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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Середа О.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

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

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

Середі Олексію Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 47-02 від 17.01.2025 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

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

АНОТАЦІЯ

Середа О.В. Програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

Метою розробки є програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

Результат роботи – програмна реалізація системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

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

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

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

Програму розроблено в середовищі Visual C++.

Ключові слова: комп'ютерна інженерія, ISFW

ABSTRACT

Sereda O.V. Software for the ISFW specialized internal intersegment firewall system. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the ISFW specialized internal intersegment firewall system.

The purpose of the development is the software for the ISFW specialized internal intersegment firewall system.

The result of the work is the software implementation of the ISFW specialized internal intersegment firewall system.

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 are provided.

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

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

Keywords: computer engineering, ISFW

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. Захист інформаційної мережі від несанкціонованого доступу став для компаній життєво важливою необхідністю. Тим часом традиційних стратегій безпеки, орієнтованих на організацію оборони по всьому периметру, уже недостатньо. Щоб забезпечити діючий захист від сучасних складних погроз, необхідно впровадити спеціалізовані внутрішні міжсегментні мережні екрани.

У міру того, як підприємства та організації масштабуються та ростуть, їх мережева інфраструктура також може ставати все більшою та складнішою. Використання плоскої мережевої структури (усі пристрої, підключені до одного сервера) полегшує кіберзлочинцям вільне та безперешкодне переміщення в системі в разі успішної кібератаки . Застосування передових методів сегментації мережі може обмежити масштаб атаки, запобігти поширенню зловмисного програмного забезпечення та порушити бічні переміщення у вашій IT-екосистемі.

Однак стратегія сегментації мережі може бути дорогою та важкою для впровадження, якщо не вжито належних заходів і немає чіткого зв’язку з командою керування. У цій статті обговорюються найкращі практики кібербезпеки для переходу на повністю сегментовану мережу, щоб ваша організація могла покращити рівень безпеки .

Необхідно впроваджувати внутрішні міжсегментні мережні екрани (Internal Segmentation Firewall, ISFW, у зв’язку з тим, що традиційних стратегій безпеки, орієнтованих на захист периметра, уже недостатньо.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

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

– Огляд існуючих систем спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

– Дослідження системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

– Програмна реалізація системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сегментація мережі – це практика безпеки мережі та стратегія глибокого захисту, яка передбачає поділ основної мережі на кілька менших підмереж для кращого захисту конфіденційних даних і обмеження бокового переміщення до решти мережі. Кожна окрема підмережа або «зона» представляє додатковий рівень безпеки, який має власну точку доступу, облікові дані для входу та захист брандмауером. Хоча плоскі мережі мають швидший зв'язок і менше обмежень, вони набагато менш безпечні, ніж сегментовані мережі.

Існує три основних способи сегментації мережі, хоча більшість політик сегментації зазвичай використовують комбінацію всіх трьох: сегментація VLAN, сегментація брандмауера та сегментація SDN:

1. Сегментація VLAN (віртуальних локальних мереж). Більшість сегментованих мереж використовують VLAN для створення менших груп підмереж або підмереж, які з'єднані практично в одному широкомовному домені. Локальні мережі спільно використовують ту саму фізичну мережу в одному місці, але VLAN дозволяють кільком мережам працювати разом в одній групі. Лише користувачі однієї VLAN можуть спілкуватися один з одним. Кожна підмережа використовує іншу IP-адресу, ніж інші підмережі, з'єднані мережевими пристроями. Щоб користувачі різних VLAN могли спілкуватися, вони повинні направляти свої дані через пристрій рівня 3 (зазвичай маршрутизатор або комутатор).

2. Сегментація брандмауера. Організації можуть налаштувати брандмауери між кожним прикладним рівнем для захисту кожної внутрішньої зони мережі. Щоб перейти від однієї функціональної зони мережі до іншої, усі сторони повинні спочатку пройти через брандмауер. Брандмауери можуть

блокувати певні типи трафіку на основі контролю доступу, встановленого організацією. Однак використання лише політики сегментації брандмауера може мати багато недоліків через складність правил, необхідних для правильного сегментування внутрішніх мереж. Будь-яка неправильна конфігурація може повністю зламати програму, що потенційно може завдати шкоди бізнесу. Його також дорого використовувати як єдиний процес сегментації.

3. Сегментація SDN (Software-Defined Networking). Сегментація SDN – це форма мікросегментації, яка використовує таке програмне забезпечення, як API (інтерфейс прикладного програмування) для керування мережею замість традиційних апаратних пристроїв. SDN дозволяють мережевим адміністраторам налаштовувати мережу в одному централізованому місці контролера, який вони можуть остаточно автоматизувати для моніторингу потоку трафіку.

Оскільки адміністратори можуть розробляти нові API для керування трафіком даних, вони можуть створювати політики для проходження даних через набір брандмауерів. Теоретично це може бути надзвичайно безпечним методом сегментації мережі. Однак сегментація та автоматизація SDN часто розглядається як найскладніший метод для впровадження, оскільки багато додатків зазвичай не відповідають набору правил.

1.2 Область застосування

Впроваджуючи політику сегментації мережі, організація може:

– Контролюйте поширення кібератаки та обмежуйте завдану шкоду.

– Кращий контроль доступу для внутрішньої та зовнішньої безпеки мережі.

– Покращте потік трафіку між мережами.

– Простіший моніторинг мережевого трафіку та виявлення загроз.

– Кращий захист конфіденційних даних і кінцевих пристроїв.

– Захищені хмарні сервери.

Що стосується безпеки мережі, слід впроваджувати як політику сегментації мережі, так і політику мікросегментації. У той час як сегментація мережі зосереджена на обмеженні трафіку «північ-південь» між різними мережами або VLAN, мікросегментація забезпечує захист мережі «схід-захід». Це означає обмеження доступу до всіх пристроїв, серверів і програм, які спілкуються між собою.

Сегментацію мережі можна розглядати як всеохоплюючу політику безпеки для всієї інфраструктури, тоді як мікросегментація є більш детальним, детальним підходом до внутрішньомережевого трафіку. Навіть якщо загрозливий суб'єкт може зламати мережу, мікросегментована комп'ютерна мережа завадить йому вільному переміщенню в цій системі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Міжмережним екранам властивий певний набір функцій. От і подивимося, які функції надає (або не надає) той або інший міжмережний екран. Основна функція будь-якого міжмережного екрана – це фільтрація пакетів на підставі певного набору правил. Не дивно, але цю функцію підтримують всі брандмауери.

Також всі розглянуті брандмауери підтримують NAT. Але є досить специфічні (але від цього не менш корисні) функції, наприклад, маскування портів, регулювання навантаження, багатокористувальницьких режим роботи, контроль цілісності, розгортання програми в ActiveDirectory і вилучене адміністрування ззовні. Досить зручно, погодитеся, коли програма підтримує розгортання в ActiveDirectory – не потрібно вручну встановлювати її на кожному комп'ютері мережі. Також зручно, якщо міжмережний екран підтримує вилучене адміністрування ззовні – можна адмініструвати мережа, не виходячи з будинку, що буде актуально для адміністраторів, що звикли виконувати свої функції віддалено.

Основне завдання міжмережних екранів при захисті персональних – це захист мереж. Тому адміністраторові часто однаково, якими додатковими функціями буде володіти міжмережний екран. Йому важливі наступні фактори:

– Час захисту. Тут зрозуміло, чим швидше, тим краще.

– Зручність використання. Не всі міжмережні екрани однаково зручні, що й буде показано в огляді.

– Вартість. Часто фінансова сторона є вирішальною.

– Строк поставки. Нерідко строк поставки залишає бажати кращого, а захистити дані потрібно вже зараз.

Безпека у всіх міжмережних екранів приблизно однакова, інакше в них би не було сертифіката.

Брандмауери в огляді

Далі ми будемо порівнювати три міжмережних екрани – VipNet Office Firewall, Кіберсейф міжмережний екран і TrustAccess.

Брандмауер TrustAccess – це розподілений міжмережний екран із централізованим керуванням, призначений для захисту серверів і робочих станцій від несанкціонованого доступу, розмежування мережного доступу до ИС підприємства.

Кіберсейф міжмережний екран – потужний міжмережний екран, розроблений для захисту комп'ютерних систем і локальної мережі від зовнішніх шкідливих впливів.

ViPNet Office Firewall 4.1 – програмний міжмережний екран, призначений для контролю й керування трафіком і перетворення трафіку (NAT) між сегментів локальних мереж при їхній взаємодії, а також при взаємодії вузлів локальних мереж з ресурсами мереж загального користування.

Час захисту мереж

Що такий час захисту мереж? По суті, цей час розгортання програми на всі комп'ютери мережі й час настроювання правил. Останнє залежить від зручності використання брандмауера, а от перше – від пристосованості його настановного пакета до централізованої установки.

Всі три міжмережних екрани поширюються у вигляді пакетів MSI, а це означає, що можна використовувати засоби розгортання ActiveDirectory для їхньої централізованої установки. Здавалося б все просто. Але на практиці виявляється, що немає.

На підприємстві, як правило, використовується централізоване керування міжмережними екранами. А це означає, що на якийсь комп'ютер установлюється

сервер керування брандмауерами, а на інші встановлюються програми-клієнти або як їх ще називають агенти. Проблема вся в тому, що при установці агента потрібно задати певні параметри – як мінімум IP-адреса сервера керування, а може ще й пароль і т.д.

Отже, навіть якщо ви розгорнете MSI-файли на всі комп'ютери мережі, набудовувати їх однаково прийде вручну. А цього б не дуже хотілося, з огляду на, що мережа більша. Навіть якщо у вас усього 50 комп'ютерів ви тільки вдумайтеся – підійти до кожному ПК і налаштувати його.

Як вирішити проблему? А проблему можна вирішити шляхом створення файлу трансформації (MST-файлу), він же файл відповідей, для MSI-файлу. От тільки ні VipNet Office Firewall, ні TrustAccess цього не вміють. Розгорнути те ці програми в домені можна, але потрібна ручна робота адміністратора.

Звичайно, адміністратор може використовувати редактори начебто Orca для створення MST-файлу.

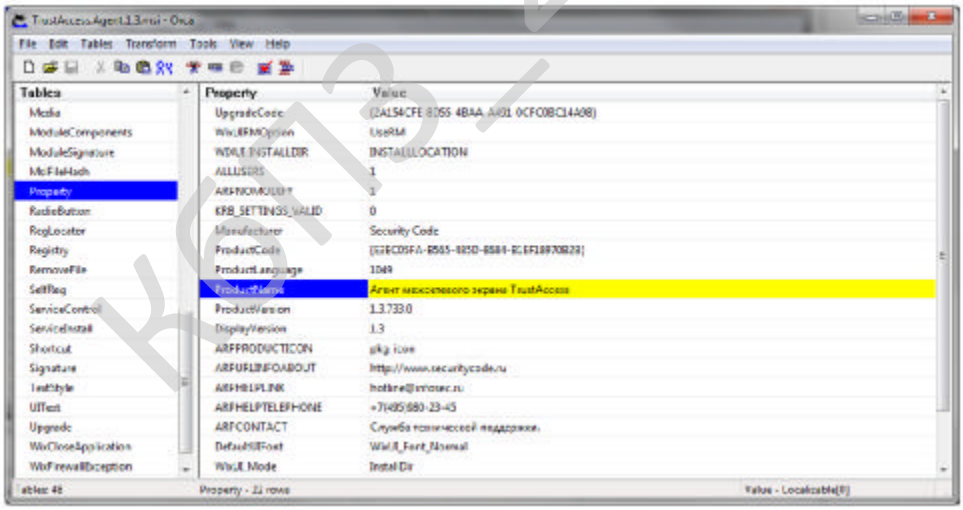

Рисунок 2.1 – Редактор Orca. Спроба створити MST-файл для TrustAccess.Agent.1.3.msi

Але невже ви думаєте, що все так просто? Відкрив MSI-файл в Orca, підправив пару параметрів і одержав готовий файл відповідей? Не отут те було!

По-перше, сам Orca просто так не встановлюється. Потрібно скачати Windows Installer SDK, з нього за допомогою 7-Zip витягти orca.msi і встановити його. Ви про це знали? Якщо ні, тоді вважайте, що витратили мінут 15 на пошук потрібної інформації, завантаження ПЗ й установку редактора. Але на цьому всі мучення не закінчуються. В MSI-файлу безліч параметрів. Подивитеся на рисунок 2.1 – це тільки параметри групи Property. Який з них змінити, щоб указати IP-адреса сервера? Ви знаєте? Якщо ні, тоді у вас два варіанти: або вручну налаштувати кожний комп'ютер або звернутися до розроблювача, чекати відповідь і т.д. З огляду на, що розроблювачі іноді відповідають досить довго, реальний час розгортання програми залежить тільки від швидкості вашого переміщення між комп'ютерами. Добре, якщо ви завчасно встановили інструмент вилученого керування – тоді розгортання пройде швидше.

Кіберсейф

Міжмережний екран самостійно створює MST-файл, потрібно лише встановити його на один комп'ютер, одержати заповітний MST-файл і вказати його в груповій політиці. Про те, як це зробити, можна прочитати в статті «Розмежування інформаційних систем при захисті персональних даних». За якісь півгодини (а те й менше) ви зможете розгорнути міжмережний екран на всі комп'ютери мережі.

Саме тому Кіберсейф міжмережний екран одержує оцінку 5, а його конкуренти – 3 (спасибі хоч інсталятори виконані у форматі MSI, а не .exe).

Зручність використання

Брандмауер – це не текстовий процесор. Це досить специфічний програмний продукт, використання якого зводиться до принципу «установив, налаштував, забув». З одного боку, зручність використання – другорядний фактор. Наприклад, iptables в Linux не можна назвати зручним, але адже ним же користуються? З іншого боку – чим зручніше брандмауер, тим швидше вийде захистити мереж і виконувати деякі функції по її адмініструванню.

Що ж, давайте подивимося, наскільки зручні розглянуті міжмережні екрани в процесі створення й захисту мереж.

Почнемо ми з VipNet Office Firewall, що, на наш погляд, не дуже зручний. Виділити комп'ютери в групи можна тільки по IP-адресах (рисунок 2.2). Інакше кажучи, є прив'язка до IP-адрес і вам потрібно або виділяти різні мереж у різні підмережі, або ж розбивати одну підмережу на діапазони IP-адрес. Наприклад, є три мереж: Керування, Бухгалтерія, IT. Вам потрібно налаштувати DHCP-сервер так, щоб комп'ютерам із групи Керування IP-адресами з діапазону 192.168.1.10 – 192.168.1.20, Бухгалтерія 192.168.1.21 – 192.168.1.31 і т.д. Це не дуже зручно. Саме за це з VipNet Office Firewall буде знятий один бал.

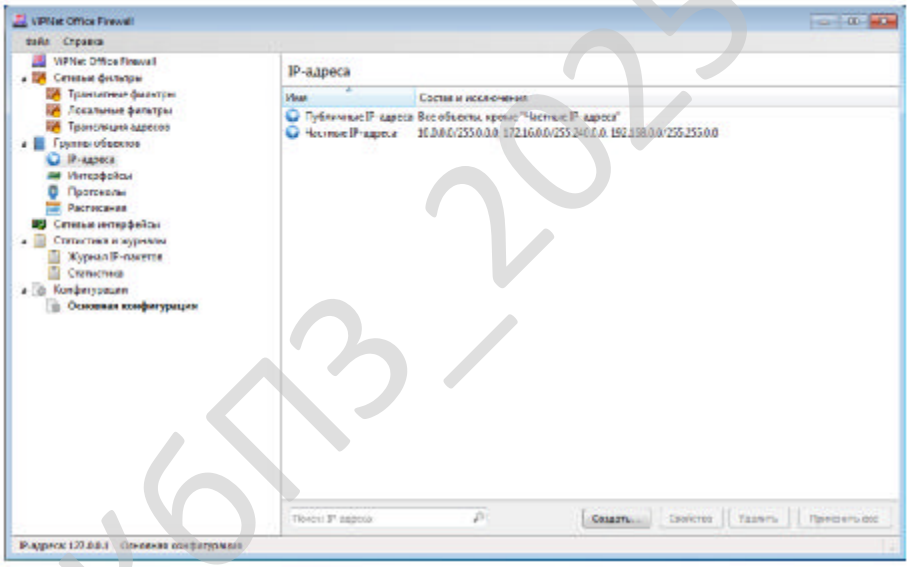

Рисунок 2.2 – При створенні груп комп'ютерів спостерігається явна прив'язка до IP-адреси

У міжмережному екрані Кіберсейф, навпаки, немає ніякої прив'язки до IP-адреси. Комп'ютери, що входять до складу групи, можуть перебувати в різних підмережах, у різних діапазонах однієї підмережі й навіть перебувати за межами мережі. Філії компанії розташовані в різних містах (Київ, Кропивницький і т.д.). Створити групи дуже просто – досить перетягнути імена комп'ютерів у потрібну

групу й нажати кнопку **Застосувати**. Після цього можна нажати кнопку **Встановити правила** для формування специфічних для кожної групи правил.

Що стосується TrustAccess, те потрібно відзначити тісну інтеграцію із самою системою. У конфігурацію брандмауера імпортуються вже створені системні групи користувачів і комп'ютерів, що полегшує керування міжмережним екраном у середовищі ActiveDirectory. Ви можете не створювати мереж у самому брандмауері, а використовувати вже наявні групи комп'ютерів у домені Active Directory.

Всі три брандмауери дозволяють створювати так звані розклади, завдяки яким адміністратор може налаштувати проходження пакетів за розкладом, наприклад, заборонити доступ до Інтернету в неробочий час. В VipNet Office Firewall розклади створюються в розділі **Розкладу**, а в Кіберсейф міжмережний екран час роботи правила задається при визначенні самого правила.

Всі три брандмауери надають дуже зручні засоби для створення самих правил. А TrustAccess ще й надає зручний майстер створення правила.

Глянемо на ще одну особливість – інструменти для одержання звітів (журналів, логів). В TrustAccess для збору звітів і інформації про події потрібно встановити сервер подій (EventServer) і сервер звітів (ReportServer). Не те, що це недолік, а скоріше особливість («feature», як говорив Білл Гейтс) даного брандмауера. Що стосується, міжмережних екранів Кіберсейф і VipNet Office, те обидва брандмауери надають зручні кошти перегляду журналу IP-пакетів. Різниця лише в тому, що в Кіберсейф міжмережний екран спочатку відображаються всі пакети, і ви можете відфільтрувати потрібні, використовуючи можливості убудованого в заголовок таблиці фільтра (рисунок 2.9). А в VipNet Office Firewall спочатку потрібно встановити фільтри, а потім уже переглянути результат.

З міжмережного екрана Кіберсейф довелося зняти 0.5 бала за відсутність функції експорту журналу в Excel або HTML. Функція далеко не критична, але

іноді корисно просто й швидко експортувати з журналу кілька рядків, наприклад, для «розбору польотів».

Що вибрати

Якщо орієнтуватися тільки за вартістю продукту й технічної підтримки, то вибір очевидний – Кіберсейф міжмережний екран. Кіберсейф міжмережний екран має оптимальне співвідношення функціонал/ціна. З іншого боку, якщо вам потрібна підтримка Secret Net, те потрібно дивитися убік TrustAccess. А от VipNet Office Firewall можемо порекомендувати хіба що як гарного персонального брандмауера, але для цих цілей існує безліч інших і до того ж безкоштовних рішень.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

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

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-додатків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Впровадження сегментації мережі може бути дорогим і трудомістким. Якщо це зробити неправильно, це може вимагати значних інвестицій для виправлення та перебудови всієї мережевої архітектури, що може спричинити крах цілих організацій або посилити ризики для безпеки. Ось найкращі методи сегментації мережі, щоб ви могли захистити свою організацію від будь-якої атаки зловмисного програмного забезпечення.

Постійний моніторинг і аудит мереж

Усі процеси сегментації повинні включати постійний моніторинг мережевого трафіку та продуктивності мережі, щоб переконатися, що в мережевій інфраструктурі немає прогалин або вразливостей . Регулярні оцінки мережевих ризиків і тести на проникнення необхідні для виявлення проблем безпеки, які потребують негайної уваги.

Щорічні мережеві аудити також важливі, оскільки вони дозволяють організаціям переоцінити ефективність поточної політики безпеки. Протягом року могли з’явитися нові користувачі, процеси або бізнес-потреби, що вимагає оновлення плану сегментації мережі.

Уникайте надмірної або недостатньої сегментації

Коли організації впроваджують сегментацію мережі, поширеною помилкою є надмірна сегментація на занадто багато мереж або недостатня сегментація на занадто малу кількість мереж. Великі корпоративні мережі часто припускають, що сегментування, наскільки це можливо, створює найвищий рівень безпеки. Має бути баланс між достатньою кількістю ресурсів для контролю та моніторингу кількох мереж без впливу на продуктивність співробітників.

Надмірна сегментація може змусити співробітників переходити через кілька точок доступу, щоб отримати доступ до даних, створюючи неефективність робочого процесу та обмежуючи потік трафіку. Це також може створити більше вразливостей, якщо кожною мережевою системою не керувати належним чином. Впровадження будь-яких оновлень безпеки через кожну окрему мережу займе набагато більше часу, що також може збільшити ризик помилок.

Недостатня сегментація мережі також може виявитися неефективною, якщо між кожною системою недостатньо відокремлено. Поділ однієї мережі лише на дві чи три не забезпечить рівень безпеки, необхідний для правильного сегментування мережі. В ідеалі є достатньо мереж, щоб максимально обмежити поверхню атаки .

Обмежте точки доступу третіх сторін

Крім захисту власних точок доступу, кожна організація повинна обмежувати ризики для третіх сторін і керувати ними . Не всім стороннім постачальникам або постачальникам потрібен повний доступ до серверів компанії, щоб продовжувати працювати на повну потужність. Організації повинні дозволяти постачальникам лише мінімальний рівень доступу для виконання своїх функцій, щоб запобігти впливу потенційного порушення даних .

Навіть у захищену мережу може проникнути скомпрометована третя сторона. Один із способів ізолювати сторонній доступ – це створити унікальні портали з індивідуальними засобами контролю доступу для кожного постачальника. Крім того, регулярні перевірки на наявність ризиків третіх сторін також можуть допомогти запобігти майбутнім витокам даних . Дотримання найкращих методів керування ризиками постачальників може запобігти тому, що потенційні точки доступу сторонніх розробників не пропадуть за вашим радаром.

Визначте та позначте вартість активів

Перед початком будь-яких процесів сегментації мережі організації повинні проаналізувати свої активи та призначити їм значення. Кожен актив, який може включати все, від пристроїв Інтернету речей (Інтернет речей) до баз

даних, має бути організовано за рівнем важливості та чутливістю даних.

Відокремлення елементів нижчої та вищої вартості при збереженні повного списку активів компанії дозволяє спростити перехід і реалізацію стратегії сегментації мережі.

Об’єднайте подібні мережеві ресурси

Після того, як інвентаризація активів буде задокументована, наступним кроком є початок групування схожих мережевих ресурсів. Елементи з нижчим рівнем безпеки слід розміщувати в одній мережі, а інші активи з вищим рівнем безпеки – в іншій. Організації можуть розміщувати посилені протоколи безпеки в мережах з більш критичними даними для захисту, коли мережева архітектура починає формуватися.

Ця практика може спростити створення та оновлення політик безпеки для кожної мережі та визначити, які мережі мають пріоритет над іншими. Це також робить моніторинг і фільтрацію мережі набагато доступнішими.

Впровадити безпеку та захист кінцевої точки

Кібератаки часто спрямовані на кінцеві пристрої, оскільки вони часто незахищені та не мають належного захисту. Один зламаний пристрій може створити точку входу для хакерів у всю основну мережу. Впровадження таких технологій, як виявлення та реагування на кінцеві точки (EDR), дозволяє організаціям забезпечити додатковий рівень безпеки шляхом проактивного моніторингу IOA (індикаторів атак) і IOC (індикаторів компрометації).

Дотримуйтеся принципу найменших привілеїв

Після реалізації сегментації мережі кожна мережа повинна дотримуватися моделі нульової довіри та принципу найменших привілеїв . Ці методи передбачають заборону доступу до мережі на всіх рівнях, що вимагає від усіх сторін периметра мережі, як внутрішніх, так і зовнішніх, забезпечити автентифікацію та перевірку перед отриманням доступу до інших частин мережі.

Завдяки архітектурі нульової довіри (ZTA) мережеві адміністратори можуть швидко ідентифікувати будь-яких зловмисників або неавторизованих

осіб, які намагаються проникнути в системи. Лише авторизовані користувачі з відповідними дозволами можуть отримати доступ до даних у цій конкретній мережі.

Створюйте простіші легітимні шляхи даних

Під час сегментації мережі важливо враховувати шлях даних авторизованого користувача порівняно з брандмауерам, що захищає дані. Законному користувачеві чи третій стороні не доведеться проходити через більше точок доступу, ніж потрібно брандмауерам, щоб зловмисники могли зламати їх. Переконайтеся, що архітектура вашої мережі має більше захисту від кіберзагроз, ніж брандмауери між вашими постачальниками та даними, до яких вони потребують доступу.

3.2 Розробка структурної схеми

У сфері мережевої безпеки концепція внутрішнього брандмауера сегментації (ISFW) набирає популярності, оскільки підприємства прагнуть зміцнити свої внутрішні мережі від бокового руху та прогресивних загроз.

ISFW забезпечує контроль доступу та правила брандмауера в центрі обробки даних, а також розширює захисну зону на маршрутизатори, VLAN і підмережі. Завдяки інтеграції ISFW компанії можуть підвищити кібербезпеку, зменшити поверхню атаки та узгодити модель нульової довіри. Давайте заглибимося в призначення, розгортання та переваги ISFW для сучасних складних корпоративних мереж.

Призначення та розгортання ISFW

Мета внутрішнього сегментованого брандмауера (ISFW) полягає в тому, щоб додати рівень безпеки у вашу внутрішню мережу, по суті розбиваючи мережу на менші, більш керовані та безпечні сегменти. Ця сегментація допомагає контролювати, хто має доступ до різних частин мережі, зменшуючи ймовірність бокового переміщення кібер-зловмисників.

Розгортання ISFW вимагає ретельного планування та розгляду. Йдеться не лише про встановлення бар'єрів; йдеться про розуміння потоку вашого мережевого трафіку та забезпечення того, що необхідний зв'язок може відбуватися без перешкод, одночасно блокуючи потенційно шкідливий трафік. Подумайте про це як про створення серії контрольних точок у вашій внутрішній мережі, причому кожна контрольна точка має власний набір правил і норм, адаптованих до конкретних потреб і вимог безпеки цього сегмента мережі.

Різниця між брандмауерам і сегментацією

Обговорюючи сегментацію мережі в кібербезпеці, дуже важливо розуміти різницю між традиційними брандмауерами периметра та сегментацією. Брандмауер периметра діє як воротар вашої мережі від зовнішнього світу. Навпаки, сегментація, а точніше ISFW, розділяє вашу внутрішню мережу на окремі зони, щоб утримувати зломи та запобігати зловмисникам від отримання доступу до всієї мережі.

Рекомендації щодо сегментації мережі

Застосування найкращих практик для сегментації корпоративної мережі є проактивним кроком до підвищення рівня кібербезпеки. Ось кілька ключових стратегій:

– Визначте чіткі правила сегментації та послідовно дотримуйтеся їх у всій мережі.

– Переконайтеся, що політики сегментації відображають критичність і чутливість даних у різних сегментах.

– Регулярно переглядайте та оновлюйте правила сегментації для адаптації до змін в архітектурі мережі або бізнес-процесах.

Переваги ISFW і Zero Trust

Розгортання ISFW добре узгоджується з моделлю Zero Trust, яка працює за принципом «ніколи не довіряй, завжди перевіряй». Сегментуючи внутрішню мережу, ISFW допомагають застосувати цю модель, гарантуючи, що кожен запит

на доступ оцінюється критично, незалежно від джерела. Цей підхід зменшує площу атаки та пом’якшує вплив потенційних порушень.

У роботі наведена реалізація внутрішніх міжсегментних мережних екранів ISFW для захисту від безлічі атак з боку IoT за допомогою адаптивної системи мережної безпеки Недавні атаки, провідником яких став IoT, продемонстрували легкість використання мільярдів пристроїв для поразки мільйонів користувачів і віртуальних економік цілих країн. Ця проблема збільшується відсутністю базових функцій захисту й керування в багатьох пристроях IoT.

Зазначена проблема найбільш актуальна для організацій, дані яких переміщаються між різними середовищами й пристроями – від планшетів до хмарних додатків – і, отже, мають потребу в особливому захисту. Організаціям необхідно звернути увагу на три стратегічні функції мережної безпеки, що забезпечують захист інфраструктури від погроз із боку IoT:

– Аналіз – повне подання про стан мережі відіграє значну роль у процесі перевірки дійсності й класифікації пристроїв IoT, розробки профілів ризику й розподілу пристроїв IoT по групах залежно від рівня їхньої надійності. В основі адаптивної системи мережної безпеки лежить рішення, що надає повні відомості про стан кожного елемента безпеки й компонент корпоративної мережі. Завдяки цьому IT-фахівці можуть ідентифікувати пристрою IoT і трафік у критичних точках інфраструктури й управляти ними.

– Сегментація – організації повинні мати можливість розділяти пристрої й канали зв'язку IoT на групи на основі політик і надавати базові повноваження залежно від профілю ризику певного пристрою IoT. Міжмережний екран підтримує реалізацію внутрішньої сегментації корпоративних мереж і пристроїв, завдяки чому IT-фахівці можуть застосовувати деталізовані політики безпеки залежно від типу пристроїв і вимог доступу до мережі.

– Захист – адаптивна система мережної безпеки підтримує зіставлення відомостей про порушення безпеки IoT з даними про погрози з метою синхронізації реагування на погрози IoT. Також система безпеки дозволяє

поміщати заражені пристрої IoT у карантин і усувати уразливості в різних точках мережі з метою запобігання поширення погроз і доступу шкідливого трафіку до важливих IT-системам і корпоративним даним.

Система безпеки мереж на основі цілей дозволяє автоматизувати реалізацію пов'язаних з IoT корпоративних стратегій і операцій шляхом задоволення потреб бізнесу за рахунок вживання погоджених заходів забезпечення мережної безпеки без втручання фахівця.

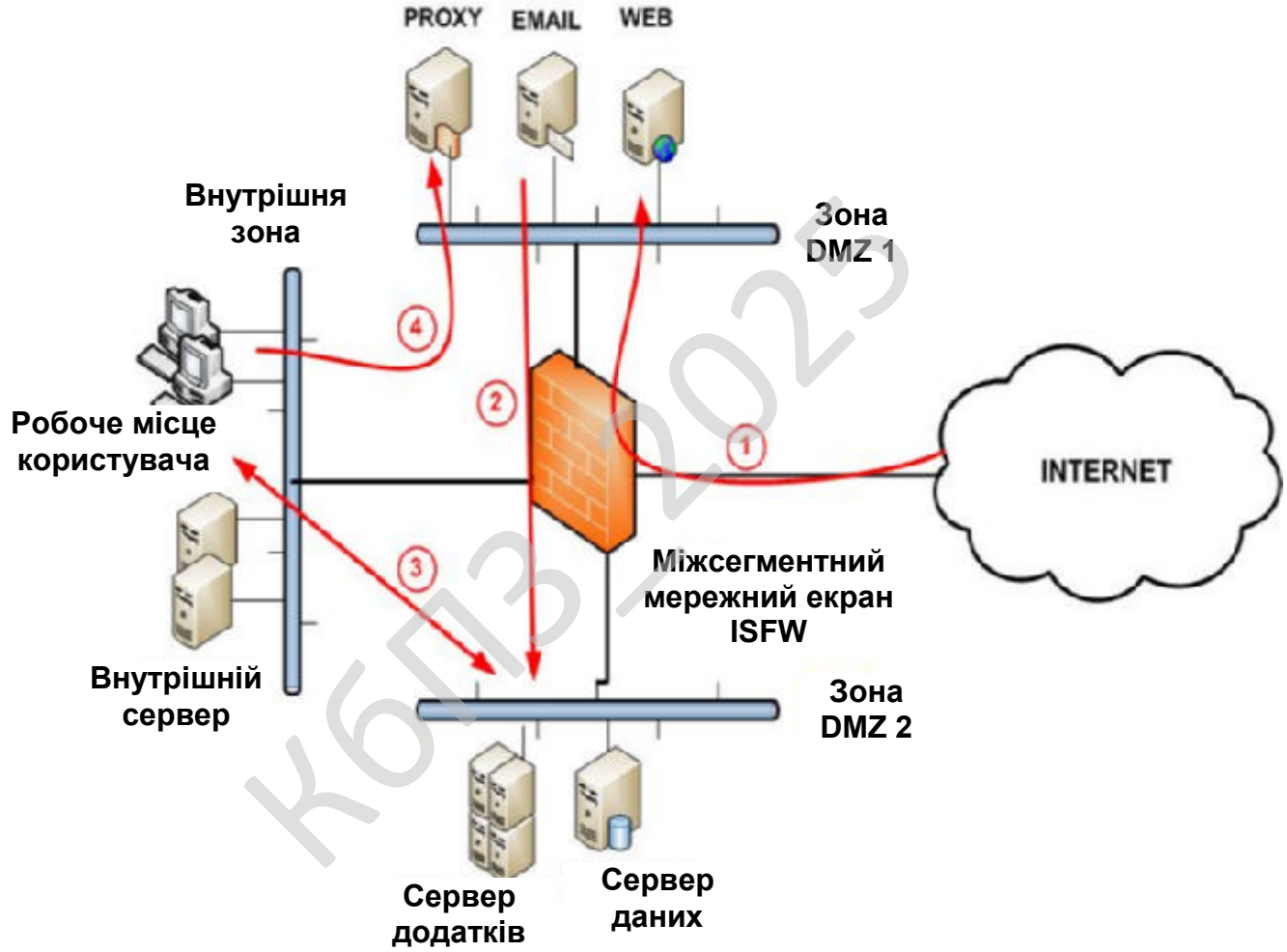

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Основною технологією ISFW залишається детальний контроль на рівні додатків, однак «підтримка додатків» у сучасних міжмережних екранах істотно

відрізняється від того, що пропонувалося 20 років тому. Технологія міжмережних екранів була значно вдосконалена – вона еволюціонувала до спеціалізованих рішень, що виконують глибокий аналіз трафіку й ідентифікацію додатків. Відповідні продукти стали працювати швидше й підтримують більше складні набори правил, чим їхні попередники.

Аналітики відзначають, що в останні два-три роки росте попит на платформи ISFW, здатні виявляти й блокувати витончені атаки, задавати (з високим ступенем деталізації) політики безпеки на рівні додатків, а не тільки портів і протоколів. Функціонал і продуктивність міжмережних екранів повинні відповідати вимогам більше складної взаємодії з додатками, а самі пристрої – мати високу пропускну здатність і підтримувати віртуалізацію. Вибір рішення визначається такими факторами, як вартість, зручність керування, простота й швидкість розгортання. Звичайно, список цим не вичерпується.

При порівнянні або розробці методики вибору міжмережних екранів аналітики оперують декількома десятками (іноді до півтори сотень) критеріїв, які варто враховувати, вибираючи рішення. Кожний замовник по-своєму розставляє пріоритети – універсального рецепта або сценарію немає й бути не може.

Нові погрози й технології Web 2.0 змушують вендорів обновляти свої пропозиції – міжмережні екрани еволюціонують. Вони оснащуються функціями глибокого аналізу трафіку й надають можливість гнучкого настроювання політики, а їхня продуктивність збільшується відповідно до росту пропускної здатності мереж. ISFW здатні контролювати мережний трафік на рівні додатків і користувачів і активно блокувати погрози. Вони можуть містити в собі цілий ряд додаткових засобів забезпечення безпеки й підтримувати розвинені мережні функції.

Великі підприємства й провайдери мають потребу у високопродуктивних рішеннях. Новітні системи будуються на потужних апаратних платформах, причому як інтегровані компоненти в них використовуються раніше розрізнені засоби й функції безпеки – IPS, глибокий аналіз пакетів, автентифікація

користувачів і багато чого іншого. Однак міжмережні екрани корпоративного рівня характеризуються не специфічним набором функцій, а масштабованістю, керованістю й надійністю, які відповідають потребам великих компаній.

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

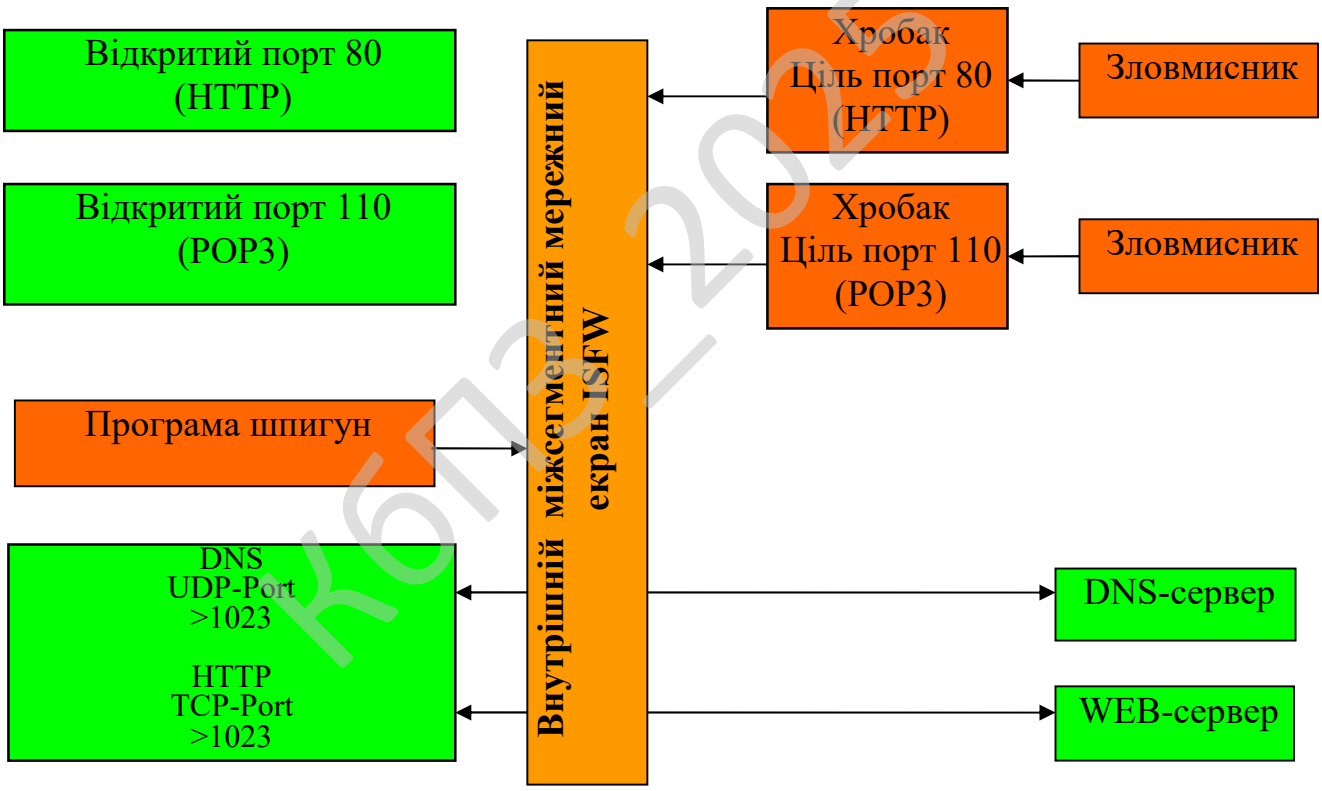

Рисунок 3.2 – Функціональна схема системи

Інтегрований маршрутизатор дозволяє налаштувати примітивну DMZ для доступу до внутрішнього сервера з вузлів за межами мережі. Для цього сервер повинен мати статичну IP-адресу, що вказується в конфігурації DMZ.

Інтегрований маршрутизатор ізолює трафік, що пересилається на зазначений IP-адрес. Цей трафік пропускається тільки на той порт комутатора, до якого підключений сервер. На всі інші вузли як і раніше поширюється захист міжмережного екрану. При активації DMZ у найпростішому виді зовнішні вузли одержують доступ до всіх портів сервера, наприклад 80 (HTTP), 21 (FTP) і 110 (POP3 для електронної пошти). Функція переадресації портів дозволяє налаштувати більш строгу конфігурацію DMZ. У цьому випадку вказуються порти, які повинні бути доступні на сервері. Пропускається тільки трафік, спрямований на ці порти. Весь інший трафік виключається. Бездротова точка доступу в складі інтегрованого маршрутизатора часто вважається частиною внутрішньої мережі. Необхідно усвідомлювати, що при роботі точки доступу в незахищеному режимі всі користувачі, що підключилися до неї, одержують доступ до внутрішньої захищеної мережі без проходження міжмережного екрану. Зловмисники можуть у такий спосіб одержати доступ до внутрішньої мережі, минаючи всі засоби захисту.

ДМЗ-хост

Деякі маршрутизатори SOHO-класу мають функцію надання доступу із зовнішньої мережі до внутрішніх серверів (режим DMZ host або exposed host). У такому режимі вони являють собою хост, у якого відкриті (не захищені) всі порти, крім тих, які транслюються іншим способом. Це не цілком відповідає визначенню щирої ДМЗ, тому що сервер з відкритими портами не відділяється від внутрішньої мережі. Тобто ДМЗ-хост може вільно підключитися до ресурсів у внутрішній мережі, у те час як з'єднання із внутрішньою мережею зі справжньої ДМЗ блокуються поділяючої їх міжмережним екраном, якщо немає спеціального розв'язного правила [1].ДМЗ-хост не надає в плані безпеки жодного з переваг, які дає використання подсетей, і часто використовується як простий метод трансляції всіх портів на інший міжмережний екран або пристрій [5] [11].

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Рисунок 3.3 – Діаграма взаємодії процесів

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації, також те, що при розробці програми слід надати особливу увагу модулю спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

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

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

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірки поточного стану та поверненням на початок схеми чи з завершенням роботи розробленого ПЗ.

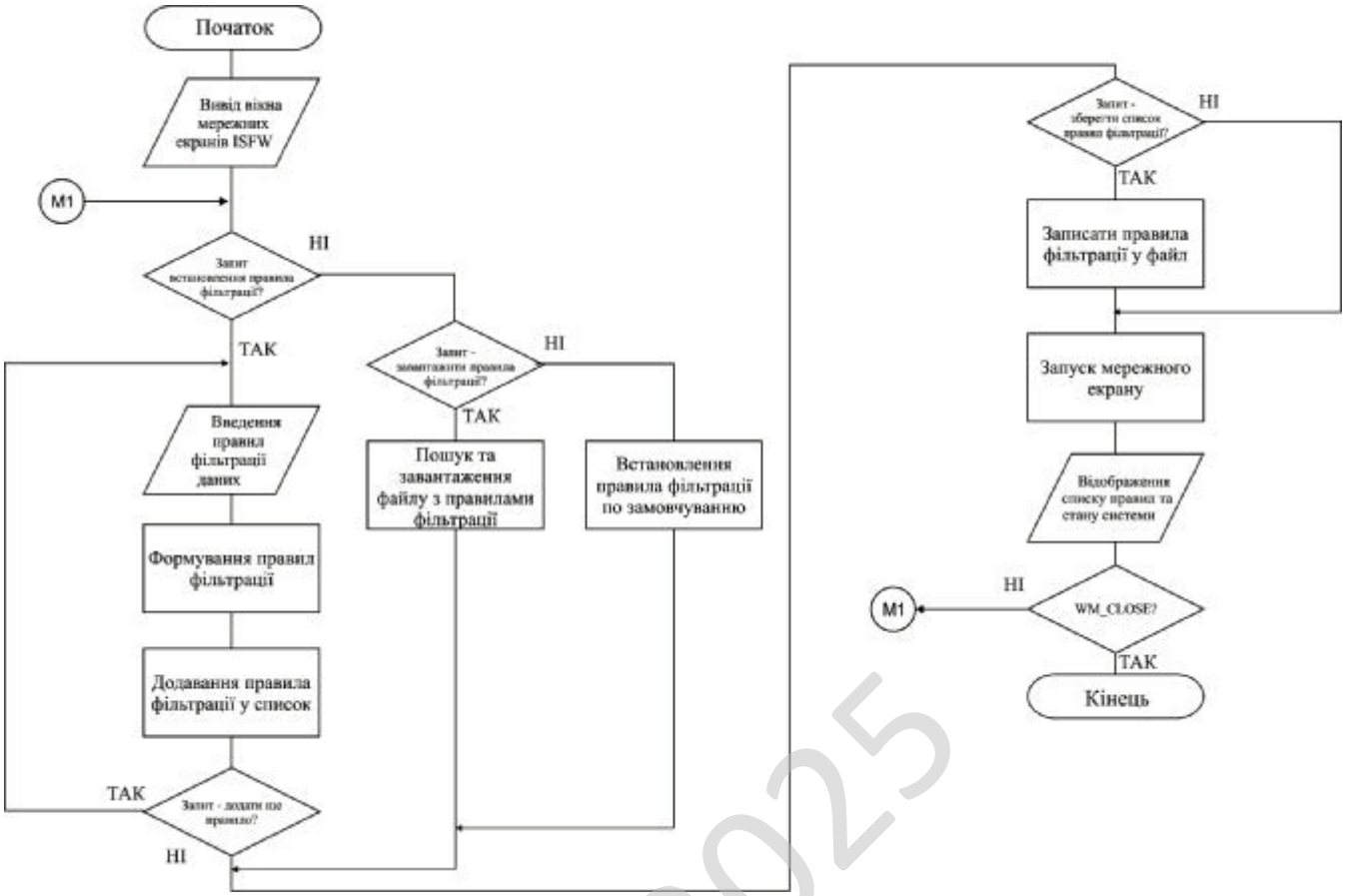

Рисунок 4.1 – Блок-схема основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

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

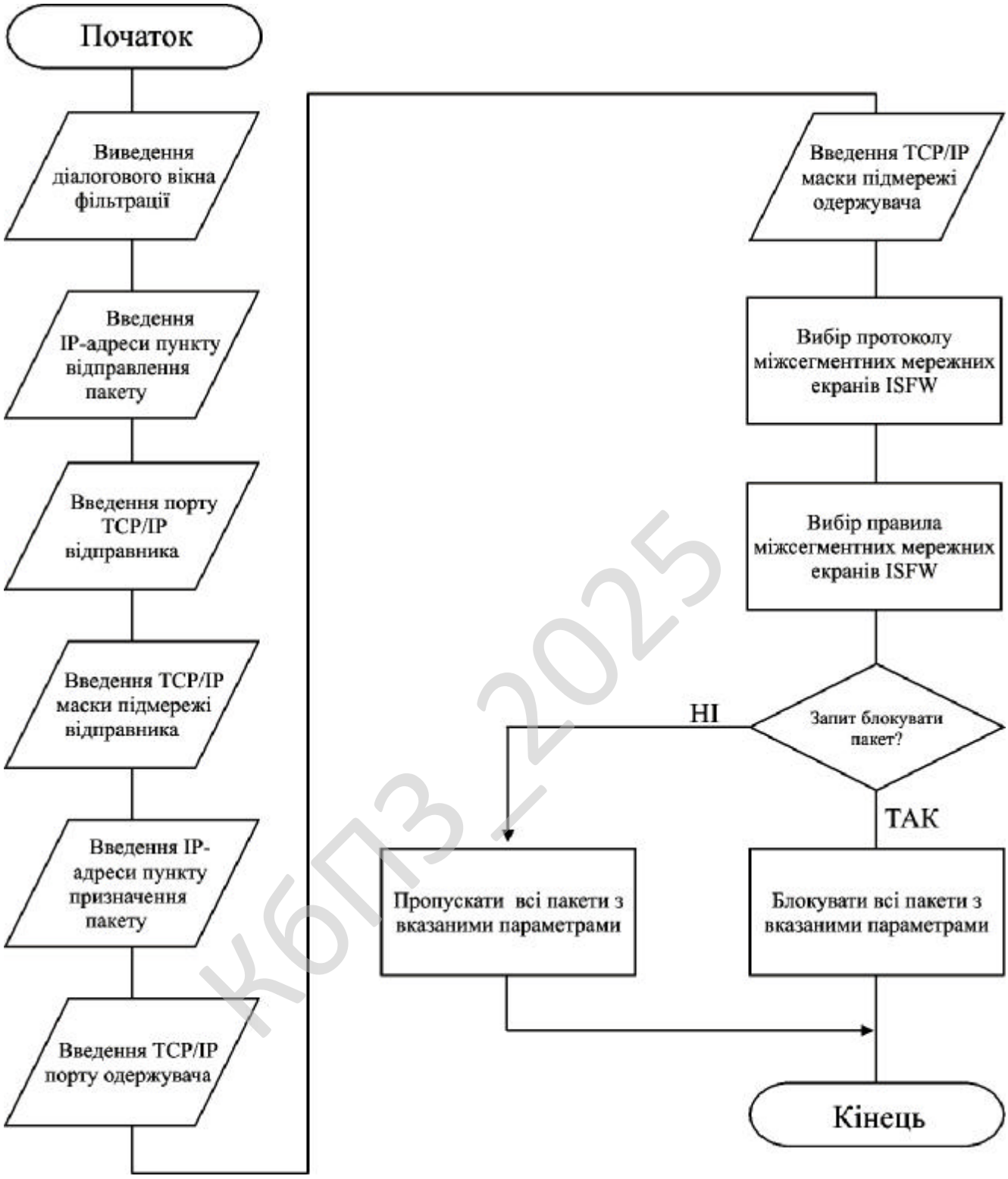

Рисунок 4.2 – Блок-схема роботи підпрограми

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень.

Подійно-орієнтована архітектура (Event-driven architecture, надалі **EDA**) – шаблон архітектури програмного забезпечення, який призначений для створення подій, їх виявлення, споживання і реагування на них.

Подія може бути визначена як значна зміна стану. Наприклад, коли споживач купує автомобіль, стан автомобіля змінюється з "на продаж" до "продано". Архітектура системи дилера автомобілів може трактувати цю зміну стану як подію, поява якої може стати відомою іншим програмам даної архітектури.

З формальної точки зору, те, що виробляється, публікується, поширюється, виявляється і споживається (як правило, асинхронно) є повідомленням, яке називають сповіщенням про подію (або нотифікацією), а не самою подією, яка є зміною стану, що викликає появу повідомлення.

Події не подорожують, вони просто відбуваються. Проте термін подія часто використовується метонімічно для позначення самого нотифікаційного повідомлення, що може призвести до певної плутанини.

Цей архітектурний шаблон може застосовуватися при проектуванні і реалізації ПЗ і систем, які передають події між слабкозв’язаними компонентами програмного забезпечення і сервісами (службами).

Подійно-орієнтована система як правило складається з емітерів подій (або агентів) і споживачів подій (або стоків).

Стоки несуть відповідальність за здійснення реагування на появу події. Реакція не завжди може бути повністю забезпечена самим стоком. Наприклад, стік, може бути відповідальним лише за фільтрацію, трансформацію і відправку події до іншого компонента або він може забезпечити повністю самостійну реакцію на таку подію. Перша категорія стоків може бути заснована на традиційних компонентах, таких як проміжне програмне забезпечення, орієнтоване на обробку повідомлень (message oriented middleware, MOM), в той

час, як друга категорія стоків (самостійна реакція в режимі он-лайн) може вимагати більш придатної платформи (фреймворку) для виконання транзакцій.

Розробка ПЗ і систем в подійно-орієнтованій архітектурі дозволяє їм бути сконструйованими способом, який більш відповідає вимогам до їх створення, оскільки такі системи в більшій мірі пристосовуються до непередбачуваних і асинхронних середовищ.

Подійно-орієнтована архітектура (EDA) може доповнювати сервісно-орієнтовану архітектуру (SOA), оскільки сервіси (служби) можуть бути активовані тригерами, які ініціюються при настанні подій.

Ця парадигма особливо корисна, коли стік не забезпечує власного виконання будь-яких дій.

Подійно-орієнтована SOA (SOA-2) розвиває архітектури SOA і EDA для забезпечення більш глибокого і надійного рівня сервісів за рахунок використання раніше невідомих причинно-наслідкових зв'язків, щоб сформувати новий шаблон подій. Цей новий шаблон бізнес-аналітики дає поштовх до небаченого раніше зростання рівня автоматизації підприємства за рахунок привнесення додаткової цінної інформації в описану раніше модель діяльності.

Обчислювальна техніка та сенсорні пристрої (сенсори, датчики, контролери) можуть виявляти зміни стану об'єктів або умов і створювати події, які потім можуть бути оброблені сервісом (службою) або системою.

Подія може складатися з двох частин: заголовка події та тіла події. Заголовок події може включати в себе інформацію таку як, наприклад, назва події, часова мітка події і тип події. Тіло події – це частина, яка описує факт, що стався в дійсності. Тіло події не слід плутати з шаблоном або логікою, яка може бути застосована як реакція на саму події.

Архітектура, керована подіями, складається з чотирьох логічних рівнів (шарів). Вона починається з виявлення факту, його технічного подання у формі події і закінчується непустою множиною реакцій на цю подію.

Генератор подій

Першим логічним шаром є генератор подій, який виявляє факт і представляє цей факт подією. Оскільки фактом може бути практично все, що може бути сприйнято, то ним може бути і генератор подій. Наприклад, генератором може бути клієнт електронної пошти, система електронної комерції або певний тип датчика.

Перетворення різних даних, отриманих від датчиків, в єдину стандартизовану форму даних, які можуть бути оцінені, є основною проблемою при розробці та реалізації цього шару. Однак, враховуючи, що подія є строго декларативною, можна легко застосовувати будь-які операції трансформації, тим самим усуваючи необхідність забезпечення високого рівня стандартизації.

Канал подій

Канал подій – це механізм, через який інформація від генератора подій передається до обробника подій (event engine) або стоку.

Це може бути з'єднання TCP/IP або вхідний файл будь-якого типу (простий текст, формат XML, e-mail тощо). В один і той же час може бути відкрито кілька каналів подій. Як правило, оскільки обробник подій повинен працювати в режимі, наближеному до реального часу, канали подій зчитуються асинхронно. Події зберігаються в черзі, очікуючи наступної обробки механізмом обробки подій.

Механізм обробки подій

Механізм обробки подій (event processing engine) є місцем, де подія ідентифікується і вибирається відповідна реакція на нього, яка потім виконується. Це також може призвести до породження ряду тверджень. Якщо подія, яка надійшла до механізму обробки подій, є наприклад такою «Запаси продукту ID досягли нижнього допустимого рівня», це може ініціювати, наприклад, такі реакції як «Замовити продукт ID» і «Сповістити персонал».

Наступна подійно-орієнтована дія (післядія)

Щодо того, як можуть проявлятися наслідки події, слід відмітити, що вони можуть проявитись багатьма різними способами і у різноманітних формах

(наприклад, повідомлення електронної пошти, надіслане комусь, або ПЗ, що виводить деяке попередження на екран). Залежно від рівня автоматизації, який забезпечується стоком (механізмом обробки подій), ці дії можуть виявитись зайвими.

Є три основні стилі обробки подій: простий, потоковий і складний. Часто ці три стилі використовуються спільно у розвинутій подійно-орієнтованій архітектурі.

Проста обробка подій

Проста обробка подій стосується подій, які безпосередньо належать до специфічних вимірних змін умов. У випадку простої обробки подій, мають справу з появою відомих подій, що ініціюють післядію (післядії). Проста обробка подій зазвичай використовується для управління потоком робіт в реальному часі, скорочуючи тим самим час затримки і вартість робіт.

Наприклад, прості події можуть створюватись (породжуватись) датчиком, що виявляє зміну тиску в шині або температуру навколишнього середовища.

Обробка потоку подій

При обробці потоку подій (event stream processing, далі ESP) відбуваються як звичайні, так і відомі події. Звичайні події (заявки, передачі RFID) перевіряються на те, чи є вони відомими, і передаються інформаційним передплатникам. Обробка потоку подій зазвичай використовується для управління потоком інформації в реальному часі і на рівні підприємства, що дозволяє своєчасно приймати рішення.

Обробка складних подій

Обробка складних подій (Complex event processing (CEP)) дозволяє за шаблонами простих і звичайних подій проводити аналіз того, чи наступила складна подія. Обробка складних подій полягає в оцінюванні взаємного впливу подій і в наступному виконанні дій. При цьому, типи подій (відомих або звичайних) можуть перетинатись, а події можуть виникати протягом тривалого періоду часу.

Кореляція подій може бути причинною, тимчасовою або просторовою. CEP вимагає використання складних інтерпретаторів подій, визначення і підбору шаблонів подій, а також відповідних кореляційних методів. Обробка складних подій зазвичай використовується для виявлення і реагування на аномальну поведінку, загрози і можливості у бізнесі.

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

```
void CMainFrame::OnButtonstart()
{
     CFirewallAppDoc *doc = (CFirewallAppDoc *)GetActiveDocument();
     unsigned int i;
     DWORD result;
     PIP_ADAPTER_INFO pAdapterInfo = NULL, aux;
     IP_ADDR_STRING *localIp;
     unsigned long len = 0;
//Пошук адаптера мережної карти
     GetAdaptersInfo(pAdapterInfo, &len);
     pAdapterInfo = (PIP_ADAPTER_INFO) malloc (len);
     result = GetAdaptersInfo(pAdapterInfo, &len);
     if(result != ERROR_SUCCESS)
     {
          AfxMessageBox("Error getting adapters info.");
          return;
     }
// Посилка правил на інтерфейс адаптера
     for(i=0;i<doc->nRules;i++)
     {
// на всі знайдені адаптери
          for(aux=pAdapterInfo;aux != NULL;aux=aux->Next)
          {
// на кожний IP адаптера
     for(localIp=&aux->IpAddressList;localIp!=NULL;localIp=localIp->Next)
               {
pckFilter.AddF(CharToIp(localIp->IpAddress.String),
                              ANY_DIRECTION,
                         doc->rules[i].sourceIp,
                         doc->rules[i].sourceMask,
                         doc->rules[i].destinationIp,
```

```
                        doc->rules[i].destinationMask,
                        doc->rules[i].sourcePort,
                        doc->rules[i].destinationPort,
                        doc->rules[i].protocol);
                }
            }
        }
        started = TRUE;
}
```

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

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

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

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

```
    }
    m_wndToolBar.EnableDocking(CBRS_ALIGN_RIGHT);
  EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);
// Установка заголовка
    this->SetWindowText("Firewall");
    return 0;
}
```

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

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

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

    CString nf=dg.GetPathName();
    if(nf.GetLength() == 0)
    {
        AfxMessageBox("Це ім’я не існує");
        return;
    }
    CFile file;
    CFileException e;
    if( !file.Open( nf, CFile::modeCreate | CFile::modeWrite, &e ) )
    {
        AfxMessageBox("Error openning the file.");
        return;
    }
    PFFORWARD_ACTION action = pckFilter.GetDefaultAction();
```

```
     file.Write(&action, sizeof(PFFORWARD_ACTION));
     unsigned int i;
          for(i=0;i<doc->nRules;i++)
     {
          file.Write(&doc->rules[i], sizeof(RuleInfo));
     }
     file.Close();
}
```

Розглянемо підпрограму 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();
}
```

Введення правила фільтрації вручну відбувається наступним чином: 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. Розглянемо підпрограму додавання правил:

```
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::DRule(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;
}
```

Опишемо функції додавання фільтра в інтерфейс:

```
int CPacketFilter::AddF(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 = PfAddFsToInterface(interfaceHandle.hInterface,
                              1, &ipFlt, 0, NULL, NULL);
     if(direction == OUT_DIRECTION || direction == ANY_DIRECTION)
          errorCode = PfAddFsToInterface(interfaceHandle.hInterface,
                         0, NULL,  1, &ipFlt, NULL);
     else
          return -2;
```

```
    if(errorCode != NO_ERROR)
        return -1;
    return 0;
}
```

Була використана водоспадна (каскадна) модель життєвого циклу ПЗ (waterfall model) – послідовний метод розробки програмного забезпечення, названий так через діаграму схожу на водоспад.

Ця модель розробки запозичена з системної інженерії у виробництві та будівництві – областях, в яких зміни на пізніх етапах дуже дорогі, або неможливі. Наприклад, для створення складних інженерних конструкцій (споруд, літаків, мостів і т.п.). Зміни в проекті фундаменту будинку після того, як покладений дах коштують дуже дорого, тому перфекціонізм на початкових етапах проектування просто необхідний. Інженери, які починали займатись розробкою програмного забезпечення перейшовши з інших галузей, просто адаптували звичну модель, тому що на ранніх етапах розвитку комп'ютерної техніки не було методологій створених саме для програмування. Проте, схожі методології застосовуються для програмного забезпечення й далі, у випадках коли вимоги фіксовані, і вимагається висока якість та надійність, наприклад в системах для військових чи медичних потреб.

Перший формальний опис водоспадної моделі, після якої вона стала популярною був здійснений В. В. Ройсом у 1970. Попри те, що стаття містить переважно критику методу, на неї часто посилаються.

Переваги методу:

– Ніяких переробок.

– Гарна специфікація перетікає в гарну документацію.

– Зрозуміла модель.

– Розробники можуть мати низьку кваліфікацію.

Недоліки:

– Необхідний перфекціонізм на кожному етапі.

– Важко вносити зміни (якщо взагалі можливо).

– Надлишкове проектування.

– Поділ розробників на "perfect" та "code monkeys".

Модифікації. Через те що цей метод погано підходить для розробки саме ПЗ, частіше використовують його модифікації.

Найвідоміша модифікація – Sashimi. Названа так через японську страву сашимі (суші нарізане і сервіроване так, що складені рядочком шматочки накладаються один на одного). В моделі розробки Сашимі фази життєвого циклу йдуть одна за одною, але при цьому перекриваються одна з одною в часі.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних [2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій: Таблична заміна γ; Лінійне перетворення π; Байтова перестановка τ; Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ спеціалізованих внутрішніх міжсегментних мережних екранів ISFW яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Правила; Довідка.

– Вікна обрання групи.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

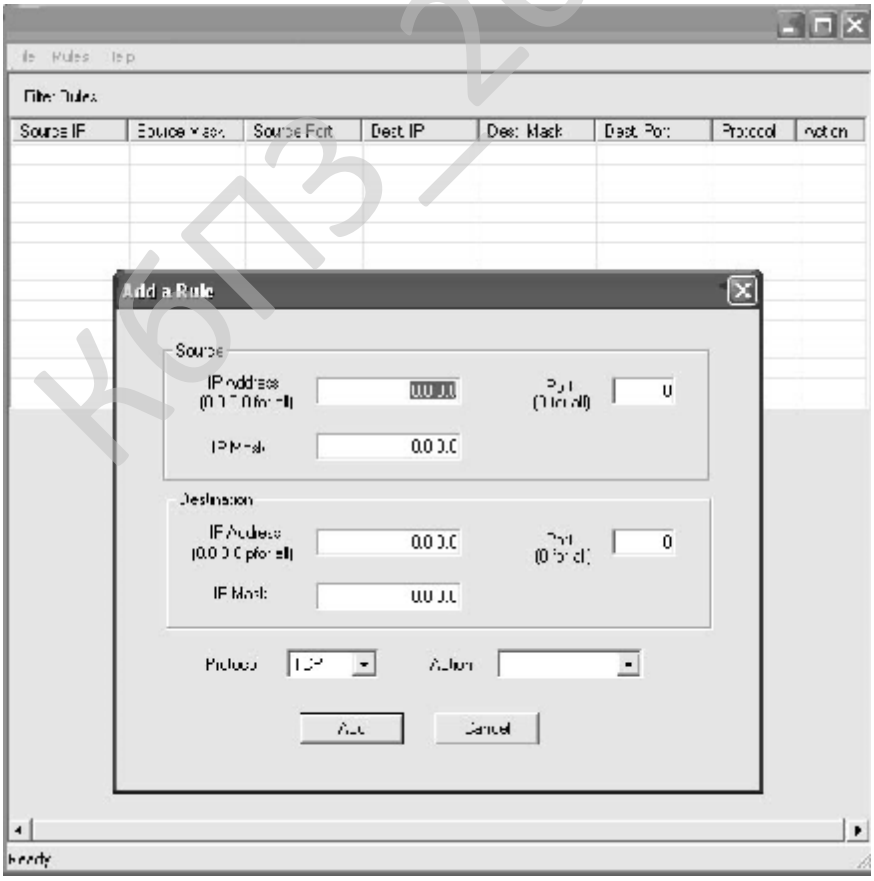

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

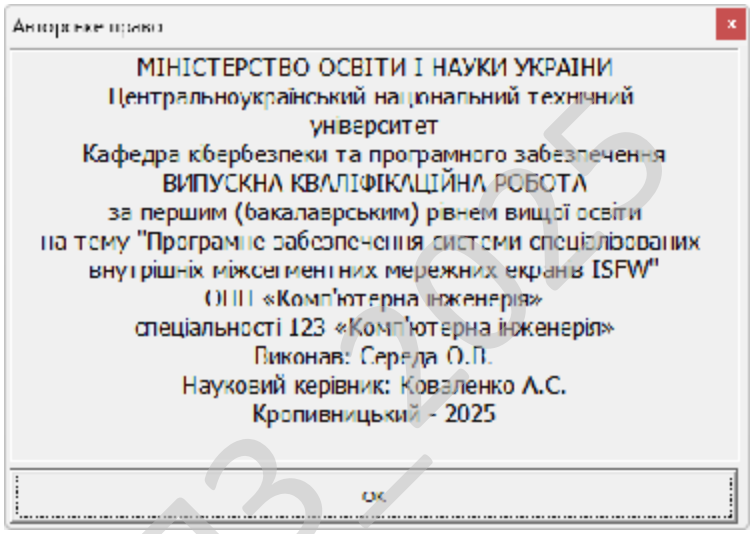

Рисунок 5.2 – Авторське право

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

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

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

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

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

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

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

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди.

Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

– Досліджена система спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

– На основі отриманих результатів досліджень створена програмна реалізація системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

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

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

Програма реалізована на мові високого рівня Visual C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW. 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 CRYPTON.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

7. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження

технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

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

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

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

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

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

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

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

15. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile*

Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

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

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

18. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

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

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

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

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

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

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

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

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

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

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

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

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

31. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

47. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. Коваленка О.В. – Кіровоград: КНТУ 2013. – 257с.

48. Смірнов О.А., Кавун С.В., Столбов В.Ф., Мелешко Є.В. Основи інформаційної безпеки. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. С.В. Кавуна. – Кіровоград: КНТУ 2012. – 442 с.

49. Смірнов О.А., Кузнецов О.О., Євсеєв С.П., Мелешко Є.В., Король О.Г. Методи та алгоритми симетричної криптографії. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102

«Комп’ютерна інженерія». За ред. О.О. Кузнецова. – Кіровоград: КНТУ 2012. – 315 с.

50. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп’ютерна інженерія». За ред. О.А. Смірнова. – Кіровоград: КНТУ 2012. – 250 с.

51. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп’ютерних мереж. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп’ютерна інженерія» та 8.0925 «Автоматизація й комп’ютерно-інтегровані технології». За ред. О.А. Смірнова. – Кіровоград: КНТУ 2012. – 454 с.

52. Смірнов О.А., Віхрова Л.Г., Осадчий С.І., Ковтун В.Ю., Мелешко Є.В. Основи захисту інформації. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп’ютерна інженерія» та 8.050201 «Системна інженерія». За ред. О.А. Смірнова. – Кіровоград: КНТУ 2011. – 322 с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 47-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 62 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Коваленко А.С.

Програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 42

Літера: РП

Кропивницький – 2025 року

Firewall_ISFW_AppDoc.cpp – формування правил

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

#include "Firewall_ISFW_AppDoc.h"

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

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

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

/////////////////////////////////////////////////////////////////////////////
// CFirewall_ISFW_AppDoc construction/destruction

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

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

	return TRUE;
}

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

/////////////////////////////////////////////////////////////////////////////
// CFirewall_ISFW_AppDoc diagnostics
//опис функцій обробки дебаг інформації
#ifdef _DEBUG
```

```
void CFirewall_ISFW_AppDoc::AssertValid() const
{
	CDocument::AssertValid();
}

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

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

Firewall_ISFW_App.cpp - головний файл програми

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

#include "MainFrm.h"
#include "Firewall_ISFW_AppDoc.h"
#include "Firewall_ISFW_AppView.h"

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

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

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

/////////////////////////////////////////////////////////////////////////////
CFirewall_ISFW_AppApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CFirewall_ISFW_AppApp initialization

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

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


      SetRegistryKey(_T("Firewall_ISFW_App"));

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

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

```
            IDR_MAINFRAME,
            RUNTIME_CLASS(CFirewall_ISFW_AppDoc),
            RUNTIME_CLASS(CMainFrame),
            RUNTIME_CLASS(CFirewall_ISFW_AppView));
      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 CFirewall_ISFW_AppApp::OnAppAbout()
{
      CAboutDlg aboutDlg;
      aboutDlg.DoModal();
}

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

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

DefaultActionDlg.cpp - Підключення основних оголошень діалогового вікна

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

Firewall_ISFW_AppView.cpp – Формування вікон

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

#include "Firewall_ISFW_AppDoc.h"
#include "Firewall_ISFW_AppView.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

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

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

	//}}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()

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

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

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

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


	return CFormView::PreCreateWindow(cs);
}

//Ініціалізація вікна
```

```
void CFirewall_ISFW_AppView::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);


}

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

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

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

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

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

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

```
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CFirewall_ISFW_AppView message handlers
//опис функції відновлення списку правил в інтерфейсі
void CFirewall_ISFW_AppView::UpdateList()
{
	CFirewall_ISFW_AppDoc *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 CFirewall_ISFW_AppView::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 "Firewall_ISFW_App.h"

#include "MainFrm.h"
#include "RuleDlg.h"
#include "DefaultActionDlg.h"
#include "Firewall_ISFW_AppDoc.h"
#include "Firewall_ISFW_AppView.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("Firewall_ISFW_");

      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()
{
	CFirewall_ISFW_AppDoc *doc = (CFirewall_ISFW_AppDoc *)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()
{
	CFirewall_ISFW_AppDoc *doc = (CFirewall_ISFW_AppDoc *)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
				CFirewall_ISFW_AppView *view = (CFirewall_ISFW_AppView
*)GetActiveView();

				view->UpdateList();
			}

		}
	}

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

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


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

		return;
	}

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

	CFirewall_ISFW_AppDoc *doc = (CFirewall_ISFW_AppDoc *)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;

	CFirewall_ISFW_AppDoc *doc = (CFirewall_ISFW_AppDoc *)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);
```

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

	view->UpdateList();
}

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

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

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

PacketFilter.cpp – Формування правил фільтру

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


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



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

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

      CPacketFilterInterface pckInt;
      unsigned long ip;

      while( pos != NULL )
      {

            interfacesTable.GetNextAssoc(pos, ip, pckInt);

            PfDeleteInterface(pckInt.hInterface);

      }

      interfacesTable.RemoveAll();
}


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

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

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

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

```
	}


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

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

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

	DWORD errorCode;

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

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

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


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

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

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

            interfacesTable[localInterfaceIp] = interfaceHandle;
      }



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


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

      DWORD errorCode;


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

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

      else
            return -2;

      if(errorCode != NO_ERROR)
            return -1;

      return 0;
}

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

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

```
	}


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

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

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

	DWORD errorCode;

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


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

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


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

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

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

```
	{
		return -1;
	}


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

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

	DWORD errorCode;

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

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

	else
		return -2;

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


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

{
	CPacketFilterInterface interfaceHandle;

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

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

		interfacesTable[CharToIp(localInterfaceIp)] = interfaceHandle;
	}


	DWORD errorCode;

	// Додаю глобальний фільтр
```

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

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}


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

{
	CPacketFilterInterface interfaceHandle;

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


	DWORD errorCode;

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

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}

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

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

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

}

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

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

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

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

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


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

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

                hInterface = NULL;

                return -2;
        }

        return 0;
}

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

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

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

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

                auxCad = strchr(auxCad, '.');

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

                auxCad++;
        }


        return lIp;
}
```

ResizeDlg.cpp – Зміна розмірів вікон

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

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

#ifndef OBM_SIZE
#define OBM_SIZE 32766
#endif

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

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

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

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

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

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

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

		st = m_stxRight;
	}

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

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

			bUpdateRect = 0;
		}

		break;

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

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

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

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

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

	default:
		break;
	}

	// зміна згори
	st = CST_NONE;
```

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

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

		st = m_styBottom;
	}

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

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

			bUpdateRect = 0;
		}

		break;

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

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

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

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

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

	default:
		break;
	}

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

```
            curRect = m_wRect;
      }

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

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

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

            if (bUpdateRect) {
                  m_wRect = ctrlRect;
            }
      }

      return hDWP;
}

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

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

CResizeDlg::~CResizeDlg()
{
      }

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

      CRect cltRect;
      CBitmap cBmpSize;
      BITMAP      Bitmap;

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

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

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

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

      cRgn.Detach();
#endif

      cBmpSize.Detach();

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

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

      return bret;
}

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

      m_nDelaySide = nSide;
}

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

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

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

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

            HDWP   hDWP;
            int    sizeType = WST_NONE;

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

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

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

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

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

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

                  ::EndDeferWindowPos(hDWP);
            }

            m_cltRect = cltRect;
      }

      m_nDelaySide = 0;
}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

      m_Items.push_back(item);

}

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

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

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

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

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

				return 0;
			}
		}
	}

	return -1;
}
```

RuleDlg.cpp – Установка умов

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