МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.5:004.056.57:004.4

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Система захисту об'єктів критичної інфраструктури

Виконавець: Антон КРУЧИНСЬКИЙ

Керівник: д.т.н., професор, завідувач кафедри КСЗІ Михайло СТЕПАНОВ

Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА

Нормоконтролер: к.т.н., доцент Анна ІЛЬЄНКО

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
здобувача вищої освіти Кручинського Антона Олександровича

1. Тема: *Система захисту об'єктів критичної інфраструктури*
затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: провести порівняльний аналіз методів захисту критичної інформаційної інфраструктури; на основі результатів аналізу визначитись з методом для розробки системи захисту об'єктів критичної інфраструктури; створити аналізатор мережевого трафіку; протестувати розроблений застосунок.
4. Зміст пояснювальної записки: аналіз методів виявлень вторгнень; розробка методики для проведення досліджень; обґрунтування вибору методу для розробки системи захисту об'єктів критичної інфраструктури; створення та конфігурація програмного застосунку; проведення тестування аналізатору мережевого трафіку.

5. КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	17.10.2023 – 18.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	19.10.2023 – 20.10.2023	*Виконано*
4.	Збір інформації	23.10.2023	*Виконано*
5.	Аналіз методів виявлень вторгень у критичну інформаційну інфраструктуру	24.10.2023 – 27.10.2023	*Виконано*
6.	Визначення створення аналізатору мережевого трафіку як ключову розробку для кваліфікаційної роботи; розробка застосунку	30.10.2023 – 01.12.2023	*Виконано*
7.	Тестування розробленого методу	04.12.2023 – 08.12.2023	*Виконано*
8.	Передзахист кваліфікаційної роботи	12.12.2023	*Виконано*
9.	Перевірка на антиплагіат	13.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	15.12.2023	*Виконано*
11.	Оформлення презентації	17.12.2023	*Виконано*
12.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Антон КРУЧИНСЬКИЙ

(підпис, дата)

Керівник кваліфікаційної роботи Михайло СТЕПАНОВ

(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота охоплює вступ, два розділи, загальні висновки, список використаних джерел, додатки і включає 96 сторінок основного тексту, 31 рисунок, 3 таблиці, 25 сторінок додатків. Список використаних джерел налічує 42 найменування та займає 5 сторінок. Загальний обсяг роботи становить 121 сторінку.

Метою даного дослідження є проведення аналізу загроз критичній інформаційній інфраструктурі, і на його основі здійснення розробки системи захисту об’єктів критичної важливості з ціллю підвищення їхньої безпеки та стійкості від внутрішніх та зовнішніх загроз.

В роботі вирішено завдання створення комплексної системи для виявлення аномальної поведінки в мережевому трафіку. Розроблено методологічні основи дослідження, включаючи критерії вибору технологій та інструментів розробки. Використано мову програмування C# та базу даних MS SQL Server.

Розроблені методи та програмне забезпечення можуть знайти застосування в галузях, де критична інформаційна інфраструктура є ключовим компонентом: енергетика, транспорт, медицина, фінанси та інші.

Можливі вектори подальшого розвитку даної роботи асоціюються з адаптацією розробленого програмного забезпечення до міжнародних стандартів кібербезпеки, таких як ISO/IEC 27001, та інтеграції з існуючими системами захисту.

Ключові слова: критична інформаційна інфраструктура, система захисту, аномальна поведінка, мережевий трафік, C#, MS SQL Server, кібербезпека.

ЗМІСТ

ВСТУП

Захист об'єктів критичної інфраструктури стає все більш актуальним завданням у сучасному світі, де інформаційні технології проникають у всі сфери життя. Об'єкти критичної інфраструктури — це системи та активи, як фізичні, так і віртуальні, чия непродуктивність або руйнування може мати серйозний негативний вплив на національну безпеку, економіку та здоров'я населення [1]. В Україні, як і в інших країнах, це включає в себе енергетичні системи, транспортну інфраструктуру, системи охорони здоров'я, фінансові інституції та інше.

Актуальність теми визначається декількома ключовими факторами. По-перше, сучасний розвиток технологій приводить до зростання кількості потенційних загроз для критично важливих систем. По-друге, у зв'язку з глобалізацією і збільшенням міжнародної взаємозалежності, наслідки атак на об'єкти критичної інфраструктури можуть мати транснаціональний характер. По-третє, в Україні існують певні специфічні виклики, такі як військовий конфлікт на сході країни, який робить задачу захисту ще більш нагальною.

З огляду на вищезазначене, існує ряд відкритих наукових та практичних проблем, які вимагають термінового вирішення. Серед них — розробка ефективних механізмів захисту, інтеграція різних систем безпеки в єдиний комплекс, розробка нормативно-правової бази для регулювання цієї сфери, а також підготовка кваліфікованих кадрів, здатних ефективно управляти системами захисту.

У відомих розв'язаннях проблеми часто фокусуються на технологічних аспектах, проте не завжди враховуються соціальні, економічні та політичні фактори, які є невід'ємною частиною комплексного заходу захисту. Тому, з урахуванням геополітичної ситуації та національних особливостей України, актуальність дослідження даної теми є беззаперечною.

Метою кваліфікаційної роботи є проведення аналізу загроз критичній інформаційній інфраструктурі, і на його основі здійснення розробки системи захисту об’єктів критичної важливості з ціллю підвищення їхньої безпеки та стійкості від внутрішніх та зовнішніх загроз.

Для вирішення поставленої мети необхідно вирішити такі **завдання**:

– провести аналіз теоретичних основ забезпечення безпеки критичної інформаційної інфраструктури, що включає в себе ретельний огляд наукових досліджень, виявлення невирішених питань та розгляд правових аспектів відповідно до чинного законодавства;

– здійснити аналіз сучасних методів і систем захисту інформації в рамках критичної інфраструктури, оцінити їх ефективність та придатність;

– розробити програмне забезпечення на основі статистичного методу для виявлення аномальної поведінки в комп'ютерних мережах, що враховує специфіку роботи об'єктів критичної інфраструктури;

– оцінити ефективність і достовірність розробленого програмного забезпечення шляхом тестування його здатності виявляти потенційні загрози та попереджати про них.

Ці задачі спрямовані на реалізацію поставленої мети та взаємопов'язані таким чином, що успішне вирішення кожної з них є кроком на шляху до досягнення кінцевої мети дослідження.

Об'єктом даного дослідження є процес забезпечення інформаційної безпеки на об'єктах критичної інфраструктури.

Предметом дослідження є методи та системи, які використовуються для забезпечення інформаційної безпеки на об'єктах критичної інфраструктури.

Наукова новизна полягає у розробці статистичного методу для виявлення аномальної поведінки в комп'ютерній мережі, з орієнтацією на об'єкти критичної інфраструктури. Метод характеризується спеціальною властивістю: здатністю ефективно виявляти потенційні вторгнення або інші загрози без необхідності ручного налаштування, що підвищує рівень захищеності від навмисних пошкоджень.

Галузь застосування даного дослідження можна виділити наступним чином:

– критична інфраструктура. Розроблений статистичний метод для виявлення аномальної поведінки в мережі може бути впроваджений у системах захисту енергетичних установ, фінансових організацій, системах охорони здоров'я, транспортних вузлах та інших об'єктах критичної інфраструктури;

– державні установи та органи влади. З урахуванням надзвичайної важливості інформаційної безпеки для державних організацій, метод може бути використаний для захисту інформаційних систем урядових органів;

– промислові підприємства. На виробничих підприємствах часто використовуються спеціалізовані системи управління, від безпеки яких залежить не лише фінансовий результат, але і життя людей;

– телекомунікаційні компанії. Інформаційні мережі є критичним компонентом для багатьох секторів економіки, і їх безпека має прямий вплив на стабільність та ефективність роботи багатьох систем;

– науково-дослідні інституції. Розроблений метод може бути використаний в наукових дослідженнях для подальшого вдосконалення методик виявлення та протидії кіберзагрозам.

Оскільки метод базується на статистичному аналізі та не вимагає спеціалізованого обладнання для імплементації, його можна впровадити в широкому спектрі об'єктів, починаючи від малих локальних мереж до великих корпоративних систем. Така універсальність робить метод особливо цінним для підвищення рівня кібербезпеки в різних секторах економіки та суспільного життя.

Практична цінність полягає у тому, що розроблений застосунок має здатність надати вичерпну інформацію для управління та підтримки безпеки мережі критичної інфраструктури, з подальшою оптимізацією її роботи та виявленням проблем, що є ключовими факторами для забезпечення захисту критично важливих структур від втрати та викрадення важливих даних, а також попередження від неправомірних дій збоку недоброчесних користувачів мережі.

Розділ 1. ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ ОБ'ЄКТІВ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

1.1. Правові основи поняття критичної інформаційної інфраструктури

Правові основи поняття КІІ займають ключове місце у забезпеченні національної безпеки та стабільності держави. Вони визначають не тільки правовий статус об'єктів КІІ, але й встановлюють конкретні рамки для компетенцій органів державної влади, які наділені відповідальністю за забезпечення безпеки цих об'єктів. Окрім того, ці правові норми окреслюють права та обов'язки учасників, що залучені до взаємодії з критичною інфраструктурою, включаючи операторів, користувачів, та інших стейкхолдерів.

У багатьох юрисдикціях, включаючи Україну, було розроблено та введено в дію законодавчі та підзаконні акти, які регламентують питання кібербезпеки КІІ. Ці нормативні акти мають на меті не тільки визначити критерії віднесення об'єктів до категорії КІІ, але й встановлюють вимоги до їх захисту від можливих кіберзагроз. Проте, часто це законодавство характеризується як фрагментарне та недостатньо систематизоване, що ускладнює процес його застосування та інтерпретації. Такі недоліки можуть призвести до прогалин у законодавстві, що у свою чергу знижує ефективність захисту критично важливих об'єктів від сучасних та еволюціонуючих кіберзагроз.

Особливу увагу слід звернути на необхідність комплексного підходу до регулювання кібербезпеки КІІ, що включає розробку та впровадження узгоджених стандартів, методологій оцінки ризиків, а також встановлення чітких процедур відповіді на інциденти кібербезпеки. Важливим є також забезпечення постійного оновлення та адаптації законодавства, щоб воно відповідало зростаючим та змінюючимся кіберзагрозам, а також сприяло створенню сильної та резилієнтної системи захисту критичної інфраструктури.

В Україні основним законодавчим актом, що регулює питання кібербезпеки об'єктів КІІ є Закон України "Про основні засади забезпечення кібербезпеки України" [2]. Закон визначає ключові поняття та терміни, що використовуються в галузі кібербезпеки, а також регулює взаємодію між органами державної влади, підприємствами, установами та організаціями у цій сфері.

Цей закон встановлює концептуальні засади державної політики в галузі кібербезпеки. Він визначає структурні елементи кібербезпеки, зокрема, що стосується об'єктів критичної інфраструктури. Однак, відзначимо, що закон не надає чіткого переліку таких об'єктів, що може ускладнити процес їх ідентифікації та захисту.

Закон також передбачає створення спеціалізованих державних органів та служб, які будуть займатися питаннями кібербезпеки. В той же час, існує питання координації їхньої діяльності, а також їх взаємодії з приватним сектором.

Можна виділити наступні ключові положення даного закону (рис. 1.1):

– визначення ключових термінів. Закон визначає основні поняття та терміни, які є критичними для розуміння та інтерпретації положень, що стосуються кібербезпеки. Це включає поняття кіберпростору, кіберінциденту, кіберзахисту, критичної інформаційної інфраструктури та інше;

– органи державної влади та їх компетенції. Закон регулює створення та функціонування спеціалізованих органів державного управління у сфері кібербезпеки, визначає їхні основні функції, повноваження та обов'язки;

– механізми реагування на кіберінциденти. Описані процедури та механізми реагування на кіберінциденти, а також координація дій між різними учасниками. Закон також передбачає необхідність ведення реєстру кіберінцидентів та проведення їх аналізу;

– юридична відповідальність. Закон встановлює систему юридичної відповідальності за порушення в сфері кібербезпеки. Хоча механізми

відповідальності описані, проте вони потребують подальшої деталізації та проробки;

– стандартизація та сертифікація. Закон наголошує на необхідності розробки національних стандартів у сфері кібербезпеки, які б повинні бути узгоджені з міжнародними нормами та стандартами;

– міжнародне співробітництво. Особливу увагу приділено міжнародному співробітництву в галузі кібербезпеки, зокрема в рамках білатеральних та багатосторонніх угод.

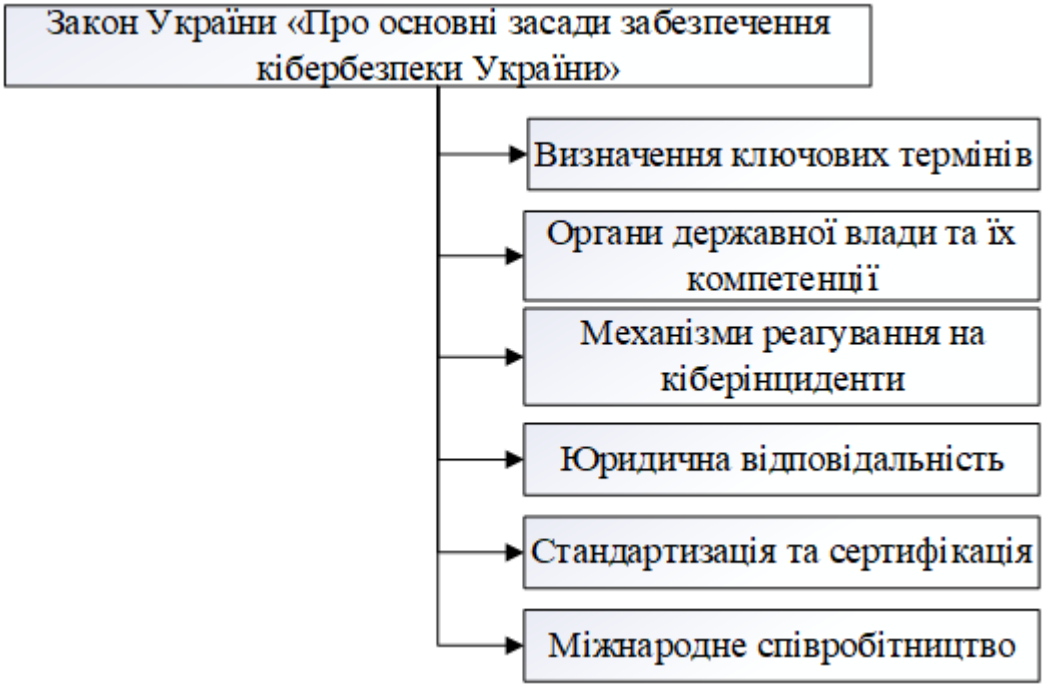

Рис. 1.1. Ключові положення Закон України "Про основні засади забезпечення кібербезпеки України"

Кожне з цих положень відкриває широкий простір для наукового аналізу, критичного огляду та подальшого удосконалення законодавчої бази в Україні.

Можна стверджувати, що даний Закон покликаний сприяти створенню умов для протидії кіберзлочинності та забезпеченню стійкості та безпеки критичної інформаційної інфраструктури. Однак, цей законодавчий акт має ряд

недоліків, зокрема, відсутність чіткої систематизації об'єктів критичної інфраструктури та механізмів їх захисту.

Також важливим є Закон України "Про Національну програму адаптації законодавства України до законодавства Європейського Союзу", який визначає стратегічні орієнтири та конкретні завдання з приведення національного законодавства відповідно до норм та стандартів Європейського Союзу. Основна мета цього закону — створення сприятливого правового середовища для реалізації Угоди про асоціацію між Україною та ЄС, а також для подальшого наближення України до європейських стандартів в різних сферах, включаючи кібербезпеку.

В рамках цього закону можуть бути розроблені та прийняті конкретні нормативні акти, які стосуються забезпечення кібербезпеки об'єктів критичної інфраструктури. Сюди можуть входити положення про обов'язкові стандарти безпеки, механізми моніторингу та реагування на кіберінциденти, а також зобов'язання щодо міжнародного співробітництва у цій сфері.

Так, цей закон не лише сприяє адаптації національного законодавства до європейських стандартів, але і підвищує рівень захищеності критично важливих інформаційних систем. Він може стати основою для розробки більш конкретних законодавчих ініціатив, які будуть цілеспрямовано регулювати питання кібербезпеки в Україні відповідно до міжнародних норм та практик.

Закон "Про Національну програму адаптації законодавства України до законодавства Європейського Союзу" включає в себе кілька положень, які мають прямий чи опосередкований вплив на питання кібербезпеки [3] (рис. 1.2):

– стратегічні орієнтири. Закон визначає загальні стратегічні напрями адаптації національного законодавства до стандартів ЄС, що має найширший вплив на всі сфери життя, включаючи кібербезпеку;

– конкретні завдання. Цей акт законодавства виокремлює конкретні завдання та етапи їхньої реалізації, зокрема в сфері забезпечення безпеки інформаційних систем;

– механізми моніторингу та контролю. Законодавчий акт встановлює механізми для моніторингу та оцінки ефективності імплементації європейських стандартів в національне законодавство;

– міжнародне співробітництво. Закон наголошує на важливості міжнародного співробітництва в процесі адаптації законодавства, що також може стосуватися обміну найкращими практиками у сфері кібербезпеки;

– санкції та відповідальність. Хоча цей аспект може бути не прямо вказаний у самому Законі, проте в контексті адаптації до європейських стандартів це може означати введення нових форм відповідальності за порушення в області кібербезпеки.

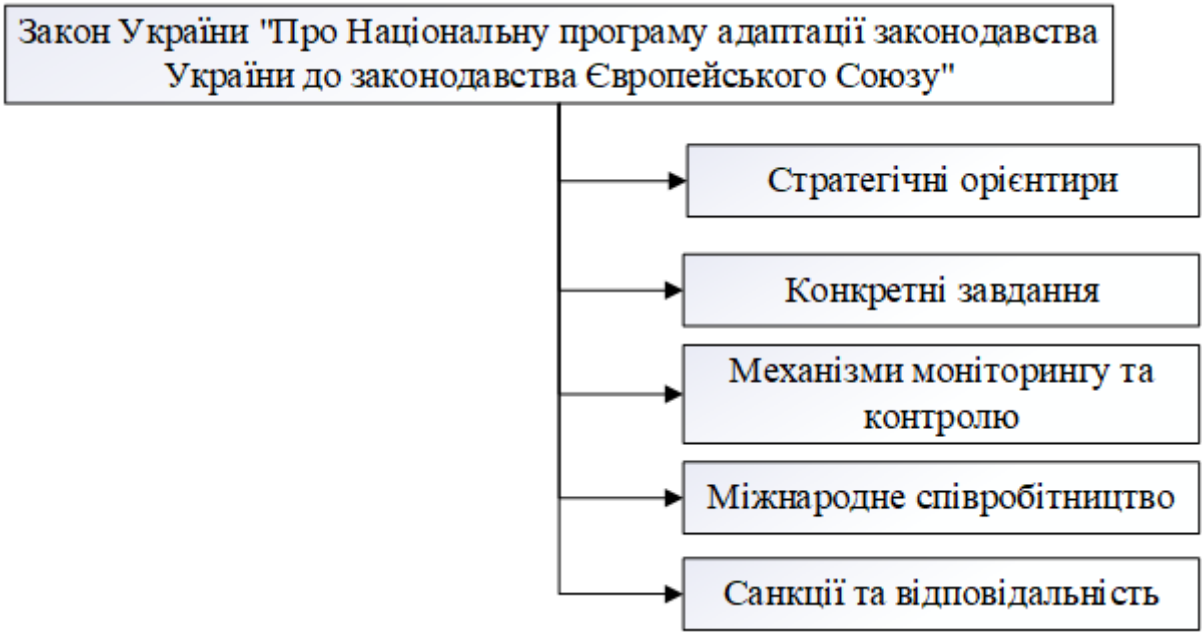

Рис. 1.2. Положення Закону України " Про Національну програму адаптації законодавства України до законодавства Європейського Союзу "

Ці положення є критичними для розуміння того, як Україна планує інтегрувати європейські норми та стандарти в своє національне законодавство з метою забезпечення вищого рівня кібербезпеки. Це особливо важливо для критичної інформаційної інфраструктури, яка вимагає особливого підходу та високого рівня захисту.

Обидва цих закони формують законодавчу базу для регулювання питань кібербезпеки в Україні, але їх ефективність у багатьох аспектах залежить від конкретних нормативно-правових актів, методів та технологій, які будуть застосовані для імплементації цих законів. Така ситуація вимагає подальшого наукового дослідження та аналізу для удосконалення існуючої системи законодавчого регулювання в галузі кібербезпеки.

На міжнародному рівні, варто звернути увагу на Директиву ЄС 2016/1148 щодо міри безпеки мереж та інформаційних систем [4]. Директива ЄС 2016/1148, відома як Директива NIS (Network and Information Systems), представляє собою один з ключових міжнародних законодавчих актів, що регулюють питання кібербезпеки. Цей документ став першою спробою Європейського Союзу створити уніфіковані правила для забезпечення безпеки мереж та інформаційних систем на території країн-членів [5].

Основні положення Директиви включають (рис. 1.3):

– встановлення загальних вимог до забезпечення безпеки КІІ. Директива пропонує єдиний набір вимог до безпеки для всіх країн-членів ЄС. Це включає стандартні процедури аудиту, методики оцінки ризиків, та механізми відстеження та реагування на інциденти кібербезпеки. Загальні вимоги сприяють створенню єдиного простору кібербезпеки в рамках ЄС, що забезпечує вищий рівень прозорості та довіри;

– організація спільних механізмів реагування на інциденти кібербезпеки. Директива передбачає створення міжнародних координаційних центрів для оперативного реагування на інциденти кібербезпеки. Це може включати в себе швидкий обмін інформацією про загрози, координацію дій для їх нейтралізації, а також розробку та впровадження засобів для відновлення нормального функціонування систем;

– координація дій між країнами-членами з метою виявлення та протидії кіберзагрозам. Цей аспект не тільки сприяє консолідації зусиль на міжнародному рівні, але й розширює можливості для оперативного реагування на загрози. Важливою частиною цієї координації є створення спільних баз даних

про кіберзагрози, які стають цінним ресурсом для аналізу та прогнозування потенційних інцидентів. Обмін інформацією та засобами її аналізу між країнами є ключовим для ідентифікації нових типів загроз та для розробки спільних стратегій протидії;

– відповідальність та зобов'язання власників та операторів об’єктів КІІ. Директива визначає не тільки права, але й обов'язки сторін, що відповідають за функціонування критичної інфраструктури. Це включає в себе необхідність регулярного інформування відповідних регуляторних органів про стан кібербезпеки, а також про всі інциденти, що мали місце. Така відповідальність покладається на операторів з метою посилення їх ролі в процесі забезпечення національної та міжнародної кібербезпеки.

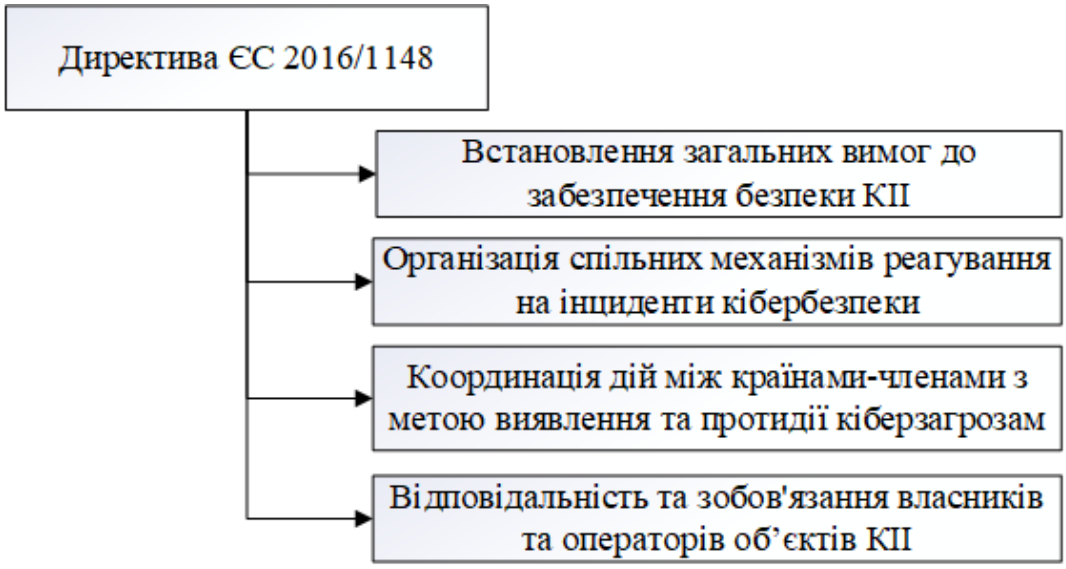

Рис. 1.3. Основні положення Директиви ЄС 2016/1148

Кожне з цих положень може слугувати відправною точкою для подальшого удосконалення національного законодавства України в області кібербезпеки.

Також необхідно врахувати рекомендації та стандарти Міжнародної організації стандартизації (ISO), зокрема ISO/IEC 27001, що є одним з найбільш важливих стандартів, що встановлює критерії для систем управління інформаційною безпекою (Information Security Management Systems) [6].

Стандарт включає в себе не тільки технічні аспекти, але і організаційні, що дозволяє створити комплексний підхід до захисту інформаційних активів.

Аналіз всіх положень ISO/IEC 27001 є складним та багатогранним завданням, оскільки цей стандарт є досить об'ємним та деталізованим. Проте, декілька ключових аспектів можна виділити (рис. 1.4):

– сфера застосування та контекст організації. Стандарт вимагає від організацій визначити межі та контекст системи управління інформаційною безпекою (ISMS). Це передбачає аналіз бізнес-процесів, стейкхолдерів, правових та регулятивних вимог;

– оцінка та управління ризиками. Однією з основних частин є процес оцінки ризиків, який повинен бути систематичним. Організації повинні ідентифікувати активи, загрози, вразливості та потенційні впливи, а також визначити адекватні заходи управління ризиками;

– розробка політики безпеки. Стандарт вимагає розробки та імплементації політики безпеки, яка була б узгоджена з загальними бізнес-цілями організації;

– організаційна структура. Стандарт зазначає, що повинні бути чітко визначені ролі, відповідальність, повноваження в області інформаційної безпеки;

– операційна діяльність та комунікація. Описує процедури та робочі інструкції, які повинні бути розроблені та дотримані для забезпечення безпеки при роботі з інформаційними системами;

– постійне вдосконалення: Стандарт закладає необхідність постійного вдосконалення системи управління безпекою, включаючи коректівні та запобіжні дії.

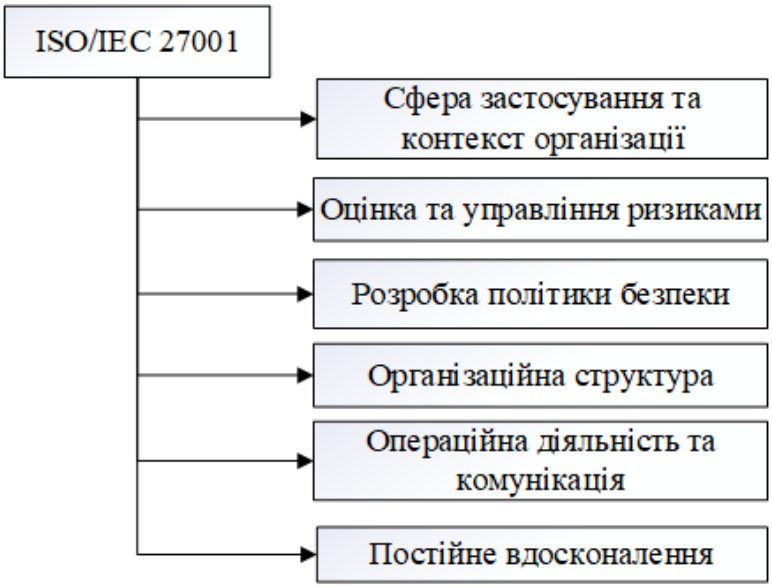

Рис. 1.4. Основні положення ISO/IEC 27001

Кожен з цих аспектів є критичним для створення ефективної системи управління інформаційною безпекою та може мати прямий вплив на забезпечення безпеки об'єктів критичної інформаційної інфраструктури. Застосування стандарту ISO/IEC 27001 може служити солідною основою для підвищення рівня кібербезпеки на національному та організаційному рівнях.

Врахування рекомендацій та вимог ISO/IEC 27001 є особливо актуальним для об'єктів критичної інформаційної інфраструктури. Зокрема, це стосується встановлення процесів управління ризиками, розробки політик безпеки, а також аудиту та моніторингу систем безпеки. Застосування цього стандарту може значно підвищити рівень захисту від кіберзагроз, забезпечити відповідність міжнародним нормам та зміцнити довіру між учасниками інформаційного простору.

В цілому, аналіз національного та міжнародного законодавства показує, що хоча і існують певні рамки для регулювання кібербезпеки об'єктів КІІ, існує необхідність їх подальшої деталізації, систематизації та адаптації до сучасних кіберзагроз.

1.2. Критична інформаційна інфраструктура як об'єкт забезпечення безпеки

В рамках сучасного законодавчого ландшафту України в сфері кібербезпеки, створено комплексну систему заходів, спрямованих на забезпечення цілісності, конфіденційності та доступності критичних інформаційних активів на території держави. Ця система включає в себе не тільки правові, але й технічні механізми для виявлення, превентивного впливу та ліквідації наслідків потенційних кібератак. Вона розроблена таким чином, щоб забезпечувати оперативний та ефективний взаємозв'язок між всіма зацікавленими сторонами: від операторів критичної інфраструктури до владних органів.

Оператори або власники критичних інформаційних активів мають обов'язок забезпечити надійний канал комунікації з компетентними державними органами. Це передбачає не лише обов'язкове та оперативне інформування про виникнення кіберінцидентів, але і активну участь в процесах їх ідентифікації, аналізу та нейтралізації.

У цьому аспекті ключову роль відіграє Державна служба експортного контролю України (ДСЕКУ), яка є централізованим органом виконавчої влади, що здійснює координацію дій та нагляд за виконанням законодавства в сфері кібербезпеки критичної інформаційної інфраструктури. ДСЕКУ має ряд специфічних повноважень, які дозволяють їй ефективно взаємодіяти з різними учасниками цієї системи. Ці повноваження та обов'язки, а також механізми їх реалізації, представлені в графічному відображенні на рис. 1.5.

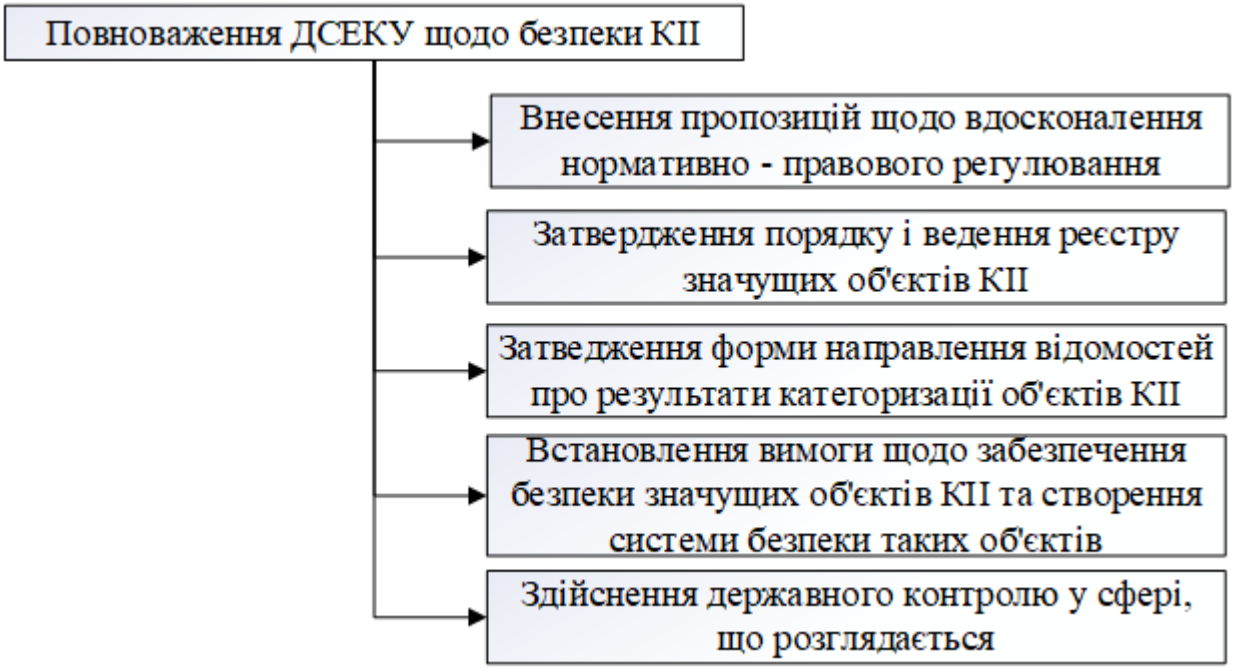

Рис. 1.5. Повноваження ДСЕКУ по забезпеченню безпеки КІІ

Власники критичних інформаційних ресурсів мають не тільки негайно інформувати компетентні органи про кіберінциденти, але й взаємодіяти з ними на етапах їх виявлення, запобігання та подальшої нейтралізації. Вони також зобов'язані забезпечувати надійну роботу відповідних систем і пристроїв, розроблених з метою забезпечення кібербезпеки.

У рамках національної стратегії забезпечення кібербезпеки, в Україні діє спеціалізована державна система. Ця система призначена для виявлення, превентивних заходів та ефективної нейтралізації наслідків кібератак на вітальні інформаційні ресурси. Основна задача цієї системи полягає у створенні надійного екрану захисту навколо критичної інформаційної інфраструктури держави.

Система конструюється з ряду ключових компонентів, які взаємодіють між собою для створення цілісної, адаптивної та реактивної схеми захисту. Ці компоненти і їх взаємозв'язки представлені на рис. 1.6. Кожен з цих компонентів відповідає за певну сферу захисту, починаючи від моніторингу та аналізу кіберзагроз і закінчуючи оперативним реагуванням на інциденти кібербезпеки.

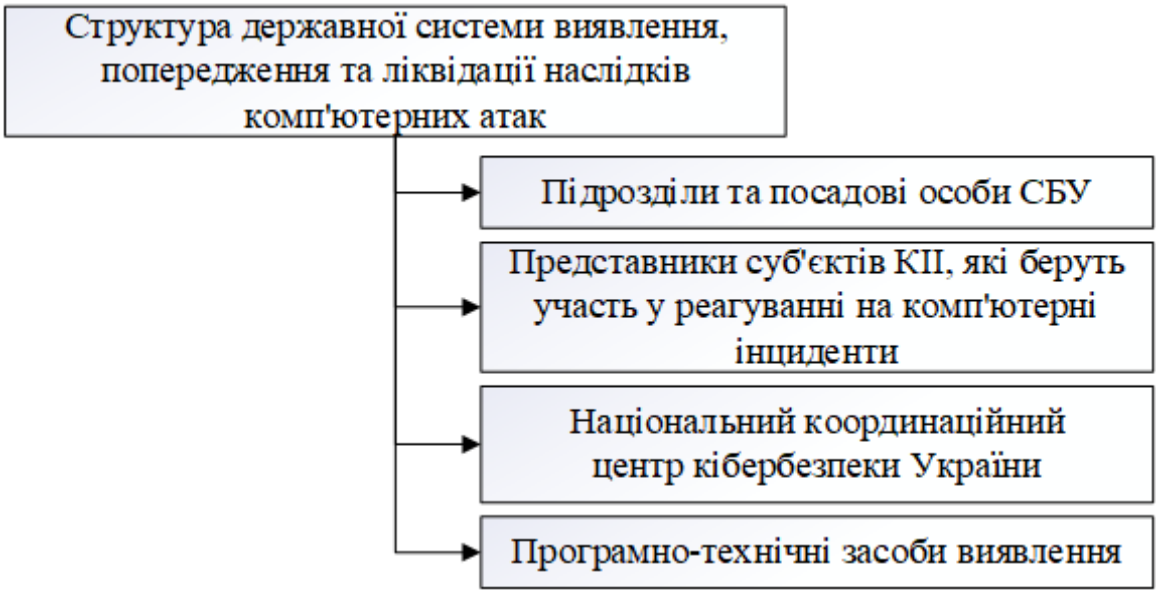

Рис. 1.6. Структура системи виявляння атак

В структурі системи забезпечення кібербезпеки КІІ можна виділити кілька ключових елементів, що взаємодіють для створення цілісної та ефективної моделі захисту [7].

Перш за все, Служба Безпеки України (СБУ) відіграє централізовану роль у координації дій з метою реагування на кіберзагрози. Підрозділи та посадові особи СБУ не тільки відстежують потенційні загрози, але й координують взаємодію із іншими суб'єктами системи.

Представники суб'єктів КІІ також активно залучені у процес реагування на комп'ютерні інциденти. Їх роль полягає в оперативному виявленні, аналізі та передачі інформації про інциденти до відповідних органів.

Національний координаційний центр кібербезпеки України є ще одним важливим елементом цієї системи. Центр забезпечує збір, аналіз та розподіл інформації про кіберзагрози, що дозволяє оперативно адаптувати стратегії захисту до змінюючогося ландшафту загроз.

Важливу роль у системі відіграють також програмно-технічні засоби виявлення, які забезпечують автоматизований моніторинг та аналіз мережевої

активності. Це дозволяє вчасно ідентифікувати аномалії та підозрілі дії, що можуть свідчити про кібератаку.

Таким чином, через скоординовану взаємодію цих елементів, система забезпечення кібербезпеки критичної інформаційної інфраструктури створює багатоплановий та гнучкий механізм реагування на широкий спектр кіберзагроз.

Для ефективного забезпечення кібербезпеки КІІ передбачається комплексний підхід, який включає в себе ряд етапів. Ці етапи, згідно з рис. 1.7, передбачають дії з боку різних органів та установ.

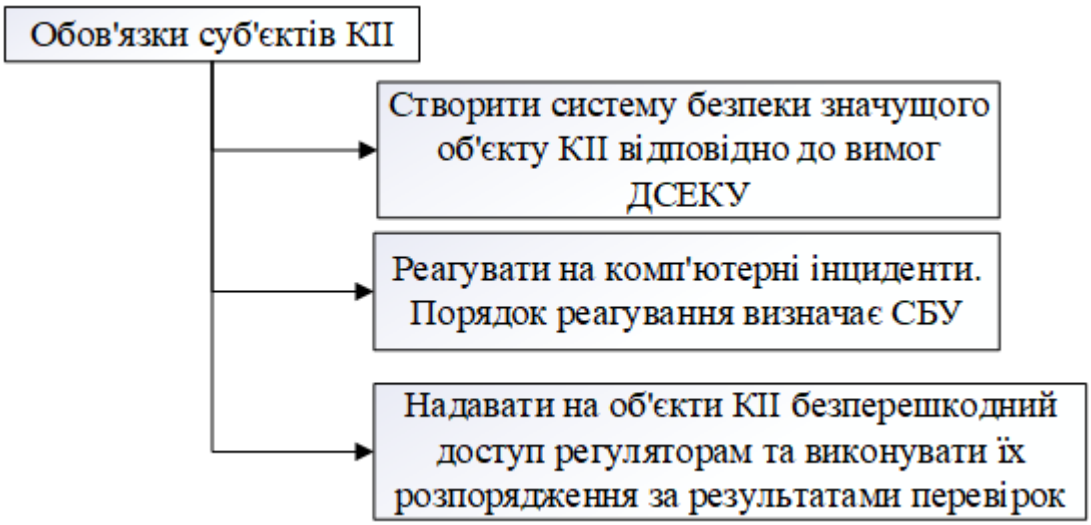

Рис. 1.7. Повноваження суб'єктів КІІ

Суб'єкти КІІ несуть відповідальність за створення системи безпеки значущих об'єктів відповідно до вимог, встановлених ДСЕКУ. Це означає, що суб'єкти КІІ мають розробити та імплементувати заходи захисту, які б відповідали стандартам та нормативам ДСЕКУ.

Важливий аспект — реагування на комп'ютерні інциденти. У цьому контексті, Служба Безпеки України (СБУ) виступає ключовим органом у визначенні порядку реагування. СБУ координує процес ідентифікації, аналізу та нейтралізації кіберзагроз, а також оповіщає інші залучені сторони.

Крім того, законодавство передбачає, що суб'єкти КІІ зобов'язані забезпечити регуляторам безперешкодний доступ до об'єктів для проведення перевірок. Це може бути як планова, так і позапланова інспекція. Результати цих перевірок можуть впливати на подальші розпорядження та рекомендації від регуляторів.

Таким чином, забезпечення кібербезпеки КІІ в Україні вимагає злагодженої взаємодії між різними суб'єктами та органами, що забезпечує комплексний та ефективний механізм реагування на потенційні кіберзагрози.

1.3. Можливі загрози КІІ

Можливі загрози КІІ можна класифікувати за різними категоріями, залежно від їх походження, характеру, та потенційного впливу. Критична інфраструктура, яка включає енергетичні системи, телекомунікації, транспорт, фінансові послуги, та інші важливі сектори, є вразливою до різноманітних форм загроз. Основні категорії загроз, яким піддається критична інфраструктура:

– кібератаки. Це одна з найбільших загроз для КІІ. Кібератаки можуть бути спрямовані на викрадення конфіденційної інформації, знищення даних, порушення роботи систем, або навіть фізичне пошкодження обладнання. Прикладами таких атак є віруси, троянські програми, фішинг, DDoS-атаки, а також більш складні форми, як APT (Advanced Persistent Threats);

– терористичні акти. Терористи можуть націлюватися на КІІ з метою викликати масові порушення, паніку серед населення або нанести шкоду економіці. Це може включати фізичні напади, вибухи, або саботаж об'єктів інфраструктури;

– природні катастрофи. Землетруси, повені, урагани, та інші природні явища можуть завдати серйозної шкоди КІІ, порушуючи її роботу та викликаючи тривалі відновлювальні роботи;

– людський фактор і помилки. Ненавмисні помилки співробітників, неправильне управління системами, або недбале обслуговування також можуть призвести до серйозних збоїв у функціонуванні КІІ;

– технологічні відмови та аварії. Несправності обладнання, відмова програмного забезпечення, або інші технічні проблеми можуть спричинити перебої у роботі критичної інфраструктури;

– інформаційно-психологічні операції. Це можуть бути спроби вплинути на громадську думку або спотворення інформації з метою дестабілізації ситуації в державі;

– економічні та політичні кризи. Стабільність КІІ також може бути під загрозою в умовах економічних коливань, політичних конфліктів, або міждержавних напружень.

Охорона та захист КІІ вимагає комплексного підходу, що включає як технічні заходи (наприклад, посилення кібербезпеки), так і організаційні (наприклад, планування реагування на надзвичайні ситуації, підготовка персоналу). Також важливою є міжнародна співпраця та обмін інформацією про загрози та найкращі практики їхнього запобігання та усунення.

1.4. Аналіз існуючих систем забезпечення захисту інформації на об'єкти критичної інфраструктури

Детальний аналіз існуючих систем забезпечення кібербезпеки на об'єктах критичної інформаційної інфраструктури (КІІ) є невід'ємною частиною підготовчого процесу для розробки цільових та ефективних стратегій і механізмів безпеки. Цей аналіз забезпечує глибоке розуміння сильних та слабких сторін існуючих систем, їх адекватності до сучасних кіберзагроз, а також потреби в інноваційних технологічних рішеннях.

Основа забезпечення кібербезпеки на таких об'єктах полягає в інтеграції декількох ключових компонентів. По-перше, це програмно-технічні засоби виявлення, які є первинним елементом в механізмі захисту. Вони складаються з різних технологічних рішень, включаючи системи моніторингу мережевого трафіку, алгоритми аналізу поведінки користувачів та інші інструменти для виявлення аномалій та несанкціонованого доступу. Ці системи повинні бути високоадаптивними, здатними швидко реагувати на нові види атак та зміни в поведінці користувачів.

По-друге, інтегровані механізми реагування на інциденти, які забезпечують оперативність процессу виявлення та нейтралізації потенційних загроз. Це може включати автоматичні системи для блокування підозрілих дій, а також протоколи зв'язку з компетентними органами.

По-третє, не менш важливим є взаємодія між різними органами та установами, що взаємодіють в цьому процесі, включаючи операторів КІІ, спеціалізовані державні органи, а також комерційні організації, що надають послуги в сфері кібербезпеки. Ця взаємодія повинна бути структурованою і координованою для забезпечення ефективної і швидкої реакції на кіберінциденти.

1.4.1. Системи антивірусного захисту

Сучасні системи антивірусного захисту, розроблені для захисту критичної інфраструктури, відрізняються високою складністю та багаторівневістю архітектури. Вони не обмежуються лише базовими функціями, такими як виявлення та нейтралізація вірусного коду, але також включають розширені аналітичні можливості. Ці можливості охоплюють поведінковий аналіз, який дозволяє ідентифікувати підозрілі поведінкові шаблони програм та процесів, моніторинг мережевого трафіку для виявлення незвичайної активності, а також

кореляцію даних з іншими системами безпеки. Наприклад, інтеграція з інтрузійними детекторами та системами управління інформаційною безпекою дозволяє формувати більш повне розуміння загроз і реагувати на них більш ефективно.

Ключовим аспектом є також інтеграція антивірусних систем з іншими компонентами системи безпеки, такими як механізми шифрування, брандмауери, та системи управління доступом. Це дозволяє створити єдиний координований фронт протидії кіберзагрозам, підвищуючи швидкість і ефективність процесів виявлення та нейтралізації кібератак. Важливою складовою також є постійне оновлення антивірусних баз, яке, враховуючи критичність об'єктів КІІ, має бути максимально автоматизованим і відбуватися в режимі реального часу. Проте, ця автоматизація також може створювати потенційні точки вразливості, особливо якщо процес оновлення стає об'єктом кібератак, що вимагає від розробників та адміністраторів системи антивірусного захисту постійного вдосконалення та оновлення захисних стратегій.

McAfee Total Protection for Data Loss Prevention

Система "McAfee Total Protection for Data Loss Prevention" є високоефективним рішенням для забезпечення кібербезпеки, яке відзначається своєю спрямованістю на захист даних (рис. 1.8). Продукт створений з урахуванням вимог до захисту критично важливої інформації в різних сферах: від корпоративних мереж до об'єктів критичної інфраструктури. Особливу увагу слід звернути на здатність системи здійснювати глибокий моніторинг даних, що передаються та обробляються.

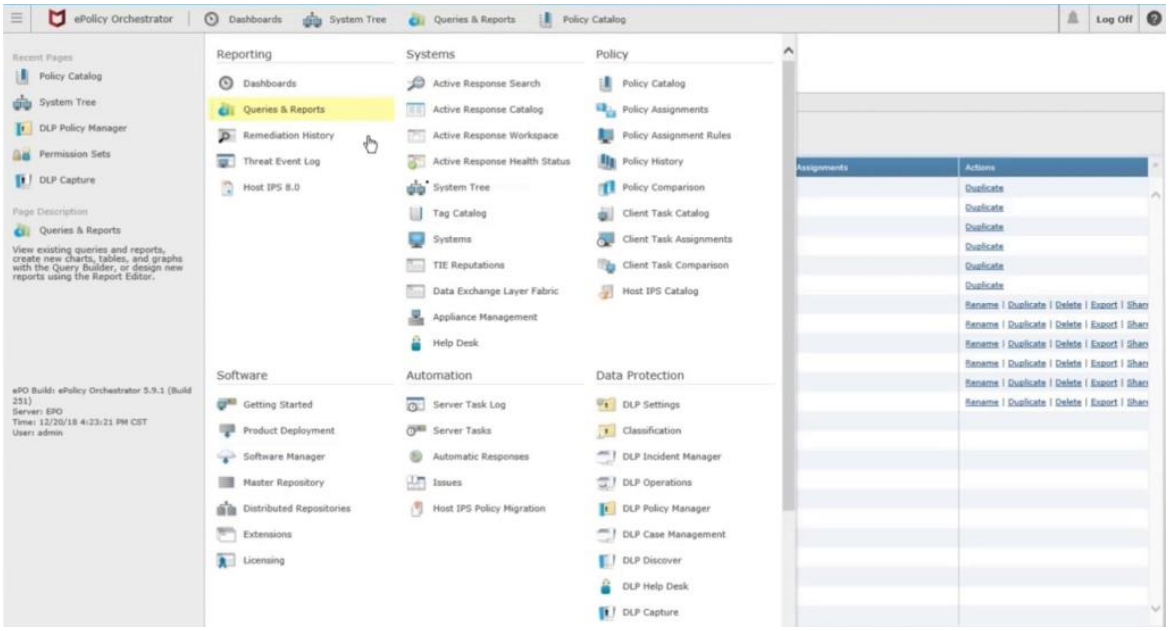

Рис. 1.8. Інтерфейс системи «McAfee Total Protection for Data Loss Prevention»

Перевагами даної системи є:

– комплексний захист. Система пропонує широкий спектр функцій для забезпечення безпеки, включаючи захист від вірусів, шпигунського програмного забезпечення, а також механізми для запобігання витоку даних;

– глибокий моніторинг даних. Можливість здійснювати деталізований аналіз даних для виявлення аномалій та потенційних внутрішніх загроз;

– автоматизація відповіді. Система може автоматично реагувати на виявлені загрози, що забезпечує швидке втручання в критичних ситуаціях;

– адаптивність. Здатність адаптуватися до нових видів загроз та виробляти прогнози на основі історичних даних;

– сумісність з іншими рішеннями. Легко інтегрується з іншими системами безпеки та управління інформаційними ресурсами.

До недоліків можна віднести

– вартість. Високий рівень функціональності може зробити систему досить коштовною, особливо для маленьких та середніх підприємств;

– складність конфігурації. Для ефективного використання системи може знадобитися кваліфікований персонал, здатний правильно налаштувати всі її компоненти;

– потенційні проблеми з приватністю. Глибокий моніторинг даних може створювати питання щодо збереження конфіденційності інформації;

– відмінності в регулятивному оточенні. Якщо система використовується в різних юрисдикціях, може виникнути необхідність адаптації до місцевих законів і нормативів.

Отже, система "McAfee Total Protection for Data Loss Prevention" є високофункціональним рішенням для забезпечення кібербезпеки об'єктів критичної інформаційної інфраструктури. Вона пропонує розширений спектр можливостей для моніторингу, аналізу та реагування на кіберзагрози, включаючи запобігання витоку даних. Однак, необхідно врахувати, що ефективне використання системи потребує значних фінансових вкладень та кваліфікованого персоналу для її налаштування і управління. Загалом, система є потужним інструментом, але її впровадження і експлуатація мають бути відповідно плановані та управлінні [8].

Cisco AMP for Endpoints

Cisco AMP (Advanced Malware Protection) for Endpoints є високотехнологічним рішенням у сфері кібербезпеки, призначеним для захисту периферійних пристроїв в корпоративних мережах (рис. 1.9). Ця система об'єднує в собі різноманітні функції та технології безпеки, забезпечуючи комплексний захист комп'ютерів, смартфонів, планшетів та інших пристроїв, які є частиною корпоративної інфраструктури.

Основною перевагою Cisco AMP for Endpoints є її здатність використовувати передові алгоритми машинного навчання для виявлення та аналізу нових та невідомих загроз. Це дозволяє системі ефективно ідентифікувати складні кіберзагрози, які традиційні антивірусні рішення можуть не виявити. Машинне навчання в Cisco AMP допомагає автоматично аналізувати

мільйони файлів та процесів, виявляючи аномалії та потенційно шкідливу поведінку.

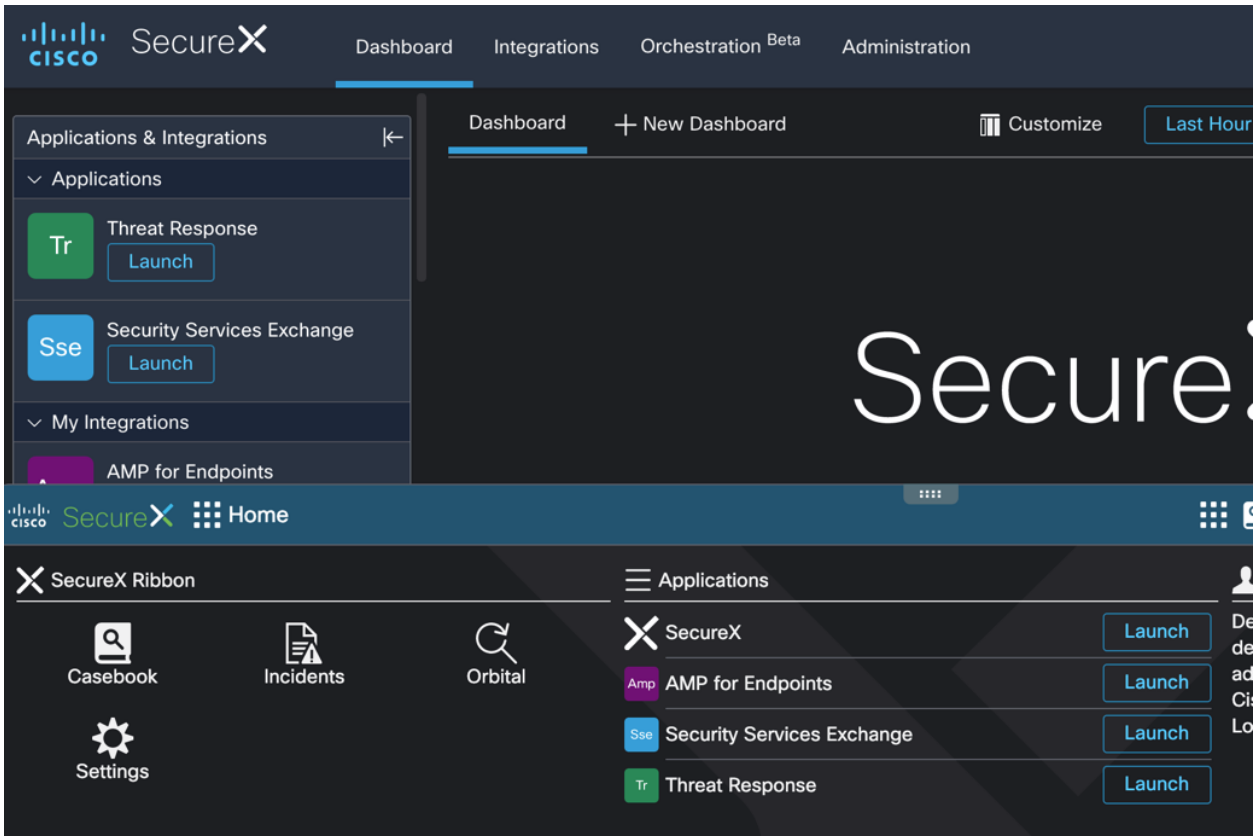

Рис. 1.9. Інтерфейс системи «Cisco AMP for Endpoints»

Переваги даної системи:

– використання алгоритмів машинного навчання для виявлення нових і невідомих загроз, що забезпечує вищий рівень прогнозування та виявлення кіберінцидентів;

– можливість інтеграції з іншими продуктами Cisco, що дозволяє створювати єдиний екосистемний підхід до забезпечення кібербезпеки.

Недоліки системи:

– вимагає від організації наявності висококваліфікованого персоналу для ефективного управління системою;

– висока вартість ліцензування та підтримки, що може бути проблематичним для малого та середнього бізнесу;

– обмеженість в можливостях інтеграції з продуктами інших виробників, що може бути незручним для організацій, які вже використовують інші системи захисту.

Отже, система Cisco AMP for Endpoints являє собою потужний інструмент для захисту ендпоїнтів, особливо великих організацій з високим рівнем кіберзагроз. Її переваги полягають у використанні алгоритмів машинного навчання для прогнозування та виявлення кіберінцидентів, а також у можливості інтеграції з іншими продуктами Cisco для створення єдиної системи кібербезпеки. Однак ця система не позбавлена недоліків, зокрема, висока вартість та потреба в кваліфікованому персоналу для її управління [9].

Symantec Critical System Protection

Система Symantec Critical System Protection є високоспеціалізованою платформою для захисту ендпоїнтів та серверів, включаючи такі особливо важливі системи, як системи управління промисловими процесами (рис. 1.10). Ця система відрізняється глибоким рівнем аналізу даних та поведінки обладнання, що дозволяє забезпечити більш ефективний захист від різних видів кіберзагроз.

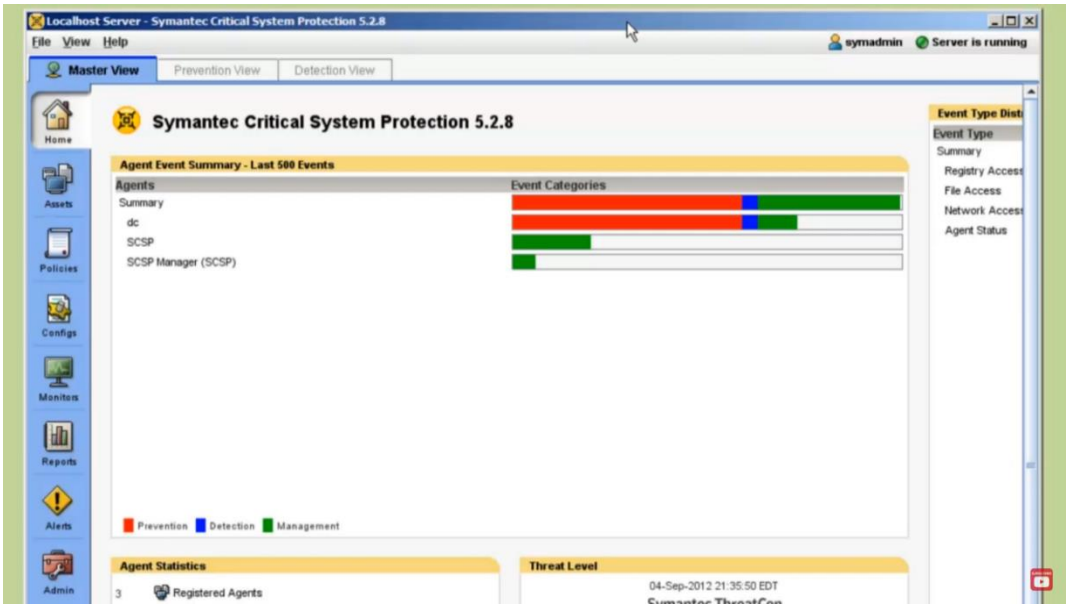

Рис. 1.10. Інтерфейс системи «Symantec Critical System Protection»

Переваги системи:

– глибокий рівень аналізу. Система забезпечує докладний аналіз поведінки та даних на ендпоїнтах та серверах, що дозволяє виявляти та блокувати складні загрози;

– фокус на промислових системах. Одна з небагатьох систем захисту, яка спеціалізується на захисті промислових контрольних систем, що є критично важливими в багатьох секторах;

– виявлення аномалій. Вбудовані механізми для виявлення нестандартних аномалій в поведінці систем, що може вказувати на потенційну кіберзагрозу;

– комплексний підхід. Система надає широкий спектр засобів для захисту, від базового моніторингу до розширених аналітичних інструментів.

Недоліки системи:

– висока складність. Для ефективного використання системи потрібна висока кваліфікація персоналу, що може бути проблематично для деяких організацій;

– вартість. З огляду на спеціалізацію та глибокий рівень аналізу, вартість розгортання та підтримки може бути висока;

– ресурсоємність. Глибокий аналіз та моніторинг можуть вимагати значних обчислювальних ресурсів, що може бути проблемою для старіших або менш потужних систем;

– спеціалізація. Через фокус на промислових системах, система може бути менш ефективною в інших контекстах, таких як звичайні офісні мережі.

Отже, «Symantec Critical System Protection» є високофункціональною системою захисту, спроектованою для забезпечення комплексної безпеки ендпоїнтів та серверів, з особливим акцентом на промислові контрольні системи. Її глибокий рівень аналізу та спеціалізовані засоби для промислових систем роблять її вибором для організацій, що шукають високий рівень захисту для своїх критично важливих активів. Проте, ця система може бути складною в управлінні та ресурсоємною, що вимагає від організацій високої кваліфікації персоналу та

значних фінансових вкладень. Її спеціалізований характер також може обмежувати її застосування в менш специфічних контекстах [10].

З урахуванням вищезгаданих особливостей, системи антивірусного захисту є невід'ємною частиною загальної системи забезпечення кібербезпеки об'єктів КІІ. Вони мають бути не лише технологічно ідеальними, але і гнучкими, інтегрованими та адаптивними до специфіки конкретного об'єкта та загрозового ландшафту.

1.4.2. Системи виявлення вторгнень

Системи виявлення вторгнень (Intrusion Detection Systems, IDS) спрямовані на неперервний моніторинг мережевого трафіку або діяльності на конкретних пристроях з метою виявлення аномалій, що можуть вказувати на небезпечну або незаконну діяльність.

Специфіка застосування IDS в контексті критичної інфраструктури полягає у тому, що тут особливий акцент робиться на надійність та безперебійність роботи систем. Оскільки діяльність об'єктів критичної інфраструктури часто має прямий вплив на життєдіяльність населення та функціонування держави, невідкладність виявлення та нейтралізації кіберзагроз є пріоритетним завданням [12].

Suricata

Suricata є однією з передових систем виявлення та запобігання вторгненню (IDS/IPS) з відкритим вихідним кодом (рис. 1.11). Ця система здатна обробляти великі об'єми мережевого трафіку в реальному часі, виконуючи глибокий аналіз пакетів для виявлення підозрілих або шкідливих дій. Однією з ключових особливостей є підтримка багатопотокової обробки, що значно збільшує продуктивність системи.

Інша важлива характеристика — гнучкість настройки. Suricata дозволяє користувачам створювати деталізовані правила для моніторингу мережевого трафіку, що може бути особливо корисним для захисту об'єктів критичної інформаційної інфраструктури.

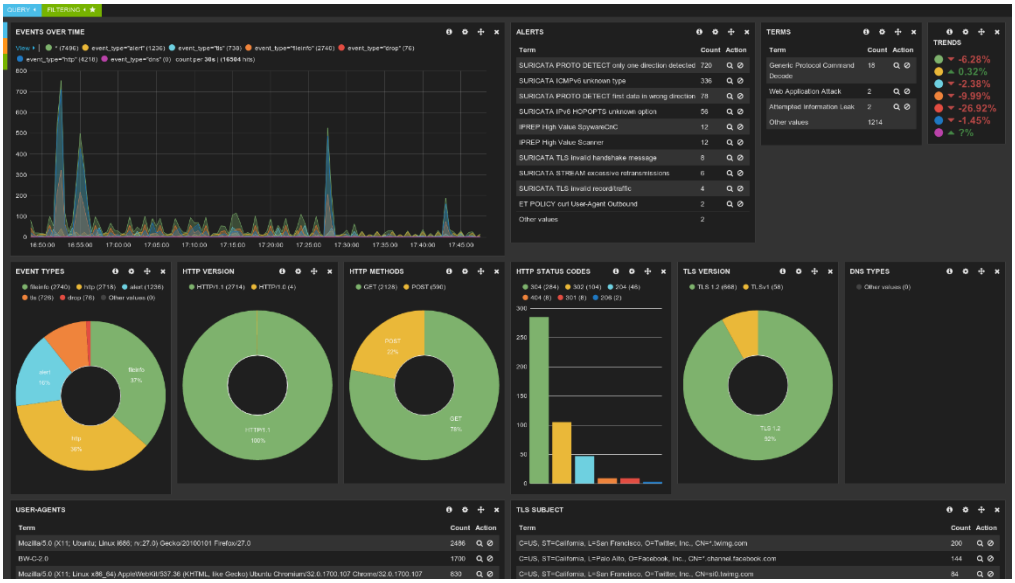

Рис. 1.11. Інтерфейс системи «Suricata»

Система також включає модулі для додаткового аналізу, такі як виявлення відомих вразливостей за допомогою сигнатур або навіть застосування евристичного аналізу для ідентифікації невідомих загроз. Це робить Suricata здатною адаптуватися до еволюції кіберзагроз, що є важливим для об'єктів з високим рівнем критичності [13].

Слід зазначити, що Suricata може бути інтегрована в більш комплексні системи захисту інформації, що дозволяє створити багаторівневу модель безпеки. Така модель може включати різні методики забезпечення безпеки, від базового моніторингу мережі до аналізу поведінки користувачів та систем.

Переваги системи:

– висока продуктивність. Завдяки багатопотоковій архітектурі Suricata може ефективно обробляти великі об'єми даних в реальному часі;

– гнучкість настройки. Система дозволяє користувачам створювати докладні правила для моніторингу та відстеження підозрілих або шкідливих дій;

– модульність та масштабованість. Suricata може легко інтегруватися в існуючі системи безпеки, що дозволяє створити багаторівневу архітектуру захисту;

– підтримка новітніх методів аналізу. Включає в себе алгоритми машинного навчання для еврістичного аналізу, що дозволяє виявляти навіть невідомі загрози.

Недоліки системи:

– складність настройки. Необхідність деталізованого конфігурування може бути складною для недосвідчених користувачів;

– вимоги до ресурсів. Для ефективної роботи Suricata потребує відносно великих обчислювальних ресурсів, що може бути проблематичним для старіших або менш потужних систем;

– потенційні помилки спрацювання. Як і усі системи IDS/IPS, Suricata може генерувати помилкові спрацювання, що потребують додаткового аналізу;

– відкритий код. Хоча це і є перевагою з точки зору гнучкості та адаптивності, відкритий код також може слугувати потенційним вектором для цільових атак на саму систему Suricata.

Отже, Suricata представляє собою високоефективну систему виявлення вторгнень та блокування (IDS/IPS), яка відзначається гнучкістю, модульністю та можливістю масштабування. Однак для ефективної роботи системи потрібно враховувати високі вимоги до обчислювальних ресурсів та потенційну складність настройки. Крім того, як і в будь-якій системі виявлення вторгнень, існує ризик помилкових спрацювань, що вимагає додаткового аналізу[14].

Cisco Firepower

Cisco Firepower є комерційним рішенням в сфері кібербезпеки, що розроблено для інтегрованого захисту периметру мережі, ендпоїнтів та внутрішньої інфраструктури (рис. 1.12). Однією з ключових переваг цієї системи є можливість її безшовної інтеграції з іншими продуктами компанії Cisco [15], що забезпечує, централізований механізм управління кібербезпекою. Система

має вбудовані розширені аналітичні засоби, які дозволяють проводити глибокий аналіз мережевого трафіку та поведінки користувачів.

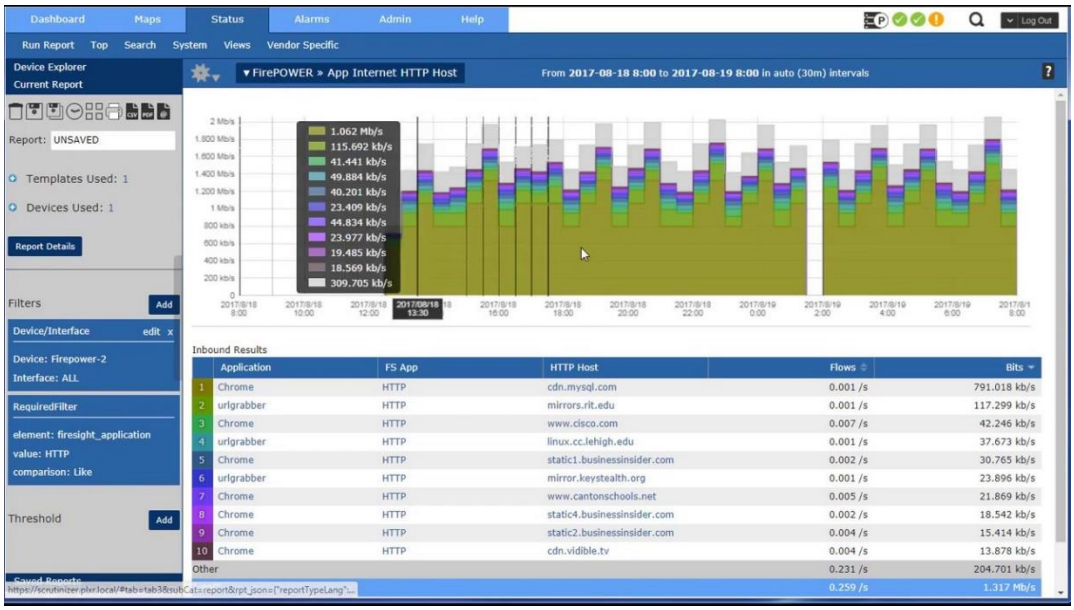

Рис. 1.12. Інтерфейс системи «Cisco Firepower»

Переваги системи:

– безшовна інтеграція з іншими продуктами Cisco, що дозволяє створювати єдиний централізований механізм управління кібербезпекою;

– розширені аналітичні можливості для глибокого аналізу мережевого трафіку і поведінки користувачів;

– централізоване управління політиками безпеки через один інтерфейс, що спрощує управління;

– високий рівень масштабування та гнучкість налаштувань, що дозволяє адаптувати систему до потреб конкретної організації.

Недоліки:

– висока вартість продукту та ліцензування може стати преградою для малого та середнього бізнесу;

– потреба в кваліфікованих фахівцях для налаштування та управління системою;

– можливі складнощі при інтеграції з продуктами інших виробників, що може обмежити функціональність;

– потенційна складність в адаптації системи до специфічних потреб і вимог організації.

Отже, система Cisco Firepower представляє собою потужний інструмент для захисту інформаційних систем, що володіє високим рівнем інтеграції з іншими продуктами Cisco та розширеними аналітичними можливостями. Централізоване управління та гнучкість налаштувань роблять її привабливою для великих організацій з складною мережевою інфраструктурою. Проте, висока вартість та потреба в кваліфікованих ресурсах для ефективного управління можуть стати значущими обмеженнями, особливо для малого та середнього бізнесу. Слід також враховувати можливі складнощі при інтеграції з не-Cisco продуктами. Загалом, вибір Cisco Firepower варто робити на основі конкретних організаційних потреб та ресурсних можливостей [16].

IBM QRadar

Система IBM QRadar вважається одним з найбільш передових рішень у галузі кібербезпеки, оскільки вона пропонує комплексний та багатофункціональний підхід до захисту інформації (рис. 1.13). Відмінною рисою QRadar є її здатність виконувати не лише базові функції виявлення вторгнень, як це роблять багато систем IDS/IPS, але й надавати розширені можливості для глибокого аналізу безпеки на різних рівнях [17].

Цей інструмент інтегрує в себе різноманітні функції безпеки, зокрема, авансований аналіз мережевого трафіку, управління журналами подій, виявлення зловмисних дій, а також оцінку вразливостей. QRadar ефективно об'єднує ці функції, використовуючи потужні алгоритми та інтелектуальний аналіз, що дозволяє виявляти складні загрози, які можуть залишитися непоміченими іншими системами.

Однією з ключових особливостей QRadar є його здатність до кореляції подій з різних джерел, що дозволяє створити повне уявлення про загрози та вразливості в мережі. Система здатна аналізувати великі обсяги даних в режимі

реального часу, надаючи адміністраторам безпеки інструменти для швидкого реагування на потенційні інциденти.

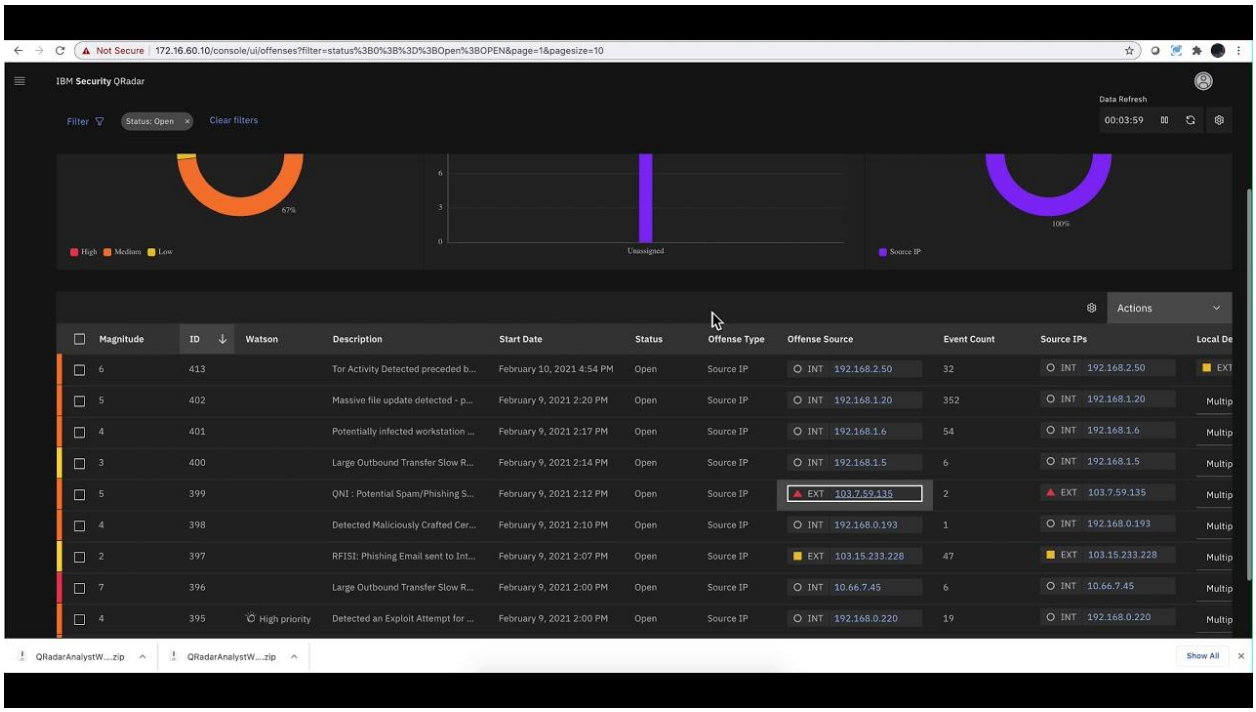

Рис. 1.13. Інтерфейс системи «IBM QRadar»

Ключова особливість QRadar полягає в її здатності агрегувати та аналізувати дані з різноманітних джерел. Це означає, що система може збирати журнали подій з операційних систем, баз даних, мережевого обладнання та інших компонентів інфраструктури. Така інтеграція дозволяє аналізувати збіги подій або аномалії, які можуть вказувати на потенційну загрозу [18].

Крім того, QRadar може аналізувати реальний мережевий трафік, що надає ще один рівень контролю та можливість для раннього виявлення загроз. Таким чином, система не просто реагує на уже відбуті події, але й дозволяє здійснювати прогнозування потенційних інцидентів на основі аналізу широкого спектру індикаторів.

Переваги:

– комплексний аналіз безпеки. QRadar не просто виявляє вторгнення, але й аналізує цілісну картину безпеки, інтегруючи дані з різних джерел;

– гнучкість інтеграції. Система може інтегруватися з різноманітними джерелами даних, що робить її універсальним рішенням для організацій з різними технологічними стеками;

– високий рівень автоматизації. Включає автоматичний збір даних, їх аналіз та реагування на інциденти, що підвищує швидкість та ефективність відгуку;

– підтримка машинного навчання. Вбудовані алгоритми машинного навчання дозволяють системі неперервно вдосконалювати свої алгоритми виявлення загроз.

Недоліки:

– висока вартість. Ліцензування та підтримка можуть бути досить дорогими, особливо для малого та середнього бізнесу;

– складність налаштування. Для ефективного використання системи потрібні кваліфіковані спеціалісти з глибоким розумінням кібербезпеки;

– великий обсяг ресурсів для обробки. Система вимагає значних обчислювальних ресурсів для аналізу великих обсягів даних;

– можливі проблеми з інтеграцією в уже існуючі системи. Хоча QRadar і є гнучким у плані інтеграції, конфлікти та несумісність можуть виникнути з деякими вже встановленими системами безпеки.

Отже, система IBM QRadar відрізняється багатофункціональністю та глибоким аналізом безпеки, що робить її видаючимся рішенням в області кібербезпеки, особливо для об'єктів критичної інфраструктури. Її здатність інтегрувати дані з різних джерел і проводити комплексний аналіз індикаторів загрози надає можливість для раннього виявлення, моніторингу та прогнозування потенційних кіберінцидентів. Така комплексність і глибина аналізу роблять QRadar цінним інструментом для забезпечення високого рівня безпеки [19].

Palo Alto Networks Threat Prevention

Система Palo Alto Networks Threat Prevention представляє собою передове рішення в сфері кібербезпеки, яке спеціалізується на виявленні нових та

невідомих кіберзагроз (рис. 1.14). Однією з ключових характеристик даної системи є використання хмарних аналітичних засобів, що дозволяє забезпечити глибокий і швидкий аналіз мережевого трафіку та інших індикаторів. Це, у свою чергу, надає можливість для оперативного виявлення нових типів загроз та їх ефективної нейтралізації.

Даний продукт інтегрується з іншими компонентами інфраструктури безпеки, включаючи файрволи, системи управління подіями в області безпеки (SIEM) та інші засоби моніторингу. Це дозволяє створити єдину, гармонізовану систему захисту, яка може працювати в реальному часі для моніторингу, виявлення та реагування на потенційні кіберзагрози [20].

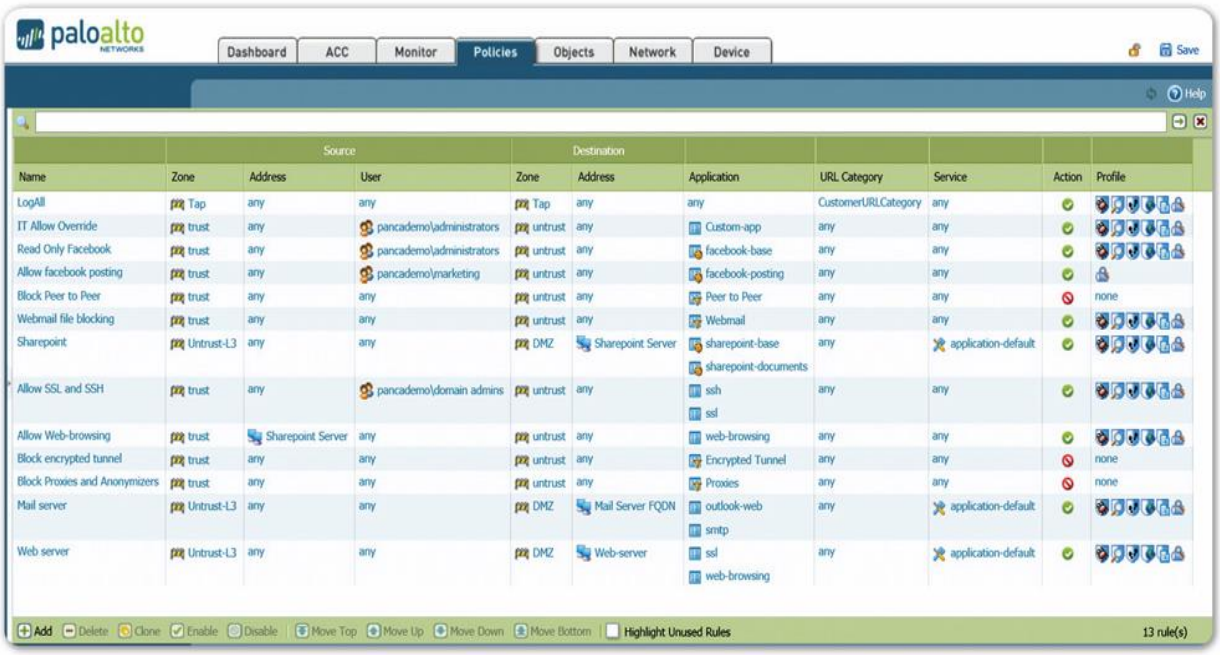

Рис. 1.14. Інтерфейс системи «Palo Alto Networks Threat Prevention»

Переваги:

– адаптивність та машинне навчання. Система здатна адаптуватися до нових типів загроз і "вчитися" на основі аналізу попередніх інцидентів та актуальних даних;

– інтеграція з іншими системами. Високий рівень сумісності з іншими компонентами інфраструктури безпеки, такими як файрволи та системи управління подіями в області безпеки (SIEM);

– хмарна аналітика. Використання хмарних ресурсів для аналітики дозволяє забезпечити високу швидкість та ефективність аналізу даних;

– комплексний аналіз безпеки. Система не просто виявляє вторгнення, але і проводить глибокий аналіз стану безпеки на різних рівнях організації.

Недоліки:

– вартість. Це комерційний продукт, і його впровадження та обслуговування може бути досить коштовним;

– складність налаштування. Для ефективної роботи системи потребує кваліфікованих фахівців і складного процесу налаштування;

– потенційні проблеми з приватністю. Збір та аналіз великих обсягів даних може порушувати питання конфіденційності та приватності;

– залежність від хмарних сервісів. Хоча хмарна аналітика є перевагою, вона також може стати недоліком у випадку проблем з доступом до хмарних ресурсів або їх надійністю.

Отже, система Palo Alto Networks Threat Prevention являє собою високотехнологічне рішення для виявлення та протидії кіберзагрозам. З одного боку, її адаптивність, можливість інтеграції з іншими системами, використання хмарної аналітики та комплексний аналіз безпеки роблять її ефективним інструментом для захисту критичної інформаційної інфраструктури. З іншого боку, висока вартість, складність налаштувань, потенційні ризики для приватності та залежність від хмарних сервісів можуть ускладнити її впровадження та подальше використання. У підсумку, це рішення вимагає ретельного аналізу з урахуванням специфіки організації та доступних ресурсів для його ефективного використання.

З урахуванням вищезгаданих аспектів, системи виявлення вторгнень є невід'ємною частиною стратегії забезпечення кібербезпеки об'єктів критичної

інформаційної інфраструктури, відіграючи ключову роль у запобіганні, виявленні та нейтралізації кіберзагроз [21].

1.5. Вибір сфери критичної інфраструктури та аналіз її інформаційних процесів

Критичні інфраструктури формують основу стабільного та безпечного функціонування сучасного суспільства та економіки [21]. Вони включають різні сектори, як-от енергетика, здоров'я, транспорт, телекомунікації та, звісно, фінансові послуги. Кожен з цих секторів відіграє вирішальну роль у підтримці повсякденного життя та економічної діяльності, а їхнє безперебійне функціонування є критично важливим для національної безпеки та процвітання.

Серед цих секторів фінансова сфера є особливо значущою, оскільки вона становить фундамент для всієї економічної системи. Фінансові установи, які включають банки, страхові компанії, біржі, інвестиційні фонди та інші організації, забезпечують рух капіталу, кредитування, інвестиції та багато інших критично важливих економічних функцій.

Фінансові установи не тільки управляють значними фінансовими ресурсами та зберігають конфіденційні дані, але й постійно стикаються з широким спектром кіберзагроз. Це може включати спроби фінансового шахрайства, крадіжку даних, атаки на інфраструктуру для проведення платежів, DDoS-атаки та інші кіберзлочини. Тому розробка нових методів захисту, які здатні виявляти загрози, є ключовим для забезпечення безпеки фінансових установ, а отже, і всієї фінансової системи країни.

Для глибокого розуміння та аналізу інформаційних процесів у фінансовій сфері КІ, ефективним інструментом є побудова діаграми цих процесів. Діаграма дозволяє візуалізувати різні аспекти інформаційних потоків, включаючи збір, обробку, зберігання та захист даних, а також взаємодії між різними підсистемами і відділами фінансових установ. Така візуалізація сприяє кращому розумінню структури та вразливостей системи, що є ключовим для розробки ефективних

стратегій кіберзахисту та управління ризиками. Наочний приклад такої діаграми інформаційних процесів критичної інфраструктури фінансової сфери представлений на рис 1.15 нижче.

Рис. 1.15. Інформаційні потоки КІ фінансової сфери

Критична інфраструктура фінансової сфери включає широкий спектр інформаційних процесів, які є важливими для її стабільного функціонування та безпеки. Вона обробляє різноманітні типи даних, включаючи особисту інформацію клієнтів, історію транзакцій, фінансові дані, ринкові тенденції та регуляторну інформацію [22]. Захист цих даних та їх конфіденційність є ключовим пріоритетом для фінансових установ.

Передача інформації в цій сфері часто здійснюється через захищені канали, щоб забезпечити безпечний обмін даними між різними фінансовими установами та їхніми клієнтами. Це особливо важливо для запобігання несанкціонованому доступу до фінансової інформації та забезпечення цілісності транзакцій.

Зберігання цієї інформації вимагає високого рівня безпеки, включаючи використання захищених баз даних та систем резервного копіювання для запобігання втраті важливих даних. Обробка інформації включає в себе застосування різноманітних автоматизованих інструментів та аналітичних рішень, які дозволяють обробляти великі обсяги даних, виконувати фінансові операції та аналізувати ринкові тенденції.

Захист інформації забезпечується через заходи кібербезпеки, які включають фаєрволи, антивірусне програмне забезпечення, інструменти виявлення та реагування на інциденти. Регулярні аудити безпеки та тренінги для співробітників з питань безпеки інформації також відіграють важливу роль.

Відповідність регулятивним вимогам, таким як GDPR або PCI DSS, є ще одним ключовим елементом, що забезпечує, що фінансові установи дотримуються міжнародних та національних стандартів захисту даних [23]. Це допомагає уникнути правових проблем та забезпечує високий рівень довіри клієнтів.

Таким чином, інформаційні процеси в фінансовій сфері вимагають комплексного підходу до захисту даних, обробки та передачі інформації, забезпечуючи таким чином надійне функціонування цієї критично важливої інфраструктури.

1.6. Аналіз основних методів захисту вибраної сфери

Аналіз основних методів захисту в фінансовій сфері виявляє широкий спектр стратегій та технологій, спрямованих на забезпечення безпеки інформаційних систем та даних. Ці методи включають фізичний захист інфраструктури, шифрування даних, захист периметра мережі за допомогою фаєрволів та систем виявлення вторгнень, а також впровадження політик безпеки, тренінгів для співробітників та регулярних аудитів.

Однак, одним з найважливіших аспектів захисту фінансової сфери є виявлення та раннє запобігання вторгненням у комп'ютерні мережі. Це критично важливо, оскільки вчасне виявлення аномалій може запобігти значним фінансовим втратам, витокам конфіденційної інформації та іншим серйозним наслідкам кібератак. В фінансовій сфері, де транзакції відбуваються неперервно та великі обсяги даних передаються щодня, виявлення незвичайної активності в

мережі є надзвичайно важливим для забезпечення надійності та безпеки операцій.

Для виявлення аномалій у мережах фінансових організацій використовуються різноманітні методи. Кожен із них має свої особливості та призначений для виявлення певних типів аномалій. Перелік найбільш популярних методів для виявлення аномалій в мережі складається із:

– статистичний аналіз. Використовує статистичні моделі для аналізу мережевого трафіку та виявлення відхилень від звичних патернів [24]. Наприклад, метод, який порівнює кількість прийнятих/переданих пакетів за одиницю часу з типовими показниками, виявляє відхилення, які можуть вказувати на аномальну активність;

– машинне навчання та штучний інтелект. Використання алгоритмів машинного навчання для ідентифікації аномальних патернів, які можуть бути неочевидними для традиційних методів [25]. Ці системи можуть навчатися з часом, вдосконалюючи свою здатність виявляти складні аномалії;

– поведінковий аналіз. Зосереджується на виявленні аномалій у поведінці користувачів або систем. Наприклад, неочікуване збільшення обсягу запитів на транзакції або незвичайні часові шаблони доступу можуть вказувати на потенційні загрози;

– аналіз підписів. Виявляє відомі підписи шкідливих програм або атак. Хоча цей метод ефективний проти відомих загроз, він може не виявити нові або модифіковані атаки;

– аналіз аномалій в системних журналах. Перевірка журналів систем на наявність підозрілих записів, що можуть вказувати на зловмисні дії або технічні неполадки;

– мережевий аналіз. Включає в себе моніторинг і аналіз мережевого трафіку для виявлення незвичних патернів, таких як незвичайно високий обсяг трафіку або транзакцій, що відбуваються з підозрілих джерел.

Проведений аналіз різноманітних методів виявлення аномалій у мережах фінансових організацій підкреслює значущість і доцільність використання

статистичного аналізу для розробки програмного забезпечення. Перелік аргументів, які обґрунтовують цей вибір складаються із:

– простота імплементації та використання. Статистичний аналіз базується на порівнянні поточної активності з історичними даними, що дозволяє легко імплементувати цей метод у програмне забезпечення. Він не вимагає складних алгоритмів машинного навчання або розробки складних поведінкових моделей;

– ефективність у виявленні відхилень. Статистичний аналіз ефективно виявляє відхилення від норми, які можуть вказувати на аномальну активність або потенційні загрози. Наприклад, значне збільшення або зменшення кількості переданих/прийнятих пакетів за одиницю часу може свідчити про спроби DDoS-атак, витоку даних або інших форм кіберзлочинності;

– швидкість реагування. Програмне забезпечення, що базується на статистичному аналізі, здатне швидко виявляти аномалії та сповіщати адміністраторів системи. Це дозволяє оперативно реагувати на потенційні загрози, мінімізуючи можливі збитки;

– низькі вимоги до ресурсів. На відміну від деяких більш складних систем, які використовують машинне навчання або поведінковий аналіз, статистичний аналіз зазвичай не вимагає великої кількості обчислювальних ресурсів, що робить його більш доступним для широкого спектру організацій;

– легкість адаптації та оновлення. Статистичні моделі можуть бути легко адаптовані до змін у мережевій активності та оновлені для відповідності поточним умовам, що забезпечує гнучкість і актуальність системи.

Враховуючи ці переваги, статистичний аналіз є дуже доцільним вибором для розробки програмного забезпечення, спрямованого на виявлення аномальних поведінок у мережах фінансових організацій.

1.7. Основні принципи забезпечення комплексної безпеки критичної інформаційної інфраструктури

Основні засади, що забезпечують комплексну безпеку важливих інформаційних систем, являють собою ключові настанови та методи, спрямовані на створення стійкого, гнучкого і безпечного середовища для охорони необхідних інформаційних ресурсів. Ці основоположні засади включають в себе різноманітні аспекти захисту, починаючи від технічних до управлінських, і служать фундаментом для планування, впровадження та керування системами безпеки. Також вони використовуються як стратегічна база для виявлення, аналізу та зменшення ризиків, що виникають через кібернебезпеки та інші потенційні загрози.

Основні принципи цього процесу можна розглянути через наступні аспекти:

– розподілена відповідальність. Відповідальність за безпеку КІІ має бути ясно розподілена між усіма зацікавленими сторонами, включаючи державні органи, операторів КІІ та постачальників технологічних рішень;

– багаторівневий захист. Захист КІІ має бути організований на декількох рівнях – від фізичного захисту до захисту інформації та мережевого рівня. Це допомагає створити багаторівневу систему, яка може протистояти різним видам загроз;

– принцип мінімальних прав. Доступ до ресурсів та інформації має бути обмежений наскільки можливо, а доступ надаватися лише тим особам, яким це дійсно необхідно для виконання їхніх функціональних обов'язків;

– проактивний моніторинг та реагування. Системи моніторингу та реагування на інциденти повинні бути не лише реактивними, але й проактивними, здатними ідентифікувати та блокувати нові загрози;

– контроль і аудит. Регулярний контроль та аудит безпеки систем є ключовими для підтримки високого рівня захисту. Це включає в себе як внутрішній аудит, так і зовнішню незалежну експертизу;

– неперервна освіта та підготовка персоналу. Освіта та підготовка персоналу в області кібербезпеки є важливою складовою ефективного захисту КІІ;

– комплексний підхід до ризиків. Розробка та впровадження політики безпеки повинні базуватися на всебічному аналізі ризиків, що включає технічні, організаційні та людські фактори;

– законодавча підтримка. Ефективна система захисту КІІ вимагає належної законодавчої підтримки, що визначає права та обов'язки всіх учасників процесу;

– міжнародна співпраця. З урахуванням глобалізації та міжнародного характеру кіберзагроз, співпраця на міжнародному рівні є ключовою для ефективного захисту КІІ.

Ці принципи формують основу для створення робустної, ефективної та адаптивної системи забезпечення безпеки критичної інформаційної інфраструктури.

1.8. Висновок

У даному розділі було розглянуто комплексне коло питань, що стосуються забезпечення безпеки КІІ. Аналіз правових основ, як на національному, так і на міжнародному рівнях, виявив структурований підхід до регулювання та захисту КІІ, що є ключовим для розуміння загального контексту кібербезпеки.

Були детально розглянуті різні аспекти КІІ як об'єкту забезпечення безпеки, включно з повноваженнями Державної служби екстреної кібернетичної безпеки України та структурою системи виявлення атак. Також були

проаналізовані можливі загрози КІІ, що охоплюють широкий спектр від кібератак до природних катастроф та економічних криз, підкреслюючи складність та багатогранність викликів, з якими стикаються об'єкти КІІ.

Аналіз існуючих систем забезпечення захисту інформації виявив різноманітність доступних інструментів і технологій, включаючи системи антивірусного захисту та системи виявлення вторгнень, що є ключовими компонентами захисту КІІ.

Рішення про вибір фінансової сфери як об'єкту дослідження було аргументовано її критичною важливістю та високою вразливістю до кіберзагроз, а також був проведений аналіз основних інформаційних процесів у цій сфері.

Особлива увага була приділена аналізу основних методів захисту вибраної сфери, у результаті якого було прийняте рішення щодо використання методу статистичного аналізу для виявлення вторгнень, який демонструє високу ефективність і доцільність у контексті фінансової сфери.

Загалом, розділ надає вичерпне розуміння та обґрунтування необхідності комплексного підходу до забезпечення безпеки КІІ, висвітлюючи як теоретичні, так і практичні аспекти в цій ключовій області.

Розділ 2. АНАЛІЗ МЕТОДІВ ТА ПОСТАНОВКА ЗАДАЧІ

2.1. Обґрунтування вибору напрямку дослідження

Вибір напрямку дослідження даної теми ґрунтується на конкретних потребах державних та приватних структур України в систематизації підходів до захисту важливих інформаційних активів. Недоліками існуючих систем є їх фрагментарність, високі витрати на інтеграцію та підтримку, а також відсутність єдиних стандартів. Дослідження спрямоване на аналіз можливостей для оптимізації ресурсів та ефективного реагування на кіберзагрози, зокрема через впровадження автоматизованих систем моніторингу та управління.

Це у свою чергу вимагає не тільки глибокого розуміння алгоритмів та технологій, що лежать в основі систем виявлення вторгнень, але й знання з особливостей архітектури комп'ютерних мереж, протоколів передачі даних, а також методів криптографічного захисту.

Така ціль також зобов'язує до вивчення потенційних векторів атак і механізмів, які можуть бути використані зловмисниками для вторгнення в мережу. Це, в свою чергу, веде до необхідності аналізу сучасних методів і засобів, які вже застосовуються в галузі кібербезпеки для виявлення та протидії таким загрозам.

Також, розробка методу для виявлення вторгнень вимагає інтеграції з існуючими системами захисту та моніторингу, що забезпечує не тільки технічну, але й організаційну складову дослідження. Це може включати в себе розробку процедур реагування на інциденти, адаптацію до вимог відповідних нормативно-правових актів, а також забезпечення сумісності з іншими системами захисту.

Отже, вибір даного напрямку дослідження виправданий не тільки актуальністю проблеми, але і комплексністю вимог, які пред'являються до сучасних систем кібербезпеки.

2.2. Розробка загальної методики проведення досліджень

У рамках даної роботи одним із ключових етапів є розробка загальної методики проведення досліджень. Ця методика служить структурним основам для систематичного аналізу, оцінки та валідації розробленого методу виявлення вторгнень в комп'ютерну мережу КІІ. Етапи дослідження будуть включати:

Підготовчий етап

На цьому етапі здійснюється первинний аналіз вимог до безпеки, вивчаються потенційні ризики та вразливості, а також формулюються цілі та завдання дослідження.

На підготовчому етапі проводиться ряд ключових дій, які становлять основу для всього подальшого дослідження. Важливою частиною цього етапу є первинний аналіз вимог до безпеки. Для цього вивчається існуюча нормативно-правова база, технічні специфікації об'єктів критичної інфраструктури, а також рекомендації з кібербезпеки від відомих організацій та стандартів (наприклад, ISO/IEC 27001, NIST).

Паралельно з цим проводиться оцінка потенційних ризиків та вразливостей. Це повинно у свою чергу включати аналіз історії інцидентів безпеки на подібних об'єктах, вивчення звітів про вразливості, а також проведення попередніх тестів на проникнення для виявлення слабких місць в системі.

Особлива увага приділяється формулюванню цілей та завдань дослідження. На основі зібраної інформації і виходячи з актуальних потреб та викликів, формулюються конкретні цілі, які має досягнути дослідження. Це буде розробка методу виявлення вторгнень, який максимально ефективно виявляє аномальну поведінку мережі з мінімальними витратами на ресурси системи.

Завершується підготовчий етап формуванням плану дослідження, в якому детально описуються етапи, методи, інструменти та ресурси, які будуть використовуватися в процесі наукової роботи.

Вибір та аналіз методів

Етап вибору та аналізу методів визначає напрямки для подальшого практичного застосування. Після проведення підготовчого етапу, де було зібрано необхідну інформацію та сформульовані цілі, наступний крок полягає в виборі методів, які будуть піддані подальшому аналізу та тестуванню.

Для цього перш за все формується список потенційно ефективних методів виявлення вторгнень. Список може формуватися на основі аналізу наукової літератури, патентів, технічних звітів, а також на основі рекомендацій від експертів в галузі кібербезпеки.

Кожен метод потім піддається детальному аналізу з урахуванням ряду параметрів, таких як ефективність виявлення різних типів атак, швидкість реагування, потреби в обчислювальних ресурсах, а також можливість інтеграції з існуючими системами.

На основі зібраних даних формується рейтинг методів, який відображає їх потенційну ефективність та придатність для реалізації в контексті конкретного дослідження. Після цього вибираються методи, які будуть взяті для подальшого експериментального тестування та аналізу.

Розробка тестового середовища

Етап розробки тестового середовища передбачає створення віртуального або фізичного середовища, яке максимально точно імітує реальні умови роботи об'єктів критичної інфраструктури.

Першочергово визначаються ключові параметри та характеристики мережі, які будуть емульовані. Це повинно включати конфігурацію мережевого обладнання, параметри трафіку, типи даних, які будуть передаватися, а також потенційні точки входу для зовнішніх атак.

Наступним кроком є вибір платформи для реалізації тестового середовища. Це може бути спеціалізоване програмне забезпечення для емуляції мереж, або ж реальне обладнання, налаштоване відповідно до вимог тестування. Важливо забезпечити можливість моніторингу та логування всіх подій в системі для подальшого аналізу.

Після налаштування основних параметрів проводиться калібрування тестового середовища. Це важливий етап, на якому перевіряється правильність налаштувань і виправляються можливі неточності. Також проводиться ряд пробних тестів для перевірки стабільності та надійності середовища.

З завершенням цього етапу дослідження можна переходити до безпосереднього тестування методів виявлення вторгнень в умовах, що максимально наближені до реальних. Таким чином, розробка тестового середовища є ключовим етапом, який забезпечує високу якість та достовірність результатів подальших досліджень.

Проведення експериментів

Етап проведення експериментів є кульмінацією всього наукового дослідження і вимагає докладної підготовки та виконання. Після створення та калібрування тестового середовища, наступним кроком є планування самого експерименту. Для цього встановлюються конкретні метрики ефективності, які будуть вимірюватися, та сценарії тестування.

Сценарії тестування включають різні типи атак, моделі поведінки користувачів, різні рівні навантаження на систему та інші фактори, які можуть вплинути на ефективність методів виявлення вторгнень. Сценарії розробляються таким чином, щоб максимально точно відтворити реальні умови експлуатації.

Після встановлення параметрів та сценаріїв розпочинається безпосереднє проведення експериментів. Важливо забезпечити постійний моніторинг і логування даних, що дозволить подальше їх детальне аналізування. Кожен експеримент проводиться в кілька ітерацій для забезпечення повторюваності результатів.

Після завершення експериментальної частини слідує етап обробки та аналізу отриманих даних. Використовуючи вибрані метрики, проводиться оцінка ефективності, швидкодії, надійності та інших критичних параметрів вибраних методів виявлення вторгнень.

Таким чином, проведення експериментів не лише дозволяє оцінити вибрані методи на практиці, але і формує базу для подальших наукових досліджень, вносячи вклад у розвиток цієї важливої галузі кібербезпеки.

Аналіз результатів

Етап аналізу результатів є одним з найбільш критичних в науковому дослідженні, оскільки саме на основі цього аналізу будуть зроблені висновки та рекомендації. Після завершення експериментальної частини роботи, всі зібрані дані піддаються первинній обробці та нормалізації. Це необхідно для видалення "шуму" та виокремлення корисної інформації.

Далі проводиться статистичний аналіз отриманих даних. Застосовуються різноманітні методи статистичного аналізу, такі як дисперсійний аналіз, t-тестування, кореляційний аналіз тощо, для визначення ступеню надійності результатів та їх загальної тенденції.

Однією з ключових частин аналізу є порівняльна оцінка ефективності методів виявлення вторгнень на основі визначених критеріїв і метрик. Це може бути, наприклад, швидкість реагування на загрозу, кількість помилково позитивних сигналів, або здатність методу виявляти нові, раніше невідомі види атак.

На підставі проведеного аналізу формулюються висновки про ефективність кожного з тестованих методів. Також враховуються можливі обмеження кожного методу, його придатність для конкретних умов експлуатації та рекомендації щодо подальшого вдосконалення.

В кінцевому підсумку, аналіз результатів не лише визначає успішність проведеного дослідження, але й підказує напрямки для подальших наукових розробок в області кібербезпеки критичної інфраструктури.

Таким чином, розроблена методика є комплексним інструментом для систематичного підходу до дослідження проблеми захисту КІІ.

2.3. Аналіз методів виявлення вторгнень

Даний розділ розглядає чотири основних типи методів виявлення вторгнень: поведінкові методи, методи на основі знань, методи машинного навчання та методи обчислювального інтелекту. Кожен із цих методів аналізується з позицій ефективності, швидкодії, адаптивності та здатності до ідентифікації нових, раніше невідомих типів загроз.

Важливо підкреслити, що методи виявлення вторгнень не існують ізольовано, а найчастіше є частиною комплексної системи кібербезпеки, яка може включати в себе також криптографічні методи захисту, фізичні заходи безпеки, системи управління доступом та інші.

2.3.1. Поведінкові методи

Поведінкові методи виявлення вторгнень базуються на аналізі нормальної (очікуваної) поведінки системи або користувача і виявленні відхилень від цієї норми. В основі цих методів лежить принцип, що несанкціонована або шкідлива діяльність буде суттєво відрізнятися від стандартного поведінкового профілю.

Суть поведінкових методів полягає в створенні базових профілів, які рефлектують "здорову" активність в системі [31]. Це може бути, наприклад, звичний трафік мережі, частота запитів до бази даних, або поведінка користувача при роботі з додатками. Після створення такого базового профілю, система постійно моніторить активність, звертаючи увагу на будь-які аномалії або відхилення. Якщо таке відхилення виявлено, система може автоматично вжити певних заходів, що можуть варіюватися від виведення сповіщення для адміністратора до блокування потенційно шкідливої активності.

Цей метод є особливо ефективним для виявлення нульового дня атак та інших невідомих загроз, оскільки він не залежить від попереднього знання про специфічні характеристики атаки. Проте, важливо зазначити, що ефективність поведінкових методів в значній мірі залежить від точності базового профілю, а також від алгоритмів, які використовуються для виявлення аномалій.

Переваги поведінкових методів [32]:

– адаптивність. Здатність адаптуватися до змін у системі та виявляти нові, невідомі загрози;

– висока чутливість. Здатність виявляти найдрібніші відхилення від нормальної поведінки, що може свідчити про атаку;

– універсальність. Можливість застосування в різних доменах та системах без потреби в спеціальних знаннях про конкретні вектори атак;

– проактивність. Здатність виявляти атаки на ранніх стадіях, навіть перед тим, як вони призведуть до реальної шкоди.

Недоліки поведінкових методів:

– висока складність налаштування. Для ефективної роботи потребує глибокого аналізу нормальної поведінки системи;

– часові витрати на «навчання» системи. Система потребує часу для збору достатньої кількості даних для формування точного базового профілю;

– ризик «хибних спрацювань». Якщо атака мімікує нормальну поведінку, вона може залишитися невиявленою;

– витрати на ресурси. Аналіз та моніторинг поведінки в реальному часі може бути ресурсоємким;

– складність інтерпретації результатів. Не завжди очевидно, яке відхилення в поведінці є критичним, а яке — ні.

Отже, поведінкові методи виявлення вторгнень відзначаються високою адаптивністю та чутливістю, що робить їх ефективними для виявлення нових та невідомих кіберзагроз. Вони також універсальні та здатні працювати в різних системах без специфічних налаштувань. Проте, їх використання

супроводжується високою складністю налаштування та потребою у значних ресурсах для аналізу.

2.3.2. Методи на основі знань

Методи на основі знань в системах виявлення вторгнень базуються на використанні попередньо визначених сценаріїв атак або зразків недопустимої поведінки для ідентифікації потенційних загроз. Ці методи оперують на основі бази даних "сигнатур", які є унікальними характеристиками відомих векторів атак.

Суть методів на основі знань полягає в тому, що система виявлення вторгнень періодично сканує мережевий трафік або системні події, порівнюючи їх з базою сигнатур [33]. Якщо виявляється відповідність між аналізованими даними та однією з сигнатур, система генерує сповіщення про можливу атаку. Цей підхід дозволяє з високою точністю виявляти відомі види атак, але не завжди ефективний для виявлення нових, раніше невідомих загроз.

Переваги методів на основі знань [34]:

– висока точність виявлення відомих загроз. Оскільки сигнатури базуються на аналізі реальних атак, методи на основі знань відрізняються високою точністю виявлення вже відомих типів вторгнень;

– швидкість реагування. Процес порівняння поточного мережевого трафіку з базою сигнатур зазвичай є досить швидким, що забезпечує миттєве виявлення вторгнень;

– низька кількість помилкових спрацьовувань. Завдяки точному визначенню сигнатур, ймовірність помилкового виявлення (false positives) є зазвичай нижчою в порівнянні з іншими методами.

– простота налаштування та обслуговування: Оскільки база знань уже вбудована в систему, адміністраторам не потрібно витрачати час на додаткове навчання моделі.

Недоліки методів на основі знань:

– нездатність виявляти нові види атак. Ці методи ефективні лише для виявлення відомих атак і не можуть адекватно реагувати на нові, невідомі загрози;

– потреба в регулярному оновленні бази сигнатур. Для підтримки актуальності системи необхідно постійно оновлювати базу знань, що може бути трудомістким.

– відсутність контекстуального аналізу. Методи на основі знань розглядають кожну подію окремо і не враховують контекст мережевої активності, що може ускладнити виявлення складних атак;

– можливість обходу за допомогою поліморфних атак. Зловмисники можуть спеціально модифікувати код атаки так, щоб він не співпадав із відомими сигнатурами, що уникне виявлення.

Отже, методи виявлення вторгнень на основі знань є ефективними для виявлення відомих типів атак завдяки високій точності та швидкості реагування. Вони також характеризуються низькою кількістю помилкових спрацьовувань і простотою в обслуговуванні. Проте, ці методи мають певні обмеження, такі як неспроможність виявляти нові, невідомі загрози та потреба в регулярних оновленнях бази сигнатур. Відсутність контекстуального аналізу та можливість обходу захисту через поліморфні атаки також є їх слабкими сторонами.

2.3.3. Методи машинного навчання

Методи машинного навчання в контексті виявлення вторгнень засновані на використанні алгоритмів машинного навчання для класифікації аномальної та

нормальної поведінки в мережі або системі. Ці методи намагаються "навчити" модель розпізнавати шаблони або закономірності в даних, що можуть вказувати на наявність атаки.

Суть методів машинного навчання полягає в тому, що вони використовують набір даних для тренування моделі, яка потім може бути використана для аналізу нових, невідомих даних [35]. Процес тренування зазвичай включає в себе вибір фіч (ознак), що найкраще описують аномальні та нормальні дії в системі. Після тренування модель здатна автоматично класифікувати нові спостереження як "нормальні" або "аномальні" на основі навчених шаблонів. Залежно від вибраного алгоритму, можна також визначити ступінь "впевненості" моделі в її класифікації, що може бути корисним для подальшого аналізу та вжиття відповідних заходів.

Переваги методів машинного навчання [36]:

– адаптивність. Моделі машинного навчання можуть адаптуватися до нових типів атак та змін у поведінці системи, що робить їх ефективними проти невідомих загроз;

– висока точність. З правильно підібраними параметрами та достатньою кількістю даних для тренування, методи машинного навчання можуть досягти високої точності виявлення вторгнень;

– автоматизація. Після етапу тренування, процес виявлення вторгнень стає великою мірою автоматизованим, що зменшує навантаження на адміністраторів системи;

– гнучкість у виборі фіч. Методи машинного навчання дозволяють використовувати складні комбінації фіч, що може поліпшити якість класифікації.

Недоліки методів машинного навчання:

– велика кількість даних для тренування. Для ефективної роботи моделі потребують значних обсягів тренувальних даних, що може бути проблематично для нових або маленьких систем;

– чутливість до шуму та викидів в даних. Наявність шуму або неправильно маркованих даних може значно вплинути на якість моделі;

– витрати на обчислення. Залежно від складності моделі, процес тренування та власне виявлення вторгнень може вимагати значних обчислювальних ресурсів;

– потреба в експертних знаннях. Для ефективного використання методів машинного навчання потрібно мати поглиблені знання у даній області, що включає вибір підходящих алгоритмів, фіч та параметрів моделі.

Отже, методи машинного навчання представляють собою потужний інструмент для виявлення вторгнень в системах критичної інфраструктури. Їх адаптивність і висока точність роблять їх привабливим вибором для сучасних систем безпеки. Проте, ефективність цих методів залежить від кількості та якості тренувальних даних, а також від обчислювальних ресурсів і експертних знань для їх налаштування. Тому, при виборі методів машинного навчання для системи захисту, необхідно ретельно враховувати всі ці фактори.

2.3.4. Методи обчислювального інтелекту

Методи обчислювального інтелекту є підмножиною широкого спектру технік штучного інтелекту, які застосовуються для розв'язання складних проблем, зокрема в домені кібербезпеки. Ці методи включають в себе нейронні мережі, нечітку логіку, еволюційні алгоритми та інші техніки, які імітують аспекти природного інтелекту для аналізу, вивчення та реагування на відомі та невідомі загрози.

Суть методів обчислювального інтелекту полягає в застосуванні імітації природних процесів обробки інформації для виявлення аномалій та патернів у даних. Наприклад, нейронні мережі можуть бути натреновані для ідентифікації патернів в мережевому трафіку, які можуть вказувати на вторгнення [37].

Еволюційні алгоритми можуть бути використані для оптимізації правил фільтрації мережевого трафіку, а системи на основі нечіткої логіки можуть допомогти в управлінні неоднозначністю та невизначеністю в сценаріях виявлення вторгнень.

Переваги методів обчислювального інтелекту [38]:

– адаптивність. Методи обчислювального інтелекту здатні адаптуватися до нових видів загроз та відмінностей у мережевому трафіку без необхідності ручного оновлення правил;

– обробка неструктурованих даних. Ці методи ефективно працюють з неструктурованими та напівструктурованими даними, що є корисним при аналізі великих наборів даних;

– висока точність. За рахунок використання складних алгоритмів, методи обчислювального інтелекту часто демонструють високу точність у виявленні аномалій та несанкціонованої активності;

– самонавчання. Деякі методи обчислювального інтелекту, такі як нейронні мережі, здатні до самонавчання на основі нових даних, що покращує їх ефективність з часом.

Недоліки методів обчислювального інтелекту:

– висока складність. Не всі організації мають доступ до експертів, здатних ефективно розробляти та підтримувати системи на основі обчислювального інтелекту;

– високі вимоги до обчислювальних ресурсів. Методи такого типу часто вимагають значних обчислювальних потужностей для тренування моделей та обробки даних в реальному часі;

– чутливість до якості даних. Погано підготовлені або неповні дані можуть суттєво погіршити ефективність методів обчислювального інтелекту;

– ризик перенавчання. При некоректному проектуванні алгоритмів існує ризик перенавчання, коли модель стає занадто "пристрасною" до тренувальних даних і погано справляється з новими, невідомими ситуаціями.

Отже, методи обчислювального інтелекту представляють собою потужний інструмент для виявлення вторгнень, який відрізняється високою адаптивністю, точністю та можливістю самонавчання. Водночас, їх застосування може бути обмеженим через високі вимоги до обчислювальних ресурсів та експертних знань для ефективної реалізації. Також важливою є проблема якості вхідних даних та ризик перенавчання. З урахуванням цих факторів, методи обчислювального інтелекту найбільш ефективно можуть бути використані в складних системах захисту, де їх переваги можна максимально реалізувати.

2.3.5. Порівняльний аналіз методів вторгнення та обґрунтування вибору методу для розробки системи захисту об’єктів критичної інфраструктури

Для визначення актуальної потреби в надійних методах виявлення вторгнень в системи критичної інфраструктури, було здійснено аналітичний огляд доступних наукових та практичних підходів. Основною метою цього аналізу є виокремлення оптимальних методів, які можуть бути застосовані в рамках розроблюваної системи. Для цього інформація була агрегована і структурована у вигляді табл. 2.1.

Таблиця 2.1

Порівняльний аналіз виявлення методів вторгнення

Характеристика	Поведінкові методи	Методи на основі знань	Методи машинного навчання	Методи обч. інтелекту
Витрати часу	+	+	–	–
Витрати пам’яті	–	+	+	+
Складність реалізації	+	–	+	+

Продовження табл. 2.1

Немає адаптивності	+	+	–	–
Неточність (на відомих даних)	+	+	–	–

Позначення "+" і "-" в таблиці вказують на наявність або відсутність певного недоліку для конкретного методу. Наприклад, у поведінкових методах є недоліки у всіх розглянутих категоріях, тоді як методи машинного навчання та обчислювального інтелекту виявилися більш вигідними з точки зору адаптивності та точності на відомих даних.

Тепер потрібно розглянути кожну групу методів окремо з погляду обраних критеріїв:

– поведінкові методи. Ці методи показали недоліки по всіх розглянутих параметрах: вони є затратними по часу, вимагають великих об'ємів пам'яті, складні в реалізації, не адаптивні та можуть виявлятися неточними на відомих даних;

– методи на основі знань. Ці методи також виявилися затратними по часу і складними в реалізації, але їх основна проблема полягає в можливій неточності виявлення вторгнень на відомих даних;

– методи машинного навчання. Ці методи є менш затратними по часу та показують високу точність на відомих даних. Проте їх слабкість полягає в тому, що вони можуть бути складними в реалізації та вимагають значних об'ємів пам'яті;

– методи обчислювального інтелекту. Подібно до методів машинного навчання, ці методи також є менш затратними по часу та виявляють високу точність на відомих даних. Їх основний недолік — великі вимоги до пам'яті та потенційна складність реалізації.

На підставі проведеного аналізу можна зробити висновок про доцільність розробки нового методу виявлення вторгнень, який би враховував переваги методів машинного навчання та обчислювального інтелекту, але уникав їх

недоліків. Ідеальний метод повинен бути ефективним по часу, мінімізувати витрати пам'яті, бути простим у реалізації, адаптивним до змін у вхідних даних, а також точним на відомих даних.

2.4. Вибір засобів для розробки системи захисту об’єктів критичної інфраструктури

Для розробки комплексної системи захисту критичної інфраструктури важливо вибрати оптимальну мову програмування, яка відповідає всім функціональним та технічним вимогам. В даному розділі буде проведено аналіз декількох популярних мов програмування: C#, Python та Java.

C# - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft у 2000 році як частина платформи .NET [39]. Вона комбінує елементи мов програмування C и C++, а також має схожість з Java. C# широко використовується для розробки веб-додатків, десктопних програм, гр, а також рішень в сфері кібербезпеки [40]. Ця мова відрізняється високою продуктивністю, сильною типізацією та наявністю розширеної стандартної бібліотеки, що забезпечує широкі можливості для розробки складних систем.

Python - це високорівнева, інтерпретована мова програмування з динамічною типізацією, яка була створена Гвідо ван Россумом у 1991 році. Завдяки своєму простому та чистому синтаксису, Python став популярним вибором для початківців у програмуванні. Він знайшов широке застосування в розробці веб-додатків, автоматизації, наукових досліджень, аналізу даних, штучного інтелекту та машинного навчання.

Java - це високорівнева, об'єктно-орієнтована мова програмування, яку створила компанія Sun Microsystems у 1995 році. Завдяки принципу "Напиши одного разу, запускай будь-де" і використанню віртуальної машини Java (JVM), Java здатна працювати на різних платформах. Вона широко використовується у

розробці корпоративних додатків, веб-додатків, мобільних додатків для платформи Android, а також в системах вбудованих пристроїв та Інтернету речей.

У табл. 2.2 подано систематичний аналіз ключових функціональних характеристик трьох мов програмування: Python, Java та C#. Оцінка виконана за такими критеріями, як система типів, парадигми програмування, методи виконання коду, особливості синтаксису, наявність фреймворків для розробки веб-додатків, мобільних платформ, наукових досліджень, ігрової розробки, а також ступінь крос-платформенності.

Таблиця 2.2

Порівняльний аналіз мов програмування

Властивість	Python	Java	C#
1	2	3	4
Система типів	Змінна	Фіксована	Фіксована
Парадигми програмування	Об'єктно-орієнтований, імперативний	Об'єктно-орієнтований	Об'єктно-орієнтований, модульний
Метод виконання	Інтерпретація	JVM, байт-код	CLR, проміжний код
Правила синтаксису	Інтуїтивний, блоки через відступи	Структурований, схожий на C++	Структурований, наслідує C++ та Java
Фреймворки для веба	Django, Flask	Spring, Servlets	ASP.NET, Blazor
Платформи для мобільної розробки	Kivy, SL4A	Android Native	Xamarin, .NET MAUI

Продовження табл. 2.2

1	2	3	4
Дослідницькі та аналітичні бібліотеки	SciPy, TensorFlow	Weka, MOA	Math.NET, Accord.NET
Сумісність з платформами	Велика	Велика	Велика (через .NET Core)
Геймдев фреймворки	Pygame, Godot	LibGDX, jMonkey	Unity, Stride

З цього аналізу можна зробити висновок, що C# має ряд переваг, які роблять його ідеальним вибором для розробки системи захисту критичної інфраструктури. Серед них: фіксована система типів для раннього виявлення помилок, висока сумісність з різними платформами через .NET Core, а також наявність розширеного набору фреймворків і бібліотек для реалізації широкого спектру задач.

При проектуванні системи захисту об'єктів критичної інфраструктури також важливим є вибір системи управління базами даних (СУБД). Цей вибір має прямий вплив на ряд оперативних характеристик, зокрема на швидкість доступу до даних, масштабованість, стабільність роботи та, безумовно, на рівень безпеки зберіганої інформації. Враховуючи ці аспекти, розглядаються три основні СУБД: Microsoft SQL Server, PostgreSQL та MongoDB.

Microsoft SQL Server є частиною екосистеми продуктів Microsoft і відзначається високою надійністю [41]. Особливо цінним є її зручна інтеграція з іншими технологіями від Microsoft, включаючи .NET Framework. Крім того, ця СУБД пропонує розширений набір інструментів для аналізу та оптимізації SQL-запитів, а також автоматичного резервного копіювання.

PostgreSQL є відкритою системою управління реляційними базами даних, яка є відомою своєю високою продуктивністю та масштабованістю [42]. Система підтримує широкий спектр типів даних, у тому числі і користувацькі типи, та

дозволяє використовувати складні SQL-запити. Гнучкість системи забезпечується її відкритим кодом, що дозволяє адаптувати СУБД під конкретні задачі.

MongoDB є представником баз даних NoSQL і оптимізована для роботи з великими об'ємами неструктурованих даних. Ця СУБД демонструє високу швидкість читання та запису, а також можливість гнучкої модифікації схеми бази даних.

У табл. 2.3 проведено детальний порівняльний аналіз трьох ключових систем управління базами даних: Microsoft SQL Server, PostgreSQL та MongoDB. Цей аналіз охоплює такі характеристики як надійність, масштабованість, сумісність з іншими технологіями, рівень безпеки, швидкість читання та запису даних, а також підтримку транзакцій.

Таблиця 2.3

Порівняльний аналіз сховищ БД

Характеристика	Microsoft SQL Server	PostgreSQL	MongoDB
Інтеграція з технологіями	Висока	Низька	Низька
Висока доступність	Так	Часткова	Часткова
Підтримка транзакцій	Повна	Повна	Обмежена
Характеристика	Microsoft SQL Server	PostgreSQL	MongoDB
Надійність	Висока	Висока	Висока
Масштабованість	Висока	Висока	Висока
Сумісність	Висока (.NET)	Середня	Низька
Безпека	Висока	Висока	Середня
Швидкість читання/запису	Висока	Висока	Високий

На основі аналізу таблиці можна зробити висновок, що Microsoft SQL Server переважає конкурентів за такими характеристиками як "Сумісність" та

"Безпека". Особливо важливою є висока сумісність із платформою .NET, що є ключовим фактором при виборі СУБД для системи, розробленої на C#. Це забезпечує гармонійну інтеграцію і спрощує розробку та підтримку системи. Крім того, висока безпека в Microsoft SQL Server дозволяє реалізувати надійні механізми захисту даних, що є критичним для об'єктів критичної інфраструктури.

2.5. Постановка задачі

На підставі аналітичного огляду методів виявлення вторгнень в інформаційні системи критичної інфраструктури визначено основні напрямки для розробки інноваційної системи безпеки. Одним з ключових завдань є створення механізму, який забезпечить неперервний моніторинг усього мережевого трафіку та ефективне виявлення аномальних відхилень в поведінці мережі.

Паралельно акцентується увага на розробці інтуїтивно зрозумілого користувацького інтерфейсу, який має супроводжуватися чіткими та інформативними повідомленнями для спрощення процесу управління та моніторингу. Це передбачає також імплементацію навігаційного меню в головному вікні програми для забезпечення швидкого та зручного доступу користувача до всіх необхідних інструментів та функцій системи.

Програмне забезпечення буде містити три ключові модулі: модуль забезпечення персональної безпеки користувача, який відповідатиме за аутентифікацію та авторизацію; модуль вводу та редагування інформації, що дозволить користувачам додавати, змінювати або видаляти дані; а також модуль аналізу та моніторингу мережевого трафіку, відповідальний за виявлення аномалій та можливих вторгнень.

Ця багатомодульна структура сприятиме не тільки спрощенню процесу розробки, але й забезпечить високу гнучкість та масштабованість системи. Розробка ведеться з використанням інтегрованого середовища розробки Visual Studio 2022 та мови програмування C#. Програма, що розробляється, отримає назву "Аналізатор мережевого трафіку", відображаючи її основну функціональну спрямованість.

2.6. Висновок

У другому розділі було проведено аналіз різних методів виявлення вторгнень, що дозволило глибоко зрозуміти потенціал та обмеження кожного підходу у контексті захисту об'єктів КІ. З особливим акцентом на поведінкові методи, методи на основі знань, методи машинного навчання та методи штучного інтелекту, було проведено порівняльний аналіз, який підтвердив значущість і ефективність вибраного напрямку дослідження.

На основі цього аналізу була розроблена загальна методика проведення досліджень, що охоплює підготовчий етап, вибір та аналіз методів, розробку тестового середовища, проведення експериментів та аналіз результатів. Це дозволило систематично та об'єктивно підійти до процесу визначення найбільш ефективного методу для розробки системи захисту.

Вибір мови програмування C# для розробки системи захисту об'єктів критичної інфраструктури був обумовлений її функціональністю, гнучкістю та зручністю використання. Це дозволило ефективно реалізувати задумані функції системи та забезпечити необхідний рівень безпеки.

Завершення цього розділу включало постановку чітко визначених задач для розробки системи захисту.

Розділ 3. РОЗРОБКА ТА АПРОБАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ ДЛЯ ЗАХИСТУ ОБ'ЄКТІВ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

3.1. Реалізація схеми аналізатора мережевого трафіку

В результаті деталізованого аналітичного огляду методів виявлення вторгнень в інформаційні системи критичної інфраструктури, було визначено ключові критерії для розробки інноваційної системи безпеки. Однією з важливих цілей є розробка механізму, який забезпечить здатність системи до неперервного моніторингу 100% мережевого трафіку, а також ефективно виявляти аномальні відхилення в поведінці мережі.

Акцентується увага на створенні інтуїтивно зрозумілого користувацького інтерфейсу. Взаємодія користувача з системою має бути супроводжена чіткими та інформативними повідомленнями, які спростять процес управління та моніторингу.

Додатково, планується впровадити в системі навігаційне меню в головному вікні програми. Ця функціональність матиме на меті надання користувачу максимально швидкого та зручного доступу до всіх необхідних інструментів та функцій системи.

В архітектурі програмного забезпечення передбачається реалізація трьох ключових модулів:

– модуль забезпечення персональної безпеки користувача, який відповідатиме за аутентифікацію та авторизацію;

– модуль вводу та редагування інформації, що надасть можливість користувачам додавати, змінювати або видаляти дані в системі;

– модуль аналізу та моніторингу мережевого трафіку, який буде відповідальний за виявлення аномалій та можливих вторгнень.

Ця багатомодульна структура не лише спростить процес розробки, але і забезпечить високу гнучкість та масштабованість системи.

Для забезпечення виконання поставлених завдань та реалізації обраного методу виявлення вторгнень в системах критичної інфраструктури буде використовуватися інтегроване середовище розробки Visual Studio 2022. Мовою програмування, обраною для реалізації даного проекту, є C#. Створена програма отримає назву "Аналізатор мережевого трафіку".

В контексті архітектурного проектування системи, розробляється діаграма взаємодії між різними компонентами — клієнтами та серверами — в мережі. Ця діаграма представлена на рис. 3.1 і служитиме основою для детального аналізу функціональності та взаємодії модулів розробленого програмного забезпечення "Аналізатор мережевого трафіку".

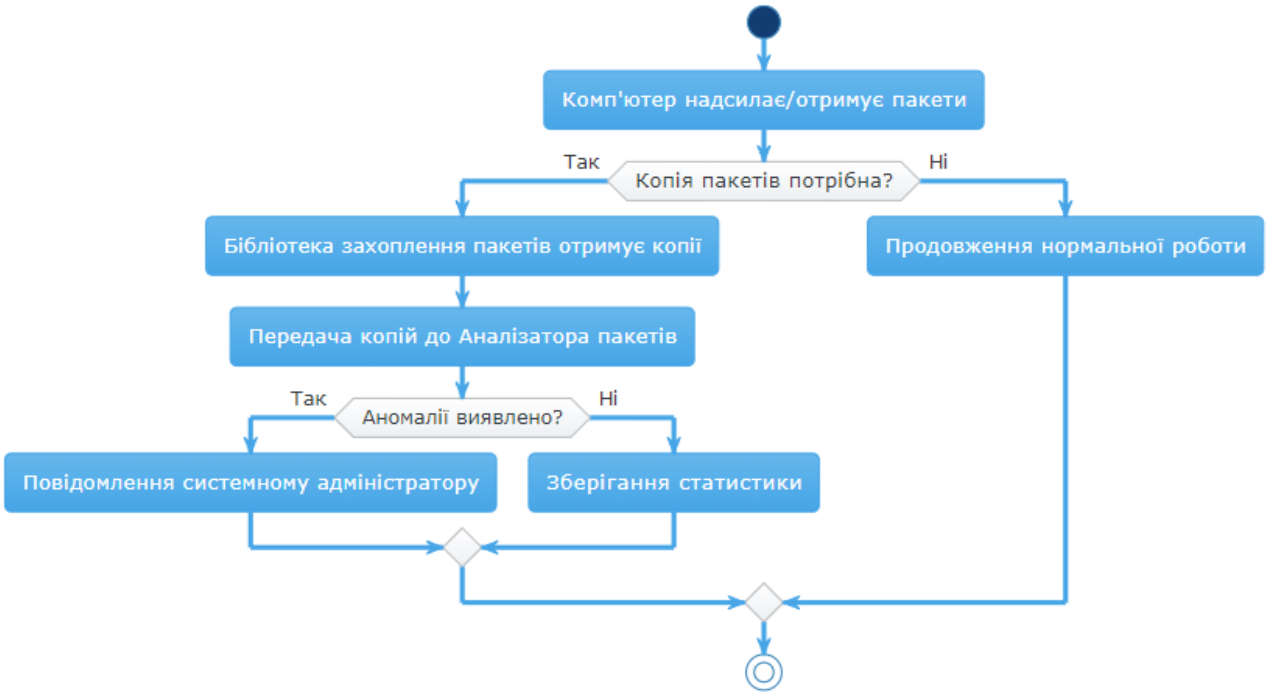

Рис. 3.1. Діаграма активності аналізатора мережевого трафіку

Задля аналізу пакетів даних в програмному забезпеченні розроблено спеціалізований аналізатор пакетів. Цей аналізатор є пасивним компонентом, який не втручається в роботу мережі, а лише спостерігає за даними, що

передаються. Комп'ютер, на якому запущено дане програмне забезпечення, взаємодіє з різноманітними протоколами та програмами для надсилання та отримання пакетів даних.

В рамках цього процесу, бібліотека захоплення пакетів отримує копії всіх пакетів, які надійшли або були відправлені комп'ютером. Ці копії пакетів передаються до аналізатора пакетів. Аналізатор пакетів, в свою чергу, здійснює глибокий аналіз цих пакетів. Він "розуміє" структуру кадрів Ethernet, IP-дейтаграм, TCP-сегментів та інших протокольних елементів, що дозволяє йому витягувати конкретні поля та інформацію з пакетів.

Під час аналізу пакетів, аналізатор також збирає статистичні дані про мережеву активність. Ця інформація може бути корисною для системного адміністратора для моніторингу загального стану мережі. У випадку виявлення аномальної мережевої активності, такої як нехарактерні піки в трафіку або незвичайні патерни передачі даних, аналізатор пакетів автоматично повідомляє системного адміністратора для подальшого розслідування та вжиття необхідних заходів.

Таким чином, розроблений аналізатор пакетів виступає як важливий інструмент для моніторингу та аналізу мережевого трафіку, що дозволяє забезпечити високий рівень безпеки та ефективності мережевої інфраструктури.

3.2. Практична реалізація системи виявлення вторгнень

Для створення високоефективної програми, спеціалізованої на аналітичному дослідженні TCP пакетів мережі з акцентом на виявлення нерегулярних або аномальних патернів поведінки, вирішено використати архітектурний підхід на основі трьох рівнів. Цей підхід є відомим своєю універсальністю та широким спектром застосувань в сфері програмної інженерії:

– рівень користувацького інтерфейсу. Цей рівень відповідає за створення та управління графічним користувацьким інтерфейсом. Він включає в себе елементи управління, такі як вікна, кнопки, навігаційні меню та візуальні віджети, які забезпечують зручну взаємодію користувача з програмою;

– рівень логіки додатку. Це сердце програмної системи, де інкапсульовані основні функціональні алгоритми та бізнес-процеси;

– рівень управління даними: Це базовий рівень архітектури, який контролює взаємодію з зовнішніми джерелами даних, такими як бази даних, файли або мережеві ресурси.

За допомогою такої трьохрівневої моделі можливо забезпечити гнучкість, модульність та легкість управління кодом.

На початку було реалізовано діаграму класів рівня даних, що зображена на рис. 3.2.

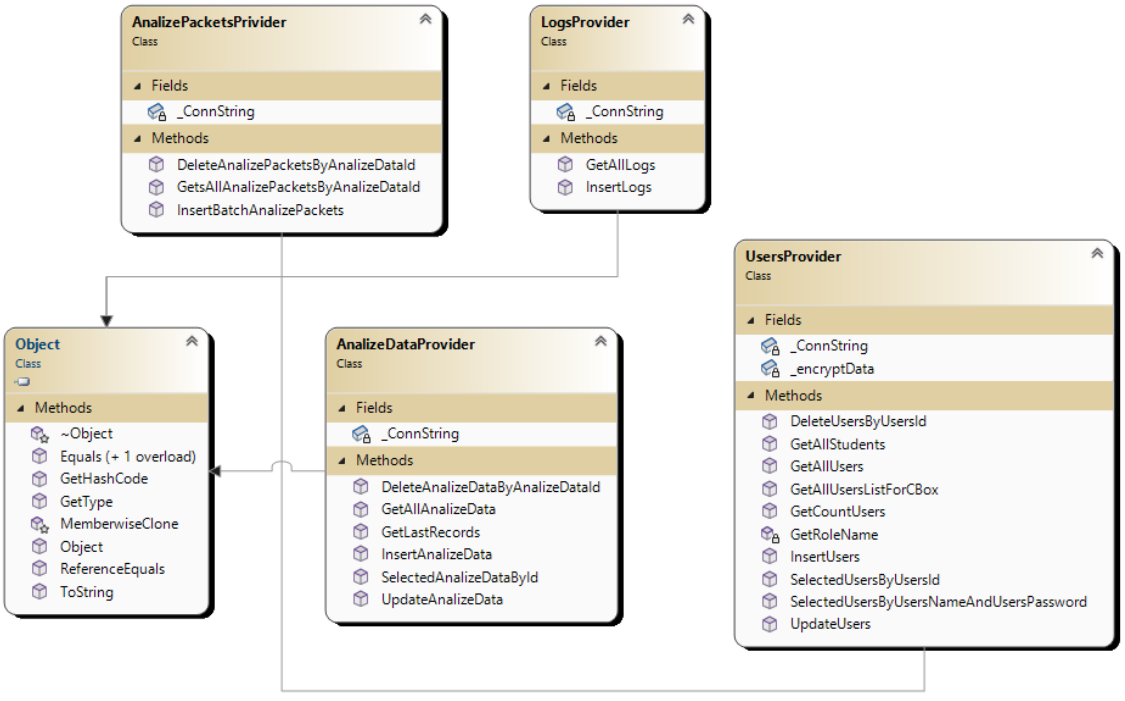

Рис. 3.2. Діаграма класів рівня даних

В архітектурі програмного забезпечення рівень управління даними відіграє ключову роль у забезпеченні стійкої та ефективної взаємодії між логікою додатку та зовнішніми джерелами інформації. В цьому контексті, ізольовані

класи, що спеціалізуються на роботі з певними видами даних, стають необхідними для модулярності та зручності управління. Опис призначення кожного з названих класів:

– AnalizeDataProvider. Цей клас відповідає за збір, обробку та зберігання даних, що стосуються аналітичних засобів системи. Він включає методи для виконання SQL-запитів, завантаження даних з бази або інтерфейсів API та їх подальшої обробки для аналізу;

– AnalizePacketsProvider. Спеціалізований клас для роботи з мережевими пакетами. Його основне призначення — збір, фільтрація та аналіз TCP пакетів. Клас включає в себе алгоритми для виявлення аномальних патернів або несанкціонованого доступу;

– LogsProvider. Клас, що концентрується на роботі з системними журналами. Його завдання — зберігання, управління та аналіз логів, що генеруються іншими компонентами системи. Це корисно для аудиту, моніторингу стану системи та виявлення можливих проблем або зловживань;

– UsersProvider. Клас, відповідальний за управління даними користувачів. Це включає в себе автентифікацію, авторизацію, зберігання користувацьких профілів та іншу інформацію, яка відноситься до взаємодії користувача з системою.

Для забезпечення користувацького інтерфейсу системи для аналізу мережевих пакетів TCP була створена набір інтерактивних форм. Ці форми діють як візуальний інтерфейс, забезпечуючи гладкий перехід між користувачем та основними алгоритмами та бізнес-процесами системи. Інтерфейс включає різноманітні елементи управління: кнопки для запуску конкретних задач, текстові області для внесення користувацьких даних, а також решіткові компоненти для візуалізації отриманих результатів аналізу.

На рис. 3.3 представлена схема, що демонструє архітектуру та взаємозв'язки компонентів на рівні користувацького інтерфейсу системи.

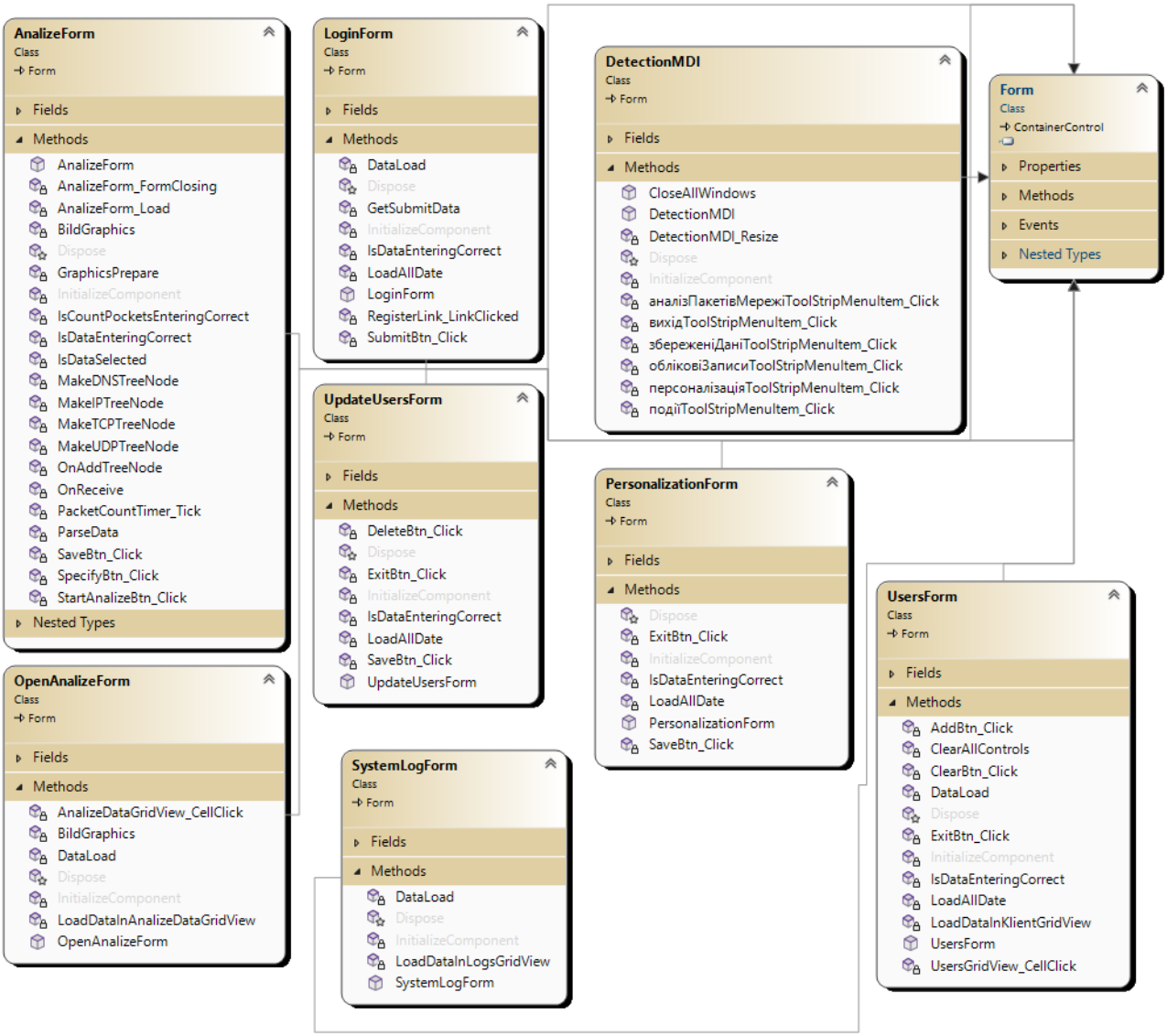

Рис. 3.3. Діаграма класів інтерфейсу системи

Діаграма класів даного рівня складається із:

– DetectionMDI. Цей клас слугує основним контейнером для інших форм на рівні користувацького інтерфейсу. Він є точкою входу для взаємодії користувача з системою і координує переключення між різними підвікнами аналізу, логування та персоналізації;

– AnalizeForm. Ця форма забезпечує інтерфейс для взаємодії користувача з алгоритмами аналізу пакетів. Вона включає елементи керування для налаштування параметрів аналізу та візуалізації результатів;

– OpenAnalizeForm. Форма, призначена для відкриття збережених результатів аналізу. Вона дає можливість користувачу переглядати раніше здійснені аналітичні операції;

– LoginForm. Форма автентифікації, що дозволяє користувачу ввести свої облікові дані для доступу до системи;

– PersonalizationForm. Ця форма надає користувачу інструменти для персоналізації робочого простору та налаштування системних параметрів;

– SystemLogForm. Форма для перегляду системних логів. Вона дозволяє користувачам відстежувати дії, здійснені в системі, та забезпечує додатковий рівень прозорості;

– UpdateUsersForm. Форма, що дозволяє адміністраторам системи змінювати облікові дані користувачів, такі як паролі, ролі доступу та інші персональні параметри;

– UsersForm. Ця форма слугує для управління списком користувачів, їхніми ролями та обліковими записами. Вона може бути використана адміністраторами для додавання, видалення чи редагування користувачів.

Кульмінаційним моментом у створенні програмного забезпечення є розробка бізнес-логіки, яка служить нейронним центром додатку. Вона регулює основні операції та бізнес-процеси, на яких базується функціональність системи. Однією з її ключових особливостей є модульність та незалежність від рівнів презентації та доступу до даних, що гарантує гнучкість та масштабованість додатку.

Для ілюстрації архітектурних рішень на рівні бізнес-логіки, розроблено діаграму, яка зображена на рис. 3.4. Цей графічний засіб дає можливість візуалізувати структурні елементи та їхні взаємовідносини в рамках центрального шару програми. Діаграма є інструментом для глибшого розуміння механізмів, що лежать в основі бізнес-логіки, і сприяє ефективній комунікації між розробниками та іншими зацікавленими сторонами.

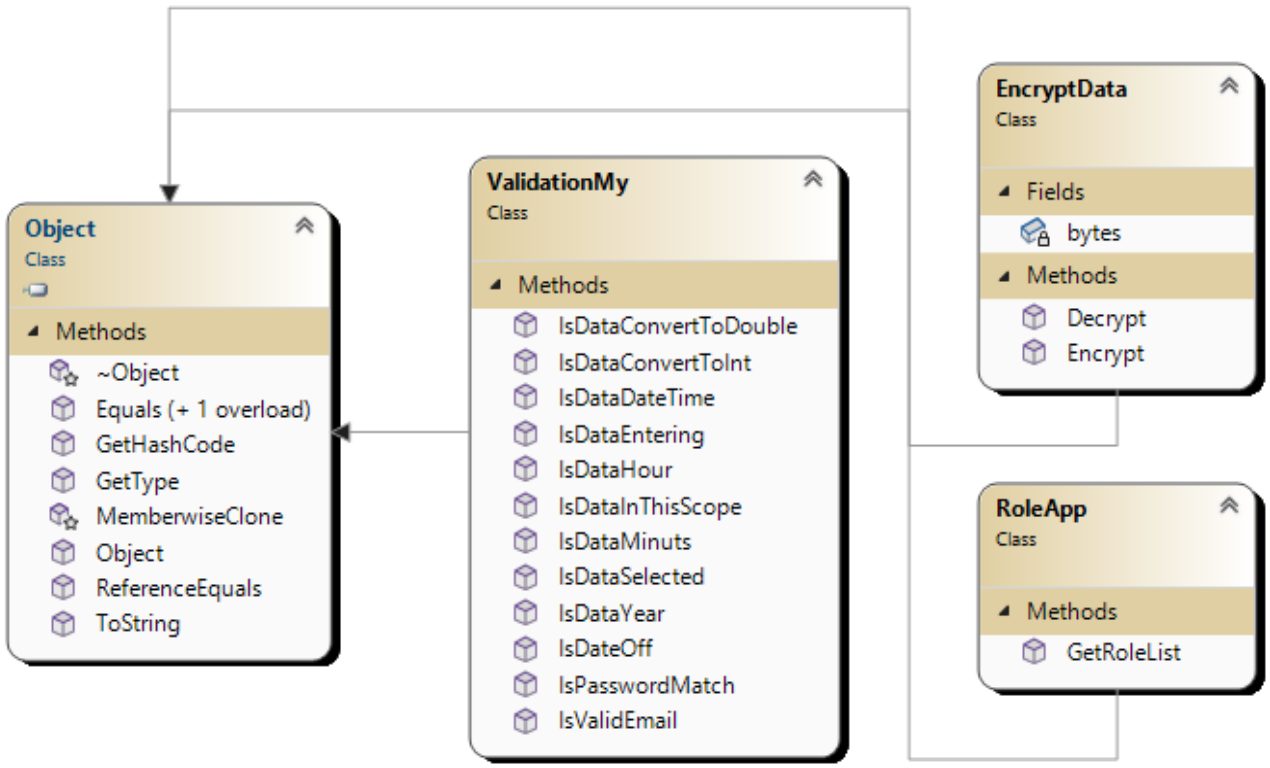

Рис. 3.4. Діаграма класів бізнес-логіки

Діаграма класів рівня бізнес-логіки складається із таких класів:

– RoleApp. Клас, призначений для управління ролями користувачів в додатку. Він слугує для визначення та перевірки дозволів і обмежень, що накладаються на різні типи користувачів (наприклад, адміністратор, звичайний користувач, аналітик тощо). Така модель дозволяє забезпечити контроль доступу до різних функціональних частин системи в залежності від ролі користувача.

– ValidationMy. Клас, який займається валідацією даних, що надходять від користувача або інших частин системи. Це може включати в себе перевірку формату, достовірності, а також санітарну обробку вхідних даних. Клас спрямований на попередження некоректної роботи системи або можливих векторів атак.

– EncryptData. Клас, розроблений для шифрування та дешифрування даних. Це особливо важливо для захисту конфіденційної інформації в процесах

зберігання та передачі даних. Використовуючи сучасні алгоритми шифрування, цей клас забезпечує інтегритет та конфіденційність інформації.

У рамках початкової фази втілення програмного забезпечення активовано процес розробки основного користувацького вікна (рис. 3.5). Це вікно функціонує як головний інтерфейсний елемент, через який користувач матиме доступ до всіх ключових функцій програми. Він становить базову платформу для інтеграції додаткових вікон, управлінських елементів, інструментальних засобів та компонентів для візуалізації даних.

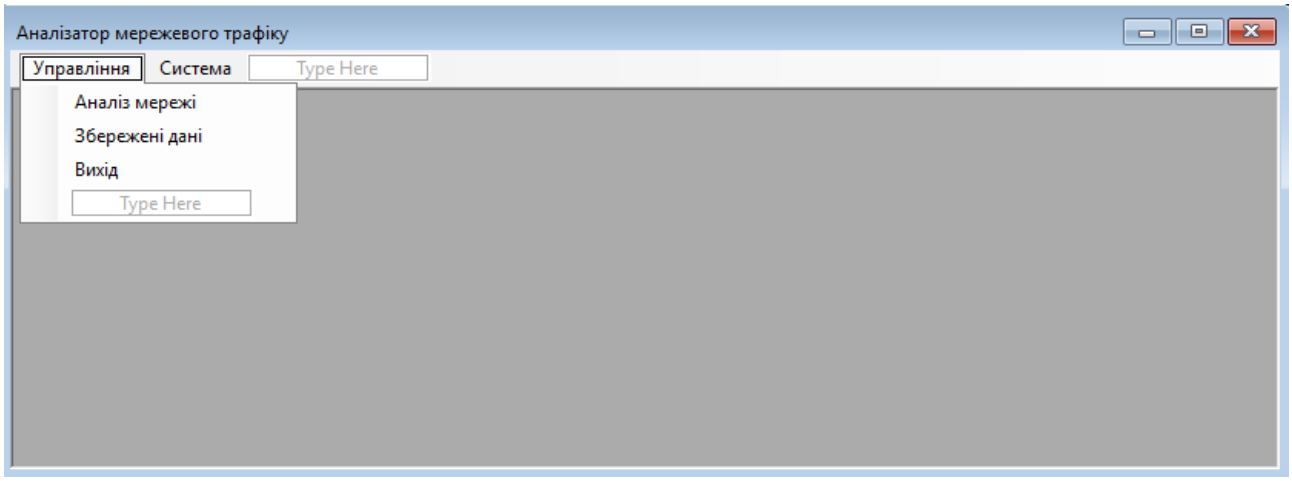

Рис. 3.5. Створене головне вікно та меню системи

Інтеграція компонента menuStrip виконана для створення головного навігаційного меню програми. Ця компонента служить засобом доступу до важливих функцій програми, включаючи управління файлами, конфігурацію та допоміжні опції. Процес її інтеграції охоплював визначення основних розділів меню, настроювання їх візуальних параметрів та прив'язка до відповідних функцій програми. В рамках подальшої оптимізації користувацького досвіду, для кожного пункту меню була розроблена та втілена специфічна логіка взаємодії (рис. 3.6).

```
private void аналізПакетівМережіToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWindows();
    AnalizeForm analizeForm = new AnalizeForm();
    analizeForm.MdiParent = this;
    analizeForm.WindowState = FormWindowState.Maximized;
    analizeForm.Show();
}
```

Рисунок 3.6 – Приклад програмування пункту меню

В рамках комплексного проекту, спрямованого на аналіз мережевих пакетів за протоколом TCP для виявлення аномальних мережевих активностей, одним з найбільш критичних елементів є організована та ефективна взаємодія з базою даних. З метою оптимізації цього процесу та централізації відповідної бізнес-логіки, було розроблено спеціфічну архітектуру класів. Ці класи розміщено в окремій папці "Providers" в структурі програмного проекту. Така організація коду не тільки полегшує його навігацію та підтримку, але також сприяє зрозумілості логічної структури програмного засобу.

Процес з'єднання з базою даних відбувається через методи "Open" та "Close", які входять до складу класу "SqlConnection". Використання цих методів забезпечує не тільки надійність з'єднання, але і консистентність та цілісність даних, що є особливо важливим в контексті аналізу мережевих пакетів.

У центрі взаємодії з базою даних перебуває клас "AnalizePacketsProvider", розроблений спеціально для обробки інформації про мережеві пакети. Клас містить методи, що дозволяють ефективно додавати зібрану інформацію про мережеві пакети до бази даних, а також забезпечують можливість витягувати цю інформацію для подальшого аналізу. Така структурна організація класів і методів відіграє ключову роль в архітектурі системи, забезпечуючи її гнучкість, масштабованість та надійність.

Сумарно, ці архітектурні рішення складають гармонійну систему для надійної та ефективної роботи з даними. Рис. 3.7 подає візуальний огляд структури коду, який відповідає за додавання даних про перехоплені пакети в

базу даних, що сприяє кращому розумінню механізмів взаємодії з базою даних в рамках даного проекту.

```
public void InsertBatchAnalizePackets(List<AnalizePackets> AnalizePackets) {
  string SqlString = "INSERT INTO AnalizePackets (PacketsCount, PacketsDate, AnalizeDataId) " +
    "Values(@PacketsCount, @PacketsDate, @AnalizeDataId)";
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      conn.Open();
      for (int i = 0; i < AnalizePackets.Count; i++) {
        cmd.Parameters.AddWithValue("@PacketsCount", AnalizePackets[i].PacketsCount);
        cmd.Parameters.AddWithValue("@PacketsDate", AnalizePackets[i].PacketsDate);
        cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizePackets[i].AnalizeDataId);
        cmd.ExecuteNonQuery();
        cmd.Parameters.Clear();
      }
      conn.Close();
    }
  }
}
```

Рисунок 3.7 – Реалізація методу для додавання інформації про пакети у БД

Метод "InsertBatchAnalizePacket" представляє собою критичний компонент в архітектурі системи аналізу мережевих пакетів за протоколом TCP. Інтегрований в клас, що спеціалізується на взаємодії з базою даних, цей метод є відповідальним за ефективне зберігання зібраної інформації про пакети в реляційній базі даних. Його реалізація взаємодіє із схемами бази даних, що розроблені спеціфічно для структурованого зберігання параметрів мережевих пакетів. В цьому контексті метод стає інструментом, що сприяє глибокому аналізу мережевих активностей та виявленню аномалій.

Паралельно було сконструйовано метод для вибірки інформації про раніше зібрані та збережені пакети. Цей метод характеризується високою ступенем складності, адже він не просто забезпечує доступ до збережених даних, але і реалізує алгоритми для їх оптимальної вибірки з урахуванням різних параметрів, таких як час, тип пакету та інші. Все це відображено в коді цього методу, що є частиною більшого екосистеми для обробки та аналізу мережевого трафіку.

Рис. 3.8 служить візуальним підтвердженням структури та функціональних можливостей згаданого методу вибірки. Це дає можливість зрозуміти, як метод

взаємодіє із базою даних та іншими компонентами системи, сприяючи таким чином комплексному аналізу мережевого трафіку.

```
public List<AnalizePackets> GetsAllAnalizePacketsByAnalizeDataId(int AnalizeDataId) {
  int i = 0;
  string SqlString = "SELECT * FROM AnalizePackets WHERE AnalizeDataId=@AnalizeDataId";

  List<AnalizePackets> AnalizePackets = new List<AnalizePackets>();
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizeDataId);
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          AnalizePackets oneAnalizePackets = new AnalizePackets();
          oneAnalizePackets.Number = ++i;
          oneAnalizePackets.AnalizePacketsId = Convert.ToInt32(reader["AnalizePacketsId"]);
          oneAnalizePackets.PacketsCount = Convert.ToInt32(reader["PacketsCount"]);
          oneAnalizePackets.PacketsDate = Convert.ToDateTime(reader["PacketsDate"]);
          oneAnalizePackets.AnalizeDataId = Convert.ToInt32(reader["AnalizeDataId"]);
          AnalizePackets.Add(oneAnalizePackets);
        }
      }
      conn.Close();
    }
  }
  return AnalizePackets;
}
```

Рисунок 3.8 – Код методу «GetAllAnalizePacketByAnalizeDataId»

В наступних етапах розробки програмного забезпечення особливий акцент було зроблено на конструюванні специфічної форми, яка зосереджена на обробці пакетів мережевого протоколу TCP. Цей важливий компонент програми визначає основну логіку взаємодії з мережевими пакетами, включаючи їх аналіз, фільтрацію та відображення в узгодженому форматі. Форма була розроблена з урахуванням можливості масштабування та адаптації до різних видів мережевого трафіку.

Візуальна концепція цієї форми, в якій інкапсульовано ключові аспекти функціональності, піддавалася множинному коректуванню та тестуванню для оптимізації користувацького досвіду. Зокрема, було приділено увагу елементам керування, які дозволяють користувачеві взаємодіяти з системою в максимально інтуїтивний та ефективний спосіб.

Деталізоване візуальне представлення створеної форми і її інтерфейсних елементів можна побачити на рис. 3.9. Цей рисунок служить не лише ілюстративною доповненням, але і відображає структурну організацію та ієрархію елементів управління на розробленій формі.

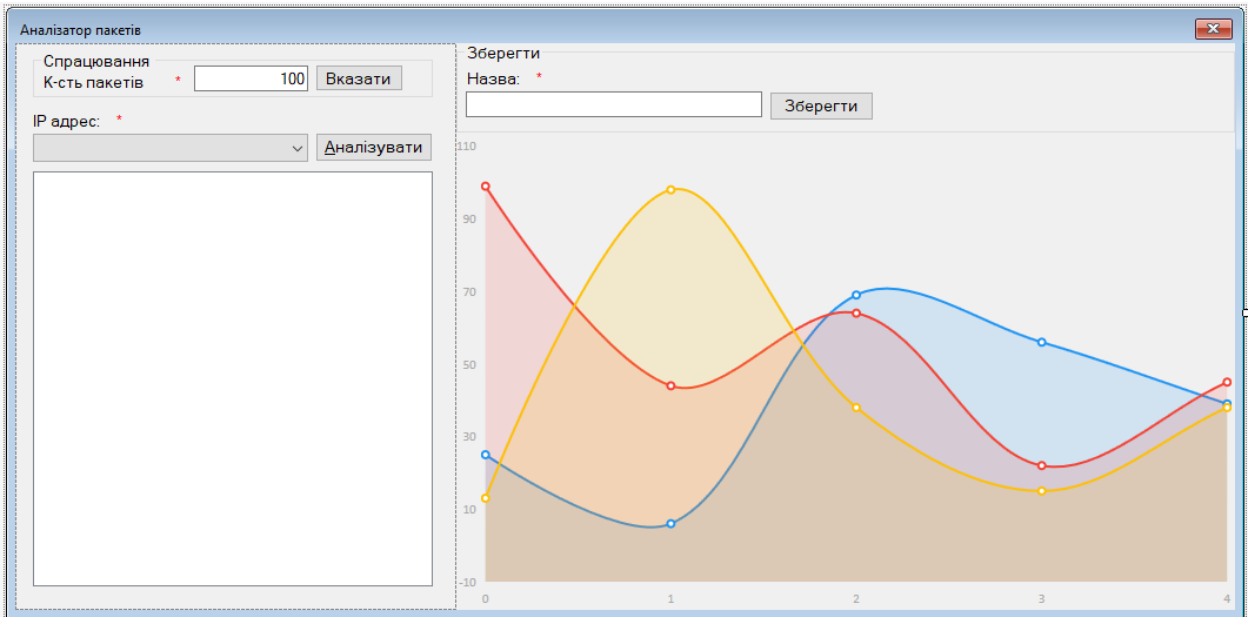

Рисунок 3.9 – Форма для перехоплення пакетів даних

Під час ініціації завантаження форми для аналізу пакетів протоколу TCP, система автоматично генерує подію, відому як подія завантаження форми. У контексті цієї події викликається метод «AnalizeForm_Load», який відіграє критичну роль в архітектурі програмного рішення. Цей метод є спеціалізованим компонентом, який забезпечує ініціалізацію даних, необхідних для подальшого аналізу пакетів.

Однією з основних функцій методу «AnalizeForm_Load» є збір інформації про всі доступні IP-адреси вузла, на якому запущено додаток. Ця інформація є вкрай необхідною для аналізу пакетів, адже вона стає вихідним параметром для фільтрації мережевого трафіку та визначення потенційно аномальних активностей.

Реалізація цього методу була ретельно протестована для забезпечення точності збору даних та їхньої надійності. Даний метод використовує ряд системних та додаткових бібліотек для доступу до мережевих інтерфейсів і здійснення необхідних операцій з IP-адресами.

Для наглядного представлення архітектурних рішень та внутрішньої логіки методу «AnalizeForm_Load», в документації зазначено візуальний знімок цього процесу, який можна переглянути на рис. 3.10.

```
private void AnalizeForm_Load(object sender, EventArgs e) {
  string strIP = null;
  IPHostEntry HosyEntry = Dns.GetHostEntry((Dns.GetHostName()));
  if (HosyEntry.AddressList.Length > 0) {
    foreach (IPAddress ip in HosyEntry.AddressList) {
      strIP = ip.ToString();
      IPHostInterfacesCBox.Items.Add(strIP);
    }
  }
}
```

Рисунок 3.10 – Метод збирання інформації IP адрес всіх вузлів

Під час етапу розробки користувацького інтерфейсу для системи аналізу мережевих пакетів протоколу TCP було вирішено включити спеціалізовану кнопку «Аналізувати». Ця кнопка відіграє важливу роль в керуванні функціональними можливостями додатку, забезпечуючи взаємодію користувача з системою. Для реалізації логіки, що забезпечує функціональність цієї кнопки, було створено специфічний обробник подій, який отримав назву «StartAnalizeBtn_Click».

Обробник подій «StartAnalizeBtn_Click» є програмним компонентом, який активізується автоматично, коли користувач натискає кнопку «Аналізувати». Цей обробник подій запускає складний ланцюг операцій, який включає в себе збір, обробку та аналіз пакетів протоколу TCP.

Важливо відзначити, що реалізація цього обробника подій вимагала розуміння не тільки основ протоколу TCP, але й складних алгоритмів аналізу даних. Код методу, що реалізує логіку обробника події, є досить складним і

використовує ряд системних та додаткових бібліотек для забезпечення необхідної функціональності.

Для наглядності та глибшого розуміння принципів роботи обробника подій «StartAnalizeBtn_Click», в документації приведено візуальний знімок цього коду. Цей знімок можна знайти на рис. 3.11, який служить відмінним інструментом для аналізу та перевірки внутрішньої структури та логіки коду.

```
private void StartAnalizeBtn_Click(object sender, EventArgs e) {
  if (IsDataSelected()) {
    if (!PacketCountTimer.Enabled) {
      StartAnalizeBtn.Text = "&Зупинити";
      _BContinueCapturing = true;
      try {
        //Почати захоплення пакетів
        _MainSockets = new Socket(AddressFamily.InterNetwork, SocketType.Raw, ProtocolType.IP);
        //Прив'язуємо сокет до вибраної IP-адреси
        _MainSockets.Bind(new IPEndPoint(IPAddress.Parse(IPHostInterfacesCBox.Text), 0));
        //Встановлюємо параметри сокету
        _MainSockets.SetSocketOption(SocketOptionLevel.IP,            //Застосовується лише до IP-пакетів
                                     SocketOptionName.HeaderIncluded, //Встановлюємо включення заголовка в true
                                     true);
        byte[] byTru = new byte[4] { 1, 0, 0, 0 };
        byte[] byOut = new byte[4] { 1, 0, 0, 0 }; //Захоплюємо вихідні пакети

        //Socket.IOControl аналогічний методу  WSAIoctl
        _MainSockets.IOControl(IOControlCode.ReceiveAll, byTru, byOut); //Еквівалент константи SIO_RCVALL Winsock 2
        //Починаємо отримувати пакети асинхронно
        _MainSockets.BeginReceive(_BytesData, 0, _BytesData.Length, SocketFlags.None,  new AsyncCallback(OnReceive), null);
      } catch (Exception ex) {
        MessageBox.Show(ex.Message, "Аналіз пакетів", MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
    } else {
      StartAnalizeBtn.Text = "&Аналізувати";
      PacketCountTimer.Stop();
      _BContinueCapturing = false;
      //Щоб припинити захоплення пакетів, закриваємо сокет
      _MainSockets.Close();
    }
  }
```

Рисунок 3.11 – Код події «StartAnalizeBtn_Click»

В рамках задачі ефективного аналізу мережевого вузла було сконцентровано увагу на створенні коду для зберігання відповідної інформації. Спеціалізований кодовий блок було спроектовано таким чином, що він здійснює комплексну обробку інформації, яка є критичною для аналітичних задач щодо вузла в мережі. Цей код не тільки капіталізує отримані дані, але і форматує їх для оптимального зберігання та подальшого аналізу.

Для забезпечення наочності та глибшого технічного розуміння даного аспекту, візуальний відтворення цього коду було включено в проектну документацію. Це відображення можна переглянути на рис. 3.12, який служить

як ілюстративний матеріал для ознайомлення з архітектурними та функціональними особливостями цієї критично важливої частини системи.

```
private void SaveBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    StartAnalizeBtn.Text = "&Аналізувати";
    PacketCountTimer.Stop();
    _AnalizeDataProvider.InsertAnalizeData(AnalizeDataNameTBox.Text);
    int lastAnalizeDataId = _AnalizeDataProvider.GetLastRecords();
    for (int i = 0; i < _AnalizePacketsList.Count; i++) {
      _AnalizePacketsList[i].AnalizeDataId = lastAnalizeDataId;
    }
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було проведено аналіз пакетів та збережено дані під назвою '" +
      AnalizeDataNameTBox.Text + "'", DateTime.Now);
    _AnalizePacketsPrivider.InsertBatchAnalizePackets(_AnalizePacketsList);
    _PocketsValues.Clear();
    _DateTimes.Clear();
    treeView.Nodes.Clear();
    AnalizeDataNameTBox.Text = String.Empty;
  }
}
```

Рисунок 3.12 – Код події «SaveBtn_Click»

На етапі завершення взаємодії користувача з формою системи запускається певна подія, що розроблена для управління процесом закриття з'єднання через сокет. Ця подія контролює завершення всіх активних сесій та зв'язків з мережевим інтерфейсом, на яких базується функціональність системи.

Активація цієї події ініціює ряд операцій, що включає в себе відключення від сокету та звільнення ресурсів, що були використані під час роботи з формою. Це забезпечує не тільки ефективну завершеність сеансу роботи, але і забезпечення цілісності даних та стабільність системи в цілому.

Рис. 3.13 ілюструє конкретні рядки коду, які імплементують логіку цієї важливої операції, надаючи можливість глибшого аналізу та розуміння її функціональних аспектів.

```
1 reference
private void AnalizeForm_FormClosing(object sender, FormClosingEventArgs e) {
  if (_BContinueCapturing) {
    _MainSockets.Close();
  }
}
```

Рисунок 3.13 – Подія для закриття з’єднання із сокетом

У підсумку, цей сегмент надає комплексний аналіз ключових етапів реалізації програмного рішення. Зокрема, розглянуто фундаментальні принципи архітектурної організації, методики взаємодії з базою даних, а також специфічні алгоритми та механізми, що підтримують основну функціональність системи.

3.3. Перевірка ефективності розробленого додатку

Для забезпечення методичного тестування програмного засобу для аналізу мережевих пакетів за протоколом TCP, важливим є створення адекватного тестового середовища. Для цього ми використовували віртуальну машину, базуючись на операційній системі Windows 10, що дозволяє симулювати реальні умови роботи системи в контрольованому середовищі.

В цьому віртуальному контексті, було інтегровано утиліту Nmap, яка відзначається високими здібностями до сканування мереж та ідентифікації активних портів. Для забезпечення максимальної гнучкості та коректності в роботі цієї утиліти, було створено конфігураційний файл "nmap.conf". В цьому файлі були детально описані параметри, необхідні для оптимізації процедур сканування відповідно до специфіки нашого дослідження. Цей файл служить як ключовий елемент управління поведінкою утиліти Nmap в рамках встановленого тестового середовища.

Конфігураційні налаштування, які були внесені в "nmap.conf", відображають ретельно плановану стратегію тестування, спрямовану на вимірювання різних аспектів роботи системи аналізу пакетів. Для наочності та глибшого розуміння цих налаштувань, вони були відображені на рис. 3.14, який служить додатковим візуальним матеріалом для аналізу структури та логіки проведеного тестування.

```
# Вибираємо тільки відкриті порти
openports=true
# Вибраємо тільки TCP порти
sS=true
# Вибрати порти від 1 до 1024
p1-1024=true
# Використовуємо швидке сканування
T4=true
# Зберігати результати сканування у файлі
oN=<output result.txt>
# Вказати цільову IP-адресу
target=<target IP>
```

Рисунок 3.14 – Налаштування утиліти Nmap

Завдяки такому конфігурованому тестовому середовищу, вдалося відтворити умови реального функціонування системи, провести інтенсивну експертизу її якісних показників, а також оцінити її ефективність у виявленні аномалій в мережевому трафіку.

Після завершення етапів конфігурації та калібрування тестового середовища, була запущена фаза активного сканування основної оперативної системи. Для цього використовувалася конкретна команда "nmap 192.168.1.101", де вказана IP-адреса представляла цільовий вузол в мережі для аналізу.

Використовуючи задані параметри, утиліта Nmap успішно виконала комплексну операцію сканування, включаючи аналіз доступних портів та інших ключових характеристик мережевого середовища. Результати цього сканування були систематизовані та представлені у формі виводу, який не тільки деталізує стан мережі, але й надає цінну інформацію про взаємодію з цільовою оперативною системою. Ці результати були зібрані та візуалізовані на рис. 3.15, який служить важливим матеріалом для подальшого аналізу та оцінки здатності розробленої системи до аналізу мережевих пакетів за протоколом TCP.

```
nmap -O 192.168.1.101
nmap -O --osscan-guess 192.168.1.101
nmap -v -O --osscan-guess 192.168.1.101
Starting Nmap 5.00 ( http://nmap.org ) at 2022-11-27 01:29 IST
NSE: Loaded 0 scripts for scanning.
Initiating ARP Ping Scan at 01:29
Scanning 192.168.1.101 [1 port]
Completed ARP Ping Scan at 01:29, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 01:29
Completed Parallel DNS resolution of 1 host. at 01:29, 0.22s elapsed
Initiating SYN Stealth Scan at 01:29
Scanning 192.168.1.101 [1000 ports]
Discovered open port 80/tcp on 192.168.1.101
Discovered open port 22/tcp on 192.168.1.101
Completed SYN Stealth Scan at 01:29, 0.16s elapsed (1000 total ports)
Initiating OS detection (try #1) against 192.168.1.104
Retrying OS detection (try #2) against 192.168.1.101
Retrying OS detection (try #3) against 192.168.1.101
Retrying OS detection (try #4) against 192.168.1.101
Retrying OS detection (try #5) against 192.168.1.101
Host 192.168.1.101 is up (0.00049s latency).
Interesting ports on 192.168.1.101:
Not shown: 998 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
MAC Address: BC:AE:C5:C3:16:93 (Unknown)
Device type: WAP|general purpose|router|printer|broadband router
```

Рисунок 3.15 – Дані IP-адресу «192.168.1.101»

До активації фази сканування головної операційної системи на віртуальній машині за допомогою утиліти Nmap, була ініційована робота спеціалізованої програми, що має назву "Аналізатор мережевого трафіку". Ця програма була конструйована з пріоритетним фокусом на аналізі та ідентифікації аномальних мережевих активностей, зокрема в контексті пакетів TCP протоколу.

З моменту запуску сканування через Nmap, програмний комплекс "Аналізатор мережевого трафіку" активував процедури моніторингу та детального аналізу трафіку в мережі. Відзначено було вищий рівень мережевої активності, що тут же спричинило генерацію системного повідомлення про виявлені аномалії.

Сформована інформація про активність в мережі була зібрана та систематизована в структурованому форматі, що надав можливість зрозуміти характер та динаміку змін у мережевому трафіку. Візуальна репрезентація цих даних та висновки з аналізу представлені на рис. 3.16, який виступає ключовим елементом в оцінці функціональної ефективності розробленого програмного рішення.

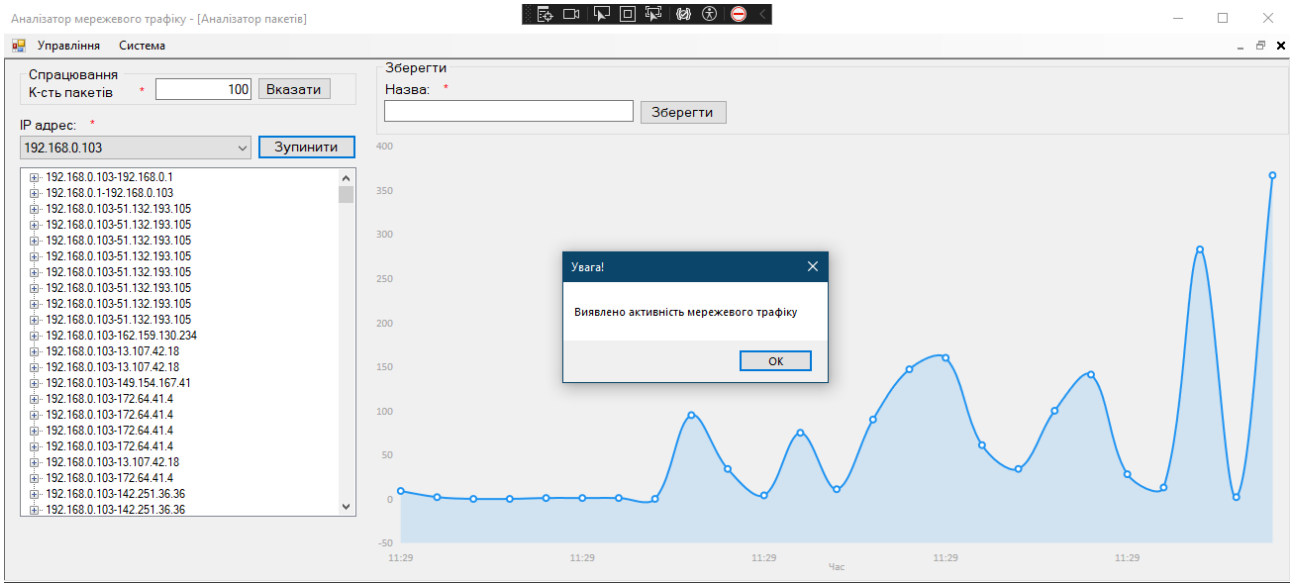

Рисунок 3.16 – Приклад виявлення активності у мережевому трафіку

По завершенню глибокого та мінуціозного експериментального вивчення розробленого програмного комплексу було констатовано, що він демонструє функціональність, відповідну заданим технічним параметрам та вимогам. Зокрема, програма ефективно ідентифікує спроби проведення сканування вразливостей через утиліту Nmap і забезпечує адекватні контрзаходи на такі інциденти.

Експериментальна верифікація підкреслила виправданість застосування методології аналізу мережевого трафіку, що вкладена в архітектуру розробленого програмного рішення. Виявлено, що для якісної оцінки ситуації достатньо аналізу статистичних показників потоку пакетів, без необхідності деталізованого розгляду атрибутів конкретної атаки та механізмів її проведення.

Однією з визначальних переваг прийнятого методологічного підходу є спроможність виявляти атаки, що не були раніше документовані або які зловмисники реалізовують вперше. Це розширює горизонти в сфері мережевої безпеки та оборони від потенційних ризиків.

Така концепція не тільки дозволяє ефективно виявляти та нейтралізувати вже відомі загрози, але також може бути адаптована для проактивного виявлення

нових вразливостей. Такий підхід підвищує загальний рівень захищеності мережевої інфраструктури, тим самим стаючи важливою ланкою в загальній стратегії забезпечення безпеки мережевих систем та їхньої стабільної функціональності.

3.4. Оцінка повноти вирішення поставлених задач і достовірності результатів

Проведене дослідження характеризується високим рівнем методичної підготовки. Використання віртуальної машини на базі операційної системи Windows 10 дозволило емулювати реальні умови функціонування системи в контрольованому середовищі. Інтеграція утиліти Nmap, зокрема з використанням спеціалізованого конфігураційного файлу "nmap.conf", додала необхідної глибини та гнучкості до експериментального плану.

Забезпечення комплексності аналізу підтверджується наявністю кількох етапів дослідження. Спочатку відбувалася конфігурація та калібрування тестового середовища, потім проводилася фаза активного сканування, і, нарешті, аналіз отриманих даних. Це дає підстави вважати, що результати дослідження є комплексними та багатогранними.

Результати сканування та аналізу були представлені в структурованому вигляді, що забезпечує високий рівень їхньої достовірності. Візуальні репрезентації (п. 3.3) служать важливими інструментами для подальшого аналізу, що підкреслює обґрунтованість отриманих висновків.

В ході експерименту було підтверджено, що розроблена система адекватно відповідає заданим технічним параметрам і вимогам, зокрема в контексті ідентифікації аномальних мережевих активностей. Програмний комплекс "Аналізатор мережевого трафіку" показав здатність до виявлення вищого рівня мережевої активності та генерації системних повідомлень про виявлені аномалії.

Однією з ключових переваг системи є її спроможність виявляти атаки, що не були раніше документовані. Це підкреслює високий інноваційний потенціал розробленого рішення і його придатність для адаптації в мінливих умовах кіберзагроз.

На підставі проведеного аналізу можна зробити висновок, що поставлені задачі були вирішені повною мірою, і отримані результати є достовірними. Розроблена система не тільки демонструє високу ефективність в ідентифікації аномальних мережевих активностей, але й має потенціал для подальших інноваційних розробок в області захисту критичної інфраструктури.

3.5. Висновок

Даний розділ присвячений методології та практичній розробці системи захисту об'єктів критичної інформаційної інфраструктури, представляє собою комплексний аналіз та обґрунтування вибору напрямку дослідження. В рамках розділу було проведено зіставлення та аналіз різних методів виявлення вторгнень, включаючи поведінкові методи, методи на основі знань, методи машинного навчання та методи обчислювального інтелекту. Це дало можливість прийняти обґрунтоване рішення щодо методологічних підходів та технологічних інструментів (C# і MS SQL Server) для реалізації проекту.

Розроблена загальна методика проведення досліджень стала фундаментом для імплементації програмного забезпечення. На основі проведеного порівняльного аналізу методів вторгнення було прийнято рішення про розробку власного ПЗ для виявлення аномальної поведінки в мережевому трафіку.

Практична реалізація системи базується на трирівневій архітектурі і включає в себе реалізацію вибраних методів на мові програмування C#. Ефективність розробленого додатку була перевірена шляхом емуляції реальних умов роботи на віртуальній машині з операційною системою Windows 10 і

застосуванням утиліти Nmap для генерації мережевого трафіку. Результати перевірки підтвердили високу ефективність системи в частині виявлення аномальних активностей.

Щодо повноти вирішення поставлених задач, можна стверджувати, що розроблене програмне забезпечення не тільки відповідає всім зазначеним параметрам та вимогам, але й відкриває нові перспективи в області захисту критичної інформаційної інфраструктури. Достовірність результатів підтверджена ретельним експериментальним аналізом та може бути вважатися високою.

РОЗДІЛ 4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Гідросфера, або водяна оболонка Землі, - це її моря й океани, крижані шапки приполярних районів, річки, озера й підземні води. Космонавти кажуть, що коли дивитися на землю з висоти космічної орбіти, то око розрізняє переважно два кольори: білий колір хмар і крижаних полярних шапок і блакитний колір Світового океану, що вкриває 71 % поверхні нашої планети; морська вода – найпоширеніша на Землі речовина.

Вода виконує дуже важливі екологічні функції:

- вода – це головна складова частина всіх живих організмів;
  - за участю води здійснюється численні процеси в екосистемах(наприклад, обмін речовин, тепла);
- вода – один із найважливіших видів мінеральної сировини, основний природний ресурс ,що споживається людством.

Величезну роль відіграє гідросфера у формуванні поверхні Землі, її ландшафтів, у розвитку екзогенних процесів, в перенесенні хімічних речовин, у тому числі й забруднювачів довкілля.

Для багатьох організмів вода – це середовище їхнього життя. Хімічний склад морської води дуже схожий на склад плазми крові людини: містить ті самі хімічні елементи й приблизно в тих самих пропорціях. Це один із доказів того, що предки людини, як і решти ссавців, колись жили в морі.

Якщо виміряти весь земний запас води, то ми одержимо цифру в 1390 мільйонів кубічних кілометрів, більша частина якого доводиться на океани й моря - 96,4%. Близько 1,86% води (у гірських льодовиках 0,2%) містять постійні сніги й льодовики на суші. Близько 1,7% від усього обсягу гідросфери становлять підземні води, вода суші становить близько 0,02% – це ріки, болота, озера й штучні водойми. Невелика кількість води мають живі організми, що проживають в атмосфері й біосфері. 2,64% становить прісна вода. Наша планета в природних умовах має воду в трьох агрегатних станах:

рідкому - вода; твердому - лід; газоподібному – водяний пар.

Це відрізняє воду від інших речовин, які існують на Землі тільки у двох видах: твердому (метали, мінерали) і газоподібному (вуглекислий газ, азот, кисень).

Завдяки воді, що має аномальні хімічні й фізичні властивості, на Землі зародилося життя. Молекули води мають дуже сильне притягання між собою, воно майже в 10 разів сильніше, ніж притягання всіх інших рідин. Саме тому кипіння води відбувається при 100°С, плавлення води при 0°С.

Вода має аномально високу теплоємність, тому для того щоб розплавити лід або розігріти воду до випару потрібно затратити набагато більше енергії, чим якби нагрівали інші рідини. При цьому вода має дуже малу теплопровідність, вона досить повільно нагрівається й приблизно з такою ж швидкістю остигає

Коли вода замерзає, вона розширюється, і її щільність у твердому стані, менше ніж у рідкому. Тому лід легше води - це до речі ще одна прекрасна властивість води, цією властивістю вода відрізняється від більшої частини інших рідин.

Підземні води за своїм хімічним складом дуже різноманітні : від прісних, що використовуються для пиття й водопостачання, до мінералізованих і навіть до ропи із солоністю 600% ; деякі мінералізовані підземні води мають лікувальні властивості.

Головне джерело води для України – річка Дніпро. Крім того, потреби у воді забезпечуються річками Дунай. Дністер, Південний Буг, Тиса, Прут та ін.. Стан води й повноводність цих артерій залежать в основному від стану їхніх приток – малих річок, яких в Україні налічується близько 63 тис. Їхня роль величезна: досить згадати, що 90% населених пунктів нашої країни розташовані саме в долинах малих річок і користуються їхньою водою. Однак стан малих річок України сьогодні викликає велику тривогу. Понад 20 тис. їх уже зникло, пересохло. Це невідворотно веде до деградації великих річок.

Тому проблема їх збереження та оздоровлення – одна з найгостріших для нашої молодої держави.

Підземні води України мають не менше значення для забезпечення водою населення: близько 70% населення сіл і селищ міського типу задовольняють свої потреби в питній воді за рахунок ґрунтових вод чи глибших водоносних горизонтів . Стан підземних вод України в цілому кращий, ніж поверхневого стоку , хоча місцями вони забруднюються стоками промислових підприємств, тваринницьких комплексів тощо. В деяких промислових районах розробка шахт і кар`єрів негативно впливає на якість і запаси підземних вод. У результаті багаторічного відкачування води з цих об`єктів рівень дуже понизився, а з деяких водоносних горизонтів вода зникла зовсім.

Отже , Гідросфера – це саме та оболонка, яка охоплює в своєму складі всі моря, океани, річки, озера. Як з`ясувалося, що вода має теж свої функції для людей, цілющі властивості, тому й людство повинно дбати про водні ресурси. Проблема нестачі водних ресурсів, забруднення водоймищ торкнулося навіть України, стан річок і морів України на даний час не є задовільним, в кращому стані перебувають лише підземні води України, хоча їхній стан і не найкращий.

Споживання прісної води. Всі галузі господарства стосовно водних ресурсів поділяються на споживачів і користувачів. Споживачі забирають воду з джерела водопостачання, використовують її для виготовлення продукції. А потім повертають, але вже в меншій кількості й іншої якості. Користувачі воду не забирають, а використовують її як середовище (водний транспорт, рибальство, спорт тощо) або як джерело енергії (ГЕС). Протей вони можуть змінювати якість води (наприклад водний транспорт забруднює воду).

Промисловість використовує близько 20 % води, споживаної людством. Кількість води, що споживається підприємством, залежить від того, яку

продукцію воно випускає, від системи водопостачання (прямоточна чи оборотна) та від інших причин.

Питоме водоспоживання під час зрошення залежить від виду вирощуваних сільськогосподарських культур, клімату, технічного стану зрошуваних систем і способів поливу.

Більша частина води (20-60%), що використовується для зрошення, безповоротно втрачається (випаровується), певна ї ї кількість повертається назад у водойми у вигляді так званих поворотних вод, сильно забруднених солями. Водопостачання населення(близько 10% усієї споживаної людством води) задовольняє потреби в питній воді й комунально-побутові потреби (робота підприємств побутового обслуговування, поливання вулиць і зелених насаджень, протипожежні заходи тощо). Є поняття питоме водоспоживання, тобто добовий об`єм води, необхідний для задоволення потреб одного жителя міста або села.

Можемо підвести підсумок про те , що основними джерелами забруднення води є підприємства , які забирають близько 20% води на свої потреби та сільське господарство, яке інтенсивно використовує воду для зрощення полів, більшість цієї води все ж таки повертається у вогляді так званих поворотних вод, але вони сильно забруднені і потребують очищення.

Забруднення води. В результаті діяльності людей гідросфера змінюється : кількісно (зменшення кількості води, придатної для використання) та якісно (забруднення). Серед забруднень розрізняють фізичне, хімічне, біологічне й теплове.

Фізичне забруднення води відбувається внаслідок: накопичення в ній нерозчинних домішок – піску, глини, мулу в результаті змивання дощовими водами з розораних ділянок (полів); надходження суспензій з підприємств гірничорудної промисловості; потрапляння пилу, що переноситься вітром у суху погоду тощо. Тверді частинки знижують прозорість води, пригнічують розвиток водних рослин, забивають зябра риб та інших водяних тварин,

погіршують смакові якості води, а іноді роблять її взагалі непридатною для споживання

Хімічне забруднення відбувається через надходження у водойми зі стічними водами різних шкідливих домішок, неорганічного(нафта й нафтопродукти, мийні засоби, пестициди тощо) складу. Шкідлива дія токсичних речовин , що потрапляють у водойми, посилюється за рахунок так званого кумулятивного ефекту (прогресуюче збільшення вмісту шкідливих сполук у кожній наступній ланці трофічного ланцюга). Так, у фітопланктоні концентрація шкідливої сполуки часто виявляється в декілька разів вищою, ніж у воді, у зоопланктоні (личинки, дрібні рачки тощо) – в десятки разів вищою, ніж у фітопланктоні, в рибі, яка харчується зоопланктоном, – ще в десятки разів вищою. А в організмі хижих риб (таких, як щука чи судак) концентрація отрути збільшується ще в десять разів і, отже, буде в десять тисяч разів вищою, ніж у воді.

Особливої шкоди водоймам завдають нафта й нафтопродукти, які утворюють на поверхні води плівку, що перешкоджає газообмінові між водою та атмосферою й знижує вміст у воді кисню. В результаті розливу 1 т нафти плівкою покривається 12 км кВ. води. Згустки мазуту, осідаючи на дно, вбивають донні мікроорганізми, які беруть участь у процесі самоочищення води. Внаслідок гниття осадів, забруднених органічними речовинами виділяються шкідливі сполуки, зокрема сірководень, що отруюють усю воду в річці чи озері.

До основних забруднювачів води належать хімічні, нафтопереробні й целюлозно-паперові комбінати, великі тваринницькі комплекси, гірничорудна промисловість. Серед забруднювачів води особливе місце посідають синтетичні мийні засоби. Ці речовини надзвичайно стійкі, зберігаються у воді роками.

Забруднення води речовинами, що містять фосфор, сприяє бурхливому розмноженню синьо-зелених водоростей і «цвітінню» водойм , яке супроводжується різким зниженням у воді вмісту кисню, «заморами» риби,

загибеллю інших водяних тварин. Під час «цвітіння» Каховського та інших «рукотворних» морів на Дніпрі стоїть сморід, а хвилі викидають на берег трупи риби, що задихнулася.

Біологічне забруднення водойм полягає в надходженні до них зі стічними водами різних мікроорганізмів (бактерій. вірусів) , спор грибів, яєць гельмінтів і т.д. , багато з яких є хвороботворними для людей , тварин і рослин. Серед біологічних забруднювачів перше місце посідають комунально-побутові стоки (особливо, якщо вони не очищені або очищені недостатньо) , а також стоки цукрових заводів , м`ясо комбінатів,підприємств з обробки шкір, деревообробних комбінатів. Особливо небезпечне біологічне забруднення водойм у місцях масового відпочинку людей(курортні зони узбереж морів). Через поганий стан каналізаційних систем та очисних споруд останніми роками нерідко закривалися пляжі в Одесі, Маріуполі та інших містах на узбережжях Чорного та Азовського морів, оскільки в морській воді було виявлено збудників таких небезпечних захворювань, як холера, дизентерія, вірусний гепатит та ін.

Теплове забруднення води відбувається внаслідок спускання у водойми підігрітих вод від ТЕС, АЕС та інших енергетичних об`єктів. Тепла вода змінює термічний і біологічний режим водойм і шкідливо впливає на їхніх мешканців. Як показали дослідження гідробіологів, вода нагріта до температури 26-30 градусів, діє на риб та інших мешканців водойм пригнічуючи. А якщо температура води піднімається до 36 градусів, то риба гине. Найбільшу кількість теплої води скидають у водойми атомні електростанції.

До основних джерел забруднення водоймищ відносять:

– атмосферні опади, що містять забруднюючі речовини промислового походження, які вимиваються з атмосфери;

– міські стічні води (побутові, каналізаційні стоки, що містять шкідливі для здоров`я синтетичні миючі засоби та ін.);

– промислові стічні води;

– сільськогосподарські стічні води (відходи тваринницьких комплексів, змив з полів пестицидів дощами і весняними талими водами та ін.).

Найбільш значущу частку забруднення водоймищ становлять промислові стічні води, половина обсягу яких скидається у водоймища без очищення, а велика частина другої половини – в недостатньо очищеному вигляді. Тому майже всі річки забруднені нафтопродуктами, важкими металами, органічними і мінеральними сполуками. Сільськогосподарські стічні води несуть у річки та озера величезну кількість добрив і пестицидів. Скидання стічних вод у водоймища супроводжується накопиченням забруднюючих речовин у донних осадках у великих концентраціях, що може призводити до різкого підвищення рівня забруднення, пов`язаного з утворенням нових хімічних сполук.

Морські води також піддаються забрудненню. З ріками і стоками прибережних промислових і сільськогосподарських підприємств щорічно виносяться в моря мільйони тонн хімічних відходів , а з комунальними стоками і органічних з`єднань. Через аварії танкерів і нафтовидобувних установок в океан попадає по різним джерелам не менше 5 млн. тонн нафти в рік, що викликає загибель багатьох водних тварин, морських птахів. Хвилювання викликають захоронення ядерних відходів на дні морів, кораблі, які потонули з ядерними реакторами і ядерною зброєю.

Отже, можна відмітити те, що найбільш небезпечними для водоймищ є фізичне, хімічне, біологічне і теплове забруднення. Вони призводять не тільки до погіршення якості води , але й до того, що починають з`являтися трупи риб, що в наш час не є несподіванкою, окрім цього забруднення водоймищ згубно впливає і на саму людину. Оскільки може призводити до таких страшних хвороб як холера, дизентерія, вірусний гепатит та ін. Забруднюються не тільки річки , а й моря, вони теж часто забруднюються нафтою та нафтопродуктами, що має найбільш згубний вплив. Поширилося

так зване «цвітіння води», яке супроводжується різким зниженням у воді вмісту кисню, «заморами» риби, загибеллю інших водяних тварин.

ВИСНОВКИ

У ході виконання наукової роботи було здійснено ряд ключових дій та досягнуто важливих результатів. Проведено детальний огляд наукової літератури, який дозволив ідентифікувати ряд невирішених проблем у сфері кібербезпеки, таких як вразливість систем, людський фактор, комплексність систем, відсутність стандартів, політичні та правові обмеження, економічні фактори та активні кіберзагрози.

У контексті правового регулювання кібербезпеки було проведено глибокий аналіз актуальних законів України, зокрема "Про основні засади забезпечення кібербезпеки України" та "Про Національну програму адаптації законодавства України до законодавства Європейського Союзу". Додатково було вивчено Директиви ЄС 2016/1148 та стандарт ISO/IEC 27001, що мають великий вплив на формування стратегій та практичних методів захисту інформації в об'єктах критичної інфраструктури.

В рамках аналізу існуючих систем захисту було здійснено систематичний огляд лідерів ринку в сфері антивірусних рішень, таких як McAfee Total Protection for Data Loss Prevention, Cisco AMP for Endpoints та Symantec Critical System Protection. Це дало можливість оцінити їх ефективність, адаптивність та надійність в контексті захисту об'єктів критичної інфраструктури.

Крім того, було вивчено системи виявлення вторгнень, такі як Suricata, Cisco Firepower, IBM QRadar, та Palo Alto Networks Threat Prevention. Оцінка цих систем базувалася на ряді параметрів, зокрема швидкість виявлення аномалій, точність аналізу та інтеграція з іншими системами захисту.

З метою виявлення оптимальних методів захисту було проведено комплексний аналіз основних методологій захисту, включаючи криптографічні методи, фізичні методи захисту, мережеві та програмні методи. Це дозволило визначити принципи комплексного підходу до забезпечення безпеки, які враховують специфіку та потреби критичної інформаційної інфраструктури.

На етапі методології та практичної розробки основний акцент було зроблено на обґрунтуванні вибору конкретного напрямку дослідження. Це включало в себе аналіз потреб критичної інформаційної інфраструктури, оцінку зовнішніх та внутрішніх загроз, а також вивчення можливостей сучасних технологій у контексті кібербезпеки.

В рамках розробки системи захисту було проведено детальний аналіз доступних інструментів розробки, на основі якого було вирішено зосередитися на використанні мови програмування C# та системи управління базами даних MS SQL Server. Вибір цих технологій був обґрунтований їхньою високою продуктивністю, гнучкістю та здатністю забезпечити надійний та ефективний захист інформації.

Для систематизації дослідницького процесу була розроблена загальна методика проведення досліджень. Ця методика описує ключові етапи виявлення аномальної поведінки в мережевому трафіку, включаючи збір даних, їхній аналіз та інтерпретацію результатів.

Практична реалізація системи базувалася на трирівневій архітектурі та включала в себе застосування методів машинного навчання та обчислювального інтелекту. Ефективність розробленого додатку була підтверджена за допомогою віртуальної машини з ОС Windows 10, на якій було встановлено утиліту Nmap для створення активності мережевого трафіку. Розроблене ПЗ виявило аномальну активність та вивело відповідне повідомлення.

Таким чином, робота в повній мірі вирішила поставлені задачі, що дозволяє розглядати її як цілісний та виважений внесок у розвиток сфери кібербезпеки критичної інформаційної інфраструктури. Результати роботи можуть бути використані як в науковому, так і в практичному контексті для подальшого вдосконалення систем захисту і забезпечення кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке об'єкти критичної інфраструктури. URL: https://smarttender.biz/terminy/view/ob-yekti-kritichnoyi-infrastrukturi/ (дата звернення 10.11.2023).

2. Закон України "Про основні засади забезпечення кібербезпеки України" : від 17.08.2022 , № 45, ст.403 URL: https://zakon.rada.gov.ua/laws/show/2163-19#top (дата звернення 25.10.2023).

3. Закон України "Про Національну програму адаптації законодавства України до законодавства Європейського Союзу": від 04.11.2018, 2004, № 29, ст.367. URL: https://zakon.rada.gov.ua/laws/show/1629-15#Text (дата звернення 10.11.2023).

4. Директива ЄС 2016/1148 : Міжнародний документ від 06.07.2016 № 2016/1148. URL: https://zakon.rada.gov.ua/laws/show/984_013-16#Text (дата звернення 25.10.2023).

5. Second Post-Implementation Review of the Network and Information Systems Regulations/ URL: https://www.gov.uk/government/publications/second-post-implementation-review-of-the-network-and-information-systems-regulations-2018 (дата звернення 25.10.2023).

6. ISO/IEC 27001 :2013// ISO/IEC. – 2013. – URL: https://www.slideshare.net/DOMDepartmentofMarke/iso-27001pdf-253872904 (дата звернення 25.10.2023).

7. Державна служба експортного контролю України (ДСЕКУ) URL :https://www.dsecu.gov.ua/ (дата звернення 25.10.2023).

8. Закон України “Служба Безпеки України (СБУ) “ № 2469-VIII від 21.06.2018.

9. McAfee Total Protection for Data Loss Prevention : веб-сайт. URL: https://www.websecurityworks.com/Data-Loss-Prevention.asp (дата звернення 25.10.2023).

10. Cisco AMP for Endpoints: веб-сайт. URL: https://www.cisco.com/site/us/en/products/security/endpoint-security/secure-endpoint/index.html (дата звернення 25.10.2023).

11. Symantec Critical System Protection: веб-сайт. URL: https://techdocs.broadcom.com/us/en/symantec-security-software/endpoint-security-and-management/Symantec-Critical-System-Protection/8-0-0-MP1.html (дата звернення 25.10.2023).

12. Intrusion Detection Systems, IDS: веб-сайт. URL: https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/ (дата звернення 25.10.2023).

13. Suricata : веб-сайт. URL: https://suricata.io/ (дата звернення 25.10.2023).

14. What is Suricata - Open Information Security Foundation – OISF : веб-сайт. URL: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/What_is_Suricata (дата звернення 25.10.2023).

15. Cisco Firepower: веб-сайт. URL: https://www.cisco.com/c/en/us/td/docs/security/firepower/630/configuration/guide/fpmc-config-guide-v63/introduction_to_the_cisco_firepower_system.html (дата звернення 25.10.2023).

16. What is Cisco FirePOWER? The introduction: веб-сайт. URL: https://www.grandmetric.com/knowledge-base/design_and_configure/cisco-firepower-introduction-licensing-architecture-fmc/ (дата звернення 25.10.2023).

17. QRadar SIEM Ratings Overview. URL: https://www.gartner.com/reviews/market/security-information-event-management/vendor/ibm/product/qradar-siem (дата звернення 10.11.2023).

18. IBM QRadar: веб-сайт. URL: https://www.ibm.com/qradar (дата звернення 25.10.2023).

19. IBM QRadar Security Intelligence Platform - IC-Systems : веб-сайт. URL: https://www.techtarget.com/searchsecurity/feature/IBM-Security-QRadar-SIEM-product-overview (дата звернення 25.10.2023).

20. Palo Alto Networks Threat Prevention : веб-сайт. URL: https://docs.paloaltonetworks.com/advanced-threat-prevention (дата звернення 25.10.2023).

21. DEVELOPING THE CRITICAL INFRASTRUCTURE PROTECTION SYSTEM IN UKRAINE. URL: https://niss.gov.ua/sites/default/files/2017-11/niss_Engl_findruk-0e9af.pdf (дата звернення 10.11.2023).

22. The Appropriate Use of Customer Data in Financial Services. URL: https://www3.weforum.org/docs/WP_Roadmap_Appropriate_Use_Customer_Data.pdf (дата звернення 10.11.2023).

23. California Consumer Privacy Act: win-win or GDPR parody: веб-сайт. URL: https://legalitgroup.com/en/california-consumer-privacy-act-win-win-or-gdpr-parody/?gclid=Cj0KCQiAjMKqBhCgARIsAPDgWlyKVM9iqkcfXnNcmq0EZetSxkOOzXyVLnqyHteLXOr0aW7iufzg4MkaAljhEALw_wcB (дата звернення 10.11.2023).

24. Succeed at Telecom Analytics - Data and Analytics Solutions: веб-сайт. URL: https://uniathena.com/lms/student-dashboard/course/123/diploma-in-data-analytics?utm_source=conversion&utm_medium=google&utm_campaign=GO-SEL-DPL-161023-CNV-SC-123-DA&utm_page=Login&gad=1&gclid=Cj0KCQiAjMKqBhCgARIsAPDgWlzmfumuTySzP1wGeAeG_LT5ZcwnH8ByvNrE5RBn-R4-_FyLhnrbmaMaAh2bEALw_wcB (дата звернення 10.11.2023).

25. A Comparative Study of Machine Learning Algorithms for Anomaly Detection in Industrial Environments: веб-сайт. URL: https://www.researchgate.net/publication/372074373_A_Comparative_Study_of_Machine_Learning_Algorithms_for_Anomaly_Detection_in_Industrial_Environments_Performance_and_Environmental_Impact (дата звернення 10.11.2023).

26. What is threat prevention Palo Alto Networks? : веб-сайт. URL: https://www.trustradius.com/products/palo-alto-networks-advanced-threat-prevention/reviews?qs=pros-and-cons (дата звернення 25.10.2023).

27. SSL/TLS : веб-сайт. URL: https://www.f5.com/glossary/ssl-tls-encryption. (дата звернення 25.10.2023).

28. Man-in-the-Middle : веб-сайт. URL: https://csrc.nist.gov/glossary/term/man_in_the_middle_attack (дата звернення 25.10.2023).

29. DDoS-атаки: веб-сайт. URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/distributed-denial-of-service/ (дата звернення 25.10.2023).

30. The VPN that just works : веб-сайт. https://www.expressvpn.com/ (дата звернення 25.10.2023).

31. А. Бандура. Соціальне навчання: теорія і практика. Київ: Видавництво "Психологія", 2019. 432 с.

32. Б. Скіннер. Поведінка організмів: експериментальна аналіз. Київ: Видавництво "Наукова думка", 2018. 296 с.

33. С. Рассел, П. Норвіг. Штучний інтелект: сучасний підхід. 3-є видання. Сан-Франциско: Prentice Hall, 2020. 1132 с.

34. Д. Ленат. Building Large Knowledge-Based Systems. San Francisco: Addison-Wesley, 2019. 704 с.

35. Я. Гудфелло, Й. Бенджіо, А. Курвіль. Глибоке навчання. Кембрідж: MIT Press, 2020. 800 с.

36. Д. Гарет, В. Дані, Х. Тревор, Р. Тібширані Введення в статистичне навчання. Нью-Йорк: Springer, 2019. 426 с.

37. Ф. Клаус. Основи м'якого обчислювання: нечіткі системи та генетичні алгоритми. Берлін: Springer, 2018. 372 с.

38. Х. Фенгцен. Нейромережі: теорія та практика. Лондон: CRC Press, 2020. 528 с.

39. Andrew Troelsen. Pro C# 7: With .NET and .NET Core. Berkeley: Apress, 2018. 1376 с.

40. Albahari Joseph. C# 7.0 in a Nutshell: The Definitive Reference. Seattle: O'Reilly Media, 2018. 1088 с.

41. Delameter Paul, Miller Matthew. SQL Server 2019 Administration Inside Out. Redmond: Microsoft Press, 2020. 992 c.

42. Ben-Han Yitzhak. T-SQL Fundamentals. 3rd edition. Redmond: Microsoft Press, 2016. 512 c.

ДОДАТКИ

Додаток А. Скрипти створення бази даних

```
USE [master]
GO
/****** Object:  Database [APockets]    Script Date: 11.08.2023 13:47:15 ******/
CREATE DATABASE [APockets]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'APockets', FILENAME = N'C:\Program Files (x86)\Microsoft SQL
Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\APockets.mdf' , SIZE = 5120KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'APockets_log', FILENAME = N'C:\Program Files (x86)\Microsoft SQL
Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\APockets_log.ldf' , SIZE = 1024KB ,
MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
ALTER DATABASE [APockets] SET COMPATIBILITY_LEVEL = 120
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [APockets].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [APockets] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [APockets] SET ANSI_NULLS OFF
GO
ALTER DATABASE [APockets] SET ANSI_PADDING OFF
GO
ALTER DATABASE [APockets] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [APockets] SET ARITHABORT OFF
GO
ALTER DATABASE [APockets] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [APockets] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [APockets] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [APockets] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [APockets] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [APockets] SET CONCAT_NULL_YIELDS_NULL OFF
GO
```

```
ALTER DATABASE [APockets] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [APockets] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [APockets] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [APockets] SET  DISABLE_BROKER
GO
ALTER DATABASE [APockets] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [APockets] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [APockets] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [APockets] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [APockets] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [APockets] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [APockets] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [APockets] SET RECOVERY SIMPLE
GO
ALTER DATABASE [APockets] SET  MULTI_USER
GO
ALTER DATABASE [APockets] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [APockets] SET DB_CHAINING OFF
GO
ALTER DATABASE [APockets] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [APockets] SET TARGET_RECOVERY_TIME = 0 SECONDS
GO
ALTER DATABASE [APockets] SET DELAYED_DURABILITY = DISABLED
GO
USE [APockets]
GO
/****** Object:  Table [dbo].[AnalizeData]    Script Date: 11.08.2023 13:47:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AnalizeData](
	[AnalizeDataId] [int] IDENTITY(1,1) NOT NULL,
	[AnalizeDataName] [nvarchar](200) NULL,
PRIMARY KEY CLUSTERED
(
	[AnalizeDataId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO
/****** Object:  Table [dbo].[AnalizePackets]    Script Date: 11.08.2023 13:47:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AnalizePackets](
	[AnalizePacketsId] [int] IDENTITY(1,1) NOT NULL,
	[PacketsCount] [int] NULL,
	[PacketsDate] [datetime] NULL,
	[AnalizeDataId] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[AnalizePacketsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 11.08.2023 13:47:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Logs](
	[LogsId] [int] IDENTITY(1,1) NOT NULL,
	[UsersId] [int] NULL,
	[EventNameShow] [nvarchar](max) NULL,
	[EventDate] [datetime] NULL,
PRIMARY KEY CLUSTERED
(
	[LogsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Users]    Script Date: 11.08.2023 13:47:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
	[UsersId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](50) NULL,
	[LastName] [nvarchar](50) NULL,
	[UsersName] [nvarchar](50) NULL,
	[UsersPassword] [nvarchar](max) NULL,
	[RoleId] [int] NULL,
	[Email] [nvarchar](50) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[UsersId] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [APockets] SET  READ_WRITE
GO
```

Додаток Б. Лістинги розробленого ПЗ

Лістинг 1. Код класу «TCPHeader»

```
using System.Net;
using System.Text;
using System;
using System.IO;
using System.Windows.Forms;

namespace DetectionApp {
  public class TCPHeader {
    //TCP header fields
    private ushort usSourcePort;              //Шістнадцять біт для номера порту джерела
    private ushort usDestinationPort;         //Шістнадцять біт для номера порту призначення
    private uint uiSequenceNumber = 555;          //Тридцять два біти для порядкового номера
    private uint uiAcknowledgementNumber = 555;   //Тридцять два біти для номера
підтвердження
    private ushort usDataOffsetAndFlags = 555;      //Шістнадцять біт для прапорів і зміщення
даних
    private ushort usWindow = 555;                  //Шістнадцять біт для розміру вікна
    private short sChecksum = 555;                 //Шістнадцять біт для контрольної суми
                                      //(контрольна сума може бути від’ємною, тому її можна
вважати короткою)
    private ushort usUrgentPointer;           //Шістнадцять бітів для термінового покажчика
                                    //Кінцеві поля заголовка TCP

    private byte byHeaderLength;            //Довжина заголовка
    private ushort usMessageLength;           //Довжина даних, що передаються
    private byte[] byTCPData = new byte[4096]; //Дані, що передаються пакетом TCP

    public TCPHeader(byte[] byBuffer, int nReceived) {
      try {
        MemoryStream memoryStream = new MemoryStream(byBuffer, 0, nReceived);
        BinaryReader binaryReader = new BinaryReader(memoryStream);

        //Перші шістнадцять бітів містять вихідний порт
        usSourcePort = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Наступні шістнадцять містять порт призначення
        usDestinationPort = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Наступні тридцять два мають порядковий номер
        uiSequenceNumber = (uint)IPAddress.NetworkToHostOrder(binaryReader.ReadInt32());

        //Наступні тридцяти мають два номери підтвердження
        uiAcknowledgementNumber =
(uint)IPAddress.NetworkToHostOrder(binaryReader.ReadInt32());

        //Наступні шістнадцять бітів містять прапори та зміщення даних
        usDataOffsetAndFlags = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());
```

```
        //Наступні шістнадцять містять розмір вікна
        usWindow = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //У наступних шістнадцяти ми маємо контрольну суму
        sChecksum = (short)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Наступні шістнадцять містять терміновий покажчик
        usUrgentPointer = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Зміщення даних вказує, де починаються дані, тому використовуючи його,
        //ми обчислюємо довжину заголовка
        byHeaderLength = (byte)(usDataOffsetAndFlags >> 12);
        byHeaderLength *= 4;

        //Довжина повідомлення = Загальна довжина TCP-пакета - Довжина заголовка
        usMessageLength = (ushort)(nReceived - byHeaderLength);

        //Скопіюйте дані TCP у буфер даних
        Array.Copy(byBuffer, byHeaderLength, byTCPData, 0, nReceived - byHeaderLength);
      } catch (Exception ex) {
        MessageBox.Show(ex.Message, "MJsniff TCP" + (nReceived), MessageBoxButtons.OK,
MessageBoxIcon.Error);
      }
    }

    public string SourcePort {
      get {
        return usSourcePort.ToString();
      }
    }

    public string DestinationPort {
      get {
        return usDestinationPort.ToString();
      }
    }

    public string SequenceNumber {
      get {
        return uiSequenceNumber.ToString();
      }
    }

    public string AcknowledgementNumber {
      get {
        //Якщо встановлено прапорець ACK, лише ми маємо дійсне значення в полі
підтвердження,
        //тому перевіряємо його, перш ніж повертати що-небудь
        if ((usDataOffsetAndFlags & 0x10) != 0) {
          return uiAcknowledgementNumber.ToString();
        } else
          return "";
```

```
  }
}

public string HeaderLength {
  get {
    return byHeaderLength.ToString();
  }
}

public string WindowSize {
  get {
    return usWindow.ToString();
  }
}

public string UrgentPointer {
  get {
    //If the URG flag is set then only we have a valid value in
    //the urgent pointer field, so check for it beore returning
    //anything
    if ((usDataOffsetAndFlags & 0x20) != 0) {
      return usUrgentPointer.ToString();
    } else
      return "";
  }
}

public string Flags {
  get {
    //Останні шість бітів зміщення даних і прапорів містять контрольні біти

    //Спочатку витягуємо прапорці
    int nFlags = usDataOffsetAndFlags & 0x3F;

    string strFlags = string.Format("0x{0:x2} (", nFlags);

    //Тепер ми починаємо перевіряти, встановлені окремі біти чи ні
    if ((nFlags & 0x01) != 0) {
      strFlags += "FIN, ";
    }
    if ((nFlags & 0x02) != 0) {
      strFlags += "SYN, ";
    }
    if ((nFlags & 0x04) != 0) {
      strFlags += "RST, ";
    }
    if ((nFlags & 0x08) != 0) {
      strFlags += "PSH, ";
    }
    if ((nFlags & 0x10) != 0) {
      strFlags += "ACK, ";
    }
```

```
        if ((nFlags & 0x20) != 0) {
          strFlags += "URG";
        }
        strFlags += ")";

        if (strFlags.Contains("()")) {
          strFlags = strFlags.Remove(strFlags.Length - 3);
        } else if (strFlags.Contains(", )")) {
          strFlags = strFlags.Remove(strFlags.Length - 3, 2);
        }

        return strFlags;
      }
    }

    public string Checksum {
      get {
        //Повертає контрольну суму в шістнадцятковому форматі
        return string.Format("0x{0:x2}", sChecksum);
      }
    }

    public byte[] Data {
      get {
        return byTCPData;
      }
    }

    public ushort MessageLength {
      get {
        return usMessageLength;
      }
    }
  }
}
```

Лістинг 2. Код класу «IPHeader»

```
using System.Net;
using System.Text;
using System;
using System.IO;
using System.Windows.Forms;
using System.Net;

namespace DetectionApp {
  public class IPHeader {
    //Поля заголовка IP
    private byte byVersionAndHeaderLength;   //Вісім бітів для версії та довжини заголовка
    private byte byDifferentiatedServices;    //Вісім біт для диференційованих послуг (TOS)
    private ushort usTotalLength;            //Шістнадцять біт для загальної довжини дейтаграми
(заголовок + повідомлення)
    private ushort usIdentification;          //Шістнадцять біт для ідентифікації
    private ushort usFlagsAndOffset;          //Вісім біт для прапорів і зміщення фрагментації
```

```
    private byte byTTL;                  //Вісім біт для TTL (Time To Live)
    private byte byProtocol;              //Вісім біт для основного протоколу
    private short sChecksum;              //Шістнадцять бітів, що містять контрольну суму
заголовка (контрольна сума може бути від’ємною, тому розглядається як коротка)
    private uint uiSourceIPAddress;         //Тридцятидвохбітна IP-адреса джерела
    private uint uiDestinationIPAddress;     //Тридцять два бітові поля IP-адреси кінцевого IP-
заголовка


    private byte byHeaderLength;            //Довжина заголовка
    private byte[] byIPData = new byte[4096];  //Дані, що передаються дейтаграмою


    public IPHeader(byte[] byBuffer, int nReceived) {

      try {
        //Створити MemoryStream з отриманих байтів
        MemoryStream memoryStream = new MemoryStream(byBuffer, 0, nReceived);
        //Далі ми створюємо BinaryReader з MemoryStream
        BinaryReader binaryReader = new BinaryReader(memoryStream);

        //Перші вісім бітів IP-заголовка містять версію та довжину заголовка, тому ми їх
читаємо
        byVersionAndHeaderLength = binaryReader.ReadByte();

        //Наступні вісім бітів містять диференційовані послуги
        byDifferentiatedServices = binaryReader.ReadByte();

        //Наступні вісім бітів містять загальну довжину дейтаграми
        usTotalLength = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Наступні шістнадцять мають ідентифікаційні байти
        usIdentification = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Наступні шістнадцять бітів містять прапори та зміщення фрагментації
        usFlagsAndOffset = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Наступні вісім бітів мають значення TTL
        byTTL = binaryReader.ReadByte();

        //Наступні вісім представляють протокол, інкапсульований у датаграмі
        byProtocol = binaryReader.ReadByte();

        //Наступні шістнадцять бітів містять контрольну суму заголовка
        sChecksum = IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

        //Наступні тридцять два біти містять IP-адресу джерела
        uiSourceIPAddress = (uint)(binaryReader.ReadInt32());

        //Наступні тридцять два містять IP-адресу призначення
        uiDestinationIPAddress = (uint)(binaryReader.ReadInt32());
```

```
    //Тепер обчислюємо довжину заголовка

    byHeaderLength = byVersionAndHeaderLength;
    //Останні чотири біти поля версії та довжини заголовка містять довжину заголовка,
    //ми виконуємо кілька простих бінарних операцій airthmatic, щоб витягти їх
    byHeaderLength <<= 4;
    byHeaderLength >>= 4;
    //Помножте на чотири, щоб отримати точну довжину заголовка
    byHeaderLength *= 4;

    //Скопіювати дані, які передає датаграма, в інший масив так, щоб
    //відповідно до протоколу, який передається в дейтаграмі IP
    Array.Copy(byBuffer,
               byHeaderLength,  //почати копіювання з кінця заголовка
               byIPData, 0,
               usTotalLength - byHeaderLength);
  } catch (Exception ex) {
    MessageBox.Show(ex.Message, "Sniffer", MessageBoxButtons.OK,
       MessageBoxIcon.Error);
  }
}

public string Version {
  get {
    //Розрахувати версію IP

    //Чотири біти заголовка IP містять версію IP
    if ((byVersionAndHeaderLength >> 4) == 4) {
      return "IP v4";
    } else if ((byVersionAndHeaderLength >> 4) == 6) {
      return "IP v6";
    } else {
      return "Unknown";
    }
  }
}

public string HeaderLength {
  get {
    return byHeaderLength.ToString();
  }
}

public ushort MessageLength {
  get {
    //MessageLength = Загальна довжина дейтаграми - Довжина заголовка
    return (ushort)(usTotalLength - byHeaderLength);
  }
}

public string DifferentiatedServices {
  get {
```

```
    //Повертає диференційовані служби в шістнадцятковому форматі
    return string.Format("0x{0:x2} ({1})", byDifferentiatedServices,
        byDifferentiatedServices);
  }
}

public string Flags {
  get {
    //Перші три біти прапорів і поле фрагментації
    //представляють прапорці (які вказують, чи дані  фрагментований чи ні)
    int nFlags = usFlagsAndOffset >> 13;
    if (nFlags == 2) {
      return "Don't fragment";
    } else if (nFlags == 1) {
      return "More fragments to come";
    } else {
      return nFlags.ToString();
    }
  }
}

public string FragmentationOffset {
  get {
    //Останні тринадцять бітів прапорів і поле фрагментації
    //містить зміщення фрагментації
    int nOffset = usFlagsAndOffset << 3;
    nOffset >>= 3;

    return nOffset.ToString();
  }
}

public string TTL {
  get {
    return byTTL.ToString();
  }
}

public Protocol ProtocolType {
  get {
    //Поле протоколу представляє протокол у частині даних дейтаграми
    if (byProtocol == 6)        //Значення шість представляє протокол TCP
    {
      return Protocol.TCP;
    } else if (byProtocol == 17)  //Сімнадцять для UDP
      {
      return Protocol.UDP;
    } else {
      return Protocol.Unknown;
    }
  }
}
```

```
    public string Checksum {
      get {
        //Повертає контрольну суму в шістнадцятковому форматі
        return string.Format("0x{0:x2}", sChecksum);
      }
    }

    public IPAddress SourceAddress {
      get {
        return new IPAddress(uiSourceIPAddress);
      }
    }

    public IPAddress DestinationAddress {
      get {
        return new IPAddress(uiDestinationIPAddress);
      }
    }

    public string TotalLength {
      get {
        return usTotalLength.ToString();
      }
    }

    public string Identification {
      get {
        return usIdentification.ToString();
      }
    }

    public byte[] Data {
      get {
        return byIPData;
      }
    }

  }
}
```

Лістинг 3. Код класу «UDPHeader»

```
using System.Net;
using System.Text;
using System;
using System.IO;
using System.Windows.Forms;

namespace DetectionApp {
  public class UDPHeader {
    //Поля заголовка UDP
    private ushort usSourcePort;          //Шістнадцять біт для номера порту джерела
```

```
  private ushort usDestinationPort;       //Шістнадцять біт для номера порту призначення
  private ushort usLength;               //Довжина заголовка UDP
  private short sChecksum;               //Шістнадцять біт для контрольної суми (контрольна сума
може бути від’ємною, тому розглядається як коротка)
                                 //Кінцеві поля заголовка UDP

  private byte[] byUDPData = new byte[4096];  //Дані, що передаються пакетом UDP

  public UDPHeader(byte[] byBuffer, int nReceived) {
    MemoryStream memoryStreams = new MemoryStream(byBuffer, 0, nReceived);
    BinaryReader binaryReaders = new BinaryReader(memoryStreams);

    //Перші шістнадцять бітів містять вихідний порт
    usSourcePort = (ushort)IPAddress.NetworkToHostOrder(binaryReaders.ReadInt16());

    //Наступні шістнадцять бітів містять порт призначення
    usDestinationPort = (ushort)IPAddress.NetworkToHostOrder(binaryReaders.ReadInt16());

    //Наступні шістнадцять бітів містять довжину UDP-пакета
    usLength = (ushort)IPAddress.NetworkToHostOrder(binaryReaders.ReadInt16());

    //Наступні шістнадцять бітів містять контрольну суму
    sChecksum = IPAddress.NetworkToHostOrder(binaryReaders.ReadInt16());

    //Скопіюйте дані, які передає UDP-пакет, у буфер даних
    Array.Copy(byBuffer,
            8,             //Заголовок UDP має 8 байт, тому ми починаємо копіювати після нього
            byUDPData,
            0,
            nReceived - 8);
  }

  public string SourcePort {
    get {
      return usSourcePort.ToString();
    }
  }

  public string DestinationPort {
    get {
      return usDestinationPort.ToString();
    }
  }

  public string Length {
    get {
      return usLength.ToString();
    }
  }

  public string Checksum {
    get {
```

```
      //Повертає контрольну суму в шістнадцятковому форматі
      return string.Format("0x{0:x2}", sChecksum);
    }
   }

   public byte[] Data {
    get {
     return byUDPData;
    }
   }
  }
}
```

Лістинг 4. Код класу «AnalizeDataProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DetectionApp.AppCode;

namespace DetectionApp.Providers {
  class AnalizeDataProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertAnalizeData(string AnalizeDataName) {
      string SqlString = "INSERT INTO AnalizeData (AnalizeDataName)
Values(@AnalizeDataName)";

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("@AnalizeDataName", AnalizeDataName);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<AnalizeData> GetAllAnalizeData() {
      int i = 0;
      string SqlString = "SELECT * FROM AnalizeData";

      List<AnalizeData> listAnalizeData = new List<AnalizeData>();
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
```

```
            AnalizeData oneAnalizeData = new AnalizeData();
            oneAnalizeData.Number = ++i;
            oneAnalizeData.AnalizeDataId = Convert.ToInt32(reader["AnalizeDataId"]);
            oneAnalizeData.AnalizeDataName = reader["AnalizeDataName"].ToString();
            listAnalizeData.Add(oneAnalizeData);
          }
        }
        conn.Close();
      }
    }

    if (listAnalizeData.Count == 0) {
      AnalizeData noAnalizeData = new AnalizeData();
      noAnalizeData.AnalizeDataId = 0;
      noAnalizeData.Message = NamesMy.NoDataNames.NoDataInAnalizeData;
      listAnalizeData.Add(noAnalizeData);
    }
    return listAnalizeData;
  }

  public AnalizeData SelectedAnalizeDataById(int AnalizeDataId) {
    string SqlString = "SELECT * FROM AnalizeData WHERE AnalizeDataId=@AnalizeDataId";

    AnalizeData oneAnalizeData = new AnalizeData();
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizeDataId);
        conn.Open();
        using (SqlDataReader reader = cmd.ExecuteReader()) {
          while (reader.Read()) {
            oneAnalizeData.AnalizeDataId = Convert.ToInt32(reader["AnalizeDataId"]);
            oneAnalizeData.AnalizeDataName = reader["AnalizeDataName"].ToString();
          }
        }
        conn.Close();
      }
    }
    return oneAnalizeData;
  }

  public void UpdateAnalizeData(string AnalizeDataName, string Description, int AnalizeDataId)
{
    string SqlString = "UPDATE AnalizeData SET AnalizeDataName=@AnalizeDataName
WHERE AnalizeDataId=@AnalizeDataId";

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@AnalizeDataName", AnalizeDataName);
        cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizeDataId);
        conn.Open();
        cmd.ExecuteNonQuery();
```

```
        conn.Close();
       }
      }
     }

     public void DeleteAnalizeDataByAnalizeDataId(int AnalizeDataId) {
      string SqlString = "DELETE FROM AnalizeData WHERE AnalizeDataId=@AnalizeDataId";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
       using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizeDataId);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
       }
      }
     }

     public int GetLastRecords() {
      int lastRecordNumber = 0;
      string SqlString = "SELECT MAX(AnalizeDataId) FROM AnalizeData";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
       using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        conn.Open();
        using (SqlDataReader reader = cmd.ExecuteReader()) {
         while (reader.Read()) {
          lastRecordNumber = Convert.ToInt32(reader.GetValue(0));
         }
        }
        conn.Close();
       }
      }
      return lastRecordNumber;
     }

    }
   }


  public class AnalizeData {
   private int _Number;
   private int _AnalizeDataId;
   private string _AnalizeDataName;
   private string _Message;

   public AnalizeData() {
    _Number = 0;
    _AnalizeDataId = 0;
    _AnalizeDataName = String.Empty;
    _Message = String.Empty;
   }

   public int Number {
```

```
    set { _Number = value; }
    get { return _Number; }
   }
   public int AnalizeDataId {
    set { _AnalizeDataId = value; }
    get { return _AnalizeDataId; }
   }
   public string AnalizeDataName {
    set { _AnalizeDataName = value; }
    get { return _AnalizeDataName; }
   }
   public string Message {
    set { _Message = value; }
    get { return _Message; }
   }
}
```

Лістинг 5. Код класу «AnalizePacketsPrivider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DetectionApp.Providers {
  class AnalizePacketsPrivider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertBatchAnalizePackets(List<AnalizePackets> AnalizePackets) {
      string SqlString = "INSERT INTO AnalizePackets (PacketsCount, PacketsDate, AnalizeDataId)
" +
        "Values(@PacketsCount, @PacketsDate, @AnalizeDataId)";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          conn.Open();
          for (int i = 0; i < AnalizePackets.Count; i++) {
            cmd.Parameters.AddWithValue("@PacketsCount", AnalizePackets[i].PacketsCount);
            cmd.Parameters.AddWithValue("@PacketsDate", AnalizePackets[i].PacketsDate);
            cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizePackets[i].AnalizeDataId);
            cmd.ExecuteNonQuery();
            cmd.Parameters.Clear();
          }
          conn.Close();
        }
      }
    }
```

```
    public List<AnalizePackets> GetsAllAnalizePacketsByAnalizeDataId(int AnalizeDataId) {
      int i = 0;
      string SqlString = "SELECT * FROM AnalizePackets WHERE
AnalizeDataId=@AnalizeDataId";

      List<AnalizePackets> AnalizePackets = new List<AnalizePackets>();
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizeDataId);
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              AnalizePackets oneAnalizePackets = new AnalizePackets();
              oneAnalizePackets.Number = ++i;
              oneAnalizePackets.AnalizePacketsId = Convert.ToInt32(reader["AnalizePacketsId"]);
              oneAnalizePackets.PacketsCount = Convert.ToInt32(reader["PacketsCount"]);
              oneAnalizePackets.PacketsDate = Convert.ToDateTime(reader["PacketsDate"]);
              oneAnalizePackets.AnalizeDataId = Convert.ToInt32(reader["AnalizeDataId"]);
              AnalizePackets.Add(oneAnalizePackets);
            }
          }
          conn.Close();
        }
      }
      return AnalizePackets;
    }


    public void DeleteAnalizePacketsByAnalizeDataId(int AnalizeDataId) {
      string SqlString = "DELETE FROM AnalizePackets WHERE
AnalizeDataId=@AnalizeDataId";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@AnalizeDataId", AnalizeDataId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}

public class AnalizePackets {
  private int _Number;
  private int _AnalizePacketsId;
  private int _PacketsCount;
  private DateTime _PacketsDate;
  private int _AnalizeDataId;
  private string _Message;
```

```
  public AnalizePackets() {
    _Number = 0;
    _AnalizePacketsId = 0;
    _PacketsCount = 0;
    _PacketsDate = new DateTime();
    _AnalizeDataId = 0;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int AnalizePacketsId {
    set { _AnalizePacketsId = value; }
    get { return _AnalizePacketsId; }
  }
  public int PacketsCount {
    set { _PacketsCount = value; }
    get { return _PacketsCount; }
  }
  public DateTime PacketsDate {
    set { _PacketsDate = value; }
    get { return _PacketsDate; }
  }
  public int AnalizeDataId {
    set { _AnalizeDataId = value; }
    get { return _AnalizeDataId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 6. Код класу «AnalizeForm»

```
using LiveCharts;
using LiveCharts.Wpf;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using DetectionApp.AppCode;
using DetectionApp.Providers;
using DetectionApp.Forms.Systems;

namespace DetectionApp.Forms.Sniffer {
```

```
  public partial class AnalizeForm : Form {
    private Socket _MainSockets;                    //Сокет, який перехоплює всі вхідні пакети
    private byte[] _BytesData = new byte[4096];
    private bool _BContinueCapturing = false;          //Прапорець, який перевіряє, чи потрібно
перехоплювати пакети чи ні

    private delegate void AddTreeNode(TreeNode node);
    private int _Count = 0;
    private int _DetectCountPockets = 0;
    private AnalizeDataProvider _AnalizeDataProvider = new AnalizeDataProvider();
    private AnalizePacketsPrivider _AnalizePacketsPrivider = new AnalizePacketsPrivider();
    private List<AnalizePackets> _AnalizePacketsList = new List<AnalizePackets>();
    private ChartValues<double> _PocketsValues = new ChartValues<double>();
    private ValidationMy _Validation = new ValidationMy();
    private List<string> _DateTimes = new List<string>();
    private LineSeries _PocketsLine = new LineSeries();
    private SeriesCollection _Series = new SeriesCollection();
    private LogsProvider _LogsProvider = new LogsProvider();
    private bool _IsDetect = false;

    public AnalizeForm() {
      InitializeComponent();
      GraphicsPrepare();
    }
    private void StartAnalizeBtn_Click(object sender, EventArgs e) {
      if (IsDataSelected()) {
        if (!PacketCountTimer.Enabled) {
          _AnalizePacketsList.Clear();
          _PocketsValues.Clear();
          _DateTimes.Clear();
          PacketCountTimer.Start();
          StartAnalizeBtn.Text = "&Зупинити";
          _BContinueCapturing = true;
          try {
            //Почати захоплення пакетів
            _MainSockets = new Socket(AddressFamily.InterNetwork, SocketType.Raw,
ProtocolType.IP);
            //Прив’язуємо сокет до вибраної IP-адреси
            _MainSockets.Bind(new IPEndPoint(IPAddress.Parse(IPHostInterfacesCBox.Text), 0));
            //Встановлюємо параметри сокету
            _MainSockets.SetSocketOption(SocketOptionLevel.IP,          //Застосовується лише до
IP-пакетів
                                SocketOptionName.HeaderIncluded, //Встановлюємо включення
заголовка в true
                                true);
            byte[] byTru = new byte[4] { 1, 0, 0, 0 };
            byte[] byOut = new byte[4] { 1, 0, 0, 0 }; //Захоплюємо вихідні пакети

            //Socket.IOControl аналогічний методу  WSAIoctl
            _MainSockets.IOControl(IOControlCode.ReceiveAll, byTru, byOut); //Еквівалент
константи SIO_RCVALL Winsock 2
            //Починаємо отримувати пакети асинхронно
```

```
          _MainSockets.BeginReceive(_BytesData, 0, _BytesData.Length, SocketFlags.None,  new
AsyncCallback(OnReceive), null);
        } catch (Exception ex) {
          MessageBox.Show(ex.Message, "Аналіз пакетів", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
       } else {
        StartAnalizeBtn.Text = "&Аналізувати";
        PacketCountTimer.Stop();
        _BContinueCapturing = false;
        //Щоб припинити захоплення пакетів, закриваємо сокет
        _MainSockets.Close();
       }
     }
   }
   private void OnReceive(IAsyncResult ar) {
     try {
       int nReceived = _MainSockets.EndReceive(ar);
       //Алізуємо отримані байти...
       ParseData(_BytesData, nReceived);
       if (_BContinueCapturing) {
         _BytesData = new byte[4096];
         //Ще один виклик BeginReceive, для продовження отримання вхідних пакетів
         _MainSockets.BeginReceive(_BytesData, 0, _BytesData.Length, SocketFlags.None,
            new AsyncCallback(OnReceive), null);
       }
     } catch (ObjectDisposedException) {
     } catch (Exception ex) {
       MessageBox.Show(ex.Message, "Аналіз пакетів", MessageBoxButtons.OK,
MessageBoxIcon.Error);
     }
   }
   private void ParseData(byte[] byteData, int nReceived) {
     TreeNode rootNode = new TreeNode();
     //Оскільки всі пакети протоколу інкапсульовані в дейтаграмі IP, ми починаємо з аналізу
заголовка IP і дивимось, які дані протоколу він передає
     IPHeader ipHeader = new IPHeader(byteData, nReceived);
     TreeNode ipNode = MakeIPTreeNode(ipHeader);
     rootNode.Nodes.Add(ipNode);

     //Тепер відповідно до протоколу, що передається дейтаграмою IP, ми аналізуємо поле
даних дейтаграми
     switch (ipHeader.ProtocolType) {
       case Protocol.TCP:
         TCPHeader tcpHeader = new TCPHeader(ipHeader.Data,              //IPHeader.Data зберігає
наявні дані переноситься дейтаграмою IP
                                             ipHeader.MessageLength);    //Довжина поля даних

         TreeNode tcpNode = MakeTCPTreeNode(tcpHeader);
         rootNode.Nodes.Add(tcpNode);
         //Якщо порт дорівнює 53, то основним протоколом є DNS
```

```
        //Примітка: DNS може використовувати TCP або UDP, тому перевірка виконується
двічі
        if (tcpHeader.DestinationPort == "53" || tcpHeader.SourcePort == "53") {
          TreeNode dnsNode = MakeDNSTreeNode(tcpHeader.Data,
(int)tcpHeader.MessageLength);
          rootNode.Nodes.Add(dnsNode);
        }
        break;
      case Protocol.UDP:
        UDPHeader udpHeader = new UDPHeader(ipHeader.Data,          //IPHeader.Data зберігає
дані, які передаються IP-дейтаграмою
                                  (int)ipHeader.MessageLength); //Довжина поля даних
        TreeNode udpNode = MakeUDPTreeNode(udpHeader);
        rootNode.Nodes.Add(udpNode);

        //Якщо порт дорівнює 53, то основним протоколом є DNS
        //Примітка: DNS може використовувати TCP або UDP, тому перевірка виконується
двічі
        if (udpHeader.DestinationPort == "53" || udpHeader.SourcePort == "53") {

          TreeNode dnsNode = MakeDNSTreeNode(udpHeader.Data,
                                  //Довжина заголовка UDP завжди становить вісім байтів, тому ми
віднімаємо її із загальної суми,  щоб знайти довжину даних
                                  Convert.ToInt32(udpHeader.Length) - 8);
          rootNode.Nodes.Add(dnsNode);
        }
        break;
      case Protocol.Unknown:
        break;
    }
    AddTreeNode addTreeNode = new AddTreeNode(OnAddTreeNode);
    rootNode.Text = ipHeader.SourceAddress.ToString() + "-" +
ipHeader.DestinationAddress.ToString();
    //Потокове безпечне додавання вузлів
    treeView.Invoke(addTreeNode, new object[] { rootNode });
  }

  //Допоміжна функція, яка повертає інформацію, що міститься в заголовку UDP, як вузол
дерева
  private TreeNode MakeIPTreeNode(IPHeader ipHeader) {
    TreeNode ipNode = new TreeNode();
    ipNode.Text = "IP";
    ipNode.Nodes.Add("Ver: " + ipHeader.Version);
    ipNode.Nodes.Add("Header Length: " + ipHeader.HeaderLength);
    ipNode.Nodes.Add("Differntiated Services: " + ipHeader.DifferentiatedServices);
    ipNode.Nodes.Add("Total Length: " + ipHeader.TotalLength);
    ipNode.Nodes.Add("Identification: " + ipHeader.Identification);
    ipNode.Nodes.Add("Flags: " + ipHeader.Flags);
    ipNode.Nodes.Add("Fragmentation Offset: " + ipHeader.FragmentationOffset);
    ipNode.Nodes.Add("Time to live: " + ipHeader.TTL);

    ////////////////////////////////
```

```
      _Count++;
      ipNode.Nodes.Add("Count: " + _Count);
      switch (ipHeader.ProtocolType) {
        case Protocol.TCP:
          ipNode.Nodes.Add("Protocol: " + "TCP");
          break;
        case Protocol.UDP:
          ipNode.Nodes.Add("Protocol: " + "UDP");
          break;
        case Protocol.Unknown:
          ipNode.Nodes.Add("Protocol: " + "Unknown");
          break;
      }
      ipNode.Nodes.Add("Checksum: " + ipHeader.Checksum);
      ipNode.Nodes.Add("Source: " + ipHeader.SourceAddress.ToString());
      ipNode.Nodes.Add("Destination: " + ipHeader.DestinationAddress.ToString());
      return ipNode;
    }

    //Допоміжна функція, яка повертає інформацію, що міститься в заголовку UDP, як вузол
дерева
    private TreeNode MakeTCPTreeNode(TCPHeader tcpHeader) {
      TreeNode tcpNode = new TreeNode();
      tcpNode.Text = "TCP";
      tcpNode.Nodes.Add("Source Port: " + tcpHeader.SourcePort);
      tcpNode.Nodes.Add("Destination Port: " + tcpHeader.DestinationPort);
      tcpNode.Nodes.Add("Sequence Number: " + tcpHeader.SequenceNumber);
      if (tcpHeader.AcknowledgementNumber != "")
        tcpNode.Nodes.Add("Acknowledgement Number: " + tcpHeader.AcknowledgementNumber);
      tcpNode.Nodes.Add("Header Length: " + tcpHeader.HeaderLength);
      tcpNode.Nodes.Add("Flags: " + tcpHeader.Flags);
      tcpNode.Nodes.Add("Window Size: " + tcpHeader.WindowSize);
      tcpNode.Nodes.Add("Checksum: " + tcpHeader.Checksum);
      if (tcpHeader.UrgentPointer != "")
        tcpNode.Nodes.Add("Urgent Pointer: " + tcpHeader.UrgentPointer);
      return tcpNode;
    }

    //Допоміжна функція, яка повертає інформацію, що міститься в заголовку UDP, як вузол
дерева
    private TreeNode MakeUDPTreeNode(UDPHeader udpHeader) {
      TreeNode udpNode = new TreeNode();
      udpNode.Text = "UDP";
      udpNode.Nodes.Add("Source Port: " + udpHeader.SourcePort);
      udpNode.Nodes.Add("Destination Port: " + udpHeader.DestinationPort);
      udpNode.Nodes.Add("Length: " + udpHeader.Length);
      udpNode.Nodes.Add("Checksum: " + udpHeader.Checksum);
      return udpNode;
    }

    //Допоміжна функція, яка повертає інформацію, що міститься в заголовку UDP, як вузол
дерева
```

```
    private TreeNode MakeDNSTreeNode(byte[] byteData, int nLength) {
      DNSHeader dnsHeader = new DNSHeader(byteData, nLength);
      TreeNode dnsNode = new TreeNode();
      dnsNode.Text = "DNS";
      dnsNode.Nodes.Add("Identification: " + dnsHeader.Identification);
      dnsNode.Nodes.Add("Flags: " + dnsHeader.Flags);
      dnsNode.Nodes.Add("Questions: " + dnsHeader.TotalQuestions);
      dnsNode.Nodes.Add("Answer RRs: " + dnsHeader.TotalAnswerRRs);
      dnsNode.Nodes.Add("Authority RRs: " + dnsHeader.TotalAuthorityRRs);
      dnsNode.Nodes.Add("Additional RRs: " + dnsHeader.TotalAdditionalRRs);
      return dnsNode;
    }
    private void OnAddTreeNode(TreeNode node) {
      treeView.Nodes.Add(node);
    }
    private void AnalizeForm_Load(object sender, EventArgs e) {
      string strIP = null;
      IPHostEntry HosyEntry = Dns.GetHostEntry((Dns.GetHostName()));
      if (HosyEntry.AddressList.Length > 0) {
        foreach (IPAddress ip in HosyEntry.AddressList) {
          strIP = ip.ToString();
          IPHostInterfacesCBox.Items.Add(strIP);
        }
      }
    }
    private void AnalizeForm_FormClosing(object sender, FormClosingEventArgs e) {
      if (_BContinueCapturing) {
        _MainSockets.Close();
      }
    }

    private void PacketCountTimer_Tick(object sender, EventArgs e) {
      AnalizePackets oneAnalizePackets = new AnalizePackets();
      oneAnalizePackets.PacketsCount = _Count;
      oneAnalizePackets.PacketsDate = DateTime.Now;
      _AnalizePacketsList.Add(oneAnalizePackets);
      if (_Count > _DetectCountPockets && _IsDetect == false) {
        _IsDetect = true;
        _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було виявлено активність
мережевого трафіку IP адреси '" +
          IPHostInterfacesCBox.Text + "'", DateTime.Now);
        MessageBox.Show("Виявлено активність мережевого трафіку", "Увага!");
      }
      _Count = 0;
      BildGraphics();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        StartAnalizeBtn.Text = "&Аналізувати";
        PacketCountTimer.Stop();
        _AnalizeDataProvider.InsertAnalizeData(AnalizeDataNameTBox.Text);
```

```
        int lastAnalizeDataId = _AnalizeDataProvider.GetLastRecords();
        for (int i = 0; i < _AnalizePacketsList.Count; i++) {
          _AnalizePacketsList[i].AnalizeDataId = lastAnalizeDataId;
        }
        _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було проведено аналіз пакетів
та збережено дані під назвою '" +
          AnalizeDataNameTBox.Text + "'", DateTime.Now);
        _AnalizePacketsPrivider.InsertBatchAnalizePackets(_AnalizePacketsList);
        _PocketsValues.Clear();
        _DateTimes.Clear();
        treeView.Nodes.Clear();
        AnalizeDataNameTBox.Text = String.Empty;
      }
    }
    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataEntering(AnalizeDataNameTBox.Text)) {
        AnalizeDataNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        AnalizeDataNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
    private bool IsDataSelected() {
      bool isCorrect = true;
      if (IPHostInterfacesCBox.SelectedItem != null) {
        IPHostInterfacesValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        IPHostInterfacesValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
        MessageBox.Show("Виберіть інтерфейс для захоплення пакетів.", "Увага!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
      return isCorrect;
    }

    private bool IsCountPocketsEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataConvertToInt(CountPocketsTBox.Text)) {
        CountPocketsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CountPocketsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void GraphicsPrepare() {
      SpecifyBtn_Click(SpecifyBtn, EventArgs.Empty);

      _PocketsValues.Add(0);
```

```
      _DateTimes.Add(DateTime.Now.ToShortTimeString());

      GraphicsCC.AxisX.Add(new LiveCharts.Wpf.Axis() {
        Title = "Час",
        Labels = _DateTimes
      });

      _PocketsLine.Title = "Кількість пакетів";
      _PocketsLine.Values = _PocketsValues;

      _Series.Add(_PocketsLine);
      GraphicsCC.Series = _Series;
    }
    private void BildGraphics() {
      _PocketsValues.Add(_AnalizePacketsList[_AnalizePacketsList.Count - 1].PacketsCount);
      _DateTimes.Add(_AnalizePacketsList[_AnalizePacketsList.Count -
1].PacketsDate.ToShortTimeString());
    }

    private void SpecifyBtn_Click(object sender, EventArgs e) {
      if (IsCountPocketsEnteringCorrect()) {
        _DetectCountPockets = Convert.ToInt32(CountPocketsTBox.Text);
      }

    }
  }
}
```