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
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-21М-1,4
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Шевченко В.В.
«____» _____ 2022 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Шевченку Володимиру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Шевченко В.В.
(прізвище та ініціали)

АНОТАЦІЯ

Шевченко В.В. Дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи протоколів стеку TCP/IP у хмарних сервісах.

Метою розробки є дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

Об'єктом дослідження є процес протоколів стеку TCP/IP у хмарних сервісах.

Предметом дослідження є методи протоколів стеку TCP/IP у хмарних сервісах.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Visual C++.

Ключові слова: комп'ютерні науки, TCP/IP, хмарні сервіси

ABSTRACT

Shevchenko V.V. Research and software implementation of the TCP/IP stack protocol system in cloud services. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the TCP/IP stack protocol system in cloud services.

The goal of the development is the research and software implementation of the TCP/IP stack protocol system in cloud services.

The object of research is the process of TCP/IP stack protocols in cloud services.

The subject of research is the methods of TCP/IP stack protocols in cloud services.

Research methods are based on methods of cloud technologies, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the TCP/IP protocol stack system in cloud services.

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 software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

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

Keywords: computer science, TCP/IP, cloud services

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЛОМ	–	локальна обчислювальна мережа
MME	–	міжмережеві екрани
ATM	–	асинхронний режим передачі
BSD	–	адаптована для Internet реалізація операційної системи UNIX
ICMP	–	міжмережевий протокол управляючих повідомлень
IP	–	Internet Protocol – міжмережевий протокол
NFS	–	мережева файлова система
PPP	–	протокол передачі від точки до точки
RFC	–	опис набору протоколів Internet
RPC	–	віддалений виклик процедури
SLIP	–	міжмережевий 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. Коли з'єднується мережа з Internet або з іншою мережею, фактор забезпечення безпеки доступу у мережу має критичне значення. Додатки, створені зловмисниками, здатні проникнути по мережі на незахищений комп'ютер і запускатися непомітно. Вони можуть:

– Збирати й пересилати персональну інформацію (реквізити, паролі, номери кредитних карт і т.п.) своїм творцям або просто видаляти важливі дані.

– Використовувати заражений комп'ютер для розсилання спаму, поширення вірусів, злому вилучених сервісів, здійснення інших протиправних дій.

– Генерувати велику кількість паразитного трафіку й робити дзвінки на платні телефонні номери через модем.

– Показувати рекламні вікна й перенаправляти інтернет-браузер на рекламні сторінки.

– Підмінювати сторінки відомих сайтів на свої й використовувати це для фінансових махінацій (так званий "фішинг").

– Порушувати роботу інших програм і операційної системи в цілому.

Файрвол, для захисту протоколів стеку TCP/IP у хмарних сервісах – це програма, що представляє собою захисний бар'єр між комп'ютером і зовнішнім миром. Хакери використовують спеціальне програмне забезпечення для сканування інтернету й пошуку незахищених комп'ютерів. Такі програми посилають маленький пакет даних комп'ютеру. Якщо на комп'ютері немає файрволу, то він автоматично відповідає на прийняте повідомлення, і це означає для хакера, що система відкрита й може бути зламана. Файрвол розпізнає такі випадки й не відповідає на подібні повідомлення. Таким чином, хакери навіть не можуть довідатися, що Ваш комп'ютер підключений до мережі. Модуль

файрвола – найбільш інтелектуальна частина програмного продукту, забезпечує зв'язок пакетів між мережами з додатковими функціями, такими, як

– перетворення IP адрес (NAT – Network Address Translation) і номерів портів;

– перенапрямок;

– VPN (віртуальна захищена мережа);

– проксі;

– міст LAN в VPN;

– фільтрацію пакетів мережевих протоколів відповідно до заданих правил, часом, з розгалуженням вихідних маршрутів (спліттери);

– авторизацію, автентифікацію, аккаунтинг і біллінг користувачів (вхід у систему, надання користувачам прав на використання ресурсів мережі, ведення статистики використання трафика мережі).

Усередині локальної мережі, від зовнішніх погроз захищає корпоративний файрвол, робоча станція залишається беззахисною. Настроювання загального файрвола не дозволяють дозволити або заборонити активність тих або інших додатків, які запущені на робочій станції, а так само запобігти поширенню вірусів. За допомогою спеціальних типів атак зловмисник може в локальній мережі одержати будь-які дані, які передаються по мережі з комп'ютера. Переговори по ICQ, поштові паролі, листи й будь-яка інша конфіденційна інформація може бути перехоплена до того, як вона дійде до одержувача. Навіть у випадку якщо використовуються захищені з'єднання (SSL), однаково є відомі способи одержувати перехоплені дані відразу в розшифрованому виді. Більшість існуючих мереж використовує IP-протокол.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

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

– Огляд існуючих систем протоколів стеку TCP/IP у хмарних сервісах.

– Дослідження системи протоколів стеку TCP/IP у хмарних сервісах.

– Програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

Об'єктом дослідження є процес протоколів стеку TCP/IP у хмарних сервісах.

Предметом дослідження є методи протоколів стеку TCP/IP у хмарних сервісах.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод протоколів стеку TCP/IP у хмарних сервісах.

– Розроблено вітчизняний продукт протоколів стеку TCP/IP у хмарних сервісах, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі протоколів стеку TCP/IP у хмарних сервісах.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Розроблювальна система призначена для захисту протоколів стеку TCP/IP у хмарних сервісах від відалених атак. У магістерському проекті пропонується у якості такого захисту використовувати файрвол (fire wall).

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

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

– Контроль за додатками, що використовують порти. На відміну від звичайних міжмережевих 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 мережевої авторизації.

– Режим навчання, коли при першому звернені програми до мережевих 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 мережевих протоколів – від другого (перевірка на фальсифікацію

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

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

1.2 Область застосування

Розроблювальний програмний продукт – це комплексний інструмент для з'єднання локальної мережі з хмарним сервісом через глобальну мережу Інтернет і захисту мережі від несанкціонованого доступу.

Функції, які реалізуються в області застосування розроблювального продукту наступні:

– Прозорий доступ в Інтернет. Технологія передачі мережевої адреси (Network Address Translation (NAT)) дозволяє з'єднувати локальну мережу з Інтернет через одну загальну IP адресу (статичну або динамічну). На відміну від проксі серверів, технологія NAT робить доступними всі Інтернет-служби з будь-якої робочої станції. При цьому можна використовувати стандартні мережеві додатки, як якби всі комп'ютери локальної мережі мали власні з'єднання з Інтернет.

– Безпека. Інтегрований брандмауер захищає всю локальну мережу, включаючи робочу станцію, на якій він установлений, незалежно від того, чи використовується функція NAT (передача IP) або програма використовується як "нейтральний" маршрутизатор між двома мережами. Розроблювальний програмний продукт пропонує такий же стандарт безпеки, як і набагато більш дорогі програмні продукти.

– Контроль доступу. Всіма установками безпеки в розроблювальний програмний продукт можна управляти через так звані правила політики трафіку. Це забезпечує ефективний захист мережі від зовнішніх атак, при цьому забезпечуючи легкий доступ до всіх служб, що працюють на серверах у межах захищеної локальної мережі (наприклад, Web сервер, поштовий сервер, FTP сервер і ін.). Правила комунікації в політиці трафіку можуть обмежувати доступ локальних користувачів до певних служб в Інтернет.

– Підтримка протоколів (інспектори протоколів). Вам можуть зустрітися додатки, які не підтримують стандартні зв'язки, які можуть використовувати несумісні протоколи з'єднань, і т.д. Для рішення цієї проблеми програма включає інспектори протоколів (protocol inspectors), які визначають необхідний додатку протокол і динамічно модифікують роботу брандмауера.

– Конфігурація мережі. Програма має убудований сервер DHCP, що встановлює параметри TCP/IP для кожної робочої станції вашої локальної мережі. Параметри для кожної робочої станції можуть установлюватися централізовано з одного місця. Це зменшує витрати часу, необхідні для конфігурації мережі й мінімізує можливість помилки. Модуль DNS форвардер забезпечує легку конфігурацію DNS і прискорює відповіді на запити DNS. Це простий тип кешування імені сервера, при якому запити пересилаються іншому серверу DNS. Відповіді зберігаються в кеш-пам'яті. Це значно підвищує швидкість відповідей на часті запити. У комбінації із сервером DHCP і системними файлами вузлів, DNS форвардер може використовуватися як динамічний DNS сервер для локального домену.

– Віддалене адміністрування. Всі настроювання виконуються в адміністраторському терміналі (administration console), це незалежний адміністраторський термінал, використовуваний для керуванням всіма функціями сервера. Він може працювати й на робочій станції з Розроблювальний програмний продукт, і на іншому вузлі в межах локальної мережі або Інтернет.

Зв'язок між програмою й адміністраторським терміналом кодується й захищена від перехоплення або несанкціонованого використання.

– Різні операційні системи в межах локальної мережі. Розроблювальний програмний продукт працює зі стандартним протоколом TCP/IP. З погляду робочих станцій локальної мережі, він діє як стандартний маршрутизатор, при цьому не потрібно ніяких спеціальних додатків для клієнтів. Отже, у локальній мережі може працювати будь-яка операційна система з TCP/IP, наприклад, Windows, Unix/Linux, Mac OS і ін. Розроблювальний програмний продукт може працювати тільки з установками протоколу TCP/IP. Він не впливає на роботу інших протоколів (тобто IPX/SPX, NetBEUI, AppleTalk і ін.).

– Фільтр контенту. Розроблювальний програмний продукт може відслідковувати всі комунікації HTTP і FTP і блокувати об'єкти, що не відповідають установленим критеріям. Установки можуть бути глобальними або визначатися окремо для кожного користувача. Об'єкти, які скачали, можуть також прозоро перевірятися зовнішніми антивірусними додатками.

– Поштові повідомлення. Розроблювальний програмний продукт може відправляти користувачам поштові повідомлення, інформуючи їх про різні події. Ця функція полегшує роботу адміністратора, тому що не доводиться часто з'єднуватися з програмою і повністю її перевіряти. Всі відправлені повідомлення зберігаються в реєстраційному файлі.

– Статистика. Розроблювальний програмний продукт дозволяє переглядати докладну статистику інтерфейсу брандмауера (поточна швидкість передачі даних, кількість переданої інформації за певний період) і окремих користувачів (кількість переданої інформації, використовувані служби, категорії відвідуваних сайтів і ін.).

– Особистий VPN сервер і клієнт. Розроблювальний програмний продукт також вирішує проблему особистого VPN, якому можна використовувати в режимах сервер-сервер і клієнт-сервер. VPN може з обох сторін використовувати NAT (навіть множинний). Програма VPN Client включена в розроблювальний

програмний продукт і може використовуватися для створення клієнт-сервер VPN (з'єднання віддалених клієнтів з локальною мережею).

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Для того, щоб захищати протоколи стеку TCP/IP у хмарних сервісах, розглянемо конкретні приклади атак на TCP/IP.

Пасивні атаки на рівні TCP

При даному типі атак зловмисники ніяким образом не виявляють себе й не вступають прямо у взаємодію з іншими системами. Фактично все зводитися до спостереження за доступними даними або сесіями зв'язку.

Підслуховування

Атака полягають у перехопленні мережевого потоку і його аналізі (англомовний термін – "sniffing"). Для здійснення підслуховування зловмиснику необхідно мати доступ до машини, розташованої на шляху мережевого потоку, якому необхідно аналізувати; наприклад, до маршрутизатора або PPP-серверу на базі UNIX. PPP (point-to-point protocol) – протокол передачі від точки до точки, протокол двуточечного з'єднання (набір протоколів фреймування й автентификації, що є частиною сервісу RAS системи Windows NT; зв'язує конфігураційні параметри численних рівнів моделі OSI). UNIX – багатокористувальницька багатозадачна операційна система, спочатку розроблена Кеном Томпсоном (Ken Thompson) і Денисом Ритчи (Dennis Ritchie) у компанії AT&T Bell Laboratory в 1969 р. для використання в міні-комп'ютерах; у цей час існує в різних формах і реалізаціях; уважається потужною операційною системою, що менш машинозалежна, ніж інші операційні системи; написана мовою С. Якщо зловмиснику вдасться одержати достатні права на цій машині, то за допомогою спеціального програмного забезпечення зможе переглядати весь

трафік, що проходить через заданий інтерфейс.

Другий варіант – зловмисник одержує доступ до машини, що розташована в одному сегменті мережі із системою, який має доступ до мережевого потоку. Наприклад, у мережі "тонкий ethernet" мережева карта може бути переведена в режим, у якому вона буде одержувати всі пакети, що циркулюють по мережі, а не тільки адресованої їй конкретно. У цьому випадку зловмиснику не потрібен доступ до UNIX – досить мати PC з DOS або Windows (часта ситуація в університетських мережах)

Оскільки TCP/IP-трафік, як правило, не шифрується (ми розглянемо виключення нижче), зловмисник, використовуючи відповідний інструментарій, може перехоплювати TCP/IP-пакети, наприклад, telnet-сесій і витягати з них імена користувачів і їхні паролі.

Варто помітити, що даний тип атаки неможливо відстежити, не маючи доступ до системи зловмисника, оскільки мережевий потік не змінюється. Єдиний надійний захист від підслуховування – шифрування TCP/IP-потоку (наприклад, secure shell – захищена оболонка) або використання одноразових паролів (наприклад, S/KEY)

Інший варіант рішення – використання інтелектуальних комутаторів і UTP, у результаті чого кожна машина одержує тільки той трафік, що адресовано їй.

У кожного ціпка два кінці. Природно, підслуховування може бути й корисно. Так, даний метод використовується великою кількістю програм, що допомагають адміністраторам в аналізі роботи мережі (її завантаженості, працездатності й т.д.). Один з яскравих прикладів – загальновідомий tcpdump

Активні атаки на рівні TCP

При даному типі атак зловмисник взаємодіє з одержувачем інформації, відправником і/або проміжними системами, можливо, модифікуючи й/або фільтруючи вміст TCP/IP-пакетів. Дані типи атак часто здаються технічно складними в реалізації, однак для гарного програміста не становить праці

реалізувати відповідний інструментарій. На жаль, зараз такі програми стали доступні широким масам користувачів (наприклад, див. SYN-затоплення).

Активні атаки можна розділити на дві частини. У першому випадку зловмисник уживає певних заходів для перехоплення й модифікації мережевого потоку або спроб "прикинутися" іншою системою. У другому випадку протокол TCP/IP використовується для того, щоб привести систему-жертву в неробочий стані.

Маючи достатні привілеї в Unix (або попросту використовуючи DOS або Windows, що не мають системи обмежень користувачів), зловмисник може вручну формувати IP-пакети й передавати їх по мережі. Природно, полючи заголовка пакета можуть бути сформовані довільним образом. Одержавши такий пакет, неможливо з'ясувати звідки реально він був отриманий, оскільки пакети не містять шляхи їхнього проходження. Звичайно, при установці зворотної адреси не співпадаючим з поточної IP-адресою, зловмисник ніколи не одержить відповідь на відісланий пакет. Однак, як ми побачимо, часто це й не потрібно.

Можливість формування довільних IP-пакетів є ключовим пунктом для здійснення активних атак.

Пророкування порядкового номера TCP

Дана атака була описана ще Робертом Моррисом (Robert T. Morris) в A Weakness in the 4.2BSD Unix TCP/IP Software. Англомовний термін – IP-spoofing (IP-spoofing одержання доступу шляхом обману (ситуація, коли користувач намагається з'єднатися із сервером Internet, proxy-сервером або брандмауером, використовуючи помилкову IP-адресу)). У цьому випадку ціль зловмисника – прикинутися іншою системою, якій, наприклад, "довіряє" система-жертва (у випадку використання протоколу rlogin/rsh для безпарольного входу). Метод також використовується для інших цілей – наприклад, для використання SMTP жертви для посилки підроблених листів.

Згадаємо, що установка TCP-з'єднання відбувається в три стадії (3-way handshake): клієнт вибирає й передає серверу порядковий номер (назвемо його C-

SYN), у відповідь на це сервер висилає клієнтові пакет даних, що містить підтвердження (C-ACK) і власний порядковий номер сервера (S-SYN). Тепер уже клієнт повинен вислати підтвердження (S-ACK). Схематично це можна представити так:

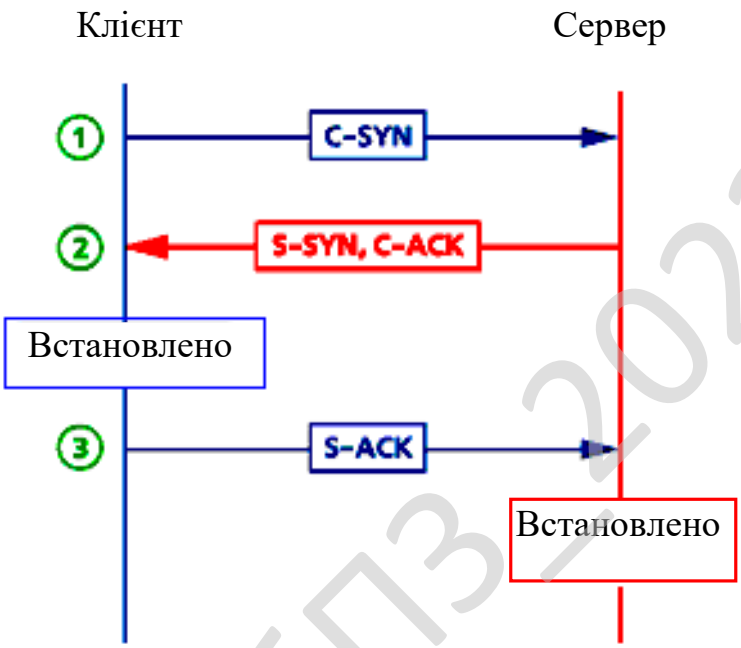

Рисунок 2.1 – Установка TCP-з'єднання

Після цього з'єднання вважається встановленим і починається обмін даними. При цьому кожний пакет має в заголовку поле для порядкового номера й номера підтвердження. Дані числа збільшуються при обміні даними й дозволяють контролювати коректність передачі.

Припустимо, що зловмисник може пророчити, який порядковий номер (S-SYN за схемою) буде висланий сервером. Це можливо зробити на основі знань про конкретну реалізацію TCP/IP. Наприклад, в 4.3BSD значення порядкового номера, що буде використано при установці наступного значення, щосекунди збільшується на 125000. Таким чином, послався один пакет серверу, зловмисник одержить відповідь і зможе (можливо, з декількох спроб і з виправленням на швидкість з'єднання) пророчити порядковий номер для наступного з'єднання.

Якщо реалізація TCP/IP використовує спеціальний алгоритм для визначення порядкового номера, то він може бути з'ясований за допомогою посилки декількох десятків пакетів серверу й аналізу його відповідей.

Отже, припустимо, що система А довіряє системі В, так, що користувач системи В може зробити "rlogin A" (вхід у систему А) і виявитися на А, не вводячи пароля. Припустимо, що зловмисник розташований на системі С. Система А виступає в ролі сервера, системи В і С – у ролі клієнтів.

Перша задача зловмисника – ввести систему В у стан, коли вона не зможе відповідати на мережеві запити. Це може бути зроблено декількома способами, у найпростішому випадку потрібно просто дочекатися перезавантаження системи В. Декількох хвилин, у плині яких вона буде непрацездатна, повинно вистачити. Інший варіант – використання описаними в наступних розділах методів.

Після цього зловмисник може спробувати прикинутися системою В, для того, що б одержати доступ до системи А (хоча б короткочасний).

Зловмисник висилає декілька IP-пакетів, що ініціюють з'єднання, системі А, для з'ясування поточного стану порядкового номера сервера.

Зловмисник висилає IP-пакет, у якому як зворотна адреса зазначена вже адреса системи В.

Система А відповідає пакетом з порядковим номером, що направляється системі В. Однак система В ніколи не одержить його (вона виведена з ладу), як, втім, і зловмисник. Але він на основі попереднього аналізу догадується, який порядковий номер був висланий системі В

Зловмисник підтверджує "одержання" пакета від А, виславши від імені В пакет з передбачуваним S-ACK (помітимо, що якщо системи розташовуються в одному сегменті, зловмиснику для з'ясування порядкового номера досить перехопити пакет, посланий системою А). Після цього, якщо зловмиснику повезло й порядковий номер сервера був вгаданий вірно, з'єднання вважається встановленим.

Тепер зловмисник може вислати черговий фальшивий IP-пакет, що буде

вже містити дані. Наприклад, якщо атака була спрямована на rsh, він може містити команди створення файлу .rhosts або відправлення /etc/passwd зловмиснику по електронній пошті.

Представимо це у вигляді схеми:

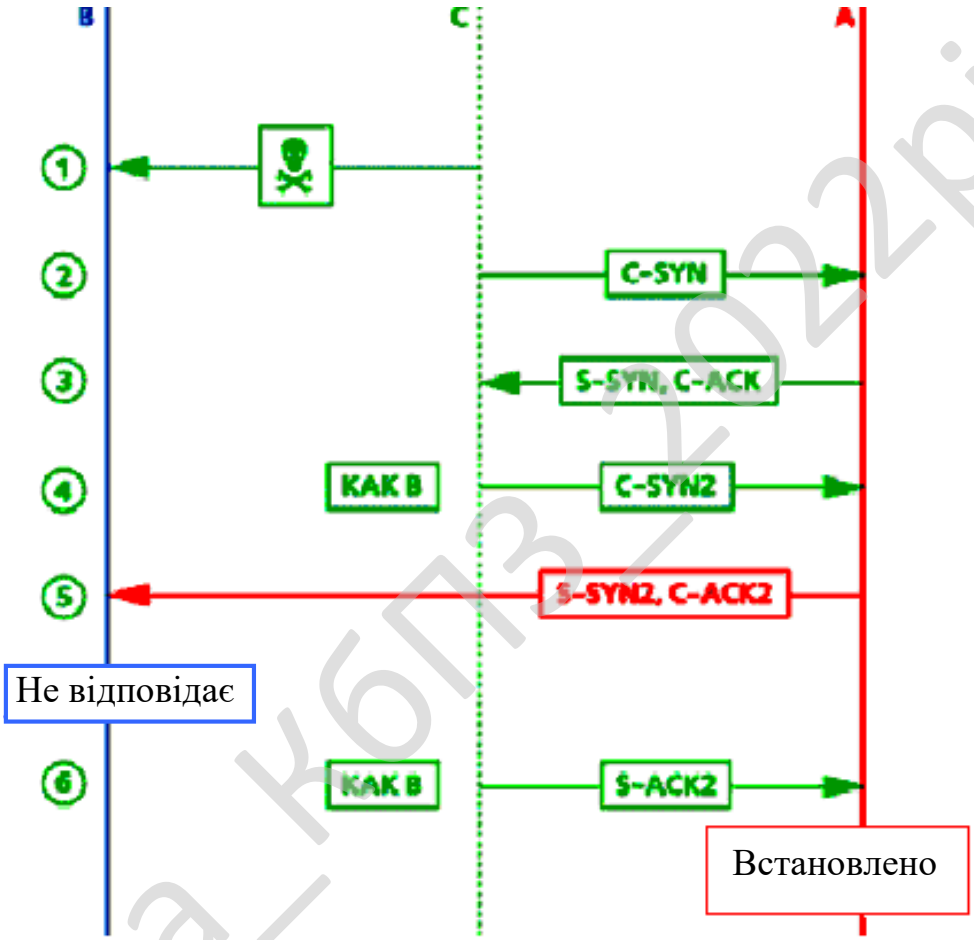

Рисунок 2.2 – IP-spoofing

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 міжмережевий 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 мережевий потік, модифікуючи його й фільтруючи довільним образом. Метод є комбінацією 'підслуховування' і IP-

спуфінгу.

Необхідні умови – зловмисник повинен мати доступ до машини, що перебуває на шляху мережевого потоку й мати достатні права на ній для генерації й перехоплення IP-пакетів.

Нагадаємо, що при передачі даних постійно використовуються порядковий номер і номер підтвердження (обоє поля перебувають в IP-заголовку). Виходячи з їхнього значення, сервер і клієнт перевіряють коректність передачі пакетів.

Існує можливість увести з'єднання в "десінхронизированное стан", надсилаються коли сервером порядковий номер і номер підтвердження не будуть збігатися з очікуваним значенями клієнта, і навпаки. У цьому випадку зловмисник, "прослуховуючи" лінію, може взяти на себе функції посередника, генеруючи коректні пакети для клієнта й сервера й перехоплюючи їхні відповіді.

Метод дозволяє повністю обійти такі системи захисту, як, наприклад, одноразові паролі, оскільки зловмисник починає роботу вже після того, як відбудеться авторизація користувача.

Є два способи розсинхронізувати з'єднання:

– Рання десинхронізація.

– Десинхронізація нульовими даними.

Рання десинхронізація

З'єднання десинхронізується на стадії його установки.

Зловмисник прослуховує сегмент мережі, по якому будуть проходити пакети його сесії, що цікавить.

Дочекавшись пакета S-SYN від сервера, зловмисник висилає серверу пакет типу RST (скидання), звичайно, з коректним порядковим номером, і, негайно, слідом за ним фальшивий C-SYN-пакет від імені клієнта.

Сервер скидає першу сесію й відкриває нову, на тому же порту, але вже з новим порядковим номером, після чого посилає клієнтові новий S-SYN-пакет.

Клієнт ігнорує S-SYN-пакет, однак зловмисник, що прослуховує лінію,

висилає серверу S-ACK-пакет від імені клієнта.

Отже, клієнт і сервер перебувають в встановленому стані ESTABLISHED (установлене), однак сесія десинхронізована.

Представимо це у вигляді схеми:

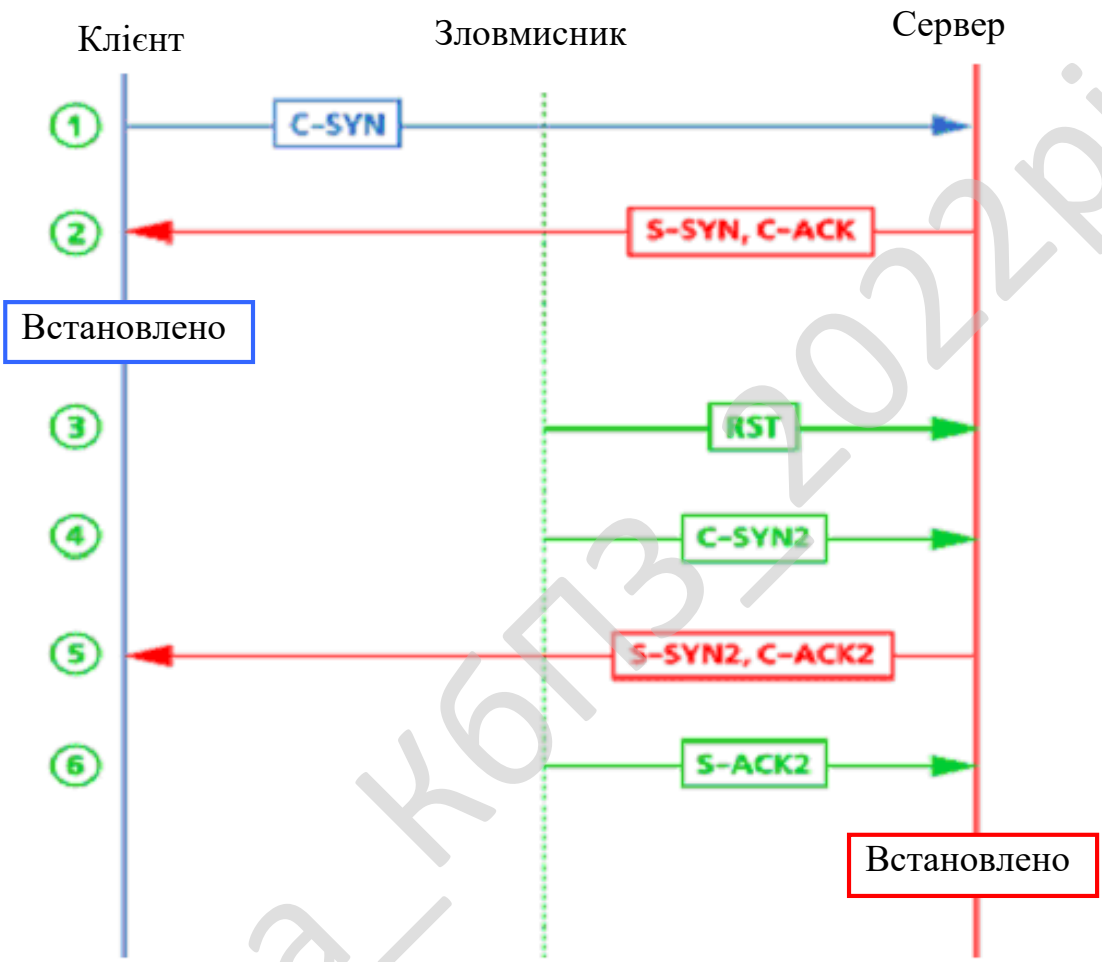

Рисунок 2.3 – Рання десинхронізація

Природно, 100% спрацьовування в цієї схеми ні, наприклад, вона не застрахована від того, що по дорозі не втратяться якісь пакети, послані зловмисником. Для коректної обробки цих ситуацій програма повинна бути ускладнена.

Десинхронізація нульовими даними

У цьому випадку зловмисник прослуховує сесію й у якийсь момент посилає серверу пакет з "нульовими" даними, тобто такими, які фактично будуть

зігноровані на рівні прикладної програми й не видні клієнтові. Аналогічний пакет посилає клієнтові. Очевидно, що після цього сесія переходить у десинхронізованное стан.

ACK-буря

Одна із проблем IP Hijacking полягає в тім, що будь-який пакет, висланий у момент, коли сесія перебуває в десинхронізованном стані викликає так званий ACK-бурю. Наприклад, пакет висланий сервером, і для клієнта він є неприйнятним, тому той відповідає ACK-пакетом. У відповідь на цей неприйнятний уже для сервера пакет клієнт знову одержує відповідь... І так нескінченно.

Але сучасні мережі будуються за технологіями, коли допускається втрата окремих пакетів. Оскільки ACK-пакети не несуть даних, повторні передачі не відбувається й "буря стихає".

Як показали досвіди, ніж сильніше ACK-буря, тим швидше вона "утихомирює" себе – на 10MB ethernet це відбувається за частки секунди. На ненадійних з'єднаннях типу SLIP – ненабагато більше. SLIP (Serial Line Internet Protocol) – міжмережевий протокол для послідовного каналу (протокол Internet, що забезпечує можливість реалізації мережевих протоколів при з'єднанні двох систем послідовними (телефонними) лініями; у цей час замість SLIP в основному використовується протокол PPP)

Детектування й захист

Є кілька шляхів. Наприклад, можна реалізувати TCP/IP-стек, які будуть контролювати перехід у десинхронізований стан, обмінюючись інформацією про порядковий номер/номері підтвердження. Однак у цьому випадку ми не застраховані від зловмисника, що міняє й ці значення.

Тому більше надійним способом є аналіз завантаженості мережі, відстеження виникаючих ACK-бурь. Це можна реалізувати за допомогою конкретних засобів контролю за мережею.

Якщо зловмисник не потрудитися підтримувати десинхронізоване

з'єднання до його закриття або не стане фільтрувати вивід своїх команд, це також буде відразу замічено користувачем. На жаль, переважна більшість просто відкриють нові сесії, не звертаючись до адміністратора.

Стовідсотковий захист від даної атаки забезпечує, як завжди, шифрування TCP/IP-трафіку (на рівні додатків – secure shell) або на уровн протоколу – IPsec). Це виключає можливість модифікації мережевого потоку. Для захисту поштових повідомлень може застосовуватися PGP.

Варто помітити, що метод також не спрацьовує на деяких конкретних реалізаціях TCP/IP. Так, деякі системи генерують зустрічний RST-пакет. Це унеможливлює ранню десинхронізацію.

Пасивне сканування

Сканування часто застосовується зловмисниками для того, щоб з'ясувати, на яких TCP-портах працюють демони, що відповідають на запити з мережі. Звичайна програма-сканер послідовно відкриває з'єднання з різними портами. У випадку, коли з'єднання встановлюється, програма скидає його, повідомляючи номер порту зловмиснику.

Даний спосіб легко детектується за повідомленнями демонів, здивованих миттєво прерваним після установки з'єднанням, або за допомогою використання спеціальних програм. Кращі з таких програм мають деякі спроби внести елементи штучного елемента у відстеження спроб з'єднання з різними портами.

Однак зловмисник може скористатися іншим методом – пасивним скануванням (англійський термін "passive scan"). При його використанні зловмисник посилає TCP/IP SYN-пакет на всі порти підряд (або по якомусь заданому алгоритму). Для TCP-портів, що приймають з'єднання ззовні, буде повернутий SYN/ ACK-пакет, як запрошення продовжити 3-way handshake. Інші повернуть RST-пакети. Проаналізувавши дані відповідь, зловмисник може швидко зрозуміти, на яких портах працюють програма. У відповідь на SYN/ ACK-пакети він може також відповісти RST-пакетами (скидання), показуючи, що процес установки з'єднання продовжений не буде (у загальному випадку RST-

пакетами автоматичний відповість TCP/IP-реалізація зловмисника, якщо він не почне спеціальних мір).

Метод не детектується попередніми способами, оскільки реальне TCP/IP-з'єднання не встановлюється. Однак (залежно від поводження зловмисника) його можно відслідковувати:

– різко зросла кількість сесій, що перебувають у стані SYN_RECEIVED. (SIN пакет прийнятий) за умови, що зловмисник не посилає у відповідь RST);

– прийом від клієнта RST-пакета у відповідь на SYN/ACK.

На жаль, при досить розумному поводженні зловмисника (наприклад, сканування з низькою швидкістю або перевірка лише конкретних портів) детектировать пасивне сканування неможливо, оскільки воно нічим не відрізняється від звичайних спроб установити з'єднання.

Як захист можна лише порадити закрити на firewall всі сервіси, доступ до яких не потрібно ззовні.

Затоплення ICMP-пакетами

Традиційний англомовний термін – "ping flood". З'явився він тому, що програма "ping", призначена для оцінки якості лінії, має ключ для "агресивного" тестування. У цьому режимі запити посилають із максимально можливою швидкістю й програма дозволяє оцінити, як працює мережа при максимальному навантаженні. Дана атака жадає від зловмисника доступу до швидких каналів в Інтернет.

Згадаємо, як працює ping. Програма посилає ICMP-пакет типу ECHO REQUEST (запит відгуку ICMP), виставляючи в ньому час і його ідентифікатор. Ядро машини-одержувача відповідає на подібний запит пакетом ICMP ECHO REPLY (відповідь на відгук ICMP) . Одержавши його ping видає швидкість проходження пакета.

При стандартному режимі роботи пакети висилаються через деякі проміжки часу, практично не навантажуючи мережу. Але в "агресивному" режимі потік ICMP echo request/ reply-пакетів може викликати перевантаження невеликої

лінії, позбавивши її здатності передавати корисну інформацію.

Природно, випадок з ping є часткою случаємо більше загальної ситуації, пов'язаний з перевантаженням каналів. Наприклад, зловмисник може посилати множину UDP-пакетів на 19-й порт машини-жертви, і горі їй, якщо вона, дотримуючись загальноприйнятих правил, має на 19-м UDP-порту знакогенератор, що відповідає на пакети рядками по 80 байт. UDP (User Datagram Protocol) протокол передачі дейтаграм користувача (швидкий мережевий протокол транспортного рівня Internet, на якому базуються мережева файлова система, служба імен і ряд інших служб, однак, на відміну від TCP, UDP забезпечує обмін дейтаграмами без підтвердження доставки).

Помітимо, що зловмисник може також підробляти зворотну адресу подібних пакетів, утрудняючи його виявлення. Відстежити його допоможе хіба що скоординована робота фахівців на проміжних маршрутизаторах, що практично нереально.

Однієї з варіантів атаки – посилати ICMP echo request-пакети з вихідною адресою, що вказує на жертву, на широкомовні-адреси великих мереж. У результаті кожна з машин відповість на цей фальшивий запит, і машина-відправник одержить більше кількість відповідей. Посилка множина echo requests від імені "жертви" на широкомовні-адреси великих мереж, можна викликати різке заповнення каналу "жертви".

Прикмети затоплення – різко зросле навантаження на мережу (або канал) і підвищення кількість специфічних пакетів (таких, як ICMP).

Як захист можна порекомендувати настроювання маршрутизаторів, при яких вони будуть фільтрувати той же ICMP трафік, що перевищують деяку задану заздалегідь величину (пакетів/од. часу). Для того, щоб переконатися, що Ваші машини не можуть служити джерелом ping flood'a, обмежте доступ до ping.

Локальна буря

Зробимо невеликий відступ убік реалізації TCP/IP і розглянемо "локальні бури" на приклад UDP-бурі. Як правило, за замовчуванням системи підтримують

роботу таких UDP-портів, як 7 ("луна", отриманий пакет відсилається назад), 19 ("знакогенератор", у відповідь на отриманий пакет відправникові висилається рядок знакогенератора) і інших.

У цьому випадку зловмисник може послати єдиний UDP-пакет, де як вихідний порт буде зазначений 7, як одержувач – 19-й, а як адреса одержувача й відправника будуть зазначені, приміром, дві машини вашої мережі (або навіть 127.0.0.1). Одержавши пакет, 19-й порт відповідає рядком, що попадає на порт 7. Сьомий порт дублює її й знову відсилає на 19.. і так нескінченно.

Нескінченний цикл з'їдає ресурси машин і додає на канал безглузде навантаження. Звичайно, при першому загубленому UDP-пакеті буря припинитися. Як недавно стало відомо, дана атака тимчасово виводить із ладу (до перезавантаження) деякі старі моделі маршрутизаторів.

Очевидно, що в нескінченну розмову можуть бути залучені багато демонів (у випадку TCP/IP може бути застосований TCP/IP spoofing, у випадку UDP/ICMP досить пари фальшивих пакетів)

Як захист коштує ще раз порекомендувати не пропускати в мережі пакети із внутрішніми адресам, але пришедшие ззовні. Також рекомендується закрити на firewall використання більшості сервісів.

Затоплення SYN-пакетами

Мабуть, затоплення SYN-пакетами ("SYN flooding") – найвідоміший спосіб нападу, відтоді, як хакерский електронний журнал "2600" опублікував вихідні тексти програми, що дозволяють занятись цим навіть некваліфікованим користувачам. Варто помітити, що вперше ця атака була згадана ще в 1986 році всі тим же Робертом Т. Моррисом.

Згадаємо, як працює TCP/IP у випадку вхідних з'єднань. Система відповідає на що прийшов C-SYN-пакет S-SYN/C-ACK-пакетом, переводить сесію в стан SYN_RECEIVED (SYN пакет прийнятий) і заносить її в чергу. Якщо в плині заданого часу від клієнта не прийде S-ACK, з'єднання видаляється із черги, у противному випадку з'єднання переводиться у встановлений стан

ESTABLISHED.

Розглянемо випадок, коли черга вхідних з'єднань уже заповнена, а система одержує SYN-пакет, що запрошує до установки з'єднання. По RFC він буде мовчачи зігнорований. RFC (Requests for Comments) – серія документів IETF, почата в 1969 році й утримуюча опис набору протоколів Internet і пов'язану з ними інформацію

Затоплення SYN-пакетами засновано на переповненні черги сервера, після чого сервер перестає відповідати на запити користувачів. Найвідоміша атака такого роду – атака на Panix, нью-йоркського провайдера. Panix не працював у плині 2-х тижнів.

У різних системах робота із чергою реалізована по різному. Так, в BSD-системах, кожний порт має свою власну чергу розміром в 16 елементів. BSD (Berkeley Software Distribution) – програмний виріб Каліфорнійського університету (адаптована для Internet реалізація операційної системи UNIX з комплектом її утиліт, розроблювальних і розповсюджуваних університетом) У системах SunOS, навпроти, такого поділу й немає й система просто має у своєму розпорядженні велику загальну чергу. Відповідно, для того, що б заблокувати, приміром, WWW-порт на BSD досить 16 SYN-пакетів, а для Solaris 2.5 їхня кількість буде набагато більше.

Після витікання деякого часу (залежить від реалізації) система видаляє запити із черги. Однак нічого не заважає зловмиснику послати нову порцію запитів. Таким чином, навіть перебуваючи на з'єднання 2400 bps, зловмисник може посилати кожні півтори хвилини по 20-30 пакетів на FreeBSD-сервер, підтримуючи його в неробочому стані (природно, ця помилка була скоректована в останніх версіях FreeBSD).

Як звичайно, зловмисник може скористатися випадковими зворотними IP-адресами при формуванні пакетів, що утрудняє його виявлення й фільтрацію його трафіку.

Детектування нескладно – велика кількість з'єднань у стані

SYN_RECEIVED (SYN пакет прийнятий), ігнорування спроб з'єднається з даним портом. Як захист можна порекомендувати патчі, які реалізують автоматичне "прорідження" черги. Для того, що б довідатися, є у системи захист від SYN-затоплення, треба звернутися до постачальника системи.

Інший варіант захисту – настроїти мережевий екран firewall так, що б всі вхідні TCP/IP-з'єднання встановлював він сам, і тільки після цього перекидав їх усередину мережі на задану машину. Це дозволить обмежити syn-затоплення й не пропустити його усередину мережі.

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 застосунків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка застосунків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки застосунків Windows дозволяють автоматизувати процес створення застосунка. Для цього

використовуються генератори застосунків. Програміст відповідає на питання генератора застосунків і визначає властивості застосунка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор застосунків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої застосунки. Подібні засоби автоматизованого створення застосунків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики застосунка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні застосунки, а також застосунки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину застосунка програмістові прийдеться розробляти самостійно. Вихідний текст застосунка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон застосунка – це вже половина всієї роботи. Вихідні тексти застосунків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти застосунків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої застосунки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-застосунки на базі об’єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows

вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який 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 другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи протоколів стеку TCP/IP у хмарних сервісах.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Існує три основних типи FireWall (файрволу, міжмережевих екранів) – пакетний фільтр (packet filtering), шлюз на сеансовому рівні (circuit-level gateway) і шлюз на прикладному рівні (application-level gateway). Далеко не всі існуючі міжмережеві екрани можуть бути однозначно віднесені до одному з названих типів. Як правило, MME сполучає в собі функції двох або трьох типів.

Як ми вже відзначали, для того, щоб ефективно забезпечувати безпеку мережі, firewall зобов'язаний відслідковувати й управляти всім потоком, що проходить через нього. Для прийняття керуючих рішень для TCP/IP-сервісів (тобто передавати, блокувати або відзначати в журналі спроби встановлення з'єднань), firewall повинен одержувати, запам'ятовувати, вибирати й обробляти інформацію, отриману від всіх комунікаційних рівнів і від інших додатків.

Недостатньо просто перевіряти пакети окремо. Інформація про стан з'єднання, отримана з інспекції з'єднань у минулому й інших додатках – головний фактор в ухваленні керуючого рішення при спробі встановлення нового з'єднання. Для ухвалення рішення можуть ураховуватися як стан з'єднання (отримане з минулого потоку даних), так і стан додатка (отримане з інших додатків).

Отже рішення, що управляють вимагають щоб firewall мав доступ, можливість аналізу й використання наступних речей:

1. Інформація про з'єднання – інформація від всіх семи рівнів у пакеті.

2. Історія з'єднань – інформація, отримана від попередніх з'єднань. Наприклад, що виходить команда PORT сесії FTP повинна бути збережена для того, щоб надалі з його допомогою можна було перевірити вхідне з'єднання FTP data.

3. Стан рівня додатка – інформація про стан, отриманий з інших додатків. Наприклад, автентифікованому до цього моменту користувачеві можна надати доступ через firewall тільки для авторизованих видів сервісу.

4. Маніпулювання інформацією – обчислення різноманітних виражень, заснованих на всіх перерахованих вище факторах.

Порівняння альтернатив

У цьому розділі описані границі, у яких доступні технології firewall забезпечують ці чотири своїх основних властивості.

Маршрутизатори

Маршрутизатори діють на мережевому рівні і їхнім очевидним недоліком є нездатність забезпечувати безпеку навіть для найбільш відомих сервісів і протоколів. Маршрутизатори не є пристроями забезпечення безпеки, тому що вони не мають основних можливостей firewall:

1. Інформація про з'єднання – маршрутизатори мають доступ лише до обмеженої частини заголовка пакетів.

2. Наслідувана інформація про з'єднання й додаток – маршрутизатори не підтримують зберігання інформації про історію з'єднання або додатка.

3. Дії над інформацією – маршрутизатори мають дуже обмежені можливості по діях над інформацією.

До того ж, маршрутизатори важко конфігурувати, стежити за їхнім станом і управляти. Вони не забезпечують належного рівня журналювання подій і механізмів оповіщення.

Proxy

Proxy – модуль доступу (наприклад до мережі Internet); програма-посередник, агент (механізм, за допомогою якого одна система представляє іншу у відповідь на запити протоколу; proxy-системи використовуються в мережевому керуванні, щоб позбутися від необхідності реалізації повного стека протоколів для таких простих пристроїв, як модеми)

Proxy є спробою реалізувати firewall на рівні додатка. Її основна

перевага – підтримка повної інформації про додатки. Proxy забезпечують часткову інформацію про історію з'єднань, повну інформацію про додаток і часткову інформацію про поточне з'єднання. Proxy також мають можливість обробки й дій над інформацією.

Однак, є очевидні труднощі у використанні proxy на рівні додатка в якості firewall, включаючи:

1. Обмеження на з'єднання – кожний сервіс вимагає наявності свого власного proxy, так що число доступних сервісів і їхня масштабованість обмежені.

2. Обмеження технології – шлюзи прикладного рівня не можуть забезпечити proxy для UDP, RPC і інших сервісів загальних сімейств протоколів. RPC (Remote Procedure Call) – віддалений виклик процедури (засіб передачі повідомлень, що дозволяє розподіленому додатку викликати сервіс різних комп'ютерів у мережі; забезпечує процедурно-орієнтований підхід у роботі з мережею; застосовується в розподілених об'єктних технологіях, таких як, DCOM, CORBA, Java RMI)

3. Продуктивність – реалізація на рівні додатка має значні втрати в продуктивності.

У додавання, proxy беззахисні до помилок у додатках і ОС, невірної інформації в нижніх рівнях протоколів і у випадку традиційних proxy дуже рідко є прозорими.

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 мережевому 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 мережевому 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".

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 (рисунок 3.1).

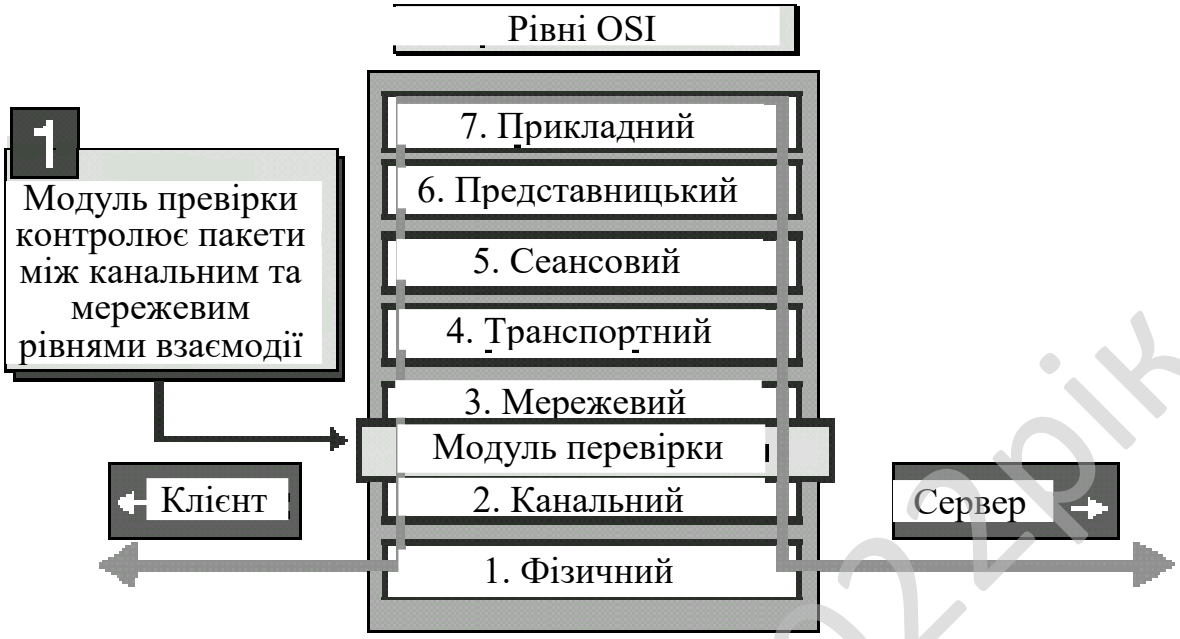

Рисунок 3.1 – З'єднання управляється модулем перевірки FireWall-1

Взаємодія FireWall-1 з різними протколами

– 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)").

Автентифікація на сервері безпеки й безпека змісту

FireWall-1 дозволяє адміністраторові визначати політикові безпеки для кожного користувача, де не тільки джерело з'єднання, призначення й сервіс

перевіряються, але й кожний користувач повинен бути автентифікован. Більше того, з'єднання можуть бути дозволені або заборонені виходячи з їхнього змісту. Приміром, пошта для або від певних адрес може бути заборонена або переспрямована, доступ може бути заборонений до заданих URL, і включена антивірусна перевірка над переданими файлами. URL (uniform resource locator) – уніфікований покажчик інформаційного ресурсу (стандартизований рядок символів, що вказує місцезнаходження документа в мережі Internet).

Автентифікація й перевірка вмісту забезпечуються набором серверів безпеки FireWall-1, що працюють на рівні додатка (рисунок 3.2).

Коли працює сервер безпеки FireWall-1, модуль перевірки перенаправляє всі пакети з'єднання до сервера безпеки, що виконує необхідну автентифікацію й/або перевірку вмісту.

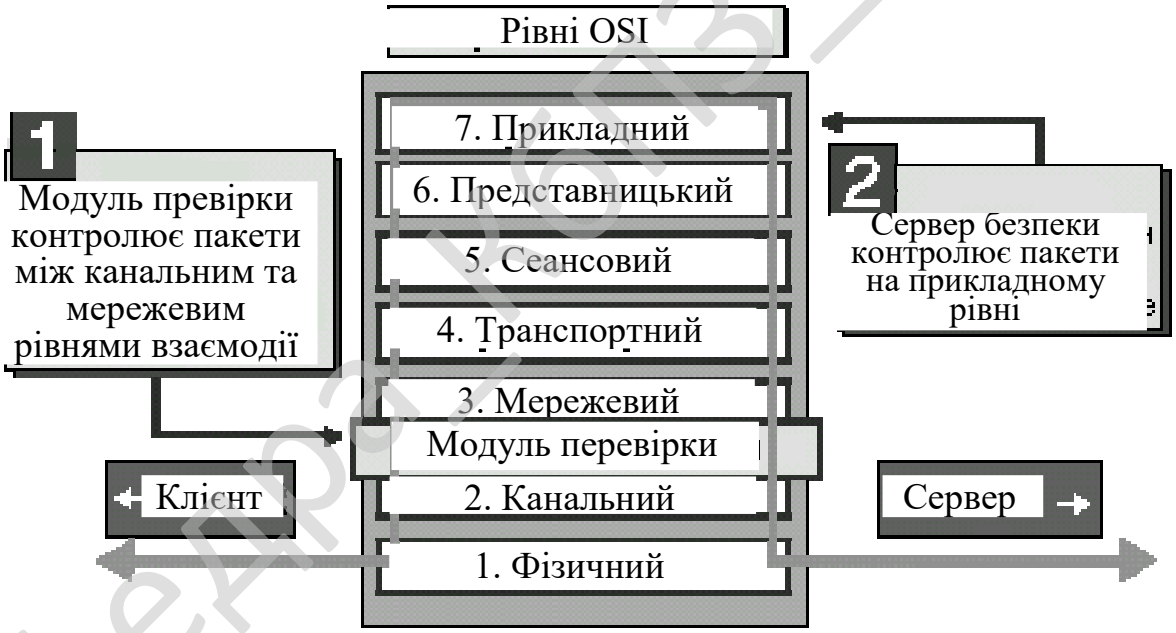

Рисунок 3.2 – З'єднання через сервер безпеки FireWall-1

Існують п'ять серверів безпеки FireWall-1, як показано в таблиці 3.2.

Таблиця 3.2 – Сервери безпеки FireWall-1 – властивості

Сервер	Автентифікація	Перевірка вмісту
TELNET	Так	Немає
RLOGIN	Так	Немає
FTP	Так	Так
HTTP	Так	Так
SMTP	Немає	Так

Перевірка потоку даних

Можливість перевірки змісту розширює набір можливостей по перевірці даних до протоколів самого верхнього рівня. Ця можливість дозволяє максимально гнучко управляти доступом до мережевих ресурсів.

FireWall-1 забезпечує перевірку змісту для HTTP, SMTP і FTP з використанням серверів безпеки FireWall-1. Для кожного з'єднання, установленого через сервер безпеки FireWall-1, адміністратор може управляти доступом відповідно до різних параметрів протоколу даного сервісу: URL, іменами файлів, командами FTP PUT/GET, типами запитів і так далі.

Найбільш важливе застосування перевірки змісту – антивірусна перевірка переданих файлів. Антивірусна підтримка повністю інтегрована з FireWall-1.

Перевірка змісту реалізується через тип обьекта FireWall-1 Resource. Визначення об'єкту Resource задає ряд складових, які можуть бути доступні через певний протокол. Ви можете задавати FireWall-1 Resource на основі протоколів HTTP, FTP і SMTP.

Наприклад, ви можете задати URI ресурс, чиїми атрибутами буде список URL і схем HTTP і FTP. Ресурс може бути використаний у базі правил точно в такий же спосіб, як і будь-який інший тип сервісу, і для нього також можуть бути визначені стандартні процедури запису події в журнал і оповіщення адміністратора системи для забезпечення можливості спостереження за використанням даного ресурсу.

HTTP – Ресурси URI можуть визначати схеми (HTTP, FTP, GOPHER і т.д.), методи (GET,PUT, і т.д.), машини (наприклад, "*.com"), шляхи й запити. Також може бути заданий файл, що містить список IP адрес і серверів.

SMTP – Протокол SMTP, розроблений для забезпечення найбільш зручного спілкування між людьми через Internet, і розширений для підтримки не тільки e-mail, але й передачі файлів, надає вибір системному адміністраторові, що хоче одночасно підтримуючи прозорість з'єднань, не дозволяти порушникам проникнути усередину мережі.

FireWall-1 пропонує сервер SMTP, що забезпечує максимально детальний контроль над з'єднаннями SMTP. Визначаючи SMTP ресурси, адміністратор безпеки має можливості:

– сховати у вихідній пошті адреса From під стандартною загальною адресою, закривши тим самим внутрішню архітектуру мережі й реальні імена користувачів;

– перенаправляти пошту, послану на дану адресу To (наприклад для root) на іншу поштову адресу;

– знищувати всю пошту від заданої адреси;

– обрізати всі прикріплені до листів файли;

– забирати поля Received з вихідної пошти для закриття внутрішньої структури мережі;

– знищувати всі листи розміру більше заданого.

FTP – Сервер безпеки FTP забезпечує автентифікацію й перевірку вмісту, ґрунтуючись на командах FTP (PUT/GET), обмеженнях на імена файлів і антивірусній перевірці файлів.

Антивірус

Антивірусна перевірка є складовою частиною такої властивості FireWall-1, як перевірка вмісту потоків даних і значно знижує уразливість захищених машин. Перевірка всіх переданих файлів проводиться з використанням убудованого антивірусного модуля. Конфігурація цього механізму (які файли перевіряти, що

робити із зараженими файлами) повністю інтегрована в політику безпеки (базу правил). Всі інструменти керування й перевірки FireWall-1 доступні для журналювання й оповіщення про випадки знаходження заражених файлів.

Дослідження недоліків файрволів

Нижче перерахуємо існуючі недоліки файерволів та запропонуємо можливості їх усунення.

Відсутність захисту від авторизованих користувачів

Проблема. Неможливість захисту від користувачів, що знають ідентифікатор і пароль для доступу в сегмент корпоративної мережі, що захищається. Міжмережевий екран може обмежити доступ сторонніх осіб до ресурсів, але він не може заборонити авторизованому користувачеві скопіювати коштовну інформацію або змінити які-небудь параметри фінансових документів, до яких цей користувач має доступ.

Рішення. Для усунення цього недоліку потрібні нові підходи й технології. Наприклад, використання систем виявлення атак (intrusion detection systems). RealSecure, виявляють і блокують несанкціоновану діяльність у мережі незалежно від того, хто її реалізує – авторизований користувач (у т.ч. і адміністратор) або зловмисник.

Відсутність захисту нових мережевих сервісів

Проблема. Неможливість захисту нових мережевих сервісів. Як правило, ММЕ розмежовують доступ по широко розповсюджених протоколах, таким як HTTP, Telnet, SMTP, FTP і ряд інших. Реалізується це за допомогою за допомогою механізму "посередників" (proxy), що забезпечують контроль трафіку, переданого по цих протоколах або за допомогою зазначених сервісів. І хоча число таких "посередників" досить велико, вони існують не для всіх нових протоколів і сервісів.

Рішення. Багато виробників міжмережевих екранів намагаються вирішити зазначену проблему, але вдається це далеко не всім. Деякі виробники створюють proxy для нових протоколів і сервісів, але завжди існує часовий інтервал від

декількох днів до декількох місяців між появою протоколу й відповідного йому proxy. Інші розроблювачі міжмережевих екранів пропонують засобу для написання своїх proxy У цьому випадку необхідні висока кваліфікація й час для написання ефективного proxy, що враховує специфіку нового сервісу й протоколу. Аналогічна можливість існує й у міжмережевого екрана CheckPoint Firewall-1, що містить у собі потужна мова INSPECT, що дозволяє описувати різні правила фільтрації трафіку.

Обмеження функціональності мережевих сервісів

Проблема. Деякі корпоративні мережі використовують топологію, що важко "уживається" з міжмережевим екраном, або використовують деякі сервіси (наприклад, NFS) таким чином, що застосування MME вимагає істотної перебудови всієї мережевої інфраструктури. NFS (Network File System) – мережева файлова система (набір протоколів на основі транспортного протоколу UDP, що дозволяє Unix-машинам, PC і ПК Macintosh спільно використовувати файли в локальній мережі). У такій ситуації відносні витрати на придбання й настроювання міжмережевого екрана можуть бути порівнянні зі збитком, пов'язаним з відсутністю MME.

Рішення. Вирішити дану проблему можна тільки шляхом правильного проектування топології мережі на початковому етапі створення корпоративної інформаційної системи.

Потенційна небезпека обходу міжмережевого екрана

Проблема. Міжмережеві екрани не можуть захистити ресурси корпоративної мережі у випадку неконтрольованого використання в ній модемів. Доступ у мережу через модем по протоколах SLIP (міжмережевий протокол для послідовного каналу) або PPP (протокол двоточкового з'єднання) в обхід міжмережевого екрана робить мережу практично незахищеною.

Рішення. Для рішення цієї задачі необхідно строго контролювати всі наявні в корпоративній мережі модеми й програмне забезпечення віддаленого доступу. Для цих цілей можливе застосування як організаційних, так і технічних

мір. Наприклад, використання систем розмежування доступу, у т.ч. і до COM-портів (наприклад, Secret Net) або систем аналізу захищеності (наприклад, Internet Scanner і System Scanner).

Потенційно небезпечні можливості

Проблема. Нові можливості, які з'явилися недавно, і які полегшують життя користувачам Internet, розроблялися практично без обліку вимог безпеки. Наприклад, WWW, Java, ActiveX і інші сервіси, орієнтовані на роботу з даними. ActiveX – назва групи технологій, розроблених Microsoft, для програмування компонентних об'єктних додатків на основі моделі COM Вони є потенційно небезпечними, тому що можуть містити в собі ворожі інструкції, що порушують установлену політику безпеки.

Рішення. Захист від таких корисних, але потенційно небезпечних можливостей повинна вирішуватися в кожному конкретному випадку по-своєму. Можна проаналізувати необхідність використання нової можливості й зовсім відмовитися від її; а можна використовувати спеціалізовані захисні засоби, наприклад, систему SurfinShield компанії Finjan або SafeGate компанії Security-7 Software, що забезпечують безпеку мережі від ворожого "мобільного" коду.

Віруси й атаки

Проблема. Практично жоден міжмережевий екран не має убудованих механізмів захисту від вірусів і, у загальному випадку, від атак. Як правило, ця можливість реалізується шляхом приєднання до ММЕ додаткових модулів або програм третіх розроблювачів (наприклад, система антивірусного захисту ViruSafe для ММЕ CyberGuard Firewall або система виявлення атак RealSecure для ММЕ CheckPoint Firewall-1). Використання нестандартних архіваторів або форматів переданих даних, а також шифрування трафіку, зводить весь антивірусний захист "на ні". Як можна захиститися від вірусів або атак, якщо вони проходять через міжмережевий екран у зашифрованому виді й розшифровуються тільки на кінцевих пристроях клієнтів?

Рішення. У такому випадку краще перестрахуватися й заборонити проходження через міжмережевий екран даних у невідомому форматі.

Зниження продуктивності

Проблема. Незважаючи на те, що приєднання до мереж загального користування або вихід з корпоративної мережі здійснюється по низькоскоростних каналах (як правило, за допомогою dialup-доступу на швидкості до 56 Кбит або використання виділених ліній до 256 Кбит), зустрічаються варіанти підключення по каналах із пропускною здатністю в кілька сотень мегабит і вище (ATM, T1, E3 і т.п.). ATM (asynchronous transfer mode) – асинхронний режим передачі (стандартизована ITU технологія комутації пакетів фіксованої довжини; є асинхронною в тому розумінні що пакети від окремих користувачів передаються аперіодично; забезпечує ефективну передачу різних типів даних (голос, відео, multimedia, трафік ЛОМ) на значні відстані). У таких випадках міжмережеві екрани є самим вузьким місцем мережі, знижуючи її пропускну здатність. У деяких випадках доводиться аналізувати не тільки заголовок (як це роблять пакетні фільтри), але й зміст кожного пакета ("proxy"), а це істотно знижує продуктивність міжмережевого екрана. Для мереж з напруженим трафіком використання міжмережевих екранів стає недоцільним.

Рішення. У таких випадках на перше місце треба ставити виявлення атак і реагування на них, а блокувати трафік необхідно тільки у випадку виникнення безпосередньої погрози.

Відсутність контролю своєї конфігурації

Проблема. Навіть якщо всі описані вище проблеми вирішені, залишається небезпека, що міжмережевий екран неправильно зконфігурований.

Рішення. У цьому випадку допоможуть засобу аналізу захищеності. Засоби аналізу захищеності можуть тестувати міжмережевий екран як на мережевому рівні (наприклад, схильність атакам типу "відмова в обслуговуванні"), так і на рівні операційної системи (наприклад, права доступу до конфігураційних файлів міжмережевого екрана). Крім того, при скануванні

можлива реалізація атак типу "підбор пароля", що дозволяють виявити "слабкі" паролі або паролі, установлені виробником за замовчуванням. До засобів, що проводять такі перевірки, можна віднести, наприклад, систему Internet Scanner американської компанії Internet Security Systems (ISS).

Ознайомившись із описаними проблемами, багато хто може зробити висновок, що міжмережеві екрани не можуть забезпечити захист корпоративної мережі від несанкціонованого втручання. Це не так. Міжмережеві екрани є необхідним, але явно недостатнім засобом забезпечення інформаційної безпеки. Вони забезпечують лише першу лінію оборони. Не варто купувати міжмережевий екран тільки тому, що він визнаний кращим за результатами незалежних випробувань. При виборі й придбанні міжмережевих екранів необхідно ретельно все продумати й проаналізувати. У деяких випадках досить установити найпростіший пакетний фільтр, вільно розповсюджуваний у мережі Internet або поставляється разом з операційною системою, наприклад squid. В інших випадках міжмережевий екран необхідний, але застосовувати його треба разом з іншими засобами забезпечення

3.2 Розробка структурної схеми

FireWall дозволяє розподілити обчислювальне навантаження на будь-яке число серверів. На рисунку 3.3 зображена структурна схема розробленої системи. На ньому покано конфігурацію, у якій сервіси FTP і HTTP обслуговуються декількома серверами.

Розміщення файрволу як граничного пристрою між внутрішньою мережею (інтранетом) і Інтернетом дозволяє контролювати весь вихідний і вхідний Інтернет-трафік і управляти його проходженням. Між внутрішньою й зовнішньою мережею створюється чіткий захисний рубіж. Однак деяким зовнішнім клієнтам може знадобитися доступ до внутрішніх ресурсів. Для цього можна передбачити демілітаризовану зону (DMZ).

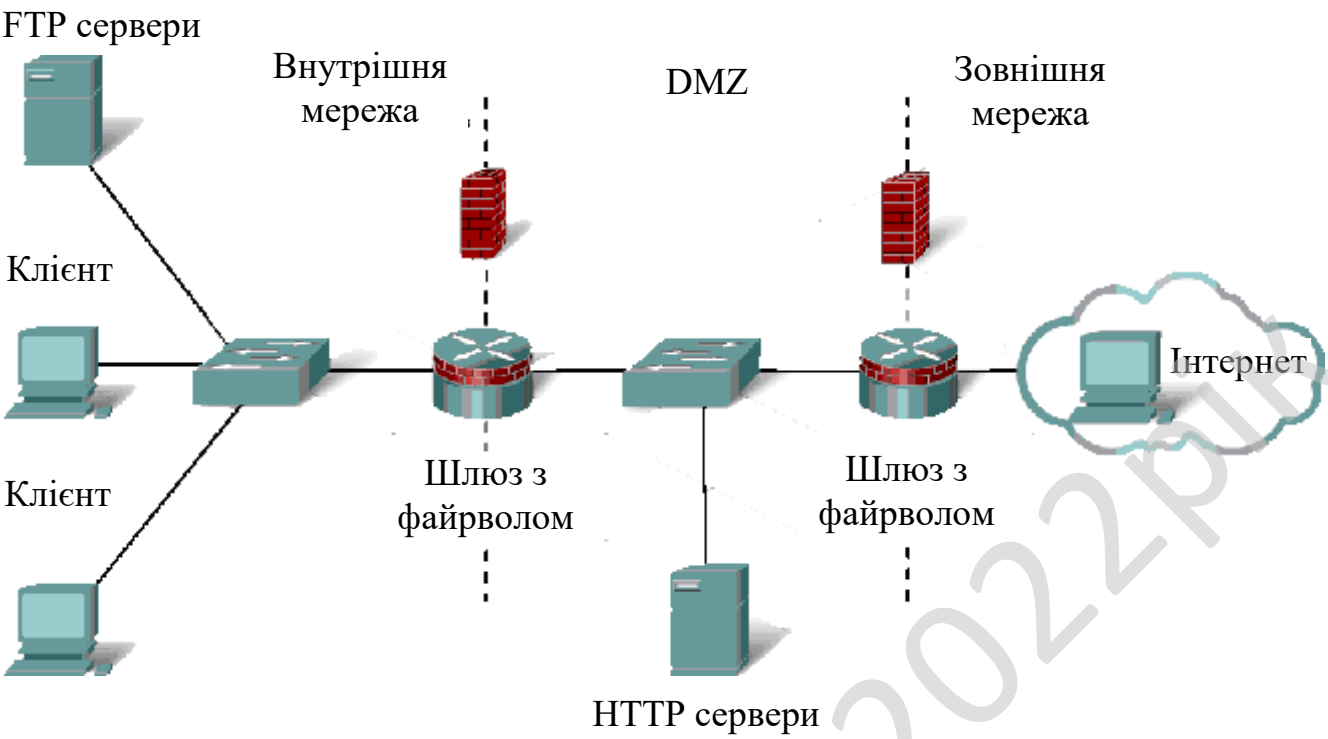

Рисунок 3.3 – Структурна схема системи

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.

Конфігурація з одним файрволом

Один файрвол ділить мережевий простір на три зони: зовнішня мережа, внутрішня мережа й DMZ. Весь трафік надходить на файрвол із зовнішньої мережі. Файрвол повинен контролювати трафік і ухвалювати рішення щодо його пересиланні в DMZ або у внутрішню мережу, або про заборону пересилання.

Конфігурація із двома файрволами

У конфігурації із двома файрволами передбачені внутрішній і зовнішній файрволи, між якими розташовується DMZ. Зовнішній файрвол застосовує менш

строгі обмеження й дозволяє доступ користувачів з Інтернету в DMZ, а також наскрізне проходження трафіка, запитаного внутрішніми користувачами. Внутрішній файрвол застосовує більше строгі обмеження й захищає внутрішню мережу від несанкціонованого доступу.

Для невеликих мереж з низьким трафіком підходить конфігурація на основі одного файрволу, що, однак, є критичною точкою відмови й може виявитися перевантаженим. Конфігурація із двома файрволами доцільна для великих і складних мереж зі значно більшими об'ємами трафіка.

Всі HTTP сервери здатні дати клієнтові однакові сервіси (однак не всі HTTP сервери за захищеним мостом). У такий же спосіб, всі FTP сервери забезпечують клієнтів однаковим сервісом.

Системному адміністраторові важливо, щоб завантаження по обслуговуванню була розподілена між серверами. Клієнт не буде підозрювати про наявність декількох різних серверів. З погляду клієнта, є тільки один HTTP і один FTP сервер. Коли запит на обслуговування досягає FireWall, FireWall визначає який із серверів буде обслуговувати даний запит, ґрунтуючись на алгоритмі балансування завантаження, заданому системним адміністратором.

Алгоритм розподілу навантаження

Доступні наступні алгоритми розподілу:

1. По завантаженню сервера. FireWall запитує сервери щоб визначити, що з них найкраще здатний обслужити нове з'єднання.

2. За часом відповіді на ping (round trip). FireWall використовує ping для визначення часу проходження пакета між FireWall і кожним із серверів і вибирає сервер з найменшим часом відповіді.

3. По колу. FireWall просто призначає наступний сервер у списку.

4. Випадковий. FireWall призначає сервер у випадковому порядку.

5. По доменному ім'ю. FireWall призначає "найближчий" сервер, ґрунтуючись на доменних іменах.

Автентифікація

FireWall-1 забезпечує три види автентифікації:

1. Автентифікація користувача, що дозволяє адміністраторові давати кожному користувачеві свої привілеї доступу. Автентифікація користувача включає сервер безпеки HTTP FireWall, що надає механізм для автентифікації користувачів сервісу HTTP, що як входить так і вихідного.

2. Автентифікація клієнтів дає механізм для автентифікації користувача будь-якого додатка, стандартного або власної розробки.

3. Автентифікація сесій, що дає прозору автентифікацію кожної сесії, що може бути інтегроване з будь-яким додатком.

Таблиця 3.3 – Порівняння типів автентифікації

	Користувач	Клієнт	Сесія
Сервіси	TELNET, FTP, RLOGIN, HTTP	Всі сервіси	Всі сервіси
Автентифікація виконується один раз на …	Сесію	Адреса IP (багато сесій) в окремій непрозорій сесії автентифікації	Сесії
Прозорість (користувач ініціює сесію безпосередньо до сервера)?	Так	Так, тільки після непрозорої сесії автентифікації	Так

Зверніть увагу: автентифікація клієнта й сесії забезпечуються не серверами безпеки, а іншими механізмами, описаними в "Автентифікація клієнта" і "Автентифікація сесії".

FireWall підтримує наступні схеми автентифікації для кожного користувача:

1. S/Key – користувачеві потрібно ввести значення S/Key для даної ітерації.

2. SecurID – користувачеві потрібно ввести номер, показаний на SecurID карті Security Dynamics (безпечна динаміка).

3. По паролю – від користувача вимагають ввести його пароль ОС.

4. Внутрішня – користувач повинен ввести його (або її) внутрішній пароль FireWall-1 на мосту.

5. RADIUS – користувач повинен ввести відповідь на запит сервера RADIUS.

6. AssureNet Pathways – користувач повинен вувести відповідь на запит сервера AssureNet Pathway (забезпечення мережевого шляху).

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема системи.

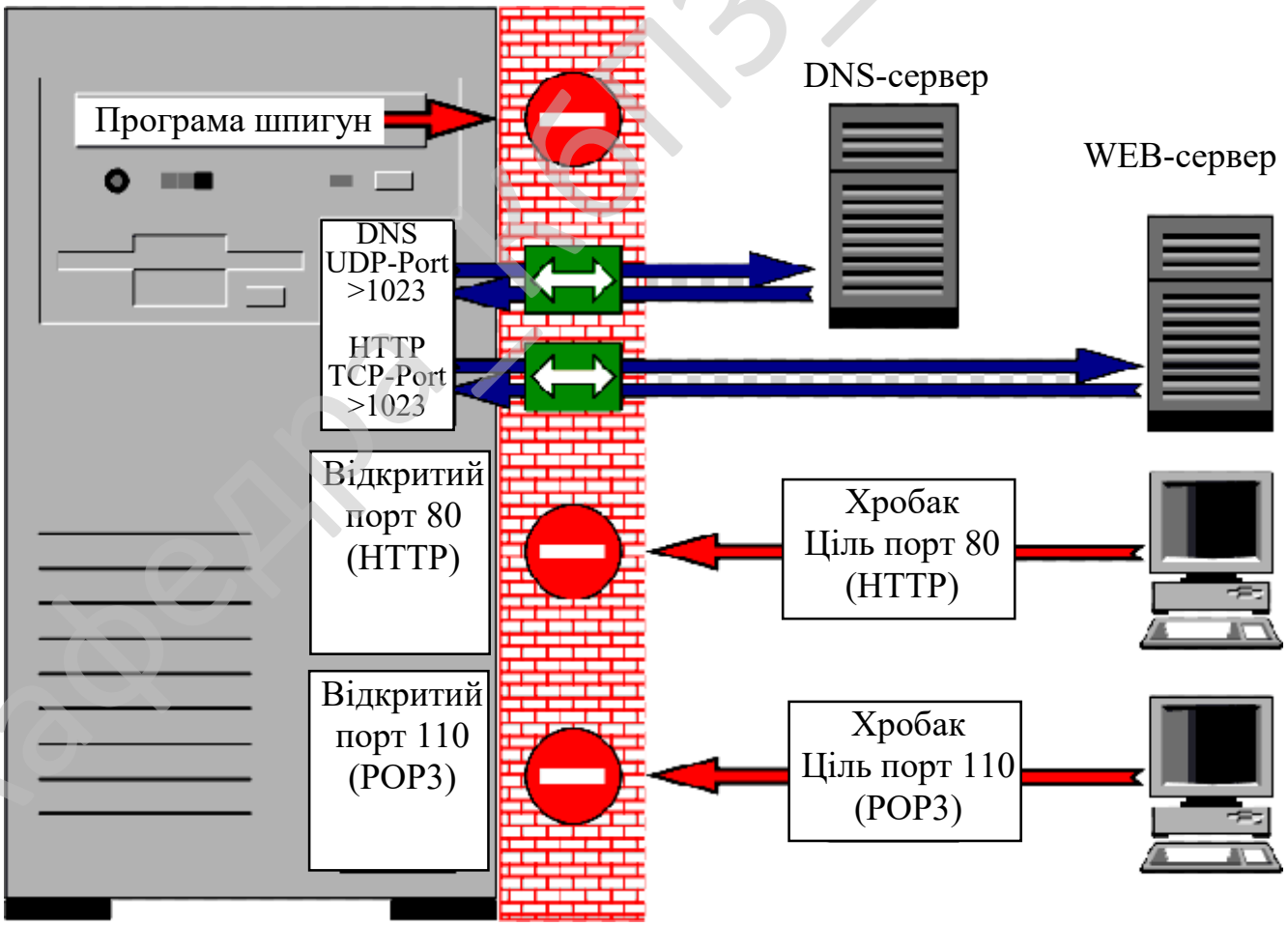

Рисунок 3.4 – Функціональна схема системи

У багатьох пристроях для домашніх мереж, наприклад інтегрованих маршрутизаторах, часто є багатофункціональні програмні файрволи. Такі файрволи звичайно реалізують трансляцію мережевих 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).

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.

3.6 Розробка діаграми процесів

Розглянемо діаграму процесів розробленої системи, що зображена на рисунку 3.5. Після запуску програми, запускається процес встановлення правил

фільтрації. Користувач вводить інформацію про пункт відправлення пакета, інформацію про пункт призначення пакета, вибирає протокол та вказує правило фільтрації. Під правилом фільтрації мається на увазі одна з наступних дій: блокувати, чи пропускати вказаний пакет. Додавати можна довільну кількість правил. Створені списки правил фільтрації можна зберігати та завантажувати в майбутньому.

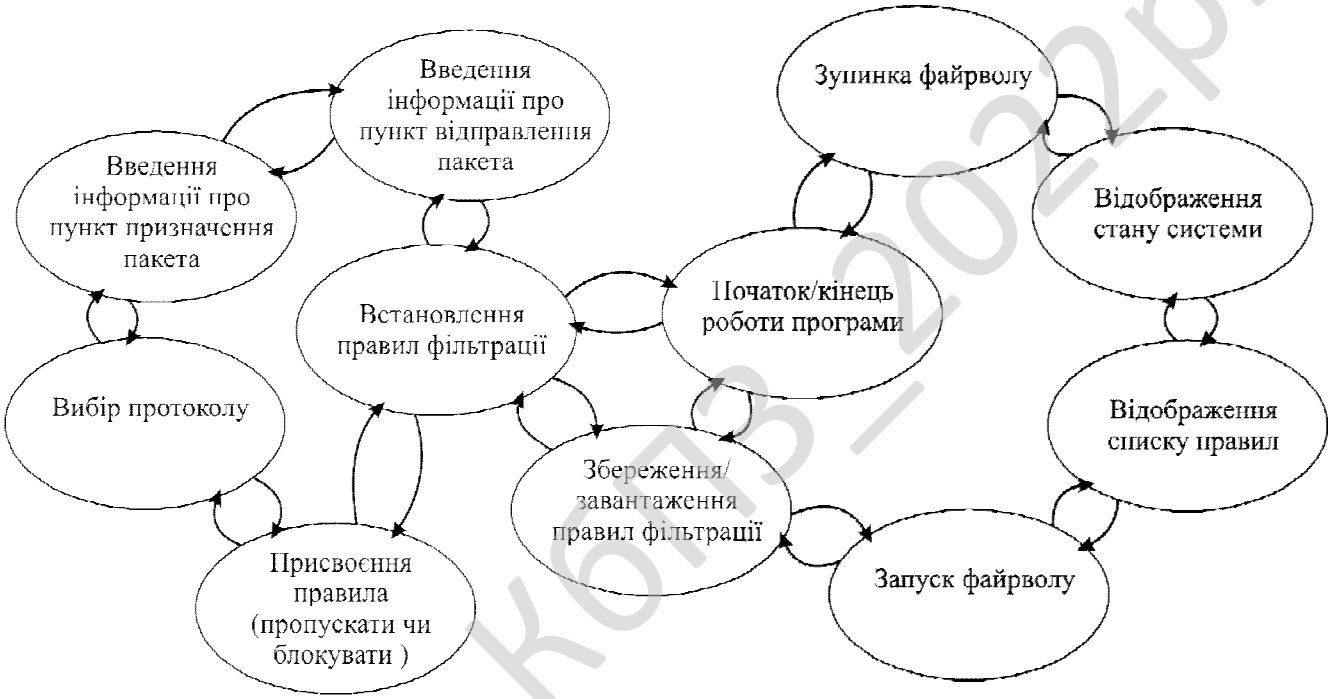

Рисунок 3.5 – Діаграма процесів системи

Після того як список правил фільтрації створений, або завантажений, можна запускати файрвол. Після запуску відбувається відображення списку правил та стану системи. Для тимчасового припинення фільтрації пакетів, файрвол слід зупинити.

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 РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо блок-схему основної програми, зображену на рисунку 4.1. Як видно з рисунку, після завантаження системи відбувається вивід основного вікна програми. Підпрограма виводу головного вікна:

```
//ініціалізація й створення вікна і його компонентів
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  //створення вікна
if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
   return -1;
  //створення ToolBar
if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE |
CBRS_TOP
   | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
   !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{
   TRACE0("Failed to create toolbar\n");
   return -1;      // 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("Firewall");
return 0;
}
```

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

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

```
void CMainFrame::OnSaveRules()
{
CFirewallAppDoc *doc = (CFirewallAppDoc *)GetActiveDocument();
if(doc->nRules == 0)
{
   AfxMessageBox("There isnt Rules to Save.");
   return;
}
CFileDialog dg(FALSE, NULL, NULL, OFN_HIDEREADONLY | OFN_CREATEPROMPT,"Rule
Files(*.rul)|*.rul|all(*.*)|*.*||", NULL);
if(dg.DoModal()==IDCANCEL)
   return;

CString nf=dg.GetPathName();
if(nf.GetLength() == 0)
{
   AfxMessageBox("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();
}
```

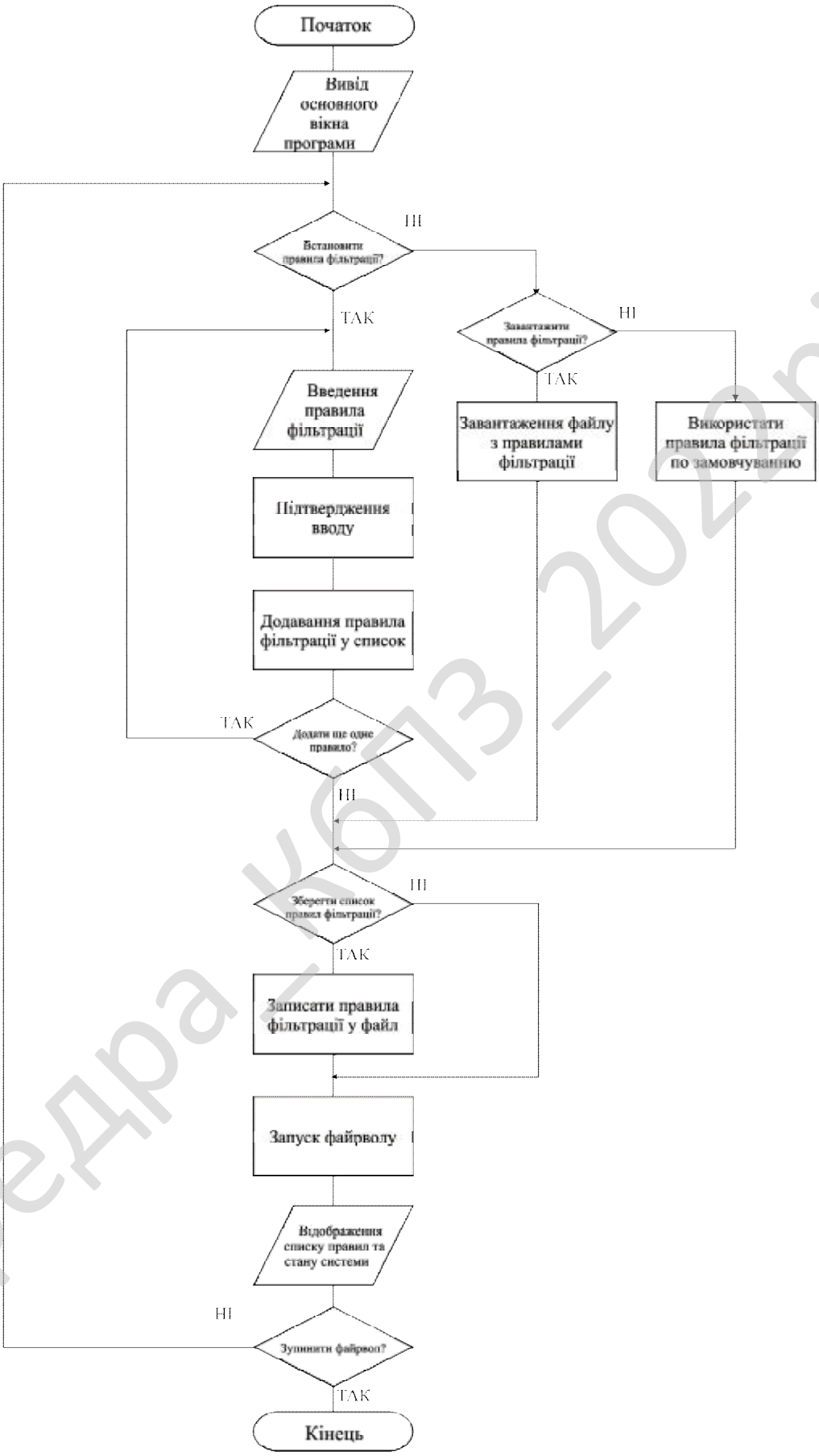

Рисунок 4.1 – Блок-схема основної програми

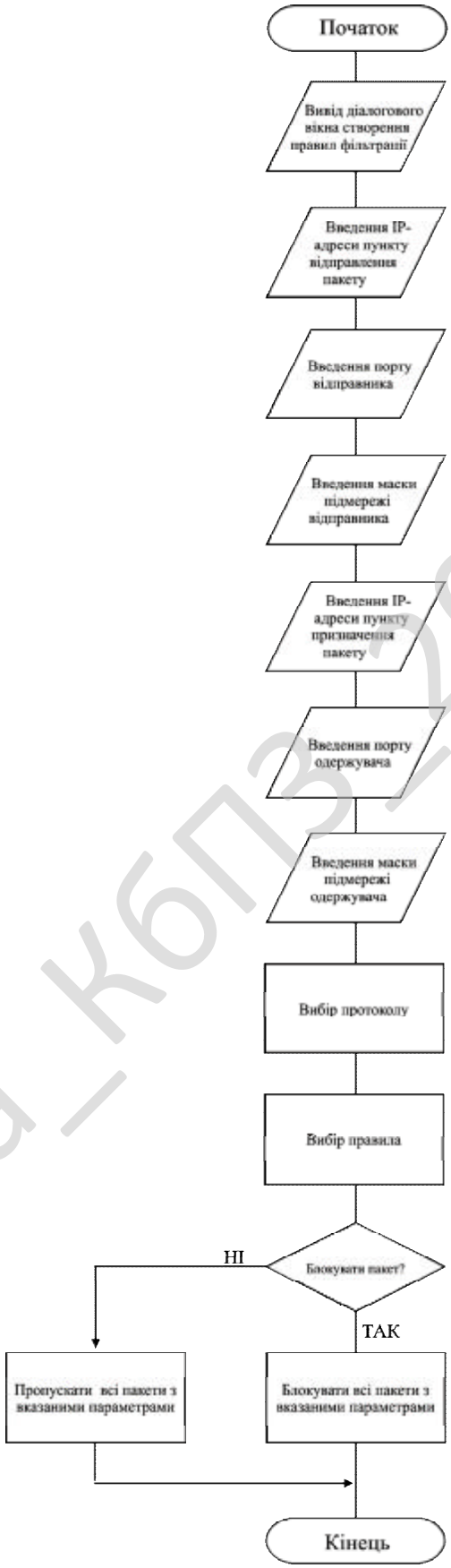

Рисунок 4.2 – Блок-схема підпрограми створення правил фільтрації

Підпрограма завантаження списку правил з файлу:

```
void CMainFrame::OnLoadRules()
{
CFile file;
CFileException e;
DWORD nRead;
CFirewallAppDoc *doc = (CFirewallAppDoc *)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);
CFirewallAppView *view = (CFirewallAppView *)GetActiveView();
view->UpdateList();
}
```

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

```
void CMainFrame::OnButtonstart()
{
CFirewallAppDoc *doc = (CFirewallAppDoc *)GetActiveDocument();
unsigned int i;
DWORD result;
PIP_ADAPTER_INFO pAdapterInfo = NULL, aux;
IP_ADDR_STRING *localIp;
unsigned long len = 0;
  //Пошук адаптера мережевої 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 файрвол. Підпрограма зупинки файрволу виглядає наступним чином:

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

На рисунку 4.2 показана блок-схема алгоритму роботи підпрограми створення правил фільтрації.

Введення правила фільтрації вручну відбувається наступним сином: спочатку користувач вводить інформацію про пункти відправлення та призначення пакету, в яку входять IP-адреси, порти та маска під мережі, потім вибирає протокол передачі даних та правило для нього. Під правилом розуміється блокування чи дозвіл на передачу даного пакету.

Підпрограма додавання правил виглядає наступним чином:

```
int CFirewallAppDoc::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;
}
```

При бажанні користувач може видалити правило зі списку. Підпрограма видалення правила фільтрації:

```
void CFirewallAppDoc::DeleteRule(unsigned int position)
{
// видалення з діапазону
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;
```

```
}
```

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;
}
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм REDOC III, який оперує з 80-бітовим блоком. Довжина ключа може змінюватися й досягати 2560 байт (204800 біт). Алгоритм складається тільки з операцій XOR над байтами ключа й відкритого тексту, перестановки й підстановки не використовуються.

1. Створюють таблицю ключів з 256 10-байтових ключів, використовуючи секретний ключ.

2. Створюють два 10-байтових блоки масок M1 і M2. M1 являє собою результат операції XOR перших 128 10-байтових ключів, а M2 – результат операції XOR других 128 10-байтовых ключів.

3. Для шифрування 10-байтового блоку:

a. Виконують операцію XOR з першим байтом блоку даних і першим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконують операцію XOR з кожним, крім першого, байтом блоку даних і відповідним байтом обраного ключа.

b. Виконують операцію XOR із другим байтом блоку даних і другим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконаєте операцію

XOR з кожним, крім другого, байтом блоку даних і відповідним байтом обраного ключа.

c. Продовжують ці дії з усім блоком даних (з 3-10 байтами), поки не буде використаний кожний байт для вибору ключа з таблиці після виконання операції XOR з ним і відповідним значенням M1. Потім виконують операцію XOR з кожним, крім використаного для вибору ключа, байтом, і ключем.

d. Повторюють етапи a-c для M2.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Основні реалізовані функції:

– завдання основної дії роботи FireWall (пропускати/блокувати мережеві пакети);

– завдання правил (фільтрів) пакетів відмінних від основної дії;

– збереження/завантаження правил на диск;

Set Default Action – Вікно завдання основної дії FireWall:

– Forward Packets – пропускати всі пакети.

– Drop Pakets – блокувати всі пакети.

Add Rule – Вікно додавання правила (фільтра) у роботу FireWall:

Source – Інформація про пункт відправлення пакета:

– IP Address– IP адреса відправника

(0.0.0.0 – для будь-якої адреси).

– Port– порт відправника

(0– для будь-якого порту).

– IP Mask– Маска підмережі відправника

(0.0.0.0 – для будь-якої адреси).

Destination– Інформація про пункт призначення пакета:

– IP Address– IP адреса одержувача

(0.0.0.0 – для будь-якої адреси).

– Port– порт одержувача.

(0 – для будь-якого порту)

– IP Mask– Маска підмережі одержувача.

(0.0.0.0 – для будь-якої адреси).

– Protocol– тип мережевого протоколу, на який призначене правило.

– Action– дія правила (зворотне основній дії).

Remove Rule– Видалення обраного правила (фільтра) з роботи FireWall.

Start Filtering– запуск FireWall на роботу.

Stop Filtering– зупинка FireWall.

Load Rules– завантаження правил (фільтрів) з файлу.

Save Rules– збереження правил (фільтрів) у файл.

Help– довідка по програмі FireWall.

About FireWall– про програму FireWall.

На рисунках 5.1 – 5.9 показані елементи інтерфейсу розроблювальної системи.

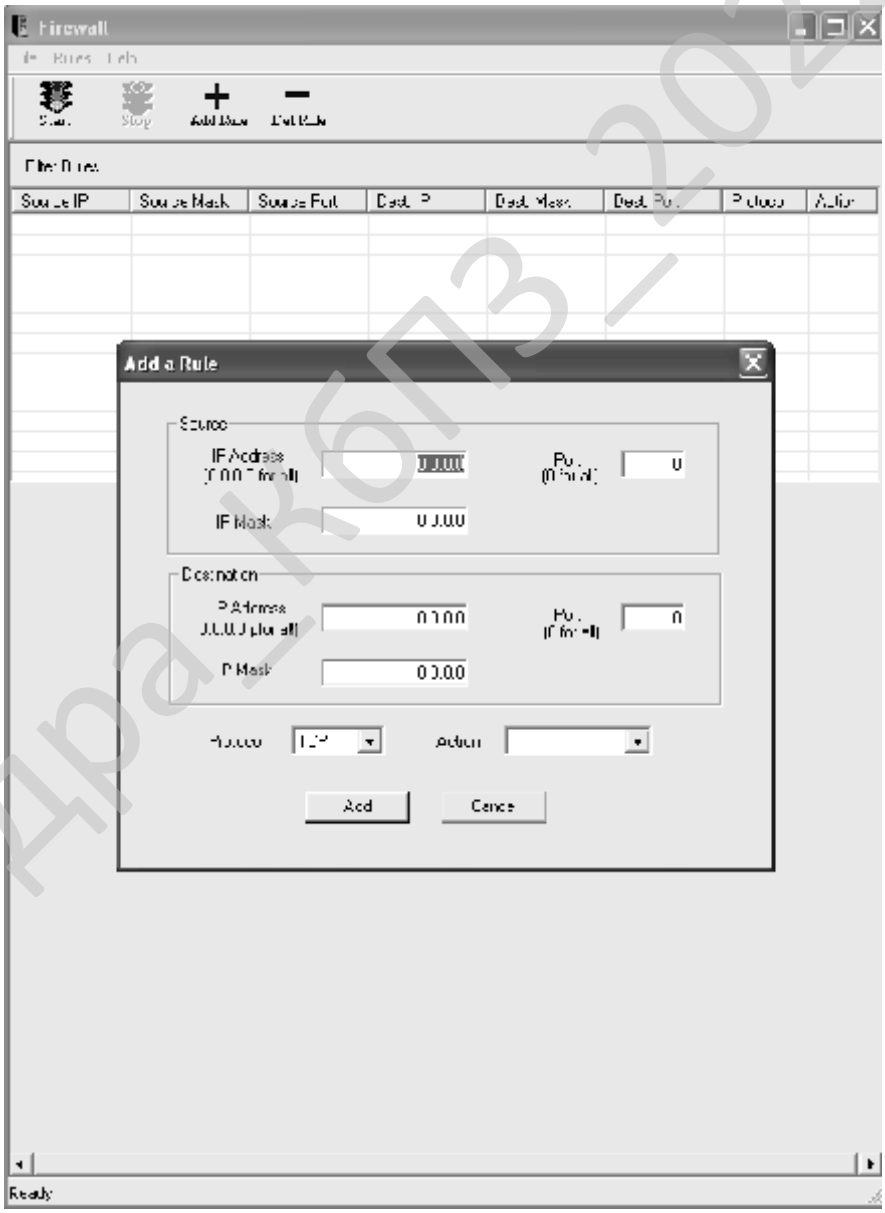

Рисунок 5.1 – Вікно додавання правила (фільтра) у роботу FireWall

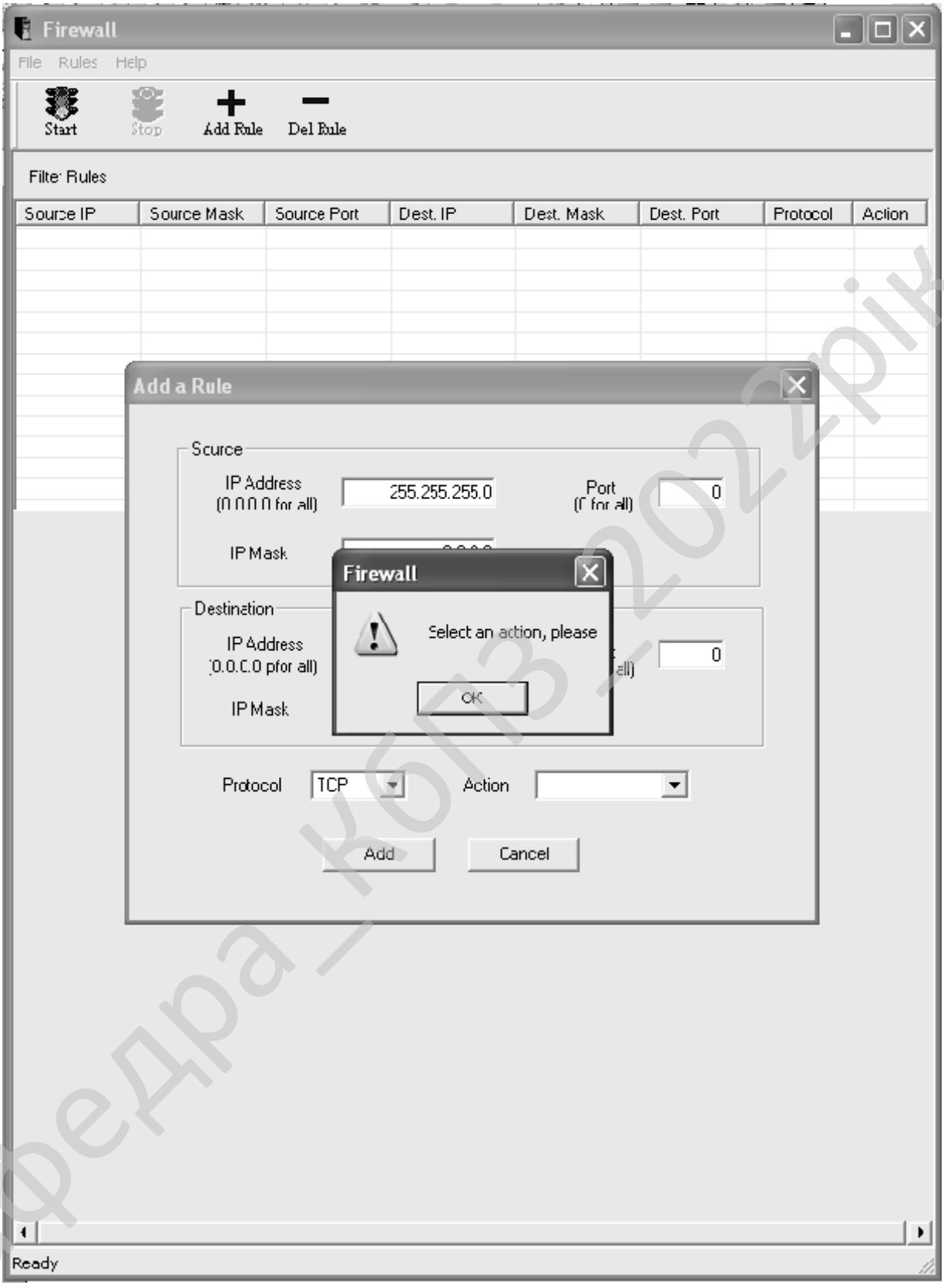

Рисунок 5.2 – Результат неправильного виконання

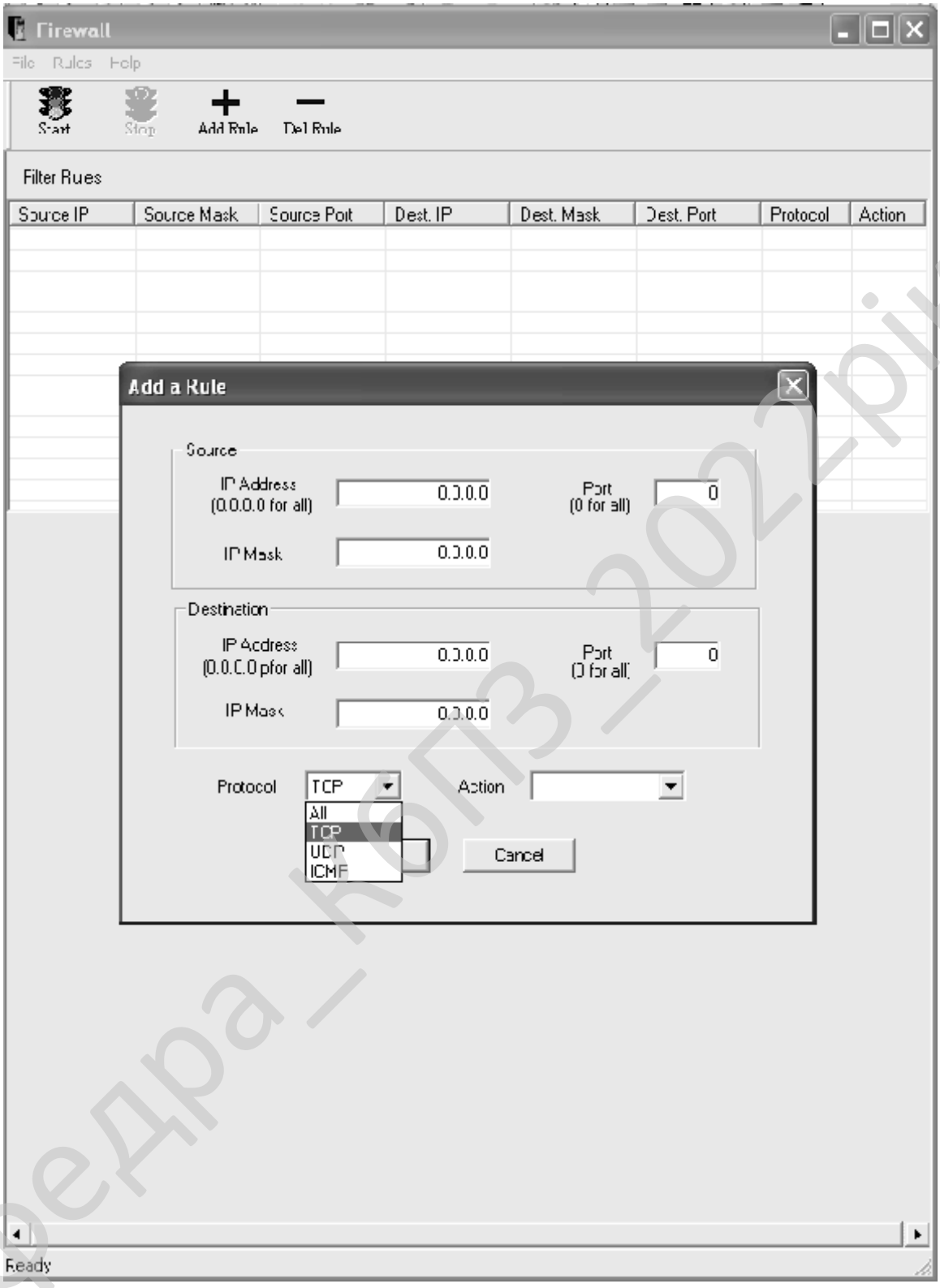

Рисунок 5.3 – Вибір протоколу

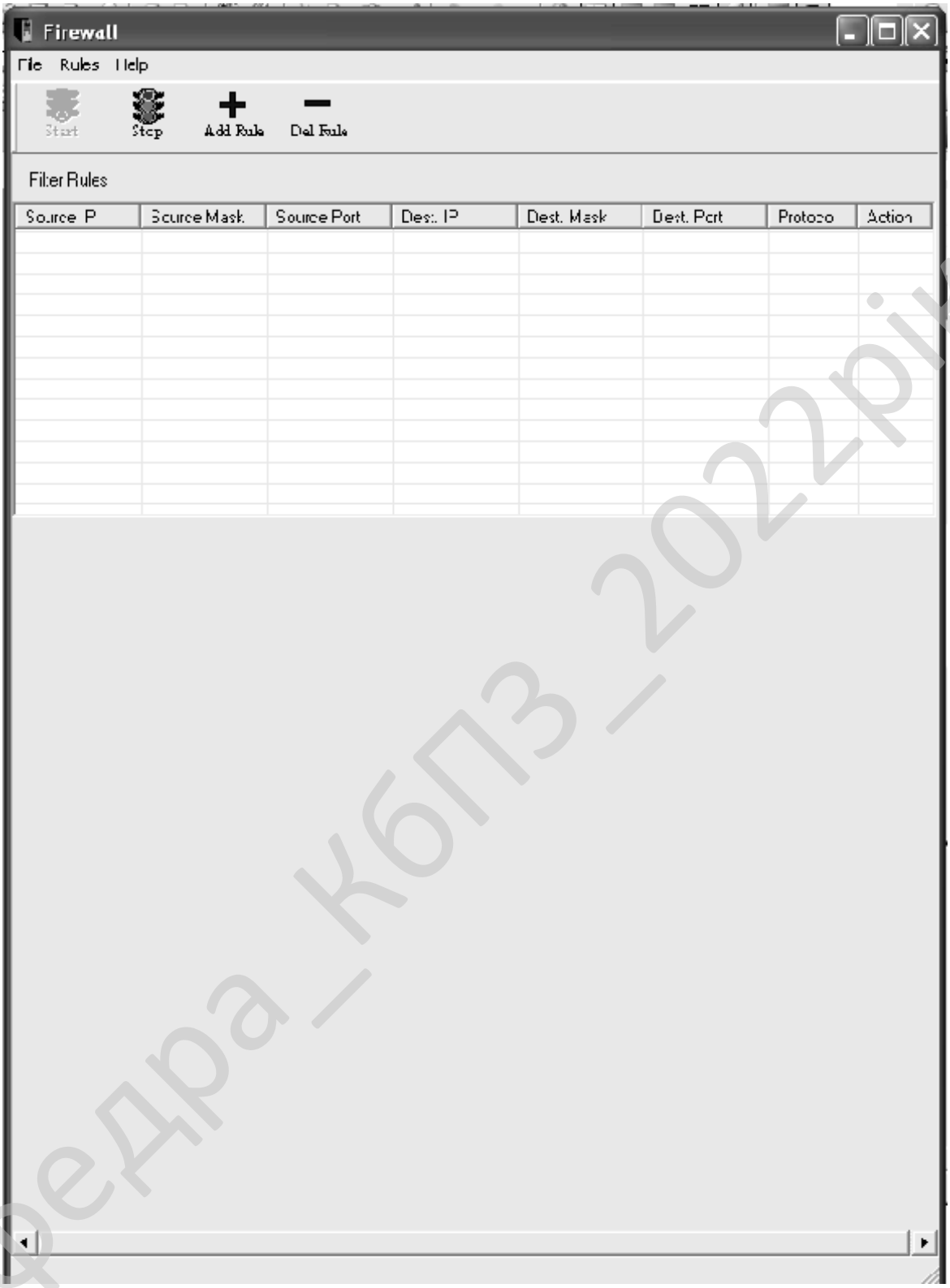

Рисунок 5.4 – Загальний вигляд програми

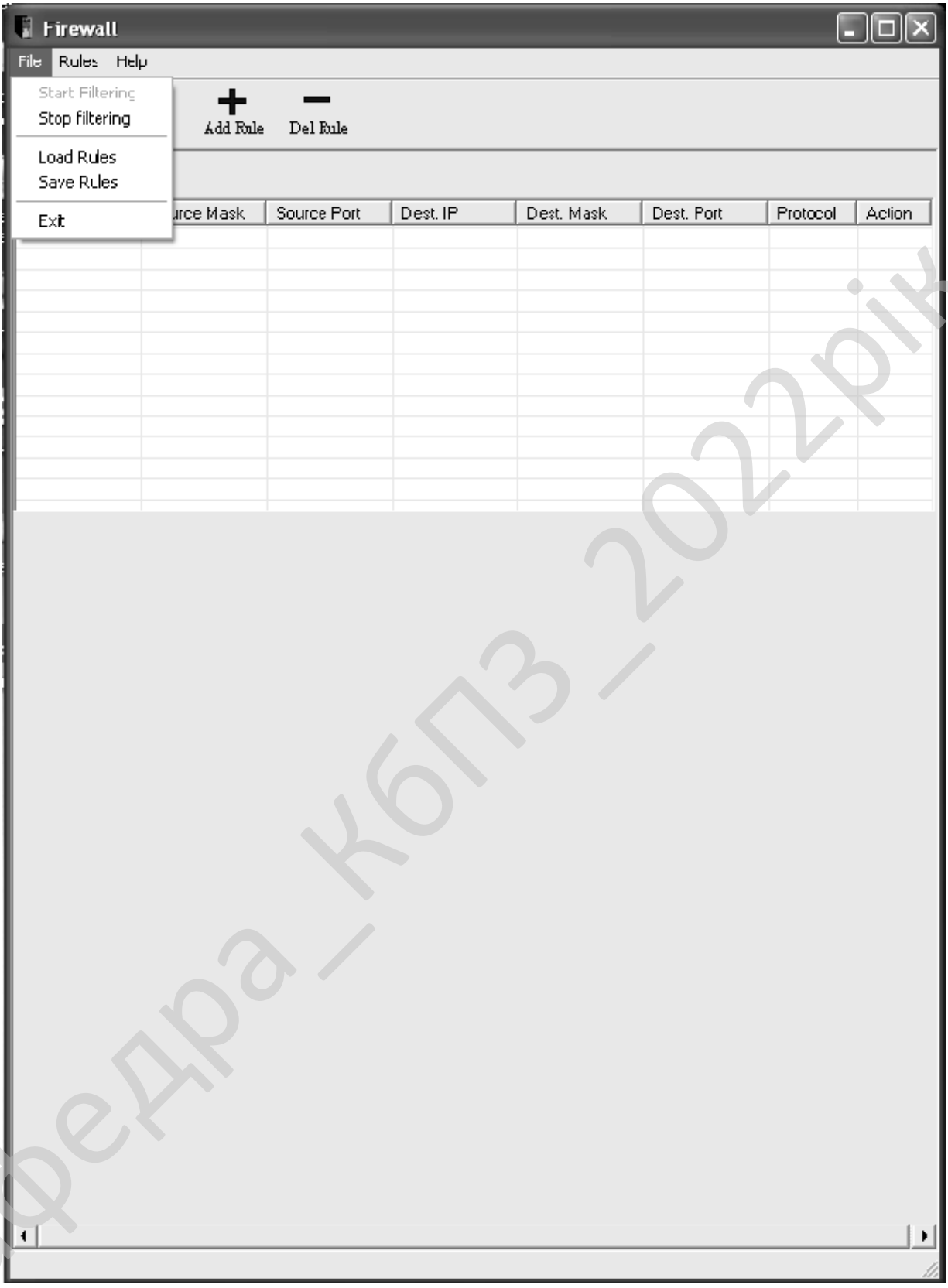

Рисунок 5.5 – Меню файл

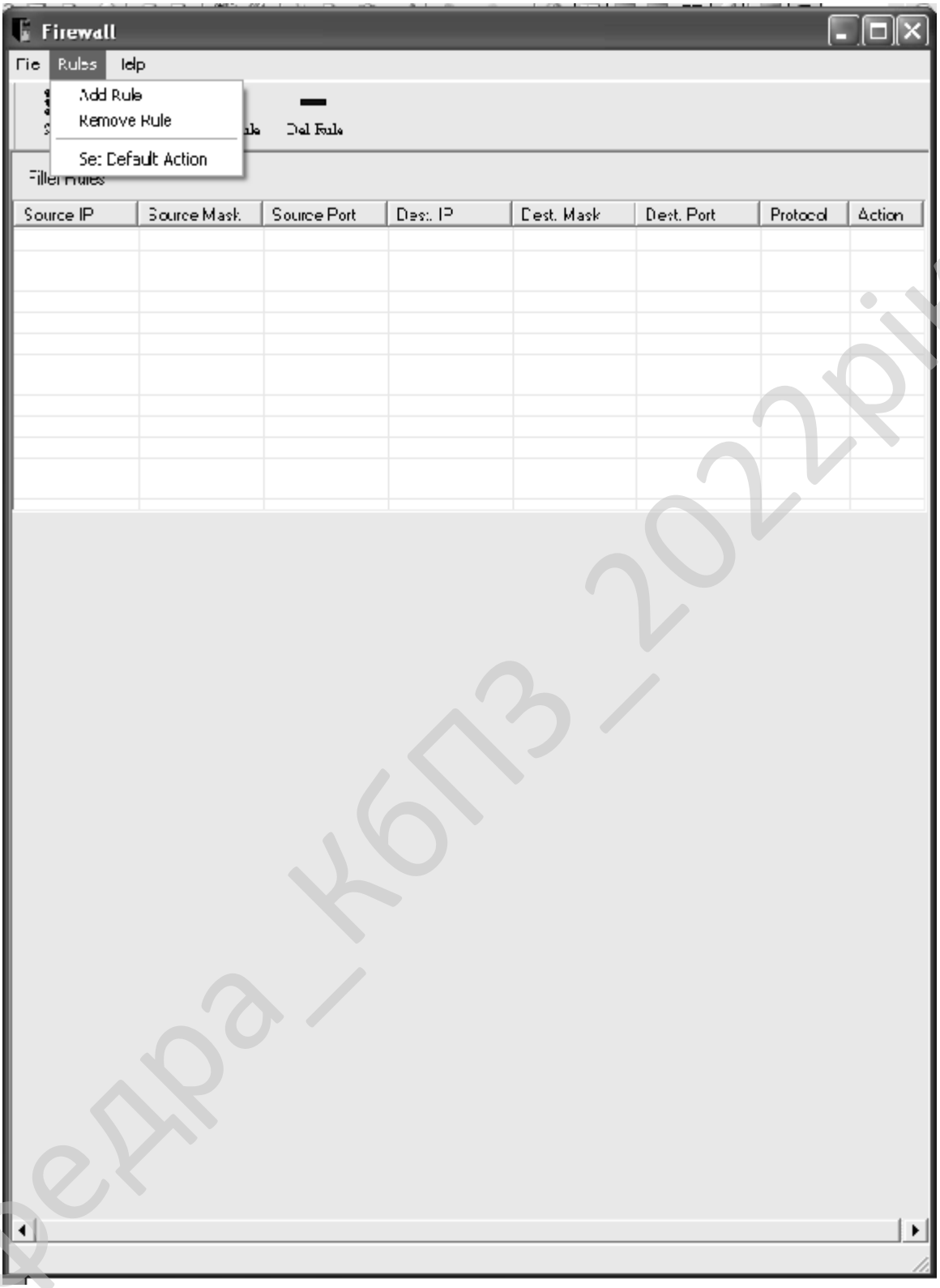

Рисунок 5.6 – Завдання правил (фільтрів) пакетів

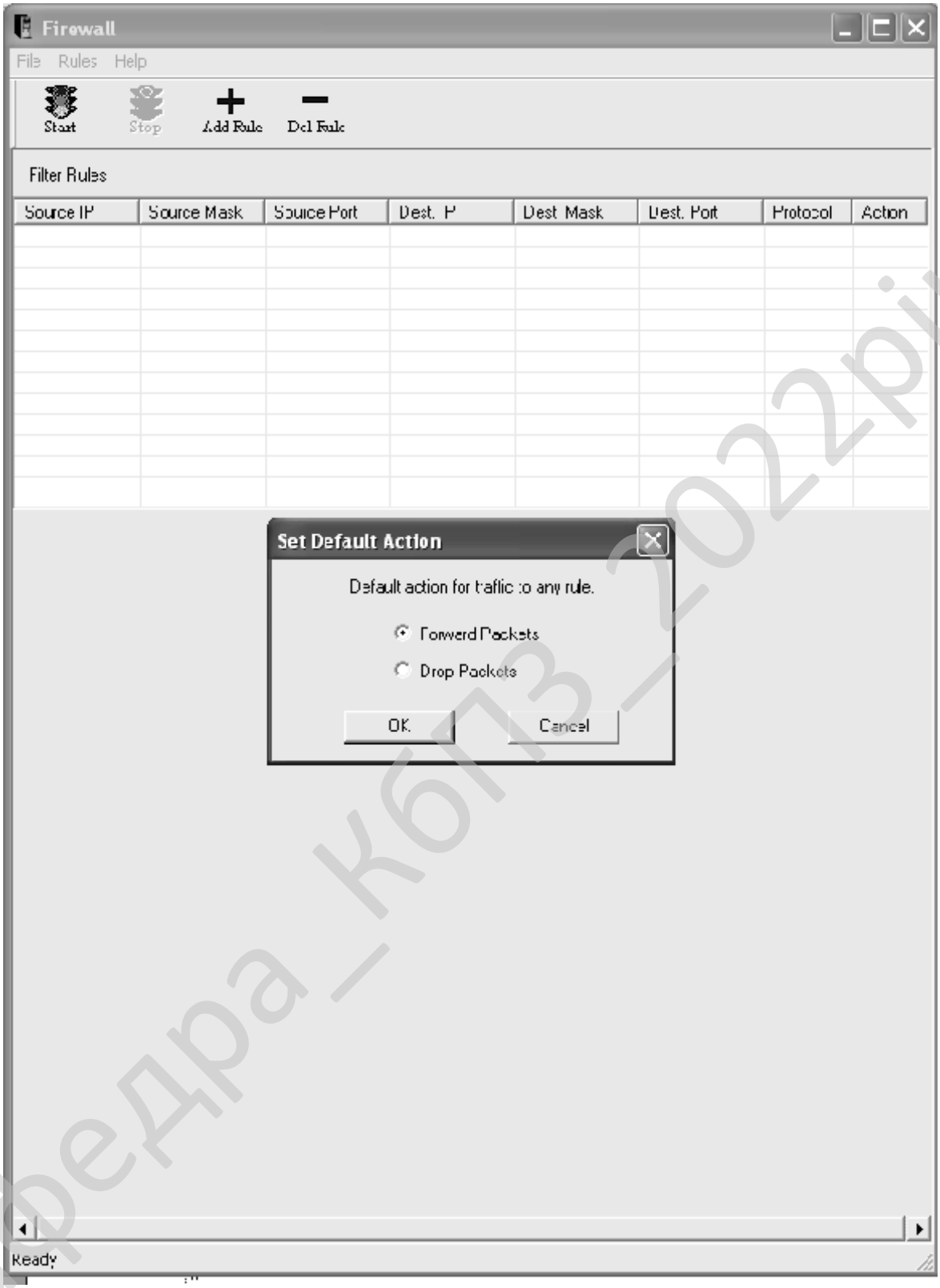

Рисунок 5.7 – Вікно завдання основної дії FireWall

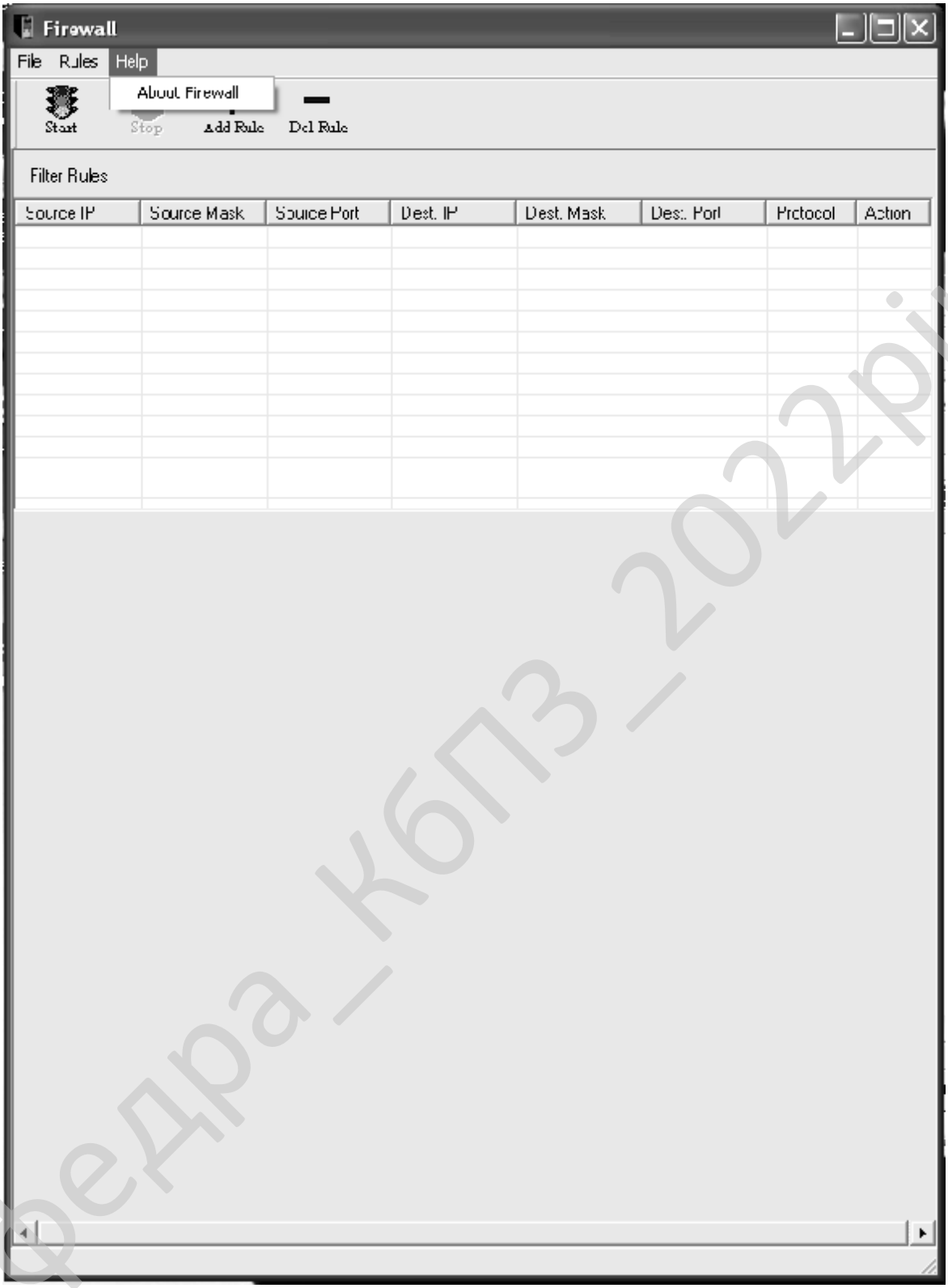

Рисунок 5.8 – Вибір меню про програму

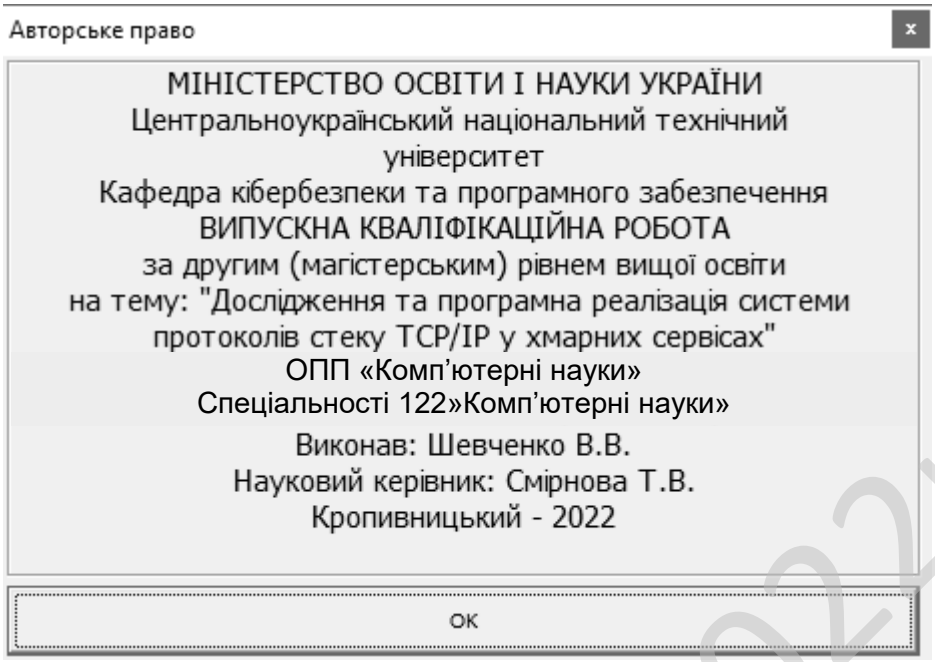

Рисунок 5.9 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи протоколів стеку TCP/IP у хмарних сервісах.

Метою розробки є дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

Об'єктом дослідження є процес протоколів стеку TCP/IP у хмарних сервісах.

Предметом дослідження є методи протоколів стеку TCP/IP у хмарних сервісах.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод протоколів стеку TCP/IP у хмарних сервісах.

– Розроблено вітчизняний продукт протоколів стеку TCP/IP у хмарних сервісах, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	100
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	100000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 60 = 123 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	123	Ф 7.1-7.4
Впровадження	13	Д13
Всього	164	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{164 \cdot 1}{60 - 5} = 3 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	39,49

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1{,}2}, \tag{7.6}$$

$$Ф_{др}^{с} = \frac{39{,}49 \cdot 3}{1{,}2} = 99 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \tag{7.7}$$

$$Ч_{ел} = 99/(60 \cdot 8) = 0{,}2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,25	9500	7125
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3	8000	72000
Інженер - електронщик	0,2	7000	4200
Інженер-системотехнік	0,25	7000	5250
Адміністратор мережі	0,5	7000	10500
Системний програміст	0,25	7000	5250
Дизайнер WEB	0,25	7000	5250
Інженер-верстальник	0,25	7000	5250
Бухгалтер-економіст	0,5	7000	10500
Всього за період розробки	$R_{сп} = 5,7$	-	$Ф_{роб} = 131325$

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{131325}{5,7 \cdot 60} = 384 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 29000 = 1858000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 185800 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 20.10.22 – джерело http://computorg.ua/ru/price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i7-870 / 4 (8) ядра по 2.93-3.6GHz 8Mb cache	-
Системна плата	MACHINIST H55, MACHINIST H55, Inte H55, 1x PCI Expless x1; 1x PCI Express x16 4x SATA II (300MB/s), 6x USB; 1x HDMI; 1 VGA; 1x PS/2 keyboard; 1x PS/2 mouse; 1 RJ-45; 1x audio in\out	-
Відеокарта	PCIeX: ATI HD SAPPHIR 1Gb/128bit/DDR4/TV/DualDVI	-
Жорсткий диск	SSD: 480 Gb	-
Оперативна пам’ять	4 GB DDR3 -2 модуля	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-450PNR 12cm), black, (front bezel – black+light silve body material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 3.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1858000	-	-
2. Передавальні пристрої	185800	-	-
Всього по групі	2043800	5	102190
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
4. Нематеріальні активи	100000	10	10000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	37857,75
Разом	$К_р$ = 2523008		$А_р$ = 249636,25

Примітка: вартість автомобіля Renault Scenic 2004 взята за даними електронного ресурсу, джерело https://auto.ria.com/uk/auto_renault_scenic_33598032.html, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 384 \cdot 164/100 = 629 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 629 \cdot 10 \cdot 0{,}01 = 63 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(629+63) = 256 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 629 \cdot 15 \cdot 0{,}01 = 94 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 1,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=206 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 206 \cdot 1{,}5 = 309 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 50):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 24 грн./шт., DVD-R box – 39 грн./шт.

$$З_{М2} = 49 \cdot 24 + 1 \cdot 39 = 1215 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (309 + 1215 + 1702)/100 = 32 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 629 \cdot 15 \cdot 0{,}01 = 94 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 100 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 249636 \cdot 3/(100 \cdot 12) = 624 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	629
2. Додаткова зарплата виконавців	$З_д$	63
3. Відрахування на соціальні потреби	$С_{оц}$	256
4. Загальногосподарські витрати	$Г_{осп}$	94
5. Витрати на матеріали	$З_М$	32
6. Освоєння нових операційних систем, мов програмування	$О_п$	94
7. Амортизація основних фондів	$А_м$	624
8. Повна собівартість програмного забезпечення	$С_п$	1792
9. Плановий прибуток	$П_р$	896
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2688
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	537,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3225,6

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 629+63+256+94+32+94+624 = 1792 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 1792 = 896 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3226
Всього капітальних витрат	–	3226

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування)	$З_р$	24156	5033
2. Витрати на електроенергію	$З_{ел}$	276	58
3. Витрати на амортизацію	$З_{ам}$	0	807
Всього витрат за рік	$І$	24432	5898

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, $грн/год$.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 50 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 75 \cdot 1{,}1 \cdot 1{,}22 = 24156\ грн,$$

до:

$$З_{р\,нов} = 50 \cdot 75 \cdot 1{,}1 \cdot 1{,}22 = 5033\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

$$З_{ел\,баз} = 0{,}5 \cdot 240 \cdot 2{,}3 = 276 \text{ грн.}$$

$$З_{ел\,нов} = 0{,}5 \cdot 50 \cdot 2{,}3 = 58 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	3226	–	806,5
Всього відрахувань	-	–	3226	–	806,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (2688\text{-}1792) \cdot 100\text{-}(0{,}05 \cdot 2043800 + 0{,}4 \cdot 199177 + 0{,}25 \cdot 37031 + $$
$$+0{,}1 \cdot 100000 + 0{,}2 \cdot 143000) \cdot 3/12 = 32170 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{479208}{(2688\text{-}1792)\cdot 100\cdot 12/3} = 1{,}3 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	100
2. Повна собівартість розробленої програми	Грн.	1792
3. Ціна розробленої програми	Грн.	2688
4. Плановий прибуток від реалізації розробленої програми	Грн.	896
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2523008
7. Загальний прибуток від реалізації програмної продукції	Грн.	89600
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	32170
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,3
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	17728
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,2

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (24432\text{-}5898)\text{-}0{,}25 \cdot 3226 = 17728 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \qquad (7.28)$$

$$T_{сп} = \frac{3226}{24432 - 5898} = 0{,}2 \ року .$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Комп'ютер – невід'ємна складова сучасного життя. За допомогою обчислювальної техніки вирішують складні робочі задачі, ведуться наукові дослідження, створюються архітектурні креслення і твори мистецтва. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп'ютерної техніки. Незважаючи на видиму безпеку та розвитку сучасних технологій, при роботі за комп'ютером є ряд чинників, які можу вплинути на здоров'я людини. Через масовий характер робіт, що виконуються працівниками за допомогою комп'ютера, законодавством України чітко врегульовано норми та вимоги до використання комп'ютерної техніки на підприємстві, безпосередньо й охорона праці на підприємстві при роботі за комп'ютером.

Законом України "Про охорону праці" [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями, затверджені наказом Мінсоцполітики від 14.02.2018р. № 207, зареєстровані в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 [5].

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ.

У розділі даної магістерської роботи висвітлюються основні питання охорони праці працівників, робота яких пов'язана з роботою за комп'ютером, планування робочого приміщення, де працюють користувачі ПК; параметри мікроклімату, освітленість робочих місць та виробничих приміщень; шумові завади.

Правильна організація і раціональне устаткування робочого місця можливість ефективно і з якнайменшими витратами праці виконувати свої функції, плідно спілкуватися співробітниками і підлеглими, підтримувати високу працездатність і робочий настрій.

Велике значення має раціональна конструкція і розташовує елементів робочого місця, що важливе для підтримки оптимальної робочої пози людини-оператора, а також необхідно дотримувати правильний режим праці і відпочинку.

Що стосується питання охорони праці людини необхідно вирішувати на всіх стадіях трудового процесу незалежно від виду професійної діяльності.

Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

Робота працівників пов'язана з роботою за комп'ютером, тому актуальною є розгляд саме умов праці та стану охорони праці працівників які постійно працюють з комп'ютерною технікою.

Завдання даного розділу полягає у тому, щоб розробити якісний програмний продукт необхідно організувати безпеку на робочому місці програміста. Під час проектування безпеки робочому місці з ПК необхідно домагатися високої якості та надійності технічного забезпечення, але й створювати комфортні параметри довкілля для розробників.

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному і психічному напруженні.

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним

напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину 20... 40 $м^3$ на людину Більше 40 $м^3$на людину	Не менше 30 Не менше 20 Може біти використана природна вентиляція

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів.

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [2]

8.3 Розробка заходів з умов поліпшення охорони праці

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців it-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців it-індустрії і стандартів підприємств,

центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів.

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [4].

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.4 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: прокат сталевий круглого перерізу діаметром *Dв=30 мм.*, довжиною *L=1,5 м.,* та горизонтальний електрод – металева полоса з перетином *30·4 мм.* Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,5 м*. Відстань між вертикальними заземлювачами (електродами) *A=1,5 м*. Глибина закладення горизонтального контура заземлення *t=0,7 м*. Опір заземлювача, який нормується: $R_{3H} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+1,5/2=1,45 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho_2 = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де

$\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичної зони у багатошаровому грунті [10];

$\rho = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [10].

Діаметр вертикального електрода (задан):

$$Dв = 30 \text{ мм.} = 0,030 \text{ м.}$$

Відношення *A/L=1,5/1,5=1.*

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [10]:

$$R_O=0,366(\rho/L)[lg(2L/Dв)+(1/2)lg((4T+L)/(4T-L))]=$$

$$=0,366(54,5/1,5)[lg(2\cdot1,5/0,030)+(1/2\ )lg((4\cdot1,45+1,5)/(4\cdot1,45-1,5)]=28\ Ом.$$

Визначаємо коефіцієнт екранування вертикальних електродів $Кев=0,8$ при орієнтовній кількості вертикальних електродів, яке дорівнює 4 [10].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН} = 4\ Ом$:

$$N=R_O /(Кев\ R_{ЗН})= 28 / (0,8\cdot4)=8,77 \approx 9\ шт.$$

Визначаємо довжину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot1,5\cdot9=14,1 \approx 14\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [10]:

$$R_П =0,366(\rho_2\cdot K_П/L_П\ )lg(2(L_П\cdot L_П\ )/(B\cdot t))=$$

$$=0,366(40\cdot5/15,1)\cdot lg((2\cdot15,1^2)/(0,03\cdot0,7))=22,1\ Ом.$$

де

$K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [10]:

$B = 30\ мм. = 0,03\ м.$- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [10]:

$$R =(R_O\cdot R_П)/(\ R_O\cdot\eta_П+\ N\cdot R_П\cdot Кев)=$$

$$=(28\cdot\ 22,1)/(28\cdot\ 0,75+\ 9\cdot\ 22,1\cdot\ 0,7)=3,44\ Ом.$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [10].

Умова $R \le R_{ЗН}$ виконується ($3,44 \le 4$).

Так як R суттєво більше $R_{ЗН}$, зменшимо кількість вертикальних електродів до 8 і виконаємо перерахунок. У результаті остаточно отримали: кількість вертикальних електродів дорівнює 8 при $R =3,86\ Ом.$

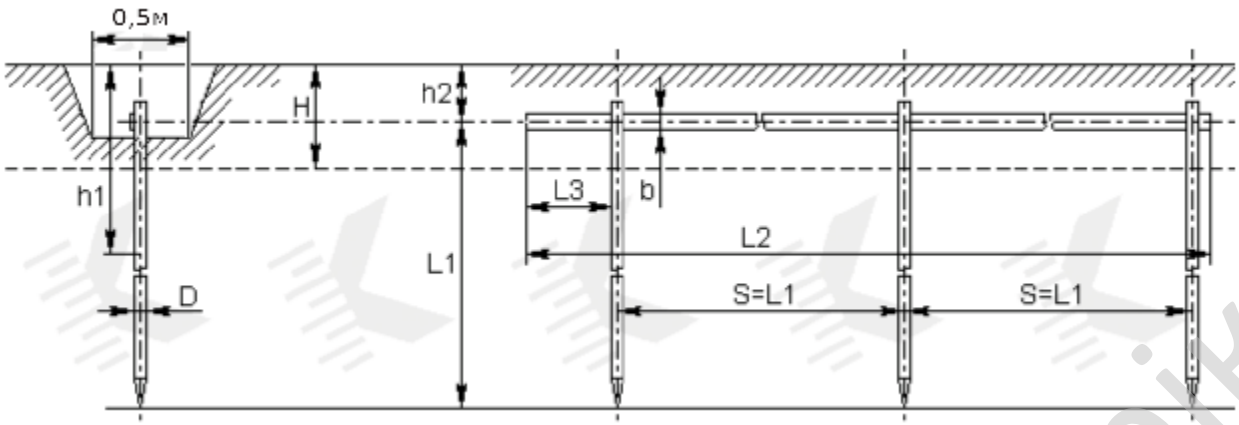

Рисунок 8.1 – Схема штучного заземлення .

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Можна зробити наступний висновок, що шкідливі та небезпечні виробничі фактори існують практично на будь якому робочому місці. Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів о питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи протоколів стеку TCP/IP у хмарних сервісах.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів протоколів стеку TCP/IP у хмарних сервісах.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем протоколів стеку TCP/IP у хмарних сервісах.

– Досліджена система протоколів стеку TCP/IP у хмарних сервісах.

– На основі отриманих результатів досліджень створена програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання протоколів стеку TCP/IP у хмарних сервісах.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм REDOC III.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 17728 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шевченко В.В. Дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

3. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

4. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

5. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

6. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

7. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-

телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

8. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

9. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

10. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

11. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

12. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

13. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка

Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

14. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. – практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

15. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

16. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

17. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

18. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

19. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

20. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам

Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

21. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

22. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

23. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

24. Смирнов С. А. технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

25. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

26. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

27. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

28. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

29. Смирнов С. А. Разработка комплекса gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук.-практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

30. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

31. Смирнов С. А. gert-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського

суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

32. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

33. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

34. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

35. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук.-практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

36. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-

практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

37. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня – 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

38. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

39. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

40. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

41. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая – 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

42. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. – техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

43. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

44. Столлингс В. Современные компьютерные сети / Вильям Столлингс. –СПб.: Питер, 2003. – 778 с.

45. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

46. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

47. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

48. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

49. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

50. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

51. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

52. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

53. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

54. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

55. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

56. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-122.22.0019.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Шевченко В.В.			Дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнова Т.В.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КН-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи протоколів стеку TCP/IP у хмарних сервісах.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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 з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи протоколів стеку TCP/IP у хмарних сервісах;

– 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 архітектури IBM PC, 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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці на робочому місці ІТ-фахівця.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 113 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 21.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнова Т.В.

Дослідження та програмна реалізація системи протоколів стеку TCP/IP у хмарних сервісах

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 42

Літера: РП

Кропивницький – 2022 року

TCP_IP_CloudApp.cpp - головний файл програми

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

#include "MainFrm.h"
#include "TCP_IP_CloudAppDoc.h"
#include "TCP_IP_CloudAppView.h"

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

/////////////////////////////////////////////////////////////////////////////
// CTCP_IP_CloudAppApp
//Мапінг Windows подій, перехоплення пост повідомлень
BEGIN_MESSAGE_MAP(CTCP_IP_CloudAppApp, CWinApp)
	//{{AFX_MSG_MAP(CTCP_IP_CloudAppApp)
	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()

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

/////////////////////////////////////////////////////////////////////////////
CTCP_IP_CloudAppApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CTCP_IP_CloudAppApp initialization

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

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


	SetRegistryKey(_T("TCP_IP_CloudApp"));

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

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

```
            IDR_MAINFRAME,
            RUNTIME_CLASS(CTCP_IP_CloudAppDoc),
            RUNTIME_CLASS(CMainFrame),
            RUNTIME_CLASS(CTCP_IP_CloudAppView));
      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 CTCP_IP_CloudAppApp::OnAppAbout()
{
      CAboutDlg aboutDlg;
      aboutDlg.DoModal();
}

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

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

DefaultActionDlg.cpp - Підключення основних оголошень діалогового вікна

```
//Описувач класу програми
//Підключення основних оголошень діалогового вікна
#include "stdafx.h"
#include "TCP_IP_Cloudapp.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();
}
```

TCP_IP_CloudAppDoc.cpp – формування правил

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

#include "TCP_IP_CloudAppDoc.h"

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

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

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

/////////////////////////////////////////////////////////////////////////////
// CTCP_IP_CloudAppDoc construction/destruction

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

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

	return TRUE;
}

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

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

```
	CDocument::AssertValid();
}

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

/////////////////////////////////////////////////////////////////////////////
// CTCP_IP_CloudAppDoc commands
//обробка подій (пост повідомлень) Windows по додаванню правила
int CTCP_IP_CloudAppDoc::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 CTCP_IP_CloudAppDoc::ResetRules()
{
	nRules = 0;
}
//обробка подій (пост повідомлень) Windows по видаленню правила
void CTCP_IP_CloudAppDoc::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;
        }
```

TCP_IP_CloudAppView.cpp – Формування вікон

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

#include "TCP_IP_CloudAppDoc.h"
#include "TCP_IP_CloudAppView.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

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

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

	//}}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()

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

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

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

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


	return CFormView::PreCreateWindow(cs);
}

//Ініціалізація вікна
```

```
void CTCP_IP_CloudAppView::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);


}

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

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

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

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

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

CTCP_IP_CloudAppDoc* CTCP_IP_CloudAppView::GetDocument() // non-debug version is
inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CTCP_IP_CloudAppDoc)));
	return (CTCP_IP_CloudAppDoc*)m_pDocument;
}
#endif //_DEBUG
```

```
////////////////////////////////////////////////////////////////////////////
// CTCP_IP_CloudAppView message handlers
//опис функції відновлення списку правил в інтерфейсі
void CTCP_IP_CloudAppView::UpdateList()
{
	CTCP_IP_CloudAppDoc *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 CTCP_IP_CloudAppView::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 "TCP_IP_CloudApp.h"

#include "MainFrm.h"
#include "RuleDlg.h"
#include "DefaultActionDlg.h"
#include "TCP_IP_CloudAppDoc.h"
#include "TCP_IP_CloudAppView.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("TCP_IP_Cloud");

      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()
{
	CTCP_IP_CloudAppDoc *doc = (CTCP_IP_CloudAppDoc *)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()
{
	CTCP_IP_CloudAppDoc *doc = (CTCP_IP_CloudAppDoc *)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
				CTCP_IP_CloudAppView *view = (CTCP_IP_CloudAppView
*)GetActiveView();

				view->UpdateList();
			}

		}
	}

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

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


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

		return;
	}

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

	CTCP_IP_CloudAppDoc *doc = (CTCP_IP_CloudAppDoc *)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;

	CTCP_IP_CloudAppDoc *doc = (CTCP_IP_CloudAppDoc *)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);
```

```
	CTCP_IP_CloudAppView *view = (CTCP_IP_CloudAppView *)GetActiveView();

	view->UpdateList();
}

//функція обробки збереження правил у файл
void CMainFrame::OnSaveRules()
{
	CTCP_IP_CloudAppDoc *doc = (CTCP_IP_CloudAppDoc *)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);

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

                    CTCP_IP_CloudAppView *view = (CTCP_IP_CloudAppView
*)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 створенй, 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;
}
```

```
**/
#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 "TCP_IP_CloudApp.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;
}
```