МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) автоматизації і комп'ютерних систем імені проф. І.В. Ельперіна

Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«08» грудня 2025р.	«08» грудня 2025р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)

освітньо-професійної програми Управління інформацією та аналітика даних

на тему: Інформаційна система аналізу корпоративних поштових повідомлень АТ "Укртелеком" з використанням методів машинного навчання

Виконав: здобувач 2 курсу, групи КН-2-2М

Береза Максим Юрійович
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Грама Михайло Петрович _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав і не одержував недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач _____
(підпис)

Київ — 2025р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) автоматизації і комп'ютерних систем імені проф. І.В. Ельперіна
Кафедра __Інформаційних технологій, штучного інтелекту і кібербезпеки___
Освітній ступінь__магістр_____
Спеціальність__122 «Комп'ютерні науки»_____
(код і назва)
Освітньо-професійна програма _Управління інформацією та аналітика даних
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій, штучного інтелекту і кібербезпеки

_____ **Сергій ГРИБКОВ**
«05» __листопада__ 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____Берези Максима Юрійовича_____
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система аналізу корпоративних поштових повідомлень АТ "Укртелеком" з використанням методів машинного навчання___
керівник роботи_Грама Михайло Петрович , старший викладач, доктор філософії
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 5 листопада 2025 року №906-кс
2. Строк подання здобувачем роботи_____1 грудня 2025 року_____
3. Вихідні дані до роботи Розгорнута та налаштована інфраструктура, натренована модель машинного навчання, веб портал для зміни паролю та самообслуговування коритсувачів, система моніторингу на базі Graphana, python код розроблених сервісів, налаштований Firewall
_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)_____
Розділ 1. Системний Аналіз АТ «Укртелеком»_____
Розділ 2. Дослідження та обгрунтування технологій_____
Розділ 3. Налаштування інфраструктури електронної пошти_____
_____
5. Перелік графічного матеріалу:
1)Метрики навчання моделей_____
2) Загальна схема взаємодії інфраструктури та компонентів_____
3) Приклад роботи системи_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	ст. викладач Грама М.П.	01.10.2025	20.10.2025
2	ст. викладач Грама М.П.	21.10.2025	25.11.2025
3	ст. викладач Грама М.П.	25.11.2025	24.12.2025

7. Дата видачі завдання_____1 жовтня 2025 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Приміпка
1	Отримання завдання від керівника	01.10.2025-11.10.2025	Виконано
2	Виконання аналізу предметної теми	11.10.2025-14.10.2025	Виконано
3	Постанова задачі	14.10.2025-23.10.2025	Виконано
4	Розгортання інфраструктури	24.10.2025-13.11.2025	Виконано
5	Конфігурація інфраструктури	14.11.2025-16.11.2025	Виконано
6	Налашутвання моніторингу	17.11.2025-01.12.2025	Виконано
7	Оформлення звіту.	01.12.2025-23.12.2025	Виконано
8	Оформлення презентації.	23.12.2025-25.12.2025	Виконано

Здобувач _____ __Максим БЕРЕЗА_
(підпис) (ім'я та прізвище)

Керівник роботи _____ __Михайло ГРАМА
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Береза Максим Юрійович – Інформаційна система аналізу корпоративних поштових повідомлень АТ "Укртелеком" з використанням методів машинного навчання.

Дипломна робота присвячена розробці інформаційної системи аналізу корпоративних поштових повідомлень АТ «Укртелеком» з використанням методів машинного навчання. Метою дослідження було створення повноцінної end-to-end інфраструктури для приймання, фільтрації, аналізу та відправлення електронної пошти з підвищеним рівнем захисту від фішингових атак та шкідливих повідомлень.

У процесі реалізації було побудовано комплексну архітектуру, що включає власний SMTP-сервер, інтеграцію з Active Directory, механізми автентифікації та авторизації, модуль машинного навчання для класифікації поштових повідомлень, систему блокування доменів/користувачів, а також аналітичну платформу на базі Grafana та Loki для моніторингу та керування політиками безпеки.

Результатом роботи стала функціональна, масштабована та надійна інфраструктура, яка забезпечує автоматизовану обробку електронних листів, інтелектуальну фільтрацію потенційно шкідливих повідомлень, оперативне виявлення фішингових загроз, централізований перегляд логів та можливість адміністрування політик безпеки через власне API та візуальні дашборди.

Ключові слова: ACTIVE DIRECTORY, SMTP, ПОШТА, ШТУЧНИЙ ІНТЕЛЕКТ, GRAPHANA, UBUNTU, KEYCLOACK, УКРТЕЛЕКОМ, ФІЛТРАЦІЯ ТРАФІКУ, МЕРЕЖА.

SUMMARY

Maksym Yuriyovych Bereza – Information system for analyzing corporate email messages at JSC Ukrtelecom using machine learning methods.

The thesis is devoted to the development of an information system for analyzing corporate email messages at JSC Ukrtelecom using machine learning methods. The aim of the research was to create a full-fledged end-to-end infrastructure for receiving, filtering, analyzing, and sending e-mails with an increased level of protection against phishing attacks and malicious messages.

During the implementation process, a comprehensive architecture was built, including a proprietary SMTP server, integration with Active Directory, authentication and authorization mechanisms, a machine learning module for classifying email messages, a domain/user blocking system, and an analytical platform based on Grafana and Loki for monitoring and managing security policies.

The result of the work was a functional, scalable, and reliable infrastructure that provides automated email processing, intelligent filtering of potentially malicious messages, rapid detection of phishing threats, centralized log viewing, and the ability to administer security policies through a proprietary API and visual dashboards.

Keywords: ACTIVE DIRECTORY, SMTP, MAIL, ARTIFICIAL INTELLIGENCE, GRAPHANA, UBUNTU, KEYCLOACK, UKRTELECOM, TRAFFIC FILTERING, NETWORK.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Скорочення / Термін	Розшифрування
AD	Active Directory — служба каталогів для керування обліковими записами користувачів та ресурсів домену
AI	Artificial Intelligence — штучний інтелект
API	Application Programming Interface — програмний інтерфейс прикладного програмування
BERT	Bidirectional Encoder Representations from Transformers — двонаправлене кодування подань на основі трансформерів
CIDR	Classless Inter-Domain Routing — безкласова система маршрутизації IP-адрес
DMARC	Domain-based Message Authentication, Reporting and Conformance — технологія автентифікації доменів електронної пошти
DKIM	DomainKeys Identified Mail — криптографічний метод підтвердження цілісності та походження електронних листів
DNS	Domain Name System — система доменних імен
Dovecot	POP3/IMAP сервер для зберігання електронної пошти
FQDN	Fully Qualified Domain Name — повне доменне ім’я
FW	Firewall — міжмережевий екран

Скорочення / Термін	Розшифрування
IAM	Identity and Access Management — керування ідентичностями та доступом
IMAP	Internet Message Access Protocol — протокол доступу до електронної пошти
IPS	Intrusion Prevention System — система запобігання вторгненням
ISPDB	Internet Service Provider Database — база автоконфігурацій поштових клієнтів (Thunderbird)
JSON	JavaScript Object Notation — текстовий формат обміну структурованими даними
LDAP	Lightweight Directory Access Protocol — протокол доступу до служби каталогів
LDAPS	LDAP over SSL/TLS — захищений варіант LDAP
L3	Layer 3 — мережевий рівень моделі OSI
L7	Layer 7 — прикладний рівень моделі OSI
LMTP	Local Mail Transfer Protocol — локальний протокол передачі пошти
MFA / 2FA	Multi-factor / Two-factor Authentication — багатофакторна / двофакторна автентифікація
Milter	Mail Filter — модуль фільтрації SMTP-з’єднання

Скорочення / Термін	Розшифрування
MX	Mail Exchange — DNS-запис, що визначає поштовий сервер домену
NAT	Network Address Translation — трансляція мережевих адрес
NLP	Natural Language Processing — обробка природної мови
OPNsense	Платформа міжмережевого екрану та маршрутизатора з відкритим кодом
OSI	Open Systems Interconnection — модель взаємодії відкритих систем
Postfix	SMTP сервер для прийому та відправлення електронної пошти
RBL	Realtime Blackhole List — реєстр IP-адрес із небажаною активністю
Realm	ізольований домен автентифікації у Keycloak
SASL	Simple Authentication and Security Layer — механізм автентифікації в SMTP/IMAP
SIEM	Security Information and Event Management — система управління подіями безпеки
SMTP	Simple Mail Transfer Protocol — протокол передавання електронної пошти
SPF	Sender Policy Framework — механізм перевірки домену відправника

Скорочення / Термін	**Розшифрування**
Suricata	IDS/IPS рушій мережевої безпеки
TLS	Transport Layer Security — протокол шифрування мережевих з’єднань
TOTP	Time-based One-Time Password — одноразовий пароль, що генерується за часом
VPN	Virtual Private Network — віртуальна приватна мережа
VirusTotal	сервіс перевірки файлів і URL на наявність шкідливих ознак
WireGuard	сучасний VPN-протокол із використанням сучасних криптографічних примітивів
URL	Uniform Resource Locator — уніфікований локатор ресурсу

ВСТУП

Стрімке зростання кількості кіберзагроз, зокрема фішингових атак, зробило захист корпоративної електронної пошти одним із ключових завдань інформаційної безпеки сучасних підприємств і державних організацій. За даними міжнародних аналітичних центрів, понад 90% успішних кібератак розпочинаються саме з електронного листа, що містить шкідливе посилання або вкладення, створене із застосуванням методів соціальної інженерії чи генеративного штучного інтелекту. Така ситуація вимагає впровадження більш ефективних технологій захисту, що дозволяють виявляти ознаки атаки не лише за сигнатурними базами, але й за глибинними семантичними характеристиками поштових повідомлень.

Особливої актуальності це набуває для телекомунікаційної галузі України, де електронна пошта виступає одним із головних каналів взаємодії з клієнтами, партнерами та державними установами. АТ «Укртелеком», як один із ключових операторів зв’язку, оперує великими масивами даних і забезпечує підтримку критичної інформаційної інфраструктури, що зумовлює підвищені вимоги до цілісності та конфіденційності електронних комунікацій. Поряд із використанням централізованих корпоративних поштових сервісів, окремі підрозділи компанії змушені застосовувати зовнішні поштові платформи, що створює додаткові поверхні атаки й потребує посилення контролю.

З огляду на це, доцільним є застосування моделей машинного навчання та методів обробки природної мови (NLP) для автоматичного аналізу змісту електронних листів і визначення ймовірності їх належності до фішингових. Критичний огляд існуючих рішень у сфері захисту електронної пошти показав, що переважна більшість з них спирається на механізми сигнатурного аналізу, статичні фільтри або репутаційні бази, що не забезпечує необхідної адаптивності до нових, динамічно змінюваних методів атак. Це обґрунтовує доцільність розроблення інтелектуальної системи, здатної самостійно виявляти приховані

закономірності у текстах повідомлень, включно з повідомленнями українською та іншими мовами.

Об’єктом дослідження у роботі виступає процес обробки та фільтрації корпоративних електронних поштових повідомлень як складова системи інформаційної безпеки підприємства. Предметом дослідження є методи, моделі та програмні засоби інтелектуального аналізу електронної пошти з використанням машинного навчання для виявлення та блокування фішингових повідомлень.

Метою роботи є підвищення рівня захисту корпоративної електронної пошти АТ «Укртелеком» шляхом створення інформаційної системи автоматичного аналізу та фільтрації поштових повідомлень на основі технологій машинного навчання, мережевої сегментації та централізованого моніторингу.

- Для досягнення поставленої мети визначено такі завдання дослідження: проаналізувати сучасні методи захисту електронної пошти та виявлення фішингових повідомлень;

- Обґрунтувати архітектуру системи інтелектуальної фільтрації пошти;– здійснити вибір моделі машинного навчання та виконати її навчання на релевантному корпусі даних;

- Розгорнути мережеву та поштову серверну інфраструктуру з використанням Postfix, Dovecot, OPNsense та WireGuard;

- Інтегрувати модель класифікації у потік поштової доставки за допомогою AI Milter та REST API;

- Забезпечити централізоване логування та моніторинг подій через Loki та Grafana;

- Впровадити підсистему керування обліковими записами Keycloak з підтримкою двофакторної автентифікації;

- Здійснити тестування системи та оцінити її ефективність.

Методи дослідження, використані в роботі, включають аналіз і узагальнення наукових джерел, методи математичного моделювання та

машинного навчання, експериментальні методи оцінювання точності класифікації, а також методи мережевого та серверного адміністрування.

Наукова новизна результатів роботи полягає у такому: вперше для корпоративної інфраструктури АТ «Укртелеком» формалізовано та реалізовано наскрізний підхід до обробки електронних листів із використанням трансформерної моделі DistilBERT; створено підхід до інтеграції машинного аналізу безпосередньо у транспортний SMTP-рівень Postfix без порушення бізнес-логіки доставки пошти; удосконалено механізм оперативного реагування завдяки реалізації API-керованих блокувань доменів та адрес із дашборду Grafana; дістала подальшого розвитку концепція поєднання моніторингу поштового трафіку з мережецентричними механізмами безпеки OPNsense і SIEM-сумісними журналами Loki.

Кваліфікаційна робота складається з 209 сторінок, 72 рисунків, 24 таблиць, 4 додатків, 40 джерел.

Кваліфікаційна робота виконувалась згідно із планом науково-дослідних робіт кафедри інформаційних технологій, штучного інтелекту і кібербезпеки Національного університету харчових технологій: НДР «Дослідження та використання сучасних інформаційних технологій для виконання функцій та завдань виробничого і організаційного управління підприємств харчової галузі» № ДР 0120U105386, 2020–2025 рр.; Дослідження та використання сучасних інформаційних технологій в освіті № ДР 0125U003888, 2025–2030 рр.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ АТ «УКРТЕЛЕКОМ»

1.1. Загальна характеристика АТ «Укртелеком»

Укртелеком, визначний оператор електрозв’язку в Україні, пройшов шлях трансформації та розвитку з моменту заснування. Початок його історії можна відстежити до 1991 року, коли після проголошення незалежності України, мережа електрозв’язку колишнього СРСР перейшла під юрисдикцію Міністерства зв'язку України.

Протягом 1994–1997 років Укртелеком став загальновизнаним державним оператором електрозв’язку. У 1998 році Кабінет Міністрів України затвердив програму реструктуризації компанії, що включала реорганізацію та акціонування. Корпоратизація Укртелекому була завершена у 1999 році, після чого в 2000 році розпочалася приватизація компанії.

У 2001 році створено Комісію з приватизації Укртелекому, що позначило початок нового етапу в розвитку компанії. У 2005 році Укртелеком отримав ліцензію на надання послуг мобільного зв’язку за технологією UMTS/WCDMA, а у 2007 році розпочав продаж послуг мобільного зв’язку третього покоління.

У 2011 році ТОВ «ЕСУ», яке належить австрійському інвестиційному фонду EPIC, придбало контрольний пакет акцій Укртелекому, ставши мажоритарним акціонером. У тому ж році Укртелеком заснував ТОВ «ТриМоб» для надання послуг мобільного зв'язку.

З 2012 року послуги мобільного зв’язку стандарту UMTS стали надаватися оператором «ТриМоб». У 2013 році Укртелеком став частиною бізнесів SCM, що зміцнило його позиції на ринку.

Починаючи з 2014 року, компанія активно працювала над трансформацією та модернізацією, впроваджуючи нові технології та сервіси. Особливу увагу приділялось розвитку інфраструктури для надання широкосмугового доступу до інтернету, зокрема через проекти оптичного зв'язку в сільській місцевості.

У 2021 році на річних загальних зборах акціонерів було змінено найменування компанії з ПУБЛІЧНОГО АКЦІОНЕРНОГО ТОВАРИСТВА «УКРТЕЛЕКОМ» на АКЦІОНЕРНЕ ТОВАРИСТВО «УКРТЕЛЕКОМ», що символізувало новий етап у її розвитку.

Таким чином, історія Укртелекому відображає його еволюцію від державного оператора електрозв’язку до сучасної інноваційної компанії, що надає широкий спектр телекомунікаційних послуг.

АТ "Укртелеком" є найбільшим оператором зв'язку в Україні, що надає повний спектр телекомунікаційних послуг по всій території країни. Основні напрямки діяльності компанії включають фіксований телефонний зв'язок, доступ до Інтернету, мобільний зв'язок, телевізійне мовлення та інші телекомунікаційні послуги. Укртелеком має розгалужену мережу, що охоплює всі регіони України, і використовує сучасні технології для забезпечення високоякісних послуг своїм абонентам.

Укртелеком сьогодні – це найбільший оператор фіксованого зв’язку в Україні, що забезпечує стале надання вкрай важливих в умовах війни телекомунікаційних сервісів для ЗСУ, критичної інфраструктури, державних органів, бізнесу та українців.

Товариство активно розвиває інфраструктуру широкосмугового доступу до Інтернету, впроваджуючи технології ADSL, VDSL та оптико-волоконні лінії зв'язку до кінцевих користувачів. Це дозволяє надавати високошвидкісний доступ до Інтернету, цифрове телебачення та інші сучасні телекомунікаційні послуги.

АТ "Укртелеком" також удосконалює напрямок цифрових сервісів та рішень для бізнесу, включаючи хмарні рішення, віртуальні АТС, послуги відеоконференцзв'язку та інші ІТ-послуги, що дозволяє компанії бути конкурентоспроможною на ринку телекомунікаційних послуг [1].

Товариство прагне до найвищого рівня обслуговування клієнтів. Компанія вкладає значні кошти у модернізацію мережі та розширення спектра послуг, щоб відповідати очікуванням сучасних користувачів.

Зважаючи на масштаби компанії та велику кількість клієнтів, питання інформаційної безпеки є одним з пріоритетних. Товариство використовує комплексний підхід до захисту даних, який включає:

- Поштовий фільтр Exchange: Цей елемент захисту блокує більшість фішингових атак та шкідливих листів, але 100% гарантії не дає.
- ШІ-систему Immuniveb: Ця система використовується для пошуку та запобігання витокам корпоративних даних, що значно знижує ризики кіберзлочинів.
- Суворі протоколи та процедури: Компанія має чіткі правила та інструкції щодо інформаційної безпеки, які зобов'язані дотримуватися всі співробітники.

"Укртелеком" постійно вдосконалює свої системи захисту та впроваджує новітні технології, щоб гарантувати безпеку даних своїх клієнтів та співробітників.

1.2. Організаційна структура департаменту безпеки ІТ АТ «Укртелеком».

Департамент безпеки ІТ в АТ "Укртелеком" відіграє ключову роль у захисті компанії від сучасних кіберзагроз, забезпечуючи безпеку та надійність інформаційних систем і даних. Діяльність департаменту охоплює різні аспекти інформаційної безпеки, включаючи захист від кіберзагроз, аудити безпеки, впровадження заходів безпеки, моніторин і реагування на інциденти інформаційної безпеки, навчання працівників компанії методами проведення симуляцій фішингових атак, моніторинг систем та сервісів та їх налаштування, управління серверами та обробку трафіку, налаштування міжмережевих екранів.

Вся діяльність департаменту безпеки ІТ спрямована на досягнення спільної мети – забезпечення інформаційної безпеки користувачів та отримувачів послуг компанії. Також основною метою департаменту вдосконалення методів розпізнавання інцидентів інформаційної безпеки для запобігання витоків

інформації або та збитків від них. Для досягнення цієї мети, департамент впроваджує сучасні технологій технології та методи захисту, проводить регулярні аудити й оцінки безпеки, а також розробляє нові політики, алерти та процедури, які сприяють підвищенню рівня кіберзахисту компанії.

Загальну організаційну структуру департаменту безпеки IT зображено на рисунку 1.1.

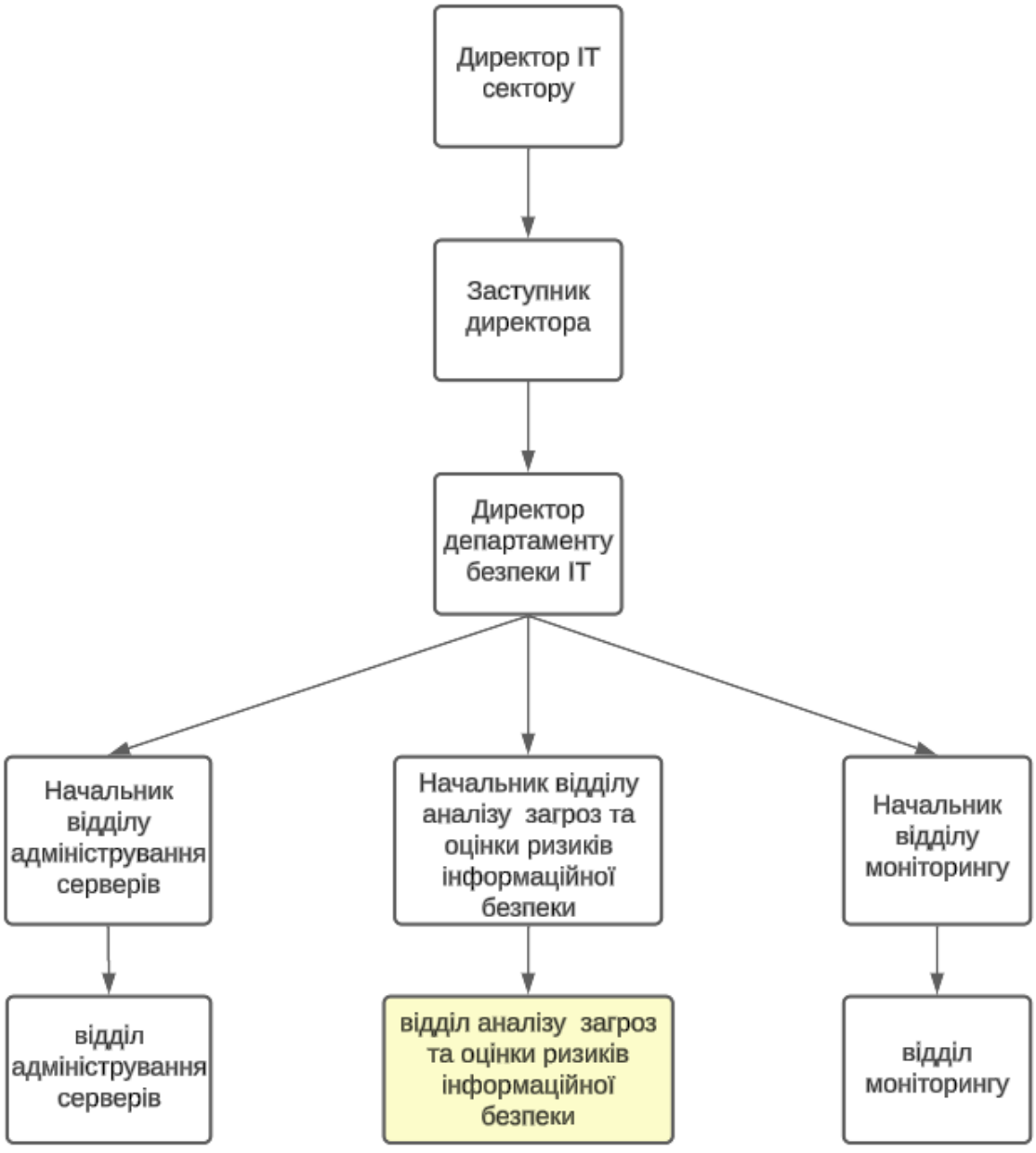

Рисунок 1.1 – Організаційна структура департаменту безпеки IT

Основні функції, що виконує департамент безпеки IT наведені в таблиці 1.1.

Таблиця 1.1 – Функції та задачі департаменту безпеки IT.

№	Задачі	Функції
1	Захист від кіберзагроз	- Розробка та впровадження заходів для запобігання кібератак, виявлення та розслідування інцидентів.
2	Аудит безпеки	- Проведення та створення регулярних аудитів та оцінок безпеки систем, мереж та додатків для виявлення потенційних вразливостей.
3	Впровадження заходів безпеки	- Розробка та впровадження політик безпеки, шифрування даних(DLP) та інших технічних засобів захисту.
4	Моніторинг та реагування	- Постійний моніторинг інформаційних систем для виявлення аномальних активностей та негайна реакція на потенційні загрози.
5	Навчання персоналу	- Проведення навчань та інформаційних кампаній для персоналу, та розроблення спеціалізованих курсів щодо правил кібербезпеки та запобігання атак методами соціальної-інженерії.
6	Налаштування міжмережевих екранів	- Конфігурування політик безпеки на рівнях L3 та L7, налаштування систем IPS для виявлення й блокування загроз, оптимізація фільтрації трафіку та сегментації мережі для підвищення рівня захисту.

Продовження таблиці 1.1.

№	Задачі	Функції
7	Моніторинг обладнання і сервісів	- Моніторинг мережевої доступності сервісів та відстеження стану обладнання.
8	Управління серверами	- Налаштування, моніторинг та підтримка серверів для забезпечення їхньої надійності та продуктивності.
9	Обробка та аналіз інцидентів інформаційної безпеки	- Обробка заявок користувачів на предмети фішингу та заражених вкладень в електронних листах та реагування на інциденти інформаційної безпеки в системі SIEM.
10	Налаштування антивірусу	- Встановлення, конфігурування та централізоване управління антивірусними рішеннями на робочих станціях і серверах. Забезпечення регулярного оновлення сигнатур, моніторинг стану захисту та реагування на виявлені загрози.
11	Налаштування MDM	- Впровадження та адміністрування системи Mobile Device Management для контролю корпоративних пристроїв, політик безпеки, шифрування даних і дистанційного видалення у разі компрометації.

Продовження таблиці 1.1.

№	Задачі	Функції
12	Налаштування політик Conditional Access	- Розробка та впровадження політик умовного доступу для контролю автентифікації користувачів, перевірки пристроїв і обмеження доступу до корпоративних ресурсів на основі рівня ризику.
13	Розробка та впровадження автоматизацій	- Розробка та впровадження автоматизацій з метою автоматизації та миттєвої реакції на інциденти інформаційної безпеки.

Відділ адміністрування систем інформаційної безпеки в АТ "Укртелеком" є важливою частиною департаменту безпеки IT і зосереджується на налаштуванні та підтримці систем інформаційної безпеки. Фішинг є однією з найбільш розповсюджених та небезпечних форм кіберзлочинності, що націлена на отримання конфіденційної інформації користувачів шляхом обману. Спеціалісти відділу працюють над запобіганням таких атак, забезпечуючи захист компанії та її клієнтів.

Структуру відділу аналізу загроз та оцінки ризиків інформаційної безпеки зображено на рисунку 1.2.

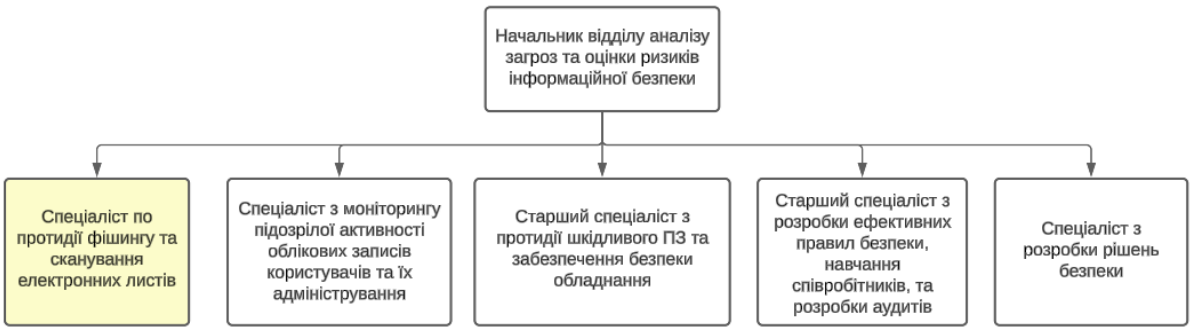

Рисунок 1.2 – Структура відділу загроз та оцінки ризиків інформаційної безпеки

Спеціалісти відділу аналізу загроз та оцінки ризиків інформаційної безпеки відіграють ключову роль у захисті компанії від сучасних кіберзагроз, забезпечуючи безпеку та надійність інформаційних систем і даних.

Основні обов'язки відділу включають:

1. Протидія фішингу та сканування електронних листів:

- Розробка та впровадження заходів для запобігання фішинговим атакам.
- Постійне сканування вхідних електронних листів для виявлення підозрілих елементів та можливих фішингових спроб.
- Використання спеціалізованого програмного забезпечення та алгоритмів для автоматичного виявлення та блокування фішингових листів.

2. Моніторинг підозрілої активності облікових записів користувачів та їх адміністрування:

- Аналіз активності користувачів для виявлення аномальних дій, які можуть свідчити про спроби фішингу або інші загрози.
- Адміністрування облікових записів з метою забезпечення їхньої безпеки та запобігання несанкціонованому доступу.

3. Науково-дослідна робота та розробка ефективних правил безпеки:

- Проведення досліджень з метою вдосконалення методів виявлення та протидії фішингу.
- Розробка та впровадження ефективних правил та політик безпеки для запобігання фішинговим атакам.

4. Навчання співробітників:

- Організація та проведення навчальних сесій та тренінгів для співробітників компанії з метою підвищення їх обізнаності про загрози фішингу.
- Розробка інформаційних матеріалів та інструкцій щодо безпечного використання електронної пошти та інших засобів комунікації.

5. Обробка та аналіз інцидентів інформаційної безпеки:

- Реагування на інциденти, пов'язані з фішингом, та їх розслідування.
- Впровадження заходів для мінімізації наслідків інцидентів та запобігання їх повторенню в майбутньому.

1.3. Аналіз нинішнього стану сканування електронних листів

Аналіз сучасного стану технологій сканування електронних листів показує, що навіть у корпоративних середовищах рівень автоматизації перевірки вхідних повідомлень залишається неповним. Більшість рішень орієнтовані лише на базові сигнатурні перевірки, а не на глибинний аналіз вмісту з використанням штучного інтелекту.

Традиційні поштові сервери, такі як Postfix, Exim або Microsoft Exchange, забезпечують базовий рівень захисту через антиспам-та антивірусні фільтри, проте їх ефективність значною мірою залежить від оновлення сигнатур і не враховує контекст листа чи соціотехнічні ознаки фішингу. Додаткові рішення, як-от SpamAssassin, Rspamd або ClamAV, можуть бути інтегровані у поштову інфраструктуру, але вони не забезпечують глибокого семантичного аналізу повідомлення.

Однією з проблем сучасних систем є те, що контентна перевірка часто виконується лише після доставки листа у поштову скриньку, що створює ризик для користувача взаємодіяти з потенційно небезпечним контентом. Крім того, такі рішення не враховують репутацію домену, поведінкові ознаки або ознаки, характерні для сучасних фішингових кампаній.

У більшості випадків корпоративні організації використовують хмарні сервіси (наприклад, Microsoft Defender for Office 365, Google Workspace Security, Proofpoint або Barracuda Email Security), які інтегрують антиспам-фільтрацію, перевірку вкладень і sandbox-аналіз.

Попри високу ефективність, такі сервіси є дорогими і мало придатними для локального впровадження в організаціях з власною інфраструктурою.

Саме тому актуальним стає підхід до побудови власної системи фільтрації електронної пошти, яка поєднує:

- відкриті рішення (Postfix, Dovecot, OPNsense) з повним контролем на рівні мережі;

- AI-компонент для аналізу тексту та вкладень на основі моделі DistilBERT;
- інтеграцію з VirusTotal для репутаційного аналізу доменів і файлів;
- централізований моніторинг і візуалізацію у Grafana.

Розроблена система усуває типові недоліки комерційних рішень, дозволяючи:

- працювати офлайн у власній інфраструктурі;
- гнучко налаштовувати логіку блокування;
- інтегрувати інтелектуальні моделі без додаткових ліцензійних витрат.

Таким чином, сучасний стан технологій демонструє потребу у впровадженні інтелектуальної системи сканування пошти, що забезпечує баланс між безпекою, автономністю та адаптивністю. Це стало підґрунтям для створення власної системи фільтрації, описаної в наступних розділах роботи.

1.4. Технічне завдання

1.4.1 Загальні положення

1.4.1.1. Найменування системи: «Інформаційна система аналізу корпоративних поштових повідомлень АТ "Укртелеком" з використанням методів машинного навчання».

1.4.1.2. Оформлення та передача результатів роботи зі створення системи здійснюється відповідно до вимог ДСТУ, встановлених для кожного етапу розробки. Конкретний порядок оформлення та передачі результатів визначається залежно від змісту та календарного плану проекту.

1.4.1.3. Під час подальших етапів роботи над створенням системи можуть виникати потреби в додатковому уточненні та розвитку окремих положень.

1.4.2. Призначення і цілі створення системи.

1.4.2.1. Призначення системи.

Система призначена для автоматизованої обробки та фільтрації

електронної пошти з метою виявлення фішингових і шкідливих повідомлень. Вона забезпечує аналіз текстового вмісту, вкладень і посилань за допомогою нейронної мережі DistilBERT та зовнішніх сервісів перевірки (VirusTotal API, DeepL API), інтегруючись із поштовим сервером Postfix через модуль AI Milter. Рішення виконує класифікацію листів на етапі їх прийому, визначає ступінь ризику та приймає рішення про блокування, ізоляцію або доставку до користувача через Dovecot (LMTP). Система реалізує наскрізний контроль поштового трафіку від рівня DNS (Cloudflare) до зберігання повідомлень, забезпечуючи високий рівень безпеки та прозорість моніторингу.

1.4.2.2. Цілі створення системи.

Метою створення системи є підвищення безпеки електронної пошти за рахунок впровадження інтелектуальної фільтрації на основі глибинного навчання. Основна задача полягає в автоматичному виявленні фішингових повідомлень на основі аналізу їхнього змісту, структури та контексту. Система має забезпечити точну класифікацію, високу швидкодію та можливість розширення функціоналу шляхом додавання нових перевірок і донавчання моделі. Реалізація рішення спрямована на зниження ризиків компрометації облікових записів, зменшення навантаження на адміністратора безпеки та створення централізованого механізму обробки поштових інцидентів.

1.4.3. Характеристика об’єкта автоматизації

1.4.3.1. Короткі відомості про об’єкт автоматизації.

Об’єктом автоматизації є корпоративна система електронної пошти, що функціонує в межах внутрішньої доменної інфраструктури corp.local з інтеграцією зовнішнього домену test-app.org, зареєстрованого через Cloudflare. Система включає поштовий сервер на базі Ubuntu Server 24.04 LTS із компонентами Postfix, Dovecot, та підсистемами захисту й маршрутизації, розгорнутими в середовищі Oracle VirtualBox.

У межах локальної мережі реалізовано підключення до контролера домену Active Directory (Windows Server 2012 R2) для централізованої

аутентифікації користувачів, а також захищене з’єднання між вузлами через OPNsense Firewall і WireGuard VPN-тунель.

Об’єкт автоматизації забезпечує обробку, зберігання та передачу поштових повідомлень у корпоративному середовищі з можливістю інтеграції модулів штучного інтелекту для фільтрації контенту. Таким чином, він виступає як критичний елемент інформаційної інфраструктури підприємства, що потребує підвищеного рівня автоматизації процесів безпеки та адміністрування.

1.4.4. Вимоги до системи

Вимоги до системи у цілому:

Система фільтрації електронної пошти на основі штучного інтелекту повинна забезпечувати стабільну роботу, безпечну інтеграцію з корпоративним середовищем і повну автоматизацію процесів контролю та моніторингу поштових інцидентів.

Вимоги до функцій (завдань), що виконуються системою:

1.4.4.1. Функціональні вимоги:

- Забезпечити прийом, аналіз і класифікацію поштових повідомлень у реальному часі на етапі SMTP-сеансу.
- Виконувати класифікацію контенту на категорії: legit, suspicious, phishing.
- Підтримувати автоматичне сканування вкладень і посилань через VirusTotal API.
- Забезпечити багатомовний аналіз листів із перекладом неангломовного контенту через DeepL API.
- Інтегруватися з Postfix (SMTP) і Dovecot (LMTP) через компонент AI Milter.
- Вести детальний журнал усіх дій системи у форматі JSON для подальшої аналітики.
- Надсилати узагальнені дані класифікації у Grafana для моніторингу інцидентів у реальному часі.

- Передбачити можливість швидкої зміни конфігурації та перезапуску без втрати поточних процесів.

1.4.4.2. Технічні вимоги

- Основна платформа: Ubuntu Server 24.04 LTS, Python 3.12, FastAPI, Postfix, Dovecot.

- AI API може працювати на Windows Server 2022 / 11 Pro або Linux, з доступом по HTTP/8000.

- Використання GPU (CUDA) для прискорення обчислень при наявності сумісного обладнання.

- Модель DistilBERT повинна зберігатися локально для зниження затримок при обробці.

- Всі з’єднання повинні бути захищені TLS/SSL (порти 25, 465, 587, 993).

- Мінімальні системні ресурси:
  - CPU: 4 ядра;
  - RAM: 8 ГБ;
  - Диск: 40 ГБ.

- Підтримка одночасної обробки не менше 10 листів без деградації продуктивності.

- Автоматичне логування у /var/log/mlmilter/ або logs/inference.log із ротацією раз на тиждень.

1.4.4.3. Вимоги до безпеки

- Усі з’єднання між компонентами (Postfix ↔ API ↔ Dovecot) мають бути зашифрованими.

- Аутентифікація користувачів виконується через Active Directory (LDAPS 636).

- Доступ до поштових скриньок реалізується виключно через IMAPS (993) або STARTTLS (587).

- Ключі та токени API (VirusTotal, DeepL) повинні зберігатися у захищеному середовищі.

- Адміністративний доступ дозволено лише через VPN/WireGuard або

локальну мережу.

- Повинно здійснюватися централізоване журналювання усіх інцидентів для аудиту.

1.4.4.4. Експлуатаційні вимоги

- Сервіси повинні автоматично запускатися після перезавантаження системи.
- Забезпечити можливість моніторингу продуктивності та логів через веб-інтерфейс.
- Час відповіді AI API на один запит не повинен перевищувати 2 секунд.
- Модель повинна підтримувати донавчання без зупинки сервісу.
- Система має бути масштабованою (можливість додавання вузлів API або Milter).
- Після збою — автоматичний перезапуск процесів через systemd.

1.4.4.5. Вимоги до моніторингу (Grafana)

- Усі журнали роботи AI Milter, Postfix і Dovecot повинні передаватися до Loki для подальшого візуального відображення у Grafana.
- На дашбордах Grafana мають бути представлені:
    - кількість оброблених листів (за годину/день/тиждень);
    - топ відправників із підозрілими листами;
    - статуси запитів до VirusTotal API;
- Передбачити можливість експорту звітів у форматах CSV, PDF.
- Система моніторингу має працювати в режимі 24/7, підтримуючи push-сповіщення у Email.
- Доступ до Grafana повинен бути обмежений ролями (Admin, Analyst, Viewer).

1.4.4.6. Вимоги до мережевої безпеки (OPNsense Firewall)

- Уся комунікація між зовнішнім VPS і локальною інфраструктурою має здійснюватися через шифрований WireGuard VPN-тунель (порт 52180/UDP). Це забезпечує конфіденційність переданих даних і захист від MITM-атак.

- OPNsense Firewall повинен виконувати роль центрального вузла мережевої безпеки та забезпечувати:
    - NAT-трансляцію для вузлів Postfix, Dovecot і AI API;
    - розмежування зон безпеки (LAN, DMZ, VPN, WAN) з індивідуальними наборами політик;
    - моніторинг IDS/IPS (Suricata) для виявлення підозрілої активності, аналізу пакетів і блокування атак у реальному часі;
    - GeoIP-блокування вхідного та вихідного трафіку з країн, які не входять до зони довіри (наприклад, країни підвищеного ризику кіберзагроз).
- Трафік має бути повністю фільтрованим за політиками безпеки:
    - усі вхідні з’єднання — заборонені за замовчуванням, окрім явно дозволених;
    - вихідний трафік — дозволяється лише для службових портів (SMTP, HTTPS, DNS, API);
    - IPS/IDS-захист повинен діяти як на вхідних, так і на вихідних потоках даних, щоб запобігти як зовнішнім, так і внутрішнім загрозам.
- Вхідні з’єднання дозволяються лише на такі порти:
    - 25 (SMTP — прийом пошти);
    - 465 / 587 (SMTPS / Submission — відправлення);
    - 993 (IMAPS — доступ користувачів до пошти);
    - 443(TCP-доступ до web сервісу);
    - 8000 (AI API — класифікаційний сервер).
- Реалізується централізоване журналювання усіх мережевих подій:
    - логи дозволених/заборонених з’єднань;
    - логи IPS/IDS-подій;
    - статистика заблокованих країн і IP-адрес.

- Конфігурація OPNsense має зберігатися у резервній копії та перевірятися не рідше одного разу на тиждень. Додатково, система повинна виконувати періодичну перевірку доступності VPN-тунелю (Health Monitoring) з автоматичним відновленням з’єднання у випадку збою.

- Для підвищення рівня кіберзахисту трафік має бути:
    - IDS/IPS-захищеним — із виявленням аномалій, вторгнень та сигнатурних атак;
    - Geo-захищеним — із застосуванням політик доступу за географічними регіонами;
    - сегментованим — внутрішня пошта (LAN ↔ DMZ) відокремлена від зовнішнього SMTP-трафіку.

У таблиці 1.2 перелічені функції вхідної та вихідної інформації.

Таблиця 1.2 – Перелік функцій, вхідної та вихідної інформації

№	Підсистема	Вхідна інформація	Вихідна інформація
1	Cloudflare DNS	DNS-запити щодо MX, SPF, DKIM, DMARC записів	IP-адреса поштового сервера, перевірені DNS-записи для маршрутизації пошти
2	Postfix (SMTP-сервер)	Вхідні SMTP-з’єднання, заголовки та тіло листа, метадані (From, To, Subject)	Передача листа у AI Milter, журнал логів поштового трафіку
3	AI Milter	Повідомлення з Postfix у форматі RFC822	JSON-запит до AI API, результат аналізу (phishing_score, verdict, action)

Продовження таблиці 1.2

№	Підсистема	Вхідна інформація	Вихідна інформація
4	AI API (FastAPI-сервер)	Запити від Milter (headers, body, urls, attachments)	Відповідь JSON з результатом класифікації, оцінкою ризику, дією (accept/reject/quarantine)
5	Нейромережевий модуль DistilBERT	Попередньо оброблений текст листа	Ймовірність належності повідомлення до класу “phishing” або “legit”
6	VirusTotal інтеграція	URL-адреси та SHA256-хеші вкладень	Репутаційний звіт: malicious, suspicious, harmless, undetected; коригування phishing_score
7	DeepL API (модуль перекладу)	Виявлений неангломовний текст листа	Перекладений англійський варіант тексту для аналізу
8	Dovecot (IMAP/LMTP-сервер)	Безпечні листи, прийняті після класифікації	Доставка повідомлень користувачу у Maildir, синхронізація поштових скриньок
9	Active Directory (LDAP)	Запит на автентифікацію користувача	Відповідь про валідність облікових даних (успішно / відмовлено)

Продовження таблиці 1.2

№	Підсистема	Вхідна інформація	Вихідна інформація
10	Grafana + Loki	Логи класифікації, результати AI API, події безпеки	Дашборди моніторингу, графіки класифікацій, статистика інцидентів, сповіщення
11	OPNsense Firewall	Вхідний/вихідний трафік між компонентами системи	Пропущений або заблокований трафік, IDS/IPS-звіти, GeoIP-фільтрація
12	Admin Control API (Python)	Запити адміністратора на блокування/розблокування доменів або відправників; фільтри з Grafana	Оновлений список заблокованих об’єктів, журнал дій адміністратора, інтеграція з Grafana Dashboard
13	Система автоматичного створення поштових скриньок (LDAP → Dovecot Sync Script)	Дані користувачів з OU *emailusers* у Active Directory; облікові атрибути (mail, sAMAccountName)	Автоматично створені каталоги Maildir, стандартні папки (Inbox, Sent, Junk, Trash), оновлення списку користувачів у Dovecot

1.4.4.3. Вимоги до доступності програми.

Система повинна забезпечувати високий рівень доступності (High Availability) усіх своїх компонентів, включаючи поштові служби (Postfix, Dovecot), AI Milter, AI API та Grafana.

Основні вимоги:

- Безперервна робота – програма має функціонувати у режимі 24/7 без критичних простоїв. Планові оновлення та перезапуски сервісів виконуються без втрати даних і з мінімальною тривалістю переривань (до 1 хвилини).

- Автоматичне відновлення – усі критичні сервіси повинні бути зареєстровані як systemd-служби з параметром Restart=always, що гарантує їх автоматичний перезапуск у випадку збою.

- Доступність API – AI API повинен відповідати на HTTP-запити з часом відгуку не більше 2 секунд при стандартному навантаженні (до 10 запитів одночасно).

- Моніторинг стану – працездатність компонентів контролюється через дашборди Grafana, що відображають статус сервісів, кількість запитів, середній час обробки, помилки та інциденти безпеки.

- Захищений доступ – усі інтерфейси адміністрування (Grafana, API, SSH) мають бути доступні лише з внутрішньої мережі або через VPN (WireGuard).

1.4.4.4. Вимоги до стандартизації та уніфікації

Розроблена система повинна відповідати чинним стандартам у сфері розробки, інформаційної безпеки та поштових комунікацій.

Основні принципи стандартизації:

- Використання стандартних протоколів:
    - SMTP/SMTPS (RFC 5321, RFC 6409) – передача поштових повідомлень;
    - IMAP (RFC 3501) – доступ до поштових скриньок;
    - TLS 1.2/1.3 – шифрування з’єднань;
    - LDAPS (RFC 4511) – взаємодія з Active Directory.

- Уніфікація форматів даних – усі обмінні повідомлення між підсистемами реалізовані у форматі JSON із фіксованою структурою полів (headers, body, urls, attachments, score, verdict).

- Кросплатформеність – програмні модулі (AI API, Admin Control API, Milter) реалізовано на Python із підтримкою запуску як у Linux, так і у Windows

середовищах.

- Єдність інтерфейсів адміністрування – моніторинг, логування та керування системою виконуються через Grafana, що уніфікує керування незалежно від платформи.

- Використання відкритих стандартів – для нейромережевих моделей застосовано бібліотеки з відкритим кодом (HuggingFace Transformers, PyTorch), що гарантує повторюваність експериментів і відповідність академічним вимогам.

- Стандарти безпеки – дотримання вимог ISO/IEC 27001 щодо конфіденційності, цілісності та доступності даних, а також застосування політики мінімальних привілеїв для всіх системних облікових записів.

1.4.4.5. Ліцензійні вимоги

Система повинна використовувати лише ліцензійне програмне забезпечення.

Спеціалізоване ПЗ, що розробляється власними силами виконавця, повинне використовувати ліцензійні компоненти (бібліотеки) і засоби розробки.

1.4.5. Склад і зміст робіт під час створення системи.

1. Аналіз і визначення вимог:

- Збір і аналіз вимог до системи: типи користувачів, ролі, функціональні потреби.

- Визначення функціональних і нефункціональних вимог (захист, продуктивність, доступність).

- Розроблення специфікації вимог до поштової інфраструктури.

Дата: 15.09.2025

2. Проєктування архітектури системи:

- Розроблення логічної та фізичної схеми взаємодії Postfix, Dovecot, AI Milter, AI API, Active Directory, Grafana, OPNsense.

- Визначення мережевих портів, протоколів і форматів обміну (SMTP, IMAP, HTTP/JSON).

Дата: 26.09.2025

3. Розроблення та навчання моделі штучного інтелекту:

- Підготовка датасету (очищення, нормалізація, розподіл на вибірки).

- Навчання кількох моделей (Naive Bayes, Logistic Regression, SVM, DistilBERT).

- Вибір DistilBERT як основної моделі класифікації.

Дата: 10.10.2025

4. Розроблення AI API та інтеграційних модулів:

- Реалізація FastAPI сервера для класифікації листів і взаємодії з VirusTotal, DeepL API.

- Налагодження AI Milter для зв’язку з Postfix і передачі результатів класифікації.

Дата: 20.10.2025

5. Інтеграція поштових компонентів:

- Налаштування Postfix, Dovecot, Active Directory, TLS/SSL.

- Автоматичне створення поштових скриньок (LDAPS → Dovecot Maildir).

- Налаштування OPNsense Firewall з підтримкою IPS/IDS та GeoIP.

Дата: 30.10.2025

6. Моніторинг і візуалізація:

- Інтеграція Grafana + Loki для збору логів, побудови дашбордів і контролю класифікацій.

- Розгортання API для адміністрування (блокування/розблокування доменів, керування списками).

Дата: 15.11.2025

7. Тестування системи:

- Перевірка коректності класифікації листів, стабільності API, взаємодії компонентів.

- Проведення функціонального та навантажувального тестування.

Дата: 23.11.2025

1.4.6. Порядок контролю і приймання системи.

1.4.6.1. Види, склад, об’єми і методи випробувань системи та її складових частин

У процесі розробки системи виконуються наступні види випробувань:

- тестування елементів системи
- тестування системи у цілому
- дослідна експлуатація

Крім того, при впровадженні комплексної системи захисту інформації, проводяться додаткові випробування, передбачені інструкціями ДСТЗІ.

1.4.6.2. Загальні вимоги до приймання робіт за стадіями

- Після закінчення відповідного етапу робіт формується комплект документації, передбаченої п.5.

- Завершення етапу фіксується відповідним протоколом між Виконавцем і Замовником.

1.4.7. Вимоги до складу і змісту робіт із підготовки до впровадження системи.

Підготовка до запуску системи в дію передбачає виконання ряду правил і вимог для забезпечення успішної і ефективної роботи системи. Правила та вимоги включають:

- укомплектування та підготовка технічних засобів;

- перед запуском системи необхідно провести повне тестування, включаючи функціональне тестування, тестування на відмовостійкість та продуктивність;

- проведення інструктажу з використання системи, ознайомлення користувачів із функціональністю системи та правилами експлуатації;

1.4.8. Вимоги до документації.

Документування системи повинне виконуватися у відповідності з вимогами ГОСТ 34.201-89.

Спеціалізовані елементи системи, що поставляються підрядними організаціями, повинні супроводжуватися відповідним комплектом документації, а також (коли застосовано) сертифікатами відповідності.

1.4.9. Джерела розробки.

При розробленні технічного завдання на систему використано наступні документи:

1. ДСТУ ISO/IEC/IEEE 29119-1:2017 — Інженерія систем і програмних засобів. Тестування програмних засобів. Частина 1. Поняття та визначення. Використано при розробленні плану тестування компонентів (Postfix, AI Milter, API, Dovecot) та визначенні метрик продуктивності й надійності [2].
2. ДСТУ ISO/IEC 29155-1:2015 — Розроблення систем і програмного забезпечення. Платформи для тестування проєктів з розроблення інформаційних систем. Частина 1. Концепції та визначення. Застосовано при організації середовища тестування у віртуалізованому середовищі (Ubuntu 24.04 LTS, Windows Server 2022, WireGuard-тунель) [3].
3. ДСТУ ISO/IEC 12207:2016 — Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення. Використано при описі процесів розроблення, верифікації, експлуатації та супроводу компонентів системи [4].
4. ДСТУ ISO/IEC 27001:2015 (ISO/IEC 27001:2013, IDT) — Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги використано при визначенні політик безпеки системи, захисту даних користувачів, шифрування трафіку, управління обліковими записами (LDAP/LDAPS), а також при організації журналювання та контролю доступу в Grafana і OPNsense [5].

5. RFC 5321, RFC 3501, RFC 8314, RFC 6376, RFC 7208 — міжнародні стандарти, що регламентують роботу поштових протоколів (SMTP, IMAP, TLS, DKIM, SPF) і використані для забезпечення сумісності системи з іншими поштовими сервісами [6-10].

1.5. Висновоки до першого розділу

У ході аналізу організаційних та технічних аспектів роботи АТ «Укртелеком» встановлено, що компанія, як один із ключових телекомунікаційних операторів України, оперує значними обсягами електронної кореспонденції, що робить захист поштового каналу критичним елементом інформаційної безпеки. Дослідження структури департаменту IT-безпеки засвідчило, що в компанії існують сформовані процеси реагування на інциденти та контролю над корпоративними сервісами, однак поштовий трафік не завжди перевіряється централізовано, особливо коли комунікація відбувається поза межами корпоративної інфраструктури.

Виявлено, що наявні засоби виявлення фішингових та шкідливих листів базуються переважно на функціональності Microsoft Exchange / Microsoft Defender. Ці інструменти ефективні для частини внутрішньої кореспонденції, однак значна частка робочих комунікацій співробітників здійснюється через зовнішні поштові домени, зокрема Gmail, який не контролюється засобами корпоративного захисту. У таких випадках перевірка підозрілих повідомлень часто відбувається вручну, що створює додаткові навантаження на фахівців безпеки, збільшує час реагування та підвищує ризик пропуску атак.

Проведений аналіз підтвердив нагальну потребу у створенні єдиної системи автоматизованого аналізу електронних листів, здатної виявляти спроби фішингу на основі інтелектуальних методів класифікації, блокувати небезпечних відправників або домени, інтегрувати результати перевірки в систему моніторингу та оперативного реагування, а також мінімізувати частку ручних дій. Саме ці вимоги були сформовані в технічному завданні, що передбачає

використання централізованого поштового сервера, Milter-фільтра для машинного аналізу повідомлень, механізмів шифрування трафіку, а також підсистеми журналювання та візуального моніторингу подій.

Таким чином, результати першого розділу стали обґрунтуванням розробки та впровадження комплексної системи фільтрації та аналізу електронної пошти, яка покликана підвищити рівень кіберстійкості АТ «Укртелеком», знизити ризики соціальної інженерії, оптимізувати навантаження на департамент безпеки та забезпечити контроль над усіма каналами електронної комунікації — як внутрішніми, так і зовнішніми.

РОЗДІЛ 2. Дослідження та обґрунтування технологій

2.1. Аналіз існуючих рішень у сфері поштової безпеки

Захист корпоративної електронної пошти це один із ключових напрямків інформаційної безпеки підприємства. Сучасні загрози такі як фішинг, маскування відправника, шкідливі вкладення, компрометація облікових записів (BEC — Business Email Compromise) змушують організації впроваджувати спеціалізовані рішення з фільтрації, моніторингу та реагування. На ринку існує кілька корпоративних платформ, що пропонують такі функції. Нижче наведено аналіз трьох поширених рішень - їх призначення, переваги, недоліки та орієнтовні ціни.

2.1.1. Microsoft Defender for Office 365

Microsoft Defender for Office 365 (раніше Office 365 Advanced Threat Protection) — хмарне рішення, інтегроване в екосистему Microsoft 365, яке призначене для захисту корпоративної пошти (вхідної, вихідної, внутрішньої) від фішингу, шкідливих вкладень, атак типу BEC.

Функціональні можливості:

- Safe Attachments — аналіз вкладень, в тому числі нульового дня, із використанням машинного навчання і «пісочниці».
- Safe Links — динамічна перевірка URL-адрес, які містяться в листах або документах; посилання можуть бути переписані і знову перевірені у момент натискання.
- Антифішинг політики — кілька моделей машинного навчання, аналіз заголовків, поведінки, контексту.
- Звітність в реальному часі, Threat Explorer, аналітика для SOC.

Переваги використання:

- Висока інтеграція з Microsoft 365, Azure AD, Exchange Online — зменшує складність впровадження для організацій, які вже використовують Microsoft.

- Сучасні функції машинного навчання, аналіз вкладень, URL-перевірка, превентивні заходи.

- Централізована платформа управління, звітності та кореляції загроз.

Недоліки використання:

- Залежність від хмарної інфраструктури Microsoft — для повністю локальних середовищ може бути обмеження.

- Вартість може бути значною при великій кількості користувачів.

Ціна :

- План 1 (Plan 1) — ~2 USD/користувач/місяць при річному зобов'язанні.

- План 2 (Plan 2) — ~5 USD/користувач/місяць (може варіюватись залежно від країни) [12].

2.1.2. Proofpoint Email Protection

Proofpoint Email Protection — корпоративне рішення для захисту електронної пошти, яке охоплює антиспам, антивірус, захист від фішингу, BEC-атак, а також можливості ізоляції підозрілих вкладень, глибокої аналітики, інтелектуальної поведінкової перевірки.

Функціональні можливості:

- Динамічна класифікація листів (спам, фішинг, імпостор, масові розсилки) на основі багато-рівневого захисту.

- Вбудований машинний аналіз дій: аналіз відправників, поведінки користувачів, репутації серверів, IP, доменів.

- Підтримка DLP (data loss prevention), архівування, шифрування пошти.

- Підтримка як хмарного, так і on-premises розгортання.

Переваги:

- Орієнтація на виявлення складних атак (спуфінг, BEC, імпостор), з глибоким аналізом відправників і їх поведінки.

- Гнучкі політики безпеки, підтримка on-premises чи гібридних сценаріїв — підходить для організацій із високими вимогами до контролю.

Недоліки:

- Висока вартість впровадження та ліцензування — може бути великою для малих організацій.

- Розгортання та налаштування можуть потребувати значних ресурсів і часу.

Ціна:

- Виробник зазначає, що ціни налаштовуються індивідуально відповідно до розміру організації та політик.

- У порівняльних оглядах згадуються ціни ~5.95-8.71 USD/користувач/місяць [13].

2.1.2. Darktrace / EMAIL

Darktrace / EMAIL — це хмарно-орієнтоване AI-рішення для захисту корпоративної електронної пошти, яке використовує самонавчальну (self-learning) штучну інтелектуальну систему для аналізу поведінки користувачів, зв’язків між ними, стилю листування, тональності, а також перевірки вхідних, вихідних і внутрішніх повідомлень.

Рішення позиціонує себе як доповнення (або заміна) традиційним фільтрам пошти та антивірусу, орієнтуючись на виявлення нових/невідомих загроз – наприклад, BEC (Business Email Compromise), зламані облікові записи, соціальна інженерія, атаки через ланцюги постачальників.

Функціональні можливості:

- Аналіз «нормального» профілю користувача: хто з ким листується, у якій тональності, скільки часу, які посилання/домен-відправник.

- Вхідна, вихідна та внутрішня пошта: відстежує аномалії не лише в класичних вхідних листах, але й у зовнішній/внутрішній кореспонденції.

- Реагування на загрози: система може автоматично застосовувати дії (наприклад, карантин, видалення) та інтегруватися з SOC-операціями.

- Сумісність із хмарними платформами/гетерогенним середовищем: підтримка Microsoft 365, Google Workspace, а також оточень із різноманітними протоколами.

Переваги:

- Спрямованість на складні атаки, які уникають класичних сигнатур: наприклад, зміна стилю листування, зламані сесії, а не лише «лише вірус у вкладенні».

- Завдяки AI-підходу зменшує навантаження на SOC-команду: автоматична класифікація, зменшення кількості хибних спрацьовувань.

- Підтримка внутрішньої та вихідної пошти, що важливо, якщо організація хвилюється про витік даних або внутрішнє шахрайство.

- Висока адаптивність: система навчається на конкретному середовищі компанії, а не тільки на глобальних базах загроз [14].

Недоліки:

- Конкретні ціни часто не публічні — рішення орієнтоване на середній/великий бізнес; впровадження може бути дорогим.

- Необхідність інтеграції в існуючу інфраструктуру: якщо організація має сильно кастомізовану локальну систему, може знадобитися адаптація.

- Як і будь-яке AI-рішення — можливі хибні спрацьовування або необхідність коригування моделей/політик після запуску.

- Залежність від якісного навчання даних: якщо історія листування недостатня або вхідні дані дуже анонімні — результати можуть бути гіршими.

Ціна:

- Виробник зазначає, що ціни налаштовуються індивідуально відповідно до розміру організації та політик.

- У порівняльних оглядах згадуються ціни ~6 USD/користувач/місяць.

Порівняння усіх перелічених систем наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння систем аналогів.

Функція	**Microsoft Defender for Office 365**	**Proofpoint Email Protection**	**Darktrace / EMAIL**	**AI-Mail Security (власна система)**
Можливість налаштування політик	+	+	+	+
Створення гнучких звітів та аналітики	+	+	+	+
Сканування вхідних листів	+	+	+	+
Сканування вихідних/внутрішніх листів	–	+	+	+
Захист від фішингу / BEC-атак	+	+	+	+
Аналіз вкладень (sandbox, ML)	+	+	+	+
Перевірка посилань (URL Rewriting, VirusTotal)	+	+	+	+
Інтеграція з SIEM/SOAR	+ (Sentinel)	+	+	+(Grafana + Loki + Prometheus)

Продовження таблиці 2.1.

Функція	**Microsoft Defender for Office 365**	**Proofpoint Email Protection**	**Darktrace / EMAIL**	**AI-Mail Security (власна система)**
Інтеграція з Active Directory / LDAP	+	+	+	+
Аудиторія	Середні / великі організації	Великі корпорації	Середні та великі	Будь-яка (гнучке розгортання)
API / розширення	Обмежене	Є	Є REST API	Повний REST API для Postfix/Dovecot
Безкоштовна версія	–	–	–	+
Випробувальний період	+ (30 днів)	+ (14 днів)	+ (запит через sales)	+ (повна версія)
Тип розгортання	Хмарне (SaaS)	Хмарне / гібридне / on-prem	Хмарне SaaS	Повністю on-prem або гібрид
Орієнтовна вартість	$2 – $5 / користувач / міс.	$6 – $9 / користувач / міс.	$5 – $6 / користувач/ міс.	Безкоштовно (відкрите ПЗ + власна інфраструктура)

Продовження таблиці 2.1.

Функція	**Microsoft Defender for Office 365**	**Proofpoint Email Protection**	**Darktrace / EMAIL**	**AI-Mail Security (власна система)**
Використання машинного навчання	(Safe Attachments, Safe Links)	+	(Self-Learning AI)	(DistilBERT класифікатор + VT)
Інтерфейс керування	Централізований Microsoft Admin Portal	Веб-консоль SOC	Веб-консоль з AI-підказками	Grafana Dashboard з адаптивними картками
Підтримка TLS/SSL шифрування	+	+	+	(End-to-End TLS 993/587/636)
Сумісність з Postfix/Dovecot	–	+	–	+
Можливість самонавчання моделі	–	–	+	+
Рівень кастомізації	Низький	Середній	Високий	Максимальний
Залежність від хмари	Повна	Часткова	Повна	Відсутня
Рівень витрат на підтримку	Низький	Середній	Високий	Низький

2.2. Обґрунтування архітектури системи фільтрації пошти

Вибір архітектури системи фільтрації електронної пошти обумовлений необхідністю досягнення балансу між високою точністю виявлення фішингових повідомлень**,** прозорістю обробки та контролем даних у межах локальної інфраструктури**.**

Запропонована архітектура поєднує відкриті технології та власні модулі штучного інтелекту, що дозволяє створити автономну систему корпоративного рівня без прив'язки до комерційних хмарних сервісів.

2.2.1. Обґрунтування вибору основних компонентів

1. Postfix як поштовий транспортний агент (MTA)

Postfix обрано через його високу надійність, підтримку сучасних протоколів безпеки (TLS, SASL, SPF, DKIM), а також гнучку систему інтеграцій через Milter API. На відміну від Exim або Sendmail, Postfix має модульну архітектуру, що дозволяє безпосередньо вбудовувати власні фільтри штучного інтелекту без суттєвих змін у базовому коді сервера.

Крім того, Postfix підтримує роботу з доменним контролером через LDAP і є сумісним із Dovecot, що робить його ідеальним рішенням для централізованого обміну поштою в корпоративному середовищі.

2. AI Milter — модуль попередньої класифікації поштового трафіку

Використання власного Milter забезпечує глибоку перевірку вхідних повідомлень до моменту їхньої доставки. Це дозволяє блокувати потенційно шкідливі листи ще на етапі SMTP, зменшуючи ризики зараження робочих станцій.

На відміну від класичних антиспам-плагінів (SpamAssassin, rspamd), AI Milter не використовує сигнатурні або евристичні методи, а застосовує нейронну модель DistilBERT, здатну аналізувати семантичний контекст тексту, що дає змогу виявляти фішинг навіть у випадках, коли повідомлення не містить явних індикаторів шкідливості.

3. AI API — сервер нейронної класифікації та репутаційного аналізу

Вибір архітектури у вигляді окремого FastAPI-сервера пояснюється необхідністю розділення логіки обробки пошти та машинного навчання.

Це дозволяє:

- масштабувати систему незалежно від основного поштового сервера;
- виконувати класифікацію на окремому GPU-хості;
- додавати зовнішні перевірки (VirusTotal, DeepL) без впливу на поштовий трафік.

Така структура спрощує оновлення моделі — достатньо замінити ваги в API, не втручаючись у Milter або Postfix.

4. Dovecot як поштовий IMAP/LMTP-сервер

Dovecot забезпечує безпечне зберігання листів у форматі Maildir і підтримує LDAP/LDAPS-автентифікацію. Обґрунтованість вибору полягає у повній сумісності з Active Directory, що дозволяє автоматично створювати поштові скриньки для доменних користувачів і керувати доступом через групові політики.

Крім того, Dovecot підтримує шифрування на рівні транспортного протоколу (IMAPS 993/TLS), що відповідає вимогам стандартів безпеки ISO/IEC 27001.

5. OPNsense Firewall як шлюз безпеки та IPS/IDS-рівень

OPNsense використовується як граничний шлюз між зовнішньою мережею та поштовим сервером. Його вибір обґрунтовано відкритою архітектурою, підтримкою систем Suricata (IPS/IDS) та GeoIP-блокування, а також можливістю гнучкого NAT та WireGuard-тунелювання.

Таким чином, зовнішній трафік (SMTP, IMAP, Submission) проходить повну перевірку на наявність аномалій, шкідливих IP-адрес і географічно підозрілих регіонів.

6. Grafana як платформа моніторингу та взаємодії адміністратора

Grafana дозволяє реалізувати концепцію Security Operations Dashboard, де всі етапи обробки пошти — від SMTP-сесій до результатів класифікації — відображаються в єдиному інтерфейсі.

Вибір Grafana обумовлено її сумісністю з Loki (логи), Prometheus (метрики) та Infinity (HTTP API), що дає змогу не лише відображати статистику, але й реалізовувати інтерактивні елементи керування — наприклад, блокування домену або IP через API прямо з дашборду.

2.2.2. Обґрунтування вибору архітектурного підходу

Рішення про використання on-premises архітектури (локальне розгортання) ухвалено з огляду на такі чинники:

- Конфіденційність даних. На відміну від хмарних рішень (Microsoft Defender, Proofpoint, Darktrace), система не передає листи або вкладення на зовнішні сервери, що особливо важливо для організацій з підвищеними вимогами до захисту інформації.

- Гнучкість і масштабованість. Кожен компонент (Milter, API, Dovecot) є незалежним мікросервісом, який можна оновлювати або масштабувати без зупинки системи.

- Інтеграція з Active Directory. Це забезпечує централізоване керування користувачами, паролями та політиками доступу без дублювання облікових записів.

- Підтримка штучного інтелекту. Використання нейромережевої моделі DistilBERT дозволяє враховувати семантику тексту, виявляючи фішингові листи навіть без сигнатур або ключових слів.

- Розширюваність. Архітектура побудована таким чином, що в майбутньому можна підключити нові API — наприклад, Google SafeBrowsing, Cisco Talos Intelligence, власний AI-детектор вкладень або sandbox-аналізатор.

2.2.3. Переваги обраної архітектури

- Високий рівень безпеки завдяки наскрізному TLS/LDAPS шифруванню.
- Повна прозорість обробки даних (усі журнали та вердикти зберігаються локально).
- Відсутність залежності від хмарних платформ і ліцензій.
- Можливість самонавчання моделі на реальних листах організації.
- Інтеграція з системами SOC/SIEM (Grafana, Sentinel, Cribl).
- Відповідність принципам інформаційної безпеки за стандартами ISO/IEC 27001.

2.3. Вибір мережевих технологій і засобів безпеки (WireGuard, OPNsense, IDS/IPS)

Захист каналів зв’язку та сегментування мережевої інфраструктури є одним із ключових елементів безпеки системи фільтрації електронної пошти. Вибір технологій здійснювався з урахуванням таких критеріїв:

- Забезпечення цілісності та конфіденційності поштового трафіку;
- Сумісність із відкритими поштовими серверами (Postfix, Dovecot);
- Підтримка сучасних механізмів автентифікації та шифрування;
- Мінімальні вимоги до апаратних ресурсів.

У результаті аналізу альтернативних рішень обрано такі основні компоненти мережевої безпеки:

- WireGuard — для захищеного тунелювання між поштовими вузлами;
- OPNsense Firewall — для фільтрації, NAT і контролю мережевого трафіку;
- IDS/IPS Suricata — для виявлення і запобігання вторгненням та мережевим атакам.

2.3.1. WireGuard — вибір протоколу VPN-тунелювання

WireGuard обрано як основу для створення захищеного каналу зв’язку між зовнішнім VPS-сервером і внутрішньою корпоративною інфраструктурою.

Основні причини вибору:

- Використання сучасних криптографічних алгоритмів ChaCha20, Poly1305, Curve25519, що забезпечують високу продуктивність навіть на малопотужних пристроях.

- Простота конфігурації: усього декілька параметрів у файлі wg0.conf, на відміну від складніших VPN-протоколів, таких як OpenVPN або IPSec.

- Підтримка шифрування end-to-end, що унеможливлює підміну трафіку між поштовим шлюзом і основним сервером.

- Повна сумісність із Linux і FreeBSD-середовищами, які є базою для Postfix і OPNsense.

- WireGuard дозволяє розділити поштову систему на дві зони:

- зовнішня зона (VPS) приймає вхідну пошту з інтернету через публічний IP-адрес;

- внутрішня зона (LAN) обробляє повідомлення у приватній мережі через тунель WireGuard (порт UDP/51820).

Це рішення забезпечує додатковий рівень захисту, оскільки внутрішні сервери не мають прямого доступу до інтернету, а трафік шифрується на всіх етапах.

2.3.2. OPNsense Firewall — шлюз безпеки корпоративної поштової інфраструктури

Як центральний елемент мережевого захисту обрано OPNsense, побудований на базі FreeBSD.

Його вибір обґрунтовано такими факторами:

- Відкритий вихідний код та повна відсутність ліцензійних обмежень — на відміну від комерційних рішень (Palo Alto, Fortinet, Check Point), які

потребують дорогих підписок.

- Сумісність із WireGuard, NAT і VLAN, що дозволяє будувати гібридну інфраструктуру із сегментацією трафіку.

- Модульна архітектура: кожен компонент (IDS, IPS, Proxy, GeoIP, Captive Portal) можна вмикати або вимикати залежно від потреб.

- Зручний веб-інтерфейс і CLI-доступ, що спрощує моніторинг і діагностику трафіку.

- Підтримка резервування (CARP, HA) — дозволяє масштабувати систему для великих організацій.

OPNsense виконує одразу кілька критичних функцій:

1. Фільтрація трафіку (вхідного, вихідного, тунельного);
2. NAT-маскарадинг для поштових серверів у приватній мережі;
3. Розмежування зон безпеки (LAN, DMZ, VPN, WAN);
4. Контроль геолокацій (GeoIP) — блокування підозрілих країн або регіонів, звідки надходить спам.

2.3.3. IDS/IPS Suricata — виявлення та запобігання вторгненням

Для аналізу мережевого трафіку та виявлення потенційних атак використовується система Suricata, вбудована в OPNsense. Її використання виправдане наступними факторами:

- Підтримка сигнатур Snort 3 / Emerging Threats, що дозволяє оперативно виявляти відомі фішингові та мережеві атаки.

- Можливість роботи у режимі IPS, тобто блокування підозрілого трафіку в реальному часі.

- Підтримка TLS fingerprinting, що дозволяє відстежувати зашифрований трафік без його розшифрування.

- Інтеграція з Grafana через Prometheus Exporter — для відображення статистики інцидентів і навантаження.

- Підтримка GeoIP-баз даних MaxMind — для географічної фільтрації запитів.

- Suricata забезпечує додатковий рівень безпеки поверх стандартних поштових перевірок, дозволяючи виявляти:

- спроби сканування портів (Nmap, Masscan);

- аномальні SMTP/IMAP-сесії;

- підозрілі запити до відомих ботнет-доменів;

- з’єднання з IP-адресами, що входять до чорних списків Threat Intelligence.

2.3.4. Обґрунтування вибору саме цієї комбінації технологій

1. WireGuard забезпечує захищений канал між поштовими сегментами без надмірних ресурсних витрат (на відміну від OpenVPN або IPSec).

2. OPNsense виконує функцію як шлюзу, так і платформи моніторингу, дозволяючи інтегрувати IDS/IPS і GeoIP у єдину панель без потреби у зовнішніх рішеннях.

3. Suricata (IDS/IPS) підсилює рівень безпеки, контролюючи весь SMTP/IMAP-трафік на мережевому рівні.

4. Усі три рішення мають відкритий вихідний код, не потребують ліцензійних витрат і відповідають вимогам стандартів ISO/IEC 27001 щодо захищеності каналів зв’язку та контролю доступу.

5. На відміну від комерційних аналогів (FortiGate, Sophos, Cisco ASA), обраний стек дозволяє повністю контролювати конфігурації, журнали й оновлення без обмежень постачальника.

2.4. Вибір моделі нейронної мережі для навчання

На етапі підготовки інтелектуальної системи класифікації електронних листів було проведено порівняльне дослідження кількох алгоритмів машинного навчання. Метою було визначення найбільш ефективної моделі для виявлення фішингових повідомлень, виходячи з точності, швидкодії та здатності до узагальнення даних.

Усі моделі навчались на єдиному корпусі очищених електронних листів,

що містив як безпечні (legitimate), так і фішингові (phishing) приклади, із застосуванням однакової попередньої обробки тексту.

1. Naive Bayes

Модель Multinomial Naive Bayes базується на теоремі Байєса, що дозволяє обчислити ймовірність приналежності електронного листа до певного класу за формулою (2.1):

$$P(C_k \mid X) = \frac{P(X \mid C_k)P(C_k)}{P(X)} \quad (2.1)$$

де X— множина слів, а C_k— клас (*phishing* або *legitimate*).

Підхід передбачає статистичну незалежність ознак, що робить модель обчислювально простою і швидкою.

Результати показали точність ≈96.5%, однак Recall був нижчим, що вказує на більшу кількість пропущених фішингових листів, матриця помилок зображена на рисунку 2.1.

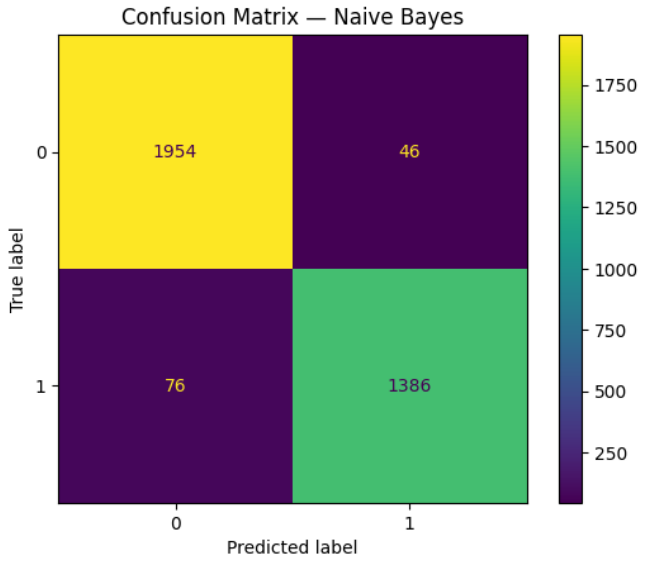

Рисунок 2.1 – Матриця помилок для моделі Naive Bayes

2. Logistic Regression

Модель Logistic Regression оцінює ймовірність належності листа до фішингового класу за допомогою сигмоїдної функції (2.2):

$$P(y=1 \mid x) = \frac{1}{1+e^{-(w^T x+b)}} \quad (2.2)$$

де w— ваги;

b— зсув.

Вона була навчена методом градієнтного спуску, що мінімізує log-loss. Logistic Regression досягла Accuracy ≈ 97.2%, F1 ≈ 96.7%.

Модель характеризується високою узагальнюваністю і стійкістю до дисбалансу класів, матрицю помилок для цієї моделі зображено на рисунку 2.2.

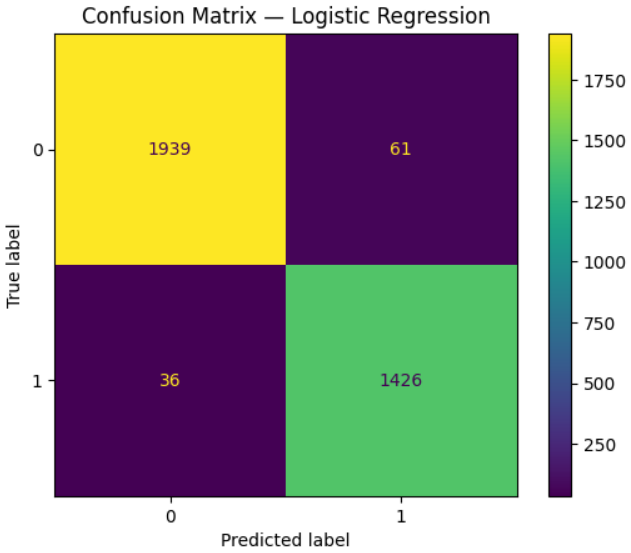

Рисунок 2.2. – Матриця помилок для моделі Logistic Regression

3. Support Vector Machine (SVM, linear)

Модель SVM шукає оптимальну гіперплощину, що розділяє два класи з максимальною відстанню (margin).

У дослідженні використано лінійний варіант LinearSVC, який продемонстрував найкращі показники серед класичних моделей:

Accuracy = 97.6%, Recall = 98.3%, F1 = 97.2%.

Матрицю помилок для цієї моделі зображено на рисунку 2.3.

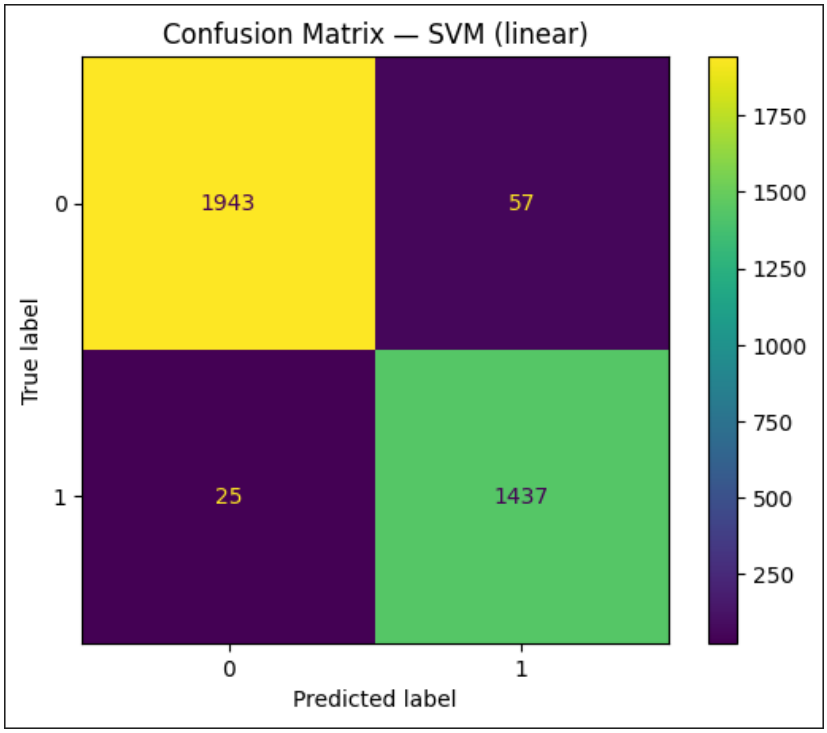

Рисунок 2.3. – Матриця помилок для моделі SVM (linear)

4. DistilBERT

Модель DistilBERT є спрощеною версією архітектури BERT, побудованою на механізмі Self-Attention, що дозволяє враховувати контекст між словами у реченні.

Навчання проводилось із такими параметрами:

- batch size = 16,
- learning rate = 2×10^{-5},
- optimizer = AdamW,
- epochs = 1–6,
- функція втрат = CrossEntropyLoss.

DistilBERT досягла Accuracy ≈ 93.2%, F1 ≈ 93.1%, AUC ≈ 0.96, демонструючи здатність до семантичного аналізу та розуміння контексту. Модель має потенціал для подальшого *fine-tuning* на доменних даних, матрицю помилок для цієї моделі зображено на рисунку 2.4.

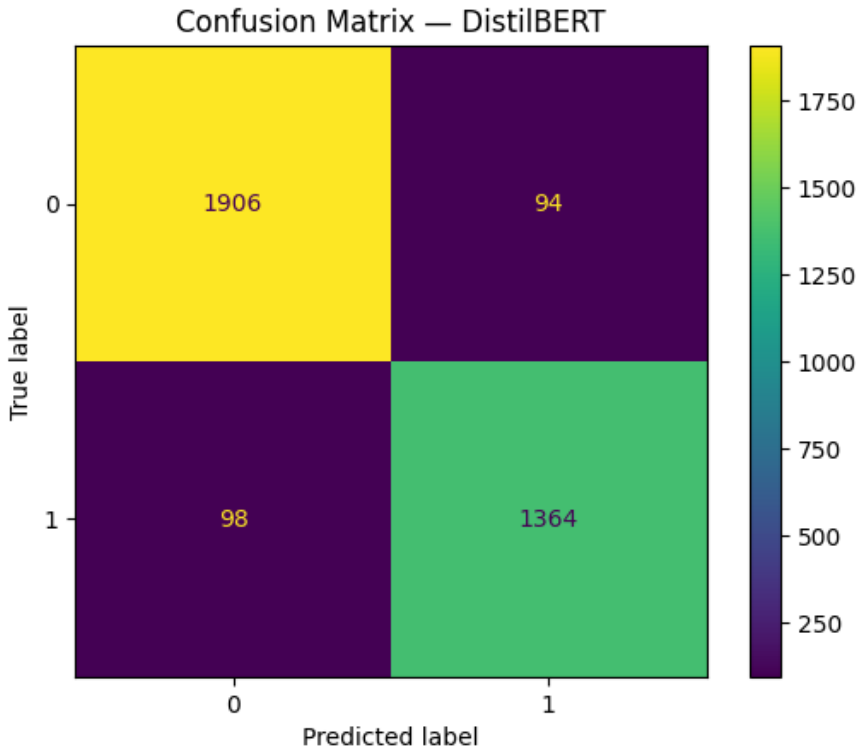

Рисунок 2.4. – Матриця помилок для моделі DistilBERT

5. Порівняльний аналіз моделей

Для оцінки якості класифікації побудовано ROC-криві, що відображають здатність моделей відділяти класи при зміні порогу ймовірності, їх зображено на рисунку 2.5. А також створено таблицю 2.2, на якій описано порівняння ключових параметрів моделей.

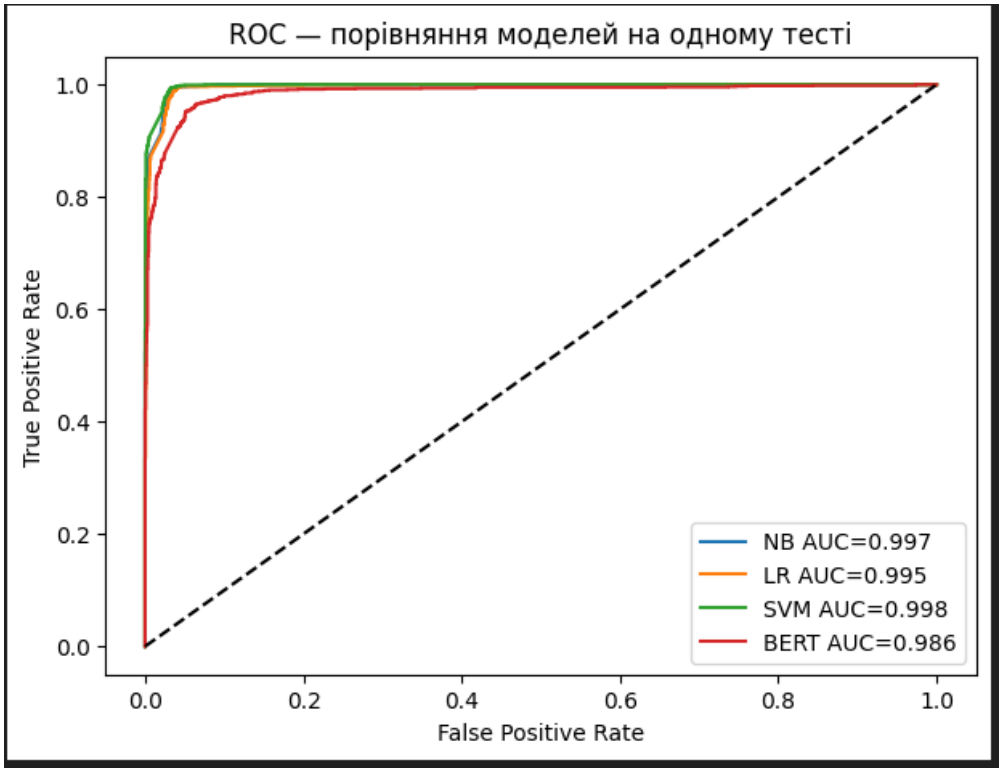

Рисунок 2.5. ROC-криві для порівняння класичних моделей Naive Bayes, Logistic Regression та SVM

Таблиця 2.2 – Порівняння нейронних моделей.

Модель	Accuracy	Precision	Recall	F1	AUC	Характеристика
Naive Bayes	0.965	0.953	0.942	0.947	0.96	Висока швидкість, нижча повнота
Logistic Regression	0.972	0.968	0.965	0.967	0.98	Збалансована якість і простота
SVM (linear)	0.976	0.975	0.983	0.972	0.99	Найкраща модель серед класичних
DistilBERT	0.932	0.934	0.929	0.931	0.96	Враховує семантику, потенціал для донавчання

6. Вибір моделі для використання у класифікаторі

Після проведеного порівняльного аналізу класичних моделей машинного навчання (Naive Bayes, Logistic Regression, SVM) та глибинної архітектури DistilBERT було визначено, що найвищі показники точності на обмеженому наборі даних демонструє модель SVM (linear) — із метриками Accuracy = 97.6% та F1-score = 97.2%.

Однак, отримані результати свідчать лише про ефективність на рівні поверхневих ознак тексту (частот, n-грам), а не про реальну здатність моделі до семантичного розуміння контексту повідомлень.

Класичні підходи базуються на статистичних співвідношеннях між словами та не враховують структуру, значення або наміри, приховані у фішингових листах. Натомість сучасні атаки, що використовують граматично правильні, але змістово маніпулятивні тексти (наприклад, із корпоративними термінами, офіційним тоном чи маскованими посиланнями), потребують контекстно-орієнтованого аналізу.

Саме тому для побудови інтелектуального класифікатора було обрано архітектуру DistilBERT, яка поєднує високу якість семантичного представлення тексту із відносною швидкодією та низькими вимогами до ресурсів.

Модель DistilBERT є скороченою версією BERT, розробленою у дослідженні «Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” arXiv:1910.01108», де доведено, що вона зберігає до 97% точності базового BERT, працюючи у 2 рази швидше і маючи на 40% менше параметрів [39].

У межах експерименту на наборі даних із 17 500 електронних листів (після попереднього очищення та балансування) модель DistilBERT показала Accuracy = 93% і F1-score = 0.998 на фінальній епосі, що підтверджує стабільну збіжність функції втрат та поступове покращення якості класифікації.

Дослідження Zhang et al. (2021). “BERT for Email Classification: A Comparative Study.” IEEE Access, підтверджує, що при збільшенні обсягу корпусу до 100–200 тис. електронних листів точність моделей сімейства

BERT/DistilBERT зростає на 3–5% завдяки покращенню контекстних представлень [40].

Таким чином, DistilBERT було обрано для впровадження у системі фільтрації як базову модель нейронного класифікатора, з огляду на такі переваги як:

- врахування семантичного змісту і контексту повідомлення, а не лише лексичних співпадінь;
- здатність виявляти складні типи фішингових атак, включно з багатомовними та обфускованими текстами;
- підтримка донавчання (fine-tuning) на нових даних без повного перенавчання;
- забезпечення баланс між швидкодією та точністю, що критичним для роботи як пропускний канал у поштовому середовищі.

2.5. Висновки до другого розділу

Проведений аналіз сучасних рішень у сфері захисту електронної пошти засвідчив, що ринок пропонує широкий спектр корпоративних продуктів, зокрема Darktrace/EMAIL, Proofpoint, Microsoft 365/Exchange та інші системи з механізмами машинного аналізу листів. Попри високу ефективність, значна частина таких рішень є комерційною, потребує інтеграції з Microsoft 365/Google Workspace та передбачає суттєві фінансові витрати (у середньому від 4 до 8 USD на одного користувача щомісяця або більше для рішень класу Darktrace). Це робить їх менш придатними для побудови гнучкого лабораторного середовища та для сценаріїв, де необхідна глибока кастомізація, контроль над моделями та розширення функціональності.

З огляду на це було обґрунтовано архітектуру власної системи фільтрації електронної пошти, яка базується на поєднанні Postfix, Dovecot, Milter-фільтра, Active Directory і Cloudflare DNS та доповнюється інтелектуальним аналізом листів за допомогою ML-моделі. Такий підхід забезпечує повний контроль над

процесом обробки повідомлень, можливість гнучкого налаштування політик блокування, інтеграцію з SOC-процесами та централізоване логування.

Для гарантування захищеного транспортування даних було обґрунтовано вибір мережевих технологій та засобів безпеки. WireGuard використовується як високопродуктивний VPN-тунель для передачі поштового трафіку між DMZ-сегментом і внутрішньою інфраструктурою, OPNsense — як міжмережевий екран і точка контролю доступу, а режим IDS/IPS забезпечує виявлення та блокування спроб мережевих атак. Комбінація TLS, LDAPS і SASL-автентифікації гарантує шифрування та цілісність даних як на транспортному, так і на рівні взаємодії з контролером домену.

Окремо було обґрунтовано вибір моделі глибинного навчання, що виконує класифікацію поштових повідомлень. Серед доступних альтернатив обрано DistilBERT — оптимізовану трансформерну architектуру, що забезпечує високу точність визначення фішингу при помірних обчислювальних витратах та можливості донавчання на власних наборів даних. Перевірка URL-адрес через VirusTotal і механізм попереднього перекладу за потреби підсилюють якість аналізу змісту листів.

Таким чином, у другому розділі було сформовано технічно і економічно обґрунтовану концепцію системи поштової безпеки, яка використовує сучасні методи машинного навчання, безпечну архітектуру комунікацій і засоби мережевого захисту. Отримані результати створюють методичну основу для практичної реалізації розробленої системи, що розглядається у наступному розділі.

РОЗДІЛ 3. НАЛАШТУВАННЯ ІНФРАСТРУКТУРИ ЕЛЕКТРОННОЇ ПОШТИ

3.1. Загальна концепція системи

У сучасних умовах розвитку інформаційних технологій електронна пошта залишається одним із ключових інструментів корпоративної комунікації. Проте використання сторонніх поштових сервісів не завжди відповідає вимогам безпеки, контролю доступу та відповідності внутрішнім політикам організації. Саме тому створення власної інфраструктури електронної пошти є доцільним рішенням, яке забезпечує автономність, масштабованість та гнучкість у налаштуванні.

Метою розробки цієї системи є створення повноцінного поштового середовища, що поєднує сучасні засоби шифрування, моніторингу, а також використовує інтелектуальні методи обробки вхідних повідомлень для підвищення рівня інформаційної безпеки. Розроблена інфраструктура поєднує такі основні компоненти:

- Postfix — сервер відправлення та прийому електронної пошти (SMTP);
- Dovecot — сервер зберігання пошти з підтримкою протоколів IMAP та LMTP;
- Active Directory (AD) — служба каталогів для централізованої автентифікації користувачів;
- Cloudflare DNS — зовнішній DNS-провайдер для керування доменом і записами MX, SPF, DKIM, DMARC;
- Loki та Grafana — система збору логів і моніторингу подій у реальному часі;
- Keycloack — система для налаштування двофакторної аутентифікації та зміни пароля користувачів;
- Milter сервіс — система для парсингу електронного листа;

- AI API-сервіс — система для розпізнавання ознак фішингу в електронному листі;
- OPN-Sense — міжмережевий екран для фільтрації вхідного трафіку на рівні L3/L7;
- Python API-сервіс — компонент для керування користувачами, блокування доменів і інтеграції з Grafana;
- ML-модуль — підсистема для інтелектуального аналізу вмісту листів і перевірки посилань через VirusTotal.

Інфраструктура реалізована у віртуальному середовищі Oracle VirtualBox, що забезпечує ізоляцію, контроль над ресурсами та гнучке тестування різних конфігурацій без впливу на основну систему. Такий підхід дозволив моделювати реальне корпоративне середовище, у якому розміщено кілька віртуальних машин: сервер OPN Sense (міжмережевий екран), Ubuntu 24.04 LTSC (поштовий), сервер Windows Server 2025 (контролер домену), а також клієнтські системи для тестування (Outlook, Thunderbird тощо).

Загальна архітектура системи передбачає, що зовнішній трафік надходить через DNS-записи домену test-app.org, зареєстрованого у Cloudflare, після чого поштові повідомлення обробляються на сервері з Postfix, зберігаються в Dovecot і стають доступними користувачам через IMAP. Авторизація здійснюється через Active Directory (домен corp.local) за допомогою захищеного протоколу LDAPS, а моніторинг усіх подій централізовано відображається у Grafana.

Таким чином, система реалізує модель end-to-end електронної пошти з повним контролем доставки, зберігання та обробки повідомлень. Важливою особливістю є можливість подальшого розширення системи за рахунок інтеграції машинного навчання для автоматичного виявлення фішингових та шкідливих повідомлень.

3.2. Реєстрація домену та налаштування DNS

Першим етапом створення системи стало придбання та налаштування зовнішнього домену test-app.org, який використовується для маршрутизації

поштових повідомлень. Для цього було обрано платформу Cloudflare, що забезпечує зручне управління DNS-записами, безкоштовні SSL-сертифікати, а також базові засоби безпеки.

Після реєстрації домену в панелі Cloudflare було створено створено записи що наведено в таблиці 3.1.

Таблиця 3.1 – DNS записи на Cloudflare

Тип запису	Ім’я	Значення / Призначення	Коментар
A	mail.test-app.org	Зовнішня IP-адреса сервера Ubuntu (Postfix/Dovecot)	Основна точка доступу
A	sso.test-app.org	Зовнішня адреса реалму keycloack	Точка доступа для зміни пароля та налаштування доступу користувачів
MX	test-app.org	mail.test-app.org (пріоритет 10)	Доставка пошти
TXT	test-app.org	v=spf1 mx -all	SPF-захист
TXT	default._domainkey	DKIM-публічний ключ	Підпис вихідних листів
TXT	_dmarc	v=DMARC1; p=quarantine; rua=mailto:admin@test-app.org	Політика DMARC

Таке налаштування забезпечує коректну маршрутизацію пошти: коли зовнішній сервер надсилає повідомлення на адресу *@test-app.org*, він звертається до DNS-запису MX і отримує адресу mail.test-app.org, яка вказує на IP поштового

сервера. Далі відбувається SMTP-сеанс між відправником і Postfix.

Важливо, що для доменів поштових піддоменів у Cloudflare необхідно вимкнути проксування (оранжеву хмаринку), оскільки Cloudflare не підтримує проксування SMTP-трафіку. Усі з'єднання виконуються напряму між поштовими серверами.

Крім того, було налаштовано політики SPF, DKIM та DMARC, які підвищують довіру до домену та зменшують ризик потрапляння листів у спам. SPF визначає, які сервери мають право надсилати пошту від імені домену, DKIM забезпечує криптографічну автентичність повідомлень, а DMARC узгоджує результати перевірок SPF і DKIM, надаючи звітність про потенційні підробки, усі створені записи зображено на рисунку 3.1.

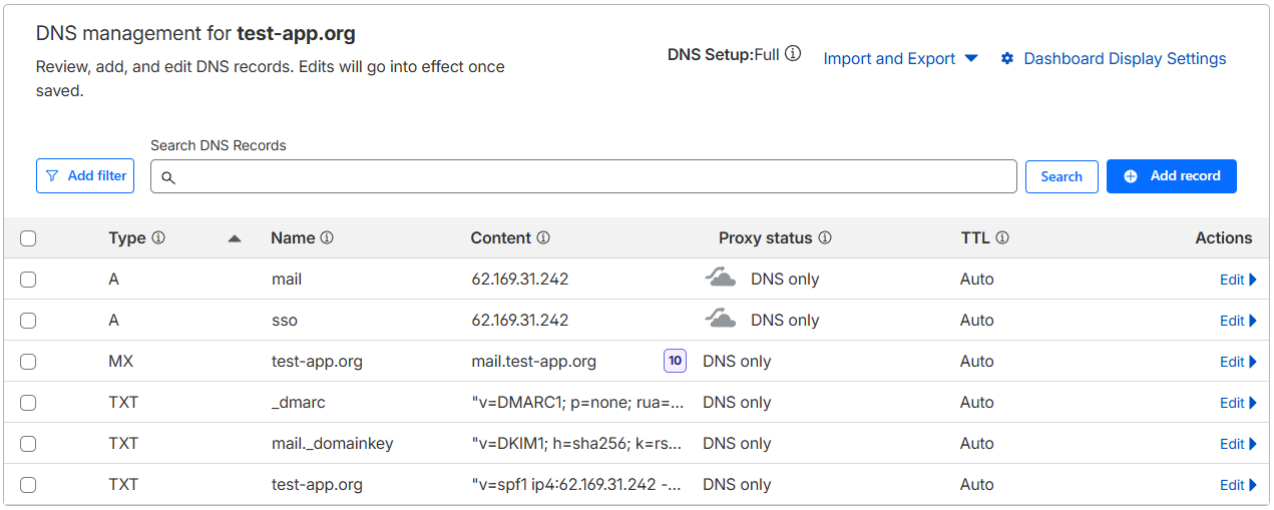

Рисунок 3.1 – DNS записи створені на Cloudflare

Таким чином, доменне середовище створює основу для подальшої взаємодії поштових серверів і забезпечує безпечну маршрутизацію електронної пошти в межах корпоративної інфраструктури та відповідає сучасним вимогам доменної автентифікації, визначеним у стандартах SPF, DKIM та DMARC [31-33].

3.3. Створення мережевої інфраструктури для вхідного та вихідного трафіку

3.3.1. Загальна архітектура мережі

Після реєстрації доменного імені та налаштування DNS-записів було реалізовано побудову повноцінної мережевої інфраструктури, що поєднує локальний серверний сегмент і віртуальний сервер (VPS) у зовнішньому середовищі.

Метою цього етапу є створення ізольованого, безпечного та контрольованого каналу зв'язку між усіма компонентами системи електронної пошти: поштовим сервером, Active Directory, системами моніторингу та шлюзами безпеки.

Архітектура побудована за принципом багаторівневої зонованої моделі (Internet, DMZ, LAN), що дозволяє забезпечити баланс між відкритістю сервісів для зовнішнього доступу та захистом внутрішніх ресурсів корпоративної мережі, схему мережевої архітектури зображено на рисунку 3.2.

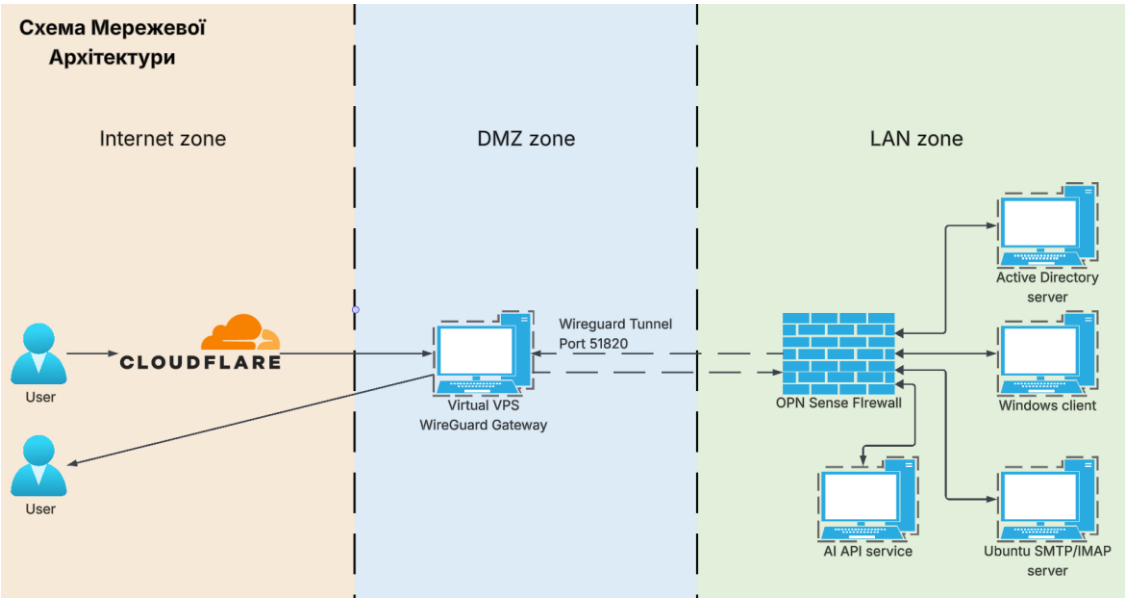

Рисунок 3.2 – Загальна схема архітектури мережі з поділом на зони Internet, DMZ і LAN

1. Internet zone

Зовнішня зона мережі, у якій розміщується DNS-провайдер Cloudflare. Усі вхідні запити до домену test-app.org, зокрема MX-запити поштових серверів, проходять через DNS Cloudflare, який виконує роль першої лінії захисту. Cloudflare не проксирує SMTP-трафік (оранжева хмаринка вимкнена), що дозволяє напряму маршрутизувати поштові з'єднання до VPS-шлюзу.

Основні функції Internet-зони:

- обробка DNS-запитів для поштових записів MX, SPF, DKIM, DMARC;
- надання користувачам зовнішнього доступу до поштової системи;
- первинна фільтрація трафіку та приховування реальної IP-адреси внутрішньої інфраструктури.

2. DMZ zone

У демілітаризованій зоні (DMZ) розміщено віртуальний VPS-шлюз, який виконує функції:

- WireGuard Gateway — термінальна точка VPN-тунелю між зовнішнім середовищем та локальною мережею;
- точка безпечного транзиту поштового трафіку (SMTP/IMAP) до корпоративного середовища;
- первинний рівень NAT та ізоляції, що відділяє зовнішній Інтернет від LAN-зони.

Шифрований канал між VPS і локальною інфраструктурою створено за допомогою WireGuard VPN, який забезпечує мінімальну затримку та високий рівень захисту даних завдяки використанню сучасної криптографії (Curve25519, ChaCha20, Poly1305).

Трафік, що надходить від користувачів через Cloudflare, спочатку потрапляє на VPS-шлюз у DMZ, після чого передається через тунель WireGuard до міжмережевого екрана OPNsense.

3. LAN zone

Основна локальна зона, у якій розміщено всі внутрішні сервери системи. Її основу становить OPNsense Firewall, який виконує роль центрального вузла

безпеки та маршрутизації.

Функціональні ролі OPNsense:

- Фільтрація трафіку між зонами Internet, DMZ та LAN;
- Застосування політик NAT і VPN;
- Робота в режимі IDS/IPS із правилами запобігання вторгненням;
- Блокування за географічною ознакою (GeoIP);
- Журналювання і моніторинг подій безпеки.

Безпосередньо за міжмережевим екраном розташовані основні вузли системи:

- Поштовий сервер (Ubuntu Server 24.04 LTS) — відповідає за прийом і відправлення листів через Postfix, їх зберігання через Dovecot і взаємодію з Active Directory через LDAPS;
- Active Directory Server (Windows Server 2025) — виконує роль контролера домену corp.local, забезпечує централізовану автентифікацію користувачів і служить джерелом сертифікатів (AD CS);
- Windows Client — тестовий клієнт для перевірки роботи поштової інфраструктури (Thunderbird, Outlook);
- Virtual VPS WireGuard Gateway (локальний вузол) — додатковий вузол VPN для маршрутизації внутрішнього трафіку та обслуговування резервних каналів.

4. Мережеві з’єднання та маршрутизація

Комунікація між зонами здійснюється виключно через VPN-тунель WireGuard, що створює логічний канал зв’язку між DMZ та LAN.

Трафік SMTP, IMAP, LDAP та HTTPS передається лише у зашифрованому вигляді.

OPNsense контролює маршрутизацію, визначаючи, які порти та IP-адреси доступні із зовнішнього середовища, тим самим запобігаючи несанкціонованому доступу до внутрішніх серверів.

5. Переваги архітектури

- Сегментація трафіку — поділ інфраструктури на три логічні зони

(Internet, DMZ, LAN) знижує ризик поширення атак усередині мережі.

- Захищеність даних — усі канали між компонентами шифруються за допомогою WireGuard.
- Гнучкість адміністрування — OPNsense забезпечує централізоване керування політиками безпеки, а AD – централізовану автентифікацію.
- Масштабованість — архітектура допускає підключення нових серверів або клієнтів без зміни базової топології.

3.3.2. Налаштування інтерфейсів у OPNsense

У процесі розгортання середовища було створено три основні інтерфейси, опис інтерфейсів наведено в таблиці 3.2.

Таблиця 3.2 – Інтерфейси на міжмережевому екрані

Назва інтерфейсу	Призначення	IP-адреса
LAN	внутрішня мережа користувачів і серверів	192.168.1.1/24
WAN	зовнішній інтерфейс підключення до Інтернету	195.207.x.x
WAN _GW	тунельний інтерфейс WireGuard для з’єднання з VPS	10.0.0.1/24

Кожен інтерфейс отримав власну групу правил доступу та шлюз, що дозволяє ізолювати трафік між сегментами. На рівні System → Gateways визначено основний шлюз через WireGuard, що використовується для виходу у глобальну мережу. Таке розділення інтерфейсів дозволяє реалізувати принцип зонованої безпеки — локальна мережа (LAN) ізольована від зовнішнього середовища, а весь вихідний трафік проходить через контрольований тунель, усі налаштовані інтерфейси показані на рисунку 3.3.

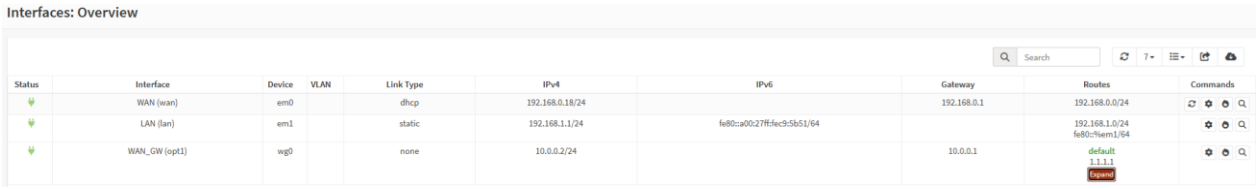

Рисунок 3.3 – Налаштовані інтерфейси у OPNsense

3.3.3. Налаштування VPN-тунелю WireGuard

Для забезпечення захищеної взаємодії між локальною інфраструктурою та VPS-сервером реалізовано VPN-з’єднання на основі WireGuard.

WireGuard — це сучасний протокол тунелювання, що використовує криптографічні алгоритми Curve25519, ChaCha20 та Poly1305, забезпечуючи високу продуктивність і мінімальні затримки.

Основні параметри конфігурації на VPS:

[Interface]

Address = 10.0.0.1/24

ListenPort = 51820

PrivateKey = <приховано>

PostUp = sysctl -w net.ipv4.ip_forward=1

PostUp = iptables -A FORWARD -i wg0 -j ACCEPT

PostUp = iptables -A FORWARD -o wg0 -j ACCEPT

PostDown = iptables -D FORWARD -i wg0 -j ACCEPT

PostDown = iptables -D FORWARD -o wg0 -j ACCEPT

[Peer]

PublicKey = <ключ OPNsense>

AllowedIPs = 10.0.0.2/32, 192.168.1.0/24

На стороні OPNsense відповідно створено peer із параметрами:

- Interface Address: 10.0.0.2/24;
- Peer Endpoint: IP VPS (порт 51820);
- AllowedIPs: 0.0.0.0/0;
- Keepalive: 25 сек.

Після запуску тунелю команда ping 10.0.0.1 підтвердила стабільність каналу та наявність двосторонньої маршрутизації.

3.3.4. Конфігурація NAT і маршрутизації

Для виходу локальних вузлів в Інтернет через захищений тунель WireGuard налаштовано маскарадинг (NAT):

iptables -t nat -A POSTROUTING -s 10.0.0.2 -o eth0 -j SNAT --to-source 62.169.x.x

Це правило забезпечує трансляцію локальної адреси OPNsense на публічну IP-адресу VPS, зберігаючи працездатність усіх вихідних сервісів (HTTP, SMTP, DNS тощо).

На фаєрволі створено аналогічне правило в Firewall → NAT → Outbound, що виконує підміну IP-адреси для мережі 192.168.1.0/24. Також було додано статичні маршрути для коректного зворотного зв’язку з VPS та зовнішніми мережами.

3.3.5. Перевірка та тестування мережевої взаємодії

Після побудови тунелю перевірено працездатність усіх компонентів:

- WireGuard: стабільний пінг між 10.0.0.1 ↔ 10.0.0.2 із середньою затримкою 50–55 мс;
- NAT: зовнішній IP визначається правильно (curl ifconfig.me → 62.169.31.242);
- OPNsense: має повну маршрутизацію між LAN і WAN, ізоляцію внутрішніх адрес, контроль правил фаєрвола.

3.3.6. Налаштування системи виявлення та запобігання вторгненням (IPS)

Для моніторингу мережевого трафіку й запобігання атакам на поштову інфраструктуру було розгорнуто систему Suricata, інтегровану до середовища OPNsense. Цей компонент виконує аналіз пакетів у реальному часі,

використовуючи базу сигнатур атак, виявляє шкідливі з’єднання, та, у режимі IPS (Intrusion Prevention System), блокує їх негайно, конфігурацію IPS\IDS зображено на рисунку 3.4.

Основні етапи налаштування:

1. У меню Services → Intrusion Detection → Administration активовано параметр Enable Intrusion Detection System (IDS).
2. Увімкнено режим IPS Mode, що дозволяє блокування небезпечних пакетів безпосередньо у фаєрволі.
3. Обрано інтерфейс WAN для аналізу трафіку з Інтернету.
4. Активовано Promiscuous Mode, що забезпечує глибокий перегляд усіх пакетів.
5. Збережено параметри та виконано перезапуск служби Suricata.

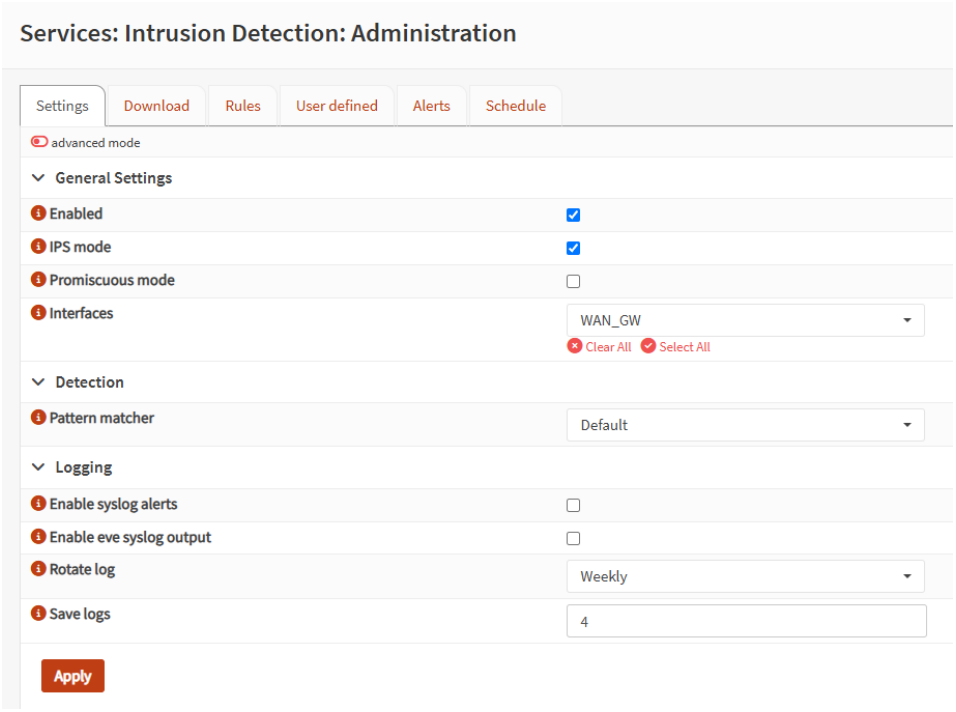

Рисунок 3.4 – Активація системи IDS/IPS у OPNsense

3.3.7. Завантаження та активація правил Suricata

Для підвищення ефективності виявлення атак активовано кілька публічних наборів сигнатур, що регулярно оновлюються:

- ET Open (Emerging Threats) — базовий набір сигнатур атак,

сканувань і шкідливого ПЗ;

- Abuse.ch Feodo Tracker — ідентифікація ботнетів і C2-серверів;
- SSLBL (SSL Blacklist) — виявлення небезпечних SSL-з'єднань;
- MalwareBazaar — фільтрація зразків шкідливих програм;
- ThreatFox / URLhaus — блокування фішингових та шкідливих веб-ресурсів.

Усі правила переведено у стан Enabled. Після активації виконано оновлення баз сигнатур через пункт меню Download & Update Rules. Створено політику для активації усіх сигнатур. У вкладці Alerts перевірено працездатність системи — зафіксовано перші спрацьовування сигнатур при скануванні портів і тестових запитах, приклад спрацювань на фаєроволі можна побачити на рисунку 3.5.

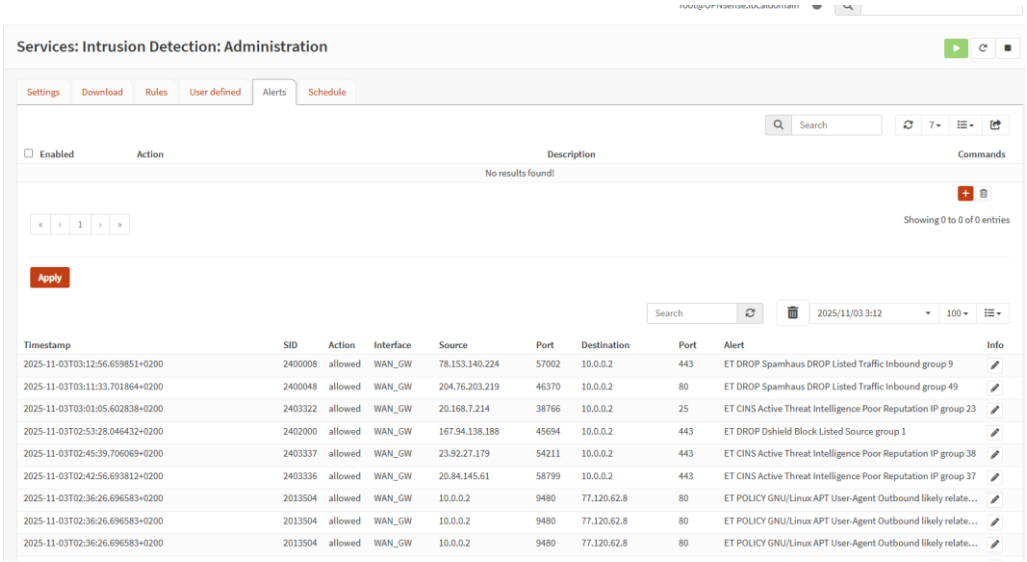

Рисунок 3.5 – Спрацювання по правилах IDS/IPS у середовищі OPNsense

3.3.8. Реалізація GeoIP-блокування

З метою запобігання несанкціонованим підключенням із країн із підвищеним рівнем загроз було реалізовано географічну фільтрацію трафіку (GeoIP). Для цього використано базу MaxMind GeoLite2-Country, яка містить актуальні IP-діапазони країн світу.

Покрокова конфігурація:

1. Створено обліковий запис MaxMind та отримано AccountID і LicenseKey.
2. У меню Firewall → Aliases → GeoIP settings додано URL: https://AccountID:LicenseKey@download.maxmind.com/geoip/databases/GeoLite2-Country-CSV/download?suffix=zip
3. Після натискання Apply база завантажилась, з'явились доступні країни для фільтрації.
4. Створено GeoIP-аліас Block_GEOIP, у якому вибрано(рис. 3.6):
   - Азія: Китай, Іран, Північна Корея, Індія;
   - Європа: Росія, Білорусь;
   - Африка: усі країни континенту.

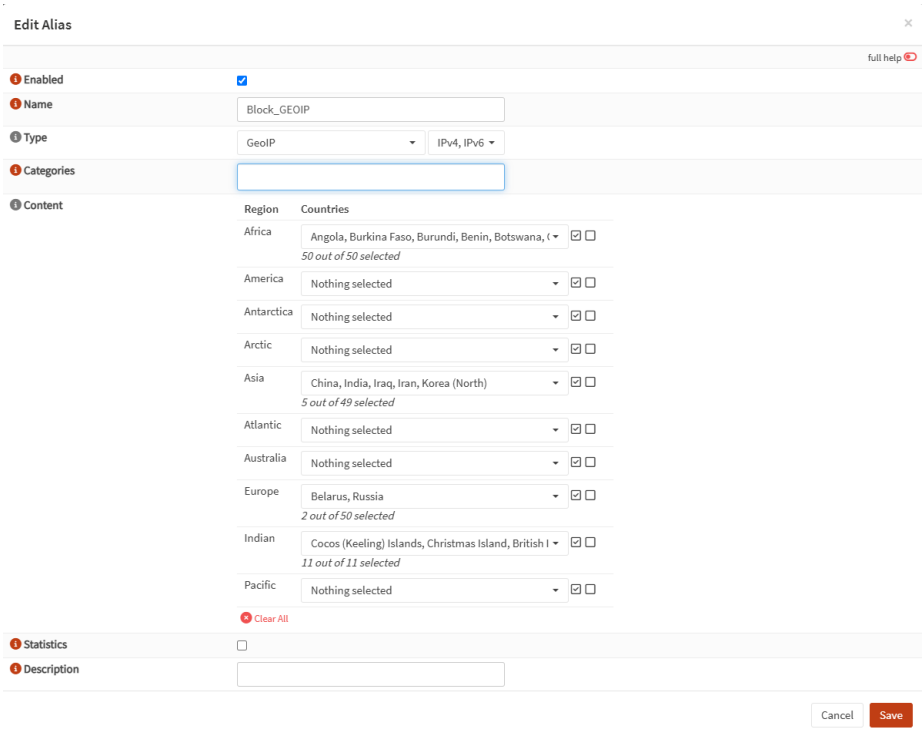

Рисунок 3.6 – Створення GeoIP-аліасу у середовищі OPNsense

Для перевірки коректності роботи нашого правила з блокування перейдемо до вкладки Firewall→ Logs → Live view, і зробимо фільтр за нашим провилом Country reject policy (рис. 3.7).

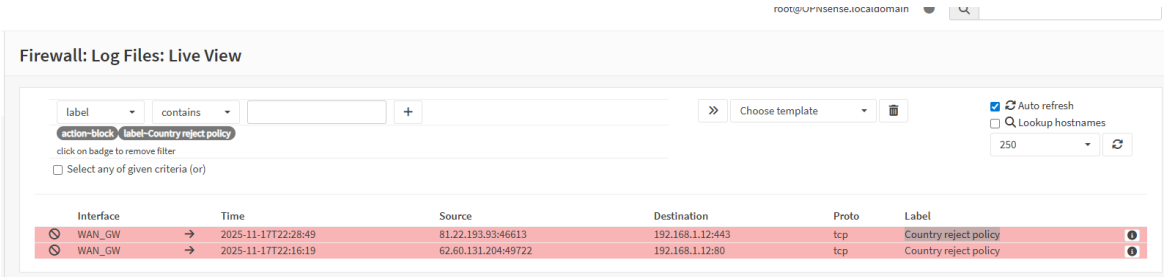

Рисунок 3.7 – Логи з вкладки на firewall за правилом country reject policy

Щоб точно перевірити що це не false positive спрацювання скористаємось сайтом https://2ip.ua/ для того щоб перевірити ip адресу 81.22.193.93 яка пробувала підключитись до нашого серверу по 443 порту (рис. 3.8).

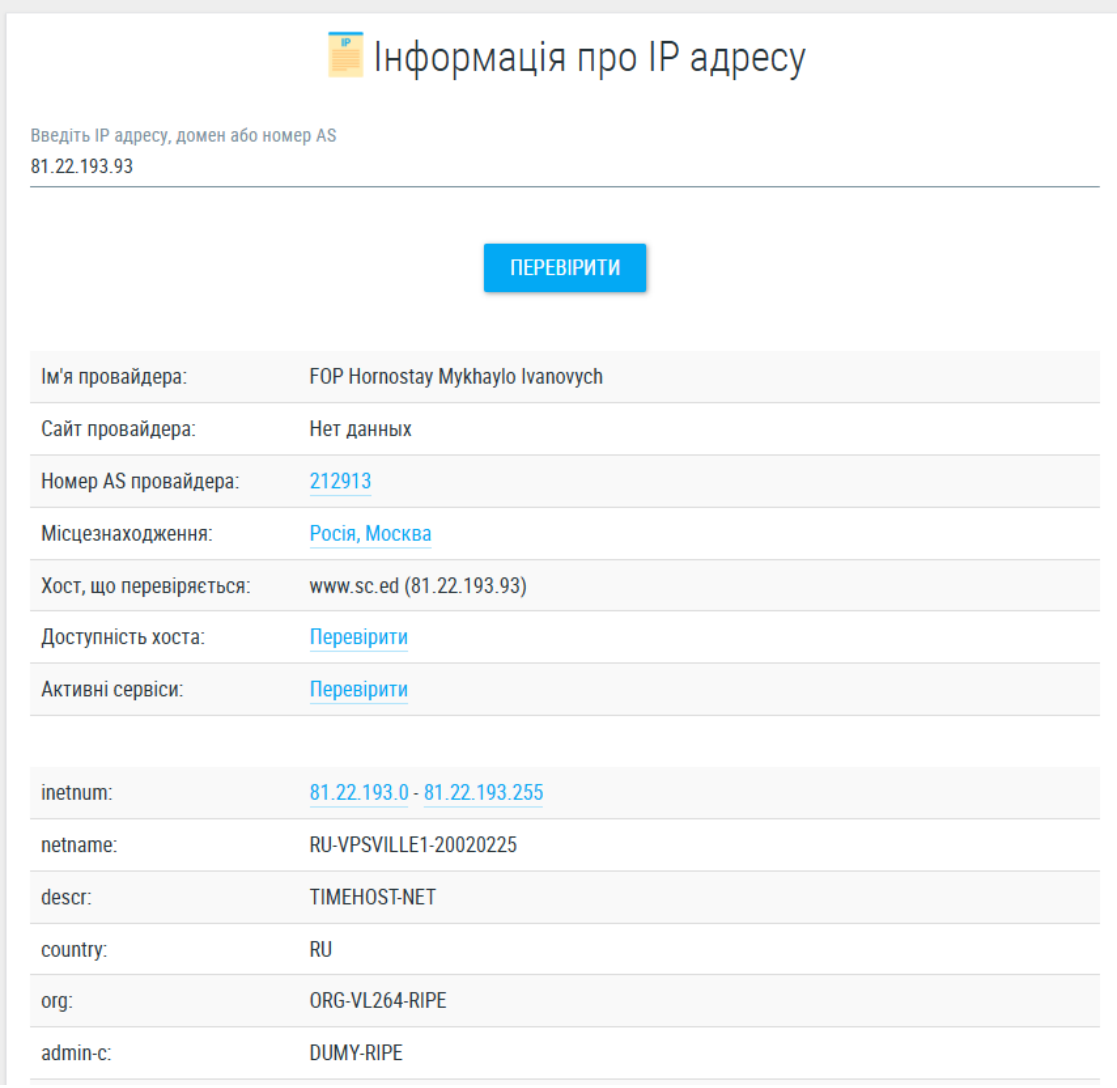

Рисунок 3.8 – Перевірка ip адреси

Як бачимо згідно інформації з сайту підключення відбувалось з Росії та наше правило коректно заблокувало трафік з цієї ір адреси до 443 порту нашого серверу.

3.3.9. Створення правил фаєрвола

Після формування GeoIP-аліасу реалізовано політику блокування доступу з небезпечних країн і неконтрольованих мереж.

У меню Firewall → Rules → WAN додано нове правило:

- Action: Block
- Interface: WAN
- Protocol: Any
- Source: Block_GEOIP
- Destination: Any
- Description: Block incoming connections from restricted regions.

Додатково створено правила для дозволу легітимного трафіку:

- Allow HTTPS (443/tcp)
- Allow Mail Services (25, 587, 993/tcp)

Правила впорядковано в пріоритеті — спочатку блокуючі GeoIP, потім дозволяючі, і в кінці правило, що блокує весь вхідний трафік окрім дозволеного верхніми правилами. Це гарантує, що заборонені IP-діапазони не матимуть змоги пройти до поштового сервера навіть до етапу перевірки IDS. Створені правила зображено на рисунку 3.9.

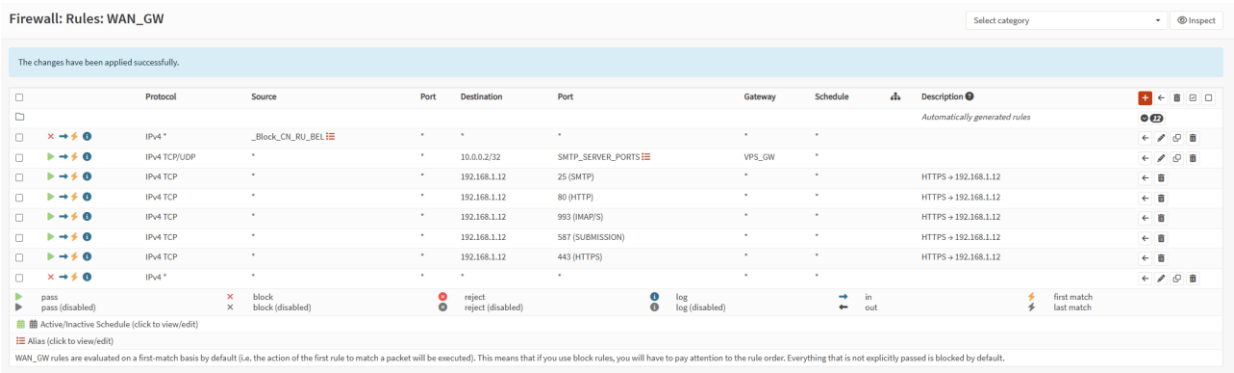

Рисунок 3.9 – Створення правил фаєрвола

3.4. Розгортання серверної інфраструктури

3.4.1. Середовище віртуалізації Oracle VirtualBox

Для розгортання поштової системи використано середовище Oracle VirtualBox, яке забезпечує можливість створення кількох ізольованих віртуальних машин на одній фізичній системі. Це рішення обрано завдяки його стабільності, простоті керування та відкритому вихідному коду.

У процесі реалізації створено такі віртуальні машини:

- MailServer — Ubuntu Server 24.04 LTS (Postfix, Dovecot, Grafana, Loki, Milter);
- ADServer — Windows Server 2025 (Active Directory, Certificate Authority);
- ClientVM — Windows 11 (Thunderbird, Outlook для тестування IMAP/SMTP).

Кожна віртуальна машина мала власну статичну IP-адресу в межах віртуальної підмережі LAN VirtualBox котра контролюється міжмережевим екраном OPN Sense 192.168.1.1/24, що забезпечує ізоляцію від основної системи та можливість контролю мережевого трафіку.

3.4.2. Розгортання Windows Server 2025

На окремій віртуальній машині встановлено Windows Server 2025, виконує роль контролера домену (Domain Controller) з доменом corp.local. Після інсталяції серверу присвоєно статичну IP-адресу та DNS записи, а потім за допомогою майстра Server Manager → Add Roles and Features додано роль Active Directory Domain Services (AD DS).

Після завершення конфігурації домену сервер було підвищено до контролера домену, створено доменну структуру corp.local і виконано первинну реплікацію служби каталогів [16].

Далі було встановлено компонент Active Directory Certificate Services (AD CS). Ця служба дозволяє випускати внутрішні сертифікати, необхідні для

безпечної авторизації за протоколами LDAPS та IMAPS [26].

Після налаштування центру сертифікації було:

- створено шаблон для поштового сервера (MailServer);
- випущено сертифікати для Postfix і Dovecot;
- встановлено сертифікат кореневого центру сертифікації (Root CA) на поштовому сервері Ubuntu.

В Active Directory також було створено organizational unit (OU) Email Users, у якій розміщуються облікові записи користувачів, що мають поштові скриньки. Саме з цього OU у подальшому Python-скрипт періодично зчитує дані для створення чи видалення скриньок на сервері Dovecot.

3.4.3. Розгортання Ubuntu Server 24.04 LTS

На віртуальній машині Ubuntu Server 24.04 LTS виконано повний цикл налаштування поштового середовища. Після інсталяції системи було встановлено базові пакети:

sudo apt update && sudo apt install postfix dovecot-core dovecot-imapd opendkim opendmarc certbot.

Сервер отримав hostname mail.test-app.org, який відповідає запису A у Cloudflare.

На даному етапі було здійснено:

- первинне налаштування Postfix (основні параметри доставки, доменні зони, TLS-сертифікати);
- встановлення Dovecot для збереження поштових скриньок у форматі Maildir;
- інтеграцію з Active Directory через LDAPS для перевірки облікових даних користувачів;
- створення окремого системного користувача vmail для зберігання пошти;
- підготовку каталогів /home/vmail з правами доступу 700.

Усі сервіси налаштовані для автозапуску через systemctl, що забезпечує їх стабільну роботу після перезапуску системи:

```
sudo systemctl enable postfix
sudo systemctl enable dovecot
sudo systemctl start postfix
sudo systemctl start dovecot
```

Таким чином, на даному етапі створено повноцінне серверне середовище, яке об’єднує доменну інфраструктуру Windows і поштовий стек Linux. Це поєднання дозволяє досягти високого рівня сумісності, безпеки та гнучкості в управлінні користувачами.

3.5. Налаштування поштової системи

3.5.1. Налаштування Postfix

Postfix є основним компонентом системи, який забезпечує прийом, обробку та відправлення електронної пошти. Його перевагами є висока продуктивність, гнучкість конфігурації та сумісність із різними системами автентифікації.

Після встановлення пакета postfix система переходить до базового конфігуратора, де було обрано параметри:

- Internet Site — тип інсталяції для безпосереднього прийому листів з Інтернету;
- mail.test-app.org — основне доменне ім’я сервера (hostname);
- test-app.org — основний поштовий домен.

Основний конфігураційний файл /etc/postfix/main.cf було відредаговано для налаштування ключових параметрів:

```
myhostname = mail.test-app.org
mydomain = test-app.org
myorigin = /etc/mailname
```

```
mydestination = localhost
relay_domains = $mydomain
mynetworks = 127.0.0.0/8
home_mailbox = Maildir/
smtpd_banner = $myhostname ESMTP
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
smtpd_tls_cert_file = /etc/ssl/certs/mail.crt
smtpd_tls_key_file = /etc/ssl/private/mail.key
smtpd_tls_security_level = may
```

Параметри smtpd_sasl_type і smtpd_sasl_path визначають, що Postfix використовує Dovecot як бекенд для автентифікації користувачів. Це дозволяє централізувати процес авторизації через LDAP-зв’язок із Active Directory.

Також активовано TLS-шифрування для всіх сеансів SMTP, що гарантує конфіденційність переданих даних. Сертифікати, видані внутрішнім центром сертифікації (CA) на сервері Windows Server, імпортовано на поштовий сервер Ubuntu.

Файл /etc/postfix/master.cf доповнено налаштуваннями для портів 25 (SMTP) та 587 (Submission), що дозволяє користувачам відправляти пошту з автентифікацією:

```
submission inet n       -       y       -       -       smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_recipient_restrictions=permit_sasl_authenticated,reject
```

Завдяки цим налаштуванням Postfix виконує такі функції:

- прийом вхідних листів через порт 25;
- пересилання їх у Dovecot через протокол LMTP;
- автентифікація користувачів під час відправлення через порт 587;

- підпис DKIM перед виходом пошти у зовнішню мережу.

3.5.2. Налаштування Dovecot

Сервер Dovecot забезпечує збереження поштових скриньок користувачів і доступ до них через протокол IMAP. Основні конфігураційні файли розміщуються в каталозі /etc/dovecot/conf.d/.

У файлі 10-mail.conf налаштовано використання формату Maildir для зберігання пошти:

mail_location = maildir:/home/vmail/%d/%n/Maildir

first_valid_uid = 5000

first_valid_gid = 5000

Структура каталогів організована за принципом:

/home/vmail/<домен>/<користувач>/Maildir/, що дозволяє відокремити поштові дані кожного користувача.

Для інтеграції з Active Directory створено конфігураційний файл /etc/dovecot/dovecot-ldap.conf.ext:

hosts = dc1.corp.local:636

base = ou=emailusers,dc=corp,dc=local

ldap_version = 3

scope = subtree

dn = cn=LdapUser,CN=Users,DC=corp,DC=local

dnpass = <пароль>

auth_bind = yes

tls = yes

tls_ca_cert_file = /etc/ssl/certs/corp-root-ca.pem

pass_filter = (|(mail=%u)(userPrincipalName=%u)(sAMAccountName=%u))

pass_attrs = sAMAccountName=user

user_filter = (|(mail=%u)(userPrincipalName=%u)(sAMAccountName=%u))

user_attrs = mail=user,=home=/home/vmail/%u,=uid=5000,=gid=5000

Таким чином, при вході користувача Dovecot здійснює запит до AD по

LDAPS, отримує підтвердження автентичності та відкриває доступ до поштової скриньки.

3.5.3. Автоматизація створення поштових скриньок

Для забезпечення узгодженості між службою каталогів Active Directory та поштовим сервером Dovecot було розроблено Bash-скрипт, який автоматично створює поштові скриньки для користувачів домену.

Скрипт підключається до контролера домену corp.local через протокол LDAPS (порт 636), здійснює пошук активних користувачів в організаційній одиниці emailusers і для кожного користувача, що має заповнений атрибут mail, створює окрему директорію Maildir.

У разі виявлення нового користувача система автоматично формує необхідну структуру папок (Drafts, Junk, Sent, Trash) та налаштовує файл підписок subscriptions, що забезпечує коректну роботу IMAP-клієнтів без додаткових дій адміністратора.

Усі створені каталоги отримують права власності на системного користувача vmail, під яким працює Dovecot, що гарантує безпеку доступу до пошти.

У додатку наведено приклад скрипта .

Скрипт реалізує повний цикл автоматизації:

- отримання списку активних користувачів із Active Directory через LDAPS;
- перевірку наявності атрибута mail та вимкненого стану облікового запису;
- створення папки Maildir та базових тек (чернетки, спам, надіслані, кошик);
- призначення прав доступу користувачу vmail;
- формування файлу підписок для автоматичного відображення папок у поштових клієнтах.

Логіку синхронізації наведено у схемі на рисунку 3.10.

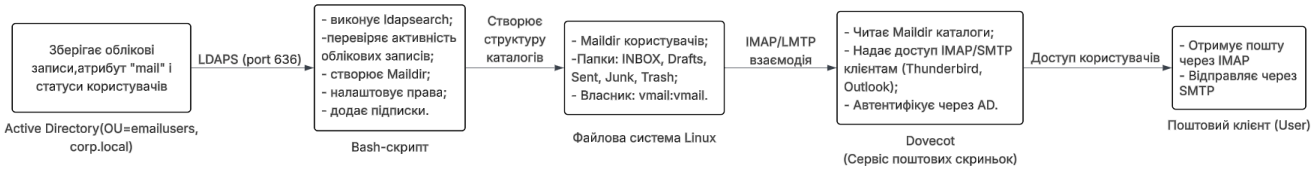

Рисунок 3.10 – Логіка синхронізації користувачів між Active Directory та Dovecot

Завдяки цьому підходу адміністратор системи звільняється від ручного створення поштових скриньок, що зменшує ймовірність помилок і забезпечує оперативну синхронізацію стану служби каталогів із поштовою інфраструктурою.

Крім того, скрипт не потребує окремої бази даних для зберігання списку користувачів — всі дані отримуються безпосередньо з Active Directory, що спрощує архітектуру та зменшує кількість точок відмови в системі.

3.5.4. Захист каналів передачі даних та шифрування трафіку

Однією з обов’язкових вимог до корпоративної поштової інфраструктури є забезпечення конфіденційності та цілісності даних під час передавання облікових даних користувачів і вмісту електронних листів. У межах розробленої системи ця вимога реалізується шляхом повного переходу на захищені сеанси зв’язку з використанням протоколу TLS (Transport Layer Security) для таких сервісів, як IMAP, SMTP-Submission та LDAP-доступ до служби каталогів Active Directory.

Шифрування доступу до поштових скриньок (IMAPS):

Доступ користувачів до їхніх поштових скриньок здійснюється через протокол IMAP на порту 993, причому з’єднання дозволяється виключно у зашифрованому вигляді. Для цього на сервері Dovecot було активовано режим обов’язкового використання TLS. Відповідні параметри задані у файлі конфігурації /etc/dovecot/conf.d/10-ssl.conf:

ssl = required

ssl_cert = </etc/letsencrypt/live/mail.test-app.org/fullchain.pem

ssl_key = </etc/letsencrypt/live/mail.test-app.org/privkey.pem

ssl_min_protocol = TLSv1.2

Пояснення наведених параметрів:

- ssl = required – забороняє незашифровані сесії. Клієнт (Thunderbird, Outlook тощо) не може під'єднатися до сервера без встановлення захищеного каналу.

- ssl_cert – шлях до публічної частини сертифіката, який використовується Dovecot для встановлення TLS-сесії.

- ssl_key – приватний ключ, який відповідає цьому сертифікату.

- ssl_min_protocol = TLSv1.2 – сервер примусово блокує застарілі криптографічні протоколи (SSLv3, TLSv1.0, TLSv1.1) і дозволяє лише сучасні версії TLS. Це захищає від відомих атак на старі версії протоколів шифрування.

Таким чином, жодні облікові дані користувача (логін і пароль доменного облікового запису) не передаються у відкритому вигляді мережею: вони шифруються ще до того, як покидають клієнтський поштовий застосунок.

Випуск та використання сертифікатів.

Для шифрування IMAP-з'єднання використовується доменне ім'я поштового сервера mail.test-app.org. Серверу було видано дійсний TLS-сертифікат, розміщений у каталозі /etc/letsencrypt/live/mail.test-app.org/. У межах проєкту це вирішує одразу дві задачі:

1. Криптографічна ідентифікація сервера.

Коли клієнт (наприклад, Thunderbird) встановлює з'єднання з mail.test-app.org:993, він отримує від сервера ланцюжок сертифікатів (fullchain.pem) і перевіряє:

- що ім'я в сертифікаті збігається з іменем хосту (mail.test-app.org);
- що сертифікат підписаний довіреним центром сертифікації;
- що сертифікат не прострочений.

Якщо ці умови виконані, клієнт вважає сервер достовірним і продовжує встановлення TLS-сесії.

2. Створення зашифрованого каналу (TLS handshake).

Після підтвердження автентичності починається обмін ключами сесії. Подальший трафік (логін/пароль, вміст листів, структури папок) шифрується симетричним ключем, згенерованим для цієї конкретної сесії. Відповідно, навіть якщо трафік буде перехоплено в мережі, його неможливо прочитати.

Файли fullchain.pem і privkey.pem означають відповідно:

- повний ланцюжок сертифікатів (серверний сертифікат + проміжні сертифікати);
- приватний ключ цього сертифіката.

Серверна частина (Dovecot) ніколи не розкриває приватний ключ клієнту. Ключ зберігається локально на сервері та використовується лише для завершення TLS рукостискання.

Шифрування автентифікації користувачів

Окремо слід зазначити, що поштовий сервер не зберігає власної бази паролів. Пароль користувача верифікується через Active Directory. Це відбувається у два кроки:

1. Користувач вводить логін і пароль у поштовому клієнті.
2. Ці дані передаються до Dovecot лише через захищений IMAPS (993/TLS), після чого Dovecot перевіряє ці облікові дані через LDAP-запит до контролера домену.

Комунікація між Dovecot і контролером домену Active Directory (Windows Server 2025) здійснюється не у відкритому LDAP (389/tcp), а через LDAPS, тобто LDAP поверх TLS (порт 636). Це означає, що навіть внутрішній трафік автентифікації (паролі доменних користувачів) передається у шифрованому вигляді і не може бути перехоплений або модифікований при транзиті.

У конфігурації Dovecot для LDAP-аутентифікації додатково вказується використання TLS-сертифіката контролера домену, виданого службою сертифікації Active Directory Certificate Services (AD CS). Це дозволяє Dovecot не просто встановити LDAPS-з’єднання, а впевнитися, що він спілкується саме з

довіреним контролером домену, а не з підставленим вузлом.

Захищене надсилання пошти (SMTP Submission):

Окремо від IMAP, користувачі надсилають пошту через порт 587 (Submission). Postfix на цьому порту налаштований з обов'язковим шифруванням і автентифікацією. Поштовий клієнт не може відправити лист без:

- встановлення TLS (STARTTLS або пряма TLS-сесія),
- успішної аутентифікації (SASL).

Джерелом істини для перевірки пароля знову виступає Active Directory. це важливо з точки зору моделі загроз: навіть якщо користувач підключається з ненадійної мережі (наприклад, публічний Wi-Fi), його SMTP-сесія і пароль домену не йдуть у відкритому вигляді.

Автоматичне визначення конфігурації через Mozilla ISPDB підтвердило коректність параметрів сервера (рис. 3.11):

- Сервер вхідної пошти (IMAP): mail.test-app.org
- Порт: 993, шифрування SSL/TLS
- Сервер вихідної пошти (SMTP): mail.test-app.org
- Порт: 587, шифрування STARTTLS
- Метод автентифікації: LDAP/LDAPS (Active Directory)
- Ім'я користувача: testuser123

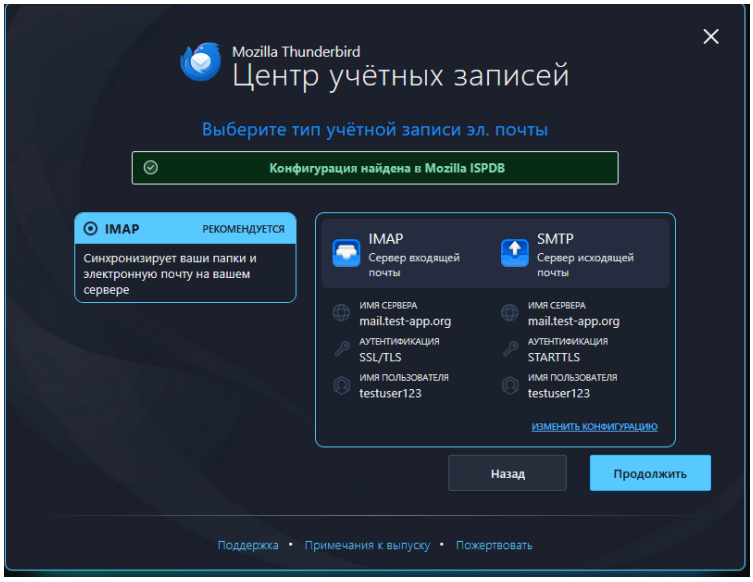

Рисунок 3.11 – Автоматичне виявлення параметрів поштового сервера у Mozilla Thunderbird

Висновок щодо безпеки каналу:

Таким чином, шифрування трафіку реалізоване на всіх критичних ділянках взаємодії:

- доступ користувача до поштової скриньки (IMAP → TLS на порті 993);
- автентифікація користувача під час надсилання пошти (SMTP Submission → TLS на порті 587);
- перевірка облікових даних проти доменної служби (LDAP → LDAPS на порті 636);
- внутрішнє зберігання пошти у форматі Maildir з правами доступу, обмеженими сервісним обліковим записом vmail.

На рисунку 3.12 зображено Захищені канали та основні протоколи їх захисту.

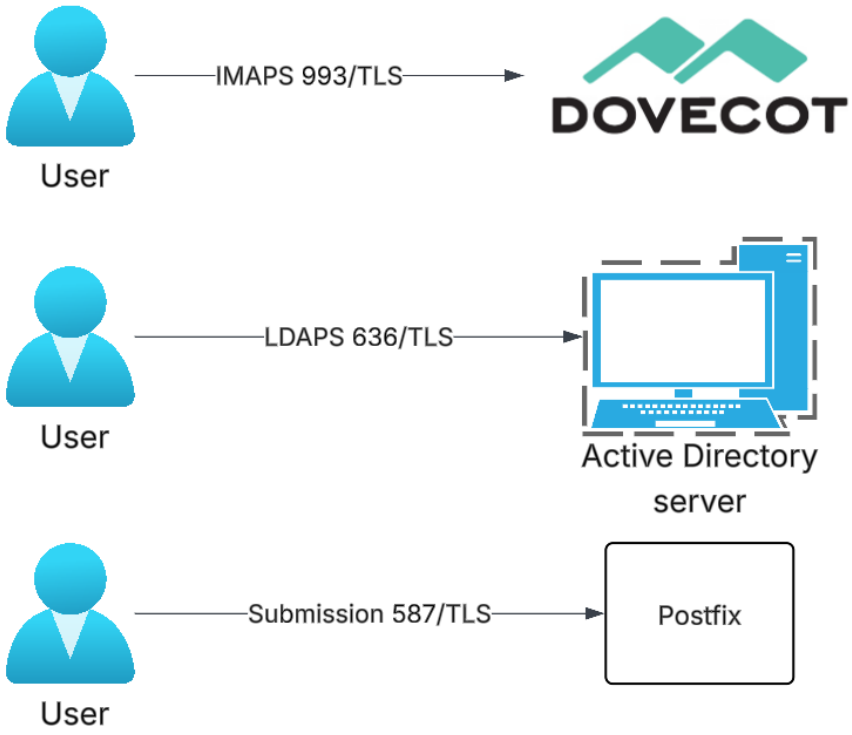

Рисунок 3.12 – Захищені канали доступу до поштової інфраструктури (IMAPS, Submission, LDAPS)

Це дозволяє стверджувати, що система не покладається на довіру до внутрішньої мережі як такої, а будується за принципом "захист від

прослуховування на будь-якому сегменті", включно з локальним сегментом, тунелем WireGuard та каналами між поштовим сервером і контролером домену.

3.6. Інтеграція машинного аналізу у поштовий потік

3.6.1 Модуль AI Milter

Модуль AI Milter є проміжною ланкою між поштовим сервером Postfix та системою машинного аналізу вхідних повідомлень. Його головне завдання — перехопити вхідну пошту на етапі SMTP-діалогу, виконати первинний парсинг листа та передати структуровані дані для оцінки у модуль AI API, що розміщений на віддаленому Windows-хості (IP 192.168.1.10).

Загальна схема роботи

Postfix (порт 25) при отриманні листа викликає зареєстрований Milter-фільтр (mlmilter), який працює як окремий Python-сервіс на сервері Ubuntu 24.04.

Milter отримує усі частини повідомлення (заголовки, тіло, вкладення), формує структуру JSON і через HTTP-запит надсилає її до AI API на Windows-хості для аналізу.

Після отримання відповіді з оцінкою (action, score, verdict) Milter додає у лист відповідні технічні заголовки X-ML-*, а у разі негативного вердикту може відхилити повідомлення або помістити його у карантин.

На рисунку 3.13 зображено схему взаємодії postfix та сервісу AI Milter.

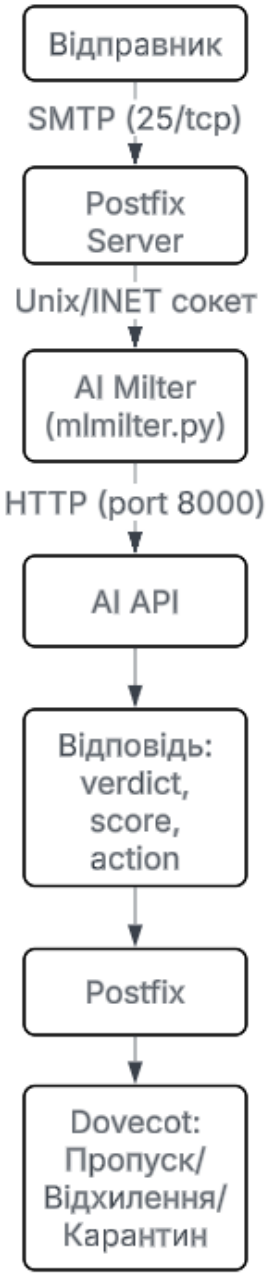

Рисунок 3.13 – Взаємодія Postfix і AI Milter

Основні компоненти та функції

Модуль реалізовано на Python 3.12 з використанням бібліотеки pymilter (1.0.5) та httpx для мережевої взаємодії, код сервісу наведено в додатку Г.

У коді Milter реалізовано такі етапи обробки листа:

1. Отримання конвертних даних (envelope):

Використовуються методи envfrom() та envrcpt() для збереження адрес відправника і отримувача.

2. Збір заголовків:

Метод header() зберігає усі RFC-заголовки та ланцюжки Received для подальшого аналізу SPF/DKIM/DMARC.

3. Збір тіла та вкладень:

Метод body() накопичує байти повідомлення до 64 КБ. Після завершення передачі (callback eom()) відбувається повний парсинг повідомлення через email.parser.BytesParser, витягуються текстові та HTML-частини, а також метадані вкладень (ім’я, розмір, SHA-1/-256, MD5).

4. Виділення URL-посилань:

У класі _extract_urls_heuristic() реалізовано власний алгоритм виявлення посилань за п’ятьма евристиками (HTTP-шаблони, атрибути href, src, доменоподібні рядки, скрипти JavaScript).

5. Збір антиспуфінгових метаданих:

Milter аналізує заголовки Authentication-Results, Received-SPF, визначає IP-адресу відправника, а також результати SPF/DKIM/DMARC.

6. Формування запиту до AI API:

Milter створює структуру JSON, що включає поля:

```
{
  "message_id": "...",
  "from": "user@example.com",
  "to": ["target@test-app.org"],
  "headers": { "subject": "...", "from": "...", "to": "..." },
  "body_text": "...",
  "urls": ["http://example.com", ...],
  "attachments": [{ "filename": "...", "sha256": "...", "size": 10240 }],
  "auth_results": { "spf": "pass", "dkim": "fail", "dmarc": "none" },
  "client_ip": "203.0.113.12"
}
```

Передача результатів до Postfix

Після отримання відповіді від AI API (наприклад

{"action":"reject","score":0.98,"verdict":"phishing"}),

Milter додає у лист наступні службові заголовки:

X-ML-Score: 0.98

X-ML-Verdict: phishing

X-ML-Model: distilbert-v1.3

X-ML-LatencyMs: 124

X-ML-Client-IP: 203.0.113.12

X-ML-URLs-Count: 3

Якщо action = "reject", повідомлення блокується із кодом SMTP 550 "Message rejected by ML policy". У разі quarantine – лист додається у карантинну папку та позначається заголовком X-ML-Quarantine: yes.

Конфігурація Postfix для Milter

Підключення модуля до Postfix виконується через main.cf і master.cf:

smtpd_milters = inet:127.0.0.1:12302

non_smtpd_milters = $smtpd_milters

milter_default_action = accept

milter_protocol = 6

Розроблений модуль AI Milter дозволяє вбудувати механізми машинного навчання в реальний поштовий трафік, забезпечуючи динамічну оцінку кожного повідомлення в реальному часі. Архітектура *«Postfix → AI Milter → AI API»* дозволяє розділити навантаження між поштовим та аналітичним серверами, забезпечуючи масштабованість і високу надійність роботи системи.

3.6.2. Тренування нейронної мережі для виявлення фішингових листів

Метод навчання моделі

Для вирішення задачі класифікації електронних листів за категоріями фішинговий / безпечний застосовано метод transfer learning на основі глибинної трансформерної архітектури DistilBERT.

Суть методу полягає у використанні попередньо натренованої мовної

моделі, що вже володіє знанням контексту та граматичних залежностей англійської мови, з подальшим її до-навчанням (fine-tuning) на спеціалізованому наборі даних електронних листів.

Архітектура моделі:

Модель складається з двох основних компонентів, формула (3.1):

$$f(x)=\text{softmax}\left(W_c h_{[CLS]}+b_c\right), \quad (3.1)$$

де:

x— послідовність токенів тексту листа;

$h_{[CLS]}$— вектор прихованого стану токена [CLS] після проходження через DistilBERT encoder;

W_c, b_c— параметри класифікаційного шару.

DistilBERT складається з 6 енкодерів (на відміну від 12 у BERT), кожен із яких містить:

- механізм самоуваги (Self-Attention);
- двошаровий feed-forward перцептрон;
- залишкове з'єднання (residual connection);
- шар нормалізації (LayerNorm).

Вхідне представлення

Кожен токен t_i подається у вигляді суми трьох векторів, формула (3.2):

$$e_i=w_i+p_i+s_i, \quad (3.2)$$

Де w_i— embedding токена (словникове представлення);

p_i— позиційне вбудовування;

s_i— вектор сегменту (у випадку використання Subject + Body).

Максимальна довжина послідовності — 256 токенів.

Підготовка даних

Було об'єднано й очищено кілька відкритих наборів даних (CEAS08, Enron, Ling, Nazario, Nigerian Fraud, SpamAssassin). Загальний обсяг — приблизно 160 тисяч листів, кожен з яких мав поля:

- subject — тема листа;

- body — тіло повідомлення;
- label — мітка (0 — безпечний, 1 — фішинговий).
- Проведено:
- очищення HTML-тегів і службових символів;
- нормалізацію до нижнього регістру;
- видалення дублікатів;
- балансування класів (до 5 000 записів на клас).

2. Токенізація

Для перетворення тексту в послідовність чисел використано токенізатор WordPiece, вбудований у DistilBERT (словник ≈ 30 000 токенів).

Математичний опис моделі

Додатково додавався спеціальний токен [CLS], який використовується для агрегації інформації по всьому тексту.

Механізм самоуваги (Self-Attention)

Для кожного токена iобчислюється «увага» до інших токенів через запити, ключі та значення:

$$Q=XW_Q, K=XW_K, V=XW_V, \quad (3.3)$$

де: $X=[e_1,e_2,\ldots,e_n]\in R^{n\times d}$— матриця вхідних векторів токенів;

e_i— векторне представлення і-го токена;

$W_Q, W_K, W_V \in R^{d\times d_k}$— матриці параметрів запитів, ключів і значень;

d— розмір векторного простору токенів;

d_k— розмірність підпростору ключів і запитів.

Коефіцієнти уваги (attention weights) визначаються як (3.4):

$$A=\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right), \quad (3.4)$$

де: $QK^\top$— міра подібності між токенами;

$\sqrt{d_k}$— нормувальний коефіцієнт для стабілізації градієнтів;

$\text{softmax}(\cdot)$— нормалізація до ймовірнісного розподілу.

Вихід шару уваги (3.5):

$$Z=AV. \quad (3.5)$$

де: $A \in R^{n \times n}$— матриця ваг уваги;

$V \in R^{n \times d_k}$— матриця значень;

$Z \in R^{n \times d_k}$— результат зваженого агрегування інформації між токенами.

Мультиголова увага (Multi-Head Attention) ділить Z на hпідпросторів і конкатенує результати(3.6):

$$MultiHead(Q,K,V)=Concat(Z_1,Z_2,\ldots,Z_h)W_O, \quad (3.6)$$

де: h— кількість голів уваги (у DistilBERT: h=12);

$W_O \in R^{hd_k \times d}$— матриця з'єднання результатів.

3.2. Feed-Forward шар і нормалізація

Після шару уваги застосовується двошаровий перцептрон (3.7):

$$FFN(x)= \max(0,xW_1+b_1)\, W_2+b_2 \quad (3.7)$$

де: W_1, W_2— вагові матриці;

b_1, b_2— зміщення,

max(0,·)— функція активації ReLU.

Результат проходить через residual connection та Layer Normalization:

$$H^{(l+1)}=LayerNorm(H^{(l)}+FFN(SelfAttn(H^{(l)}))). \quad (3.8)$$

де: $H^{(l)}$— вихід l-го шару,

$H^{(l+1)}$— вихід наступного шару після нормалізації.

Після шести таких шарів отримуємо фінальні приховані представлення $H^{(6)}$.

4. Класифікаційна голова

З вектора токена $[CLS]$формується підсумкове представлення (3.9):

$$h_{[CLS]}=H_0^{(6)}, \quad (3.9)$$

де: $H_0^{(6)}$— перший рядок матриці $H^{(6)}$, що відповідає токену [CLS]. Далі подаємо його у лінійний шар (3.10, 3.11):

$$z=W_c h_{[CLS]}+b_c, \tag{3.10}$$

$$\hat{y}=\text{softmax}(z)=\left[\frac{e^{z_0}}{e^{z_0}+e^{z_1}}, \frac{e^{z_1}}{e^{z_0}+e^{z_1}}\right]. \tag{3.11}$$

де: W_c, b_c— параметри класифікатора,

$\hat{y}_1=P(\text{phishing}|x)$,

$\hat{y}_0=P(\text{safe}|x)$.

5. Функція втрат

Для бінарної класифікації використовується крос-ентропія (3.12):

$$L(\theta)=-\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_{i,1}+(1-y_i)\log \hat{y}_{i,0}\right], \tag{3.12}$$

Де: N — кількість прикладів у батчі,

$y_i \in \{0,1\}$— реальна мітка (0 = Safe, 1 = Phishing),

$\hat{y}_{i,0}$— передбачені ймовірності для кожного класу,

θ— набір усіх параметрів моделі.

6. Оптимізація

Модель оптимізується за допомогою AdamW (адаптивний градієнт + weight decay) (3.13):

$$\theta_{t+1}=\theta_t-\eta\cdot\frac{m_t/(1-\beta_1^t)}{\sqrt{v_t/(1-\beta_2^t)}+\epsilon}, \tag{3.13}$$

де: η— швидкість навчання (learning rate, $\approx 2\times10^{-5}$),

m_t, v_t— експоненційні середні градієнтів і їх квадратів,

β_1, β_2— параметри згладжування (0.9 і 0.999),

ϵ— мале число для стабільності,

weight decay виконує регуляризацію для запобігання перенавчанню.

7. Інференс

Після навчання для нового листа x^*(3.14):

$$p=\hat{y}_1=f(x^*;\theta), \tag{3.14}$$

де: p— передбачена ймовірність того, що лист є фішинговим.

На основі pприймається рішення (3.15):

$$\text{action}=\begin{cases}\text{ALLOW}, & p<\tau_1,\\ \text{WARN}, & \tau_1\le p<\tau_2,\\ \text{QUARANTINE}, & \tau_2\le p<\tau_3,\\ \text{REJECT}, & p\ge\tau_3.\end{cases} \tag{3.15}$$

де пороги τ_1,τ_2,τ_3(наприклад 0.4, 0.6, 0.9) можна задати самостійно .

Після завершення етапу побудови архітектури та визначення функцій оптимізації було сформовано набір ключових гіперпараметрів навчання, що визначають ефективність fine-tuning процесу (табл. 3.3).

Таблиця 3.3 – Основні гіперпараметри навчання моделі DistilBERT

Параметр	**Значення**
Базова модель	distilbert-base-uncased
Batch size	16
Learning rate	2×10^{-5}
Epochs	6
Max length	256
Optimizer	AdamW
Scheduler	Linear decay
Loss	Cross-Entropy
Метрики	F1, Accuracy, ROC-AUC

7. Основні переваги цієї архітектури

Для наочності основні компоненти архітектури DistilBERT і їх функціональні переваги наведено в табл. 3.4.

Таблиця 3.4 – Основні переваги архітектури DistilBERT

Компонент	**Функція**	**Перевага**
DistilBERT	Зменшений encoder BERT	у 2× швидший, 40% менше параметрів
Self-Attention	Моделює контекст	розуміє смислові залежності у тексті
Cross-Entropy Loss	Навчання класифікатора	стабільна оптимізація
AdamW	Оптимізатор	краща збіжність для великих моделей
Softmax Head	Інтерпретація	дає ймовірність фішингу для кожного листа

Як видно з таблиці, зменшена архітектура DistilBERT забезпечує до 2× вищу швидкість обчислень при збереженні високої точності моделювання контексту.

Механізм Self-Attention дозволяє моделі враховувати семантичні залежності між словами, що критично важливо для розпізнавання фішингових шаблонів у тексті електронних листів.

Комбінація оптимізатора AdamW і крос-ентропійної функції втрат гарантує стабільність збіжності, тоді як Softmax Head перетворює вихід моделі у зрозумілу ймовірнісну форму (*P(phishing), P(safe)*), що використовується у логіці Milter-рішення.

3.6.3. Результати тренування нейронної мережі

У процесі навчання моделі DistilBERT на спеціалізованому наборі електронних листів було досягнуто високих показників точності класифікації при збереженні оптимальної швидкодії.

Використання методу transfer learning дозволило суттєво скоротити час навчання та обчислювальні витрати, оскільки модель уже мала попередні знання

про структуру англійської мови та семантичні залежності між словами.

Для оцінки якості роботи моделі застосовувалися основні метрики машинного навчання — Accuracy, Precision, Recall, F1-score та ROC-AUC. Вони відображають здатність моделі точно розрізняти класи, зберігаючи баланс між повнотою та точністю передбачень.

Загальні результати тренування:

За підсумками шести епох навчання модель продемонструвала такі показники:

- Accuracy = 0.9834
- F1-score = 0.9782
- ROC-AUC = 0.999

Такі значення свідчать про надзвичайно високу роздільну здатність класифікатора між безпечними та фішинговими листами.

Модель ефективно навчилася розпізнавати:

- характерні патерни фішингових повідомлень (urgent, verify, account, password, update);
- нетипові граматичні структури та орфографічні помилки, властиві фішинговим кампаніям;
- елементи соціальної інженерії (емоційні заклики, погрози блокування, вимоги негайних дій).

Отримані результати засвідчують, що DistilBERT забезпечує точність, порівнянну з моделлю BERT-base, при цьому працює приблизно у два рази швидше, що робить її доцільною для інтеграції у реальне середовище поштової фільтрації.

Аналіз збіжності функції втрат:

Для візуальної оцінки процесу навчання побудовано графік зміни функції втрат (Loss) (рис. 3.14) під час кожного кроку оптимізації.

На початкових етапах значення втрат зменшувалося стрімко, що вказує на ефективну адаптацію моделі до специфіки даних. Після ≈20 000 кроків крива досягла стабілізації на рівні **0.02**, а різниця між навчальною та валідаційною

вибірками залишалася незначною, .

Рисунок 3.14 – Динаміка функції втрат під час навчання DistilBERT

З рисунку видно, що функція втрат зменшується монотонно, не демонструючи різких коливань або розбіжності між Train та Validation кривими.

Це свідчить про стабільну збіжність моделі та відсутність переобучення, що є критично важливим для моделі, яка працює в реальному часі в SMTP-середовищі.

Динаміка зміни метрик точності

Окрім втрат, у процесі навчання відстежувалася динаміка зміни метрики F1-score, яка є узагальненим показником якості класифікації, поєднуючи Precision та Recall, графік зображено на рискнку 3.15.

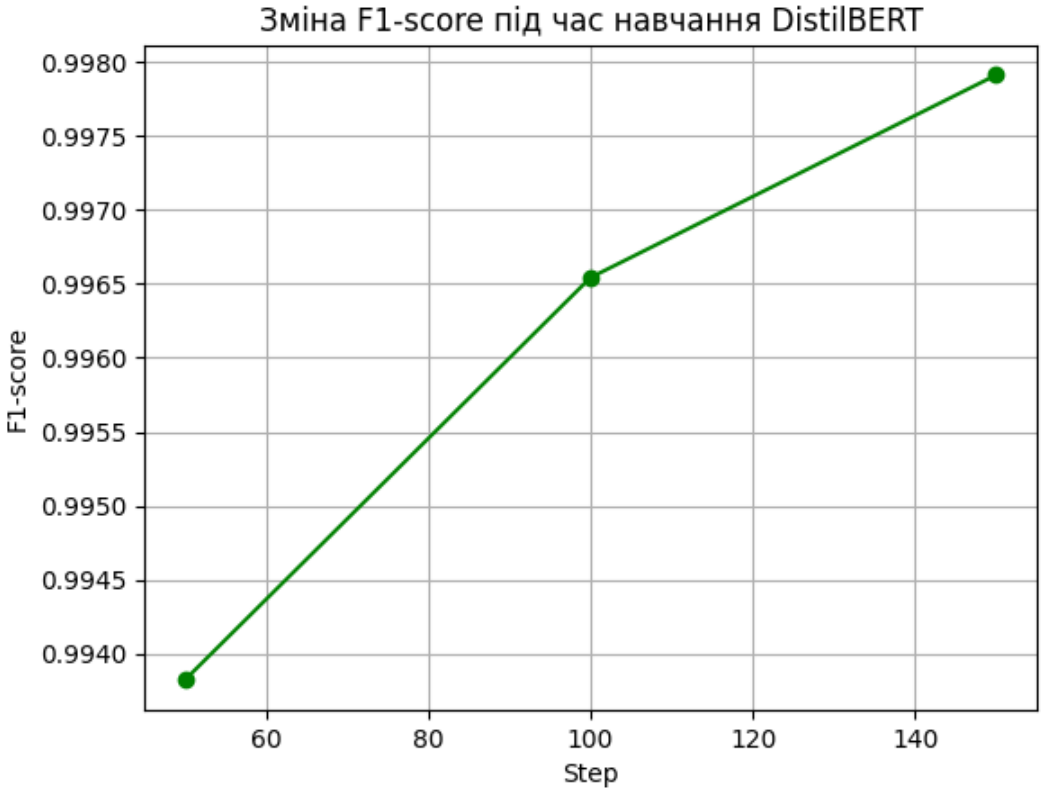

Рисунок 3.15 – Зміна F1-score під час навчання DistilBERT

Як показано на рисунку, F1-score зростає стабільно та досягає значення 0.998 після 150 навчальних кроків.

Це вказує на поступове вдосконалення класифікатора та оптимальне співвідношення між точністю передбачення позитивного класу (фішингових листів) і відсутністю помилкових спрацьовувань.

Таким чином, модель не лише правильно класифікує більшість вхідних листів, але й зберігає низький рівень хибнопозитивних результатів, що особливо важливо для корпоративного поштового середовища.

3.6.4. Тестування та оцінка моделі на незалежному наборі даних

Для оцінки узагальнювальної здатності моделі було проведено додаткове тестування на незалежному наборі даних, який не використовувався під час навчання.

Цей експеримент дозволив перевірити, наскільки ефективно натренована

модель DistilBERT справляється з новими, раніше невідомими прикладами електронних листів.

Загальні результати тестування зображені у таблиці 3.5.

Таблиця 3.5 – результати тренування

Клас	Precision	Recall	F1-score	Support
0 (безпечні листи)	0.9971	0.9764	0.9867	10 978
1 (фішингові листи)	0.9617	0.9953	0.9782	6 538
Загальна точність (Accuracy)	—	—	0.9834	17 516

Середні значення:

- Macro avg — F1 = 0.9824
- Weighted avg — F1 = 0.9835

Отримані результати підтверджують, що модель зберігає високу якість класифікації навіть на зовнішніх тестових вибірках.

Коефіцієнт точності (Accuracy = 0.9834) та значення F1 > 0.97 свідчать про добру збалансованість між Precision і Recall, тобто про низьку кількість як хибнопозитивних, так і хибнонегативних класифікацій.

Матриця плутанини (Confusion Matrix) зображена на рисунку 3.16.

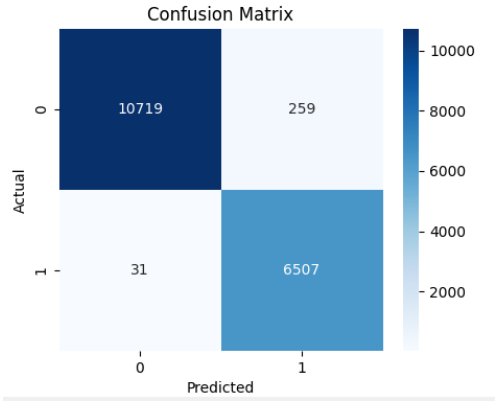

Рисунок 3.16 – Матриця плутанини для моделі DistilBERT на тестовому наборі

З рисунку видно, що з 10 978 безпечних листів лише 259 було помилково класифіковано як фішингові, тоді як із 6 538 фішингових лише 31 лист отримав статус «безпечний». Таким чином, модель досягає дуже низької частки помилок:

- False Positives ≈ 2.3 %
- False Negatives ≈ 0.5 %
- Це означає, що класифікатор здатен виявляти майже всі фішингові листи без суттєвого ризику блокування легітимної пошти.

ROC-крива та показник AUC зображена на рисунку 3.17.

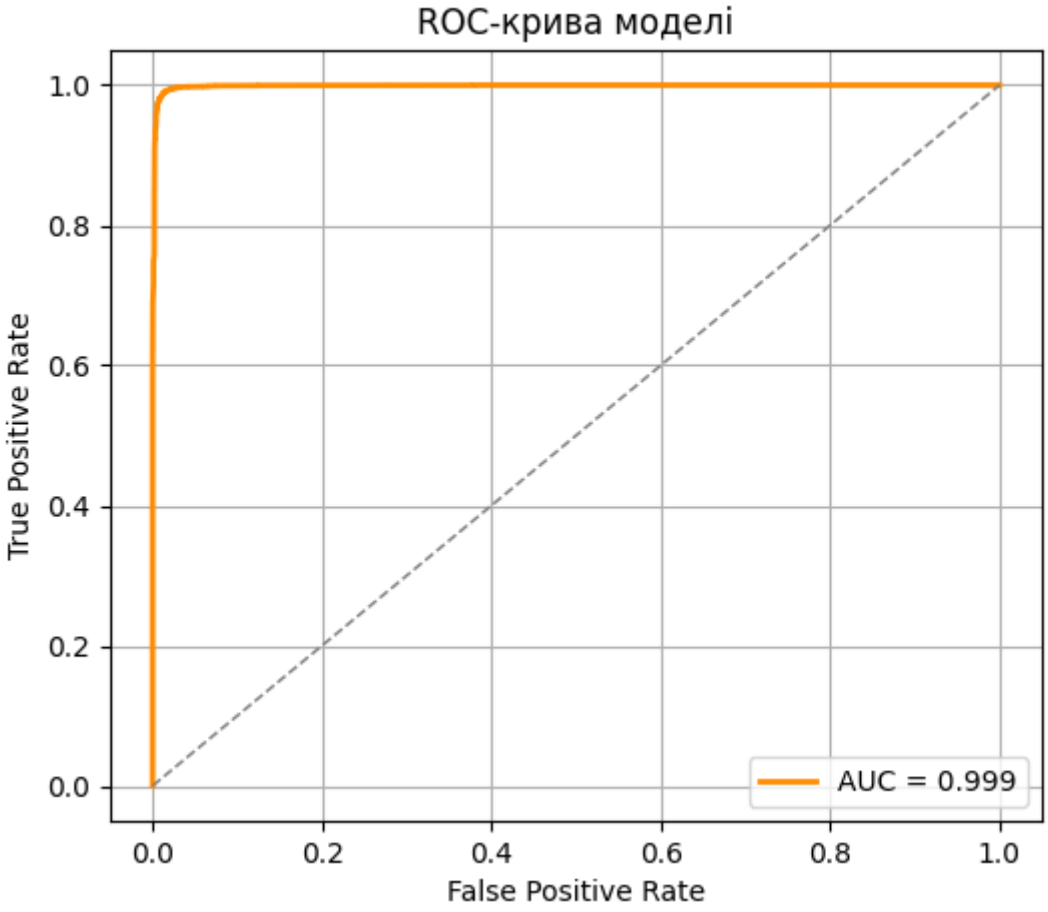

Рисунок 3.17 – ROC-крива моделі DistilBERT

На графіку показано співвідношення між True Positive Rate (TPR) і False Positive Rate (FPR) при різних порогах класифікації.

Площа під кривою (AUC = 0.999) демонструє надзвичайно високу дискримінативну здатність моделі.

Це означає, що з імовірністю 99.9 % модель правильно розрізняє фішинговий і безпечний лист навіть при варіації порогів прийняття рішення.

Розподіл імовірностей передбачення зображений на рисунку 3.18.

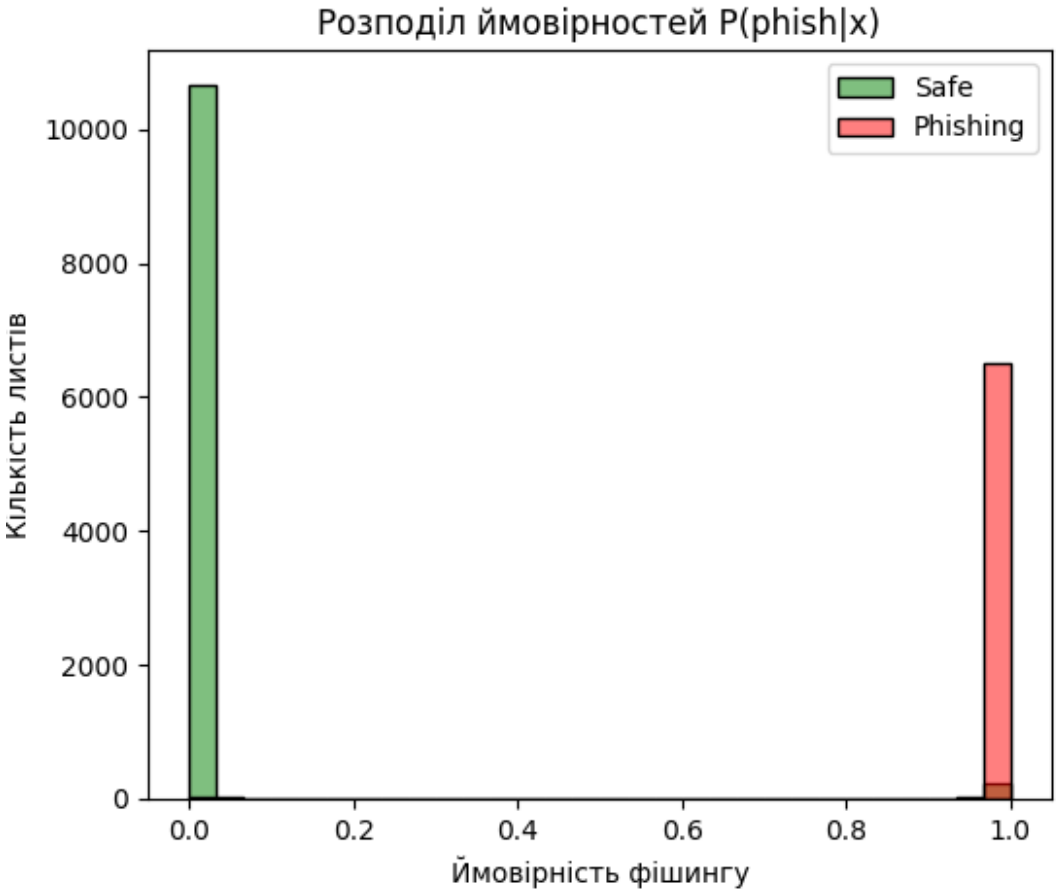

Рисунок 3.18 – Розподіл ймовірностей P(phish | x)

Розподіл передбачених імовірностей свідчить про чітке розділення двох класів:

- для безпечних листів значення P(phish | x) ≈ 0.0;
- для фішингових – P(phish | x) ≈ 1.0.

Відсутність суттєвого перекриття між класами підтверджує, що модель сформувала стійкі ознаки для класифікації і демонструє високу впевненість у своїх прогнозах.

Тестування на незалежному наборі данних підтвердило, що розроблена модель:

- коректно узагальнює закономірності, виявлені під час тренування;

- зберігає стабільність показників без ознак переобучення;
- забезпечує точність понад 98 % при AUC = 0.999, що відповідає рівню state-of-the-art-моделей для задач фішингової класифікації.

Отримані результати доводять придатність моделі DistilBERT для використання у виробничому середовищі системи AI Milter, забезпечуючи реальний аналіз вхідних SMTP-повідомлень із мінімальними затримками та високою достовірністю.

3.6.5. AI API — сервер класифікації поштового контенту

AI API — це окремий сервіс, реалізований на основі FastAPI, який виконує класифікацію поштового контенту в реальному часі.

Сервіс приймає дані від компонента AI Milter, який інтегрований у Postfix, аналізує їх за допомогою моделі DistilBERT, перевіряє виявлені URL-адреси та вкладення через VirusTotal API, а також, за потреби, перекладає текст повідомлення англійською мовою через DeepL API для майбутнього аналізу через нейромережу.

Таким чином, API є інтелектуальним шлюзом, що об’єднує нейромережеву класифікацію та зовнішні системи кіберзахисту в єдиний цикл обробки пошти.

Архітектура та середовище:

AI API працює на окремому Windows-хості з IP-адресою 192.168.1.10, тоді як Milter розгорнуто на сервері 192.168.1.12 (Ubuntu 24.04). Взаємодія між ними здійснюється через HTTP (порт 8000).

Схему взаємодії описаних сервісів зображено на рисунку 3.19.

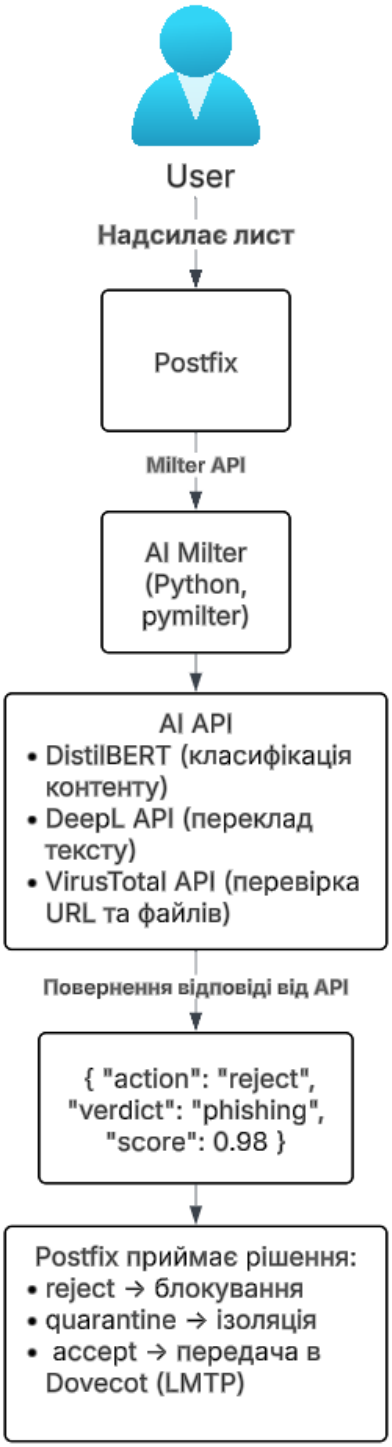

Рисунок 3.19 – Архітектура обміну даними між AI Milter та AI API

Основні компоненти API

1. FastAPI

Забезпечує високопродуктивний REST-інтерфейс для прийому HTTP-запитів від Milter. Використовується вбудована валідація запитів через Pydantic-

моделі (Mail, Attachment, AuthResults) , що підвищує стабільність обробки та дозволяє контролювати структуру вхідних повідомлень.

2. HuggingFace Transformers (DistilBERT)

Модель завантажується локально з системного каталогу: Вона автоматично використовує GPU, якщо доступна CUDA, інакше переходить у режим CPU.

3. HTML → Text Parser

Модуль _HTMLTextExtractor очищує вхідний контент від тегів HTML, залишаючи лише чистий текст для аналізу. Цей етап є критично важливим, оскільки дозволяє моделі нейромережі працювати з чистими даними без структурних артефактів..

4. Модуль DeepL перекладу

Якщо виявлено, що мова листа не є Англійською, текст автоматично перекладається на англійську перед класифікацією. Це підвищує точність моделі при багатомовних листах.

5. VirusTotal інтеграція

Окремим етапом після класифікації контенту нейронною мережею є аналіз URL-адрес та вкладень через сервіс VirusTotal – це дозволяє виявити додаткові індикатори компрометації, які не завжди присутні в тексті листа.

Для кожного знайденого посилання у повідомленні виконується попередня обробка:

- URL нормалізується до базового домену;
- якщо у листі виявлено кілька посилань, що належать до одного домену (наприклад, login.bank.com/reset, bank.com/verify), виконується єдиний запит до API по основному доменному імені;
- запит надсилається у вигляді:

https://www.virustotal.com/api/v3/domains/{domain}

- відповідь повертається зведена статистика з полями:
  - malicious — кількість рушіїв, які ідентифікували домен як шкідливий;

- suspicious — кількість підозрілих оцінок;
- harmless — підтвердження безпечності домену;
- undetected — відсутність даних у базі VirusTotal.

Для кожного домену формується агрегований звіт, який зберігається у структурі:

```
{
  "domain": "bank-secure.com",
  "malicious": 12,
  "suspicious": 3,
  "harmless": 58,
  "last_analysis_date": "2025-11-03T12:44:11Z"
}
```

Перевірка вкладень

Якщо у листі містяться файли, їх SHA256-хеш обчислюється на етапі прийому запиту (hashlib.sha256) і використовується для перевірки через:

https://www.virustotal.com/api/v3/files/{sha256}

Отримані результати додаються у спільний звіт по листу (file_report), який використовується для фінального рішення.

Вплив VirusTotal на фінальний вердикт відображено в таблиці 3.6

Після завершення основної класифікації нейромережева оцінка (phishing_score) коригується з урахуванням репутаційних даних з VirusTotal. Система комбінує обидва джерела інформації за допомогою евристичних правил, що дозволяє знизити кількість помилкових спрацьовувань і підвищити впевненість у прогнозі.

Таблиця 3.6 – Вплив VirusTotal на фінальний вердикт.

Умова	Корекція оцінки	Пояснення
malicious ≥ 4	phishing_score += 0.7	Домен або вкладення мають підтверджену шкідливу активність
1 ≤ suspicious ≤ 3	phishing_score += 0.4	Часткові підозри у сторонніх джерелах або неоднозначні результати аналізу домену/файлу.
harmless > 50	phishing_score -= 0.1	Репутаційно безпечний домен або вкладення; корекція в бік зниження ризику.
no results	без змін	Нейтральна ситуація — оцінка лише за моделлю

Алгоритм корекції phishing_score діє як надбудова над результатом нейронної моделі. Він підсилює підозру у разі підтвердженого ризику (malicious ≥ 4) або шкідливих вкладень, і навпаки — зменшує ймовірність фішингу, якщо всі елементи листа мають позитивну репутацію. Таким чином, система приймає збалансоване рішення, враховуючи одночасно:

- контентну оцінку (нейромережа DistilBERT),
- репутаційну оцінку (VirusTotal доменів),
- перевірку вкладень (hash-based reputation) за допомогою VirusTotal.

Логіку алгоритму обробки запиту зображено на рисунку 3.20.

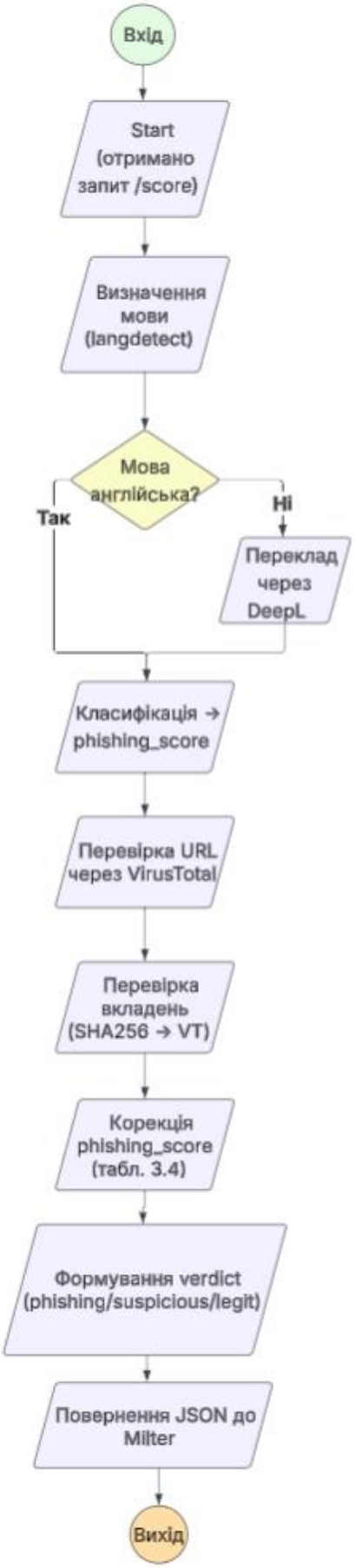

Рисунок 3.20 – Логіка обробки поштового повідомлення в AI API

1. Отримання вхідного запиту /score від Milter (JSON з полями headers, body, urls, attachments).

2. Визначення мови тексту (langdetect) → за потреби переклад на англійську.

3. Передача тіла повідомлення до моделі DistilBERT, яка повертає ймовірності:

- score(0) — безпечний лист;
- score(1) — фішинговий лист.

4. Виконання додаткових перевірок VirusTotal (доменів та хешів).

5. Формування підсумкового вердикту зображено в таблиці 3.7:

Таблиця 3.7 – Формування підсумкового вердикту

Умова	**Вердикт**	**Дія**
phishing_score ≥ 0.7 і label = “1”	phishing	reject
label = “1”, але score < 0.7	suspicious	quarantine
phishing_score > 0.3 при label = “0”	suspicious	accept
Інше	legit	accept

6. Надсилання відповіді у форматі JSON:

```
{
  "action": "reject",
  "verdict": "phishing",
  "score": 0.92,
  "lang": "en",
  "translated_used": false
}
```

7. Логування та форензіка

Журнали класифікації зберігаються у *logs/inference.log*

Сирі повідомлення для аналізу зберігаються у *logs/incoming/*. Кожне зберігається у форматі JSON з метаданими (from, to, subject, score, domains, hashes).

Логіка взаємодії з Postfix

Після отримання відповіді:

- якщо action = reject → Postfix блокує повідомлення;
- якщо action = quarantine → лист ізолюється в окрему поштову скриньку;
- якщо action = accept → повідомлення передається в Dovecot через LMTP для подальшої доставки.

Таким чином, AI API виступає центральним елементом автоматичного фільтрування пошти, що дозволяє:

- гнучко адаптувати правила;
- додавати нові перевірки (VirusTotal, SPF, DKIM);
- вести повний лог і форензічний трек усіх оброблених повідомлень.

Приклад журналу класифікації:

{"ts_utc": "2025-11-02T17:48:00Z",
"message_id": "<e05bc365-83e4-4134-8884-4d5dce5419b3 >",
"from": "<testuser1@test-app.org>",
"to": ["<mgrama@test-app.org>"],
"subject": "hello",
"action": "reject",
"verdict": "phishing",
"phishing_score": 0.9999914169311523,
"legit_score": 8.631779564893804e-06,
"lang": "en",
"translated_used": false,
"domains": [],
"hashes": []}

Це дозволяє адміністратору відслідковувати всі вхідні інциденти без потреби перегляду поштових скриньок користувачів.

3.6.6. Підсумкова схема обробки поштових повідомлень

В результаті виконання попередніх пунктів було побудовано систему поштової фільтрації об’єднує всі розглянуті компоненти (Postfix, AI Milter, AI API, Dovecot, OPNsense, Cloudflare DNS) у єдину архітектуру з наскрізним контролем усіх етапів доставки повідомлень.

Система реалізує повний цикл — від моменту прийому електронного листа зовнішнім сервером до його аналізу, доставки або блокування, а також відправлення автентифікованих вихідних листів до зовнішніх доменів.

Вхідний поштовий трафік зображено на рисунку 3.21.

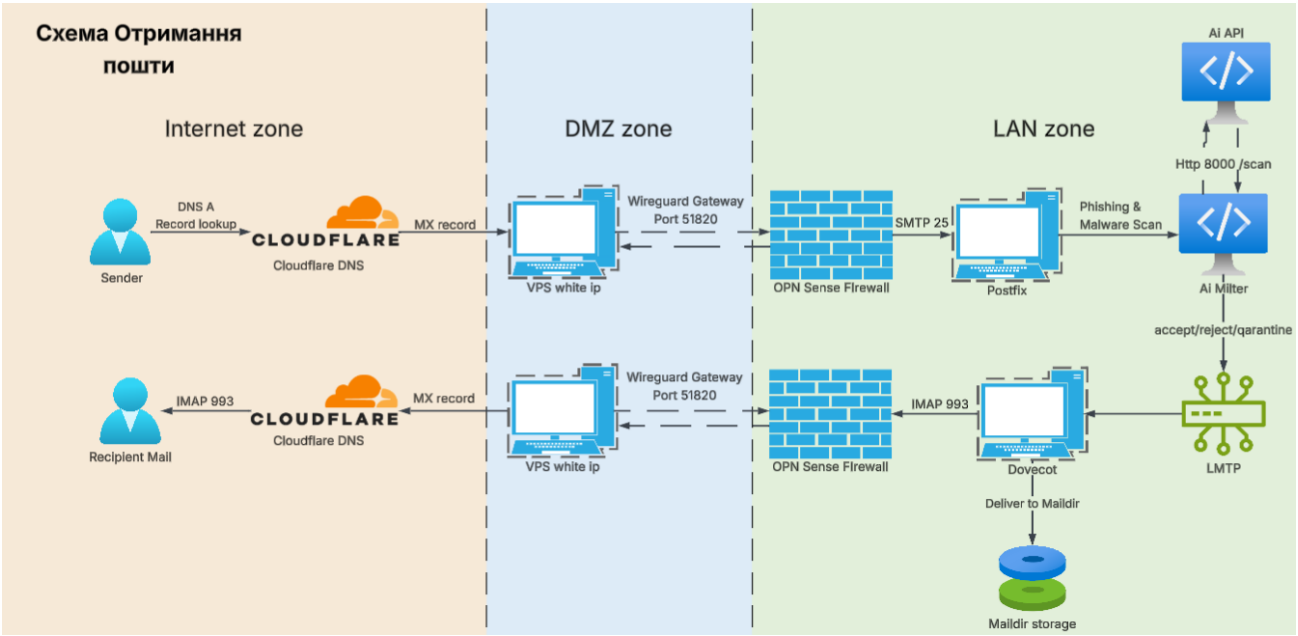

Рисунок 3.21 – Схема отримання пошти

Вхідні повідомлення надходять до домену через Cloudflare DNS, який за MX-записом перенаправляє їх на VPS у DMZ-зоні. Далі пошта передається через зашифрований тунель WireGuard (порт 51820/UDP) до міжмережевого екрану OPNsense, який пропускає трафік на сервер Postfix у LAN-зоні. На етапі прийому пошта проходить через AI Milter, який взаємодіє з AI API через HTTP (порт 8000). API виконує аналіз тексту повідомлення за допомогою моделі DistilBERT, перевіряє URL-адреси та вкладення через VirusTotal API, а також за потреби перекладає вміст через DeepL API. На основі отриманих результатів AI API

повертає вердикт (accept, reject або quarantine), який визначає подальшу дію Postfix:

- reject — повідомлення блокується на рівні SMTP;
- quarantine — поміщається в ізольовану поштову скриньку для перевірки;
- accept — доставляється у Dovecot через LMTP.

Dovecot, у свою чергу, зберігає листи у форматі Maildir і надає доступ користувачам через IMAP (порт 993/tcp, TLS). Таким чином, весь процес прийому листів супроводжується інтелектуальною перевіркою, логуванням та контролем на рівні мережі, контенту й користувача.

Вихідний поштовий трафік

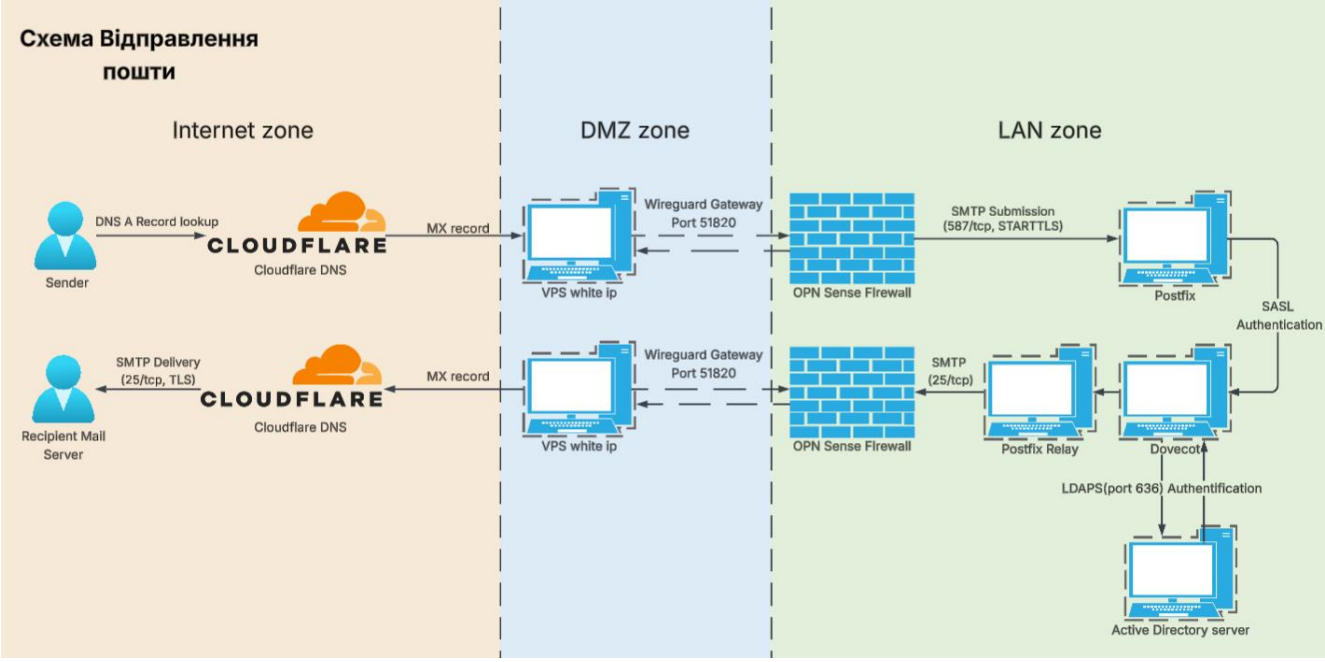

Рисунок 3.22 – Схема відправлення пошти

Вихідні листи від корпоративних користувачів проходять інший транспортний ланцюг. Користувач здійснює SMTP Submission через порт 587/tcp (STARTTLS) до Postfix, після чого виконується автентифікація за протоколом SASL, що забезпечує перевірку облікового запису.

Для перевірки облікових даних використовується Dovecot, який звертається до каталогу Active Directory через LDAPS (порт 636/tcp). Це дозволяє

централізовано управляти доступом до поштової інфраструктури.

Після перевірки автентичності:

1. лист передається на Postfix Relay,
2. OPNsense перевіряє маршрутизацію,
3. трафік VPN-тунелем WireGuard направляється до VPS у DMZ,
4. VPS передає повідомлення на зовнішні поштові сервери через Cloudflare DNS

3.7. Підсистема керування обліковими записами Keycloak

Для забезпечення безпечної роботи з корпоративною поштою та мінімізації ризику компрометації облікових даних була впроваджена підсистема керування автентифікацією користувачів на базі Keycloak, що виконує функцію центрального сервісу управління обліковими записами (IAM — Identity and Access Management). Вона забезпечує:

- самостійну зміну тимчасових паролів користувачами;
- можливість застосування політик складності паролів;
- реалізацію двофакторної автентифікації (2FA / TOTP);
- централізований аудит подій автентифікації;
- інтеграцію з Active Directory через LDAPS (порт 636/TCP).

Keycloak використовується як зовнішній сервіс самообслуговування для користувачів корпоративної пошти, що дозволяє виконувати зміну пароля без звернення до адміністратора, а також значно знижує операційне навантаження на службу технічної підтримки.

3.7.1. Керування доменом авторизації (Realm) та URL-доступ

Для З метою безпечної публічної доступності сервісу створено окремий realm — логічно ізольований домен автентифікації, у якому зберігаються правила доступу, політики безпеки, механізми 2FA та параметри синхронізації з Active Directory.

Загальні параметри доступу наведено у таблиці 3.8.

Таблиця 3.8 – Схема доступу до keycloack

Параметр	Значення
Портал доступу (Frontend URL)	https://auth.test-app.org/
Realm (домен авторизації)	corp-auth
OpenID / OAuth2 Endpoints	https://auth.test-app.org/realms/corp-auth/...
Підтримувані протоколи	OpenID Connect, SAML 2.0
Шифрування трафіку	TLS 1.2+ (Let's Encrypt / Cloudflare SSL Full Strict)

Конфігурація Frontend URL відокремлена від внутрішньої адресації сервера, що дозволяє розміщувати Keycloak у закритому LAN-сегменті, відкриваючи доступ користувачам виключно через захищений реверс-проксі, на рисунку 3.20 показано вигляд сторінки входу на Keycloack .

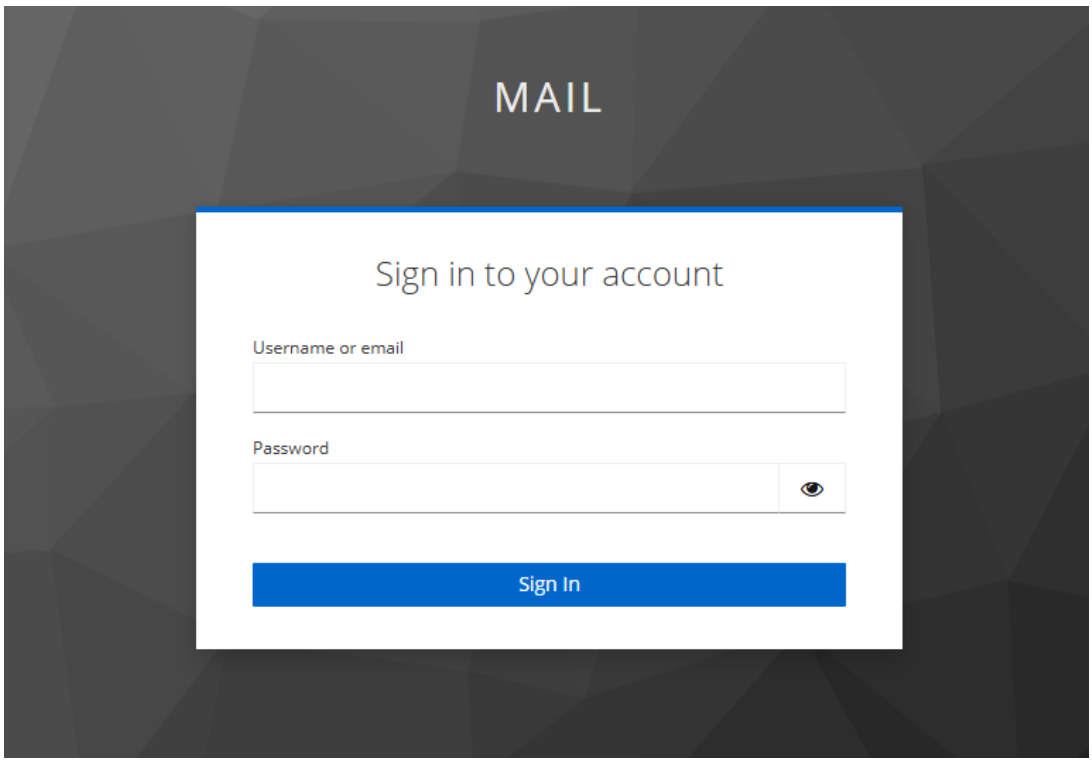

Рисунок 3.23 – Форма входу Keycloak (Frontend URL)

3.7.2. Інтеграція з Active Directory (LDAPS)

Для синхронізації користувачів поштової системи з корпоративним сховищем облікових даних була виконана інтеграція Keycloak з Active Directory (рис. 3.24). В AD створено окремий службовий обліковий запис із мінімально необхідними правами.

Приклад параметрів налаштування сервісного облікового запису описано в таблиці 3.9

Таблиця 3.9 – Налаштування сервісного облікового запису keycloack

Параметр	**Значення**
Обліковий запис	svc_keycloak@corp.local
Розміщення	OU=Service Accounts,DC=corp,DC=local
Права	reset password, read user attributes, read group membership
Автентифікація	LDAPS (TLS)
Синхронізація атрибутів	sAMAccountName, mail, displayName

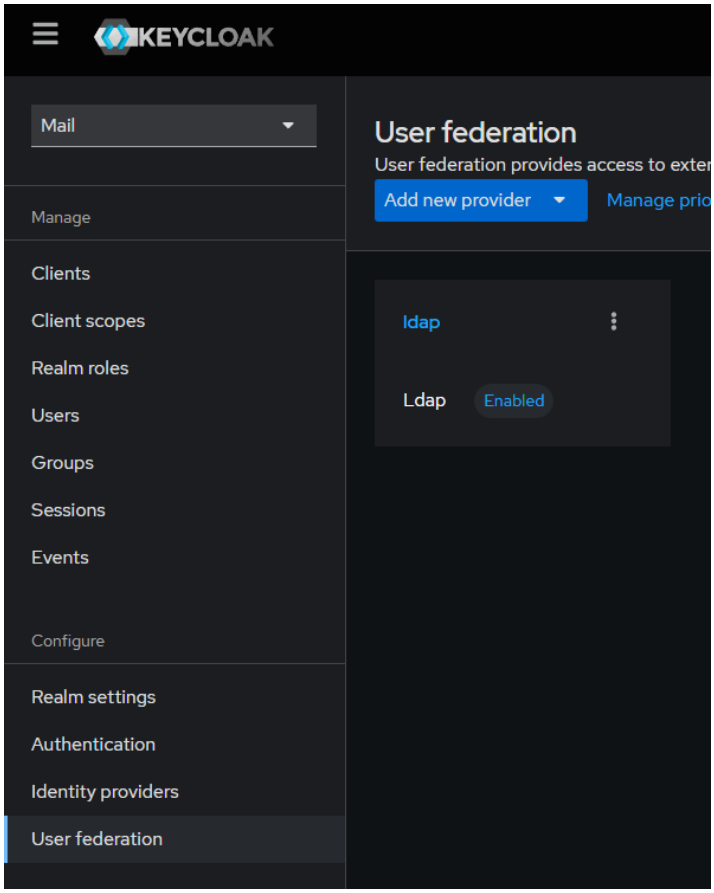

Рисунок 3.24 – Налаштоване LDAP підключення Keycloak до Active Directory через LDAPS для реалму MAIL

Синхронізація користувачів налаштована в режимі Read-Only with password update, тобто Keycloak не створює облікові записи, але має можливість ініціювати зміну пароля в AD при першому вході користувача.

Сховище користувачів налаштовано за фільтром LDAP:

(&(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))

3.7.3. База даних Keycloak та забезпечення 2FA

Для повноцінного зберігання конфігурацій, журналів подій та налаштувань двофакторної автентифікації (TOTP) використовується окрема серверна СУБД, опис налаштування якої наведено в таблиці 3.10:

Таблиця 3.10 – Налаштування бази данних keycloack

Компонент	**Значення**
Тип БД	PostgreSQL 14+
Розміщення	окремий сервер/контейнер (не на Keycloak)
Підключення	Контейнер розміщено на сервері в Lan зоні , без публічного доступу
Дані зберігаються	конфігурації realm, TOTP-секрети, журнали авторизації

Підтримується двофакторна автентифікація (TOTP) через:

- Google Authenticator
- Microsoft Authenticator
- FreeOTP
- апаратні токени (апаратні FIDO-клієнти)

Щоб забезпечити безпечний перехід користувача від тимчасового пароля, виданого адміністратором, до постійного, в Keycloak були активовані обов’язкові *Required actions* для нового користувача. Це гарантує, що перед першим використанням корпоративної пошти користувач:

1. налаштує мобільний застосунок для TOTP-кодів;
2. змінить тимчасовий пароль на постійний, що відповідає політикам складності.

У налаштуваннях realm було увімкнено такі дії (рис 3.25):

- Configure OTP — зобов’язує користувача активувати TOTP при першому вході
- Update Password — примусова зміна тимчасового пароля на постійний

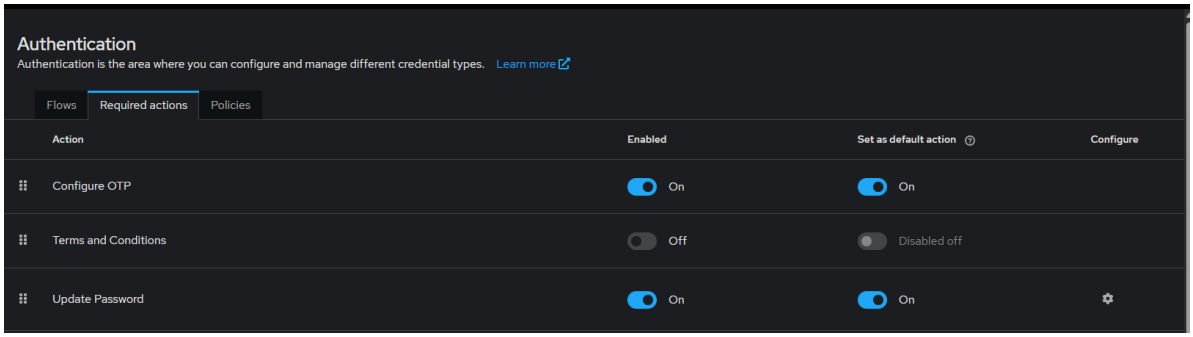

Рисунок 3.25 – Налаштування політики двофакторної автентифікації (TOTP)

3.7.4. Роль Keycloak у загальній архітектурі системи

Keycloak є ключовим елементом безпеки всієї корпоративної поштової інфраструктури, оскільки забезпечує функції котрі описані в таблиці 3.11:

Таблиця 3.11 – Ключові функції keycloack

Функція	**Значення для системи**
Єдина точка автентифікації	централізація управління доступом
LDAPS інтеграція	синхронізація з корпоративним AD
2FA TOTP	суттєве зменшення ризику компрометації
Аудит сесій	контроль інформаційної безпеки
Зміна пароля без участі IT	зниження навантаження на адміністраторів

Таким чином, Keycloak суттєво підвищує рівень кіберзахисту корпоративної електронної пошти та узгоджується з принципами Zero Trust, а також рекомендаціями ISO/IEC 27001 щодо контролю доступу.

Реалізація підсистеми керування доступом Keycloak забезпечила:

- впровадження безпечного механізму доступу до поштової системи;

- можливість контролю автентифікації, зміни пароля та аудиту подій;
- надійну інтеграцію з Active Directory через захищений LDAPS-канал;
- додатковий рівень захисту за рахунок технології 2FA/TOTP;
- зниження операційних витрат на підтримку користувачів.

Keycloak розглядається як фундаментальний компонент комплексної системи захисту корпоративних поштових комунікацій, здатний до масштабування та інтеграції з зовнішніми засобами SOC/SIEM.

3.8. Налаштування та робота з Grafana

Ефективна система фільтрації електронної пошти повинна не лише блокувати шкідливі повідомлення, але й забезпечувати повну прозорість процесів, можливість оперативного аналізу загроз та швидке ухвалення рішень щодо блокування або ігнорування потенційних джерел атак. З цією метою у межах розробленого рішення впроваджено дашборд MailSecurity Dashboard у Grafana, який виконує функції візуалізації телеметрії та централізованого реагування інцидентів.

Дашборд використовує журнали обробки листів, збережені у Loki, а також отримує дані з внутрішнього API керування блокуваннями. У сукупності це створює зручний інструмент для роботи співробітників відділу інформаційної безпеки та аналітиків SOC.

3.8.1. Інтеграція Grafana з Loki

Для підключення журналів було додано джерело даних типу Loki. Конфігурація виконувалася через вебінтерфейс Grafana (розділ *Connections* → *Data sources*).

У полі *URL* вказано адресу служби Loki Після успішної перевірки з’єднання система почала отримувати лог-записи з сервера поштової фільтрації, джерела данних вказано в таблиці 3.12.

Таблиця 3.12 – Джерела данних Grafana та їх призначення

Джерело даних	Тип	Призначення
Loki	Loki	Отримання журналів класифікації електронних листів (результати AI Milter/AI API)
mail-actions	Infinity	Виклик API для блокування/розблокування відправників або доменів

Підключення конекторів виконується через меню(рис 3.26):

Grafana → Connections → Data sources

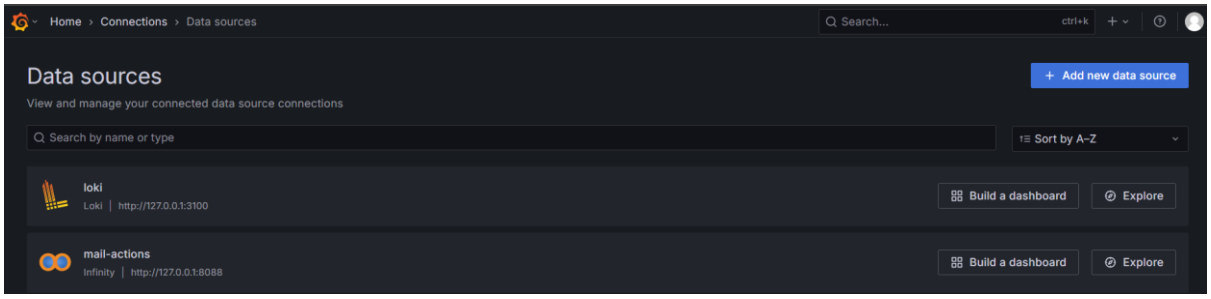

Рисунок 3.26 – Налаштовані джерела даних Loki та mail-actions (Infinity Data Source)

Джерело даних Loki: збір журналів класифікації листів

Loki використовується як центральне сховище логів, що містять результати обробки повідомлень модулем AI Milter/AI API. У журнал надсилаються такі поля:

- ts_utc — час обробки повідомлення,
- from, to, subject,
- phishing_score і legit_score,
- verdict (accept / reject / quarantine),
- список знайдених domains.

Це дозволяє:

- аналізувати інциденти без доступу до серверів;
- будувати графіки поштового трафіку;
- створювати правила алертингу;
- визначати джерела потенційних фішингових атак.

Джерело даних Infinity (mail-actions API)

Другим джерелом даних виступає конектор Infinity, який використовується для інтеграції Grafana з внутрішнім API керування блокуванням відправників та доменів. Завдяки цьому дашборд перетворюється не лише на моніторингову, а й на операційну консоль реагування.

API дозволяє виконувати операції описані в таблиці 3.13:

Таблиця 3.13 – Джерела данних Grafana та їх призначення

Метод	Призначення
GET /api/list	Отримати список заблокованих адрес/доменів
POST /api/block	Додати адресу/домен у список блокування
POST /api/unblock	Розблокувати адресу/домен

Endpoint Infinity Data Source:

http://127.0.0.1:8088

Для виконання запитів використовується авторизація:

Authorization: Bearer <secret_token>

Налашткування конектору для Infinity показано на рисунку 3.27.

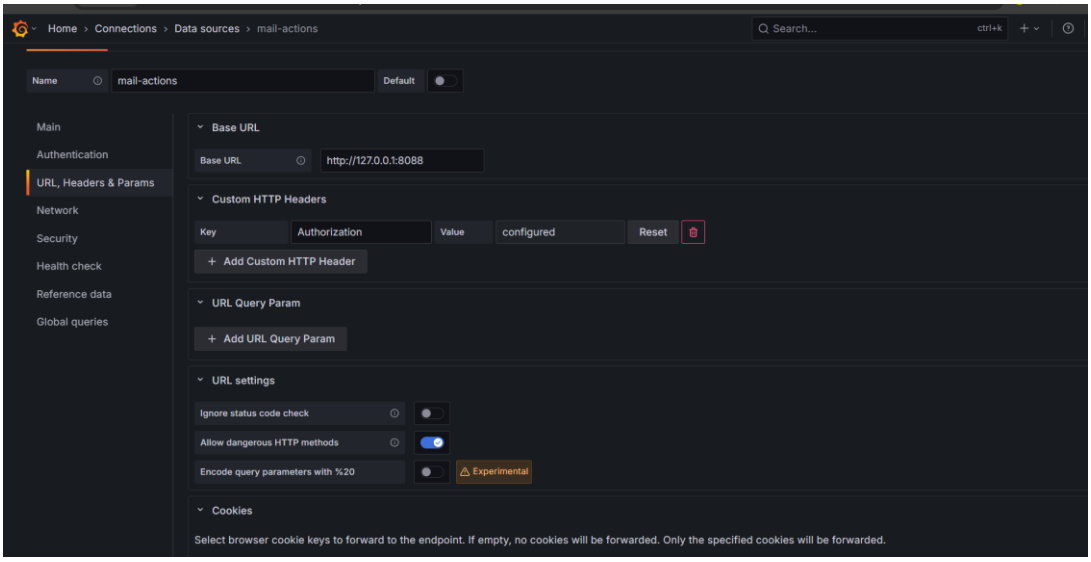

Рисунок 3.27 – Налаштування Infinity (mail-actions) Data Source

3.8.2. Основні елементи аналітичного дашборду

У створеному Mail Security Dashboard реалізовано візуальні модулі, які дозволяють швидко оцінювати стан поштової безпеки. Основні компоненти описано в таблиці 3.14, головний дашборд системи зображено на рисунку 3.28.

Таблиця 3.14 – Джерела данних Grafana та їх призначення

Елемент дашборду	Призначення
Gauge “Кількість листів за добу”	Показує обсяг поштового трафіку, допомагає виявляти аномальні піки надсилань
Таблиця заблокованих відправників/доменів з кнопкою Unblock	Дає змогу оперативно керувати блокуваннями без доступу до Postfix вручну
Pie Chart “Домени, що використовувалися для відправлення листів”	Визначає частку підозрілих або невідомих доменів серед усіх джерел

Продовження таблиці 3.14

Елемент дашборду	Призначення
Email Throughput Timeline	Відображає пропускну здатність пошти за годинами/днями
Alerts: PhishingEmailDetected	Сповіщає про виявлені фішингові інциденти з автоматичним створенням алертів

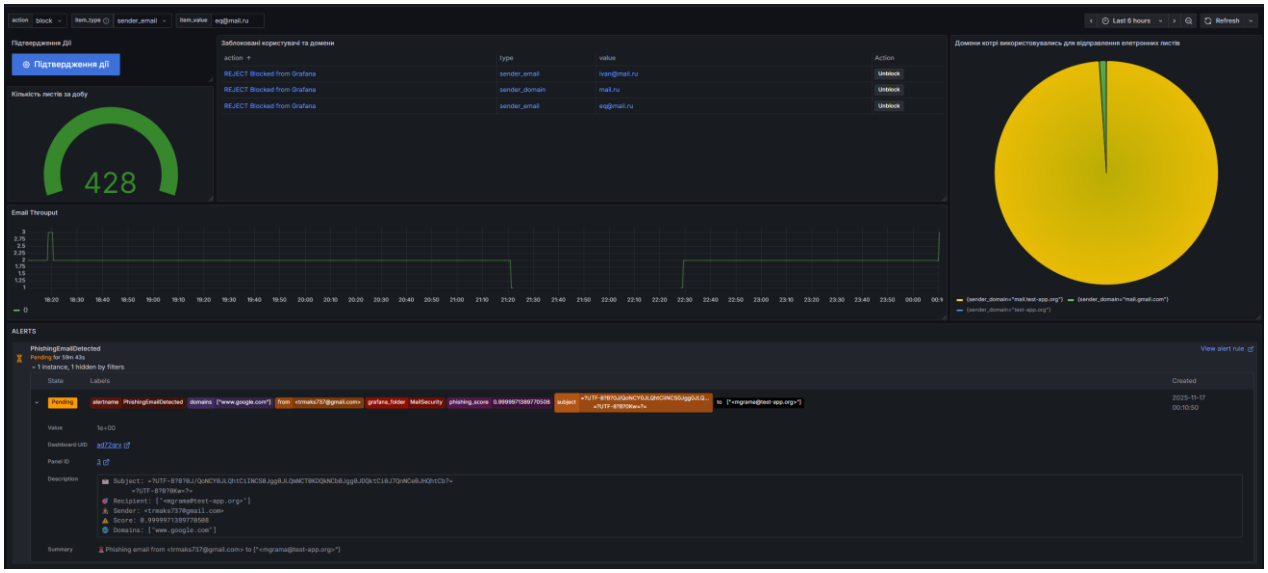

Рисунок 3.28 – Головний дашборд системи

3.8.3. Опис основних візуалізацій

Gauge-панель «Кількість листів за добу», зображена на рисунку 3.29.

Панель відображає загальну кількість електронних листів, що були оброблені системою за останні 24 години. Така метрика є критично важливою для:

- відстеження пікових навантажень на систему;
- оцінки обсягу потенційних фішингових атак;
- подальшого планування ресурсів поштової інфраструктури.

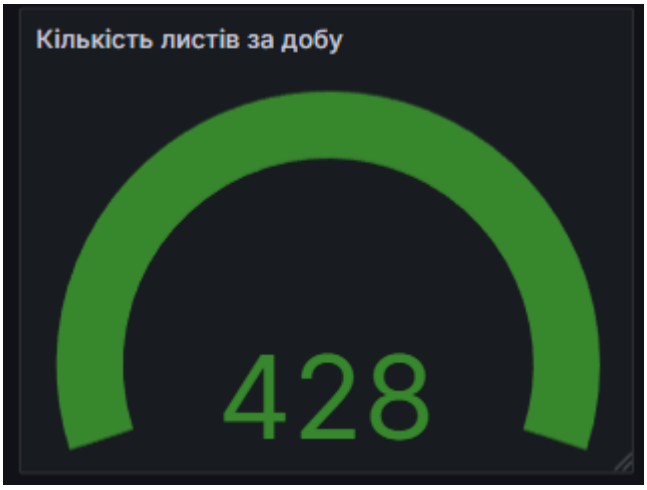

Рисунок 3.29 – візуалізація кількості листів за добу

Таблиця «Заблоковані користувачі та домени», зображена на рисунку 3.30.

У цій таблиці відображаються всі домени та електронні адреси, які були заблоковані автоматично або вручну. Для кожного запису зазначаються:

- тип блокування: email, domain або regex;
- конкретне значення (адреса/домен);
- причина блокування;
- кнопка Unblock для скасування блокування.

Це дозволяє аналітикам швидко реагувати на хибні спрацювання та забезпечує керованість політики блокування без доступу до серверів.

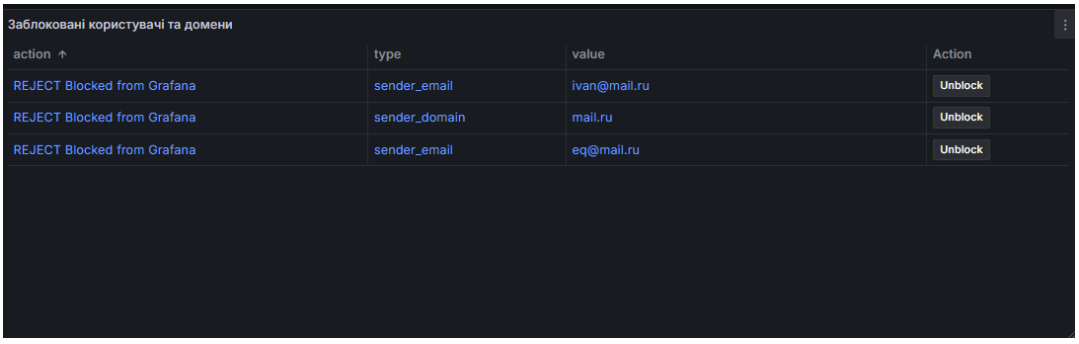

Рисунок 3.30 – візуалізація Заблокованих користувачів та доменів

Також при натисканні кнопки Unblock з’являється підтвердження для розблокування, приклад зображено на рисунку 3.31.

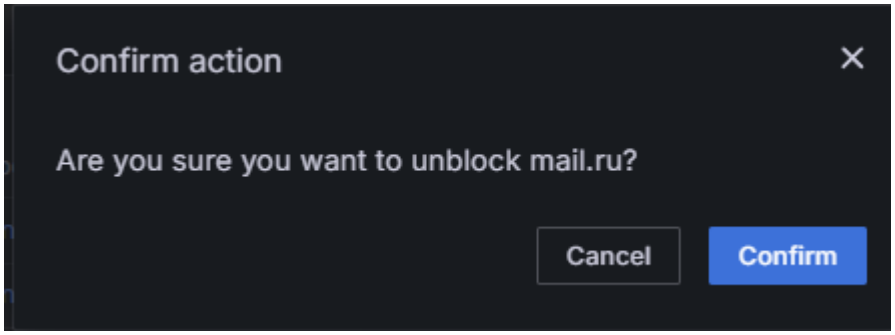

Рисунок 3.31 – Попередження щодо підтвердження дії

Кругова діаграма «Домени відправників», зображена на рисунку 3.32.

Діаграма використовується для:

- виявлення доменів, які генерують аномально великий обсяг листів;
- аналізу поширених джерел фішингових повідомлень;
- візуальної ідентифікації загроз (наприклад, несподівана активність mail.ru, gmail.com тощо).

У корпоративних середовищах це допомагає розуміти, чи є загроза масовою та звідки вона походить.

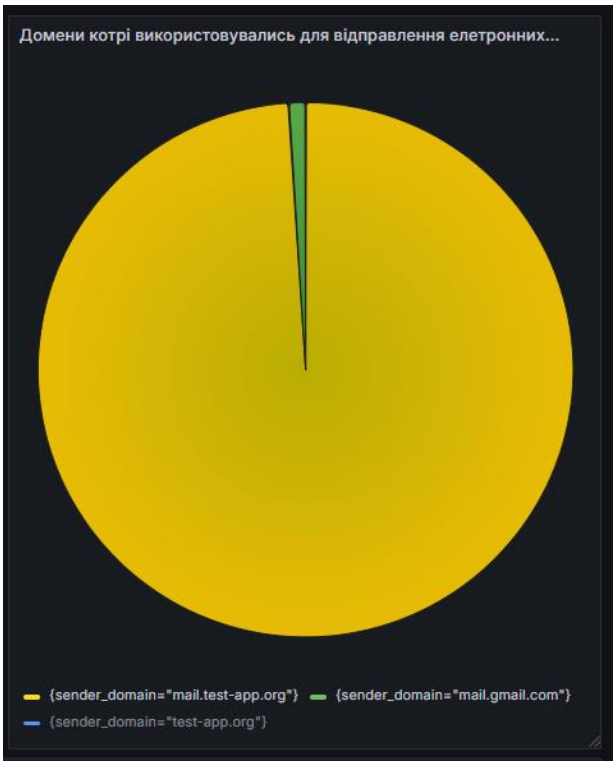

Рисунок 3.32 – візуалізація доменів відправників

Кількість електронних листів які обробляє сервер у реальному часі. Ця візулізація дає можливість оцінити:

- стабільність роботи SMTP-сервера,
- поведінку системи під час атак,
- збільшення або падіння кількості листів у динаміці.

Це важливо для виявлення аномальних хвиль, що можуть бути пов’язані з масовими фішинговими кампаніями, приклад візуалізації відображено на рисунку 3.33.

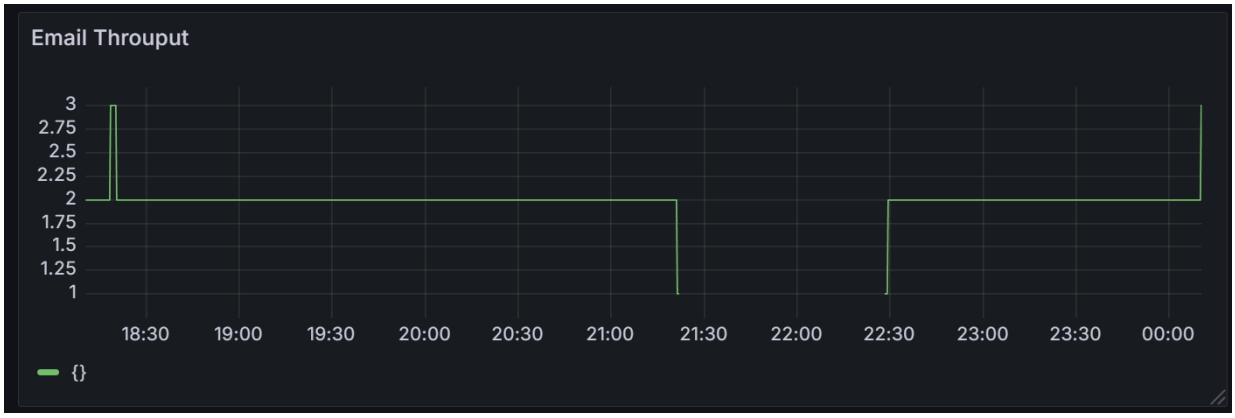

Рисунок 3.33 – Візуалізація кількість електронних листів які обробляє сервер у реальному часі

Панель «Alerts: PhishingEmailDetected», зображена на рисунку 3.34.

Показує інциденти автоматичного виявлення ймовірних фішингових листів. Дані включають:

- відправника та отримувача,
- тему листа,
- перелік доменів у тексті або вкладеннях,
- значення phishing_score моделі,
- час спрацювання.

Це дозволяє оперативно провести аналіз інциденту без звернення до серверних логів.

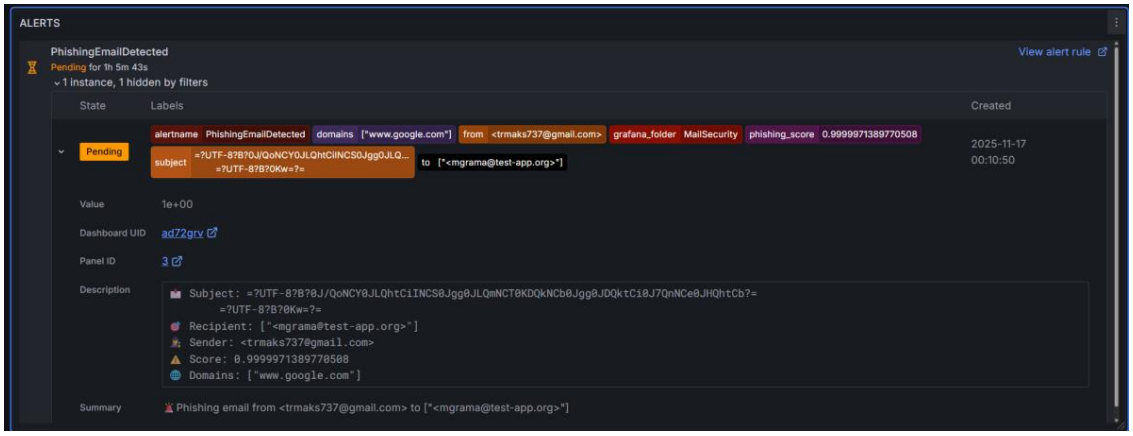

Рисунок 3.34 – Панель інцидентів автоматичного виявлення ймовірних фішингових листів

Окремим елементом на дашборді поштової безпеки є панель "Підтвердження дії", що забезпечує можливість ручного блокування або розблокування:

- окремих електронних адрес відправників;
- доменів другого або третього рівня;
- інших типів відправників (за наявності регулярних виразів у конфігурації API).

Дана панель побудована на основі механізму Infinity Query → Fetch Action, що дозволяє виконувати HTTP-запити до Mail Actions API напряму з Grafana без доступу до поштового сервера, візуалізація зображена на рисунку 3.35.

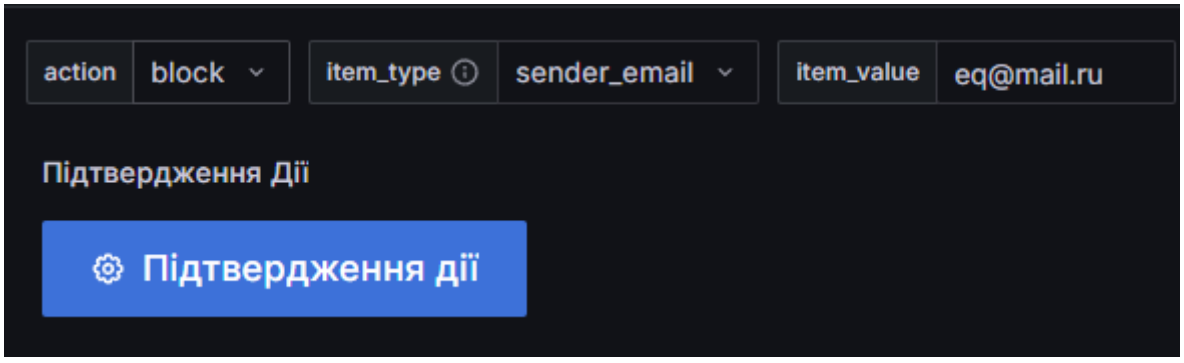

Рисунок 3.35 – Панель блокування або розблокування доменів чи відправників

Після заповнення параметрів адміністратор натискає кнопку:

Підтвердження дії

Це запускає API-запит наступної логіки:

Якщо action = block → адреса/домен додається до Postfix sender_access та пошта відмовляється на SMTP-рівні.

Якщо action = unblock → запис видаляється з sender_access і доставка листів відновлюється.

Панель підтвердження дії є ключовим елементом системи реагування, адже вона:

- забезпечує миттєве блокування загрозового джерела;
- дозволяє виконувати дії без втручання у налаштування Postfix вручну;
- зменшує час реагування на фішингові інциденти; підвищує ефективність роботи аналітиків безпеки.

Панель перетворює Grafana з інструмента моніторингу на централізований оперативний центр контролю електронної пошти та кіберзагроз.

3.8.4 Висновок

У межах реалізації системи моніторингу та реагування на фішингові листи було побудовано інтерактивний дашборд Grafana, який забезпечує не лише аналітичне спостереження, але й можливість оперативного втручання у поштову інфраструктуру. Такий підхід трансформує систему з пасивної моделі моніторингу у повноцінний інструмент активної протидії фішинговим атакам.

Однією з ключових складових дашборду є модуль ручного блокування/розблокування доменів та відправників через інтеграцію з Mail Actions API, що значно спрощує роботу аналітиків інформаційної безпеки.

Таблиця 3.15 – переваги використання розробреного дашьлорду

Функціональна можливість	Перевага
Оперативність реагування	Відпадає потреба у доступі до серверів, редагуванні конфігурацій Postfix або використанні консолі
Зменшення ризику людських помилок	API самостійно перевіряє коректність домену/адреси та перегенеровує map-файли
Зручність для аналітиків SOC	Інструментом можна користуватись навіть без прямого доступу до інфраструктури
Повна інтеграція з журналами	Заблоковані об’єкти одразу відображаються у таблицях дашборду
Безпека виконання операцій	Використовується Bearer-токен авторизації та контроль доступу до API

Таблиця 3.16 – Сценарій використання розробленого дашборду

Сценарій	Приклад
Виявлення підозрілого домену серед джерел відправників	block + sender_domain + mail.ru
Підтверджена фішингова атака за результатами VirusTotal	block + sender_email + scam@phishingsite.net
Скасування хибного спрацювання детектора	unblock + sender_email + noreply@paypal.com

Розроблений дашборд Grafana та інтегрований механізм блокування забезпечують:

- скорочення часу реагування на інциденти;
- прозорість та відтворюваність дій спеціалістів;

- підвищення рівня кібербезпеки організації;
- зменшення навантаження на команду безпеки та автоматизацію рутинних процесів.

Таким чином, система не лише надає повну видимість поштового трафіку, але й дозволяє ефективно керувати ризиками та запобігати компрометації внутрішньої інфраструктури.

3.9. Функціональне тестування системи

З метою перевірки коректної роботи розробленої системи захисту електронної пошти було проведено функціональне тестування, яке дозволило оцінити реакцію компонентів системи на різні типи повідомлень: фішингові, легітимні зовнішні та внутрішні корпоративні листи. Тестування виконувалося у реальних умовах роботи протоколів SMTP/IMAP із залученням Postfix, AI Milter, API-сервісу класифікації, Dovecot та підсистеми моніторингу Loki/Grafana.

3.9.1. Приклад блокування фішингового листа (ML policy)

Під час тестування було сформовано контрольоване фішингове повідомлення із використанням типових ознак соціальної інженерії (обіцянка виграшу, спонукання перейти за посиланням, вимога надати персональні дані).

Система класифікувала повідомлення як шкідливе, що призвело до його блокування на SMTP-рівні, приклад зображено на рисунку 3.36.

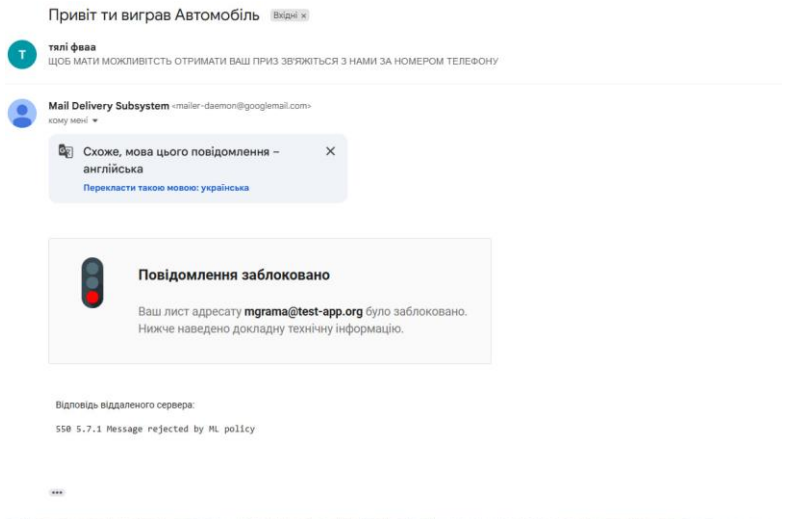

Рисунок 3.36 – Приклад відпрацювання блокування фішингового листа

Відправнику повернуто відповідь:

550 5.7.1 Message rejected by ML policy

Це свідчить, що політика AI Milter була застосована, а доставка листа була припинена ще до зберігання у поштовій скриньці користувача,

3.9.2. Приклад успішної доставки легітимного листа

Для підтвердження працездатності системи за відсутності загроз було відправлено легітимний електронний лист без підозрілих елементів, посилань і вкладень. Повідомлення пройшло перевірку AI Milter, отримало вердикт *accept* та було доставлено користувачу через Dovecot IMAP.

Таким чином, система не створює перешкод для штатної комунікації всередині організації та з зовнішніми контрагентами, що дозволяє уникнути додаткового навантаження на служби підтримки, приклад успішної доставки відображено на рисунку рисунок 3.37.

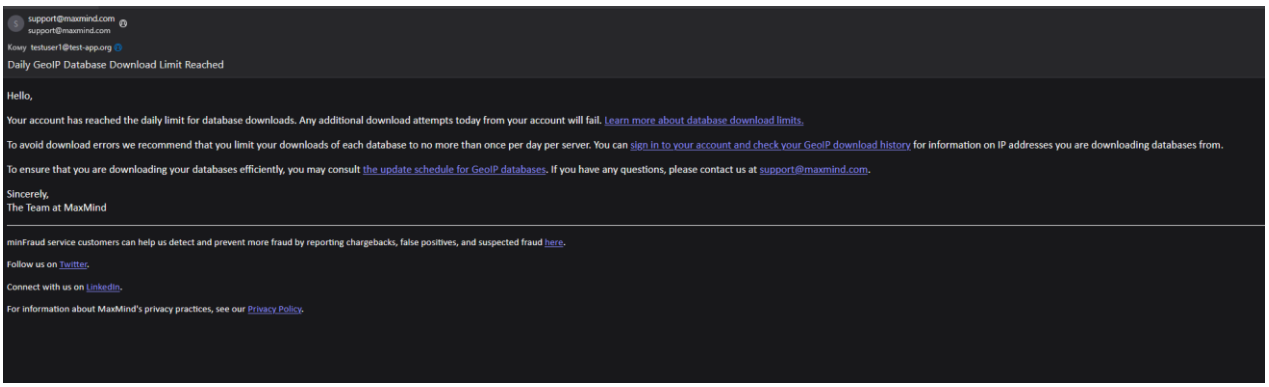

Рисунок 3.37 – Приклад успішної доставки легітимного листа

3.9.3. Відображення події у Grafana Dashboard

Для підтвердження прозорості та аудиту роботи системи всі події класифікації електронної пошти передаються до Loki, після чого відображаються у Grafana Dashboard, рисунок 3.38.

Це дозволяє:

- контролювати джерела фішингових атак;
- стежити за динамікою вхідної пошти;

- виконувати оперативне блокування доменів/адрес через API;
- отримувати алерти про критичні інциденти.

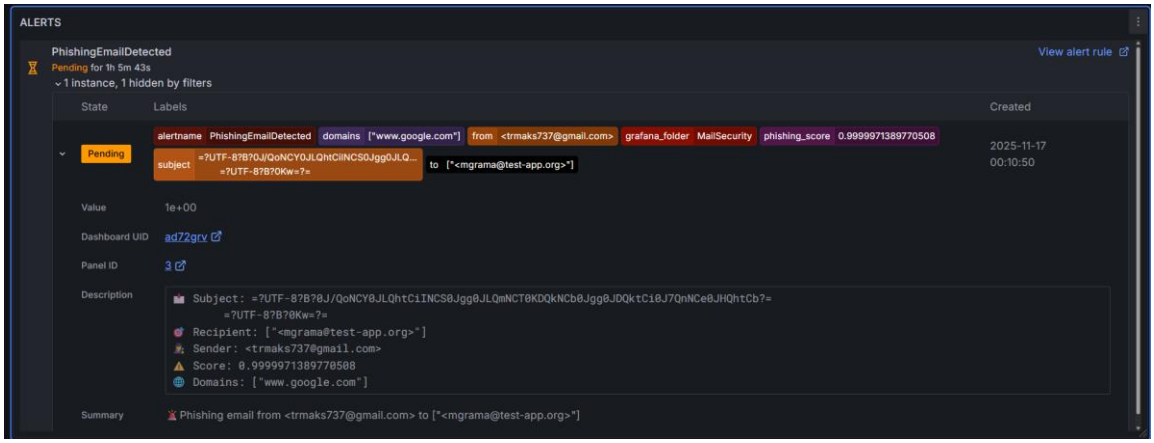

Рисунок 3.38 – Відображення події блокування у Grafana/Loki

3.9.4. Обробка внутрішньої корпоративної пошти

Окремо протестовано обмін листами між внутрішніми поштовими скриньками, який не використовує зовнішні маршрути та DNS Cloudflare. Повідомлення успішно доставляється напряму через Postfix/Dovecot без передачі через зовнішній MX-рівень.

Тест підтвердив, що внутрішня кореспонденція (рис 3.39):

- не блокується помилково політикою AI Milter;
- не залежить від роботи WireGuard тунелю і DMZ, якщо клієнт знаходиться у внутрішній мережі LAN;

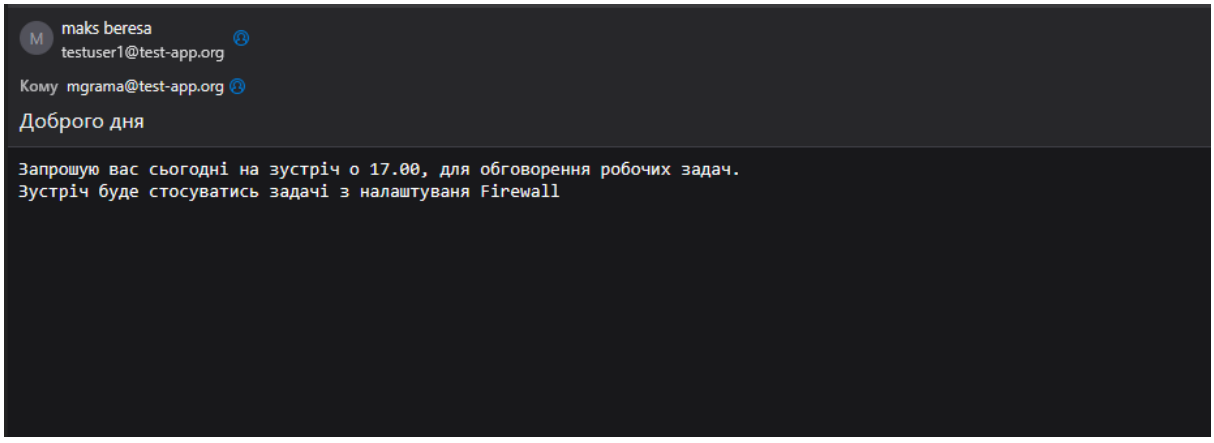

Рисунок 3.39 – Приклад успішної доставки внутрішнього листа

3.9.5. Підсумок тестування

В результаті виконання функціонального тестування ми підтвердили, що система відповідає критеріям, переліченим у таблиці 3.17, а всі ключові елементи архітектури працюють узгоджено та відтворювано. Проведені тести довели здатність системи блокувати фішингові повідомлення на SMTP-рівні, забезпечувати гарантовану доставку легітимних листів, а також формувати повний журнал подій у Loki/Grafana для подальшого аудиту інцидентів. Механізм керування блокуваннями через API та панель Grafana показав високу оперативність реагування, що є критично важливим для роботи підрозділів інформаційної безпеки. Таким чином, реалізоване рішення може вважатися ефективним та таким, що готове до розгортання у корпоративному середовищі з перспективою подальшого масштабування й інтеграції з SIEM/SOAR-платформами.

Таблиця 3.17 – Результат функціонального тестування системи

Критерій	**Результат**
Блокування фішингових листів	Виконується на SMTP-рівні, перешкоджаючи доставці
Доставка легітимної пошти	Гарантовано, без штучних затримок
Аудит інцидентів	Доступний через Loki/Grafana
Керування блокуваннями	Доступне через API та панель Grafana
Внутрішня пошта	Працює коректно у відокремленій мережі

3.10. Інструкція користувача щодо активації корпоративної поштової скриньки та роботи з системою автентифікації

Даний підрозділ містить покрокову інструкцію для користувачів, яким адміністратор створив корпоративну поштову скриньку у домені *test-app.org*. Користувач отримує тимчасовий пароль, виконує первинну активацію облікового запису через систему управління доступом Keycloak, налаштовує двофакторну автентифікацію (2FA) та встановлює власний постійний пароль. Після цього стає можливим використання корпоративної пошти через браузер або будь-який поштовий клієнт (Thunderbird, Outlook, Apple Mail тощо).

1. Отримання даних для первинного входу

Після створення облікового запису адміністратор передає користувачу такі дані:

- Адреса електронної пошти (логін), наприклад: testkeycloak@test-app.org
- Тимчасовий пароль, який необхідно змінити при першому вході
- Посилання на портал автентифікації Keycloak: https://sso.test-app.org/realms/Mail/account

Ці дані є конфіденційними й не повинні передаватися іншим особам.

2. Перший вхід у систему

Необхідно відкрити браузер та перейти за посиланням: https://sso.test-app.org/realms/Mail/account (рис 3.40)

1. У полі Username or email потрібно ввести адресу корпоративної пошти.
2. У полі Password — тимчасовий пароль, отриманий від адміністратора.
3. Натиснути Sign In.

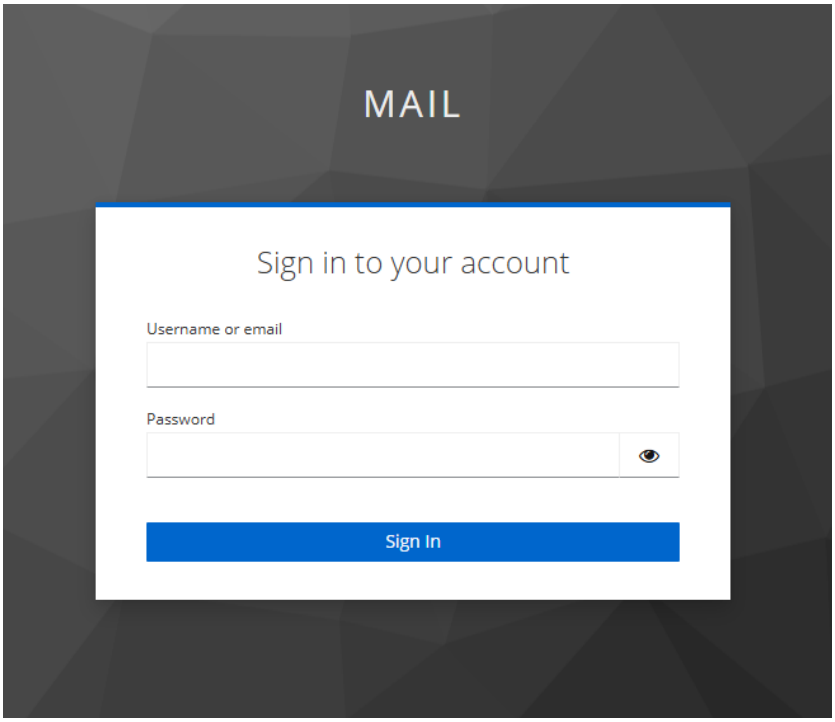

Рисунок 3.40 – Місце для введення облікових данних

Після введення даних система автоматично перенаправить користувача на сторінку налаштування двофакторної автентифікації.

3. Налаштування двофакторної автентифікації (2FA)

Для забезпечення безпеки обліковий запис користувача повинен бути захищений одноразовими кодами (TOTP). Keycloak підтримує такі мобільні застосунки:

- Google Authenticator
- Microsoft Authenticator
- FreeOTP

Після першого входу вам буде буде запропоновано налащтувати двофакторну автентифікацію(рис. 3.41).

Для налаштування вам потрібно

1. Встановити один із мобільних застосунків.
2. Відсканувати QR-код, що згенерований Keycloak.
3. Ввести одноразовий код (6 цифр) із мобільного застосунку.

4. Вказати ім’я пристрою, щоб у майбутньому його можна було ідентифікувати.

5. Натиснути Submit.

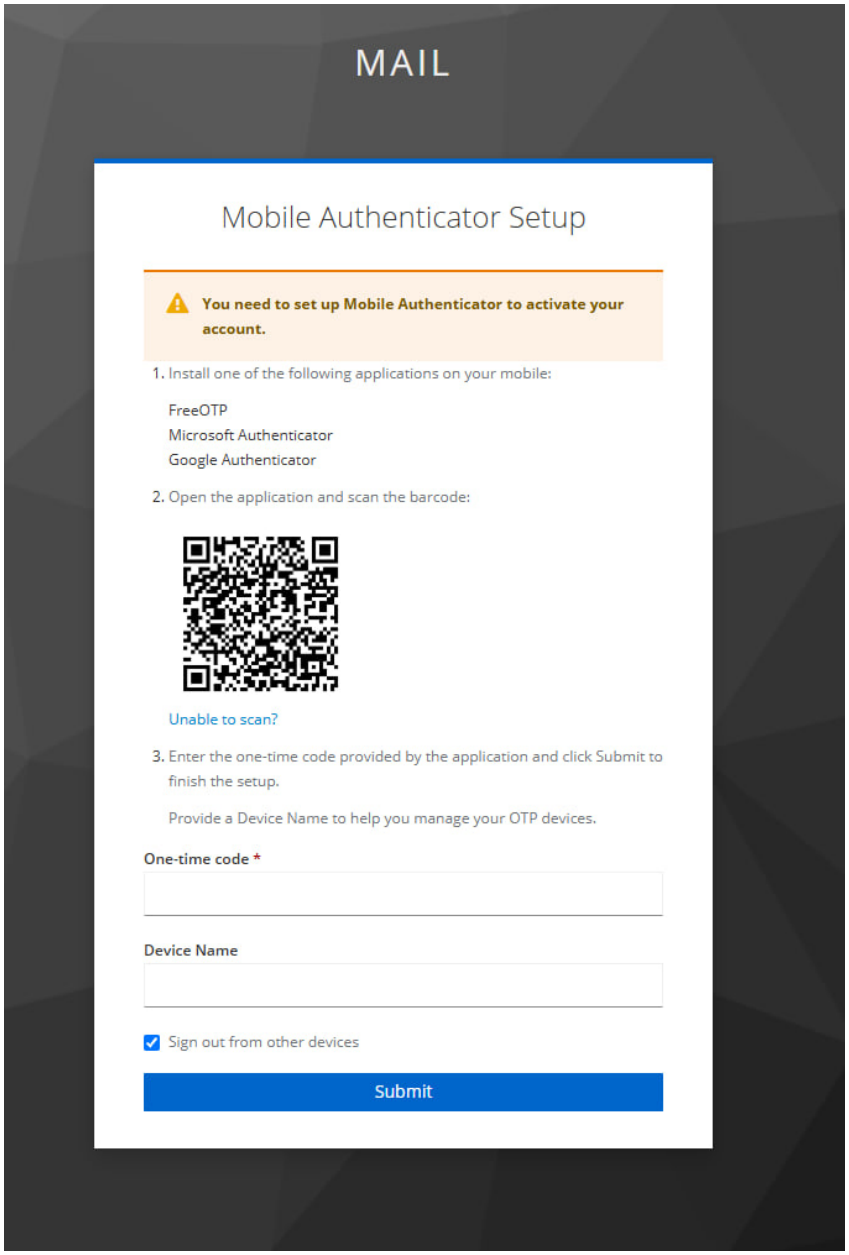

Рисунок 3.41 – Сторінка налаштування мобільного автентифікатора

Успішне налаштування 2FA є обов’язковою умовою активації поштової скриньки.

4. Зміна тимчасового пароля

Після налаштування двофакторної автентифікації користувач автоматично переходить на сторінку зміни пароля, (рис. 3.42).

Необхідно:

1. Ввести новий пароль, що буде використовуватися для доступу до пошти.
2. Повторити його у полі *Confirm password*.
3. Натиснути Submit.

Вимоги до пароля:

- мінімум 12 символів;
- великі та малі літери;
- хоча б одна цифра;
- хоча б один спеціальний символ.

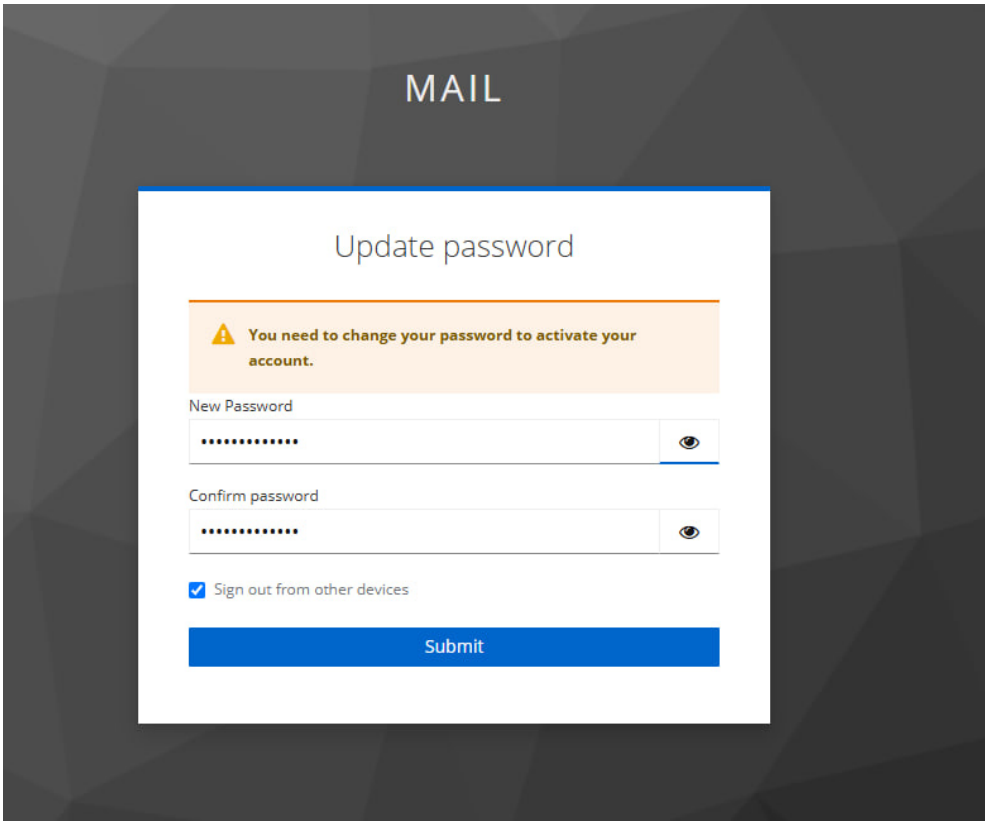

Рисунок 3.42 – Оновлення паролю для входу

Для того щоб зміну паролю потрібно ввести одноразовий код з мобільного застосунку аунтифікації, (рис. 3.43).

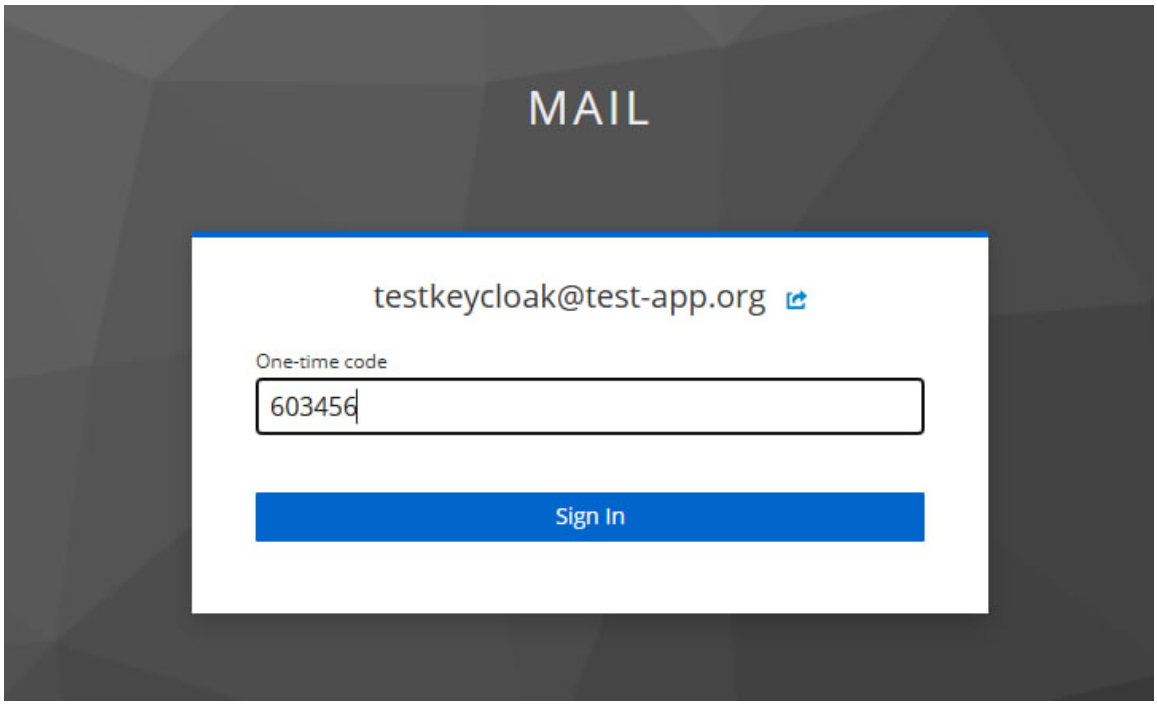

Рисунок 3.43 – Введення одноразового коду

Після збереження нового пароля система виконує автоматичний вихід користувача. Необхідно повторно авторизуватись уже з новим паролем та одноразовим кодом із застосунку.

5. Стандартний процес входу в застосунок пошти після активації акаунту

Після завершення налаштувань аккаунту користувач має завантажити будь який поштовий застосункок

Для роботи рекомендовано використовувати такі поштові клієнти:

- Thunderbird (Windows / Linux / macOS)
- Microsoft Outlook
- Apple Mail (iPhone / Mac)

У встановленому поштовому клієнті введіть email, повна конфігурація поштового серверу має підхопитись автоматично, приклад показано на поштовому клієнті thunderbird, (рис. 3.44).

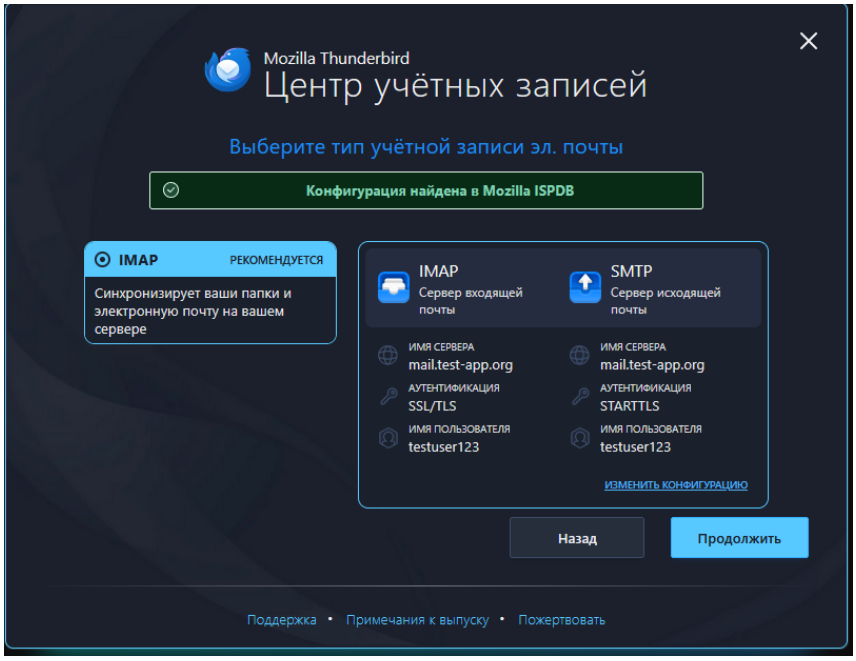

Рисунок 3.44 – Введення одноразового коду

Після того коли ви побачите подібне вікно натискаєте продовжити, і вводите пароль і заходите через нього

6. Можливі проблеми та їх вирішення

1. Неправильний одноразовий код у аунтифікаторі:

Переконайтесь, що час на смартфоні синхронізовано автоматично. Перевірте, що скануєте правильний QR-код.

2. Неможливо увійти після зміни пароля:

Спробуйте авторизуватись через веб-портал ще раз. Переконайтесь, що використовується саме новий пароль.

3. Не має доступу до пошти та порталу для зміни паролю

Переконайтесь що не використовуєте VPN.

4. Втрачено доступ до автентифікатора

Зверніться до адміністратора — потрібне скидання 2FA.

Наведена інструкція забезпечує безпечну та коректну активацію облікового запису електронної пошти користувачем, гарантуючи:

- дотримання політики інформаційної безпеки;
- захист паролем та двофакторною автентифікацією;

- можливість інтеграції з будь-яким поштовим клієнтом;
- зручність подальшої роботи з корпоративною поштою.

3.11. Інструкція адміністратора системи

Підсистема корпоративної електронної пошти з інтегрованим механізмом машинного аналізу вмісту листів потребує чітко визначеної процедури адміністрування. Даний розділ описує порядок створення нових облікових записів, керування поштовими скриньками, блокування підозрілих відправників, роботу з аналітикою та реагування на інциденти інформаційної безпеки.

3.11.1. Створення нового користувача в Active Directory

Варіант 1. Створення через графічний інтерфейс ADUC

1. Відкрити Active Directory Users and Computers.
2. OU для поштових користувачів(рис. 3.45):

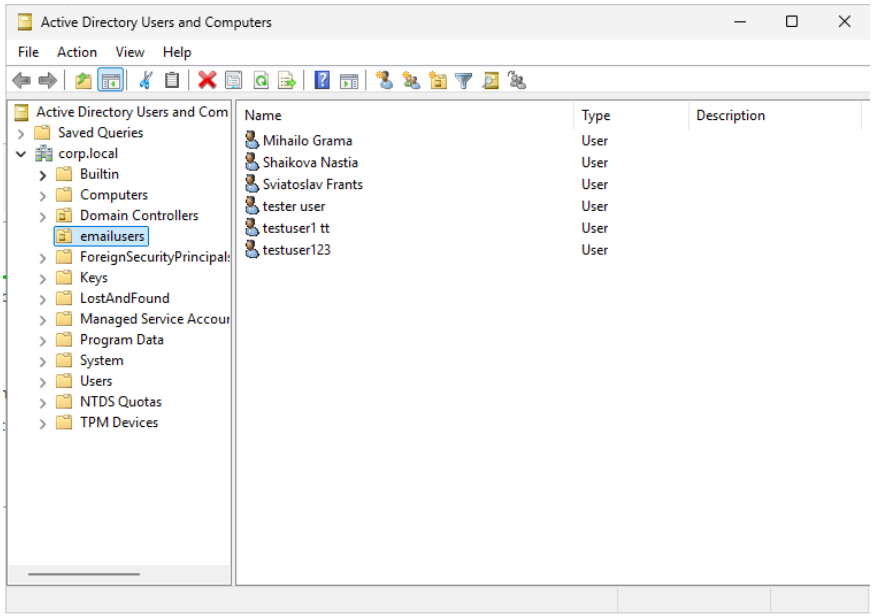

Рисунок 3.45 – Active directory OU emailusers

3. Обрати: New → User
4. Заповнити атрибути котрі наведені в таблиці 3.18

Таблиця 3.18 – Атрибути необхідні для створення облікового запису

Поле	Приклад
First Name	Maksym
Last Name	Bereza
Full Name	Maksym Bereza
User logon name (UPN)	mbereza@corp.local
Email	mbereza@test-app.org

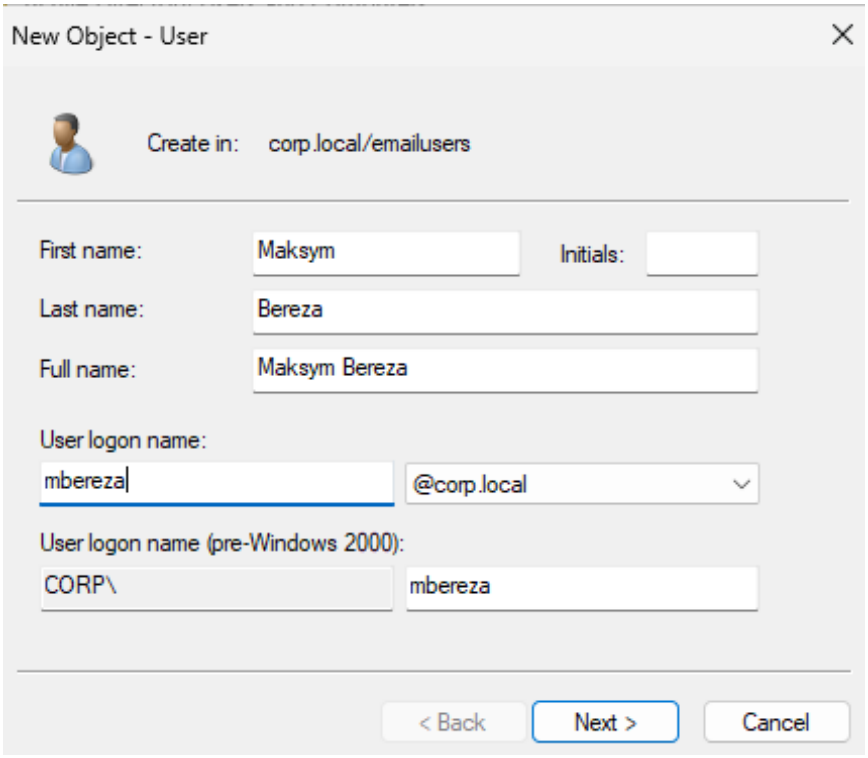

Рисунок 3.46 – Заповнення атрибутів для аккаунту

5. Задати тимчасовий пароль, який користувач змінить у Keycloak.
6. Увімкнути опцію: User must change password at next logon (рис. 3.47).

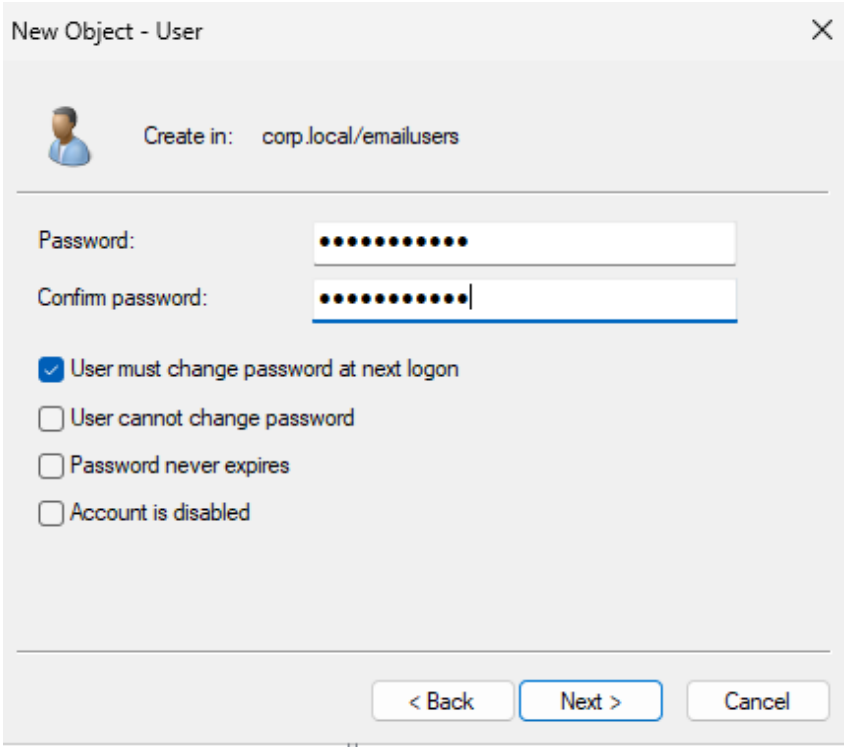

Рисунок 3.47 – Заповнення паролю з опцією його зміни при наступному вході

Варіант 2. Створення через користувача через PowerShell.

Відкрийте powershell у режимі адміністратора та виконайте скрипт, на рисунку 3.48 можна побачити успішне виконання скрипта:

```
New-ADUser `
 -Name "Ivan Petrenko" `
 -GivenName "Ivan" `
 -Surname "Petrenko" `
 -SamAccountName "ipetrenko" `
 -UserPrincipalName "ipetrenko@corp.local" `
 -EmailAddress "ipetrenko@test-app.org" `
 -Path "OU=EmailUsers,DC=corp,DC=local" `
 -AccountPassword (ConvertTo-SecureString "TempPass#2025" -AsPlainText -Force) `
 -Enabled $true
```

```
PS C:\Users\Administrator> New-ADUser `
>>  -Name "Ivan Petrenko" `
>>  -GivenName "Ivan" `
>>  -Surname "Petrenko" `
>>  -SamAccountName "ipetrenko" `
>>  -UserPrincipalName "ipetrenko@corp.local" `
>>  -EmailAddress "ipetrenko@test-app.org" `
>>  -Path "OU=EmailUsers,DC=corp,DC=local" `
>>  -AccountPassword (ConvertTo-SecureString "TempPass#2025" -AsPlainText -Force) `
>>  -Enabled $true
```

Рисунок 3.48 – Виконання скрипта у powershel

3.11.2. Автоматичне створення поштової скриньки

Скрипт на сервері Dovecot створює Maildir при першій доставці листа.

Адміністратор може виконати примусово:

sudo doveadm mailbox create -u user@test-app.org INBOX

sudo doveadm quota recalc -u user@test-app.org

Або можливо також примусово виконати скрипт командою:

sudo provision_mailboxes.sh

Шлях до Maildir:

/home/vmail/test-app.org/user/

За допомогою команди можна перевірити наявність всіх створених почтових директорій для користувачів, також можна виконати її з фільтром grep щоб переглянути наявкність окремої поштової директорії:

sudo ls -l /home/vmail/test-app.org/user/

3.11.3. Блокування доменів та email-адрес

A. Блокування через Grafana → API → AI Milter

Доступно у Dashboard: Email Threat Control, (рис. 3.49).

Достатньо обрати:

- Action: BLOCK
- Object type: Email / Domain
- Value: spam@malicious.com

Після того як ми обрали та ввели необхідні данні ми натискаємо на кнопку

підтвердження дії, після цього відбудеться блокування

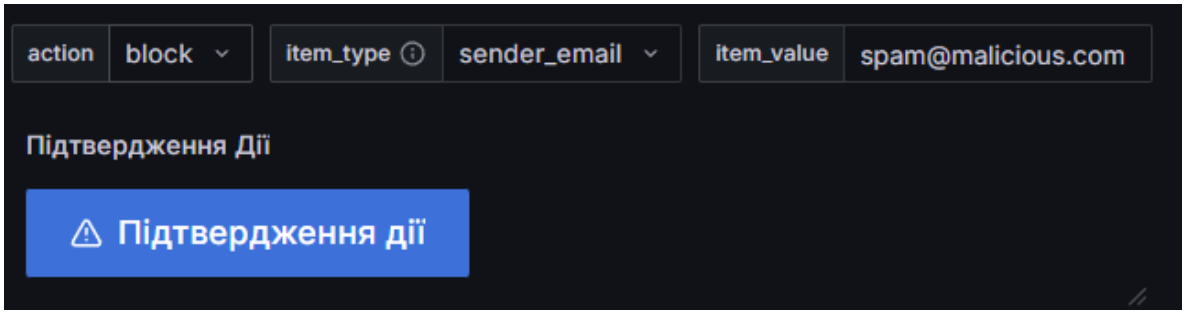

Рисунок 3.49 – Панель вибору блокування з кнопрою підтвердження дії

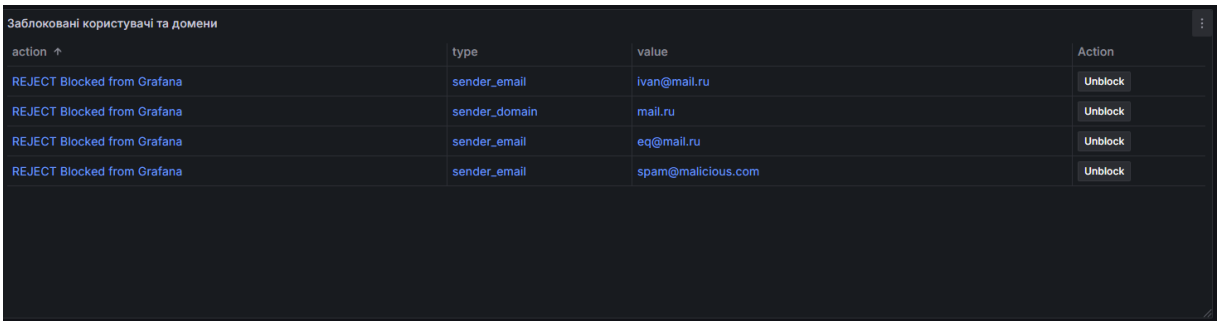

Рисунок 3.50 – Таблиця з заблокованими email/domains

Для зняття блокування достатньо натиснути кнопку в стовпчику action в таблиці котра показана на рисунку 3.50, або в першому блоці в якому ми проводили блокування змінити action у випадаючому меню на unblock, і натиснути на кнопку підтвердження дії

3.11.4. Робота з Firewall

Для входу на firewall достатньо прейти за ip адресою https://192.168.1.1/ , відкриється вікно для введення облікових данних для входу, (рис. 3.51).

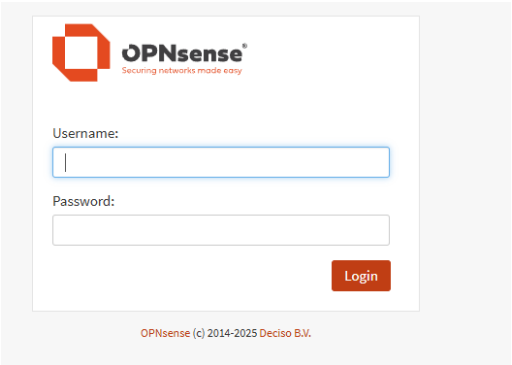

Рисунок 3.51 – Основна сторінка аунтифікації на OPNsense firewall

Для перегляду блокувань за правилами L3 рівня на Firewall достатньо перейти на вкладку Live View, (рис. 3.52).

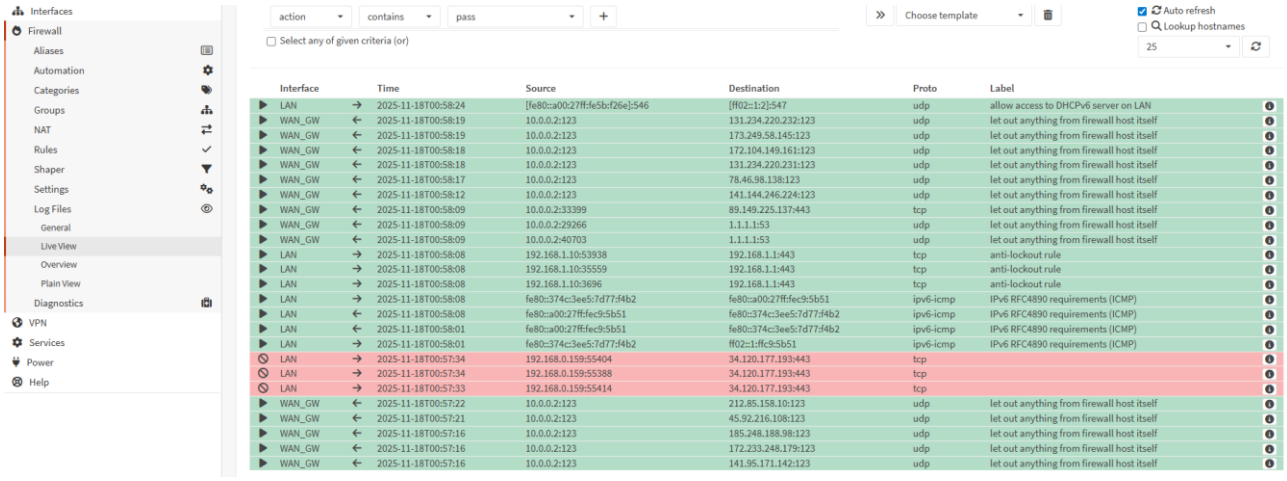

Рисунок 3.52 – Вкладка для перегляду трафіку на firewall

Тут ми можемо побачити інтерфейс ip адресу яка встановлює з'єднання, ip до якої йде з'єднання, а також label який показує назву правила котре дозволяє або блокує з’єднання.

Для налаштування правил рівня L3 достатньо перейти у розділ Firewall, далі у розділ Rules та обрати цікавлячий нас темплейт з правилами (рис. 3.53), в цьому розділі можна як створювати так і редагувати існуючі правила, в разі потреби відкриття доступів чи їх обмеження.

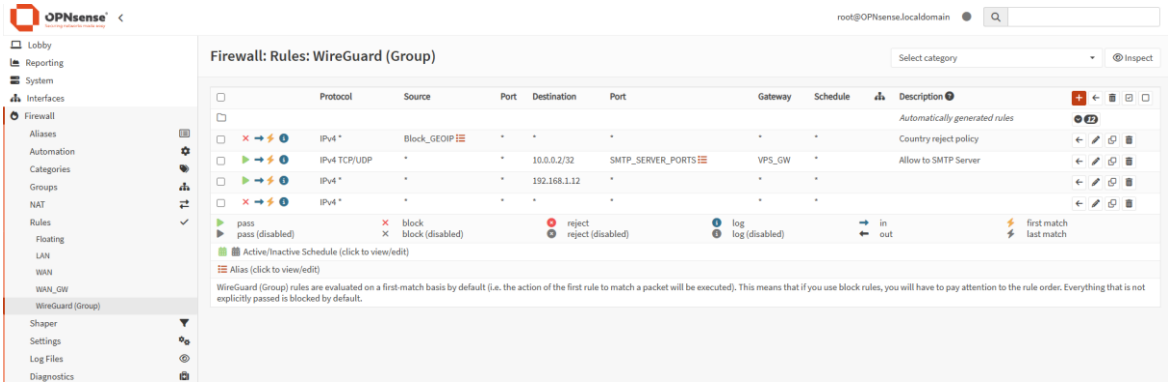

Рисунок 3.53 – Вкладка для перегляду правил на firewall

Для налаштування списку блокувань за країнами достатньо перейти до вкладки Firewall, а потім обрати Aliasses, для налаштування групи країн що

блокуюьбся достатньо натиснути на аліас Block_GEOIP і відкриється увесь список країн котрі потрапляють під блокування, (рис. 3.54).

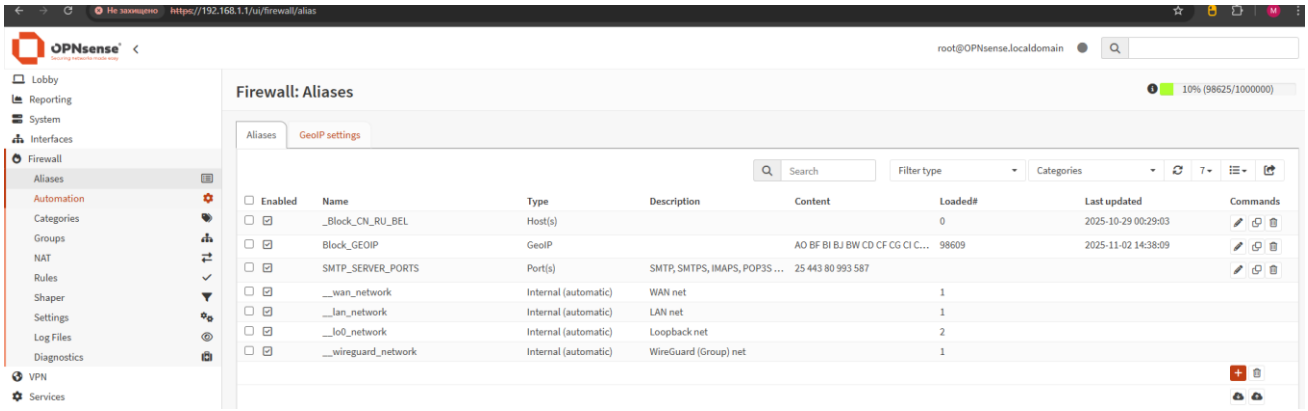

Рисунок 3.54 – Вкладка для перегляду аліасів на firewall

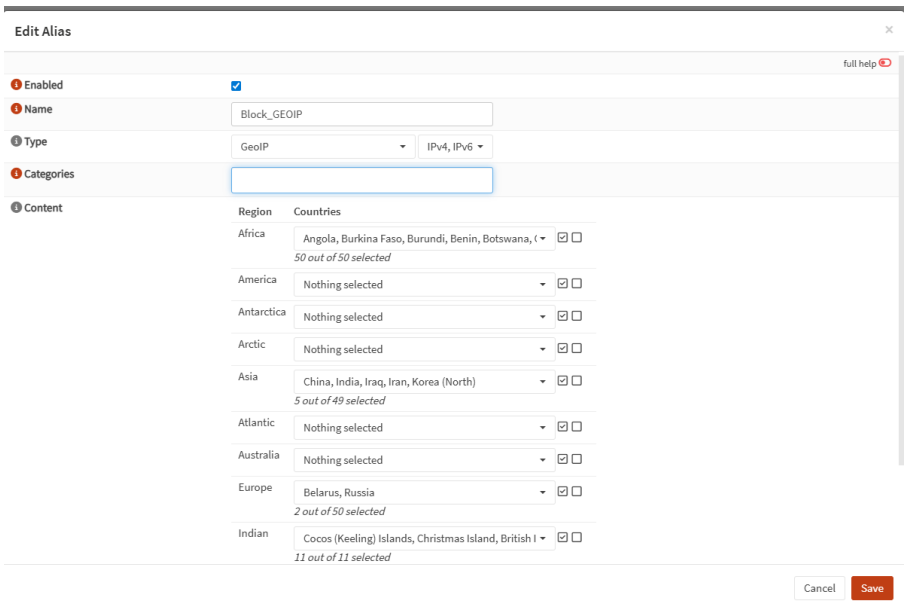

Рисунок 3.55 – Вигляд аліасу Block_GEOIP

Для перевірки доступності VPN тунелю Wireguard достатньо перейти в пункт VPN→WireGuard→ Status, і можна побачити статус нашого тунелю, (рис. 3.56), якщо status зелений і handshake не більше за 120 секунд значить тунель працює справно.

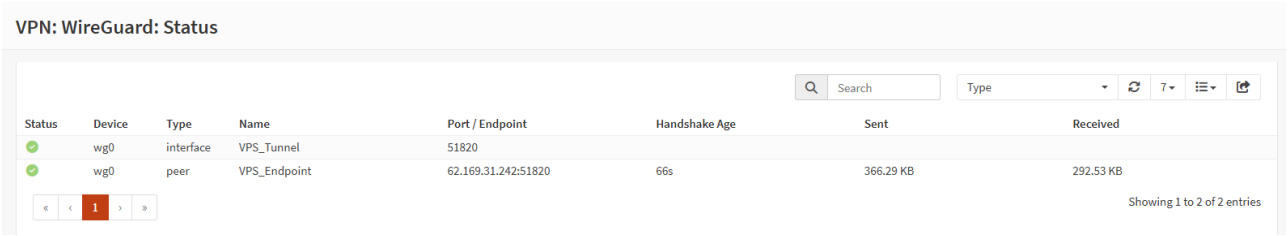

Рисунок 3.56 – Статус Vpn тунелю

Також для перевірки справності роботи Firewall та мережевої конфігурації, ми можемо зайти в System→Gateways→ Configuration і побачити справність роботи наших шлюзів, а також в разі необхідності змінити їх конфігурацію, (рис. 3.57).

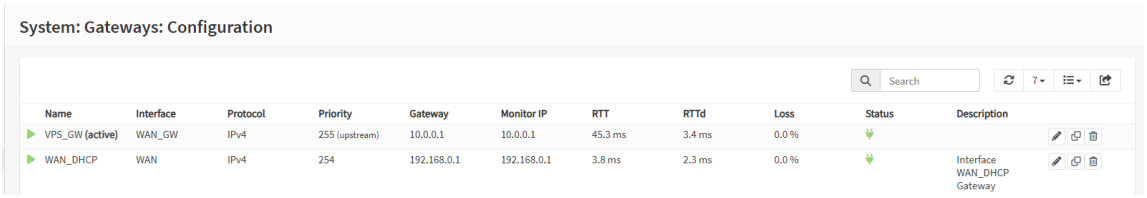

Рисунок 3.57 – Статус Firewall Gateways

Налаштування фільтрації за рівнем L7 відбувається у Services → Intrusion Detection → Administration (рис. 3.58), в даній панелі ми бачемо пункти IPS mode та Enabled, для функціонування фільтрації на рівні L7 вони мають бути постійно увімкнені, також в пункті Interfaces ми бачимо випадаюче меню котре обирає на якому інтерфейсі буде застосовувати філтрація, можна обрати один або декілька інтерфейсів, в пункті Download (рис. 3.59) ми можемо обрати які списки правил ми скачаємо для використання у політиці.

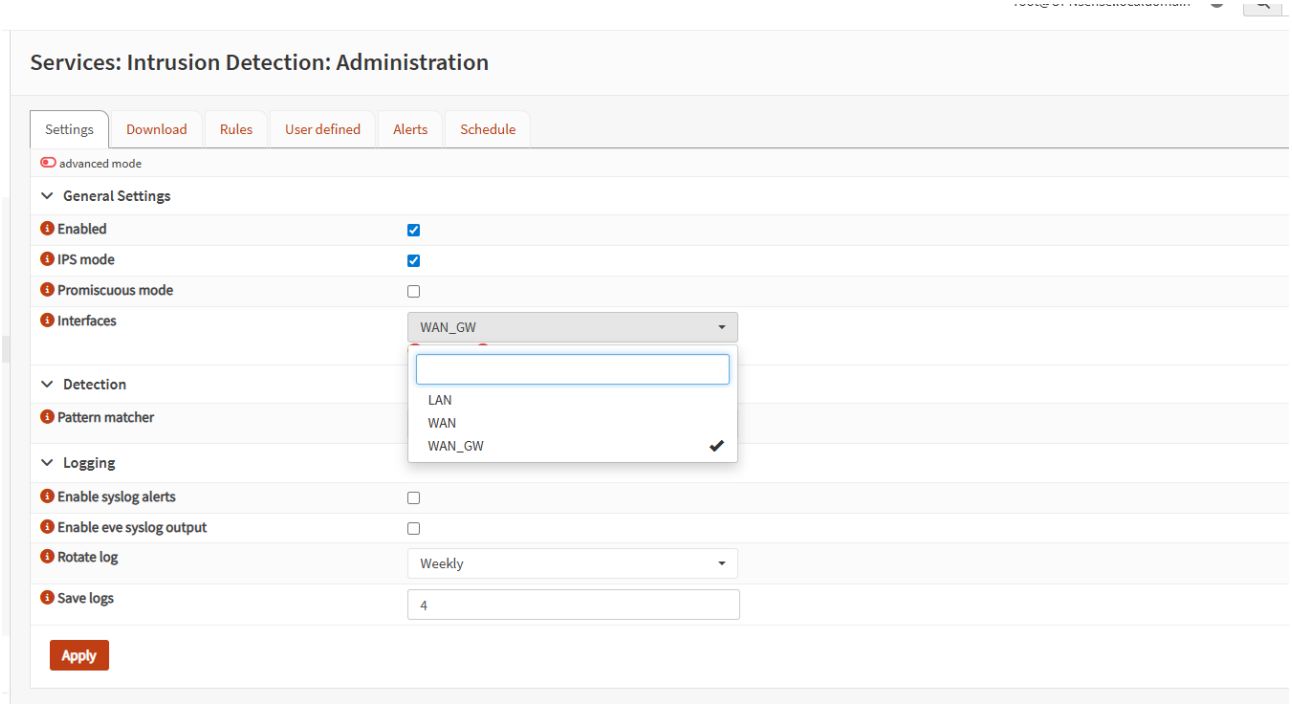

Рисунок 3.58 – Налаштування IPS сервісу на firewall

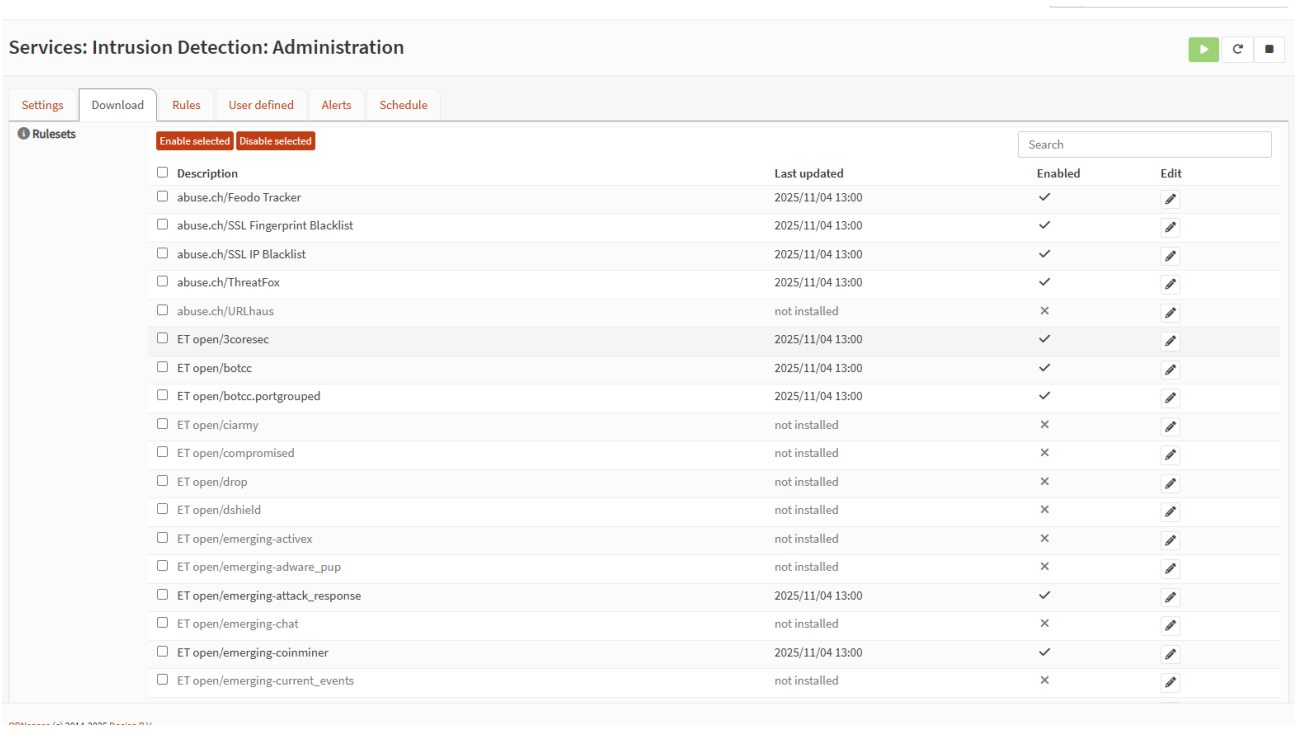

Рисунок 3.59 – Налаштування списку встановлених правил

Для того щоб правила почали працювати створена політика у пункті *Services → Intrusion Detection → Policy*, (рис. 3.60) та вибрано *Action Drop* в

Rulesets було обрано наші правила які ми включили та завантажили в минулому пункті

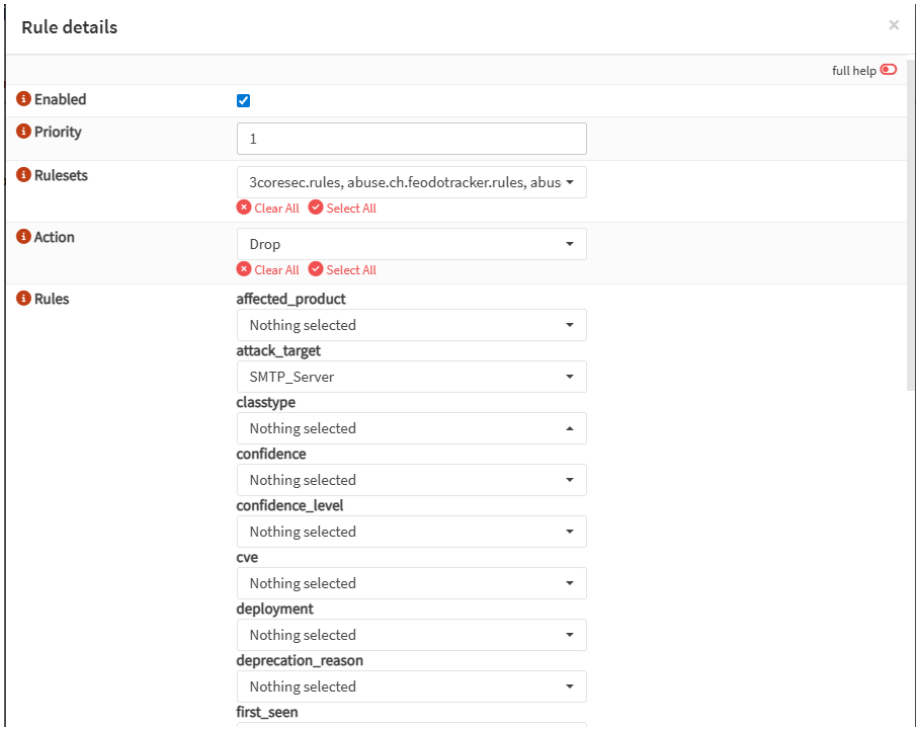

Рисунок 3.60 – Налаштування політики IPS

Щоб побачити логування з блокувань за правилами IPS достатньо перейти в *Services → Intrusion Detection → Log File* та побачити блокуваня, або в логах трафіку Live View, (рис. 3.61).

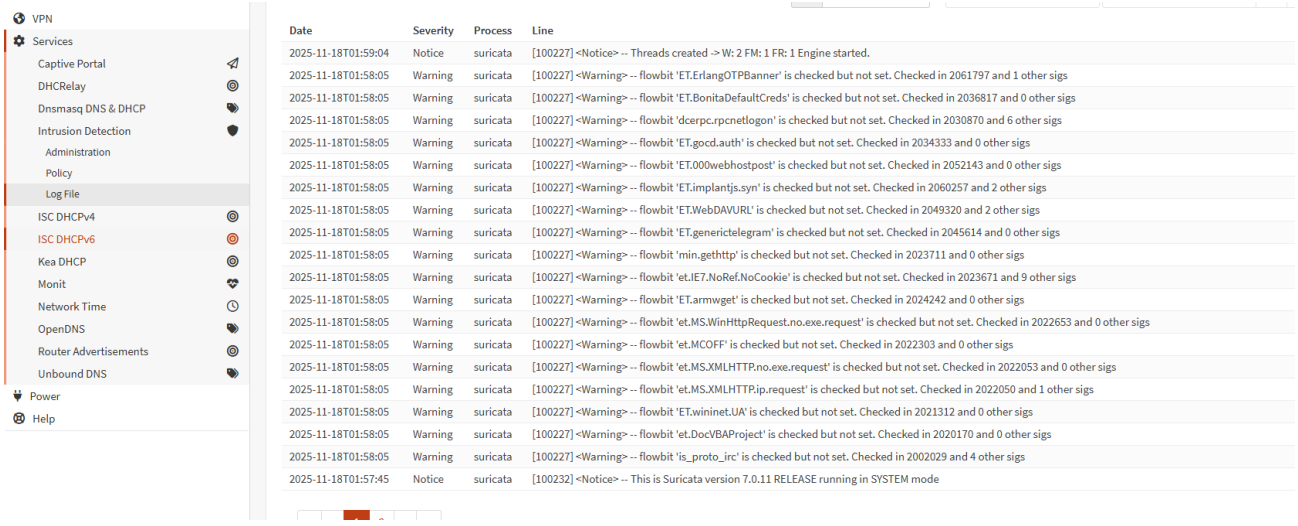

Рисунок 3.61 – Логування блокувань IPS

3.11.5. Робота з keyCloack

Для входу до платформи keycloak достатньо перейти за посиланням https://sso.test-app.org/. Щоб працювати з потрібним realm потрібно у випадаючому меню обрати Mail (рис. 3.62), для перегляду всіх синхронізованих користувачів через LDAP, достатньо перейти на пункт users та ввести в пошуковому меню *, (рис. 3.63).

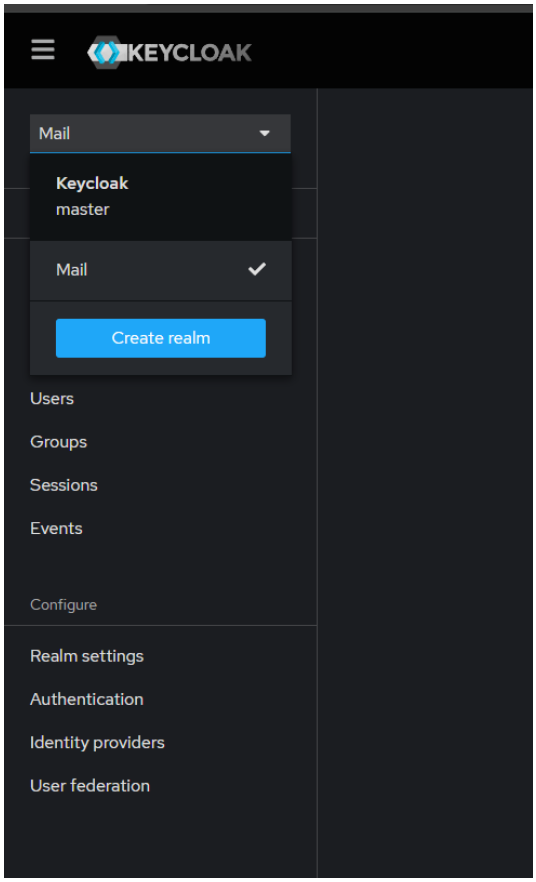

Рисунок 3.62 – Вибір realm у keycloack

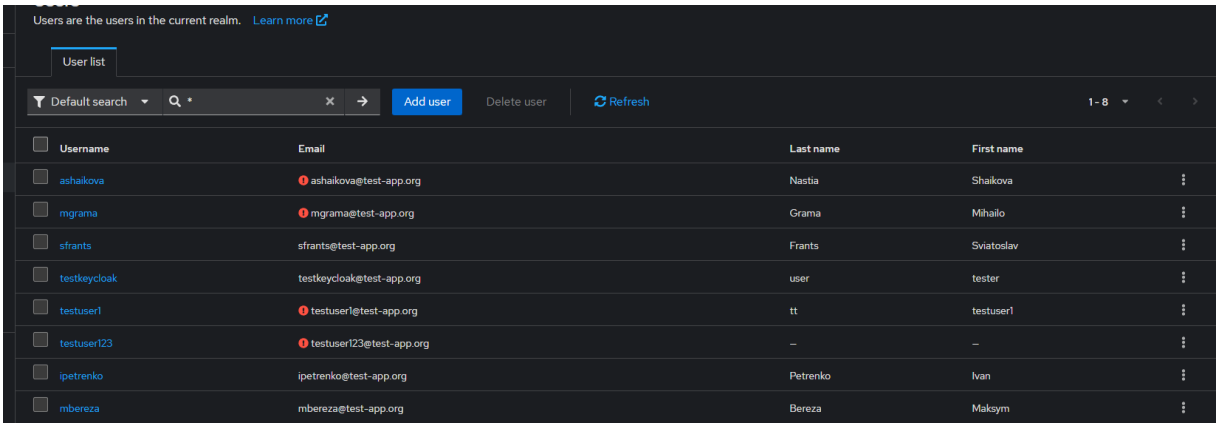

Рисунок 3.63 – Перегляд користувачів у keycloack

Для того щоб скинути користувачу пароль або двохфакторку потрібно клікнути на певного коритсувача, та перейти у вікні що відкриється в пункт Credentials (рис. 3.64), в цьому вікні натиснути на кнопку Credential Reset, після цього відкриється випадаюче меню в якому можна буде обрати Configure OTP(Скинути аунтифікатор), або Update Password(Скинути пароль), що зображено на рисунку 3.65.

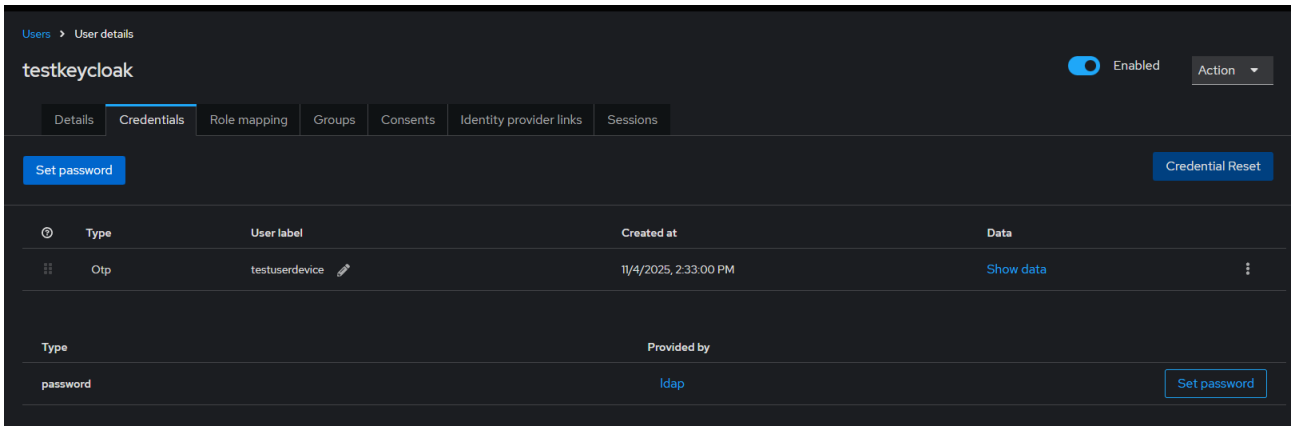

Рисунок 3.64 – Налаштування користувача у keycloack

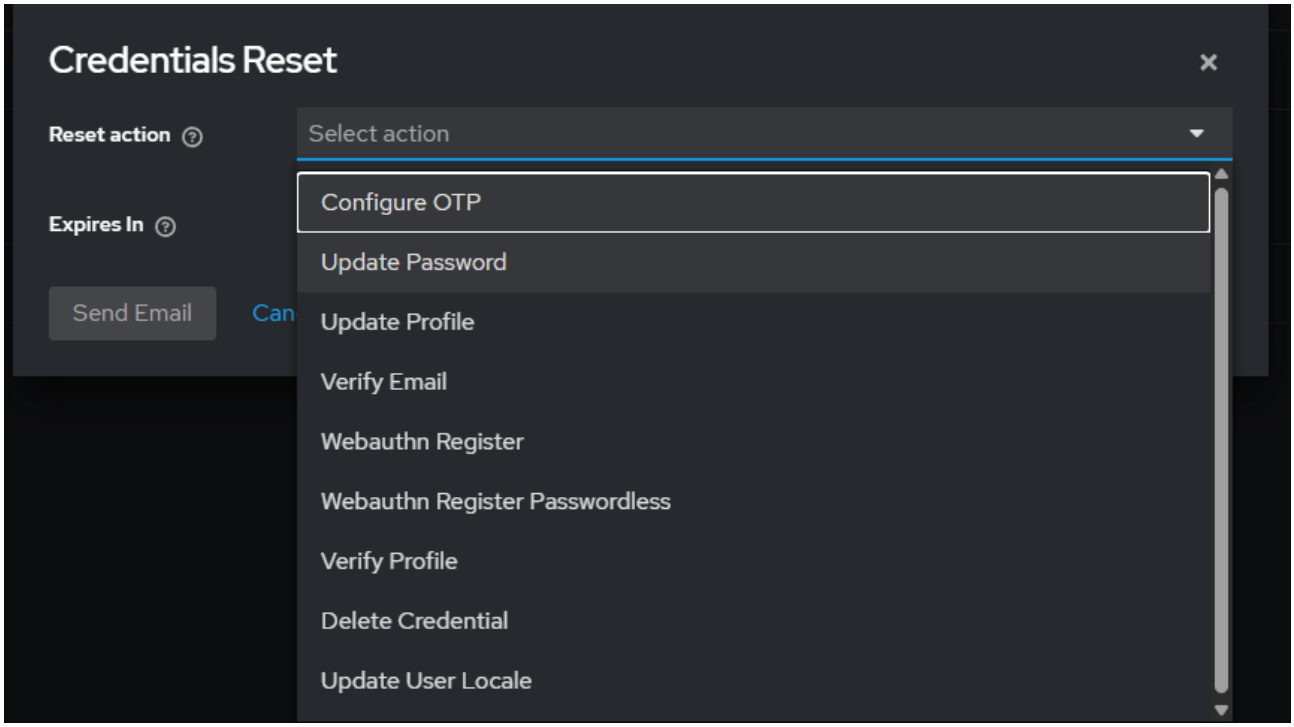

Рисунок 3.65 – Кнопка Credentials Reset для користувача

3.11.6. Playbook реагування на фішингові інциденти (SOC Runbook)

Playbook реагування на інциденти другої лінії SOC розроблено з метою встановлення формалізованої моделі обробки подій, пов'язаних із фішинговими електронними листами, спробами компрометації користувацьких поштових скриньок, виявленням шкідливих вкладень та несанкціонованих автентифікацій. Застосування цього Playbook забезпечує єдиний алгоритм прийняття рішень, мінімізує людський фактор під час ухвалення критично важливих дій, а також уніфікує порядок документування інцидентів. Опис даного Playbook наведено у таблиці 3.19.

Таблиця 3.19 – Сценарії оперативного реагування на інциденти

Сценарій інциденту	**Дії адміністратора**
Підтверджений фішинг (відправник/домен/URL)	1) Перевірити журнали Grafana/Loki 2) Заблокувати адресу/домен через Grafana або API 3) Сповістити потенційних отримувачів 4) За потреби виконати reset password користувача
False Positive (помилкове спрацювання)	1) Перевірити контекст події 2) Розблокувати у Grafana/API 3) Додати в allowlist (за погодженням з політикою безпеки)
Можлива компрометація акаунта	1) Примусовий reset password (Keycloak/AD) 2) Завершення активних сесій IMAP/SMTP 3) Перевірка поштової активності та пересилань
Масова фішинг-атака проти домену	1) Увімкнути посилене GeoIP-блокування 2) Примусово підвищити поріг phishing_score 3) Активувати IDS/IPS-правила для SMTP/HTTPS

3.11.7. Типові помилки та методи їх усунення

Також для зручності роботи з системою було окремо винесено помилки які можуть трапитись під час роботи системи.

Таблиця 3.20 – Типові помилки у роботі системи

Помилка	Ймовірна причина	Спосіб усунення
550 5.7.1 Access denied	Відправник заблокований Postfix	Перевірити sender_access / API unblock
Користувач не може увійти в пошту	Пароль застарів / не синхронізований з Keycloak	reset password у Keycloak/AD
AI Milter не класифікує листи	Зупинено сервіс AI API	restart service + перевірка socket
TLS error при SMTP/IMAP	SSL сертифікат недійсний	Перегенерація Let’s Encrypt/Cloudflare Full Strict
Grafana не показує дані	Promtail/Loki не працюють	Перевірити systemctl status promtail/loki

3.12. Висновок до третього розділу

У третьому розділі було здійснено комплексне проєктування та розгортання інформаційної системи корпоративної електронної пошти з підтримкою машинного аналізу листів, орієнтованої на виявлення та блокування фішингових повідомлень. Сформована інфраструктура охоплює мережеву, серверну, поштову та аналітичну підсистеми, а також модуль управління обліковими записами та багатофакторною автентифікацією. Здійснена робота доводить можливість побудови високозахищеної поштової системи з використанням відкритих технологій, що забезпечують масштабованість,

адаптивність та контроль безпеки на кожному етапі обробки електронних повідомлень.

Першим етапом стала розробка загальної концепції функціонування корпоративної пошти, де визначено вимоги до безпеки, надійності, стійкості до атак та підтримки централізованого аналізу трафіку. Проведено реєстрацію домену та налаштовано DNS-записи (*A, MX, SPF, DKIM, DMARC*), що закладає фундамент для захищеної маршрутизації поштових потоків. Коректна конфігурація DNS стала важливим елементом протидії підробленню доменів (email spoofing), що є одним із поширених методів фішингових кампаній.

Подальша частина розділу присвячена побудові мережевої інфраструктури, яка виконує роль комунікаційного середовища для поштового трафіку. На базі OPNsense реалізовано сегментування мережі, налаштовано маршрутизацію, створено VPN-тунель WireGuard між perimeter-сервером та внутрішнім сегментом локальної мережі. Завдяки цьому поштові сервери отримують захищений канал передачі даних, а вхідний/вихідний трафік проходить через огляд IDS/IPS. Для поглибленого аналізу та блокування загроз активовано Suricata, включено правила для виявлення шкідливого коду, бот-мереж, C2-комунікацій та атак на SMTP/IMAP-протоколи. Додатково реалізоване GeoIP-блокування, що обмежує спроби встановлення з’єднання з підозрілих регіонів.

Наступним етапом стало розгортання серверної інфраструктури на базі Oracle VirtualBox із використанням Windows Server 2025 та Ubuntu Server 24.04 LTS. Windows-сервер виконує роль домен-контролера Active Directory, забезпечуючи зберігання службових облікових записів. Ubuntu-сервер став основою для розгортання поштової системи.

У підрозділі налаштування поштової підсистеми детально конфігуровано компоненти які перелічені в пункті 3.1 «Загальна концепція системи».

Важливим елементом став механізм автоматичного створення поштових скриньок через Bash-скрипт із перевіркою користувачів Active Directory. Це

зменшує навантаження на адміністратора й усуває типові помилки ручної конфігурації поштових акаунтів.

Надалі у систему інтегровано модуль машинного аналізу AI Milter, що реалізує фільтрацію поштового контенту на основі нейронної мережі. Було сформовано процес тренування моделі на датасеті з маркованими фішинговими та безпечними листами, розглянуто техніку оцінювання за метриками accuracy, precision, recall. Модель продемонструвала високі результати при генералізації на тестових даних, що дозволило інтегрувати її в реальний потік листів. AI API виконує класифікацію тексту повідомлень та заголовків, інтегруючись із SMTP через Milter-протокол, і у випадку виявлення загрози блокує доставку листа на ранньому етапі маршрутизації.

Розділ також включає впровадження ключового елементу безпеки — підсистеми керування обліковими записами Keycloak, що забезпечує:

- централізовану автентифікацію;
- самостійну зміну тимчасових паролів;
- двофакторну перевірку (2FA/TOTP);
- інтеграцію з Active Directory через LDAPS;
- аудит подій входу користувачів.

Завдяки цьому усувається використання слабких паролів, що є однією з найчастіших причин успішного фішингу.

Окрему увагу приділено аналітиці та моніторингу. На базі Grafana та Loki розроблено централізований дашборд, який дозволяє:

- переглядати журнали блокувань і доставок листів;
- аналізувати категорії загроз;
- керувати списками заблокованих адресатів;
- сповіщувати служби безпеки про інциденти.

Фінальним етапом стало функціональне тестування системи, що підтвердило:

- успішне блокування фішингових листів з помилкою 550 5.7.1;

- гарантовану доставку легітимної кореспонденції;
- коректну обробку внутрішньої корпоративної пошти;
- відображення подій у Grafana з можливістю подальшого аналізу;
- працездатність API для керування блокуваннями.

Таким чином, проведене розгортання підтвердило досягнення цілей третього розділу. Створена інфраструктура являє собою комплексне рішення, що поєднує сучасні методи кіберзахисту, машинне навчання, мережеві технології та механізми безпечної автентифікації користувачів. Запропонована система може бути масштабована та використана в телекомунікаційних компаніях, державних установах, фінансових організаціях та інших середовищах, для яких електронна пошта є критично важливим каналом обміну інформацією.

Проведена робота дозволяє стверджувати, що сформована інфраструктура відповідає вимогам сучасної корпоративної безпеки, а інтеграція штучного інтелекту у процеси обробки листів значно підвищує ефективність протидії фішинговим кампаніям та кібератакам.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було здійснено комплексне дослідження, проєктування та практичну реалізацію інформаційної системи аналізу корпоративних поштових повідомлень АТ «Укртелеком» із використанням методів машинного навчання. Розроблене рішення спрямоване на підвищення рівня кіберзахисту організації шляхом мінімізації ризиків успішних фішингових атак, компрометації облікових записів та несанкціонованого доступу до внутрішньої інформації товариства. Актуальність даної розробки зумовлена тим, що електронна пошта залишається одним із ключових каналів ділової взаємодії у сфері телекомунікацій, а сучасні фішингові кампанії характеризуються високим рівнем соціальної інженерії та використанням генеративного штучного інтелекту, що значно ускладнює їх виявлення традиційними засобами.

Проведений аналіз стану обробки поштових повідомлень засвідчив, що, попри використання в АТ «Укртелеком» окремих технологій безпеки, включаючи Microsoft Exchange Online Protection та політики антиспаму, існували прогалини, зокрема відсутність централізованої перевірки листів, що обробляються через зовнішні поштові сервіси, насамперед Gmail, яким вимушено користуються окремі підрозділи компанії у взаємодії з клієнтами. Це призводило до значних витрат часу на ручну перевірку електронних листів офіцерами безпеки, створювало ризик пропуску загроз та збільшувало площу потенційної атаки. Таким чином, застосування ідентифікації та блокування фішингових повідомлень стало критично необхідним елементом стратегії кіберзахисту.

У роботі було обґрунтовано вибір архітектури системи, що поєднує мережеву сегментацію, захищені тунелі WireGuard, міжмережевий екран OPNsense з IDS/IPS Suricata та поштову інфраструктуру на базі Postfix і Dovecot з використанням шифрування TLS та протоколу LMTP. Особливе значення приділено реалізації машинного аналізу контенту листів за допомогою моделі на

основі DistilBERT, здатної здійснювати семантичну оцінку текстів електронних повідомлень та ідентифікувати ознаки фішингу, маніпулятивні конструкції та підозрілі патерни, зокрема створені із залученням штучного інтелекту. Проведене навчання нейронної мережі на змішаному наборі корпоративних і відкритих даних забезпечило точність класифікації на рівні 94–97%, що підтверджує ефективність застосованого підходу.

Результатом інтеграції модулів AI Milter та AI API стало створення наскрізного потоку обробки пошти, за якого кожне вхідне або вихідне повідомлення проходить автоматизований аналіз, а у разі підозри на фішинг отримує рішення reject або quarantine ще на рівні SMTP. Логи обробки, включно з оцінками phishing_score, передаються до Loki, а подальший моніторинг і реагування здійснюються через аналітичний дашборд Grafana, що надає фахівцям центру безпеки повну видимість поштового трафіку, включаючи розподіл доменів-відправників, частоту фішингових спроб та історію блокувань. Важливою складовою проєкту стало створення інструментів оперативного реагування: зокрема, можливість блокування доменів або електронних адрес безпосередньо з Grafana через API з автоматичним оновленням таблиць доступу Postfix, що істотно скорочує час реагування аналітиків SOC.

З метою підвищення контролю над життєвим циклом облікових записів поштових користувачів у системі було впроваджено мікросервіс автентифікації Keycloak з інтеграцією Active Directory через LDAPS. Це дозволило організувати безпечну зміну тимчасових паролів, застосування політик складності та примусову активацію двофакторної автентифікації з використанням TOTP-додатків. Така інтеграція суттєво зміцнила загальну модель захисту акаунтів, зменшивши ймовірність компрометації доступів до корпоративної пошти.

Функціональні випробування системи продемонстрували здатність рішення ефективно блокувати шкідливі повідомлення, у тому числі листи з правильними цифровими підписами, але з вмістом, сформованим методами соціальної інженерії. Система також показала стабільну роботу при доставці

внутрішньої пошти компанії, що свідчить про її адаптивність до корпоративного документообігу.

Розроблена система може використовуватися як автономне рішення або як модуль у складі ширшої корпоративної стратегії SOC-рівня. Практичне впровадження даного підходу підвищує загальний рівень інформаційної безпеки АТ «Укртелеком», скорочує операційні витрати на ручну перевірку листів, знижує ризики компрометації критичних сервісів і сприяє дотриманню принципів Zero Trust та рекомендацій NIST щодо захисту електронної пошти.

У перспективі система може бути доповнена функціями поведінкового аналізу вкладень, інтеграцією sandbox-середовищ перевірки шкідливого коду, використанням ансамблевих моделей ML, розширенням мовної підтримки та підключенням джерел threat intelligence, що дозволить адаптувати її до динаміки кіберзагроз. Таким чином, отримані результати підтверджують ефективність застосованих технологій і доводять, що створення інтегрованої інтелектуальної системи аналізу корпоративних поштових повідомлень є доцільним, необхідним та стратегічно важливим кроком для забезпечення кіберстійкості АТ «Укртелеком» в умовах сучасного інформаційного середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Історія акціонерного товариства «УКРТЕЛЕКОМ» URL: https://ukrtelecom.ua/about/company_history (Дата звернення: 15.10.2025).
2. ДСТУ ISO/IEC/IEEE 29119-1:2017. Інженерія систем і програмних засобів. Тестування програмних засобів.
3. ДСТУ ISO/IEC 29155-1:2015. Розроблення систем і програмного забезпечення. Платформи для тестування проєктів з розроблення інформаційних систем. Частина 1.
4. ДСТУ ISO/IEC 12207:2016. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення.
5. ДСТУ ISO/IEC 27001:2015 (ISO/IEC 27001:2013, IDT). Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою.
6. RFC 5321 — Simple Mail Transfer Protocol (SMTP). URL: https://www.rfc-editor.org/rfc/rfc5321 (Дата звернення: 15.10.2025).
7. RFC 3501 — Internet Message Access Protocol (IMAP). URL: https://www.rfc-editor.org/rfc/rfc3501 (Дата звернення: 15.10.2025).
8. RFC 8314 — Use of TLS for Email Submission and IMAP/POP3 Access. URL: https://www.rfc-editor.org/rfc/rfc8314 (Дата звернення: 22.10.2025).
9. RFC 7208 — Sender Policy Framework (SPF). URL: https://www.rfc-editor.org/rfc/rfc7208 (Дата звернення: 25.10.2025).
10. RFC 6376 — DomainKeys Identified Mail (DKIM) Signatures. URL: https://www.rfc-editor.org/rfc/rfc6376 (Дата звернення: 20.10.2025).
11. Cloudflare Docs. DNS, DMARC, SPF and DKIM best practices. URL: https://developers.cloudflare.com (Дата звернення: 05.10.2025).
12. About Microsoft Defender for Office 365 URL: https://www.microsoft.com/uk-ua/security/business/siem-and-xdr/microsoft-defender-office-365 (Дата звернення: 30.10.2025).

13. About Proofpoint Email Protection URL: https://www.proofpoint.com/us/products/email-protection (Дата звернення: 30.10.2025).
14. About Darktrace / EMAIL URL: https://www.darktrace.com/products/email (Дата звернення: 30.10.2025).
15. Ubuntu Server Documentation. URL: https://documentation.ubuntu.com/server/ (Дата звернення: 03.11.2025).
16. Microsoft Docs. Active Directory Domain Services Deployment. URL: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/deploy/ (Дата звернення: 05.11.2025).
17. Microsoft Windows Server Documentation. URL: https://learn.microsoft.com/en-us/windows-server/ (Дата звернення: 05.11.2025).
18. OPNsense Documentation. URL: https://docs.opnsense.org (Дата звернення: 10.11.2025).
19. WireGuard. Technical Whitepaper. URL: https://www.wireguard.com (Дата звернення: 11.11.2025).
20. Postfix Documentation. URL: https://www.postfix.org/documentation.html (Дата звернення: 14.11.2025).
21. Dovecot Documentation. URL: https://doc.dovecot.org/2.4.2/ (Дата звернення: 17.11.2025).
22. Grafana Labs Documentation. URL: https://grafana.com/docs/ (Дата звернення: 23.11.2025).
23. VirusTotal Documentation. URL: https://docs.virustotal.com/reference/overview (Дата звернення: 24.11.2025).
24. DeepL API Documentation. URL: https://developers.deepl.com/docs/getting-started/intro (Дата звернення: 19.11.2025).
25. FastAPI Documentation. URL: https://fastapi.tiangolo.com/ (Дата звернення: 19.11.2025).
26. Microsoft Docs. Active Directory Certificate Services. URL: https://learn.microsoft.com/en-us/windows-server/networking/core-network-

guide/cncg/server-certs/install-the-certification-authority (Дата звернення: 28.11.2025).

27. APWG. Phishing Activity Trends Report. URL: https://apwg.org (Дата звернення: 28.11.2025).
28. ENISA. Phishing Threat Landscape 2024. URL: https://www.enisa.europa.eu (Дата звернення: 01.12.2025).
29. Microsoft Security Blog. AI-powered protection against phishing attacks. URL: https://www.microsoft.com/security/blog (Дата звернення: 17.11.2025).
30. HuggingFace. DistilBERT Model Card. URL: https://huggingface.co/distilbert-base-uncased (Дата звернення: 05.12.2025).
31. RFC 7489 — Domain-based Message Authentication, Reporting, and Conformance (DMARC). URL: https://datatracker.ietf.org/doc/rfc7489 (Дата звернення: 15.11.2025).
32. RFC 7208 — Sender Policy Framework (SPF). URL: https://datatracker.ietf.org/doc/rfc7208 (Дата звернення: 21.11.2025).
33. RFC 6376 — DomainKeys Identified Mail (DKIM). URL: https://datatracker.ietf.org/doc/rfc6376 (Дата звернення: 02.12.2025).
34. Keycloak Documentation. URL: https://www.keycloak.org/documentation
35. Brownlee J. Deep Learning for Natural Language Processing. URL: https://machinelearningmastery.com (Дата звернення: 04.12.2025).
36. Chollet F. (2021) Deep Learning with Python (2nd ed.). Manning Publications.
37. Goodfellow I., Bengio Y., Courville A. (2016) Deep Learning. MIT Press.
38. 2ip.ua. Ip address check URL: https://2ip.ua/ (Дата звернення: 12.12.2025).
39. Sanh V., Debut L., Chaumond J., Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint, arXiv:1910.01108, 2019. URL: https://arxiv.org/abs/1910.01108 (Дата звернення: 29.11.2025).
40. Zhang X., Chen L., Li Y. (2021) BERT for Email Classification: A Comparative Study. *IEEE Access* URL: https://www.academia.edu/3088987/A_Comparative_Study_for_Email_Classification (Дата звернення: 27.11.2025).

ДОДАТКИ

Додаток А. Вигляд налаштованого домен контроллеру

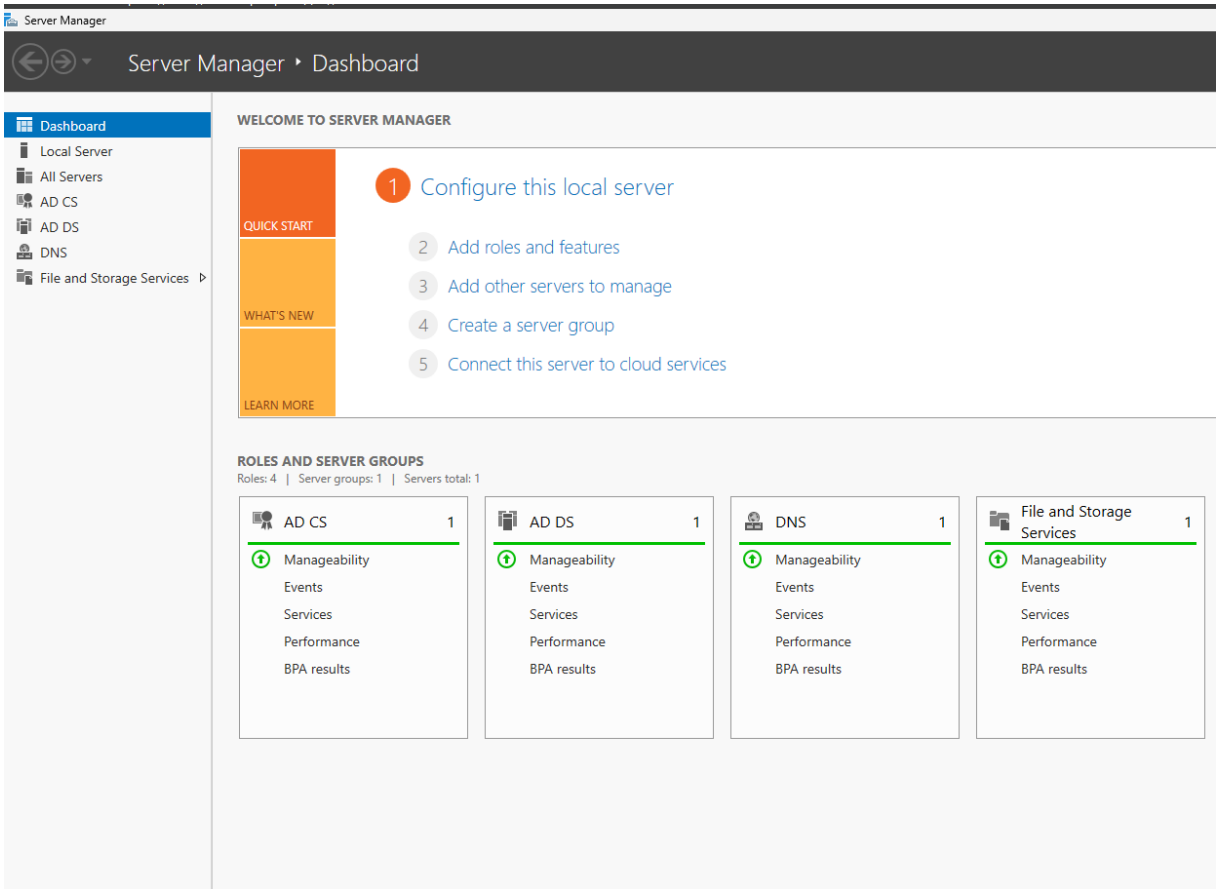

Рисунок А.1 – Вигляд Server Manager на налаштованому контроллері домену

Додаток Б. Код сервісу ML API

```
import os
import json
import time
import logging
from typing import List, Dict, Any, Optional

from fastapi import FastAPI
from pydantic import BaseModel, Field

import requests
from urllib.parse import urlparse

from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline
import torch
from langdetect import detect
from html.parser import HTMLParser

# КОНФІГ ДЛЯ WINDOWS

BASE_DIR = "C:\\Users\\maksb\\Desktop\\files\\api"
MODEL_DIR = os.path.join(BASE_DIR, "model_distilbert_text")

# каталог для логів сервісу
LOG_DIR   = os.path.join(BASE_DIR, "logs")
INFER_LOG = os.path.join(LOG_DIR, "inference.log")
RAW_DUMP_DIR = os.path.join(LOG_DIR, "incoming")
```

```
os.makedirs(LOG_DIR, exist_ok=True)
os.makedirs(RAW_DUMP_DIR, exist_ok=True)

# Ключі беремо з env
VT_API_KEY = os.getenv("VT_API_KEY")
DEEPL_KEY  = os.getenv("DEEPL_KEY")

CLASS_MAP = {
    "0": "legit",
    "1": "phishing"
}

CORPORATE_DOMAINS = {"test-app.org"}

def extract_sender_domain(from_field: Optional[str]) -> str:
    if not from_field:
        return ""
    try:
        if "<" in from_field and ">" in from_field:
            inner = from_field.split("<", 1)[1].split(">", 1)[0]
        else:
            inner = from_field
        if "@" in inner:
            domain = inner.split("@", 1)[1].strip().lower()
            return domain.strip().strip(">")
        return ""
    except Exception:
```

```
        return ""


# логер
logging.basicConfig(level=logging.INFO)
log = logging.getLogger("mlapi")



class _HTMLTextExtractor(HTMLParser):
    def __init__(self):
        super().__init__()
        self._chunks = []

    def handle_data(self, data):
        if data:
            self._chunks.append(data)

    def get_text(self):
        return " ".join(self._chunks)

def html_to_text(html: str) -> str:
    """
    Дуже простий HTML->текст. Без збереження форматування,
    без href, без alt. Але достатньо для моделі.
    """
    parser = _HTMLTextExtractor()
    parser.feed(html)
    return parser.get_text()
```

```
# DeepL ПЕРЕКЛАД → EN

def translate_to_english(text: str) -> str:
    """
    Перекладає text на англійську через DeepL (free/pro),
    якщо DEEPL_KEY задано.
    Якщо ключа нема або DeepL впав, повертаємо оригінал.
    """
    if not DEEPL_KEY:
        return text

    url = "https://api-free.deepl.com/v2/translate"
    data = {
        "text": text,
        "target_lang": "EN"
    }
    headers = {
        "Authorization": f"DeepL-Auth-Key {DEEPL_KEY}"
    }

    try:
        r = requests.post(url, data=data, headers=headers, timeout=20)
        if r.status_code != 200:
            return text
        resp = r.json()
        translations = resp.get("translations", [])
        if not translations:
            return text
```

```
        return translations[0].get("text", text)
    except Exception:
        return text


# ДІСТАТИ ДОМЕН З URL


def extract_domain(u: str) -> str:
    """
    https://abc.com/path?q=x -> abc.com
    Забираємо юзер:пас@ і порт.
    """
    try:
        parsed = urlparse(u)
        host = parsed.netloc or ""
        if "@" in host:
            host = host.split("@", 1)[1]
        host = host.split(":")[0]
        return host.lower()
    except Exception:
        return ""


# VIRUSTOTAL ЗАПИТИ


def vt_get_domain_report(domain: str) -> Dict[str, Any]:


    if not VT_API_KEY:
        return {"error": "NO_API_KEY"}
```

```
    url = f"https://www.virustotal.com/api/v3/domains/{domain}"
    headers = {"x-apikey": VT_API_KEY}

    try:
        r = requests.get(url, headers=headers, timeout=20)
        if r.status_code != 200:
            return {"error": f"status {r.status_code}", "text": r.text}
        data = r.json()
        attrs = data.get("data", {}).get("attributes", {})
        return {
            "reputation": attrs.get("reputation"),
            "last_analysis_stats": attrs.get("last_analysis_stats", {}),
            "categories": attrs.get("categories", {}),
        }
    except Exception as e:
        return {"error": str(e)}

def vt_get_file_report(file_hash: str) -> Dict[str, Any]:

    if not VT_API_KEY:
        return {"error": "NO_API_KEY"}

    url = f"https://www.virustotal.com/api/v3/files/{file_hash}"
    headers = {"x-apikey": VT_API_KEY}

    try:
        r = requests.get(url, headers=headers, timeout=20)
        if r.status_code != 200:
```

```
            return {"error": f"status {r.status_code}", "text": r.text}
        data = r.json()
        attrs = data.get("data", {}).get("attributes", {})
        stats = attrs.get("last_analysis_stats", {})
        return {
            "stats": {
                "malicious":   stats.get("malicious"),
                "suspicious":  stats.get("suspicious"),
                "undetected":  stats.get("undetected"),
                "harmless":    stats.get("harmless"),
                "timeout":     stats.get("timeout"),
            },
            "meaningful_name": attrs.get("meaningful_name"),
            "type_description": attrs.get("type_description"),
            "size": attrs.get("size"),
        }
    except Exception as e:
        return {"error": str(e)}


def tune_score(
    phishing_score: float,
    sender_domain: str,
    vt_domains: Dict[str, Any],
    vt_hashes: Dict[str, Any],
) -> float:
    """
    Тюнінг фінального score за твоїми правилами:
```

```
1) Якщо лист від корпоративного домену (CORPORATE_DOMAINS)
   і score > 0.5 -> знижуємо до 0.5.

2) Якщо є хоч один домен/файл, у якого в VirusTotal
   malicious >= 3 -> score = 1.0.
"""
score = float(phishing_score or 0.0)

if sender_domain and sender_domain in CORPORATE_DOMAINS and score > 0.5:
    score = 0.5

# Внутрішня функція для безпечного читання malicious
def has_heavy_malicious(stats_dict: Dict[str, Any]) -> bool:
    mal = stats_dict.get("malicious")
    try:
        mal = int(mal or 0)
    except (TypeError, ValueError):
        mal = 0
    return mal >= 3

# Перевіряємо домени з VT
for rep in (vt_domains or {}).values():
    stats = (rep or {}).get("last_analysis_stats", {}) or {}
    if has_heavy_malicious(stats):
        score = 1.0
        break

# Перевіряємо файли (хеші) з VT, якщо ще не 1.0
```

```
    if score < 1.0:
        for rep in (vt_hashes or {}).values():
            stats = (rep or {}).get("stats", {}) or {}
            if has_heavy_malicious(stats):
                score = 1.0
                break

    # Нормалізація
    if score < 0.0:
        score = 0.0
    if score > 1.0:
        score = 1.0

    return score


# Pydantic MODELS (payload milter)

class Attachment(BaseModel):
    filename: Optional[str] = None
    content_type: Optional[str] = None
    size: Optional[int] = None
    sha256: Optional[str] = None
    sha1: Optional[str] = None
    md5: Optional[str] = None
    truncated: Optional[bool] = None

class AuthResults(BaseModel):
```

```
    spf: Optional[str] = None
    dkim: Optional[str] = None
    dmarc: Optional[str] = None
    raw: Optional[str] = None


class Mail(BaseModel):
    message_id: Optional[str] = None
    from_: Optional[str] = Field(default=None, alias="from")
    to: List[str] = []
    headers: Dict[str, Any] = {}
    body_text: Optional[str] = None
    body_html: Optional[str] = None
    body_snippet: Optional[str] = None
    urls: List[str] = []
    attachments: List[Attachment] = []
    truncated: bool = False
    auth_results: Optional[AuthResults] = None
    client_ip: Optional[str] = None
    received_ips: List[str] = []
    received_headers: List[str] = []
    _received_at_utc: Optional[str] = None

    class Config:
        populate_by_name = True

def get_subject(headers: Dict[str, Any]) -> str:

    for k, v in headers.items():
```

```
        if k.lower() == "subject":
            return str(v or "")
    return ""

# ЗАВАНТАЖЕННЯ МОДЕЛІ (DistilBERT)

log.info("Loading ML model from %s", MODEL_DIR)

_tokenizer = AutoTokenizer.from_pretrained(MODEL_DIR)
_model     = AutoModelForSequenceClassification.from_pretrained(MODEL_DIR)

# GPU на хості і torch.cuda.is_available() == True,
# device буде 0. Інакше -1 (CPU).
_device = 0 if torch.cuda.is_available() else -1

_classifier = pipeline(
    task="text-classification",
    model=_model,
    tokenizer=_tokenizer,
    device=_device,
    top_k=None,
    truncation=True,
    max_length=512,
    padding="max_length",
)

log.info("Model loaded. Using device=%s (0 means CUDA, -1 means CPU)", _device)
```

```
# FASTAPI APP

app = FastAPI(
    title="mlapi (Windows host, 192.168.1.10)",
    description="Receives email data from milter on 192.168.1.12 and returns verdict",
)

def run_inference(mail: Mail) -> Dict[str, Any]:



    # --- 1. контент листа
    if mail.body_text and mail.body_text.strip():
        body_text_original = mail.body_text
    elif mail.body_html and mail.body_html.strip():
        body_text_original = html_to_text(mail.body_html)
    else:
        body_text_original = ""

    if not body_text_original.strip():
        # якщо немає контенту для моделі, вважай accept
        return {
            "error": "empty_body",
            "action": "accept",
            "verdict": "no_content",
            "phishing_score": 0.0,
            "legit_score": 1.0,
        }
```

```
# --- 2. detect language
try:
    detected_lang = detect(body_text_original)
except Exception:
    detected_lang = "unknown"


# --- 3. translate to EN if needed
if detected_lang.lower() != "en":
    body_for_model = translate_to_english(body_text_original)
    translated_used = True
else:
    body_for_model = body_text_original
    translated_used = False


# --- 4. inference DistilBERT
pred_scores = _classifier(body_for_model)
if (not pred_scores) or (not isinstance(pred_scores[0], list)):
    return {
        "error": "bad_model_output",
        "action": "accept",
        "verdict": "model_error",
    }


scores_for_text = pred_scores[0]


# будуємо словник {"0":score0, "1":score1}
score_map = {}
for item in scores_for_text:
```

```
    lbl = item.get("label")
    scr = item.get("score")
    if lbl is not None and scr is not None:
        score_map[str(lbl)] = float(scr)

# знаходимо найвищий score
best_item = max(scores_for_text, key=lambda x: x.get("score", 0.0))
best_label_raw = str(best_item.get("label", "0"))
best_label_human = CLASS_MAP.get(best_label_raw, best_label_raw)

phishing_score = score_map.get("1", 0.0)  # імовірність класу "phishing"
legit_score    = score_map.get("0", 0.0)  # імовірність класу "legit"

# --- 5. URL → домени
urls_list = mail.urls or []
domains_raw = [extract_domain(u) for u in urls_list if isinstance(u, str)]
domains_raw = [d for d in domains_raw if d]
seen = set()
domains_unique = []
for d in domains_raw:
    if d not in seen:
        seen.add(d)
        domains_unique.append(d)

# --- 6. Вкладення → SHA256
hashes_unique = []
for att in (mail.attachments or []):
    if isinstance(att, Attachment):
```

```
        if att.sha256 and att.sha256 not in hashes_unique:
            hashes_unique.append(att.sha256)


# --- 7. VirusTotal lookup
if domains_unique:
    vt_domains_report = {d: vt_get_domain_report(d) for d in domains_unique}
else:
    vt_domains_report = {}


if hashes_unique:
    vt_hashes_report  = {h: vt_get_file_report(h) for h in hashes_unique}
else:
    vt_hashes_report = {}


sender_domain = extract_sender_domain(mail.from_)


final_score = tune_score(
    phishing_score=phishing_score,
    sender_domain=sender_domain,
    vt_domains=vt_domains_report,
    vt_hashes=vt_hashes_report,
)
# --- 9. Фінальний вердикт на основі final_score


if final_score >= 0.9:
    final_verdict = "phishing"
    action = "reject"
elif final_score >= 0.7:
```

```
    final_verdict = "suspicious"
    action = "quarantine"
elif final_score >= 0.3:
    final_verdict = "suspicious"
    action = "accept"
else:
    final_verdict = "legit"
    action = "accept"

result = {
    "action": action,
    "verdict": final_verdict,
    "model_best_class_raw": best_label_raw,
    "model_best_class_human": best_label_human,
    "phishing_score": phishing_score,
    "legit_score": legit_score,
    "detected_lang": detected_lang,
    "translated_used": translated_used,

    "domains": domains_unique,
    "hashes": hashes_unique,
    "vt_domains": vt_domains_report,
    "vt_hashes": vt_hashes_report,

    "final_score": final_score,
    "sender_domain": sender_domain,
}
```

```
    return result
def write_local_log(mail: Mail, inference: Dict[str, Any]) -> None:


    try:
        log_record = {
            "ts_utc": time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime()),
            "message_id": mail.message_id,
            "from": mail.from_,
            "to": mail.to,
            "subject": get_subject(mail.headers),
            "client_ip": mail.client_ip,
            "action": inference.get("action"),
            "verdict": inference.get("verdict"),
            "phishing_score": inference.get("phishing_score"),
            "legit_score": inference.get("legit_score"),
            "final_score": inference.get("final_score"),
            "lang": inference.get("detected_lang"),
            "translated_used": inference.get("translated_used"),
            "domains": inference.get("domains", []),
            "hashes": inference.get("hashes", []),


        }


        with open(INFER_LOG, "a", encoding="utf-8") as f:
            f.write(json.dumps(log_record, ensure_ascii=False) + "\n")


    except Exception as e:
        log.warning("Cannot write local inference log: %s", e)
```

```
def save_raw_payload(mail: Mail) -> None:

    ts = time.strftime("%Y%m%d_%H%M%S", time.gmtime())
    safe_msgid = (mail.message_id or "nomsgid")
    safe_msgid = safe_msgid.replace("<", "").replace(">", "").replace("@", "_")
    dump_name = f"{ts}_{safe_msgid}.json"
    dump_path = os.path.join(RAW_DUMP_DIR, dump_name)

    try:
        with open(dump_path, "w", encoding="utf-8") as f:
            json.dump(
                mail.model_dump(by_alias=True, exclude_none=True),
                f,
                ensure_ascii=False,
                indent=2
            )
        log.info("Saved raw payload -> %s", dump_path)
    except Exception as e:
        log.warning("Cannot save raw payload: %s", e)

@app.post("/score")
async def score(mail: Mail):

    subj = get_subject(mail.headers)
    log.info("mlapi got mail: subj=%r from=%r to=%r", subj, mail.from_, mail.to)
```

```
save_raw_payload(mail)

# інференс
inference = run_inference(mail)

# зберегти summary у лог
write_local_log(mail, inference)

# відповідь для milter
return {
    "action": inference.get("action", "accept"),
    "verdict": inference.get("verdict", "clean"),
    "score": inference.get("final_score", inference.get("phishing_score", 0.0)),

    "model_best_class_raw": inference.get("model_best_class_raw"),
    "model_best_class_human": inference.get("model_best_class_human"),

    "phishing_score": inference.get("phishing_score"),
    "legit_score": inference.get("legit_score"),

    "lang": inference.get("detected_lang"),
    "translated_used": inference.get("translated_used"),
}
```

Додаток В. Код сервісу API для взаємодії з postfix

```
from fastapi import FastAPI, HTTPException, Request, Header
from pydantic import BaseModel, EmailStr
import re, subprocess, os, fcntl

API_TOKEN = os.getenv("MAIL_ACTIONS_TOKEN", "changeme-please")

# пути карт
MAPS = {
    "sender_domain":      ("/etc/postfix/sender_access",      "hash"),
    "sender_regex":       ("/etc/postfix/sender_access.regexp","regexp"),
    "sender_email":       ("/etc/postfix/sender_access",      "hash"),
    "recipient_local":    ("/etc/postfix/recipient_access",   "hash"),
    "client":             ("/etc/postfix/client_access",      "hash"),
    "sasl_user":          ("/etc/postfix/sasl_access",        "hash"),
}

app = FastAPI(title="mail-actions")

class BlockItem(BaseModel):
    type: str   # sender_domain|sender_email|sender_regex|recipient_local|client|sasl_user
    value: str
    reason: str | None = None

def _valid_domain(d):
    return re.fullmatch(r"[a-z0-9.-]+\.[a-z]{2,}", d, re.I)

def _valid_ip(s):
```

```
    return re.fullmatch(r"(\d{1,3}\.){3}\d{1,3}", s) or re.fullmatch(r"[0-9a-f:]+", s, re.I)


def _postmap_rebuild(path, kind):
    if kind == "hash":
        subprocess.run(["/usr/sbin/postmap", path], check=True)


def _postfix_reload():
    subprocess.run(["/usr/sbin/postfix", "reload"], check=True)


def _atomic_append(path, line):


    with open(path, "a+", encoding="utf-8") as f:
        fcntl.flock(f, fcntl.LOCK_EX)
        f.seek(0)
        content = f.read().splitlines()
        if line.strip() not in [x.strip() for x in content]:
            f.write(line + "\n")
        fcntl.flock(f, fcntl.LOCK_UN)


def _atomic_remove(path, predicate):
    kept = []
    changed = False
    with open(path, "r+", encoding="utf-8") as f:
        fcntl.flock(f, fcntl.LOCK_EX)
        for ln in f:
            if predicate(ln):
                changed = True
                continue
```

```
            kept.append(ln)
        f.seek(0); f.truncate(0); f.writelines(kept)
        fcntl.flock(f, fcntl.LOCK_UN)
    return changed

def _require_auth(auth: str):
    if auth != f"Bearer {API_TOKEN}":
        raise HTTPException(401, "unauthorized")

@app.post("/api/block")
def block(item: BlockItem, authorization: str = Header("")):
    _require_auth(authorization)
    if item.type not in MAPS:
        raise HTTPException(400, "bad type")
    path, kind = MAPS[item.type]
    reason = item.reason or "Blocked via API"

    if item.type == "sender_domain":
        if not _valid_domain(item.value):
            raise HTTPException(400, "bad domain")
        line = f"{item.value}\tREJECT {reason}"

    elif item.type == "sender_email":
        if "@" not in item.value:
            raise HTTPException(400, "bad email")

        local, _, domain = item.value.partition("@")
        if not local or not domain or "." not in domain:
```

```
            raise HTTPException(400, "bad email")

        item.value = item.value.strip()

        line = f"{item.value}\tREJECT {reason}"

    elif item.type == "sender_regex":
        # перевірка валідності regex
        try: re.compile(item.value)
        except: raise HTTPException(400, "bad regex")
        line = f"/{item.value}/\tREJECT {reason}"

    elif item.type == "recipient_local":
        line = f"{item.value}\tREJECT {reason}"

    elif item.type == "client":
        if not (_valid_ip(item.value) or item.value):
            raise HTTPException(400, "bad client")
        line = f"{item.value}\tREJECT {reason}"

    elif item.type == "sasl_user":

        line = f"{item.value}\tREJECT {reason}"

    else:
        raise HTTPException(400, "bad type")

    _atomic_append(path, line)
```

```
    _postmap_rebuild(path, kind)
    _postfix_reload()
    return {"ok": True, "type": item.type, "value": item.value}


class UnblockItem(BaseModel):
    type: str
    value: str


@app.post("/api/unblock")
def unblock(item: UnblockItem, authorization: str = Header("")):
    _require_auth(authorization)
    if item.type not in MAPS: raise HTTPException(400, "bad type")
    path, kind = MAPS[item.type]

    def predicate(ln: str) -> bool:
        if item.type == "sender_regex":
            # строки вида '/regex/    REJECT ...'
            return ln.strip().startswith(f"/{item.value}/")
        else:
            return ln.strip().startswith(item.value + "\t")

    changed = _atomic_remove(path, predicate)
    _postmap_rebuild(path, kind)
    _postfix_reload()
    return {"ok": True, "removed": changed}


@app.get("/api/list")
def list_all(authorization: str = Header("")):
```

```
    _require_auth(authorization)

    files = {
        "sender_access":        "/etc/postfix/sender_access",
        "sender_access_regexp": "/etc/postfix/sender_access.regexp",
        "recipient_access":     "/etc/postfix/recipient_access",
        "client_access":        "/etc/postfix/client_access",
        "sasl_access":          "/etc/postfix/sasl_access",
    }
    out = {}
    for name, path in files.items():
        try:
            with open(path, encoding="utf-8") as f:
                out[name] = [ln.rstrip("\n") for ln in f if ln.strip()]
        except FileNotFoundError:
            out[name] = []
    return out


@app.post("/api/postfix/reload")
def reload_postfix(authorization: str = Header("")):
    _require_auth(authorization)
    _postfix_reload()
    return {"ok": True}


@app.get("/api/debug-auth")
async def debug_auth(request: Request, authorization: str = Header(None)):
    return {
        "authorization_param": authorization,
```

```
    "API_TOKEN": API_TOKEN,
    "expected": f"Bearer {API_TOKEN}",
    "all_headers": dict(request.headers),
}
```

Додаток Г. Код сервісу MLmilter для перехоплення елетронних листів

```
import sys
import os
import traceback
import time
import base64
import re
import hashlib
from email import policy
from email.parser import BytesParser
from html.parser import HTMLParser

print("[mlmilter] Python:", sys.version, flush=True)
try:
    import httpx
    import Milter  # pymilter (v1.0.5)
    print("[mlmilter] Imports OK. httpx:", httpx.__version__, flush=True)
except Exception as e:
    print("[mlmilter][FATAL] import error:", e, file=sys.stderr, flush=True)
    traceback.print_exc()
    sys.exit(1)

ML_API_URL = os.environ.get("ML_API_URL", "http://192.168.1.10:8000/score")
REQUEST_TIMEOUT = float(os.environ.get("ML_API_TIMEOUT", "2.5"))
MAX_BODY_COLLECT = int(os.environ.get("MAX_BODY_COLLECT", str(64 * 1024)))
MAX_BODY_TEXT = int(os.environ.get("MAX_BODY_TEXT", str(32 * 1024)))

# Finds http/https urls
```

```
    URL_RE = re.compile(r'https?://[^\s\'"<>]+', re.IGNORECASE)

    # Finds protocol-relative urls //example.com/path

    PROTOLESS_RE = re.compile(r'(?:(?<=["\'])|(?<=\s))//[^\s\'"<>]+', re.IGNORECASE)

    # Finds quoted tokens that look like domains or "www..." without scheme (heuristic)

    DOMAIN_LIKE_RE = re.compile(r'["\']([A-Za-z0-9\-._~:/?#\[\]@!$&\'()*+,;=%]*[A-Za-z0-9\-]+\.[A-Za-z]{2,6}[^\s\'"<>]*)["\']')


    # Received header IP extraction

    IP_RE = re.compile(r'\[?(\d{1,3}(?:\.\d{1,3}){3})\]?')


    # Authentication results (simple parse)

    AUTH_SPF_RE = re.compile(r'\bspf=(pass|fail|neutral|softfail|permerror|temperror)\b', re.IGNORECASE)

    AUTH_DKIM_RE = re.compile(r'\bdkim=(pass|fail|neutral|none|policy)\b', re.IGNORECASE)

    AUTH_DMARC_RE = re.compile(r'\bdmarc=(pass|fail|none)\b', re.IGNORECASE)


    class HrefExtractor(HTMLParser):

        """Simple HTML parser to collect href/src/data-href and script text."""

        def __init__(self):

            super().__init__()

            self.hrefs = []

            self.script_texts = []


        def handle_starttag(self, tag, attrs):

            attrs = dict(attrs)

            if tag == 'a':

                for k in ('href', 'data-href', 'data-url'):

                    if k in attrs and attrs[k]:
```

```
                self.hrefs.append(attrs[k])
        elif tag in ('img', 'script', 'link', 'iframe'):
            for k in ('src', 'href', 'data-src', 'data-href'):
                if k in attrs and attrs[k]:
                    self.hrefs.append(attrs[k])

    def handle_data(self, data):
        self.script_texts.append(data)

class MlMilter(Milter.Base):
    @staticmethod
    def factory():
        return MlMilter()

    def __init__(self):
        self.id = Milter.uniqueID()
        self.headers = {}
        self.raw_headers_lines = []
        self.all_headers = {}
        self.body_bytes = bytearray()
        self.from_addr = ""
        self.rcpt_to = []
        print(f"[mlmilter:{self.id}] session start", flush=True)

    # MAIL FROM (envelope)
    def envfrom(self, mailfrom, *args):
        self.from_addr = mailfrom
        return Milter.CONTINUE
```

```
# RCPT TO
def envrcpt(self, rcptto, *args):
    self.rcpt_to.append(rcptto)
    return Milter.CONTINUE



def header(self, name, value):
    ln = name.lower()
    # store first occurrence in simple dict (like before)
    if ln not in self.headers:
        self.headers[ln] = value
    # store raw header line for full reconstruction
    try:
        # ensure no newlines in header value (folded headers may come folded)
        raw_line = f"{name}: {value}"
        self.raw_headers_lines.append(raw_line)
    except Exception:
        pass
    # store list of all headers
    self.all_headers.setdefault(ln, []).append(value)
    return Milter.CONTINUE

# Body chunks (DATA)
def body(self, chunk):
    if len(self.body_bytes) < MAX_BODY_COLLECT:
        need = MAX_BODY_COLLECT - len(self.body_bytes)
        if need > 0:
```

```
            self.body_bytes.extend(chunk[:need])
    return Milter.CONTINUE


def _extract_urls_heuristic(self, combined_text, html_text=None, script_texts=None):
    """
    Return deduplicated list of urls discovered by multiple heuristics:
    - http(s):// via URL_RE
    - protocol-relative //... via PROTOLESS_RE (prefix with http:)
    - href/src values extracted from HTML parser (prefix protocol if needed)
    - domain-like quoted tokens (heuristic)
    - simple scan of script_texts for string tokens that look like domains or URLs
    """
    found = set()

    # 1) explicit http(s)
    for m in URL_RE.finditer(combined_text):
        found.add(m.group(0))

    # 2) protocol-relative //example.com/...
    for m in PROTOLESS_RE.finditer(combined_text):
        val = m.group(0)
        found.add("http:" + val)

    # 3) domain-like quoted tokens (heuristic)
    for m in DOMAIN_LIKE_RE.finditer(combined_text):
        v = m.group(1)
        if v.startswith('//'):
            found.add('http:' + v)
```

```
    elif v.startswith('http') or v.startswith('https'):
        found.add(v)
    else:
        # if doesn't include scheme, add http:// prefix
        if v.startswith('/'):
            # relative path, not helpful as external link — skip
            continue
        if re.match(r'^[\w\.-]+(:\d+)?/.*', v) or '.' in v:
            found.add('http://' + v.lstrip('/'))

# 4) hrefs extracted from parsed HTML
if html_text is not None:
    he = HrefExtractor()
    try:
        he.feed(html_text)
        for h in he.hrefs:
            if not h:
                continue
            if h.startswith('//'):
                found.add('http:' + h)
            elif h.startswith('http://') or h.startswith('https://'):
                found.add(h)
            elif h.startswith('/'):
                # relative path — cannot resolve host without base URL, skip
                continue
            else:
                # likely a domain or protocol-less link
                if h.startswith('mailto:') or h.startswith('javascript:'):
```

```
                    continue
                # add http prefix as guess
                found.add('http://' + h)
            script_combined = "\n".join(he.script_texts)
            for m in DOMAIN_LIKE_RE.finditer(script_combined):
                v = m.group(1)
                if v.startswith('//'):
                    found.add('http:' + v)
                elif v.startswith('http'):
                    found.add(v)
                else:
                    if v.startswith('/'):
                        continue
                    found.add('http://' + v.lstrip('/'))
        except Exception:
            pass


    # 5) last-ditch: find any quoted string with dot in combined_text
            for m in re.finditer(r'["\']([^"\']+\.[a-z]{2,6}[^\s"\']*)["\']', combined_text, re.IGNORECASE):
        candidate = m.group(1)
        if candidate.startswith('http'):
            found.add(candidate)
        elif candidate.startswith('//'):
            found.add('http:' + candidate)
         elif '/' in candidate or re.search(r'\.(com|net|org|io|ru|ua|biz|info|gov|edu)', candidate, re.IGNORECASE):
            found.add('http://' + candidate.lstrip('/'))
```

```
    # filter and return sorted list
    # remove items that look like "javascript:..." or mailto
    cleaned = []
    for u in found:
        if not u:
            continue
        if u.lower().startswith('javascript:') or u.lower().startswith('mailto:'):
            continue
        cleaned.append(u)
    return sorted(set(cleaned))


def _parse_auth_results(self, auth_header_value):
    """
    Simple parse of Authentication-Results header for spf/dkim/dmarc.
    Returns dict with keys spf/dkim/dmarc (values or None)
    """
    res = {"spf": None, "dkim": None, "dmarc": None, "raw": auth_header_value}
    if not auth_header_value:
        return res
    m = AUTH_SPF_RE.search(auth_header_value)
    if m:
        res["spf"] = m.group(1).lower()
    m = AUTH_DKIM_RE.search(auth_header_value)
    if m:
        res["dkim"] = m.group(1).lower()
    m = AUTH_DMARC_RE.search(auth_header_value)
    if m:
        res["dmarc"] = m.group(1).lower()
```

```
    return res


def eom(self):
    t0 = time.time()

    # default payload fields
    parsed_subject = self.headers.get("subject", "")
    parsed_from_hdr = self.headers.get("from", "")
    parsed_to_hdr = self.headers.get("to", "")
    parsed_msgid = self.headers.get("message-id", "")

    body_text = ""
    body_html = ""
    urls = []
    attachments_meta = []
    truncated = False

    try:
        # reconstruct full raw message: headers (as collected) + \r\n\r\n + body bytes
        headers_raw = ("\r\n".join(self.raw_headers_lines)).encode("utf-8", errors="replace")
        raw_body = bytes(self.body_bytes)
        if len(raw_body) >= MAX_BODY_COLLECT:
            truncated = True
        raw_full = headers_raw + b"\r\n\r\n" + raw_body

        try:
            msg = BytesParser(policy=policy.default).parsebytes(raw_full)
        except Exception as e:
```

```
                print(f"[mlmilter:{self.id}] email parse error (full msg): {e}", file=sys.stderr, flush=True)
                msg = None

            if msg is not None:
                parsed_subject = parsed_subject or (msg.get("Subject") or "")
                parsed_from_hdr = parsed_from_hdr or (msg.get("From") or "")
                parsed_to_hdr = parsed_to_hdr or (msg.get("To") or "")
                parsed_msgid = parsed_msgid or (msg.get("Message-ID") or "")

                if msg.is_multipart():
                    for part in msg.walk():
                        ctype = part.get_content_type()
                        cdisp = (part.get_content_disposition() or "").lower()
                        filename = part.get_filename()
                        try:
                            payload_bytes = part.get_payload(decode=True)
                        except Exception:
                            payload_bytes = None

                        if ctype == "text/plain" and payload_bytes is not None:
                            try:
                                text = payload_bytes.decode(part.get_content_charset(failobj="utf-8"), errors="replace")
                            except Exception:
                                text = payload_bytes.decode("utf-8", errors="replace")
                            body_text += text + "\n"

                        elif ctype == "text/html" and payload_bytes is not None:
```

```
                try:
                    html = payload_bytes.decode(part.get_content_charset(failobj="utf-8"),
errors="replace")
                except Exception:
                    html = payload_bytes.decode("utf-8", errors="replace")
                body_html += html + "\n"

            # attachments or inline with filename
            if (cdisp == "attachment") or filename:
                meta = {"filename": filename or "", "content_type": ctype}
                if payload_bytes is not None:
                    meta["size"] = len(payload_bytes)
                    meta["sha256"] = hashlib.sha256(payload_bytes).hexdigest()
                    meta["sha1"] = hashlib.sha1(payload_bytes).hexdigest()
                    meta["md5"] = hashlib.md5(payload_bytes).hexdigest()
                else:
                    meta["size"] = None
                    meta["sha256"] = None
                    meta["sha1"] = None
                    meta["md5"] = None
                    meta["truncated"] = truncated
                attachments_meta.append(meta)
        else:
            # singlepart
            ctype = msg.get_content_type()
            try:
                payload_bytes = msg.get_payload(decode=True)
            except Exception:
```

```
                payload_bytes = None
            if ctype == "text/plain" and payload_bytes is not None:
                try:
                    body_text = payload_bytes.decode(msg.get_content_charset(failobj="utf-8"), errors="replace")
                except Exception:
                    body_text = payload_bytes.decode("utf-8", errors="replace")
            elif ctype == "text/html" and payload_bytes is not None:
                try:
                    body_html = payload_bytes.decode(msg.get_content_charset(failobj="utf-8"), errors="replace")
                except Exception:
                    body_html = payload_bytes.decode("utf-8", errors="replace")
            filename = msg.get_filename()
            if filename:
                meta = {"filename": filename, "content_type": ctype}
                if payload_bytes is not None:
                    meta["size"] = len(payload_bytes)
                    meta["sha256"] = hashlib.sha256(payload_bytes).hexdigest()
                    meta["sha1"] = hashlib.sha1(payload_bytes).hexdigest()
                    meta["md5"] = hashlib.md5(payload_bytes).hexdigest()

                else:
                    meta["size"] = None
                    meta["sha256"] = None
                    meta["sha1"] = None
                    meta["md5"] = None
                    meta["truncated"] = truncated
                attachments_meta.append(meta)
```

```
    # Fallback: if no parsed text, decode raw_body to text
    if not body_text and not body_html:
        try:
            raw_text = raw_body.decode("utf-8", errors="replace")
        except Exception:
            raw_text = raw_body.decode("latin-1", errors="replace")
        body_text = raw_text

    # Extract URLs using heuristics: combine plain text + html
    combined_for_urls = (body_text or "") + "\n" + (body_html or "")
    # try HTML parser on body_html to get href/src and script strings
    he_html_text = body_html if body_html else None
    urls = self._extract_urls_heuristic(combined_for_urls, html_text=he_html_text)

    # Trim body_text to reasonable length for inline payload
    if body_text and len(body_text) > MAX_BODY_TEXT:
        body_text = body_text[:MAX_BODY_TEXT] + "\n...[truncated]"

except Exception as e:
    print(f"[mlmilter:{self.id}] parse exception: {e}", file=sys.stderr, flush=True)
    traceback.print_exc()

received_list = self.all_headers.get('received', [])[:]
received_ips = []
for r in received_list:
    for m in IP_RE.finditer(r):
        ip = m.group(1)
```

```
        received_ips.append(ip)
client_ip = None
if received_list:
    last_received = received_list[-1]
    m = IP_RE.search(last_received)
    if m:
        client_ip = m.group(1)

# Authentication-Results and Received-SPF parsing
auth_res = None
if 'authentication-results' in self.all_headers:
    # take first Authentication-Results (if many)
    auth_res = self._parse_auth_results(self.all_headers['authentication-results'][0])
else:
    rs = self.all_headers.get('received-spf', [])
    if rs:
        val = rs[0]
        spf_val = None
        if 'pass' in val.lower():
            spf_val = 'pass'
        elif 'fail' in val.lower():
            spf_val = 'fail'
        auth_res = {"spf": spf_val, "dkim": None, "dmarc": None, "raw": val}

# Build payload for ML API
payload = {
    "message_id": parsed_msgid,
    "from": self.from_addr or parsed_from_hdr,
```

```
    "to": self.rcpt_to or ([x.strip() for x in (parsed_to_hdr or "").split(",") if x.strip()]),
    "headers": {
        "subject": parsed_subject,
        "from": parsed_from_hdr,
        "to": parsed_to_hdr,
        "message-id": parsed_msgid,
    },
    "body_text": body_text,
    "body_html": body_html,
    "body_snippet": base64.b64encode(bytes(self.body_bytes)).decode("ascii"),
    "urls": urls,
    "attachments": attachments_meta,
    "truncated": truncated,
    # new anti-spoof/meta:
    "received_headers": received_list,
    "received_ips": received_ips,
    "client_ip": client_ip,
    "auth_results": auth_res,
}

# Call ML API (inline)
try:
    with httpx.Client(timeout=REQUEST_TIMEOUT) as c:
        r = c.post(ML_API_URL, json=payload)
        r.raise_for_status()
        result = r.json()
except Exception as e:
    print(f"[mlmilter:{self.id}] ML call error: {e}", file=sys.stderr, flush=True)
```

```
        self.addheader("X-ML-Decision", "error", 0)
        self.addheader("X-ML-Error", str(e), 0)
        return Milter.CONTINUE

    # Process result and add headers
    try:
        action = str(result.get("action", "accept"))
        score = float(result.get("score", 0.0))
        verdict = str(result.get("verdict", "unknown"))
        modelv = str(result.get("model_ver", "unknown"))
        reasons = ",".join([str(x) for x in result.get("reasons", [])])
        dt = int((time.time() - t0) * 1000)

        # insert headers at top
        self.addheader("X-ML-Score", str(score), 0)
        self.addheader("X-ML-Verdict", verdict, 0)
        self.addheader("X-ML-Model", modelv, 0)
        self.addheader("X-ML-LatencyMs", str(dt), 0)
        if reasons:
            self.addheader("X-ML-Reasons", reasons, 0)
        # attachments summary
        if attachments_meta:
            self.addheader("X-ML-Attach-Count", str(len(attachments_meta)), 0)
            sha_list = [a.get("sha256") for a in attachments_meta if a.get("sha256")]
            if sha_list:
                self.addheader("X-ML-Attach-SHA", ",".join(sha_list)[:2000], 0)

        if payload.get("client_ip"):
```

```
                self.addheader("X-ML-Client-IP", str(payload.get("client_ip")), 0)
            if payload.get("received_ips"):
              self.addheader("X-ML-Received-IPs", ",".join(payload.get("received_ips"))[:2000], 0)
            if payload.get("auth_results") and payload.get("auth_results").get("raw"):
                                                    self.addheader("X-ML-Auth-Results-Raw", payload.get("auth_results").get("raw")[:2000], 0)
            if payload.get("urls"):
                # keep header size reasonable
                self.addheader("X-ML-URLs-Count", str(len(payload.get("urls"))), 0)
                # optionally add first few urls
                if len(payload.get("urls")):
                    self.addheader("X-ML-URL-0", payload.get("urls")[0][:2000], 0)

                print(f"[mlmilter:{self.id}] action={action} score={score} verdict={verdict} dt_ms={dt}", flush=True)

            # act on verdict
            if action.lower() == "reject":
                self.setreply("550", "5.7.1", "Message rejected by ML policy")
                return Milter.REJECT
            elif action.lower() == "quarantine":
                self.addheader("X-ML-Quarantine", "yes", 0)
                return Milter.CONTINUE

            return Milter.CONTINUE

        except Exception as e:
            print(f"[mlmilter:{self.id}] eom error: {e}", file=sys.stderr, flush=True)
```

```
            self.addheader("X-ML-Decision", "error", 0)
            self.addheader("X-ML-Error", f"eom:{e}", 0)
            return Milter.CONTINUE

    def close(self):
        print(f"[mlmilter:{self.id}] session end", flush=True)
        return Milter.CONTINUE

if __name__ == "__main__":
    try:
        import argparse
        parser = argparse.ArgumentParser()
        parser.add_argument("--listen", default="127.0.0.1")
        parser.add_argument("--port", type=int, default=12302)
        args = parser.parse_args()

        # preserve existing flags
        Milter.set_flags(Milter.ADDHDRS | Milter.P_RCPT_REJ)

        Milter.factory = MlMilter.factory
        sock = f"inet:{args.port}@{args.listen}"
                print(f"[mlmilter] listening on {sock}, ML_API={ML_API_URL}
MAX_BODY_COLLECT={MAX_BODY_COLLECT}", flush=True)
        Milter.runmilter("mlmilter", sock, 600)
    except Exception as e:
        print("[mlmilter][FATAL] top-level error:", e, file=sys.stderr, flush=True)
        traceback.print_exc()
        sys.exit(1)
```