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
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки пірінгових мереж”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Волошин Є.О.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2025 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 *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

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

Волошину Єгору Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки пірінгових мереж*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 51-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки пірінгових мереж*

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 кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2025 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	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Волошин Є.О.
(прізвище та ініціали)

АНОТАЦІЯ

Волошин Є.О. Програмне забезпечення системи кібербезпеки пірінгових мереж. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки пірінгових мереж.

Метою розробки є програмне забезпечення системи кібербезпеки пірінгових мереж.

Результат роботи – програмна реалізація системи кібербезпеки пірінгових мереж.

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: кібербезпека, пірінгові мережі

ABSTRACT

Voloshyn E.O. Software for the cybersecurity system of peer-to-peer networks. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the cybersecurity system of peer-to-peer networks.

The purpose of the development is the software for the cybersecurity system of peer-to-peer networks.

The result of the work is the software implementation of the cybersecurity system of peer-to-peer networks.

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 Windows 10/11.

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

Keywords: cybersecurity, peer-to-peer networks

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. Однорангова мережа (або пірінгова, від англ. Peer-to-Peer, P2P – рівний до рівного) – це логічна комп'ютерна мережа, створювана поверх іншої комп'ютерної мережі, у якій всі учасники мають рівні права й функції. Звичайно в такій мережі відсутнє явне розмежування між клієнтом і сервером, а кожний мережний вузол виконує функції як клієнта, так і сервера.

Зростання рівноправних розподілених обчислень (P2P) можна віднести до ключових досягнень в інформаційних технологіях, які за своєю суттю змінили спосіб використання зв’язку та мереж для обміну та представлення даних в Інтернеті. У той час як традиційна модель доставки вмісту через Інтернет включає централізований сервер, який обмінюється даними з машинами, що запитують, системи P2P покладаються на різноманітні машини, які надають свої відповідні обчислювальні ресурси децентралізованим способом для виконання функцій, специфічних для програми.

Популярність програм P2P зросла на початку 2000-х із поширенням програмного забезпечення для обміну файлами, наприклад BitTorrent, яке все ще широко використовується для незаконного розповсюдження носіїв, захищених авторським правом. Хоча піратство є серйозною проблемою безпеки, яка сильно впливає на конфіденційність інтелектуальної власності, це, можливо, вважається прийнятним ризиком для програм для обміну файлами P2P, оскільки вони не несуть прямої відповідальності за зберігання чи поширення будь-якого піратського вмісту. Програми обміну файлами були лише початком динамічних програм розподілених і децентралізованих мереж. Деякі школи вважають, що природа та функції програм P2P представляють звільнення Інтернету, перехід від неавторитарної моделі до більш гнучкої моделі з необмеженою масштабованістю, яка покладається на вузли-учасники (комп’ютери) у розподіленій мережі. Децентралізовані програми (dApps) представляють одне з найсучасніших

застосувань розподілених обчислень P2P. Додатки dApps характеризуються своїм існуванням у блокчейні криптовалюти, як мережа Ethereum Layer-2, і зазвичай представляють передбачуваний перехід від відносно традиційного Web 2.0 до децентралізованого та відкритого Web 3.0. Дослідження, представлене в цьому документі, намагається вивчити компоненти та функції P2P-додатків, визнаючи структуровану, неструктуровану або гібридну мережеву архітектуру, на якій вони побудовані, і виявляючи критичні вразливості безпеки, якими можуть скористатися зловмисники, що призведе до ефективного порушення конфіденційності, цілісності та/або доступності даних користувача та децентралізованого сервісу.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки пірінгових мереж.

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

– Огляд існуючих систем пірінгових мереж.

– Дослідження системи кібербезпеки пірінгових мереж.

– Програмна реалізація системи кібербезпеки пірінгових мереж.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі пірінгових мереж.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки пірінгових мереж, 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

Децентралізований файлообмін є класичною сферою застосування однорангових мереж, оскільки обмін більшими файлами вимагає гарної масштабованості, що досягається за рахунок прямого обміну даними. Найпоширенішим протоколом для цієї мети є BitTorrent. Відповідно до актуальних досліджень, на нього доводиться від 20 до 33% усього интернет-трафіку.

Сучасні засоби забезпечення анонімності в Інтернеті, як правило, будуються на основі однорангових мереж. Ця проблема викликає великий інтерес у науковому середовищі, добре вивчена, і, як результат, зараз засобу анонімізації є для самих різних прикладних завдань. Найбільшу популярність одержали низьколатентні анонімні мережі. Дві самі великі з них – Tor і I2P, при цьому в першій вже понад 2 млн користувачів.

Щодо новою областю застосування однорангових мереж стали криптовалюти. В 2009 році була представлена цифрова валюта Bitcoin, творцям якої вдалося дозволити головні труднощі при побудові криптовалют на основі однорангової мережі – проблему повторної витрати засобів. Вона була вирішена за допомогою створення єдиного ланцюжка транзакцій, для захисту якої використовується система доказу виконаної обчислювальної роботи (proof-of-work). Незважаючи на існування безлічі аналогів, заснованих на тих же принципах, але які відміняються деталями, Bitcoin залишається самої популярною криптовалютою, а її капіталізація становить 3 млрд доларів.

Описані вище переваги пірінгових мереж – лише одна сторона медалі. Необхідно ще забезпечити їхня безпека. У строгій одноранговій мережі (де немає централізованих сервісів) кожний користувач – потенційний порушник. Довіряти

не можна нікому. У таких умовах забезпечення безпеки будь-якого роду представляється дуже складним завданням.

1.2 Область застосування

Розроблювальна система призначена для безпеки пірінгових мереж. У дипломному проекті пропонується у якості такого захисту використовувати файрвол (fire wall).

Персональний файрвол ("персональний брендмауеэр") – додаток, що виконує роль міжмережного екрана для окремого комп'ютера (звичайно персонального), запущений на цьому ж самому комп'ютері.

Більша частина функцій персонального файрволу дублює функції міжмережного екрана, однак персональний файрвол так само може забезпечувати додаткові можливості:

– Контроль за додатками, що використовують порти. На відміну від звичайних міжмережних екранів, персональний файрвол може визначати не тільки використовуваний протокол і адреси, але й точну назву додатка, що запитує з'єднання (або намагається слухати на якімсь порту), зокрема, можливий контроль за незмінністю додатка (у випадку зміни додатка вірусами або троянами, додаток, що встановлюються в якості плагінів, блокується).

– Призначення роздільних правил різним користувачам без додаткової мережної авторизації.

– Режим навчання, коли при першому звернені програми до мережних ресурсів користувачеві видається запит (звичайно виду «заборонити завжди, заборонити однократно, завжди дозволити, дозволити однократно, створити правило»).

– Режим змішаної фільтрації (при якій перевіряються різні параметри на різних рівнях мережних протоколів – від другого (перевірка на фальсифікацію

MAC-адреси) до 4 (фільтрація портів), і навіть вищестоящих рівнів (фільтрація вмісту веб-сайтів, перевірка пошти, відсівання спаму).

При цьому персональний файрвол звичайно не призначений для використання в якості "міжмережного екрана" і не може здійснювати фільтрацію маршрутизуємих і/або трансльованих пакетів, фільтрувати пакети на основі адреси відправника, використовувати різні правила для різних мережних інтерфейсів (звичайно в рамках моделі персонального файрволу вважається, що в комп'ютера єдина IP-адреса і єдиний зовнішній мережний інтерфейс).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки пірінгових мереж, 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

Розглянемо конкретні приклади атак на TCP/IP.

Детектування й захист

Найпростішим сигналом IP-spoofing будуть служити пакети із внутрішніми адресами, що прийшли із зовнішнього миру. Програмне забезпечення маршрутизатора може попередити про це адміністратора. Однак не варто зваблюватися – атака може бути й зсередини Вашої мережі.

У випадку використання більш інтелектуальних засобів контролю за мережею адміністратор може відслідковувати (в автоматичному режимі) пакети від систем, які перебувають у недоступному стані. Втім, що заважає зловмиснику імітувати роботу системи В відповіддю на ICMP-пакети?

Які способи існують для захисту від IP-spoofing? По-перше, можна ускладнити або унеможливити вгадування порядкового номера (ключовий елемент атаки). Наприклад, можна збільшити швидкість зміни порядкового номера на сервері або вибирати коефіцієнт збільшення порядкового номера випадково (бажано, використовуючи для генерації випадкових чисел криптографічно стійкий алгоритм).

Якщо мережа використовує міжмережний екран firewall (або інший фільтр IP-пакетів), варто додати йому правила, по яких всі пакети, що прийшли ззовні й мають зворотними адресами з нашого адресного простору, не повинні пропускатися усередину мережі. Крім того, варто мінімізувати довіра машин один одному. В ідеалі не повинні існувати способу, прямо потрапити на сусідню машину мережі, діставши права суперкористувача на одній з них. Звичайно, це не

врятує від використання сервісів, не потребуючі авторизації, наприклад, IRC – чат (зловмисник може прикинутися довільною машиною Internet і передати набір команд для входу на канал IRC, видачі довільних повідомлень і т.д.).

Шифрування TCP/IP-потоку вирішує в загальному випадку проблему IP-спуфінга (за умови, що використовуються криптографічно стійкі алгоритми).

Для того, щоб зменшити число таких атак, рекомендується також настроїти firewall для фільтрації пакетів, посланих нашою мережею назовні, але адреси, що має, не приналежному нашому адресному простору. Це захистить мир від атак із внутрішньої мережі, крім того, детектування подібних пакетів буде означати порушення внутрішньої безпеки й може допомогти адміністраторові в роботі.

IP Hijacking – Напад на IP

Якщо в попередньому випадку зловмисник ініціював нове з'єднання, то в цьому випадку він перехоплює весь мережний потік, модифікуючи його й фільтруючи довільним образом. Метод є комбінацією 'підслуховування' і IP-спуфінгу.

Таким чином розглянувши усі види атак на IP-мережу, зробимо висновок, що застосування файрволу у значній мірі протидіє цим атакам, тому у подальшому ми розглянемо файрвол.

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 додатків 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 додатків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки додатків Windows дозволяють автоматизувати процес створення додатка. Для цього використовуються генератори додатків. Програміст відповідає на питання генератора додатків і визначає властивості додатка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор додатків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої додатки. Подібні засоби автоматизованого створення додатків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики додатка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні додатки, а також додатки, 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 додатка програмістові прийдеться розробляти самостійно. Вихідний текст додатка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон додатка – це вже половина всієї роботи. Вихідні тексти додатків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти додатків 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 додатки 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-додатки 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-додатків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованого 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 додаток 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 кібербезпеки пірінгових мереж.

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 кібербезпеки 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 кібербезпеки 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

Проблеми починаються вже на першому кроці при спробі досягнення стандартних цілей безпеки. Як робити автентифікацію в строгій P2P-системі (без глобального центра сертифікації)? Теоретичні вишукування приводять до невтішного виводу: на сьогоднішній день не існує способу реалізації надійної внутрісистемної автентифікації без залучення довіреної третьої сторони й без попередніх контактів між користувачами по інших каналах.

Інша невирішена проблема безпеки однорангових мереж – так звана Sybil-атака. Вона заснована на тому, що зловмисник додає в мережу свої вузли неодноразово – щораз із новим ідентифікатором. Ця атака є допоміжною для проведення цілого спектра інших атак і, таким чином, займає ключове місце в питаннях безпеки. Класичний метод протидії – введення обмеження по тим або інших ресурсах. На жаль, застосування даного методу найчастіше вимагає дотримання нездійсненних умов, і поки немає іншого способу повністю виключити Sybil-атаку, крім використання централізованого сервісу.

Проблематика однорангових мереж дуже широка й не обмежується наведеними прикладами. Так, згадані вище Tor і інші низьколатентні анонімні мережі залишаються уразливими для атак, заснованих на кореляції тимчасових затримок (timing-атаки). Проведення цих атак вимагає колосальних ресурсів (глобального прослуховування мережі), однак цілком під силу, наприклад, спецслужбам.

Погроза Sybil-атаки поряд із проблемою автентифікації нерідко змушує розроблювачів однорангових мереж прибігати до використання централізованих сервісів, що виконують обмежений спектр ключових функцій. Це приводить до

появи точок загальносистемної відмови в мережі й частково нівелює основні переваги пірінгових мереж.

Таким чином, будучи бурхливо розвиваємим й, безумовно, перспективним напрямком інформатики, однорангові мережі залишаються областю з безліччю невирішених проблем, відкритою для нових ідей і привабливої для гострих розумів.

3.2 Розробка структурної схеми

Традиційні програми зазвичай надають вміст і інформацію на основі конкретного запиту від кінцевого користувача. Вони часто розміщуються на центральному сервері з повноваженнями контролювати та керувати безпекою системи та надавати гарантії щодо конфіденційності особистої інформації, яку вони можуть зберігати та обробляти. Розглядаючи типовий веб-сайт як приклад, відвідувачі отримують доступ до веб-вмісту, надсилаючи запит HTTPGET на виділений веб-сервер, який має повний контроль над схваленням або відхиленням запиту на підключення. Після застосування відповідних заходів безпеки веб-сервер може встановити довіру з хостом, який запитує, на рівні Інтернету та додатків і продовжити показ веб-вмісту, який запитується, у віддаленому браузері відвідувача. Деякі загальні заходи безпеки включають встановлення захищеного сеансу через SSL і підтримку чорного списку IP-адрес для запобігання відомим запитам зловмисників.

Навпаки, програми P2P характеризуються трьома основними компонентами. Компоненти визначаються таким чином:

1. Сервер входу в систему – зазвичай це точка доступу служби P2P, яка забезпечує платформу для підключення користувачів і участі в децентралізованій мережі, виконання конкретних завдань, визначених програмним забезпеченням, як-от обмін файлами.

2. Трекер – це окремий сервер, який діє як одноранговий каталог,

зберігаючи записи про доступні вузли в розподіленій мережі, активно ділячись їхньою доступністю з новими одноранговими вузлами, які підключаються через сервер входу.

3. Однорангові вузли – цей компонент представляє вузли, що не залежать від розташування, з усього світу, які беруть участь у децентралізованій мережі, вони «є автономними клієнтськими машинами, які приєднуються до системи та залишають її за бажанням.

Існує кілька переваг безпеки, притаманних децентралізованій мережі, одна з яких полягає в підвищених труднощах, з якими зіткнеться зловмисник, щоб успішно демонтувати функціональність програми P2P. Хоча централізована мережа Сервер можна розглядати як координаційну точку компромісу, децентралізована мережа розподіляє ризик між своїми учасниками та забезпечує резервування для підтримки цілісності даних на основі колективного консенсусу. Сучасна демонстрація цих функцій безпеки є неявною в технології блокчейну

Кілька поведінкових функцій є обов'язковими для успішної роботи програми P2P. Розглядаючи мережу блокчейн як приклад, основні функції включають:

1. Виявлення – коли новий вузол приєднується до мережі блокчейну, він підключається до існуючих вузлів за допомогою методів виявлення та ідентифікації, таких як розподілені хеш-таблиці (DHT), початкові вузли та протоколи однорангового обміну.

2. Розташування та отримання даних – ця функція передбачає використання різноманітних ідентифікаторів, щоб гарантувати, що новий вузол у мережі може отримати інформацію, як-от транзакції та коди смарт-контрактів, використовуючи механізми розподіленого зберігання даних.

3. Передача даних – мабуть, найважливіша функція мережі блокчейн, коли учасники ініціюють транзакцію або коли новий блок додається до блокчейну, ці дані поширюються на інші вузли мережі. Піри перевіряють достовірність даних, і після досягнення консенсусу дані постійно записуються в облікову книгу.

Інші внутрішні функції в мережах блокчейнів включають механізми консенсусу, криптографічну безпеку, незмінність даних, смарт-контракти, управління мережею, токенізацію, структури стимулів, конфіденційність і масштабованість рішень, а також децентралізацію.

Три основні мережеві архітектури, навколо яких будуються програми P2P, це структурована, неструктурована та гібридна. Оскільки гібридна інфраструктура є комбінацією структурованих і неструктурованих мережевих топологій, її визначення формує адекватне розуміння того, як розподілені системи спілкуються.

Мережна технологія блокчейн представляє ще одне релевантне дослідження наслідків вибору відповідної мережевої архітектури. Айзенбарт, Чолез і Перрін пропонують програму на основі блокчейну, відповідну з точки зору вартості, продуктивності та простоти розробки. Аналізуючи різні мережеві архітектури, вони дійшли висновку, що гібридний підхід був би найбільш придатним для задоволення визначених вимог. "У випадку системи, яка вимагала б взаємодії з компонентами поза блокчейном, цікаво розробити гібридний підхід: додаток може отримати вигоду від джерела подій, що стосується зберігання станів як послідовності незмінних подій, які можна відтворити в будь-який час. Цей підхід дуже цікавий для нашої системи: список підозрілих вузлів, знайдених нашою системою моніторингу, запускає подію зі смарт-контракту, який незмінно додається до надійного сховища подій.

Популярна блокчейн-мережа, відома як Polygon, використовує гібридну мережеву архітектуру, щоб забезпечити масштабованість і зручність використання мережі Etheruem, одночасно зменшуючи транзакційні витрати, широко відомі як плата за газ. Будучи блокчейном рівня 2 на вершині мережі Ethereum, Polygon збирає транзакції в пакетах і безперешкодно інтегрує його з основною мережею Ethereum після необхідної перевірки консенсусного механізму Proof-of-Stake. Елементи структурованої архітектури на Polygon знаходяться у створенні вузлів Validator, які допомагають захистити мережу.

Участь в екосистемі Polygon в якості вузла Validator вимагає підтримки великої частки в блокчейні, ця інвестиція забезпечує впевненість у тому, що оператор вузла відданий добробуту мережі. Існує механізм заохочення для винагороди валідаторів за їх роль у мережі. Елементи неструктурованої архітектури на Polygon представлені звичайними вузлами, які беруть участь у мережі. Polygon зарекомендував себе як одна із найрентабельніших блокчейн-мереж для роботи, що робить їх ефективною платформою для розробки dApps.

Отже, основні характеристики гібридної мережевої архітектури можна підсумувати як перетин структурованих і неструктурованих топологій. Структуровані вузли, які іноді називають супервузлами, підвищують безпеку та забезпечують цілісність даних, забезпечуючи важливі функції балансування навантаження для інших учасників мережі. Вони часто використовують підхід кільцевої теплової мережі, щоб забезпечити односпрямовану передачу пакетів. Навпаки, неструктуровані вузли не мають чіткої топології, а підключаються до P2P-програми розсіяним способом. Коли звичайні вузли підключаються до супервузлів, вони називаються кластерами. Супервузли відстежують і керують розподіленими обчислювальними ресурсами та одноранговими запитами в межах своїх кластерів, забезпечуючи ефективну екосистему.

Атаки Sybil і DDoS-атаки

Хоча різноманітні заходи безпеки, застосовані до додатків P2P, є відносно достатніми, вони не забезпечують повного захисту від загроз від зловмисників. Серед цих загроз, що насуваються, атаки Sybil і атаки Distributed Denial-of-Service (DDoS) є одними з основних векторів, які використовуються для порушення конфіденційності, цілісності та доступності P2P-сервісів.

Досліджена ймовірність успіху при виконанні подвійних витрат за допомогою атаки Sybil у децентралізованій мережі Bitcoin, визнаючи відносну ймовірність враховуючи необхідну кількість обчислювальної потужності. Багато дослідників довели атаку Sybil як велику загрозу мережевим системам P2P, зазначивши, що атаки Sybil також можуть завдати шкоди системам Bitcoin. Їхній

процес атаки був описаний одним зловмисником, який створив численні вузли Sybil із кількома підробленими ідентифікаторами, щоб затримати поширення дійсних блоків, щоб домінувати в змаганні за майнінг проти чесних майнерів. Тим часом, під впливом вузлів Sybil відбуватимуться розгалуження блокчейну, що призведе до втрати обчислювальної потужності деяких чесних вузлів. Результативний рівень успішності їх підходу до атаки вимагав 32% частки обчислювальної потужності в мережі біткойн, що є відносно великою кількістю приблизно 32 екзахешів на секунду.

Історично додатки proxy рівня задовольняли застосуванню загальному їхньому застосуванню й потребам Internet. Однак, у міру перетворення Internet у постійно мінливе динамічне середовище, що постійно пропонує нові протоколи, сервіси й додатки, proxy більше не здатні обробити різні типи взаємодій в Internet або відповідати новим потребам бізнесу, високим вимогам до пропускної здатності й безпеки мереж.

Check Point FireWall-1 технологія перевірки з урахуванням стану протоколу (Stateful Inspection Technology)

На відміну від описаних альтернатив, FireWall-1 вводить передову архітектуру, названу технологією перевірки з урахуванням стану протоколу, що реалізує всі необхідні можливості firewall на мережному рівні.

Пропонуючи перевірку з урахуванням стану протоколу, FireWall-1 модуль інспекції має доступ і аналізує дані, отримані від всіх рівнів комунікацій. Ці дані про "стан" і "контексті" запам'ятовуються й обновляються динамічно, забезпечуючи віртуальну інформацію про сесію для відстеження протоколів без установки з'єднань (таких як RPC і додатків заснованих на UDP). Дані, зібрані зі станів з'єднань і додатків, конфігурації мережі й правил безпеки, використовуються для генерації відповідної дії й або прийняття, або відкидання, або шифрації каналу зв'язку. Будь-який трафік, що навмисно не дозволений правилами безпеки блокується за замовчуванням і одночасно в реальному часі генеруються сигнали оповіщення, забезпечуючи системного адміністратора

повною інформацією про стан мережі.

Таблиця 3.1 – Порівняння технологій

Властивість firewall	Маршрутизатори	Proxy	Перевірка з урахуванням стану протоколу
Інформація про канал зв'язку	Частково	Частково	Так
Інформація про історію з'єднань	Немає	Частково	Так
Інформація про стан додатка	Немає	Так	Так
Дії над інформацією	Частково	Так	Так

Отже, FireWall-1 сполучає прозорість на мережному рівні, повноту, надійність і високу продуктивність із гнучкістю на рівні додатків, забезпечуючи тим самим чудове рішення для забезпечення безпеки, що значно перевершує можливості попередніх рішень.

Політика безпеки

Політика безпеки FireWall-1 виражається у вигляді бази правил і властивостей. База правил – це впорядкований набір правил, за допомогою яких перевіряється кожне з'єднання. Якщо джерело з'єднання, призначення й тип сервісу відповідають правилу, із з'єднанням буде виконане дія, описана в правилі (Accept – прийняти, Encrypt-зашифрувати, Reject-відхилити, Drop-залишити). Якщо з'єднання не відповідає жодному із правил, воно блокується, відповідно до принципу "Що спеціально не дозволено, завжди заборонене".

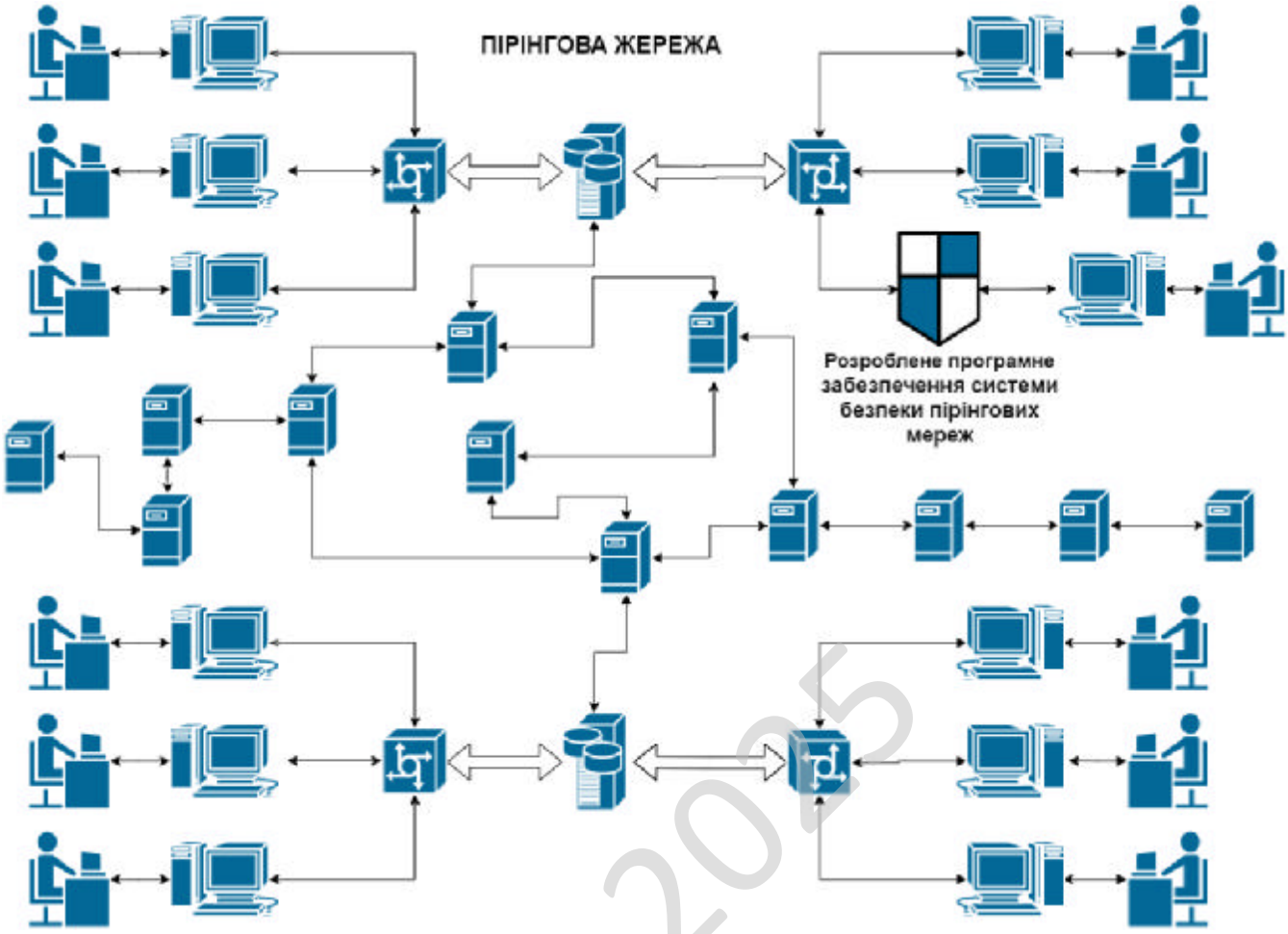

Рисунок 3.1 – Структурна схема системи

Модуль перевірки FireWall-1

Модуль перевірки FireWall-1 динамічно завантажується в ядро операційної системи, між рівнем Data Link – каналом передачі даних і мережею (рівні 2 і 3). Коли приходить перший пакет нового з'єднання, модуль перевірки FireWall-1 перевіряє базу правил для визначення повинне чи бути дозволене це з'єднання. Як тільки з'єднання встановлене, FireWall-1 додає його у внутрішню таблицю з'єднань. З міркувань ефективності, наступні пакети з'єднання перевіряються по таблиці з'єднань, а не по базі правил. Пакету дозволяється бути переданим тільки, якщо з'єднання є в таблиці з'єднань. У такому з'єднанні як тут, з'єднання повністю ведеться модулем перевірки FireWall-1.

Взаємодія FireWall-1 з різними протоколами

– UDP – З інформації в заголовку TCP/UDP, FireWall-1, використовуючи свої унікальні здатності, моделює стан комунікаційного протоколу, на основі чого відслідковує й управляє з'єднаннями UDP.

– FTP – Для відстеження зворотного з'єднання FTP-data, FireWall-1 витягає інформацію з області додатка в пакеті. Така унікальна здатність використання інформації із всіх рівнів дозволяє FireWall-1 моделювати стан протоколу, на основі чого зворотне з'єднання може бути встановленоі ("Вихідні з'єднання FTP").

– RPC – FireWall-1 використовує всі описані вище можливості, включаючи інформацію про стан, отриманий з додатка для відстеження динамічного перепризначення номерів програм і портів цього складного протоколу ("RPC (Remote Procedure Call)").

У комп'ютерних мережах демілітаризованою зоною називається ділянка мережі, доступна як внутрішнім, так і зовнішнім користувачам. Вона більше захищена у порівнянні із зовнішньою мережею, але менш захищенай у порівнянні із внутрішньою мережею. DMZ створюється за допомогою одного або декількох файрволів, що розмежовують внутрішню мережу, DMZ і зовнішню мережу. В DMZ часто розміщаються веб-сервери, відкриті для доступу ззовні.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. У багатьох пристроях для домашніх мереж, наприклад інтегрованих маршрутизаторах, часто є багатофункціональні програмні файрволи. Такі файрволи 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.

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 файрволу.

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).

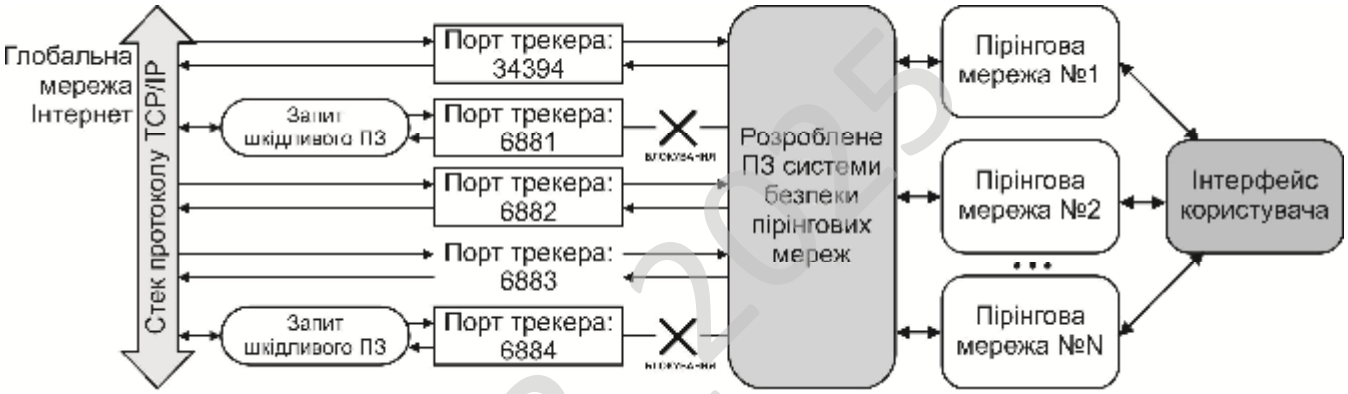

Рисунок 3.2 – Функціональна схема системи

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 файрволу. 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

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

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

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. 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 оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи безпеки пірінгових мереж.

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 підпрограми 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.

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.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

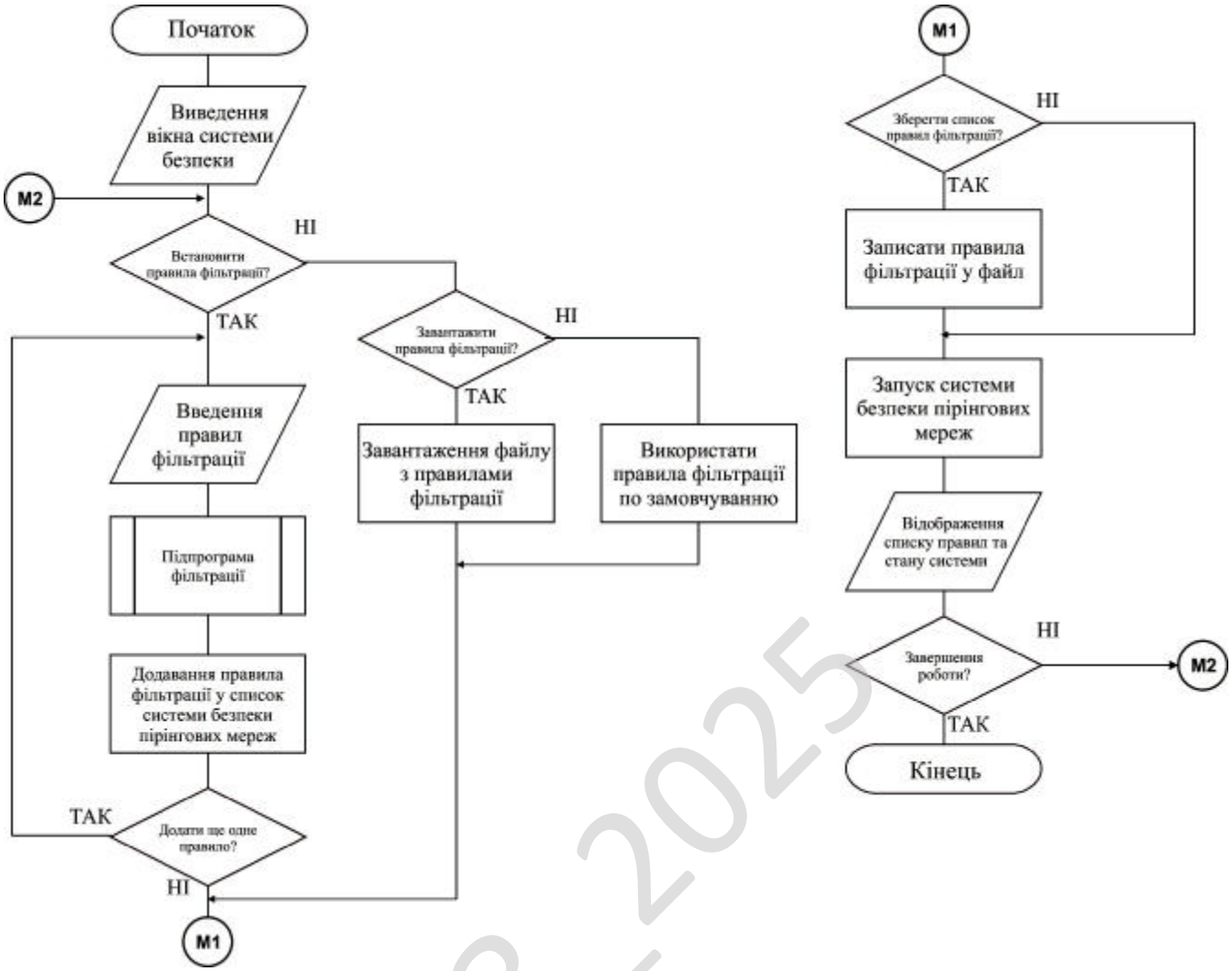

Рисунок 4.1 – Блок-схема основної програми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще

більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

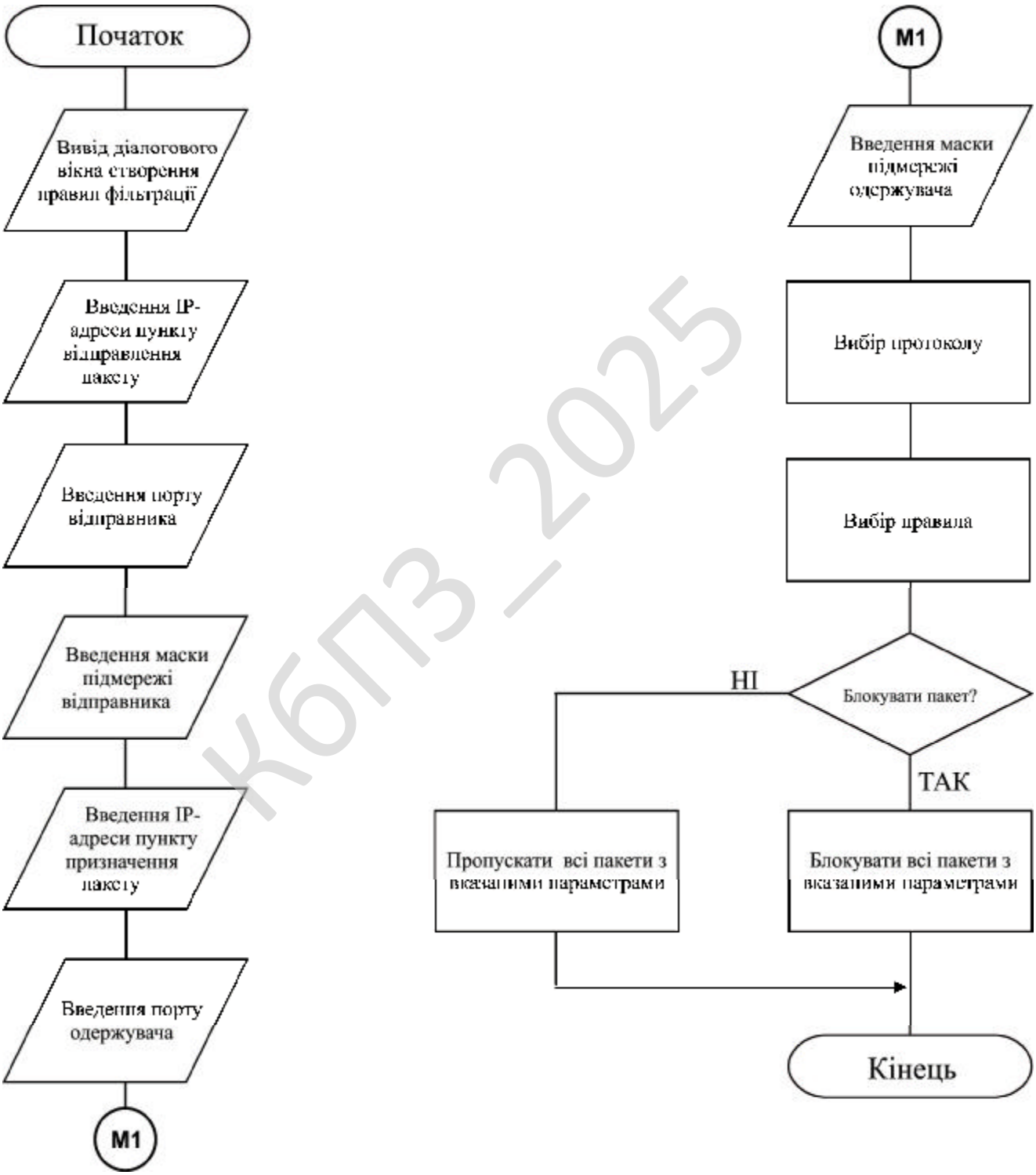

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

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.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі

випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з

класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Підпрограма виводу головного вікна системи безпеки пірінгових мереж:

```
//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;
// fail to create
     }
  //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("System");
     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::OnSaveRules()
{
     CSystemAppDoc *doc = (CSystemAppDoc *)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("This file name isn't valid.");
        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();
}
```

Підпрограма завантаження списку правил з файлу:

```
void CMainFrame::OnLoadRules()
{
    CFile file;
    CFileException e;
    DWORD nRead;
    CSystemAppDoc *doc = (CSystemAppDoc *)GetActiveDocument();
    CFileDialog dg(TRUE,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("This file name isn't valid.");
        return;
    }
    if(!file.Open(nf, CFile::modeRead, &e ) )
    {
        AfxMessageBox("Error openning the file.");
        return;
    }
    doc->ResetRules();
    PFFORWARD_ACTION action;
    file.Read(&action, sizeof(PFFORWARD_ACTION));
    if(action != pckFilter.GetDefaultAction())
    {
        pckFilter.RemoveAll();
        pckFilter.SetDefaultAction(action);
        doc->defaultAction = action;
    }
        RuleInfo rule;
       do
   {
        nRead = file.Read(&rule, sizeof(RuleInfo));
        if(nRead == 0)
            break;
        if(doc->AddRule(rule.sourceIp,
                        rule.sourceMask,
                         rule.sourcePort,
                         rule.destinationIp,
                         rule.destinationMask,
                         rule.destinationPort,
                         rule.protocol,
                         1)  != 0)
        {
            AfxMessageBox("Error adding a rule.");
            break;
        }
   }while (1);
    CSystemAppView *view = (CSystemAppView *)GetActiveView();
    view->UpdateList();}
```

Після того як створені чи відкриті правила фільтрації можна запустити систему безпеки пірінгових мереж.

Після запуску системи безпеки пірінгових мереж, програма починає фільтрувати пакети по вказаним правилам. Підпрограма запуску:

```
void CMainFrame::OnButtonstart()
{
     CSystemAppDoc *doc = (CSystemAppDoc *)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)
     {
          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.AddFilter(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 ПЗ. Підпрограма зупинки файрволу виглядає наступним чином:

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

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою MISTY1. MISTY1 – блоковий алгоритм шифрування, створений для компанії Mitsubishi Electric криптологом Міцуру Мацуї. Назва є абревіатурою Mitsubishi Improved Security Technology. Алгоритм був розроблений в 1995-1996 рр. Відомі також дві модифікації алгоритму MISTY1: MISTY2 і KASUMI

Шифр став переможцем на Європейському конкурсі NESSIE. У результаті аналізу алгоритму експерти зробили вивід, що ніяких серйозних уразливостей даний алгоритм не має (переважно, завдяки вкладеним мережам Фейстеля, що суттєво утрудняє криптоаналіз). У нього високий запас криптостійкості, алгоритм має високу швидкість шифрування й досить ефективний для апаратної реалізації.

Алгоритм був розроблений на основі теорії «підтвердженої безпеки» проти диференціального й лінійного криптоаналізу. Цей алгоритм був спроектований, щоб протистояти криптоатакам, відомим на момент створення.

З моменту публікації MISTY1 було проведено багато досліджень, щоб оцінити його рівень безпеки.

Диференціальний і неможливий диференціальний криптоаналіз високого порядку ефективно застосовується до блокових шифрів з малим ступенем.

Найкращі результати для обох варіантів були отримані для 5-рівневого алгоритму MISTY1 без FL функцій.

Саме FL функції й широкобітні AND/OR операції в сильно утрудняють використання диференціального криптоаналізу, що не заважає проведенню в цьому напрямку всі нових досліджень і досягненню усе більш близьких до розв'язку результатів.

Параметри вихідних даних

MISTY1 – це шифр на основі вкладених мереж Фейстеля з варьюємим числом раундів. Рекомендоване використання 8-раундової версії, але може використовуватися будь-яка кількість раундів, кратне 4-м. Розмір блоку вихідного тексту – 64 біта, розмір ключа – 128 біт.

Для роботи алгоритму також попередньо виконується процедура розширення ключа, яка для 8-мі раундів обчислює 1216 бітів ключової інформації з 128-бітного ключа шифрування.

Структура алгоритму

Для задоволення вимогам конкурсу NESSIE, а також для задоволення завдання мультиплатформеності, в алгоритмі MISTY1 використовувалися наступні методи шифрування:

– Логічні операції.

– Арифметичні операції.

– Операції зрушення.

– Таблиці перестановок.

Як говорилося вище, алгоритм MISTY1 заснований на «вкладених» мережах Фейстеля. Спочатку блок вихідного тексту розбивається на два 32-бітних субблоки, після чого виконується r раундів наступних перетворень[1]:

– У кожному непарному раунді обоє субблоки обробляються операцією FL

– Над оброблюваним субблоком виконується операція FO.

– Результат цих операцій накладається логічною операцією «, що виключає або» (XOR) на неопрацьований субблок.

– Субблоки міняються місцями. Після заключного раунду обоє субблоки ще раз обробляються операцією FL.

Операція FL

Оброблюваний 32-бітний субблок розбивається на два 16-бітних фрагмента, до яких застосовуються операції, де:

– L і R – вхідні значення лівого й правого фрагментів відповідно;

– L' і R' – вихідні значення;

– і – фрагменти j-го підключа і-го раунду для функції FL (процедура розширення ключа докладно описана далі);

– і – побітіві логічні операції «і» і «або» відповідно.

Операція FO

Саме ця функція є вкладеною мережею Фейстеля. Тут, як і раніше, виконується розбивка вхідного значення на два 16-бітних фрагмента, що проходять 3 раунду наступних перетворень:

– На лівий фрагмент операцією XOR накладається фрагмент ключа, де k – номер раунду функції FO.

– Лівий фрагмент обробляється операцією FI.

– На лівий фрагмент накладається операцією XOR значення правого фрагмента.

– Фрагменти міняються місцями.

Після третього раунду операції FO на лівий фрагмент накладається операцією XOR додатковий фрагмент ключа.

Операція FI

Дана операція також представляє собою третій рівень вкладеності мережі Фейстеля. На відміну від двох верхніх рівнів, дана мережа є незбалансованою: оброблюваний 16-бітний фрагмент ділиться на дві частини: 9-бітну ліву й 7-бітну праву. Потім виконуються 3 раунду перетворень, що випливають:

– Ліва частина зазнає обробці S-box.9-бітна частина (в 1-м і 3-м раундах) обробляється таблицею S9, а 7-бітна (в 2-м раунді) – таблицею S7. Дані таблиці описані нижче.

– На ліву частину операцією XOR накладається поточне значення правої частини. При цьому, якщо праворуч 7-бітна частина, вона доповнюється нулями ліворуч, а в 9- бітної частини віддаляються ліворуч два біти.

– У другому раунді на ліву частину операцією XOR накладається фрагмент ключа раунду, а на праву – фрагмент. В інших раундах ці дії не виконуються.

– Ліва й права частини міняються місцями.

Для оптимального розв'язку завдання мультиплатформеності, таблиці S7 і S9 алгоритму MISTY1 можуть бути реалізовані як за допомогою обчислень, так і безпосередньо таблицями.

Розширення ключа

Для 8 раундів алгоритму результатом процедури розширення ключа буде наступний набір ключових значень:

– 20 фрагментів ключа (), кожний з яких має розмір по 16 бітів;

– 32 16-бітних фрагмента ();

– 24 7-бітних фрагмента (при k=4, тобто в 4-м раунді функції FO, операція FI не виконується);

– 24 9-бітних фрагмента.

Виконується дане обчислення в такий спосіб:

1. 128-бітний ключ ділиться на 8 фрагментів … по 16 бітів кожний.

2. Формуються значення: у якості використовується результат обробки значення функцією FI, яка в якості ключа (тобто сукупності необхідних 7- і 9-бітного фрагментів) використовує значення(якщо індекс n фрагмента ключа перевищує 8, то замість нього використовується індекс n-8).

Необхідні фрагменти розширеного ключа «набираються» у міру виконання перетворень із відповідних масивів і згідно з відповідними таблицями

16-бітний фрагмент ділиться на 7-бітний фрагмент і 9-бітний .

Розшифрування

Розшифрування проводиться виконанням тих же операцій, що й при зашифруванні, але з наступними змінами:

– фрагменти розширеного ключа використовуються у зворотній послідовності,

– замість операції FL використовується зворотна їй операція – FLI.

Схеми виконання функції FLI і процедури розшифрування наведено на малюнках 6 і 7 відповідно:

Методи аналізу

Як говорилося на початку розділу, диференціальний і неможливий диференціальний аналізи виявилися ефективні лише до версій шифру з меншою кількістю раундів і без операції FL [2][3]. Проте, на даний момент цей напрямок аналізу, особливе використання слабких ключів, найбільше перспективно, тому що наближене до реальних можливих допущень при використанні алгоритму.

Так само, ученим з Японії був проведений інтегральний аналіз повного алгоритму, використовуючи відкритих текстів зі складністю обчислення, рівної [4].

Лінійний аналіз дав результати тільки для 7-раундової версії шифру, і також без операції FL[5].

Так як MISTY1 створювався, у тому числі, з розрахунку на апаратну реалізацію, має сенс диференціальний аналіз, заснований на використанні атаки по помилках обчислень, що в цьому випадку наближене до реальності.

Таким чином, була докладно описана структура алгоритму шифрування MISTY1 і розглянуті методи його аналізу, найбільш прагматичні напрямки дослідження. Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу до аналізу MISTY1.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення системи безпеки пірінгових мереж працює у автоматичному режимі та замість головного вікна має вікно налаштувань роботи яке складається з наступних функціональних блоків:

– Блоку обрання параметрів поточного вузла.

– Блоку обрання параметрів одного або групи віддалених вузлів.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

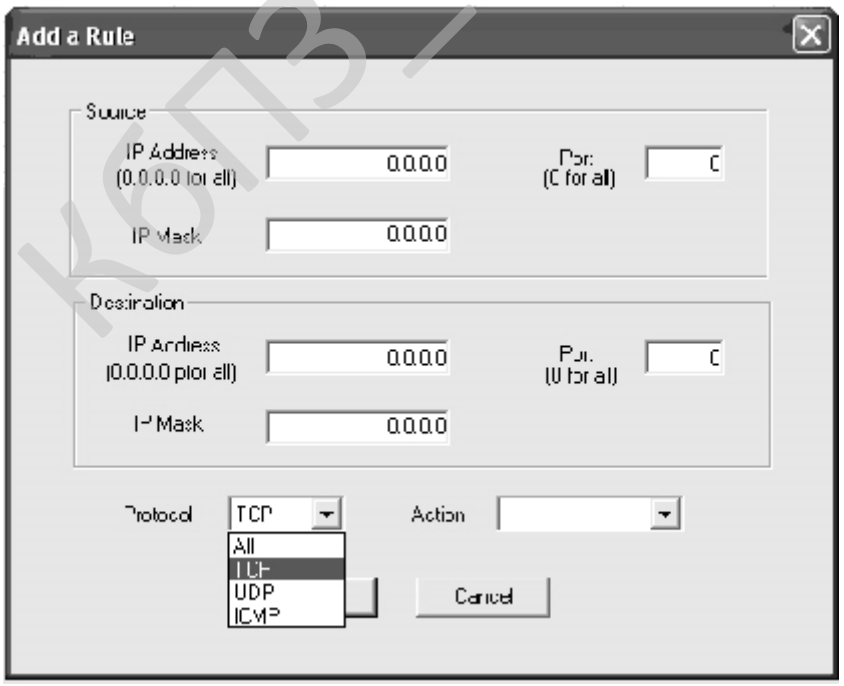

Рисунок 5.1 – Вікно налаштувань роботи розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

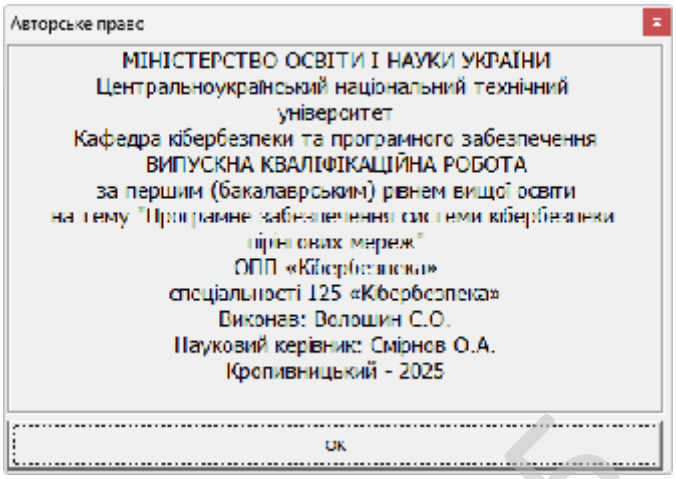

Рисунок 5.2 – Вікно розробника ПЗ

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 та чорної скриньки.

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.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

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.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки пірінгових мереж.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем пірінгових мереж.

– Досліджена система пірінгових мереж.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки пірінгових мереж.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання пірінгових мереж.

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 кібербезпеки пірінгових мереж. 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 кібербезпеки 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 MISTY1.

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

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

3. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

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

10. 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

11. 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.

12. 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.

13. 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,

14. 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.

15. 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.

16. 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

17. 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.

18. 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.

19. 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.*

20. 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.

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

22. 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.

23. 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.

24. 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.

25. 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.

26. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

27. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

28. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

29. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

30. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

31. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

32. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

33. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

34. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

35. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

36. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

37. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

38. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced*

Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18 - 21 September 2019. P.713-718.

39. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

40. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

41. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

42. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

43. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

44. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

45. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

46. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

47. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

48. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

49. 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.

50. 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.

51. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки пірінгових мереж.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 51-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки пірінгових мереж.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки пірінгових мереж;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 59 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнов О.А.

Програмне забезпечення системи кібербезпеки пірінгових мереж

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 42

Літера: РП

Кропивницький – 2025 року

SecurityPeerToPeerApp.cpp - головний файл програми

```
//Описувач головного класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SecurityPeerToPeerApp.h"

#include "MainFrm.h"
#include "SecurityPeerToPeerAppDoc.h"
#include "SecurityPeerToPeerAppView.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppApp
//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CSecurityPeerToPeerAppApp, CWinApp)
      //{{AFX_MSG_MAP(CSecurityPeerToPeerAppApp)
      ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
      ON_COMMAND(ID_APP_HELP, OnAppHelp)

      //}}AFX_MSG_MAP
      // стандартний файл для команд
      ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
      ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
      // Стандартне розпечатування файлів установки
      ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppApp construction
//Опис конструктора
CSecurityPeerToPeerAppApp::CSecurityPeerToPeerAppApp()
{
}

/////////////////////////////////////////////////////////////////////////////
CSecurityPeerToPeerAppApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppApp initialization

//Ініціалізація вікна
BOOL CSecurityPeerToPeerAppApp::InitInstance()
{
      AfxEnableControlContainer();

        //Ініціалізація лінковки статичного управління MFC
#ifdef _AFXDLL
      Enable3dControls();
#else
      Enable3dControlsStatic();
#endif


      SetRegistryKey(_T("SecurityPeerToPeerApp"));

  // Завантаження стандартних INI файлів настроювання (підключення MRU)
      LoadStdProfileSettings();

  //Створення  SDI фрейму вікна
      CSingleDocTemplate* pDocTemplate;
      pDocTemplate = new CSingleDocTemplate(
```

```
            IDR_MAINFRAME,
            RUNTIME_CLASS(CSecurityPeerToPeerAppDoc),
            RUNTIME_CLASS(CMainFrame),
            RUNTIME_CLASS(CSecurityPeerToPeerAppView));
      AddDocTemplate(pDocTemplate);

        // Аналіз командного рядка, DDE, відкриття файлу
      CCommandLineInfo cmdInfo;
      ParseCommandLine(cmdInfo);

        //Видалення параметрів командного рядка
      if (!ProcessShellCommand(cmdInfo))
            return FALSE;

        // Ініціалізація вікна, його відображення й відновлення
      m_pMainWnd->ShowWindow(SW_SHOW);
      m_pMainWnd->UpdateWindow();

      return TRUE;
}


////////////////////////////////////////////////////////////////////////////
// клас обробки й відображення вікна інформації про програму
class CAboutDlg : public CDialog
{
public:
  //Конструктор
      CAboutDlg();

// Діалогові дані
      //{{AFX_DATA(CAboutDlg)
  // Покажчик на ресурс оголошення діалогового вікна
      enum { IDD = IDD_ABOUTBOX };
      //}}AFX_DATA

            //{{AFX_VIRTUAL(CAboutDlg)
      protected:
    // DDX/DDV підтримка обміну даних
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
супроводження
      //}}AFX_VIRTUAL

// Реалізація програми
protected:
      //{{AFX_MSG(CAboutDlg)
      //}}AFX_MSG
  //Декларація мапінгу повідомлень
      DECLARE_MESSAGE_MAP()
};

//Конструктор
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
      //{{AFX_DATA_INIT(CAboutDlg)
      //}}AFX_DATA_INIT
}

//функція зчитування й установки даних вікна
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CAboutDlg)
      //}}AFX_DATA_MAP
}
//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
      //{{AFX_MSG_MAP(CAboutDlg)
      //}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CHelpDlg dialog used for App Help
// клас обробки й відображення вікна допомоги по програмі
class CHelpDlg : public CDialog
{
public:
  //Конструктор
      CHelpDlg();

// Діалогові дані
      //{{AFX_DATA(CHelpDlg)
  // Покажчик на ресурс оголошення діалогового вікна
      enum { IDD = IDD_HELPBOX };
      //}}AFX_DATA

            //{{AFX_VIRTUAL(CHelpDlg)
      protected:
    // DDX/DDV підтримка обміну даних
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
      //}}AFX_VIRTUAL

// Implementation
protected:
      //{{AFX_MSG(CHelpDlg)
      //}}AFX_MSG
  //Декларація мапінгу повідомлень
      DECLARE_MESSAGE_MAP()
};

//Конструктор
CHelpDlg::CHelpDlg() : CDialog(CHelpDlg::IDD)
{
      //{{AFX_DATA_INIT(CHelpDlg)
      //}}AFX_DATA_INIT
}

//функція зчитування й установки дані вікна
void CHelpDlg::DoDataExchange(CDataExchange* pDX)
{
  FILE * f = NULL;
  if (fopen_s(&f, "help.txt", "r+t") == 0) {
#define BUFFER_SIZE 10240
    size_t count = 0;
    char buff[BUFFER_SIZE];
    char text[BUFFER_SIZE];
    count = fread(buff, sizeof( char ), BUFFER_SIZE, f);
    fclose(f);
    size_t index = 0;
    for (size_t i = 0 ; i < count; i++) {
      if (buff[i] == 0x0A) {
        text[index] = '\r';
        index++;
      }
      text[index] = buff[i];
      index++;
    }
    text[index] = 0;
    SetDlgItemText(IDC_HELP_TEXT, text);
  }

  CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CHelpDlg)
      //}}AFX_DATA_MAP
}
```

```
//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CHelpDlg, CDialog)
      //{{AFX_MSG_MAP(CHelpDlg)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

//функція створення й відкриття діалогового вікна інформації про програму
void CSecurityPeerToPeerAppApp::OnAppAbout()
{
      CAboutDlg aboutDlg;
      aboutDlg.DoModal();
}

//функція створення й відкриття діалогового вікна допомоги по програмі
void CSecurityPeerToPeerAppApp::OnAppHelp()
{
      CHelpDlg helpDlg;
      helpDlg.DoModal();
}

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

DefaultActionDlg.cpp - Підключення основних оголошень діалогового вікна

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SecurityPeerToPeerapp.h"
#include "DefaultActionDlg.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDefaultActionDlg dialog


//Опис конструктора
CDefaultActionDlg::CDefaultActionDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CDefaultActionDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CDefaultActionDlg)
	//}}AFX_DATA_INIT

  //Завдання первісної основної дії
	action = PF_ACTION_FORWARD;
}

//функція зчитування й установки даних вікна
void CDefaultActionDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CDefaultActionDlg)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}

//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CDefaultActionDlg, CDialog)
	//{{AFX_MSG_MAP(CDefaultActionDlg)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//Ініціалізація вікна
BOOL CDefaultActionDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	//Установка значень управління вікна
	if(action == PF_ACTION_DROP)
		CheckRadioButton(IDC_RADIOFORWARD, IDC_RADIODROP, IDC_RADIODROP);

	else
		CheckRadioButton(IDC_RADIOFORWARD, IDC_RADIODROP, IDC_RADIOFORWARD);

	return TRUE;
}

//обробка подій (пост повідомлень) Windows
void CDefaultActionDlg::OnOK()
{
  //Зчитування значення
	int id = GetCheckedRadioButton(IDC_RADIOFORWARD, IDC_RADIODROP);

  //Збереження  поточної основної дії
	if(id == IDC_RADIODROP)
		action = PF_ACTION_DROP;
```

```
        else
              action = PF_ACTION_FORWARD;

    //Виклик оброблювача події предка для завершення коректної реакції на подію
        CDialog::OnOK();
}
```

SecurityPeerToPeerAppDoc.cpp – формування правил

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SecurityPeerToPeerApp.h"

#include "SecurityPeerToPeerAppDoc.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppDoc
//Мапінг Windows подій, перехоплення пост повідомлень
IMPLEMENT_DYNCREATE(CSecurityPeerToPeerAppDoc, CDocument)

BEGIN_MESSAGE_MAP(CSecurityPeerToPeerAppDoc, CDocument)
	//{{AFX_MSG_MAP(CSecurityPeerToPeerAppDoc)
		!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppDoc construction/destruction

//Опис конструктора
CSecurityPeerToPeerAppDoc::CSecurityPeerToPeerAppDoc()
{
	nRules = 0;
	defaultAction = PF_ACTION_FORWARD;
}
//Опис деструктора
CSecurityPeerToPeerAppDoc::~CSecurityPeerToPeerAppDoc()
{
}

//обробка подій (пост повідомлень) Windows
BOOL CSecurityPeerToPeerAppDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppDoc serialization
//обробка подій (пост повідомлень) Windows
void CSecurityPeerToPeerAppDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
	}
	else
	{
	}
}

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppDoc diagnostics
//опис функцій обробки дебаг інформації
#ifdef _DEBUG
void CSecurityPeerToPeerAppDoc::AssertValid() const
{
```

```
	CDocument::AssertValid();
}

void CSecurityPeerToPeerAppDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppDoc commands
//обробка подій (пост повідомлень) Windows по додаванню правила
int CSecurityPeerToPeerAppDoc::AddRule(unsigned long srcIp,
                                       unsigned long srcMask,
                                       unsigned short srcPort,
                                       unsigned long dstIp,
                                       unsigned long dstMask,
                                       unsigned short dstPort,
                                       unsigned int protocol,
                                       int action)
{

	if(nRules >= MAX_RULES)
	{
		return -1;
	}

	else
	{
		rules[nRules].sourceIp          = srcIp;
		rules[nRules].sourceMask        = srcMask;
		rules[nRules].sourcePort        = srcPort;
		rules[nRules].destinationIp   = dstIp;
		rules[nRules].destinationMask = dstMask;
		rules[nRules].destinationPort = dstPort;
		rules[nRules].protocol          = protocol;
		rules[nRules].action            = action;

		nRules++;
	}

	return 0;
}
//обробка подій (пост повідомлень) Windows по скиданню правил
void CSecurityPeerToPeerAppDoc::ResetRules()
{
	nRules = 0;
}
//обробка подій (пост повідомлень) Windows по видаленню правила
void CSecurityPeerToPeerAppDoc::DeleteRule(unsigned int position)
{
	// out of range
	if(position >= nRules)
		return;

		if(position != nRules - 1)
	{
		unsigned int i;

		for(i = position + 1;i<nRules;i++)
		{
			rules[i - 1].sourceIp          = rules[i].sourceIp;
			rules[i - 1].sourceMask        = rules[i].sourceMask;
			rules[i - 1].sourcePort        = rules[i].sourcePort;
			rules[i - 1].destinationIp    = rules[i].destinationIp;
			rules[i - 1].destinationMask  = rules[i].destinationMask;
			rules[i - 1].destinationPort  = rules[i].destinationPort;
			rules[i - 1].protocol         = rules[i].protocol;
			rules[i - 1].action                 = rules[i].action;
```

```
            }
        }
        nRules ---i;
    }
```

SecurityPeerToPeerAppView.cpp – Формування вікон

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SecurityPeerToPeerApp.h"

#include "SecurityPeerToPeerAppDoc.h"
#include "SecurityPeerToPeerAppView.h"
#include "SockUtil.h"
#include "PacketFilter.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppView
//Мапінг Windows подій, перехоплення пост повідомлень
IMPLEMENT_DYNCREATE(CSecurityPeerToPeerAppView, CFormView)

BEGIN_MESSAGE_MAP(CSecurityPeerToPeerAppView, CFormView)
	//{{AFX_MSG_MAP(CSecurityPeerToPeerAppView)

	//}}AFX_MSG_MAP
	// Standard printing commands
	ON_COMMAND(ID_FILE_PRINT, CFormView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_DIRECT, CFormView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_PREVIEW, CFormView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppView construction/destruction
//Опис конструктора
CSecurityPeerToPeerAppView::CSecurityPeerToPeerAppView()
	: CFormView(CSecurityPeerToPeerAppView::IDD)
{
	//{{AFX_DATA_INIT(CSecurityPeerToPeerAppView)
	//}}AFX_DATA_INIT
	}

//Опис деструктора
CSecurityPeerToPeerAppView::~CSecurityPeerToPeerAppView()
{
}

//функція зчитування й установки дані вікна
void CSecurityPeerToPeerAppView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSecurityPeerToPeerAppView)
	DDX_Control(pDX, IDC_LIST1, m_rules);
	//}}AFX_DATA_MAP
}

//обробка повідомлення передстворення основного вікна з викликом оброблювача
предка для коректної обробки події
BOOL CSecurityPeerToPeerAppView::PreCreateWindow(CREATESTRUCT& cs)
{


	return CFormView::PreCreateWindow(cs);
}

//Ініціалізація вікна
```

```
void CSecurityPeerToPeerAppView::OnInitialUpdate()
{
	CFormView::OnInitialUpdate();
	GetParentFrame()->RecalcLayout();
	ResizeParentToFit();

	RECT rc;
	m_rules.GetClientRect(&rc);

	int width=rc.right-rc.left-110;
	m_rules.InsertColumn(0, "Source IP",LVCFMT_LEFT , width/6, 0);
	m_rules.InsertColumn(1, "Source Mask",LVCFMT_LEFT , width/6, 1);
	m_rules.InsertColumn(2, "Source Port",LVCFMT_LEFT ,width/6, 2);
	m_rules.InsertColumn(3, "Dest. IP",LVCFMT_LEFT , width/6, 3);
	m_rules.InsertColumn(4, "Dest. Mask",LVCFMT_LEFT , width/6, 4);
	m_rules.InsertColumn(5, "Dest. Port",LVCFMT_LEFT , width/6, 5);
	m_rules.InsertColumn(6, "Protocol",LVCFMT_LEFT ,60, 6);
	m_rules.InsertColumn(7, "Action",LVCFMT_LEFT , 50, 7);

	m_rules.SetExtendedStyle(LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES);


}

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppView printing
//Не використовується, але взагалі для друку , і виводу інформації на друк
BOOL CSecurityPeerToPeerAppView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// default preparation
	return DoPreparePrinting(pInfo);
}
//Не використовується, але взагалі для друку , і виводу інформації на друк
void CSecurityPeerToPeerAppView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{

}
//Не використовується, але взагалі для друку , і виводу інформації на друк
void CSecurityPeerToPeerAppView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{

}
//Не використовується, але взагалі для друку , і виводу інформації на друк
void CSecurityPeerToPeerAppView::OnPrint(CDC* pDC, CPrintInfo* /*pInfo*/)
{
	//
}

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppView diagnostics
//опис функцій обробки дебаг інформації
#ifdef _DEBUG
void CSecurityPeerToPeerAppView::AssertValid() const
{
	CFormView::AssertValid();
}

void CSecurityPeerToPeerAppView::Dump(CDumpContext& dc) const
{
	CFormView::Dump(dc);
}

CSecurityPeerToPeerAppDoc* CSecurityPeerToPeerAppView::GetDocument() // non-
debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSecurityPeerToPeerAppDoc)));
	return (CSecurityPeerToPeerAppDoc*)m_pDocument;
```

```
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CSecurityPeerToPeerAppView message handlers
//опис функції відновлення списку правил в інтерфейсі
void CSecurityPeerToPeerAppView::UpdateList()
{
	CSecurityPeerToPeerAppDoc *doc = GetDocument();

	int action = (doc->defaultAction == PF_ACTION_FORWARD) ? 1:0;

	// обновлення листа керування
	m_rules.DeleteAllItems();

	unsigned int i;
	for(i=0;i<doc->nRules;i++)
	{
		AddRuleToList(doc->rules[i].sourceIp,
				doc->rules[i].sourceMask,
				doc->rules[i].sourcePort,
				doc->rules[i].destinationIp,
				doc->rules[i].destinationMask,
				doc->rules[i].destinationPort,
				doc->rules[i].protocol,
				action);
	}
}

//опис функції додавання правила в інтерфейс еа відображення інформації про
правило
void CSecurityPeerToPeerAppView::AddRuleToList(unsigned long srcIp,
					unsigned long srcMask,
					unsigned short srcPort,
					unsigned long dstIp,
					unsigned long dstMask,
					unsigned short dstPort,
					unsigned int protocol,
					int action)
{
	char ip[16];
	char port[6];
	LVITEM it;
	int pos;


	it.mask		= LVIF_TEXT;
	it.iItem	= m_rules.GetItemCount();
	it.iSubItem = 0;
	it.pszText	= (srcIp == 0) ? "All" : IpToString(ip, srcIp);
	pos		= m_rules.InsertItem(&it);

	it.iItem	= pos;
	it.iSubItem = 1;
	it.pszText	= IpToString(ip, srcMask);
	m_rules.SetItem(&it);

	it.iItem	= pos;
	it.iSubItem = 2;

	if(protocol != ICMP_PROTOCOL)
		it.pszText	= (srcPort == 0) ? "All" : itoa(srcPort, port, 10);

	else
		it.pszText	= (srcPort == 255) ? "All" : itoa(srcPort, port, 10);

	m_rules.SetItem(&it);

	it.iItem	= pos;
```

```
        it.iSubItem = 3;
        it.pszText  = (dstIp == 0) ? "All" : IpToString(ip, dstIp);
        m_rules.SetItem(&it);

        it.iItem    = pos;
        it.iSubItem = 4;
        it.pszText  = IpToString(ip, dstMask);
        m_rules.SetItem(&it);

        it.iItem    = pos;
        it.iSubItem = 5;

        if(protocol != ICMP_PROTOCOL)
              it.pszText  = (dstPort == 0) ? "All" : itoa(dstPort, port, 10);

        else
              it.pszText  = (dstPort == 255) ? "All" : itoa(dstPort, port, 10);

        m_rules.SetItem(&it);


        it.iItem    = pos;
        it.iSubItem = 6;

        if(protocol == 1)
              it.pszText = "ICMP";

        else if(protocol == 6)
              it.pszText = "TCP";

        else if(protocol == 17)
              it.pszText = "UDP";

        else
              it.pszText = "All";

        m_rules.SetItem(&it);


        it.iItem    = pos;
        it.iSubItem = 7;
        it.pszText = action ? "Drop" : "Forward";
        m_rules.SetItem(&it);
}
```

MainFrm.cpp – Ініціалізація й створення головного вікна і його компонентів, основна робота програми

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SecurityPeerToPeerApp.h"

#include "MainFrm.h"
#include "RuleDlg.h"
#include "DefaultActionDlg.h"
#include "SecurityPeerToPeerAppDoc.h"
#include "SecurityPeerToPeerAppView.h"
#include "SockUtil.h"
#include "rules.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame
//Мапінг Windows подій, перехоплення пост повідомлень
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
      //{{AFX_MSG_MAP(CMainFrame)
      ON_WM_CREATE()
      ON_COMMAND(ID_BUTTONSTART, OnButtonstart)
      ON_COMMAND(ID_BUTTONADD, OnButtonadd)
      ON_COMMAND(ID_BUTTONDEL, OnButtondel)
      ON_COMMAND(ID_BUTTONSTOP, OnButtonstop)
      ON_UPDATE_COMMAND_UI(ID_BUTTONSTART, OnUpdateButtonstart)
      ON_UPDATE_COMMAND_UI(ID_BUTTONSTOP, OnUpdateButtonstop)
      ON_COMMAND(IDMENU_ADDRULE, OnMenuAddrule)
      ON_COMMAND(IDMENU_DELRULE, OnMenuDelrule)
      ON_COMMAND(ID_MENUSTART, OnMenustart)
      ON_UPDATE_COMMAND_UI(ID_MENUSTART, OnUpdateMenustart)
      ON_COMMAND(ID_MENUSTOP, OnMenustop)
      ON_UPDATE_COMMAND_UI(ID_MENUSTOP, OnUpdateMenustop)
      ON_COMMAND(ID_APP_EXIT, OnAppExit)
      ON_COMMAND(IDMENU_LOADRULES, OnLoadRules)
      ON_COMMAND(IDMENU_SAVERULES, OnSaveRules)
      ON_COMMAND(IDMENU_SETDEFAULT, OnMenuSetdefault)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
      ID_SEPARATOR,
/*    ID_INDICATOR_CAPS,
      ID_INDICATOR_NUM,
      ID_INDICATOR_SCRL,
  */
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
//Опис конструктора
CMainFrame::CMainFrame()
{
      started = FALSE;
}
//Опис деструктора
CMainFrame::~CMainFrame()
{
```

```
}

//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;      // помилка створення
      }


      //m_wndToolBar.EnableDocking(CBRS_ALIGN_RIGHT);
  //EnableDocking(CBRS_ALIGN_ANY);
      //DockControlBar(&m_wndToolBar);

  //Установка заголовка
      this->SetWindowText("SecurityPeerToPeer");

      return 0;
}

//обробка повідомлення передстворення основного вікна з викликом оброблювача
предка для коректної обробки події
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
      if( !CFrameWnd::PreCreateWindow(cs) )
            return FALSE;


            cs.style &= ~ FWS_ADDTOTITLE;

      return TRUE;
}

////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
//опис функцій обробки дебаг інформації
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
      CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
      CFrameWnd::Dump(dc);
}

#endif //_DEBUG
```

```
////////////////////////////////////////////////////////////////////////////
// CMainFrame заголовок повідомлення

//Функція запуску програми
void CMainFrame::OnButtonstart()
{
	CSecurityPeerToPeerAppDoc *doc = (CSecurityPeerToPeerAppDoc
*)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)
	{
		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.AddFilter(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;
}

//функція вимикання програми
void CMainFrame::OnButtonstop()
{
	pckFilter.RemoveAll();

	started = FALSE;
}
```

```
//функція додавання правила
void CMainFrame::OnButtonadd()
{
	CSecurityPeerToPeerAppDoc *doc = (CSecurityPeerToPeerAppDoc
*)GetActiveDocument();
	CRuleDlg dlg;

	// перевірка правильності номерів
	if(doc->nRules < MAX_RULES )
	{
		dlg.defaultAction = pckFilter.GetDefaultAction();

		if(dlg.DoModal() == IDOK)
		{
			// додавання правильного номера до листа правильних адрес

			if(doc->AddRule(dlg.srcIp, dlg.srcMask, dlg.srcPort,
dlg.dstIp, dlg.dstMask, dlg.dstPort, dlg.protocol, dlg.cAction) != 0)
				AfxMessageBox("Error adding the rule.");

			else
			{
				// Після цього коректуються правила
				CSecurityPeerToPeerAppView *view =
(CSecurityPeerToPeerAppView *)GetActiveView();

				view->UpdateList();
			}

		}
	}

	else
		AfxMessageBox("You can't add more rules.");
}

//функція видалення правила
void CMainFrame::OnButtondel()
{
	CSecurityPeerToPeerAppView *view = (CSecurityPeerToPeerAppView
*)GetActiveView();


		POSITION pos = view->m_rules.GetFirstSelectedItemPosition();
	if (pos == NULL)
	{
		AfxMessageBox("Select a Rule, please.");

		return;
	}

	int position;
	position = view->m_rules.GetNextSelectedItem(pos);

	CSecurityPeerToPeerAppDoc *doc = (CSecurityPeerToPeerAppDoc
*)GetActiveDocument();
	doc->DeleteRule(position);


	// перегляд змін в правилах
	view->UpdateList();

}

//функція перемикання стану меню по запуску програми
void CMainFrame::OnUpdateButtonstart(CCmdUI* pCmdUI)
{
	if(started)
```

```
            pCmdUI->Enable(FALSE);

      else
            pCmdUI->Enable(TRUE);
}

//функція перемикання стану меню по зупинці програми
void CMainFrame::OnUpdateButtonstop(CCmdUI* pCmdUI)
{
      if(started)
            pCmdUI->Enable(TRUE);

      else
            pCmdUI->Enable(FALSE);
}

//функція обробки меню по додаванню правила
void CMainFrame::OnMenuAddrule()
{
      OnButtonadd();
}

//функція обробки меню по видаленню правила
void CMainFrame::OnMenuDelrule()
{
      OnButtondel();
}


//функція обробки меню по старту програми
void CMainFrame::OnMenustart()
{
      OnButtonstart();
}

//функція обробки відновлення меню
void CMainFrame::OnUpdateMenustart(CCmdUI* pCmdUI)
{
      if(started)
            pCmdUI->Enable(FALSE);

      else
            pCmdUI->Enable(TRUE);

}

//функція обробки меню по зупинці програми
void CMainFrame::OnMenustop()
{
      OnButtonstop();
}

//функція перемикання стану головного меню по зупинці програми
void CMainFrame::OnUpdateMenustop(CCmdUI* pCmdUI)
{
      if(started)
            pCmdUI->Enable(TRUE);

      else
            pCmdUI->Enable(FALSE);

}

//функція обробки виходу із програми
void CMainFrame::OnAppExit()
{

}
```

```
//функція обробки завантаження правил з файлу
void CMainFrame::OnLoadRules()
{
	CFile file;
	CFileException e;
	DWORD nRead;

	CSecurityPeerToPeerAppDoc *doc = (CSecurityPeerToPeerAppDoc
*)GetActiveDocument();

	CFileDialog dg(TRUE,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("This file name isn't valid.");

		return;
	}

	if( !file.Open(nf, CFile::modeRead, &e ) )
	{
		AfxMessageBox("Error openning the file.");

		return;
	}

	doc->ResetRules();

	PFFORWARD_ACTION action;
	file.Read(&action, sizeof(PFFORWARD_ACTION));

	if(action != pckFilter.GetDefaultAction())
	{
		pckFilter.RemoveAll();

		pckFilter.SetDefaultAction(action);
		doc->defaultAction = action;
	}



	RuleInfo rule;

    do
    {
		nRead = file.Read(&rule, sizeof(RuleInfo));

		if(nRead == 0)
			break;

		if(doc->AddRule(rule.sourceIp,
                        rule.sourceMask,
                            rule.sourcePort,
                            rule.destinationIp,
                            rule.destinationMask,
                            rule.destinationPort,
                            rule.protocol,
                            1)  != 0)
		{

			AfxMessageBox("Error adding a rule.");

			break;
		}
```

```
    }while (1);



      CSecurityPeerToPeerAppView *view = (CSecurityPeerToPeerAppView
*)GetActiveView();

      view->UpdateList();
}

//функція обробки збереження правил у файл
void CMainFrame::OnSaveRules()
{
      CSecurityPeerToPeerAppDoc *doc = (CSecurityPeerToPeerAppDoc
*)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("This file name isn't valid.");

            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();
}

//скидання даних і реініціалізації програми у вихідне положення
void CMainFrame::OnMenuSetdefault()
{
      CDefaultActionDlg dlg;
```

```
	dlg.action = pckFilter.GetDefaultAction();

	if(dlg.DoModal() == IDOK)
	{
		if(dlg.action != pckFilter.GetDefaultAction())
		{
			pckFilter.RemoveAll();

			pckFilter.SetDefaultAction(dlg.action);

			CSecurityPeerToPeerAppDoc *doc = (CSecurityPeerToPeerAppDoc
*)GetActiveDocument();
			doc->defaultAction = dlg.action;

			CSecurityPeerToPeerAppView *view = (CSecurityPeerToPeerAppView
*)GetActiveView();
			view->UpdateList();
		}
	}
}
```

PacketFilter.cpp – Формування правил фільтру

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "PacketFilter.h"


#include <stdlib.h>
#include <stdio.h>



/*********************************************
Class CPacketFilter.
*********************************************/
//Опис конструктора
CPacketFilter::CPacketFilter()
{
      defaultAction = PF_ACTION_FORWARD;
}
//Опис деструктора
CPacketFilter::~CPacketFilter()
{
      RemoveAll();
}

//Опис функції видалення всіх фільтрів з інтерфейсу
void CPacketFilter::RemoveAll()
{
      POSITION pos = interfacesTable.GetStartPosition();

      CPacketFilterInterface pckInt;
      unsigned long ip;

      while( pos != NULL )
      {

            interfacesTable.GetNextAssoc(pos, ip, pckInt);

            PfDeleteInterface(pckInt.hInterface);

      }

      interfacesTable.RemoveAll();
}


//Опис функції додавання фільтра в інтерфейс
int CPacketFilter::AddFilter(char  *localInterfaceIp,
                                          int  direction,
                                          char *srcIp,
                                          char *srcMask,
                                          char *dstIp,
                                          char *dstMask,
                                          int  srcPort,
                                          int  dstPort,
                                          int  protocol)
{
      CPacketFilterInterface interfaceHandle;

            if(!interfacesTable.Lookup(CharToIp(localInterfaceIp),
interfaceHandle))
      {

            interfaceHandle.Create(CharToIp(localInterfaceIp), defaultAction);

            interfacesTable[CharToIp(localInterfaceIp)] = interfaceHandle;
```

```
	}


	// Заповнюю структуру фільтру
	PF_FILTER_DESCRIPTOR ipFlt;
	ipFlt.dwFilterFlags     = FD_FLAGS_NOSYN;
	ipFlt.dwRule            = 0;
	ipFlt.pfatType          = PF_IPV4;
	ipFlt.dwProtocol        = protocol;
	ipFlt.fLateBound        = 0;
	ipFlt.wSrcPort          = srcPort;
	ipFlt.wSrcPortHighRange = srcPort;
	ipFlt.wDstPort          = dstPort;
	ipFlt.wDstPortHighRange = dstPort;

	unsigned long lIpSrc    = CharToIp(srcIp);
	unsigned long lIpDst    = CharToIp(dstIp);
	unsigned long lMaskSrc  = CharToIp(srcMask);
	unsigned long lMaskDst  = CharToIp(dstMask);

	ipFlt.SrcAddr = (PBYTE) &lIpSrc;
	ipFlt.SrcMask = (PBYTE) &lMaskSrc;
	ipFlt.DstAddr = (PBYTE) &lIpDst;
	ipFlt.DstMask = (PBYTE) &lMaskDst;

	DWORD errorCode;

	if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                    1,
                                                    &ipFlt,
                                                    0,
                                                    NULL,
                                                    NULL);

	if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                    0,
                                                    NULL,
                                                    1,
                                                    &ipFlt,
                                                    NULL);

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції додавання фільтра в інтерфейс по іншому набору параметрів
int CPacketFilter::AddFilter(unsigned long  localInterfaceIp,
                                        int             direction,
                                        unsigned long   srcIp,
                                        unsigned long   srcMask,
                                        unsigned long   dstIp,
                                        unsigned long   dstMask,
                                        int  srcPort,
                                        int  dstPort,
                                        int  protocol)
{
	CPacketFilterInterface interfaceHandle;

		if(!interfacesTable.Lookup(localInterfaceIp, interfaceHandle))
	{
```

```
            interfaceHandle.Create(localInterfaceIp, defaultAction);

            interfacesTable[localInterfaceIp] = interfaceHandle;
      }



      PF_FILTER_DESCRIPTOR ipFlt;
      ipFlt.dwFilterFlags     = 0;
      ipFlt.dwRule            = 0;
      ipFlt.pfatType          = PF_IPV4;
      ipFlt.dwProtocol        = protocol;
      ipFlt.fLateBound        = 0;
      ipFlt.wSrcPort          = srcPort;
      ipFlt.wSrcPortHighRange = srcPort;
      ipFlt.wDstPort          = dstPort;
      ipFlt.wDstPortHighRange = dstPort;


      ipFlt.SrcAddr = (PBYTE) &srcIp;
      ipFlt.SrcMask = (PBYTE) &srcMask;
      ipFlt.DstAddr = (PBYTE) &dstIp;
      ipFlt.DstMask = (PBYTE) &dstMask;

      DWORD errorCode;


      if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
            errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                1,
                                                &ipFlt,
                                                0,
                                                NULL,
                                                NULL);

      if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
            errorCode = PfAddFiltersToInterface(interfaceHandle.hInterface,
                                                0,
                                                NULL,
                                                1,
                                                &ipFlt,
                                                NULL);

      else
            return -2;

      if(errorCode != NO_ERROR)
            return -1;

      return 0;
}

//Опис функції видалення зазначеного фильтру з інтерфейсу
int CPacketFilter::RemoveFilter(char     *localInterfaceIp,
                                int         direction,
                                char  *srcIp,
                                char  *srcMask,
                                char  *dstIp,
                                char  *dstMask,
                                int         srcPort,
                                int         dstPort,
                                int         protocol)
{
      CPacketFilterInterface interfaceHandle;

            if(!interfacesTable.Lookup(CharToIp(localInterfaceIp),
interfaceHandle))
      {
            return -1;
```

```
	}


	// Заповнюю структуру фільтру
	PF_FILTER_DESCRIPTOR ipFlt;
	ipFlt.dwFilterFlags     = FD_FLAGS_NOSYN;
	ipFlt.dwRule            = 0;
	ipFlt.pfatType          = PF_IPV4;
	ipFlt.dwProtocol        = protocol;
	ipFlt.fLateBound        = 0;
	ipFlt.wSrcPort          = srcPort;
	ipFlt.wSrcPortHighRange = srcPort;
	ipFlt.wDstPort          = dstPort;
	ipFlt.wDstPortHighRange = dstPort;

	unsigned long lIpSrc    = CharToIp(srcIp);
	unsigned long lIpDst    = CharToIp(dstIp);
	unsigned long lMaskSrc  = CharToIp(srcMask);
	unsigned long lMaskDst  = CharToIp(dstMask);

	ipFlt.SrcAddr = (PBYTE) &lIpSrc;
	ipFlt.SrcMask = (PBYTE) &lMaskSrc;
	ipFlt.DstAddr = (PBYTE) &lIpDst;
	ipFlt.DstMask = (PBYTE) &lMaskDst;

	DWORD errorCode;

		if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
                                                         1,
                                                         &ipFlt,
                                                         0,
                                                         NULL);


	if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
                                                         0,
                                                         NULL,
                                                         1,
                                                         &ipFlt);

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції видалення зазначеного фильтру з інтерфейсу по іншому наборі
параметрів
int CPacketFilter::RemoveFilter(unsigned long  localInterfaceIp,
                                                int
	direction,
                                                unsigned long     srcIp,

                                                unsigned long     srcMask,
                                                unsigned long     dstIp,
                                                unsigned long     dstMask,
                                                int                     srcPort,
                                                int                     dstPort,
                                                int                     protocol)
{
	CPacketFilterInterface interfaceHandle;

	// Якщо інтерфейс не створений, то створюю його.
	if(!interfacesTable.Lookup(localInterfaceIp, interfaceHandle))
```

```
	{
		return -1;
	}


	// Заповнюю структуру фільтру
	PF_FILTER_DESCRIPTOR ipFlt;
	ipFlt.dwFilterFlags     = FD_FLAGS_NOSYN;
	ipFlt.dwRule            = 0;
	ipFlt.pfatType          = PF_IPV4;
	ipFlt.dwProtocol        = protocol;
	ipFlt.fLateBound        = 0;
	ipFlt.wSrcPort          = srcPort;
	ipFlt.wSrcPortHighRange = srcPort;
	ipFlt.wDstPort          = dstPort;
	ipFlt.wDstPortHighRange = dstPort;

	ipFlt.SrcAddr = (PBYTE) &srcIp;
	ipFlt.SrcMask = (PBYTE) &srcMask;
	ipFlt.DstAddr = (PBYTE) &dstIp;
	ipFlt.DstMask = (PBYTE) &dstMask;

	DWORD errorCode;

	// Додаю фільтр
	if(direction == IN_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
		                                         1,
		                                         &ipFlt,
		                                         0,
		                                         NULL);

	if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
		errorCode = PfRemoveFiltersFromInterface(interfaceHandle.hInterface,
		                                         0,
		                                         NULL,
		                                         1,
		                                         &ipFlt);

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції додавання глобального фільтра інтерфейсу
int CPacketFilter::AddGlobalFilter(char   *localInterfaceIp,
                                   int globalFilter)

{
	CPacketFilterInterface interfaceHandle;

	// Якщо інтерфейс не створений, то створюю його.
	if(!interfacesTable.Lookup(CharToIp(localInterfaceIp), interfaceHandle))
	{

		interfaceHandle.Create(CharToIp(localInterfaceIp), defaultAction);

		interfacesTable[CharToIp(localInterfaceIp)] = interfaceHandle;
	}


	DWORD errorCode;

	// Додаю глобальний фільтр
```

```
	errorCode = PfAddGlobalFilterToInterface(interfaceHandle.hInterface,
(GLOBAL_FILTER)globalFilter);

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


//Опис функції видалення глобального фільтра інтерфейсу
int CPacketFilter::RemoveGlobalFilter(char	*localInterfaceIp,
						int	globalFilter)

{
	CPacketFilterInterface interfaceHandle;

	// Якщо інтерфейс не створений, то створюю його.
	if(!interfacesTable.Lookup(CharToIp(localInterfaceIp), interfaceHandle))
	{
		return -1;
	}


	DWORD errorCode;

	// Додаю глобальний фільтр
	errorCode = PfRemoveGlobalFilterFromInterface(interfaceHandle.hInterface,
(GLOBAL_FILTER)globalFilter);

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}

//установка основної дії інтерфейсу
void CPacketFilter::SetDefaultAction(PFFORWARD_ACTION action)
{
	defaultAction = action;
}

//зчитування основної дії інтерфейсу
PFFORWARD_ACTION  CPacketFilter::GetDefaultAction()
{
	return defaultAction;
}

/*********************************************
Class CPacketFilterInterface.
*********************************************/
//інтерфейсний клас мережного адаптера
CPacketFilterInterface::CPacketFilterInterface()
{

}

CPacketFilterInterface::~CPacketFilterInterface()
{
//	PfDeleteInterface(hInterface);
}

// Створення інтерфейсу й асоціація його з IP
int CPacketFilterInterface::Create(unsigned long ip, PFFORWARD_ACTION
defaultAction)
{

	// Створення Інтерфейсу  й завдання початкової дії пропускати всі пакети
	DWORD errorCode = PfCreateInterface(0,
```

```
                                                defaultAction,
                                                defaultAction,
                                                FALSE,
                                                TRUE,
                                                &hInterface);

        if(errorCode != NO_ERROR)
        {
                return -1;
        }


        // Асоціація IP с інтерфейсом
        PBYTE lIp = (PBYTE)&ip;
        errorCode = PfBindInterfaceToIPAddress(hInterface, PF_IPV4, lIp);

        if(errorCode != NO_ERROR)
        {
                PfDeleteInterface(hInterface);

                hInterface = NULL;

                return -2;
        }

        return 0;
}

// перетворення строкового значення IP у цифрове представлення
unsigned long CharToIp(const char *sIp)
{
        int octets[4];
        int i;
        const char * auxCad = sIp;
        unsigned long lIp = 0;

        for(i = 0; i < 4; i++)
        {
                octets[i] = atoi(auxCad);

                if(octets[i] < 0 || octets[i] > 255)
                        return 0;

                lIp |= (octets[i] << (i*8));

                auxCad = strchr(auxCad, '.');

                if(auxCad == NULL && i!=3)
                        return -1;

                auxCad++;
        }


        return lIp;
}
```

ResizeDlg.cpp – Зміна розмірів вікон

```
**/
#include "stdafx.h"
#include "ResizeDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#ifndef OBM_SIZE
#define OBM_SIZE 32766
#endif

CItemCtrl::CItemCtrl()
{
      m_nID = 0;
      m_stxLeft = CST_NONE;
      m_stxRight = CST_NONE;
      m_styTop = CST_NONE;
      m_styBottom = CST_NONE;
      m_bFlickerFree = 1;
      m_xRatio = m_cxRatio = 1.0;
      m_yRatio = m_cyRatio = 1.0;
}

CItemCtrl::CItemCtrl(const CItemCtrl& src)
{
      Assign(src);
}

CItemCtrl& CItemCtrl::operator=(const CItemCtrl& src)
{
      Assign(src);
      return *this;
}

void CItemCtrl::Assign(const CItemCtrl& src)
{
      m_nID = src.m_nID;
      m_stxLeft = src.m_stxLeft;
      m_stxRight = src.m_stxRight;
      m_styTop = src.m_styTop;
      m_styBottom = src.m_styBottom;
      m_bFlickerFree = src.m_bFlickerFree;
      m_bInvalidate = src.m_bInvalidate;
      m_wRect = src.m_wRect;
      m_xRatio = src.m_xRatio;
      m_cxRatio = src.m_cxRatio;
      m_yRatio = src.m_yRatio;
      m_cyRatio = src.m_cyRatio;
}

HDWP CItemCtrl::OnSize(HDWP hDWP,int sizeType, CRect *pnRect, CRect *poRect,
CRect *p0, CWnd *pDlg)
{
      CRect ctrlRect = m_wRect, curRect;
      CWnd   *pWnd = pDlg->GetDlgItem(m_nID);
      int         delta = pnRect->Width() - poRect->Width();
      int         delta = pnRect->Height()- poRect->Height();
      int       delta0 = pnRect->Width() - p 0-0->Width();
      int         delta0 = pnRect->Height()- p 0-0->Height();
      int         newCx, newCy;
      int         st, bUpdateRect = 1;
```

```
	// зміна зліва-горизонитально
	st = CST_NONE;
	if ((sizeType & WST_LEFT) && m_stxLeft != CST_NONE) {
		ASSERT((sizeType & WST_RIGHT) == 0);

		st = m_stxLeft;
	}
	else if ((sizeType & WST_RIGHT) && m_stxRight != CST_NONE) {
		ASSERT((sizeType & WST_LEFT) == 0);

		st = m_stxRight;
	}

	switch(st) {
	case CST_NONE:
		if (m_stxLeft == CST_ZOOM || m_stxRight == CST_ZOOM ||
				m_stxLeft == CST_DELTA_ZOOM || m_stxRight ==
CST_DELTA_ZOOM)
		{
			pWnd->GetWindowRect(&curRect);
			pDlg->ScreenToClient(&curRect);

			ctrlRect.left = curRect.left;
			ctrlRect.right = curRect.right;

			bUpdateRect = 0;
		}

		break;

	case CST_RESIZE:
		ctrlRect.right += delta;
		break;

	case CST_REPOS:
		ctrlRect.left += delta;
		ctrlRect.right += delta;
		break;

	case CST_RELATIVE:
		newCx = ctrlRect.Width();
		ctrlRect.left = (int)((double)m_xRatio * 1.0 * pnRect->Width() -
newCx / 2.0);
		ctrlRect.right = ctrlRect.left + newCx; /* (int)((double)m_xRatio *
1.0 * pnRect->Width() + newCx / 2.0); */
		break;

	case CST_ZOOM:
		ctrlRect.left = (int)(1.0 * ctrlRect.left  * (double)pnRect->Width()
/ p 0-0->Width());
		ctrlRect.right = (int)(1.0 * ctrlRect.right * (double)pnRect-
>Width() / p 0-0->Width());
		bUpdateRect = 0;
		break;

	case CST_DELTA_ZOOM:
		newCx = ctrlRect.Width();
		ctrlRect.right = (int)(ctrlRect.left * 1.0 + delta0 * 1.0 * m_xRatio
+ newCx * 1.0 + delta0 * m_cxRatio);
		ctrlRect.left += (int)(delta0 * 1.0 * m_xRatio);
		bUpdateRect = 0;
		break;

	default:
		break;
	}

	// зміна згори
	st = CST_NONE;
```

```
	if ((sizeType & WST_TOP) && (m_styTop != CST_NONE)) {
		ASSERT((sizeType & WST_BOTTOM) == 0);

		st = m_styTop;
	}
	else if ((sizeType & WST_BOTTOM) && (m_styBottom != CST_NONE)) {
		ASSERT((sizeType & WST_TOP) == 0);

		st = m_styBottom;
	}

	switch (st) {
	case CST_NONE:
		if (m_styTop == CST_ZOOM || m_styTop == CST_ZOOM ||
				m_styBottom == CST_DELTA_ZOOM || m_styBottom ==
CST_DELTA_ZOOM)
		{
			pWnd->GetWindowRect(&curRect);
			pDlg->ScreenToClient(&curRect);

			ctrlRect.top = curRect.top;
			ctrlRect.bottom = curRect.bottom;

			bUpdateRect = 0;
		}

		break;

	case CST_RESIZE:
		ctrlRect.bottom += delta;
		break;

	case CST_REPOS:
		ctrlRect.top += delta;
		ctrlRect.bottom += delta;
		break;

	case CST_RELATIVE:
		newCy = ctrlRect.Width();
		ctrlRect.top = (int)((double)m_yRatio * 1.0 * pnRect->Width() -
newCy / 2.0);
		ctrlRect.bottom = ctrlRect.top + newCy; /* (int)((double)m_yRatio *
1.0 * pnRect->Width() + newCy / 2.0); */

	case CST_ZOOM:
		ctrlRect.top = (int)(1.0 * ctrlRect.top * (double)pnRect->Height() /
p 0-0->Height());
		ctrlRect.bottom = (int)(1.0 * ctrlRect.bottom * (double)pnRect-
>Height() / p 0-0->Height());
		bUpdateRect = 0;
		break;

	case CST_DELTA_ZOOM:
		newCy = ctrlRect.Height();
		ctrlRect.bottom = (int)(ctrlRect.top * 1.0 + delta0 * 1.0 * m_yRatio
+ newCy * 1.0 + delta0 * m_cyRatio);
		ctrlRect.top += (int)(delta0 * 1.0 * m_yRatio);
		bUpdateRect = 0;
		break;

	default:
		break;
	}

	if (!bUpdateRect) {
		pWnd->GetWindowRect(&curRect);
		pDlg->ScreenToClient(&curRect);
	}
	else {
```

```
            curRect = m_wRect;
      }

      if (ctrlRect != curRect) {
            if (m_bInvalidate) {
                  pWnd->Invalidate();
            }

            hDWP = ::DeferWindowPos(hDWP, (HWND)*pWnd, NULL, ctrlRect.left,
ctrlRect.top, ctrlRect.Width(), ctrlRect.Height(), SWP_NOZORDER);

#if 1 /* why No effect!!!! */
            if (m_bInvalidate) {
                  pDlg->InvalidateRect(&curRect);
                  pDlg->UpdateWindow();
            }
#endif      /* No effect???? */

            if (bUpdateRect) {
                  m_wRect = ctrlRect;
            }
      }

      return hDWP;
}

IMPLEMENT_DYNAMIC(CResizeDlg, CDialog)
BEGIN_MESSAGE_MAP(CResizeDlg, CDialog)
      ON_WM_SIZING()
      ON_WM_SIZE()
      ON_WM_GETMINMAXINFO()
      ON_WM_ERASEBKGND()
END_MESSAGE_MAP()

CResizeDlg::CResizeDlg(const UINT resID,CWnd *pParent)
        : CDialog(resID,pParent)
{
      m_xSt = CST_RESIZE;
      m_xSt = CST_RESIZE;
      m_xMin = 32;
      m_yMin = 32;
      m_nDelaySide = 0;
}

CResizeDlg::~CResizeDlg()
{
      }

BOOL CResizeDlg::OnInitDialog()
{
      BOOL bret = CDialog::OnInitDialog();

      CRect cltRect;
      CBitmap cBmpSize;
      BITMAP      Bitmap;

      GetClientRect(&cltRect);
      m_clt0 = cltRect;
      ClientToScreen(&m_clt0);
      m_cltRect = m_clt0;

      cBmpSize.LoadOEMBitmap(OBM_SIZE);
      cBmpSize.GetBitmap(&Bitmap);

      m_wndSizeIcon.Create( NULL,
                        WS_CHILD | WS_VISIBLE | SS_BITMAP,
                        CRect(0, 0, Bitmap.bmWidth, Bitmap.bmHeight),
                        this, m_idSizeIcon );
      m_wndSizeIcon.SetBitmap(cBmpSize);
```

```
      m_wndSizeIcon.SetWindowPos(&wndTop,
                        cltRect.right - Bitmap.bmWidth, cltRect.bottom -
Bitmap.bmHeight,
                        0, 0,
                        SWP_NOSIZE );
#if 0
      CRgn  cRgn;
      POINT bmpPt[3] = { { cltRect.right - Bitmap.bmWidth, cltRect.bottom },
                                 { cltRect.right, cltRect.bottom -
Bitmap.bmHeight},
                                 { cltRect.right, cltRect.bottom } };
      cRgn.CreatePolygonRgn(bmpPt, 3, WINDING);
      m_wndSizeIcon.SetWindowRgn(cRgn, TRUE);

      cRgn.Detach();
#endif

      cBmpSize.Detach();

      AddControl(m_idSizeIcon, CST_REPOS, CST_REPOS, CST_REPOS, CST_REPOS);

      CRect wRect;
      GetWindowRect(&wRect);
      m_xMin = wRect.Width();       // вбудована межа x
      m_yMin = wRect.Height();      // вбудована межа y

      return bret;
}

void CResizeDlg::OnSizing(UINT nSide, LPRECT lpRect)
{
      CDialog::OnSizing(nSide, lpRect);

      m_nDelaySide = nSide;
}

void CResizeDlg::OnSize(UINT nType,int cx,int cy)
{
      int    nCount;

      std::vector<CItemCtrl>::iterator it;

      CDialog::OnSize(nType,cx,cy);

      if((nCount = m_Items.size()) > 0) {
            CRect  cltRect;
            GetClientRect(&cltRect);
            ClientToScreen(cltRect);

            HDWP   hDWP;
            int    sizeType = WST_NONE;

#if 0
            int    delta = cltRect.Width() - m_cltRect.Width();
            int      delta = cltRect.Height()- m_cltRect.Height();
            int      mid = (cltRect.left + cltRect.right) / 2;
            int    mid = (cltRect.top + cltRect.bottom) / 2;
            CPoint csrPt(::GetMessagePos());

            if (delta) {
                  if (csrPt.x < mid)
                        sizeType |= WST_LEFT;
                  else
                        sizeType |= WST_RIGHT;
            }

            if (delta) {
                  if (csrPt.y < mid)
                        sizeType |= WST_TOP;
```

```
                  else
                        sizeType |= WST_BOTTOM;
            }
#else
            switch (m_nDelaySide) {
            case WMSZ_BOTTOM:
                  sizeType = WST_BOTTOM;
                  break;
            case WMSZ_BOTTOMLEFT:
                  sizeType = WST_BOTTOM|WST_LEFT;
                  break;
            case WMSZ_BOTTOMRIGHT:
                  sizeType = WST_BOTTOM|WST_RIGHT;
                  break;
            case WMSZ_LEFT:
                  sizeType = WST_LEFT;
                  break;
            case WMSZ_RIGHT:
                  sizeType = WST_RIGHT;
                  break;
            case WMSZ_TOP:
                  sizeType = WST_TOP;
                  break;
            case WMSZ_TOPLEFT:
                  sizeType = WST_TOP|WST_LEFT;
                  break;
            case WMSZ_TOPRIGHT:
                  sizeType = WST_TOP|WST_RIGHT;
                  break;
            default:
                  break;
            }
#endif

            if (sizeType != WST_NONE) {
                  hDWP = ::BeginDeferWindowPos(nCount);

                  for (it = m_Items.begin(); it != m_Items.end(); it++)
                        hDWP = it->OnSize(hDWP, sizeType, &cltRect, &m_cltRect,
&m_clt0, this);

                  ::EndDeferWindowPos(hDWP);
            }

            m_cltRect = cltRect;
      }

      m_nDelaySide = 0;
}

void CResizeDlg::OnGetMinMaxInfo(MINMAXINFO *pmmi)
{
      if ((HWND)m_wndSizeIcon == NULL)
            return;

      pmmi->ptMinTrackSize.x = m_xMin;
      pmmi->ptMinTrackSize.y = m_yMin;

      if (m_xSt == CST_NONE)
            pmmi->ptMaxTrackSize.x = pmmi->ptMaxSize.x = m_xMin;
      if (m_ySt == CST_NONE)
            pmmi->ptMaxTrackSize.y = pmmi->ptMaxSize.y = m_yMin;
}

BOOL CResizeDlg::OnEraseBkgnd(CDC *pDC)
{
      if (!(GetStyle() & WS_CLIPCHILDREN)) {
            std::vector<CItemCtrl>::const_iterator it;
```

```
            for(it = m_Items.begin();it != m_Items.end(); it++) {
                  // пропускається зміна іконки, якщо він скритий
                  if(it->m_nID == m_idSizeIcon &&
!m_wndSizeIcon.IsWindowVisible())
                        continue;

                  if(it->m_bFlickerFree && ::IsWindowVisible(GetDlgItem(it-
>m_nID)->GetSafeHwnd())) {
                        pDC->ExcludeClipRect(&it->m_wRect);
                  }
            }
      }

      CDialog::OnEraseBkgnd(pDC);
      return FALSE;
}

void CResizeDlg::AddControl( UINT nID,int xl, int xr, int yt, int yb, int
bFlickerFree,
                                                double xRatio, double cxRatio,
double yRatio, double cyRatio )
{
      CItemCtrl   item;
      CRect       cltRect;

      GetDlgItem(nID)->GetWindowRect(&item.m_wRect);
      ScreenToClient(&item.m_wRect);

      item.m_nID = nID;
      item.m_stxLeft = xl;
      item.m_stxRight = xr;
      item.m_styTop = yt;
      item.m_styBottom = yb;
      item.m_bFlickerFree = !!(bFlickerFree & 0x01);
      item.m_bInvalidate = !!(bFlickerFree & 0x02);
      item.m_xRatio = xRatio;
      item.m_cxRatio = cxRatio;
      item.m_yRatio = yRatio;
      item.m_cyRatio = cyRatio;

      GetClientRect(&cltRect);
      if (xl == CST_RELATIVE || xl == CST_ZOOM || xr == CST_RELATIVE || xr ==
CST_ZOOM)
            item.m_xRatio = (item.m_wRect.left + item.m_wRect.right) / 2.0 /
cltRect.Width();

      if (yt == CST_RELATIVE || yt == CST_ZOOM || yb == CST_RELATIVE || yb ==
CST_ZOOM)
            item.m_yRatio = (item.m_wRect.bottom + item.m_wRect.top ) / 2.0 /
cltRect.Height();

      std::vector<CItemCtrl>::iterator it;
      int  nCount;

      if((nCount = m_Items.size()) > 0) {
            for (it = m_Items.begin(); it != m_Items.end(); it++) {
                  if (it->m_nID == item.m_nID) {
                        *it = item;
                        return;
                  }
            }
      }

      m_Items.push_back(item);

}

void CResizeDlg::AllowSizing(int xst, int yst)
{
```

```
	m_xSt = xst;
	m_ySt = yst;
}

void CResizeDlg::HideSizeIcon(void)
{
	m_wndSizeIcon.ShowWindow(SW_HIDE);
}

int CResizeDlg::UpdateControlRect(UINT nID, CRect *pnr)
{
	std::vector<CItemCtrl>::iterator it;
	int  nCount;

	if ((nCount = m_Items.size()) > 0) {
		for (it = m_Items.begin(); it != m_Items.end(); it++) {
			if (it->m_nID == nID) {
				if (pnr != NULL) {
					it->m_wRect = *pnr;
				}
				else {
					GetDlgItem(nID)->GetWindowRect(&it->m_wRect);
					ScreenToClient(&it->m_wRect);
				}

				return 0;
			}
		}
	}

	return -1;
}
```

RuleDlg.cpp – Установка умов

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "SecurityPeerToPeerApp.h"
#include "RuleDlg.h"

#include "sockUtil.h"

// Ініціалізація дебаг інформації
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CRuleDlg діалог

//Опис конструктора
CRuleDlg::CRuleDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CRuleDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CRuleDlg)
  //Ініціалізація первісних значень
	m_ipsource = _T("0.0.0.0");
	m_portsource = 0;
	m_ipdestination = _T("0.0.0.0");
	m_portDestination = 0;
	m_action = _T("Tirar");
	m_protocol = _T("TCP");
	m_srcMask = _T("0.0.0.0");
	m_dstMask = _T("0.0.0.0");
	//}}AFX_DATA_INIT

	defaultAction = PF_ACTION_FORWARD;
}

//функція зчитування й установки даних вікна
void CRuleDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CRuleDlg)
	DDX_Control(pDX, IDC_COMBO3, m_actionCombo);
	DDX_Text(pDX, IDC_EDIT1, m_ipsource);
	DDV_MaxChars(pDX, m_ipsource, 15);
	DDX_Text(pDX, IDC_EDIT2, m_portsource);
	DDX_Text(pDX, IDC_EDIT3, m_ipdestination);
	DDV_MaxChars(pDX, m_ipdestination, 15);
	DDX_Text(pDX, IDC_EDIT4, m_portDestination);
	DDX_CBString(pDX, IDC_COMBO3, m_action);
	DDX_CBString(pDX, IDC_COMBO4, m_protocol);
	DDX_Text(pDX, IDC_EDIT6, m_srcMask);
	DDV_MaxChars(pDX, m_srcMask, 15);
	DDX_Text(pDX, IDC_EDIT5, m_dstMask);
	DDV_MaxChars(pDX, m_dstMask, 15);
	//}}AFX_DATA_MAP
}

//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CRuleDlg, CDialog)
	//{{AFX_MSG_MAP(CRuleDlg)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CRuleDlg заголовок повідомлень
```

```
//обробка подій (пост повідомлень) Windows
void CRuleDlg::OnOK()
{
	int result;

  //Зчитування, перевірка на коректність вводу
  //вивід помилок із описом проблем
	UpdateData(TRUE);

	result = inet_addr(m_ipsource, &srcIp);

	if(result == -1)
	{
		AfxMessageBox("Source Address isn't valid.");

		return;
	}

	if(srcIp == 0)
		srcMask = 0;
	else
	{
		result = inet_addr(m_srcMask, &srcMask);

		if(result == -1)
		{
			AfxMessageBox("Source Mask isn't valid.");

			return;
		}
	}


	result = inet_addr(m_ipdestination, &dstIp);

	if(result == -1)
	{
		AfxMessageBox("Destination Address isn't valid.");

		return;
	}

	if(dstIp == 0)
		dstMask = 0;

	else
	{
		result = inet_addr(m_dstMask, &dstMask);

		if(result == -1)
		{
			AfxMessageBox("Destination Mask isn't valid.");

			return;
		}
	}

	srcPort = m_portsource;
	dstPort = m_portDestination;

	if(m_protocol == "TCP")
		protocol = 6;

	else if(m_protocol == "UDP")
		protocol = 17;

	else if(m_protocol == "ICMP")
	{
		protocol = 1;
```

```
            if(srcPort == 0x00)
                  srcPort = 0xff;

            if(dstPort == 0x00)
                  dstPort = 0xff;

      }

      else
            protocol = 0;


      if(m_action == "")
      {
            AfxMessageBox("Select an action, please");

            return;
      }

      else
      {
            if(m_action == "Forward")
                  cAction = 0;

            else
                  cAction = 1;

      }

      //Виклик оброблювача події предка для завершення коректної реакції на
подію
      CDialog::OnOK();
}

//Ініціалізація вікна
BOOL CRuleDlg::OnInitDialog()
{
      CDialog::OnInitDialog();

  //Установка значень управління вікна
      if(defaultAction == PF_ACTION_DROP)
            m_actionCombo.AddString("Forward");

      else
            m_actionCombo.AddString("Drop");

      return TRUE;  //
}
```

sockUtil.cpp – Опис системних функцій по роботі з IP адресою перетворення форматів

```
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "sockutil.h"
#include <stdlib.h>
#include <string.h>

//Опис системних функцій по роботі з IP адресами й перетворення форматів.
int inet_addr(const char *sIp, unsigned long *lIp)
{
	int octets[4];
	int i;
	const char * auxCad = sIp;
	*lIp = 0;

	for(i = 0; i < 4; i++)
	{
		octets[i] = atoi(auxCad);

		if(octets[i] < 0 || octets[i] > 255)
			return -1;

		*lIp |= (octets[i] << (i*8));

		auxCad = strchr(auxCad, '.');

		if(auxCad == NULL && i!=3)
			return -1;

		auxCad++;
	}


	return 0;
}



unsigned short htons(unsigned short port)
{
	unsigned short portRet;

	portRet = ((port << 8) | (port >> 8));

	return portRet;
}

char *IpToString(char *ip, unsigned long lIp)
{
	char octeto[4];

	ip[0] = 0;

	itoa(lIp & 0xff, octeto, 10);

	strcat(ip, octeto);
	strcat(ip, ".");


	itoa((lIp >> 8) & 0xff, octeto, 10);

	strcat(ip, octeto);
	strcat(ip, ".");
```

```
        itoa((lIp >> 16) & 0xff, octeto, 10);

        strcat(ip, octeto);
        strcat(ip, ".");

        itoa((lIp >> 24) & 0xff, octeto, 10);

        strcat(ip, octeto);


        return ip;
}
```