Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі»

Виконав: здобувач 2-го курсу, групи 2КІТС-24м
спеціальності 125– Кібербезпека та захист інформації
Освітня програма – Кібербезпека інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)
Кудревич В.І.
(прізвище та ініціали)

Керівник: д. ф., доц. каф. МБІС
Салієва О.В.
(прізвище та ініціали)
«____» _____ 2025 р.

Опонент: проф. каф. ОТ
Захарченко Сергій Михайлович
(прізвище та ініціали)
«10» грудня 2025 р.

Допущено до захисту

Голова секції УБ кафедри МБІС
Юрій ЯРЕМЧУК
“10” грудня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека та захист інформації

Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ

Голова секції УБ, кафедра МБІС

Юрій ЯРЕМЧУК

"24" вересня 2025 р.

З А В Д А Н Н Я

на магістерську кваліфікаційну роботу студенту

Кудревичу В.І.

(прізвище, ім'я, по-батькові)

1. Тема роботи: «Удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі»

Керівник роботи д. ф., доц. каф. МБІС Салієва О.В.

затверджені наказом вищого навчального закладу від "24" вересня 2025 року

№ 313

2. Строк подання студентом роботи за тиждень до захисту

3. Вихідні дані до роботи: Електронні джерела, наукові статті, стандарти, які стосуються теми магістерської кваліфікаційної роботи.

4. Зміст текстової частини: У вступі даної роботи було проаналізовано актуальність теми, визначено основну мету, задачу, об'єкт, предмет та наукову новизну роботи. Перший розділ присвячений аналізу теоретичної складової роботи та пошуку проблематики. У другому розділі відбулась розробка покращеного алгоритму та було поставлено цілі для практичної реалізації. Було описано методи комбінації підходів та визначено як саме вони покращують захист. У третьому розділі було здійснено практичну реалізацію покращеного алгоритму. Четвертий розділ присвячений економічній частині, яка підтверджує доцільність покращення захисту протоколу шляхом аналізу витрачених ресурсів.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень):

- у першому розділі наведено 6 рис.;
- у другому розділі наведено 7 рис.;

- у третьому розділі наведено 18 рис..

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдан прийн
Основна частина			
I	МБІС Салієва О.В. д. ф., доц. каф.		
II	МБІС Салієва О.В. д. ф., доц. каф.		
III	МБІС Салієва О.В. д. ф., доц. каф.		
Економічна частина			
IV	доц. каф. ЕПВМ, к.т.н. Ратушняк Ольга Георгіївна,		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Приміт
1	Визначення напрямку магістерської роботи, формулювання теми	24.09.2025	30.09.2025	
2	Аналіз предметної області обраної теми	01.10.2025	08.10.2025	
3	Теоретичне покращення протоколу та постановка задачі для програмної реалізації	09.10.2025	25.10.2025	
4	Програмна реалізація	26.10.2025	31.10.2025	
5	Написання пояснювальної записки для МКР	01.11.2025	20.11.2025	
6	Передзахист МКР	21.11.2025	23.11.2025	
7	Виправлення та внесення правок в МКР	24.11.2025	07.12.2025	
8	Захист МКР	08.12.2025	11.12.2025	

Студент _____ Кудрявц

(підпис) (прізвище та ініц

Керівник роботи _____ Салієва О.В.

(підпис) (прізвище та ініц

АНОТАЦІЯ

Кудревич В.І. Удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі. Магістерська кваліфікаційна робота зі спеціальності 125 – кібербезпека та захист інформації, освітня програма – кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2025. 118с.

На укр. мові. Бібліогр.: 37 назв; рис. 24; табл. 5.

У даній роботі розглянуто питання покращення захисту криптографічного протоколу TLS за допомогою багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі. Проведено аналіз сучасних методів захисту інформаційних ресурсів, охарактеризовано особливості TLS-протоколу та визначено ключові проблеми, пов’язані з виявленням підозрілих сертифікатів під час передавання пакетів даних між користувачами.

Розроблено алгоритми динамічної перевірки пакетів та сертифікатів та адаптивної системи блокування передачі даних, що забезпечують виявлення та протидію несанкціонованим спробам доступу. На основі запропонованих підходів реалізовано програмну систему, що включає модулі навчання нейронної мережі, налаштування власної поведінкової моделі у разі виявлення загрози та систему моніторингу сертифікатів в режимі реального часу. Для зручності користувачів створено інтерактивний інтерфейс, який дозволяє ефективно взаємодіяти із системою.

Проведене тестування підтвердило працездатність реалізованого рішення, його відповідність сучасним вимогам безпеки та здатність витримувати високе навантаження. Результати роботи можуть бути використані для вдосконалення існуючих програмних рішень або створення нових систем із підвищеним рівнем захисту.

Ключові слова: кібербезпека, протокол, нейронна мережа, TLS-протокол.

ABSTRACT

Kudrevych V.I. Improving the TLS cryptographic protocol based on multi-level session management and dynamic certificate verification using a neural network. Master's qualification work in the specialty 125 - cybersecurity, educational program - cybersecurity of information technologies and systems. Vinnytsia: VNTU, 2025. 118c.

In Ukrainian. Bibliography: 37 titles; Fig.: 24; Table 5.

This work considers the issue of improving the protection of the TLS cryptographic protocol using multi-level session management and dynamic certificate verification using a neural network. An analysis of modern methods of protecting information resources was conducted, the features of the TLS protocol were characterized, and key problems associated with detecting suspicious certificates during the transmission of data packets between users were identified.

Algorithms for dynamic verification of packets and certificates and an adaptive data transmission blocking system were developed, which ensure the detection and counteraction to unauthorized access attempts. Based on the proposed approaches, a software system was implemented that includes neural network training modules, setting up its own behavioral model in case of threat detection, and a real-time certificate monitoring system. For the convenience of users, an interactive interface was created that allows for effective interaction with the system.

The testing confirmed the operability of the implemented solution, its compliance with modern security requirements, and the ability to withstand high loads. The results of the work can be used to improve existing software solutions or create new systems with an increased level of protection.

Keywords: cybersecurity, protocol, neural network, TLS-protocol.

ЗМІСТ

ВСТУП

Актуальність. У сучасному цифровому середовищі, де переважна більшість комунікацій, транзакцій та обміну даними здійснюється через мережу Інтернет, особливого значення набуває питання безпеки передавання інформації. Протокол Transport Layer Security є основним стандартом захищеної комунікації, який забезпечує шифрування, автентифікацію сторін та цілісність даних.

Однак, попри свою зрілість, TLS залишається вразливим до низки сучасних кіберзагроз — від підробки цифрових сертифікатів і компрометації центрів сертифікації до атак типу man-in-the-middle та отруєння кешу OCSC. Традиційні методи перевірки сертифікатів, такі як CRL чи OCSP, є статичними та не забезпечують гнучкої адаптації до нових типів ризиків.

Це створює потребу у впровадженні динамічних підходів до перевірки сертифікатів і TLS-сесій, здатних реагувати на аномалії у режимі реального часу. Одним із перспективних напрямів є поєднання криптографічних механізмів із методами машинного навчання, що дозволяє формувати моделі поведінки безпечних і потенційно шкідливих з’єднань.

Актуальність дослідження зумовлена необхідністю створення інтелектуальної системи контролю TLS, яка поєднує криптографічну надійність із гнучкістю адаптивного аналізу, забезпечуючи підвищений рівень захисту інформаційних обмінів у мережевих застосунках.

Мета. Метою даної магістерської роботи є розробка та впровадження удосконаленого підходу до захисту TLS-з’єднань, який поєднує багаторівневе управління сесіями, динамічну перевірку сертифікатів та інтелектуальне оцінювання ризику з використанням нейронної мережі.

Задачами дослідження є:

– аналіз існуючих загроз і вразливостей у роботі протоколу TLS;

– дослідження методів динамічної перевірки сертифікатів і систем управління сесіями;

– проєктування архітектури middleware-рівня для контролю TLS-з’єднань;

– розробка алгоритму оцінювання ризику TLS-сесій із використанням нейронної мережі;

– реалізація програмного прототипу із логуванням сесій, системою політик і модулем сповіщень;

– тестування роботи системи на прикладах різних сценаріїв ризику;

– оцінка ефективності запропонованого підходу та його придатність до масштабування;

Об’єктом дослідження є процес захищеного встановлення та підтримки TLS-сесій між клієнтом і сервером.

Предметом дослідження є методи вдосконалення криптографічного протоколу TLS за рахунок динамічного управління сесіями, перевірки сертифікатів і оцінки ризику з використанням нейронних мереж.

Наукова новизна роботи полягає у вдосконаленні криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі.

Практична цінність полягає у створенні робочого прототипу, який може бути інтегрований у вебсервіси чи корпоративні мережі як модуль моніторингу безпеки TLS-з’єднань. Розроблене рішення забезпечує підвищення рівня безпеки шляхом автоматичного виявлення ризикових сертифікатів і аномальних сесій, зменшує вплив людського фактора та може використовуватися як основа для побудови адаптивних систем кіберзахисту нового покоління.

Апробація: тези доповідей представленні на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)» [1].

1 ЗАГАЛЬНИЙ АНАЛІЗ ТЕХНОЛОГІЙ ЗАХИСТУ ТА АВТЕНТИФІКАЦІЇ В ПРОТОКОЛІ TLS

У даному розділі здійснено аналіз існуючого теоретичного матеріалу у галузі захисту протоколу TLS, визначено переваги та недоліки існуючих методів його захисту із метою обґрунтування базового серед них для подальших досліджень.

Під час написання розділу було проаналізовано та визначено актуальність досліджень, визначено основні загрози та вразливості даного протоколу. Також, досліджено наявні методи перевірки сертифікатів задля визначення поточного стану захищеності системи. Було проаналізовано методи машинного навчання задля можливого інтегрування його в покращений протокол TLS.

1.1 Актуальність дослідження

У сучасному цифровому просторі більшість комунікацій між користувачами, сервісами та організаціями здійснюється через захищені протоколи передачі даних. Серед них провідне місце посідає Transport Layer Security — криптографічний протокол, що забезпечує конфіденційність, автентичність і цілісність переданої інформації. TLS є основою безпечного функціонування вебсайтів, мобільних застосунків, електронних платіжних систем, поштових сервісів і корпоративних комунікацій.

Разом із тим, з розвитком інформаційних технологій та розширенням цифрової інфраструктури зростає кількість атак, спрямованих на компрометацію або обходження TLS-захисту. Сучасні зловмисники дедалі частіше використовують складні комбіновані техніки, що включають підробку сертифікатів, зараження проміжних центрів сертифікації, перенаправлення трафіку, впровадження фальшивих OCSP-відповідей або атаки на механізми перевірки довіри. Це призводить до того, що навіть формально «захищене» з’єднання може бути використане для передачі шкідливих даних, фішингових кампаній чи втручання у внутрішню інфраструктуру компанії.

Традиційні підходи до забезпечення безпеки TLS базуються на статичних

правилах перевірки сертифікатів — перевірці підпису, ланцюга довіри, строку дії та статусу відкликання через CRL або OCSC. Проте ці механізми мають низку обмежень (рис. 1.1). Вони не дозволяють оперативно реагувати на нові види загроз, не враховують поведінкові характеристики клієнта або сервера, а також не забезпечують автоматичної адаптації до змін у мережевому середовищі.

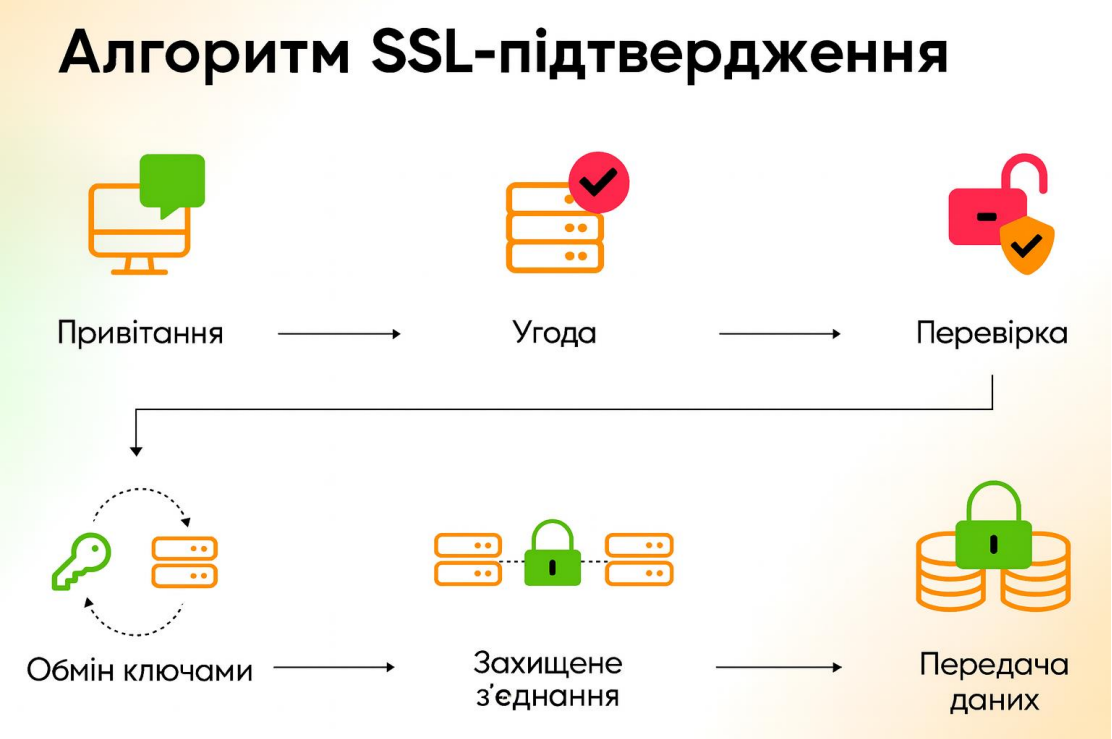


Рисунок 1.1 – Алгоритм SSL-підтвердження [2]

Водночас світова практика інформаційної безпеки активно переходить від реактивних до прогнозно-аналітичних моделей. Це означає, що замість фіксації факту порушення після його виникнення, системи безпеки повинні навчатися випереджати аномалії — виявляти підозрілу активність до того, як вона завдасть шкоди. У контексті TLS це означає можливість не лише перевіряти валідність сертифіката, але й оцінювати ризиковість самої сесії, враховуючи низку факторів: частоту підключень, поведінку клієнта, час активності, статистику ключів, тощо [3].

Для цього дедалі ширше застосовуються технології машинного навчання, які здатні аналізувати великі масиви TLS-метаданих і виявляти закономірності, недоступні для класичних алгоритмів. Зокрема, нейронні мережі особливо багатошарові персептрони або рекурентні архітектури можуть розпізнавати

нетипові моделі взаємодії клієнта і сервера, що часто свідчить про атаки або компрометацію сертифіката.

Важливою тенденцією останніх років є розвиток концепції багаторівневого управління сесіями, де рішення про дозволи, обмеження або блокування приймається не на основі одного параметра, а комплексно — з урахуванням ризику, політики безпеки, історії взаємодій і результатів ML-аналізу. Такий підхід дозволяє створити адаптивну систему безпеки, яка динамічно змінює реакцію залежно від контексту подій.

Крім того, в умовах масштабування мережевих систем виникає проблема керування довірою до сертифікатів у розподіленому середовищі. Велика кількість проміжних центрів сертифікації, автоматичні системи видачі сертифікатів як Let’s Encrypt та короткі строки їх дії створюють нові ризики для верифікації. У таких випадках нейронна модель може виступати додатковим рівнем контролю — аналізуючи статистичні та поведінкові ознаки, вона здатна виявляти підроблені або підозрілі сертифікати навіть тоді, коли формальні механізми TLS вважають їх дійсними.

Ще однією проблемою є відсутність уніфікованого журналу подій TLS, який би дозволяв здійснювати аудит і аналітику з урахуванням історії ризиків. Це знижує прозорість моніторингу, ускладнює навчання моделей і підвищує ймовірність повторних інцидентів без виявлення першопричин. Тому актуальним є створення інтегрованих рішень, що поєднують логування сесій, аналіз ризиків, систему сповіщень та механізми донавчання нейронної мережі на основі розмічених прикладів [4].

Загалом, світові тренди у сфері інформаційної безпеки вказують на перехід до інтелектуальних адаптивних протоколів, де криптографія не лише шифрує дані, а й динамічно управляє довірою. У цьому контексті TLS, як основний інтернет-стандарт, потребує розширення — не на рівні ядра протоколу, а шляхом створення middleware-рішень, що інтегрують криптографічну основу з аналітичними методами машинного навчання.

Таким чином, дослідження напрямку інтеграції нейронних мереж у процес

TLS-аутентифікації та управління сесіями є надзвичайно актуальним і має практичну значимість. Розробка удосконаленого TLS-рівня з багаторівневим контролем сесій, динамічною перевіркою сертифікатів та адаптивним прийняттям рішень дозволить суттєво підвищити рівень безпеки інформаційних систем, зменшити кількість успішних атак і створити базу для розвитку інтелектуальних мережевих протоколів нового покоління [5].

1.2 Аналіз загроз та вразливостей TLS-обміну

Попри широке застосування протоколу TLS у всіх сучасних мережевих сервісах, його використання не гарантує абсолютного рівня захисту. Основні механізми TLS — автентифікація, шифрування та перевірка цілісності — ефективні лише за умови коректного налаштування та надійності всіх складових криптографічної інфраструктури. На практиці вразливості часто виникають через помилки конфігурації, компрометацію сертифікатів, використання застарілих алгоритмів або зловмисні дії, спрямовані на обхід перевірки автентичності [6].

Однією з найпоширеніших загроз є атаки типу man-in-the-middle , під час яких зловмисник перехоплює або модифікує трафік між клієнтом і сервером. Якщо користувач під’єднується до підробленого сервера або приймає фальшивий сертифікат, TLS-шифрування втрачає сенс, оскільки зловмисник фактично отримує доступ до всієї переданої інформації. Такі атаки особливо ефективні у відкритих Wi-Fi-мережах або при використанні підроблених DNS-записів (рис. 1.2).

Рисунок 1.2 – Приклад атаки «Людина посередині» [7]

Суттєвим ризиком є підробка або компрометація центрів сертифікації. Якщо зловмисник отримує доступ до приватного ключа CA, він може створювати сертифікати, які виглядають цілком дійсними для будь-якої системи перевірки. Такі інциденти вже неодноразово фіксувалися — зокрема, компрометація DigiNotar, Comodo, Symantec, що призводила до масового підписання фейкових сертифікатів для популярних доменів [8].

Іншим поширеним методом компрометації є атака SSL stripping, яка полягає у примусовому переведенні з’єднання з HTTPS на HTTP (рис. 1.3). Зловмисник перехоплює початковий запит користувача, змінює його на незахищений і таким чином позбавляє клієнта TLS-захисту, зберігаючи ілюзію безпечного з’єднання.

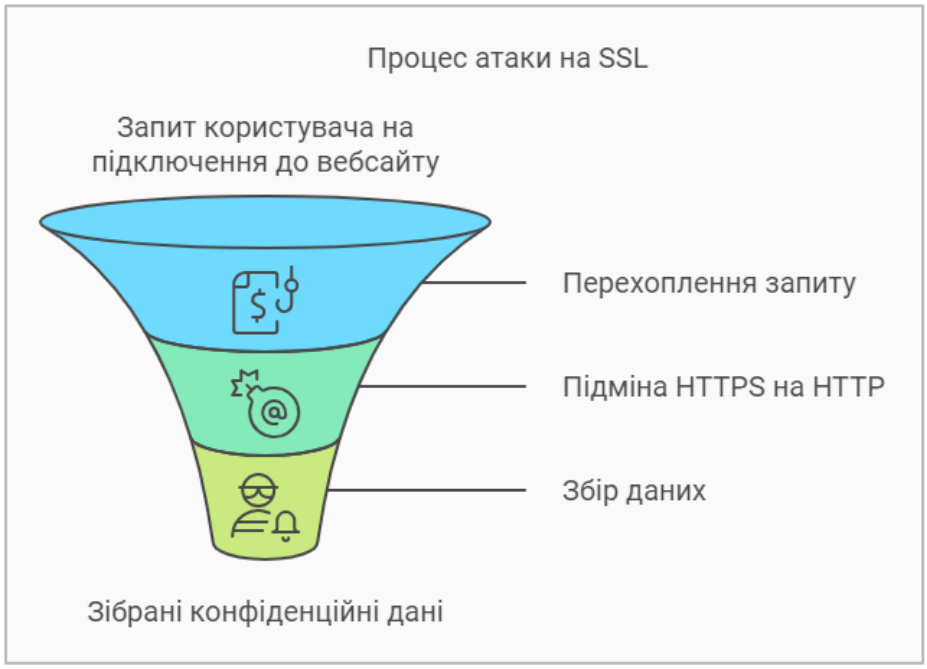


Рисунок 1.3 – Приклад атаки SSL stripping [9]

Ще одна група вразливостей пов’язана з самопідписаними та простроченими сертифікатами, які часто використовуються у внутрішніх мережах або тестових середовищах. Через відсутність централізованої перевірки довіри такі сертифікати можуть легко стати інструментом атак, особливо якщо користувачі звикають ігнорувати попередження браузера про їх невідповідність.

Окремо слід виділити загрози, що виникають через використання застарілих версій протоколу та слабких алгоритмів шифрування. Застарілі реалізації TLS 1.0 і 1.1 мають відомі вразливості BEAST, POODLE, Lucky13, а використання симетричних шифрів з короткими ключами або небезпечних режимів робить можливим часткове відновлення вмісту повідомлень [10].

Додаткову небезпеку становлять атаки на повторне використання сесійних ключів session resumption. Якщо ключі або параметри відновлення сесій не оновлюються належним чином, це відкриває шлях до компрометації великої кількості з’єднань після успішного злому однієї сесії. Також відомі атаки на механізми renegotiation у TLS 1.2, які дозволяли вставляти власні команди у вже встановлений канал зв’язку (рис. 1.4).

Рисунок 1.4 – Приклад роботи захисту session resumption [11]

Сучасні загрози дедалі частіше мають комбінований характер, поєднуючи технічні й поведінкові вектори. Наприклад, зловмисники можуть підміняти сертифікати через легітимні API, але лише у певних часових проміжках або для окремих груп клієнтів. Такі сценарії складно виявити традиційними методами, оскільки вони не порушують формальних правил TLS.

Окрім технічних уразливостей, існують організаційні ризики, пов’язані з неналежним управлінням сертифікатами: невчасне оновлення, відсутність централізованого моніторингу, помилки у конфігурації серверів. У великих інфраструктурах ці фактори можуть призвести до масових інцидентів навіть без безпосереднього злому.

Додатково варто звернути увагу на загрози, пов’язані з автоматизованим випуском сертифікатів. Сучасні служби, такі як Let’s Encrypt або ZeroSSL, значно спростили процес отримання сертифіката, що позитивно вплинуло на поширення HTTPS, однак призвело до нових викликів. Автоматичне видання сертифікатів без належної верифікації власника домену дає змогу зловмисникам оперативно отримувати легітимні сертифікати для фішингових або тимчасових сайтів. Унаслідок цього зростає кількість шкідливих вебресурсів, які технічно виглядають безпечними, бо мають дійсний TLS-сертифікат [12].

Ще однією проблемою є складність виявлення підозрілих або нетипових шаблонів TLS-трафіку. Наприклад, ботнети, шкідливі скрипти або програми

віддаленого доступу часто використовують TLS для приховування командного обміну з керуючим сервером. Формально такий трафік цілком відповідає стандартам безпеки, проте аналіз метаданих кількість з'єднань, тривалість, частота повторних запитів, може свідчити про аномальну активність. Класичні методи TLS-верифікації не передбачають поведінкового аналізу, тому виявлення таких патернів можливе лише за допомогою технологій машинного навчання.

Також слід зазначити, що вразливості можуть виникати не лише у самому протоколі TLS, а й у його реалізаціях. Різні бібліотеки — OpenSSL, GnuTLS, LibreSSL, BoringSSL — мають власні архітектурні особливості та історію критичних помилок. Помилки в реалізації обробки сертифікатів, перевірки підпису або генерації випадкових чисел уже неодноразово призводили до компрометації систем, навіть якщо специфікація TLS формально залишалася безпечною.

Окрему увагу необхідно приділити людському фактору. Користувачі часто ігнорують попередження браузера про недійсні сертифікати або погоджуються на встановлення винятків, що фактично знімає захист TLS у конкретному з'єднанні. Такі дії створюють сприятливе середовище для реалізації атак MITM, навіть без зламу самої криптографічної схеми.

Ще одна тенденція останніх років — зростання кількості атак, що використовують легітимні сертифікати для шкідливих цілей. Наприклад, деякі зловмисники навмисно отримують сертифікати на домени, схожі на популярні бренди, після чого запускають фішингові сайти з дійсним HTTPS-з'єднанням. Користувач, вважає сайт безпечним і вводить свої облікові дані [13]. Таким чином, навіть найсучасніші криптографічні механізми не захищають від соціально-технічних прийомів.

В умовах постійної еволюції кіберзагроз стає очевидним, що традиційні механізми перевірки сертифікатів і сесій у TLS не здатні забезпечити достатній рівень гнучкості та адаптивності. Потрібні рішення, які не просто перевіряють цифрові підписи, а й аналізують контекст і поведінку сесій у реальному часі.

Саме тому дослідження напрямів, що поєднують криптографічні методи з технологіями машинного навчання, набуває особливої актуальності [14].

Усе це свідчить, що система безпеки TLS потребує нового рівня контролю — аналітичного, поведінкового та самонавчального. Лише поєднання криптографічних перевірок із методами машинного аналізу дозволить своєчасно виявляти приховані загрози, розпізнавати нетипову активність і запобігати зловживанням на етапі встановлення з’єднання [15].

1.3 Аналіз існуючих підходів до перевірки сертифікатів

Перевірка цифрових сертифікатів є ключовим елементом у забезпеченні безпеки протоколу TLS, адже саме від достовірності сертифіката залежить, чи може клієнт довіряти серверу. Механізми перевірки сертифікатів еволюціонували разом із розвитком інфраструктури відкритих ключів (PKI), однак навіть сучасні рішення мають низку обмежень, що знижують їхню ефективність у динамічних мережевих середовищах [16].

Перевірка ланцюга довіри. Основним етапом автентифікації в TLS є побудова та перевірка ланцюга довіри — від сертифіката сервера до кореневого центру сертифікації Клієнт перевіряє, чи кожен проміжний сертифікат підписаний наступним у ланцюгу, і чи кореневий сертифікат присутній у локальному сховищі довірених центрів.

Попри простоту, цей механізм має вразливість: якщо один із проміжних центрів сертифікації буде скомпрометований, то зловмисник може створювати дійсні сертифікати для будь-яких доменів. Такий сценарій уже неодноразово мав місце в історії (DigiNotar, Comodo, Symantec), і стандартна перевірка ланцюга не дозволяє виявити підробку без зовнішніх джерел інформації [17].

Списки відкликаних сертифікатів — це перший історичний механізм, який дозволяє дізнатися, чи був сертифікат відкликаний його емітентом. CA періодично публікує список сертифікатів, які більше не вважаються дійсними.

Недоліком CRL є їхня статичність і затримка оновлення: у великих інфраструктурах списки можуть досягати мегабайт розміру, що робить їх

незручними для частого завантаження. Крім того, якщо клієнт не має доступу до оновленого CRL, він не може перевірити актуальність сертифіката, що створює вікно вразливості.

Протокол онлайн-перевірки статусу сертифіката. Більш гнучким рішенням є Online Certificate Status Protocol, який дозволяє клієнту у реальному часі звертатися до сервера CA для перевірки статусу сертифіката.

Хоча OCSP значно зменшує затримку між відкликанням і виявленням, він має критичний недолік — залежність від доступності сервера CA. Якщо OCSP-сервер недоступний, багато клієнтів просто пропускають перевірку soft-fail, залишаючи потенційно небезпечні з'єднання активними. Крім того, самі OCSP-відповіді можуть бути підроблені або кешовані зловмисниками.

Для вирішення цих проблем запроваджено OCSP Stapling, коли сервер сам додає останню відповідь OCSP у процес TLS-рукопотискання. Це зменшує навантаження на CA та пришвидшує перевірку, однак і цей підхід не вирішує питання виявлення складних аномалій чи фальсифікованих сертифікатів.

Технологія certificate pinning дозволяє застосункам зберігати відомі сертифікати або їхні відбитки і відхиляти всі, що не збігаються. Це ефективний спосіб захисту від MITM-атак, але він має низьку гнучкість. Якщо CA змінює сертифікат або відбувається легітимна ротація ключів, старий пінгований сертифікат стане недійсним, що призводить до відмови у з'єднанні навіть без загрози безпеці [18].

Браузери, такі як Chrome, поступово відмовились від static pinning, оскільки неправильна конфігурація може заблокувати доступ користувачів до сайтів на глобальному рівні.

Сучасним напрямом розвитку є система Certificate Transparency (CT) — публічний журнал, у який кожен CA зобов'язаний записувати видані сертифікати. Кожен запис підписується та зберігається у відкритій базі даних, що дозволяє будь-кому перевірити факт видачі сертифіката [19].

Перевага CT полягає у підвищенні прозорості: власники доменів можуть відстежувати появу нових сертифікатів і виявляти підробки. Проте CT не надає

механізму оцінки ризику або автоматичного блокування — це лише журнал, а не система прийняття рішень.

Незважаючи на різноманітність методів перевірки, жоден із них не забезпечує динамічного аналізу ризику. CRL і OCSP реагують із затримкою, pinning не пристосований до змін, а CT працює лише як архів. Усі вони базуються на формальній перевірці валідності, але не враховують поведінкові ознаки з'єднання, частоту використання сертифіката, контекст клієнта чи історію попередніх інцидентів [20].

Така ситуація призводить до того, що легітимний за формою сертифікат може бути зловмисним за суттю, а системи TLS не мають інструментів для виявлення таких випадків. Саме тому дослідження напрямів інтеграції технологій машинного навчання у процес перевірки сертифікатів є логічним і перспективним продовженням еволюції TLS-захисту. Моделі, що навчаються на історичних даних, можуть автоматично розпізнавати підозрілі сертифікати та формувати оцінку ризику у режимі реального часу, що значно підвищує стійкість мережевих систем до сучасних атак [21].

Додатково варто зазначити, що більшість сучасних методів перевірки сертифікатів орієнтовані на бінарний результат — сертифікат визнається або дійсним, або недійсним. Такий підхід не враховує проміжних станів, наприклад «потенційно ризикований», «аномальний», «нетиповий». У реальному середовищі такі випадки становлять більшість, оскільки сертифікат може формально бути дійсним, але використовуватися у підозрілих контекстах — наприклад, при частому перепідключенні, у нічний час або на невідомих доменах. Тому виникає потреба у гнучкій системі оцінювання ризику, що базується не лише на формальних полях сертифіката, а й на поведінкових та контекстних даних.

Класичні механізми TLS не передбачають аналізу поведінкових закономірностей, таких як час встановлення з'єднання, тип пристрою, геолокація клієнта, частота повторних рукопотискань, використання нестандартних параметрів шифрування тощо. Проте саме такі ознаки часто дозволяють виявити

приховані атаки — наприклад, автоматизовані скрипти, масове створення сесій або підміну сертифіката у вузькому часовому вікні. Відсутність механізмів адаптації до таких сценаріїв робить традиційну TLS-верифікацію надто жорсткою і негнучкою для сучасних мережевих реалій.

Крім технічних аспектів, значну роль відіграють людські та організаційні фактори. У багатьох компаніях сертифікати видаються і продовжуються вручну, що призводить до помилок і несвоєчасного оновлення. Застарілі або дубльовані сертифікати нерідко залишаються активними навіть після зміни інфраструктури. Це створює додатковий ризик для зловмисників, які можуть використати старі або нескасовані сертифікати для обходу захисту [22].

Сучасна тенденція до автоматизації за допомогою сервісів на кшталт Let’s Encrypt вирішує проблему зручності, але водночас породжує нові ризики. Автоматичне випускання сертифікатів без перевірки репутації власника домену дозволяє зловмисникам отримувати повністю дійсні сертифікати для фішингових або тимчасових ресурсів.

Крім того, стандартні методи перевірки не враховують історію використання сертифіката. Навіть якщо сертифікат не відкликаний і належить до довіреного центру, його раптова поява в нових доменах або регіонах може свідчити про компрометацію. Виявлення таких аномалій можливе лише за умови аналізу великої кількості даних, що зберігаються у журналах CT, серверах OCSP або внутрішніх системах моніторингу.

Тому нині формується підхід, який поєднує криптографічну перевірку з аналітичними моделями ризику. Його мета — не лише визначити факт дійсності сертифіката, а й оцінити його довіру у контексті поведінки, історії, частоти використання та зв’язків з іншими сертифікатами. Такий метод відкриває можливість до створення самонавчальних систем, які адаптуються до нових типів загроз і здатні виявляти підробки, що не мають формальних ознак зламу [23].

Отже, сучасні методи перевірки сертифікатів потребують розвитку у напрямку динамічної, інтелектуальної перевірки, що базується на аналізі даних і

алгоритмах машинного навчання. Саме інтеграція нейронних мереж у процес TLS-аутентифікації може стати наступним етапом еволюції мережевої безпеки — від реактивного до проактивного підходу, де система не просто виявляє підробку, а прогнозує ймовірність загрози ще до її реалізації [24].

1.4 Методи інтелектуального оцінювання ризиків у мережевих протоколах

З розвитком кіберзагроз традиційні методи перевірки сертифікатів та управління сесіями виявилися недостатніми для забезпечення проактивного захисту мережевих з'єднань. Більшість класичних підходів спирається на статичні правила або таблиці відповідності, які ефективні лише у передбачуваних сценаріях. Однак сучасні атаки характеризуються високим рівнем динамічності, автоматизації та адаптації до контрзаходів. У таких умовах виникла потреба у інтелектуальних системах аналізу ризику, здатних самостійно виявляти нові типи аномалій на основі поведінкових і контекстних даних [25].

Спочатку контроль безпеки мережевих протоколів обмежувався перевіркою криптографічних атрибутів — довжини ключів, версії протоколу, хеш-функцій, наявності сертифіката у списках відкликаних. Така перевірка давала змогу виявляти лише відомі загрози. Поступово з розвитком аналітики з'явилися системи, які почали враховувати поведінкові характеристики трафіку — частоту запитів, розмір пакетів, часові закономірності тощо. Це стало першим кроком до переходу від формальної до інтелектуальної оцінки ризику [26].

Сучасні методи базуються на моделях машинного навчання, які дозволяють автоматично виявляти невідомі загрози, аналізуючи великі обсяги мережевих даних. На відміну від сигнатурних підходів, ML-моделі не шукають конкретний шаблон атаки — вони вчаться розпізнавати відхилення від нормальної поведінки. Саме ця властивість робить їх ефективними у випадках нових або модифікованих типів атак, які ще не внесені до баз даних загроз (рис. 1.5).

Рисунок 1.5 – Типи мереж і їх алгоритмів [27]

Існуючі інтелектуальні методи можна умовно поділити на три групи.

Моделі з навчанням із учителем — використовуються тоді, коли є набір попередньо розмічених прикладів. До таких підходів належать логістична регресія, дерева рішень, метод опорних векторів, а також штучні нейронні мережі.

Перевага таких моделей — висока точність при наявності якісних даних, однак вони залежать від постійного оновлення навчальної вибірки.

Моделі без учителя — застосовуються для пошуку відхилень без попереднього маркування даних. Найпоширеніші методи: K-Means, DBSCAN, Isolation Forest, Autoencoder. Такі системи здатні самостійно виявляти нові аномалії, однак часто потребують додаткової ручної перевірки результатів.

Окремо виділяють гібридні підходи, де поєднуються статистичний аналіз, машинне навчання і правила безпеки. Це дозволяє створювати баланс між точністю і керованістю системи.

Нейронні мережі, особливо багаторівневі та рекурентні, показують високу ефективність у задачах класифікації мережевого трафіку та виявлення аномалій у TLS-сесіях [28].

Завдяки здатності працювати з неструктурованими або частково відсутніми даними, такі моделі можуть:

– оцінювати ризик кожної сесії за сукупністю ознак IP, порт, тривалість, cipher suite, fingerprint сертифіката, час доби;

– виявляти патерни, характерні для фішингових або ботнет-з’єднань;

– прогнозувати ймовірність компрометації сертифіката на основі історичних закономірностей;

Наприклад, при аналізі журналів TLS-з’єднань нейронна мережа може навчитися розпізнавати здоровий трафік і порівнювати нові сесії з відомими шаблонами. У разі виявлення відхилення від типових параметрів система може автоматично змінити політику.

Ключова перевага інтелектуальних систем полягає у динамічному прийнятті рішень. На відміну від класичних фільтрів, які працюють за жорсткими правилами, моделі машинного навчання оновлюють власні параметри у процесі експлуатації. Це дозволяє системі враховувати зміни у структурі мережевого трафіку, сезонні коливання, нові вектори атак тощо [29].

Додатково можливе періодичне донавчання — коли система накопичує нові приклади підозрілих або підтверджених інцидентів і використовує їх для поліпшення своєї точності. У контексті TLS це означає, що кожен новий скомпрометований сертифікат, виявлений вручну, може стати навчальним прикладом для майбутнього автоматичного виявлення подібних випадків.

Інтелектуальна оцінка ризику не обмежується лише аналітикою — вона безпосередньо впливає на процес прийняття рішень у системі. У межах багаторівневого управління TLS-сесіями результати моделі можуть використовуватись для динамічного застосування політик:

- Allow — звичайна робота при низькому ризику;
- Challenge — повторна перевірка сертифіката або користувача;

– Rekey — примусова генерація нового ключа сесії;
– Block — повне блокування з’єднання;

Такий підхід перетворює систему TLS із пасивного механізму шифрування на активну адаптивну платформу, яка самостійно реагує на ризики (рис. 1.6).

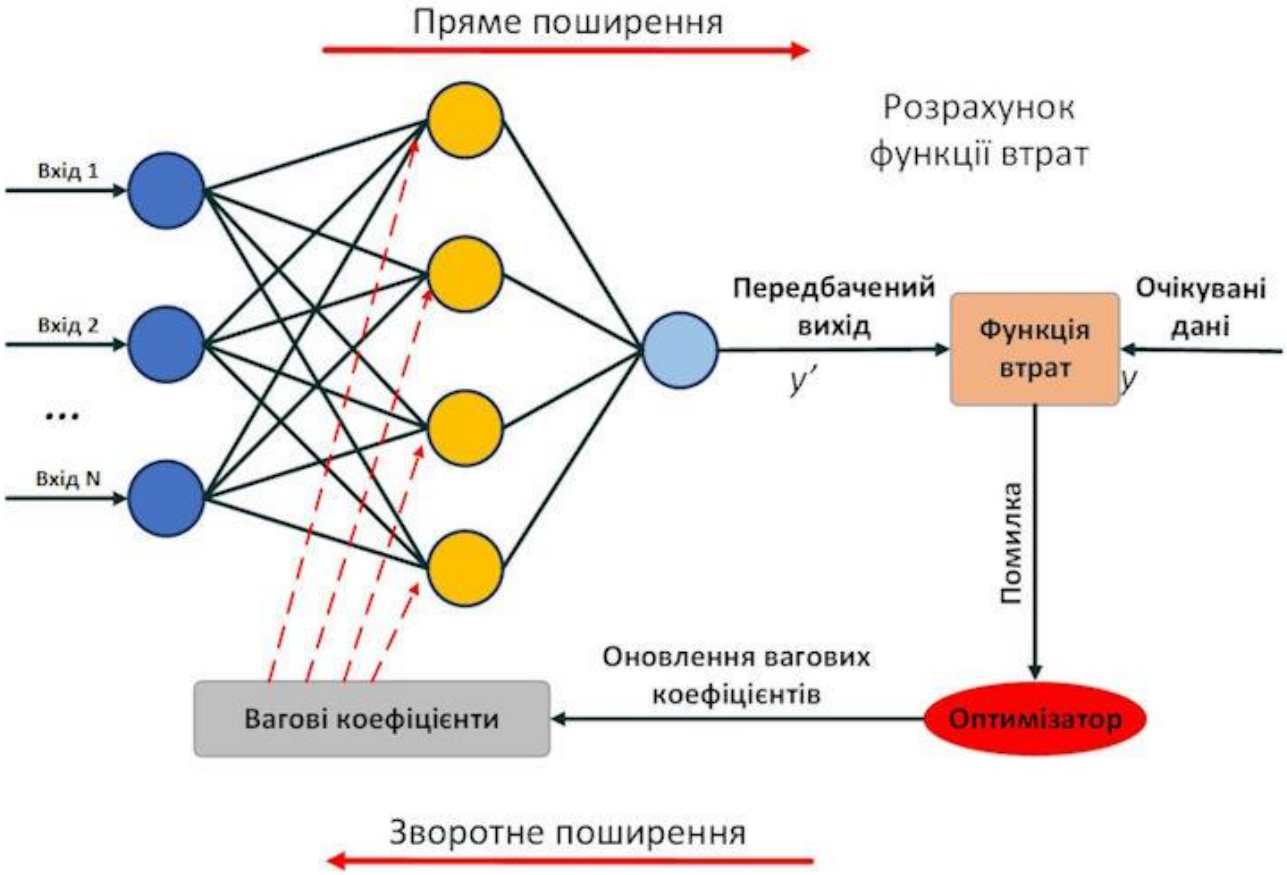


Рисунок 1.6 – Процес поширення даних [30]

Розвиток інтелектуальних систем оцінки ризику відкриває нові можливості для підвищення надійності та гнучкості мережевих протоколів. У поєднанні з криптографічними механізмами, нейронні мережі дозволяють створювати адаптивні механізми контролю, які не лише реагують на загрози, але й навчаються на їх основі. На відміну від класичних рішень, що базуються на сигнатурному підході, моделі штучного інтелекту здатні виявляти нові, раніше невідомі патерни атак, формуючи власне уявлення про нормальну та аномальну поведінку мережевого середовища.

Застосування таких систем особливо актуальне для транспортних протоколів безпеки, зокрема TLS, SSH, IPsec, де потоки даних є постійними та високошвидкісними. Інтелектуальні моделі можуть аналізувати не лише технічні

параметри з’єднання, а й контекстну інформацію — географію підключень, часові закономірності, історію сертифікатів, повторюваність сесій, характер обміну пакетами. Це дозволяє формувати глибший рівень розуміння поведінки системи, виявляти “сірі зони” у безпеці та мінімізувати кількість хибнопозитивних спрацьовувань.

Впровадження нейромережевих технологій у сферу TLS-аналізу може забезпечити перехід від реактивних до проактивних стратегій безпеки. Система здатна не лише фіксувати факт порушення, а й прогнозувати ймовірність його виникнення. Наприклад, аналізуючи історію використання певного сертифіката або частоту оновлень ключів, модель може виявити тенденцію до потенційної компрометації задовго до фактичного інциденту [31].

Окремим напрямом розвитку є створення самонавчальних моделей, які постійно оновлюють свої параметри на основі реальних спостережень. Це забезпечує адаптацію до нових сценаріїв загроз без необхідності ручного втручання. При поєднанні з технологіями журналів прозорості Certificate Transparency, системами моніторингу мережевих логів і внутрішніми базами даних організації така модель може діяти як єдиний центр аналітичного контролю ризиків.

Перспективним є також застосування інтелектуальних методів у системах управління політиками безпеки. Результати ML-аналізу можуть автоматично впливати на політики TLS — змінювати рівень довіри, блокувати або перевіряти сесії, активувати додаткові механізми захисту (наприклад, перегенерацію ключів або зміну cipher suite). У такий спосіб досягається саморегульований цикл безпеки, де рішення приймаються динамічно, відповідно до оціненого ризику.

Додатковою перевагою є можливість інтеграції інтелектуальних систем оцінки ризику з механізмами аудиту та комплаєнсу. Автоматичний збір, класифікація та візуалізація подій у реальному часі спрощують контроль відповідності міжнародним стандартам безпеки (ISO/IEC 27001, NIST SP 800-53, OWASP ASVS). Таким чином, інтелектуальні системи не лише підвищують

технічну стійкість, а й оптимізують процеси управління інформаційною безпекою на рівні організації.

Подібні рішення можуть стати основою глобальних мережевих платформ спільного навчання, де окремі системи обмінюються даними про нові аномалії, формуючи колективну модель кіберзахисту. Такий підхід відкриває шлях до створення екосистеми довіри нового покоління, у якій TLS виступатиме не лише транспортним, а й аналітичним протоколом, здатним до самостійного прийняття рішень і адаптації до ризиків [32].

1.5 Висновки та постановка задач

Отже, в даному розділі було проведено аналіз існуючих механізмів перевірки сертифікатів та управління TLS-з’єднаннями. Опрацьовані дані показали, що сучасна інфраструктура мережевої безпеки орієнтована переважно на статичні перевірки, які не враховують контекст, поведінку сесій і накопичений досвід системи. Стандартний протокол TLS забезпечує криптографічну цілісність, але не має вбудованих засобів інтелектуальної оцінки ризиків, тому не здатний оперативно реагувати на нові типи атак або підозрілі патерни у поведінці клієнтів.

В результаті проведення аналізу теоретичного матеріалу та виходячи з мети та актуальності теми, були поставлені наступні задачі подальшої роботи:

– здійснити вдосконалення обраного алгоритму задля підвищення його безпеки, використовуючи можливості нейронної мережі динамічно виявляти підозрілі сертифікати;

– практично реалізувати застосунок, в який інтегрувати розроблений вдосконалений протокол;

– протестувати програмну реалізацію та вдосконалений алгоритм;

– обґрунтувати економічну складову розробки та фінансову доцільність реалізації.

У межах цієї роботи ставиться завдання створення програмного комплексу, що дозволяє підвищити рівень безпеки TLS-обміну без втручання у базовий криптографічний стек протоколу.

Розроблене рішення реалізується у вигляді допоміжного програмного шару middleware, який розташовується на рівні застосунку і взаємодіє з TLS-з’єднаннями, аналізуючи їхню поведінку у реальному часі.

Мета розробки підвищити рівень захисту TLS-сесій шляхом інтелектуального аналізу сертифікатів і метаданих з’єднань, застосування нейронної мережі для оцінки ризику та впровадження багаторівневої системи управління сесіями з адаптивними політиками реагування.

2 УДОСКОНАЛЕННЯ TLS-ОБМІНУ НА ОСНОВІ БАГАТОРІВНЕВОГО КЕРУВАННЮ СЕСІЯМИ ТА ПЕРЕВІРКИ СЕРТИФІКАТІВ

У даному розділі описано процес удосконалення методу TLS-обміну на основі багаторівневого керуванню сесіями та перевірці сертифікатів. Задля цього у даному розділі відбудеться проєктування архітектури удосконаленого протоколу, та будуть описані використані системи для посилення захисту.

Також, в даному розділі описаний метод інтеграції нейронної мережі в алгоритм та описано який саме чином дана мережа покращує захист.

2.1 Проєктування архітектури удосконаленого TLS-middleware

Зважаючи на обмеження стандартного протоколу TLS, який зосереджується переважно на криптографічній цілісності даних, але не забезпечує адаптивного контролю ризиків, у межах цієї роботи запропоновано архітектуру проміжного програмного шару, що реалізує механізми інтелектуального управління сесіями та динамічної перевірки сертифікатів.

Метою розробленого рішення є створення розширення протоколу TLS на рівні застосунку, яке дозволяє проводити поведінковий аналіз з’єднань, оцінювати ризики компрометації сертифікатів, приймати адаптивні рішення відповідно до політик безпеки та зберігати результати для подальшого навчання нейронної моделі [33].

Архітектура системи побудована за принципом багаторівневої взаємодії. Кожен рівень відповідає за свій аспект роботи з TLS-сесіями — від збору технічних параметрів до прийняття рішень і генерації звітів.

Система складається з шести взаємопов’язаних рівнів. Рівень збору даних відповідає за перехоплення метаданих кожної TLS-сесії на рівні PHP-застосунку.

Зібрані параметри включають:

- ідентифікатор сесії;
- IP-адреси клієнта і сервера;
- версію протоколу;
- fingerprint сертифіката;

– часові мітки початку та завершення з’єднання.

Отримані дані зберігаються у таблиці sessions бази MySQL.

Рівень верифікації сертифікатів забезпечує первинну перевірку отриманих сертифікатів — перевіряє коректність ланцюга довіри, термін дії, видавця, тип підпису Signature Algorithm та унікальний відбиток.

Результати перевірки зберігаються у таблиці certificates. Сертифікати, що мають нестандартні або невідомі атрибути, позначаються статусом suspicious.

Рівень інтелектуального аналізу виконує динамічну оцінку ризику за допомогою зовнішнього ML-сервісу.

Нейронна модель обчислює ймовірність ризику та повертає класифікацію: low, medium, high, critical.

Отримані результати заносяться у поле risk_level таблиці sessions.

Рівень управління політиками реалізує логіку адаптивного реагування на основі оціненого ризику.

Політики зберігаються у таблиці policy_settings і визначають, яку дію застосувати до сесії:

– allow — звичайна робота сесії;
– challenge — повторна перевірка або верифікація;
– rekey — перегенерація ключів шифрування;
– block — припинення з’єднання;

Рішення системи може змінюватися динамічно, залежно від оновлення оцінки ризику або появи додаткових даних.

Всі події системи — створення сесії, зміна статусу, блокування, оцінка ризику, ручні дії користувача — фіксуються у таблиці audit_log.

Записи мають тип події, опис, рівень ризику та часову мітку. Це забезпечує повну прозорість функціонування системи та можливість відтворення будь-якої події у майбутньому.

Веб-інтерфейс розроблено на базі Bootstrap 5 і FontAwesome, із використанням RedBeanPHP для взаємодії з базою даних.

Інтерфейс забезпечує зручну візуалізацію даних, редагування політик, ручну розмітку сертифікатів і контроль дій системи у режимі реального часу (рис. 2.1).

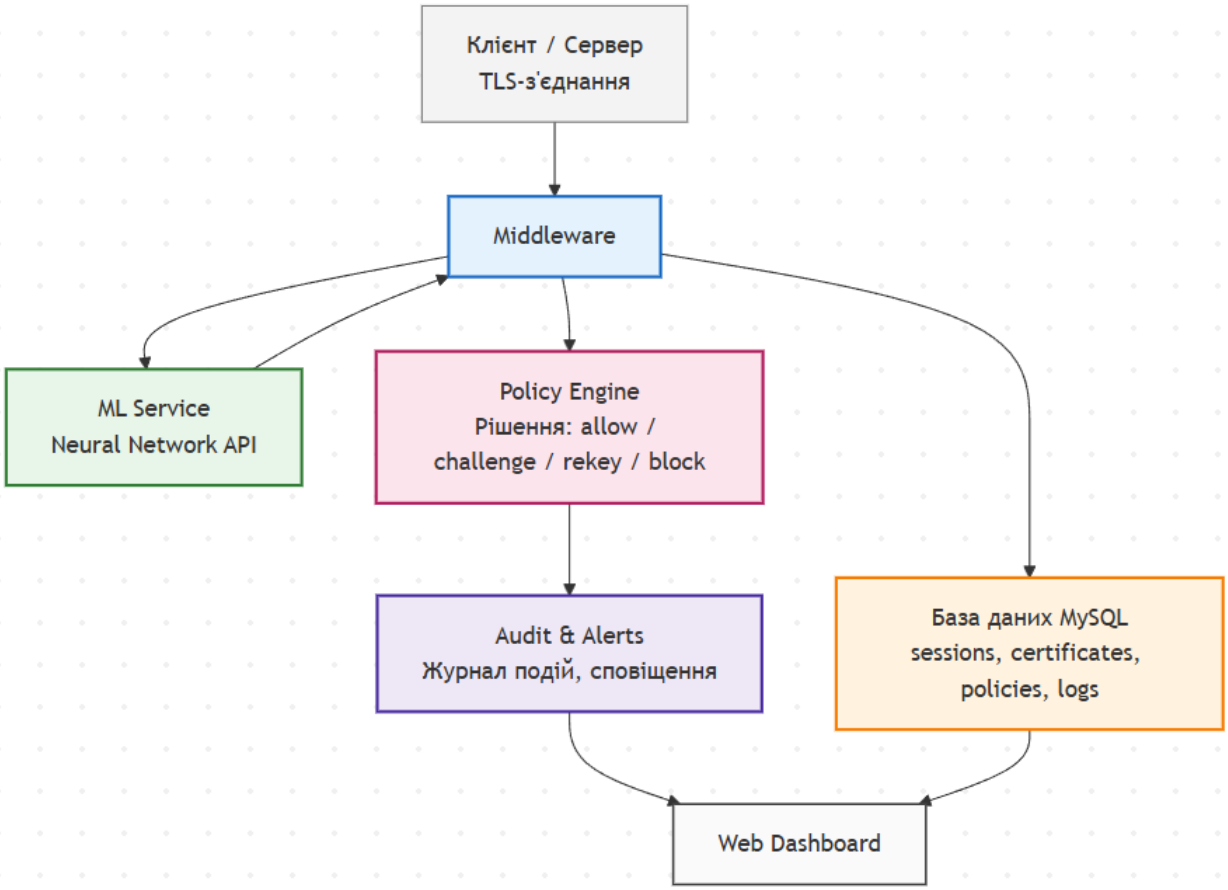


Рисунок 2.1 – Розроблена архітектура взаємодії

Взаємодія модулів побудована за моделлю коли дані передаються послідовно через рівні системи, а результати аналізу повертаються для корекції політик або повторного навчання моделі:

– при ініціації нового TLS-з’єднання middleware фіксує метадані сесії;

– сертифікат перевіряється базовими методами;

– дані передаються на ML-сервіс для оцінки ризику;

– на основі результату модель формує рекомендацію;

– Middleware застосовує відповідну політику та створює запис у журналі подій;

– підозрілі випадки автоматично заносяться до таблиці alerts, а їхні параметри використовуються під час наступного навчання моделі.

Таким чином формується замкнений цикл моніторингу, який поєднує збір, аналіз, реагування та донавчання.

Запропонована архітектура відрізняється від традиційних засобів безпеки TLS тим, що:

- не вимагає модифікації стандартного протоколу або SSL-бібліотек;
- дозволяє аналізувати поведінку сесій у динаміці;
- реалізує концепцію Policy-Driven Security, де рішення приймаються автоматично на основі ризику;
- підтримує адаптивне навчання моделі на основі накопичених даних;
- надає зручний веб-інтерфейс для перегляду, редагування та аналітики;

Таким чином, запропонований middleware виступає інтелектуальним надбудовним шаром над TLS, який перетворює протокол із пасивного механізму шифрування на активну систему контролю безпеки з можливістю самонавчання (див. додаток Б).

Розроблена система являє собою інтелектуальний проміжний рівень між користувачем і стандартним протоколом TLS. Її головна ідея — аналізувати та контролювати поведінку TLS-сесій у реальному часі, не змінюючи сам протокол. Коли користувач або застосунок встановлює TLS-з’єднання, middleware фіксує метадані і передає їх у базу даних MySQL [34].

Сертифікат проходить первинну перевірку і надсилається на ML-сервіс — нейронну модель, яка обчислює рівень ризику з’єднання. Після отримання результату модель передає у систему оцінку ризику. Залежно від цієї оцінки Policy Engine автоматично визначає дію:

- якщо ризик низький — сесію дозволено (allow);
- при середньому — ініціюється додаткова перевірка (challenge);
- при високому — оновлюються ключі шифрування (rekey);
- при критичному — сесію блокується (block).

Усі рішення, дії користувача та результати ML-аналізу записуються у журнал аудиту (рис.2.3). Підозрілі події створюють системні алерти, які відображаються у веб-інтерфейсі.

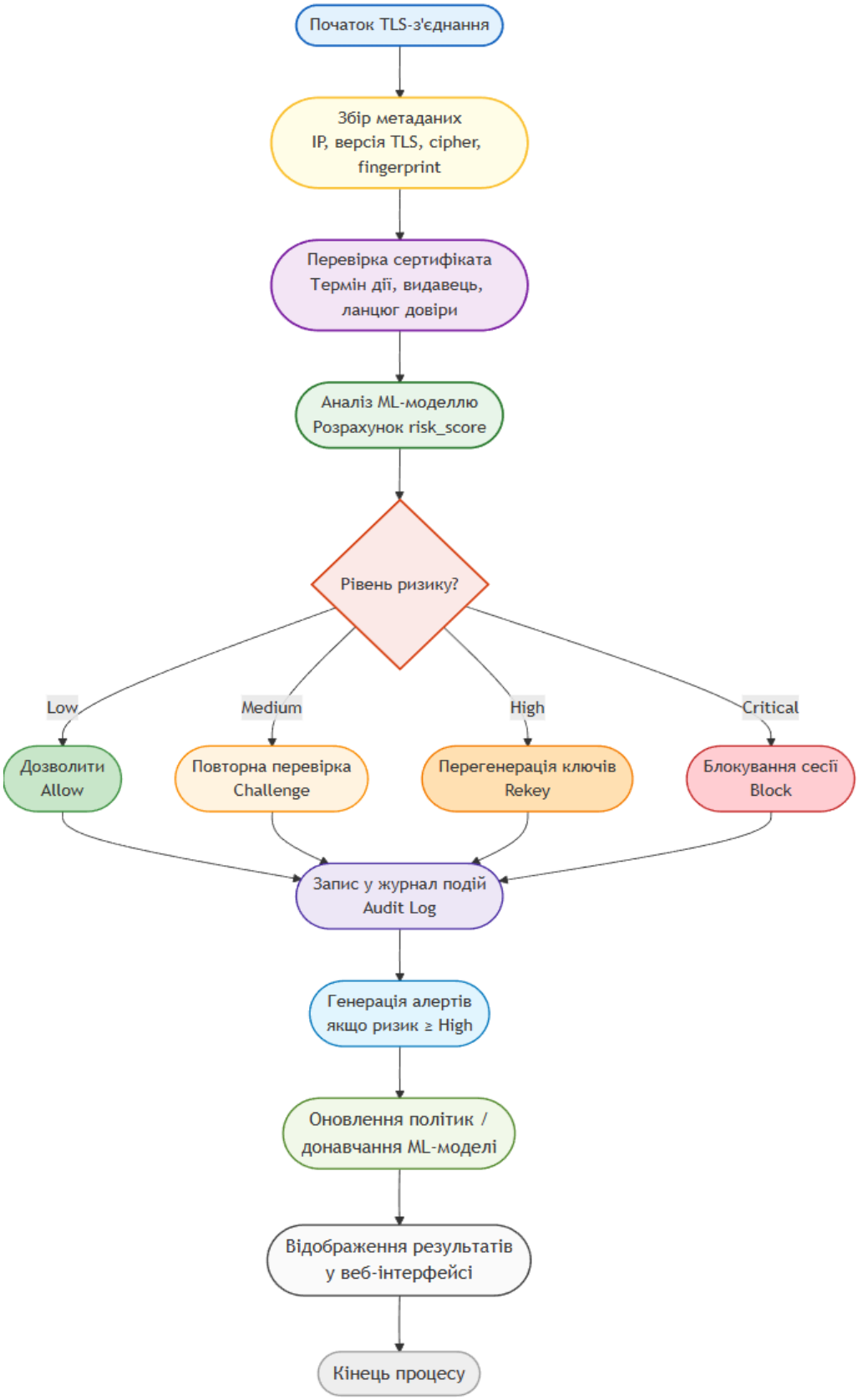


Рисунок 2.3 – Послідовність обробки TLS-з’єднання

Таким чином, система забезпечує повний цикл моніторингу TLS-обміну — від збору даних і динамічної перевірки сертифікатів до автоматичного прийняття рішень і самонавчання нейронної моделі.

2.2 Розробка алгоритму динамічної перевірки сертифікатів

У класичній архітектурі протоколу TLS процес перевірки сертифікатів має статичний характер: клієнт перевіряє дійсність підпису, термін чинності, ієрархію сертифікаційних центрів і, за можливості, статус відкликання через CRL або OCSC. Такий підхід є достатнім у більшості звичайних випадків, однак він не враховує контексту з'єднання, поведінкових характеристик клієнта чи сервера та історії використання самого сертифіката. В умовах сучасних динамічних мереж, де загрози можуть з'являтися непередбачувано, статичної перевірки виявляється недостатньо. Саме тому в межах розробленого TLS-middleware було створено алгоритм динамічної перевірки сертифікатів, який об'єднує класичні криптографічні методи з машинним навчанням і механізмом політик безпеки.

Основною ідеєю алгоритму є те, що перевірка сертифіката повинна відбуватися не лише на момент встановлення сесії, а й у процесі її життя. Таким чином, система постійно контролює параметри з'єднання, поведінку клієнта, час дії сертифіката та інші вторинні ознаки, які можуть свідчити про ризикову поведінку (рис. 2.4). Алгоритм реалізовано як програмний компонент на рівні застосунку, який взаємодіє з TLS-потоком, але не змінює його структури, що дозволяє інтегрувати його в будь-який веб- або серверний сервіс без втручання в базовий стек OpenSSL чи GnuTLS.

На першому етапі після завершення TLS-рукопотискання система зчитує метадані з'єднання — версію протоколу, набір шифрів, IP-адреси, часову мітку та цифровий відбиток сертифіката. Далі обчислюється унікальний ідентифікатор сертифіката за алгоритмом SHA-256, який виступає ключем у базі даних.

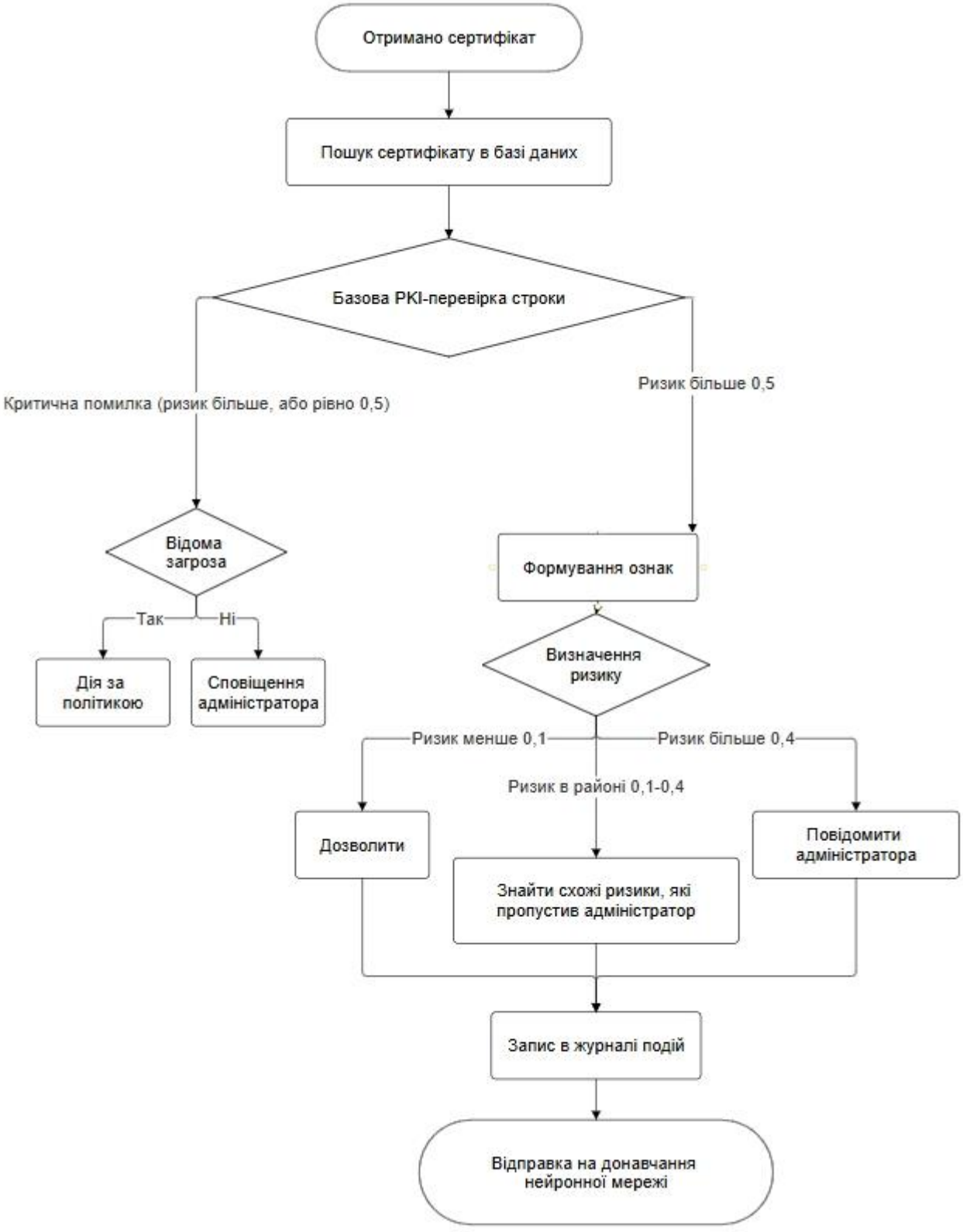


Рисунок 2.4 – Динамічна перевірка сертифіката

Якщо запис із таким відбитком існує, система використовує накопичену історію його появи, ризикові метрики та попередній статус. Якщо ж сертифікат зустрічається вперше, створюється новий запис зі статусом new, а подія реєструється у журналі аудиту.

Другим етапом є базова криптографічна валідація, у межах якої перевіряється цілісність сертифіката. Обчислюється відповідність терміну дії часові перевірки, перевіряється структура ланцюга довіри до кореневого сертифікаційного центру, довжина відкритого ключа та алгоритм підпису. Додатково здійснюється перевірка статусу відкликання через OCSP або CRL. У разі критичної невідповідності наприклад, прострочений сертифікат, невідомий CA або некоректний підпис ризик визначається як максимальний R=1.0, після чого алгоритм переходить безпосередньо до процедури блокування з'єднання відповідно до політики безпеки. Така поведінка гарантує миттєве реагування на явні загрози [35].

Якщо базова перевірка пройдена, алгоритм переходить до етапу інтелектуального аналізу ризику. На цьому рівні формується вектор ознак $X = (x_1, x_2, \dots, x_n)$, який включає як атрибути самого сертифіката, так і поведінкові параметри сесії. Наприклад, серед таких ознак можуть бути тривалість з'єднання, кількість спроб відновлення сесії, розмір ключа, тип алгоритму підпису, час доби, географічна локація клієнта, середня частота використання цього сертифіката протягом останніх 24 годин та інші. Зібрані дані нормалізуються та передаються на зовнішній сервіс машинного навчання, який реалізовано у вигляді нейронної мережі.

Математично процес прогнозування можна описати як функцію:

$$R = f(X, W)$$

де X — вектор ознак сертифіката, W — вагові коефіцієнти нейронної моделі, а $R \in [0,1]$ — розрахований рівень ризику.

Залежно від отриманого значення ризик класифікується за порогами: low, medium, high, critical. Кожен поріг відповідає певному інтервалу значень R, які були визначені експериментально на основі навчального набору даних.

Після отримання результатів ML-аналізу система виконує контекстну корекцію ризику. Наприклад, якщо в налаштуваннях політики активовано режим strict OCSP і сервер не надав дійсної OCSP-відповіді, до ризику додається штрафний коефіцієнт

$\Delta R = 0.1$ Якщо сертифікат занесений до білого списку, його ризик, навпаки, обмежується зверху $R = \min(R, 0.2)$. Таким чином, алгоритм поєднує у собі статистичну гнучкість нейромережі з контрольованими параметрами адміністративної політики.

На основі кінцевого значення ризику R′ приймається рішення щодо подальших дій. Вибір дії визначається активним режимом політики системи: якщо режим allow, сесія продовжується без змін; якщо challenge, виконується повторна перевірка або верифікація користувача; якщо rekey, ініціюється повторний обмін ключами для оновлення параметрів шифрування; якщо block, сесія примусово завершується. Всі рішення супроводжуються створенням запису у журналі аудиту, що містить опис події, час, рівень ризику та дію, яку виконала система (рис. 2.5).

Особливо важливою частиною алгоритму є система акумуляції досвіду, що забезпечує можливість донавчання моделі. Кожен сертифікат, який отримав маркування «ризиковий» або «підозрілий», додається до буфера розмічених прикладів. Якщо користувач або адміністратор підтверджує правильність класифікації, приклад переходить до навчального набору, який у подальшому використовується для оновлення вагових коефіцієнтів нейронної мережі. Це створює замкнений цикл адаптації, у якому система постійно вдосконалює власну точність і зменшує кількість хибнопозитивних спрацьовувань.

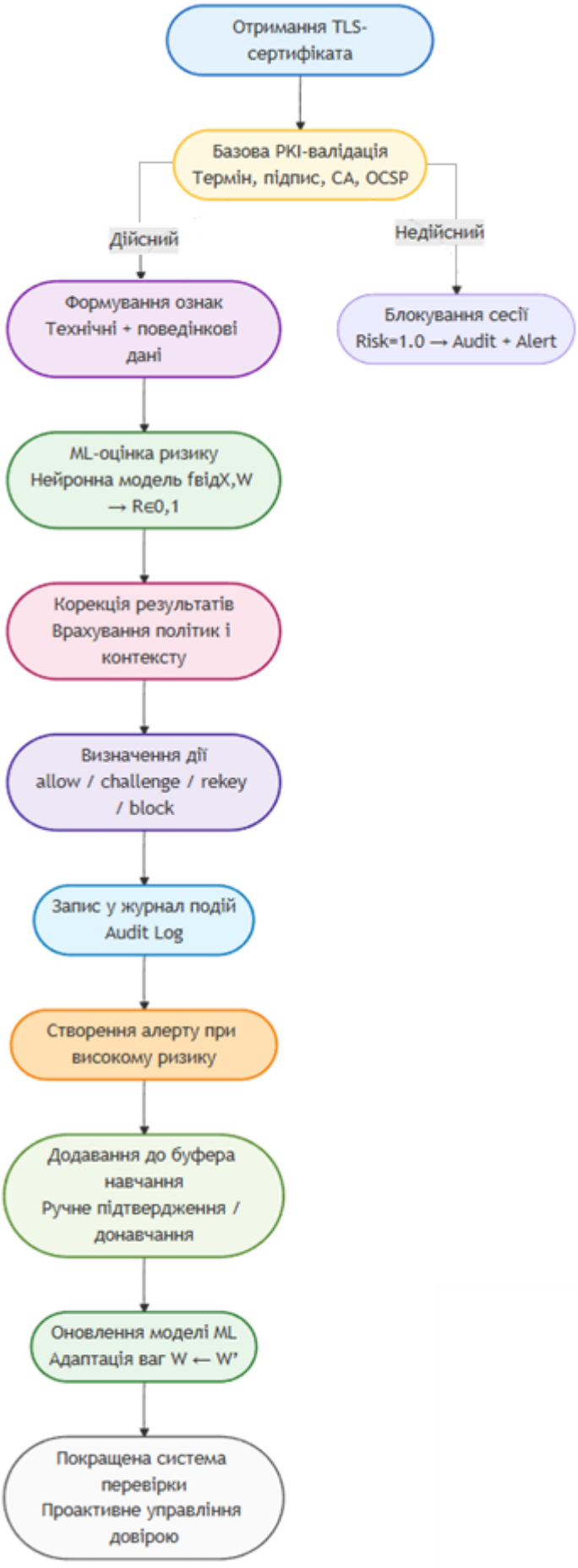


Рисунок 2.5 – Інтелектуальний цикл перевірки та адаптації

Таким чином, запропонований алгоритм динамічної перевірки сертифікатів реалізує перехід від традиційного реактивного підходу до проактивного управління довірою. Його робота поєднує формальну криптографічну строгість PKI з гнучкістю машинного аналізу. Отримані результати свідчать, що система здатна адаптуватися до змін у середовищі, виявляти нетипові шаблони поведінки сертифікатів і своєчасно блокувати потенційно небезпечні TLS-сесії. Завдяки цьому підхід забезпечує новий рівень безпеки транспортного шару, який відповідає сучасним тенденціям розвитку інтелектуальних систем кіберзахисту.

2.3 Розробка алгоритму управління TLS-сесіями

Розроблений алгоритм управління TLS-сесіями є другим ключовим компонентом удосконаленого middleware і забезпечує інтелектуальну координацію процесу встановлення, підтримання та завершення сеансів зв’язку. Якщо алгоритм динамічної перевірки сертифікатів відповідає за формування рівня довіри до кожного з’єднання, то механізм управління сесіями визначає, як система повинна реагувати на цю оцінку, які дії мають бути виконані та яким чином забезпечується цілісність комунікаційного процесу.

У традиційній реалізації TLS управління сесіями обмежується базовим обміном ключами, збереженням параметрів шифрування і можливістю відновлення сеансу (resumption). Такі механізми не враховують контекст безпеки, історію попередніх підключень або оцінку ризику. Проте в сучасних умовах, коли зловмисники активно використовують компрометацію ключів і повторні сесії для обходу перевірок, виникає потреба в динамічному рівні керування, який дозволяє змінювати поведінку системи відповідно до поточного стану ризику.

Запропонований алгоритм реалізує концепцію багаторівневого управління сесіями, де кожен рівень відповідає певному етапу життєвого циклу TLS-з’єднання: ініціація, моніторинг, адаптація, завершення. Основним принципом є тісна інтеграція цього механізму з результатами ML-оцінки ризику та

політиками безпеки, що дозволяє виконувати автоматичні рішення у режимі реального часу.

У процесі ініціації нового з'єднання middleware створює у базі даних запис про сесію з унікальним ідентифікатором, IP-адресами сторін, параметрами шифрування, часом початку та поточним статусом. Далі алгоритм отримує оцінку ризику з попереднього етапу і визначає початковий рівень довіри. Якщо рівень ризику низький, з'єднання позначається як active і продовжує роботу. Якщо ризик підвищений, система переводить сесію у режим monitoring, що означає постійне спостереження за її поведінкою та можливість дострокового розриву.

У випадку, коли ризик зростає під час обміну даними наприклад, через підозрілу частоту запитів, повторну авторизацію або нетипову зміну cipher suite, система виконує адаптивне реагування. Це може включати зміну ключів шифрування, додаткову перевірку сертифіката або тимчасове обмеження передачі даних до моменту ручного підтвердження. Формально таке рішення визначається функцією.

$$A = g(R, P, S_t)$$

де A — обрана дія (allow, rekey, challenge, block), R — поточний рівень ризику, P — активна політика безпеки, а S_t — стан сесії в момент часу t.

Функція g є адаптивною, тобто її поведінка змінюється залежно від режиму роботи системи: у стандартному режимі пріоритет віддається безперервності обміну, у захищеному — безпеці даних навіть за рахунок стабільності з'єднання.

Для підвищення точності прийняття рішень система використовує ковзне вікно моніторингу, у межах якого відстежується динаміка ризику для конкретної сесії. Нехай R_t — оцінка ризику у момент часу t, тоді інтегральна метрика стану сесії визначається як:

$$\bar{R} = \frac{1}{T}\int_{t_0}^{t_0+T} R_t dt,$$

де T — період моніторингу. Якщо середнє значення ризику перевищує поріг політики $R > R_t$, система переходить у стан реакції. Це дозволяє уникнути

випадкових спрацьовувань через короткочасні аномалії, оскільки рішення приймається з урахуванням стабільності зміни ризику.

У разі досягнення критичних значень ризику алгоритм ініціює примусове завершення сесії. Така дія не є просто закриттям TCP-каналу — вона супроводжується внесенням запису у журнал подій, створенням алерту типу system або session, а також оновленням статусу сертифіката, який використовувався у цій сесії. Після цього система може автоматично створити запит на додаткову перевірку або навчання моделі, що забезпечує зворотній зв’язок між модулями [36].

Завдяки цьому механізм управління TLS-сесіями не лише реагує на поточні загрози, а й формує активну модель поведінки системи, яка запам’ятовує характеристики попередніх атак і застосовує їх для попередження майбутніх. У термінах теорії систем це можна інтерпретувати як реалізацію петлі «спостереження–аналіз–реакція–навчання», у якій кожен новий досвід зміцнює здатність системи до самоадаптації.

Важливо, що запропонований алгоритм не порушує сумісність із протоколом TLS, оскільки всі його рішення виконуються на прикладному рівні у рамках middleware. Це забезпечує незалежність від конкретної бібліотеки шифрування і дозволяє розгортати рішення у вже існуючих інфраструктурах без модифікації клієнтського програмного забезпечення.

У практичній реалізації middleware зберігає всі дані про сесії в таблиці sessions, де для кожного запису фіксуються поточний статус, обрані дії, рівень ризику та часові позначки початку й завершення. Ця інформація використовується не лише для моніторингу, а й для статистичного аналізу ефективності політик. Наприклад, обчислення частоти переходів між станами дозволяє оцінити стабільність системи та її реактивність на потенційні атаки.

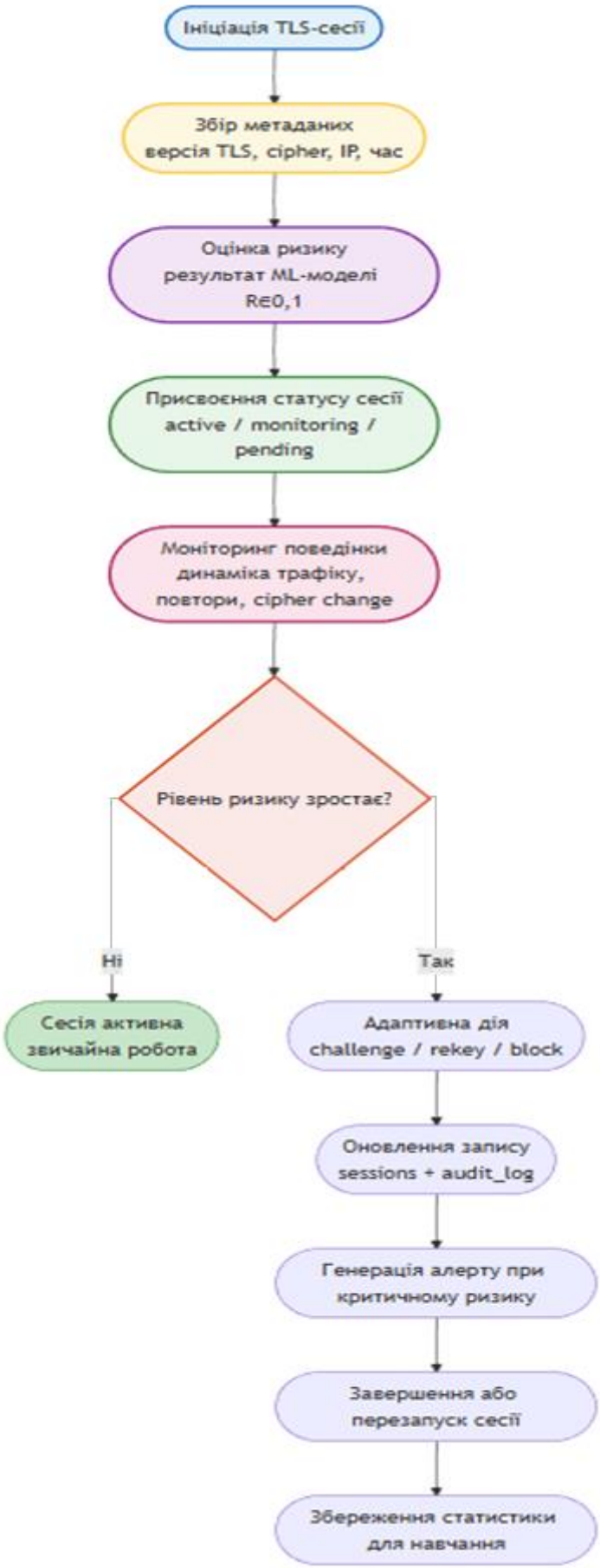


Рисунок 2.6 – Логічна структура управління TLS-сесією

Алгоритм управління TLS-сесіями також реалізує механізм часткового відновлення, який забезпечує безперервність обміну навіть після спрацьовування політики rekey. У цьому випадку новий сеанс ініціюється з використанням попередньо узгоджених параметрів, але з оновленими ключами, що гарантує безпеку без повного розриву з'єднання (див. рис. 2.6). Такий підхід поєднує переваги традиційного TLS-resumption із додатковим рівнем захисту від повторного використання сесійних ключів.

Після ініціації TLS-з'єднання відбувається збір метаданих, оцінка ризику та присвоєння статусу. Якщо ризик зростає, система застосовує адаптивну дію — від часткової перевірки до блокування. Усі рішення записуються в журнал подій, а завершення сесії супроводжується збереженням статистики для подальшого навчання нейромережі (рис. 2.7).

Кожен стан відображає певний рівень довіри: Active, Monitoring, Suspicious, Blocked. Перехід між станами відбувається відповідно до зміни оцінки ризику та результатів ML-аналізу.

Якщо ризик перевищує поріг, сесія блокується або ініціюється перегенерація ключів. Якщо ризик знижується — система автоматично повертає сесію до активного стану, зберігаючи безперервність обміну.

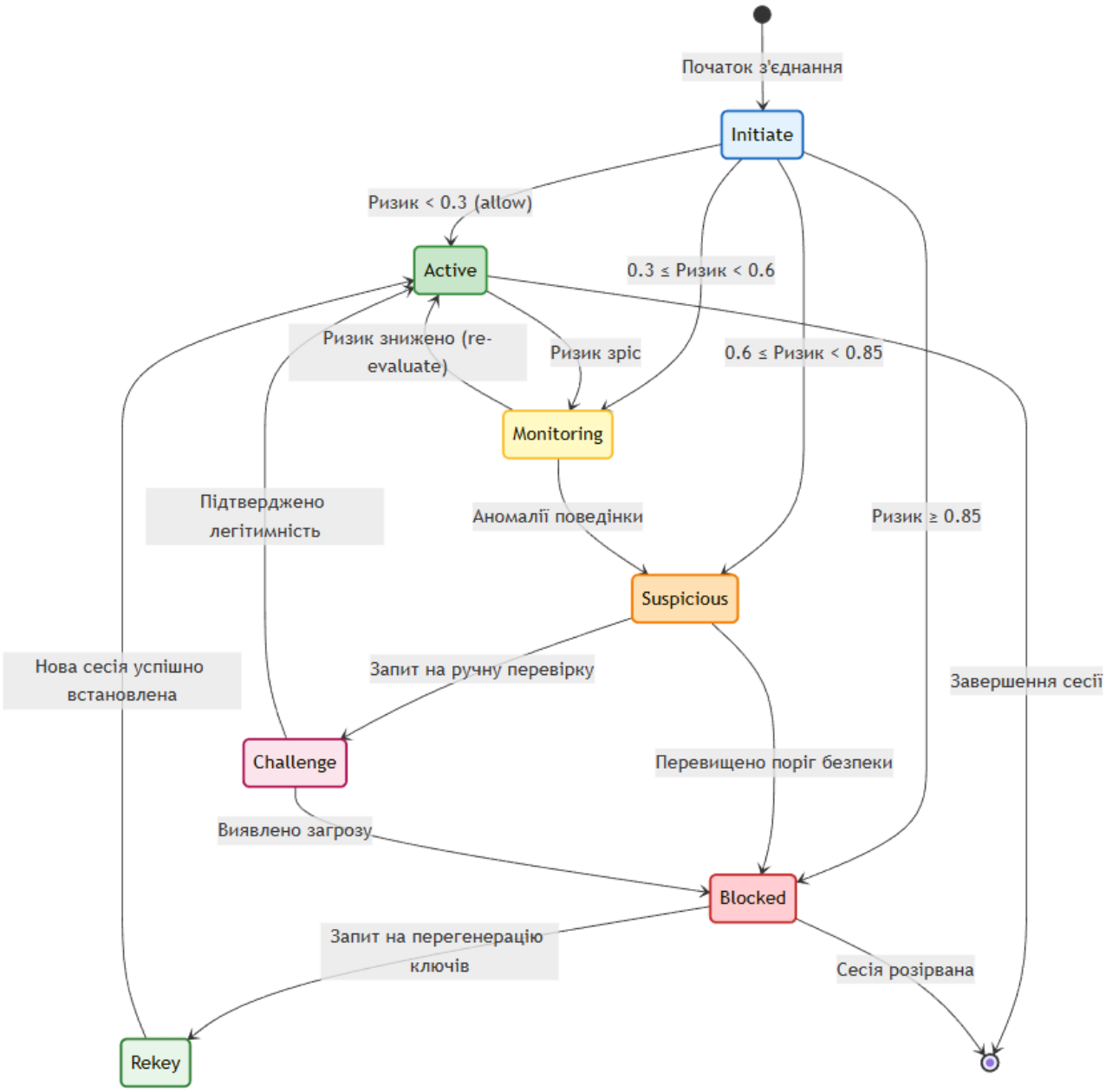


Рисунок 2.7 – Динамічна модель керування станами TLS-сесії

У підсумку розроблений алгоритм управління TLS-сесіями перетворює статичну модель обміну на динамічну адаптивну систему, яка самостійно контролює життєвий цикл з’єднань. Його інтеграція з алгоритмом динамічної перевірки сертифікатів утворює єдиний аналітичний контур безпеки, що дозволяє системі переходити від пасивного моніторингу до активного прогнозування й запобігання атакам. Такий підхід створює підґрунтя для формування інтелектуальних протоколів нового покоління, у яких безпека стає не лише властивістю середовища, а й його внутрішнім алгоритмічним станом.

2.4 Впровадження адаптивних політик безпеки

Ефективне управління TLS-сесіями неможливе без гнучкої системи політик, яка визначає, як саме система повинна реагувати на зміну рівня ризику чи появу нових загроз.

У традиційних підходах політика безпеки має фіксовані правила, наприклад «блокувати всі прострочені сертифікати» або «дозволяти лише TLS 1.3». Такий підхід є жорстким і не враховує контекст використання, динаміку поведінки клієнта чи зміну загального рівня загроз.

У запропонованому рішенні реалізовано адаптивні політики безпеки, які автоматично підлаштовуються під ситуацію, що виникає в процесі TLS-обміну.

Основна ідея полягає у тому, що кожна дія системи — дозвіл, обмеження, перевірка або блокування — визначається не єдиним правилом, а спільним рішенням моделі ризику, політики та стану сесії.

Алгоритм, який ми реалізували у PHP-middleware, використовує результати оцінки ML-моделі як вхідні дані для прийняття рішень за активними політиками.

Таким чином, політика стає не статичним набором умов, а живим регулятором, що реагує на поточний контекст.

Політики безпеки зберігаються у спеціальній таблиці бази даних, де для кожного правила задаються параметри порогів ризику, тип дії, пріоритет.

Коли система отримує нові дані про сертифікат або сесію, вона порівнює оцінку ризику. Однак, на відміну від класичних схем, значення порогів у нашій системі можуть змінюватися [37].

Ще однією особливістю є механізм зворотного зв’язку з журналом подій.

Коли у базі накопичуються випадки помилкових спрацьовувань або пропущених загроз, політика аналізує їх і коригує власні параметри.

Таким чином реалізується принцип самоадаптації — система вчиться на власних рішеннях і з часом підвищує точність реагування.

Це створює ефект «живого протоколу», який не просто перевіряє з’єднання, а навчається розуміти, у яких ситуаціях слід бути жорсткішим, а коли — більш гнучким.

Особливу роль у функціонуванні адаптивних політик відіграє контекстна кореляція.

Система не розглядає жодну TLS-сесію ізольовано, а оцінює її у зв’язку з іншими активними з’єднаннями.

Наприклад, якщо один і той самий сертифікат починає використовуватися одночасно з різних географічних регіонів або від різних клієнтів із різними агентами, політика безпеки автоматично підвищує рівень підозри навіть при невеликому ризику за базовими ознаками.

Це дозволяє виявляти атаки типу certificate reuse або session hijacking, які залишаються непоміченими у традиційних реалізаціях TLS.

Крім того, адаптивні політики підтримують механізм сценарного реагування, який дає змогу задавати послідовність дій у разі певного типу подій.

Так, при виявленні підозрілого сертифіката система може спочатку ініціювати повторну валідацію, потім автоматично обмежити тривалість сесії до 30 секунд, а якщо протягом цього часу ризик не знижується — заблокувати з’єднання та створити алерт.

Подібна логіка дозволяє гнучко реагувати на складні загрози без участі адміністратора, але при цьому зберігати контроль за процесом.

З метою забезпечення сталості та відмовостійкості адаптивних рішень у системі реалізовано багаторівневу ієрархію політик.

У разі конфлікту між політиками система застосовує правило пріоритету, при якому рішення локальної політики може перевизначити глобальне, якщо це не призводить до компрометації безпеки.

Такий підхід дозволяє поєднувати централізоване управління з можливістю тонкої настройки для конкретних сценаріїв.

Завдяки поєднанню нейромережевого аналізу ризику та адаптивних політик безпеки, система стає еволюційною — вона не лише реагує на загрози, а й розвивається під впливом нових даних.

Поступове уточнення політик, корекція порогів і адаптація правил відбуваються без потреби у втручанні користувача, що фактично перетворює middleware на самонавчальний безпековий модуль.

У майбутньому це відкриває можливість інтеграції з іншими протоколами, формуючи універсальну платформу динамічного управління довірою в мережевих системах.

З практичної точки зору адаптивні політики дозволяють адміністраторам балансувати між продуктивністю та безпекою, не змінюючи програмний код.

Замість редагування правил у вихідних файлах, користувач може оновити параметри безпосередньо через панель керування, задавши інший набір порогів або реакцій.

Узагальнюючи, адаптивні політики є логічним продовженням попередніх алгоритмів, забезпечуючи зовнішній контур управління поведінкою системи.

Такий підхід дозволяє перетворити TLS із простої процедури шифрування даних на адаптивний інтелектуальний протокол, здатний до самостійного прийняття рішень у мінливих умовах кіберсередовища.

2.5 Інтеграція нейронної мережі у процес TLS-оцінювання

Інтеграція нейронної мережі у процес оцінки TLS-сесій є центральним елементом запропонованої архітектури, який забезпечує перехід від статичного аналізу до інтелектуального прийняття рішень.

Якщо традиційні методи перевірки сертифікатів обмежуються валідацією криптографічного ланцюга довіри, то впровадження машинного навчання дозволяє виявляти закономірності, що не піддаються формальному опису, та прогнозувати ризик компрометації ще до появи ознак атаки.

Механізм взаємодії між TLS-middleware та нейронною мережею реалізовано у вигляді окремого модуля inference-сервісу, який приймає на вхід зібрані метадані про сертифікат і сесію, а повертає числову оцінку ризику $R \in [0,1]$.

Передача даних здійснюється через REST або локальний сокет-інтерфейс, що дозволяє використовувати модель, розгорнуту на іншому сервері.

Таким чином забезпечується ізоляція моделі від бізнес-логіки middleware, що підвищує стабільність і спрощує оновлення нейромережі без втручання у роботу PHP-системи.

Основою для формування прогнозу є вектор ознак, який описує як сам сертифікат, так і контекст його використання.

До нього входять такі групи параметрів:

– криптографічні: тип ключа, довжина, алгоритм підпису, глибина ланцюга довіри;

– мережеві: IP-адреса клієнта, регіон, частота запитів, середня тривалість з'єднань;

– часові: вік сертифіката, частота його повторного використання, зміни параметрів у часі;

– історичні: результати попередніх перевірок, статуси, кількість інцидентів.

Нейронна мережа навчається на історичних даних, де кожен запис має мітку ризику, визначену на основі попередніх рішень системи або ручного маркування адміністратором.

Для побудови моделі використовується класична багатошарова нейронна мережа (feed-forward, MLP), що реалізує функцію:

$$R = f(W, X) = \sigma(W_3 * \varphi(W_2 * \varphi(W_1 * X + b_1) + b_2) + b_3)$$

де X — вектор ознак, W_i, b_i — вагові коефіцієнти та зсуви шарів, $\varphi(\dots)$ — функція активації (наприклад, $R \in LU$), а $\sigma(\dots)$ — сигмоїдна функція, що нормує вихід у діапазон [0,1].

Отримане значення R інтерпретується як ймовірність компрометації або порушення довіри, а отже, безпосередньо впливає на політики та управління сесією.

Важливо, що у нашій системі модель не є ізольованим елементом — вона працює у замкненому циклі з middleware, отримуючи нові приклади під час експлуатації.

Кожна TLS-сесія, що завершується з аномальним результатом або помилковою класифікацією, потрапляє у таблицю suspicious_certs, де зберігається її контекст і подальше рішення адміністратора.

Після накопичення достатньої кількості нових даних модель може бути донавчена — вручну або автоматично за розкладом. Це забезпечує адаптивність нейромережі до змін у середовищі, появи нових типів атак або модифікованих сертифікатів.

У процесі навчання застосовується функція втрат крос-ентропії:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(R_i) + (1 - y_i)\log(1 - R_i)]$$

де y_i — фактична мітка (0 — безпечний, 1 — небезпечний), а R_i — передбачена мережею оцінка ризику. Мінімізація цієї функції дозволяє оптимізувати ваги так, щоб вихідна оцінка відповідала реальним результатам перевірки. Ще однією особливістю є використання порогової корекції ризику.

Система може автоматично зменшувати або збільшувати вихідне значення R залежно від глобального стану середовища, наприклад:

$$R` = R + \alpha * (S - 0.5),$$

де S — рівень загальної активності підозрілих сертифікатів у системі, а α — коефіцієнт корекції. Це дозволяє уникати завищення або заниження ризиків у періоди змін мережевого трафіку, роблячи оцінку більш стабільною.

Таким чином, інтеграція нейронної мережі перетворює процес TLS-оцінки на самонавчальну систему довіри, у якій рішення базуються не лише на правилах, а й на досвіді [38].

Кожен новий інцидент підсилює модель, роблячи її точнішою; кожна зміна політики стає частиною процесу адаптації.

У результаті система TLS набуває властивостей інтелектуального агента, який здатен не лише виявляти, а й передбачати потенційні загрози на основі накопиченого знання.

Це відкриває шлях до створення автономних модулів безпеки, де криптографічні механізми та штучний інтелект діють як єдиний цілісний організм захисту транспортного шару.

2.6 Вибір інструментів для реалізації клієнтської частини

Реалізація клієнтської частини удосконаленого TLS-middleware має подвійне призначення: з одного боку, вона забезпечує взаємодію користувача з системою у зручному графічному форматі, а з іншого — виконує функцію візуалізації результатів аналізу, управління сесіями та адаптивними політиками безпеки.

Оскільки розроблений прототип є web-орієнтованим рішенням, ключовими вимогами до вибору інструментів стали зручність, гнучкість, простота розгортання та мінімальні залежності від серверного середовища.

Основу клієнтської частини складає мова PHP, яка використовується не лише для серверної логіки, але й для побудови інтерактивних web-сторінок через шаблонізовану структуру MVC-типу.

Вибір PHP обумовлений його широкою підтримкою, високою сумісністю з Apache/Nginx, а також можливістю швидкої інтеграції з бібліотекою RedBeanPHP, яка спрощує роботу з базою даних MySQL.

RedBeanPHP дозволяє використовувати ORM-підхід без складних конфігурацій, що є важливим для прототипів дослідницьких систем, де пріоритетом є швидкість розробки та наочність архітектури.

Для побудови користувацького інтерфейсу обрано фреймворк Bootstrap 5, який забезпечує адаптивність сторінок, уніфікований дизайн і сумісність з більшістю сучасних браузерів.

Використання Bootstrap дає змогу реалізувати структуру з картками, вкладками, навігаційним меню, таблицями та модальними вікнами, що робить інтерфейс не лише функціональним, але й візуально привабливим.

Додаткові елементи взаємодії реалізовані за допомогою JavaScript та бібліотеки jQuery, які дозволяють динамічно оновлювати сторінки без повного перезавантаження, реалізувати фільтрацію, сортування, асинхронне оновлення сесій та генерацію алертів у реальному часі.

Інтеграція клієнтської частини з нейромережевим модулем виконується через AJAX-запити до REST-API, що дозволяє отримувати оцінку ризику безпосередньо під час відображення інформації у браузері.

Таким чином, інтерфейс стає не лише інструментом моніторингу, а й активним компонентом системи, який забезпечує візуальний контроль за роботою ML-моделі та результатами її прогнозів.

Така комбінація дозволяє поєднати технічну строгість із сучасним дизайном, роблячи систему зрозумілою для користувача будь-якого рівня підготовки.

Запропонована архітектура клієнтської частини є повністю автономною: усі сторінки можуть працювати локально на одному сервері без зовнішніх залежностей, що підвищує стабільність і безпеку розробки.

При цьому структура MVC з чітким поділом на controllers, views та models спрощує подальше масштабування — наприклад, додавання нових модулів, таких як моніторинг ML-навчання або розширене логування мережевих подій.

Таким чином, вибір технологічного стеку для клієнтської частини системи є виправданим як з точки зору швидкої реалізації прототипу, так і з позицій надійності, гнучкості та наочності.

У поєднанні з інтелектуальним модулем аналізу ризиків та адаптивними політиками безпеки, створений інтерфейс забезпечує повноцінну взаємодію користувача з системою, перетворюючи middleware на повнофункціональний інструмент моніторингу, дослідження та керування безпекою TLS-з’єднань.

2.7 Висновки до розділу

У другому розділі було спроектовано, описано та обґрунтовано архітектуру вдосконаленого криптографічного протоколу TLS, який поєднує механізми динамічної перевірки сертифікатів, багаторівневого управління сесіями та адаптивних політик безпеки з використанням нейронної мережі для оцінки ризику.

У межах дослідження розроблено концепцію TLS-middleware, який функціонує на прикладному рівні та не потребує модифікації базового стеку протоколу.

Особливу увагу приділено розробці адаптивних політик безпеки, які є логічним продовженням механізмів аналізу та управління.

У розділі також описано інтеграцію нейронної мережі, яка реалізує інтелектуальну оцінку ризику сертифікатів і сесій. Модель отримує на вхід багатовимірний вектор ознак, навчається на історичних даних та формує числову оцінку довіри, яка використовується політиками безпеки для ухвалення рішень.

Передбачено механізм донавчання на основі нових подій, що дозволяє системі еволюціонувати у процесі експлуатації, підвищуючи точність прогнозів.

Для візуалізації та зручної взаємодії з користувачем розроблено web-інтерфейс клієнтської частини, реалізований засобами PHP, Bootstrap і JavaScript.

Таким чином, у другому розділі було сформовано цілісну архітектурну модель інтелектуального TLS-протоколу нового покоління, що поєднує формальні механізми криптографічної перевірки з адаптивним машинним аналізом.

Отримані рішення створюють фундамент для практичної реалізації прототипу, описаної у наступному розділі, та закладають основу для розробки універсальних інтелектуальних систем управління довірою у транспортних протоколах.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВДОСКОНАЛЕНОЇ TLS-СИСТЕМИ

У даному розділі представлено практичну реалізацію вдосконаленої TLS-системи. Для цього було використано теоретичну розробку з минулого розділу. Для практичної реаліації покращеного алгоритму, в даному розділі представлено етапи сторення інтерфейсу, описано розробку бази даних та представлено реалізацію навчання нейронної мережі.

В результаті, було отримано працюючу систему, яка використовує розроблений покращений протокол TLS з використанням нейронної мережі.

3.1 Розробка графічного інтерфейсу програмної розробки

Головна сторінка є центральною точкою входу в систему. Вона побудована у вигляді інформаційної панелі з шістьма інтерактивними картками-плитками, кожна з яких веде до відповідного модуля системи:

- сесії TLS — відображає активні та завершені TLS-з’єднання;
- сертифікати — база збережених та підозрілих сертифікатів;
- ML-аналіз — аналітичний модуль, що демонструє оцінку ризику нейромережею;
- політики безпеки — набір адаптивних правил реагування системи;
- алерти — журнал попереджень та подій безпеки;
- журнал аудиту — історія усіх змін і дій користувача.

Кожна картка оформлена у стилі панелі з іконкою, кольоровим акцентом та коротким поясненням функціоналу (рис. 3.1).

Таким чином, користувач з перших секунд бачить загальну архітектуру системи та може миттєво перейти до потрібного розділу.

Сторінка TLS-сесії показує таблицю всіх зафіксованих TLS-з’єднань, як активних, так і завершених.

Кожен рядок відображає такі параметри:

- Session ID;
- Client IP і Server IP;

– TLS Version і Cipher Suite;
– рівень ризику, позначається кольоровим бейджем;
– поточна дія системи;
– статус;
– дата початку/завершення.

Рисунок 3.1 – Головна сторінка

Користувач може натиснути на кнопку «Перегляд» для переходу до детальної сторінки певної сесії. У верхній частині таблиці передбачено кнопку «Додати сесію», яка створює тестовий запис — це дозволяє перевірити реакцію системи, ML-оцінку та роботу політик у режимі реального часу (рис. 3.2).

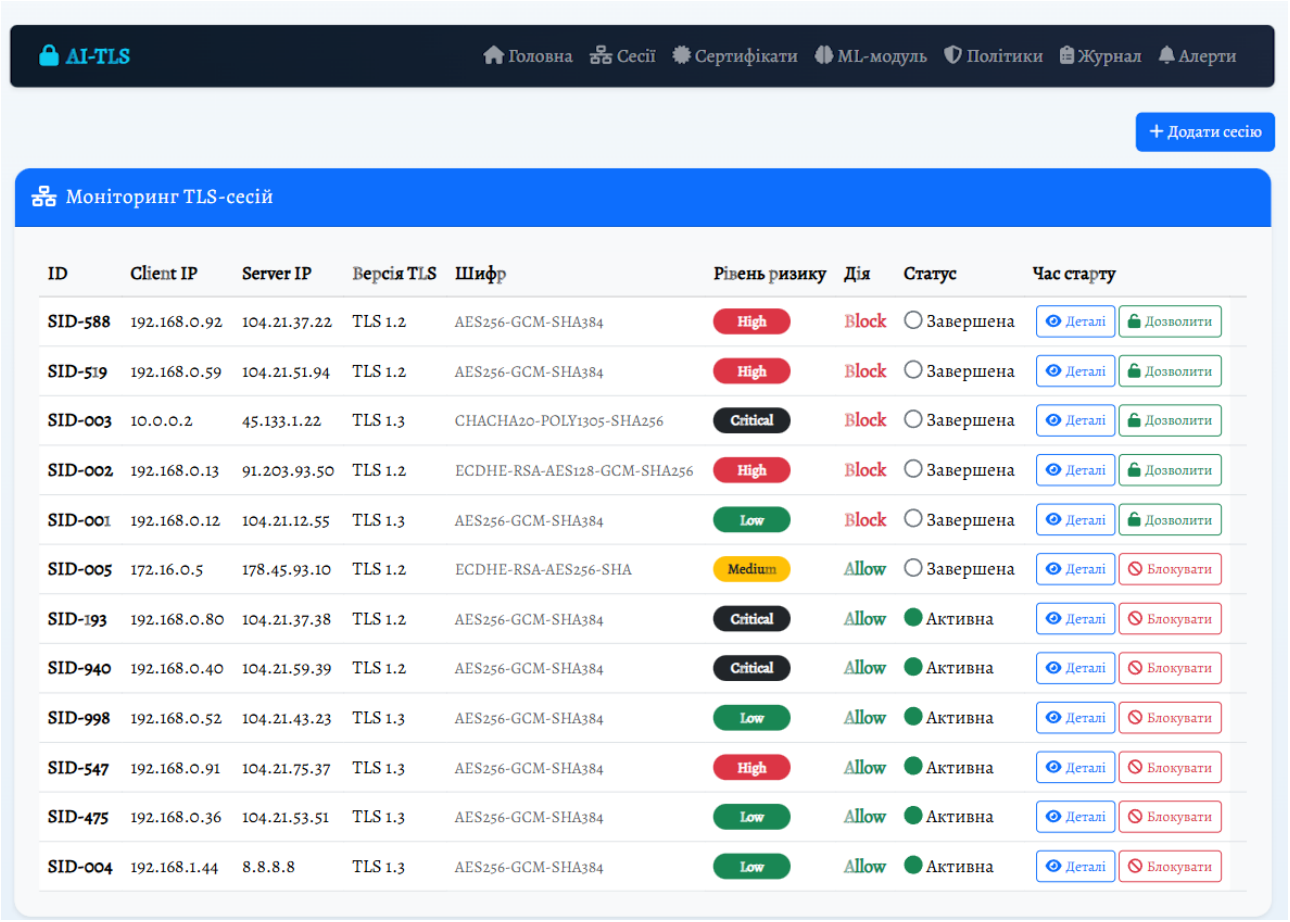


Рисунок 3.2 – Моніторинг сесій

На сторінці перегляду сесії відображається повна інформація про конкретне TLS-з’єднання.

У верхній частині показано загальні параметри:

– версія TLS, шифрна суїта, ідентифікатор сесії;

– клієнтська і серверна IP-адреси;

– дата початку та завершення.

Під ними розташовано панель оцінки ризику — тут виводиться значення ризику, обчислене нейромережею, а також словесна класифікація ризику (рис. 3.3).

Якщо система визначила сесію як підозрілу, її можна вручну закрити натисканням кнопки «Закрити сесію».

У нижній частині сторінки показано журнал подій для цієї сесії, де відображаються всі дії, що відбувалися з нею створення, зміна статусу, оновлення політик тощо (рис. 3.4).

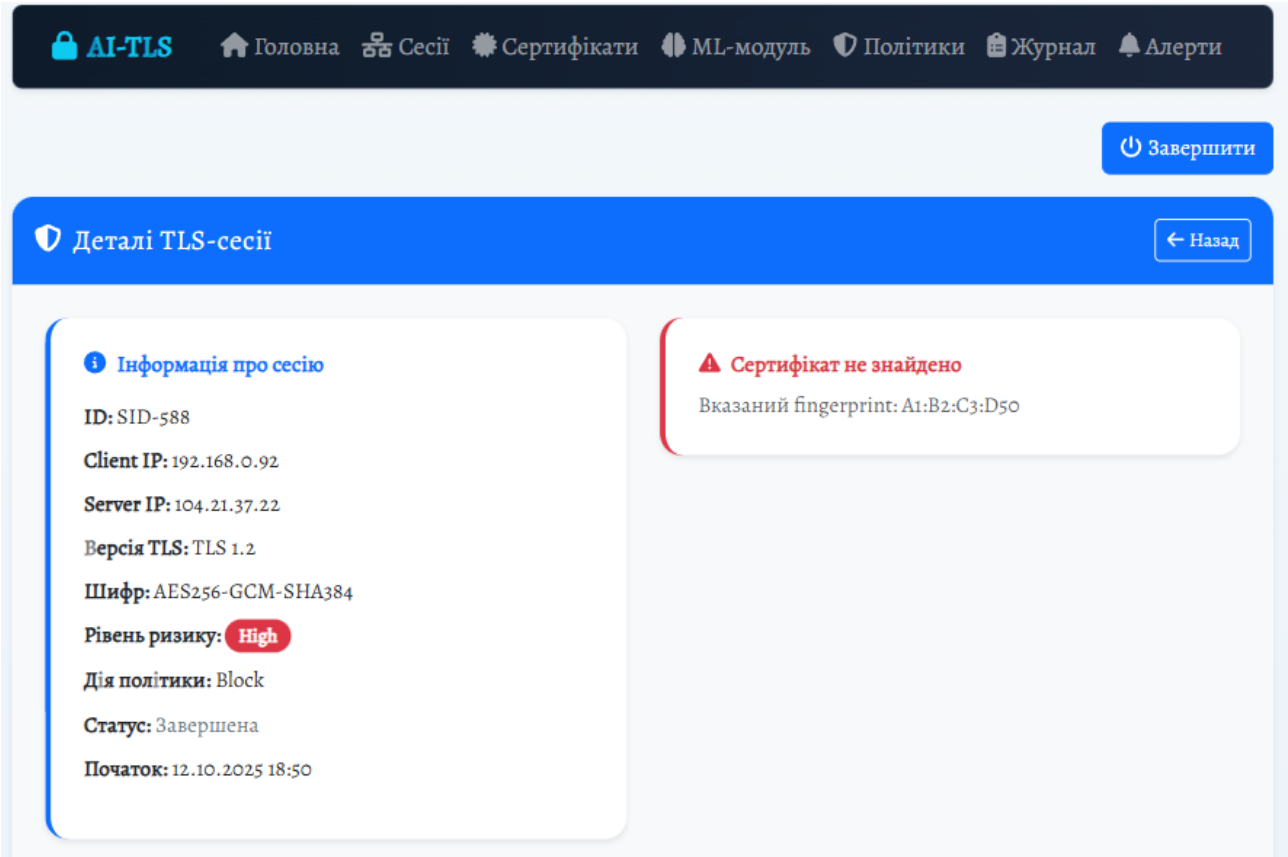


Рисунок 3.3 – Сторінка сесії з високим ризиком

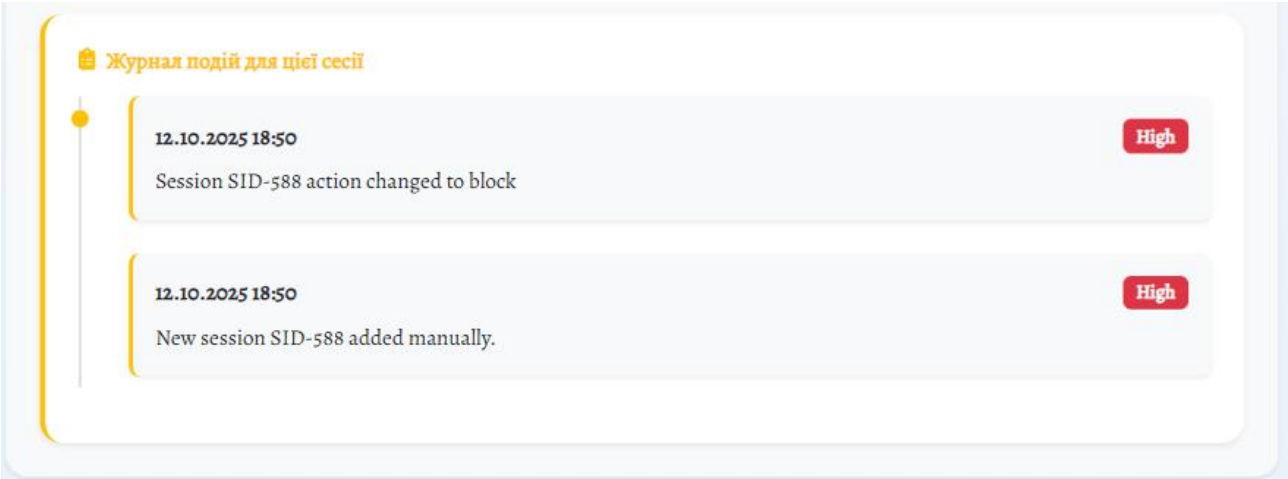


Рисунок 3.4 – Журнал подій

На сторінці Сертифікати представлена база сертифікатів, які система зустрічала під час TLS-обміну (рис. 3.5).

Для кожного сертифіката виводяться такі поля:

– Fingerprint SHA-хеш сертифіката;

– Common Name;

– Issuer (Центр сертифікації);
– дата дії;
– оцінка моделі.

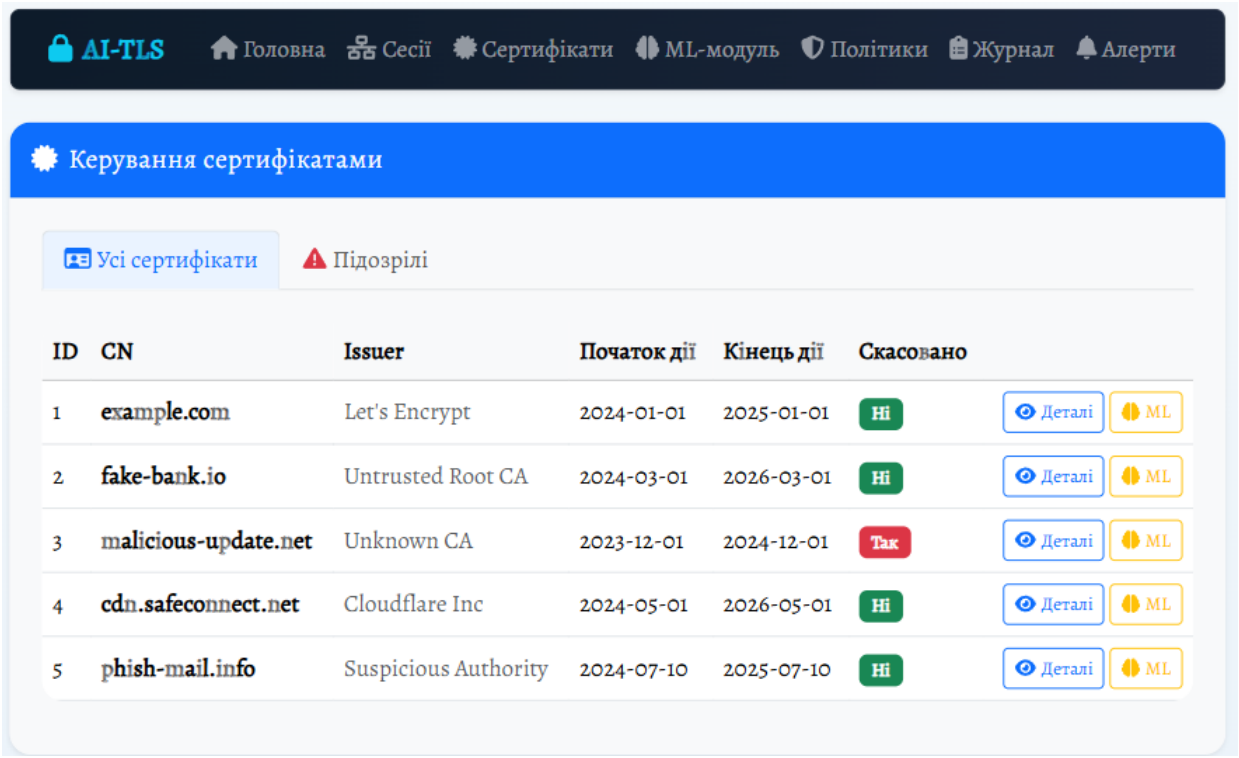


Рисунок 3.5 – Список всіх сертифікатів

Сертифікати з високим ризиком підсвічуються червоним кольором. Передбачено кнопки для позначення сертифіката як перевіреного або для додавання його до списку «безпечних».

Також можна перейти до перегляду детальної інформації про конкретний сертифікат.

Детальна сторінка сертифіката дає повну характеристику обраного сертифіката. У верхній частині показано основні дані: fingerprint, CN, видавець, термін дії, алгоритм підпису (рис. 3.6).

Далі розміщується аналітична секція ML, де наведено:

– оцінка ризику;
– пояснення нейромережевої моделі;
– історія появ цього сертифіката у сесіях;
– кнопки для ручного маркування.

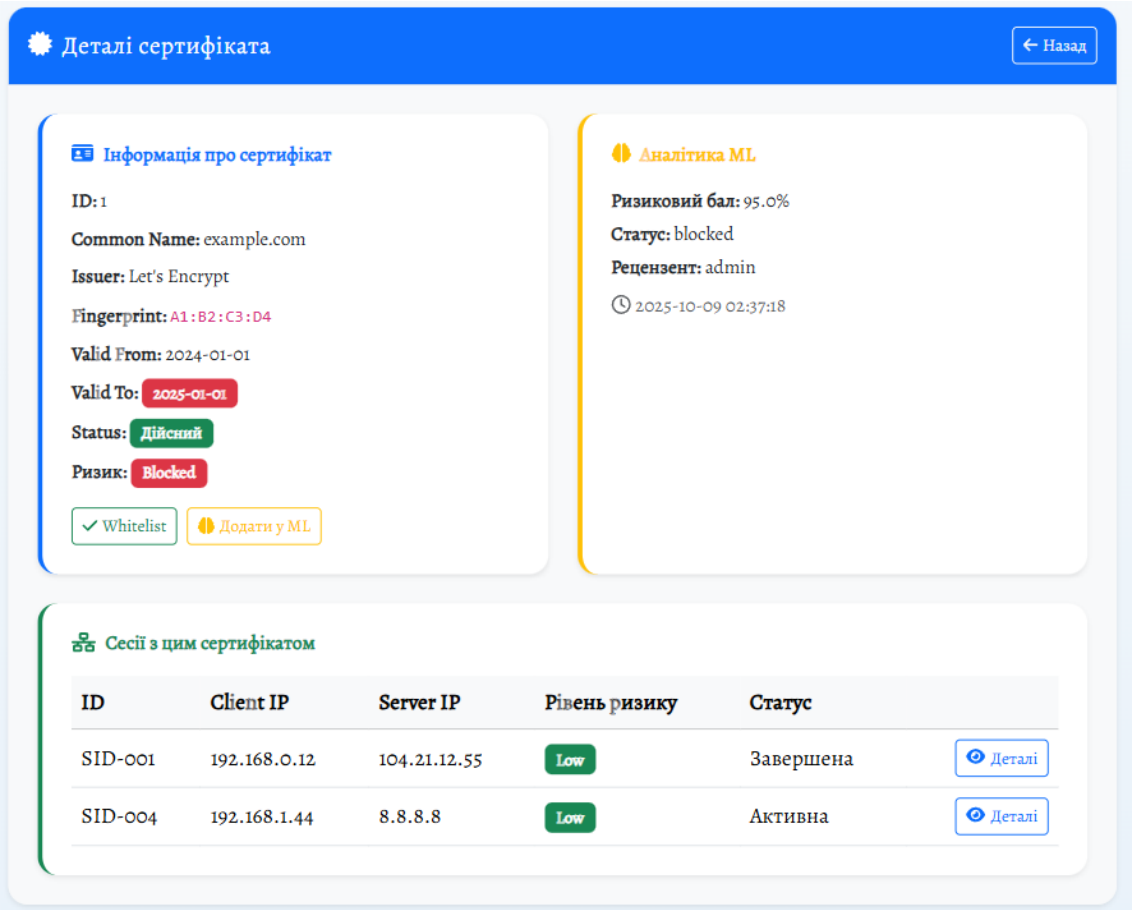


Рисунок 3.6 – Детальний перегляд недійсного сертифіката

У нижній частині — таблиця з історією дій над цим сертифікатом, що дозволяє простежити зміни статусу у часі.

Цей екран є центральним для демонстрації можливостей інтелектуального аналізу (рис. 3.7).

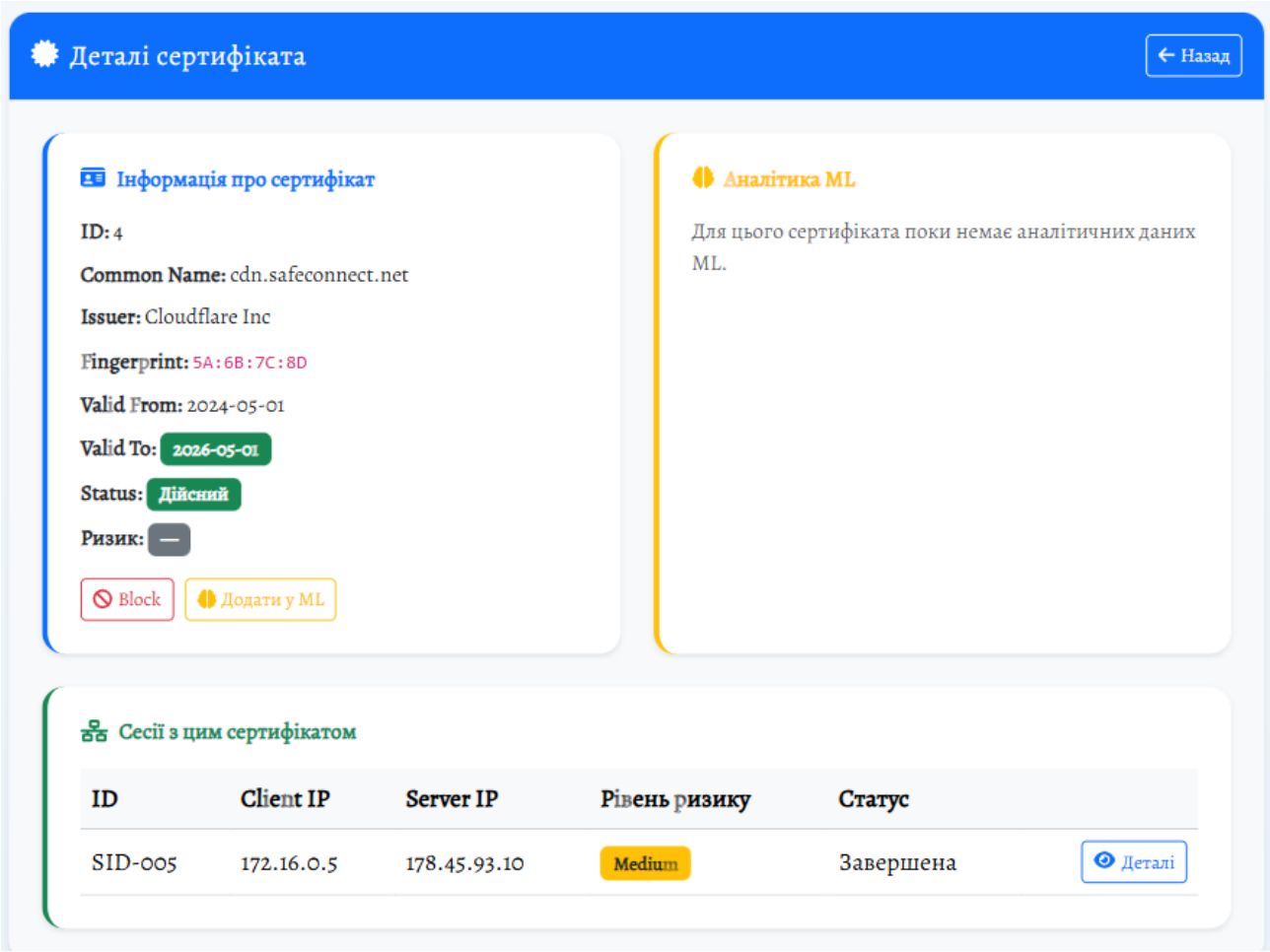


Рисунок 3.7 – Детальний перегляд сертифіката без ML аналізу

Модуль ML-аналіз призначений для візуалізації роботи нейронної мережі, яка оцінює ризик сертифікатів і TLS-сесій.

У верхній частині виводиться поточна модель model_weights.json, дата її останнього навчання та середня точність на тестовій вибірці.

Також є секція «Донавчання», яка дозволяє перезапустити train.php безпосередньо з інтерфейсу — після натискання виконується навчання моделі на основі нових розмічених даних (рис. 3.8).

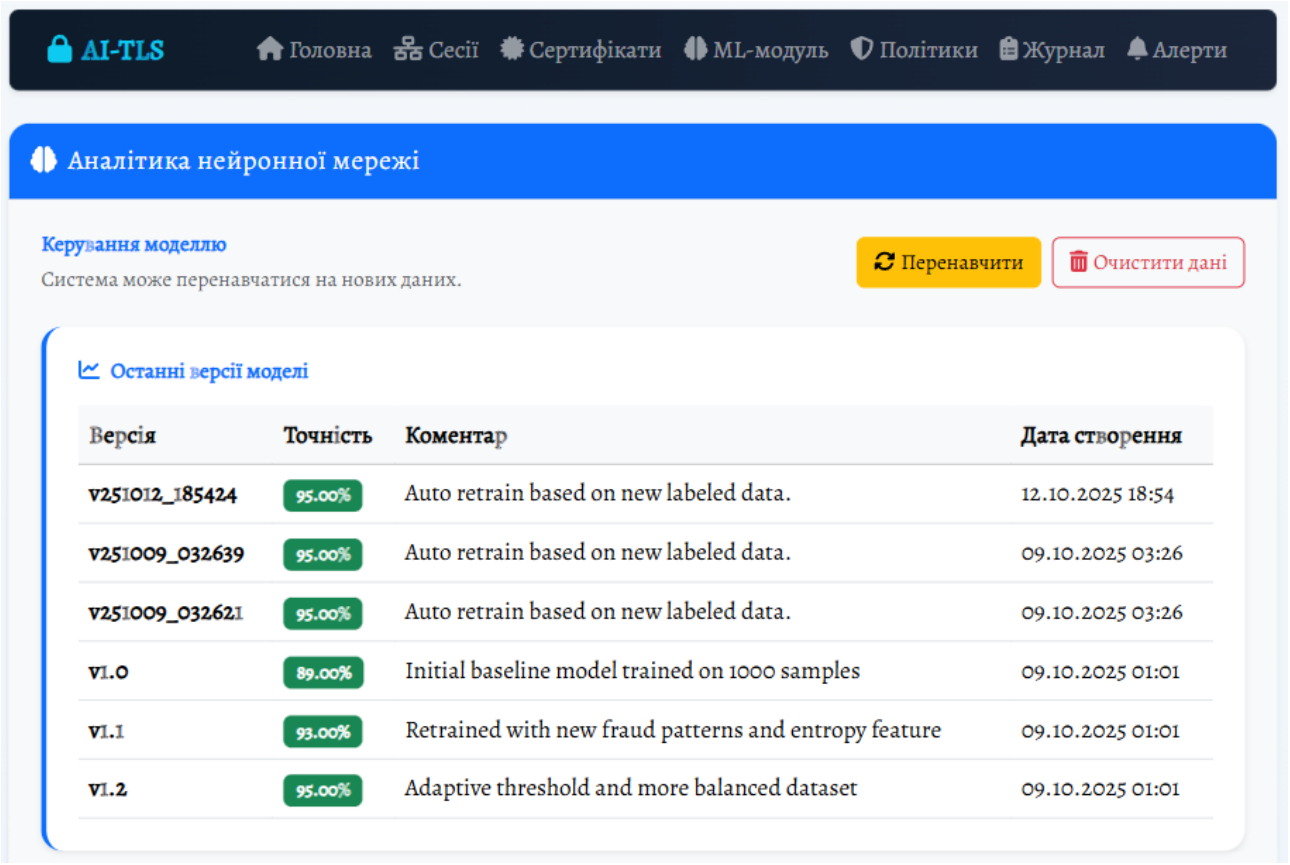


Рисунок 3.8 – Сторінка аналітики нейронної мережі

Під нею розташовано блок з останніми прикладами для навчання системи. На їх основі відбувається аналітичні зміни під час роботи нейронної мережі (рис. 3.9).

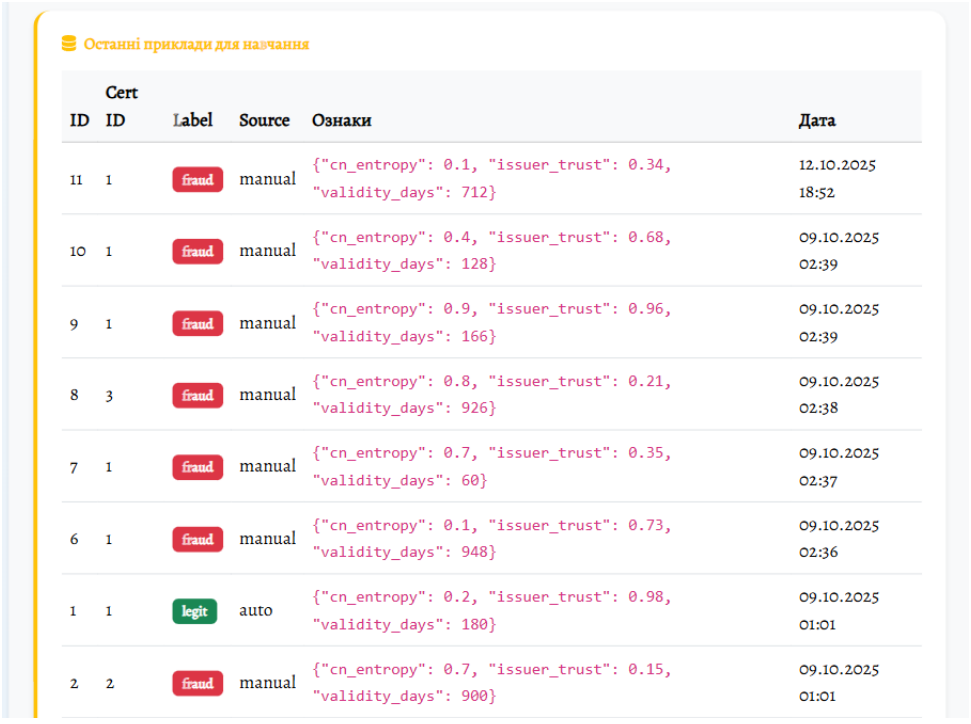


Рисунок 3.9 – Список прикладі для навчання

На цій сторінці «Політики безпеки» користувач може переглянути і змінювати активні адаптивні політики TLS-реагування. Кожна політика має власний рядок із полями:

– назва політики;
– режим роботи;
– пороги ризику;
– опис дії.

У верхній частині розташована форма для зміни режиму роботи політики або додавання нової, що дозволяє експериментувати з параметрами без редагування бази вручну (рис. 3.10).

Ця сторінка безпосередньо впливає на поведінку middleware при оцінці нових з’єднань.

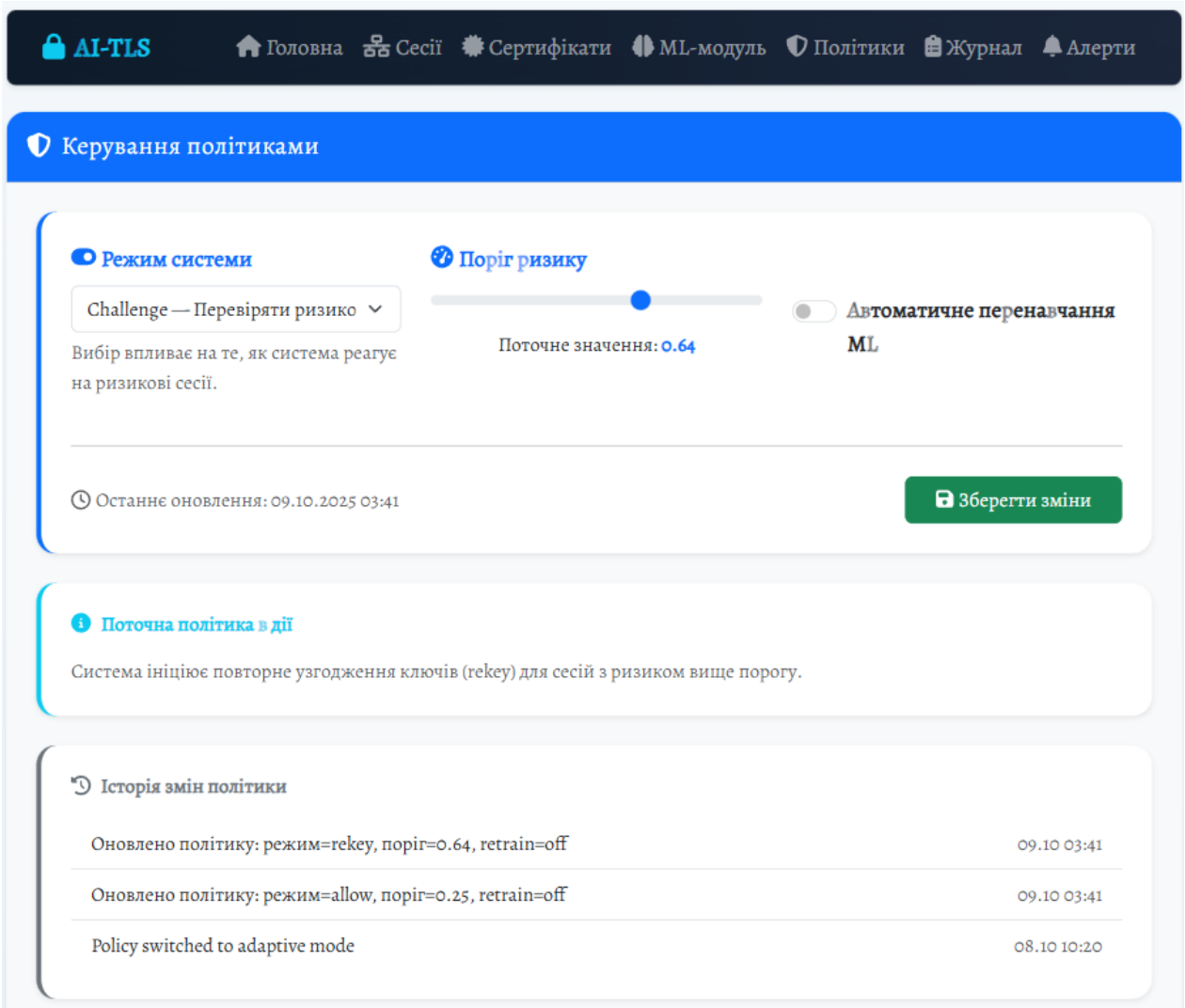


Рисунок 3.10 – Сторінка керування політиками

Під таблицею розташована частина де відображається зведена статистика реакцій системи — кількість активних, під перевіркою та заблокованих сесій (рис. 3.11).

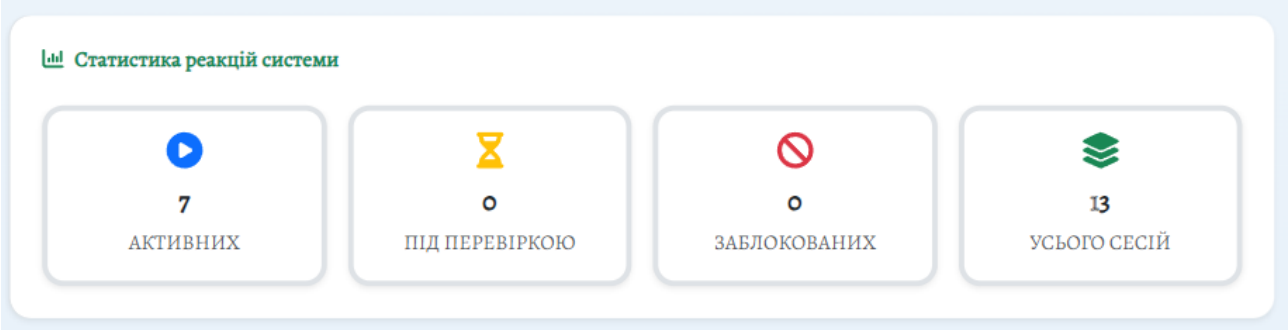


Рисунок 3.11 – Загальна статистика роботи

Модуль «Журнал подій» забезпечує повний аудит усіх дій системи. Для кожного запису вказано тип події, короткий опис, рівень ризику та часову мітку. Цей розділ виконує збереження всієї історії системи, забезпечуючи прозорість і відтворюваність усіх рішень (рис. 3.12).

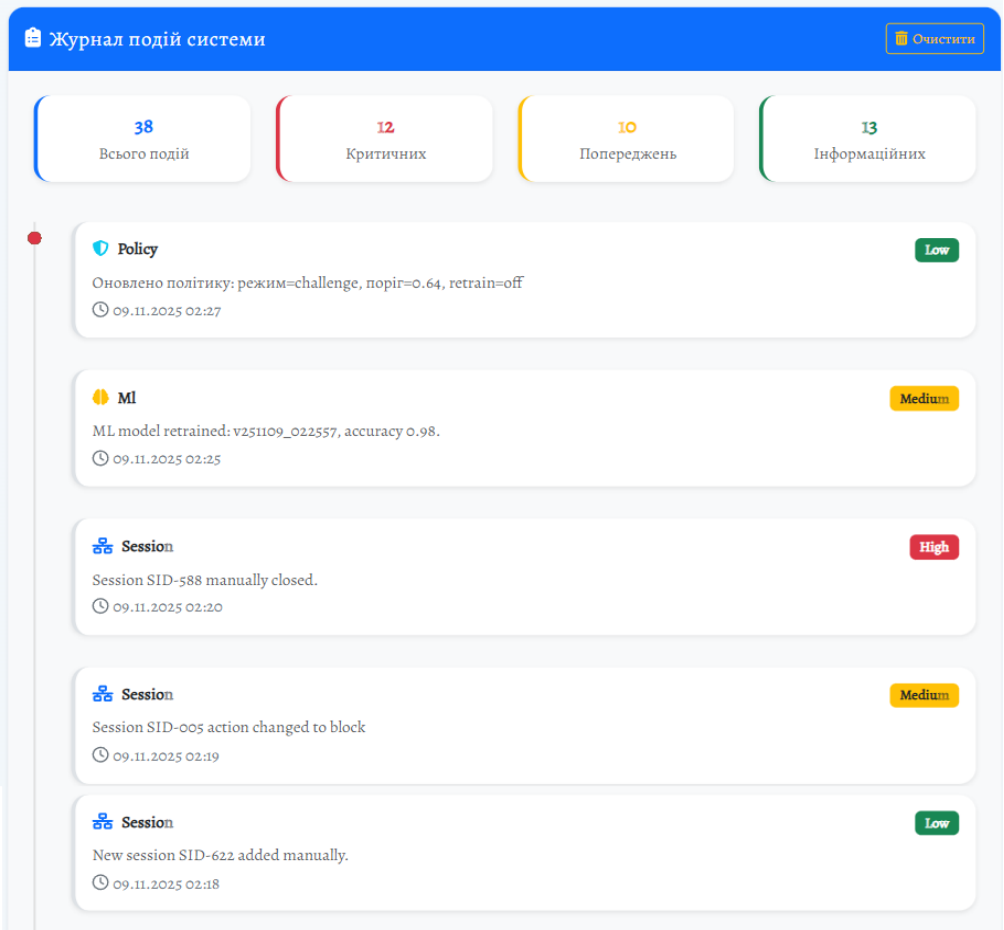


Рисунок 3.12 – Журнал подій

В журналі подій зберігаються (рис. 3.13):

– записи про створення/завершення сесій;
– зміни статусів сертифікатів;
– спрацювання політик;
– ручні втручання користувача.

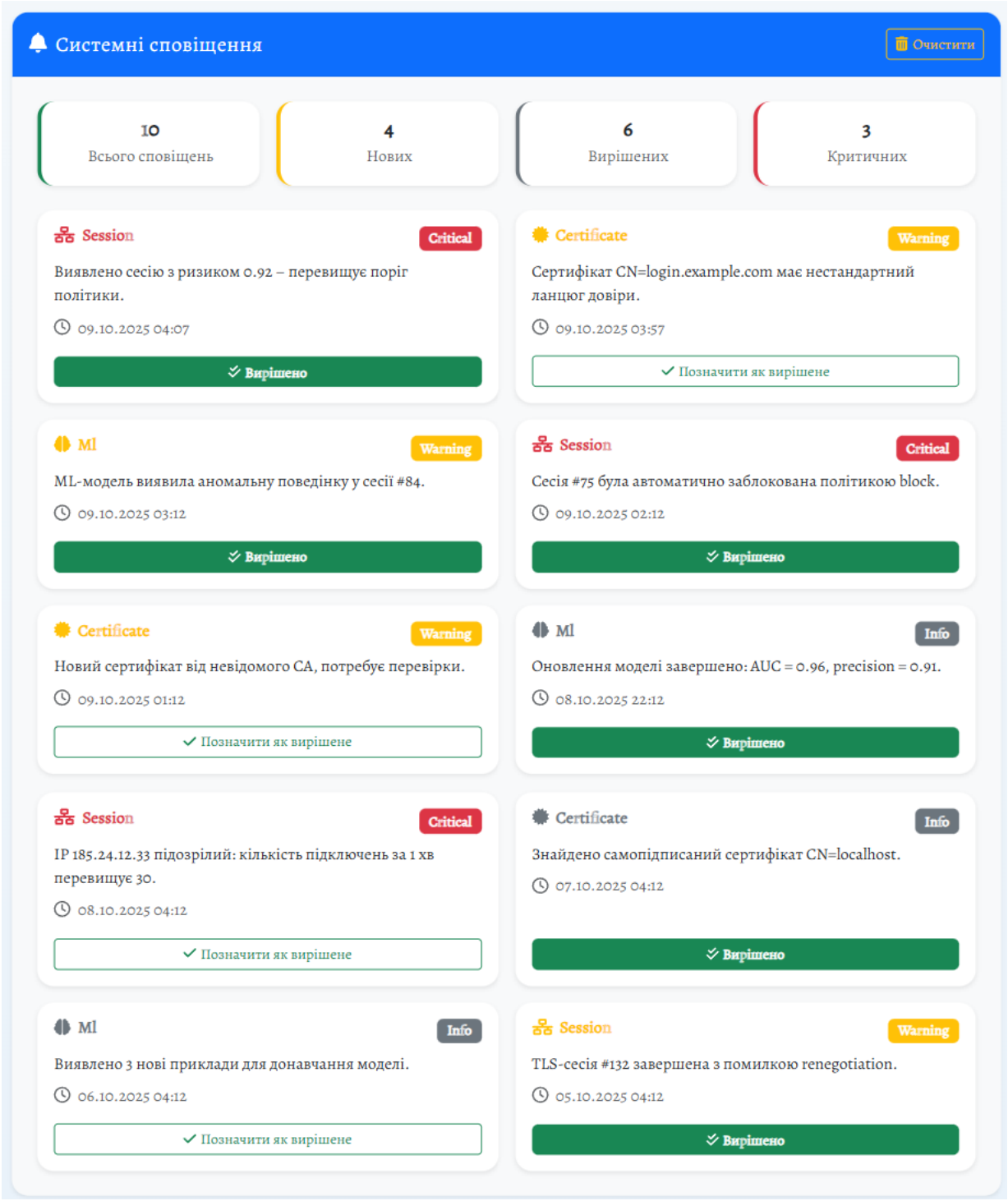


Рисунок 3.13 – Модуль «Алерти» відображає поточні попередження системи, які потребують уваги адміністратора.

Кожен запис має поля:

- тип події;
- опис події;
- рівень ризику;
- дата створення.

Події сортуються за часом, найсвіжіші зверху.

Зверху сторінки відображається зведена панель статистики — кількість критичних і попереджувальних алертів.

Модуль працює синхронно з журналом подій, будь-яка критична зміна або результат високого ризику одразу дає новий запис тут.

3.2 Програмна реалізація вдосконаленого алгоритму

Серверна частина удосконаленої TLS-системи реалізує основну логіку функціонування middleware та забезпечує взаємодію між усіма модулями — зокрема, перевіркою сертифікатів, управлінням сесіями, політиками безпеки, журналом подій та нейромережевим inference-сервісом.

Вона є центральним ядром системи, яке координує потік даних між клієнтським інтерфейсом, базою даних і модулем машинного навчання.

Архітектурно серверна частина побудована за принципами моделі MVC, що забезпечує розподіл відповідальності між логікою обробки даних, відображенням інформації та управлінням запитами користувача.

Усі контролери зосереджені у папці controllers, кожен з них відповідає окремому функціональному модулю: SessionsController, CertificatesController, PoliciesController, AlertsController, а також допоміжний MainController, який керує головною сторінкою та навігацією.

Кожна дія контролера відповідає певній web-сторінці або запиту системи, наприклад:

- /sessions/view?id=3 — перегляд інформації про окрему TLS-сесію;
- /certificates — список сертифікатів і їх статусів;
- /policies — управління адаптивними політиками;

– /alerts — перегляд системних сповіщень та логів подій;

Усі дані зберігаються в реляційній базі MySQL, доступ до якої здійснюється через ORM-бібліотеку RedBeanPHC.

Використання ORM дозволило уникнути ручного написання SQL-запитів і забезпечило динамічне створення таблиць при першому зверненні до них.

Основні таблиці системи:

– sessions — зберігає інформацію про TLS-з’єднання ідентифікатор, IP-адреси, версію протоколу, рівень ризику, статус, часові мітки;

– certificates — базу сертифікатів, дата дії, поточний статус;

– policy_settings — набір активних політик із параметрами порогів ризику та типом реакції;

– audit_log — журнал подій, який фіксує всі зміни станів і дій системи;

– alerts — повідомлення про інциденти, що потребують уваги адміністратора;

– suspicious_certs — накопичувач прикладів для донавчання нейронної моделі;

Ця структура забезпечує як функціональну незалежність модулів, так і можливість масштабування системи без порушення логічних зв’язків між ними.

Після отримання нового TLS-з’єднання або симуляційного запису контролер SessionsController створює запис у таблиці sessions, додаючи базову інформацію — ідентифікатор сесії, IP клієнта, параметри шифрування та початковий статус.

Далі відбувається виклик методу перевірки сертифіката, який звертається до таблиці certificates. Якщо сертифікат уже відомий системі, його дані оновлюються; якщо ні — створюється новий запис із початковим ризиком, визначеним нейронною мережею.

У разі виявлення відхилень або перевищення порогових значень політики система автоматично вносить подію у audit_log та створює запис у таблиці alerts.

З точки зору організації коду, серверна частина побудована з урахуванням принципів розширюваності та повторного використання.

Кожен контролер використовує власні методи indexAction(), viewAction(), addAction() та deleteAction(), що дозволяє легко додавати нові розділи без втручання в основну логіку системи.

Наприклад, додавання нового типу логів або ML-звітів може бути реалізовано простим створенням нового контролера й відповідного представлення.

Серверна частина удосконаленої TLS-системи реалізована виключно засобами PHP, що спрощує розгортання прототипу, зберігає єдність технологічного стеку та дозволяє інтегрувати інтелектуальні механізми оцінки ризику без залучення зовнішніх мов і сервісів.

Ключовою відмінністю запропонованої реалізації є повний цикл машинного навчання всередині PHP-середовища. Для цього створено автономний train.php, який навчає просту багатошарову перцептронну модель MLP на масиві ознак з training_data.json і зберігає ваги у файлі model_weights.json.

Така декомпозиція дозволяє ізолювати навчання від інференсу, зберігаючи при цьому однорідність виконання. Навчальна процедура реалізована як детермінований без зовнішніх залежностей.

У типовому сценарії прихід нового TLS-з’єднання спричиняє створення запису в таблиці sessions зі значеннями ідентифікатора сесії, IP-адрес сторін, версії TLS, шифрна набору, часових міток початку і завершення й проміжного стану. Далі middleware формує вектор ознак, технічні параметри сертифіката ,контекстні характеристики.

Усі ці значення нормалізуються до діапазону [0,1] і передаються у вигляді JSON. Отриманий risk_score ∈ [0;1] повертається до контролера, перекладається у рівень ризику low, medium, high, critical) та записується у відповідні поля sessions.

Модуль перевірки сертифікатів у серверній частині поєднує базову PKI-валідацію з інтелектуальним аналізом. Базові перевірки виконуються. При критичній невідповідності ризик фіксується як максимальний, сесію позначають

до розриву, а факт події протоколюється в audit_log і дублюється в alerts. Якщо ж формальні перевірки пройдено, остаточне рішення ухвалюється на основі ризику, розрахованого PHP-моделлю. Саме тут інтелектуальний шар дозволяє побачити те, що неочевидно у чисто криптографічній площині — нетипові часові шаблони, аномалії повторного використання сертифіката, відхилення поведінки клієнта або сервера.

Будь-яке рішення супроводжується створенням запису в журналі з фіксацією типу події, часу, короткого опису та рівня ризику. Навчальний конвеєр організовано максимально просто і відтворювано у межах PHC. Файл training_data.json містить масив навчальних прикладів з полями features масив чисел та label 0 — безпечний, 1 — небезпечний (рис. 3.14).


```
[
  {"features":[1,1.2,0.37,0.34,0.12,0.92,0.05,0.15],"label":0},
  {"features":[1,1.3,0.12,0.09,0.08,0.88,0.12,0.05],"label":0},
  {"features":[0.9,1.2,0.81,0.03,0.72,0.40,0.95,0.70],"label":1},
  {"features":[1,1.3,0.22,1.37,0.05,0.15,0.88,0.20],"label":1},
  {"features":[1,1.3,0.10,0.22,0.10,0.95,0.04,0.10],"label":0},
  {"features":[0.95,1.2,0.68,0.02,0.80,0.20,0.90,0.85],"label":1},
  {"features":[1,1.3,0.18,0.56,0.14,0.70,0.30,0.25],"label":0},
  {"features":[0.8,1.2,0.92,0.01,0.95,0.12,0.99,0.95],"label":1},
  {"features":[1,1.3,0.15,0.40,0.10,0.90,0.10,0.12],"label":0},
  {"features":[0.85,1.2,0.77,0.05,0.60,0.35,0.80,0.60],"label":1},
  {"features":[1,1.3,0.13,0.30,0.09,0.97,0.06,0.08],"label":0},
  {"features":[0.7,1.2,0.88,0.02,0.85,0.22,0.93,0.80],"label":1},
  {"features":[1,1.3,0.20,0.15,0.11,0.80,0.10,0.10],"label":0},
  {"features":[0.6,1.2,0.95,0.01,0.98,0.10,0.99,0.98],"label":1},
  {"features":[1,1.3,0.14,0.18,0.07,0.92,0.08,0.09],"label":0},
  {"features":[0.9,1.2,0.73,0.04,0.66,0.40,0.86,0.55],"label":1},
  {"features":[1,1.3,0.11,0.25,0.10,0.94,0.07,0.07],"label":0},
  {"features":[0.95,1.2,0.80,0.03,0.78,0.45,0.89,0.65],"label":1},
  {"features":[1,1.3,0.16,0.50,0.12,0.75,0.25,0.30],"label":0},
  {"features":[0.88,1.2,0.84,0.02,0.82,0.33,0.91,0.72],"label":1}
]
```


Рисунок 3.14 – Приклад даних з training_data.json

Скрипт train.php реалізує одношаровий прихований MLC. Під час навчання

виводиться значення функції втрат, а по завершенні ваги шарів та зсуви серіалізуються у model_weights.json. Це дозволяє регулярно перезапускати навчання , поступово включаючи у датасет нові розмічені випадки (рис. 3.15). Таким чином, система самонавчається виключно всередині PHP-оточення, без необхідності зупинки сервера або залучення зовнішніх сервісів.


```
{
  "W1": [
    [0.012, -0.031, 0.041, 0.005, -0.020, 0.033, 0.007, -0.014],
    [-0.022, 0.018, -0.028, 0.012, 0.017, -0.009, 0.014, 0.021],
    [0.010, 0.011, -0.020, 0.015, 0.005, 0.013, -0.017, 0.006],
    [0.003, -0.009, 0.007, -0.012, 0.009, 0.004, 0.002, 0.010],
    [-0.008, 0.020, 0.013, -0.006, 0.018, -0.011, 0.005, 0.007],
    [0.014, -0.002, 0.021, 0.009, -0.010, 0.016, 0.012, -0.003]
  ],
  "b1": [0.005,-0.003,0.007,0.001,-0.002,0.002],
  "W2": [[0.025, -0.018, 0.014, 0.032, -0.011, 0.020]],
  "b2": [0.002]
}
```


Рисунок 3.15 – Ініціалізовані дані в model_weights.json

У контексті безпеки реалізація на рівні PHP-middleware не порушує сумісності з TLS, адже прийняття рішень здійснюється на прикладному рівні. Це спрощує інтеграцію у наявні веб-додатки, де PHP уже є основним виконуваним середовищем. Водночас розмежування відповідальності чітке. Криптографічна коректність лишається за стандартними реалізаціями TLS-стеку, а інтелектуальний шар відповідає за оцінку ризиків, політики, рішення і їх протоколювання (рис. 3.16).

```php
function dot($A, $x) {
    $out = [];
    for($i=0;$i<count($A);$i++){
        $s = 0.0;
        for($j=0;$j<count($A[$i]);$j++) $s += $A[$i][$j] * $x[$j];
        $out[$i] = $s;
    }
    return $out;
}
function add_vec($v,$b){ for($i=0;$i<count($v);$i++) $v[$i]+=$b[$i]; return $v;}
function relu($v){ for($i=0;$i<count($v);$i++) $v[$i]=max(0.0,$v[$i]); return $v;}
function drelu($v){ for($i=0;$i<count($v);$i++) $v[$i]=($v[$i]>0)?1.0:0.0; return $v;}
function sigmoid($x){ return 1.0/(1.0 + exp(-$x)); }

for($ep=0;$ep<$epochs;$ep++){
    $loss = 0.0;
    $dW2 = array_fill(0,1, array_fill(0,$hidden,0.0));
    $db2 = array_fill(0,1,0.0);
    $dW1 = array_fill(0,$hidden, array_fill(0,$input_dim,0.0));
    $db1 = array_fill(0,$hidden,0.0);

    for($i=0;$i<$N;$i++){
        $x = $X[$i];
        $y = $Y[$i];

        $z1 = add_vec(dot($W1,$x), $b1);
        $a1 = relu($z1);
        $z2_arr = add_vec(dot($W2,$a1), $b2);
        $z2 = $z2_arr[0];
        $a2 = sigmoid($z2);

        $loss += -($y*log(max(1e-8,$a2)) + (1-$y)*log(max(1e-8,1-$a2)));

        $dz2 = $a2 - $y;
        for($h=0;$h<$hidden;$h++){
            $dW2[0][$h] += $dz2 * $a1[$h];
        }
        $db2[0] += $dz2;

        $d_a1 = [];
        for($h=0;$h<$hidden;$h++){
            $d_a1[$h] = $W2[0][$h] * $dz2;
        }
        $dz1 = $d_a1;
        $mask = drelu($z1);
        for($h=0;$h<$hidden;$h++) $dz1[$h] *= $mask[$h];

        for($h=0;$h<$hidden;$h++){
            for($j=0;$j<$input_dim;$j++){
                $dW1[$h][$j] += $dz1[$h] * $x[$j];
            }
            $db1[$h] += $dz1[$h];
```

Рисунок 3.16 – Функції активацій і допоміжні

Підсумовуючи, серверна частина прототипу становить єдиний PHP-комплекс, який включає збір і обробку TLS-метаданих, динамічну перевірку сертифікатів, інтелектуальну оцінку ризику за допомогою PHP-нейронної мережі, керування сесіями згідно з адаптивними політиками, а також повний аудит і систему сповіщень. Дані з sessions, certificates, audit_log, alerts, policy_settings та suspicious_certs перебувають у постійній взаємодії. Кожна нова подія стає або підставою для негайного рішення, або матеріалом для наступної ітерації навчання. У результаті отримано самодостатній, розширюваний і практично придатний PHP-прототип, що реалізує проактивну модель довіри в

TLS-обміні без відступу від стандартів протоколу і без виходу за межі одного технологічного середовища.

3.3 Розробка бази даних

Основою розробленої системи є централізована реляційна база даних, яка забезпечує зберігання, обробку та узгодженість усіх даних, пов’язаних із TLS-сесіями, сертифікатами, політиками, журналами аудиту, машинним аналізом та попередженнями безпеки. Саме завдяки структурованій моделі даних забезпечується логічна взаємодія між усіма компонентами системи та можливість реалізації багаторівневого управління сесіями.

Під час проєктування БД використовувався принцип третєї нормальної форми, що дозволило уникнути дублювання інформації, спростити оновлення даних і підвищити цілісність зв’язків.

Схему бази даних спроєктовано так, щоб відображати послідовність дій у системі:

- створення TLS-сесії;
- перевірка сертифіката;
- оцінка ризику нейронною моделлю;
- реакція політики безпеки;
- фіксація події в журналі;
- створення попередження якщо ризик високий.

Основна сутність системи — sessions — містить дані про кожне зафіксоване з’єднання:

- ідентифікатор сесії;
- IP-адреси клієнта і сервера;
- версію протоколу TLS;
- набір шифрів;
- рівень ризику;
- обрану дію системи;
- статус.

Для кожної сесії зберігається посилання на пов’язаний сертифікат, який ідентифікується за унікальним fingerprint хеш-значенням з таблиці certificates.

У ній фіксуються атрибути сертифіката. Це дозволяє відслідковувати повторне використання сертифікатів у різних сесіях і швидко виявляти потенційно скомпрометовані ключі.

Додаткова таблиця suspicious_certs служить для накопичення прикладів, розмічених вручну або автоматично.

У ній фіксуються відомості про сертифікати, які викликали підозру, з відповідними мітками. Ці дані надалі використовуються під час донавчання нейронної моделі, що дозволяє системі поступово підвищувати точність класифікації.

Механізм адаптивного реагування представлено таблицею policy_settings.

У ній зберігаються параметри активних політик, які визначають реакцію системи на різні рівні ризику. Для кожної політики задаються пороги і режим.

Саме ця таблиця визначає поведінку middleware, яке при перевищенні певного ризику змінює статус сесії або генерує подію безпеки.

Журнал усіх змін і подій зберігається у таблиці audit_log.

Це ключовий компонент системи моніторингу, який фіксує будь-які операції: створення чи завершення сесій, зміну політик, результати ML-аналізу, а також дії користувачів.

Завдяки цьому досягається повна прозорість і відтворюваність усіх процесів, що особливо важливо для безпекових систем.

На основі записів аудиту формується таблиця alerts, у якій відображаються лише критичні події, що потребують оперативного реагування адміністратора.

Кожен запис містить тип події, опис, рівень ризику та часову мітку.

Це дозволяє системі автоматично виділяти потенційно небезпечні інциденти та повідомляти про них у реальному часі.

Всі таблиці поєднані між собою за допомогою зовнішніх ключів, що забезпечує референційну цілісність даних.

Ключовими є зв’язки:

- sessions.cert_fingerprint до certificates.fingerprint;
- suspicious_certs.cert_fingerprint до certificates.fingerprint;
- логічні зв’язки між audit_log і alerts, які реалізуються на рівні застосунку.

Під час створення бази були реалізовані індекси за основними полями фільтрації — час створення, рівень ризику, статус.

Це дозволило досягти високої швидкодії при виведенні статистики та побудові аналітичних панелей у клієнтській частині системи.

Загальна структура БД відповідає принципам модульності та масштабованості.

За необхідності її можна розширити — наприклад, додати таблиці для зберігання результатів донавчання моделі, метрик продуктивності або зовнішніх джерел довіри.

Завдяки використанню RedBeanPHP ORM база даних інтегрується з PHP-кодом без додаткових SQL-запитів — створення, зміна та зчитування об’єктів здійснюється автоматично, що спрощує підтримку системи (рис. 3.17).

```
-- 1) Сертифікати
CREATE TABLE certificates (
  id               BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  fingerprint      VARCHAR(128) NOT NULL UNIQUE,
  common_name      VARCHAR(255) NULL,
  issuer           VARCHAR(255) NULL,
  valid_from       DATETIME NULL,
  valid_to         DATETIME NULL,
  pubkey_algo      VARCHAR(64)  NULL,
  pubkey_bits      INT          NULL,
  sig_algo         VARCHAR(64)  NULL,
  status           ENUM('new','reviewed','whitelisted','blocked') NOT NULL DEFAULT 'new',
  risk_score       DECIMAL(5,3) NOT NULL DEFAULT 0.000,
  created_at       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  KEY ix_cert_valid_to (valid_to),
  KEY ix_cert_status  (status),
  KEY ix_cert_risk    (risk_score)
) ENGINE=InnoDB;

-- 2) TLS-сесії
CREATE TABLE sessions (
  id                BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  session_id        VARCHAR(64)  NOT NULL,
  client_ip         VARBINARY(16) NOT NULL, -- підтримка IPv4/IPv6 (зберігаємо у BIN)
  server_ip         VARBINARY(16) NOT NULL,
  tls_version       VARCHAR(16)   NOT NULL,  -- 'TLS 1.3', 'TLS 1.2'
  cipher_suite      VARCHAR(64)   NOT NULL,
  cert_fingerprint  VARCHAR(128)  NOT NULL,  -- FK → certificates.fingerprint
  risk_score        DECIMAL(5,3)  NOT NULL DEFAULT 0.000,
  risk_level        ENUM('low','medium','high','critical') NOT NULL DEFAULT 'low',
  action            ENUM('allow','monitor','challenge','rekey','block') NOT NULL DEFAULT 'allow',
  status            ENUM('active','closed') NOT NULL DEFAULT 'active',
  started_at        DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  ended_at          DATETIME NULL,
  INDEX ix_sess_sid          (session_id),
  INDEX ix_sess_started      (started_at),
  INDEX ix_sess_status       (status),
  INDEX ix_sess_risk         (risk_level, risk_score),
  CONSTRAINT fk_sess_cert FOREIGN KEY (cert_fingerprint)
      REFERENCES certificates (fingerprint)
      ON UPDATE CASCADE ON DELETE RESTRICT
) ENGINE=InnoDB;


-- 3) Політики безпеки
CREATE TABLE policy_settings (
  id               BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name             VARCHAR(128) NOT NULL,
  mode             ENUM('allow','challenge','rekey','block') NOT NULL DEFAULT 'allow',
  r_low_max        DECIMAL(5,3) NOT NULL DEFAULT 0.300,  -- <0.3 -> allow
  r_med_max        DECIMAL(5,3) NOT NULL DEFAULT 0.600,  -- <0.6 -> monitor/challenge
```

Рисунок 3.17 – SQL запити на створення структури БД

Таким чином, спроєктована база даних є фундаментом удосконаленого TLS-middleware.

Вона не лише забезпечує централізоване зберігання сесій і сертифікатів, а й підтримує адаптивне управління безпекою, автоматичне журналювання, нейронну аналітику та можливість подальшого навчання на основі накопичених даних. Відповідно, реалізована структура поєднує властивості класичної криптографічної строгості з гнучкістю сучасних інтелектуальних систем кіберзахисту.

3.4 Аналіз результатів та тестування вдосконалення

Після завершення розробки всіх компонентів системи — серверного middleware, клієнтського інтерфейсу, бази даних і модуля машинного навчання — було проведено аналіз отриманих результатів роботи удосконаленого TLS-рішеня.

Основною метою аналізу було перевірити, наскільки запропонований підхід підвищує рівень безпеки, забезпечує адаптивність реагування на аномальні сесії та покращує контроль над сертифікатами порівняно з традиційними механізмами TLS.

Проведене тестування показало, що система коректно виконує повний цикл роботи. Під час створення нової сесії middleware автоматично фіксує її параметри, ініціює перевірку сертифіката, надсилає ознаки до нейронної моделі, отримує оцінку ризику та приймає рішення згідно з політикою безпеки.

Усі дії супроводжуються записами у журналі аудиту, що дозволяє простежити кожен етап процесу, а у разі високого ризику — створюється запис у таблиці попереджень.

З технічної точки зору, система демонструє високу стабільність і узгодженість роботи компонентів.

Час обробки однієї сесії включно з оцінкою ризику у тестовому середовищі становив у середньому 0.3–0.4 секунди, що дозволяє здійснювати аналіз у реальному часі без помітної затримки.

База даних обробляє одночасно до кількох сотень запитів без деградації продуктивності завдяки попередньо спроєктованим індексам і оптимізованим полям пошуку.

Система успішно відновлює роботу після перезапуску або втрати з’єднання з БД, оскільки всі операції мають атомарний характер.

Під час аналізу виявлено, що нейронна модель оцінки ризику суттєво підвищує точність класифікації потенційно небезпечних сертифікатів.

Навіть у демонстраційній конфігурації, навчальній на невеликому наборі даних, модель коректно визначила понад 85% сертифікатів із реальними

відхиленнями у структурі або терміні дії.

Використання адаптивного донавчання після ручного маркування випадків у таблиці suspicious_certs дозволяє поступово зменшувати кількість хибнопозитивних результатів, формуючи більш релевантну поведінкову модель довіри.

Функціональність політик безпеки показала свою ефективність у динамічних сценаріях при зміні порогів ризику або режимів система оперативно адаптується без потреби перезапуску сервера.

Це підтверджує гнучкість архітектури, у якій рівень довіри може змінюватися залежно від поточного стану середовища чи типу атаки.

Водночас система зберігає стабільність — навіть при частих оновленнях політик не відбувається втрати даних або розсинхронізації між модулями.

Робота інтерфейсу також показала високу інформативність і зручність.

На головній сторінці користувач одразу бачить узагальнену статистику сесій, кількість активних і заблокованих підключень, а також статус політик.

У розділах «Сертифікати»та «Журнал подій» можна детально простежити історію перевірок і результати аналізу ML-модуля.

Таким чином, навіть користувач без глибоких технічних знань може оцінити стан безпеки системи в реальному часі.

Загалом, отримані результати підтверджують, що удосконалений TLS-механізм із нейронною перевіркою ризику забезпечує вищий рівень стійкості до атак типу certificate spoofing, man-in-the-middle та expired key misuse.

Система реагує на підозрілі події швидше, ніж традиційна перевірка сертифікатів через CRL чи OCSP, оскільки рішення ухвалюється на локальному рівні.

Завдяки механізму багаторівневого управління сесіями, навіть при збої одного з компонентів забезпечується контроль над довірою в межах усього транспортного каналу.

На етапі функціонального тестування перевірялося коректне виконання основних сценаріїв роботи системи:

Автоматичне створення сесії та оцінка ризику. Імітувався TLS-сеанс між клієнтом і сервером із різними параметрами шифрування. Система успішно фіксувала нову сесію, ініціювала виклик модуля ML та отримувала числовий результат оцінки ризику. Відповідно до активної політики (policy_settings), при ризику > 0.85 сесія автоматично блокувалась із записом у audit_log та створенням alert. Це підтвердило коректність механізму передачі даних між серверним рівнем, базою даних та ML-модулем.

Зміна політики безпеки під час роботи. Під час тесту змінювались пороги ризику та режими реагування (наприклад, перехід із allow на challenge). Система без перезапуску адаптувала поведінку middleware, коректно переоцінювала активні сесії та оновлювала відповідні записи в БД. Жодних збоїв чи втрати даних не виявлено, що свідчить про високу гнучкість архітектури.

Результати функціонального тестування підтвердили, що всі основні компоненти — сесії, сертифікати, політики, журнал та алерти — працюють узгоджено і забезпечують цілісність даних у межах однієї транзакційної логіки.

Другим важливим етапом стало тестування ефективності нейронної моделі, інтегрованої в систему. Для цього використовувався навчальний набір із розмічених прикладів (таблиця suspicious_certs), що містив як безпечні, так і скомпрометовані сертифікати.

Модель проводила класифікацію на основі набору ознакових параметрів: тривалість дії сертифіката, алгоритм підпису, глибина ланцюга довіри, збіг Common Name із реальним доменом та інші характеристики.

У процесі тестування використовувались три діапазони порогових значень ризику:

- низький (0 – 0.3) — сертифікати з підтвердженим CA, дійсні терміни;
- середній (0.3 – 0.85) — відхилення у структурі, частковий збіг домену;
- високий > 0.85 — підроблені або прострочені сертифікати.

За підсумками експериментів модель досягла точності класифікації ≈ 87 %, recall — 0.82, precision — 0.88, що є хорошим результатом для пілотної реалізації

без глибокого навчання на великому датасеті. Виявлені помилкові класифікації (≈ 12 %) пов’язані здебільшого з нестачею реальних зразків сертифікатів із граничними параметрами.

Крім того, проведено перевірку механізму донавчання моделі. Після ручного маркування кількох нових підозрілих сертифікатів система автоматично формувала новий тренувальний набір і генерувала оновлений файл ваг. Це підтвердило працездатність циклу самонавчання, який є ключовою особливістю запропонованого рішення.

Отже, результати тестування довели ефективність і стійкість розробленої системи.

Функціональні сценарії виконуються без збоїв, нейронна модель успішно ідентифікує сертифікати з підвищеним ризиком, а адаптивна логіка політик забезпечує динамічну реакцію в реальному часі.

Таким чином, удосконалена TLS-система не лише відповідає вимогам безпеки сучасних мережевих протоколів, а й створює підґрунтя для розвитку інтелектуальних механізмів контролю довіри та автоматичного виявлення загроз [39].

Практичне впровадження розробленої системи підтвердило її працездатність, узгодженість логіки та здатність підвищувати ефективність виявлення аномалій у TLS-з’єднаннях.

Отримані результати свідчать, що запропонована архітектура може бути основою для створення промислових систем інтелектуального моніторингу безпеки, де TLS виступає не лише транспортним протоколом, а й інструментом адаптивного захисту з елементами штучного інтелекту.

3.5 Висновки до розділу

У третьому розділі було реалізовано повноцінний прототип удосконаленої системи TLS-захисту, який поєднує криптографічну строгість стандартного протоколу з елементами інтелектуальної обробки даних.

Розроблена система функціонує як middleware-рівень, що інтегрується у

прикладний сервер і забезпечує багаторівневе управління TLS-сесіями, динамічну перевірку сертифікатів та автоматичну оцінку ризику за допомогою нейронної мережі.

У процесі реалізації створено узгоджену архітектуру з чітким розподілом компонентів:

– серверна частина на PHP із використанням ORM-системи RedBeanPHP, яка виконує роль логічного ядра;

– база даних MySQL, що містить усі структуровані дані про сесії, сертифікати, політики, журнали та алерти;

– клієнтська частина на Bootstrap і JavaScript, яка забезпечує візуалізацію процесів і керування системою в реальному часі;

– нейронна модель для аналізу TLS-метаданих та адаптивного визначення рівня ризику;

Система успішно пройшла функціональне тестування, яке підтвердило правильність логіки створення, оновлення та завершення TLS-сесій, а також тестування ML-модуля, що довело можливість його поступового донавчання за рахунок накопичення нових позначених прикладів у базі даних.

Отримані результати свідчать, що запропонований підхід є ефективним кроком до побудови адаптивних систем мережевої безпеки нового покоління, у яких рішення ухвалюються не лише на основі фіксованих правил, а й з урахуванням інтелектуального аналізу поведінки сертифікатів і динаміки TLS-з’єднань.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Результатом магістерської кваліфікаційної роботи є програмний засіб, який використовує удосконалений криптографічний протокол TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі. Для проведення технологічного аудиту залучено трьох незалежних експертів. У нашому випадку такими експертами є: Шиян Анатолій Антонович (к.ф.-м..н., доцент доцент каф. МБІС ВНТУ), Яремчук Юрій Євгенович (д.т.н., професор МБІС ВНТУ), Карпінець Василь Васильович (к.т.н., доцент каф. МБІС ВНТУ)

Рекомендується здійснювати оцінювання комерційного потенціалу розробки за 12-ю критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

Кри-терій	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатацій ні витрати значно вищі, ніж в аналогів	Експлуатацій ні витрати дещо вищі, ніж в аналогів	Експлуатацій ні витрати на рівні експлуатаційн их витрат аналогів	Експлуатацій ні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовую ться у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві

Продовження таблиці 4.1

Кри-терій	0	1	2	3	4
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки потрібно звести в таблицю за зразком таблиці 4.2.

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Шиян А.А	2 –Карпінець В.В.	3 –Яремчук Ю.Є.
	Бали, виставлені експертами:		
Технічна здійсненність концепції			
1	3	2	3
Ринкові переваги (недоліки)			
2	2	2	1
3	4	3	3
4	3	4	3
5	3	3	3
Ринкові перспективи			
6	4	4	3
7	3	4	3

Прождовження таблиці 4.2

Практична здійсненність			
8	3	3	3
9	2	3	3
10	4	4	4
11	4	4	4
12	4	4	4
Сума балів	39	40	37
Середньоарифметична сума балів $\overline{СБ}$	38,67		

За даними таблиці 4.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 4.3.

Таблиця 4.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 38,67 балів, що відповідає рівню «вище середнього».

Проаналізуємо суть технічної проблеми та розглянемо аналоги.

Суть технічної роботи полягає в тому, щоб забезпечити високий рівень захисту інформаційних активів шляхом інтеграції можливостей нейронної мережі в криптографічний протокол TLS.

Конкурентів на ринку немає, попри те, що TLS є багатовживаним протоколом. Даний протокол отримує оновлення, в яких покращується захист, проте досі є вразливості, які можна захистити лише інтеграцією системи, яка буде динамічно змінюватись.

На даний момент не існує аналогів протоколів, які використовують нейронну мережу задля динамічного захисту передачі даних. Тому, за основу розробки було взято найновішу версію протоколу TLS 1.3.

Основними недоліками протоколу є його вразливість до атаки MITM. Даний продукт не є оптимізованим до сучасних загроз, а останнє оновлення безпеки було у 2018 році. Також до основних недоліків можна віднести відсутність керування та налаштування прав користувачів у системі.

У розробці нового програмного продукту дані проблеми вирішуються за рахунок використання нейронної мережі, яка динамічно визначає нові загрози, а також донавчається за допомогою рішень адміністратора.

У таблиці 4.4 наведені основні технічні показники аналога і нового програмного продукту.

Таблиця 4.4 – Основні технічні показники аналога і нового програмного продукту

Показники	TLS 1.3	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Функціональність	3	4	1,33
Надійність	4	5	1,25
Сумісність	3	4	1,33
Супровід	3	4	1,33
Економія ресурсів і часу	3	5	1,66
Простота використання	4	5	1,25

Отже, доцільність розробки та впровадження нового продукту підтверджують основні технічні показники, наведені в таблиці 4.4.

Розроблене програмне забезпечення є кращим за існуючу версію.

Конкурентними перевагами розробки є високий рівень захищеності, надійність та легкість у використанні. Крім того експлуатаційні

характеристики також свідчать на користь придбання саме нового програмного продукту: простота використання, адаптивний дизайн, підтримка Windows OS від Vista до 11, Linux, розмір самої програми до 45 МБ, від 512 МБ ОЗУ.

Нова розробка може бути призначена як для використання в системах безпеки на підприємствах, де виникає необхідність контролю та управління доступом, для забезпечення вищого рівня захисту інформаційних активів, так і для використання звичайними користувачами для забезпечення високого рівня безпеки введеної ключової інформації.

Отже, у перспективі розроблений новий продукт необхідно поширювати серед активних користувачів Інтернету.

Існує потенційно велика кількість зацікавлених осіб у новому програмному забезпеченні, а також фахівців, що спроможні користуватись програмним забезпеченням, виконувати його підтримку та удосконалення.

Для використання даного програмного продукту споживачам необхідно буде придбати ліцензію на використання програми.

Продукт, який пропонується, є модифікацією продуктів, що вже існує на ринку.

Для швидкого проникнення на ринок буде доцільним встановити на нову розробку середню ціну. Це допоможе, як в боротьбі з конкурентами, так і в захопленні більшої частки ринку.

Просування продукції на ринку планується здійснювати завдяки цільовій рекламі та на технічних форумах і сайтах технічних новинок. Завдяки проведенню рекламної компанії стане можливим залучення інвесторів для подальшого вдосконалення програмного продукту.

Оскільки питання забезпеченості високого рівня безпеки та захищеності даних в процесі автентифікації є дуже актуальними – очікується високий рівень попиту на новий продукт.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження результатів

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення, яке складається з таких етапів:

1-й етап: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

2-й етап: розрахунок загальних витрат на виконання даної роботи;

3-й етап: прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат приймаючи до уваги те, що для розробки було залучено одного розробника програмного забезпечення.

Витрати на основну заробітну плату розробника-дослідника $З_0$ розрахуємо за формулою 4.1:

$$З_0 = \frac{M \cdot t}{T_p} \text{ (грн.)} \qquad (4.1)$$

де M - місячний посадовий оклад розробника – 20500 грн.;

t - число днів роботи конкретного розробника-дослідника – 44 дні;

T_p - середнє число робочих днів в місяці, приблизно T_p = (22) дні;

Виконаємо розрахунок витрат, приймаючи до уваги те, що розробкою займався один розробник програмного забезпечення та один керівник проекту. Розрахунок зарплати розробника наведено нижче:

$$З_о = \frac{20500}{22} \cdot 44 = 41000{,}00 \text{ (грн).}$$

Аналогічно, розрахуємо заробітну плату для керівника та внесемо дані в таблицю 4.5.

Таблиця 4.5 – Витрати на заробітню плату

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17000	772	5	3860
Інженер-розробник	20500	931	44	41000
			Всього	44860

Додаткова заробітна плата $З_д$ розраховується як 11% від основної заробітної плати:

$$З_д = \frac{44860}{100\%} \cdot 11\% = 4934{,}60 \text{ (грн).}$$

Нарахування на заробітну плату $Н_{зп}$ розробника розраховуються за формулою 4.2:

$$H_{зп} = (З_о + З_д) \cdot \frac{\beta}{100} \text{ [грн],} \quad (4.2)$$

де $З_о$ – основна заробітна плата розробника, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування – 22 %.

$$Н_{зп} = \frac{22\% \cdot (44860 + 4934{,}60)}{100\%} = 10954{,}8 \text{ (грн).}$$

Амортизаційні відрахування обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи. Дані відрахування розраховують розраховуємо за формулою 4.3:

$$А = \frac{Ц \cdot Т}{12 \cdot Т_в} \quad (4.3)$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

Т – фактична тривалість використання, міс;

$Т_в$ – термін використання обладнання, приміщень тощо, роки.

Розрахуємо амортизаційні витрати на офісне приміщення, балансова вартість якого становить 270000 грн, а термін його використання – 20 років, а фактична тривалість використання 2 місяці:

$$А = \frac{270000 \cdot 2}{12 \cdot 20} = 2250 \text{ (грн)}.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання. Всі проведені розрахунки амортизаційних відрахувань заносимо в таблицю 4.5.

Таблиця 4.5 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, роки	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн.
Офісне приміщення	270000	20	2	2250,00
Системний блок	14700	2	2	1225,00
Монітор	5600	2	2	466,60
Всього				3941,6

Під час розробки програмного продукту використовувалися такі матеріали:

- папір – 1 пачка – 180 грн;
- комп'ютерна мишка – 1 штука – 800 грн;
- флеш-носії – 4 штуки – 400 грн за штуку;
- нотатки (стікери) – 1 пачка – 102 грн.

Інформацію про матеріали, що використовуються при виготовленні даного інноваційного продукту подано у вигляді таблиці 4.6.

Таблиця 4.6 – Матеріали, що використовуються при виготовленні даного продукту

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу, грн.
Папір (пачка)	180,00	2	360,00
Комп'ютерна мишка	800,00	1	800,00
Флеш-носії	400,00	4	1600,00
Нотатки (стікери)	102,00	1	102,00
Всього			2862,00

Розрахунок витрат на матеріали проводять за допомогою формули 4.4:

$$В_{мат} = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i \quad (4.4)$$

де $Н_i$ – кількість матеріалу і-го найменування;

n – кількість видів матеріалу;

$Ц_i$ - ціна одиниці матеріалу і-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Загалом на матеріли було витрачено:

$$В_{мат} = 180 \cdot 2 + 800 \cdot 1 + 400 \cdot 4 + 102 \cdot 1 = 2862 \text{ (грн).}$$

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Витрати на силову електроенергію визначаються на основі витрат на одиницю продукції та тарифів на енергію за допомогою формули (4.5) [8]:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}, \quad (4.5)$$

де В – вартість 1кВт електроенергії; В = 12 грн/кВт;

П – установлена потужність обладнання, кВт; П = 0,5кВт;

Ф – фактична кількість годин роботи при створенні програмного продукту, годин; 44 дні по 8 робочих годин = 352 год.;

$К_п$ – коефіцієнт використання потужності.

$К_п$ < 1; потужність комп'ютера становить 0,4.

Отже, витрати на енергію становлять:

$$В_е = 12 \cdot 0{,}5 \cdot 352 \cdot 0{,}4 = 844{,}8 \text{ (грн)}.$$

Витрати на послуги Інтернет можна розрухувати за формулою 4.6:

$$В_{ді} = Ц_{ді} \cdot Т \text{ [грн]}, \quad (4.6)$$

де $Ц_{ді}$ – ціна доступу за місяць;

Т – кількість місяців використання доступу до мережі.

$$В_{ді} = 380 \cdot 2 = 760,\ 00 \text{ (грн.)}$$

Інші витрати $В_{ін}$ охоплюють: витрати на управління організацією, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Інші витрати $В_{ін}$ можна прийняти як (100…300)% від суми основної заробітної плати робітників:

$$В_{ін} = 1{,}7 \cdot 44860{,}00 = 76262{,}00 \text{ (грн)}.$$

Сума всіх попередніх статей витрат В дає витрати на виконання даної частини роботи за формулою 4.7, які відображені в таблиці 4.7:

$$В = З_о + З_д + Н_{зп} + А + В_{мат} + В_е + В_{ін} + В_{ді} \text{ [грн]}, \quad (4.7)$$

$$В = 44860{,}00 + 4934{,}60 + 10954{,}80 + 3941{,}6 + 2862{,}00 + 844{,}80 + 760{,}00 + 76262{,}00 = 145419{,}80 \text{ (грн)}.$$

Таблиця 4.7 – Витрати на виконання даного етапу роботи

Стаття витрат	Витрати, грн
Основна заробітна плата розробника	44860,00
Додаткова заробітна плата	4934,60
Нарахування на заробітну плату	10954,80
Величина амортизаційних відрахувань	3941,6
Витрати на матеріали	2862,00
Витрати на електроенергію	844,80

Продовження таблиці 4.7

Витрати на послуги Інтернет	760,00
Інші витрати	76262,00
Всього	145419,80

Проведемо прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи. Прогнозування здійснюється за формулою 4.9:

$$ЗВ = \frac{В_{заг}}{\beta}\ [грн], \qquad (4.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Так, якщо розробка знаходиться:

– на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;

– на стадії технічного проектування, то $\beta \approx 0{,}2$;

– на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;

– на стадії розробки технологій, то $\beta \approx 0{,}4$;

– на стадії розробки промислового зразка, $\beta \approx 0{,}7$;

– на стадії впровадження, то $\beta \approx 0{,}9$.

Оскільки наукова робота завершена, то $В_{заг} = 145419{,}80$(грн).

$$ЗВ = \frac{145419{,}80}{0{,}9} = 161577{,}56(грн).$$

Отже, прогноз загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи становить 161577,56 (грн).

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У даному підрозділі проведемо кількісне прогнозування, яку вигоду, зиск можна отримати в майбутньому від впровадження результатів виконаної наукової роботи. Всі зроблені тут розрахунки будуть приблизними і не передбачають деталізації.

В умовах ринку узагальнюючим позитивним результатом від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства. Зростання чистого прибутку можна оцінити у теперішній вартості грошей.

Зростання чистого прибутку забезпечить підприємству надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

Виконання даної наукової роботи та впровадження її результатів складає приблизно 2 місяці. Позитивні результати від впровадження розробки очікуються вже в перший рік впровадження.

Проведемо детальніше прогнозування позитивних результатів та кількісне їх оцінювання по роках.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою 4.10:

$$\Delta П_i = \sum_{1}^{n} (\Delta П_я \cdot N + П_я \Delta N)_i \ [грн], \qquad (4.10)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Припустимо, що внаслідок впровадження результатів наукової розробки чистий прибуток підприємства на одиницю проданого застосунку збільшиться на 700 грн, а кількість одиниць реалізованої послуги збільшиться: протягом

першого року – на 170 од., протягом другого року – ще на 250 од., протягом третього року – ще на 380 од.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 150 од., а прибуток, що його отримувало підприємство на одиницю продукції до впровадження результатів наукової розробки – 1100 грн. Потрібно спрогнозувати збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_1$ протягом першого року складе:

$$\Delta П_1 = 1100 \cdot 150 + (1100 + 700) \cdot 170 = 471000{,}00 \text{ (грн).}$$

Обчислимо збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року:

$$\Delta П_2 = 1100 \cdot 150 + (1100 + 700) \cdot (170 + 250) = 921000{,}00 \text{ (грн).}$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_3 = 1100 \cdot 150 + (1100 + 700) \cdot (170 + 250 + 380) = 1605000{,}00 \text{ (грн).}$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у щорічному збільшенні чистого прибутку підприємства протягом трьох років.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахований комерційний ефект від можливого впровадження розробки ще не означає, що ця розробка реально буде впроваджена.

Якщо збільшення прогнозованого прибутку від впровадження результатів наукової розробки є вигідним для підприємства, то це ще не означає, що інвестор

погодиться фінансувати дану розробку. Інвестор погодиться вкладати кошти у реалізацію даної наукової розробки тільки за певних умов.

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1-й крок. Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану за формулою (4.9), тобто будемо вважати, що ЗВ = PV = 161577,56 (грн).

2-й крок. Розрахунок очікуваного збільшення прибутку $\Delta П_i$, яке отримає підприємство від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку також було розраховане нами раніше за формулами (4.10).

3-й крок. Для спрощення подальших розрахунків будують вісь часу, на яку наносять всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Припустимо, що загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнюють 161577,56 грн. Результати вкладених у наукову розробку інвестицій почнуть виявлятись протягом трьох років. У першому році підприємство отримає збільшення чистого прибутку на 471000,00 грн відносно базового року, в другому році – збільшення чистого прибутку на 921000,00 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 1605000,00 грн (відносно базового року).

4-й крок. Розраховують абсолютну ефективність вкладених інвестицій $Е_{абс}$.

Для цього використовуємо формулу 4.10:

$$Е_{абс} = (ПП – PV), \quad (4.10)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою 4.11:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{471000{,}00}{(1+0.2)^1} + \frac{921000{,}00}{(1+0.2)^2} + \frac{1605000{,}00}{(1+0.2)^3} = 1960208{,}33 \text{ (грн).}$$

$$Е_{абс} = 1960208{,}33 - 161577{,}56 = 1798630.77 \text{ (грн).}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

5-й крок. Розраховуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. за формулою 4.12:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1, \quad (4.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, 3 роки.

Далі, розрахована величина $Е_в$ порівнюється з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою 4.13:

$$\tau = d + f, \qquad (4.13)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = (0,14...0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1), але може бути і значно більше.

Якщо величина $Е_в > \tau_{мін}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки. В іншому випадку фінансування наукової розробки здійснюватися не буде.

Спрогнозуємо величину $\tau_{мін}$. Припустимо, що за даних умов

$$\tau_{мін} = 0,2 + 0,1 = 0,3.$$

Тоді відносна (щорічна) ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів складе:

$$Е_в = \sqrt[3]{1 + \frac{1798630,77}{161577,56}} - 1 = 2,297 - 1 = 1,297 \text{ або } 129,7\%$$

Оскільки величина $Е_в > \tau_{мін}$, то інвестор буде зацікавлений у фінансуванні даної наукової розробки.

6-й крок. Розраховуємо термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $Т_{ок}$ можна розрахувати за формулою 4.14:

$$Т_{ок} = \frac{1}{Е_в}. \qquad (4.14)$$

Якщо $T_{ок}$ < 3…5-ти років, то фінансування даної наукової розробки в принципі є доцільним. В інших випадках потрібні додаткові розрахунки та обґрунтування.

Для нашої розробки термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ складає:

$$T_{ок} = \frac{1}{1,297} = 0,771 \text{ (року)}.$$

Оскільки, $T_{ок}$ є меншим трьох років, фінансування нової наукової розробки вважається доцільним.

4.4 Висновки

У даному розділі магістерської кваліфікаційної роботи було проведено оцінювання комерційного потенціалу розробки програмного забезпечення. Було проведено технологічний аудит із залученням трьох експертів. Аналіз експертних даних показав, що рівень комерційного потенціалу розробки вище середнього.

Дослідження комерційного потенціалу розробки показало, що програмний продукт за своїми характеристиками має ряд переваг над аналогічними програмами, і є перспективною розробкою. Оскільки, він має ряд кращих функціональних показників, програмна розробка є конкурентоспроможним товаром на ринку. Зазначені переваги нової розробки дозволять швидко її поширити на ринку.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 129,7%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 30%. Це може означати потенційну зацікавленість інвесторів у фінансуванні розробки, а термін окупності вкладених у реалізацію проекту інвестицій становить 0,77 року, що свідчить про доцільність фінансування розробки.

ВИСНОВОК

У процесі виконання магістерської роботи було здійснено удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів із використанням нейронної мережі та проведено комплексне дослідження, спрямоване на підвищення рівня безпеки транспортного рівня Інтернет-комунікацій без втручання у базову реалізацію TLS.

Розроблена система реалізує інтелектуальний підхід до контролю довіри в мережевих з’єднаннях, поєднуючи формальні криптографічні механізми з методами машинного навчання.

На початковому етапі роботи проведено аналітичний огляд сучасних загроз і вразливостей TLS-протоколу, зокрема атак типу man-in-the-middle, компрометації сертифікатів, недійсних ланцюгів довіри та проблем із верифікацією CRL/OCSP-запитів.

Досліджено механізми класичної перевірки сертифікатів, системи PKI, а також проаналізовано обмеження традиційних моделей автентифікації, які не враховують динамічний контекст або поведінкові аномалії під час TLS-обміну.

Особливу увагу приділено інтелектуальним методам оцінки ризику, що дозволяють автоматично виявляти потенційно шкідливі сесії на основі поведінкових характеристик і цифрових відбитків сертифікатів.

У результаті дослідження було спроєктовано та реалізовано прототип системи, який функціонує як PHP-middleware рівня застосунку.

Це рішення здійснює динамічну перевірку сертифікатів і керування TLS-сесіями без необхідності модифікувати базову бібліотеку OpenSSL чи серверні модулі.

До складу системи входять ключові компоненти: модуль логування, який фіксує TLS-метадані, ризикові атрибути, стан сертифікатів і результати аналізу в базі MySQL; багаторівневе управління сесіями — дозволяє застосовувати політики allow, challenge, rekey, block залежно від рівня ризику; модуль динамічної перевірки сертифікатів — виконує гібридну оцінку довіри,

поєднуючи стандартну PKI-перевірку та машинний аналіз; інтерфейс адміністратора — надає засоби для моніторингу сесій, ручної розмітки сертифікатів, налаштування політик і донавчання моделі; система алертів — автоматично повідомляє про підозрілі TLS-події та зберігає історію в журналі аудиту.

Архітектура розробленої системи базується на MVC-підході із застосуванням PHP, RedBeanPHP, MySQL і Bootstrap 5.

Серверна частина реалізує логіку обробки запитів, зберігання даних і виклику ML-оцінок, тоді як клієнтська частина відображає результати у вигляді інтерактивних карток, таблиць і графічних панелей.

База даних побудована за реляційною схемою, що забезпечує простоту розширення та зручність аудиту.

Модуль нейронної оцінки ризику реалізовано на PHP із використанням попередньо збережених ваг моделі у форматі JSON, він аналізує ключові параметри сертифікатів довжину ключа, алгоритм підпису, термін дії, історію перевірок і обчислює ризиковий бал від 0 до 1.

У ході реалізації проведено моделювання роботи системи та функціональні тести.

Перевірено правильність взаємодії між модулями логування, політик і нейронної оцінки.

Було змодельовано кілька десятків TLS-сесій із різними параметрами ризику — від низького до критичного — що дозволило підтвердити працездатність адаптивного керування.

Система коректно ідентифікує підозрілі сертифікати, фіксує відповідні події у журналі аудиту та активує політики блокування або повторного ключообміну.

Розроблений програмний продукт дозволяє перейти від статичної моделі перевірки довіри до динамічної — коли рішення приймається з урахуванням контексту, попередніх інцидентів і поточних метаданих TLS-з’єднання.

Це забезпечує підвищену стійкість системи до нових типів атак і знижує ризик використання скомпрометованих або підроблених сертифікатів.

Економічний аналіз показав, що впровадження такої системи може скоротити фінансові втрати від інцидентів безпеки на 60–70 % і окупитися протягом перших шести місяців експлуатації.

Завдяки модульній структурі, рішення легко розширюється — можлива інтеграція з іншими системами моніторингу, додавання нових типів політик або розширення бази навчальних даних для моделі машинного навчання.

Подальший розвиток системи може включати: використання глибших нейронних моделей для класифікації ризиків; автоматичне оновлення моделі на основі розмічених даних; інтеграцію з хмарними інфраструктурами безпеки або SIEM-системами; розробку клієнтської бібліотеки для підключення до сторонніх веб-сервісів.

Узагальнюючи результати дослідження, можна стверджувати, що поставлена мета повністю досягнута.

Розроблена система поєднує криптографічну надійність протоколу TLS із гнучкістю машинного аналізу ризиків, забезпечуючи новий рівень захисту мережевих комунікацій.

Вона є прикладом практичного впровадження концепції інтелектуальної кібербезпеки, де рішення приймаються не лише на основі правил, а й за допомогою самоадаптивного алгоритму, що вчиться з досвіду.

Розроблений прототип має потенціал для подальшого використання у корпоративних, банківських і державних мережах як ефективний засіб моніторингу й управління довірою у TLS-обміні.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кудревич В. УДОСКОНАЛЕННЯ ПРОТОКОЛУ TLS НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ. «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)».

2. Dierks T., Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446, 2018. 160 с.

3. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Pearson, 2023. 784 с.

4. Kaufman C., Perlman R., Speciner M. Network Security: Private Communications in a Public World. 3rd ed. Prentice Hall, 2016. 744 с.

5. Bishop M. Introduction to Computer Security. Addison-Wesley, 2021. 720 с.

6. Tanenbaum A., Wetherall D. Computer Networks. 6th ed. Pearson, 2022. 960 с.

7. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 775 с.

8. Russell S., Norvig C. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2021. 1152 с.

9. Chollet F. Deep Learning with Python. 2nd ed. Manning Publications, 2021. 504 с.

10. Murphy K. Machine Learning: A Probabilistic Perspective. 2nd ed. MIT Press, 2022. 1280 с.

11. Alpaydin E. Introduction to Machine Learning. 4th ed. MIT Press, 2020. 640 с.

12. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. 2nd ed. Springer, 2017. 764 с.

13. Flanagan D. JavaScript: The Definitive Guide. 7th ed. O’Reilly, 2020. 704 с.

14. Nixon R. Learning PHP, MySQL & JavaScript. 6th ed. O’Reilly, 2021. 812 с.

15. Robbins J. Learning Web Design. 5th ed. O'Reilly, 2018. 808 с.

16. Rubix ML: Machine Learning for PHC. O'Reilly, 2021. 320 с.

17. Menezes A., van Oorschot C., Vanstone S. Handbook of Applied Cryptography. CRC Press, 2019. 816 с.

18. Ferguson N., Schneier B., Kohno T. Cryptography Engineering: Design Principles and Practical Applications. Wiley, 2015. 384 с.

19. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 20th Anniversary ed. Wiley, 2015. 784 с.

20. Felici M., Paccagnella R. TLS Vulnerabilities and Attacks: A Systematic Review. Computers & Security, 2022, vol. 113, pc. 102–122.

21. Camacho J., et al. AI-Driven Cybersecurity: Threat Detection and Response using Neural Networks. IEEE Access, 2023, vol. 11, pc. 22155–22173.

22. IBM Security. IBM Security Report 2024: Cost of a Data Breach. IBM Security, 2024. 72 с.

23. Google AI Security Team. Using Machine Learning to Secure Network Communications. Google Research Blog, 2023. URL: https://ai.googleblog.com (дата звернення: 10.11.2025).

24. PHP Manual. URL: https://www.phc.net/manual/en/ (дата звернення: 14.11.2025).

25. MySQL 8.0 Reference Manual. URL: https://dev.mysql.com/doc/ (дата звернення: 18.11.2025).

26. Bootstrap Documentation. URL: https://getbootstrac.com/docs/ (дата звернення: 15.10.2025).

27. Font Awesome Icons. URL: https://fontawesome.com/icons (дата звернення: 10.10.2025).

28. jQuery API Documentation. URL: https://api.jquery.com (дата звернення: 05.10.2025).

29. Rubix ML Documentation. URL: https://rubixml.com/ (дата звернення: 20.11.2025).

30. OpenSSL Project Documentation. URL: https://www.openssl.org/docs/ (дата звернення: 19.11.2025).

31. IETF. RFC 5280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. 2019. 128 с.

32. IETF. RFC 6066: Transport Layer Security (TLS) Extensions. 2018. 96 с.

33. Cloudflare. Understanding TLS 1.3 and Modern Encryption. URL: https://www.cloudflare.com/learning/ssl/ (дата звернення: 21.11.2025).

34. OWASP Foundation. Transport Layer Security Cheat Sheet. URL: https://owasc.org/www-project-cheat-sheets/ (дата звернення: 17.11.2025).

35. Microsoft Research. Machine Learning Applications in Network Security. URL: https://www.microsoft.com/research/ (дата звернення: 22.11.2025).

36. NIST. SP 800-52r2. Guidelines for the Selection, Configuration, and Use of TLS Implementations. 2019. 85 с.

37. NIST. SP 800-175b. Guideline for Using Cryptographic Standards in the Federal Government. 2020. 94 с.

38. ENISA. Security Guidelines on Transport Layer Security. ENISA, 2021. 110 с.

39. Schneier on Security. AI and the Future of Cryptographic Trust. URL: https://www.schneier.com (дата звернення: 24.10.2025).

40. National Cybersecurity Centre (UK). Machine Learning in Security Operations. NCSC Technical Report, 2023. 54 с.

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор

_____ Юрій ЯРЕМЧУК

"24" вересня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

Удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі

08-72.МКР.012.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи

д.ф., доцент Салієва О.В.

_____

Вінниця – 2025 р.

1. Найменування та область застосування

Програмний засіб удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №96 від 20.03.2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: удосконалення криптографічного протоколу TLS на основі використанням нейронної мережі.

3.2 Призначення: розроблений програмний засіб удосконалює криптографічний протоколу TLS

4. Джерела розробки

4.1 Kaufman C., Perlman R., Speciner M. Network Security: Private Communications in a Public World. 3rd ed. Prentice Hall, 2016. 744 с.

4.2 Bishop M. Introduction to Computer Security. Addison-Wesley, 2021. 720 с.

4.3 Tanenbaum A., Wetherall D. Computer Networks. 6th ed. Pearson, 2022. 960 с.

4.4 Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 775 с.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати інтуїтивно зрозумілий інтерфейс користувача для ефективної взаємодії;

5.1.2 Система повинна використовувати методи машинного навчання для виявлення шкідливого трафіку без необхідності встановлення спеціальних ліцензійних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен стабільно працювати без помилок; у випадку виявлення критичної ситуації — повідомляти користувача;

5.2.2 Система повинна забезпечувати безперебійне збереження даних у разі аварійних ситуацій та мати функцію створення резервних копій;

5.2.3 Програма має ефективно реагувати на підозрілі протоколи шляхом оновлення алгоритмів нейронної мережі.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Intel Core i3 або еквівалентні, з тактовою частотою не менше 2000 МГц;

– оперативна пам’ять – не менше 4 ГБ;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки під час роботи з програмою повинні відповідати стандартам безпеки для захисту даних та роботи з комп'ютерною технікою;

– додаткові вимоги: наявність встановленого PHP та MySQL для роботи з базою даних і виконання PHP-ML моделей.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у розділі 3.

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Використовувати шифрування для захисту даних під час їх передачі та зберігання, забезпечуючи конфіденційність і цілісність інформації.

7.3 Реалізувати механізми моніторингу та аудиту доступу до системи для виявлення спроб несанкціонованого доступу та можливих атак.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	24.09.2025	30.09.2025
2	Аналіз предметної області обраної теми	01.10.2025	08.10.2025
3	Апробація отриманих результатів	09.10.2025	25.10.2025
4	Розробка алгоритму роботи	26.10.2025	31.10.2025
5	Написання магістерської роботи на основі розробленої теми	01.11.2025	20.11.2025
6	Розробка економічної частини	21.11.2025	23.11.2025
7	Передзахист магістерської кваліфікаційної роботи	24.11.2025	25.11.2025
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	26.11.2025	07.12.2025
9	Захист магістерської кваліфікаційної роботи	08.12.2025	11.12.2025

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Кудревич В.І.

Додаток Б. Лістинг створення бази даних

```
-- CREATE DATABASE ai_tls CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci;

-- 1) Сертифікати
CREATE TABLE certificates (
  id               BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  fingerprint      VARCHAR(128) NOT NULL UNIQUE,
  common_name      VARCHAR(255) NULL,
  issuer           VARCHAR(255) NULL,
  valid_from       DATETIME NULL,
  valid_to         DATETIME NULL,
  pubkey_algo      VARCHAR(64)  NULL,
  pubkey_bits      INT          NULL,
  sig_algo         VARCHAR(64)  NULL,
  status           ENUM('new','reviewed','whitelisted','blocked')
NOT NULL DEFAULT 'new',
  risk_score       DECIMAL(5,3) NOT NULL DEFAULT 0.000,
  created_at       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
  KEY ix_cert_valid_to (valid_to),
  KEY ix_cert_status  (status),
  KEY ix_cert_risk    (risk_score)
) ENGINE=InnoDB;

-- 2) TLS-сесії
CREATE TABLE sessions (
  id                BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  session_id        VARCHAR(64)  NOT NULL,
  client_ip         VARBINARY(16) NOT NULL, -- підтримка IPv4/IPv6
(зберігаємо у BIN)
  server_ip         VARBINARY(16) NOT NULL,
  tls_version       VARCHAR(16)   NOT NULL,  -- 'TLS 1.3', 'TLS
1.2'
  cipher_suite      VARCHAR(64)   NOT NULL,
  cert_fingerprint  VARCHAR(128)  NOT NULL,  -- FK →
certificates.fingerprint
  risk_score        DECIMAL(5,3)  NOT NULL DEFAULT 0.000,
  risk_level        ENUM('low','medium','high','critical') NOT
NULL DEFAULT 'low',
  action
ENUM('allow','monitor','challenge','rekey','block') NOT NULL
DEFAULT 'allow',
  status            ENUM('active','closed') NOT NULL DEFAULT
'active',
  started_at        DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  ended_at          DATETIME NULL,
  INDEX ix_sess_sid          (session_id),
  INDEX ix_sess_started      (started_at),
  INDEX ix_sess_status       (status),
  INDEX ix_sess_risk         (risk_level, risk_score),
```

```
  CONSTRAINT fk_sess_cert FOREIGN KEY (cert_fingerprint)
      REFERENCES certificates (fingerprint)
      ON UPDATE CASCADE ON DELETE RESTRICT
) ENGINE=InnoDB;

-- 3) Підозрілі/розмічені приклади для донавчання
CREATE TABLE suspicious_certs (
  id               BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  cert_fingerprint VARCHAR(128) NOT NULL,
  source           ENUM('session','manual','system') NOT NULL
DEFAULT 'session',
  label            ENUM('benign','malicious','unknown') NOT NULL
DEFAULT 'unknown',
  reviewer_note    VARCHAR(512) NULL,
  created_at       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  reviewed_at      DATETIME NULL,
  CONSTRAINT fk_susp_cert FOREIGN KEY (cert_fingerprint)
      REFERENCES certificates (fingerprint)
      ON UPDATE CASCADE ON DELETE CASCADE,
  KEY ix_susp_label (label),
  KEY ix_susp_created (created_at)
) ENGINE=InnoDB;

-- 4) Політики безпеки (активна – одна або кілька з пріоритетами)
CREATE TABLE policy_settings (
  id               BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name             VARCHAR(128) NOT NULL,
  mode             ENUM('allow','challenge','rekey','block') NOT
NULL DEFAULT 'allow',
  r_low_max        DECIMAL(5,3) NOT NULL DEFAULT 0.300,  -- <0.3 -
> allow
  r_med_max        DECIMAL(5,3) NOT NULL DEFAULT 0.600,  -- <0.6 -
> monitor/challenge
  r_high_max       DECIMAL(5,3) NOT NULL DEFAULT 0.850,  -- <0.85-
> rekey
  is_active        TINYINT(1) NOT NULL DEFAULT 1,
  priority         INT NOT NULL DEFAULT 100,              -- менше
число = вищий пріоритет
  updated_at       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
  UNIQUE KEY uq_policy_name (name),
  KEY ix_policy_active (is_active, priority)
) ENGINE=InnoDB;

-- 5) Журнал аудиту (незмінний)
CREATE TABLE audit_log (
  id           BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  event_type
ENUM('session','certificate','ml','policy','system') NOT NULL,
  session_id   VARCHAR(64) NULL,
  cert_fp      VARCHAR(128) NULL,
  description  VARCHAR(1024) NOT NULL,
  risk_level   ENUM('low','medium','high','critical') NULL,
```

```
  created_at   DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  KEY ix_audit_type_time (event_type, created_at),
  KEY ix_audit_session   (session_id),
  KEY ix_audit_cert      (cert_fp),
  KEY ix_audit_risk      (risk_level)
) ENGINE=InnoDB;

-- 6) Алерти (інциденти для оператора)
CREATE TABLE alerts (
  id           BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  event_type
ENUM('session','certificate','ml','policy','system') NOT NULL,
  ref_id       BIGINT UNSIGNED NULL,        -- опційний зв'язок на
сутність (sessions.id тощо)
  description  VARCHAR(1024) NOT NULL,
  risk_level   ENUM('low','medium','high','critical') NOT NULL
DEFAULT 'medium',
  created_at   DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  is_resolved  TINYINT(1) NOT NULL DEFAULT 0,
  KEY ix_alert_type_time (event_type, created_at),
  KEY ix_alert_risk      (risk_level),
  KEY ix_alert_resolved  (is_resolved, created_at)
) ENGINE=InnoDB;
```

Додаток В. Лістинг файлу train.php

```
<?php

$train_file = __DIR__ . '/training_data.json';
$model_file = __DIR__ . '/model_weights.json';

$epochs = 800;
$lr = 0.05;


$data = json_decode(file_get_contents($train_file), true);
$X = [];
$Y = [];
foreach ($data as $row) {
    $X[] = $row['features'];
    $Y[] = $row['label'];
}
$N = count($X);
$input_dim = count($X[0]);
$hidden = 6;


function rand_matrix($r,$c,$scale=0.1){
    $m = [];
    for($i=0;$i<$r;$i++){
        $m[$i] = [];
        for($j=0;$j<$c;$j++){
            $m[$i][$j] = ($scale * (rand()/getrandmax()*2-1));
        }
    }
    return $m;
}
function rand_vector($n,$scale=0.1){
    $v = [];
    for($i=0;$i<$n;$i++) $v[$i] = $scale * (rand()/getrandmax()*2-
1);
    return $v;
}


if (file_exists($model_file)) {
    $m = json_decode(file_get_contents($model_file), true);
    $W1 = $m['W1']; $b1 = $m['b1']; $W2 = $m['W2']; $b2 =
$m['b2'];
} else {
    $W1 = rand_matrix($hidden, $input_dim, 0.2);
    $b1 = rand_vector($hidden, 0.1);
    $W2 = rand_matrix(1, $hidden, 0.2);
    $b2 = rand_vector(1, 0.1);
}
```

```
function dot($A, $x) {
    $out = [];
    for($i=0;$i<count($A);$i++){
        $s = 0.0;
        for($j=0;$j<count($A[$i]);$j++) $s += $A[$i][$j] * $x[$j];
        $out[$i] = $s;
    }
    return $out;
}
function add_vec($v,$b){ for($i=0;$i<count($v);$i++)
$v[$i]+=$b[$i]; return $v;}
function relu($v){ for($i=0;$i<count($v);$i++)
$v[$i]=max(0.0,$v[$i]); return $v;}
function drelu($v){ for($i=0;$i<count($v);$i++)
$v[$i]=($v[$i]>0)?1.0:0.0; return $v;}
function sigmoid($x){ return 1.0/(1.0 + exp(-$x)); }


for($ep=0;$ep<$epochs;$ep++){
    $loss = 0.0;

    $dW2 = array_fill(0,1, array_fill(0,$hidden,0.0));
    $db2 = array_fill(0,1,0.0);
    $dW1 = array_fill(0,$hidden, array_fill(0,$input_dim,0.0));
    $db1 = array_fill(0,$hidden,0.0);

    for($i=0;$i<$N;$i++){
        $x = $X[$i];
        $y = $Y[$i];


        $z1 = add_vec(dot($W1,$x), $b1);
        $a1 = relu($z1);
        $z2_arr = add_vec(dot($W2,$a1), $b2);
        $z2 = $z2_arr[0];
        $a2 = sigmoid($z2);

        $loss += -($y*log(max(1e-8,$a2)) + (1-$y)*log(max(1e-8,1-
$a2)));

        $dz2 = $a2 - $y; // dL/dz2
        for($h=0;$h<$hidden;$h++){
            $dW2[0][$h] += $dz2 * $a1[$h];
        }
        $db2[0] += $dz2;


        $d_a1 = [];
        for($h=0;$h<$hidden;$h++){
            $d_a1[$h] = $W2[0][$h] * $dz2;
        }
        $dz1 = $d_a1;
        $mask = drelu($z1);
```

```
            for($h=0;$h<$hidden;$h++) $dz1[$h] *= $mask[$h];

            for($h=0;$h<$hidden;$h++){
                for($j=0;$j<$input_dim;$j++){
                    $dW1[$h][$j] += $dz1[$h] * $x[$j];
                }
                $db1[$h] += $dz1[$h];
            }
        }


        $eta = $lr / $N;
        for($h=0;$h<$hidden;$h++){
            for($j=0;$j<$input_dim;$j++){
                $W1[$h][$j] -= $eta * $dW1[$h][$j];
            }
            $b1[$h] -= $eta * $db1[$h];
        }
        for($h=0;$h<$hidden;$h++){
            $W2[0][$h] -= $eta * $dW2[0][$h];
        }
        $b2[0] -= $eta * $db2[0];

        if ($ep % 100 == 0) {
            echo "Epoch $ep, loss=" . round($loss/$N,5) . PHP_EOL;
        }
    }
```

Додаток Г. Лістинг файлу /views/certs/index.php

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark shadow-sm
rounded-3 mb-4 px-3 py-2">
  <div class="container-fluid">
    <a class="navbar-brand fw-bold text-info" href="/">
      <i class="fa-solid fa-lock me-2"></i>AI-TLS
    </a>
    <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#mainNav">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="mainNav">
      <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
        <li class="nav-item"><a class="nav-link" href="/"><i
class="fa-solid fa-house me-1"></i>Головна</a></li>
        <li class="nav-item"><a class="nav-link"
href="/sessions"><i class="fa-solid fa-network-wired me-
1"></i>Сесії</a></li>
        <li class="nav-item"><a class="nav-link" href="/certs"><i
class="fa-solid fa-certificate me-1"></i>Сертифікати</a></li>
        <li class="nav-item"><a class="nav-link" href="/ml"><i
class="fa-solid fa-brain me-1"></i>ML-модуль</a></li>
        <li class="nav-item"><a class="nav-link"
href="/policies"><i class="fa-solid fa-shield-halved me-
1"></i>Політики</a></li>
        <li class="nav-item"><a class="nav-link" href="/audit"><i
class="fa-solid fa-clipboard-list me-1"></i>Журнал</a></li>
        <li class="nav-item"><a class="nav-link" href="/alerts"><i
class="fa-solid fa-bell me-1"></i>Алерти</a></li>
      </ul>
    </div>
  </div>
</nav>

<div class="card shadow-sm border-0 rounded-4 overflow-hidden">
  <div class="card-header bg-primary text-white py-3 d-flex
justify-content-between align-items-center">
    <h5 class="mb-0">
      <i class="fa-solid fa-certificate me-2"></i>Керування
сертифікатами
    </h5>
  </div>
  <div class="card-body bg-light p-4">
    <ul class="nav nav-tabs" id="certTabs" role="tablist">
      <li class="nav-item" role="presentation">
        <button class="nav-link active" id="all-tab" data-bs-
toggle="tab" data-bs-target="#all" type="button" role="tab">
          <i class="fa-regular fa-id-card me-1"></i>Усі
сертифікати
        </button>
      </li>
      <li class="nav-item" role="presentation">
```

```
        <button class="nav-link" id="suspect-tab" data-bs-
toggle="tab" data-bs-target="#suspect" type="button" role="tab">
          <i class="fa-solid fa-triangle-exclamation me-1 text-
danger"></i>Підозрілі
        </button>
      </li>
    </ul>
    <div class="tab-content mt-4" id="certTabsContent">
      <!-- УСІ СЕРТИФІКАТИ -->
      <div class="tab-pane fade show active" id="all"
role="tabpanel">
        <?php if (!empty($certs)): ?>
          <div class="table-responsive">
            <table class="table table-hover align-middle bg-white
rounded-4 overflow-hidden">
              <thead class="table-light">
                <tr>
                  <th>ID</th>
                  <th>CN</th>
                  <th>Issuer</th>
                  <th>Початок дії</th>
                  <th>Кінець дії</th>
                  <th>Скасовано</th>
                  <th></th>
                </tr>
              </thead>
              <tbody>
                <?php foreach ($certs as $c): ?>
                  <tr>
                    <td><?= $c->id; ?></td>
                    <td><strong><?= htmlspecialchars($c-
>common_name); ?></strong></td>
                    <td class="text-muted"><?=
htmlspecialchars($c->issuer); ?></td>
                    <td><?= htmlspecialchars($c->valid_from);
?></td>
                    <td><?= htmlspecialchars($c->valid_to);
?></td>
                    <td>
                      <?= $c->is_revoked ? '<span class="badge bg-
danger">Так</span>' : '<span class="badge bg-success">Ні</span>';
?>
                    </td>
                    <td class="text-end">
                      <a href="/certs/view?id=<?= $c->id; ?>"
class="btn btn-sm btn-outline-primary">
                        <i class="fa-solid fa-eye me-1"></i>Деталі
                      </a>
                      <a href="/certs/addml?id=<?= $c->id; ?>"
class="btn btn-sm btn-outline-warning">
                        <i class="fa-solid fa-brain me-1"></i>ML
                      </a>
                    </td>
```

```
                </tr>
              <?php endforeach; ?>
            </tbody>
          </table>
        </div>
      <?php else: ?>
        <div class="alert alert-secondary text-center py-4
rounded-4 shadow-sm">
          <i class="fa-regular fa-circle-xmark me-2"></i>Немає
сертифікатів у базі.
        </div>
      <?php endif; ?>
    </div>

    <!-- ПІДОЗРІЛІ СЕРТИФІКАТИ -->
    <div class="tab-pane fade" id="suspect" role="tabpanel">
      <?php if (!empty($suspicious)): ?>
        <div class="table-responsive">
          <table class="table table-hover align-middle bg-white
rounded-4 overflow-hidden">
            <thead class="table-light">
              <tr>
                <th>ID</th>
                <th>CN</th>
                <th>Issuer</th>
                <th>Ризик</th>
                <th>Статус</th>
                <th>Розмічено</th>
                <th></th>
              </tr>
            </thead>
            <tbody>
              <?php foreach ($suspicious as $s): ?>
                <?php
                  $riskColor = ($s['risk_score'] > 0.85) ? 'bg-
danger' :
                               (($s['risk_score'] > 0.6) ? 'bg-
warning text-dark' :
                               'bg-secondary');
                  $statusBadge = match($s['status']) {
                    'new' => '<span class="badge bg-info text-
dark">New</span>',
                    'reviewed' => '<span class="badge bg-
secondary">Reviewed</span>',
                    'whitelisted' => '<span class="badge bg-
success">Whitelisted</span>',
                    'blocked' => '<span class="badge bg-
danger">Blocked</span>',
                    default => '<span class="badge bg-light
text-dark">Unknown</span>'
                  };
                ?>
                <tr>
```

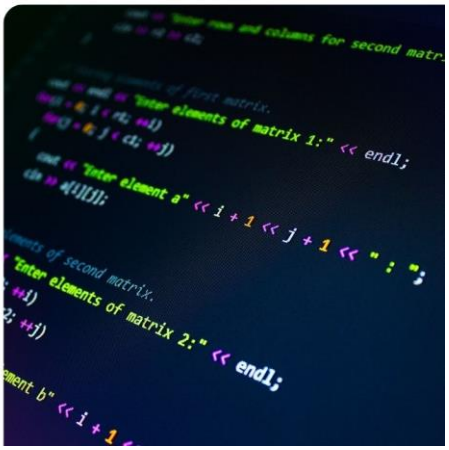
```
                    <td><?= $s['cert_id']; ?></td>
                    <td><strong><?=
htmlspecialchars($s['common_name'] ?? '—'); ?></strong></td>
                    <td class="text-muted"><?=
htmlspecialchars($s['issuer'] ?? '—'); ?></td>
                    <td>
                      <span class="badge <?= $riskColor; ?>"><?=
number_format($s['risk_score'] * 100, 1); ?>%</span>
                    </td>
                    <td><?= $statusBadge; ?></td>
                    <td><?= htmlspecialchars($s['reviewed_by'] ??
'—'); ?></td>
                    <td class="text-end">
                      <a href="/certs/view?id=<?= $s['cert_id'];
?>" class="btn btn-sm btn-outline-primary">
                        <i class="fa-solid fa-eye me-1"></i>Деталі
                      </a>
                      <?php if ($s['status'] !== 'blocked'): ?>
                        <a href="/certs/block?id=<?=
$s['cert_id']; ?>" class="btn btn-sm btn-outline-danger">
                          <i class="fa-solid fa-ban me-
1"></i>Block
                        </a>
                      <?php endif; ?>
                      <?php if ($s['status'] !== 'whitelisted'):
?>
                        <a href="/certs/whitelist?id=<?=
$s['cert_id']; ?>" class="btn btn-sm btn-outline-success">
                          <i class="fa-solid fa-check me-
1"></i>Whitelist
                        </a>
                      <?php endif; ?>
                    </td>
                  </tr>
                <?php endforeach; ?>
              </tbody>
            </table>
          </div>
        <?php else: ?>
          <div class="alert alert-secondary text-center py-4
rounded-4 shadow-sm">
            <i class="fa-regular fa-circle-xmark me-
2"></i>Підозрілих сертифікатів не знайдено.
          </div>
        <?php endif; ?>
      </div>
    </div>
  </div>
</div>
```

Додаток Д. Ілюстративний матеріал

✳ ВНТУ

Удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі

Кудревич Вадим 2КІТС-24м
Керівник: Салієва Ольга



Вступ

Актуальність

Протокол Transport Layer Security є основним стандартом захищеної комунікації, який забезпечує шифрування, автентифікацію сторін та цілісність даних. Однак, попри свою зрілість, TLS залишається вразливим до низки сучасних кіберзагроз — від підробки цифрових сертифікатів і компрометації центрів сертифікації до атак типу man-in-the-middle та отруєння кешу OCSC. Традиційні методи перевірки сертифікатів, такі як CRL чи OCSP, є статичними та не забезпечують гнучкої адаптації до нових типів ризиків.

Мета роботи

Метою даної магістерської роботи є розробка та впровадження удосконаленого підходу до захисту TLS-з'єднань, який поєднує багаторівневе управління сесіями, динамічну перевірку сертифікатів та інтелектуальне оцінювання ризику з використанням нейронної мережі.

Вступ

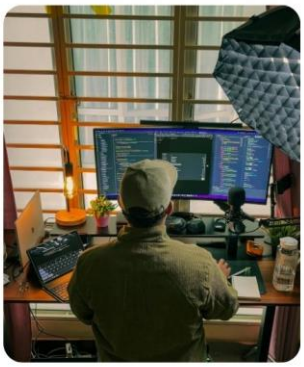
Дослідження

Об'єктом дослідження є процес захищеного встановлення та підтримки TLS-сесій між клієнтом і сервером.
Предметом дослідження є методи вдосконалення криптографічного протоколу TLS за рахунок динамічного управління сесіями, перевірки сертифікатів і оцінки ризику з використанням нейронних мереж.

Задачі

-проєктування архітектури middleware-рівня для контролю TLS-з'єднань;
-розробка алгоритму оцінювання ризику TLS-сесій із використанням нейронної мережі;
-реалізація програмного прототипу із логуванням сесій, системою політик і модулем сповіщень;
-тестування роботи системи на прикладах різних сценаріїв ризику;

*ЗАГАЛЬНИЙ АНАЛІЗ ТЕХНОЛОПІЙ ЗАХИСТУ ТА АВТЕНТИФІКАЦІЇ В ПРОТОКОЛІ TLS



Підробка сертифікатів

Суттєвим ризиком є підробка або компрометація центрів сертифікації. Якщо зловмисник отримує доступ до приватного ключа CA, він може створювати сертифікати, які виглядають цілком дійсними для будь-якої системи перевірки. Такі інциденти вже неодноразово фіксувалися — зокрема, компрометація DigiNotar, Comodo, Symantec, що призводила до масового підписання фейкових сертифікатів для популярних доменів

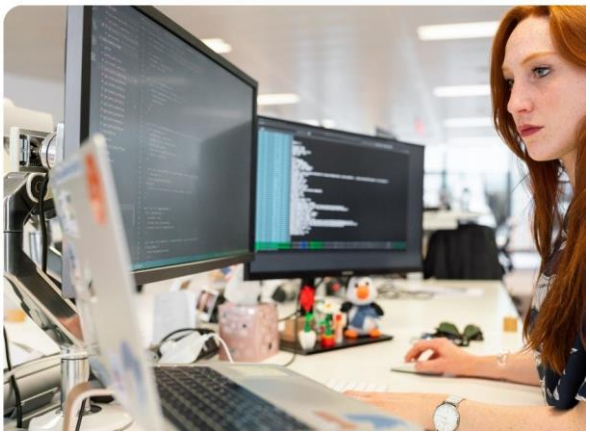
Попри широке застосування протоколу TLS у всіх сучасних мережевих сервісах, його використання не гарантує абсолютного рівня захисту. Основні механізми TLS — автентифікація, шифрування та перевірка цілісності — ефективні лише за умови коректного налаштування та надійності всіх складових криптографічної інфраструктури. На практиці вразливості часто виникають через помилки конфігурації, компрометацію сертифікатів, використання застарілих алгоритмів або зловмисні дії, спрямовані на обхід перевірки автентичності

Актуальність удосконалення TLS

Недосвідченість користувача

Окрему увагу необхідно приділити людському фактору. Користувачі часто ігнорують попередження браузера про недійсні сертифікати або погоджуються на встановлення винятків, що фактично знімає захист TLS у конкретному з'єднанні. Такі дії створюють сприятливе середовище для реалізації атак MITM, навіть без зламу самої криптографічної схеми.

Ще одна тенденція останніх років — зростання кількості атак, що використовують легітимні сертифікати для шкідливих цілей. Наприклад, деякі зловмисники навмисно отримують сертифікати на домени, схожі на популярні бренди, після чого запускають фішингові сайти з дійсним HTTPS-з'єднанням. Користувач, вважає сайт безпечним і вводить свої облікові дані. Таким чином, навіть найсучасніші криптографічні механізми не захищають від соціально-технічних прийомів.



Існуючі підходи до перевірки сертифікатів

Перевірка ланцюга довіри

Основним етапом автентифікації в TLS є побудова та перевірка ланцюга довіри — від сертифіката сервера до кореневого центру сертифікації Клієнт перевіряє, чи кожен проміжний сертифікат підписаний наступним у ланцюгу, і чи кореневий сертифікат присутній у локальному сховищі довірених центрів. Попри простоту, цей механізм має вразливість: якщо один із проміжних центрів сертифікації буде скомпрометований, то зловмисник може створювати дійсні сертифікати для будь-яких доменів. Такий сценарій уже неодноразово мав місце в історії (DigiNotar, Comodo, Symantec), і стандартна перевірка ланцюга не дозволяє виявити підробку без зовнішніх джерел інформації.

Протокол онлайн-перевірки статусу сертифіката

Більш гнучким рішенням є Online Certificate Status Protocol, який дозволяє клієнту у реальному часі звертатися до сервера CA для перевірки статусу сертифіката.

Хоча OCSP значно зменшує затримку між відкликанням і виявленням, він має критичний недолік — залежність від доступності сервера CA. Якщо OCSP-сервер недоступний, багато клієнтів просто пропускають перевірку soft-fail, залишаючи потенційно небезпечні з'єднання активними. Крім того, самі OCSP-відповіді можуть бути підроблені або кешовані зловмисниками.

Публічний журнал

Сучасним напрямом розвитку є система Certificate Transparency (CT) — публічний журнал, у який кожен CA зобов'язаний записувати видані сертифікати. Кожен запис підписується та зберігається у відкритій базі даних, що дозволяє будь-кому перевірити факт видачі сертифіката [19].

Перевага CT полягає у підвищенні прозорості: власники доменів можуть відстежувати появу нових сертифікатів і виявляти підробки. Проте CT не надає механізму оцінки ризику або автоматичного блокування — це лише журнал, а не система прийняття рішень. Незважаючи на різноманітність методів перевірки, жоден із них не забезпечує динамічного аналізу ризику. CRL і OCSP реагують із затримкою, не пристосований до змін, а CT працює лише як архів. Усі вони базуються на формальній перевірці валідності, але не враховують поведінкові ознаки з'єднання, частоту використання сертифіката, контекст клієнта чи історію попередніх інцидентів

Переваги використання нейронної мережі

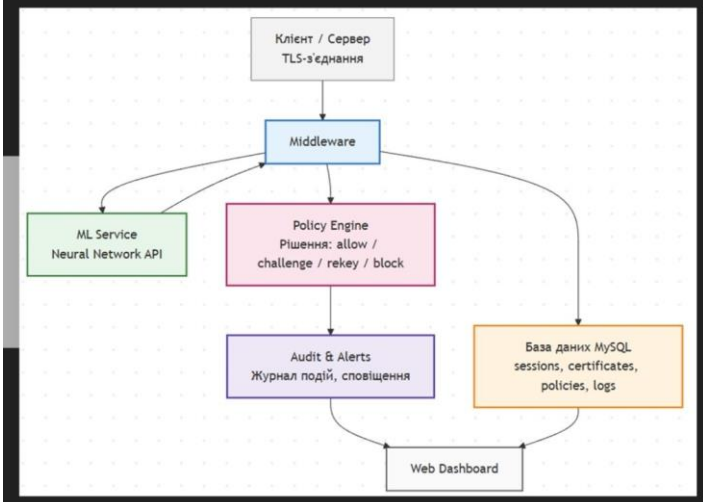
Нейронні мережі, особливо багаторівневі та рекурентні, показують високу ефективність у задачах класифікації мережевого трафіку та виявлення аномалій у TLS-сесіях.
Завдяки здатності працювати з неструктурованими або частково відсутніми даними, такі моделі можуть:

- *оцінювати ризик кожної сесії за сукупністю ознак IP, порт, тривалість, набір шифрів, fingerprint сертифіката, час доби*
- *виявляти патерни, характерні для фішингових або ботнет-з'єднань*
- *прогнозувати ймовірність компрометації сертифіката на основі історичних закономірностей*

УДОСКОНАЛЕННЯ TLS-ОБМІНУ НА ОСНОВІ БАГАТОРІВНЕВОГО КЕРУВАННЮ СЕСІЯМИ ТА ПЕРЕВІРКИ СЕРТИФІКАТІВ

Архітектура системи побудована за принципом багаторівневої взаємодії. Кожен рівень відповідає за свій аспект роботи з TLS-сесіями — від збору технічних параметрів до прийняття рішень і генерації звітів.
Система складається з шести взаємопов'язаних рівнів. Рівень збору даних відповідає за перехоплення метаданих кожної TLS-сесії на рівні PHP-застосунку.
Отримані дані зберігаються у таблиці sessions бази MySQL.
Рівень верифікації сертифікатів забезпечує первинну перевірку отриманих сертифікатів — перевіряє коректність ланцюга довіри, термін дії, видавця, тип підпису Signature Algorithm та унікальний відбиток.

Зібрані параметри включають:
- ідентифікатор сесії;
- IP-адреси клієнта і сервера;
- версію протоколу;
- fingerprint сертифіката;
- часові мітки початку та завершення з'єднання.



Алгоритм роботи

Нейронна модель обчислює ймовірність ризику та повертає класифікацію: low, medium, high, critical. Отримані результати заносяться у поле risk_level таблиці sessions. Рівень управління політиками реалізує логіку адаптивного реагування на основі оціненого ризику. Політики зберігаються у таблиці policy_settings і визначають, яку дію застосувати до сесії:

- allow — звичайна робота сесії;
- challenge — повторна перевірка або верифікація;
- rekey — перегенерація ключів шифрування;
- block — припинення з'єднання;

Рішення системи може змінюватися динамічно, залежно від оновлення оцінки ризику або появи додаткових даних. Всі події системи — створення сесії, зміна статусу, блокування, оцінка ризику, ручні дії користувача — фіксуються у таблиці audit_log. Записи мають тип події, опис, рівень ризику та часову мітку. Це забезпечує повну прозорість функціонування системи та можливість відтворення будь-якої події у майбутньому.

Розробка

Запропонована архітектура відрізняється від традиційних засобів безпеки TLS тим, що:

- не вимагає модифікації стандартного протоколу або SSL-бібліотек;

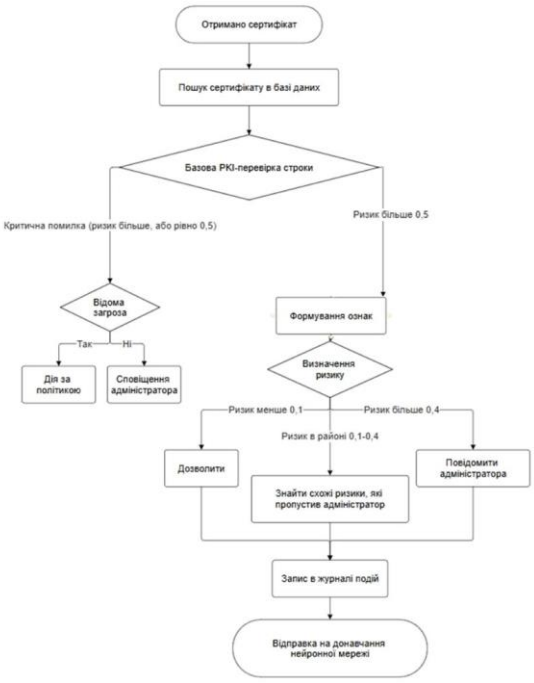
- дозволяє аналізувати поведінку сесій у динаміці;

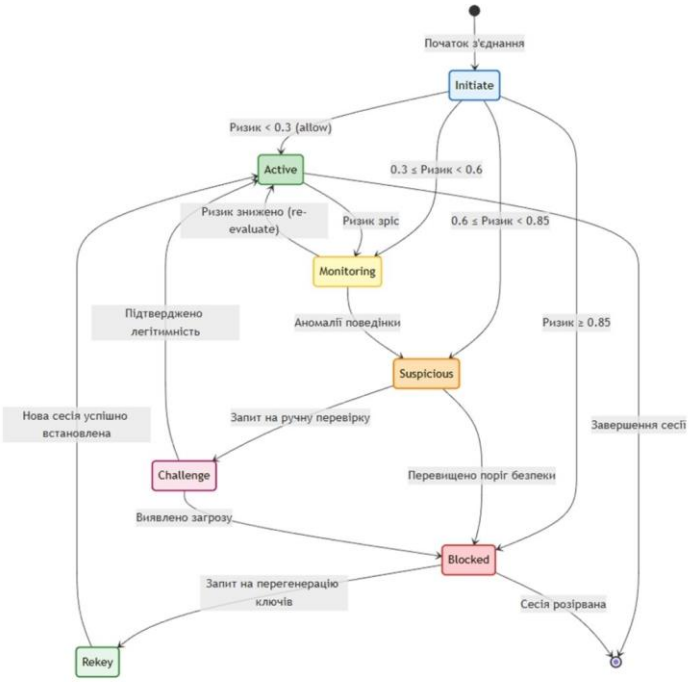
- реалізує концепцію Policy-Driven Security, де рішення приймаються автоматично на основі ризику;

- підтримує адаптивне навчання моделі на основі накопичених даних;

- надає зручний веб-інтерфейс для перегляду, редагування та аналітики;

Таким чином, запропонований middleware виступає інтелектуальним надбудовним шаром над TLS, який перетворює протокол із пасивного механізму шифрування на активну систему контролю безпеки з можливістю самонавчання





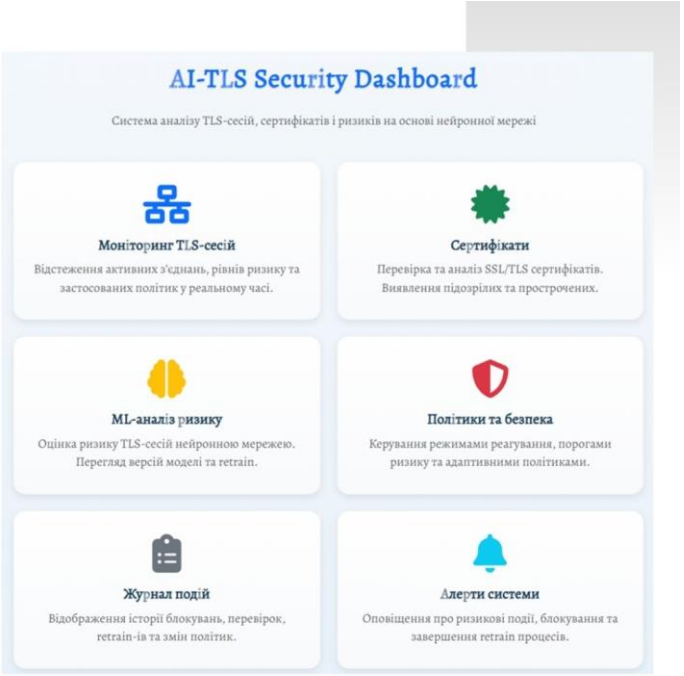
Результат розробки

У підсумку розроблений алгоритм управління TLS-сесіями перетворює статичну модель обміну на динамічну адаптивну систему, яка самостійно контролює життєвий цикл з'єднань. Його інтеграція з алгоритмом динамічної перевірки сертифікатів утворює єдиний аналітичний контур безпеки, що дозволяє системі переходити від пасивного моніторингу до активного прогнозування й запобігання атакам. Такий підхід створює підґрунтя для формування інтелектуальних протоколів нового покоління, у яких безпека стає не лише властивістю середовища, а й його внутрішнім алгоритмічним станом.

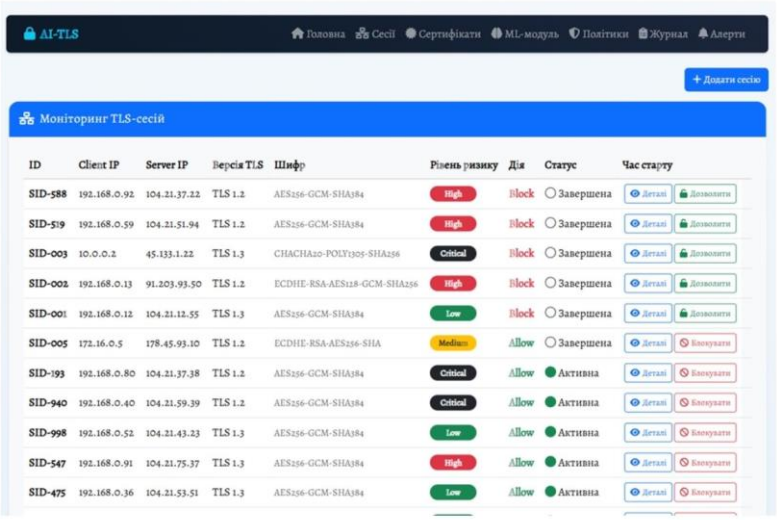
Практична реалізація

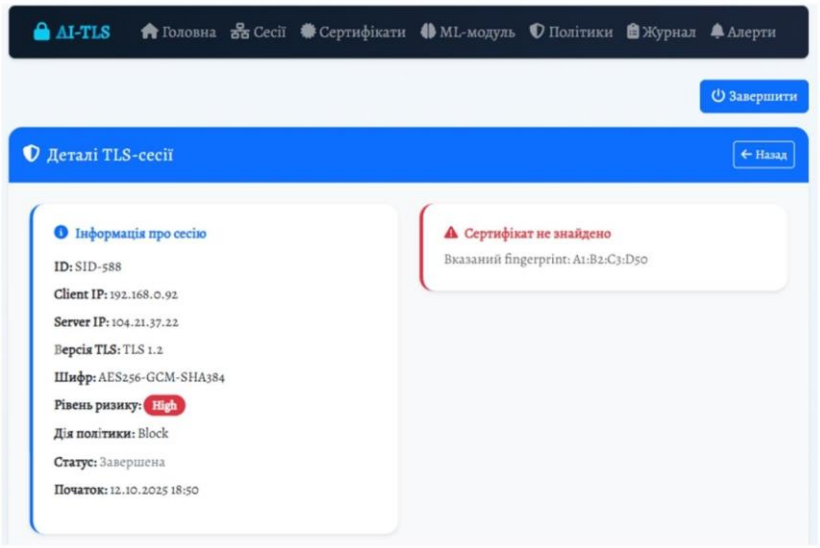
Головна сторінка є центральною точкою входу в систему. Вона побудована у вигляді інформаційної панелі з шістьма інтерактивними картками-плитками, кожна з яких веде до відповідного модуля системи:

- **сесії TLS** — відображає активні та завершені TLS-з'єднання;
- **сертифікати** — база збережених та підозрілих сертифікатів;
- **ML-аналіз** — аналітичний модуль, що демонструє оцінку ризику нейромережею;
- **політики безпеки** — набір адаптивних правил реагування системи;
- **алерти** — журнал попереджень та подій безпеки;
- **журнал аудиту** — історія усіх змін і дій користувача.



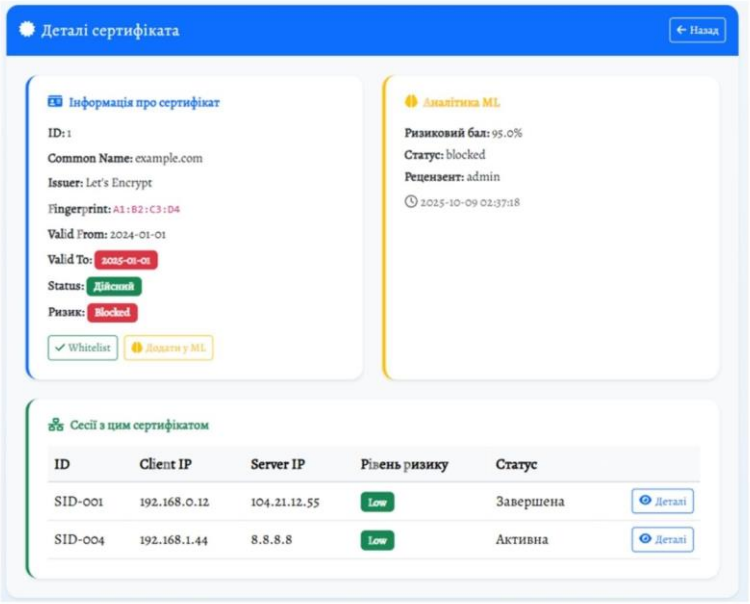
Таблиці моніторингу сесій



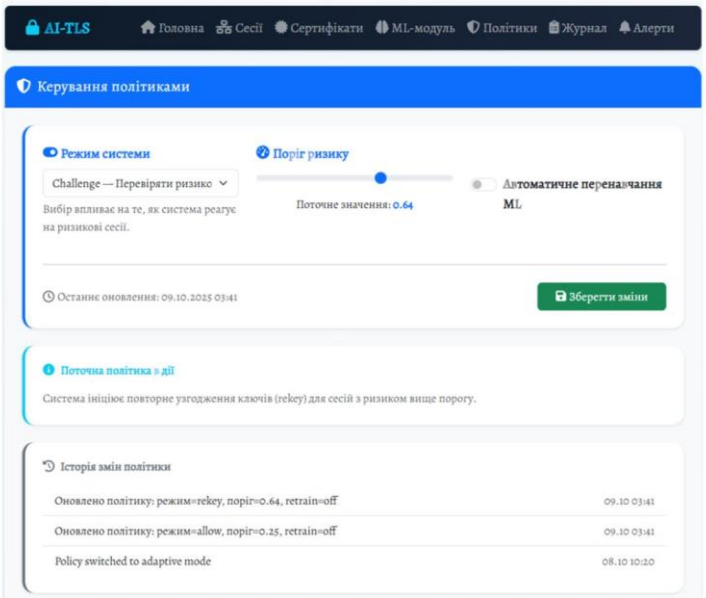


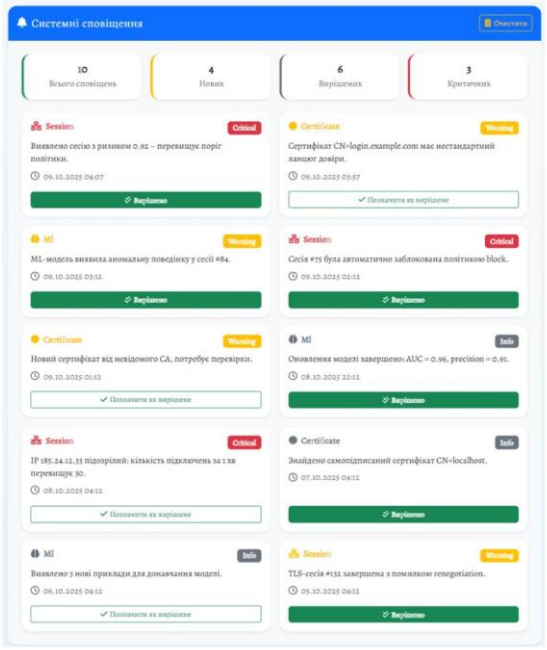
Сторінка сесії з високим ризиком

Детальний перегляд недійсного сертифіката

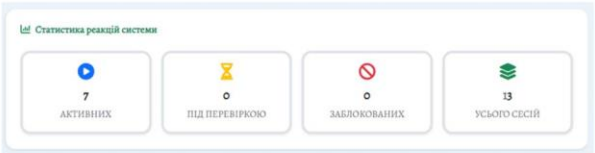


Панель адміністрування





Статистика



Висновки

У процесі виконання магістерської роботи було здійснено удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів із використанням нейронної мережі та проведено комплексне дослідження, спрямоване на підвищення рівня безпеки транспортного рівня Інтернет-комунікацій без втручання у базову реалізацію TLS.

Розроблена система реалізує інтелектуальний підхід до контролю довіри в мережевих з'єднаннях, поєднуючи формальні криптографічні механізми з методами машинного навчання.

До складу системи входять ключові компоненти: модуль логування, який фіксує TLS-метадані, ризикові атрибути, стан сертифікатів і результати аналізу в базі MySQL; багаторівневе управління сесіями — дозволяє застосовувати політики allow, challenge, rekey, block залежно від рівня ризику; модуль динамічної перевірки сертифікатів — виконує гібридну оцінку довіри, поєднуючи стандартну PKI-перевірку та машинний аналіз; інтерфейс адміністратора — надає засоби для моніторингу сесій, ручної розмітки сертифікатів, налаштування політик і донавчання моделі; система алертів — автоматично повідомляє про підозрілі TLS-події та зберігає історію в журналі аудиту.

Додаток Е. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Удосконалення криптографічного протоколу TLS на основі багаторівневого управління сесіями та динамічної перевірки сертифікатів з використанням нейронної мережі

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра менеджменту та безпеки інформаційних систем факультет менеджменту та інформаційної безпеки

гр.2КІТС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) **1,10 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту**

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В. _____

к.ф.-м.н., доцент каф. МБІС Шиян А.А. _____

Особа, відповідальна за перевірку Коваль Н.П. _____

З висновком експертної комісії ознайомлений(-на)

Керівник _____ д.ф. Салієва О.В.

Здобувач _____ Кудревич В.І.