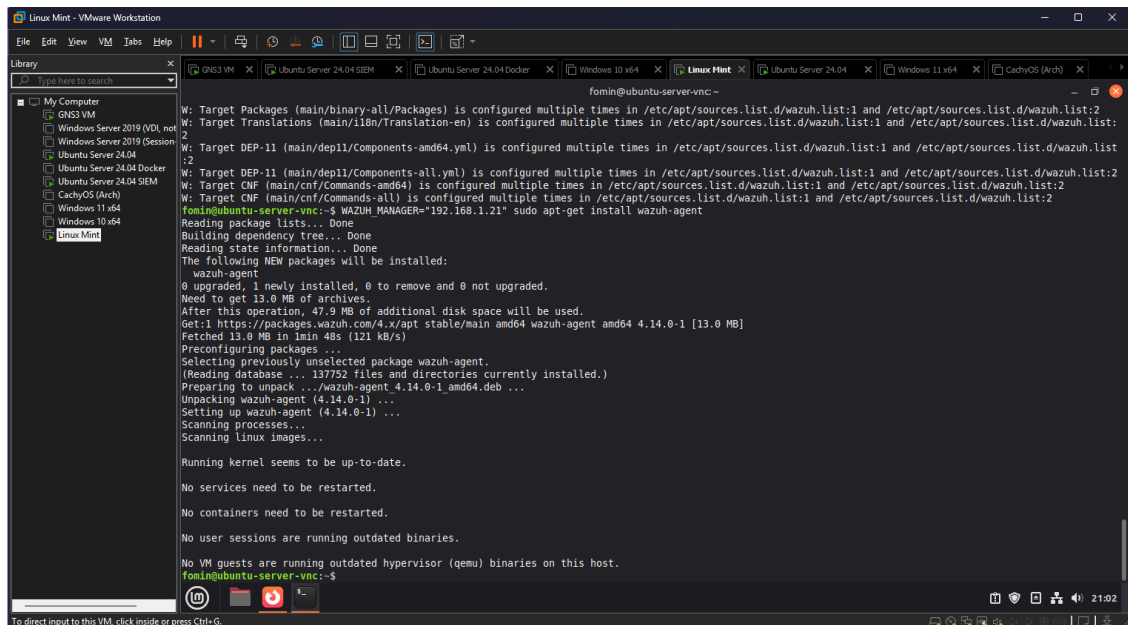


Рисунок 3.50 – Встановлення Wazuh-agent на Serv-Ubuntu-Session

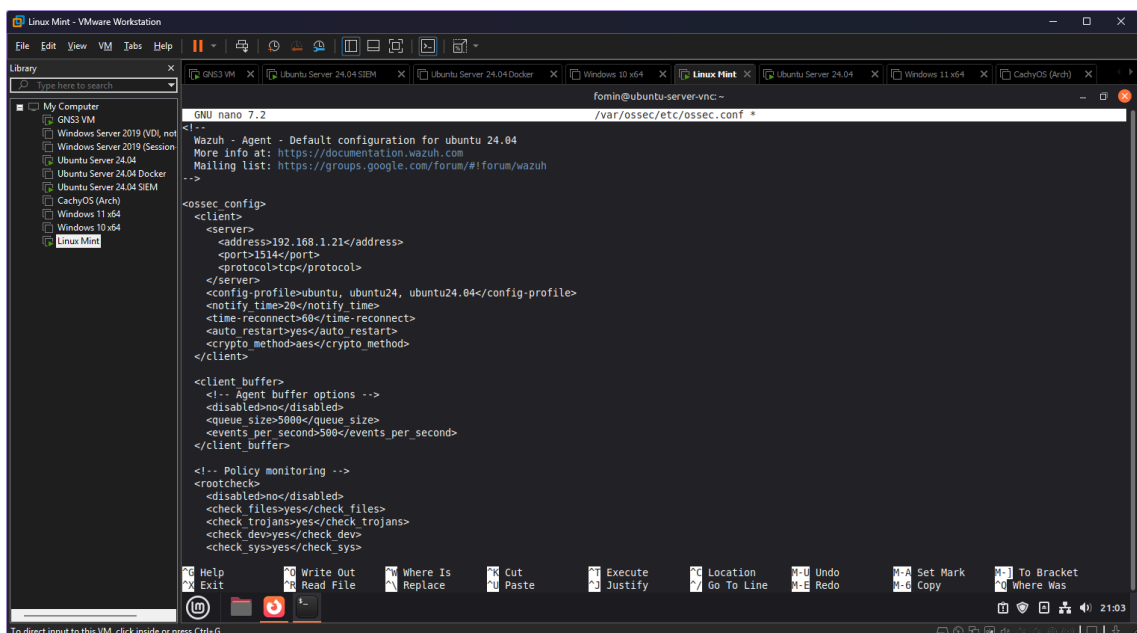


Рисунок 3.51 – Встановлення Wazuh-agent на Serv-Ubuntu-Session

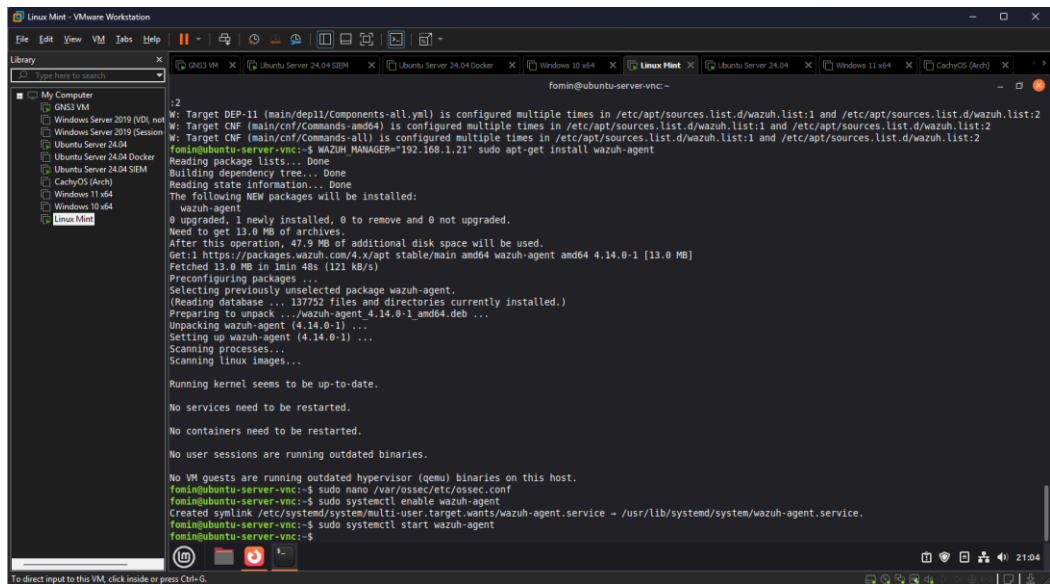


Рисунок 3.52 – Ввімкнення Wazuh-agent на Serv-Ubuntu-Session

На цьому етапі конфігурації Wazuh-agent на даному сервері завершена: можна переходити до наступного пристрою

Для Windows необхідно завантажити wazuh-agent.msi з офіційного сайту Wazuh. Далі у встановленому Wazuh agent потрібно встановити IP-адресу Wazuh-сервера. На цьому етапі налаштування клієнта на його пристрої завершено. Налаштування Wazuh Agent на Windows Server 2019 шляхом встановлення IP-адреси Wazuh-сервера відображено на рисунку 3.53.

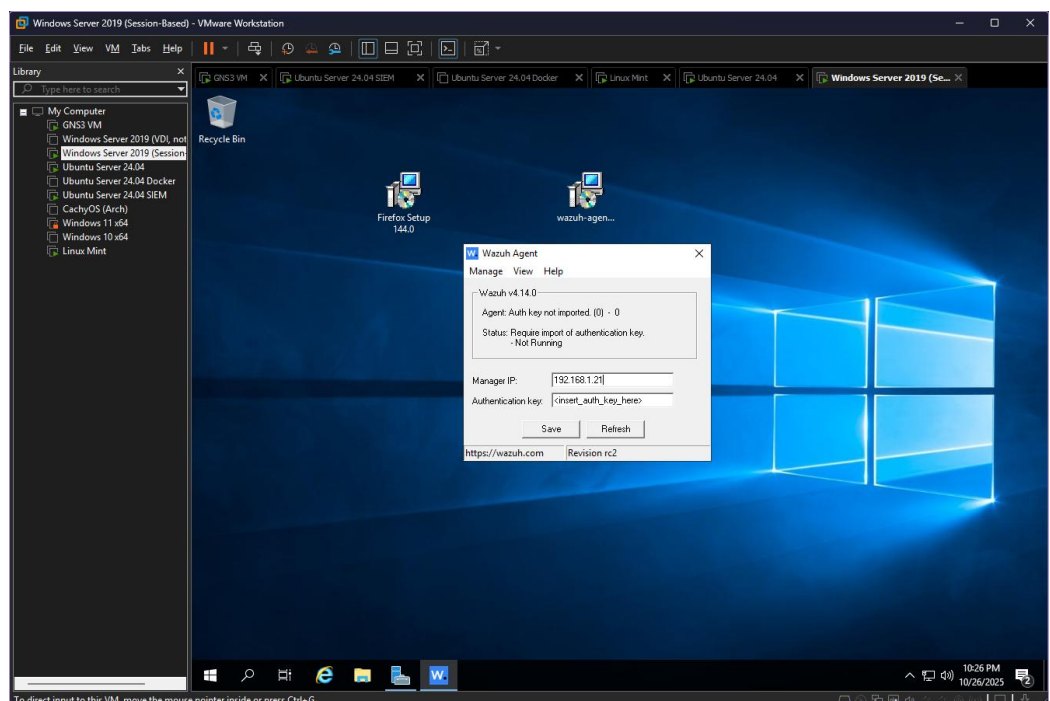


Рисунок 3.53 – Підключення клієнта до Wazuh

Після цих налаштувань можна побачити, що усі клієнти успішно відображаються на Wazuh Dashboard. На рисунку 3.54 показано, що усі підключені клієнти успішно відображаються на Wazuh Dashboard.

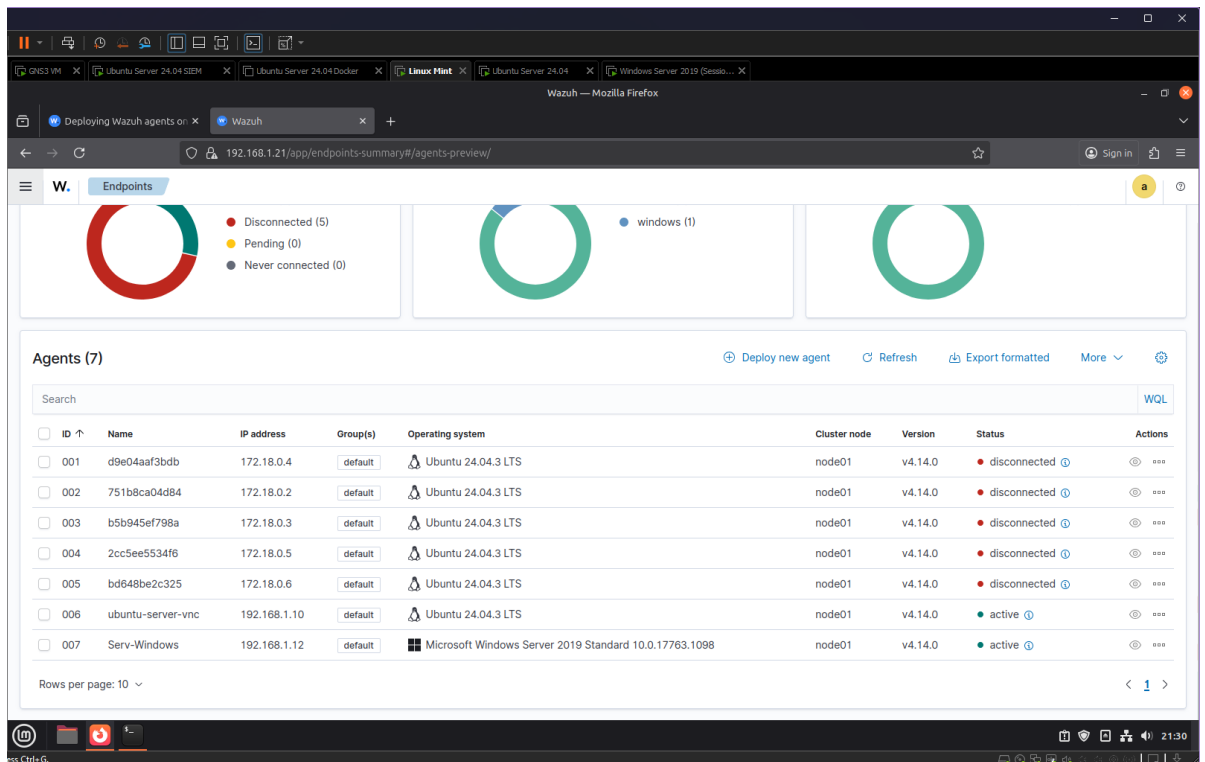


Рисунок 3.54 – Перегляд Wazuh Dashboard з усіма клієнтами

Тепер варто налаштувати власні правила, аби адаптувати Wazuh до наших потреб. Для цього необхідно створити файл з правилами, у якому ми будемо пропишемо правила моніторингу, що збиратимуть такі дані:

1. Успішні RDP-логіни.
2. Невдалі RDP-логіни.
3. Декілька невдалих RDP-логінів поспіль.
4. Успішних VNC-підключень.
5. Невдалих VNC-підключень.

Для використовуватиметься набір правил з файлу custom-rules.xml, його лістинг наведено у додатках. На рисунку 3.55 відображено додавання власних правил безпеки custom-rules.xml до Wazuh.

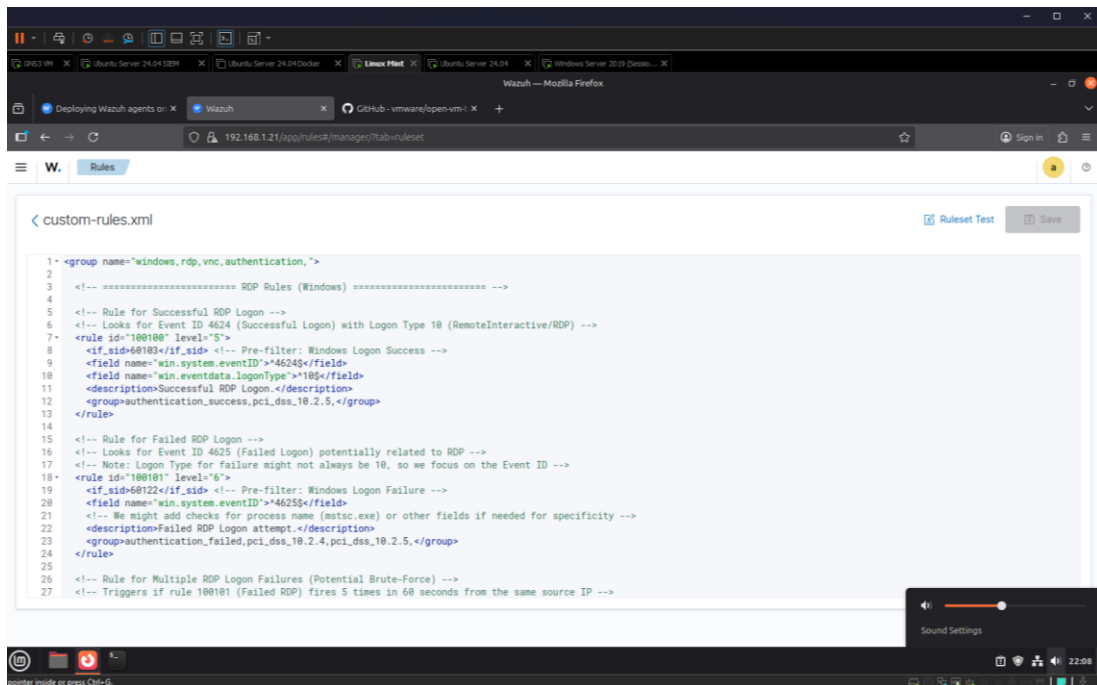


Рисунок 3.55 – Додавання власних правил до Wazuh Dashboard

Для того щоб здійснювати моніторинг логів VNC-серверу необхідно додати додаткову конфігурацію у кожен локальний agent.conf наших пристроїв. Однак аби не редагувати кожен файл окремо, це можна зробити прямо у Wazuh Dashboard. Для цього заходимо до групи до якої належать наші пристрої і там налаштовуємо загальний agent.conf. Додавання загальної конфігурації agent.conf для групи агентів через Wazuh Dashboard показано на рисунку 3.56.

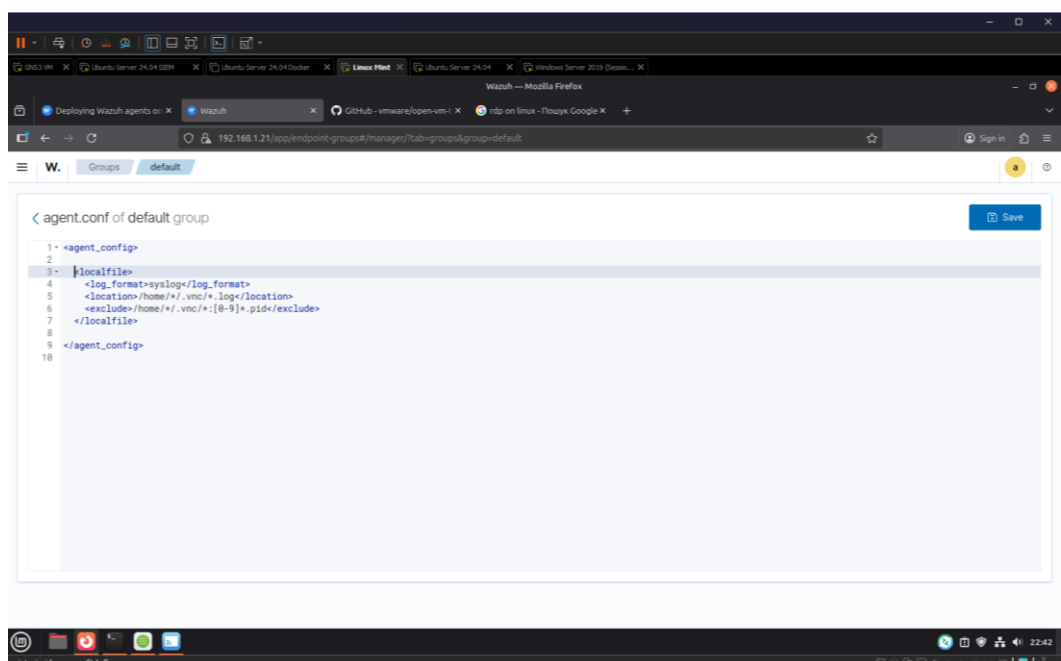


Рисунок 3.56 – Додавання загальної конфігурації для агентів

Також для обробки змісту логів VNC-серверу необхідно написати власний декодер для Wazuh. На рисунку 3.57 відображено реалізацію власного декодера custom-decoder.xml для Wazuh для обробки логів VNC-сервера.

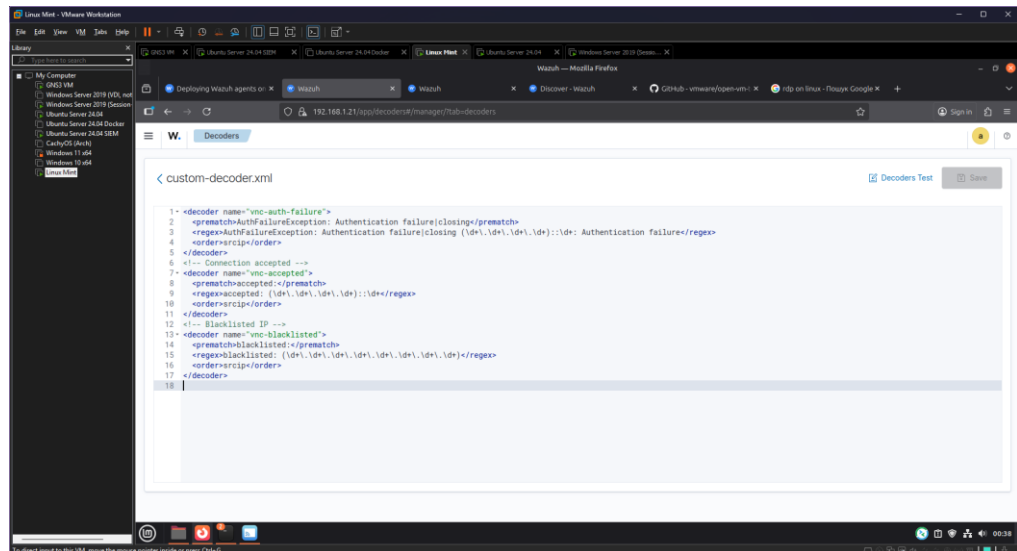


Рисунок 3.57 – Реалізація власного декодера для Wazuh

3.7. Дослідження роботи тестового стенду

Після всіх виконаних налаштувань варто перейти до аналізу роботи стенду: дослідження роботи засобів захисту та порівняння використовуваних ресурсів для кожного виду розгортання віддаленого доступу. На рисунку 3.58 наведено вигляд створеної та налаштованої мережі.

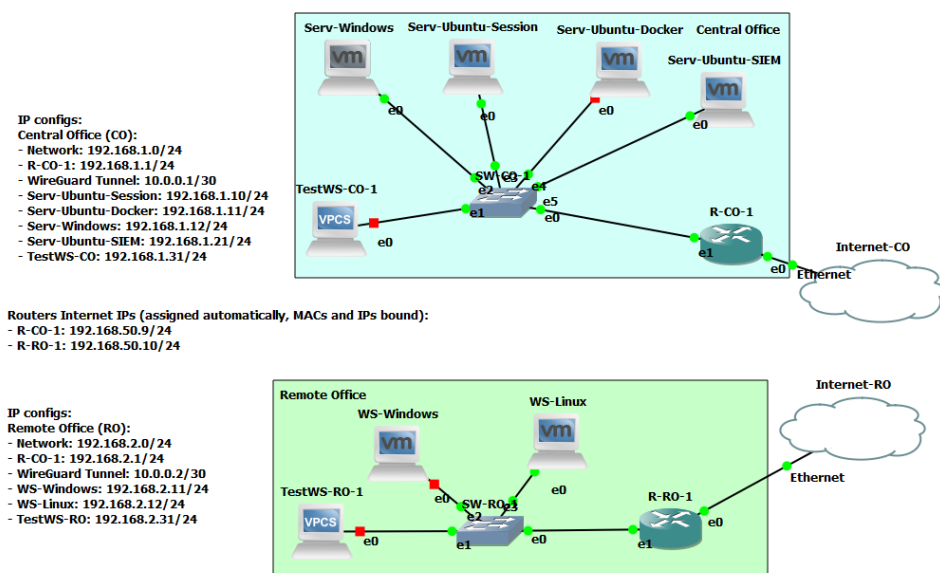


Рисунок 3.58 – Вигляд мережі

Для початку варто дослідити як Wazuh збирає дані про невдалі спроби входу по RDP. Для цього спробуємо декілька разів неправильно ввести пароль. Симуляція brute-force атаки на RDP-сервер шляхом введення невірних облікових даних відображено на рисунку 3.59. На рисунку 3.60 показано реакцію Wazuh Dashboard на симульовану атаку перебору паролів RDP. Рисунок 3.61 відображає спрацювання раніше налаштованих правил Wazuh на невдалі спроби входу по RDP.

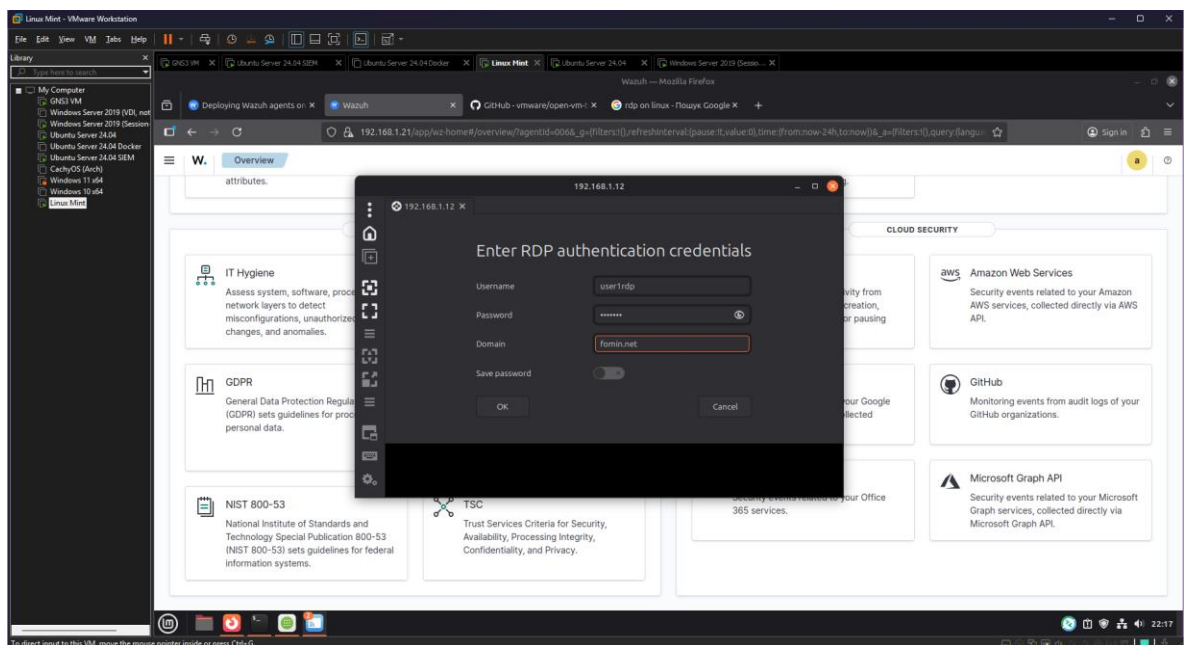


Рисунок 3.59 – Брут-форс атака на RDP-сервер

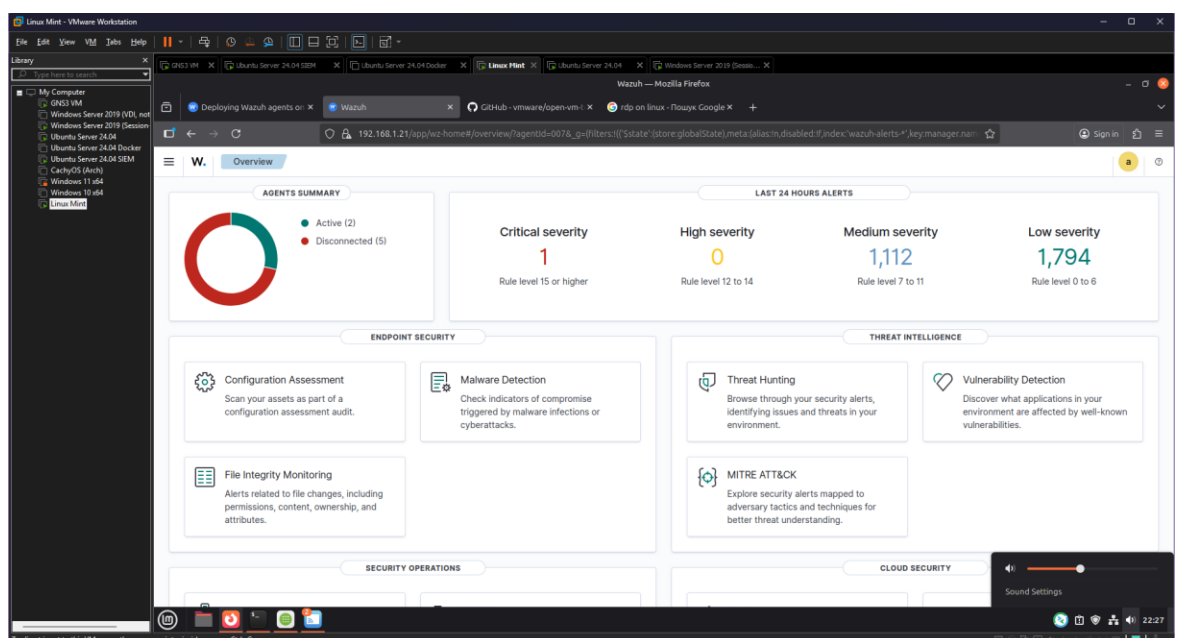


Рисунок 3.60 – Реакція Wazuh на атаку

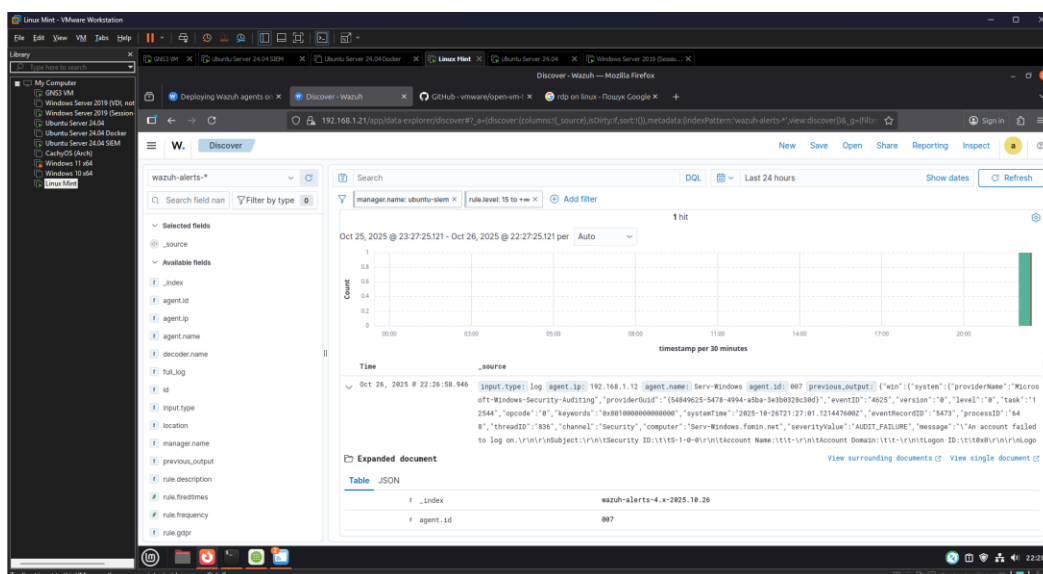


Рисунок 3.61 – Раніше налаштовані правила успішно спрацювали

Як можна побачити, Wazuh коректно “впіймав” злочинця: видно кількість невдалих спроб входу, адресу RDP-серверу до якого він намагався підключитися та IP-адреса пристрою злочинця.

Наступним кроком варто переглянути захист наших VNC-серверів. Відповідно до налаштованих правил, ми повинні бачити усі невдалі спроби, а також спроби підключення з уже заблокованих IP-адрес. VNC-сервер також автоматично блокує IP-адресу злочинця після 5 невдалих спроб авторизації. Невдалі спроби авторизації на VNC-сервері, задокументовані в Wazuh, відображено на рисунку 3.62.

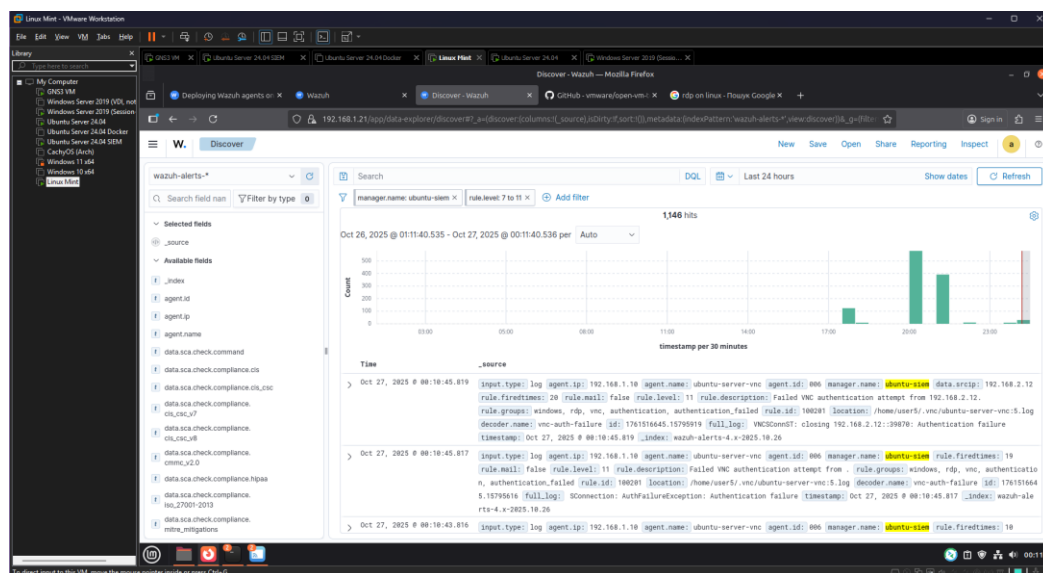


Рисунок 3.62 – Невдалі спроби авторизації задокументовано

На рисунку 3.63 показано блокування доступу до VNC-сервера після багаторазових невдалих спроб входу та фіксацію цього в Wazuh.

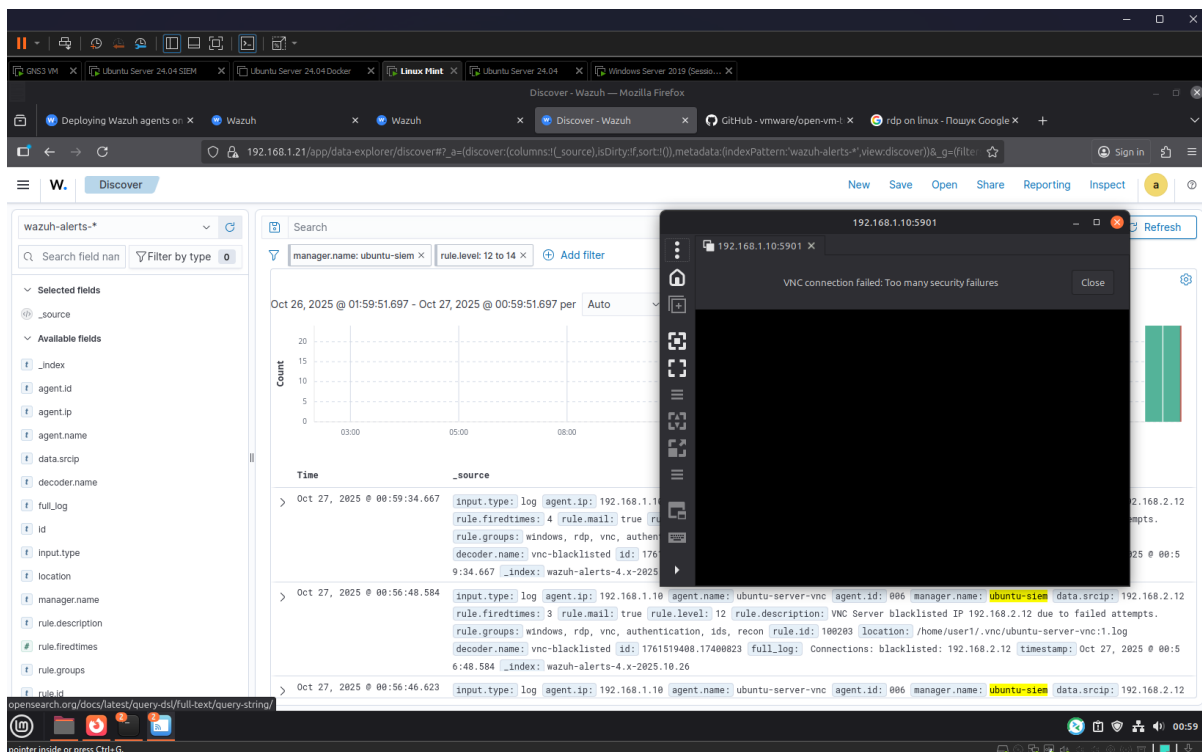


Рисунок 3.63 – Доступ до VNC заблоковано, а всі спроби підключень з заблокованого IP задокументовано

На цьому етапі можна сказати, що налаштована система моніторингу працює коректно, перевіривши усі журнали, створені нашими правилами.

Крім власних журналів, Wazuh також є дуже корисним інструментом для забезпечення безпечного середовища на робочих станціях агентів. Спеціаліст з кібербезпеки за допомогою цього інструменту може детально переглянути усі знайдені вразливості. Для прикладу візьмемо наш RDP-сервер.

Результат аналізу вразливостей (Vulnerability Detection) на Serv-Windows за допомогою Wazuh відображено на рисунку 3.64. Детальна інформація про критичну вразливість на Serv-Windows з Wazuh Dashboard показана на рисунку 3.65.

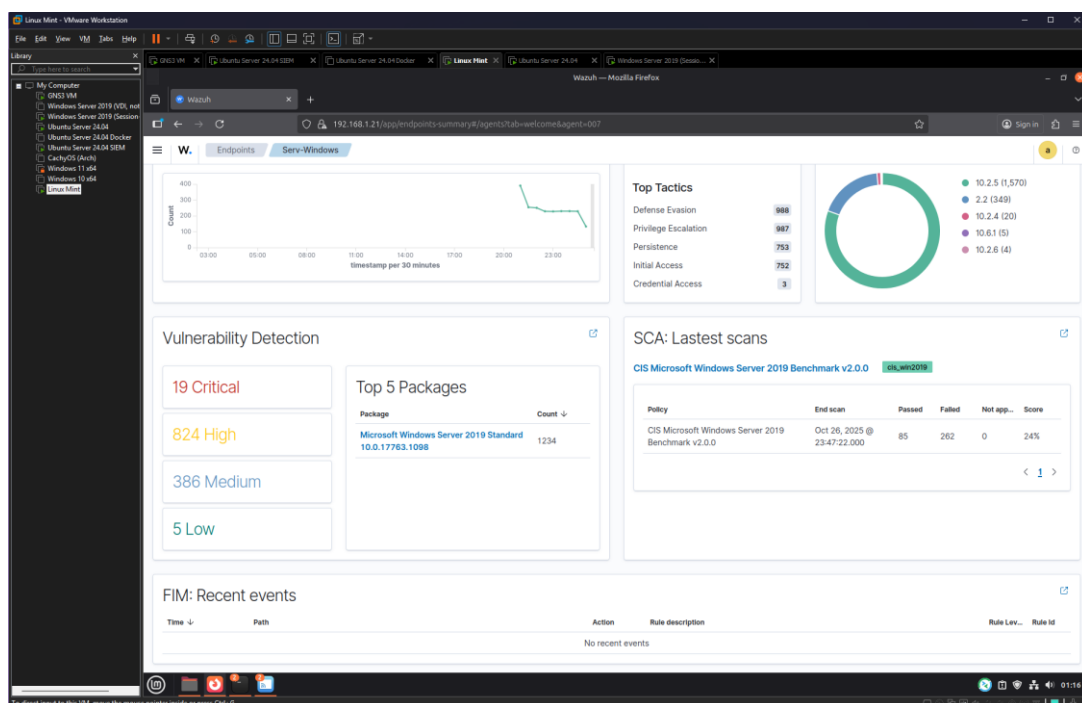


Рисунок 3.64 – Результат аналізу вразливостей на Serv-Windows

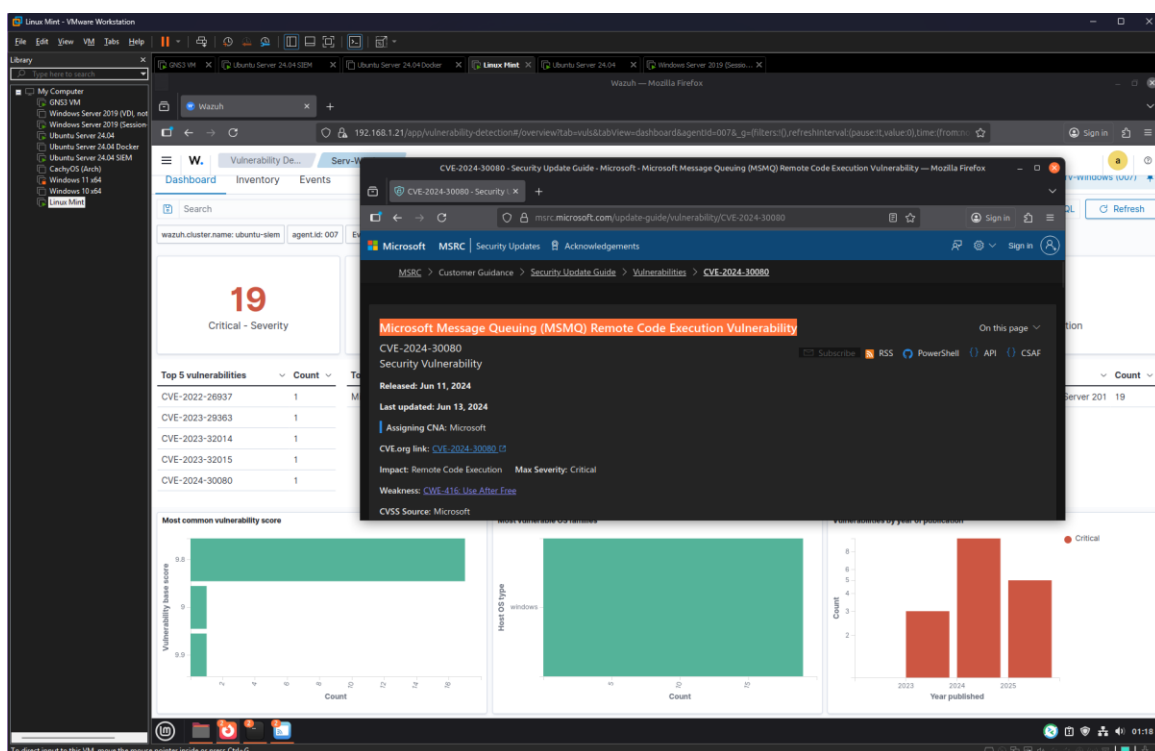


Рисунок 3.65 – Результат аналізу вразливостей на Serv-Windows

За допомогою цього інструментарію можна провести дослідження вразливостей та успішно їх усунути, аби уникнути проблем в майбутньому.

Спираючись на такі детальні результати моніторингу, можна підтвердити що налаштування засобів моніторингу – це один з найкращих методів

забезпечення безпеки у корпоративній мережі. У даному дослідженні засіб моніторингу Wazuh дав змогу захистити усі налаштовані стенди-сервери.

Також варто не забувати про безпеку мережі. Потрібно практично перевірити, що маршрутизатори коректно налаштовані та захищають мережу від небажаних зовнішніх підключень.

Щоб переконатися, що з Інтернету доступний лише порт WireGuard UDP, а всі інші порти закриті, можна провести скан портів маршрутизаторів з хост машини за допомогою nmap. Сканування проводитиметься в 2 сценарії: ззовні та зсередини.

ІР-адреси маршрутизаторів за якими вони виходять в Інтернет це 192.168.50.9 та 192.168.50.10 відповідно. Налаштування безпеки у них ідентичні тому наведено приклад на першому маршрутизаторі.

Для перевірки здійснено сканування TCP та сканування порту, який використовується для Wireguard. Результат зовнішнього сканування TCP-портів маршрутизатора R1 за допомогою Zenmap (Nmap) показано на рисунку 3.66. Результат сканування UDP-порту WireGuard (12321) маршрутизатора R1 за допомогою Zenmap (Nmap) відображено на рисунку 3.67.

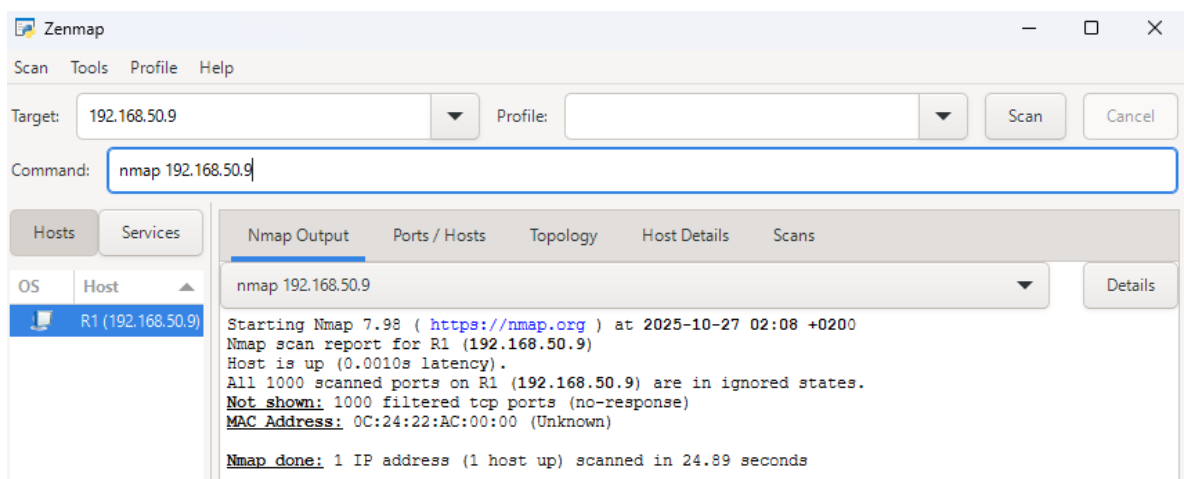


Рисунок 3.66 – Результат сканування відкритих портів

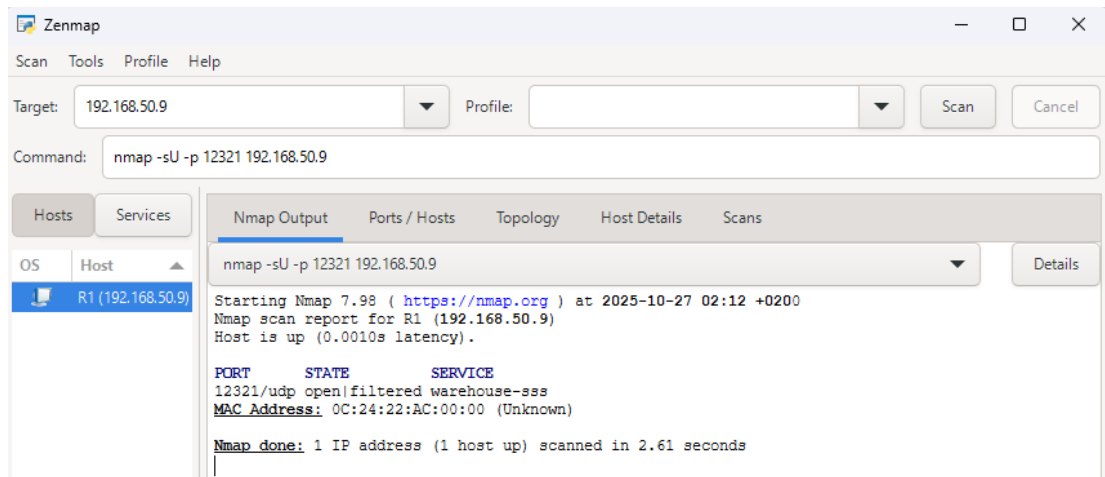


Рисунок 3.67 – Результат сканування відкритих портів

Сканування TCP показує що всі 1000 сканованих портів закриті, і немає жодного відкритого порту. Стан open|filtered для UDP означає, що сканер отримав відповідь, яка не вказує однозначно, чи порт відкритий, але й не вказує, що він точно закритий, що є очікуваною поведінкою для WireGuard.

Ця перевірка підтверджує ефективність зовнішнього фаєрволу, який пропускає лише UDP-трафік на порт WireGuard.

Тепер варто перейти до внутрішнього сканування. Результат цього сканування наведено на рисунку 3.68.

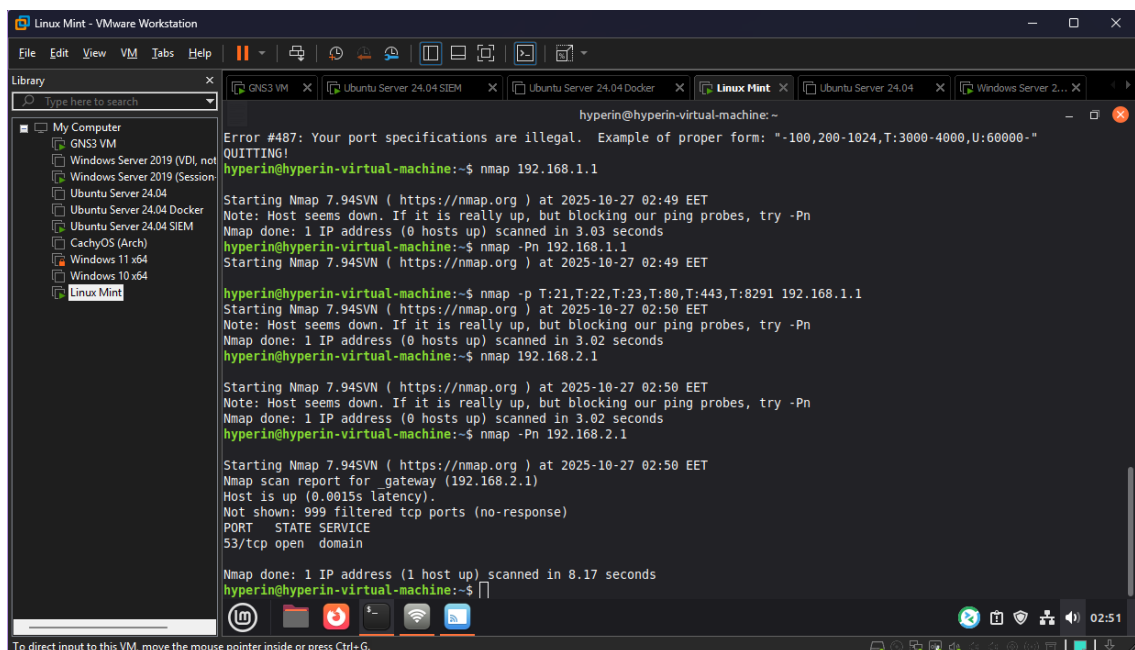


Рисунок 3.68 – Результат сканування відкритих портів

Як можна побачити, всі раніше налаштовані параметри безпеки працюють коректно – доступ до маршрутизатора з віддаленої підмережі блокується, а локальний маршрутизатор лише має відкритий порт для DNS. Все працює коректно, а головне – безпечно.

Наступним етапом дослідження варто провести експерименти з продуктивності кожного підходу до реалізації віддаленого доступу. Для цього на кожному зі стендів буде запущено одночасно по 5 сесій користувачів та виконано заміри навантаження на CPU та RAM кожного з серверів. Для забезпечення справедливого порівняння усі сервери мають 2 CPU-ядра та 4 ГБ оперативної пам'яті. В якості тесту, порівнювання здійснюватиметься між двома замірами навантаженнями на систему: перший раз з лише запущеними сесіями користувачів, другий – з запущеним браузером Firefox, щоб провести емуляцію роботи користувача.

Для початку, необхідно перевірити RDP-сервер на основі сесій – Serv-Windows. На рисунку 3.69 показано 5 одночасно запущених RDP-сесій на Serv-Windows. Навантаження на CPU та RAM Serv-Windows у стані спокою (5 RDP-сесій у простої) відображено на рисунку 3.70.

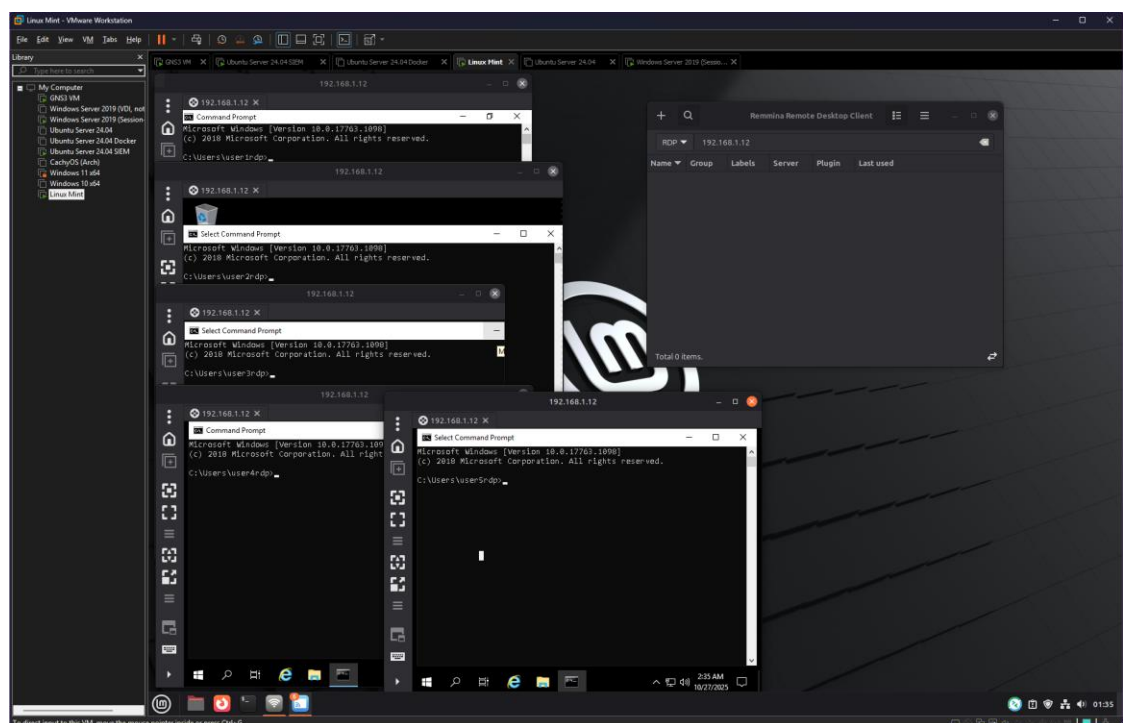


Рисунок 3.69 – Запущено 5 RDP-сесій

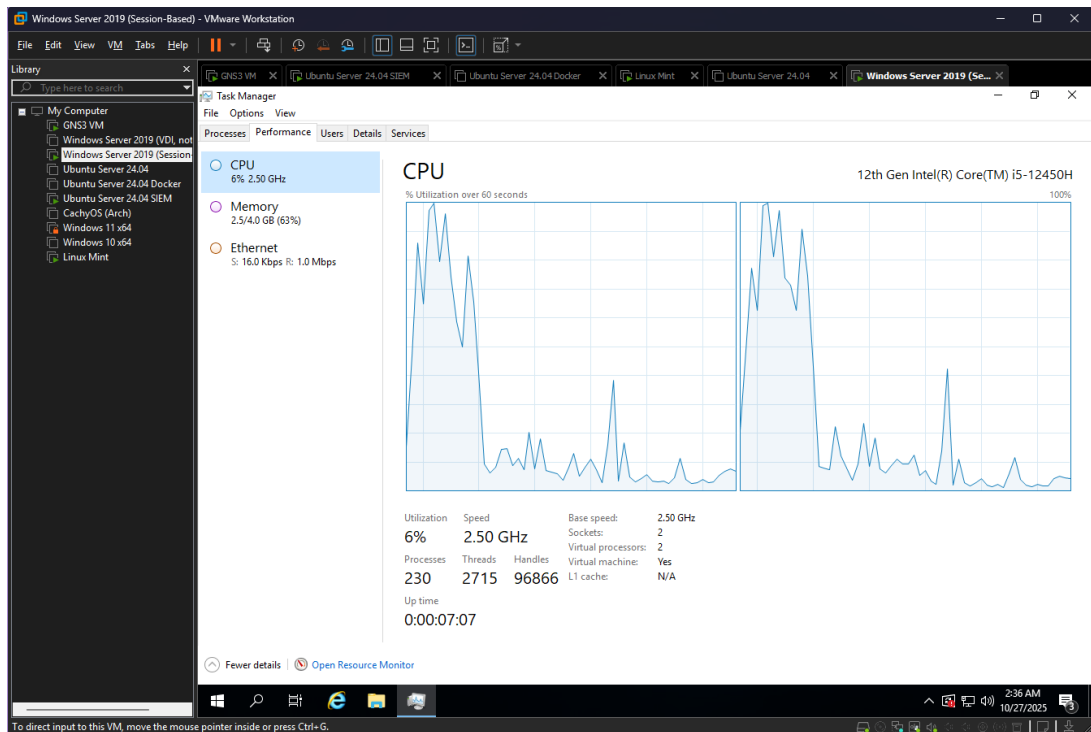


Рисунок 3.70 – Навантаження на систему

Наразі система успішно працює, завантаження на процесор досить мале, та зайнято більше ніж половину оперативної пам'яті. Переглянемо що станеться якщо користувачі будуть виконувати якусь роботу. Запуск браузера Firefox в усіх 5 RDP-сесіях для симуляції активності користувачів показано на рисунку 3.71.

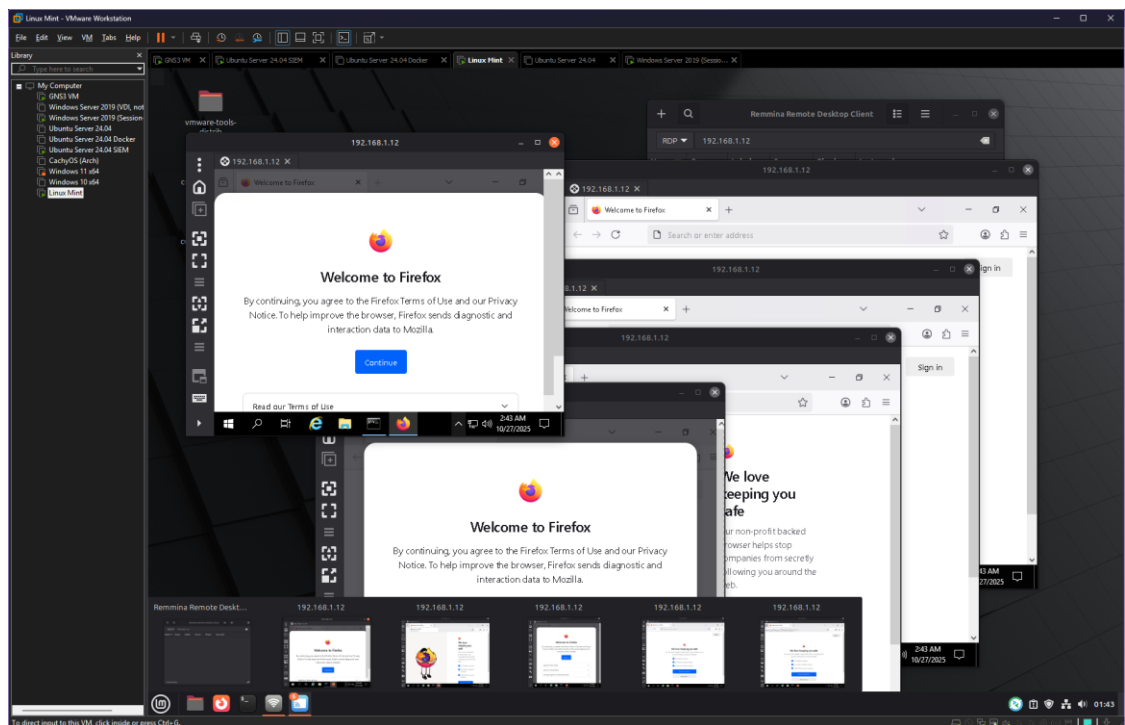


Рисунок 3.71 – Запуск програм у кожного користувача

Навантаження на CPU та RAM Serv-Windows при активності (5 RDP-сесій з запущеним браузером) відображено на рисунку 3.72.

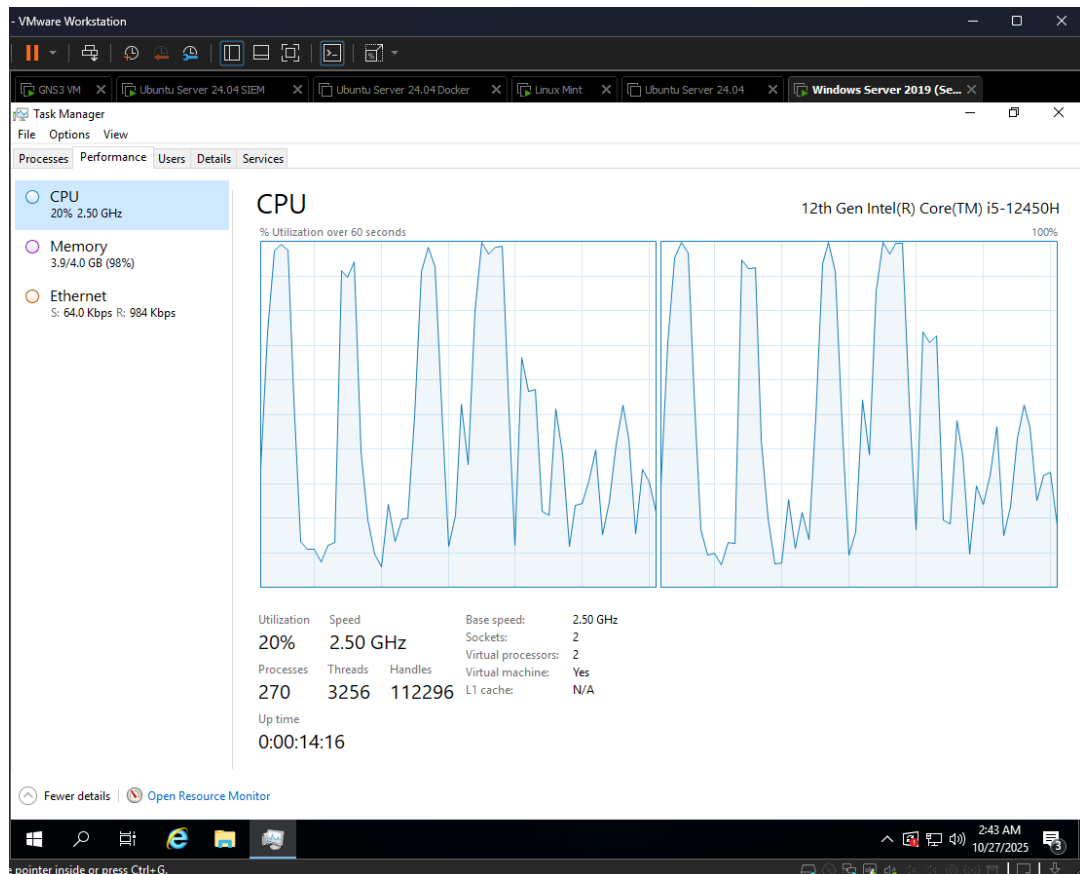


Рисунок 3.72 – Завантаження на сервер із запущеними програмами

Тепер ситуація дещо гірша: піднялося навантаження на процесор та оперативної пам'яті вже не вистачає для комфортної роботи.

Запишемо зібрані результати:

- Перший замір (Простій):
 - Процесор: в середньому ~6% навантаження.
 - Оперативна пам'ять: 2.5 ГБ.
- Другий замір (Активність):
 - Процесор: в середньому ~50% з частими піками до 95%.
 - Оперативна пам'ять: 3.9 ГБ.

Також було здійснено спостереження за навантаженням на маршрутизатори при роботі з Firefox через RDP. Навантаження на CPU маршрутизаторів R1 та R2 у стані спокою (5 RDP-сесій у простої) відображено на рисунку 3.73.

```
[admin@R2] > /system resource monitor
cpu-used: 0%
free-memory: 307204KiB
- [Q quit|D dump|C-z pause]

[admin@R1] > /system resource monitor
cpu-used: 1%
free-memory: 305004KiB
- [Q quit|D dump|C-z pause]
```

Рисунок 3.73 – Навантаження на маршрутизатори в просторі

Далі залишивши лише 1 сесію RDP і ввімкнувши відео в браузері, можна переглянути яку кількість трафіку використовує RDP при передачі по інтерфейсу Wireguard. Аналіз мережевого трафіку на WireGuard-інтерфейсі R2 при активній RDP-сесії з відтворенням відео показано на рисунку 3.74.

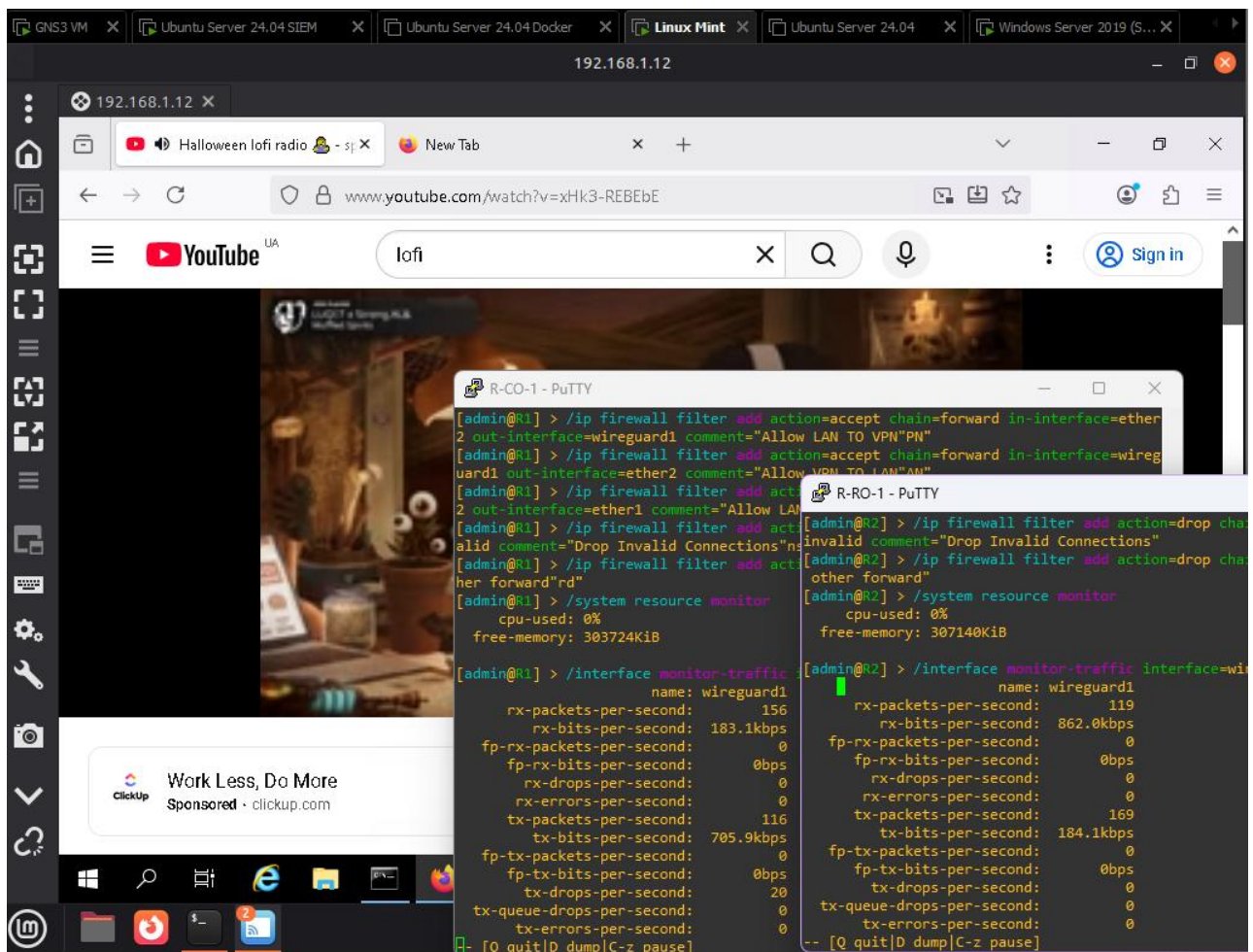


Рисунок 3.74 – Аналіз трафіку для RDP

Аналіз навантаження на CPU маршрутизаторів R1 та R2 при активній RDP-сесії з відтворенням відео відображено на рисунку 3.75.

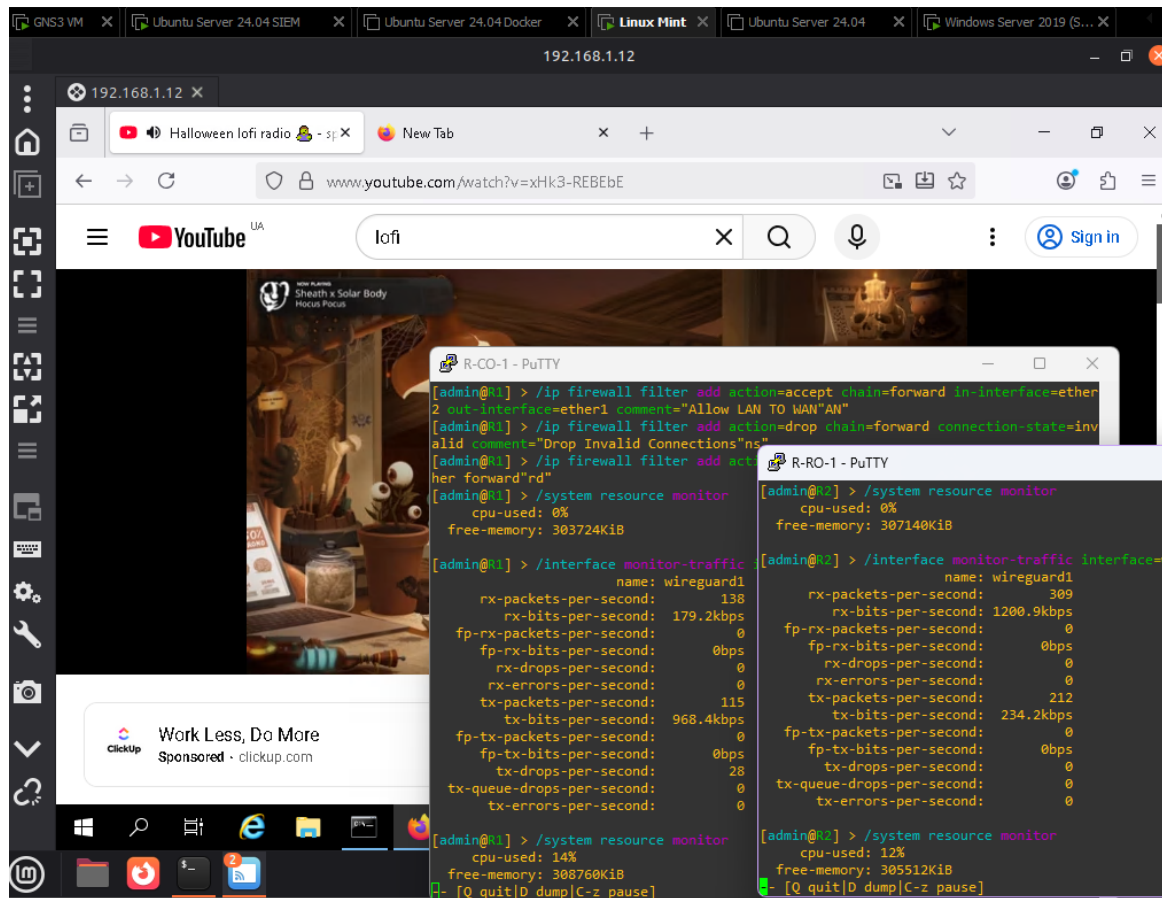


Рисунок 3.75 – Аналіз навантаження на маршрутизатори

За результатами спостережень, RDP споживає десь 700 – 1200 кбіт/с. І навантажує процесори обидвох маршрутизаторів на 12-14%.

Цей скачок навантаження можна пояснити тим, що CPU на маршрутизаторах MikroTik під час роботи VPN навантажується в основному через процеси шифрування та дешифрування трафіку для тунелю WireGuard.

Для передачі відео з віддаленої-сесії, генерується значний обсяг графічних оновлень, які потрібно передати через VPN. Пояснення для цих процесів таке:

1. R1 отримує невеликий потік даних від клієнтів (клавіатура/миша) і шифрує великий потік вихідних даних (часте оновлення екрану) для відправки до R2. Шифрування є більш ресурсоємним процесом. Тому навантаження на R1 вище (14%).

2. R2 шифрує невеликий потік даних (клавіатура/миша) і дешифрує великий потік вхідних даних (оновлення екрану від R1). Дешифрування зазвичай менш навантажує CPU. Тому навантаження на R2 нижче (12%).

Аналогічно перейдемо до Serv-Ubuntu-Session. Варто зазначити, що дана конфігурація VNC-серверу не може динамічно змінювати розширення, як це робить RDP, тому всі ці заміри робилися з розширенням 1920x1080. Навантаження на CPU та RAM Serv-Ubuntu-Session у стані спокою (5 VNC-сесій у простої) відображено на рисунку 3.76.

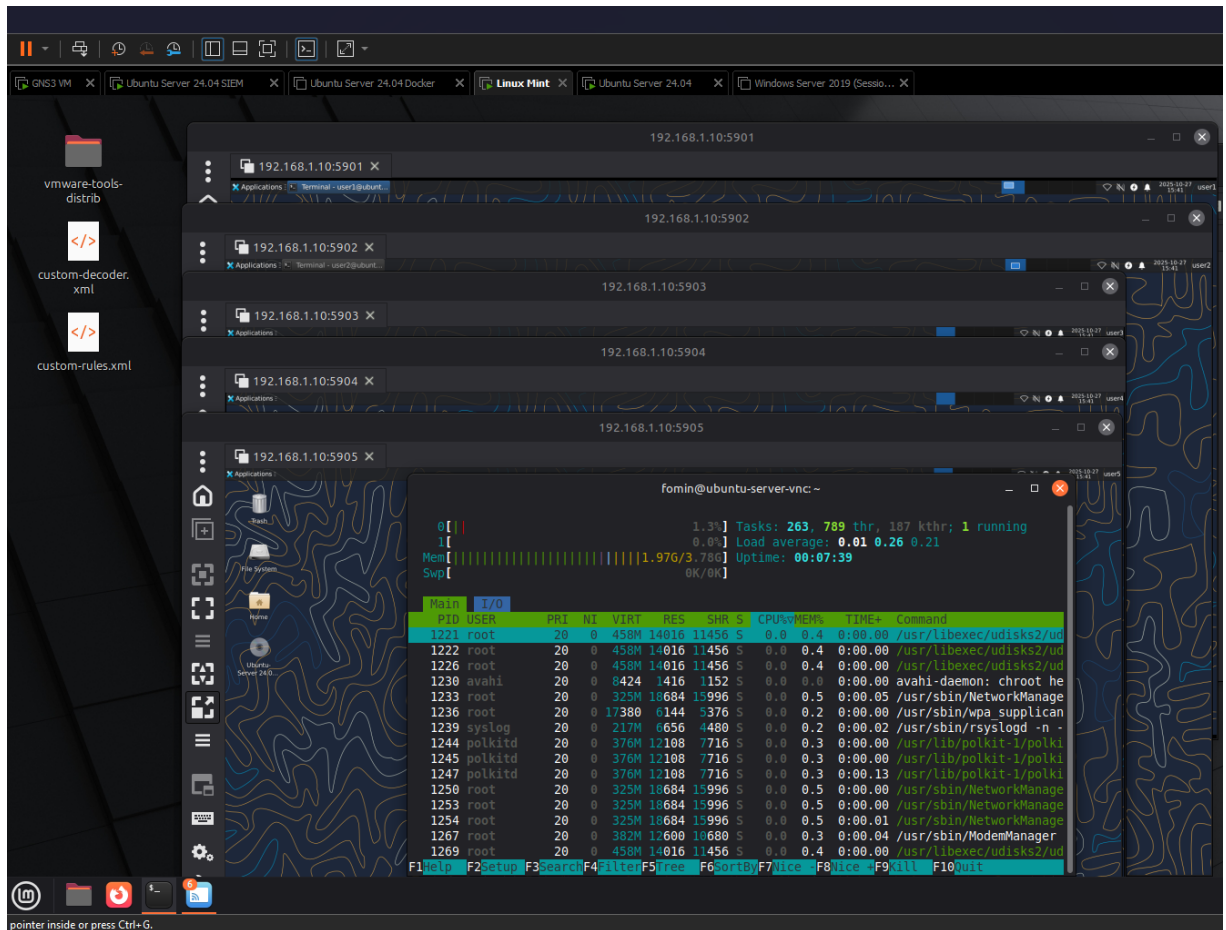


Рисунок 3.76 – Навантаження при 5 запущених сесіях

5 запущених сесій у простої не споживають багато ресурсів, навіть на процесор навантаження досить мале. Навантаження на CPU та RAM Serv-Ubuntu-Session при активності (5 VNC-сесій з запущеним браузером) відображено на рисунку 3.77.

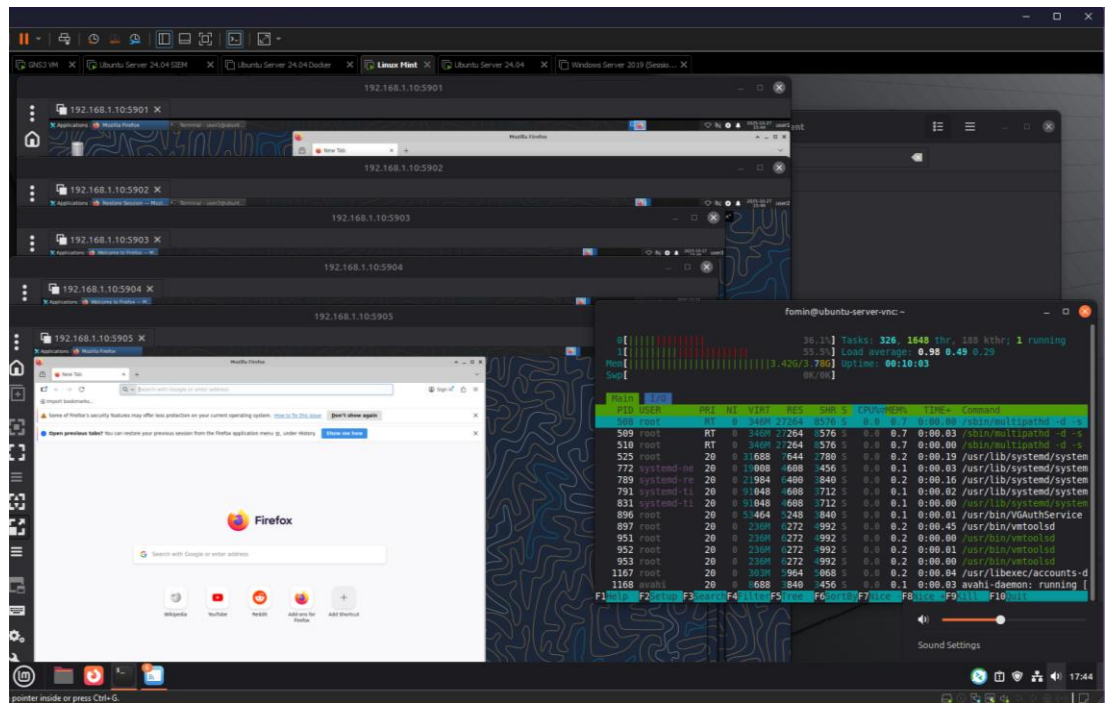


Рисунок 3.77 – Навантаження при 5 “активних” сесіях

З активними сесіями ситуація гірша: збільшилося навантаження на процесор та на оперативну пам’ять. Запишемо отримані результати:

- Перший замір (Простій):
 - Процесор: в середньому ~1% навантаження.
 - Оперативна пам’ять: 1.97 ГБ.
- Другий замір (Активність):
 - Процесор: в середньому ~45% з піками до 100% при активній роботі.
 - Оперативна пам’ять: 3.42 ГБ.

Тепер перейдемо до аналізу навантаження на маршрутизатори при роботі з Firefox через VNC. Навантаження на CPU маршрутизаторів R1 та R2 у стані спокою (5 VNC-сесій у простої) відображено на рисунку 3.78.

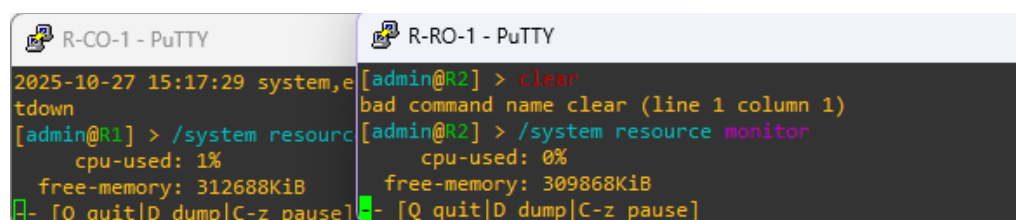


Рисунок 3.78 – Навантаження на маршрутизатори в простої

Аналогічно до тесту RDP, залишено лише одну активну VNC-сесію з активним відео. На рисунку 3.79 показано аналіз навантаження на маршрутизатори при активній VNC-сесії з відтворенням відео (частина 1). На рисунку 3.80 показано аналіз навантаження на маршрутизатори при активній VNC-сесії з відтворенням відео (частина 2).

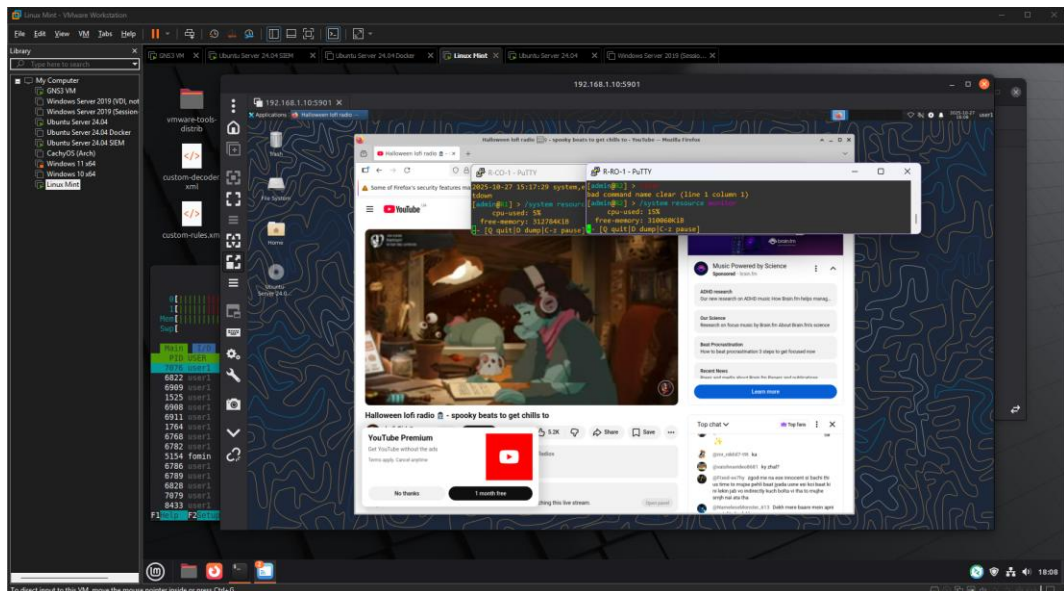


Рисунок 3.79 – Навантаження на маршрутизатори при активному відео через VNC

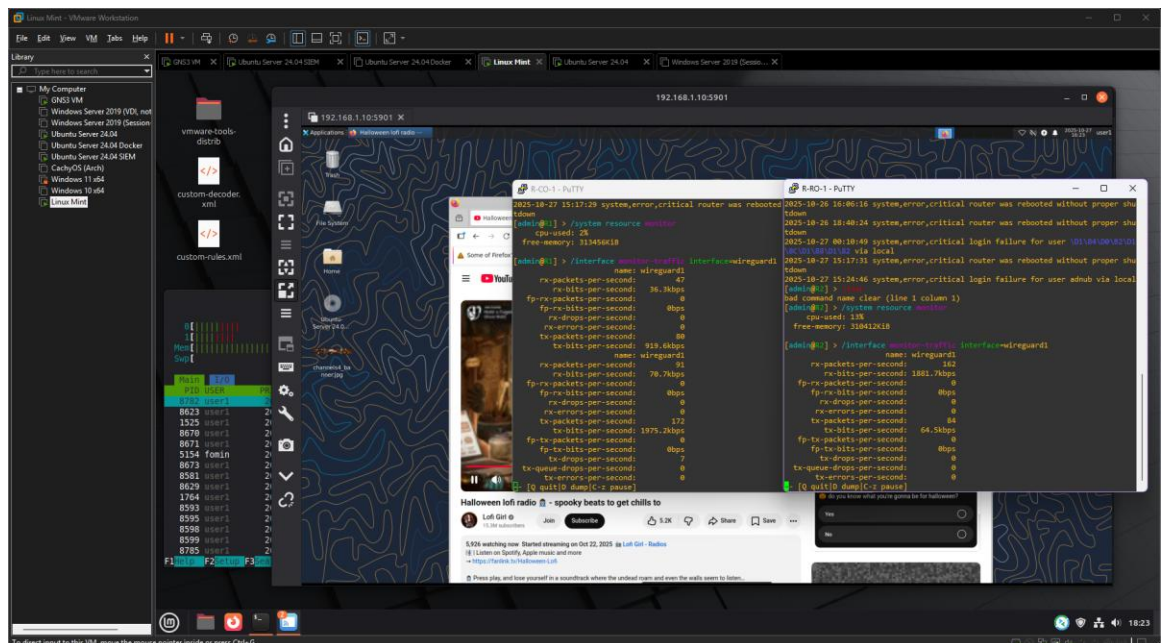


Рисунок 3.80 – Навантаження на маршрутизатори при активному відео через VNC

Як можна побачити з замірів використовуваного трафіку, VNC набагато більш ресурсоємний ніж RDP. VNC у даному дослідженні споживає близько 900-2000 кбіт/с швидкості передачі даних. Також варто зазначити, що ~1 Мбіт/с – це “стеля” швидкості у налаштованому середовищі емуляції мережі GNS3 і її не вистачає для комфортної роботи через VNC: робота з ним дуже повільна, іноді необхідно почекати декілька секунд поки дії користувача відобразяться на екрані.

Це можна пояснити тим, що VNC передає "знімки" екрану (framebuffer) через що він набагато “важче” на мережу, тоді як RDP передає графічні команди.

Останнім кроком проаналізуємо Serv-Ubuntu-Docker. Аналогічно, підключимось до 5 сесій і переглянемо навантаження на систему. Навантаження на CPU та RAM Serv-Ubuntu-Docker у стані спокою (5 VNC-контейнерів у просторі) відображено на рисунку 3.81.

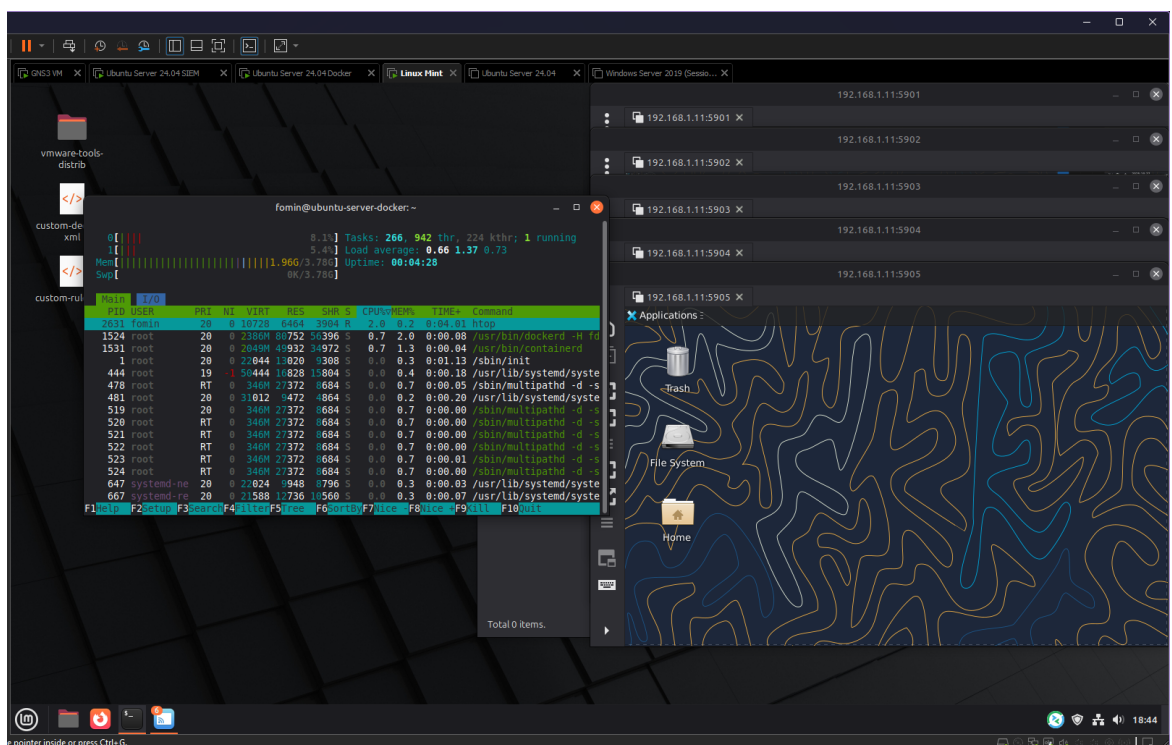


Рисунок 3.81 – Перегляд навантаження на Serv-Ubuntu-Docker в просторі

Тут вже можна помітити, що навантаження на процесор вище, ніж у іншого сервера; навантаження на оперативну пам’ять те ж саме. Тепер запусимо браузер у кожній сесії.

Навантаження на CPU та RAM Serv-Ubuntu-Docker при активності (5 VNC-контейнерів з запущеним браузером) відображено на рисунку 3.82.

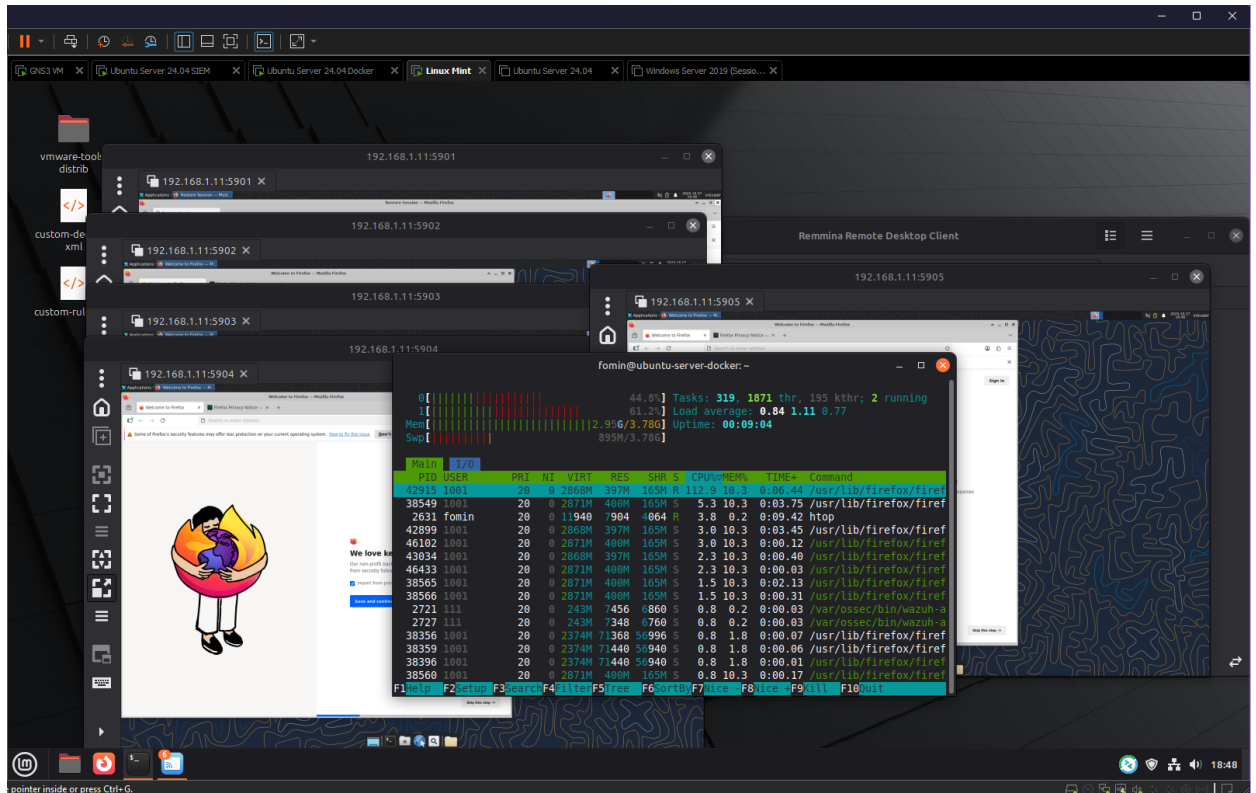


Рисунок 3.82 – Перегляд навантаження на Serv-Ubuntu-Docker з активними сесіями

Запишемо отримані результати:

- Перший замір (Простий):
 - Процесор: в середньому ~6% навантаження.
 - Оперативна пам'ять: 1.96 ГБ.
- Другий замір (Активність):
 - Процесор: в середньому ~50% з піками до 100% при активній роботі.
 - Оперативна пам'ять: 2.95 ГБ.

Тепер перевіримо навантаження на маршрутизатори. Навантаження на CPU маршрутизаторів R1 та R2 у стані спокою (5 VNC-контейнерів у простої) відображено на рисунку 3.83.

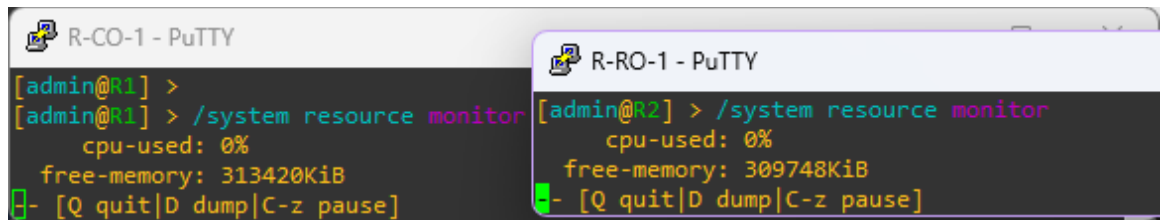


Рисунок 3.83 – Навантаження на маршрутизатори в просторі

На рисунку 3.84 показано аналіз навантаження на маршрутизатори R1 та R2 при активній VNC-сесії в Docker-контейнері.

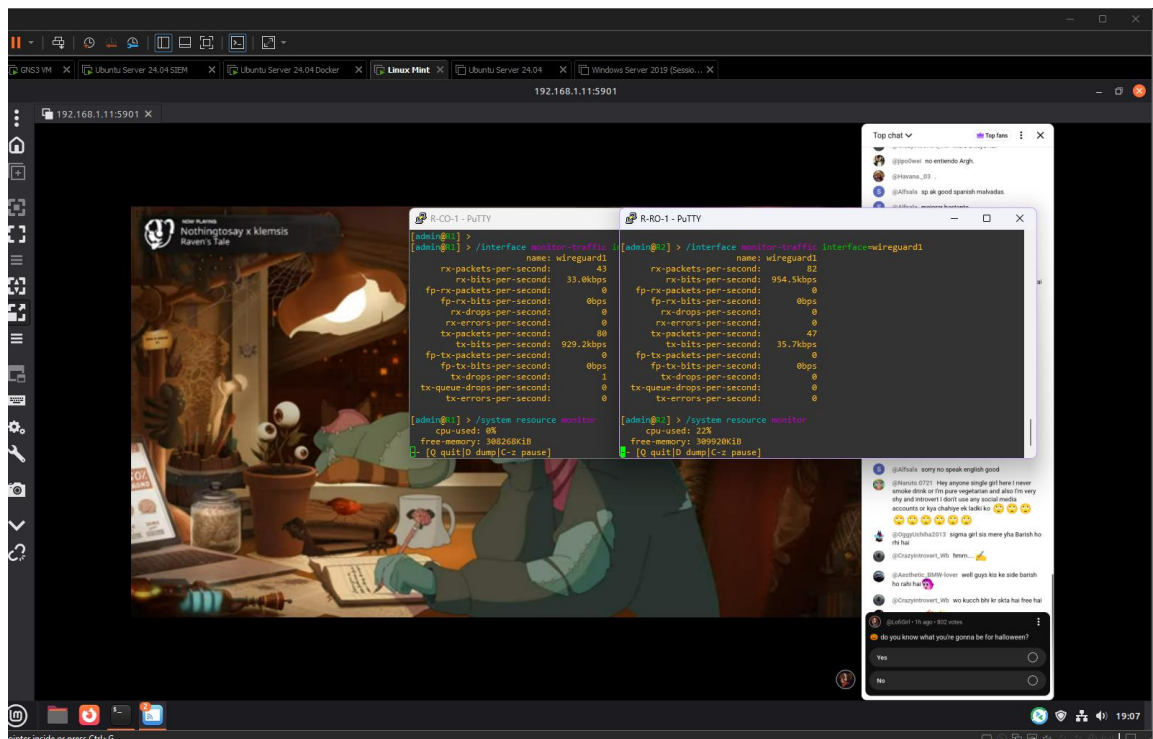


Рисунок 3.84 – Навантаження на маршрутизатори при активності

Як можна побачити, результати практично ідентичні, адже ми використовуємо VNC у обох випадках.

Головною особливістю використання Docker-контейнерів є те, що клієнти ізольовані один від одного: у кожного з них своя машина. Для цього експерименту запустимо декілька процесів від одного користувача і спробуємо їх переглянути з-під іншого користувача, а також спробуємо отримати доступ до його домашнього каталогу.

Для перегляду процесів іншого користувача використано команду “ps aux | grep user2”. На рисунку 3.85 продемонстровано відсутність ізоляції процесів користувачів на Serv-Ubuntu-Session.

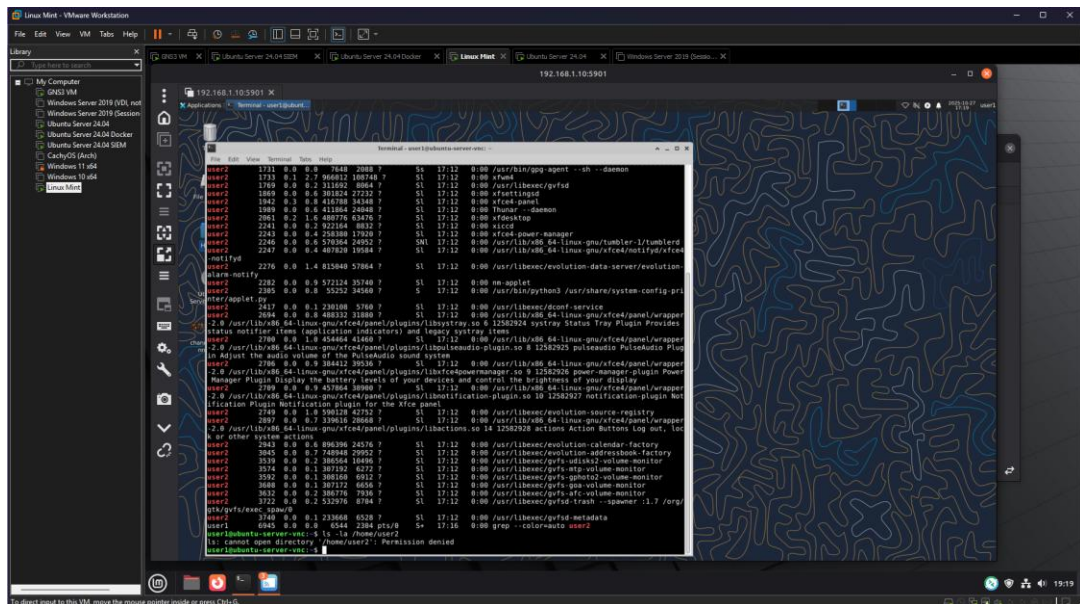


Рисунок 3.85 – Перевірка ізоляції процесів користувачів

Як можна побачити користувачі бачать процеси один одного, однак доступ до чужих домашніх каталогів блокується. Ми, звісно, отримали Permission denied (через стандартні права Linux), але сам факт того, що ми можемо спробувати отримати доступ і бачимо, що такий каталог існує, вже свідчить про спільний простір.

А тепер аналогічна перевірка на іншому сервері. На рисунку 3.86 продемонстровано повну ізоляцію процесів користувачів у Docker-контейнерах на Serv-Ubuntu-Docker.

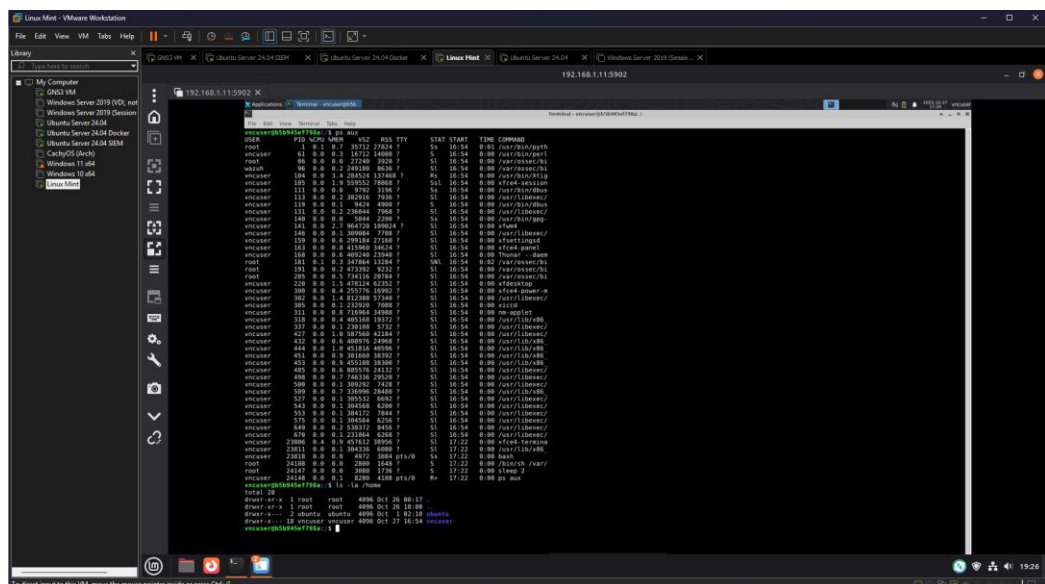


Рисунок 3.86 – Перевірка ізоляції процесів користувачів

Отже тут можна підведемо підсумки. На Serv-Ubuntu-Session користувачі працюють в одній ОС, можуть бачити процеси один одного. Файлова система спільна (хоча доступ обмежений стандартними правами). Це створює ризики для безпеки та конфіденційності.

На Serv-Ubuntu-Docker кожен користувач повністю ізольований у своєму контейнері. Немає доступу до процесів чи файлової системи інших користувачів або хостової ОС. Це забезпечує значно вищий рівень безпеки та відповідає принципу найменших привілеїв.

3.8. Підсумки дослідження роботи тестового стенду

Проведене експериментальне дослідження дозволило практично перевірити та порівняти різні аспекти спроектованої системи безпечного віддаленого доступу, зокрема ефективність засобів захисту та продуктивність різних архітектур.

Система моніторингу Wazuh успішно виявила симульовані атаки перебору паролів (brute-force) на RDP-сервер, коректно спрацювавши налаштованими правилами та ідентифікувавши джерело атаки. Для VNC-серверів було підтверджено фіксацію невдалих спроб автентифікації та спрацювання вбудованого механізму блокування IP-адреси після 5 невдалих спроб, що також було задокументовано у Wazuh. Додатково, продемонстровано можливості Wazuh як інструменту проактивного пошуку та аналізу вразливостей на підключених агентах, що є важливим елементом підтримки безпеки інфраструктури. Сканування мережевого периметра за допомогою nmap підтвердило ефективність налаштувань фаєрволу: зовнішнє сканування показало доступність лише UDP-порту WireGuard, а внутрішнє сканування підтвердило блокування доступу до інтерфейсів керування маршрутизаторів між сегментами VPN, дозволяючи при цьому легітимний трафік до серверів.

Тестування ізоляції користувачів виявило суттєві відмінності між архітектурами. На сервері з сесійним доступом VNC (Serv-Ubuntu-Session) користувачі могли бачити процеси один одного, що становить потенційний

ризик для конфіденційності та безпеки. Натомість, на сервері з Docker-контейнерами (Serv-Ubuntu-Docker) кожен користувач працював у повністю ізольованому середовищі, без доступу до процесів чи файлової системи інших користувачів або хостової ОС, що повністю відповідає принципу найменших привілеїв та значно підвищує рівень безпеки.

Для порівняння продуктивності було проведено стрес-тестування трьох конфігурацій (RDP Session, VNC Session, VNC Docker) під навантаженням 5 одночасних користувачів на серверах з однаковими ресурсами (2 CPU, 4 ГБ RAM). Результати вимірювань навантаження на CPU та RAM серверів, а також мережевого трафіку та навантаження на CPU маршрутизаторів зведені у Таблицю 3.1.

Таблиця 3.1

Порівняння продуктивності рішень віддаленого доступу (5 користувачів)

Показник	Стан	RDP Session (Windows)	VNC Session (Ubuntu)	VNC Docker (Ubuntu)
Навантаження CPU сервера	Простій	~6%	~1%	~6%
	Активність	~50%(піки 95-100%)	~45% (піки 100%)	~50%(піки 100%)
Використання RAM сервера	Простій	2.5 ГБ	1.97 ГБ	1.96 ГБ
	Активність	3.9 ГБ	3.42 ГБ	2.95 ГБ
Мережевий трафік (Відео)	Активність	700 - 1200 kbps	900 - 2000 kbps	900 - 2000 kbps
Навантаження CPU R1 (Пік)	Активність	~14%	~6%	~6%
Навантаження CPU R1 (Пік)	Активність	~12%	~5%	~5%

У стані простою VNC-сесії на Ubuntu виявилися найменш ресурсоємними, особливо щодо CPU. Під активним навантаженням (запуск Firefox) усі три рішення показали схоже високе навантаження на CPU сервера, що свідчить про те, що основне навантаження створюється саме клієнтськими програмами (браузером). Щодо оперативної пам'яті, Windows RDP виявився найбільш вимогливим, досягнувши ліміту в 4 ГБ. Обидва рішення на Ubuntu споживали менше RAM, причому Docker-контейнери показали найкращий результат (2.95

ГБ), що може свідчити про кращу оптимізацію використання ресурсів завдяки ізоляції.

Аналіз мережевого трафіку підтвердив, що RDP є більш ефективним протоколом для передачі графічно насиченого контенту (відео), споживаючи менше мережевого трафіку (700-1200 kbps) порівняно з VNC (900-2000 kbps). VNC, хоч і генерував більше "сирого" трафіку, ймовірно, не досягав максимальної пропускної здатності каналу в GNS3 (~1 Мбіт/с), що призводило до меншого пікового навантаження на CPU маршрутизаторів (~5-6%) та суб'єктивно повільної роботи з затримками. Це обмеження тестового середовища GNS3 могло вплинути на реальну картину навантаження VNC на мережеве обладнання в умовах ширшого каналу.

Проведені експерименти підтвердили працездатність та ефективність спроектованої системи. Налаштовані засоби захисту (фаєрвол, VPN, SIEM) успішно виявляли та блокували симульовані атаки. Порівняльний аналіз продуктивності показав переваги Linux-рішень у споживанні RAM, особливо при використанні Docker-контейнерів, хоча під активним навантаженням усі варіанти вимагали значних ресурсів CPU. RDP продемонстрував кращу ефективність використання мережевого каналу, тоді як VNC виявився більш вимогливим до пропускної здатності. Рішення на базі Docker забезпечило найвищий рівень ізоляції та безпеки для користувачів.

Висновки до третього розділу: у третьому розділі було проведено практичну реалізацію та дослідження спроектованої системи на тестовому стенді в GNS3. Налаштовано VPN WireGuard на маршрутизаторах MikroTik та реалізовано три варіанти віддаленого доступу: RDP Session-Based, VNC Session-Based та ізольовані VNC-сесії у Docker-контейнерах (Ubuntu). Розгорнуто SIEM-систему Wazuh з власними правилами для моніторингу RDP та VNC. Дослідження безпеки (симуляція brute-force атак, перевірка nmap та тестування ізоляції) підтвердило переваги Docker-архітектури. Виконано порівняльне дослідження продуктивності трьох рішень, результати якого зведено у таблицю.

ВИСНОВКИ

У кваліфікаційній роботі вирішено задачу підвищення безпеки та ефективності віддаленого доступу в корпоративних мережах. За результатами виконання поставлених завдань отримано наступні висновки:

1. Проведено аналіз технологій, який виявив переваги протоколу WireGuard у швидкодії над OpenVPN та доцільність використання гібридного підходу до віддаленого доступу (RDP для Windows, VNC для Linux).

2. Спроектовано архітектуру мережі, що передбачає розгортання VPN на маршрутизаторах MikroTik, що дозволило приховати структуру внутрішньої мережі та централізувати керування доступом. Розроблено схему ізоляції користувачів Linux за допомогою технології Docker.

3. Реалізовано повнофункціональний стенд системи. Налаштовано правила фаєрволу для захисту периметра та внутрішньої сегментації. Впроваджено SIEM-систему Wazuh, для якої розроблено власні правила та декодери для виявлення специфічних атак на RDP та VNC.

4. Експериментально підтверджено ефективність запропонованих рішень. Тести показали, що використання Docker для VNC знижує споживання оперативної пам'яті на 14% порівняно з сесійним підходом та забезпечує повну ізоляцію процесів користувачів. Система моніторингу успішно розпізнала симульовані атаки перебору паролів.

Таким чином, мета роботи досягнута: створена система забезпечує надійний захист віддаленого доступу при збереженні високої продуктивності роботи користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Corporate Network Security. SoftTeco. URL: <https://softteco.com/blog/corporate-network-security> (дата звернення: 21.01.2025).
2. What Is Remote Access? How Does It Work? Fortinet. URL: <https://www.fortinet.com/resources/cyberglossary/remote-access> (дата звернення: 21.01.2025).
3. VDI vs VPN vs RDP: Choosing a Secure Remote Access Solution. eSecurityPlanet. URL: <https://www.esecurityplanet.com/endpoint/secure-access-for-remote-workers-rdp-vpn-vdi/> (дата звернення: 24.01.2025).
4. Remote Access Made Easy: A Guide to Accessing Ubuntu from Anywhere. The Linux Code. URL: <https://thelinuxcode.com/access-ubuntu-remotely/> (дата звернення: 24.01.2025).
5. Setup VPN for small business. NordLayer. URL: <https://nordlayer.com/blog/setup-vpn-for-small-business/> (дата звернення: 22.01.2025).
6. How to Configure a Business VPN. Network Computing. URL: <https://www.networkcomputing.com/network-security/how-to-configure-a-business-vpn-a-setup-guide-for-your-business> (дата звернення: 22.01.2025).
7. WireGuard vs. OpenVPN. Astrill. URL: <https://www.astrill.com/blog/wireguard-vs-openvpn> (дата звернення: 22.01.2025).
8. OpenVPN vs WireGuard. Windscribe. URL: <https://windscribe.com/blog/openvpn-vs-wireguard/> (дата звернення: 23.01.2025).
9. What Is OpenVPN? An Explanation of Community Edition vs. Access Server. OpenVPN. URL: <https://blog.openvpn.net/openvpn-community-edition-vs-access-server/> (дата звернення: 12.12.2024).

10. How Does OpenVPN Work. VPN Central. URL: <https://vpncentral.com/how-does-openvpn-work/#1> (дата звернення: 12.12.2024).
11. Overview of OpenVPN. OpenVPN Community Wiki. URL: <https://community.openvpn.net/openvpn/wiki/OverviewOfOpenvpn> (дата звернення: 15.11.2024).
12. WireGuard vs OpenVPN: Which VPN Protocol Is Better? Cybernews. URL: <https://cybernews.com/what-is-vpn/wireguard-vs-openvpn/> (дата звернення: 15.11.2024).
13. WireGuard. WireGuard Official Website. URL: <https://www.wireguard.com/> (дата звернення: 15.11.2024).
14. WireGuard Configuration. OPNsense Documentation. URL: <https://docs.opnsense.org/manual/vpnet.html#wireguard> (дата звернення: 12.12.2024).
15. A Practical Guide for Implementing and Managing Remote Access Solutions. Netwrix. URL: <https://blog.netwrix.com/remote-access-management/> (дата звернення: 24.01.2025).
16. Аналіз технологій VPN для використання у корпоративному середовищі // Комп'ютерні технології: інновації, проблеми, рішення: тези доп. VII Всеукр. наук.-техн. конф. (м. Житомир, 2025). Житомир, 2025. С. 117. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/01/povnyj-tekst-1.pdf> (дата звернення: 25.10.2025).
17. Аналіз технологій віддаленого доступу для використання у корпоративному середовищі // Інформаційно-комп'ютерні технології: тези доп. XV Міжнар. наук.-техн. конф. (м. Житомир, 2025). Житомир, 2025. С. 83. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/povnyj-tekst.pdf> (дата звернення: 25.10.2025).

ДОДАТКИ

- IP configs:
- Central Office (CO):
- Network: 192.168.1.0/24
 - R-CO-1: 192.168.1.1/24
 - WireGuard Tunnel: 10.0.0.1/30
 - Serv-Ubuntu-Session: 192.168.1.10/24
 - Serv-Ubuntu-Docker: 192.168.1.11/24
 - Serv-Ubuntu-SIEM: 192.168.1.12/24
 - Serv-Ubuntu-SIEM: 192.168.1.21/24
 - TestWS-CO: 192.168.1.31/24

Routers Internet IPs (assigned automatically, MACs and IPs bound):

- R-CO-1: 192.168.50.9/24
- R-RO-1: 192.168.50.10/24

- IP configs:
- Remote Office (RO):
- Network: 192.168.2.0/24
 - R-CO-1: 192.168.2.1/24
 - WireGuard Tunnel: 10.0.0.2/30
 - WS-Windows: 192.168.2.11/24
 - WS-Linux: 192.168.2.12/24
 - TestWS-RO: 192.168.2.31/24

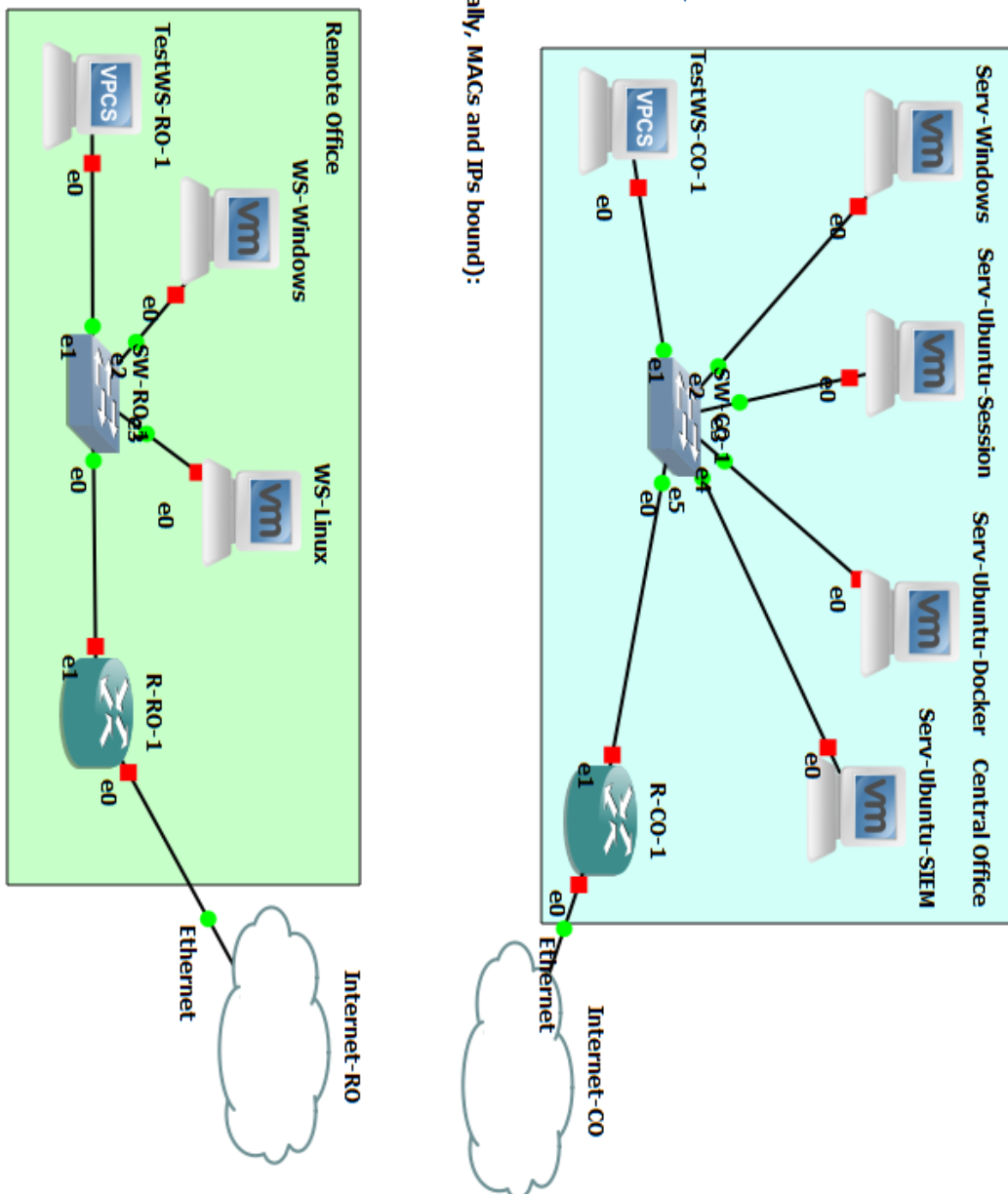


Рисунок А.1 – Проект мережі

ДОДАТОК В

Нижче представлені конфігураційні файли пристроїв мережі

R-CO-1 (R1):

```
[admin@R1] > /export
# 2025-10-27 17:53:33 by
RouterOS 7.20.2
# system id = QlPFHW3O0wC
#
/interface ethernet
set [ find default-name=ether1 ]
disable-running-check=no
set [ find default-name=ether2 ]
disable-running-check=no
/interface wireguard
add listen-port=12321 mtu=1420
name=wireguard1
/port
set 0 name=serial0
/interface wireguard peers
add allowed-
address=192.168.2.0/24 endpoint-
address=192.168.50.10 \
    endpoint-port=12321
interface=wireguard1
name=peer1 public-key=\
"dZM7NaFIQoImSDa2S69Jq8P
KC00ugRW+zClhBEqSdWY="
/ip address
add address=192.168.1.1/24
interface=ether2
network=192.168.1.0
add address=10.0.0.1/30
interface=wireguard1
network=10.0.0.0
/ip dhcp-client
add interface=ether1
/ip dns
set allow-remote-requests=yes
servers=8.8.8.8,1.1.1.1
/ip firewall filter
```

```
add action=accept chain=input
comment="Allow
Established/Related" \
```

```
connection-
state=established,related
```

```
add action=accept chain=input
comment="Allow WireGuard
UDP from WAN" \
```

```
dst-port=12321 in-
interface=ether1 protocol=udp
```

```
add action=accept chain=input
comment="Allow Ping TO
Router from LAN" \
```

```
in-interface=ether2
protocol=icmp
```

```
add action=accept chain=input
comment="Allow DNS from
LAN" dst-port=53 \
```

```
in-interface=ether2
protocol=udp
```

```
add action=accept chain=input
comment="Allow DNS from
LAN (TCP)" dst-port=53 \
```

```
in-interface=ether2
protocol=tcp
```

```
add action=drop chain=input
comment="Drop ALL other
WAN input" in-interface=\
```

```
ether1
```

```
add action=drop chain=input
comment="Drop ALL other
input"
```

```
add action=accept chain=forward
comment="Allow
Established/Related" \
```

```
connection-
state=established,related
```

```
add action=drop chain=forward
comment="Drop access TO R1
from VPN" \
```

```
dst-address=192.168.1.1 in-
interface=wireguard1
```

```
add action=accept chain=forward
comment="Allow LAN TO
VPN" in-interface=\
```

```
ether2 out-
interface=wireguard1
```

```
add action=accept chain=forward
comment="Allow VPN TO
LAN" in-interface=\
```

```
wireguard1 out-
interface=ether2
```

```
add action=accept chain=forward
comment="Allow LAN TO
WAN" in-interface=\
```

```
ether2 out-interface=ether1
```

```
add action=drop chain=forward
comment="Drop Invalid
Connections" \
```

```
connection-state=invalid
```

```
add action=drop chain=forward
comment="Drop ALL other
forward"
```

```
/ip firewall nat
```

```
add action=masquerade
chain=srcnat out-interface=ether1
```

```
/ip route
```

```
add dst-address=192.168.2.0/24
gateway=wireguard1
```

```
/system identity
```

```
set name=R1
```

```
[admin@R1] >
```

R-RO-1 (R2):

```
[admin@R2] > /export
# 2025-10-27 17:54:15 by
RouterOS 7.20.2
# system id = M/hpWjw0DQO
#
/interface ethernet
set [ find default-name=ether1 ]
disable-running-check=no
```

```

set [ find default-name=ether2 ]
disable-running-check=no

/interface wireguard

add listen-port=12321 mtu=1420
name=wireguard1

/port

set 0 name=serial0

/interface wireguard peers

add allowed-
address=192.168.1.0/24 endpoint-
address=192.168.50.9 \

    endpoint-port=12321
interface=wireguard1
name=peer1 public-key=\

"vmRQf1ASwBQqm/HgqFPNrP
z/+Tyh99m5bASkxgTWO="

/ip address

add address=192.168.2.1/24
interface=ether2
network=192.168.2.0

add address=10.0.0.2/30
interface=wireguard1
network=10.0.0.0

add address=192.168.50.10/24
interface=ether1
network=192.168.50.0

/ip dhcp-client

add interface=ether1

/ip dns

set allow-remote-requests=yes
servers=8.8.8.8,1.1.1.1

/ip firewall filter

add action=accept chain=input
comment="Allow
Established/Related" \

    connection-
state=established,related

add action=accept chain=input
comment="Allow WireGuard
UDP from WAN" \

    dst-port=12321 in-
interface=ether1 protocol=udp

```

```

add action=accept chain=input
comment="Allow Ping TO
Router from LAN" \

    in-interface=ether2
protocol=icmp

add action=accept chain=input
comment="Allow DNS from
LAN" dst-port=53 \

    in-interface=ether2
protocol=udp

add action=accept chain=input
comment="Allow DNS from
LAN (TCP)" dst-port=53 \

    in-interface=ether2
protocol=tcp

add action=drop chain=input
comment="Drop ALL other
WAN input" in-interface=\

    ether1

add action=drop chain=input
comment="Drop ALL other
input"

add action=accept chain=forward
comment="Allow
Established/Related" \

    connection-
state=established,related

add action=drop chain=forward
comment="Drop access TO R2
from VPN" \

    dst-address=192.168.2.1 in-
interface=wireguard1

add action=accept chain=forward
comment="Allow LAN TO
VPN" in-interface=\

    ether2 out-
interface=wireguard1

add action=accept chain=forward
comment="Allow VPN TO
LAN" in-interface=\

    wireguard1 out-
interface=ether2

add action=accept chain=forward
comment="Allow LAN TO
WAN" in-interface=\

    ether2 out-interface=ether1

```

```

add action=drop chain=forward
comment="Drop Invalid
Connections" \

    connection-state=invalid

add action=drop chain=forward
comment="Drop ALL other
forward"

/ip firewall nat

add action=masquerade
chain=srcnat out-interface=ether1

/ip route

add dst-address=192.168.1.0/24
gateway=wireguard1

/system identity

set name=R2

[admin@R2] >

Скрипт для налаштування Serv-
Docker-Session (VNC session)
full-setup.sh:

#!/bin/bash

# --- CONFIGURATION ---

# Set the default password for all
users (system and VNC)

DEFAULT_PASS="fominpass"

# --- END CONFIGURATION --
-

# Check if running as root

if [ "$EUID" -ne 0 ]; then

    echo "Please run this script with
sudo privileges (e.g., sudo
./full_vnc_setup.sh)"

    exit 1

fi

echo "--- [1/6] System Update
and Upgrade ---"

```

```

apt update

apt upgrade -y

echo "--- [2/6] Installing Core
Packages (XFCE, VNC, Dbus) --
_"

# Install the necessary packages
for the VNC/XFCE environment

apt install -y xfce4 xfce4-goodies
tigervnc-standalone-server
tigervnc-common dbus-x11

echo "--- [3/6] Installing Firefox
(Non-Snap version) ---"

# Add Mozilla PPA to get the
.deb version of Firefox

add-apt-repository
ppa:mozillateam/ppa -y

# Create a preference file to
prioritize the PPA version over
the Snap

echo '

Package: *

Pin: release o=LP-PPA-
mozillateam

Pin-Priority: 1001

' > /etc/apt/preferences.d/mozilla-
firefox

# Update apt lists and install
Firefox

apt update

apt install -y firefox

echo "--- [4/6] Creating and
Configuring 5 Users ---"

# Encrypt the password for the
'useradd' command

ENCRYPTED_PASS=$(openssl
passwd -1 "$DEFAULT_PASS")

for i in {1..5}
do
    USER_NAME="user$i"

    HOME_DIR="/home/$USER_N
AME"

    VNC_DIR="$HOME_DIR/.vnc"

    echo "--- Processing
$USER_NAME ---"

    # 1. Create the user (if they don't
already exist)

    if id "$USER_NAME"
    &>/dev/null; then

        echo "User $USER_NAME
already exists. Setting password."

        # Set password for existing
user

        usermod -p
"$ENCRYPTED_PASS"
"$USER_NAME"

    else

        echo "Creating user
$USER_NAME..."

        # Create user with encrypted
password, home dir, and bash
shell

        useradd -m -p
"$ENCRYPTED_PASS" -s
/bin/bash "$USER_NAME"

    fi

    # 2. Run all VNC setup
commands *as the user*

    echo "Configuring VNC for
$USER_NAME..."

    su - "$USER_NAME" -c "

        mkdir -p $VNC_DIR;

        # Set the VNC password non-
interactively

        (echo \"\$DEFAULT_PASS\";
echo \"\$DEFAULT_PASS\") |
vncpasswd;

        # Create the xstartup file

        echo '#!/bin/sh' >
$VNC_DIR/xstartup;

        echo 'unset
SESSION_MANAGER' >>
$VNC_DIR/xstartup;

        echo 'unset
DBUS_SESSION_BUS_ADDRE
SS' >> $VNC_DIR/xstartup;

        echo 'exec dbus-launch xfce4-
session' >> $VNC_DIR/xstartup;

        # Make the xstartup file
executable

        chmod +x $VNC_DIR/xstartup

        "

        echo "--- $USER_NAME
configured successfully ---"

    done

    echo "--- [5/6] Creating and
Enabling Systemd Services ---"

    for i in {1..5}
    do
        USER_NAME="user$i"

        DISPLAY_PORT=":$i"

        SERVICE_FILE="/etc/systemd/s
ystem/vncserver@$i.service"

        echo "Creating
$SERVICE_FILE..."

        # Use 'tee' to write the systemd
service file as root

```

```

tee "$SERVICE_FILE" >
/dev/null << EOF

[Unit]

Description=TigerVNC service
for $USER_NAME on display
$DISPLAY_PORT

After=syslog.target
network.target

[Service]

Type=forking

User=$USER_NAME

Group=$USER_NAME

WorkingDirectory=/home/$USER_NAME

# Kill any old session if it exists

ExecStartPre=/usr/bin/vncserver
-kill $DISPLAY_PORT

# Start command (adjust
geometry as needed)

ExecStart=/usr/bin/vncserver
$DISPLAY_PORT -localhost no

# Stop command

ExecStop=/usr/bin/vncserver -kill
$DISPLAY_PORT

[Install]

WantedBy=multi-user.target

EOF

# Enable the service to start on
boot

systemctl enable
"vncserver@$i.service"

done

echo "--- [6/6] Reloading
Systemd and Starting Services ---
"
```

```

systemctl daemon-reload

# Start all 5 services now

for i in {1..5}
do

    systemctl start
    "vncserver@$i.service"

done

echo
"=====
=====
=====
"

echo "ALL DONE!"

echo "Your 5 VNC sessions are
running."

echo "You can connect using a
VNC client:"

echo "user1 -> YOUR_IP:1 (Port
5901)"

echo "user2 -> YOUR_IP:2 (Port
5902)"

echo "user3 -> YOUR_IP:3 (Port
5903)"

echo "user4 -> YOUR_IP:4 (Port
5904)"

echo "user5 -> YOUR_IP:5 (Port
5905)"

echo ""

echo "The password for all users
(system and VNC) is:
$DEFAULT_PASS"

echo
"=====
=====
=====
"

Скрипт для налаштування Serv-
Ubuntu-Docker setup-docker.sh:

#!/bin/bash

# Check if running as root
```

```

if [ "$EUID" -ne 0 ]; then

    echo "Please run this script with
sudo privileges (e.g., sudo
./install_docker.sh)"

    exit 1

fi

echo "--- [1/6] Updating and
Upgrading System Packages ---"

apt update

apt upgrade -y

echo "--- System update complete
---"

echo "--- [2/6] Installing
Prerequisite Packages ---"

# Install packages to allow apt to
use a repository over HTTPS

apt install -y ca-certificates curl
gnupg

echo "--- Prerequisites installed --
-"

echo "--- [3/6] Adding Docker's
Official GPG Key ---"

# Create the keyring directory

install -m 0755 -d
/etc/apt/keyrings

# Download and add Docker's
GPG key

curl -fsSL
https://download.docker.com/linu
x/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

# Set permissions for the key

chmod a+r
/etc/apt/keyrings/docker.asc

echo "--- GPG Key added ---"

echo "--- [4/6] Setting Up the
Docker Repository ---"
```

```
# Add the Docker repository to
Apt sources

echo \

"deb [arch=$(dpkg --print-
architecture) signed-
by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linu
x/ubuntu \

$(. /etc/os-release && echo
"$VERSION_CODENAME")
stable" | \

tee
/etc/apt/sources.list.d/docker.list
> /dev/null

echo "--- Repository setup
complete ---"

echo "--- [5/6] Installing Docker
Engine and Docker Compose ---"

# Update apt package index again
to include the new Docker repo

apt update

# Install the latest versions

apt install -y docker-ce docker-ce-
cli containerd.io docker-buildx-
plugin docker-compose-plugin

echo "--- Docker installation
complete ---"

echo "--- [6/6] (Optional) Adding
current user to the 'docker' group
---"

# This allows running 'docker'
commands without 'sudo'

# We check if $SUDO_USER is
set, which holds the name of the
user who invoked sudo

if [ -n "$SUDO_USER" ]; then

    usermod -aG docker
"$SUDO_USER"

    echo "User '$SUDO_USER'
has been added to the 'docker'
group."
```

```
echo "!!! You MUST log out
and log back in for this change to
take effect !!!"

else

    echo "Could not find original
user. Skipping adding user to
'docker' group."

    echo "You can add your user
manually with: sudo usermod -aG
your_username"

fi

echo "--- Docker setup is
finished! ---"

Скрипт entrypoint.sh для
Docker:

#!/bin/bash

set -e

# Wait for wazuh-agent to be
ready if it's already running

# (This handles container restarts)

if [ -f /var/ossec/bin/wazuh-
control ]; then

    /var/ossec/bin/wazuh-control
stop > /dev/null 2>&1 || true

    sleep 2

fi

# --- VNC Setup ---

# Check if VNC_PASS is set,
otherwise use a default

VNC_PASS=${VNC_PASS:-
"password"}

echo "Setting VNC password..."

mkdir -p /home/vncuser/.vnc

# Set the VNC password non-
interactively

(echo "$VNC_PASS"; echo
"$VNC_PASS") | su - vncuser -c
"vncpasswd"

# Secure the password file
```

```
chmod 600
/home/vncuser/.vnc/passwd

chown vncuser:vncuser
/home/vncuser/.vnc/passwd

# --- Wazuh Setup ---

# Check if WAZUH_MANAGER
IP is set, otherwise use
placeholder

WAZUH_MANAGER_IP=${W
AZUH_MANAGER:-
"192.168.1.1"} # Default, change
in compose

echo "Configuring Wazuh Agent
to connect to
$WAZUH_MANAGER_IP..."

# Use awk to replace the
placeholder IP in the Wazuh
config

awk -v
ip="$WAZUH_MANAGER_IP"
'{gsub(/<address>MANAGER_I
P</address>/,"<address>" ip
"</address>"); print}'
/var/ossec/etc/ossec.conf >
/var/ossec/etc/ossec.conf.tmp &&
mv /var/ossec/etc/ossec.conf.tmp
/var/ossec/etc/ossec.conf

# Set permissions for supervisor
logs

mkdir -p /var/log/supervisor

chown vncuser:vncuser
/var/log/supervisor

# --- Ensure Correct Permissions -
--

echo "Ensuring correct
permissions for /home/vncuser..."

# Ensure vncuser owns
everything in their home directory

chown -R vncuser:vncuser
/home/vncuser

# Ensure VNC directory exists
and has correct permissions

mkdir -p /home/vncuser/.vnc

chmod 700 /home/vncuser/.vnc
```

```

chown -R vncuser:vncuser
/home/vncuser/.vnc

# Ensure xstartup is executable

if [ -f /home/vncuser/.vnc/xstartup
]; then

    chmod +x
/home/vncuser/.vnc/xstartup

fi

# Ensure VNC password file is
readable only by the user

if [ -f /home/vncuser/.vnc/passwd
]; then

    chmod 600
/home/vncuser/.vnc/passwd

    chown vncuser:vncuser
/home/vncuser/.vnc/passwd

fi

# Fix potential issues with /tmp
for X11 sockets (less likely but
possible)

chmod 1777 /tmp

# --- Permissions check done ---

# --- Start Supervisor ---

echo "Starting Supervisor..."

exec /usr/bin/supervisord -n -c
/etc/supervisor/supervisord.conf

```

Конфігураційний файл
supervisord.conf для Docker:

```

[supervisord]

nodaemon=true

logfile=/var/log/supervisor/superv
isord.log

pidfile=/var/run/supervisord.pid

[program:wazuh]

# Wazuh agent must run as root

command=/var/ossec/bin/wazuh-
control start

```

```

user=root

autostart=true

autorestart=true

startretries=10

stdout_logfile=/var/log/supervisor
/wazuh.log

stderr_logfile=/var/log/supervisor
/wazuh.err

[program:vnc]

# Pass display as the first
argument to vncserver

command=/usr/bin/vncserver -fg
-localhost no -passwd
/home/vncuser/.vnc/passwd :1

user=vncuser

autostart=true

autorestart=true

startretries=10

stdout_logfile=/var/log/supervisor
/vnc.log

stderr_logfile=/var/log/supervisor
/vnc.err

# Explicitly set the HOME
directory for the VNC process

environment=HOME="/home/vn
cuser",USER="vncuser"

```

Конфігураційний файл
Dockerfile для Docker:

```

# --- 1. Base Image ---

FROM ubuntu:24.04

# --- 2. Environment Variables ---

# Set non-interactive mode for apt
to prevent build from hanging

ENV
DEBIAN_FRONTEND=noninter
active

# Set default VNC screen
resolution

```

```

ENV
VNC_RESOLUTION=1920x108
0

ENV VNC_PORT=5901

ENV VNC_DISPLAY=:1

```

--- 3. Install All Dependencies
in a Single Layer ---

```

RUN apt update && \

    apt install -y \

        ca-certificates curl gnupg
software-properties-common \

        sudo \

        supervisor \

        xfce4 xfce4-goodies \

        tigervnc-standalone-server
tigervnc-common \

        dbus-x11 \

        # Add Mozilla PPA for non-
Snap Firefox (to match Shed 2)

        && add-apt-repository
ppa:mozillateam/ppa -y \

        && echo 'Package: *' >
/etc/apt/preferences.d/mozilla-
firefox \

        && echo 'Pin: release o=LP-
PPA-mozillateam' >>
/etc/apt/preferences.d/mozilla-
firefox \

        && echo 'Pin-Priority: 1001'
>> /etc/apt/preferences.d/mozilla-
firefox \

        # Install Firefox

        && apt update \

        && apt install -y firefox \

        # Add Wazuh Agent repository
and key

        && curl -s
https://packages.wazuh.com/key/
GPG-KEY-WAZUH | \

        gpg --dearmor -o
/usr/share/keyrings/wazuh-
archive-keyring.gpg && \

```

```

    echo "deb [signed-
by=/usr/share/keyrings/wazuh-
archive-keyring.gpg]
https://packages.wazuh.com/4.x/a
pt/ stable main" | \

tee
/etc/apt/sources.list.d/wazuh.list \

# Install Wazuh Agent

&& apt update \

&& apt install -y wazuh-agent \

# --- 4. Cleanup ---

&& apt-get clean \

&& rm -rf /var/lib/apt/lists/*


# --- 5. User Setup ---

# Create a non-root user to run
VNC

RUN useradd -m -s /bin/bash
vncuser && \

    mkdir -p /home/vncuser/.vnc
&& \

    chown -R vncuser:vncuser
/home/vncuser && \

    # Give vncuser passwordless
sudo rights (needed by
supervisor)

    echo "vncuser ALL=(ALL)
NOPASSWD:ALL" >
/etc/sudoers.d/vncuser


# --- 6. VNC xstartup
Configuration ---

# Create the xstartup script as
vncuser (to match Shed 2)

USER vncuser

RUN echo '#!/bin/sh' >
/home/vncuser/.vnc/xstartup && \

    echo 'unset
SESSION_MANAGER' >>
/home/vncuser/.vnc/xstartup && \

    echo 'unset
DBUS_SESSION_BUS_ADDRE
SS' >>
/home/vncuser/.vnc/xstartup && \

```

```

    echo 'exec dbus-launch xfce4-
session' >>
/home/vncuser/.vnc/xstartup && \

    chmod 700
/home/vncuser/.vnc/xstartup

USER root


# --- 7. Supervisor & Entrypoint
Setup ---

# Copy the config files into the
image

COPY supervisord.conf
/etc/supervisor/conf.d/supervisord
.conf

COPY entrypoint.sh
/usr/local/bin/entrypoint.sh


# Make entrypoint executable

RUN chmod +x
/usr/local/bin/entrypoint.sh


# --- 8. Final Steps ---

# Expose the VNC port

EXPOSE 5901


# Set the entrypoint to run our
setup script

ENTRYPOINT
["/usr/local/bin/entrypoint.sh"]


Конфігураційний файл docker-
compose.yml для Docker
compose:

#
=====

# Docker Compose for 5 VNC
Session Containers

#
=====

```

```

version: '3.8'

services:

    vnc-host1:

        build: . # Build from the
Dockerfile in this directory

        container_name: vnc-host1

        ports:

            # Map host port 5901 to
container port 5901
(VNC_DISPLAY=:1)

            - "5901:5901"

        environment:

            - VNC_PASS=user1pass #
Unique VNC password

            -
WAZUH_MANAGER=192.168.
1.21 # IP of your SIEM server

        restart: unless-stopped

        volumes:

            - vncdata1:/home/vncuser #
Persistent storage for user1


    vnc-host2:

        build: .

        container_name: vnc-host2

        ports:

            # Map host port 5902 to
container port 5901

            - "5902:5901"

        environment:

            - VNC_PASS=user2pass

            -
WAZUH_MANAGER=192.168.
1.21

        restart: unless-stopped

        volumes:

            - vncdata2:/home/vncuser

```

vnc-host3:	- "5905:5901"	<field name="win.system.eventID">^46 24\$</field>
build: .	environment:	
container_name: vnc-host3	- VNC_PASS=user5pass	<field name="win.eventdata.logonType" >^10\$</field>
ports:	-	<description>Successful RDP Logon.</description>
# Map host port 5903 to container port 5901	WAZUH_MANAGER=192.168. 1.21	
- "5903:5901"	restart: unless-stopped	
environment:	volumes:	<group>authentication_success,p ci_dss_10.2.5,</group>
- VNC_PASS=user3pass	- vncdata5:/home/vncuser	</rule>
-		
WAZUH_MANAGER=192.168. 1.21	# Define the named volumes for persistent storage	<!-- Rule for Failed RDP Logon -->
restart: unless-stopped	volumes:	
volumes:	vncdata1:	<!-- Looks for Event ID 4625 (Failed Logon) potentially related to RDP -->
- vncdata3:/home/vncuser	vncdata2:	
	vncdata3:	<!-- Note: Logon Type for failure might not always be 10, so we focus on the Event ID -->
vnc-host4:	vncdata4:	<rule id="100101" level="6">
build: .	vncdata5:	<!-- Rule for Multiple RDP Logon Failures (Potential Brute- Force) -->
container_name: vnc-host4		
ports:	Набір власних правил custom- rules.xml для Wazuh:	<!-- Triggers if rule 100101 (Failed RDP) fires 5 times in 60 seconds from the same source IP - ->
# Map host port 5904 to container port 5901	<group name="windows,rdp,vnc,authenti cation,">	<!-- Rule for Successful RDP Logon -->
- "5904:5901"		
environment:		
- VNC_PASS=user4pass		
-		
WAZUH_MANAGER=192.168. 1.21	RDP Rules (Windows)	<!-- Rule for Multiple RDP Logon Failures (Potential Brute- Force) -->
restart: unless-stopped		
volumes:		
- vncdata4:/home/vncuser		
vnc-host5:		
build: .		
container_name: vnc-host5		
ports:		
# Map host port 5905 to container port 5901		

```
<rule id="100102" level="15"
frequency="5" timeframe="60">
```

```
<if_matched_sid>100101</if_matched_sid>
```

```
<same_field>win.eventdata.ipAddress</same_field>
```

```
<description>Multiple RDP
Logon Failures from same source
IP (Potential Brute-
Force).</description>
```

```
<mitre>
```

```
<id>T1110</id> <!-- Brute
Force Mitre ATT&CK ID -->
```

```
</mitre>
```

```
<group>authentication_failures,pci_dss_10.2.4,pci_dss_10.2.5,gdp
r_IV_35.7.d,</group>
```

```
</rule>
```

```
<!-- =====
VNC Rules (Linux - Using
Custom Decoders)
===== -->
```

```
<!-- Rule for Successful VNC
Authentication -->
```

```
<rule id="100200" level="10">
```

```
<decoded_as>vnc-
accepted</decoded_as> <!-- Use
our custom decoder name -->
```

```
<description>Successful VNC
connection established from
$(srcip).</description>
```

```
<group>authentication_success,</
group>
```

```
</rule>
```

```
<!-- Rule for Failed VNC
Authentication -->
```

```
<rule id="100201" level="11">
```

```
<decoded_as>vnc-auth-
failure</decoded_as> <!-- Use
our custom decoder name -->
```

```
<description>Failed VNC
authentication attempt from
$(srcip).</description>
```

```
<group>authentication_failed,</g
roup>
```

```
</rule>
```

```
<!-- Rule for Multiple VNC
Logon Failures (Potential Brute-
Force) -->
```

```
<rule id="100202" level="15"
frequency="3" timeframe="60">
```

```
<if_matched_sid>100201</if_ma
tched_sid>
```

```
<same_field>srcip</same_field>
<!-- This should work now if the
decoder extracts srcip -->
```

```
<description>Multiple VNC
Logon Failures from same source
IP $(srcip) (Potential Brute-
Force).</description>
```

```
<mitre>
```

```
<id>T1110</id>
```

```
</mitre>
```

```
<group>authentication_failures,</
group>
```

```
</rule>
```

```
<!-- (Optional) Rule for
Blacklisted IP -->
```

```
<rule id="100203" level="12">
```

```
<decoded_as>vnc-
blacklisted</decoded_as>
```

```
<description>VNC Server
blacklisted IP $(srcip) due to
failed attempts.</description>
```

```
<group>ids,recon,</group>
```

```
</rule>
```

```
<!-- Rule for Multiple VNC
Logon Failures From Blacklisted
IP(Potential Brute-Force) -->
```

```
<rule id="100204" level="15"
frequency="3" timeframe="60">
```

```
<if_matched_sid>100203</if_ma
tched_sid>
```

```
<same_field>srcip</same_field>
<!-- This should work now if the
decoder extracts srcip -->
```

```
<description>Multiple VNC
Logon Failures from same
Blacklisted source IP $(srcip)
(Potential Brute-
Force).</description>
```

```
<mitre>
```

```
<id>T1110</id>
```

```
</mitre>
```

```
<group>ids,recon,authentication_
failures,</group>
```

```
</rule>
```

```
</group>
```

Власний декодер **custom-decoder.xml** для Wazuh:

```
<decoder name="vnc-auth-
failure">
```

```
<prematch>AuthFailureExceptio
n: Authentication
failure|closing</prematch>
```

```
<regex>AuthFailureException:
Authentication failure|closing
(\\d+\\.\\d+\\.\\d+\\.\\d+):.\\d+:
Authentication failure</regex>
```

```
<order>srcip</order>
```

```
</decoder>
```

```
<!-- Connection accepted -->
```

```
<decoder name="vnc-accepted">
```

```
<prematch>accepted:</prematch>  
>
```

```
  <regex>accepted:  
(\d+\.\d+\.\d+\.\d+):.\d+:</regex>
```

```
  <order>srcip</order>
```

```
</decoder>
```

```
<!-- Blacklisted IP -->
```

```
<decoder name="vnc-  
blacklisted">
```

```
<prematch>blacklisted:</prematch>  
h>
```

```
<regex>blacklisted:\s+(\d+\.\d+\.\d+\.\d+)</regex>
```

```
  <order>srcip</order>
```

```
</decoder>
```