

СИСТЕМА АВТОМАТИЗОВАНОГО КОНТРОЛЮ ВІДПОВІДНОСТІ КОНФІГУРАЦІЙ МЕРЕЖЕВИХ ПРИСТРОЇВ ВИМОГАМ КІБЕРБЕЗПЕКИ

Збільшення кількості підключених пристроїв, різноманіття й ускладнення мережеских топологій та постійний розвиток кіберзагроз дуже ускладнюють адміністрування мережевого обладнання. Налаштування десятків або сотень маршрутизаторів і комутаторів вручну є трудомістким, дорогим і пов'язаним з високим ризиком кіберзагроз. Саме помилки конфігурації є причиною успіху більшості кібератак. Метою дослідження є розроблення та практична апробація системи для автоматизованої валідації конфігурацій мережеских пристроїв. Зокрема, система має автоматично ідентифікувати фізичну топологію мережі, перевіряти відповідність налаштувань галузевим стандартам безпеки та автоматизовано усувати виявлені вразливості.

Систему реалізовано мовою Python за допомогою бібліотек Netmiko та PyVis. Для отримання інформації про топологію застосовано протокол LLDP (IEEE 802.1AB). Перевірку конфігурацій здійснено відповідно до 10 правил CIS Benchmark для Cisco IOS. Апробацію системи здійснено на віртуальному стенді GNS3, що складався з маршрутизаторів Cisco IOS, класифікованих у зони безпеки (Access, Restricted, DMZ) з динамічною маршрутизацією OSPF.

Модуль аудиту виявив низку вразливих налаштувань (зокрема, активований Telnet), внаслідок чого початкова оцінка відповідності стандартам безпеки склала 85%. Після автоматизованого застосування виправлень та генерації розширених списків контролю доступу (ACL) оцінку підвищено до 100%. Крім цього, система успішно заблокувала спробу несанкціонованого доступу з зони Restricted до сервера баз даних, що підтвердило коректність згенерованих політик фільтрації.

Запропоноване рішення значною мірою скорочує час проведення аудиту, усуває вплив суб'єктивного людського чинника та гарантує послідовне впровадження заходів безпеки. Отримані результати дослідження підтверджують, що принципи NetDevOps досить ефективно працюють при розв'язанні задач забезпечення захисту реальних мереж.

Ключові слова: автоматизація адміністрування; аудит безпеки; списки контролю доступу; CIS Benchmark; NetDevOps; сегментація мережі; Zero Trust; мережескі пристрої

Вступ та постановка проблеми. Мережева інфраструктура є основою функціонування будь-яких компаній. Зростання кількості підключених пристроїв, складність топологій та еволюція кіберзагроз є новими викликами для адміністраторів мереж. Ручне налаштування десятків або сотень різних маршрутизаторів і комутаторів не тільки займає багато часу, але й є дорогим. Це основна причина критичних неправильних конфігурацій, що в більшості випадків призводять до успішних кібератак. У практиці адміністрування такі помилки часто виникають не через брак знань, а через людський чинник, коли одну й ту саму дію доводиться виконувати багаторазово, зокрема, вручну.

Відповідно до звіту Verizon Data Breach Investigations Report [1], понад 80 % порушень безпеки мережевої інфраструктури ґрунтуються на неправильній конфігурації обладнання або використанні застарілих протоколів управління. Яскравим прикладом цього є протокол Telnet, облікові дані якого передаються у відкритому форматі, який досі зустрічається в корпоративних мережах попри наявність безпечних альтернатив. Недостатня сегментація мережі за функціональними зонами безпеки та погано налаштовані правила фільтрації трафіку дають зловмисникам надто легкий доступ. Після первинного проникнення вони здатні вільно пересуватися інфраструктурою організації. Відтак, контроль конфігураційних налаштувань виходить за межі простої технічної задачі. Це ще й значний організаційний виклик, адже його ефективність прямо пропорційна регулярності проведених перевірок.

Для забезпечення справді високого рівня безпеки у сучасних мережах необхідно автоматизувати рутинні адміністративні завдання. До них належать інвентаризація, збір топологічних даних та, звичайно ж, перевірка конфігураційних налаштувань. Концепція NetDevOps, що успішно перейняла найкращі практики DevOps, пропонує розглядати мережеву інфраструктуру як своєрідний код – "Infrastructure as Code". Це дозволяє автоматично тестувати, версіювати та розгортати її. Проте, на практиці, реалії часто демонструють, що втілити цей підхід на практиці не завжди є можливим. Багато обладнання та чинних процесів і досі прив'язані до традиційного, ручного адміністрування, що неминуче породжує серйозні перешкоди на шляху до повної автоматизації.

Метою статті є обґрунтування архітектури та детальне представлення результатів її практичного випробування. Йдеться про автоматизовану валідацію конфігурацій мережевих пристроїв, яка інтегрує кілька ключових функцій: збір даних про фізичну топологію, перевірку на відповідність стандартам CIS Benchmark [2] та, що особливо важливо, автоматизоване виправлення виявлених невідповідностей відповідно до визначених політик безпеки. Запропонований цілісний підхід, який ми пропонуємо покликаний відчутно зменшити операційні ризики, зробити інфраструктуру значно прозорішою та мінімізувати час, необхідний для ліквідації виявлених вразливостей. Водночас авторами підкреслюється, що систему варто розглядати не як повноцінну заміну кваліфікованого адміністратора, а радше як потужний інструмент підтримки його рішень, здатний ефективно виконувати рутинні, стандартні перевірки.

Для досягнення мети, поставлено кілька завдань:

- Детально порівняти існуючі методи фільтрації трафіку та різноманітні засоби автоматизації конфігурування мережевих систем.
- Розробити та реалізувати програмні модулі, що відповідають за збір топології мережі, проведення аудиту, а також автоматизоване усунення виявлених недоліків.
- Провести експериментальну перевірку ефективності створеної системи на підготовленому віртуальному стенді.

Практична цінність полягає у створенні програмного рішення для адміністрування, котре не тільки автоматизує складні перевірки безпеки, але й суттєво спрощує виконання численних рутинних завдань, пов'язаних із налаштуванням і підтримкою мережевої інфраструктури.

Огляд літератури (аналіз останніх досліджень та публікацій). Проблема автоматизованої валідації конфігурацій мережевого обладнання попри численні дослідження, залишається актуальною та досі не знайшла свого остаточного розв'язання. Така ситуація беззаперечно підкреслює нагальну потребу в ґрунтовній систематизації вже існуючих наукових напрацювань. Саме ретельний аналіз останніх публікацій та досліджень дозволив ідентифікувати кілька ключових, взаємопов'язаних векторів наукового пошуку.

Одним із таких напрямів, який привертає значну увагу дослідників, є мережі на основі намірів (Intent-Based Networking, IBN). У межах цього підходу автоматизація конфігурації трактується як трансформація політик високого рівня в перевірені та оптимальні налаштування мережевих пристроїв. Слід зазначити, що оглядові дослідження [3] виявили: ключовою перешкодою для IBN дотепер залишається саме верифікація та уникнення конфліктів політик, а не лише їхнє елементарне розгортання. Ця принципово важлива теза отримала подальший розвиток у пізніших працях, які пропонували методи для ідентифікації конфліктів та гарантування когерентності намірів упродовж усього життєвого циклу конфігурації [4]. Проте, глибокий аналіз доступних літературних джерел переконливо свідчить, що більшість подібних комплексних рішень здебільшого орієнтовані на промислові контролери, інтеграція котрих виявляється винятково трудомісткою, що, своєю чергою, значно обмежує їхню практичну імплементацію в інфраструктурах меншого масштабу.

Другий напрям досліджень охоплює автоматизацію конфігурування та контроль відповідності (compliance). Існує вагома доказова база, яка свідчить про те, що програмне забезпечення, призначене для автоматичного застосування та перевірки конфігурацій, здатне значно зменшити кількість помилок, а також істотно скоротити витрати праці

адміністраторів [5]. Водночас інші автори зазначають, що використання інструментів штучного інтелекту в процесах конфігурації та контролю відповідності відкриває абсолютно нові можливості, але вимагає розробки надійних засобів валідації згенерованих налаштувань [6, 7]. Це підтверджує ключову тезу, згідно з якою автоматизація, позбавлена етапу ретельної перевірки відповідності стандартам, є об'єктивно неповною та потенційно небезпечною.

Третій напрям тісно пов'язаний з концепцією Zero Trust (нульової довіри) та мікросегментацією, що розглядаються як нова модель мережевої безпеки. Огляд літератури чітко встановив основні тенденції та проблеми, що виникають при впровадженні архітектури Zero Trust у різних секторах; він також підтвердив нагальну вимогу жорсткої сегментації доступу між сегментами мережі [8]. Дослідники мікросегментації переконані, що поділ інфраструктури на ізольовані зони з неухильним дотриманням політики найменших привілеїв є надзвичайно ефективним засобом стримування латерального переміщення злоумисника всередині мережі [9]. Саме ці фундаментальні засади сформували підґрунтя для розробленої чотиризонної моделі сегментації, що застосована у цьому дослідженні.

Підсумовуючи викладене, слід з упевненістю констатувати: незважаючи на значний обсяг виконаних робіт у царині автоматизації мережевого адміністрування та зміцнення її безпеки, проблема повної інтеграції трьох критично важливих аспектів: автоматизованого збору відомостей про фізичну топологію, деталізованої перевірки конфігурації згідно з визнаними стандартами (наприклад, CIS Benchmark) та автоматичного застосування коригувальних списків контролю доступу (ACL) в межах єдиного інструменту, що адаптований для потреб невеликих та середніх корпоративних інфраструктур, і надалі залишається актуальною та не вирішеною. Отже, у цій статті ми представляємо результати нашого дослідження, що має на меті ефективно ліквідувати цю прогалину.

Методи дослідження. Методологічну основу дослідження становить поєднання аналізу предметної області та експериментального проектування програмної системи. Нижче схарактеризовано теоретичні засади (методи фільтрації трафіку, технології автоматизації та стандарти захисту) і склад розробленої системи.

Методи фільтрації трафіку та списки контролю доступу. Для забезпечення безпеки сучасних комп'ютерних мереж необхідне комплексне рішення, і одним із найважливіших його елементів є контроль потоку трафіку між різними секціями мережі. Фільтрація мережевого трафіку – це практика аналізу пакетів під час їх проходження через мережевий пристрій (маршрутизатор або комутатор L3), щоб визначити, чи слід їх передавати далі або блокувати відповідно до заздалегідь визначених правил.

Основним і найпоширенішим інструментом для фільтрації на пристроях Cisco IOS є списки контролю доступу (ACL). ACL – це набір правил (permit/deny). Принципово важливим є те, що правила перевіряються зверху вниз послідовно, і перевірка зупиняється, як тільки знайдено відповідність пакету до правила. Крім того, наприкінці кожного ACL Cisco IOS неявно додає правило "deny all", за яким весь трафік, що не відповідає жодному явному дозволу, блокується.

Залежно від рівня деталізації фільтрації, ACL можна класифікувати на два основних типи. Стандартні ACL (Standard ACL, номери 1–99 та 1300–1999) забезпечують аналіз лише IP-адреси джерела пакету і є досить простими у впровадженні. Через свої обмеження вони в основному використовуються для контролю доступу до самих пристроїв або у специфічних сценаріях маршрутизації. Розширені ACL (Extended ACL, номери 100–199 та 2000–2699) пропонують значно більшу гнучкість: вони дозволяють фільтрувати пакети одночасно за IP-адресами джерела та призначення, за протоколом (IP, ICMP, TCP, UDP, OSPF тощо), а також за номерами портів TCP/UDP для визначення конкретних сервісів та додатків.

Ефективна реалізація сегментації мережі передбачає розподіл інфраструктури на ізольовані зони безпеки з різними рівнями привілеїв доступу. У цій роботі застосовується чотиризонна модель:

- OSPF Area 0 (Core) – магістральна зона маршрутизації, що з'єднує всі інші зони;
- Access – зона доступу для корпоративних користувачів, мережа 10.0.10.0/24;

Restricted – зона обмеженого доступу з підвищеними вимогами безпеки, мережа 10.0.30.0/24;

DMZ (Demilitarized Zone) – демілітаризована зона для зовнішньо доступних серверів, мережі 10.0.40.0/24–10.0.42.0/24.

Впровадження ACL дотримується принципу найменших привілеїв, що є ключовим елементом концепції Zero Trust Security. Відповідно до цього принципу, кожен мережевий вузол або сегмент повинен мати доступ лише до ресурсів, необхідних для виконання його функцій (табл. 1). Весь інший трафік забороняється за замовчуванням.

Таблиця 1 – Функціональні зони безпеки випробувального стенду

Зона / сегмент	Адресний простір	Призначення	Політика доступу
Core (OSPF Area 0)	10.10.10.0/30 – 10.10.12.0/30	Магістральна маршрутизація між зонами	Автентифікація OSPF MD5
Access	10.0.20.0/24 – 10.0.21.0/24	Робочі місця корпоративних користувачів (PC1, PC2)	Стандартний доступ
Restricted	10.0.30.0/24	Вузол з підвищеними вимогами (PC3)	Обмежений доступ через ACL
DMZ	10.0.40.0/24 – 10.0.42.0/24	Зовнішньо доступні сервери (WEB, REDIS, DB)	Фільтрація вхідного трафіку

Технології автоматизації налаштування мережевих пристроїв. Коли інфраструктура масштабується, класичний спосіб управління мережею через інтерфейс командного рядка в реальному часі стає нежиттєздатним. Управління навіть кількома десятками пристроїв у такий спосіб є трудомістким і схильним до помилок, що не забезпечує необхідної швидкості реагування на інциденти. Сучасна парадигма NetDevOps надає різноманітні технологічні рішення для автоматизації мережевого адміністрування.

Ansible є однією з найпопулярніших платформ автоматизації, заснованою на декларативних плейбуках YAML. Вона не потребує агентів на цільових пристроях і може застосовуватися на широкому спектрі мережевого обладнання. Її реалізація вимагає увімкнення SSH або REST API на стороні пристрою.

NETCONF/RESTCONF – стандартизовані протоколи управління мережею (RFC 6241 та RFC 8040), що забезпечують структурований, заснований на моделях даних спосіб взаємодії з обладнанням. Формальні моделі даних YANG використовуються для забезпечення однозначності та валідності конфігурацій. Однак ці протоколи підтримуються переважно на сучасних версіях ОС мережевого обладнання.

У цій статті обрано підхід на основі Python-скриптів з використанням бібліотеки Netmiko з кількох причин. По-перше, Netmiko підтримує все обладнання Cisco IOS, яке може не підтримувати NETCONF. По-друге, Python добре підходить для обробки неструктурованих текстових виводів CLI за допомогою регулярних виразів.

Архітектура взаємодії системи з мережевим обладнанням налаштована таким чином: Python-скрипт ініціює окрему SSH-сесію до кожного цільового вузла через бібліотеку Netmiko, яка автоматично обробляє специфіку різних версій Cisco IOS. Далі виконується набір діагностичних команд (show lldp neighbors detail, show running-config тощо), а отриманий текстовий вивід обробляється регулярними виразами для виділення структурованих даних. Зібрані дані зберігаються у вигляді Python-словників та списків для подальшої обробки та візуалізації (рис. 2).

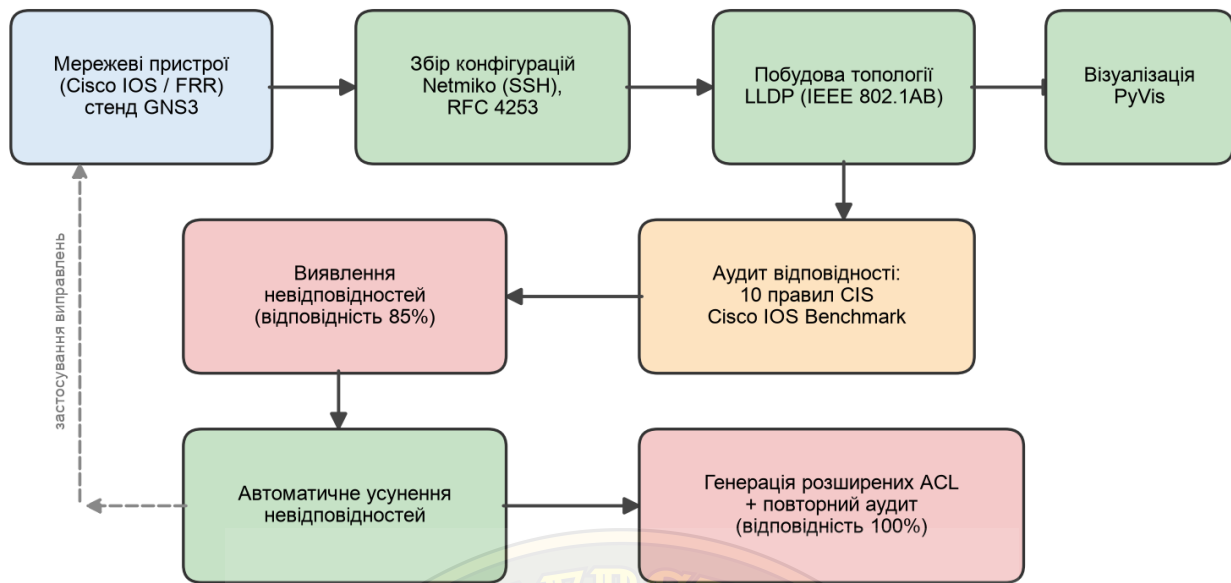


Рисунок 2 – Узагальнена архітектура системи автоматизованого контролю відповідності конфігурацій

Стандарти захисту інфраструктури та кращі практики. Однією з переваг автоматизованого підходу є можливість регулярної перевірки конфігурацій мережевого обладнання на відповідність визнаним галузевим стандартам безпеки. Провідним джерелом таких стандартів є CIS Benchmarks – великі технічні керівництва, розроблені провідними фахівцями у галузі кібербезпеки.

CIS Benchmark для Cisco IOS містить понад 80 підтверджених рекомендацій, що охоплюють усі аспекти безпечної конфігурації маршрутизаторів. Найбільш критичні вимоги, реалізовані в цій роботі.

Обов'язкове відключення протоколу Telnet (transport input none або transport input ssh) на всіх лініях VTY. Telnet передає облікові дані та дані сесії у незашифрованому вигляді, що робить його вразливим до атак типу Man-in-the-Middle.

Використання виключно протоколу SSHv2 [10] як захищеної альтернативи Telnet. SSHv2 забезпечує криптографічно стійке шифрування каналу зв'язку, автентифікацію та цілісність даних.

Захист площини управління (Control Plane Policing) для обмеження обсягу трафіку, спрямованого безпосередньо до процесора маршрутизатора.

Налаштування автоматичного виходу із сесії за неактивністю (exec-timeout), що мінімізує ризик несанкціонованого доступу до відкритої сесії.

Опис складових системи та використаних технологій. Побудована система є програмним комплексом, реалізованим мовою Python3, та складається з чотирьох функціональних модулів, кожен із яких виконує окреме завдання в рамках загального процесу аудиту та управління мережею. Модульна архітектура забезпечує незалежність компонентів та можливість їх окремого використання або розширення.

Перший модуль – модуль SSH-з'єднання та збору даних – відповідає за встановлення захищених сесій до мережевих вузлів і виконання діагностичних команд. Реалізований на базі бібліотеки Netmiko 4.x, яка надає уніфікований інтерфейс для роботи з обладнанням різних виробників та автоматично обробляє специфічні особливості різних версій Cisco IOS.

Другий модуль – модуль побудови та візуалізації топології – обробляє зібрані дані LLDP, формує граф мережевих з'єднань та генерує інтерактивну HTML-візуалізацію за допомогою бібліотеки PyVis 0.3.x [11], яка є Python-обгорткою над JavaScript-бібліотекою vis.js.

Третій модуль – модуль перевірки безпеки – перевіряє завантажені конфігурації пристроїв на відповідність контрольному списку вимог CIS Benchmark та формує структурований звіт про виявлені відхилення.

Четвертий модуль – модуль управління ACL – динамічно генерує розширені ACL відповідно до визначеної політики безпеки та застосовує їх до відповідних інтерфейсів маршрутизаторів через функцію `send_config_set()` бібліотеки Netmiko.

Для тестування системи використовувалося середовище GNS3 [12] – відкрита платформа емуляції мережевого обладнання. Стенд розгорнуто на базі маршрутизаторів FRR – відкритого рішення з підтримкою OSPF, BGP та інших протоколів маршрутизації. Такий підхід дозволяє відтворити реальну поведінку мережевого обладнання без потреби в образах Cisco IOS.

На рисунку 3 зображено головну сторінку (дашборд) веб-інтерфейсу системи SecureNet. У верхній частині відображається загальна статистика мережевої топології: 10 вузлів, 9 каналів зв'язку, 10 пристроїв, з яких 9 активні. Нижній блок призначений для управління Ansible-автоматизацією: кнопка "Запустити Ansible" ініціює виконання playbook `site.yml` через GNS3 API, а журнал виводу оновлюється у реальному часі через WebSocket-з'єднання (Flask-SocketIO).

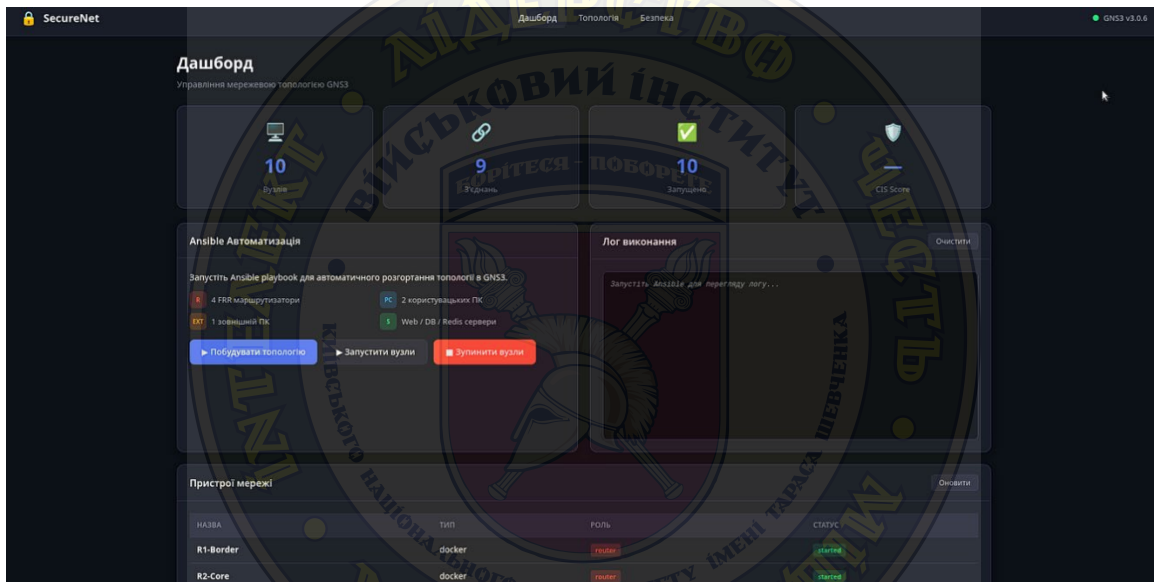


Рисунок 3 – Дашборд системи: статистика топології та блок Ansible-автоматизації

Результати. У цьому розділі викладено результати практичної реалізації та апробації системи за трьома функціональними напрямками: збір і візуалізація топології, автоматизований аудит безпеки та генерація і впровадження коригувальних ACL з подальшим розгортанням оновленої конфігурації.

Збір інформації про топологію та візуалізація зв'язків. Детальна, актуальна карта мережевих з'єднань є основою будь-якого аудиту та управління мережею. Стандартний підхід перевірки доступності через ICMP дає лише інформацію про логічну зв'язність третього рівня моделі OSI і є недостатнім для розуміння повного стану інфраструктури: фізичні з'єднання між пристроями не виявляються, конкретні порти підключення не ідентифікуються, ізольовані сегменти не визначаються.

Для вирішення цього завдання обрано відкритий стандарт IEEE 802.1AB – Link Layer Discovery Protocol. LLDP – це протокол канального рівня, через який мережеві пристрої оголошують свою ідентифікаційну інформацію та отримують аналогічні дані від безпосередньо підключених сусідів. На відміну від пропрієтарного Cisco Discovery Protocol, LLDP є відкритим стандартом, підтримується обладнанням різних виробників і не залежить від стану маршрутизації.

Топологічні дані збираються таким чином: система ініціює SSH-сесію до кожного пристрою зі списку цілей та виконує команду `show lldp neighbors detail`. Ця команда повертає детальну інформацію у форматі TLV про кожного сусіда, де ключовими полями є: System Name (ім'я сусіда), Chassis ID (MAC-адреса), Port ID (порт сусіда) та Local Interface (локальний порт підключення).

Для витягнення структурованих даних з текстового виводу CLI застосовуються регулярні вирази. Отримані пари "локальний порт – сусід – порт сусіда" утворюють ребра графа мережі. За допомогою бібліотеки PyVis будується інтерактивний HTML-граф топології, де вузли різних зон безпеки виділяються різними кольорами, а підписи на ребрах відображають назви портів підключення.

На рисунку 4 зображено список усіх 10 вузлів мережі SecureNet, що формується динамічно через GNS3 REST API. Перелік включає: 4 маршрутизатори на базі FRR (R1-Border, R2-Core, R3-Internal, R4-DMZ), 3 сервери Docker-контейнерів (Web-Server на nginx, DB-Server на PostgreSQL, Redis-Server), а також 3 кінцевих вузли VPCS (PC1-User, PC2-User, PC-External). Для кожного пристрою відображається тип, статус роботи та IP-адреса.

ІМЕНА	ТИП	РОЛЬ	СТАТУС
R1-Border	docker	router	started
R2-Core	docker	router	started
R3-Internal	docker	router	started
R4-DMZ	docker	router	started
Web-Server	docker	server	started
DB-Server	docker	server	started
Redis-Server	docker	server	started
PC-External	vpcs	external_pc	started
PC1-User	vpcs	user_pc	started
PC2-User	vpcs	user_pc	started

Рисунок 4 – Повний список пристроїв мережевої топології

На рисунку 5 зображено сторінку візуалізації мережевої топології, побудованої за допомогою бібліотеки D3.js на основі даних GNS3 API. Вузлами відображено всі 10 пристроїв, з'єднаних 9 каналами зв'язку. Центральну роль виконує R2-Core, до якого підключені: R1-Border (зовнішня мережа 192.168.100.0/24), R3-Internal (внутрішня мережа 10.10.20.0/24) та R4-DMZ (демілітаризована зона 10.10.30.0/24). Клікнувши на вузол, можна побачити деталі пристрою.

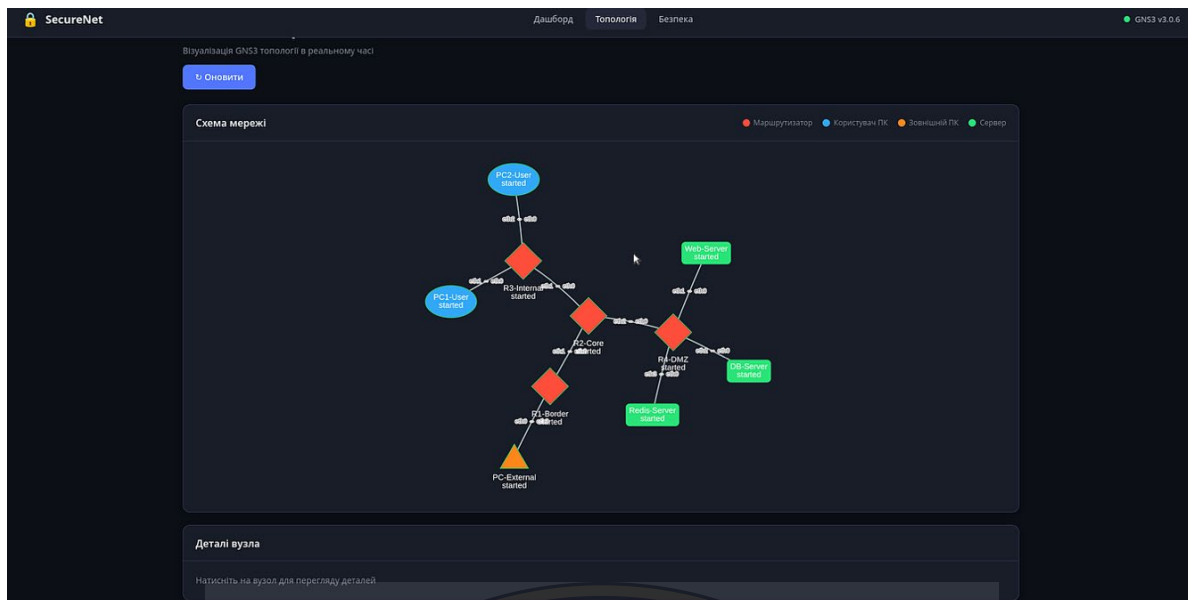


Рисунок 5 – Інтерактивна топологія мережі SecureNet

Результуюча візуалізація стенду наведена на рисунку 6. Топологія включає чотири маршрутизатори на базі FRR (R1–R4): R1 виконує роль ядрового маршрутизатора (Core/Distribution, OSPF Area 0), R2 обслуговує зону Access (мережі 10.0.20.0/24 та 10.0.21.0/24), R3 – зону Restricted (10.0.30.0/24), R4 – зону DMZ (10.0.40–42.0/24). У зоні Access розміщено два хости PC1 та PC2 (легітимні користувачі), у зоні Restricted – PC3 (хост "Зловмисник"). У зоні DMZ розгорнуто три сервери: SRV-WEB (PHP/Apache, 10.0.40.10), SRV-REDIS (Redis 7, 10.0.41.10) та SRV-DB (PostgreSQL 14, 10.0.42.10). Пунктирна лінія між R3 та PC3 позначає заблокований ACL-трафік.

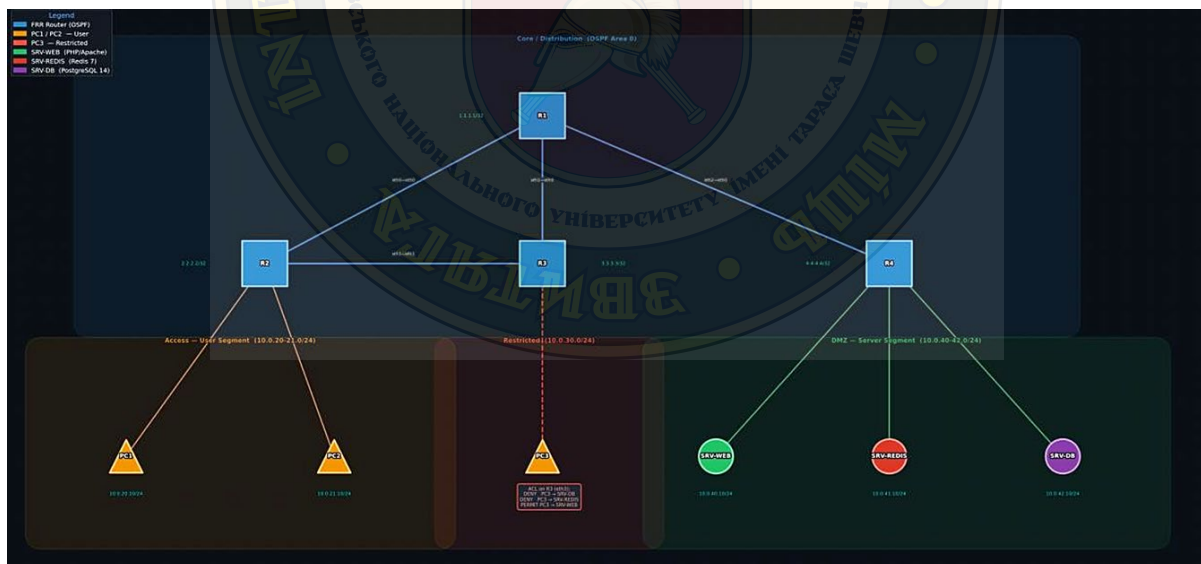


Рисунок 6 – Мережева топологія стенду (OSPF + зони безпеки + ACL)

Результатом роботи модуля є інтерактивна карта мережі, побудована на основі реальних LLDP-даних стенду. Топологія стенду включає чотири маршрутизатори на базі FRRouting: R1 виконує роль ядрового вузла (Core / OSPF Area 0) і з'єднаний з трьома граничними маршрутизаторами. R2 обслуговує зону Access (10.0.20–21.0/24) з хостами PC1 та PC2. R3 – зону Restricted (10.0.30.0/24) з хостом PC3, трафік якого фільтрується ACL. R4 – зону DMZ (10.0.40–42.0/24) з серверами SRV-WEB, SRV-REDIS та SRV-DB.

Зібрані LLDP-зв'язки між пристроями наведено у таблиці 2.

Таблиця 2 – Топологія мережі стенду, зібрана через LLDP

Пристрій А	Інтерфейс А	Пристрій В	Інтерфейс В
R1-Border	eth0	R2-Core	eth0
R1-Border	eth1	R3-Internal	eth0
R1-Border	eth2	R4-DMZ	eth0
R2-Core	eth1	R3-Internal	eth1
R3-Internal	eth3	PC3 (Restricted)	—
R4-DMZ	eth1	SRV-WEB / SRV-REDIS / SRV-DB	—

Автоматизована валідація конфігурацій. Модуль валідації безпеки формує аналітичне ядро розробленої системи. Його завданням є автоматизована перевірка конфігурацій мережевих пристроїв на відповідність попередньо визначеному набору вимог безпеки та формування звіту про виявлені порушення.

Процес валідації починається зі збору повної конфігурації кожного пристрою. Через SSH виконується команда `show running-config`, результат якої зберігається як рядок для подальшого аналізу. Аналіз виконується послідовним застосуванням чек-листу перевірок на базі стандарту CIS Benchmark для маршрутизаторів FRRouting.

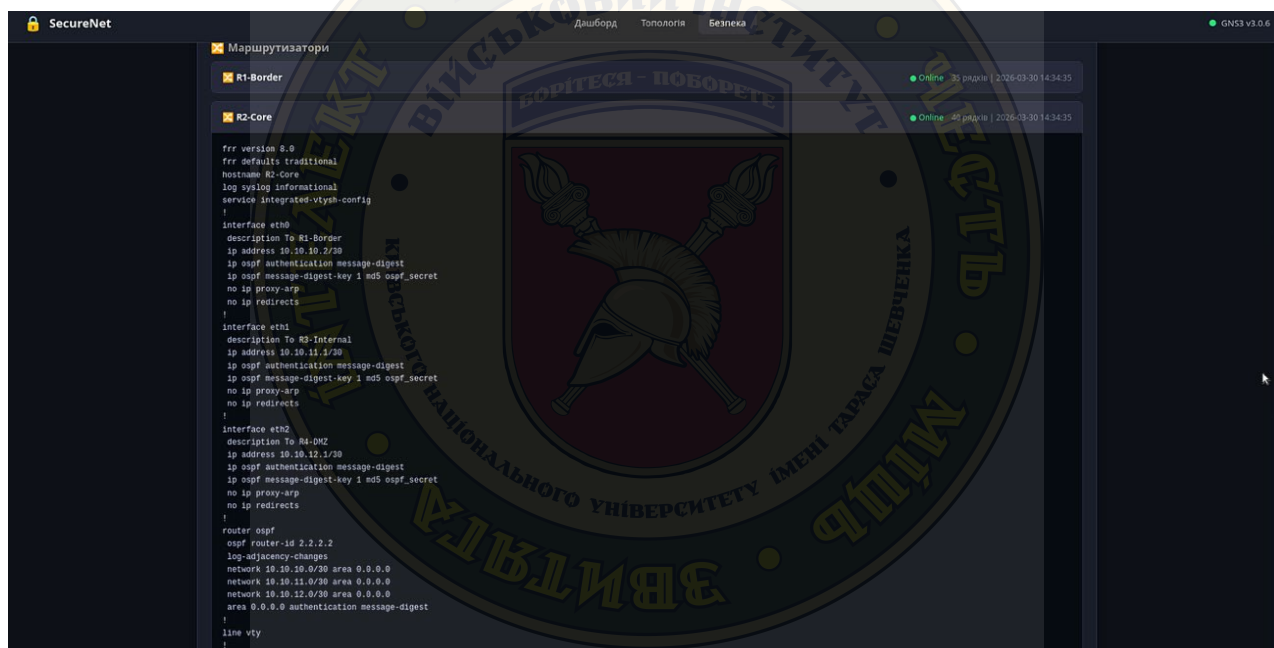


Рисунок 7 – Перегляд FRR-конфігурацій маршрутизаторів R1-Border та R2-Core

На рисунку 7 зображено розгорнуті конфігурації маршрутизаторів у форматі FRR (Free Range Routing), що збираються через GNS3 API з файлу `/etc/frr/frr.conf`. Конфігурація R1-Border містить: інтерфейси eth0/eth1 з OSPF MD5-автентифікацією, `passive-interface eth0`, мережі OSPF та розширений ACL BLOCK-EXTERNAL. Конфігурація R2-Core відображає початковий (небезпечний) стан: відсутній `passive-interface` та ACL – що відповідає правилам CIS-R5, R6, R7 зі статусом FAIL.

За результатами запуску системи 09.03.2026 о 14:21 (timestamp: 20260309_142110) для стенду з чотирьох маршрутизаторів (FRR-Router-1 – FRR-Router-4) отримано зведений звіт: із 40 перевірок (10 правил на 4 маршрутизаторах) 34 пройдено успішно (PASS), 6 – не пройдено (FAIL). Загальний індекс безпеки склав 85 балів зі 100. Детальні результати по кожному правилу наведено у таблиці 3.

Таблиця 3 – Перелік перевірок CIS Benchmark для Cisco IOS та підсумковий статус

ID	Правило CIS Benchmark	Критичність	Статус (після усунення)
CIS-R1	OSPF MD5 Authentication	критичний	ОК
CIS-R2	Logging Enabled	високий	ОК
CIS-R3	No IP Proxy-ARP	середній	ОК
CIS-R4	No IP Redirects	середній	ОК
CIS-R5	OSPF Passive Interface on LAN	високий	ОК
CIS-R6	ACL Configured	високий	ОК
CIS-R7	Telnet Blocked in ACL	критичний	ОК
CIS-R8	OSPF Router-ID Configured	середній	ОК
CIS-R9	Exec-timeout на лініях VTY	середній	ОК
CIS-R10	SSHv2 як транспорт керування	критичний	ОК

Аналіз результатів виявив одне критичне порушення (FAIL з пріоритетом HIGH), спільне для всіх чотирьох маршрутизаторів: відсутність ACL на граничних інтерфейсах (правило CIS-R5). Система автоматично сформувала конкретні команди для виправлення цього порушення. Правило CIS-R9 (SSH Management) позначено як WARN: хоча Telnet вимкнений, SSH-доступ також не налаштовано явно, що ускладнює захищене віддалене адміністрування. Ці результати підтверджують практичну цінність автоматизованого підходу – вручну перевірити 10 правил на 4 пристроях (40 перевірок) займає значно більше часу та є схильним до помилок.

Архітектура модуля розроблена з можливістю розширення: додавання нових перевірок зводиться до написання нового патерну пошуку та відповідного повідомлення без зміни базової логіки.

На рис. 8 зображено результати автоматизованого аудиту безпеки за 10 правилами CIS Benchmark до застосування виправлень. Загальний бал системи – 85%. R1-Border отримав 100% (еталонна конфігурація). R2-Core – 70% (7/10): відсутні passive-interface та ACL блокування telnet/TFTP. R3-Internal – 80% (8/10): відсутній ACL фільтр. R4-DMZ – 90% (9/10): у DMZ-IN відсутнє блокування telnet (порт 23). Виявлені порушення виділені червоним з детальним поясненням. Порівняння рівнів відповідності конфігурацій вимогам CIS Benchmark до та після автоматизованого усунення невідповідностей наведено на рис. 9.

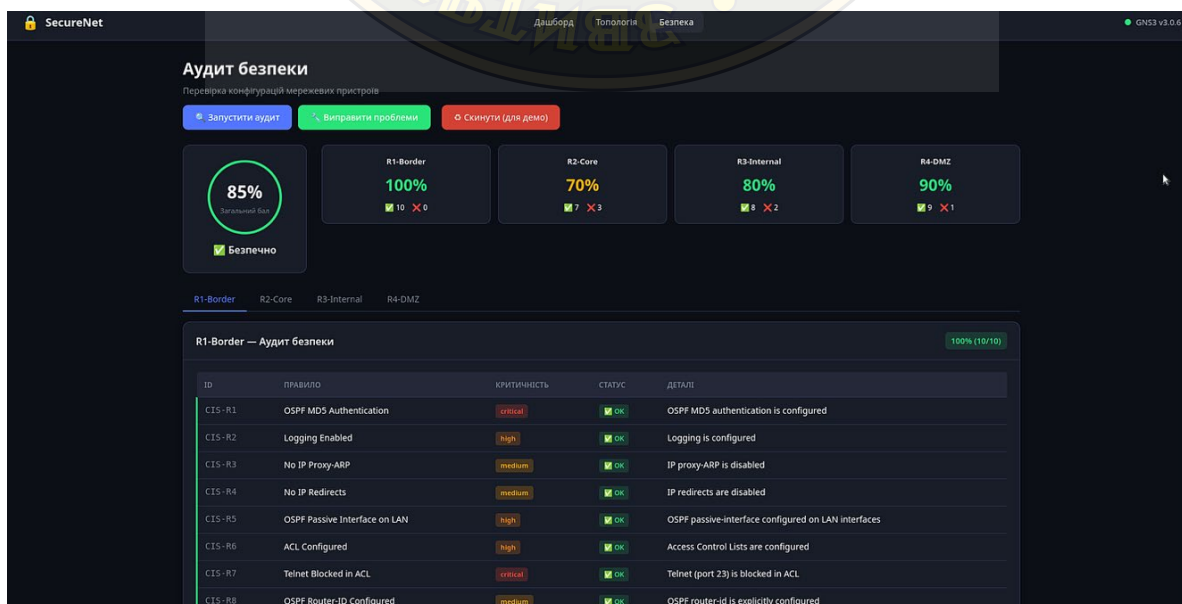


Рисунок 8 – Результати аудиту безпеки до застосування виправлень (85%)

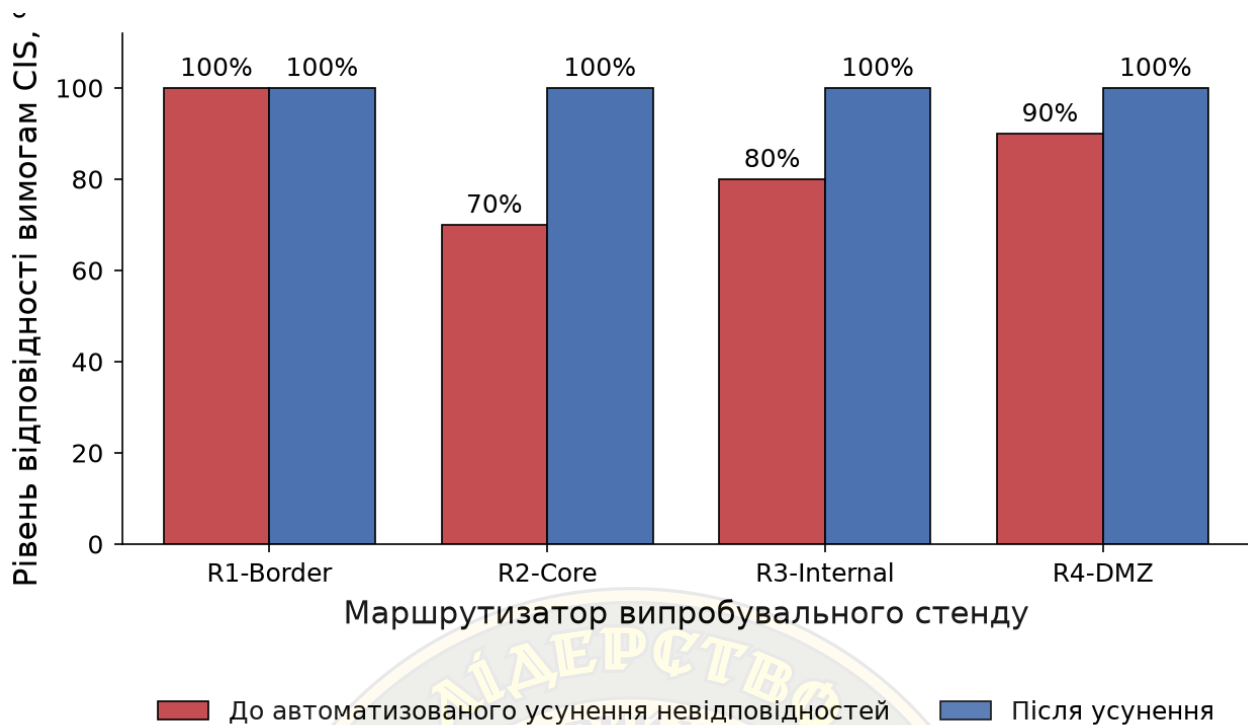


Рисунок 9 – Рівень відповідності конфігурацій вимогам CIS Benchmark до та після автоматизованого усунення невідповідностей

Генерація та впровадження розширених списків доступу. Модуль управління ACL реалізує повний цикл від визначення політики безпеки до її впровадження на мережевому обладнанні. Логіка генерації ACL побудована на основі попередньо визначеної матриці доступу між зонами.

Для зони Restricted, підключеної до маршрутизатора R3 з підмережею 10.0.30.0/24, визначено такі правила доступу до серверів у зоні DMZ (10.0.40–42.0/24):

Дозволити доступ до SRV-WEB (10.0.40.10) по протоколах HTTP (TCP 80) та HTTPS (TCP 443) – для забезпечення доступу до корпоративного вебпорталу;

Заблокувати будь-який доступ до SRV-DB (10.0.42.10) – сервер бази даних не повинен бути доступний безпосередньо для кінцевих користувачів;

Заблокувати будь-який доступ до SRV-REDIS (10.0.41.10) – сервер кешування є виключно внутрішнім компонентом;

Заблокувати весь інший трафік із зони Restricted до зони DMZ згідно принципу "deny all" за замовчуванням.

Система формує відповідний набір команд Cisco IOS та передає їх на маршрутизатор через метод `send_config_set()` бібліотеки Netmiko. Підтвердженням коректної роботи є результати тестування: ping від хоста "Зловмисник" (10.0.30.99) до SRV-DB (10.0.42.10) повертає "Destination Host Unreachable", тоді як легітимний хост із зони Access успішно відкриває з'єднання з SRV-WEB.

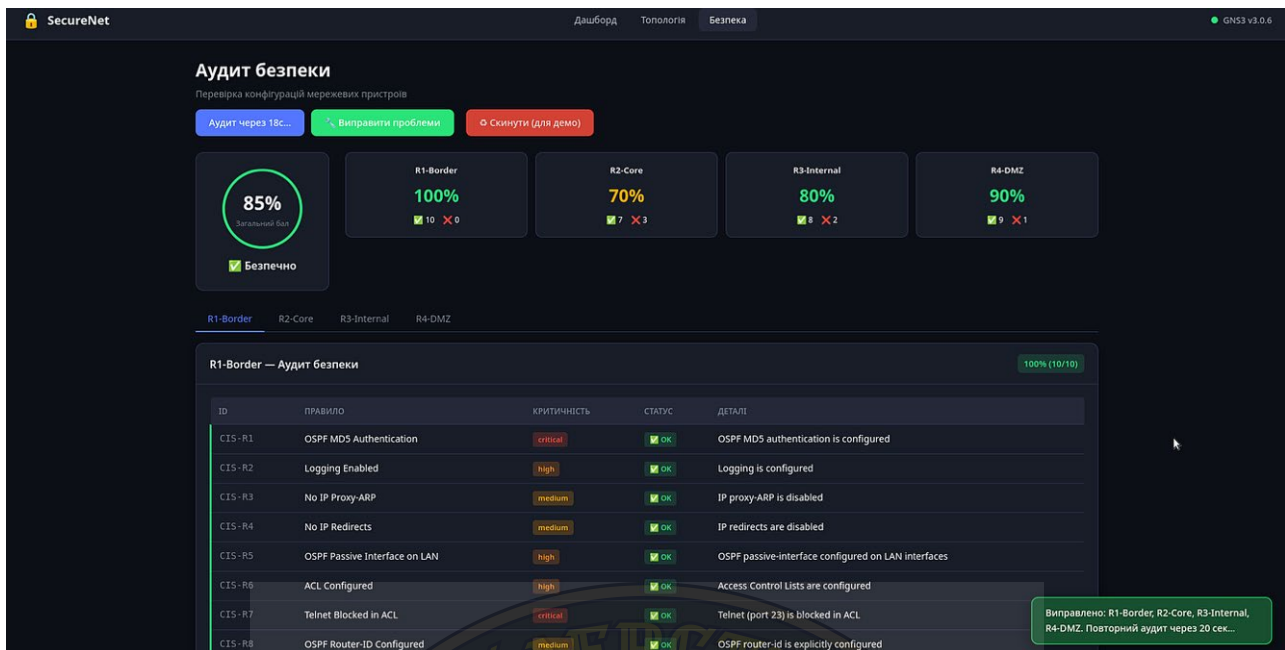


Рисунок 10 – Сповіщення про успішне застосування виправлень конфігурацій

На рисунку 10 зображено інтерфейс після натискання кнопки "Виправити проблеми". Toast-повідомлення підтверджує: "Виправлено: R2-Core, R3-Internal, R4-DMZ". Система автоматично завантажила захищені FRR-конфігурації через GNS3 API (POST /projects/{id}/nodes/{id}/files/etc/frr/frr.conf), перезапустила вузли для застосування змін та розпочала 20-секундний зворотний відлік перед автоматичним повторним запуском аудиту безпеки.

Розгортання нової конфігурації на наявній топології. Для апробації системи в умовах, максимально наближених до реального функціонування, у віртуальному середовищі GNS3 було розгорнуто детально пропрацьований віртуальний стенд. Його основу складають чотири маршрутизатори, що працюють на базі FRR (Free Range Routing): R1 виконує роль ядрового маршрутизатора (Core/Distribution) у магістральній зоні OSPF Area 0; R2 призначений для зони Access; R3 обслуговує зону Restricted; R4 функціонує як маршрутизатор зони DMZ. Хости PC1 та PC2 симулюють активність легітимних користувачів у зоні Access (підмережі 10.0.20.0/24 та 10.0.21.0/24), тоді як PC3 виступає в ролі потенційного зловмисника в зоні Restricted (10.0.30.0/24). У зоні DMZ було імплементовано три сервери: SRV-WEB (працює на базі PHP/Apache), SRV-REDIS та SRV-DB (з PostgreSQL 14).

Маршрутизація між цими зонами забезпечується за допомогою протоколу OSPF. R1 виступає центральним вузлом, який пов'язує R2, R3 та R4 через ключову зону Area 0. Після запуску емуляції первинним етапом є перевірка базової мережевої доступності між усіма елементами стенда шляхом надсилення ICMP-запитів. Позитивні відповіді на ці запити є підтвердженням коректності налаштувань OSPF, створюючи необхідну передумову для подальших етапів тестування.

Саме тестування проходило у три послідовні фази. На першому етапі на маршрутизаторі R1 навмисно залишено активний Telnet для імітації реальної вразливості. Після підключення системи та збору running-config модуль валідації виявляє цей факт і виводить попередження: "Warning: Telnet detected on device R1! CIS Benchmark violation".

На другому етапі система автоматично генерує та застосовує ACL на R3: забороняє трафік від PC3 до SRV-DB та SRV-REDIS, дозволяючи лише HTTP/HTTPS до SRV-WEB. Ping від PC3 до SRV-DB повертає "U.U.U" (Destination Unreachable), тоді як ping від PC1/PC2 до SRV-WEB завершується успішно. Пунктирна лінія на рисунку 6 наочно показує заблокований напрямок трафіку.

На третьому етапі система збирає LLDP-дані з усіх чотирьох маршрутизаторів та генерує HTML-файл з інтерактивним графом. Граф відображає ієрархічну структуру мережі: R1 у верхньому рівні, R2–R4 на рівні розподілу, хости та сервери в нижньому рівні, організовані в три чітко виділені кольором зони безпеки. Весь цикл займає менше трьох хвилин без будь-яких переривань у роботі мережі.

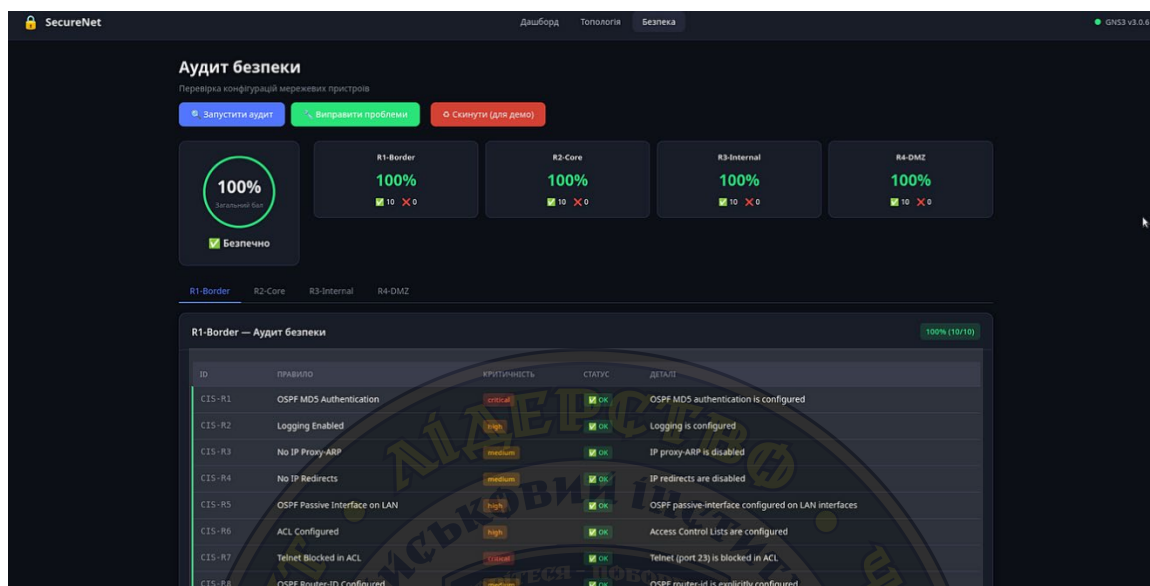


Рисунок 11 – Результати аудиту безпеки після виправлення (100%)

На рисунку 11 зображено результати аудиту безпеки після успішного застосування виправлених конфігурацій. Усі чотири маршрутизатори досягли максимального результату 100% (правила 10/10): R1-Border, R2-Core, R3-Internal, R4-DMZ. Загальний системний бал підвищився з 85% до 100%, а статус змінився з "Вразливий" на "Безпечний". Це підтверджує коректну роботу модуля автоматичного виправлення конфігурацій.

Висновки. Обговорення отриманих результатів доцільно проводити, спираючись на вищезгаданий огляд літератури. Отримані дані узгоджуються з висновками попередніх досліджень, які неодноразово підкреслювали, наскільки ефективно автоматизація конфігурацій здатна мінімізувати помилки адміністрування [5]. Саме тому, на наш погляд, цю статтю слід сприймати як доповнення до вже наявних знань. Вона свідчить, що поєднання аудиту за CIS Benchmark з автоматичною генерацією ACL може суттєво підвищити рівень безпеки: з 85% до 100% відповідно до прийнятої метрики відповідності. На нашу думку, це є вагомим аргументом на користь інтегрування етапу верифікації безпосередньо у цикл розгортання, що відповідає головній тезі дослідження мереж на основі намірів про пріоритет верифікації над простим застосуванням політики [3].

Практична реалізація чотиризонної моделі сегментації та підтверджене блокування несанкціонованого доступу між зонами Restricted і DMZ яскраво демонструють дієвість принципу найменших привілеїв, який у літературі розглядається як фундаментальна основа архітектури Zero Trust [8, 9]. Проте ми маємо одразу ж вказати на певні обмеження нашого дослідження: апробація відбувалася на віртуальному стенді GNS3 з вельми обмеженою кількістю вузлів, а контрольний список валідації охоплював лише 10 з понад 80 рекомендацій CIS Benchmark. Отже, отримані кількісні показники варто розглядати виключно як наочну демонстрацію ефективності запропонованого підходу, а не як універсальну та абсолютно точну оцінку безпеки.

Зважаючи на все вищевикладене, можна сформулювати низку ключових висновків.

Результатом нашого дослідження є розробка автоматизованої системи, призначеної для ретельної перевірки конфігурацій мережевих пристроїв. Таким чином, базуючись на отриманих результатах, ми можемо зробити кілька важливих тверджень.

У процесі вивчення предметної області проаналізовано механізми фільтрації мережевого трафіку та механізм розширених ACL на платформі Cisco IOS, а також охоплено сучасні технологічні стеки, призначені для автоматизації управління мережевим обладнанням, такі як Ansible, NETCONF, Netmiko/Python. Детально розглянуто всі вимоги стандарту CIS Benchmark для Cisco IOS. Вибір протоколу LLDP (IEEE 802.1AB) для збору даних про фізичну топологію мережі обґрунтовано його повною незалежністю від стану маршрутизації та відмінною сумісністю з обладнанням різних виробників.

Практичну реалізацію цієї системи виконано мовою програмування Python з використанням бібліотек Netmiko та PyVis, що дало змогу повністю автоматизувати збір топологічних даних через LLDP, їх подальше перетворення на структурований граф та, зрештою, генерацію інтерактивних HTML-візуалізацій мережі. Модуль аудиту безпеки продемонстрував вражаючу ефективність, успішно виявивши численні вразливі конфігурації, зокрема, активний протокол Telnet, що є прямим та очевидним порушенням стандарту CIS Benchmark.

Розроблена система пропонує три основні покращення, що значно виділяють її на тлі традиційного ручного адміністрування. По-перше, вона забезпечує скорочення часу, що необхідний для проведення аудиту безпеки. По-друге, повністю усувається суб'єктивний людський чинник у процесі перевірки конфігурацій, що гарантує об'єктивність. І, по-третє, система автоматизує застосування захисних політик, що забезпечує їхнє послідовне, безперервне та належне впровадження.

Подальший розвиток системи передбачає розширення контрольного списку валідації для охоплення абсолютно всіх вимог CIS Benchmark, інтеграцію з провідними системами моніторингу для миттєвого реагування на будь-які зміни конфігурацій у режимі реального часу, а також розробку інтелектуального механізму автоматичного виправлення виявлених вразливостей. Отримані результати доводять, що принципи NetDevOps у сфері мережевої безпеки залишаються актуальними та відкривають широкі перспективи.

ЛІТЕРАТУРА

1. Verizon. Data Breach Investigations Report. 2025. URL: <https://www.verizon.com/business/resources/reports/dbir/> (дата звернення: 10.03.2026).
2. CIS Cisco IOS Benchmark. Center for Internet Security. URL: <https://www.cisecurity.org/benchmark/cisco> (дата звернення: 10.03.2026).
3. Leivadeas A., Falkner M. A survey on intent-based networking. IEEE Communications Surveys & Tutorials. 2023. Vol. 25, No. 1. P. 625–655. DOI: 10.1109/COMST.2022.3215919.
4. Gharbaoui M., Sciarrone F., Fontana M., Castoldi P., Martini B. Assurance and conflict detection in intent-based networking: A comprehensive survey and insights on standards and open-source tools. IEEE Transactions on Network and Service Management. 2026. Vol. 23. P. 1891–1912. DOI: 10.1109/TNSM.2026.3651896.
5. Bajpai M. Automating network device configuration and compliance enforcement. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences. 2025. Vol. 7, No. 3. P. 1–6. DOI: 10.2139/ssrn.5057509.
6. Thati S. R. AI-driven automation for network configuration and compliance: Transforming enterprise security posture. World Journal of Advanced Research and Reviews. 2025. Vol. 26, No. 2. P. 1216–1223. DOI: 10.30574/wjarr.2025.26.2.1693.
7. Ficzero D., Varga P. AI-driven network configuration and operation. NOMS 2025 IEEE Network Operations and Management Symposium. IEEE, 2025. P. 1–4. DOI: 10.1109/NOMS57970.2025.11073726.
8. Mushtaq S., Mohsin M., Mushtaq M. A systematic literature review on the implementation and challenges of zero trust architecture across domains. Sensors. 2025. Vol. 25, No. 19. Art. 6118. DOI: 10.3390/s25196118.
9. Basta N., Ikram M., Kaafar M. A., Walker A. Towards a zero-trust micro-segmentation network security strategy: An evaluation framework. NOMS 2022 IEEE/IFIP Network Operations and Management Symposium. IEEE. 2022. P. 1–7. DOI: 10.1109/NOMS54207.2022.9789888.

10. Rescorla E., Ylonen T. RFC 4253: The Secure Shell (SSH) transport layer protocol. IETF, 2006. URL: <https://www.rfc-editor.org/rfc/rfc4253> (дата звернення: 10.03.2026).
11. PyVis documentation. PyVis. URL: <https://pyvis.readthedocs.io/en/latest/> (дата звернення: 10.03.2026).
12. GNS3 documentation. GNS3. URL: <https://docs.gns3.com/> (дата звернення: 10.03.2026).

REFERENCES

1. Verizon (2025) Data Breach Investigations Report. Available at: <https://www.verizon.com/business/resources/reports/dbir/>
2. Center for Internet Security (no date) CIS Cisco IOS Benchmark. Available at: <https://www.cisecurity.org/benchmark/cisco>
3. Leivadeas, A. and Falkner, M. (2023) 'A survey on intent-based networking', IEEE Communications Surveys & Tutorials, 25(1), pp. 625–655. <https://doi.org/10.1109/COMST.2022.3215919>
4. Gharbaoui, M., Sciarrone, F., Fontana, M., Castoldi, P. and Martini, B. (2026) 'Assurance and conflict detection in intent-based networking: a comprehensive survey and insights on standards and open-source tools', IEEE Transactions on Network and Service Management, 23, pp. 1891–1912. <https://doi.org/10.1109/TNSM.2026.3651896>
5. Bajpai, M. (2025) 'Automating network device configuration and compliance enforcement', SSRN Electronic Journal. <https://doi.org/10.2139/ssrn.5057509>
6. Thati, S.R. (2025) 'AI-driven automation for network configuration and compliance: transforming enterprise security posture', World Journal of Advanced Research and Reviews, 26(2), pp. 1216–1223. <https://doi.org/10.30574/wjarr.2025.26.2.1693>
7. Ficzer, D. and Varga, P. (2025) 'AI-driven network configuration and operation', in NOMS 2025 IEEE Network Operations and Management Symposium. IEEE, pp. 1–4. <https://doi.org/10.1109/NOMS57970.2025.11073726>
8. Mushtaq, S., Mohsin, M. and Mushtaq, M. (2025) 'A systematic literature review on the implementation and challenges of zero trust architecture across domains', Sensors, 25(19), 6118. <https://doi.org/10.3390/s25196118>
9. Basta, N., Ikram, M., Kaafar, M.A. and Walker, A. (2022) 'Towards a zero-trust micro-segmentation network security strategy: an evaluation framework', in NOMS 2022 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 1–7. <https://doi.org/10.1109/NOMS54207.2022.9789888>
10. Rescorla, E. and Ylonen, T. (2006) RFC 4253: The Secure Shell (SSH) Transport Layer Protocol. IETF. Available at: <https://www.rfc-editor.org/rfc/rfc4253>
11. PyVis (no date) PyVis documentation. Available at: <https://pyvis.readthedocs.io/en/latest/>
12. GNS3 (no date) GNS3 documentation. Available at: <https://docs.gns3.com/>

PhD Sashnova M., Vozniuk Y., PhD Cherkasov D., Maslov P.

AUTOMATED COMPLIANCE CONTROL OF NETWORK DEVICE CONFIGURATIONS WITH CYBERSECURITY REQUIREMENTS

The growing number of connected devices, the increasing diversity and complexity of network topologies, and the constant evolution of cyber threats greatly complicate the administration of network equipment. Configuring dozens or hundreds of routers and switches manually is labor-intensive, costly, and associated with a high risk of cyber threats. Configuration errors themselves are the reason for the success of most cyberattacks. The aim of the study is to develop and practically test a system for the automated validation of network device configurations. Specifically, the system should automatically identify the physical network topology, verify the compliance of settings with industry security standards, and automatically remediate detected vulnerabilities.

The system was implemented in Python using the Netmiko and PyVis libraries. The LLDP protocol (IEEE 802.1AB) was used to obtain topology information. Configuration checks were performed in accordance with 10 CIS Benchmark rules for Cisco IOS. The system was tested on a virtual GNS3 testbed consisting of Cisco IOS routers classified into security zones (Access, Restricted, DMZ) with dynamic OSPF routing.

The audit module detected a number of vulnerable settings (in particular, enabled Telnet), as a result of which the initial security-standard compliance score was 85%. After the automated application of fixes and the generation of extended access control lists (ACLs), the score was raised to 100%. In addition, the system successfully blocked an unauthorized access attempt from the Restricted zone to the database server, which confirmed the correctness of the generated filtering policies.

The proposed solution significantly reduces the time required to conduct an audit, eliminates the influence of the subjective human factor, and ensures the consistent implementation of security measures. The results of the study confirm that NetDevOps principles work quite effectively in solving the tasks of ensuring the protection of real networks.

Keywords: administration automation; security audit; access control lists; CIS Benchmark; NetDevOps; network segmentation; Zero Trust; network devices.

