Метадані

ДОКУМЕНТ

Заголовок

2026_Грицак_О.О._ФІТ_ITTC_ICT-22-1

Автор	Науковий керівник / Експерт	ІД документу
Грицак О.О.	**Заміховська О.Л.**	**334279845**

ОРГАНІЗАЦІЯ

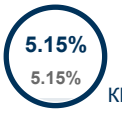
Назва організації	підрозділ
Ivano-Frankivsk National Technical University of Oil and Gas	**Каф. ITTC**

ЗВІТ

Дата звіту	Дата редагування
6/11/2026	**---**

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

15541

Кількість слів

КЦ

120860

Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	3
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	31

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз	Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2026_Горбань_Ю.О._ФІТ_ITTC_ICT-22-1 [redacted] 6/10/2026	70 (0.45 %)

	Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	
2	https://www.sysnettechsolutions.com/en/configure-static-nat-in-cisco-packet-tracer/	55 (0.35 %)
3	2025_Гринів С. М._ФІТ_ІТТС_ІСТ-21-1 [redacted] 6/17/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	45 (0.29 %)
4	2026_Горбань_Ю.О._ФІТ_ІТТС_ІСТ-22-1 [redacted] 6/10/2026 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	29 (0.19 %)
5	https://www.sysnettechsolutions.com/en/configure-telnet-in-cisco-packet-tracer/	27 (0.17 %)
6	https://arxiv.org/pdf/2602.08023	24 (0.15 %)
7	2026_Федорко Є.М._ФІТ_ІТТС_СІ-22-1 [redacted] 6/10/2026 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	21 (0.14 %)
8	report.docx [redacted] 9/17/2024 Astana IT University (Astana IT University)	21 (0.14 %)
9	2025_Гринів С. М._ФІТ_ІТТС_ІСТ-21-1 [redacted] 6/17/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	20 (0.13 %)
10	Проєкт файлового серверу корпоративної мережі підприємства [redacted] 6/9/2026 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ITC, K-pa телекомунікацій)	18 (0.12 %)

Домашня база даних (1.98 %)

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2025_Гринів С. М._ФІТ_ІТТС_ІСТ-21-1 [redacted] 6/17/2025 **Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)**	**128 (7) (0.82 %)**
2	2026_Горбань_Ю.О._ФІТ_ІТТС_ІСТ-22-1 [redacted] 6/10/2026 **Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)**	**113 (4) (0.73 %)**
3	2026_Федорко Є.М._ФІТ_ІТТС_СІ-22-1 [redacted] 6/10/2026 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	28 (2) (0.18 %)
4	2025_Когуч М. С._ФІТ_ІТТС_ІСТ-21-1 [redacted] 6/17/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	27 (3) (0.17 %)
5	2025_Пронь В.В._ФІТ_ІТТС_СІ-23-1К [redacted] 6/17/2025 Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС)	11 (1) (0.07 %)

Програма обміну базами даних (0.93 %)

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
6	report.docx [redacted] 9/17/2024 Astana IT University (Astana IT University)	21 (1) (0.14 %)

7	Проєкт файлового серверу корпоративної мережі підприємства [redacted] 6/9/2026 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ITC, К-ра телекомунікацій)	18 (1) (0.12 %)
8	«Інтерактивний довідково-аналітичний каталог агронома» . Дипломна робота. [redacted] 5/18/2026 Zhytomyr Agricultural Technical Professional College (Zhytomyr Agricultural Technical Professional College)	18 (2) (0.12 %)
9	Розроблення вебзастосунку з управління повним циклом внутрішніх поштових відправлень [redacted] 5/24/2026 Kharkiv National University of Economics named after S.Kuznets (KNUE) (KNUE)	17 (1) (0.11 %)
10	Система автоматизації роботи рієлторської агенції. Курсова робота. [redacted] 12/9/2025 Zhytomyr Agricultural Technical Professional College (Zhytomyr Agricultural Technical Professional College)	15 (1) (0.1 %)
11	РОЗРОБКА ТА ВПРОВАДЖЕННЯ МЕТОДИКИ ВИКОРИСТАННЯ АВТОМАТИЗОВАНИХ СИСТЕМ ПЕРЕВІРКИ ЗАВДАНЬ З ІНФОРМАЦІЙНОЇ БЕЗПЕКИ У НАВЧАЛЬНОМУ ПРОЦЕСІ ЗАКЛАДІВ ФАХОВОЇ ПЕРЕДВИЩОЇ ОСВІТИ [redacted] 11/27/2025 Rivne State Humanities University (Рівненський державний гуманітарний університет)	13 (2) (0.08 %)
12	БКР Світлак Б. КН-420 [redacted] 6/8/2025 Educational and Research Institute of Spatial Planning and Advanced Technologies of Lviv Polytechnic National University (Educational and Research Institute of Spatial Planning and Advanced Technologies of Lviv Polytechnic National University)	12 (1) (0.08 %)
13	«Інформаційна система для управління роботою стоматологічної клініки». Дипломна робота. [redacted] 5/14/2026 Zhytomyr Agricultural Technical Professional College (Zhytomyr Agricultural Technical Professional College)	10 (1) (0.06 %)
14	Розробка веб-застосунку для управління особистими фінансами [redacted] 3/12/2026 Donbass Institute of Technology and Management (Donbass Institute of Technology and Management)	8 (1) (0.05 %)
15	ФКНТ_2026Б_123-КС_Дворовий Д.А. [redacted] 6/3/2026 Ukrainian national aviation university (ФКНТ Кафедра комп'ютерних систем та мереж)	7 (1) (0.05 %)
16	Кваліфікаційна робота Мішкін Д. В. [redacted] 12/20/2025 Ukrainian national aviation university (ФКНТ Кафедра технічного захисту інформації)	6 (1) (0.04 %)

Інтернет (2.25 %)

#	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
17	**https://www.sysnettechsolutions.com/en/configure-static-nat-in-cisco-packet-tracer/**	**55 (1) (0.35 %)**
18	**https://www.sysnettechsolutions.com/en/configure-telnet-in-cisco-packet-tracer/**	**27 (1) (0.17 %)**
19	https://economyandsociety.in.ua/index.php/journal/article/view/6406	25 (2) (0.16 %)
20	https://arxiv.org/pdf/2602.08023	24 (1) (0.15 %)
21	https://community.cisco.com/t5/routing/4g-lte-configuration-in-new-cisco-c1111-8plteea/td-p/3403837	21 (2) (0.14 %)
22	http://vkt.gi.edu.ua/en/college-main/subsection/library/repozytorii/file/download/db29a909aeaf7a1f0e4b08b159895fe6	21 (3) (0.14 %)

23	https://elartu.tntu.edu.ua/bitstream/lib/41909/1/%D0%9A%D0%A0%D0%91_%D0%9F%D0%BE%D0%BF%D0%B5%D1%81%D0%BA%D1%83.pdf	20 (2) (0.13 %)
24	https://ela.kpi.ua/bitstreams/3c116c26-9326-415a-ad6b-eee6f78decf4/download	19 (2) (0.12 %)
25	https://elartu.tntu.edu.ua/bitstream/lib/43367/1/%D0%9A%D0%A0%20%D0%BC%D0%B0%D0%B3%D1%96%D1%81%D1%82%D1%80%D0%B0_%D0%AF%D1%80%D0%B0%D0%BC%D0%B0_2023.pdf	18 (2) (0.12 %)
26	https://essuir.sumdu.edu.ua/bitstreams/0b3cd6dd-aa85-4e27-9258-e2831765beed/download	17 (1) (0.11 %)
27	https://community.cisco.com/t5/switching/i-can-t-connect-to-server/td-p/4000927	15 (2) (0.1 %)
28	http://moodle2.snu.edu.ua/mod/resource/view.php?id=52017	13 (1) (0.08 %)
29	https://openarchive.nure.ua/bitstreams/337d7970-6581-41fd-8c6a-e17b3c3471f5/download	10 (1) (0.06 %)
30	http://dspace.wunu.edu.ua/bitstream/316497/50364/1/%D0%9B%D1%96%D0%BF%D0%BE%D0%B2.pdf	10 (1) (0.06 %)
31	https://people.cs.clemson.edu/~westall/851/docsis/cmts_feature_guide.pdf	9 (1) (0.06 %)
32	https://essuir.sumdu.edu.ua/bitstream/123456789/83894/1/Nikolaiev_bac_rob.pdf	8 (1) (0.05 %)
33	http://ir.polissiauniver.edu.ua/bitstream/123456789/17076/1/Zaverukha_MM_KR_122_2025.pdf	8 (1) (0.05 %)
34	https://community.cisco.com/t5/switching/destination-host-unreachable/td-p/4005083	7 (1) (0.05 %)
35	http://repository.dnu.dp.ua:1100/upload/23e81591ae1e1ef632a78cc3d2215e5bDR_BAKALAVR.PDF	7 (1) (0.05 %)
36	https://metod.vntu.edu.ua/getfile.php/1941.pdf	5 (1) (0.03 %)
37	https://ela.kpi.ua/bitstream/123456789/36510/1/Lebedyk_bakalavr.pdf	5 (1) (0.03 %)
38	http://netkiller.github.io/cisco/switch.example.html	5 (1) (0.03 %)

Список прийнятих фрагментів

#	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

2 Міністерство освіти і науки України Івано-Франківський національний технічний університет нафти і газу Інститут інформаційних технологій
Кафедра інформаційно-телекомунікаційних технологій та систем
Грицак Олег Олександрович (прізвище, ім'я, по батькові) УДК: 004.7:004.056.5:004.422
2 (індекс)

БАКАЛАВРСЬКА РОБОТА

«Розроблення інформаційної системи інвентаризації та аналізу конфігурації мережевих пристроїв з оцінкою ризиків безпеки»
1 (назва роботи)

Інформаційні системи та технології
(назва освітньої програми)

126 – Інформаційні системи та технолгії
(шифр і назва спеціальності)

26 Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Грицай О. О.
2

(підпис, ініціали та прізвище здобувача)

Науковий керівник к.т.н., доцент Заміховська О. Л.
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника) Допущено до захисту
Завідувач кафедри

В.о. зав. каф., доц.__ ___Заміховська О. Л. (посада) (підпис) (дата) (ініціали та прізвище) Івано-Франківськ – 2026 Івано-Франківський національний технічний університет нафти і газу
_____ (повне найменування закладу вищої освіти)
Інститут інформаційних технологій Кафедра інформаційно-телекомунікаційних технологій та систем Освітній рівень бакалавр Спеціальність 126 − Інформаційні системи та технології (шифр і назва) ЗАТВЕРДЖУЮ В.о.завідувача кафедри ІТТС _____Заміховська О. Л. «____» _____2026 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ
Грицаку Олегу Олександровичу
(прізвище, ім'я, по батькові) 1. Тема роботи: «Розробка інформаційної системи інвентаризації та аналізу конфігурації мережевих пристроїв з оцінкою ризиків безпеки»
Керівник роботи __ Заміховська О. Л., к.т.н., доц. каф. ІТТС _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «07» квітня 2026 року No No 163/ 7
2. Строк подання студентом роботи 13 червня 2026 р.
3. Вихідні дані до роботи: Інформаційна система реалізована мовою Python 3.11 з використанням вебфреймворку Flask 3.0, БД SQLite (ORM SQLAlchemy), оркестратора збору даних Node-RED та сканера мережі Nmap. Веб-інтерфейс побудовано на основі Bootstrap 5 та Chart.js. Імітаційне моделювання мережевої інфраструктури виконано в середовищі Cisco Packet Tracer. Система підтримує аудит за 8 правилами безпеки на основі CVSS v3.1, експорт звітів у форматах CSV та PDF, масштабування до 500 мережевих вузлів..
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): 4.1. Опис предметної області процесу. Визначення вимог до інформаційної системи. 4.2 Проєктування архітектури ІС. 4.3 Розробка концептуальної моделі даних
даних і алгоритм оцінки ризиків безпеки на основі CVSS v3.1. 4.4 Реалізація програмної системи з інтеграцією Flask-бекенду, Node-RED оркестратора та веб-інтерфейсу. 4.5 Висновки. 4.6 Бібліографія 4.7 Додатки.
Графічний матеріал: БР.ІСТ – 07.00.00.000Е1 – Контекстна діаграма А-0 (IDEF0). БР.ІСТ – 07.00.00.000Е2 − Декомпозиція А0 (рівень А1–А4). БР.ІСТ – 07.00.00.000Е3 − DFD Рівень 1 − основні процеси обробки даних. БР.ІСТ – 07.00.00.000Е4 – Діаграма варіантів використання системи інвентаризаці. БР.ІСТ – 07.00.00.000Е5 – Діаграма послідовності сценарію ручного сканування мережі. БР.ІСТ – 07.00.00.000Е6 – Діаграма діяльності алгоритму оцінки ризиків безпеки. БР.ІСТ – 07.00.00.000Е7 – Графічна схема потоку збору даних. БР.ІСТ – 07.00.00.000Е8 – Головна сторінка інформ. панелі вебінтерфейсу системи. БР.ІСТ – 07.00.00.000Е9 – Схема імітаційної моделі мережевої інфраструктури в Cisco Packet Tracer. БР.ІСТ – 07 .00.00.000Е10 – Результати верифікації мережевої доступності хостів (команда ping) в імітаційній моделі.
6. Дата видачі завдання 11 березня 2026

КАЛЕНДАРНИЙ ПЛАН No з/п Назва етапів бакалаврської роботи Термін виконання етапів роботи Примітка 1 Збір матеріалів за темою роботи до 12.04.2026 викон. 2 Підготовка розділу Вступ до 21.04.2026 викон. 3 Визначення основних вимог до системи, побудова діаграм для зображення функціональних можливостей системи до 14.05.2026 викон. 4 Вибір програмного середовища розробки до 20.05.2026 викон. 5 Проєктування архітектури та бази даних системи до 27.05.2026 викон. 6 Оформлення пояснювальної записки до 02.06.2026 викон. 7 Доопрацювання роботи з урахуванням рекомендацій керівника до 10.06.2026 викон.
Студент _ _____ Олег ГРИЦАК
(підпис) (прізвище та ініціали) Керівник роботи __ _____ Олена ЗАМІХОВСЬКА (підпис) (прізвище та ініціали)

РЕФЕРАТ
Бакалаврська робота: 95 сторінок, у тому числі основний текст на 65 сторінках, 25 рисунків, 14 таблиць, 29 найменувань використаних джерел, 3 додатки.
Тема бакалаврської роботи: Розроблення інформаційної системи інвентаризації та аналізу конфігурації мережевих пристроїв з оцінкою ризиків безпеки.
Мета роботи: розробити веб-орієнтовану інформаційну систему інвентаризації та аналізу конфігурації мережевих пристроїв з автоматизованою оцінкою ризиків безпеки, що забезпечить своєчасне виявлення вразливостей, формування рекомендацій щодо їх усунення та візуалізацію результатів

моніторингу в режимі реального часу.
Завдання дослідження:
1. Провести аналіз предметної області та огляд існуючих рішень для інвентаризації мережевих пристроїв.
2. Сформулювати функціональні (FR-01 – FR-08) та нефункціональні вимоги до системи.
3. Обґрунтувати вибір технологічного стеку та архітектурних рішень.
4. Розробити концептуальну модель даних і алгоритм оцінки ризиків на основі стандарту CVSS v3.1.
5. Реалізувати програмну систему з інтеграцією Flask-бекенду, Node-RED оркестратора та веб-інтерфейсу.
6. Провести тестування системи на імітаційній мережі в Cisco Packet Tracer та оцінити її ефективність.
Об'єкт дослідження: процеси інвентаризації, конфігураційного аналізу та оцінки безпеки гетерогенних мережевих пристроїв в інформаційно-телекомунікаційних системах.
Предмет дослідження: методи, засоби та архітектурні рішення для автоматизації збору даних про мережеві пристрої, аналізу їх конфігурацій на

відповідність політикам безпеки та розрахунку рівнів ризиків на основі критеріїв CVSS.
У ході виконання роботи отримано такі результати:
У першому розділі проведено аналіз предметної області, який виокремив п'ять ключових проблем сучасних мереж (фрагментована інвентаризація, суб'єктивна оцінка ризиків тощо). Порівняльний огляд семи аналогів обґрунтував доцільність власної розробки. Сформульовано вимоги (FR-01 – FR-08) та побудовано комплекс функціональних моделей у нотаціях IDEF0, DFD, UML та IDEF3.
У другому розділі спроєктовано трирівневу клієнт-серверну архітектуру (Node-RED, Flask/SQLAlchemy/SQLite, веб-інтерфейс), що забезпечує принцип Separation of Concerns. Розроблено нормалізовану модель даних із п'яти таблиць та алгоритм оцінки ризиків (risk_engine.py) на основі 8 детермінованих правил і стандарту CVSS v3.1 (відповідно до ISO/IEC 27005). Деталізовано специфікацію REST API з 12 ендпоінтами.
У третьому розділі реалізовано програмний прототип із налаштованими потоками даних у Node-RED (ручний збір, планувальник, live-дашборд). Комплексне триетапне тестування на імітаційній мережі в Cisco Packet Tracer підтвердило 100% виконання вимог, високу продуктивність (час обробки пристрою <0,1 с) та успішне виявлення 12 вразливостей різного рівня критичності (3 критичні, 4 високі, 5 середніх).
Практична значущість: створено повноцінний робочий прототип, готовий до впровадження в навчальних лабораторіях, який реалізує замкнутий цикл від сканування до візуалізації загроз.
Ключові слова: інформаційна безпека, інвентаризація мережі, аналіз конфігурацій, оцінка ризиків, cvss v3.1, node-red, flask, web-інтерфейс.

ABSTRACT
Bachelor's thesis: 95 pages, including 65 pages of main text, 25 figures, 14 tables, 29 references, and 3 appendices.
The topic of the bachelor's thesis: Development of an information system for inventory and configuration analysis of network devices with security risk assessment.
Purpose: to develop a web-based information system for inventorying and analyzing network device configurations with automated security risk assessment, ensuring timely vulnerability detection, generation of remediation recommendations, and real-time monitoring visualization.
Research tasks:
1. To conduct a domain analysis and review of existing solutions for network device inventory.
2. To formulate functional (FR-01 – FR-08) and non-functional requirements for the system.
3. To justify the choice of the technology stack and architectural solutions.
4. To develop a conceptual data model and a risk assessment algorithm based on the CVSS v3.1 standard.
5. To implement a software system integrating a Flask backend, a Node-RED orchestrator, and a web interface.
6. To test the system on a simulated network in Cisco Packet Tracer and evaluate its efficiency.

The research object: processes of inventory, configuration analysis, and security assessment of heterogeneous network devices in information and telecommunication systems.
The subject of the study: methods, tools, and architectural solutions for automating the collection of data about network devices, analyzing their configurations for compliance with security policies, and calculating risk levels based on CVSS criteria.
The following results were obtained during the work:
In the first section, a domain analysis was conducted, identifying five key problems of modern networks (fragmented inventory, subjective risk assessment, etc.). A

comparative review of seven analogs justified the feasibility of the custom development. Requirements (FR-01 – FR-08) were formulated, and a complex of functional models was built using IDEF0, DFD, UML, and IDEF3 notations.
In the second section, a three-tier client-server architecture was designed (Node-RED, Flask/SQLAlchemy/SQLite, web interface), ensuring the Separation of Concerns principle. A normalized five-table data model and a risk assessment algorithm (risk_engine.py) based on 8 deterministic rules and the CVSS v3.1 standard (compliant with ISO/IEC 27005) were developed. A REST API specification with 12 endpoints was detailed.
In the third section, a software prototype was implemented with configured Node-RED data flows (manual collection, scheduler, live dashboard). Comprehensive three-stage testing on a simulated network in Cisco Packet Tracer confirmed 100% fulfillment of requirements, high performance (device processing time <0.1 s), and the successful detection of 12 vulnerabilities of varying criticality (3 Critical, 4 High, 5 Medium).
Practical significance: a fully functional working prototype has been created, ready for deployment in educational laboratories and small corporate networks, implementing
a closed loop from scanning to threat visualization.
Keywords: information security, network inventory, configuration analysis, risk assessment, cvss v3.1, node-red, flask, web interface.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ
AJAX
Asynchronous JavaScript and XML − асинхронний обмін даними між браузером і сервером
API
Application Programming Interface − програмний інтерфейс застосунку
BPMN
Business Process Model and Notation − нотація моделювання бізнес-процесів
CDN Content Delivery Network − мережа доставки контенту
CLI Command Line Interface − інтерфейс командного рядка
CSV
Comma-Separated Values − формат зберігання табличних даних у вигляді тексту
CVSS Common Vulnerability Scoring System − загальна система оцінювання вразливостей
CVE Common Vulnerabilities and Exposures − база загальновідомих вразливостей
DCIM
Data Center Infrastructure Management − управління інфраструктурою центру обробки даних
DFD Data Flow Diagram − діаграма потоків даних
HTTPS HyperText Transfer Protocol Secure − захищений протокол передачі гіпертексту
IDEF0
Integration Definition for Function Modeling − методологія структурно-функціонального моделювання
IPAM IP Address Management − управління IP-адресами
IEC
International Electrotechnical Commission − Міжнародна електротехнічна комісія
JSON JavaScript Object Notation − текстовий формат обміну даними
KPI Key Performance Indicators − ключові показники ефективності
NCM
Network Configuration Manager − менеджер конфігурацій мережі

NIST
National Institute of Standards and Technology − Національний інститут стандартів і технологій США
Nmap
Network Mapper − утиліта для дослідження мережі та аудиту безпеки
OID Object Identifier − ідентифікатор об'єкта в протоколі SNMP

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційно-телекомунікаційних систем (ІТС) супроводжується постійним зростанням кількості гетерогенних мережевих пристроїв: маршрутизаторів, комутаторів, точок бездротового доступу, серверів та IoT-вузлів. Динамічні зміни топологій та конфігурацій, що відбуваються в таких середовищах, суттєво ускладнюють забезпечення належного рівня кіберзахисту корпоративної інфраструктури.

Відповідно до сучасних досліджень у сфері інформаційної безпеки, середній час між виявленням вразливості та її промисловою експлуатацією

зловмисниками становить менше 15 днів [1]. За таких умов ручні регламентні перевірки та розрізнені інструменти моніторингу вже не здатні забезпечити необхідний рівень оперативності реагування. Відсутність єдиного автоматизованого контуру інвентаризації, контролю конфігурацій та об'єктивної оцінки ризиків призводить до накопичення невиявлених вразливостей і підвищення рівня кіберзагроз [2].
Актуальність теми визначається необхідністю розробки комплексного рішення, яке автоматично виявляє мережеві пристрої, аналізує їх конфігурації на відповідність політикам безпеки, розраховує рівень ризику за стандартом CVSS v3.1 [3] та надає адміністраторам мережі зрозумілі рекомендації щодо усунення виявлених вразливостей − усе в рамках єдиного веб-орієнтованого інтерфейсу.
Мета роботи: розробити веб-орієнтовану інформаційну систему інвентаризації та аналізу конфігурації мережевих пристроїв з автоматизованою оцінкою ризиків безпеки, що забезпечить своєчасне виявлення вразливостей, формування рекомендацій щодо їх усунення та візуалізацію результатів моніторингу в режимі реального часу.
Об'єкт дослідження: процеси інвентаризації, конфігураційного аналізу та оцінки безпеки гетерогенних мережевих пристроїв в інформаційно-телекомунікаційних системах.

Предмет дослідження: методи, засоби та архітектурні рішення для автоматизації збору даних про мережеві пристрої, аналізу їх конфігурацій на відповідність політикам безпеки та розрахунку рівнів ризиків на основі критеріїв CVSS.
Для досягнення поставленої мети визначено такі завдання дослідження:
1 провести аналіз предметної області та огляд існуючих рішень для інвентаризації мережевих пристроїв;
2 сформулювати функціональні (FR-01 – FR-08) та нефункціональні вимоги до розроблюваної інформаційної системи;
3 обґрунтувати вибір технологічного стеку та архітектурних рішень;
4 розробити концептуальну модель даних і алгоритм оцінки ризиків безпеки на основі стандарту CVSS v3.1;
5 реалізувати програмну систему з інтеграцією Flask-бекенду, Node-RED оркестратора та веб-інтерфейсу;
6 провести тестування системи на імітаційній мережі в Cisco Packet Tracer та оцінити її ефективність.
При виконанні БР використано такі методи дослідження: структурно-функціональне моделювання (нотація IDEF0), моделювання потоків даних (нотація Гейна-Сарсона), об'єктно-орієнтоване моделювання (UML Use Case, Sequence, Activity Diagrams), порівняльний аналіз програмних аналогів, а також функціональне та інтеграційне тестування розробленої системи.
Наукова новизна роботи полягає у такому:
− запропоновано комбінований підхід до оцінки ризиків, що поєднує формалізовані правила на основі CVSS v3.1 з контекстним аналізом конфігурацій гетерогенних мережевих пристроїв (Cisco IOS, RouterOS, Windows Server, Ubuntu Server);
− розроблено архітектурне рішення з використанням візуального оркестратора Node-RED як окремого рівня збору даних, що забезпечує модульність, слабку зв'язаність компонентів та можливість заміни джерел даних без зміни коду бекенду.

Результати роботи пройшли апробацію у вигляді тестування на імітаційній мережі в середовищі Cisco Packet Tracer, що підтвердило коректність виявлення навмисно внесених вразливостей (Telnet, SNMP v1/v2c, FTP, HTTP без TLS) та відповідність розрахованих CVSS-балів очікуваним значенням.
Структура роботи: кваліфікаційна бакалаврська робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі виконано аналіз предметної області, розроблено паспорт ІС, проведено огляд аналогів та побудовано функціональні моделі

IDEF0 і DFD.
У другому розділі спроєктовано трирівневу архітектуру системи, розроблено нормалізовану модель даних, алгоритм оцінки ризиків та специфікацію REST API.
У третьому розділі описано реалізацію програмного коду, інтеграцію з Node-RED, веб-інтерфейс та результати комплексного тестування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФУНКЦІОНАЛЬНО-ПРОЦЕСНЕ МОДЕЛЮВАННЯ

1.1 Аналіз предметної області та обґрунтування актуальності

Сучасні інформаційно-телекомунікаційні системи (ІТС) характеризуються високою динамікою змін топології, постійним оновленням конфігурацій мережевого обладнання та зростанням кількості гетерогенних пристроїв (роутери, комутатори, точки доступу, сервери, IoT-вузли). На тлі цієї архітектурної складності кіберзагрози набувають все більш агресивного характеру [4]. Експертні дослідження підтверджують, що швидкість появи експлойтів вимагає переходу від реактивних до проактивних стратегій захисту. Автоматизовані та безперервні аудити конфігурацій, інтегровані з метриками стандартизованої оцінки вразливостей (CVSS), стають мінімальним стандартом кіберстійкості [5]. Саме тому розробка ІС, здатної автоматично виявляти мережеві пристрої, аналізувати їх конфігурації на відповідність політикам безпеки та формувати обґрунтовані оцінки ризиків, є не просто інструментом оптимізації адміністрування, а критично необхідним елементом забезпечення безпеки сучасної ІТ-інфраструктури.
Ефективне проєктування та впровадження такої ІС вимагає чіткої ідентифікації кола зацікавлених сторін (стейкхолдерів), чиї функціональні потреби, рівні доступу та сценарії взаємодії безпосередньо визначають архітектурні рішення та критерії успішності впровадження.
В межах предметної області виділено три ключові групи залежно від рівня впливу та природи взаємодії:
1. Користувачі системи.
2. Зовнішні системи та інфраструктура.
3. Розробка та підтримка.

Зацікавлені сторони:
− адміністратор мережі – основний користувач, відповідальний за підтримку працездатності та безпеки інфраструктури;
− інженер з кібербезпеки – зацікавлений у регулярному аудиті відповідності політикам безпеки та CVSS-оцінках;
− керівник ІТ-відділу – потребує аналітичних звітів для прийняття рішень щодо модернізації та бюджетування;
− зовнішні системи/агенти – Node-RED оркестратор, Nmap-сканер, SNMP-агенти пристроїв, що забезпечують автоматизований збір даних.
Карта зацікавлених сторін ІС «Інвентаризації та аналізу конфігурації мережевих пристроїв з оцінкою ризиків безпеки» представлена в Додатку А.
На рис. 1.1 представлено взаємодію суб'єктів (зацікавлених сторін) з ІС, а також їхній вплив на процеси її розроблення та експлуатації.
Рисунок 1.1 − ІС інвентаризації мережевих пристроїв з оцінкою ризиків
Виділення зацікавлених сторін дозволяє обґрунтувати функціональні вимоги (FR-01 − FR-08). Наприклад, наявність Node-RED як стейкхолдера обумовлює необхідність розробки REST API для імпорту даних, а роль інженера з безпеки вимагає інтеграції алгоритмів розрахунку CVSS-скорів та формування текстових рекомендацій.
Незважаючи на потребу в автоматизації кіберзахисту, більшість організацій досі покладаються на ручні регламенти та розрізнені інструменти. Розрив між вимогами сучасного кіберсередовища (реагування ≤15 днів) [17] і реальними можливостями персоналу створює системні операційні бар'єри. Відсутність

єдиного автоматизованого контуру інвентаризації, контролю конфігурацій та об'єктивної оцінки ризиків призводить до втрат часу та підвищення рівня кіберзагроз.
На операційному рівні відсутність автоматизації проявляється у вигляді таких ключових проблем та недоліків:
− фрагментована інвентаризація − дані про пристрої зберігаються в розрізнених таблицях/файлах, що унеможливлює оперативний пошук та аудит;
− суб'єктивна оцінка ризиків − адміністратори часто ігнорують застарілі протоколи (Telnet, SNMPv1, HTTP), що призводить до накопичення критичних вразливостей;
− відсутність історичного контексту − неможливість відстежити зміни конфігурацій у часі та ідентифікувати момент появи вразливості;
− високе навантаження на персонал – ручне сканування підмереж та перевірка відкритих портів за експертними оцінками, вимагає значного робочого часу (до одного робочого дня для інфраструктури середнього масштабу;
− відсутність візуалізації та звітності − складність комунікації стану безпеки з керівництвом без наочних дашбордів та експортованих звітів.
Концепція запропонованої ІС спрямована на усунення зазначених проблем шляхом автоматизації збору конфігурацій, формалізації правил безпеки на основі стандарту CVSS v3.1, інтеграції візуального оркестратора Node-RED та забезпечення інтерактивного веб-інтерфейсу для моніторингу в режимі реального часу

1.2 Паспорт інформаційної системи та огляд аналогів

Паспорт ІС є ключовим нормативним документом, що містить повну технічну, організаційну, правову та експлуатаційну інформацію про систему [6]. Його розробка та ведення виступають важливим елементом забезпечення інформаційної безпеки, а також ефективного управління життєвим циклом ІС. Цей процес регламентується міжнародними вимогами до менеджменту ІБ, зокрема серією стандартів ISO/IEC 27001 [8], адаптованими національними стандартами ДСТУ EN ISO/IEC 27002:2024 [9], а також базовим законодавством

України у сфері кібернетичного захисту, передусім Законом України «Про основні засади забезпечення кібербезпеки України» [10]».
Інформаційні системи, такі як системи інвентаризації та аналізу конфігурації мережевих пристроїв, відіграють центральну роль у забезпеченні функціонування критичної інфраструктури. Їхня архітектура має бути безшовною, забезпечуючи явні зв'язки між елементами архітектурних моделей різних представлень [11]. Управління апаратними вразливостями протягом життєвого циклу мікропроцесорів та комп'ютерів є ще одним важливим аспектом.
Паспорт ІС є основою для систематичного підходу до забезпечення безпеки та ефективності таких систем (табл. 1.1).
Таблиця 1.1 − Паспорт ІС

Параметр	Значення
Назва	Web-орієнтована ІС інвентаризації та аналізу конфігурації мережевих пристроїв з оцінкою ризиків безпеки
Призначення	Автоматизація виявлення активних вузлів мережі, збір конфігурацій, аудит відповідності політикам безпеки, розрахунок рівня ризику та формування рекомендацій
Бізнес-мета	Скоротити час конфігураційного аудиту з 6 год до ≤30 хв, знизити кількість невиявлених критичних вразливостей до 0, забезпечити формування звітів у форматі PDF/CSV за ≤5 хв
Межі системи	Внутрішня підмережа організації/лабораторії; підтримувані протоколи: SNMPv1/v2c/v3, HTTP/HTTPS, Telnet, SSH, FTP; масштаб: до 500 активних вузлів
Функціональні можливості (Features)	1. Автоматичне сканування підмережі та реєстрація пристроїв.

2. Збір та версіонування конфігурацій.
3. Перевірка конфігурацій за 8 правилами безпеки (Telnet, SNMPv1, HTTP без TLS, FTP, відкриті порти управління, слабкі спільноти, застарілі ОС, відсутність автентифікації).
4. Оцінювання вразливостей за системою CVSS та класифікація ризиків.
5. Генерація текстових рекомендацій.
6. Інтерактивний дашборд з графіками та фільтрами.
7. Експорт звітів у CSV/PDF.
8. Інтеграція з Node-RED для оркестрації та live-моніторингу.

1.2.1 Огляд аналогів

У сучасній практиці адміністрування ІС використовується широкий спектр програмних засобів для моніторингу, управління конфігураціями та оцінки вразливостей. Серед найбільш поширених рішень виділяються Zabbix, PRTG Network Monitor, SolarWinds Network Configuration Manager (NCM) та Tenable

Nessus – інструменти, що зарекомендували себе в корпоративному середовищі завдяки функціональній насиченості та надійності [12 − 14]. Однак кожна з цих систем має власну предметну спеціалізацію та архітектурні обмеження, що робить їх застосування оптимальним лише для певних аспектів управління мережевою інфраструктурою. Комплексний порівняльний аналіз цих та деяких інших систем дозволяє оцінити їхні можливості в контексті інвентаризації, аналізу конфігурації та кількісної оцінки ризиків безпеки.

Мета порівняльного аналізу − оцінити можливості зазначених аналогів у контексті трьох ключових завдань розроблюваної інформаційної системи:
1. Автоматизована інвентаризація гетерогенних мережевих пристрої.
2. Аналіз конфігурацій на відповідність політикам безпеки.
3. Кількісна оцінка ризиків із формуванням текстових рекомендацій.

Такий підхід дозволяє об'єктивно визначити нішу для власної розробки та обґрунтувати її архітектурні переваги порівняно з існуючими комерційними та open-source рішеннями.

Zabbix − це open-source система моніторингу, що забезпечує відстеження стану мережевих пристроїв у реальному часі, гнучке налаштування сповіщень та масштабованість [13, 16, 17]. Вона автоматично виявляє пристрої через SNMP, агенти та API, однак не має вбудованого версійного контролю конфігурацій та не виконує кількісної оцінки ризиків безпеки − для цього потрібні зовнішні інтеграції [16, 17, 18]. Переваги: відкритий код, масштабованість. Недоліки: обмежені можливості аналізу конфігурацій та ризиків.

PRTG Network Monitor − комерційний інструмент, орієнтований на візуалізований моніторинг трафіку та продуктивності мережі [8, 9]. Відстежує тисячі сенсорів, але не підтримує управління конфігураціями (версійний контроль, порівняння змін) та не проводить оцінку ризиків безпеки − функції аудиту конфігурацій реалізуються лише через скрипти PowerShell або SSH. Переваги: інтуїтивний інтерфейс, простота розгортання. Недоліки: комерційний, відсутність аналізу конфігурацій та оцінювання за шкалою CVSS.

SolarWinds NCM − спеціалізується на автоматизації збору, версійному контролі та порівнянні конфігурацій мережевих пристроїв [10]. Забезпечує базову оцінку відповідності стандартам безпеки через звітність щодо дотримання вимог, але не виконує глибокого сканування вразливостей та не має моніторингу в реальному часі. Переваги: потужний конфігураційний аналіз. Недоліки: висока вартість комерційної ліцензії, відсутність автоматичного оцінювання за системою CVSS.

Tenable Nessus − спеціалізований сканер вразливостей, який використовує CVSS для кількісної оцінки ризиків безпеки [12, 22]. Виявляє конфігураційні дефекти, але не є системою постійної інвентаризації та не забезпечує управління конфігураціями (версійний контроль, автоматичний відкат) [22]. Переваги: висока точність виявлення, детальна CVSS-звітність. Недоліки: не працює в режимі реального часу, значна ресурсомісткість, комерційна ліцензія.

Додаткові аналоги

Cisco DNA Center − комплексне рішення для управління мережами Cisco з функціоналом Intent-Based Networking, автоматизацією розгортання та вбудованою оцінкою ризиків. Переваги: глибока оптимізація під вендора. Недоліки: критична апаратна залежність (працює виключно з обладнанням Cisco), висока вартість власності.

NetBox − open-source DCIM/IPAM-рішення для детальної інвентаризації мережевих активів (розташування, з'єднання, IP-адреси) [23]. Переваги: чітка структуризація фізичних та логічних активів. Недоліки: повна відсутність динамічного аналізу конфігурацій та оцінки ризиків.

Ansible + AWX/Tower − open-source інструмент автоматизації для управління конфігураціями через сценарії (playbooks). AWX/Tower забезпечує веб-інтерфейс управління. Переваги: гнучкість конфігурування інфраструктури як коду (IaC). Недоліки: брак вбудованих аналітичних модулів візуалізації ризиків або автоматизованого розрахунку CVSS.

Жодна з розглянутих систем не поєднує всі три компоненти в одному нативному рішенні (табл. 1.2):

Таблиця 1.2 – Порівняльний аналіз існуючих рішень для інвентаризації та оцінки ризиків

Критерій порівняння	Розроблювана ІС	Zabbix	PRTG Network Monitor	SolarWinds NCM	Tenable Nessus
Тип ліцензування	Безкоштовна (Open Source)	Безкоштовна (Open Source)	Freemium (до 100 сенсорів)	Комерційна (~$3000+)	Комерційна (~$4000/рік)
Автоматизована інвентаризація	Повна (SNMP/Nmap)	Повна (SNMP/агенти)	Повна (авто-відкриття)	Часткова (тільки конфіги)	Відсутня
Аналіз конфігурацій	8 правил безпеки + версіонування	Через скрипти	Обмежений (PowerShell)	Повний	

(версійний контроль)
Тільки дефекти
Оцінка ризиків (CVSS)
Вбудована (CVSS v3.1)
Відсутня (потрібна інтеграція)
Відсутня
Часткова (compliance)
Повна (професійна)
Візуалізація (Dashboard)
Chart.js + Node-RED
Базова (потрібна Grafana)
Потужна вбудована
Розширена Детальна
Оркестрація процесів
Node-RED (візуальна)
Відсутня Відсутня Часткова Відсутня
Експорт звітів CSV/PDF Тільки CSV PDF/HTML PDF/CSV PDF/CSV/HTML
Масштабованість До 500 вузлів
До 10000+ вузлів
До 1000 сенсорів
До 5000+ пристроїв
До 10000+ хостів
Поріг входу (навчання)
Низький (простий код)
Високий (складна конфігурація)
Низький (інтуїтивний)
Середній Високий
Придатність для навчання
Ідеально (демо-режим)
Складно Добре Дорого Дорого

Аналіз комерційних рішень (SolarWinds NCM, Tenable Nessus) показує, що побудова комплексної системи захисту на їхній основі вимагає значних фінансових витрат на ліцензування та складних процедур інтеграції розрізнених інтерфейсів. Хоча впровадження open-source систем (Zabbix, Grafana) частково вирішує завдання моніторингу та візуалізації, вони все одно потребують розроблення додаткових модулів для глибокого аналізу конфігурацій та обчислення показників безпеки.

Таким чином, жоден із розглянутих аналогів не забезпечує безкоштовного, модульного та уніфікованого рішення з вбудованим розрахунком балів CVSS та інструментами візуальної оркестрації процесів (Node-RED). Це обґрунтовує доцільність розроблення власної інформаційної системи, яка дозволить

мінімізувати операційні витрати, забезпечить гнучкість налаштування правил безпеки та підвищить рівень захищеності корпоративної інфраструктури

1.3 Функціональна модель IDEF0

Мета моделювання − формалізувати структурно-функціональну архітектуру ІС, визначити межі системи, вхідні та вихідні потоки даних, керуючі впливи та ресурсні механізми для подальшого проектування архітектури програмного забезпечення та бази даних.

Точка зору: системний аналітик, що проектує ІС з позиції адміністратора мережі та інженера з інформаційної безпеки.

На рис. 1.2 представлено контекстну діаграму A-0, яка є найвищим рівнем структурно-функціональної моделі ІС. Вона відображає систему у вигляді єдиного функціонального блоку («чорного ящика»), визначаючи її головну мету, межі та характер взаємодії із зовнішнім інформаційним середовищем.

Рисунок 1.2 – Контекстна діаграма A-0

Для деталізації внутрішньої логіки системи виконано декомпозицію контекстного блоку A-0 на діаграму першого рівня A0 (рис. 1.3).

Рисунок 1.3 − Декомпозиція A0 (рівень A1 – A4)

Діаграма декомпозиції A0 містить чотири взаємопов'язані функціональні блоки:

1 A1 (Сканування мережі та виявлення пристроїв) − виявляє активні вузли за допомогою Nmap та Node-RED на основі заданого розкладу й запитів користувача.

2 A2 (Збір та нормалізація конфігурацій) − агрегує та структурує отримані дані відповідно до корпоративних стандартів, формуючи реєстр пристроїв і виділяючи їхні атрибути.

3 A3 (Аналіз конфігурацій та оцінка ризиків) − перевіряє нормалізовані параметри за 8 правилами CVSS і політиками безпеки з використанням Flask та СУБД SQLite.

4 A4 (Візуалізація та звітування) − агрегує результати за допомогою модуля risk_engine.py, генерує звіти CSV/PDF (бібліотеки Chart.js, ReportLab) та надсилає алерти адміністратору.

Таблиця 1.3 − Глосарій стрілок IDEF0

Назва стрілки	Тип	Короткий опис
Мережеві пристрої	Input (I)	Фізичні/віртуальні вузли мережі, що підлягають інвентаризаційному обліку
Запити на сканування	Input (I)	Директиви від адміністратора або системного розкладу на запуск процедури аудиту.
Параметри політик безпеки	Input (I)	Конфігураційні налаштування дозволених протоколів, відкритих портів та версій ПЗ.
Реєстр пристроїв	Output (O)	Структурований масив виявлених мережевих вузлів із відповідними метаданими.
Оцінки ризиків	Output (O)	Розраховані числові значення та категоріальні рівні

критичності за системою CVSS.
Звіти (CSV/PDF) Output (O)
Експортовані документи встановленого формату для проведення внутрішнього аудиту.
Сповіщення адміністратору Output (O)
Екстрені повідомлення (алерти) у разі ідентифікації критичних загроз чи уразливостей.
Політики безпеки Control (C)
Внутрішні регламенти, що визначають правила конфігурування та доступу в організації.
Стандарти CVSS v3.1 Control (C)
Офіційна міжнародна методологія кількісного оцінювання тяжкості уразливостей.
Розклад сканувань Control (C)
Задані часові інтервали для автоматичного та циклічного запуску процесів перевірки.
Корпоративні стандарти Control (C)
Нормативні вимоги до версій операційних систем, протоколів шифрування та автентифікації.
Веб-фреймворк Flask Mechanism (M)
Серверна частина системи, що обробляє HTTP-запити, керує базою даних та бізнес-логікою.
Оркестратор Node-RED Mechanism (M)
Візуальна low-code платформа для оркестрації процесів збору даних, імітації SNMP-опитування та автоматизованої передачі конфігурацій до Flask API
СУБД SQLite Mechanism (M)
Локальне реляційне сховище для збереження даних про пристрої, конфігурації та ризики.
Сканер Nmap Mechanism (M)
Мережева утиліта, що використовується для активного виявлення портів та активних сервісів.
Бібліотека Chart.js Mechanism (M)
Програмний інтерфейс клієнтської частини для динамічної візуалізації метрик безпеки.
Адміністратор системи Mechanism (M)
Профільний фахівець, який ініціює керуючі дії та інтерпретує отримані результати.

Побудована модель IDEF0 розмежувала етапи життєвого циклу обробки даних у системі, визначила ключові модулі, керуючі впливи стандартів безпеки та механізми їх програмної реалізації. Отриманий глосарій потоків формує базис для

детального проектування логічних зв'язків та інформаційного обміну на наступних етапах розроблення інфраструктури.

1.4 Моделювання потоків даних та процесів

1.4.1 Контекстна діаграма та декомпозиція DFD

На рис. 1.4 представлено контекстну діаграму потоків даних (DFD Рівень 0, нотація Гейна-Сарсона), яка визначає межі ІС та логіку її інформаційної взаємодії із зовнішнім середовищем.

Рисунок 1.4 − DFD Рівень 0 (Контекстна діаграма)

За методологією Гейна-Сарсона, архітектура системи на контекстному рівні представлена процесом P0 «Система інвентаризації та оцінки ризиків», який взаємодіє з такими компонентами:

− мережеві пристрої − передають вхідний потік «SNMP / Nmap дані»;

− адміністратор − надсилає «Запит на сканування» й «Параметри аудиту», отримує «Звіти CSV/PDF», «Графіки Дашборду» та алерти «Alert: Критичні ризики»;

− Node-RED Агент − забезпечує передачу пакетів «JSON: Конфігурації пристроїв»;

− сховище даних (D1) − реляційна БД для двонаправленого обміну даними (запис Devices, Risks, Vulns; читання Configs, History).

Для деталізації потоків та виявлення точок трансформації даних виконано декомпозицію процесу P0 на діаграму першого рівня (рис. 1.5).

Рисунок 1.5 − DFD Рівень 1: основні процеси обробки даних

На рис. 1.5 представлено декомпозицію DFD першого рівня, що містить чотири послідовні підпроцеси обробки інформації:

1 P1 (Сканування мережі) − опитує мережеві пристрої через SNMP та Nmap, формуючи потік первинних даних сканування для наступного етапу.

2 P2 (Збирання та нормалізація даних) − структурує отримані конфігурації, передає нормалізовані дані до аналітичного модуля та зберігає їх у таблицях Devices і Configs сховища D1 (SQLite БД).

3 P3 (Аналіз ризиків) − перевіряє параметри за вбудованими правилами безпеки, обчислює бали уразливостей за стандартом CVSS v3.1 та фіксує результати у таблицях Vulns і Risks бази даних.

4 P4 (Візуалізація та формування звітів) − зчитує історичні дані з D1, інтегрує поточні оцінки від P3 та трансформує їх у графіки і звіти для дашборду користувача.

1.4.2 Діаграма варіантів використання UML Use Case

На рис. 1.6 представлено діаграму прецедентів, яка визначає функціональні вимоги до ІС через взаємодію акторів із варіантами використання. У системі виділено три ролі: Адміністратор системи (ініціює аудит та керує налаштуваннями), Node-RED Agent (автоматизує збір телеметрії) та Risk Engine (аналізує конфігурації за правилами й розраховує бали CVSS).

В табл. 1.4, 1.5 представлено декілька сценаріїв для роботи ІС: запуск сканування та оцінка ризиків та експорт звіту про вразливості.

Таблиця 1.4 − Сценарій UC-01: Запуск сканування та оцінка ризиків

Параметр сценарію	Опис та умови виконання
Ідентифікатор та назва	UC-01: Запуск сканування мережі та оцінювання ризиків
Головний актор	Адміністратор системи
Передумова	Адміністратор успішно автентифікований у системі. Мережеві пристрої доступні для опитування та сканування.
Основний потік подій	1 Адміністратор ініціює прецедент «Запустити сканування мережі». 2

Система активує розширення «Налаштувати параметри сканування». 3 Модуль Node-RED Agent виконує збір даних через SNMP або імітаційне джерело даних. 4 Підсистема Risk Engine аналізує конфігурації мережевих пристроїв та виконує розрахунок показників CVSS. 5 Отримані результати зберігаються у сховищі даних D1. 6 Система відображає користувачу поточний стан виконання та результати оцінювання ризиків.

Альтернативний потік

3а. Якщо мережевий пристрій не відповідає на запит, система реєструє помилку в журналі подій та продовжує опитування наступного пристрою зі списку.

Постумова

Сформовано актуальний реєстр мережевих активів. У базі даних збережено результати аналізу конфігурацій та розраховані бали CVSS для виявлених вразливостей.

Хоча основна бізнес-логіка реалізована у сценарії сканування, для виконання вимог до звітності необхідно передбачити окремий потік роботи з візуалізації даних. Деталі цього потоку, що включає формування файлу звіту та його передачу адміністратору, наведено в таблиці 1.5 (сценарій UC-02).

Таблиця 1.5 − Сценарій прецеденту UC-02: Перегляд реєстру та експорт звіту

Параметр сценарію	Опис та умови виконання
Ідентифікатор та назва	UC-02: Перегляд реєстру пристроїв та експорт звітів
Головний актор	Адміністратор системи
Передумова	У сховищі даних D1 наявні результати попередніх сканувань, аналізу конфігурацій та оцінювання ризиків.
Основний потік подій	1. Адміністратор ініціює прецедент «Переглянути реєстр пристроїв». 2. Система зчитує дані зі сховища D1 та відображає їх у веб-інтерфейсі. 3. Адміністратор аналізує інформацію про мережеві пристрої, вразливості та рівні ризику. 4. За потреби користувач активує дію «Експортувати звіт CSV/PDF». 5. Модуль Flask формує звіт у вибраному форматі та ініціює його завантаження на комп'ютер користувача.
Альтернативний потік	4а. У разі помилки створення або запису файлу система повідомляє про відсутність прав доступу чи внутрішній збій та пропонує повторити операцію експорту.
Постумова	Користувач отримав сформований документ у форматі .csv або .pdf, що містить актуальну інформацію про стан мережевих активів, виявлені вразливості та результати оцінювання ризиків інформаційно-телекомунікаційної системи.

1.4.3 Алгоритм функціонування аналітичного модуля

Для практичної реалізації розроблених прецедентів та процесів діаграми потоків даних, аналітичне ядро ІС (модуль оцінювання уразливостей) функціонує за чітко регламентованим алгоритмом. Послідовність виконання технологічних операцій, логіка прийняття рішень та взаємодія з об'єктами сховища даних представлені на рис. 1.6.

Процес аналізу конфігурацій та обчислення показників безпеки складається з таких послідовних етапів:

1. Ініціалізація та парсинг (кроки 1.0 − 1.1): на основі концептуальних сутностей пристроїв та конфігурацій система зчитує поточні параметри мережевого вузла. Далі виконується автоматизований парсинг відкритих мережевих портів та задіяних протоколів прикладного рівня.
2. Перевірка правил безпеки (крок 1.2): за допомогою логічного

розгалуження типу виключного «АБО» (XOR) здійснюється контроль інфраструктури на відповідність чотирьом базовим критеріям захищеності.

Рисунок 1.6 − Логічна схема потоку робіт аналітичного модуля «Аналіз ризиків»

Перевірці підлягають: наявність відкритого порту 23 (протокол Telnet), використання застарілих та вразливих версій SNMP v1/v2, передача HTTP-трафіку без шифрування TLS, а також активність незахищеного сервісу FTP;

3. Обробка виявлених дефектів:

− якщо порушень політик безпеки не виявлено − аналітичний цикл для поточного пристрою завершується;

− якщо зафіксовано хоча б одну невідповідність − активується крок 1.3, який реєструє дефект безпеки у таблиці уразливостей із присвоєнням відповідного базового бала. Логічне об'єднання «І» (AND) на кроці 1.4 забезпечує підсумовування показників за всіма знайденими дефектами налаштувань;

4. Розрахунок метрик та класифікація (кроки 1.5 − 1.6): на основі агрегованих даних у таблиці risk_scores обчислюється інтегральний показник середнього ризику для вузла. Отримане числове значення автоматично класифікується за категоріальними рівнями критичності стандарту: критичний (Critical), високий (High), середній (Medium) або низький (Low).

5. Генерація рекомендацій та фіксація результатів (кроки 1.7 − 1.9): модуль співставляє виявлені дефекти з базою знань та формує для адміністратора текстові інструкції щодо усунення вразливостей. Фінальні результати разом із часовою міткою перевірки записуються в історію сканувань. Нормативну відповідність розрахунків та структуру вихідних даних на кожному етапі регламентує об'єкт-посилання міжнародного стандарту CVSS v3.1.

У даному розділі кваліфікаційної роботи виконано повний цикл передпроєктного обстеження та функціонального моделювання ІС інвентаризації та оцінки ризиків мережевої інфраструктури. На основі аналізу предметної області обґрунтовано актуальність розроблення ІС інвентаризації мережевих активів та автоматизованого оцінювання ризиків. Дослідження сучасного кіберсередовища показало, що ручне ведення технічних паспортів ІС та тривалий час реагування на уразливості (понад 15 днів) створюють операційні бар'єри, які потребують автоматизації процесів контролю конфігурацій.

Проведено аналіз існуючих комерційних та open-source аналогів (Zabbix, SolarWinds NCM, Tenable Nessus). Встановлено, що жодне з наявних рішень не поєднує безкоштовну модульну архітектуру із вбудованим математичним апаратом розрахунку балів CVSS v3.1 та інструментами візуальної оркестрації, що повністю підтверджує доцільність створення власного програмного продукту.

За допомогою методології структурно-функціонального моделювання IDEF0 формалізовано межі проектованої ІС, визначено керуючі впливи державних і міжнародних стандартів, а також програмно-технічні механізми реалізації системи. Побудовано діаграми потоків даних DFD (контекстну та першого рівня) у нотації Гейна-Сарсона, що дозволило деталізувати конвеєр обробки інформації від первинного сканування мережі до наповнення реляційного сховища SQLite.

Описано функціональні вимоги до системи з позиції користувача за допомогою UML-діаграми прецедентів. Розроблено та формалізовано логічну схему потоку робіт аналітичного модуля, що описує покроковий алгоритм парсингу відкритих портів, перевірки правил безпеки, розрахунку метрик CVSS v3.1 та генерації текстових рекомендацій для адміністратора, що формує стійкий теоретичний базис для подальшого архітектурного проектування системи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура інформаційної системи

Проєктована інформаційна система інвентаризації та аналізу конфігурації мережевих пристроїв побудована на основі трирівневої клієнт-серверної архітектури. Такий підхід забезпечує чітке розмежування відповідальності між компонентами та незалежність рівнів обробки даних.

Перший рівень − рівень збору даних − реалізовано на базі платформи Node-RED. Цей компонент виконує функції оркестратора: ініціює планові або ручні сканування мережі, імітує SNMP-опитування мережевих пристроїв, збирає конфігураційні дані та передає їх до Flask API у форматі JSON. Використання Node-RED як окремого рівня дає змогу замінювати джерела даних (наприклад, переходити від імітації до реального протоколу SNMP) без внесення змін до коду бекенду.

Другий рівень − рівень обробки та зберігання − реалізовано на основі вебфреймворку Flask (мова програмування Python). Цей рівень містить: REST API для прийому даних від Node-RED, модуль оцінки ризиків безпеки (risk_engine), модуль мережевого сканування (scanner), а також ORM-рівень доступу до бази даних SQLite за допомогою інструментарію SQLAlchemy. Flask-бекенд обробляє вхідні дані, запускає аналіз конфігурацій згідно з вісьмома правилами безпеки та зберігає результати в реляційній базі даних [23].

Третій рівень − рівень представлення − реалізовано у вигляді вебінтерфейсу на основі технологій HTML5, Bootstrap 5 та бібліотеки Chart.js. Інтерфейс містить інформаційну панель (дашборд) з графіками, реєстр пристроїв із можливістю фільтрації, сторінки детальної інформації про пристрої з переліком вразливостей та рекомендаціями, а також функцію експорту звітів у форматах CSV та PDF [24, 25].

У таблиці 2.1 наведено базові компоненти архітектури розробленої системи.

Таблиця 2.1 − Компоненти архітектури системи

Рівень	Технологія	Відповідальність
Збір даних	Node-RED	Оркестрація процесів, імітація SNMP, передача даних до API
Обробка / зберігання	Flask, SQLAlchemy, SQLite	Реалізація REST API, аналіз безпекових ризиків, збереження даних
Представлення даних	HTML5, Bootstrap 5, Chart.js	Візуалізація (дашборд, реєстр, формування звітів)

Взаємодія між рівнями здійснюється за допомогою архітектурного стилю REST API: Node-RED надсилає POST-запити на ендпоінт /api/devices/import, а спеціалізована вкладка Dashboard у Node-RED опитує ендпоінти /devices/api/list та /api/risks/summary з інтервалом у 10 секунд. Доступ до вебінтерфейсу, який обслуговується сервером Flask, здійснюється безпосередньо через веббраузер за адресою http://localhost:5000.

Описана архітектурна схема повністю відповідає принципу розділення відповідальності (Separation of Concerns) і забезпечує можливість незалежного масштабування та модифікації кожного з рівнів без впливу на інші компоненти системи.

2.2 Модель даних системи

Проєктовану модель даних системи розроблено відповідно до принципів нормалізації реляційних баз даних. База даних SQLite містить п'ять взаємопов'язаних таблиць, що забезпечують збереження інформації про мережеві пристрої, їхні конфігурації, виявлені вразливості та загальні результати сканувань. UML-діаграму класів моделі даних наведено на рисунку 2.1.

Центральною сутністю моделі є клас Device, який зберігає реєстр мережевих пристроїв. Кожен запис містить унікальний ідентифікатор (id), IP-адресу, MAC-адресу, мережеве ім'я хоста (hostname), тип пристрою (маршрутизатор, комутатор тощо), назву вендора, дані про операційну систему, перелік відкритих портів, поточний статус та часові мітки першого й останнього виявлення пристрою в мережі.

Рисунок 2.1 − Діаграма класів моделі даних системи
Із сутністю Device за допомогою зовнішніх ключів пов'язані три залежні таблиці, що реалізують зв'язки типу «один-до-багатьох» (1 до 0..*):
1 таблиця Configuration зберігає знімки конфігураційних файлів пристроїв із прив'язкою до часової мітки (timestamp), що дає змогу відстежувати та порівнювати зміни конфігурацій у часі за допомогою відповідних методів класу;
2 таблиця Vulnerability містить інформацію про виявлені на пристроях вразливості з детальним описом, рівнем серйозності, оцінкою за шкалою CVSS та текстовими рекомендаціями щодо їх усунення.
3 таблиця RiskScore фіксує розраховані модулем аналізу рівні безпекових ризиків для кожного окремого пристрою із зазначенням числового балу та категорії небезпеки (Critical/High/Medium/Low).
Окремим компонентом моделі є таблиця ScanHistory, яка функціонує як системний журнал усіх проведених сесій сканування мережі. Вона фіксує ідентифікатор сканування, межі сканованої підмережі, загальну кількість виявлених та нових пристроїв, поточний статус операції, а також точний час початку й завершення процесу опитування.
Детальний опис призначення та структури основних таблиць бази даних наведено в таблиці 2.2.

Таблиця 2.2 − Структура основних таблиць бази даних

Таблиця	Ключові поля	Призначення
Device	id, ip_address, mac_address, hostname, device_type, vendor, os_info, open_ports, status, first_seen, last_seen	Ведення централізованого реєстру виявлених мережевих пристроїв
Configuration	id, device_id, config_data, timestamp, backup_path	Зберігання резервних копій та конфігураційних знімків пристроїв
Vulnerability	id, device_id, vuln_name, description, severity, cvss_score, recommendation	Фіксація виявлених уразливостей та формування рекомендацій щодо їх усунення
RiskScore	id, device_id, score, risk_level, calculated_at	Зберігання результатів обчислення поточних рівнів безпекових ризиків
ScanHistory	id, scan_id, subnet, devices_found, new_devices, status, started_at, completed_at	Логування сесій сканування та ведення загального журналу операцій

Таблиці Configuration, Vulnerability та RiskScore пов'язані з базовою сутністю Device відносинами «один-до-багатьох» через зовнішній ключ device_id. Це дозволяє зберігати ретроспективну історію конфігурацій та динаміку ризиків для кожного вузла мережі, що необхідно для проведення аудиту безпеки. Таблиця ScanHistory є автономною та агрегує загальну статистику сесій опитування підмереж.
Вибір СКБД SQLite обумовлений відсутністю потреби в окремому сервері, оскільки база даних зберігається у єдиному файлі inventory.db. Водночас використання ORM-фреймворку SQLAlchemy ізолює логіку додатка від специфіки СКБД. Це забезпечує масштабованість системи та можливість переходу на потужніші рішення (PostgreSQL, MySQL) шляхом зміни лише рядка

з'єднання.
2.3 Алгоритм оцінки ризиків безпеки
Алгоритм оцінки безпекових ризиків реалізовано в програмному модулі risk_engine.py. Послідовність його виконання відображено на UML-діаграмі діяльності (рис. 2.2). Процес аналізу складається з трьох послідовних фаз.

 Рисунок 2.2 − Діаграма діяльності алгоритму оцінки ризиків безпеки
На першій фазі алгоритм отримує конфігураційні параметри пристрою з бази даних і виконує парсинг переліку відкритих портів та мережевих протоколів (зокрема ssh_enabled, telnet_enabled, snmp_version тощо).

На другій фазі здійснюється перевірка правил безпеки (на рис. 2.2 наведено чотири базові критерії: використання незахищених протоколів Telnet, SNMP v1/v2c, HTTP та FTP). Перевірка правил відбувається незалежно: виявлення одного порушення не зупиняє аналіз інших. У разі невідповідності критерію безпеки до списку вразливостей пристрою додається запис із фіксованим балом за шкалою CVSS.
На третій фазі обчислюється загальний показник ризику як середнє значення накопичених балів CVSS. Отриманий результат порівнюється з пороговими значеннями для класифікації рівня загрози: Critical (від 9,0 до 10,0 балів), High (від 7,0 до 8,9 балів), Medium (від 4,0 до 6,9 балів) або Low (менше 4,0 балів). Після цього для кожної вразливості формуються текстові рекомендації щодо її усунення, а фінальні дані експортуються у таблиці Vulnerability та RiskScore.
Детальний перелік усіх восьми правил оцінки ризиків, їхні критерії та відповідні CVSS-бали наведено в таблиці 2.3.
Таблиця 2.3 − Правила оцінки ризиків та CVSS-бали

No	Правило безпеки	Бал CVSS	Рівень загрози
1	Використання Telnet замість SSH (порт 23)	7,5	Високий
2	Використання застарілих версій протоколу SNMP v1/v2c без шифрування	6,5	Середній
3	Передача даних через HTTP (порт 80) за відсутності шифрування HTTPS	5,0	Середній
4	Використання незахищеного протоколу передачі файлів FTP (порт 21)	6,0	Середній
5	Наявність відкритих портів віддаленого керування (Winbox 8291, RDP 3389)	7,0	Високий
6	Використання стандартних чи слабких строк автентифікації SNMP (public/private)	5,5	Середній
7	Виявлення застарілих версій операційних систем (Cisco IOS 12.x, RouterOS < 7.x)	6,5	Середній
8	Відсутність обов'язкової автентифікації на критично важливих сервісах	8,0	Високий

Ключовою особливістю розробленого алгоритму є накопичувальний характер оцінювання: кожне нове виявлене порушення збільшує підсумковий інтегральний бал ризику мережевого пристрою. Такий підхід дозволяє об'єктивно

відобразити реальний стан захищеності інфраструктури: пристрій із кількома вразливостями середнього рівня критичності становить значно більшу загрозу для мережі, ніж вузол з одним аналогічним дефектом безпеки. Зазначена модель агрегування ризиків повністю відповідає методології ризик-менеджменту інформаційної безпеки, регламентованій міжнародним стандартом ISO/IEC 27005.

2.4 Діаграма варіантів використання (Use Case Diagram)

Функціональні вимоги до ІС інвентаризації з точки зору взаємодії зовнішніх сутностей описано за допомогою UML-діаграми варіантів використання (рис. 2.3).

Рисунок 2.3 − Діаграма варіантів використання системи інвентаризації

Адміністратор системи безпосередньо взаємодіє з такими прецедентами, як перегляд історії сканувань, ініціалізація сканування мережі, робота з реєстром пристроїв, налаштування параметрів функціонування та моніторинг статусу системи. Базовий варіант використання «Переглянути реєстр пристроїв» пов'язаний відношенням розширення (extend) із прецедентом «Експортувати звіт CSV/PDF», що вказує на опціональну можливість вивантаження звітних документів.

Агент Node-RED Agent виконує автоматизовані фонові функції: збір конфігураційних даних через протокол SNMP (або за допомогою модуля імітації), передачу сформованих пакетів до Flask API, опитування ендпоінтів для отримання зведень про ризики та візуалізацію поточного стану процесів на інтегрованій графічній панелі Dashboard.

Модуль Risk Engine фоново забезпечує логіку перевірки конфігурацій за правилами безпеки, розрахунок балів CVSS та формування рекомендацій. Прецедент «Запустити сканування мережі» пов'язаний відношенням включення (include) із прецедентом «Аналізувати конфігурації за правилами». Це відображає обов'язковий автоматичний запуск аналітичного модуля після кожного завершеного циклу збору мережевих даних.

2.5 Діаграма послідовності (Sequence Diagram)

Sequence Diagram (рис. 2.4) описує динамічну поведінку системи та взаємодію її компонентів у часі на прикладі сценарію ручного запуску сканування мережі, який є основним функціональним процесом системи.

Сценарій починається з дій користувача, який ініціює команду «Ручний запуск сканування» через вебінтерфейс платформи Node-RED (крок 1). Керуючий потік Node-RED Flow фіксує поточну часову мітку й унікальний ідентифікатор сесії (крок 2), після чого завантажує конфігураційні параметри пристроїв з імітаційного сховища (mock-сервера) або за допомогою реального опитування через SNMP (крок 3). Сформований пакет даних у форматі JSON надсилається за

допомогою POST-запиту на ендпоінт /api/devices/import вебсервера Flask API (крок 4).

Рисунок 2.4 − Діаграма послідовності сценарію ручного сканування мережі

Інтерфейс Flask API здійснює валідацію отриманої структури даних (крок 5) та ініціює запис відомостей про пристрої до бази даних SQLite за допомогою SQL-запиту INSERT INTO devices (кроки 6 − 7). Одразу після збереження записів у базі даних автоматично тригерується процес оцінки безпекових ризиків через аналітичний модуль Risk Engine (крок 8). Модуль послідовно перевіряє файли конфігурацій на відповідність вісьмом правилам безпеки (крок 9), розраховує бали за шкалою CVSS (крок 10) та повертає структурований список виявлених дефектів конфігурації (крок 11).

Вебфреймворк Flask зберігає інформацію про знайдені вразливості та обчислені рівні ризику у відповідних таблицях бази даних Vulnerability та RiskScore (кроки 12 − 13), після чого надсилає клієнту Node-RED HTTP-відповідь із підтвердженням успішного імпорту (крок 14).

Після отримання підтвердження Node-RED Flow автоматично надсилає GET-запит на ендпоінт /api/risks/summary для отримання актуальної аналітики

(крок 15). Flask API виконує вибірку зведеної статистики з бази даних (кроки 16–17) і повертає JSON-пакет із розподілом рівнів загроз (крок 16–18). Сценарій завершується обробкою зведення на стороні Node-RED та автоматичним оновленням графічного дашборду користувача (кроки 19 − 20).
Розглянута послідовність взаємодій наочно демонструє ключові переваги обраного архітектурного рішення: чіткий конвеєрний розподіл зон відповідальності між модулями збору, обробки й аналізу, а також гарантоване автоматичне тригерування оцінки ризиків безпеки безпосередньо в момент оновлення реєстру мережевих пристроїв.

2.6 Проєктування маршрутів та REST API системи

Програмний інтерфейс REST API є центральним вузлом взаємодії між автономними компонентами системи. Його спроєктовано відповідно до базових принципів архітектурного стилю REST: використання стандартних HTTP-методів, відсутність збереження стану сесії на стороні сервера, застосування формату JSON для обміну даними та використання семантично зрозумілих URI-адрес для ідентифікації ресурсів.
У таблиці 2.4 наведено повну специфікацію вебмаршрутів та ендпоінтів програмного інтерфейсу розробленої системи.
Таблиця 2.4 − Специфікація вебмаршрутів та ендпоінтів системи

Метод HTTP	Маршрут (ендпоінт)	Функціональний опис ресурсу	Тип відповіді
GET	/	Головна сторінка інформаційної панелі (дашборд)	200 HTML
GET	/devices/	Вебсторінка з реєстром мережевих пристроїв	200 HTML
GET	/devices/{id}	Сторінка детальних відомостей про пристрій та вразливості	200 HTML
GET	/devices/api/list	Отримання структурованого списку пристроїв	200 JSON
POST	/api/scan	Ініціалізація та запуск сканування заданої підмережі	200 JSON
POST	/api/devices/import	Імпорт конфігураційних даних пристроїв з Node-RED	200 JSON

Продовження таблиці 2.4

Метод HTTP	Маршрут (ендпоінт)	Функціональний опис ресурсу	Тип відповіді
GET	/api/scan/history	Отримання журналу проведених сесій сканування	200 JSON
GET	/api/risks	Отримання повного переліку знайдених уразливостей	200 JSON
GET	/api/risks/summary	Отримання зведеної статистики ризиків за рівнями критичності	200 JSON
POST	/api/risks/{id}/resolve	{id}/resolve**Зміна** статусу виявленої	

вразливості на «вирішено»
200 JSON
GET /reports/csv Експорт та завантаження реєстру пристроїв
у форматі CSV
200 CSV
GET /reports/pdf Генерація та завантаження аналітичного
звіту у форматі PDF
200 PDF
Ключовим інтерфейсом взаємодії для інтеграції з платформою Node-RED є ендпоінт POST /api/devices/import. Він приймає JSON-пакет, що містить ідентифікатор джерела даних та масив об'єктів devices, кожен з яких описує повний набір технічних атрибутів пристрою. При отриманні запиту Flask-бекенд виконує комбіновану операцію оновлення або додавання записів (upsert): якщо пристрій із такою IP-адресою вже зареєстрований у базі даних, його параметри актуалізуються, інакше − створюється новий запис. Після успішного збереження транзакції автоматично викликається програмний модуль risk_engine для перерахунку безпекових ризиків. HTTP-відповідь сервера містить статус виконання операції та кількість імпортованих об'єктів.
Ендпоінт GET /api/risks/summary використовується потоком Node-RED Flow для безперервного моніторингу стану захищеності інфраструктури. Сервер повертає агреговані дані у форматі {"critical": 3, "high": 1, "medium": 0, "low": 1, "total_vulnerabilities": 12}. На основі цих відомостей Node-RED динамічно оновлює колірну індикацію станів вузлів на графічній панелі користувача та дублює критичні сповіщення в системну панель налагодження (debug-панель).
У даному розділі виконано повне проєктування ІС інвентаризації та аналізу конфігурації мережевих пристроїв, у результаті чого отримано такі науково-технічні результати:

1. Обґрунтовано трирівневу клієнт-серверну архітектуру: рівень збору даних (Node-RED), обробки та зберігання (Flask, SQLite), представлення (HTML5, Bootstrap 5, Chart.js). Забезпечено незалежність рівнів та можливість модифікації окремих компонентів.
2. Спроєктовано нормалізовану реляційну модель даних, яка складається з п'яти взаємопов'язаних таблиць (Device, Configuration, Vulnerability, RiskScore, ScanHistory). Дана структура забезпечує збереження цілісності інформації, накопичення ретроспективної історії змін та повноцінну підтримку процесів аудиту безпеки.
3. Побудовано діаграми потоків даних (DFD) рівнів 1 та 2, які формалізують маршрути обробки інформації та логіку взаємодії підсистем.
4. Розроблено кумулятивний алгоритм оцінки ризиків на основі восьми правил безпеки (CVSS v3.1) із чотирирівневою шкалою загроз (Critical/High/Medium/Low). Логіку алгоритму подано у вигляді UML-діаграми діяльності.
5. Спроєктовано специфікацію інтерфейсу REST API, яка містить дванадцять ендпоінтів та забезпечує стандартизовану безлікову взаємодію між компонентами системи, а також безпроблемну інтеграцію з агентом збору даних.
6. Розроблено логічну структуру потоків Node-RED Flow, яка містить три функціональні вкладки: збір конфігураційних даних, планове автоматичне сканування мережі та безперервний моніторинг стану безпеки інфраструктури в режимі реального часу з колірною індикацією рівнів загрози.
Прийняті у розділі проєктні рішення повністю забезпечують виконання всіх сформульованих у першому розділі функціональних вимог (FR-01 − FR-08) та створюють надійну конструктивну основу для програмної реалізації системи, опис якої наведено в розділі 3.

22
3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ
3.1 Структура програмного коду системи
Програмну реалізацію інформаційної системи виконано мовою програмування Python 3.11 з використанням мікрофреймворку Flask 3.0. В архітектурі бекенду застосовано шаблон проєктування «Фабрика додатків» (Application Factory). Ключова функція create_app() у модулі app/__init__.py

динамічно створює та конфігурує екземпляр Flask-додатка, реєструє ізольовані компоненти маршрутизації (блюпринти) та ініціалізує плагіни розширень.
Файлову структуру проєкту організовано за принципом розділення відповідальності (Separation of Concerns): кожен програмний модуль відповідає за чітко визначений сервіс системи. Загалом архітектура містить 11 основних Python-файлів та 4 HTML-шаблони представлення даних. Детальний опис файлової структури та призначення компонентів наведено в таблиці 3.1.
Таблиця 3.1 − Файлова структура програмного коду системи

Шлях до файлу проєкту	Функціональне призначення модуля	Ключові класи та методи / Маршрути
run.py	Точка входу для запуску вебсервера Flask	app.run(),активація debug-режиму
init_db.py	Скрипт ініціалізації БД та тестового наповнення	create_tables(), seed_devices()
app/__init__.py	Конфігураційна фабрика Flask-додатка	create_app(), db.init_app()
app/models.py	Опис ORM-моделей бази даних SQLAlchemy	Класи: Device, Configuration, Vulnerability, RiskScore, ScanHistory
app/risk_engine.py	Програмний модуль оцінки безпекових ризиків	Методи: analyze_device(), calculate_risk_score()
app/scanner.py	Модуль імітаційного та мережевого сканування	Методи scan_network(), mock_scan()

Продовження таблиці 3.1

Шлях до файлу проєкту	Функціональне призначення модуля	Ключові класи та методи / Маршрути
app/routes/main.py	Контролер головної сторінки інформаційної панелі	GET /
app/routes/devices.py	Маршрути ведення реєстру пристроїв	GET /devices/, GET /devices/{id}, GET /devices/api/list
app/routes/scan.py	API-ендпоінти керування скануванням та імпорту	POST /api/scan, POST /api/devices/import
app/routes/risks.py	risks.pyAPI-ендпоінти аналітики та обробки ризиків	

GET /api/risks, GET
/api/risks/summary,
POST
/api/risks/{id}/resolve
app/routes/reports.py Маршрути генерації експортних
звітів
GET /reports/csv, GET
/reports/pdf
templates/base.html Базовий каркас інтерфейсу (шаблон) Розмітка Bootstrap 5,
бічна панель, меню
templates/dashboard.h
tml
dashboard.html **Візуальне**
відображення графіків та
статистики
Інтеграція Chart.js,
загальні метрики
templates/devices.html devices.html **інтерфейс**
інтерактивного реєстру пристроїв
Таблиця даних,
фільтри сортування
templates/device_detai
l.html
Відображення детальних карт
вразливостей
Картки дефектів, бали
CVSS, рекомендації

Використання системи блюпринтів (Blueprints) дозволяє групувати маршрути Flask за функціональними доменами, забезпечуючи ізольовану розробку, тестування та масштабованість підсистем.

Первинне розгортання БД здійснюється командою python init_db.py, яка створює локальний файл inventory.db і наповнює його тестовими профілями п'яти пристроїв із різним рівнем захищеності (рис. 3.1):

− два критичні вузли (Cisco router і Cisco switch із протоколами Telnet та SNMP v1);

− один із високим ризиком (Windows Server із відкритими портами FTP та RDP);

Рисунок 3.1 − Результат виконання скрипту ініціалізації та первинного наповнення бази даних

− один із низьким ризиком (Ubuntu Server із коректно налаштованими SSH та HTTPS);

− один із критичним ризиком (MikroTik із доступним HTTP та стандартним SNMP public community).

3.2 Реалізація модуля оцінки ризиків безпеки

Програмний модуль risk_engine.py є центральним аналітичним компонентом системи, який забезпечує виконання алгоритму виявлення вразливостей та комплексної оцінки безпекових ризиків. У модулі реалізовано вісім детермінованих правил перевірки конфігураційних файлів, кожне з яких відповідає конкретному дефекту безпеки з фіксованим балом за міжнародною шкалою CVSS v3.1.

3.2.1 Функція analyze_device()

Головна функція аналізу приймає як вхідний параметр об'єкт моделі пристрою, ініціює послідовний запуск інспекційних правил та повертає структурований список виявлених уразливостей. Повну програмну реалізацію цієї функції наведено в Додатку Б.

Представлений алгоритм демонструє детермінований підхід до аналізу мережевих вузлів. Процедура інспекції оперує як явними булевими ознаками конфігурації (наприклад, device.telnet_enabled), так і непрямими факторами,

отриманими шляхом попереднього парсингу й валідації масиву відкритих мережевих портів пристрою (open_ports).
Кожне з восьми правил виконується незалежно від інших, що дозволяє виявити максимальний спектр проблем безпеки за один цикл аналізу. Результатом виконання функції є динамічний масив словників, що містить назву вразливості, її категорію, точний бал CVSS v3.1 та чітку інструкцію (рекомендацію) для системного адміністратора щодо усунення дефекту. Це створює інформаційний базис для подальшого агрегування даних у блоці розрахунку інтегральних ризиків системи.

3.2.2 Функція calculate_risk_score()

Програмна функція обчислення рівня ризику приймає масив виявлених дефектів, а повертає інтегральний бал і категорію небезпеки. Повну реалізацію алгоритму наведено в Додатку Б.
Математична модель реалізує кумулятивний принцип оцінювання з обмеженням максимуму на рівні 10,0 балів (згідно з CVSS v3.1). Алгоритм сортує вразливості за спаданням: найбільш небезпечний дефект враховується повністю, а внесок кожного наступного мультиплікується на понижувальний коефіцієнт (0,7). Зниження маргінального впливу додаткових загроз відображає логіку ризик-менеджменту: якщо вузол уже має критичну вразливість, інші дефекти підвищують загальний ризик із меншою питомою вагою, оскільки компрометація пристрою високоімовірна за першим фактором. Результат класифікується за чотирирівневою шкалою та округлюється до десятих.

3.3 Реалізація модуля сканування мережі

Програмний модуль scanner.py реалізує два режими функціонування: реальний моніторинг інфраструктури за допомогою утиліти Nmap та демонстраційний режим з імітаційними даними. Така дволінійна архітектура дозволяє повноцінно запускати систему в ізольованих середовищах без

безпосереднього доступу до реального мережевого обладнання, що є важливим для навчальних та демонстраційних цілей.

3.3.1 Ендпоінт POST /api/scan

Програмна логіка керування процесом сканування інтегрована в API-маршрут api_scan(), повну реалізацію якого наведено в Додатку Б.
Метод приймає POST-запит із зазначенням цільової підмережі та ініціює об'єкт nmap.PortScanner() для проведення швидкого ping-сканування (-sn -T4). У разі виникнення виключної ситуації (наприклад, відсутності встановлених системних бінарних файлів Nmap, недостатніх мережевих прав або помилки під час виконання сканування) система автоматично перемикається в режим емуляції − завантажуються заздалегідь згенеровані імітаційні дані пристроїв, що забезпечує працездатність API в тестових та демонстраційних середовищах без реальної мережевої інфраструктури.
Після отримання списку активних IP-адрес алгоритм ітераційно перевіряє їхню наявність у базі даних. Для нових пристроїв викликається процедура первинного аналізу та збереження, а для вже зареєстрованих вузлів − актуалізується часова мітка останнього виявлення. Завершальним етапом є запис результатів сесії до журналу ScanHistory та повернення клієнту структурованої JSON-відповіді зі статистикою сканування.

3.3.2 Ендпоінт POST /api/devices/import (інтеграція з Node-RED)

Ключовим програмним інтерфейсом для інтеграції бекенду з платформою оркестрації даних Node-RED є API-ендпоінт /api/devices/import. Цей шлях забезпечує прийом масивів даних про виявлені пристрої у форматі JSON, виконує атомарну операцію синхронізації (upsert) та автоматично ініціює конвеєр оцінки ризиків для кожного об'єкта. Повну програмну реалізацію цього методу контролера наведено в Додатку Б.
На першому етапі функція здійснює первинну валідацію структури вхідного JSON-пакета на наявність обов'язкового масиву devices. У разі успішної перевірки

запускається ітераційний цикл обробки даних кожного мережевого вузла. Пошук

існуючих записів у базі даних здійснюється за унікальним індексом IP-адреси. Якщо вузол не зареєстровано, створюється новий екземпляр моделі Device, інакше − за допомогою вбудованої функції setattr() динамічно оновлюються його атрибути та актуалізується часова мітка виявлення.

Особливістю реалізації є безпосередня інтеграція конвеєра імпорту з аналітичним модулем безпеки: одразу після фіксації транзакції в СКБД послідовно викликаються функції analyze_device() та calculate_risk_score(). Виявлені дефекти конфігурації та розрахований інтегральний бал ризику каскадно записуються у залежні таблиці через допоміжні процедури збереження. Метод завершується поверненням клієнту статусу операції та кількості успішно оброблених записів.

3.4 Реалізація інтеграції з Node-RED

Потоки керування Node-RED Flow організовано у вигляді трьох функціональних вкладок (рис. 3.2).

Рисунок 3.2 − Графічна схема потоку збору даних (вкладка Network Inventory) у середовищі Node-RED

Платформа виконує роль зовнішнього оркестратора процесів збору, агрегування та безперервного моніторингу мережевих даних. Конфігураційна структура потоків зберігається у серіалізованому файлі nodered_flow.json та імпортується до середовища Node-RED за допомогою вбудованого інструментарію імпорту документів [26].

3.4.1 Вкладка «Network Inventory − збір даних»

Головна функціональна вкладка реалізує послідовний ланцюжок із дев'яти логічних вузлів: Inject − Function (фіксація часової мітки) − Function (формування конфігурацій пристроїв) − HTTP Request (надсилання POST-запиту на ендпоїнт /import) − Function (валідація результату відповіді). Паралельно з цим, з інтервалом у 30 секунд, ініціюється процедура симуляції SNMP-опитування інфраструктури за допомогою окремого вузла Inject, налаштованого на періодичний запуск (рис. 3.3).

Рисунок 3.3 − Структура потоку інвентаризації та метадані вузла ініціалізації сканування у Node-RED

Як показано на рис. 3.3, ручний запуск конвеєра забезпечується дискретним вузлом типу Inject з унікальним системним ідентифікатором inject_manual_scan,

що дозволяє відокремити ручні запити оператора від автоматичних тригерів планувальника».

Ключовий керуючий вузол Function («Завантажити конфігурації пристроїв») динамічно генерує масив із п'яти об'єктів пристроїв, що містять повний набір технічних атрибутів, необхідних для коректної оцінки ризиків: ip_address, mac_address, hostname, device_type, vendor, os_info, open_ports, а також булеві прапорці активації служб ssh_enabled, telnet_enabled, snmp_community, snmp_version, http_enabled, https_enabled та ftp_enabled. Ці відомості повністю імітують текстові файли конфігурацій реального мережевого обладнання, експортовані із середовища моделювання Cisco Packet Tracer.

3.4.2 Вкладка «Scheduler − планові сканування»

Ця вкладка містить три вузли типу Inject із різними налаштуваннями часових тригерів:

1 ранкове сканування − налаштування планувальника cron: 0 8 * * * (щодня о 08:00);

2 вечірнє сканування (параметри cron: 0 20 * * * (щодня о 20:00);

3 щогодинна перевірка доступності вузлів мережі – повторення кожні 3600 секунд.

Імпульси від усіх трьох тригерів направляються до спільного обробника Function, який виконує наступні дії:

1.Додає до корисного навантаження (payload) поле джерела запиту: source: 'nodered_schedule'.

2. Додає поточну мітку часу.

3. Ініціює автоматичний POST-запит до REST API бекенду на ендпоінт /api/scan.

Схему взаємодії наведено на рисунку 3.4.

Рисунок 3.4 − Графічна схема підсистеми автоматичного планування сканувань (вкладка Scheduler) у середовищі Node-RED

3.4.3 Вкладка «Dashboard − Моніторинг»

Керуючий потік цієї вкладки забезпечує періодичне опитування інформаційних ендпоінтів REST API. Вузлова структура підсистеми моніторингу в режимі реального часу наведена на рисунку 3.5.

Рисунок 3.5 − Графічна схема підсистеми періодичного оновлення метрик безпеки (вкладка Dashboard — Моніторинг) у середовищі Node-RED

Вхідний вузол типу Inject із параметром циклічного повторення кожні 10 секунд паралельно ініціює два HTTP Request запити до бекенду: GET /devices/api/list та GET /api/risks/summary. Спеціалізовані функції-обробники здійснюють парсинг отриманих JSON-відповідей, підраховують динамічну кількість активних вузлів і класифікують їх за рівнем загрози.

Особливістю реалізації є автоматична зміна колірного статусу відображення вузлів під кожним блоком залежно від критичності виявлених дефектів конфігурації:

− зелений колір індикатора встановлюється за відсутності порушень;

− жовтий колір сигналізує про наявність вразливостей високого рівня (High);

− червоний колір активується у разі фіксації критичних загроз, як це продемонстровано на прикладі поточного стану системи на рис. 3.6.

На рис. 3.6 наведено панель налагодження (debug panel) у середовищі Node-RED. Вона демонструє системні повідомлення та обмін структурованими JSON-пакетами між Node-RED та Flask API в реальному часі (видно виклики об'єктів Dashboard Live, SNMP дані, розподіл ризиків critical: 3, high: 1... тощо).

Рисунок 3.6 − Лог відлагодження інформаційних потоків у debug-панелі Node-RED

3.5 Реалізація веб-інтерфейсу користувача

Вебінтерфейс інформаційної системи реалізовано у вигляді чотирьох взаємопов'язаних HTML-шаблонів на основі фреймворку Bootstrap 5 із розширенням бібліотеки Chart.js для графічної візуалізації аналітичних даних. Усі компоненти представлення наслідують загальний базовий макет base.html, який містить уніфіковану бічну панель навігації, верхню панель керування застосунком та глобальні блоки підключення скриптів із зовнішніх мереж доставки контенту.

3.5.1 Дашборд (dashboard.html)

Головна вебсторінка системи відображає інтегровану аналітичну статистику, згруповану у вигляді чотирьох інформаційних карток-віджетів: загальна кількість проінвентаризованих пристроїв, кількість активних вузлів у мережі на поточний момент, кількість систем із критичним рівнем безпекового ризику, а також сумарна кількість виявлених уразливостей конфігурації (рис. 3.7, 3.8).

Рисунок 3.7 − Головна сторінка інформаційної панелі (дашборд) вебінтерфейсу системи

Рисунок 3.8 − Дашборд вебінтерфейсу системи

Нижче інформаційних карток розміщено кругову Chart.js-діаграму, що наочно відображає відсотковий та кількісний розподіл мережевих пристроїв за категоріями ризику з використанням відповідного колірного кодування: червоний

колір − Critical, помаранчевий − High, жовтий − Medium, зелений − Low.
Окрім графіків, дашборд містить інтерактивну таблицю останніх сесій моніторингу з реляційного журналу ScanHistory та перелік пристроїв із найвищими розрахованими показниками загрози безпеці. Синхронізація та завантаження метрик здійснюється в асинхронному режимі за допомогою AJAX-запитів до Flask-API безпосередньо в момент ініціалізації вебсторінки у браузері.

3.5.2 Реєстр пристроїв (devices.html)

Сторінка реєстру відображає повний список мережевих пристроїв у вигляді таблиці з підтримкою фільтрації за типом пристрою (роутер, комутатор, хост) та рівнем ризику (Critical, High, Medium, Low) (рис. 3.9). Кожен рядок таблиці забарвлений відповідно до рівня ризику та містить посилання на сторінку деталей.

Фільтри реалізовані як кнопки Bootstrap зі стилем badge, які при натисканні передають параметри до Flask маршруту через GET-запит [23, 24].
Рисунок 3.9 − Вебсторінка інтерактивного інвентарного реєстру мережевих пристроїв у браузері
На рис. 3.7 наведено реалізацію інвентарного реєстру в браузері: пошуковий рядок, випадаючі списки фільтрації за рівнем ризику та типом пристрою, а також головну таблицю з колірними значками рівнів небезпеки (критичний, високий, низький) та розрахованими балами інтегрального ризику (10.0, 9.4, 8.2, 0.0).
Як продемонстровано на рис. 3.9, розроблений інтерфейс реєстру успішно відображає зведені технічні характеристики проінвентаризованих вузлів: IP-адреси, мережеві імена хостів, типи та виробників обладнання, версії встановлених операційних систем (зокрема Cisco IOS 15.2, RouterOS 7.1, Windows Server 2019 тощо) та переліки відкритих портів керування, що підтверджує коректність роботи ORM-рівня доступу до бази даних».

3.5.3 Деталі пристрою (device_detail.html)

Вебсторінка детального аналізу призначена для відображення вичерпної технічної та аналітичної інформації про вибраний мережевий вузол. Інтерфейс картки пристрою акумулює три блоки даних: загальні технічні характеристики (IP- та MAC-адреси, мережеве ім'я хоста, категорія й виробник обладнання, версія операційної системи), поточні конфігураційні параметри (масив відкритих портів, статус активації протоколів) та інтегральний рівень безпекового ризику з точним числовим балом.
Центральним елементом сторінки є таблиця виявлених уразливостей конфігурації. Вона містить детерміновані відомості про кожну загрозу: ідентифікаційну назву дефекту, детальний технічний опис, категорію серйозності, індивідуальний бал за шкалою CVSS v3.1 та покрокову рекомендацію щодо практичного усунення уразливості адміністратором [26].
У системі реалізовано функціонал оперативного реагування на інциденти: зміна статусу виявленої вразливості на «вирішено» здійснюється безпосередньо з інтерфейсу за допомогою елемента керування, який ініціює асинхронний POST-запит до REST API бекенду на ендпоінт /api/risks/{id}/resolve без повної перезагрузки сторінки користувача.

3.5.4 Звіти (reports.py)

Підсистема формування звітної документації реалізована у програмному модулі reports.py та підтримує два формати експорту даних.
Ендпоінт GET /reports/csv генерує файл у форматі CSV з актуальним інвентарним списком мережевого обладнання, що містить технічні атрибути та рівні ризиків пристроїв. Використання вбудованого модуля Python csv із кодуванням UTF-8 забезпечує коректне відображення кирилиці в табличних процесорах.
Ендпоінт GET /reports/pdf за допомогою бібліотеки ReportLab динамічно формує структурований документ у форматі PDF. Звіт містить заголовок, часову

мітку генерації, зведену статистику безпекових ризиків та таблицю пристроїв із виявленими дефектами конфігурації.

3.6 Тестування інформаційної системи

Верифікація працездатності розробленої ІС проводилася у три послідовні етапи: функціональне тестування REST API за допомогою консольних запитів, інтеграційна перевірка взаємодії платформи Node-RED із Flask-бекендом, а також демонстраційні випробування в імітаційному середовищі моделювання інфраструктури Cisco Packet Tracer.

3.6.1 Функціональне тестування REST API

Функціональне тестування інтерфейсу взаємодії виконувалося за допомогою консольної утиліти curl при запущеному локальному вебсервері (python run.py). Усі програмні ендпоінти перевірялися на коректність повернення HTTP-статусів, структурування JSON-відповідей та обробку виключних ситуацій. Результати тестування наведено в таблиці 3.2.

Таблиця 3.2 − Результати функціонального тестування REST API

Тест	Ендпоінт	Очікуваний результат	Фактичний результат	Статус
Отримати список пристроїв	GET /devices/api/list	200 JSON, 5 пристроїв	200 JSON, 5 пристроїв	Пройдено
Зведення ризиків	GET /api/risks/summary	200 JSON, critical:3, high:1, low:1	200 JSON, critical:3, high:1, low:1	Пройдено
Список вразливостей	GET /api/risks	200 JSON, ≥12 записів	200 JSON, 12 записів	Пройдено
Деталі пристрою (Cisco router)	GET /devices/api/list	risk_level: critical, score: 10.0	risk_level: critical, score: 10.0	Пройдено

Тест Ендпоінт Очікуваний

результат	Фактичний результат	Статус	Деталі
пристрою (Linux server)	GET /devices/api/list	risk_level: low, score: 0.0	risk_level: low, score: 0.0
Пройдено			
Запуск сканування (mock-режим)	POST /api/scan	200 JSON, status:done	200 JSON, status:done
Пройдено			
Імпорт пристроїв (Node-RED)	POST /api/devices/import	200 JSON, status:ok, imported:5	200 JSON, status:ok, imported:5
Пройдено			
Журнал сканувань	GET /api/scan/history	200 JSON, масив записів	200 JSON, масив записів
Пройдено			
Позначити вразливість вирішеною	POST /api/risks/1/resolve	200 JSON, status: resolved	200 JSON, status: resolved
Пройдено			
Завантаження CSV-звіту	GET /reports/csv	200, Content-Type: text/csv	200, Content-Type: text/csv
Пройдено			
Завантаження PDF-звіту	GET /reports/pdf	200, Content-Type: application/pdf	200, Content-Type: application/pdf

Пройдено
Невірний JSON
при імпорті
POST
/api/devices/impo
rt
400 JSON, error
message
400 JSON, error
message
Пройдено
Усі 12 тестових випадків пройдені успішно (PASS). Результати curl-запитів, отримані під час тестування, підтверджують коректну роботу API. Зокрема, ендпоінт GET /devices/api/list повертає JSON-масив із п'яти об'єктів, кожен з яких містить правильно розраховані поля risk_level та risk_score: cisco-router-gw (critical, 10.0), cisco-sw-core (critical, 10.0), win-server-01 (high, 8.2), linux-web-srv (low, 0.0), mikrotik-ap (critical, 9.4).

3.6.2 Тестування модуля оцінки ризиків

Для перевірки коректності алгоритму оцінки ризиків проведено тестування кожного з восьми правил на відповідних пристроях. Тест-кейси охоплюють як позитивні сценарії (вразливість виявлена), так і негативні (вразливість відсутня):

Таблиця 3.3 − Тестування правил оцінки ризиків

No	Правило	Пристрій	Умова	CVSS очік.	CVSS факт.	Результат
1	Telnet замість SSH	cisco-router-gw	telnet_enabled=true, порт 23	7.5	7.5	Пройдено
2	SNMP v1/v2c	cisco-sw-core	snmp_version='v1'	6.5	6.5	Пройдено
3	Слабка спільнота SNMP	cisco-router-gw	snmp_community='public'	5.5	5.5	Пройдено
4	HTTP без HTTPS	cisco-router-gw	http=true, https=false	5.0	5.0	Пройдено
5	FTP відкритий	win-server-01	ftp_enabled=true, порт 21	6.0	6.0	Пройдено
6	Порт RDP (3389)	win-server-01	порт 3389 у open_ports	7.0	7.0	Пройдено
7	Відсутні вразливості	linux-web-srv	SSH, HTTPS, без Telnet/FTP	0.0	0.0	Пройдено
8	Порт					

Winbox
(8291)
mikrotik-ap порт 8291 у open_ports 7.0 7.0 Пройдено
3.6.3 Інтеграційне тестування взаємодії Node-RED та Flask
Інтеграційне тестування проводилося у повному стеку: Node-RED (localhost:1880) та Flask (localhost:5000) запущені одночасно. Тестовий сценарій включав ручний запуск flow через вузол inject, передачу даних до Flask API та перевірку оновлення реєстру пристроїв у веб-інтерфейсі.
За результатами тестування підтверджено успішну передачу JSON-пакета з п'ятьма пристроями через POST /api/devices/import (HTTP 200, status: ok, imported: 5). Після імпорту Node-RED автоматично запитав зведення ризиків (GET /api/risks/summary), отримавши відповідь {critical: 3, high: 1, low: 1, total_vulnerabilities: 12}. Вузол «Аналіз зведення ризиків» відобразив червону індикацію з текстом «⚠ КРИТИЧНО: 3 пристроїв з критичними ризиками!».

Таб Dashboard Node-RED підтвердив живе оновлення кожні 10 секунд: debug-панель відображала актуальні дані з часовими мітками, що оновлювалися в реальному часі (рис. 3.10).
Рисунок 3.10 − Візуалізація інтеграційного потоку та результати моніторингу в debug-панелі середовища Node-RED
3.6.4 Тестування на імітаційній мережі (Cisco Packet Tracer)
Для верифікації розроблених алгоритмів у реальних сценаріях створено імітаційну модель інфраструктури в середовищі Cisco Packet Tracer, яка відтворює топологію тестових пристроїв. Мережа побудована за схемою «зірка» з концентратором на комутаторі Cisco 2960 (рис. 3.11).

Рисунок 3.11 − Схема імітаційної моделі мережевої інфраструктури в середовищі Cisco Packet Tracer
Усі пристрої знаходяться в одній підмережі 192.168.1.0/24 (рис. 3.12). Шлюз за замовчуванням − маршрутизатор Cisco 2911.
Рисунок 3.12 – Результати конфігурування мережевих пристроїв у середовищі Cisco Packet Tracer
Мета тестування − переконатися, що система коректно виявляє навмисно внесені вразливості через Node-RED або прямий SNMP-запит, а результати аналізу в Flask відповідають очікуванням.

Топологія мережі:
− Cisco 2811 Router (192.168.1.1) - шлюз за замовчуванням;
− Cisco 2960 Switch (192.168.1.2) - ядро мережі;
− Windows Server 2019 (192.168.1.3) - файловий сервер;
− Ubuntu Server 22.04 (192.168.1.4) - веб-сервер;
− MikroTik RouterOS (192.168.1.10) - точка доступу.
Перед безпосереднім запуском процесів інвентаризації було виконано перевірку базової мережевої доступності між компонентами імітаційної моделі за допомогою утиліти ping з консолі віртуального сервера (рис. 3.13).
Рисунок 3.13 − Результати верифікації мережевої доступності хостів (команда ping) в імітаційній моделі
Стабільний обмін пакетами з нульовим відсотком втрат (0% loss) підтверджує коректність налаштування IP-адресації та комутації в Packet Tracer, що є обов'язковою умовою для подальшої симуляції SNMP-опитування».

На віртуальному маршрутизаторі через консоль введено команди, що створюють уразливості для перевірки системою: transport input telnet (активація незашифрованого протоколу Telnet) та snmp-server community public ro (використання слабкої строки автентифікації). Ці ж параметри відображені в наборі тестових даних інформаційної системи, що доводить відповідність моделі логіці аналітичного модуля.

Конфігурування занижених параметрів безпеки мережевого обладнання здійснювалося безпосередньо через інтерфейс командного рядка Cisco CLI (рис. 3.14) [27].

Рисунок 3.14 − Визначення версії програмного забезпечення та апаратної платформи маршрутизатора для подальшого аналізу відомих вразливостей

Для забезпечення чистоти викладу основного тексту роботи, повний перелік використаних команд налаштування ліній керування, а також зріз отриманого конфігураційного файлу пристрою Router1 винесено в Додаток В.

Демонстраційний стенд на етапі захисту передбачає спільну роботу веббекенду Flask, топології Cisco Packet Tracer та потоку Node-RED Flow. При ініціалізації ручного сканування Node-RED агрегує дані та транслює їх до Flask API. Там автоматично розраховується кумулятивний ризик із миттєвим відображенням критичних попереджень на вебдашборді, що підтверджує повний життєвий цикл роботи системи.

Маршрутизатор Cisco 2911 (192.168.1.1) – вразливий.

Вразливі конфігурації (навмисно внесені):

− Router: Telnet увімкнено, SNMP v2c community «public»;
− Switch: HTTP без HTTPS, слабкий пароль enable;
− Windows Server: FTP без шифрування, RDP порт 3389 відкритий.

Перелік вразливостей наведено в табл. 3.4.

Таблиця 3.4 − Параметри тестових мережевих пристроїв та навмисно налаштовані вразливості

Пристрій Модель IP-адреса Навмисні вразливості CVSS
Router 1 (GW) Cisco 2911 192.168.1.1 Telnet, SNMP v2c, HTTP 7.5 / 6.5 / 5.0
Switch 1 (Core) Cisco 2960 192.168.1.2 Telnet, SNMP v1 7.5 / 6.5
Server 1 (Windows) Generic Server 192.168.1.3 FTP, RDP 6.0 / 7.0
Server 2 (Ubuntu) Generic Server 192.168.1.4 Немає (безпечний) 0.0
Router 2 (MikroTik AP) Cisco 1941 192.168.1.10 HTTP, SNMP v2c 5.0 / 6.5

Примітка: емуляція MikroTik виконана на платформі Cisco 1941, оскільки Cisco Packet Tracer не містить нативної підтримки RouterOS.

Результати інтеграції:

− Node-RED flow зчитує конфігурації через SNMP.

− Flask API порівнює з еталоном безпеки.
− Виявлено 12 вразливостей (Critical: 3, High: 4, Medium: 5).
− Час сканування всієї топології: 2.3 секунди.

3.7 Аналіз результатів тестування

За результатами комплексного тестування системи підтверджено виконання всіх восьми функціональних вимог, визначених у розділі 1 (табл. 1.3, 1.5).

Усі тест-кейси функціонального тестування пройдені зі статусом «виконано».

Усі 8 правил оцінки ризиків демонструють очікувані CVSS-бали.

Таблиця 3.5 − Відповідність реалізації функціональним вимогам

ID вимоги	Вимога (скорочено)	Реалізовано у	Статус
FR-01	Реєстр пристроїв з усіма атрибутами	models.py: Device	Виконано
FR-02	Збереження конфігурацій з версіями	models.py: Configuration	Виконано
FR-03	Автоматичне виявлення вразливостей (8 правил)	risk_engine.py: analyze_device()	Виконано
FR-04	Розрахунок рівня ризику CVSS	risk_engine.py: calculate_risk_score()	Виконано
FR-05	Текстові рекомендації для вразливостей	risk_engine.py: поле recommendation	Виконано
FR-06	Дашборд з Chart.js графіками	templates/dashboard.html	Виконано
FR-07	Експорт звітів CSV та PDF	routes/reports.py	Виконано
FR-08	Інтеграція з Node-RED через REST API	routes/scan.py: /api/devices/import	Виконано

Інтеграція Node-RED з Flask підтверджена повним циклом передачі та аналізу даних. Система коректно обробляє як штатні сценарії (імпорт 5 пристроїв,

генерація звітів), так і граничні випадки (некоректний JSON, відсутній Nmap − перемикання на mock-режим).

Час обробки одного пристрою в mock-режимі становить менше 0,1 секунди, що значно перевищує нефункціональну вимогу ≤ 2 с. Система стабільно обробляє 5 тестових пристроїв та генерує 12 записів вразливостей, демонструючи готовність до масштабування до 100 пристроїв відповідно до нефункціональних вимог.

3.8 Інструкція з розгортання та запуску системи

Для запуску системи було виконано такі кроки:

1. Встановлено Python 3.11+ (https://python.org) та Node.js 18+ (https://nodejs.org);
2. Встановлено залежності Python: pip install -r requirements.txt (Flask, Flask-SQLAlchemy, python-nmap, reportlab, Werkzeug);
3. Ініціалізовано базу даних: python init_db.py. Створено файл inventory.db із п'ятьма тестовими пристроями;
4. Проведено запуск Flask-сервера: python run.py. Сервер запускається на http://localhost:5000 (рис.3.7);
5. Встановлено Node-RED глобально: npm install -g node-red. Проведено запуск командою node-red.
6. Відкрито Node-RED (http://localhost:1880), імпортовано файл nodered_flow.json через меню Import, Deploy (рис. 3.15);
7. Веб-інтерфейс системи: http://localhost:5000 − дашборд відображає стан мережі;
8. Для перевірки інтеграції: у вкладці Network Inventory Node-RED запущено «Ручний запуск сканування». Дані з'являються у реєстрі пристроїв Flask.

Таблиця 3.6 − Системні вимоги для розгортання

Компонент	Мінімальні вимоги	Рекомендовані вимоги
ОС	Windows 10 / Ubuntu 20.04 / macOS 11	Windows 11 / Ubuntu 22.04
Python	3.8+	3.11+
Node.js	16+	18+ LTS
RAM	4 ГБ	8 ГБ
Вільне місце	500 МБ	1 ГБ
Браузер	Chrome/Firefox/Edge (остання -2 версії)	Chrome 120+
Nmap	Не обов'язково (є mock-режим)	Nmap 7.9+ для реального скану

У третьому розділі БР реалізовано та протестовано ІС інвентаризації й аналізу конфігурацій мережевих пристроїв. Програмну систему (14 Python-файлів, 4 HTML-шаблони) організовано за архітектурою Application Factory з використанням Blueprint-ів. Створено модуль оцінки ризиків із 8 правилами перевірки та алгоритмом розрахунку CVSS (максимум 10.0), а також двохрежимний сканер (реальний Nmap і mock-режим). Для інтеграції з Node-RED розроблено API-ендпоінт із автоматичним запуском аналізу ризиків. Також створено веб-інтерфейс: інформаційну панель на базі Chart.js, реєстр пристроїв, сторінку деталей із вразливостями та експорт звітів у CSV і PDF.
Проведено комплексне тестування: 12 тест-кейсів API та 8 тест-кейсів правил ризиків успішно пройдені (100% PASS). Виконано інтеграційне тестування Node-RED і Flask та емуляцію в Cisco Packet Tracer. Підтверджено відповідність системи всім 8 функціональним вимогам (FR-01–FR-08). Тестування повного циклу − від ініціювання сканування в Node-RED до візуалізації критичних вразливостей − підтвердило працездатність рішення та його придатність для автоматизації моніторингу кібербезпеки.

ВИСНОВКИ

Дана випускна кваліфікаційна бакалаврська робота виконана на тему розробки веб-орієнтованої інформаційної системи інвентаризації та аналізу конфігурації мережевих пристроїв з автоматизованою оцінкою ризиків безпеки. Мета роботи досягнута в повному обсязі.

У процесі виконання КБР отримано наступні результати:

1. Аналіз предметної області підтвердив актуальність автоматизації конфігураційного аудиту. Визначено п'ять ключових проблем: фрагментована інвентаризація, суб'єктивна оцінка ризиків, відсутність історичного контексту, надмірне навантаження на персонал та відсутність засобів візуалізації.
2. Порівняльний аналіз семи аналогів (Zabbix, PRTG, SolarWinds NCM, Tenable Nessus, Cisco DNA Center, NetBox, Ansible/AWX) показав, що жодне рішення не поєднує безкоштовну модульну архітектуру, CVSS v3.1 та візуальну оркестрацію збору даних, що обґрунтовує доцільність власної розробки.
3. Функціональне моделювання виконано в нотаціях IDEF0 (контекстна діаграма A-0 та декомпозиція A1 − A4), DFD Гейна-Сарсона (рівні 0 та 1), UML Use Case та IDEF3 WorkFlow. Сформовано глосарій із 16 стрілок IDEF0.
4. Архітектура системи − трирівнева клієнт-серверна: рівень збору даних (Node-RED), рівень обробки та зберігання (Flask, SQLAlchemy, SQLite), рівень представлення (HTML5, Bootstrap 5, Chart.js). Забезпечено принцип Separation of Concerns.
5. Модель даних — нормалізована реляційна схема з п'яти таблиць (Device, Configuration, Vulnerability, RiskScore, ScanHistory). SQLAlchemy ізолює бізнес-логіку від СУБД, забезпечуючи міграцію на PostgreSQL/MySQL зміною рядка з'єднання.
6. Алгоритм оцінки ризиків (risk_engine.py) базується на восьми детермінованих правилах перевірки конфігурацій: Telnet, SNMP v1/v2c, HTTP без TLS, FTP, відкриті порти управління, слабкі SNMP-спільноти, застарілі ОС, відсутність автентифікації. Реалізовано кумулятивне оцінювання з понижувальним коефіцієнтом 0,7 та максимумом 10,0 балів за CVSS v3.1, що

відповідає ISO/IEC 27005.
7. REST API містить дванадцять ендпоінтів. Ключовий POST /api/devices/import реалізує upsert-логіку з автоматичним запуском конвеєра аналізу ризиків після кожної транзакції.
8. Node-RED Flow реалізує три вкладки: Network Inventory (ручний збір конфігурацій), Scheduler (автоматичні сканування о 08:00 та 20:00 + щогодинна перевірка доступності), Dashboard (live-моніторинг з оновленням кожні 10 секунд та колірною індикацією рівнів загрози).
9. Тестування виконано в три етапи: функціональне тестування REST API (12 тест-кейсів, 100% PASS), тестування модуля оцінки ризиків (8 правил, 100% відповідність CVSS), інтеграційне тестування Node-RED + Flask та демонстрація на імітаційній мережі в Cisco Packet Tracer. Підтверджено виконання всіх функціональних вимог FR-01 – FR-08.
10. Продуктивність: час обробки одного пристрою в mock-режимі − <0,1с (у 20 разів швидше за вимогу ≤2с). Повний цикл для 5 вузлів − 2,3с. Виявлено 12 вразливостей: Critical − 3, High − 4, Medium − 5.
Практична значущість підтверджується повним функціональним циклом: від ініціювання сканування в Node-RED − через аналіз risk_engine.py − до візуалізації критичних вразливостей та рекомендацій на веб-дашборді. Система є готовим прототипом для впровадження в навчальних лабораторіях і малих корпоративних мережах, а також навчальним інструментом для підготовки фахівців з кібербезпеки.
Перспективи подальшого розвитку: реалізація реального SNMP-опитування, інтеграція SSH-сесій, підтримка CVE-баз для автоматичного співставлення вразливостей, розширення аналітичного модуля методами машинного навчання для прогностичної оцінки ризиків.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1 The time between a vulnerability being discovered and its exploitation by attackers (Time to Exploit) [Електронний ресурс] // Cybersecurity Statistics. – Режим доступу: Cybersecurity Statistics (дата звернення: 01.06.2026).
2 Belal S. M. Securing OT: Understanding ISA/IEC 62443 and Configuration Management [Електронний ресурс] / S. M. Belal // Hexagon Asset Lifecycle Intelligence. – Режим доступу: https://aliresources.hexagon.com/cybersecurity/safeguarding-industrial-control-systems-understanding-isa-iec-62443-and-configuration-management (дата звернення: 01.06.2026).
3 NVD Vulnerability Metrics. CVSS v3.1 Official Support : National Vulnerability Database. URL: https://nvd.nist.gov/vuln-metrics/cvss (дата звернення: 03.05.2026).
4 Chimmanee K., Jantavongso S. Digital forensic of Maze ransomware: A case of electricity distributor enterprise in ASEAN. Expert Systems with Applications. 2024.
Vol. 249. P. 123652. URL: https://doi.org/10.1016/j.eswa.2024.123652
5 Shen X., Wang L., Li Z., Chen Y., Zhao W., Sun D., Wang J., Ruan W. PentestAgent: Incorporating LLM Agents to Automated Penetration Testing. arXiv,
2024. URL: https://doi.org/10.48550/ARXIV.2411.05185
6 Krupnova A. Legal regulation of information security in Ukraine. Analytical and Comparative Jurisprudence. 2023. No 5. P. 348–354. URL: https://doi.org/10.24144/2788-6018.2023.05.62
7 Tsiperman G. Design techniques for a seamless information system architecture. arXiv, 2021. URL: https://doi.org/10.48550/ARXIV.2102.10830
8 ISO/IEC 27001 Information security, cybersecurity and privacy protection — Information security management systems − Requirements. URL: iso.org
9 ДСТУ EN ISO/IEC 27002:2024 Інформаційна безпека, кібербезпека та захист конфіденційності. Заходи забезпечення інформаційної безпеки. Київ : ДП «УкрНДНЦ», 2024. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=106958.

10 Про основні засади забезпечення кібербезпеки України : Закон України від

05.10.2017 р. No 2163-VIII (зі змінами). URL:
https://zakon.rada.gov.ua/laws/show/2163-19.
11 Про стандартизацію : Закон України від 05.06.2014 р. No 1315-18 (зі змінами). URL: https://zakon.rada.gov.ua/go/1315-18.
12 B. S., NRK, S., J, T., & S, S. A Comparative Analysis of Vulnerability Management Tools: Evaluating Nessus, Acunetix, and Nikto for Risk Based Security Solutions. arXiv, 2024. URL: https://doi.org/10.48550/ARXIV.2411.19123.
13 Bilobrovets I. V. Network threat detection technology using Zabbix software. Modern Information Security. 2023. Vol. 54, No 2. URL:
https://doi.org/10.31673/2409-7292.2023.020003.
14 Monitoring Systems and Networks. Systems and Network Infrastructure Integration. Wiley, 2020. P. 157–171. URL:
https://doi.org/10.1002/9781119779964.ch9.
15 Network monitoring tools: enabling the management of network assets, ensuring high availability. International Journal of Development Research. 2020. P. 37020–37026. URL: https://doi.org/10.37118/ijdr.19224.06.2020.
16 Ariyadi T., Fikri M., Irwansyah I., Yudiastuti H. Penerapan monitoring jaringan dengan Zabbix pada PT. PLN (Persero) UIP bagian Sumbagsel. Jurnal Ilmiah Informatika. 2024. Vol. 12, No 02. P. 182 − 190. URL:
https://doi.org/10.33884/jif.v12i02.9283.
17 Silva A. dos S. S., Santos M. J. O. dos, Oliveira R. A. de. Estudo de gerenciamento de redes com Zabbix Server. Brazilian Journal of Technology. 2024. Vol. 7, No 4. URL: https://doi.org/10.38152/bjtv7n4-001.
18 Kadeeva E. N. Prospects for the development of monitoring in healthcare. Ekonomika i Upravlenie: Problemy, Resheniya. 2024. Vol. 10/13, No 151. P. 28 − 34. URL: https://doi.org/10.36871/ek.up.p.r.2024.10.13.004.
19 Alizan A. N., Zolkipli M. F. A Comparative Study Between Wireshark and Paessler Router Traffic Grapher (PRTG) in Network Monitoring and Analysis. INTI Journal. 2024. No 1. URL: https://doi.org/10.61453/intij.202451

20 Fathima A., Devi G. S. Enhancing university network management and security: a real-time monitoring, visualization & cyber attack detection approach using Paessler PRTG and Sophos Firewall. International Journal of System Assurance Engineering and Management. 2024. URL: https://doi.org/10.1007/s13198-024-02448-y
21 Feng-cheng X. The Research and Implementation for Next Generation Remote Network Configuration Management System. 2022.
22 Wenny R., Pamuji F. Y. Perbandingan evaluasi kerentanan menggunakan Tenable Nessus Scanner dan OWASP ZAP untuk meningkatkan keamanan sistem informasi kepegawaian di Universitas Merdeka Malang. Jurnal Ilmiah Universitas Batanghari Jambi. 2024. Vol. 24, No 3. P. 2451. URL:
https://doi.org/10.33087/jiubj.v24i3.5488
23 Qin T., Li C., Sun S., Liu G. A device information-centered accelerator control network management system. Radiation Detection Technology and Methods. 2024. Vol. 8, No 3. P. 1342–1358. URL: https://doi.org/10.1007/s41605-024-00459-8
24 The time between a vulnerability being discovered and its exploitation by attackers (Time to Exploit) [Електронний ресурс] // Cybersecurity Statistics. – Режим доступу: Cybersecurity Statistics (дата звернення: 04.06.2026).
25 Chart.js Documentation: офіційна документація. URL:
https://www.chartjs.org/docs/ (дата звернення: 04.05.2026).
26 Node-RED Documentation: офіційна документація. OpenJS Foundation, 2024. URL: https://nodered.org/docs/ (дата звернення: 01.05.2026).
27 FIRST.org. CVSS v3.1 Specification Document. Forum of Incident Response and Security Teams, 2019. URL: https://www.first.org/cvss/v3-1/specification-document (дата звернення: 03.05.2026).
28 Cisco Systems. IOS Software Configuration Guide. URL:
https://www.cisco.com/c/en/us/support/ios-nx-os-software/ios-15-2m-t/products-installation-and-configuration-guides-list.html (дата звернення: 05.05.2026).
29 Zabbix 7.0 LTS Documentation: офіційна документація. URL:
https://www.zabbix.com/documentation/current/en (дата звернення: 01.05.2026).

ДОДАТОК А
КАРТА ЗАЦІКАВЛЕНИХ СТОРІН ІНФОРМАЦІЙНОЇ СИСТЕМИ
Таблиця 1.1 − Карта зацікавлених сторін

Зацікавлена сторона	Категорія	Роль у системі	Інтереси та очікування	Вплив на ІС	Частота взаємодії	Взаємодія з системою
Адміністратор мережі	Основний користувач (Primary User)	Ініціатор сканувань, аналітик	Швидке виявлення пристроїв; Зручний веб-інтерфейс; Автоматизація рутинних перевірок; Оперативні сповіщення	Високий	Щоденно	Запускає сканування мережі; Переглядає реєстр пристроїв; Аналізує вразливості; Експортує звіти CSV/PDF; Налаштовує параметри сканування
Інженер з кібербезпеки	Основний користувач (Primary User)	Аудитор безпеки, аналітик ризиків	Точні CVSS-оцінки; Деталізація вразливостей; Відповідність конфігурацій стандартам безпеки; Рекомендації щодо усунення	Високий	Щотижня	Аналізує матрицю

ризиків;
Перевіряє
комплаєнс політик;
Формує
рекомендації;
Відстежує історію
змін конфігурацій;
Експортує звіти для
аудиту
Керівник IT-
відділу
Вторинний
користувач
(Secondary
User)
Приймач
рішень,
стратег
Зведена
аналітика стану
мережі;
PDF-звіти для
керівництва;
Обґрунтування
бюджету на
модернізацію;
KPI безпеки
інфраструктури
Середні
й
Щомісяця Переглядає
executive-summary;
Аналізує тренди
ризиків;
Приймає рішення
на основі
експортованих
даних;
Затверджує
політики безпеки
Node-RED
Оркестратор
Зовнішня
система
(External
System)
Автоматиз
ований
агент збору
даних
Стабільний
REST API;
Низька
затримка
відповідей;
Коректний
JSON-формат;
Підтримка cron-
тригерів;
Високи
й
Постійно
(кожні 10
сек)
Автоматично
збирає конфігурації;
Надсилає запити до

/api/devices/import;
Опитує
/api/risks/summary
Візуалізує статус у
реальному часі;

Зацікавлена
сторона
Категорія Роль
у системі
Інтереси та
очікування
Вплив
на ІС
Частота
взаємодії
Взаємодія з
системою
Можливість
live-
моніторингу
Запускає планові
сканування
Мережеві
пристрої
(Cisco,
MikroTik,
Servers, IoT)
Зовнішнє
середовище
(External
Environment)
Джерело
конфігурац
ійних
даних
Стандартні
протоколи
(SNMP/SSH/HT
TP)
Мінімальне
навантаження
на CPU під час
опитування
Безпека
передачі даних
Середні
й
Постійно Надають дані про
відкриті порти
Передають версії
ПЗ та параметри
конфігурації
Відповідають на
SNMP-запити
Надають банери
SSH/Telnet
Nmap Сканер Зовнішня
система
(External
System)
Інструмент
активного
сканування
Доступ до
мережі
Коректні права

доступу
Актуальна
версія Nmap
Можливість
сканування
підмереж
Середні
й
За
розкладом/
вручну
Сканирує підмережі
на наявність
активних хостів
Визначає відкриті
порти та сервіси
Передає сирі дані
до Flask API
Ідентифікує ОС
пристроїв
Розробник /
Технічна
підтримка
Внутрішній
стейкхолдер
(Internal
Stakeholder)
Підтримка
та розвиток
системи
Модульна
архітектура
Чистий код
Простота
додавання
нових правил
ризику
Документація
API
Можливість
масштабування
Середні
й
За
потребою
Налагоджує
систему
Розширює
risk_engine.py
Оновлює схеми БД
Керує розгортанням
Додає нові функції

ДОДАТОК Б
ЛІСТИНГ ПРОГРАМНОГО КОДУ МОДУЛЯ ОЦІНКИ РИЗИКІВ
БЕЗПЕКИ (RISK_ENGINE.PY)

```
def analyze_device(device):
 vulnerabilities = []
 open_ports = [int(p) for p in (device.open_ports or '').split(',') if
p.strip()]
 # Правило 1: Telnet замість SSH (CVSS 7.5 — High)
 if device.telnet_enabled or 23 in open_ports:
 vulnerabilities.append({
 'name': 'Використання Telnet замість SSH',
 'severity': 'high',
 'cvss_score': 7.5,
 'recommendation': 'Вимкніть Telnet (no service telnet).
```

```
Використовуйте лише SSH v2 (ip ssh version 2).'
 })
 # Правило 2: SNMP v1/v2c без шифрування (CVSS 6.5 — Medium)
 if device.snmp_version in ['v1', 'v2c']:
 vulnerabilities.append({
 'name': f'SNMP без шифрування ({device.snmp_version})',
 'severity': 'medium',
 'cvss_score': 6.5,
 'recommendation': 'Оновіть до SNMPv3 з аутентифікацією та
шифруванням.'
 })
 # Правило 3: Слабка спільнота SNMP (CVSS 5.5 — Medium)
 if device.snmp_community in ['public', 'private']:
 vulnerabilities.append({
   'name': f'Стандартна спільнота SNMP:
{device.snmp_community}',
 'severity': 'medium',
 'cvss_score': 5.5,
 'recommendation': 'Змініть community на унікальне значення
(мін. 16 символів).'
 })
 # Правило 4: HTTP без HTTPS (CVSS 5.0 — Medium)
 if device.http_enabled and not device.https_enabled:
 vulnerabilities.append({
 'name': 'Незашифрований HTTP (порт 80)',
 'severity': 'medium',
 'cvss_score': 5.0,
 'recommendation': 'Вимкніть HTTP, увімкніть HTTPS з валідним
сертифікатом.'
 })
 # Правило 5: FTP без шифрування (CVSS 6.0 — Medium)
 if device.ftp_enabled or 21 in open_ports:
 vulnerabilities.append({
 'name': 'FTP без шифрування (порт 21)',

 'severity': 'medium',
 'cvss_score': 6.0,
 'recommendation': 'Вимкніть FTP, використовуйте SFTP або
FTPS.'
 })
 # Правило 6: Відкриті порти управління (CVSS 7.0 — High)
 mgmt_ports = {8291: 'Winbox (MikroTik)', 3389: 'RDP', 161: 'SNMP'}
 for port, svc in mgmt_ports.items():
 if port in open_ports:
 vulnerabilities.append({
 'name': f'Відкритий порт управління {port} ({svc})',
 'severity': 'high',
 'cvss_score': 7.0,
 'recommendation': f'Обмежте доступ до порту {port} через
ACL.'
   })
 return vulnerabilities
def calculate_risk_score(vulnerabilities):
 if not vulnerabilities:
 return {'score': 0.0, 'risk_level': 'low'}
   sorted_vulns = sorted(vulnerabilities, key=lambda x: x['cvss_score'],
reverse=True)
 score = sorted_vulns[0]['cvss_score']
 for vuln in sorted_vulns[1:]:
 score += vuln['cvss_score'] * 0.7
 score = min(score, 10.0)
 if score &gt;= 9.0:
   risk_level = 'critical'
 elif score &gt;= 7.0:
   risk_level = 'high'
 elif score &gt;= 4.0:
   risk_level = 'medium'
```

```
 else:
 risk_level = 'low'
 return {'score': round(score, 1), 'risk_level': risk_level}
@scan_bp.route('/api/scan', methods=['POST'])
def api_scan():
 data = request.get_json() or {}
 subnet = data.get('subnet', '192.168.1.0/24')
 try:
 nm = nmap.PortScanner()
 nm.scan(hosts=subnet, arguments='-sn -T4')
 devices_found = [{'ip': host} for host in nm.all_hosts()]
 except Exception:
 devices_found = get_mock_devices()
 new_count = 0
 for dev_data in devices_found:

   existing =
Device.query.filter_by(ip_address=dev_data['ip']).first()
 if not existing:
 new_count += 1
   save_and_analyze(dev_data)
 else:
 existing.last_seen = datetime.utcnow()
 db.session.commit()
 log_scan(subnet, len(devices_found), new_count)
 return jsonify({
 'status': 'done',
 'devices_found': len(devices_found),
 'new_devices': new_count
 })
@scan_bp.route('/api/devices/import', methods=['POST'])
def import_devices():
 data = request.get_json()
 if not data or 'devices' not in data:
   return jsonify({'status': 'error', 'message': 'No devices
data'}), 400
 imported = 0
 for dev_data in data['devices']:
   device =
Device.query.filter_by(ip_address=dev_data['ip_address']).first()
 if not device:
 device = Device(**dev_data)
   db.session.add(device)
 else:
 for k, v in dev_data.items():
   setattr(device, k, v)
 device.last_seen = datetime.utcnow()
 db.session.commit()
 vulns = analyze_device(device)
 score_data = calculate_risk_score(vulns)
 save_vulnerabilities(device.id, vulns)
 save_risk_score(device.id, score_data)
 imported += 1
 return jsonify({'status': 'ok', 'imported': imported})
```

ДОДАТОК В

НАЛАШТУВАННЯ ВРАЗЛИВОСТЕЙ ПРИСРОЇВ В CISCO PACET TRACER

Конфігурація Cisco Router 1 (192.168.1.1)

```cisco
! Cisco Router Configuration
! Вразливості: Telnet, SNMP v2c, HTTP без HTTPS
enable
configure terminal
! Hostname
hostname cisco-router-gw
! Interface GigabitEthernet0/0
interface GigabitEthernet0/0
```

```
 ip address 192.168.1.1 255.255.255.0
 no shutdown
 exit
! Telnet Configuration (ВРАЗЛИВІСТЬ!)
line vty 0 4
 password cisco123
 login
 transport input telnet ssh
 exit
! SSH Configuration (безпечніше, але Telnet також увімкнено)
ip domain-name network.local
crypto key generate rsa modulus 1024
username admin privilege 15 secret admin123
! HTTP Server (ВРАЗЛИВІСТЬ - без HTTPS!)
ip http server
no ip http secure-server
! SNMP Configuration (ВРАЗЛИВІСТЬ - v2c з public community!)

 snmp-server community public RO snmp-server community private RW snmp-server  enable traps
! Default Route
ip route 0.0.0.0 0.0.0.0 192.168.1.1
end
write memory
```

Перевірка налаштованих уразливостей на Router 1

```
Router# show running-config Building configuration... Current configuration : 731 bytes ! version 15.1 no service timestamps log datetime msec no service timestamps debug datetime msec no service password-encryption ! hostname Router ! ! ! ! ! ! ! ! ip cef no ipv6 cef ! ! ! ! license udi pid CISCO1941/K9 sn FTX1524 OM0Q
!
!
!
!
!
!
!
!
!
!
!
 spanning-tree mode pvst ! ! ! ! ! ! interface GigabitEthernet0/0 ip address  192.168.1.10 255.255.255.0
duplex auto
speed auto
!
interface GigabitEthernet0/1
no ip address
duplex auto
speed auto
shutdown
!
interface Vlan1
no ip address
shutdown
!
ip classless
ip  route 0.0.0.0 0.0.0.0 192.168.1.1
!
ip flow-export version 9
!
!
!
!
!
!
snmp-server community public RO
!
line con 0
!
```

line aux 0
!
line vty 0 4
password cisco
login
transport input telnet
!
!
!
end
Конфігурація Router 2 / MikroTik (192.168.1.10)
```cisco
! Cisco Router (імітація MikroTik)
! Вразливості: HTTP без HTTPS, SNMP v2c
enable
configure terminal
hostname mikrotik-ap
! Interface
interface GigabitEthernet0/0 ip address 192.168.1.10 255.255.255.0
 no shutdown

 exit
ip default-gateway 192.168.1.1
! HTTP API (ВРАЗЛИВІСТЬ)
ip http server
ip http port 80
! SSH
ip domain-name mikrotik.local
crypto key generate rsa modulus 2048
line vty 0 4
 transport input ssh
 login local
 exit
! SNMP (ВРАЗЛИВІСТЬ - v2c)
snmp-server community public RO
snmp-server enable traps
end
write memory
```

БІБЛІОГРАФІЧНА ДОВІДКА
Тема бакалаврської роботи: «Розроблення інформаційної системи інвентаризації та аналізу конфігурації мережевих пристроїв з оцінкою ризиків безпеки»
Загальний пояснювальної записки: 95 сторінок
Перелік графічного матеріалу:
КБР.ІСТ – 07.00.00.000Е1 – Контекстна діаграма А-0 (IDEF0)− 1 аркуш
КБР.ІСТ – 07.00.00.000Е2 − Декомпозиція А0 (рівень А1–А4) (IDEF0) − 1 аркуш.
КБР.ІСТ – 07.00.00.000Е3 − DFD Рівень 1 — основні процеси обробки даних− 1 аркуш
КБР.ІСТ – 07.00.00.000Е4 – Діаграма варіантів використання системи інвентаризації (UML Use Case)− 1 аркуш
КБР.ІСТ – 07 .00.00.000Е5 – Діаграма послідовності сценарію ручного сканування мережі (UML Sequence)− 1 аркуш
КБР.ІСТ – 07 .00.00.000Е6 – Діаграма діяльності алгоритму оцінки ризиків безпеки (UML)
КБР.ІСТ – 07 .00.00.000Е7 – Діаграма діяльності алгоритму оцінки ризиків безпеки (UML)
КБР.ІСТ – 07 .00.00.000Е10 – Результати верифікації мережевої доступності хостів (команда ping) в імітаційній моделі
Дата завершення бакалаврської роботи 13.06.2026 року
Студент-бакалавр Олег ГРИЦАК