Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall”

Виконав здобувач вищої освіти
IV курсу, групи КН-22
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Криця О.В.
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А.
«____» _____ 2026 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“Інформаційні технології”*_____

Спеціальність 122 *“Комп’ютерні науки”*

Освітньо-професійна (освітньо-наукова) програма _*“Комп’ютерні науки”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 06 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Криці Олександру Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 116-02 від 06.02.2026 року

3. Строк подання студентом роботи до захисту *23.05.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 06 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 06 » лютого 2026 р.

Підпис керівника
_____ Лисенко І.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 06 » лютого 2026 р.

Підпис здобувача
_____ Криця О.В.
(прізвище та ініціали)

АНОТАЦІЯ

Криця О.В. Програмне забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи інтелектуального аналізу додатків рівня L7 у Firewall.

Метою розробки є програмне забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall.

Результат роботи – програмна реалізація системи інтелектуального аналізу додатків рівня L7 у Firewall.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Visual C++.

Ключові слова: комп'ютерні науки, інтелектуальний аналіз додатків

ABSTRACT

Krytsia O.V. Software for the L7 level application intelligence analysis system in Firewall. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for the L7 level application intelligence analysis system in Firewall.

The purpose of the development is the software for the L7 level application intelligence analysis system in Firewall.

The result of the work is the software implementation of the L7 level application intelligence analysis system in Firewall.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with the software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Visual C++ environment.

Keywords: computer science, application intelligence analysis

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЛОМ	–	локальна обчислювальна мережа
MME	–	міжмережні екрани
ATM	–	асинхронний режим передачі
BSD	–	адаптована для Internet реалізація операційної системи UNIX
ICMP	–	міжмережний протокол управляючих повідомлень
IP	–	Internet Protocol – міжмережний протокол
NFS	–	мережна файлова система
PPP	–	протокол передачі від точки до точки
RFC	–	опис набору протоколів Internet
RPC	–	віддалений виклик процедури
SLIP	–	міжмережний протокол для послідовного каналу
SMTP	–	Simple Mail Transfer Protocol – простий протокол передачі пошти
TCP	–	Transmission Control Protocol – протокол управління передачею
UDP	–	User Datagram Protocol – протокол користувальницьких датаграм
UNIX	–	багатозадачна операційна система
UTP	–	незахищена вита пара
URL	–	уніфікований покажчик інформаційного ресурсу

ВСТУП

Актуальність теми. Firewall 7-го рівня представляють собою значний прогрес у мережевій безпеці, надаючи можливості, що значно перевершують можливості традиційних Firewall 3-го та 4-го рівнів. Розуміючи та керуючи трафіком на прикладному рівні, ці Firewall пропонують комплексні та контекстуальні заходи безпеки, захищаючи від широкого спектру сучасних загроз. Їхня здатність перевіряти, аналізувати та контролювати дані 7-го рівня робить їх незамінним інструментом для захисту сучасних складних мережевих середовищ. Оскільки кіберзагрози продовжують розвиватися, важливість надійних, гнучких та інтелектуальних рішень для Firewall, таких як Firewall 7-го рівня, неможливо переоцінити.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем інтелектуального аналізу додатків рівня L7 у Firewall.

– Дослідження системи інтелектуального аналізу додатків рівня L7 у Firewall.

– Програмна реалізація системи інтелектуального аналізу додатків рівня L7 у Firewall.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі інтелектуального аналізу додатків рівня L7 у Firewall.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Традиційні Firewall, що працюють на рівнях 3 та 4 моделі OSI, в основному зосереджуються на IP-адресах, портах та прапорцях. Вони розглядають трафік запитів і відповідей як окремі, непов'язані потоки, що створює кілька обмежень:

1. Окремі потоки: запити та відповіді обробляються як різні потоки даних, незважаючи на те, що з точки зору користувача вони є частиною одного сеансу зв'язку.

2. Відсутність контексту: Без можливостей сеансу ці Firewall не можуть пов'язувати трафік запитів і відповідей, що обмежує їхню здатність впроваджувати контекстні заходи безпеки.

3. Обмежена видимість: Традиційні Firewall не розуміють та не перевіряють дані на вищих рівнях, таких як 7-й рівень (рівень додатків), де працюють такі протоколи, як HTTP.

1.2 Область застосування

Порівняємо журнали L4 і L7 firewall

Порівняємо запис у журналі firewall, що розбирає тільки заголовок транспортного (четвертого) рівня моделі OSI ISO: Так, є інформація про джерело й одержувача, можна догадатися за номером порту 443, що це усередині з великим ступенем імовірності, як говорять англійці, з'єднання SSL. Але навряд чи отут можна побачити якийсь інцидент.

Зрівняєте запис у журналі firewall для того ж самого TCP з'єднання, де розбирається ще й сам контент переданий у поле даних TCP/IP: Тут ви бачите, що

Іванов з відділу маркетингу виклав на Slideshare файл із позначкою «не для поширення». Це приклад реального інциденту, де співробітник виклав конфіденційні плани розвитку компанії на рік в Інтернет. Ця інформація отримана в L7 firewall на основі аналізу того ж самого з'єднання, що й вище в L4 firewall і відразу інформації стає досить, щоб зрозуміти, що був інцидент. І це зовсім інший підхід до аналізу трафіку. Але це накладає серйозне навантаження на процесор і пам'ять пристрою.

Іноді відчувається, що рівень деталізації журналів в NGFW схожий на рівень деталізації в системах SIEM, які збирають інформацію із крупиць із різних джерел. Саме тому NGFW одне із кращих джерел інформації для SIEM.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтелектуального аналізу додатків рівня L7 у Firewall, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

У минулому в операційних системах компанії Microsoft була проблема – Windows мав вічні проблеми із захистом, які хакери й інші шахраї в мережі використовували у своїх цілях.

Один зі способів боротьби із цим була установка брандмауера (firewall) – застосунку, що блокує шкідливі й/або небажані процеси в мережі і яке могло гарантувати, що хитромудрі сторонні програми не проберуться на ваш ПК.

На сьогоднішній день, коли Microsoft став випускати свій власний, убудований в Windows, брандмауер не припиняються дискусії на рахунок його надійності. Адже кібератаки залишаються серйозною погрозою. Сторонні застосунки від третіх сторін допоможуть вам зробити вашу систему більше захищеної.

Це Firewall, які, на нашу думку, заслуговують вашої уваги незалежно від того, яку версію Windows ви використовуєте.

Більшість із нас використовують брандмауера на роутері, а також окремий програмний брандмауер на Windows. Деякі виробники антивірусів сполучають брандмауера зі своїм безкоштовним антивірусом, а інші пропонують окремий застосунок.

Тут ми розглядаємо firewall, які можна використовувати паралельно з вашим стандартним антивірусом.

ZoneAlarm Free Firewall

Для багатьох з нас ZoneAlarm Free Firewall був додатком, що вперше познайомило нас із брандмауерами, і він був обов'язковою програмою в ті дні, коли Windows кричав на весь інтернет «Зламай мене! Це простіше простого!» Поточна версія ZoneAlarm Free Firewall ховає відкриті порти, визначає потенційно небезпечний трафік, відключає шкідливі програми й з'єднується з DefenseNet, що надає відновлення захисту в реальному часі, коли з'являються нові погрози.

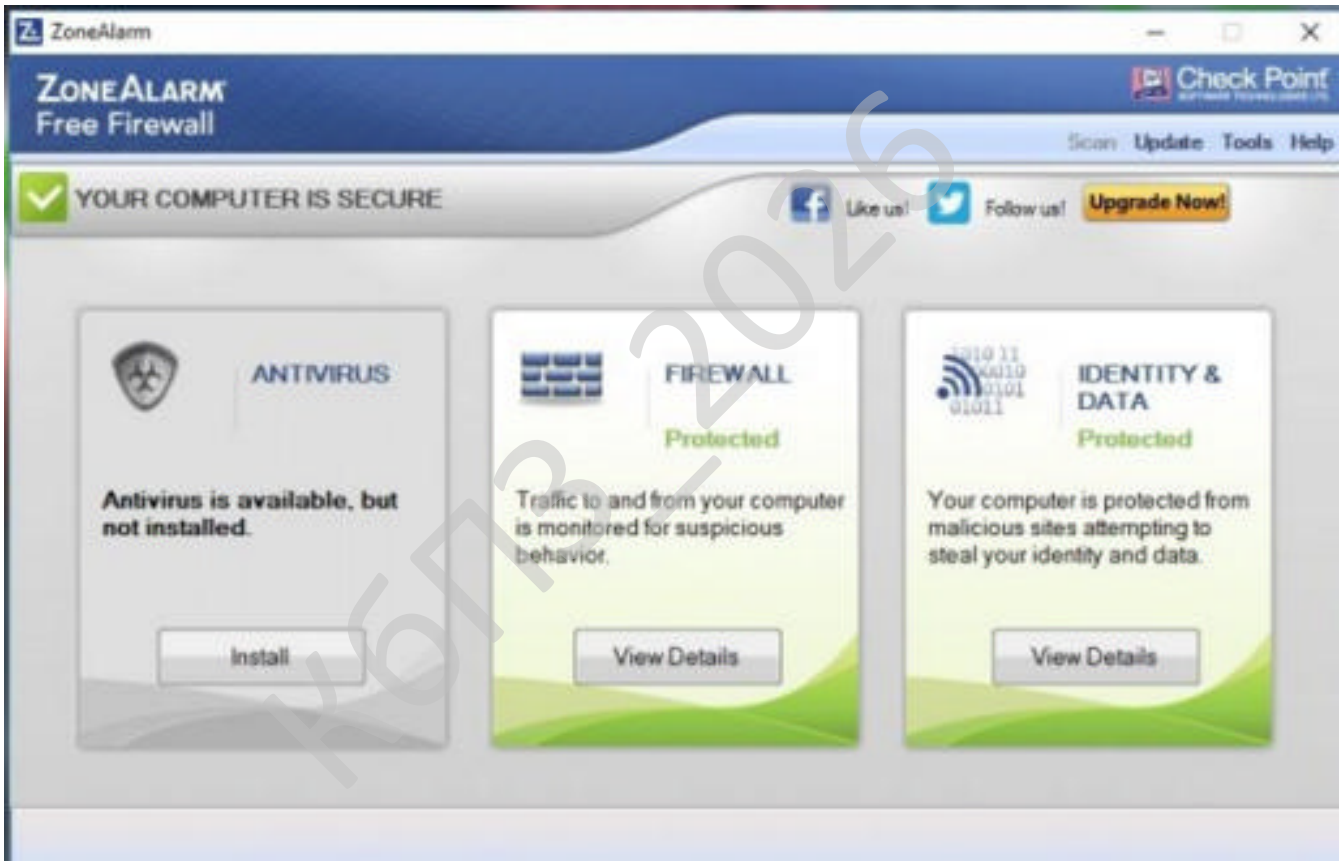

Рисунок 2.1 – Захист у реальному часі й додаткові інструменти для безпеки Wi-Fi

+ Може похвастатися режимом «невидимки».

+ Відновлення від DefenseNet.

+ Убудоване резервне копіювання онлайн.

– Потрібен адреса електронної пошти для активації.

Брандмауер також захищає ваш комп'ютер у публічних мережах Wi-Fi і дає 5 гігабайт місця в інтернеті для резервного копіювання через IDrive.

Коли ви встановлюєте це програмне забезпечення, уважно читайте кожний крок установки, якщо ви не хочете бачити Yahoo вашою домашньою сторінкою.

Кнопка «Налаштувати установку» досить добре захована.

Вам також прийдеться вказати ваш email для активації брандмауера, хоча ZoneAlarm обіцяє не передавати його третій стороні.

Якщо вас це влаштовує, ZoneAlarm Free Firewall є кращим безкоштовним брандмауером для Windows.

Comodo Free Firewall

Установці Comodo Free Firewall варто приділити увагу, оскільки в комплекті додатково йде браузер, і якщо ви будете клацати Вперед дуже швидко, те не помітете, як цей браузер установиться.

Також Comodo Free Firewall запропонує вам установити Yahoo вашою домашньою сторінкою, але опцію для скасування цього набагато простіше побачити, ніж в ZoneAlarm.

Брандмауер також запропонує зробити його власний браузер Comodo Dragon вашим браузером за замовчуванням і імпортувати ваші налаштування з Chrome.

Після установки Comodo Free Firewall перебуває в маленькому вікні зверху із правої сторони вашого робочого стола, і ви можете одним кличем одержати доступ до «піскового» версіям Chrome, Firefox і Comodo Browser для більше безпечної роботи в інтернеті.

Також ви можете запускати будь-які інші програми в пісочниці – це дуже корисно, якщо ви любите експериментувати з різними програмами.

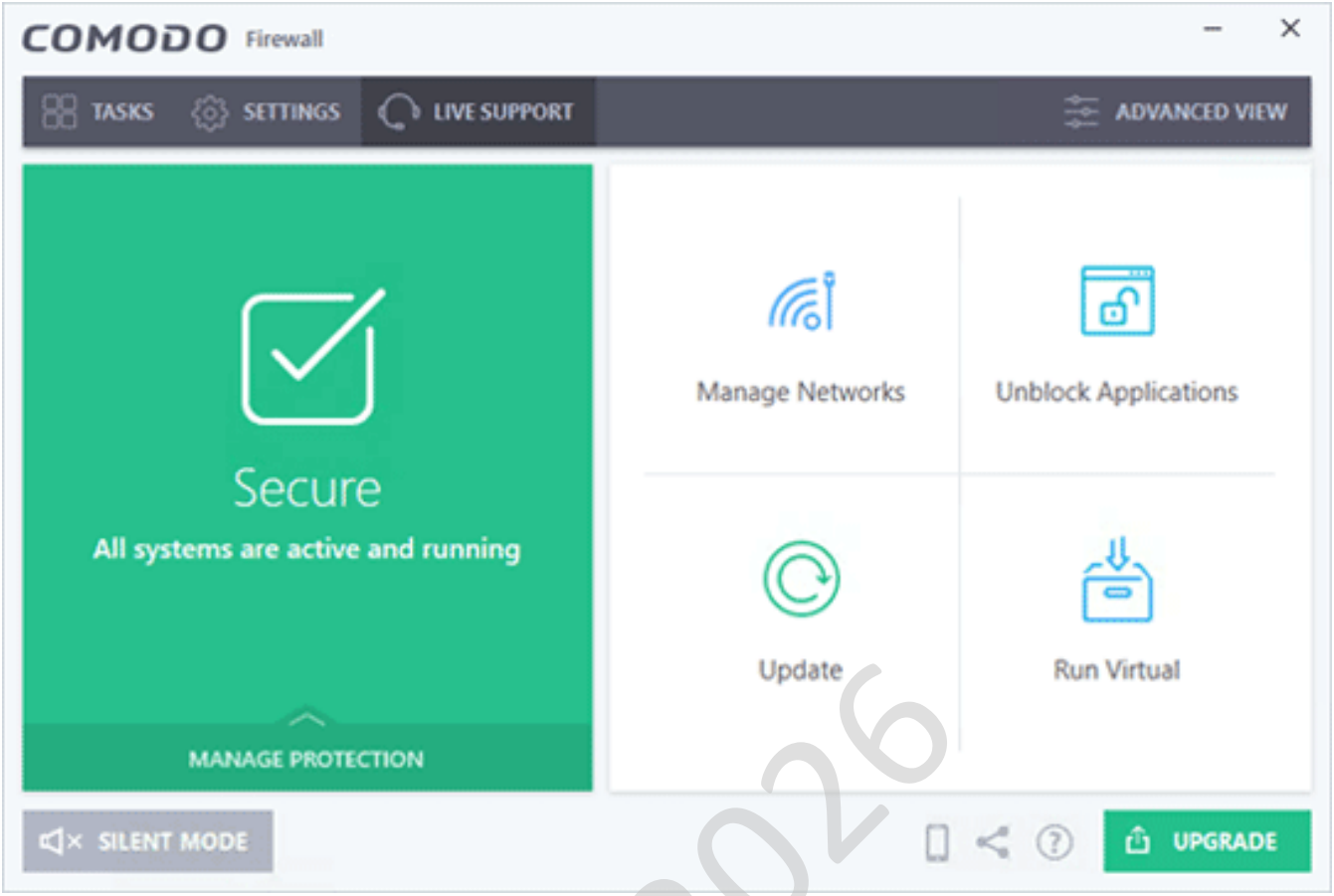

Рисунок 2.2 – Ефективне рішення зі швидким приміщенням вірусів в «пісочницю»

+ Пісочниця для ненадійних програм.

+ Монітори для підозрілої активності.

– Намагається встановити браузер Comodo Dragon.

В Comodo Free Firewall легко контролювати дозволу для різних мереж, і ви можете створювати правила для певних програм (дозволяючи вхідний трафік, що виходить трафік, обоє або жодного).

Вірускоп відслідковує поводження процесів усього, що виглядає підозріло (хоча ваш стандартний антивірус, швидше за все, також це робить), а фільтрація веб сайтів дозволяє блокувати ненадійні сайти.

Comodo Free Firewall запропонує змінити вашого DNS провайдеру на Comodo Secure DNS для більше швидкої й безпечної роботи в мережі.

GlassWire

GlassWire – безкоштовний брандмауер з гарним інтерфейсом, що показує вам які конкретно програми відправляють і одержують дані через вашу мережу і який обсяг трафіку вони використовують, що дозволяє легко помітити щось підозріле.

Рисунок 2.3 – Добре пророблений, інформативний брандмауер для забезпечення безпеки

+ Відображає рівень використання мережевого трафіку.

+ Може заблокувати окремі програми.

+ Сканує застосунки на підозріле поводження.

– Не дуже зручний і інтуїтивний.

У цьому firewall можна переглянути величезну кількість інформації, що втім може спочатку небагато заплутати.

Вас повідомлять, коли новий застосунок або сервіс захоче вийти в інтернет, і ви можете дозволити або заблокувати це одним клацанням миші.

По великому рахунку, GlassWire є всього лише інтерфейсом, побудованим на основі брандмауері Windows, тому він не краще його, незважаючи на привабливість і інформативність.

Преміум версія GlassWire містить у собі деякі додаткові функції, такі як моніторинг вашої веб камери й мікрофона, зберігання записів більше одного місяця й моніторинг декількох вилучених з'єднань.

Але для звичайної роботи відмінно підійде й безкоштовна версія.

TinyWall

На відміну від деяких безкоштовних Firewall, він не відволікає вас спливаючими вікнами або події, що інформуючими про кожному найменшому процесі.

TinyWall створений для поліпшення функціональності рідного брандмауера Windows без необхідності бути експертом, і хоча він і пропонує таку ж систему занесення в білий список, як і інші програми (для того, щоб перевірені програми могли безперешкодно з'єднуватися з інтернетом), він робить це за допомогою гарячих клавіш, а не спливаючих вікон.

TinyWall відповідає назві й займає всього приблизно 1 мегабайт місця на вашім вінчестері.

Це дуже корисне доповнення, якщо ви вирішили використовувати убудований захист Windows.

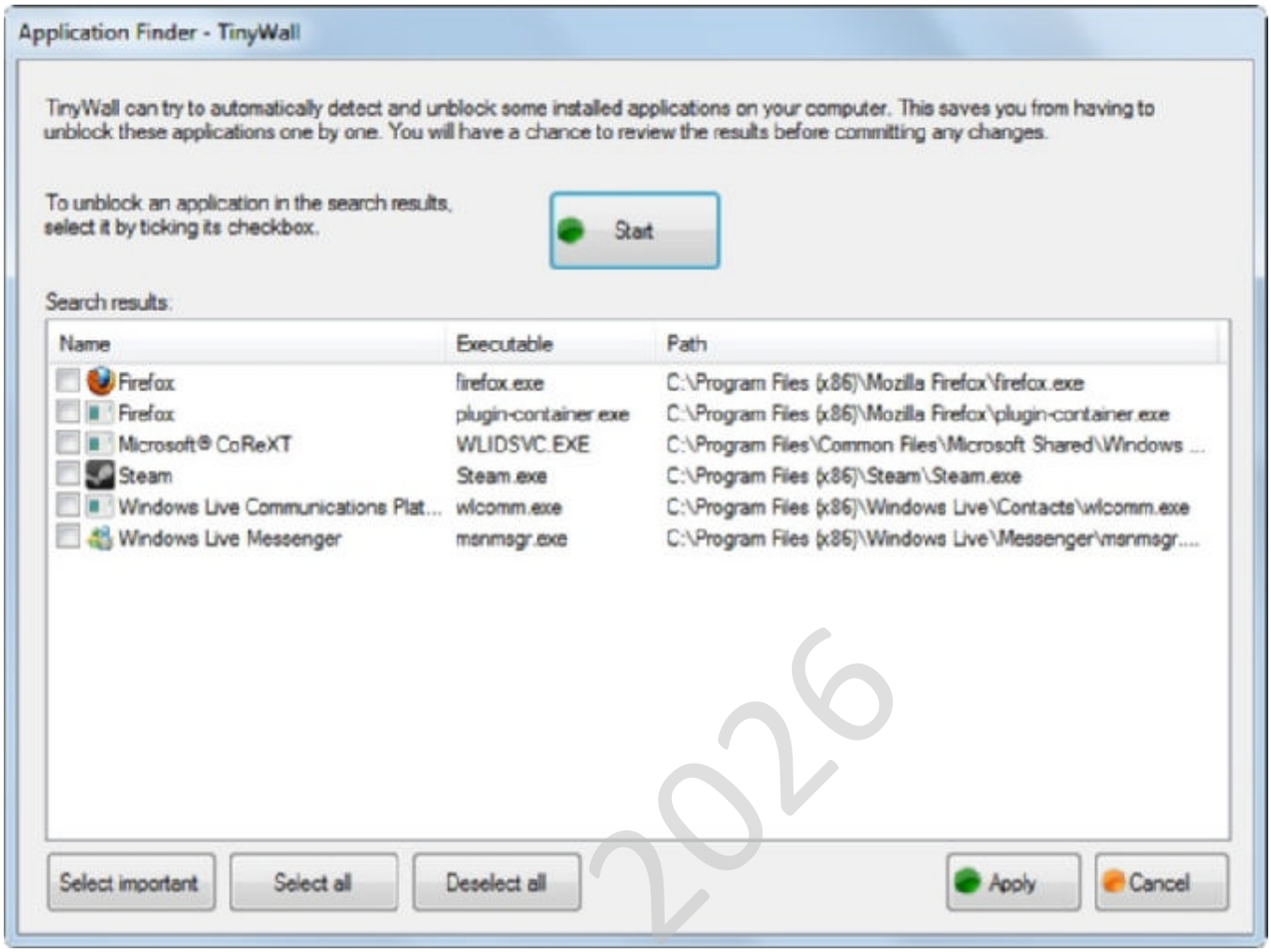

Рисунок 2.4 – Застосунок, що дає вам додатковий контроль над брандмауером Windows

+ Відсутні докучливі спливаючі вікна

+ Простий інтерфейс без надмірностей

– Вимагає ручного керування у випадку складних програм TinyWall більше відомий тим, що він не робить, ніж тим, що робить.

OpenDNS Home

Це означає, що захист застосовується до будь-якого пристрою вашої мережі. А в наш час корпоративних мереж це дуже актуально.

OpenDNS Home автоматично блокує відомі погрози й пропонує багато варіантів фільтрування інтернет даних, щоб тримати ваших дітей подалі від усякого бруду. Крім цього, він допомагає уникнути фішингових атак.

Програма безкоштовна й має зрозумілу інструкцію установки для будь-яких пристроїв.

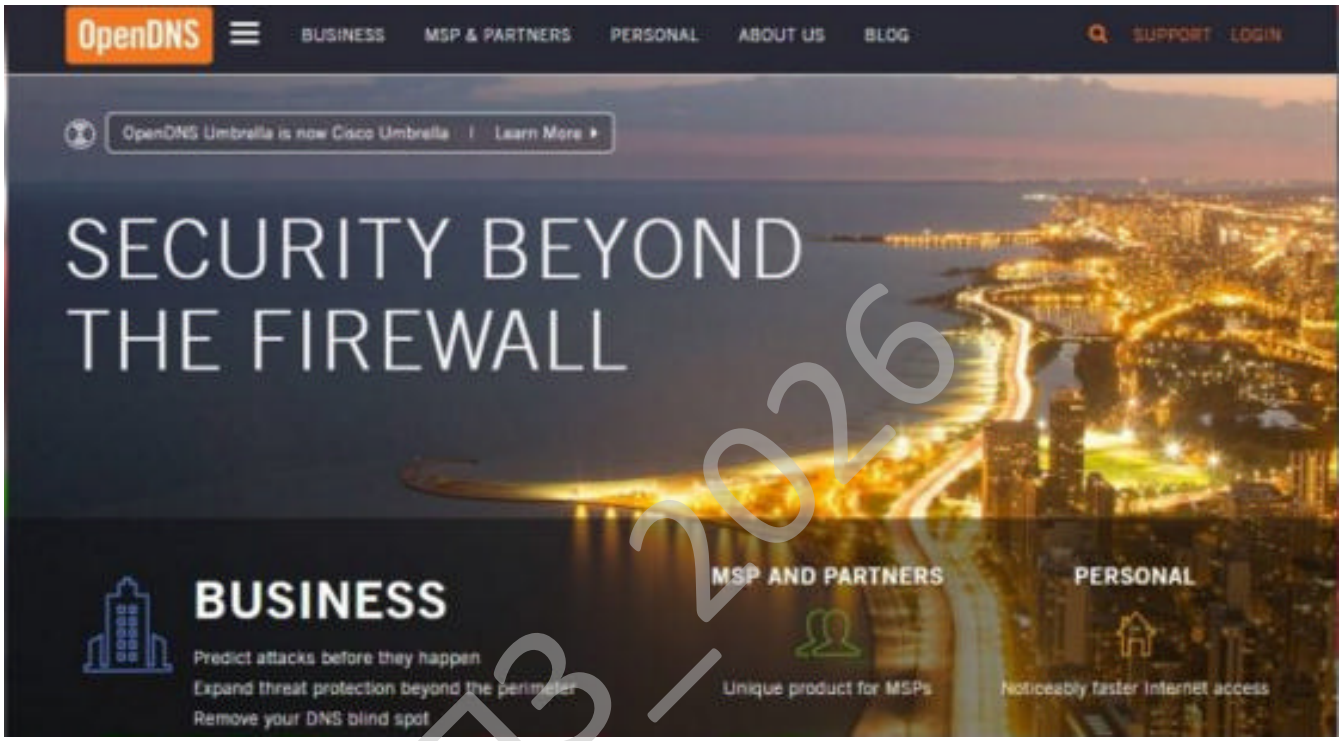

Рисунок 2.5 – Налаштовуються фильтри, які для захисту ваших персональних даних на рівні роутера

+ Альтернативний підхід до захисту брандмауера.

+ Можна використовувати додатково до існуючого брандмауера.

– Не заміняє повноцінну окрему програму OpenDNS не є безкоштовним брандмауером, якому можна скачати з інтернету; це заміна налаштувань вашого роутера, коли він з'єднується з інтернетом через сервери OpenDNS.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка застосунків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка застосунків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки застосунків Windows дозволяють автоматизувати процес створення застосунку. Для цього використовуються генератори застосунків. Програміст відповідає на питання генератора застосунків і визначає властивості застосунку – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор застосунків, створить застосунок, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої застосунки. Подібні засоби автоматизованого створення застосунків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики застосунку й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні

застосунки, а також застосунки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину застосунку програмістові прийдеться розробляти самостійно. Вихідний текст застосунку, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон застосунку – це вже половина всієї роботи. Вихідні тексти застосунків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти застосунків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої застосунки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-застосунки на базі об'єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато

елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який застосунок взаємодіє з Windows через API, що містить кілька сотень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи інтелектуального аналізу додатків рівня L7 у Firewall.

В процесі розробки випускної кваліфікаційної роботи за першим

(бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У 2026 році брандмауер більше не є «коробкою, яка блокує порти». Це критична точка контролю для ідентифікації, зашифрованого трафіку, сайтів, хмари та віддаленої роботи . І саме тому багато компаній зазнають невдачі таким самим чином: вони купують потужний NGFW… але експлуатують його так, ніби це брандмауер 10-річної давності. У цій статті ми розглянемо, що означає керований брандмауер у 2026 році та чого слід вимагати (і уникати), коли ви шукаєте професійний сервіс: NGFW, IPS, VPN та сегментація з практичним підходом, розробленим для малого та середнього бізнесу.

– Якщо вам потрібне коротке емпіричне правило: ви купуєте брандмауер один раз, а безпекою користуєтеся щодня.

– Мета: зменшити ризики та простої, а також уникнути «крихких» конфігурацій, які виходять з ладу в найневідповідніший момент.

– Підхід: що запитувати, що вимірювати та які червоні лінії слід помітити перед підписанням.

Класичний периметр більше не існує як єдина «точка входу». У вас є SaaS, віддалений доступ, сайти, мобільні пристрої, Інтернет речей і трафік, який майже завжди зашифрований. У такому випадку брандмауер без безперервної роботи стає «хибним відчуттям безпеки». Посібники з найкращих практик наполягають на тому, що справжня цінність полягає в політиці, конфігурації, тестуванні та підтримці контролю периметра, а не лише в пристрої. Ось чому «керований» важливий так само, як і «NGFW».

Керований брандмауер – це послуга, а не продукт. Він включає чіткі обов'язки, процеси та показники. Якщо хтось пропонує вам «керований», але говорить лише про встановлення та ліцензії, ви, ймовірно, купуєте підтримку, а

не експлуатацію.

– Дизайн та архітектура: зони, сегментація, VPN, безпечна публікація сервісів та висока доступність.

– Політики та правила: створення, перевірка, контроль змін та регулярне очищення застарілих правил.

– Оновлення: прошивка, сигнатури, посилення захисту та огляд небезпечних конфігурацій.

– Моніторинг: відповідні події, сповіщення, що потребують дій, та кореляція з іншими системами (XDR/SIEM, де це можливо).

– Звітність: прозорість для керівництва (ризики, тенденції) та технічна дорожня карта (пріоритетні покращення).

NGFW вирізняється тим, що може розуміти програми та контекст, а не лише IP-адреси/порти. Це дозволяє розробляти детальніші політики, проводити кращі розслідування та зменшувати кількість «дірок», створених погано розробленими винятками.

Керування на основі програм та користувачів (не лише портів)

Бізнесу потрібні зрозумілі правила: «дозволити Teams та блокувати неавторизовані програми», «тільки фінансовий відділ має доступ до ERP», «постачальники лише для певного сегмента». Це зменшує ризики, не покладаючись на величезні, некеровані набори правил.

Перевірка зашифрованого трафіку (TLS) з чіткими критеріями

Більшість трафіку зашифровано. Якщо ви нічого не перевіряєте, ви втрачаєте видимість; якщо ви перевіряєте «все», ви можете порушити продуктивність або конфіденційність. У 2026 році ключовим є правильне проектування: що перевіряти, де, які винятки застосовувати та як вимірювати вплив.

Веб/DNS-фільтрація та захист від відомих загроз

Добре керована система захисту даних NGFW зменшує кількість «уникнутих» інфекцій, блокуючи шкідливі домени, категорії високого ризику та

комунікації командного управління. Це базовий рівень, але ефективний, якщо він налаштований на ваш бізнес.

Реальна продуктивність із увімкненою безпекою

Цей момент постійно ігнорується: пропускна здатність «за даними» рідко збігається з пропускною здатністю з увімкненою IPS, фільтрацією та TLS-інспекцією. Під час покупки запитуйте реалістичні цифри та перевіряйте їх відповідно до вашого шаблону використання (сайти, VPN, SaaS, відеодзвінки). Якщо ви хочете посилити периметр як послугу, має сенс пов’язати його з повним рівнем безпеки периметра для компаній (брандмауер, політики, сегментація та безперервна робота).

IPS у 2026 році

IPS – це цінна, але делікатна функція: вона блокує (або сповіщає про) схеми атак. Типовою помилкою є її ввімкнення на «максимальній чутливості» без налаштування, що створює шум і блокування, які зрештою вимикають захист через втому сповіщень.

– Запитайте план налаштування: фаза навчання, початковий режим виявлення, потім перехід до профілактики з перевіреними правилами.

– Вимагайте постійних оновлень: актуальних підписів та переглядайте їх, коли ваші програми змінюються.

– Запитайте контекст: що було заблоковано, на який ресурс це впливає та яка дія рекомендована (не просто «сповіщення»).

– Попросіть надати добре задокументовані винятки: «чому це існує» та «коли це буде переглянуто».

Добре керована система захисту від небезпеки (IPS) інтегрується з рештою вашого захисту (кінцева точка/XDR, ідентифікація, резервне копіювання, моніторинг). Наприклад, якщо ви вже використовуєте XDR, має сенс підключити сигнали для швидшого розслідування та зменшення впливу. Див. також: у блозі ми розглядаємо, як покращити час виявлення та реагування за допомогою Sophos XDR у бізнес-середовищі, особливо коли співіснує кілька рівнів безпеки.

VPN у 2026 році

VPN залишається ключовим для сайтів, доступу постачальників та віддаленої роботи. Але у 2026 році погано розроблений VPN стає дорогою для зловмисників: викрадені облікові дані, відсутність багатофакторної автентифікації, надмірні привілеї та відсутність журналів.

Що запитувати в VPN типу "сайт-сайт" (сайти)

VPN-з'єднання типу «сайт-сайт» має бути стабільним, передбачуваним та сегментованим. Мета полягає не в тому, щоб «об'єднати мережі», а в тому, щоб підключити те, що необхідно, з найменшим рівнем привілеїв.

– Правильно налаштований IPsec/IKE та сучасні шифри.

– Вибір підмережі та контрольована маршрутизація (не «все до всього»).

– Міжзональні політики: те, що перетинає VPN, проходить через правила та журнали.

– Тести на перехід на резервний режим, якщо у вас два інтернет-провайдери або висока доступність.

Що запитувати у VPN віддаленого доступу (користувачі та постачальники)

Віддалений доступ має бути надійним щодо ідентифікації та «мінімального доступу». Йдеться не лише про шифрування, а й про обмеження шкоди у разі компрометації облікового запису:

– Багатофакторна автентифікація (MFA) обов'язкова для всіх віддалених доступів.

– Ролеві профілі (співробітник, IT-фахівець, постачальник) з окремими дозволами.

– Позитивний стан пристрою, де це можливо (керований пристрій, виправлення, EDR).

– Ведення журналу та відстеження (хто підключається, звідки, до чого вони отримують доступ і що вони змінюють).

Чого слід уникати у VPN (типові помилки)

Деякі рішення здаються зручними, але значно збільшують ризик. Це невеликі «поступки», які згодом перетворюються на інциденти.

– Віддалений доступ без багатофакторної аварійної діагностики (або багатофакторна аварійна діагностика «лише для деяких»).

– Профілі «адміністратора» за замовчуванням для постачальників, «тому що це швидше».

– Без сегментації: VPN підключається та бачить усю внутрішню мережу.

– Політика відсутності реєстрації: коли трапляється інцидент, «історії немає».

Сегментація у 2026 році

Якщо зловмисник проникає всередину, шкоду визначає не лише «чи проникає він всередину», а й «наскільки далеко він може просунутися». Сегментація (і мікросегментація, де це можливо) обмежує горизонтальне переміщення та захищає критично важливі активи. У малих і середніх підприємствах зазвичай достатньо добре виконаної зональної сегментації: користувачі, сервери, VoIP, Інтернет речей, гості, OT/виробництво, якщо воно існує, та управління IT. Важливо те, що міжзонний зв'язок регулюється політиками та журналами:

– VLAN та зони з чіткими правилами (дозволяйте необхідне, блокуйте решту).

– Розділення адміністраторів (управління IT ніколи не змішується з трафіком користувачів).

– Демілітаризована зона для опублікованих послуг (і публікувати лише те, що вкрай необхідно).

– Доступ до ERP/БД з контрольованих сегментів, а не з «будь-якого ПК».

Якщо ваша мережа потребує редизайну або розширення (нові сайти, корпоративний Wi-Fi, Інтернет речей), варто вирішити цю проблему за допомогою корпоративних мережевих рішень, щоб сегментація не була латкою, а міцною основою.

Правила, зміни та управління: де порушується 80% периметральної безпеки

Більшість «інцидентів брандмауера» спричинені не відсутніми функціями. Вони спричинені накопиченими правилами, винятками без власника, терміновими змінами без перевірки та відсутністю очищення. Керований брандмауер повинен включати процес управління.

– Контроль змін: хто запитує, хто затверджує, хто виконує та як це перевіряється.

– Регулярний огляд: невикористані правила, дублікати правил та «тимчасовий» доступ, який залишився назавжди.

– Мінімальна документація: кожне критичне правило повинно мати «чому воно існує» та «який ризик воно приймає».

– Резервні копії конфігурації: для швидкого відкату, якщо щось порушить роботу системи.

Висока доступність: якщо брандмауер вийде з ладу, ваш бізнес зупиниться

Якщо ваш брандмауер – це вихід з Інтернету, VPN сайту та шлях доступу до програм, це компонент забезпечення безперервності. А у 2026 році безперервність – це не розкіш: це базові операції:

– НА (активний/пасивний або активний/активний) з регулярним тестуванням відновлення після збою.

– Подвійний інтернет-провайдер, якщо ваша залежність від підключення висока.

– Живлення та стійка (ДБЖ, резервування), де це можливо.

– Документований план дій у надзвичайних ситуаціях: що робити у разі відключення, погіршення стану або нападу.

Контрольний список: що вимагати від керованого брандмауера у 2026 році

Використовуйте її як контрольний список для покупки та список контролю якості під час порівняння постачальників.

– Операційна сфера: правила, IPS, VPN, оновлення, перевірки та реагування на сповіщення.

– Угода про рівень обслуговування та години покриття: реальний час реагування та канал ескалації.

– Процес змін: затвердження, період обслуговування та відкат.

– Корисна звітність: ключові показники ефективності безпеки + практичні рекомендації (не «30-сторінковий PDF-файл»).

– Сегментація за зонами: початковий дизайн + еволюція в міру змін бізнесу.

– VPN з MFA: надійний віддалений доступ з дозволами на основі ролей.

– Журнали та відстеження: зберігання, експорт та доступ під час інциденту.

– Тестування: перевірка критичних правил, тести високої доступності для відновлення та відновлення, якщо враховано забезпечення безперервності.

– Ризик та відповідність: докази для аудитів (GDPR, NIS2, де це застосовується).

– Експертна підтримка: не просто «відкриття заявок», а реальна оцінка інженерів безпеки.

Чого слід уникати

Ці сигнали часто призводять до додаткових витрат, інцидентів або розчарувань. Якщо вони з'являються під час передпродажної підготовки, краще врахувати їх до укладання договору або пошукати альтернативу.

– «Керований» = встановив і забув. Без оглядів, без налаштування, без звітності.

– Непрозоре ліцензування (ви не знаєте, що входить до комплекту, термін дії яких закінчується або як це впливає на ключові функції).

– Неконтрольовані винятки («ми відкриваємо це, а потім побачимо»).

– Сегментації немає, бо «це складно» (або тому що ніхто не хоче торкатися мережі).

– VPN без багатофакторної автентифікації (MFA) або з надмірно широким доступом.

– Недоступні журнали або недостатнє зберігання для розслідувань.

Реалістичний приклад: як має виглядати добре захищений малий та середній бізнес

Уявіть собі компанію з 80–200 співробітниками, головним офісом, філією та частково віддаленою роботою. Мета полягає не в тому, щоб «захистити все» на 100%, а в тому, щоб різко зменшити ризики та вплив на операційну діяльність.

– Зони: користувачі, сервери, VoIP, гості, Інтернет речей, управління IT.

– NGFW: контроль програм/користувачів + веб/DNS-фільтрація + вибіркова перевірка TLS.

– IPS: прогресивне налаштування, профілактика лише там, де це додає цінності, не порушуючи бізнес.

– VPN типу «сайт-сайт»: лише необхідні підмережі, міжзональний трафік контролюється правилами та журналами.

– Віддалена VPN: обов'язкова багатофакторна автентифікація (MFA), профілі на основі ролей та мінімальний доступ.

– Операції: контроль змін + щоквартальний перегляд правил + щомісячна звітність.

Така конструкція зменшує горизонтальне переміщення (ключовий фактор у вимагацьських програмах) та гарантує, що у разі чогось ви зможете швидко відреагувати, оскільки у вас є видимість та можливість відстеження.

3.2 Розробка структурної схеми

Шлюз L7

Шлюз L7 працює на рівні додатків, виконуючи маршрутизацію з урахуванням контенту, балансування навантаження та функції безпеки для HTTP(S)-трафіку, підвищуючи продуктивність та захист веб-додатків.

Шлюз L7 – це мережевий пристрій, який працює на рівні OSI 7 для маршрутизації та керування HTTP- та HTTPS-запитами на основі контенту, такого як URL-шляхи, заголовки або файли cookie, що забезпечує детальний контроль трафіку та ефективну доставку програм.

Серед спільних функцій – термінація SSL/TLS для розвантаження шифрування, маршрутизація на основі URL-адрес, кешування контенту та можливості брандмауера веб-застосунків для перевірки та фільтрації трафіку на рівні застосунків. Поєднуючи безпеку та балансування навантаження, шлюз централізує політики та підвищує стійкість веб-сервісів.

Під час розгортання адміністратори налаштовують правила маршрутизації, завантажують SSL-сертифікати та визначають політики безпеки. Шлюзи L7 інтегруються з виявленням служб та виконують перевірки справності для динамічного налаштування пулів серверної частини. Правильне керування сертифікатами та організація правил є критично важливими для уникнення неправильних конфігурацій та підтримки безпеки та доступності програм.

Розширені можливості Firewall 7-го рівня

Firewall 7-го рівня, також відомі як Firewall програм, долають ці обмеження, додаючи можливості сеансів та розуміючи потоки даних від рівня 1 до рівня 7. Це дозволяє їм обробляти сеанс зв'язку цілісно, зменшуючи адміністративні накладні витрати та забезпечуючи більш детальні політики безпеки.

Основні характеристики Firewall рівня 7

Обізнаність про сеанс:

– Firewall 7-го рівня розглядають весь сеанс зв'язку як єдине ціле. Це означає, що трафік запитів і відповідей розпізнається як частина одного сеансу, що дозволяє ефективніше та контекстуальніше керувати безпекою.

– Наприклад, трафік відповідей може оброблятися по-різному, коли його розпізнають як частину початкового запиту, що підвищує релевантність і точність політик безпеки.

Комплексна перевірка даних:

– На відміну від традиційних Firewall, Firewall 7-го рівня можуть перевіряти дані на рівні додатків. Вони розуміють структуру та вміст різних протоколів 7-го рівня, таких як HTTP та HTTPS.

– Це дозволяє їм виявляти нормальні та аномальні елементи в потоці зв'язку, забезпечуючи надійний захист від атак та вразливостей, пов'язаних з певним протоколом.

Розширене розуміння протоколу:

– Firewall 7-го рівня можуть розривати HTTPS-з'єднання, видаляючи шифрування для перевірки базових HTTP-даних. Потім вони створюють нове HTTPS-з'єднання з внутрішнім сервером, забезпечуючи прозору перевірку даних як для клієнта, так і для сервера.

– Ця функція дозволяє брандмауеру аналізувати, блокувати, замінювати або позначати дані 7-го рівня, захищаючи від таких загроз, як шкідливе програмне забезпечення, спам і неприйнятний контент.

Детальний контроль вмісту:

– Маючи доступ до прикладного рівня, Firewall 7-го рівня можуть застосовувати вузькоспеціалізовані політики щодо контенту. Наприклад, вони можуть дозволяти зображення котів, блокуючи зображення собак, або замінювати контент для дорослих нешкідливими зображеннями.

– Ці Firewall також можуть блокувати певні програми, такі як Facebook, або запобігати завантаженню конфіденційних бізнес-даних до таких сервісів, як Dropbox.

Повний набір функцій:

– Firewall 7-го рівня зберігають усі функціональні можливості Firewall 3-го, 4-го та 5-го рівнів, але додають можливості реагування на елементи протоколів 7-го рівня. Це включає перевірку DNS-імен, керування швидкістю з'єднань та аналіз заголовків і контенту.

– Вони надають мережевим адміністраторам гнучкий та потужний інструмент для захисту своїх середовищ від широкого спектру загроз.

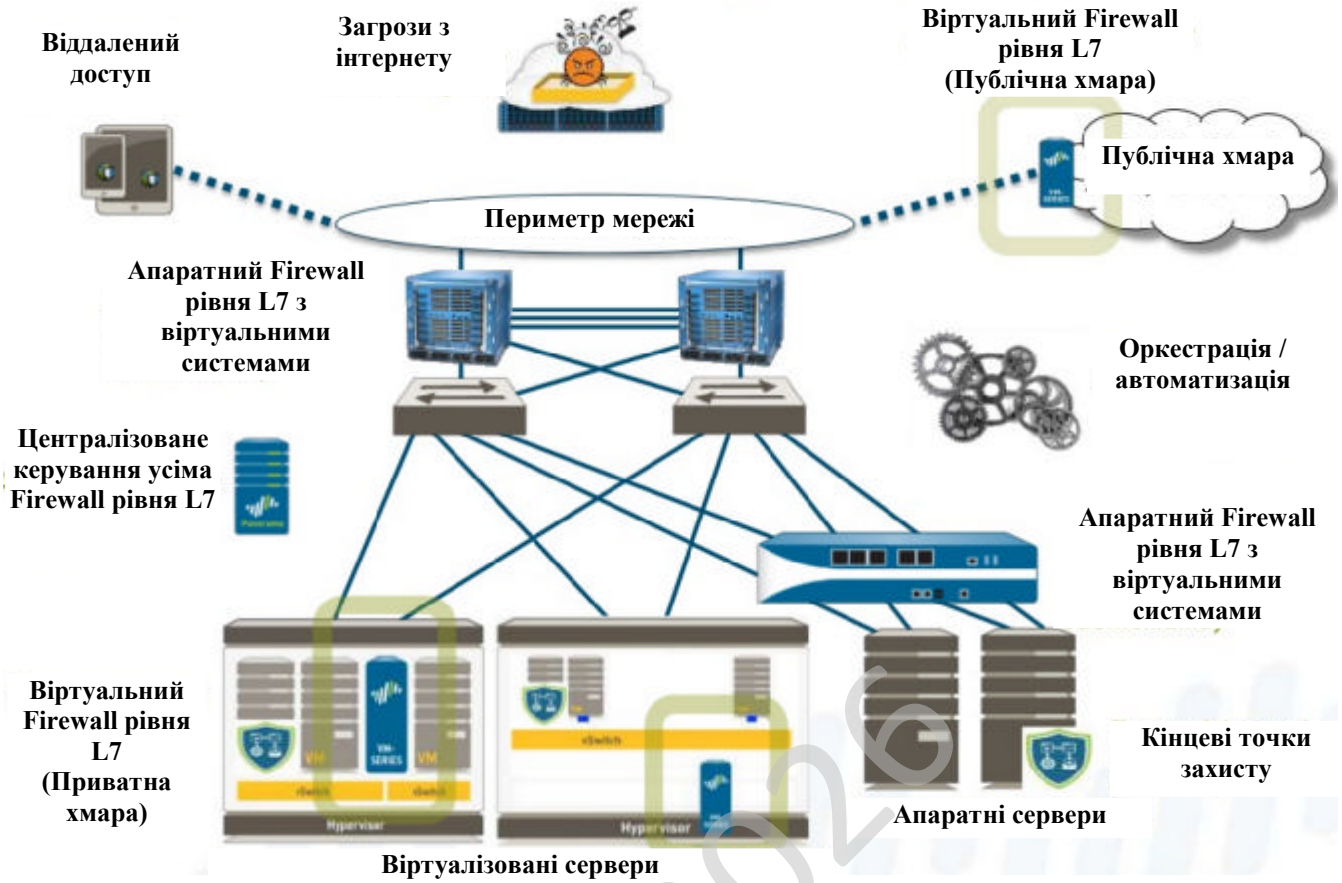

Рисунок 3.1 – Структурна схема системи

Приклади функціональності брандмауера 7-го рівня

Фільтрація контенту:

– Брандмауер 7-го рівня може перевіряти HTTP-заголовки та вміст тіла, що дозволяє адміністраторам фільтрувати спам, контент не за темою або небезпечне шкідливе програмне забезпечення. Це має вирішальне значення для підтримки цілісності та безпеки веб-застосунків.

Контроль застосунків:

– Розуміючи такі протоколи, як HTTP та SMTP, ці Firewall можуть блокувати або дозволяти певні програми. Наприклад, вони можуть забороняти доступ до сайтів соціальних мереж у робочий час або блокувати певні типи файлів у вкладеннях електронної пошти.

Прозора перевірка даних:

– Коли клієнт підключається до сервера через HTTPS, брандмауер 7-го рівня може розшифрувати з'єднання, перевірити трафік, а потім повторно зашифрувати його перед пересиланням на сервер. Цей процес гарантує, що шкідливий контент буде виявлено та усунено до того, як він потрапить у внутрішню мережу.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. У багатьох пристроях для домашніх мереж, наприклад інтегрованих маршрутизаторах, часто є багатофункціональні програмні міжмережні екрани, побудовані за технологією NGFW. Такі міжмережні екрани, побудовані за технологією NGFW звичайно реалізують трансляцію мережних адрес (NAT), динамічний аналіз пакетів (SPI), а також фільтрацію по IP-адресах, додатках і веб-сайтах. Додатково вони підтримують функції DMZ.

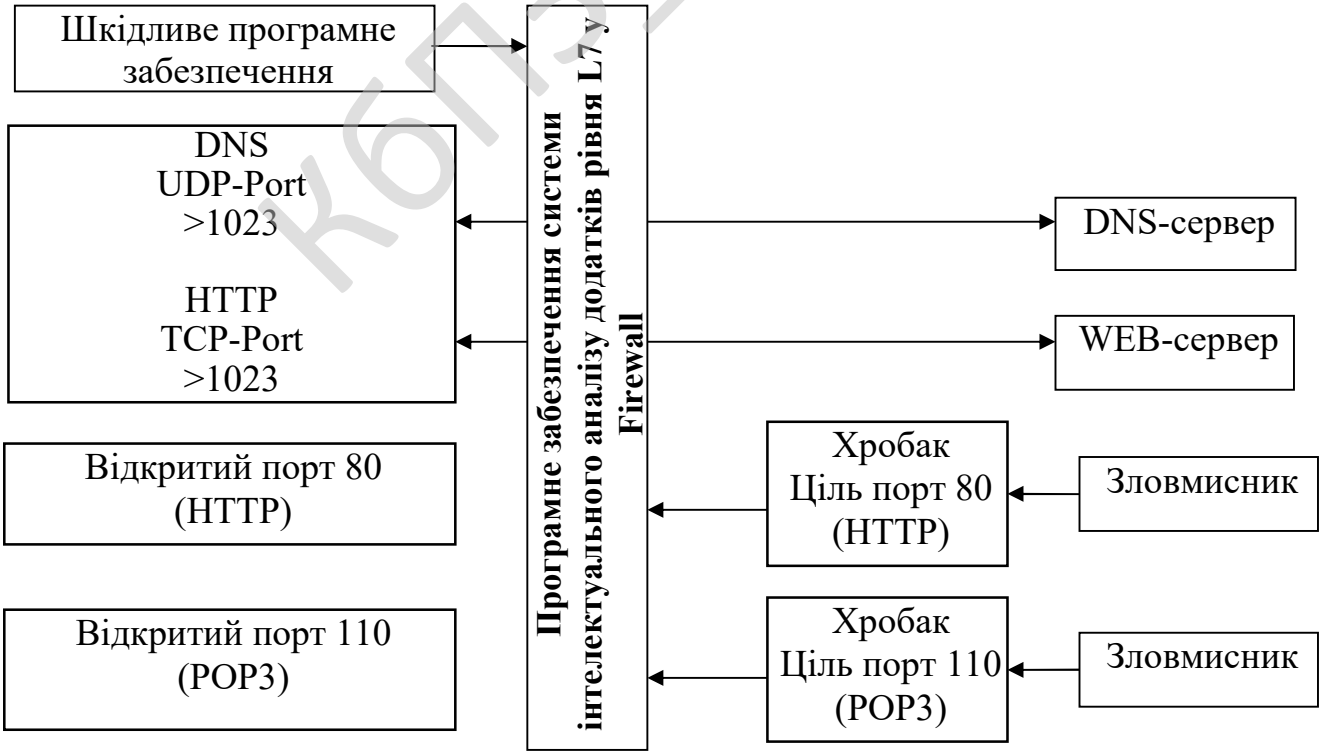

Рисунок 3.2 – Функціональна схема системи

Інтегрований маршрутизатор дозволяє настроїти примітивну DMZ для доступу до внутрішнього сервера з вузлів за межами мережі. Для цього сервер повинен мати статичну IP-адресу, що вказується в конфігурації DMZ. Інтегрований маршрутизатор ізолює трафік, що пересилається на зазначений IP-адрес. Цей трафік пропускається тільки на той порт комутатора, до якого підключений сервер. На всі інші вузли як і раніше поширюється захист міжмережного екрану.

При активації DMZ у найпростішому виді зовнішні вузли одержують доступ до всіх портів сервера, наприклад 80 (HTTP), 21 (FTP) і 110 (POP3 для електронної пошти).

Функція переадресації портів дозволяє настроїти більш строгу конфігурацію DMZ. У цьому випадку вказуються порти, які повинні бути доступні на сервері. Пропускається тільки трафік, спрямований на ці порти. Весь інший трафік виключається.

Бездротова точка доступу в складі інтегрованого маршрутизатора часто вважається частиною внутрішньої мережі. Необхідно усвідомлювати, що при роботі точки доступу в незахищеному режимі всі користувачі, що підключилися до неї, одержують доступ до внутрішньої захищеної мережі без проходження міжмережного екрану. Зловмисники можуть у такий спосіб одержати доступ до внутрішньої мережі, минаючи всі засоби захисту.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути

представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Рисунок 3.3 – Діаграма взаємодії процесів

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію бакалаврської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Розглянемо підпрограму запуску файрволу. Після того як створені чи відкриті правила фільтрації можна запустити файрвол. Після запуску файрволу, програма починає фільтрувати пакети по вказаним правилам.

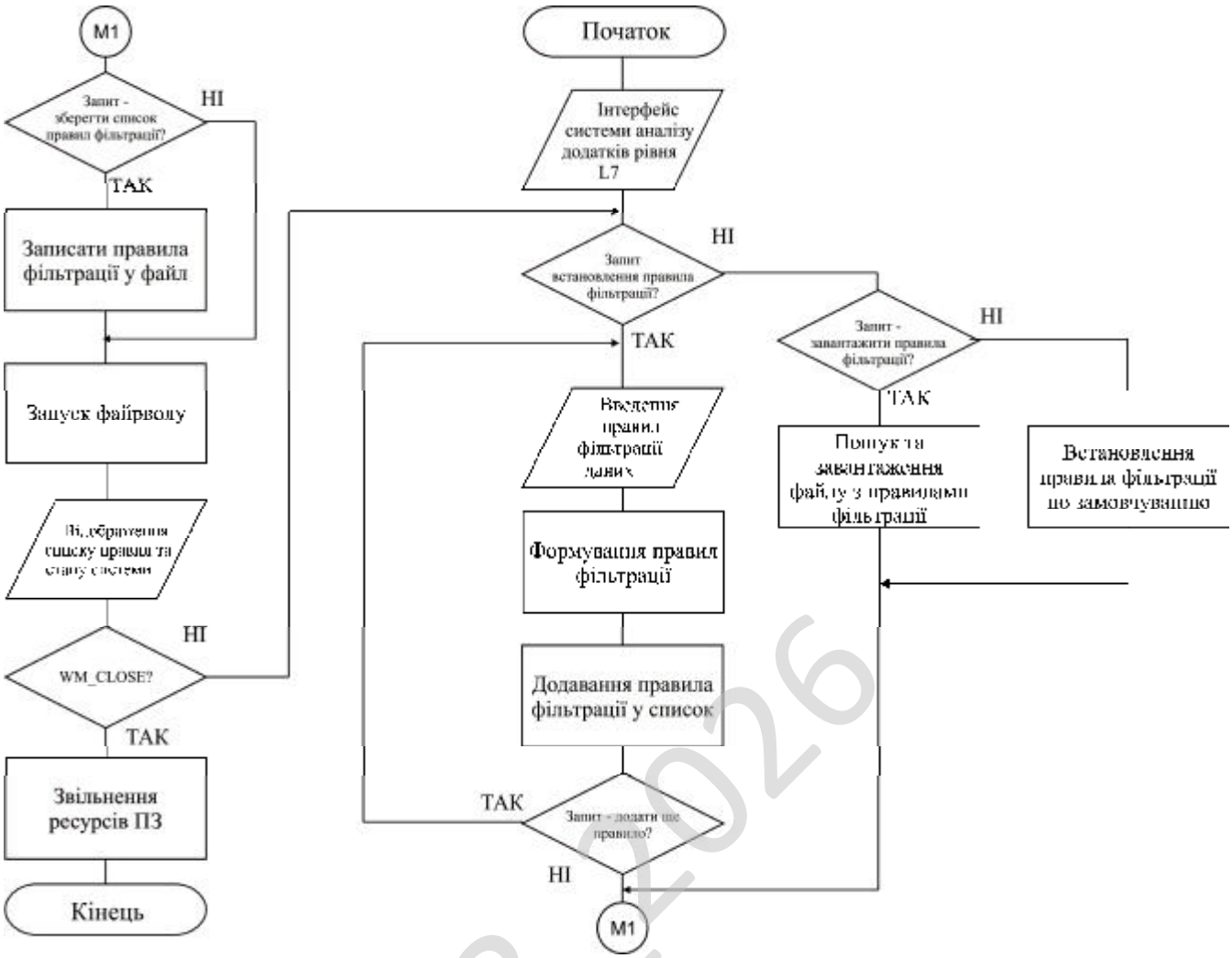

Рисунок 4.1 – Блок-схема основної програми

Розглянемо код.

```
void CMainFrame::OnButtonstart()
{
   CFirewallAppDoc *doc = (CFirewallAppDoc *)GetActiveDocument();
   unsigned int i;
   DWORD result;
   PIP_ADAPTER_INFO pAdapterInfo = NULL, aux;
   IP_ADDR_STRING *localIp;
   unsigned long len = 0;
// Пошук адаптера мережної карти
   GetAdaptersInfo(pAdapterInfo, &len);
   pAdapterInfo = (PIP_ADAPTER_INFO) malloc (len);
   result = GetAdaptersInfo(pAdapterInfo, &len);
   if(result != ERROR_SUCCESS)
```

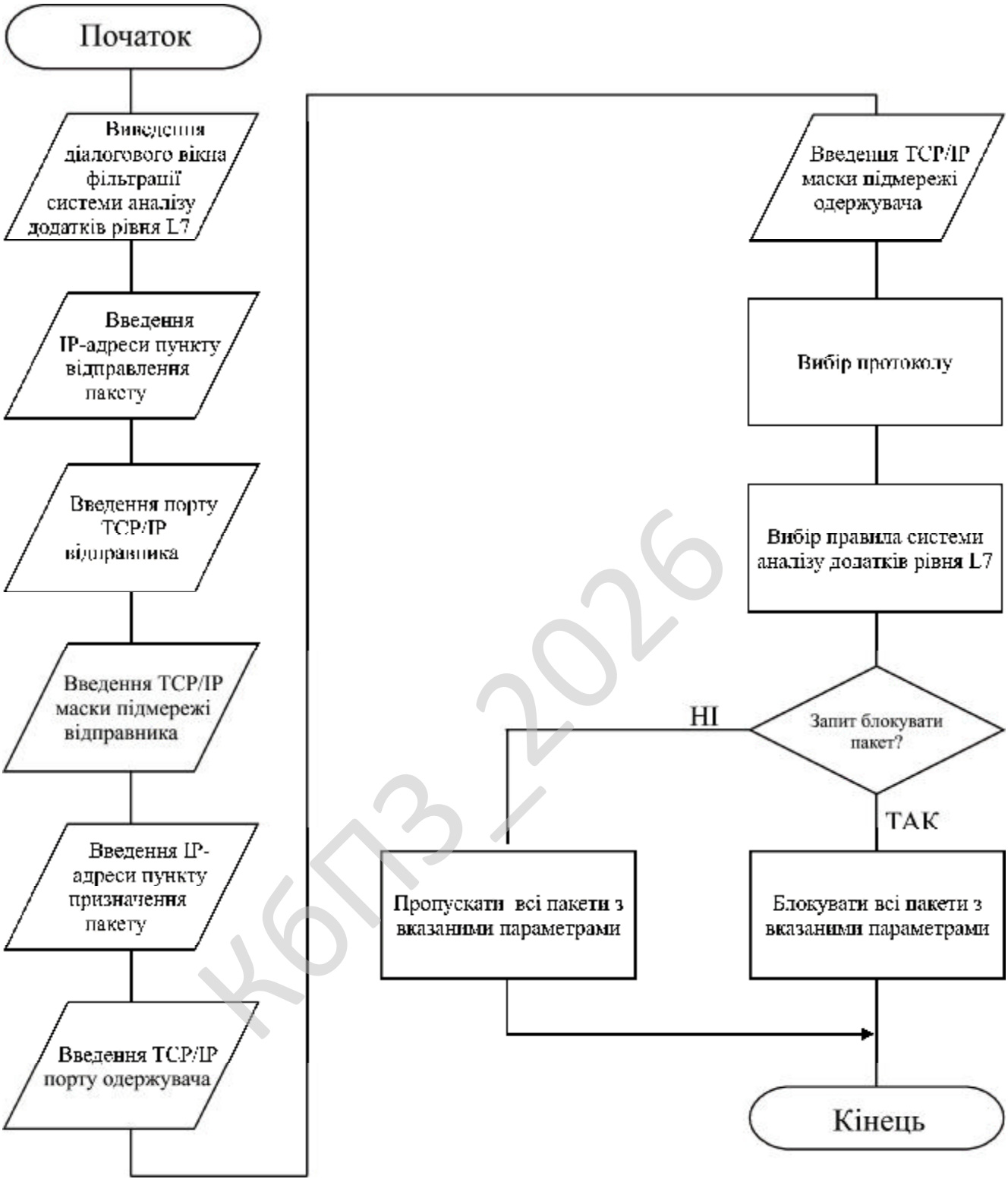

Рисунок 4.2 – Блок-схема роботи підпрограми

```
        {
             AfxMessageBox("Error getting adapters info.");
             return;
        }
      // Посилка правил на інтерфейс адаптера
        for(i=0;i<doc->nRules;i++)
        {
      // на всі знайдені адаптери
             for(aux=pAdapterInfo;aux != NULL;aux=aux->Next)
             {
      // на кожний IP адаптера
                  for(localIp=&aux->IpAddressList;localIp!=NULL;localIp=localIp-
>Next)
                  {
                  pckFilter.AddF(CharToIp(localIp->IpAddress.String),
        ANY_DIRECTION,
                              doc->rules[i].sourceIp,
                              doc->rules[i].sourceMask,
                              doc->rules[i].destinationIp,
                              doc->rules[i].destinationMask,
                              doc->rules[i].sourcePort,
                              doc->rules[i].destinationPort,
                              doc->rules[i].protocol);
                  }
             }
        }
        started = TRUE;
      }
```

Для припинення фільтрації пакетів слід зупинити файрвол. Підпрограма зупинки файрволу виглядає наступним чином:

```
      void CMainFrame::OnButtonstop()
      {
        pckFilter.RemoveAll();
        started = FALSE;
      }
```

Після завантаження системи відбувається вивід основного вікна програми. Розглянемо код підпрограми виводу головного вікна:

```
      //ініціалізація й створення вікна і його компонентів
      int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
      {
```

```
        //створення вікна
          if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
               return -1;
        //створення ToolBar
if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP
        | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
          !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
          {
               TRACE0("Failed to create toolbar\n");
               return -1;
        // помилка створення
          }
        //створення StatusBar
        if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT))
             )
          {
               TRACE0("Failed to create status bar\n");
               return -1;      // fail to create
          }
          m_wndToolBar.EnableDocking(CBRS_ALIGN_RIGHT);
          EnableDocking(CBRS_ALIGN_ANY);
          DockControlBar(&m_wndToolBar);
        // Установка заголовка
          this->SetWindowText("Firewall");
          return 0;
        }
```

Потім користувач або завантажує попередньо збережений файл з правилами фільтрації, або вводить правила вручну, чи використовує значення за замовчуванням.

Після створення списку правил, його можна зберегти у файлі, для подальшого використання. Підпрограма збереження правил у файлі:

```
        void CMainFrame::OnSRules()
        {
          CFirewallAppDoc *doc = (CFirewallAppDoc *)GetActiveDocument();
          if(doc->nRules == 0)
          {
               AfxMessageBox("There isnt Rules to Save.");
               return;
          }
```

```
        CFileDialog dg(FALSE, NULL, NULL, OFN_HIDEREADONLY |
OFN_CREATEPROMPT,"Rule Files(*.rul)|*.rul|all(*.*)|*.*||", NULL);
        if(dg.DoModal()==IDCANCEL)
             return;

        CString nf=dg.GetPathName();
        if(nf.GetLength() == 0)
        {
             AfxMessageBox("Це ім’я не існує");
             return;
        }
        CFile file;
        CFileException e;
        if( !file.Open( nf, CFile::modeCreate | CFile::modeWrite, &e ) )
        {
             AfxMessageBox("Error openning the file.");
             return;
        }
        PFFORWARD_ACTION action = pckFilter.GetDefaultAction();
        file.Write(&action, sizeof(PFFORWARD_ACTION));
        unsigned int i;
             for(i=0;i<doc->nRules;i++)
        {
             file.Write(&doc->rules[i], sizeof(RuleInfo));
        }
        file.Close();
      }
```

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

Було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу).

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими

словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так

званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики XP зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Технологія екстремального програмування була розроблена Кентом Беком, Уардом Каннінґхемом та Роном Джеффріесом під час роботи над Chrysler Comprehensive Compensation System (C3). У 1996 Кент Бек став лідером проекту і почав вдосконалювати методи розробки, що застосовувалися в роботі над проектом. Свій метод він виклав у книзі «Extreme Programming Explained», котру було видано у жовтні 1999. Після купівлі Крайслера компанією Даймлер–Бенц проект C3 було скасовано у лютому 2000.

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій".

Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану (традиційна особливість для програмного забезпечення, що проектується).

XP використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

XP намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає

зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн.

Написання коду. Прихильники XP заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми XP. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

1.3. Замовник завжди поруч (Whole team, Onsite customer).

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

3.3 Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).

3.4 Стандарт кодування (Coding standard or Coding conventions).

4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі-системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – Фотографіями, скриншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настроюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором

статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи українську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи – плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько $50 за користувача для локального і $7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

– проект використовує ліцензії, схвалені Open Source Initiative;

– вихідний код проекту доступний для скачування;

– у проекту є публічно доступна веб-сайт;

– програмне забезпечення від Atlassian є на веб-сайті проекту.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму FEAL – блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті.

У ньому використовуються 64-бітовий блок і 64-бітовий ключ. Його ідея полягає і в тому, щоб створити алгоритм, подібний DES, але з більш сильною функцією етапу. Використовуючи менше етапів, цей алгоритм міг би працювати швидше. На жаль, дійсність виявилася далекою від цілей проекту.

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції F[1] з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку зображено розроблене у бакалаврської дипломної роботі ПЗ системи аналізу додатків рівня L7 у Firewall. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Функціональних кнопок ПЗ.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Верхнього меню: Файл; Налаштування; Список блоків ПЗ; Довідка.

– Розділу виведення результату роботи системи.

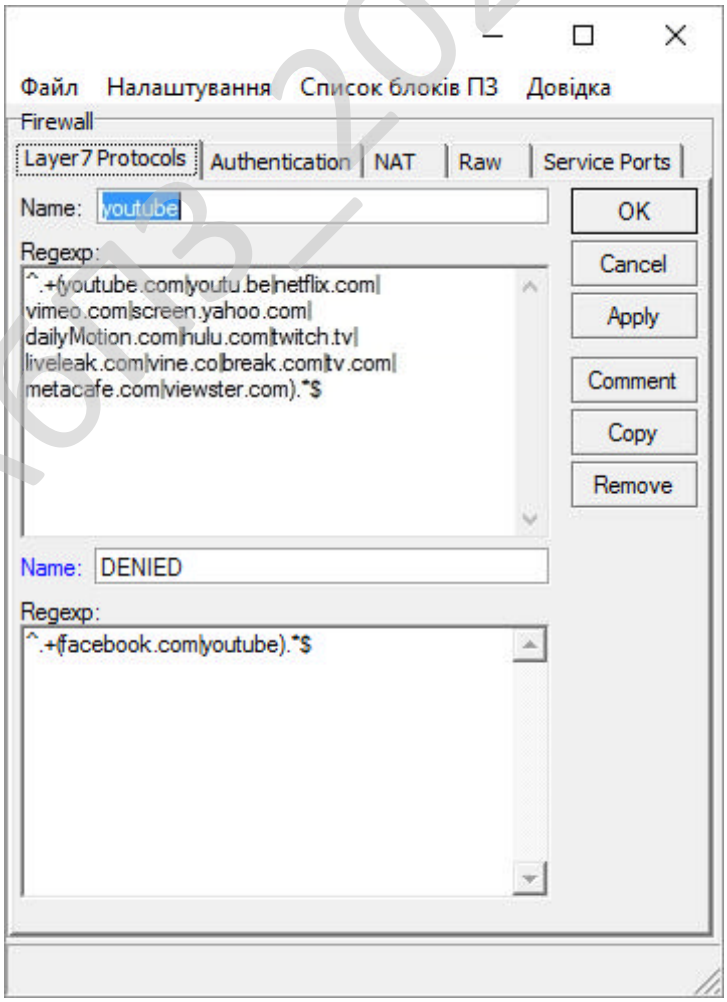

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

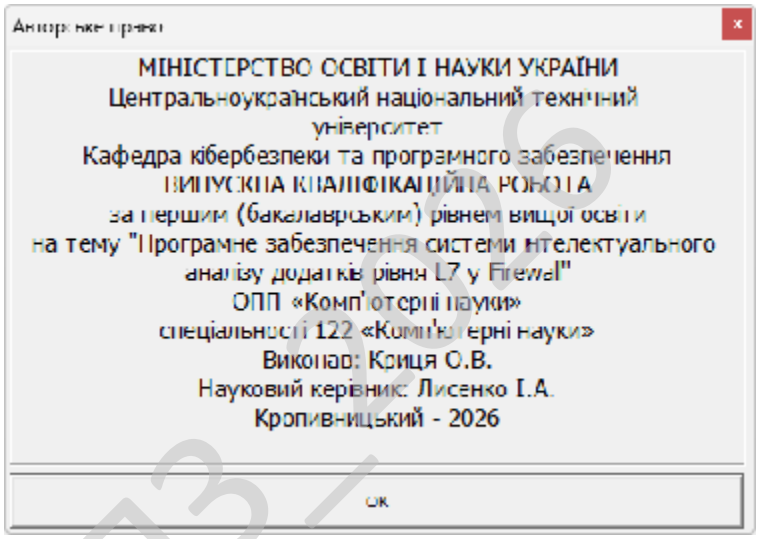

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є

унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються

в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy).

ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ).

Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи інтелектуального аналізу додатків рівня L7 у Firewall.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем інтелектуального аналізу додатків рівня L7 у Firewall.

– Досліджена система інтелектуального аналізу додатків рівня L7 у Firewall.

– На основі отриманих результатів досліджень створена програмна реалізація системи інтелектуального аналізу додатків рівня L7 у Firewall.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання інтелектуального аналізу додатків рівня L7 у Firewall.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Visual C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи інтелектуального аналізу додатків рівня L7 у Firewall. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на

загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм FEAL.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

2. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

3. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

4. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

5. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

6. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

7. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

8. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

9. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

10. Vijay Kumar Velu. Mastering Kali Linux for Advanced Penetration Testing. Packt Publishing Ltd. 2022. 573 p.

11. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing,* 2025. pp. 21-61.

12. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник / Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А, Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко

В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.– К.: Видавничий центр “Кафедра”, 2025.– 320 с.

13. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка.* 2025. Том 1 № 29. С.704–716, 2025

14. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

15. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

16. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

17. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

18. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

19. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

20. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

21. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

22. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

23. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

24. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

25. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

26. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

27. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

28. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

29. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

30. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

31. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

32. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

33. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя*

кафедри кібербезпеки та програмного забезпечення, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

34. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

35. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

36. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

37. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

38. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

39. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

40. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

41. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

42. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

43. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

44. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

45. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

46. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

47. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

48. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

49. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

50. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

51. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.