ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Застосунок моніторингу мережевої безпеки з елементами автоматичного виявлення аномалій»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ _____
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД-42
Владислав СТЕПУРА
Ім'я, ПРІЗВИЩЕ

Керівник:
науковий ступінь, вчене звання
К.т.н, доцент Оксана ТКАЛЕНКО
Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання
_____
Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____Каміла СТОРЧАК
«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Степура Владислав Олегович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Застосунок моніторингу мережевої безпеки з елементами автоматичного виявлення аномалій

керівник кваліфікаційної роботи Оксана ТКАЛЕНКО, к. т. н., доцент,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» 02 2026 р. № 51

2. Строк подання кваліфікаційної роботи «01» 06 2026р.

3. Вихідні дані до кваліфікаційної роботи: наукова та технічна література з мережевої безпеки, кіберзагроз і моніторингу трафіку; документація Python, Flask, PyShark, Scikit-learn, SQLite; матеріали щодо методів машинного навчання для виявлення аномалій у мережевому трафіку.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Проаналізувати сучасні кіберзагрози та методи моніторингу мережевої безпеки.
2. Обґрунтувати вибір технологій і методів автоматичного виявлення аномалій.
3. Розробити, реалізувати та протестувати застосунок моніторингу мережевої безпеки.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «20» 02 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та сучасних кіберзагроз.	12.03.2026	
2	Дослідження існуючих систем моніторингу мережевої безпеки.	25.03.2026	
3	Формування вимог до програмного застосунку.	03.04.2026	
4	Вибір технологій та проєктування архітектури системи.	10.04.2026	
5	Програмна реалізація застосунку та модуля виявлення аномалій.	09.05.2026	
6	Розробка демонстраційних матеріалів.	15.05.2026	
7	Попередній захист дипломної роботи.	19.05.2026	
8	Подання роботи в деканат.	01.06.2026	

Здобувач(ка) вищої освіти _____ Владислав СТЕПУРА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Оксана ТКАЛЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 62 стор., 19 рис., 14 джерел.

Мета роботи – розробка програмного застосунку для моніторингу мережевої безпеки, що забезпечує автоматичне виявлення аномальної активності в локальному трафіку за допомогою методів машинного навчання

Об’єкт дослідження – процеси моніторингу та аналізу безпеки в локальних комп’ютерних мережах.

Предмет дослідження – методи та програмні засоби автоматизованого виявлення аномалій у мережевому трафіку.

Короткий зміст роботи: у роботі проведено аналіз сучасних кіберзагроз та існуючих систем моніторингу мережевої безпеки. Обґрунтовано вибір мови Python та алгоритму Isolation Forest для створення інтелектуального ядра системи.

Розроблено архітектуру та програмний код застосунку на базі фреймворку Flask та бібліотеки PyShark. Проведено експериментальне тестування системи на виявлення DDoS-атак та сканування портів, що підтвердило ефективність обраного підходу.

КЛЮЧОВІ СЛОВА:

МЕРЕЖЕВА БЕЗПЕКА, МОНІТОРІНГ ТРАФІКУ, АНОМАЛІЯ, МАШИННЕ НАВЧАННЯ, ISOLATION FOREST, PYTHON, FLASK, PYSHARK, КІБЕРЗЛОЧИН.

ABSTRACT

The textual part of the qualification work for obtaining the Bachelor's degree consists of 62 pages, 19 figures, and 14 references.

The purpose of the work is to develop a software application for network security monitoring that provides automatic detection of anomalous activity in local traffic using machine learning methods.

The object of the research is the processes of monitoring and security analysis in local computer networks.

The subject of the research is methods and software tools for automated anomaly detection in network traffic.

Brief summary of the work: the paper analyzes modern cyber threats and existing network security monitoring systems. The choice of the Python programming language and the Isolation Forest algorithm for creating the intelligent core of the system is substantiated.

The architecture and program code of the application based on the Flask framework and the PyShark library were developed. Experimental testing of the system for detecting DDoS attacks and port scanning was carried out, which confirmed the effectiveness of the chosen approach.

KEYWORDS: NETWORK SECURITY, TRAFFIC MONITORING, ANOMALY, MACHINE LEARNING, ISOLATION FOREST, PYTHON, FLASK, PYSHARK, CYBERCRIME.

ЗМІСТ

ВСТУП

Актуальність теми. Стрімка цифровізація економічних процесів, масове впровадження хмарних технологій та перехід на дистанційні формати роботи докорінно змінили ландшафт мережевої безпеки. За статистикою 2025 року, близько 20% підприємств систематично стають жертвами кіберзлочинів, причому для великого бізнесу цей показник сягає 52%. Традиційні засоби захисту, такі як статичні міжмережеві екрани та сигнатурні антивіруси, часто виявляються неефективними проти нових типів атак та складних методів соціальної інженерії.

Використання інтелектуальних систем моніторингу, здатних виявляти аномалії в режимі реального часу за допомогою алгоритмів машинного навчання, стає критичною необхідністю для забезпечення кіберстійкості сучасної інфраструктури. Саме тому розробка легкого та ефективного застосунку для автоматизованого виявлення мережевих відхилень є актуальним науково-прикладним завданням.

Аналіз останніх досліджень і публікацій. Проблематика мережевого моніторингу та аналізу трафіку висвітлена у працях багатьох вітчизняних та закордонних фахівців. Сучасний ринок пропонує потужні SIEM-платформи та IDS/IPS-рішення. Проте їх впровадження часто супроводжується значними фінансовими витратами та складністю налаштування, що робить їх недоступними для малого та середнього бізнесу. Це зумовлює потребу в створенні гнучких інструментів на базі відкритих технологій та методів навчання без учителя, таких як Isolation Forest.

Метою роботи є розробка програмного застосунку для моніторингу мережевої безпеки, що забезпечує автоматичне виявлення аномальної активності в локальному трафіку за допомогою методів машинного навчання.

Для досягнення поставленої мети було визначено такі завдання:

1. Проаналізувати сучасний стан кіберзагроз та існуючі методи моніторингу мережевої безпеки.

2. Сформувати функціональні та технічні вимоги до розроблюваного програмного продукту.

3. Обґрунтувати вибір технологічного стеку для реалізації системи.

4. Проєктувати архітектуру застосунку та розробити інтелектуальне ядро на базі алгоритму Isolation Forest.

5. Провести тестування розробленого ПЗ на виявлення типових атак та оцінити його ефективність.

Об’єкт дослідження – процеси моніторингу та аналізу безпеки в локальних комп’ютерних мережах.

Предмет дослідження – методи та програмні засоби автоматизованого виявлення аномалій у мережевому трафіку.

Методи дослідження. Для розв’язання поставлених завдань використано: статистичні методи аналізу трафіку, методи поведінкового моделювання та алгоритми машинного навчання. Програмна реалізація виконана з використанням модульного підходу та об’єктно-орієнтованого програмування.

Наукова новизна та практичне значення. Набули подальшого розвитку підходи до автоматизації виявлення мережевих аномалій шляхом інтеграції бібліотеки аналізу пакетів PyShark з моделями Scikit-learn в межах легкого вебзастосунку на Flask. Практичне значення роботи полягає у створенні працездатного прототипу системи, який дозволяє ідентифікувати загрози з точністю детекції аномалій на рівні реальних інцидентів.

Апробація результатів. Основні положення та результати дослідження пройшли апробацію на:

1. **III Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу»** (Київ, ДУІКТ, 18 листопада 2025 р.).

Тези доповіді на тему «Гібридні підходи для інтелектуального аналізу потокових та пакетних даних» опубліковані у збірнику матеріалів конференції (стор. 261).

2. **VII Всеукраїнській науково-технічній конференції «Сучасний стан та перспективи розвитку IoT» (Київ, ДУІКТ, 20 квітня 2026 р.).** Тези доповіді на тему «Інтелектуальний моніторинг мережевих аномалій як базовий елемент безпеки Інтернету речей» опубліковані у збірнику матеріалів конференції (стор. 249).

РОЗДІЛ 1 ОСНОВИ МЕРЕЖЕВОЇ БЕЗПЕКИ ТА АНАЛІЗ ЗАГРОЗ

1.1 Сучасний стан мережевої безпеки та аналіз кіберзагроз

Стабільність функціонування організацій критично залежить від захищеності їхньої цифрової інфраструктури. Поширення хмарних сервісів, дистанційної роботи, мобільного доступу та інтеграції зовнішніх інформаційних систем призвело до суттєвого розширення потенційної поверхні атаки. Унаслідок цього, питання мережевої безпеки набуло стратегічного значення не лише для великих корпорацій, а й для підприємств середнього та малого масштабу.

Згідно з результатами дослідження уряду Сполученого Королівства Великої Британії та Північної Ірландії *Cyber Security Breaches Survey* [1], у 2025 році 20% опитаних підприємств повідомили про випадки кіберзлочинів, зафіксованих протягом останніх 12 місяців. Разом із тим більш детальний аналіз структури інцидентів, наведений на рисунку 1.1, свідчить про суттєву залежність частоти атак від масштабу підприємства. Зі збільшенням розміру бізнесу частка організацій, що зазнали кіберзлочинів, послідовно зростає: від 18% серед мікропідприємств до 52% серед великих компаній.

Водночас рівень кіберзагроз визначається не лише масштабом підприємства, а й специфікою його господарської діяльності, рівнем цифровізації бізнес-процесів та характером інформаційних активів. У зв’язку з цим доцільно розглянути розподіл кіберінцидентів за галузями діяльності підприємств, що наведено на рисунку 1.2.

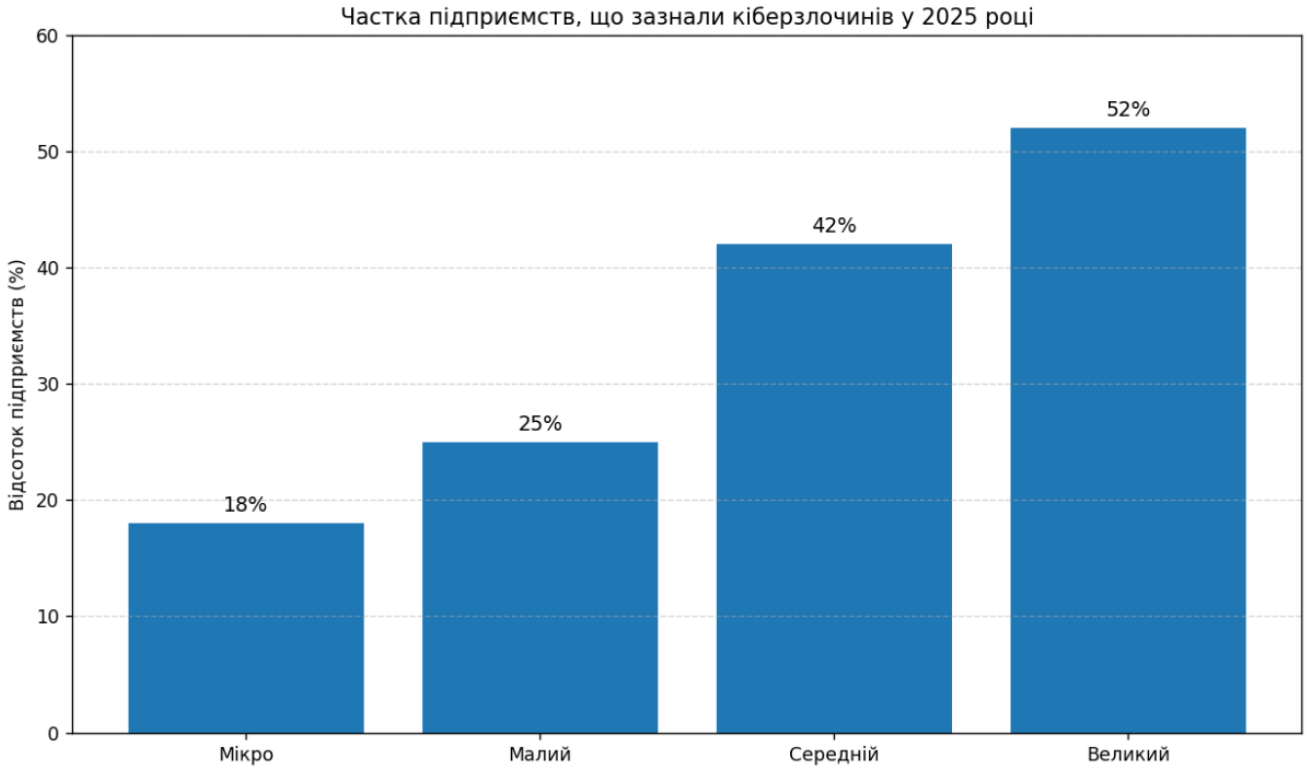

Рис 1.1 Частка підприємств, що зазнали кіберзлочинів у 2025 році (зроблено на основі таблиці [1])

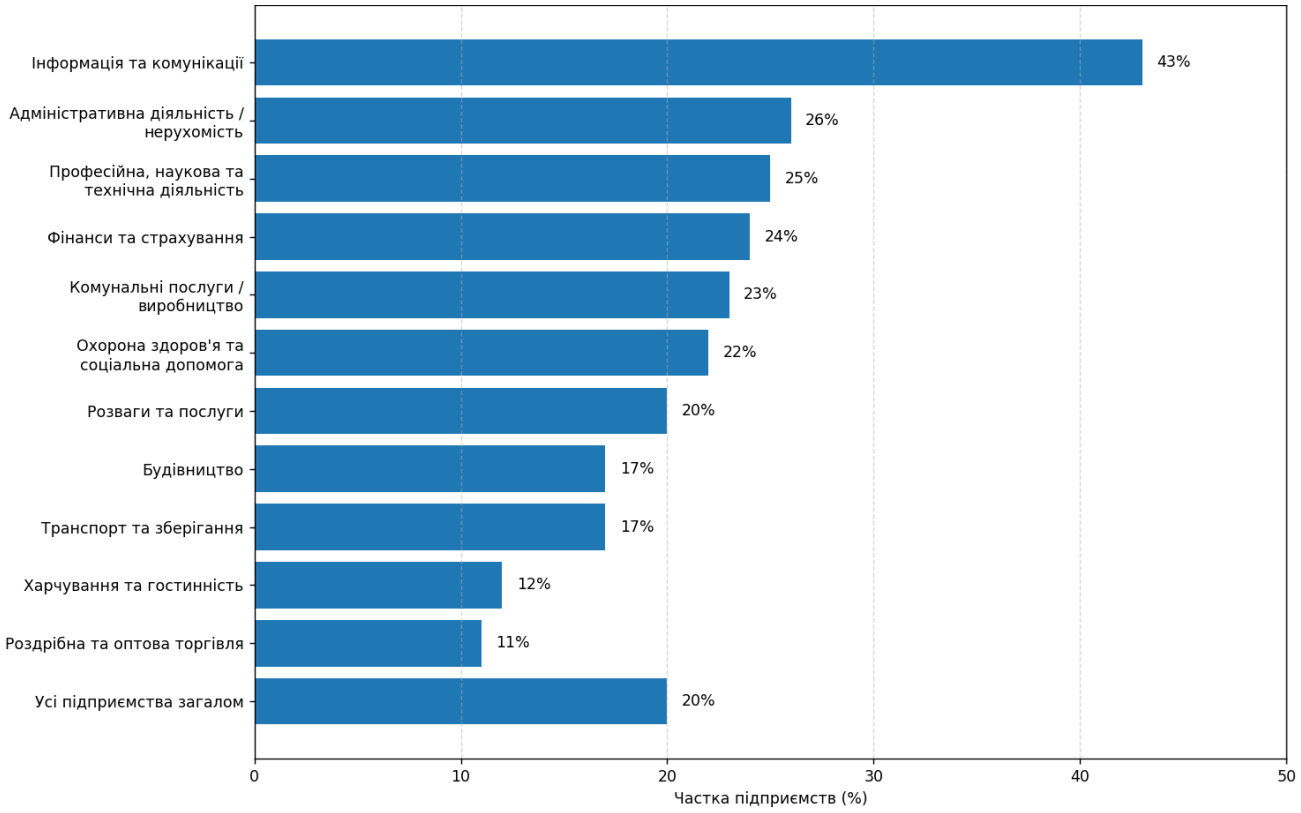

Рис 1.2 Частка підприємств, що зазнали кіберзлочинів за галузями діяльності (зроблено на основі таблиці [1])

Як видно з наведених даних, найбільш уразливими до кіберзлочинів у 2025 році виявилися підприємства сфери інформації та комунікацій, де про інциденти повідомили 43% опитаних організацій. Такий показник більш ніж удвічі перевищує середнє значення серед усіх підприємств (20%). Це може бути пояснено високим рівнем цифровізації зазначеної галузі, значною кількістю мережевих сервісів, використанням віддаленого доступу, хмарних платформ та концентрацією цінних інформаційних ресурсів.

Підвищені значення також характерні для сектору адміністративних послуг та операцій з нерухомістю (26%), професійної, наукової й технічної діяльності (25%), фінансового та страхового сектору (24%), а також сфери комунальних послуг і виробництва (23%). Спільною ознакою цих галузей є активне використання цифрових систем управління, обробка конфіденційної інформації та залежність від безперервності бізнес-процесів, що підвищує їхню привабливість для зловмисників.

Показники, наближені до середнього рівня, спостерігаються у сфері охорони здоров’я та соціальної допомоги (22%), а також у сфері розваг і послуг (20%). Це свідчить про стабільний рівень ризику, характерний для організацій, які використовують інформаційні системи у значному обсязі, однак не належать до найбільш критичних цілей атак.

Найнижчі значення зафіксовано у будівництві (17%), транспорті та зберіганні (17%), сфері харчування й гостинності (12%), а також у роздрібній та оптовій торгівлі (11%). Імовірною причиною нижчої частоти інцидентів може бути менша залежність окремих процесів від складної цифрової інфраструктури або нижчий рівень виявлення та фіксації кіберподій у таких організаціях.

Наведені вище статистичні дані свідчать про нерівномірний розподіл кіберризиків залежно від масштабу підприємства та галузі діяльності. Водночас не менш важливим аспектом є структура самих кіберінцидентів, оскільки різні типи загроз відрізняються механізмами реалізації, рівнем складності виявлення та потенційними наслідками для інформаційної системи. Доцільно проаналізувати, з

якими типами кіберзагроз найчастіше стикаються сучасні підприємства. Відповідні дані наведено на рисунку 1.3.

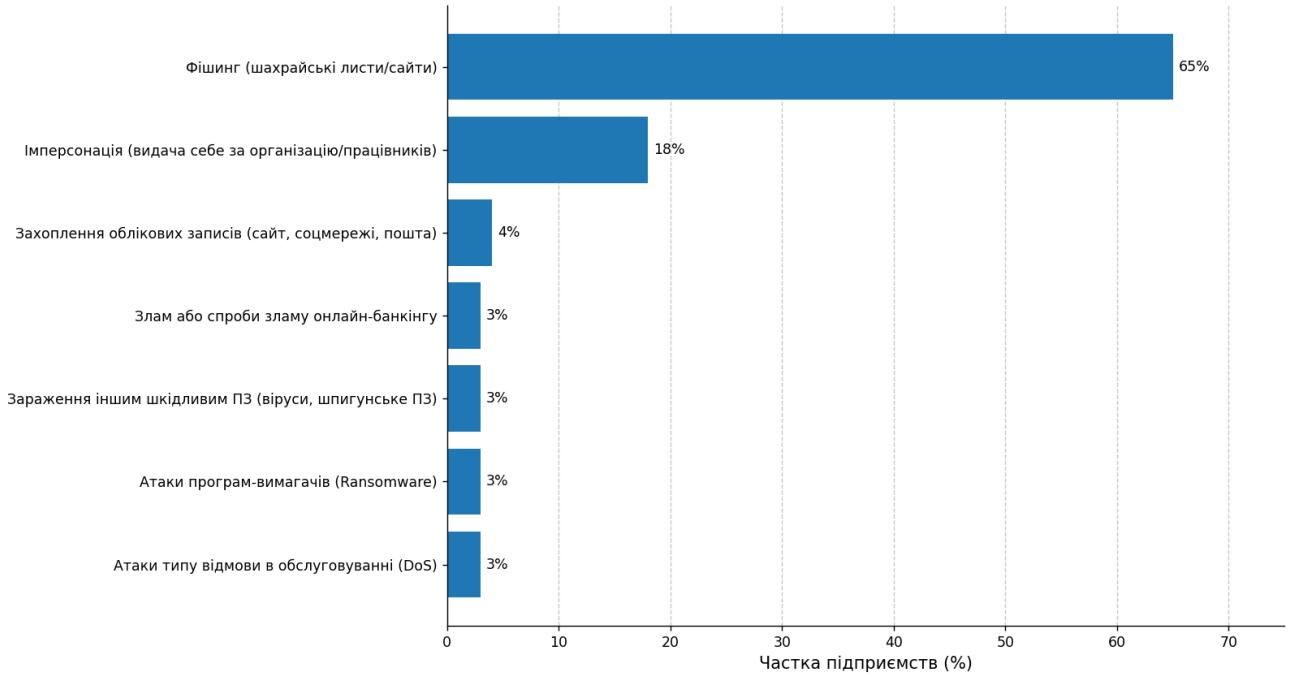

Рис 1.3 Основні типи кіберзлочинів серед підприємств у 2025 році (зроблено на основі таблиці [1])

Як видно з рисунка 1.3, переважну більшість кіберінцидентів серед підприємств становлять фішингові атаки, частка яких сягає 65%. Це свідчить про те, що методи соціальної інженерії залишаються одним із найефективніших інструментів зловмисників, оскільки орієнтовані насамперед на використання людського фактора, а не технічних недоліків інформаційної системи.

На другому місці за поширеністю знаходяться атаки, пов'язані з імперсонацією (видачею себе за організацію або її працівників в електронному листуванні чи онлайн) - з ними стикаються 18% підприємств. Значно рідше фіксуються випадки несанкціонованого захоплення облікових записів вебсайтів, пошти чи соціальних мереж (4%), а також спроби зламу онлайн-банкінгу (3%). Атаки з використанням шкідливого програмного забезпечення, програм-вимагачів та атаки типу відмови в обслуговуванні становлять по 3% відповідно.

Отримані результати підтверджують, що сучасні кіберзагрози мають різну природу походження, механізми реалізації та рівень потенційної шкоди. У зв’язку з цим забезпечення належного рівня мережевої безпеки потребує комплексного

підходу, який поєднує технічні засоби захисту, організаційні заходи та постійний моніторинг мережевого середовища. Саме тому доцільно розглянути основні класифікаційні ознаки загроз та їх характерні особливості.

1.2 Характеристика найбільш поширених кіберзагроз

Аналіз статистичних даних, наведений у попередньому підрозділі, показав, що сучасні підприємства систематично зазнають впливу різних видів кіберінцидентів. Зазначені загрози відрізняються механізмом реалізації, рівнем складності та можливими наслідками для інформаційної інфраструктури. Частина атак спрямована на отримання несанкціонованого доступу до даних, інші - на порушення роботи сервісів, викрадення фінансової інформації або зупинення бізнес-процесів. У цьому підрозділі розглянуто найбільш поширені кіберзагрози для підприємств.

Фішингові атаки

Структуру основних різновидів фішингових атак наведено на рисунку 1.4. Залежно від способу реалізації та масштабу охоплення цілей фішинг доцільно поділяти на масовий і цільовий.

Масовий фішинг орієнтований на одночасне розсилання великої кількості однотипних повідомлень без попереднього аналізу конкретної жертви. Для доставки використовуються електронна пошта, SMS-повідомлення, месенджери, рекламні сервіси та підроблені вебресурси. Типовим сценарієм є надсилання повідомлення з посиланням на фальшиву сторінку авторизації або форму введення платіжних реквізитів. Ефективність такого підходу забезпечується масштабом розсилки та низькою собівартістю реалізації атаки.

До поширених форм масового фішингу належать email-фішинг, SMS-фішинг, фішингові повідомлення у месенджерах, а також перенаправлення користувачів на підроблені сайти, що імітують банки, поштові сервіси, державні портали або популярні онлайн-платформи.

Цільовий фішинг передбачає попередній збір інформації про конкретну особу, підрозділ або організацію. Для підвищення достовірності зловмисники використовують службові контакти, посади працівників, структуру компанії, відкриті джерела та дані попередніх витоків. На основі зібраної інформації формуються персоналізовані повідомлення, які стилізуються під офіційне листування керівництва, партнерів або внутрішніх служб.

До основних різновидів цільового фішингу належать spear phishing, whaling, Business Email Compromise та clone phishing. Такі атаки спрямовані на викрадення облікових даних, компрометацію корпоративної пошти, зміну фінансових реквізитів, ініціювання несанкціонованих платежів або отримання доступу до внутрішніх інформаційних ресурсів.

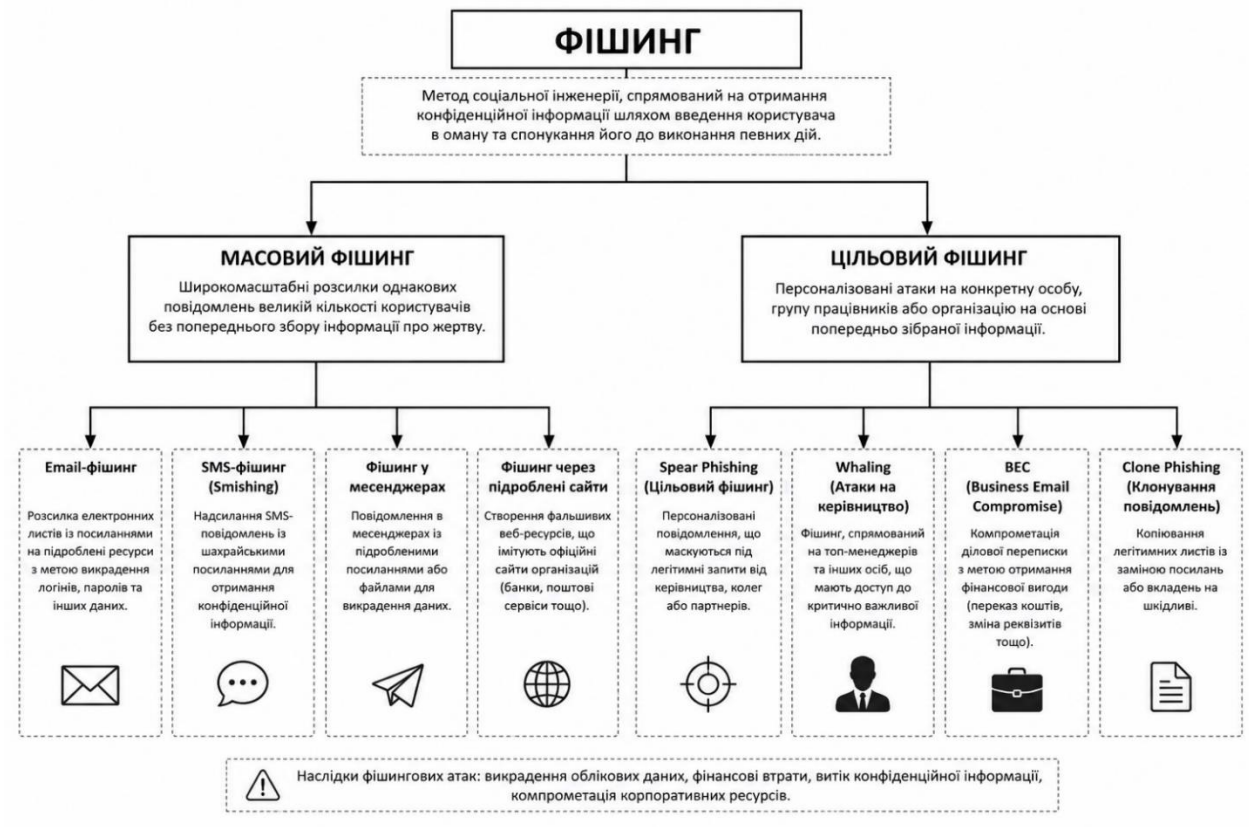

Рис 1.4 Структура фішингових атак

Практика кіберінцидентів свідчить, що масовий фішинг використовується для широкого первинного охоплення потенційних жертв, тоді як цільовий застосовується для атак на підприємства, фінансові структури та працівників із

розширеними правами доступу. З огляду на це засоби захисту мають включати технічну фільтрацію повідомлень, багатофакторну автентифікацію, контроль доменів, перевірку вкладень і постійне навчання персоналу.

Несанкціонований доступ до облікових записів і сервісів

Структуру основних способів отримання несанкціонованого доступу до облікових записів і цифрових сервісів наведено на рисунку 1.5. Зазначений тип кіберінцидентів належить до найбільш небезпечних, оскільки дозволяє зловмиснику використовувати легітимний обліковий запис без прямого порушення периметра інформаційної системи. Основною ціллю таких атак є корпоративна електронна пошта, внутрішні інформаційні системи, хмарні платформи, VPN-сервіси, системи віддаленого доступу та облікові записи працівників із розширеними правами. Після отримання доступу зловмисник може діяти від імені користувача в межах наданих йому повноважень.

Найпоширенішими методами компрометації є підбір паролів, використання даних із попередніх витоків, фішингові атаки, перехоплення одноразових кодів підтвердження, зараження пристрою шкідливим програмним забезпеченням, а також автоматизовані перевірки повторно використаних паролів на різних сервісах.

Після успішного входу можливе викрадення документів, зміна налаштувань безпеки, створення нових облікових записів, розсилання шкідливих повідомлень від імені компанії, несанкціоновані фінансові операції або використання скомпрометованого акаунта як точки подальшого проникнення у внутрішню мережу.

Зниження ризику досягається застосуванням багатофакторної автентифікації, використанням унікальних складних паролів, обмеженням прав доступу, контролем підозрілих входів, журналюванням подій безпеки та впровадженням централізованих систем керування цифровою ідентичністю [3].

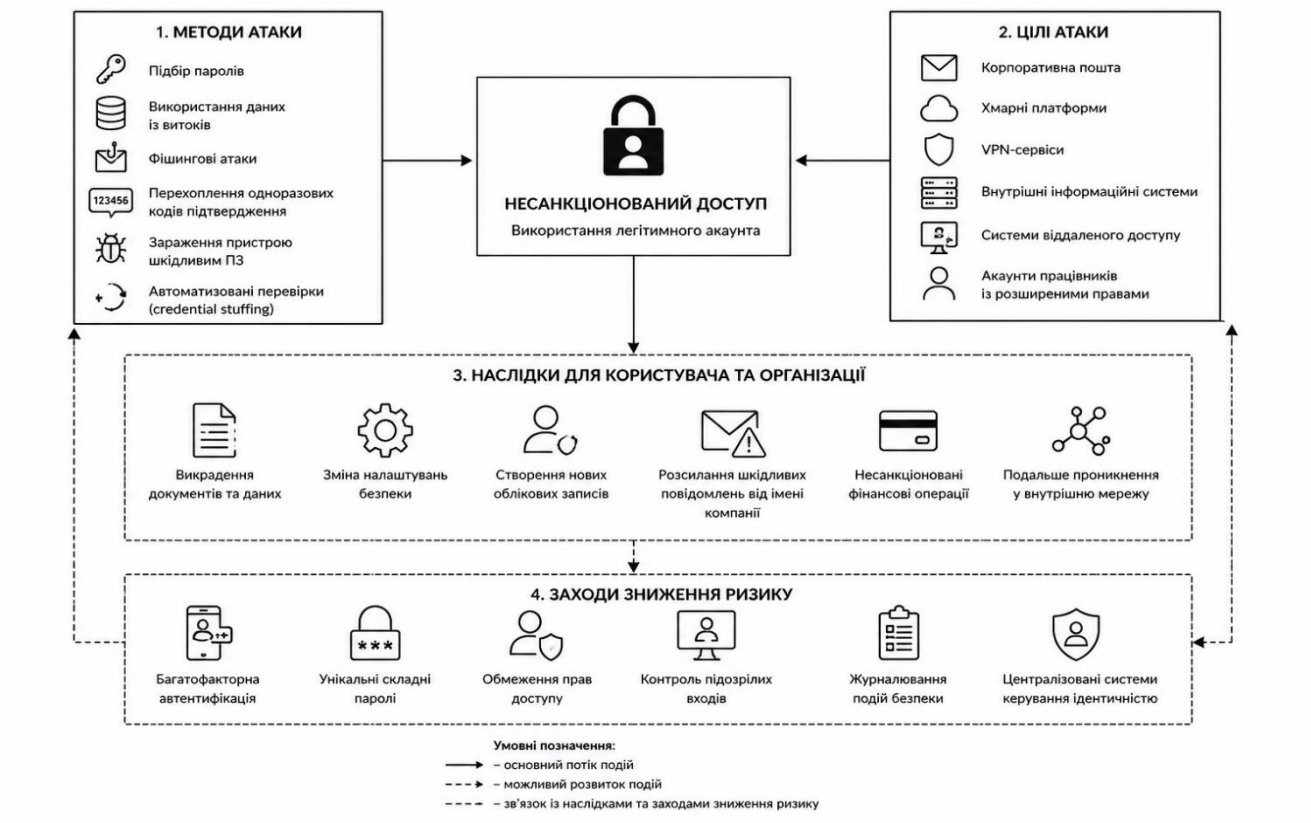

Рис 1.5 Схема несанкціонованого доступу до облікових записів

Програми-вимагачі

Програми-вимагачі належать до найбільш небезпечних сучасних кіберзагроз, оскільки їх застосування може одночасно порушити доступність інформації, стабільність функціонування інформаційної системи та економічну діяльність підприємства. Основний механізм дії таких шкідливих програм полягає у блокуванні або криптографічному шифруванні файлів, баз даних, робочих станцій чи серверного обладнання з подальшим висуненням вимоги сплати викупу за відновлення доступу до даних [4].

Особливістю сучасних програм-вимагачів є комплексний характер впливу. У багатьох випадках зловмисники не обмежуються лише шифруванням інформації, а попередньо здійснюють копіювання конфіденційних документів, комерційних даних або персональної інформації користувачів. Надалі такі відомості використовуються як додатковий інструмент тиску шляхом погроз їх публічного розголошення або продажу третім особам. Механізм реалізації атаки наведено на рисунку 1.6.

Для підприємств наслідки подібних інцидентів можуть бути критичними. Зокрема, виникає ризик повного або часткового зупинення виробничих і управлінських процесів, втрати доступу до електронного документообігу, порушення взаємодії з клієнтами та партнерами, а також суттєвих фінансових витрат на відновлення працездатності інфраструктури. Додатково значної шкоди завдають репутаційні втрати та зниження довіри до організації з боку контрагентів.

Ефективний захист від програм-вимагачів потребує комплексного підходу, що включає регулярне резервне копіювання даних, своєчасне оновлення програмного забезпечення, сегментацію мережі, використання антивірусних засобів і систем виявлення вторгнень, а також підвищення рівня обізнаності персоналу щодо методів соціальної інженерії та фішингових атак [4].

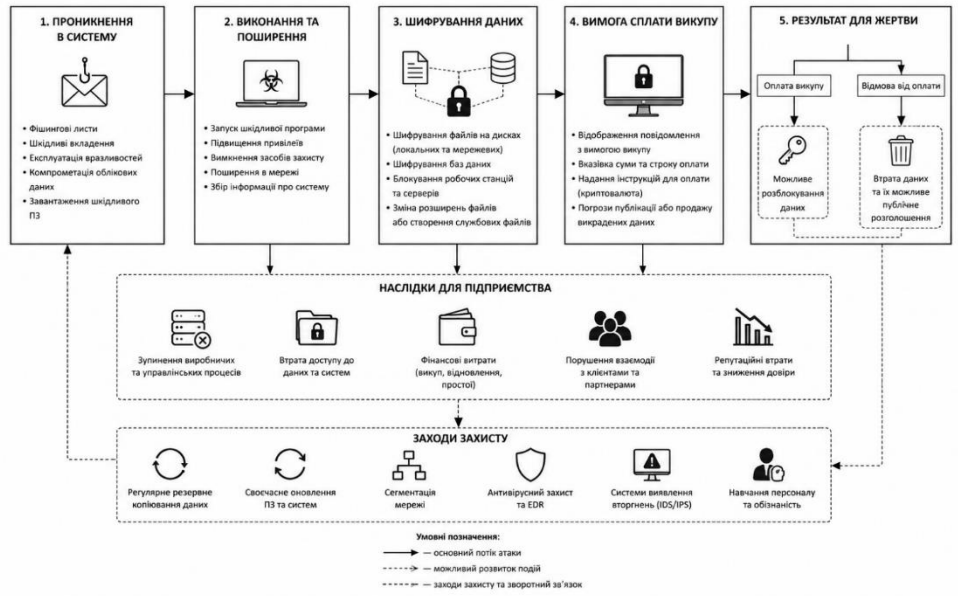

Рис 1.6 Реалізація програм-вимагачів

Шкідливе програмне забезпечення

Шкідливе програмне забезпечення охоплює широкий клас програмних засобів, створених з метою порушення штатної роботи інформаційних систем, отримання несанкціонованого доступу до ресурсів, викрадення даних або прихованого контролю над пристроєм користувача [5]. До цієї категорії належать комп'ютерні віруси, мережеві черв'яки, троянські програми, шпигунське

програмне забезпечення, бекдори, кейлогери, ботнет-клієнти та інші різновиди шкідливого коду.

Основні способи поширення шкідливого програмного забезпечення наведено на рисунку 1.7. Найбільш поширеними каналами зараження є вкладення електронної пошти, фішингові посилання, використання неліцензійного програмного забезпечення, завантаження файлів із ненадійних ресурсів, а також експлуатація вразливостей операційної системи чи прикладних програм. У корпоративному середовищі додатковим фактором ризику є використання зовнішніх носіїв інформації та недостатній контроль дій користувачів.

Особливу небезпеку становить прихований режим роботи шкідливого програмного забезпечення. Зараження може тривалий час залишатися непоміченим, одночасно виконуючи збір облікових даних, копіювання службових документів, передачу інформації третім особам або підготовку подальших атак. Окремі види malware використовуються як початковий етап складніших кіберінцидентів, зокрема програм-вимагачів або несанкціонованого проникнення до внутрішньої мережі підприємства.

Для зниження ризику зараження доцільно застосовувати комплекс превентивних заходів, серед яких регулярне оновлення програмного забезпечення, використання антивірусних систем, обмеження прав користувачів, контроль мережевої активності та підвищення рівня кібергігієни персоналу [5].

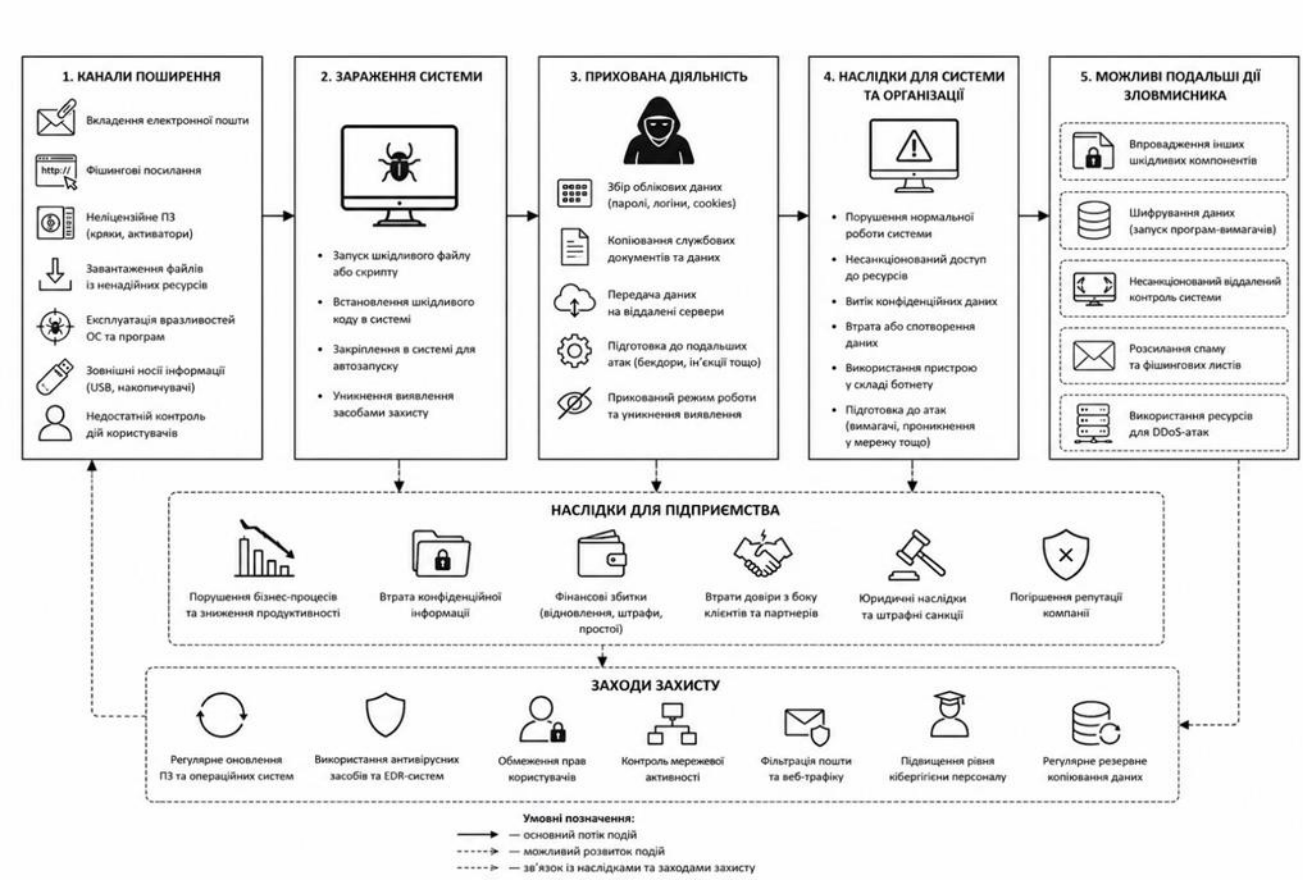

Рис 1.7 Механізм дії шкідливого програмного забезпечення

Атаки типу відмова в обслуговуванні

Атаки типу відмови в обслуговуванні спрямовані на порушення доступності інформаційних ресурсів для легітимних користувачів. Основний механізм дії полягає у створенні надмірного навантаження на серверне обладнання, мережеву інфраструктуру або прикладні сервіси, унаслідок чого система втрачає здатність своєчасно обробляти запити або повністю припиняє функціонування [6].

Структуру реалізації атак типу відмови в обслуговуванні наведено на рисунку 1.8. Більш складною формою таких інцидентів є розподілені атаки DDoS, за яких шкідливий трафік генерується одночасно з великої кількості пристроїв, попередньо скомпрометованих та об'єднаних у ботмережу. Використання численних джерел запитів істотно ускладнює виявлення атаки та блокування її джерела.

Об'єктами впливу найчастіше стають вебсайти підприємств, електронні торговельні майданчики, банківські сервіси, системи онлайн-обслуговування

клієнтів, корпоративні портали та інші ресурси, критично залежні від безперервного доступу користувачів.

Наслідками подібних інцидентів можуть бути простої бізнес-процесів, втрата прибутку, погіршення репутації, зниження рівня довіри клієнтів, а також додаткові витрати на відновлення працездатності інфраструктури.

Ефективна протидія атакам типу відмови в обслуговуванні потребує комплексного підходу, що включає використання систем фільтрації трафіку, балансування навантаження, резервування серверних ресурсів, застосування спеціалізованих DDoS-захисних сервісів, а також постійний моніторинг мережевої активності для своєчасного виявлення аномальної поведінки [6].

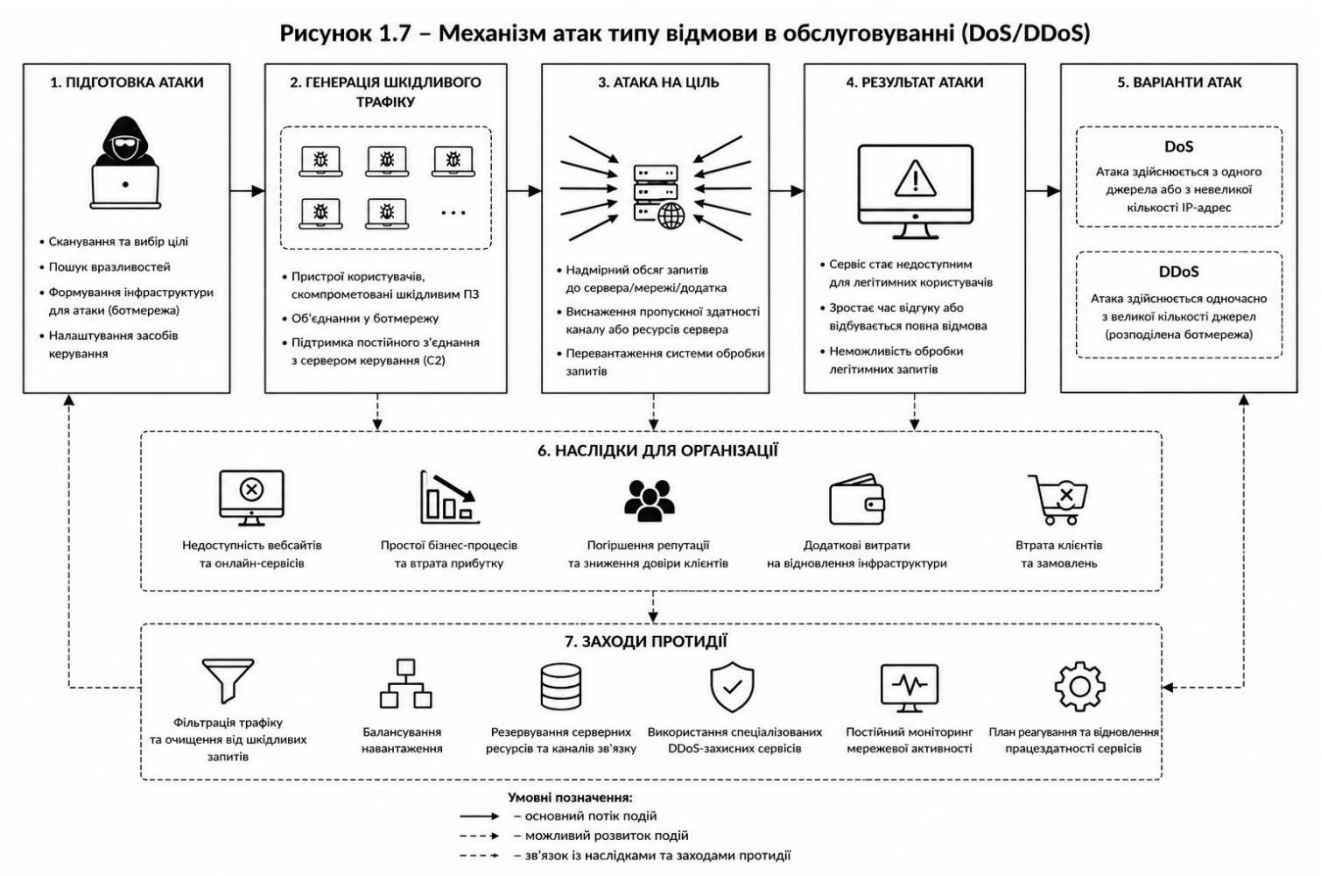

Рис 1.8 Схема DDOS атак

1.3 Методи протидії кіберзагрозам та моніторингу мережної безпеки

Ефективна протидія сучасним кіберзагрозам потребує комплексного підходу, який поєднує організаційні, програмно-технічні та аналітичні заходи. Використання лише одного засобу захисту, не забезпечує належного рівня безпеки інформаційної системи. Саме тому сучасні підприємства впроваджують багаторівневі моделі захисту, що охоплюють запобігання інцидентам, своєчасне виявлення підозрілої активності та оперативне реагування на загрози.

Захист від фішингових атак

Одним із ключових напрямів протидії фішинговим атакам є підвищення рівня обізнаності користувачів щодо методів соціальної інженерії. Практика свідчить, що навіть технічно захищені інформаційні системи можуть залишатися вразливими за умови недостатньої підготовки персоналу, оскільки значна частина фішингових інцидентів ґрунтується саме на помилкових діях користувача. Зловмисники використовують психологічний тиск, терміновість, імітацію авторитетних організацій або зацікавлення жертви для отримання необхідної інформації.

Систематичні тренінги з кібергігієни та симуляції фішингових атак ефективно мінімізують ризики соціальної інженерії. Завдяки цьому персонал здобуває стійкі навички самостійної ідентифікації загроз: верифікації адрес відправників, доменів, гіперпосилань та вкладених файлів [2].

Важливим елементом організаційного захисту є затверджений регламент реагування на інциденти: оперативне сповіщення служби інформаційної безпеки, заборона взаємодії з невідомими вкладеннями та обов'язкова верифікація запитів щодо передачі конфіденційних даних. Технічний рівень протидії забезпечується фільтрацією поштового трафіку, поведінковим аналізом вкладень в ізольованому середовищі та превентивним блокуванням шкідливих URL-адрес. Для унеможливлення поштового спуфінгу застосовуються протоколи автентифікації доменів SPF, DKIM і DMARC. Водночас критичним рубежем захисту, що нівелює наслідки успішної компрометації паролів, виступає впровадження багатофакторної

автентифікації. Відповідно, надійний захист від соціальної інженерії вимагає синергії апаратних рішень та кіберграмотності персоналу.

Захист облікових записів та цифрової ідентичності

Компрометація облікових записів часто слугує початковим вектором атаки для глибокого проникнення в корпоративну інфраструктуру. Базовим рубежем захисту цифрової ідентичності є багатофакторна автентифікація, яка унеможливлює доступ зловмисника виключно за рахунок викраденого пароля [3]. Водночас архітектура безпеки вимагає використання корпоративних менеджерів паролів та централізованого адміністрування доступу на основі принципу мінімальних привілеїв.

Для своєчасного виявлення інцидентів критичним є безперервний моніторинг та логування автентифікаційних подій: фіксація аномальних геолокацій, нетипових часових патернів та спроб брутфорсу. Лише інтеграція політик доступу з системами автоматичного моніторингу дозволяє мінімізувати ризики несанкціонованого доступу.

Протидія програмам-вимагачам

Специфіка атак програм-вимагачів полягає у миттєвій паралізації бізнес-процесів через криптографічне блокування даних. Тому сучасна парадигма захисту зміщує фокус із простого запобігання на забезпечення загальної кіберстійкості інфраструктури. Фундаментом такої стійкості є наявність ізольованих резервних копій, що гарантує відновлення працездатності системи без взаємодії зі зловмисниками [4]. На архітектурному рівні критично важливою є сегментація мережі. Розподіл інфраструктури на ізольовані VLAN унеможливлює безперешкодне горизонтальне переміщення шкідливого коду між робочими станціями та критичними серверами. Цей підхід доповнюється строгим дотриманням принципу найменших привілеїв та своєчасним патч-менеджментом, що звужує поверхню атаки та мінімізує радіус ураження у разі компрометації одного вузла.

Оскільки початковим вектором найчастіше виступає фішинг, превентивні заходи обов'язково мають працювати в синергії з інструментами активного виявлення. Впровадження EDR-платформ та систем моніторингу мережевих аномалій дозволяє ідентифікувати атаку на ранніх стадіях за специфічними поведінковими маркерами: масовим перейменуванням файлів, сплесками навантаження через процеси шифрування або підозрілим трафіком до командних серверів.

Захист від шкідливого програмного забезпечення

Для протидії шкідливому програмному забезпеченню використовуються комплексні засоби технічного та організаційного захисту, спрямовані на виявлення, блокування та запобігання проникненню небезпечних компонентів до інформаційної системи. До категорії malware належать віруси, троянські програми, шпигунське програмне забезпечення, бекдори, кейлогери, ботнет-клієнти та інші шкідливі модулі, здатні порушувати роботу пристроїв, викрадати інформацію або надавати зловмиснику прихований доступ до системи.

Базовим інструментом захисту залишаються антивірусні системи, що здійснюють перевірку файлів, оперативної пам’яті, електронної пошти та мережевого трафіку на наявність відомих ознак шкідливого коду. Ефективність такого захисту значною мірою залежить від своєчасного оновлення сигнатурних баз, оскільки нові модифікації шкідливого програмного забезпечення з’являються постійно [5].

У сучасних умовах дедалі більшого значення набувають EDR-рішення, які дозволяють не лише виявляти відомі загрози, а й аналізувати поведінку процесів у системі. Такі засоби здатні фіксувати нетипові дії програм, спроби зміни системних файлів, несанкціонований доступ до реєстру, запуск прихованих процесів або підозрілу мережеву активність. Використання поведінкового аналізу дає можливість виявляти нові або модифіковані загрози, що ще не внесені до сигнатурних баз.

Важливим напрямом захисту є контроль запуску програмного забезпечення. Доцільним є використання механізмів білого списку програм, за яких запуск дозволяється лише перевіреним застосункам, затвердженим адміністратором системи. Такий підхід істотно знижує ризик випадкового запуску шкідливих файлів користувачем.

Додатковим профілактичним заходом є заборона використання неліцензійного або піратського програмного забезпечення, оскільки подібні продукти часто містять приховані шкідливі компоненти або бекдори. Також важливим є контроль зовнішніх носіїв інформації, зокрема USB-пристроїв, карт пам’яті та переносних накопичувачів, через які malware може потрапляти до внутрішньої мережі підприємства.

Окрему увагу слід приділяти обмеженню прав користувачів на встановлення програм без дозволу адміністратора. Надмірні привілеї створюють додаткові ризики, оскільки дозволяють запускати стороннє програмне забезпечення без перевірки з боку служби безпеки. Використання принципу мінімально необхідних прав значно зменшує можливості поширення шкідливого коду.

У результаті можна зробити висновок, що ефективна протидія malware потребує поєднання антивірусного захисту, сучасних систем моніторингу кінцевих пристроїв, контролю програмного середовища та дотримання користувачами базових правил цифрової безпеки. Лише комплексний підхід дозволяє забезпечити належний рівень захисту корпоративної інфраструктури від сучасних шкідливих загроз.

Протидія атакам типу відмови в обслуговуванні

Забезпечення високої доступності інформаційних ресурсів в умовах DoS та DDoS-атак вимагає ешелонованої архітектури захисту, здатної витримувати пікові зовнішні навантаження. Базовим рівнем такої відмовостійкості є розподілення вхідного трафіку за допомогою систем балансування, що унеможливлює відмову єдиної точки та дозволяє динамічно підключати резервні обчислювальні потужності.

Активна протидія базується на глибокій фільтрації трафіку. Сучасні апаратні та програмні міжмережеві екрани аналізують структуру пакетів, географію джерел та поведінкові патерни, відсікаючи шкідливі запити ще на підступах до цільового сервера [6]. Для нейтралізації автоматизованих ботнетів цей механізм доповнюється політиками обмеження частоти запитів, що тимчасово блокують або уповільнюють з'єднання при перевищенні допустимих порогових значень. У разі масштабних волюметричних атак, які перевищують пропускну здатність каналів підприємства, трафік маршрутизується через спеціалізовані хмарні центри очищення, що пропускають у внутрішню мережу виключно легітимні з'єднання.

Проте ефективність усіх зазначених інструментів критично залежить від швидкості виявлення інциденту. Саме безперервний моніторинг телеметрії - фіксація аномальних сплесків з'єднань, нетипової утилізації каналу чи повторюваних запитів - дозволяє своєчасно активувати механізми реагування. Отже, комплексна протидія DDoS-атакам неможлива без інтелектуальних систем безперервного виявлення мережевих аномалій, що підтверджує практичну цінність та актуальність розроблюваного застосунку.

Роль систем моніторингу мережної безпеки

Сучасні стратегії кіберзахисту базуються на переході від пасивного реагування до проактивного моніторингу мережевого середовища. На відміну від сигнатурних засобів, системи моніторингу забезпечують безперервну видимість інфраструктури, дозволяючи ідентифікувати загрози на етапах розвідки або початкового проникнення. Функціонування таких рішень базується на централізованому зборі та кореляції логів із гетерогенних джерел: від мережевого обладнання та міжмережевих екранів до антивірусних платформ і прикладних сервісів.

Пріоритетним напрямом розвитку є впровадження модулів автоматичного виявлення аномалій на основі статистичних методів та алгоритмів машинного навчання. Такий підхід дозволяє виявляти нетипові часові патерни, аномальні маршрути трафіку та сплески мережевого навантаження, які характерні для 0-day

атак або модифікованих шкідливих програм, чиї сигнатури ще не внесені до баз даних. Крім зовнішніх векторів, аналіз поведінкових шаблонів ефективно виявляє внутрішні загрози: несанкціоновану ексфільтрацію даних, компрометацію службових акаунтів або звернення до критичних ресурсів поза межами службових повноважень.

Таким чином, інтеграція інтелектуальних систем моніторингу в архітектуру безпеки є необхідною умовою для мінімізації ризиків та оперативного реагування на інциденти. Це підтверджує актуальність розробки спеціалізованих застосунків для автоматизованого виявлення аномалій у мережевому трафіку.

1.4 Актуальні тенденції розвитку засобів мережної безпеки

Стрімка цифровізація економічних процесів, поширення хмарних технологій, дистанційної роботи, мобільного доступу та інтеграції розподілених сервісів суттєво змінили підходи до забезпечення мережної безпеки. Якщо раніше основна увага приділялася захисту периметра корпоративної мережі, то в сучасних умовах цього вже недостатньо. Інформаційні ресурси підприємств розміщуються не лише у внутрішньому сегменті мережі, а й у хмарних середовищах, віддалених офісах, мобільних пристроях та зовнішніх сервісах.

Одночасно зростає складність і масштабність кіберзагроз. За даними сучасних досліджень, зловмисники активно використовують автоматизовані інструменти, штучний інтелект, масові кампанії соціальної інженерії, програми-вимагачі та багатовекторні атаки, спрямовані одночасно на мережеву інфраструктуру, користувачів і прикладні сервіси [7], [8]. Це свідчить про перехід кіберзлочинності від поодиноких інцидентів до системної високотехнологічної діяльності.

У таких умовах традиційні засоби захисту, зокрема класичні антивірусні рішення та статичні міжмережеві екрани, вже не забезпечують достатнього рівня протидії сучасним загрозам. Для актуальних атак характерні швидка зміна інструментів, використання легітимних сервісів, прихована присутність у

мережевому середовищі та поступова ескалація привілеїв. Унаслідок цього поширюється концепція багаторівневого захисту, що передбачає поєднання різних механізмів безпеки в єдиній системі контролю та реагування.

Значного поширення набуває модель Zero Trust, відповідно до якої жоден користувач, пристрій або сервіс не вважається довіреним за замовчуванням. Доступ до ресурсів надається лише після перевірки цифрової ідентичності, контексту входу, рівня ризику та відповідності встановленим політикам безпеки. Такий підхід особливо актуальний для організацій із гібридною інфраструктурою та великою кількістю віддалених користувачів [3].

Важливу роль відіграють системи централізованого моніторингу та аналізу подій безпеки. Сучасні підприємства впроваджують SIEM-платформи, системи виявлення вторгнень, EDR/XDR-рішення та засоби кореляції журналів подій. Такі технології забезпечують збір інформації з різних джерел, виявлення підозрілої активності та оперативне реагування на інциденти. За результатами сучасних досліджень, інтегровані центри безпеки та централізований моніторинг стають одним із ключових напрямів розвитку корпоративного захисту [7].

Окремим напрямом є використання алгоритмів машинного навчання для автоматичного виявлення аномалій у мережевому трафіку та поведінці користувачів. На відміну від сигнатурного підходу, такі системи здатні виявляти нові або модифіковані загрози, які раніше не були відомі. Це особливо важливо в умовах постійної появи нових технік обходу традиційних засобів захисту.

Також поширюється автоматизація реагування на інциденти. Використання SOAR-рішень дозволяє автоматично запускати сценарії перевірки, ізоляції пристроїв, блокування облікових записів та формування повідомлень для служби безпеки без участі оператора. Це скорочує час реагування та знижує навантаження на персонал.

Отже, сучасний розвиток засобів мережної безпеки характеризується переходом від пасивного захисту до інтелектуальних систем безперервного контролю, аналітики та автоматизованого реагування. Основна увага приділяється моніторингу подій, виявленню аномалій, перевірці цифрової довіри та швидкому

усуненню інцидентів. Саме тому актуальним є аналіз існуючих систем моніторингу мережної безпеки та принципів їх побудови.

РОЗДІЛ 2 ТЕОРІЯ РОЗРОБКИ ЗАСТОСУНКУ МОНІТОРИНГУ МЕРЕЖНОЇ БЕЗПЕКИ

2.1 Аналіз існуючих систем моніторингу мережної безпек

У попередньому розділі було розглянуто сучасні мережеві загрози та способи їх реалізації. Практика показує, що застосування лише традиційних засобів захисту часто є недостатнім. З цієї причини для забезпечення належного рівня безпеки використовуються спеціалізовані системи моніторингу, аналізу подій та автоматизованого реагування на інциденти.

Такі програмні засоби призначені для безперервного контролю стану мережевої інфраструктури, збору журналів подій, аналізу мережевого трафіку, виявлення підозрілої активності та оперативного інформування адміністратора про потенційні загрози. Їх використання дає змогу своєчасно реагувати на спроби несанкціонованого доступу, шкідливу активність, аномальні зміни навантаження та інші інциденти інформаційної безпеки.

На сучасному етапі розвитку інформаційних технологій існує значна кількість програмних продуктів, що реалізують функції моніторингу мережної безпеки. Залежно від призначення та архітектури їх доцільно поділити на такі основні групи: SIEM-системи, IDS/IPS-рішення, системи мережевої аналітики та універсальні засоби моніторингу. До поширених представників належать Splunk, IBM QRadar, Wazuh, Snort, Suricata, Zeek та Zabbix. Кожна з наведених систем має власні переваги, функціональні можливості та обмеження, які доцільно розглянути детальніше.

SIEM-системи

Основним призначенням таких платформ є централізований збір, зберігання та аналіз подій безпеки, що надходять з різних компонентів інформаційної інфраструктури: серверів, мережевого обладнання, робочих станцій, міжмережевих екранів, антивірусних засобів, хмарних сервісів і прикладного програмного забезпечення [7].

Принцип роботи SIEM-систем полягає в агрегуванні журналів подій з багатьох джерел у єдиному середовищі з подальшою кореляцією отриманих даних. Це дозволяє виявляти пов’язані між собою інциденти, які окремо можуть виглядати несуттєвими, але в сукупності свідчать про наявність атаки або підготовчих дій зловмисника. Наприклад, численні невдалі спроби входу до облікового запису, подальше успішне підключення та нетипове копіювання файлів можуть розглядатися як ознаки компрометації акаунта.

Сучасні SIEM-платформи підтримують автоматичне формування сповіщень у режимі реального часу, побудову інформаційних панелей, пошук аномалій у поведінці користувачів та інтеграцію із засобами автоматизованого реагування на інциденти. Це скорочує час виявлення загроз і підвищує ефективність роботи фахівців з кібербезпеки. Проте, впровадження даного рішення супроводжується значними фінансовими витратами, складністю налаштування та високими вимогами до кваліфікації персоналу. Необхідність постійного оновлення правил кореляції, аналіз великих обсягів даних та відсікання хибнопозитивних спрацювань вимагають залучення досвідчених аналітиків безпеки, що створює додаткове навантаження на бюджет організації.

Серед поширених представників цього класу рішень слід виділити IBM QRadar, Splunk Enterprise Security, Microsoft Sentinel, Elastic Security та інші платформи корпоративного рівня. Їх перевагами є масштабованість, широкі можливості інтеграції з іншими системами та високий рівень аналітичної обробки даних.

IDS/IPS

IDS/IPS-системи призначені для виявлення спроб несанкціонованого доступу, експлуатації вразливостей, шкідливої активності та інших ознак компрометації інформаційної інфраструктури. На відміну від класичних засобів захисту, що переважно виконують функції обмеження доступу, такі системи орієнтовані на безперервний аналіз подій безпеки та своєчасне реагування на загрози [8].

Вони реалізують функцію виявлення підозрілої активності шляхом контролю мережевого трафіку, системних журналів або поведінки окремих вузлів. Основним результатом їх роботи є формування сповіщень для адміністратора безпеки про можливий інцидент. IPS-системи, у свою чергу, поєднують можливості виявлення з механізмами активного впливу на атаку: блокуванням мережевого з'єднання, відхиленням шкідливих пакетів, тимчасовим обмеженням доступу або автоматичним застосуванням політик захисту [8].

Залежно від місця розгортання розрізняють мережеві та хостові рішення. Мережеві IDS/IPS-системи контролюють трафік у ключових точках мережі та дозволяють виявляти сканування портів, спроби експлуатації сервісів, DDoS-активність і аномальні з'єднання між вузлами. Хостові рішення встановлюються безпосередньо на сервери або робочі станції та орієнтовані на аналіз змін у файловій системі, процесах, системних подіях і локальних журналах. У практичному застосуванні IDS/IPS використовують два основні підходи до виявлення загроз. Перший базується на сигнатурному аналізі, коли система порівнює трафік або події з відомими шаблонами атак і правилами виявлення. Другий ґрунтується на аналізі аномалій та передбачає фіксацію відхилень від типової поведінки мережі, користувачів або окремих вузлів.

Системи мережевої аналітики

Системи мережевої аналітики призначені для глибокого аналізу мережевого трафіку, виявлення прихованих загроз та контролю стану інформаційної інфраструктури. На відміну від класичних засобів захисту, що виконують переважно функції фільтрації або обмеження доступу, такі рішення орієнтовані на виявлення нетипової поведінки користувачів, пристроїв і сервісів у мережевому середовищі [8].

Основою їх роботи є безперервний збір телеметричних даних: мережевих потоків NetFlow, IPFIX, sFlow, журналів подій, DNS-запитів, інформації про сесії, а в окремих випадках - повних копій пакетів. Отримані дані проходять автоматизовану обробку, кореляцію та подальший аналіз із використанням

статистичних методів, сигнатурних правил і поведінкових моделей. Це дозволяє виявляти аномальні підключення, приховане переміщення зловмисника мережею, спроби витоку інформації, командно-контрольний трафік та інші ознаки компрометації [7], [8].

Сучасні системи мережевої аналітики часто реалізуються у форматах Network Traffic Analysis або Network Detection and Response. Їх характерною особливістю є здатність формувати базову модель нормальної поведінки мережі та автоматично визначати відхилення від типових шаблонів роботи. Це дає змогу виявляти загрози, які можуть залишатися непоміченими для класичних сигнатурних засобів, зокрема нові різновиди атак, внутрішні інциденти або компрометацію легітимних облікових записів.

Практичне значення таких рішень зростає в умовах використання хмарних сервісів, дистанційної роботи, мобільного доступу та гібридної IT-інфраструктури. У подібних середовищах периметровий захист не забезпечує повної видимості мережевих подій, тоді як аналітичні системи дозволяють контролювати як зовнішній трафік, так і внутрішні взаємодії між сегментами мережі.

До переваг систем мережевої аналітики належать висока інформативність, можливість ретроспективного розслідування інцидентів, раннє виявлення складних атак та інтеграція з платформами SIEM, SOAR, IDS/IPS і засобами реагування. Водночас їх впровадження потребує достатніх обчислювальних ресурсів, продуманої архітектури збору даних і коректного налаштування механізмів обробки подій.

Універсальні засоби моніторингу

Універсальні засоби моніторингу призначені для централізованого контролю стану інформаційної інфраструктури, мережевих сервісів, серверного обладнання, робочих станцій і прикладних систем. На відміну від спеціалізованих рішень, орієнтованих переважно на виявлення атак, такі системи забезпечують комплексний контроль працездатності, продуктивності, доступності ресурсів і подій безпеки [8].

Основним завданням універсальних платформ є безперервний збір технічних показників: завантаження процесора, використання оперативної пам’яті, стану дискових підсистем, мережевої активності, доступності вузлів, часу відповіді сервісів, змін конфігурації та системних журналів. Отримані дані агрегуються у єдиному середовищі та відображаються у вигляді графіків, панелей керування, журналів подій і автоматичних сповіщень.

Практична цінність таких систем полягає у можливості раннього виявлення відхилень від штатного режиму роботи. Різке зростання навантаження на мережевий канал, нестандартне використання процесорних ресурсів, часті помилки сервісів або неочікуване вимкнення вузлів можуть свідчити як про технічну несправність, так і про початок кіберінциденту. З цієї причини універсальні засоби моніторингу часто використовуються як джерело даних для подальшого аналізу загроз і реагування на інциденти.

До найбільш поширених представників цього класу належать Zabbix, Nagios, Prometheus, Grafana та PRTG Network Monitor. Такі рішення широко застосовуються в корпоративному середовищі завдяки масштабованості, підтримці різних протоколів збору даних, гнучким механізмам сповіщення та можливості інтеграції з іншими системами безпеки.

Ефективність їх використання значною мірою залежить від правильного налаштування. Необхідно визначити перелік ключових метрик, джерела телеметрії та порогові значення сповіщень. Надлишкова кількість подій або некоректно налаштовані правила можуть перевантажувати адміністратора та ускладнювати реагування на реальні інциденти.

Отже, універсальні засоби моніторингу забезпечують постійний контроль стану інфраструктури та можуть використовуватися як основа для виявлення аномальної активності. Такий підхід доцільно врахувати під час розробки власного застосунку моніторингу мережної безпеки.

2.2 Методи виявлення аномалій у мережевому трафіку

Проведений у попередньому підрозділі аналіз існуючих систем моніторингу мережної безпеки показав, що ефективність таких рішень значною мірою залежить не лише від функціональних можливостей програмного забезпечення, а й від застосованих методів аналізу мережевої активності. Саме алгоритми виявлення підозрілих подій визначають швидкість реагування системи, точність розпізнавання загроз та здатність виявляти нові або приховані типи атак.

У сучасних умовах мережевий трафік характеризується значною інтенсивністю, різноманітністю протоколів обміну даними та постійною зміною поведінки користувачів і сервісів. Унаслідок цього традиційні засоби ручного контролю вже не забезпечують належного рівня спостереження за інфраструктурою. Виникає потреба у використанні автоматизованих підходів, здатних оперативно виявляти відхилення від нормального режиму функціонування мережі.

Під аномалією в мережевому трафіку доцільно розуміти будь-яку нетипову або потенційно небезпечну активність, що відрізняється від звичайних параметрів роботи інформаційної системи. Такими ознаками можуть бути різке зростання кількості з’єднань, повторювані запити з одного джерела, використання нетипових портів, нехарактерні часові інтервали доступу, передавання надмірних обсягів даних або незвична поведінка окремих вузлів мережі.

На практиці для виявлення подібних відхилень застосовуються різні методи аналізу, що відрізняються принципом роботи, точністю, складністю реалізації та вимогами до ресурсів. До найбільш поширених належать сигнатурний, статистичний, поведінковий підходи, а також сучасні методи машинного навчання.

Сигнатурний підхід

Сигнатурний підхід базується на порівнянні поточного мережевого трафіку з базою відомих шаблонів атак, шкідливих дій або характерних ознак компрометації

системи. Якщо виявляється збіг із наявним правилом, система формує попередження або автоматично блокує підозрілу активність.

Основною перевагою такого методу є висока точність щодо вже відомих загроз, швидкість обробки подій та відносна простота впровадження. Саме тому сигнатурний аналіз широко використовується в IDS/IPS-системах, міжмережевих екранах та антивірусних рішеннях. Разом із тим його недоліком є обмежена ефективність проти нових або модифікованих атак, сигнатури яких відсутні у базі даних.

Статистичний підхід

Статистичний метод передбачає аналіз кількісних характеристик мережевого трафіку та порівняння поточних показників із типовими значеннями. До таких параметрів можуть належати кількість пакетів за певний проміжок часу, середній розмір трафіку, частота звернень до сервісів, кількість активних сесій або співвідношення вхідного та вихідного трафіку.

У разі суттєвого відхилення від звичних значень система може визначити подію як потенційну аномалію. Наприклад, різке зростання кількості однотипних запитів може свідчити про DDoS-атаку або масове сканування мережі. Перевагою підходу є можливість виявлення невідомих загроз, однак недоліком виступає ймовірність хибних спрацювань під час легітимних змін навантаження.

Поведінковий аналіз

Поведінковий підхід ґрунтується на формуванні моделі нормальної активності користувачів, серверів, робочих станцій або мережевих сервісів. Система накопичує інформацію про типові часові інтервали роботи, маршрути доступу, характер використання ресурсів, обсяги передавання даних та інші параметри.

Після формування базової моделі будь-які значні відхилення можуть розглядатися як підозрілі. Наприклад, вхід користувача у нетиповий час, масове копіювання файлів або спроба доступу до ресурсів, що раніше не

використовувалися, можуть свідчити про компрометацію облікового запису. Перевагою методу є здатність виявляти внутрішні загрози та складні сценарії атак.

Машинне навчання

Сучасним напрямом розвитку систем кібербезпеки є використання алгоритмів машинного навчання для автоматичного аналізу великих масивів мережевих даних. Такі методи дозволяють знаходити приховані закономірності, класифікувати типи активності та виявляти аномалії без жорстко заданих правил.

Для цього можуть застосовуватися дерева рішень, методи кластеризації, нейронні мережі, алгоритми пошуку викидів та інші математичні моделі. Їх використання дозволяє адаптувати систему до змін мережевого середовища та виявляти нові загрози, які складно знайти сигнатурними засобами. Водночас ефективність таких моделей залежить від якості навчальних даних і правильності налаштування параметрів.

Кожен із розглянутих підходів має власні переваги та обмеження. Сигнатурний метод забезпечує високу точність щодо відомих атак, проте малоефективний проти нових загроз. Статистичний аналіз дозволяє знаходити нетипові навантаження, але може генерувати хибні спрацювання. Поведінкові моделі ефективні для виявлення внутрішніх інцидентів, однак потребують часу для формування базового профілю. Алгоритми машинного навчання є найбільш гнучкими, проте складнішими в реалізації.

У сучасних системах моніторингу найкращий є комбінований підхід, що поєднує переваги декількох методів одночасно. Саме така концепція є найбільш перспективною для розробки застосунку моніторингу мережної безпеки з автоматичним виявленням аномалій, оскільки дозволяє підвищити точність виявлення інцидентів та зменшити кількість помилкових спрацювань.

2.3 Формування вимог до програмного продукту

З огляду на тему кваліфікаційної роботи, програмний продукт має бути орієнтований на автоматизований збір мережевих даних, аналіз подій безпеки та виявлення аномальної активності в локальному мережевому середовищі.

Функціональні вимоги

Основні функціональні вимоги прототипу: можливість захоплення та аналізу мережевого трафіку в режимі, наближеному до реального часу. Це необхідна умова для оперативного отримання інформації про події, що відбуваються у мережевому середовищі, а також для своєчасного реагування на потенційно небезпечну активність. Система повинна забезпечувати безперервний збір пакетних даних або мережевої статистики без суттєвого впливу на стабільність роботи пристрою.

Збір та структуроване відображення відомостей про активні мережеві з’єднання дозволить формувати загальну картину мережевої активності. До таких даних належать IP-адреси джерела та призначення, використовувані протоколи передачі даних, номери портів, тривалість з’єднань та інші технічні параметри.

Прототип повинен автоматично виявляти підозрілу або аномальну поведінку. Реалізація здійснюється на основі попередньо визначених правил, сигнатур, порогових значень або алгоритмів статистичного аналізу. Наприклад, система може фіксувати різке збільшення кількості запитів, спроби сканування портів, повторювані з’єднання з невідомими вузлами або інші нетипові відхилення від нормального режиму роботи мережі.

Система повинна забезпечувати ведення журналу подій безпеки та збереження результатів перевірок у базі даних. Реєстрація подій повинна включати дату й час інциденту, джерело активності, короткий опис події та рівень потенційної небезпеки. Наявність журналів дозволяє здійснювати подальший аналіз інцидентів, формувати звітність та відстежувати повторювані загрози.

Необхідно реалізувати відображення статистичних показників у графічному вигляді. Використання графіків, діаграм та часових шкал дозволяє швидко оцінити

навантаження мережі, активність окремих вузлів, кількість підозрілих подій та загальну динаміку змін у роботі мережевого середовища. Візуальне подання інформації значно спрощує роботу адміністратора та пришвидшує прийняття рішень.

Окремою функціональною вимогою є формування сповіщень про виявлені аномалії або потенційні інциденти безпеки. Система повинна інформувати користувача про критичні події за допомогою повідомлень у вебінтерфейсі, змін статусу системи або інших засобів оповіщення. Це забезпечить можливість оперативного реагування на небезпечні ситуації.

Користувач повинен мати можливість перегляду історії подій через вебінтерфейс застосунку. Такий функціонал дозволить аналізувати попередні інциденти, здійснювати пошук за датою або типом загрози та контролювати загальний стан мережевої безпеки протягом тривалого часу.

Нефункціональні вимоги

До нефункціональних вимог належать характеристики програмного продукту, що визначають якість його роботи, надійність експлуатації та зручність подальшого використання.

Перша нефункціональна вимога – стабільність роботи протягом тривалого часу. Програмний продукт повинен коректно функціонувати у безперервному режимі без критичних помилок, зависань або втрати працездатності. Оскільки система призначена для постійного спостереження за станом мережі, її робота має бути надійною навіть при тривалому навантаженні.

Друга вимога - це надійність збереження даних. Журнали подій, результати аналізу, статистичні показники та параметри конфігурації повинні зберігатися без пошкодження або втрати інформації. У разі аварійного завершення роботи застосунку бажаним є збереження вже накопичених даних та можливість подальшого відновлення роботи системи.

Окрему увагу слід приділити безпечному зберіганню конфігураційної інформації. До такої інформації можуть належати параметри мережевого

моніторингу, налаштування доступу, службові ключі або інші технічні дані. Програмний продукт повинен виключати можливість несанкціонованого доступу до зазначених параметрів та забезпечувати належний рівень захисту службової інформації.

Третя вимога - простота встановлення та налаштування системи. Процес розгортання програмного продукту не повинен вимагати складної технічної підготовки або великої кількості ручних операцій. Чим простіше впровадження системи, тим вищою є ймовірність її практичного використання у реальному середовищі.

Четверта – простота у супроводі програмного коду. Структура проєкту повинна бути логічною, зрозумілою та придатною до внесення змін. Це спрощує виправлення помилок, оновлення окремих модулів та розширення функціональних можливостей у майбутньому.

П’ята вимога - сумісність із сучасними операційними системами. Прототип повинен мати можливість роботи щонайменше у поширених програмних середовищах, зокрема Windows та Linux. Кросплатформність підвищує універсальність програмного рішення та спрощує його впровадження у різних організаціях.

Інтерфейс користувача

Інтерфейс програмного продукту є важливою складовою системи моніторингу мережної безпеки, оскільки саме через нього користувач взаємодіє з основними функціями застосунку, переглядає результати аналізу та отримує інформацію про стан мережевого середовища. Від якості реалізації інтерфейсу безпосередньо залежить швидкість роботи адміністратора, зручність використання системи та ефективність реагування на потенційні інциденти безпеки.

Головна вимог до інтерфейсу є його зрозумілість та простота використання. Користувач повинен мати змогу швидко орієнтуватися у структурі застосунку, знаходити необхідні функції та отримувати доступ до основної інформації без складного навчання або тривалого ознайомлення із системою. Логічне

розташування елементів керування, зрозумілі назви розділів та послідовна навігація суттєво підвищують комфорт роботи із програмним продуктом.

Відсутність перевантаження зайвими елементами: інтерфейс не повинен містити надлишкової кількості кнопок, меню чи візуальних компонентів, що можуть ускладнювати сприйняття інформації. Основна увага користувача має бути зосереджена на ключових показниках безпеки, поточному стані мережі та повідомленнях про виявлені загрози.

Зручне та доступне відображення даних щодо активності мережі, журналів подій, статистичних показників та виявлених аномалій. Використання таблиць, графіків, кольорових індикаторів статусу та коротких інформаційних блоків дозволять швидко оцінити ситуацію без детального аналізу кожного параметра окремо.

Можливість швидкого переходу між основними розділами системи буде перевагою. Користувач матиме оперативний доступ до сторінок перегляду журналів подій, статистики трафіку, налаштувань системи, переліку виявлених інцидентів та конфігурації правил моніторингу. Це скоротить час реагування у випадку підозрілої активності.

Веборієнтований інтерфейс користувача забезпечить доступність. Для користування прототипом буде потрібен тільки пристрій з браузером, без необхідності встановлення окремого клієнтського програмного забезпечення. Крім того, вебінтерфейс спрощує розгортання прототипу у локальній мережі та дозволяє отримувати доступ до нього з різних пристроїв за наявності відповідних прав доступу.

Додатковою перевагою веборієнтованого інтерфейсу є можливість подальшого масштабування, оновлення зовнішнього вигляду системи та інтеграції нових модулів без необхідності перевстановлення програмного забезпечення на кожному окремому пристрої користувача.

Швидкодія

Оскільки програмний продукт призначений для моніторингу мережевого середовища та аналізу поточного трафіку, однією з ключових вимог є достатній рівень швидкодії системи. Ефективність виявлення загроз значною мірою залежить від того, наскільки оперативно прототип здатний обробляти отримані дані, аналізувати події та повідомляти користувача про потенційно небезпечну активність. Надмірні затримки в роботі можуть призвести до несвоєчасного реагування на інциденти та зниження практичної цінності системи.

Важливою вимогою є мінімальна затримка між моментом отримання мережевих даних і відображенням результатів аналізу. Події, що фіксуються у мережі, повинні оброблятися у режимі, наближеному до реального часу, щоб користувач міг своєчасно отримувати актуальну інформацію про стан мережі, появу аномалій або підозрілих з’єднань. Це особливо важливо під час виявлення атак, що розвиваються швидко та можуть завдати шкоди за короткий проміжок часу.

Програмний продукт повинен забезпечувати стабільну роботу при типовому навантаженні локальної мережі без суттєвого зниження продуктивності комп’ютера, на якому він розгорнутий. Система не повинна створювати надмірне навантаження на центральний процесор, займати критичний обсяг оперативної пам’яті або перешкоджати роботі інших прикладних програм. Це є важливою умовою використання застосунку на звичайних робочих станціях без спеціалізованого серверного обладнання.

Окремої уваги потребує оптимізація використання оперативної пам’яті. Під час обробки значної кількості мережевих пакетів система повинна раціонально використовувати доступні ресурси, очищувати тимчасові дані та не допускати неконтрольованого зростання споживання пам’яті у процесі тривалої роботи.

Також бажаною вимогою є ефективне використання процесорних ресурсів. Алгоритми аналізу повинні бути реалізовані таким чином, щоб виконуватися достатньо швидко без зайвих обчислювальних витрат. Це особливо актуально при одночасному зборі трафіку, роботі вебінтерфейсу, записі даних до бази та побудові графічної статистики.

Для підвищення швидкодії доцільним є використання асинхронної обробки окремих задач, буферизації даних, пакетного запису до бази даних та оптимізації запитів до внутрішніх модулів системи. Такий підхід дозволяє зменшити навантаження та підвищити загальну продуктивність програмного продукту.

Масштабованість

Однією з важливих вимог до сучасного програмного продукту є можливість його подальшого розвитку без необхідності повного перепроєктування архітектури. Для системи моніторингу мережної безпеки це має особливе значення, оскільки кіберзагрози постійно змінюються, а потреби користувачів і мережевої інфраструктури з часом ускладнюються. Саме тому прототип повинен створюватися з урахуванням принципів масштабованості та модульності.

Архітектура системи має дозволяти додавання нових функціональних компонентів із мінімальним впливом на вже реалізовані модулі. Такий підхід спрощує модернізацію програмного продукту, прискорює впровадження нових можливостей та зменшує ризик появи помилок під час оновлення системи.

Одним із перспективних напрямів розвитку є підтримка декількох мережевих інтерфейсів одночасно. Це дозволить здійснювати моніторинг не лише одного каналу зв’язку, а декількох сегментів мережі, що особливо актуально для корпоративного середовища з розподіленою структурою.

Доцільною є можливість інтеграції із зовнішніми SIEM-системами, журналами подій або іншими засобами централізованого контролю безпеки. Така взаємодія дозволить передавати результати аналізу до більших систем керування інцидентами та використовувати прототип як окремий елемент комплексної інфраструктури захисту.

Подальший розвиток програмного продукту може передбачати використання алгоритмів машинного навчання для автоматичного виявлення складних аномалій, прогнозування нетипової поведінки мережі та підвищення точності спрацювань. Це особливо актуально для великих обсягів трафіку, де ручний аналіз є малоефективним.

Перспективною вимогою є реалізація централізованого моніторингу кількох вузлів мережі. У такому випадку система зможе збирати дані з декількох пристроїв або серверів та відображати загальну картину стану мережевої інфраструктури в єдиному інтерфейсі керування.

Не менш важливим напрямом масштабування є впровадження багатокористувацької системи доступу. Це дасть змогу розмежовувати права адміністраторів, операторів та інших користувачів, вести облік дій у системі та забезпечувати безпечну спільну роботу декількох осіб.

Для забезпечення масштабованості доцільно використовувати модульну структуру програмного коду, окремі логічні компоненти обробки даних та можливість переходу на продуктивніші технології зберігання чи обробки інформації у майбутньому.

2.4 Вибір технологій для реалізації

Ефективність програмного продукту значною мірою залежить від правильного вибору технологічного стеку, що використовується під час його розробки. Для застосунку моніторингу мережної безпеки особливого значення набувають швидкість обробки даних, можливість роботи з мережевими інтерфейсами, підтримка автоматизованого аналізу інформації, зручність створення вебінтерфейсу та подальша масштабованість системи. У зв’язку з цим вибір технологій має ґрунтуватися не лише на популярності окремих рішень, а й на їх відповідності поставленим функціональним завданням.

Під час проєктування прототипу продукту доцільно використовувати модульний підхід, за якого система складається з окремих взаємопов’язаних компонентів: модуля збору мережевих даних, сервера обробки інформації, бази даних, аналітичного модуля та користувацького вебінтерфейсу. Така архітектура спрощує супровід програмного забезпечення, оновлення окремих компонентів та подальше розширення функціональних можливостей.

Мова програмування

Для створення прототипу застосунку доцільним є використання мови програмування Python. Вибір обумовлений високою швидкістю розробки, зручністю реалізації базової серверної логіки та можливістю оперативного тестування функціональних сценаріїв роботи системи.

Python має значну кількість готових бібліотек, що дозволить реалізувати прототип без необхідності створення та налаштування низькорівневих механізмів з нуля. Зокрема, бібліотека PyShark використовується для аналізу мережевого трафіку та роботи з пакетними даними [9], SQLite - для організації локального зберігання даних і журналів подій [10], Flask - для побудови вебсерверної частини застосунку [11], а Matplotlib - для візуалізації статистичних показників, графіків та результатів аналізу [12].

Додатковими перевагами Python є кросплатформність, простота супроводу програмного коду та можливість подальшої інтеграції з модулями машинного навчання, що є важливим для систем автоматичного виявлення аномалій.

Для реалізації графічного інтерфейсу користувача доцільно застосувати веборієнтований інтерфейс. Такий підхід забезпечує можливість розгортання системи на різних пристроях без встановлення окремого клієнтського програмного забезпечення. Доступ до функціоналу застосунку може здійснюватися через браузер із будь-якого пристрою, що знаходиться в межах локальної мережі або має відповідний мережевий доступ до сервера. Додатковою перевагою є спрощення адміністрування, оновлення інтерфейсу та централізоване керування системою.

Pyshark

Бібліотека PyShark є програмною надбудовою над Wireshark/TShark та призначена для аналізу мережевого трафіку засобами Python [9]. Її використання дозволяє отримувати детальну інформацію про мережеві пакети, IP-адреси джерела та призначення, використовувані протоколи, номери портів, часові характеристики з’єднань та інші параметри передачі даних.

Застосування даної бібліотеки створює можливість реалізувати у програмному продукті модуль перехоплення, збору та подальшого аналізу мережевого трафіку в режимі реального часу. На основі отриманих даних система може виявляти нетипову активність, підозрілі з’єднання, різке зростання навантаження, сканування портів, повторювані запити та інші ознаки потенційних кіберзагроз.

Додатковою перевагою PyShark є інтеграція з середовищем Python, що спрощує подальшу обробку мережевих даних, формування статистики та використання алгоритмів автоматичного виявлення аномалій у межах розроблюваного застосунку.

SQLite

SQLite є вбудованою реляційною системою керування базами даних [10]. Особливістю даного рішення є зберігання всієї бази даних у вигляді локального файлу, що суттєво спрощує розгортання програмного продукту, оскільки не потребує використання окремого серверного середовища чи встановлення додаткових служб керування базами даних.

Зазначена особливість є важливою перевагою на етапі створення прототипу системи, адже дозволяє зосередитися безпосередньо на реалізації функціональних можливостей застосунку без необхідності додаткового налаштування взаємодії між серверною частиною та зовнішньою базою даних. Це скорочує час розробки, спрощує тестування та зменшує загальну складність впровадження програмного продукту.

У межах розроблюваного застосунку SQLite доцільно використовувати для зберігання журналів подій, результатів перевірок, параметрів конфігурації, статистичних показників роботи системи, а також історії виявлених аномалій. Наявність централізованого локального сховища даних забезпечує можливість подальшого аналізу інцидентів, формування звітності та відстеження змін у роботі мережевого середовища.

Таким чином, використання SQLite є раціональним рішенням для прототипу застосунку, оскільки поєднує простоту інтеграції, мінімальні системні вимоги та достатній функціонал для зберігання службової інформації.

Flask

Flask є легким вебфреймворком для мови програмування Python, що призначений для створення серверних вебзастосунків, REST API-сервісів та програмних систем із вебінтерфейсом користувача [11]. Завдяки простій структурі та модульному підходу даний інструмент широко використовується для розробки прототипів і прикладних інформаційних систем різного рівня складності.

У межах кваліфікаційної роботи Flask доцільно використовувати для реалізації серверної логіки застосунку, обробки HTTP-запитів, організації маршрутизації сторінок та забезпечення взаємодії між вебінтерфейсом і модулями аналізу мережевого трафіку. За допомогою цього фреймворку можливо реалізувати передачу даних між клієнтською та серверною частинами системи, запуск функцій моніторингу, формування журналів подій і відображення результатів аналізу у веббраузері користувача.

Важливою перевагою Flask є гнучкість архітектури, що дозволяє поступово розширювати функціональні можливості застосунку шляхом додавання нових модулів без суттєвої зміни базової структури проєкту. Крім того, фреймворк має значну кількість готових розширень для авторизації користувачів, роботи з базами даних, формування API та інтеграції із зовнішніми сервісами.

Таким чином, використання Flask є доцільним рішенням для створення прототипу системи моніторингу мережної безпеки, оскільки поєднує простоту впровадження, достатню функціональність та можливість подальшого розвитку програмного продукту.

Matplotlib

Matplotlib є бібліотекою для побудови графіків, діаграм та візуалізації даних у середовищі Python [12]. Даний інструмент широко застосовується для

представлення статистичної інформації у зручному графічному вигляді, що спрощує сприйняття результатів аналізу та подальше прийняття рішень.

У межах розроблюваного застосунку Matplotlib може використовуватися для відображення статистики мережевого навантаження, кількості активних з’єднань, часових змін мережевої активності, частоти звернень до окремих сервісів, а також результатів автоматичного виявлення аномалій. Зокрема, за допомогою бібліотеки можливо реалізувати графіки активності конкретного хоста, динаміку мережевого трафіку за визначений проміжок часу або візуальне відображення різкого зростання підозрілої активності, що може свідчити про початок атаки чи некоректну роботу мережевого вузла.

Використання графічного подання інформації суттєво підвищує зручність аналізу стану мережі адміністратором, оскільки дозволяє швидко виявляти нетипові відхилення, порівнювати показники у різні часові періоди та оперативно реагувати на потенційні загрози. Візуалізація даних також спрощує формування звітів щодо роботи системи моніторингу.

ML-Learning

Бібліотека Pandas є високорівневою надбудовою над NumPy, призначеною для маніпуляції та аналізу структурованих даних. Основними структурами даних є Series та DataFrame, що дозволяють ефективно зберігати та обробляти мережеві логи, подані у вигляді часових рядів. У межах розроблюваного застосунку Pandas використовується для попередньої обробки зібраної телеметрії: фільтрації пакетів за ознаками, агрегації трафіку за часовими інтервалами та вирахування статистичних показників, таких як середнє навантаження або частота запитів. Інтеграція з SQL-подібними операціями забезпечує швидке об’єднання даних із бази SQLite для проведення ретроспективного аналізу [13].

Scikit-learn - це інструментальна бібліотека, що реалізує широкий спектр математичних алгоритмів для статистичного аналізу та машинного навчання. Для вирішення завдання автоматичного виявлення аномалій у мережевому трафіку використовуються методи навчання без учителя, які не потребують попередньо

розміченої бази атак. Зокрема, бібліотека надає реалізації алгоритмів Isolation Forest та Local Outlier Factor. Крім того, Scikit-learn забезпечує необхідний функціонал для нормалізації та масштабування мережевих ознак, що дозволяє привести різнорідні метрики до єдиного діапазону для підвищення точності роботи алгоритмів [14].

Поєднання цих бібліотек дозволяє реалізувати повний цикл аналізу: від первинного очищення та структурування мережевих даних до автоматичного прийняття рішення про наявність аномальної активності без жорсткого задання статичних правил.

РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ БЕЗПЕКИ МЕРЕЖІ.

3.1. Архітектура та опис системи

Використання модульної архітектури дозволяє ізолювати процес захоплення трафіку від аналітичного модуля, що запобігає втраті пакетів при високому навантаженні системи. Надає зручність супроводу програмного коду, можливість подальшого розширення функціоналу та адаптацію до змінних вимог користувачів. Архітектура системи базується на принципі розподілу відповідальності, де кожен модуль виконує незалежну частину обробки телеметрії. Такий підхід передбачає поділ системи на окремі логічно завершені компоненти, кожен із яких виконує власну функцію та взаємодіє з іншими частинами через визначені внутрішні інтерфейси. Загальну структуру шарів розробленого застосунку відображено на рисунку 3.1.

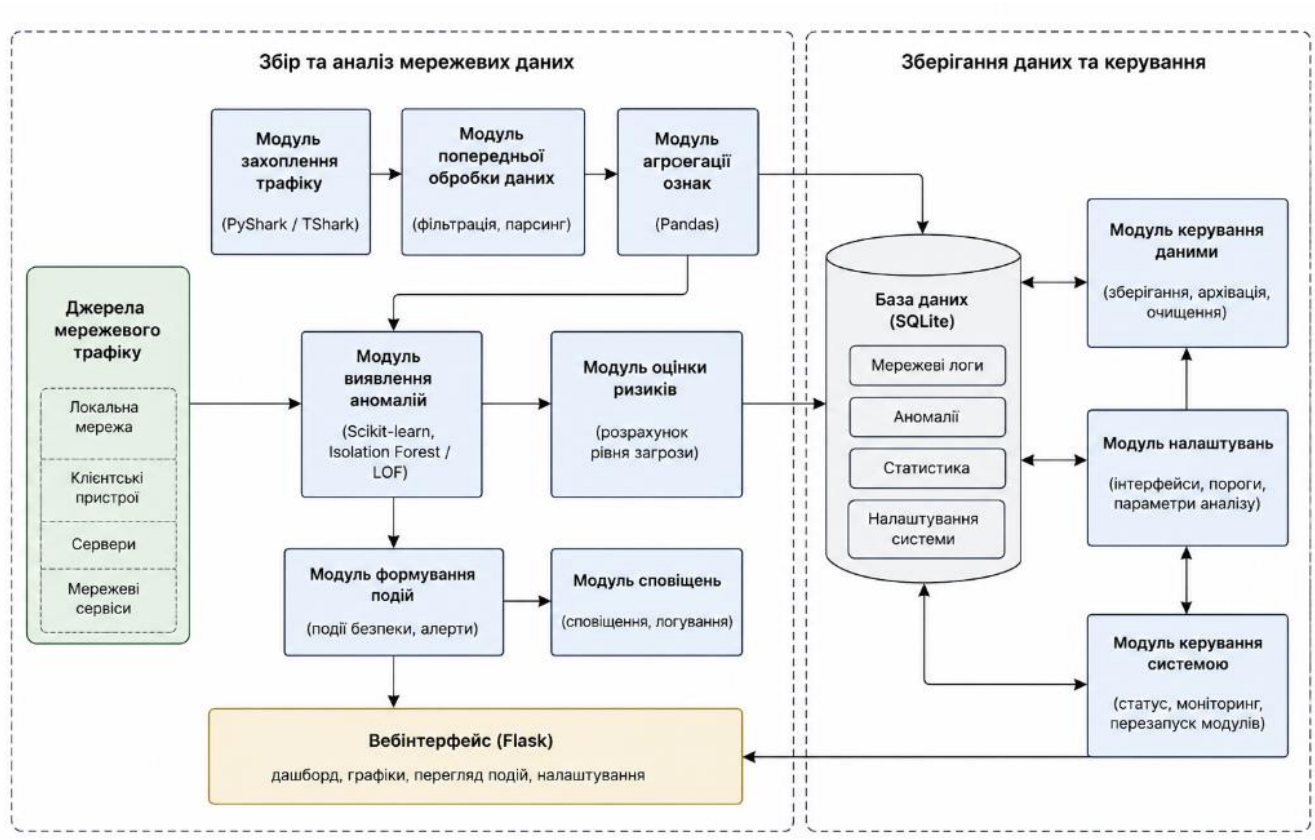

Рис 3.1 Діаграма шарів системи

Нижній рівень системи включає механізми збору телеметрії та підсистему зберігання даних. Модуль збору, реалізований за допомогою бібліотеки PyShark, відповідає за перехоплення пакетів та формування первинного набору параметрів для аналізу. Для накопичення журналів подій, статистики та історії аномалій використовується підсистема зберігання на базі СУБД SQLite.

Середній рівень представлений аналітичними компонентами та серверною логікою. Аналітичний модуль на базі Scikit-learn та Pandas здійснює обробку даних та виявлення аномальної поведінки. Цей компонент ідентифікує інциденти безпеки, як-от підозрілі з'єднання або перевищення порогових навантажень. Важливою частиною цього рівня є модуль сповіщень, який забезпечує оперативне інформування про критичні події.

Верхній рівень становить користувацький інтерфейс, реалізований як вебінтерфейс на мікрофреймворку Flask. Він призначений для візуалізації статистики, перегляду логів та керування параметрами моніторингу. Взаємодію між усіма технологічними модулями через програмні інтерфейси проілюстровано на UML-діаграмі компонентів зображеній на рисунку 3.2.

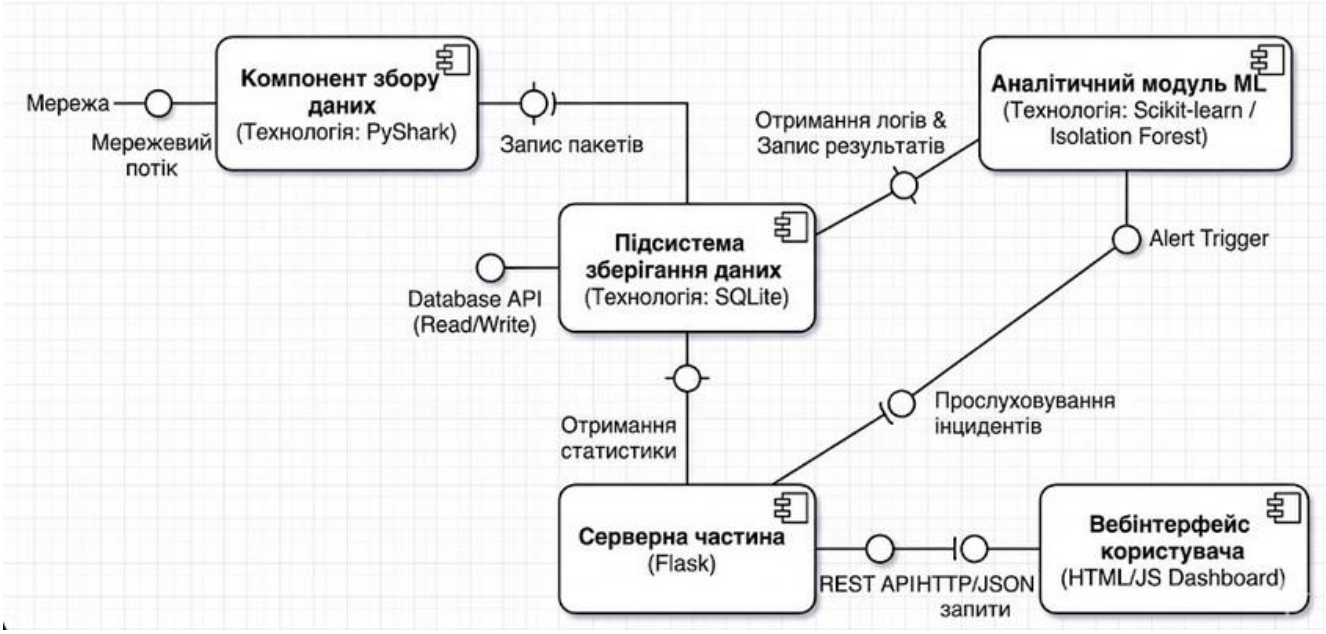

Рис 3.2 UML-діаграма компонентів

Серверна частина системи виступає сполучною ланкою, що забезпечує координацію внутрішніх модулів, обробку запитів користувача та синхронізацію даних між базою даних SQLite та вебінтерфейсом. Завдяки реалізації логіки на базі Flask, забезпечується централізоване керування всіма функціональними можливостями застосунку.

Особливу роль відіграє модуль сповіщень, інтегрований у серверну частину. Він відповідає за оперативне інформування адміністратора про виявлені аномалії або критичні порушення штатного режиму роботи мережі. Механізм оповіщення реалізовано через передачу повідомлень у реальному часі до вебінтерфейсу, що дозволяє мінімізувати час реагування на інциденти.

Фізичне розміщення частин системи, архітектуру вузлів та їх інтеграцію у локальну мережу підприємства відображено на діаграмі розгортання, яка зображена на рисунку 3.3. На діаграмі представлено трирівневу структуру взаємодії компонентів: робочу станцію адміністратора, локальний мережевий сегмент підприємства та сервер моніторингу.

Робоча станція адміністратора містить клієнтський програмний компонент у вигляді веббраузера, через який здійснюється доступ до функцій системи керування та перегляд результатів моніторингу. Обмін даними із серверною частиною виконується за протоколами HTTP або HTTPS, що забезпечує віддалений доступ до вебінтерфейсу без встановлення додаткового спеціалізованого програмного забезпечення.

Локальна мережа підприємства включає клієнтські персональні комп’ютери, сервери мережевих служб та комутаційне обладнання. Центральним вузлом сегмента є маршрутизатор або комутатор, через який проходить основний внутрішній трафік. Саме цей вузол використовується як точка спостереження за мережевими потоками та джерело даних для аналізу стану інфраструктури.

Сервер моніторингу є окремим обчислювальним вузлом, розгорнутим під керуванням операційної системи Windows або Linux. На ньому розміщено основні програмні артефакти системи: вебсервер Flask, модуль збору трафіку, аналітичний модуль, модуль сповіщень та локальну базу даних SQLite.

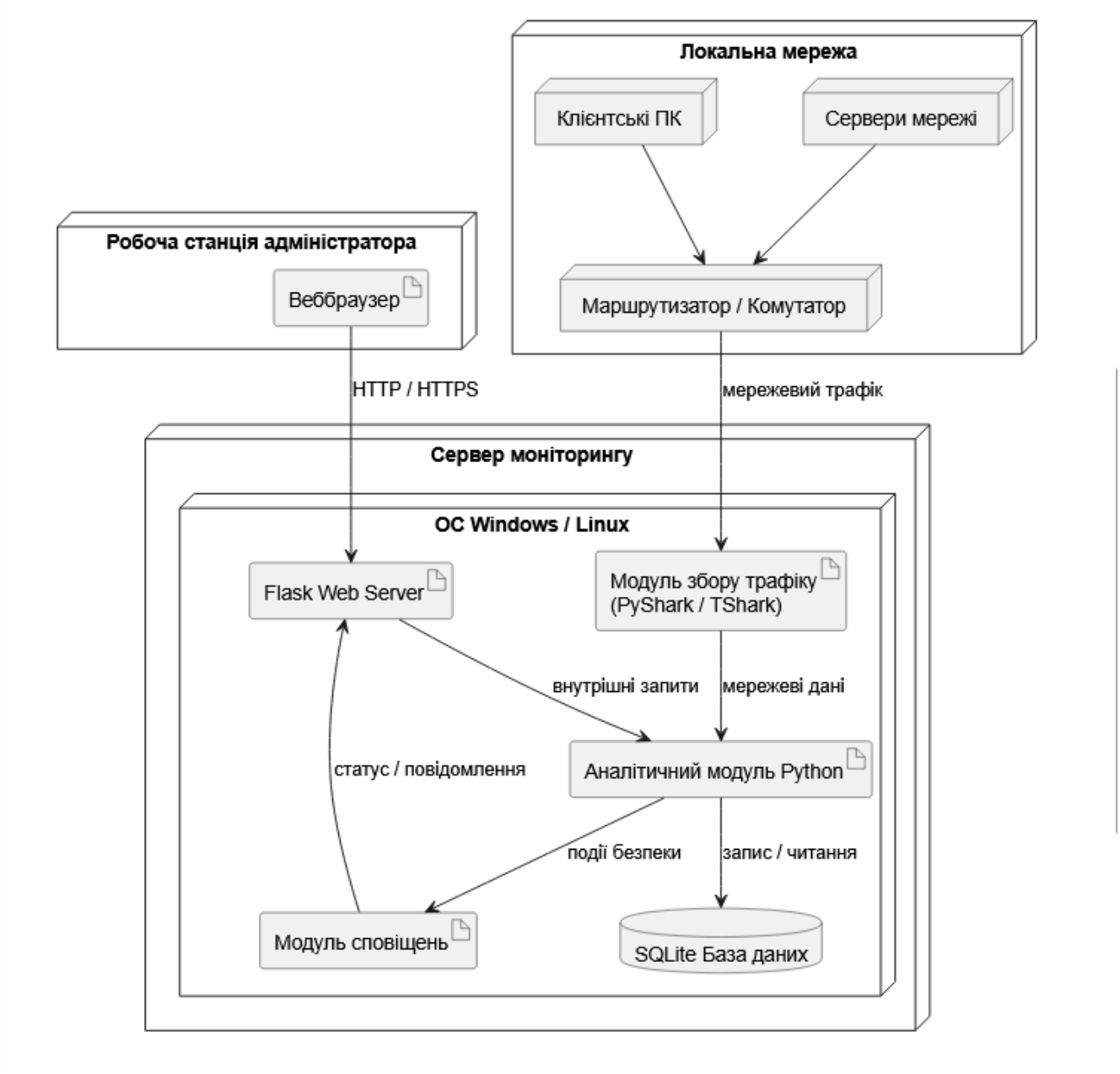

Рис 3.3 Діаграма розгортання

3.2 Проєктування внутрішньої структури та алгоритмів застосунку

Проєктування внутрішньої структури програмного застосунку визначає механізми обробки мережевих даних, логіку функціонування модулів та способи їхньої взаємодії. Детальна розробка архітектури на цьому етапі мінімізує ймовірність помилок під час програмної реалізації та забезпечує відповідність системи вимогам продуктивності.

3.2.1 Проєктування структури бази даних

Для збереження результатів моніторингу використовується локальна реляційна база даних SQLite. Архітектуру бази спроєктовано за принципом мінімізації використання дискового простору. Загальна статистика мережевих підключень реєструється для всіх перехоплених пакетів. Глибокий аналіз та збереження корисного навантаження виконується виключно для подій, класифікованих як аномалія.

Логічна структура бази даних складається з п'яти основних сутностей: Admins, Logs, Anomaly, Payload, Notifications. Зв'язки між сутностями реалізовано через зовнішні ключі. Таблиця корисного навантаження пов'язана безпосередньо з таблицею інцидентів, що повністю виключає запис вмісту безпечних пакетів до пам'яті. Логічну схему бази даних наведено на рисунку 3.4.

Таблиця Admins містить облікові записи адміністраторів системи та хешовані паролі для безпечної авторизації.

Таблиця Logs є базовим журналом мережевих подій. Вона фіксує точний час підключення, IP-адреси відправника та отримувача, мережеві порти, тип транспортного протоколу та статус безпеки пакета.

Таблиця Anomaly акумулює дані про зафіксовані інциденти. Кожен запис містить ідентифікатор базового логу, тип виявленої загрози та детальний опис підозрілої активності.

Таблиця Payload призначена для розслідування інцидентів. Вона зберігає сирі байти перехопленого пакета та розкодований текст для аналізу вектора атаки.

Таблиця Notifications забезпечує роботу системи сповіщень. Записи містять посилання на конкретну аномалію, статус опрацювання інциденту адміністратором та рівень критичності загрози.

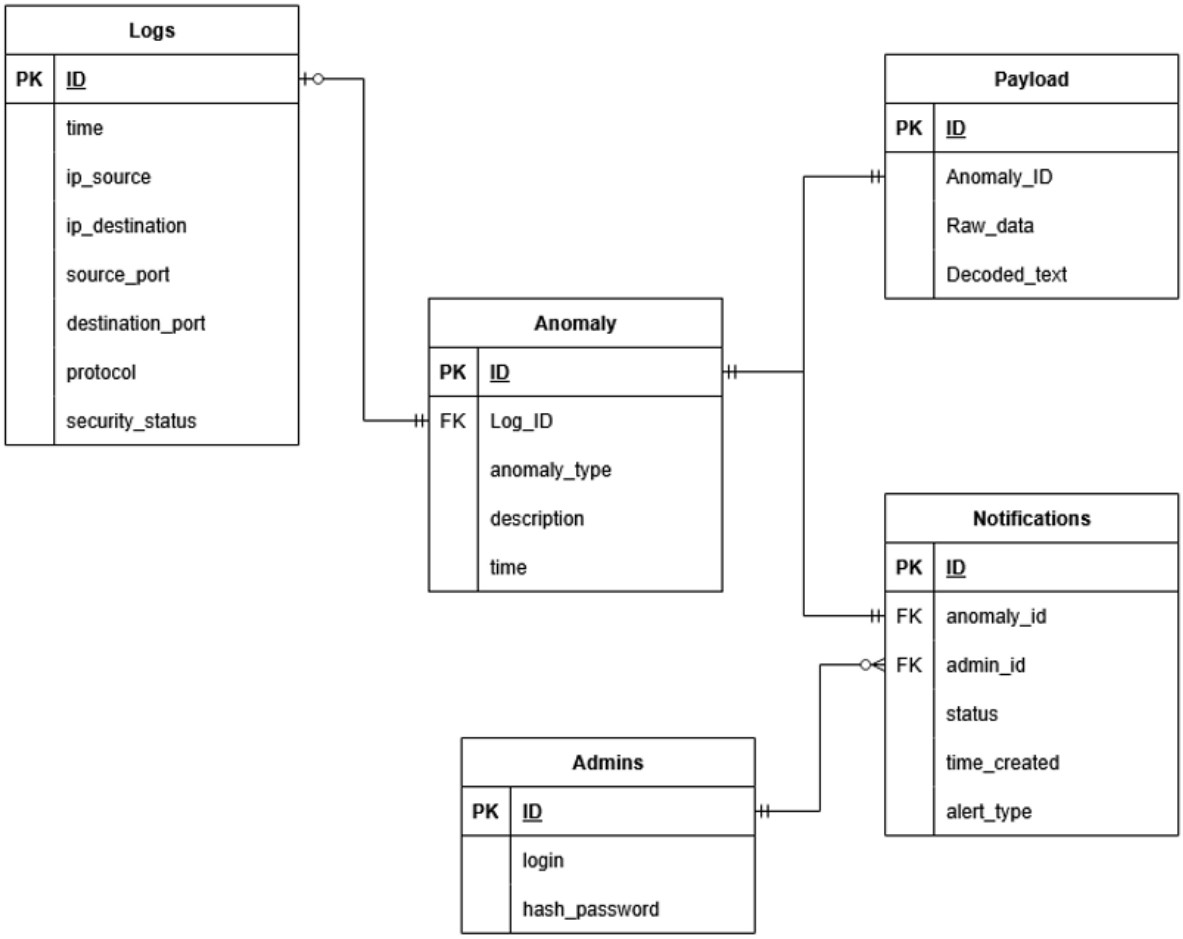

Рис 3.4 Логічна схема бази даних

3.2.2 Проєктування алгоритму функціонування системи

Блок-схему алгоритму виявлення та блокування мережевих загроз представлено на рисунку 3.5. Робота застосунку базується на подієво-орієнтованому підході з використанням багаторівневої оцінки ризиків згідно з принципами систем запобігання вторгненням.

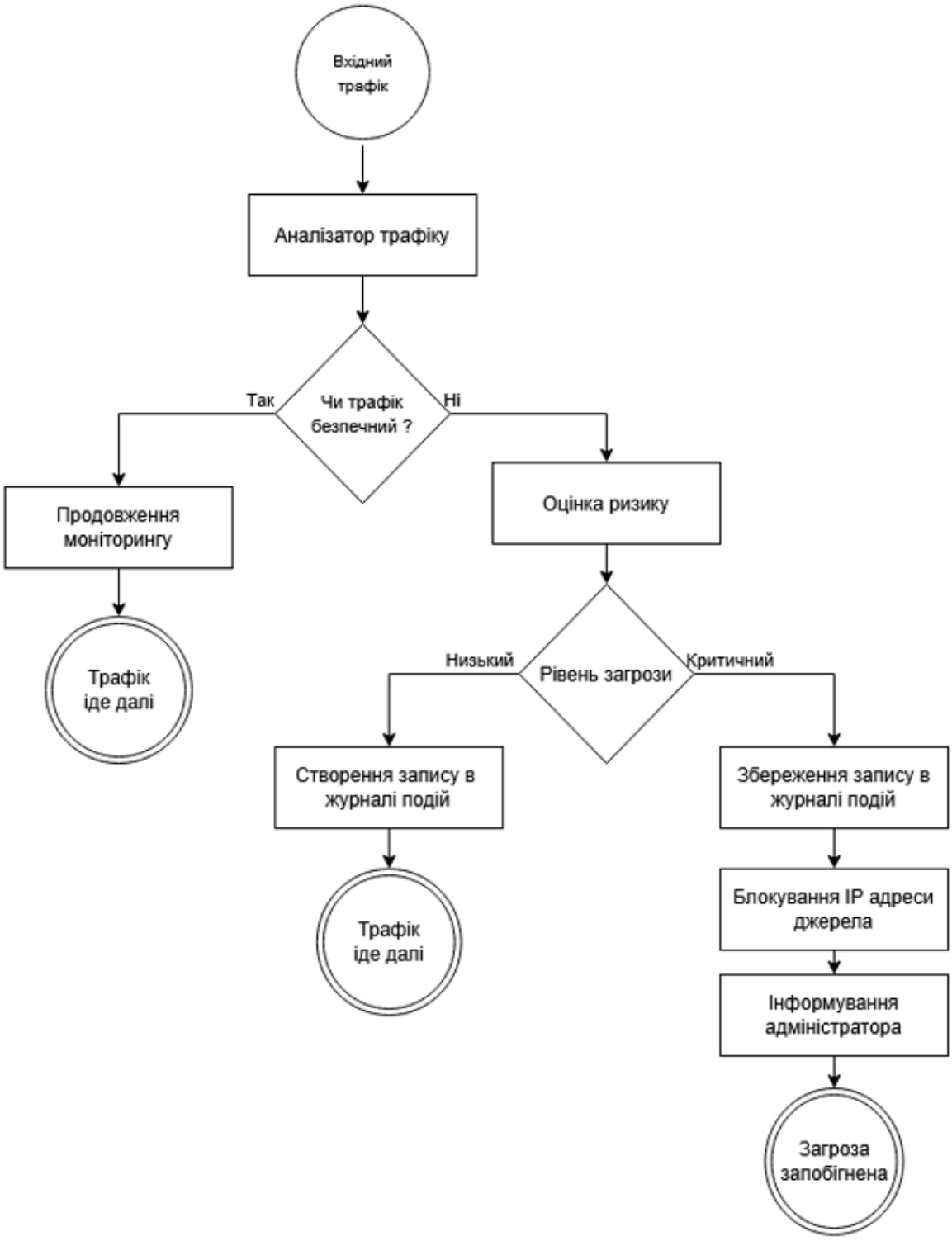

Рис 3.5 Алгоритм виявлення і блокування загроз

Процес функціонування розпочинається з безперервного прослуховування вхідного мережевого потоку модулем захоплення. Кожен пакет спрямовується до аналізатора для проведення первинної оцінки характеристик з'єднання.

Алгоритм передбачає диференційовану обробку даних залежно від статусу безпеки. Легітимний трафік проходить крізь систему в режимі пасивного спостереження. Підозрілі пакети підлягають обов'язковій класифікації для встановлення рівня загрози. Результатом ідентифікації низькорівневої або неоднозначної активності є реєстрація події у відповідному журналі та генерація попереджувального сповіщення.

Виявлення критичної загрози активує функціонал активного захисту системи. Програмний комплекс виконує збереження корисного навантаження пакета для подальшого аналізу, ініціює автоматичне блокування IP-адреси джерела на рівні міжмережевого екрана та формує термінове сповіщення для адміністратора.

3.2.3 Проєктування взаємодії програмних компонентів

Взаємодія внутрішніх об'єктів системи під час аналізу пакета визначає ефективність обробки даних у реальному часі. Процес обміну повідомленнями та викликами функцій проілюстровано на UML-діаграмі послідовності на рисунку 3.6.

Процес розпочинається із перехоплення пакета модулем PyShark. Модуль вилучає необхідні метадані та формує вектор ознак. Сформований набір даних передається до аналітичного ML-ядра для обчислення оцінки аномальності за допомогою алгоритму Isolation Forest.

Перевищення встановленого порогу аномальності активує транзакцію запису до таблиць логів, інцидентів та корисного навантаження у базі SQLite. Після успішного збереження даних генерується сповіщення, яке передається через WebSocket до вебінтерфейсу Flask. Пакет із показниками в межах норми реєструється виключно у базовому журналі без ресурсомісткого розбору вмісту. Серверна частина Flask паралельно виконує запити до бази даних для оновлення статистичних графіків на панелі керування.

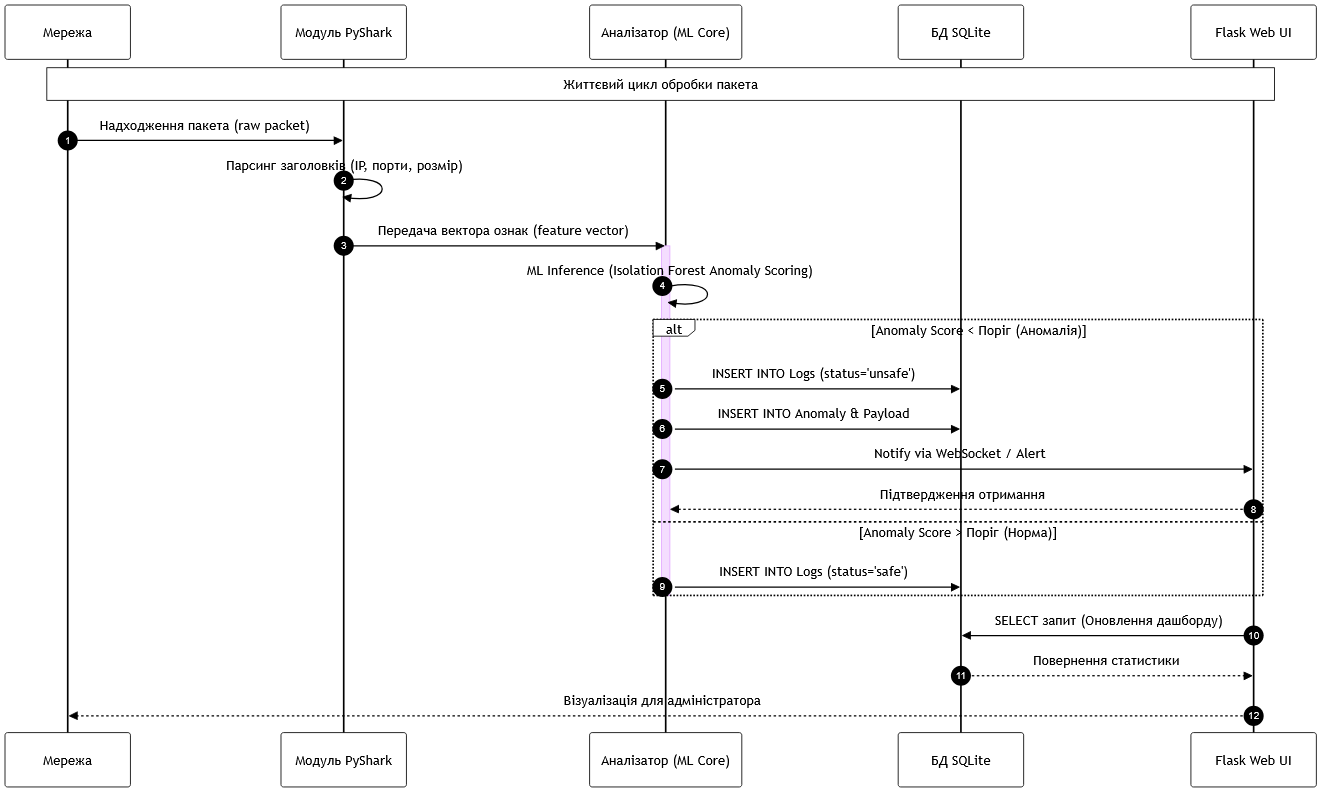

Рис 3.6 Діаграма послідовності циклу обробки мережевого пакета

3.2.4 Проєктування аналітичного модуля на основі алгоритму Isolation Forest

Інтелектуальне ядро системи побудовано на базі ансамблевого алгоритму машинного навчання Isolation Forest, реалізованого за допомогою бібліотеки Scikit-learn. Такий підхід відноситься до методів навчання без учителя і дозволяє ефективно виявляти загрози нульового дня та нестандартну мережеву активність без необхідності використання попередньо розмічених баз сигнатур.

Механізм роботи алгоритму базується на фундаментальній властивості аномалій: вони є рідкісними подіями та суттєво відрізняються від нормального трафіку за своїми ознаками. Процес аналізу полягає у побудові серії випадкових дерев рішень. Алгоритм рекурсивно розділяє вектор ознак мережевого пакета, випадковим чином обираючи параметр та точку перетину. Оскільки аномальні пакети мають нестандартні значення характеристик, алгоритму потрібно значно менше кроків для їх повної "ізоляції" від решти набору даних.

Математичним критерієм ідентифікації загрози є середня довжина шляху від кореня дерева до кінцевого вузла. На основі цієї довжини обчислюється оцінка аномальності. Пакети з короткою довжиною шляху отримують оцінку, що наближається до -1, і класифікуються як аномальні. Легітимний трафік, який потребує глибокого розгалуження для ізоляції, отримує оцінку ближче до 1.

Для забезпечення швидкодії системи під час роботи в реальному часі вектор ознак попередньо обробляється за допомогою інструментів Pandas, після чого передається до навченої моделі Isolation Forest для миттєвої класифікації та прийняття рішення щодо безпеки з'єднання.

3.3 Програмна реалізація застосунку

Програмна реалізація системи базується на принципах модульності та асинхронності, що дозволяє уникнути блокування основного потоку виконання під час інтенсивного захоплення трафіку. Проєкт реалізовано мовою Python 3.12 із чітким розподілом функцій між бекенд-ядром (збір та аналіз) та фронтенд-частиною (візуалізація). Повні лістинги вихідного коду наведено у додатках.

3.3.1 Реалізація модуля збору мережевого трафіку

Основним завданням модуля сніфера є безперервне прослуховування мережевого інтерфейсу та екстракція метаданих без втрати пакетів. Для цього використано клас LiveCapture з бібліотеки PyShark. Щоб система могла працювати в режимі реального часу, захоплення трафіку винесено в окремий фоновий потік (threading).

Програмна логіка обробника (sniffer.py) виконує парсинг кожного кадру на льоту. З пакета вилучаються ключові ознаки, які були визначені на етапі проєктування: розмір пакета (length), порти джерела та призначення (src_port, dst_port) і транспортний протокол (TCP/UDP). Зібрані дані формуються у словник і передаються до конвеєра аналізу. Фрагмент логіки екстракції мережевих ознак представлено у лістингу 3.1.

```
1. def process_packet(self, packet: Any) -> None:
2.                     packet_info    =    packet_to_dict(packet,
self._rate_meter.tick())
3.     result = analyze_packet(packet_info)
4.
5.         log_id  =  insert_log(packet_info,  result["status"],
result["score"])
6.           notify_packet(create_payload(log_id,  packet_info,
result))
7.
8.     if result["status"] != "safe":
9.          handle_anomaly(log_id, packet_info, result)
```

Лістинг 3.1 Фрагмент логіки обробки пакета

3.3.2 Реалізація аналітичного модуля машинного навчання

Інтелектуальне ядро реалізовано у файлі detector.py із використанням бібліотеки Scikit-learn. Ключовою особливістю розробленого модуля є механізм первинної ініціалізації. Під час першого запуску застосунок перевіряє наявність збережених ваг моделі формату .pkl. У разі їх відсутності скрипт автоматично переходить у режим навчання: збирає еталонну вибірку легітимного трафіку, проводить нормалізацію даних через StandardScaler та навчає алгоритм Isolation Forest.

У робочому режимі аналізатор приймає вектор ознак від модуля PyShark, нормалізує його та обчислює оцінку аномальності за допомогою методу decision_function(). Якщо отримане значення падає нижче динамічно

розрахованого порогу, пакет маркується як загроза. Основний фрагмент функції класифікації наведено у лістингу 3.2.

```
1. def analyze_packet(packet_info: dict) -> dict:
2.     features = [packet_info["packet_size"],
packet_info["packet_rate"], packet_info["entropy"]]
3.     score = float(ml_model.decision_function([features])[0])
4.
5.     status = "safe"
6.     if score < ANOMALY_THRESHOLD:
7.         status = "dangerous" if score < CRITICAL_THRESHOLD else
"unsafe"
8.
9.     return {"status": status, "score": score, "danger_level":
get_level(status)}
```

Лістинг 3.2 - Програмна оцінка аномальності вектора ознак

3.3.3 Взаємодія з базою даних SQLite

Для забезпечення консистентності даних та збереження доказів атак реалізовано модуль database.py. Взаємодія з СУБД SQLite здійснюється через вбудовану бібліотеку sqlite3. Програмна логіка суворо дотримується принципу мінімізації дискових операцій.

Легітимні пакети генерують лише базовий запис (INSERT) до таблиці Logs. Однак, якщо ML-модуль повертає статус аномалії, ініціюється комплексна транзакція: створюється запис у таблиці Anomaly, а сирі байти перехопленого

пакета конвертуються та записуються до таблиці Payload для подальшого розслідування.

3.3.4 Реалізація вебінтерфейсу та сповіщень

Вебінтерфейс забезпечує взаємодію адміністратора з ядром системи через браузер. Серверну частину побудовано на мікрофреймворку Flask. Для того, щоб адміністратор міг миттєво реагувати на загрози, традиційна REST-архітектура була доповнена технологією WebSocket.

Щойно аналітичний модуль реєструє інцидент у базі даних, він генерує тригерну подію. Flask-сервер перехоплює її та відправляє push-сповіщення на клієнтську частину без необхідності перезавантаження сторінки. На боці фронтенду ця подія оновлює DOM-дерево, додаючи червоний статусний індикатор до таблиці інцидентів та оновлюючи графіки активності..

3.4 Тестування та оцінка ефективності виявлення аномалій

Метою тестування є перевірка здатності розробленого програмного забезпечення ідентифікувати мережеві загрози в реальному часі та оцінка точності роботи алгоритму Isolation Forest при обробці нетипових пакетних послідовностей.

Після ініціалізації системи проводився збір фонового трафіку локальної мережі. На дашборді вебінтерфейсу фіксувалася стабільна активність із переважним використанням протоколів UDP (DNS, MDNS), TCP та TLS Рисунки 3.7 та 3.8 демонструють, що переважна більшість пакетів класифікується як безпечна. Оцінка аномальності для легітимних з'єднань знаходилася в діапазоні значень від 0.1 до 0.3, що є нормою.

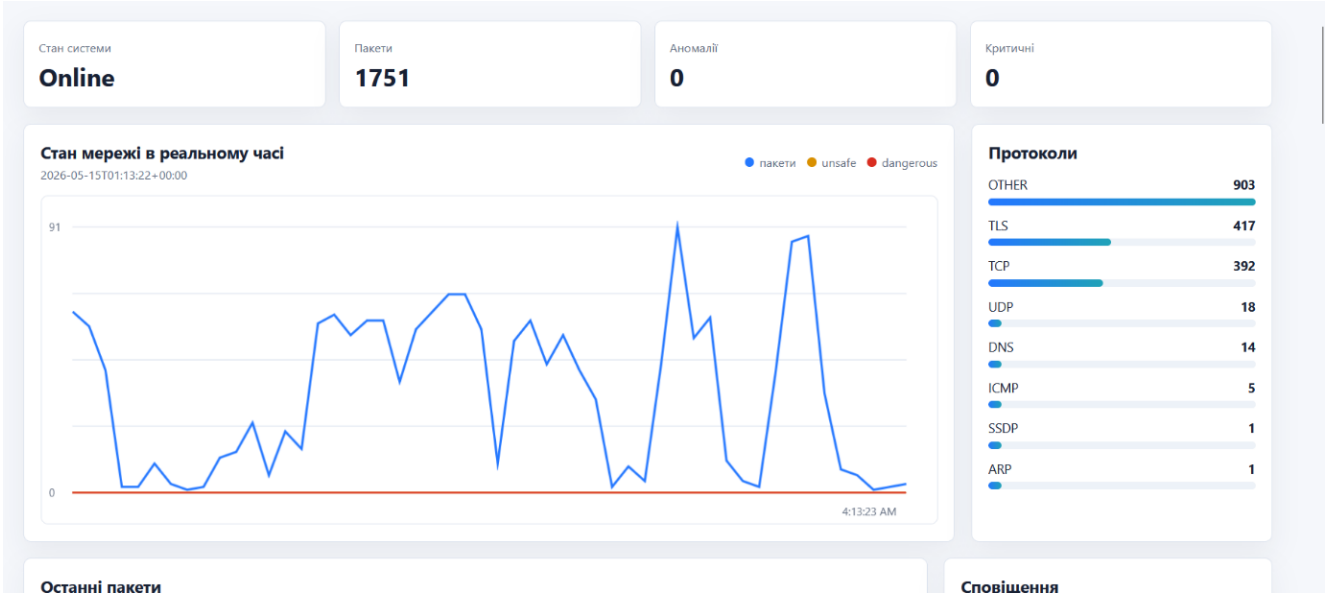

Рис 3.7 Загальний вигляд системи моніторинг

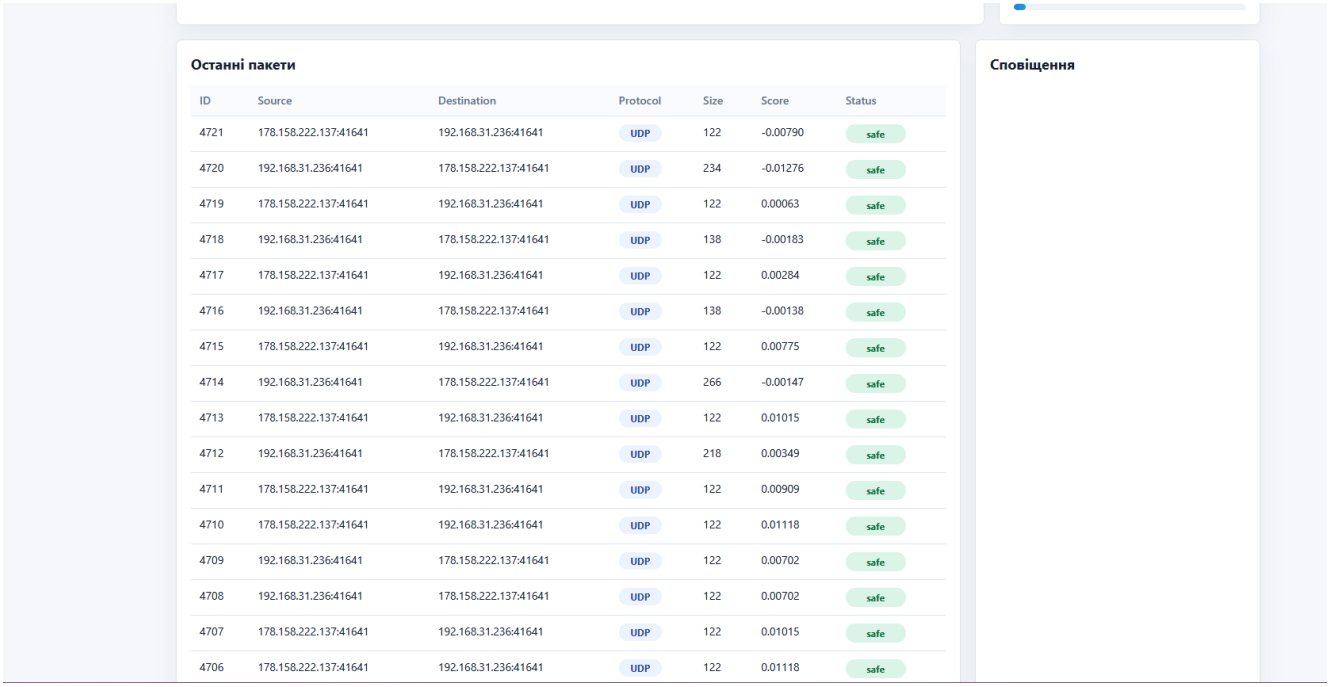

Рис 3.8 Нормальний трафік

Для перевірки реакції системи проведено стрес-тестування за допомогою спеціалізованого скрипта network_attack_simulator.py. Повний лістинг тестового скрипта наведено у Додатку Б. Тестування здійснювалося з віддаленого обчислювального вузла через VPN-з'єднання на зовнішню адресу сервера моніторингу (100.81.209.40). Скрипт ініціював багатопотокову генерацію аномалій:

TCP/UDP флуд, сканування діапазону портів (1-2048), HTTP-спам по API-ендпоінтах застосунку та імітацію beacon-активності ботнету.

Аналітичне ядро миттєво зафіксувало відхилення в структурі мережевих потоків. Значну частину трафіку на графіку активності було класифіковано як потенційно небезпечну, а в моменти пікового навантаження - як критичну загрозу. Із понад 44 тисяч пакетів, перехоплених під час тестування, 3 186 було ідентифіковано як аномалії. Детальний вигляд вебінтерфейсу з результатами виявлення інцидентів наведено на рисунку 3.9.

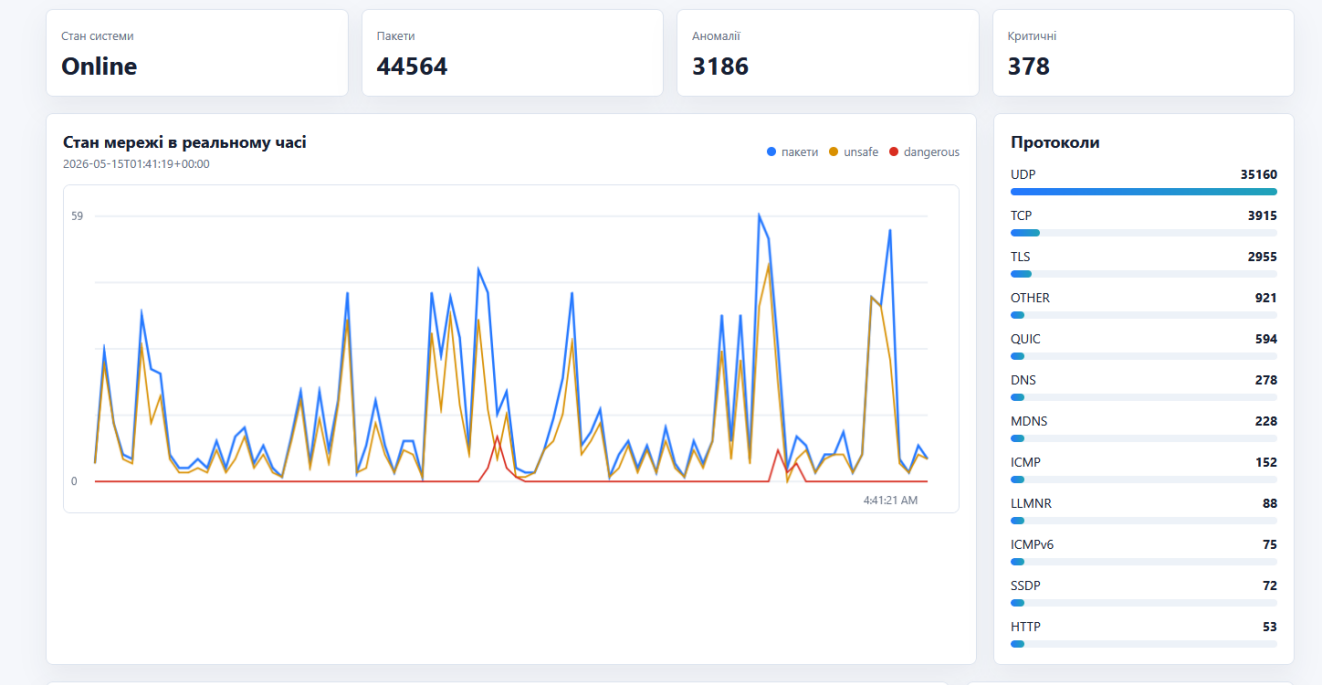

Рис 3.9 Результати стрес тестів

Виявлені загрози були задокументовані в журналі подій, що відображено на рисунку 3.10. Система ідентифікувала найбільш активні джерела атак, серед яких адреси 178.158.222.137 та 104.18.32.47. Для кожного аномального пакета в базі даних SQLite зафіксовано тип загрози та збережено вміст корисного навантаження в таблиці Payload для подальшої форензики.

Алгоритм Isolation Forest зреагував на сплеск з'єднань та нетипові розміри пакетів падінням показника Anomaly Score нижче нуля. Зафіксовано перетин порогових ліній unsafe threshold та dangerous threshold. Зокрема, для

послідовностей TCP-пакетів з високою ентропією корисного навантаження похибка реконструкції (у термінах оцінки відхилень) досягла значень -0.18682 та -0.04888, що автоматично перевело їх у статус «critical».

47555	192.168.31.236:49320	104.18.32.47:443	QUIC	1283	-0.07228	dangerous
47554	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.09188	dangerous
47553	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.07473	dangerous
47552	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.06956	dangerous
47551	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.08866	dangerous
47550	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.09022	dangerous
47549	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.08997	dangerous
47548	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.09188	dangerous
47547	192.168.31.236:49320	104.18.32.47:443	QUIC	1294	-0.08665	dangerous
47546	192.168.31.236:50854	149.154.167.41:443	TCP	54	-0.18384	dangerous
47545	149.154.167.41:443	192.168.31.236:50854	TCP	159	-0.11179	safe
47544	192.168.31.236:55142	192.200.0.110:80	TCP	54	-0.18682	dangerous
47543	192.200.0.110:80	192.168.31.236:55142	TCP	117	-0.14075	unsafe
47542	192.168.31.236:49320	104.18.32.47:443	QUIC	86	-0.18088	dangerous

Рис 3.10 Журнал подій під час тестової атаки

Програмна реалізація оновлення графіків через WebSocket дозволила візуалізувати атаку в реальному часі без затримок обробки. Динамічна діаграма розподілу протоколів підтвердила аномальне зростання частки UDP та TCP-сегментів під час симуляції флуду. Результати випробувань демонструють високу чутливість розробленого інтелектуального ядра до комбінованих методів впливу на мережеву інфраструктуру. Проте, воно може бути надмірно чутливим, і потребує корекцій та подальших випробувань.

Після випробувань та детекцій аномалій, візуалізація роботи за допомогою графіків, побудованих із використанням бібліотеки matplotlib є дуже зручною. На них відображено ключові метрики: джерела з найвищою мережевою активністю, оцінки системи Isolation Forest по коефіцієнтах, та співвідношення безпечних пакетів, до небезпечних і загрозливих. Відповідні графічні панелі наведено на рисунку 3.11.

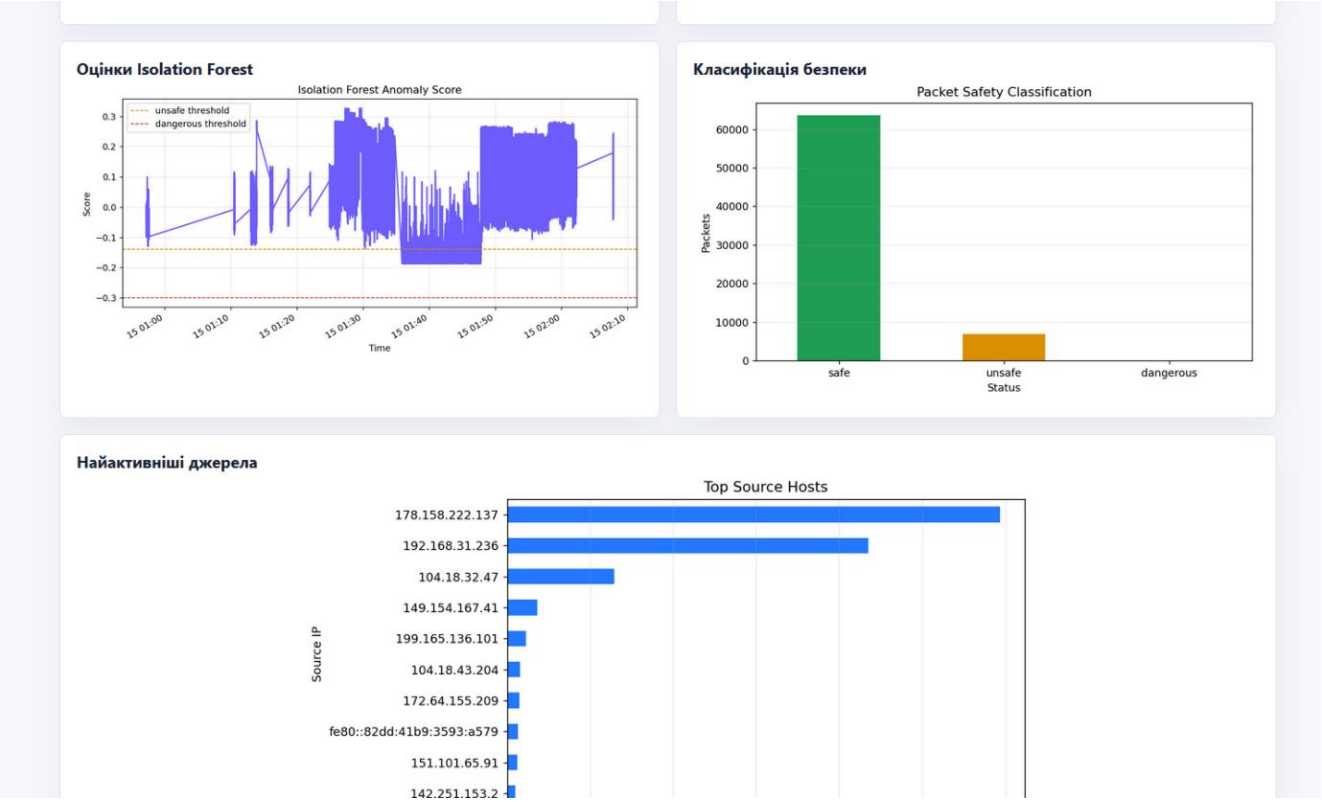

Рис 3.11 Графіки результатів роботи

Результати стрес-тестування підтверджують коректне функціонування розробленого застосунку та його відповідність поставленим вимогам. Система успішно ідентифікувала аномалії та забезпечила їх реєстрацію у журналі подій. За результатами тестування сформовано план подальшої оптимізації застосунку та усунення виявлених недоліків.

ВИСНОВКИ

У дипломній роботі проведено комплексне дослідження методів моніторингу мережевої безпеки та розроблено програмний комплекс для автоматизованого виявлення аномалій у мережевому трафіку на основі алгоритмів машинного навчання. За результатами виконання роботи сформовано такі висновки:

Аналіз сучасного стану кіберзагроз підтвердив недостатню ефективність класичних сигнатурних засобів захисту проти сучасних багатовекторних атак. Статистичний аналіз показав, що понад 65% інцидентів пов'язані з методами соціальної інженерії та складним шкідливим ПЗ, що вимагає впровадження інтелектуальних систем проактивного моніторингу.

Обґрунтовано вибір технологічного стеку та методів аналізу. Визначено, що використання ансамблевого алгоритму Isolation Forest є найбільш доцільним для ідентифікації аномалій «нульового дня», оскільки він дозволяє виявляти відхилення без попереднього розмічування бази атак. Обрана мова програмування Python та бібліотека PyShark забезпечили необхідну швидкодію для обробки пакетів у реальному часі.

Розроблено модульну архітектуру системи, яка базується на принципі розподілу відповідальності. Це дозволило ізолювати процеси захоплення трафіку, ML-аналізу та візуалізації, забезпечивши стабільність роботи застосунку при високих мережевих навантаженнях. Впровадження режиму самонавчання (Self-Learning Mode) дозволило системі автономно адаптуватися до специфіки конкретного мережевого середовища.

Реалізовано систему сповіщень та вебінтерфейс, що забезпечують візуалізацію мережевих процесів через WebSockets. Це дозволило адміністратору отримувати критичні сповіщення миттєво, без затримок на оновлення сторінок, та проводити глибокий аналіз інцидентів завдяки збереженню корисного навантаження (Payload) аномальних пакетів у базі даних SQLite.

Проведено експериментальну перевірку працездатності системи. Під час стрес-тестування із застосуванням багатопотокової генерації атак через VPN-тунель, розроблений застосунок продемонстрував високу точність детекції. Алгоритм Isolation Forest успішно ідентифікував TCP/UDP флуд, сканування портів та HTTP-спам, зафіксувавши падіння оцінки аномальності до критичних значень (нижче -0.18).

Практична цінність роботи полягає у створенні автономного інструмента моніторингу, який може бути інтегрований у локальні мережі підприємств для раннього виявлення кіберінцидентів. Подальший розвиток проєкту може бути спрямований на розширення вектора ознак для аналізу зашифрованого трафіку та впровадження механізмів автоматичного блокування джерел атак на рівні системного брандмауера.

ПЕРЕЛІК ПОСИЛАНЬ

1. Uk government: Cyber security breaches survey 2025 *https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2025/cyber-security-breaches-survey-2025#chapter-6-cyber-crime*
2. Microsoft: Що таке фішинг? *https://www.microsoft.com/uk-ua/security/business/security-101/what-is-phishing*
3. National Institute of Standards and Technology. Digital Identity Guidelines. NIST Special Publication 800-63-4. Gaithersburg, MD, 2025. DOI: 10.6028/NIST.SP.800-63-4. *https://doi.org/10.6028/NIST.SP.800-63-4*
4. Voskoboinyk V. O., Savchenko Iu. V., Karpukov L. M., Parshyna O. A., Prokopovych-Tkachenko D. I. Assessment of the State of Information Security Using Expert Systems // Системи та технології. 2024. № 1(67). С. 73–84. DOI: *https://doi.org/10.32782/2521-6643-2024-1-67.11*
5. Cloudflare. What is Malware? URL: *https://www.cloudflare.com/learning/ddos/glossary/malware/*
6. *Kumar Shankar, Singh Nandeshwar Pd., Kumar Narendra. Literature Review of Distributed Denial of Service (DDoS) Attacks, its Detection Techniques and Prevention Mechanisms // International Journal for Research in Applied Science and Engineering Technology. 2022. Vol. 10, Issue 9. P. 1681–1685. DOI: https://doi.org/10.22214/ijraset.2022.46882*
7. IBM. What is SIEM? URL: *https://www.ibm.com/think/topics/siem*
8. Stouffer K., Pease M., Tang C. та ін. Guide to Operational Technology (OT) Security. NIST Special Publication 800-82r3. Gaithersburg, MD, 2023. DOI: https://doi.org/10.6028/NIST.SP.800-82r3
9. PyShark Documentation / KimiNewt. PyShark: Python wrapper for tshark, allowing python packet parsing. URL: https://github.com/KimiNewt/pyshark
10. Python Software Foundation. Sqlite3 DB-API 2.0 interface for SQLite databases. Python Documentation. URL: https://docs.python.org/3/library/sqlite3.html
11. Pallets Project. Flask Documentation. URL: https://flask.palletsprojects.com/en/stable/
12. Matplotlib Development Team. Matplotlib Documentation. URL: https://matplotlib.org/stable/tutorials/index.html\
13. Pandas development team. Pandas documentation. URL: https://pandas.pydata.org/docs/
14. Scikit-learn: Machine Learning in Python. User Guide. URL: https://scikit-learn.org/stable/user_guide.html

ДОДАТОК А. ПРОГРАМНИЙ КОД МОДУЛІВ СИСТЕМИ ТА ЗАСОБІВ ТЕСТУВАННЯ

```
1.  // 1. Мапінг статусів для присвоєння CSS-класів бейджам у таблиці
2.  const STATUS_MAP = {
3.      'safe': { class: 'bg-green-100 text-green-800', label: 'safe' },
4.       'unsafe': { class: 'bg-yellow-100 text-yellow-800', label: 'warning' },
5.         'dangerous': { class: 'bg-red-100 text-red-800', label: 'critical' }
6.  };
7.
8.  // 2. Оновлення датасетів графіка Chart.js на основі нових даних
9.  function pushToChart(chart, payload) {
10.     const timestamp = new Date().toLocaleTimeString();
11.     chart.data.labels.push(timestamp);
12.
13.     // Розподіл значень за трьома лініями графіка
14.      chart.data.datasets[0].data.push(payload.total_packets); // синя лінія
15.    chart.data.datasets[1].data.push(payload.status === 'unsafe' ? 1 : 0); // жовта
16.             chart.data.datasets[2].data.push(payload.status   === 'dangerous' ? 1 : 0); // червона
17.
18.     chart.update('none'); // оновлення без зайвої анімації для швидкодії
19. }
```

ДОДАТОК Б. ПРОГРАМНИЙ КОД ТЕСТОВОГО СКРИПТА ДЛЯ ГЕНЕРАЦІЇ АНОМАЛЬНОГО ТРАФІКУ

```
# network_attack_simulator.py
# Локальний симулятор мережевих атак для тестування IDS/IPS

import socket
import threading
import random
import time
import requests

TARGET_IP = "100.81.209.40"   # IP машини з IDS (VPN-тунель)
WEB_PORT = 5000               # Flask/Web interface

RUNNING = True

def tcp_flood():
    while RUNNING:
        try:
            port = random.randint(1, 65535)
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(0.05)
            s.connect_ex((TARGET_IP, port))
            payload = random._urandom(random.randint(64, 2048))
            try:
                s.send(payload)
            except:
                pass
            s.close()
        except:
```

```
29.             pass
30.
31. def udp_flood():
32.     sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
33.     while RUNNING:
34.         try:
35.             payload = random._urandom(random.randint(256, 4096))
36.             port = random.randint(1, 65535)
37.             sock.sendto(payload, (TARGET_IP, port))
38.         except:
39.             pass
40.
41. def http_spam():
42.     paths = ["/", "/dashboard", "/logs", "/api/stats", "/api/alerts",
"/api/network", "/api/anomalies"]
43.     while RUNNING:
44.         try:
45.             url =
f"http://{TARGET_IP}:{WEB_PORT}{random.choice(paths)}"
46.             requests.get(url, timeout=0.2)
47.         except:
48.             pass
49.
50. def port_scan():
51.     while RUNNING:
52.         try:
53.             for port in range(1, 2048):
54.                 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
55.                 s.settimeout(0.01)
56.                 s.connect_ex((TARGET_IP, port))
```

```
57.                 s.close()
58.         except:
59.             pass
60.
61. def beacon_simulation():
62.     while RUNNING:
63.         try:
64.             s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
65.             s.settimeout(1)
66.             s.connect((TARGET_IP, WEB_PORT))
67.             beacon_payload = b"botnet_beacon_ping"
68.             s.send(beacon_payload)
69.             s.close()
70.             time.sleep(2)
71.         except:
72.             time.sleep(1)
73.
74. def random_connections():
75.     while RUNNING:
76.         try:
77.             port = random.randint(1, 65535)
78.             proto = random.choice(["tcp", "udp"])
79.             if proto == "tcp":
80.                 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
81.                 s.settimeout(0.02)
82.                 s.connect_ex((TARGET_IP, port))
83.                 s.close()
84.             else:
85.                 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
86.                 data = random._urandom(random.randint(32, 512))
87.                 s.sendto(data, (TARGET_IP, port))
88.         except:
89.             pass
90.
91. def lateral_movement():
92.     targets = [TARGET_IP, "192.168.0.11", "192.168.0.12",
"192.168.0.13"]
93.     common_ports = [21, 22, 23, 53, 80, 135, 139, 443, 445, 3389]
94.     while RUNNING:
95.         for ip in targets:
96.             for port in common_ports:
97.                 try:
98.                     s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
99.                     s.settimeout(0.03)
100.                     s.connect_ex((ip, port))
101.                     s.close()
102.                 except:
103.                     pass
104.
105. def start_attack(name, func, count):
106.     print(f"[+] Starting {name} x{count}")
107.     for _ in range(count):
108.         t = threading.Thread(target=func)
109.         t.daemon = True
110.         t.start()
111.
112. def main():
```

```
    print("=" * 60)
    print(" IDS/IPS ATTACK SIMULATOR ")
    print("=" * 60)
    print(f"[TARGET] {TARGET_IP}\n")
    start_attack("TCP Flood", tcp_flood, 5)
    start_attack("UDP Flood", udp_flood, 5)
    start_attack("HTTP Spam", http_spam, 3)
    start_attack("Port Scan", port_scan, 2)
    start_attack("Beacon Simulation", beacon_simulation, 2)
    start_attack("Random Connections", random_connections, 3)
    start_attack("Lateral Movement", lateral_movement, 1)
    print("\n[+] Attack simulation is running")
    print("[+] Press CTRL+C to stop\n")
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        global RUNNING
        RUNNING = False
        print("\n[!] Stopping simulator...")
        print("[!] Threads shutting down...")
        print("[!] Done")

if __name__ == "__main__":
    main()
```