МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Білоус Ігор Володимирович

Програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF

Спеціальність: 125 «Кібербезпека»

Освітній ступінь: бакалавр

Науковий керівник:

Смірнов Сергій Анатолійович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Білоусу Ігорю Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF*

керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 185-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF*

4. 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

5. 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*

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Білоус І.В. Програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

Метою розробки є програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

Результат роботи – програмна реалізація системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

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

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

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Visual C++.

Ключові слова: кібербезпека, WAF

ABSTRACT

Bilous I.V. Cybersecurity software to protect web applications based on WAF technology. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

This undergraduate qualification has developed software designed for a cybersecurity system to protect web applications based on WAF technology.

The purpose of the development is cybersecurity system software to protect web applications based on WAF technology.

The result is a software implementation of a cybersecurity system to protect web applications based on WAF technology.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC PC with Windows XP / Vista / 7/8/10.

The program is developed in Visual C ++.

Keywords: cybersecurity, WAF

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
WAF	–	Web Application Firewall – засоби фільтрації трафіку прикладного рівня, спеціально орієнтовані на веб-застосунки

ВСТУП

Актуальність теми. Web Application Firewall (скорочено – WAF) – засоби фільтрації трафіку прикладного рівня, спеціально орієнтовані на веб-застосунки. Застосування Web Application Firewall традиційно вважається найбільш ефективним підходом до захисту веб-ресурсів. WAF може бути реалізований як хмарний сервіс, агент на веб-сервері або спеціалізоване залізне або віртуальний пристрій.

Традиційно вважається, що прикладний рівень – це останній рівень моделі й вище його розташовуються тільки дані кінцевих застосунків, які не можуть бути формалізовані й згруповані. Однак з розвитком стандартів представлення інформації прикладними сервісами вже можна говорити про те, що, частково, дані, якими оперують визначені групи застосунків, добре формалізуються, і правила їх представлення, по суті, є якимись про прієтарними протоколами або, спрощено говорячи, закономірностями.

Таким чином, можна говорити про появу нового рівня міжмережевої взаємодії, який схований для класичних міжмережевих екранів прикладного рівня. Новий клас пристроїв – Web Application Firewall – характеризується здатністю розуміти групи протоколів і залежностей, властивих для веб-застосунків, які будуються над прикладними протоколами http/HTTPS.

Класичне розміщення WAF у мережі – у режимі зворотного проксі-сервера, перед, що захищаються веб-серверами. Залежно від виробника можуть підтримуватися й інші режими роботи – наприклад, прозорий проксі-сервер, міст або навіть пасивний режим, коли продукт працює з реплікацією трафіку.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

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

– Огляд існуючих систем кібербезпеки для захисту веб-застосунків на базі технології WAF.

– Дослідження системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

– Програмна реалізація системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для захисту веб-застосунків на базі технології WAF.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Після установки WAF і пуску продуктивного трафіку відразу ж починає роботу основний компонент захисту – машинне навчання, у ході якого складається еталонна модель комунікації з об'єктом захисту, і в такий спосіб формується «білий» список припустимих ідентифікаторів доступу. На даний момент у веб-застосунках використовуються три типи ідентифікатора доступу: Http-параметри (у виставах типу: Raw, XML, JSON), ідентифікатор ресурсу (URL, URN), ідентифікатор сесії (cookie). Завдання WAF полягає у визначенні припустимих значень ідентифікаторів для веб-застосунка. З певних значень згодом буде полягати еталонна (позитивна) модель. Включення конкретних значень ідентифікатора в модель здійснюється на основі застосування математико-статистичного алгоритму, який за допомогою вибірки продуктивного трафіку оцінює ці значення як припустимі.

Коли всі ресурси веб-застосунка додані в позитивну модель, адміністратор системи повинен переконатися у відсутності значимого кількості неправильно-позитивних спрацьовувань і перемкнути систему в режим блокування.

1.2 Область застосування

Крім машинного навчання в набір функцій WAF звичайно входять наступні типові механізми захисту:

- валідація протоколу;
- сигнатурний аналіз;
- захист від ін'єкцій і XSS (найчастіше пропрієтарна);
- можливість створення власних правил захисту;

- інтеграція з репутаційними й фрод-сервісами;
- інтеграція з іншими пристроями в ландшафті ІБ компанії.

Пріоритетом для виробника WAF є сфокусованість власних дослідних центрів на генерації відновлень політик безпеки для своїх пристроїв з урахуванням актуальних погроз веб-застосункам. Так з'являються, наприклад, сигнатури атак, властиві для конкретних веб-фреймворків і систем контролю контенту або пропрієтарні механізми захисту від XSS і SQL-ін'єкцій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Щодня кіберзлочинці зламують майже 50 000 сайтів. От чому для захисту від цих численних спроб злому веб-сервера потребують спеціалізованого програмного інструментарію (брандмауерові для веб-застосунків), такому як ModSecurity, NAXSI або інших подібних їх по функціональних можливостях.

Розглянемо переваги й недоліки NAXSI і ModSecurity, популярних WAF (Web Application Firewall, міжмережевий екран для веб-застосунків) з відкритим вихідним кодом. І хоча обоє цих програмних продукти безкоштовні, ваш вибір між NAXSI і Modsecurit багато в чому буде продиктований конкретною конфігурацією вашого сервера.

NAXSI

Nginx Anti-XSS & SQL Injection (або просто NAXSI) – це брандмауер для веб-застосунків, спеціально розроблений для серверів Nginx.

NAXSI допомагає боротися зі зловмисниками, які додають уразливі скрипти на веб-сайт. Крім того, він дозволяє уникнути атак впровадження коду (code injections) у базу даних, використовувану веб-сайтами.

А тепер давайте більш докладно розглянемо основні переваги й недоліки NAXSI.

Переваги NAXSI

Основні переваги NAXSI містять у собі.

Простий набір правил

NAXSI захищає веб-сайти за допомогою простого набору правил, використовуючи засновану на оцінці систему захисту. Так, ґрунтуючись на цих твердих правилах, кожному одержуваному URL-запиту виставляється оцінка.

Якщо ця оцінка перевищує деяке граничне значення, задане в конфігурації, NAXSI автоматично блокує цей запит до веб-сайту. Приміром, коли URL-запит містить потенційно шкідливі символи, такі як «<», «/» (слеш), «drop», а також інші «небезпечні» символи й ключові слова-запити SQL, те це автоматично підвищує значення оцінки. І виконання таких URL-запитів буде заблоковано на сервері.

Підтримка білих списків

Також сильною стороною NAXSI є підтримка створення набору правил так званого «білого» списку. Ці правила визначають, які шаблони з потенційно шкідливими паттернами й для яких застосунків є прийнятними, і, відповідно, NAXSI не буде блокувати їхнє виконання. Щоб спростити адміністраторам завдання по складанню набору правил для «білого» списку, NAXSI поставляється із програмним інструментарієм за назвою Nxtool. Цей інструмент здатний автоматично вивчати трафік веб-сайту, а також створювати й доповнювати «білий» список. Слід зазначити, що цей програмний інструментарій здатний ураховувати активність ваших користувачів. Так, якщо більш 20 % ваших користувачів постійно використовують ідентичний «небезпечний» метод для своїх запитів на веб-сайті, то він буде зареєстрований як дозволений. І всі подібні запити будуть безперешкодно проходити через веб-сервер.

Стійкість до методів обходу WAF

Під будь-які навіть самі строгі правила міжмережевих екранів для веб-застосунків хакери знаходять альтернативні способи їх обходу. Але NAXSI прекрасно справляється з нейтралізацією можливих методів обходу WAF, таких як кодування URL, об'єднання рядків у запиті і т.д.

Швидке й легке адміністрування

NAXSI не буде «з'їдати» більшу частину ваших серверних ресурсів. Крім того, він не вимагає проведення періодичних і частих відновлень, чому вигідно відрізняється від також обговорюваного в рамках даної статті міжмережевого

екрана для веб-застосунків ModSecurity. Відразу після установки NAXSI буде працювати безупинно й надійно, без яких-небудь періодів вимушеного простою.

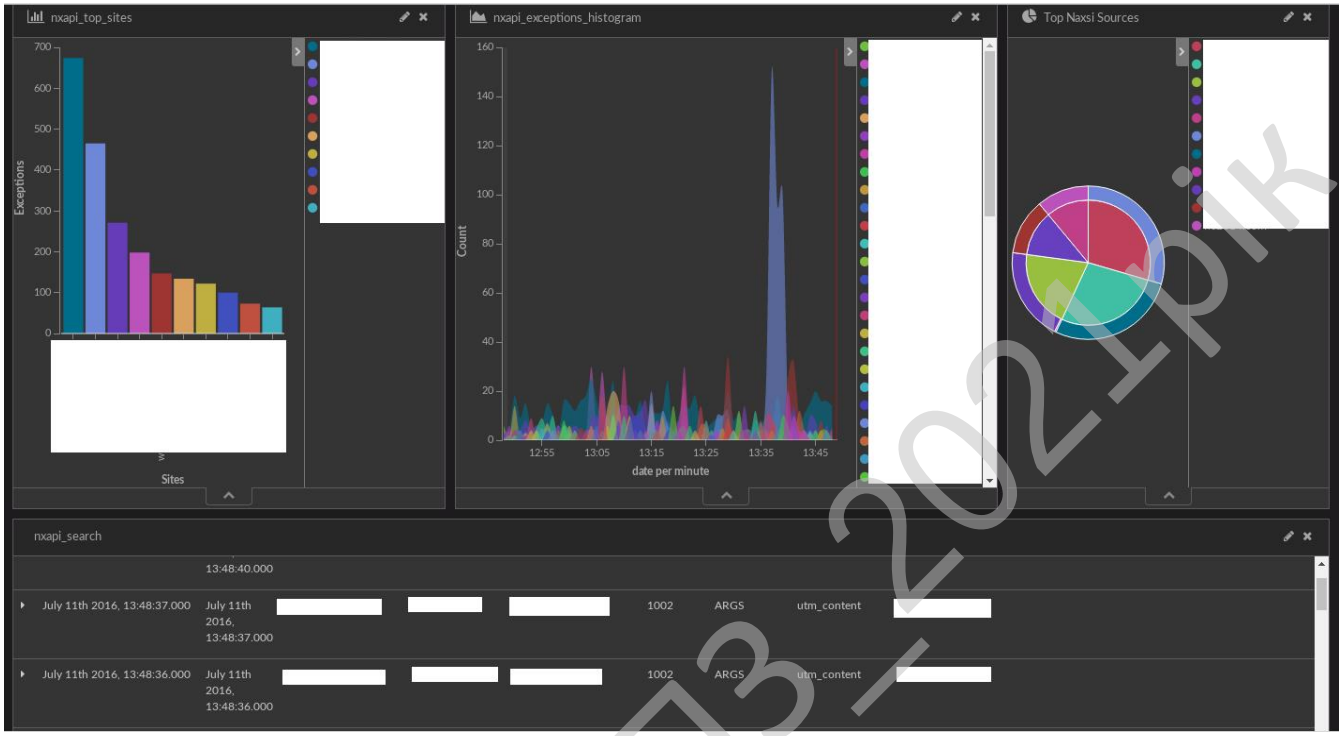

Рисунок 2.1 – Інтерфейс користувача NAXSI

Недоліки NAXSI

Хоча в NAXSI є багато переваг, у нього також є й свої недоліки:

Потрібен запуск режиму, що навчається, для кожного відновлення веб-застосунка

NAXSI має два режими роботи: робочий (Live) і навчальний (Learning). Саме в режимі навчання NAXSI не блокує запити, а вивчає поведінка користувачів, автоматично створюючи правила «білого» списку. Тому щораз, коли в коді вашого веб-сайту відбувається відновлення, вам потрібно буде запускати NAXSI у режимі навчання, щоб внести корективи в розв'язні правила «білого» списку й уникнути блокування легітимного трафіку. Це створює визначені застосункові труднощі, особливо якщо на сайті часто відбуваються модифікації коду.

Залежність від типу веб-сервера

До істотного недоліку NAXSI можна віднести те, що він працює тільки із системами, реалізованими на базі Nginx. Тобто він не підходить для використання з Apache або IIS, що накладає серйозні обмеження на параметри використовуваного веб-сервера. Проте, NAXSI прекрасно працює в застосунках на основі Docker. Тому NAXSI можна настроїти для таких популярних застосунків, як owncloud, Elasticsearch і т.д..

ModSecurity

ModSecurity – це один із самих популярних брандмауерів для веб-застосунків, який підтримує такі веб-сервери, такі як Apache, IIS, Nginx і т.д. Він працює з бібліотекою шкідливих шаблонів, відомих також як сигнатури. Коли URL-запит відповідає який-небудь із сигнатур, він блокуються. Хоча ModSecurity і є відмінним програмним інструментарієм для виявлення атак XSS (Cross-Site Scripting, міжсайтовий скриптинг), троянських атак і т.д., він також має свої переваги й недоліки.

Переваги ModSecurity

ModSecurity має такі переваги, як:

Блокування стандартних атак

ModSecurity поставляється з набором правил Core Rule, який ураховує практично всі відомі атаки в Інтернеті. Тому вам не потрібно буде самостійно прописувати правила для блокування вже відомих уразливостей застосунків.

Підтримка віртуального патчинга (Virtual patching)

Також ModSecurity надає можливість Virtual patching. Опція «Віртуальний патчинг» допоможе захистити сервер від атак, що використовують недавно виявлені уразливості. Інакше кажучи, як тільки з'являється нова атака на основі уразливості Wordpress або PHP, вам просто потрібно буде поставити новий патч застосунка ModSecurity. Це нівелює можливість проведення подальших атак на ваші сайти за допомогою цієї уразливості. Таким чином, ви одержите більше часу для відновлення кожного вразливого сайту, створеного на Wordpress або PHP.

Просунуте налаштування

Крім того, ModSecurity дозволяє писати конкретні правила для застосунків, розміщених на сервері. Ця відкриває широкі можливості для використання користувацьких модифікацій, коли у вас їсти необхідність вибудувати більш надійний захист від певних типів атак, які найбільш критичні для ваших веб-застосунків.

Недоліки ModSecurity

Знову ж, в ModSecurity є й свої недоліки:

Складне адміністрування великого набору правил

Практично всі правила ModSecurity ґрунтуються на регулярних вираженнях, адміністрування яких саме по собі є досить складним завданням. Крім того, чим більше правил, тем складніше ставати грамотно відфільтрувати «гарний» трафік від «поганого». Тим, хто працює з ModSecurity, доводитися періодично зустрічатися зі скаргами від клієнтів про те, що, крім блокування шкідливих атак, цей брандмауер для веб-застосунків часто блокує й легітимний трафік.

Ресурсоємність

Коли настроєне занадто багато правил, ModSecurity починає використовувати занадто багато ресурсів. Тому, чим більше веб-сайтів запущене на сервері, тим більше буде витрачатися ресурсів на їхній захист, і, відповідно, тем вище буде навантаження на сервер. Тому ваш вибір на користь ModSecurity також багато в чому буде залежати від специфіки ваших завдань і характеристик самого сервера.

При грамотній реалізації й ретельному налаштуванню брандмауери для веб-застосунків, такі як NA XSI і ModSecurity допоможуть вам захистити ваші сайти й уникнути зломів вашого веб-сервера. Сподіваємося, що наш порівняльний аналіз допоможе вам зробити правильний вибір.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою

об’єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який застосуноквзаємодіє 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 кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

В процесі розробки кваліфікаційної бакалаврської роботи 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 кібербезпеки, 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

Споконвічно термін firewall (брандмауер, екран) позначав мережний фільтр, який ставиться між довіреною внутрішньою мережею й зовнішнім Інтернетом (звідси прикметник «міжмережевий»). Цей фільтр був покликано блокувати підозрілі мережні пакети на основі критеріїв низьких рівнів моделі OSI: на мережному й канальному рівнях. Іншими словами, фільтр ураховував тільки IP адреси джерела й призначення, прапор фрагментації, номера портів.

Надалі можливості розширилися до шлюзів сеансового рівня, або фільтрів контролю стану каналу (stateful firewall). Ці міжмережеві екрани другого покоління підвищили якість і продуктивність фільтрації за рахунок контролю приналежності пакетів до активних TCP-Сесіям.

На жаль, подібна система захисту практично пошук проти сучасних кібер-погроз, де більш 80% атак використовують уразливості застосунків, а не уразливості мережної архітектури й сервісів. Гірше того: блокування певних портів, адрес або протоколів (основний спосіб роботи міжмережевих екранів) може вирубати цілком легітимні застосунки. Це значить, що захисна система повинна проводити більш глибокий аналіз змісту пакетів – тобто краще «розуміти» роботу застосунків.

Системи виявлення/запобігання вторгнень (IDS/IPS)

Наступним поколінням захисних екранів стали системи виявлення й запобігання вторгнень (IDS/IPS). Вони здатні вивчати в TCP-пакетах поле даних і здійснювати інспекцію на рівні застосунка по певних сигнатурах. Системи IDS пристосовані до того, щоб виявляти атаки не тільки зовні, але й усередині мережі за рахунок прослуховування SPAN-порту комутатора.

Для вдосконалювання захисних механізмів в IDS/IPS стали застосовуватися декодери (розбір полів TCP-пакета) і препроцесори (розбір частин протоколу рівня застосунка, наприклад, HTTP). Застосування препроцесорів в IPS Snort дозволило суттєво поліпшити функціональність периметрового захисту в порівнянні з пакетним фільтром, навіть якщо останній перевіряє пакети на рівні застосунків (iptables з модулем layer7).

Однак при цьому зберігся основний недолік пакетного фільтра: перевірка здійснюється попакетно, без обліку сесій, cookies і всієї іншої логіки роботи застосунка.

Паралельно для боротьби з поширенням вірусів з'являються проксі-сервери, а для застосунку завдань балансування навантаження – зворотні проксі-сервери. Вони відрізняються технічно, але головне, що й ті, і інші повноцінно працюють на рівні застосунка: відкривається два TCP з'єднання від проксі до клієнта й від проксі до сервера, аналіз трафіку ведеться винятково на рівні застосунка.

NGFW/UTM

Наступним кроком еволюції систем виявлення вторгнень стала поява пристроїв класу UTM (unified threat management, система єдиного керування погрозами) і NGFW (next generation firewall, екрани нового покоління).

Системи UTM відрізняються від NGFW лише маркетингом, при цьому їх функціонал практично збігається. Обоє класу програмних продуктів з'явилися спробою об'єднати функції різних продуктів (антивірус, IDS/IPS, пакетний фільтр, VPN-шлюз, маршрутизатор, балансувальник і ін.) в одному пристрої. У теж час, виявлення атак у пристроях UTM/NGFW нерідко здійснюється на старій технологічній базі, за допомогою згаданих вище препроцесорів.

Специфіка веб-застосунків припускає, що за один сеанс роботи користувача з веб-сервером може здійснюватися велика кількість різних TCP-з'єднань, які відкриваються з різних адрес, але мають один (можливо динамічний)

ідентифікатор сесії. Це приводить до того, що для акуратного захисту веб-трафіку необхідна платформа на основі повнофункціонального реверс-проксі-сервера.

Але різниця в технологічній платформі – не єдине, що відрізняє захист веб-застосунків.

WAF

Якщо говорити зовсім просто, то веб-застосунки відрізняються від звичайних застосунків двома речами: величезною різноманітністю й значної інтерактивності. Це створює цілий ряд нових погроз, з якими традиційні міжмережеві екрани не справляються: за нашими оцінками, в 2021 році 60% атак на корпоративні мережі здійснювалися через веб-застосунки, незважаючи на наявність традиційних захисних засобів.

Саме тут вступає в справу Web Application Firewall (WAF), захисний екран для застосунків, що здійснюють передачу даних через HTTP і HTTPS.

Multiprotocol Security

Будучи вузькоспеціалізованим засобом, WAF не захищає від проблем протоколів, відмінних від HTTP/HTTPS. Але в будь -якої медалі дві сторони. Різноманіття способів обміну даними поверх протоколу HTTP настільки велике, що орієнтуватися в ньому може тільки спеціалізований засіб. Лише для прикладу: десь змінні й значення передаються у форматі example.com/animals?dogs=32&cats=23, десь у форматі example.com/animals/dogs/32/cats/23, десь передача параметрів застосунка здійснюється в Cookie, а десь – у параметрах заголовка HTTP.

Крім того, просунуті моделі WAF можуть аналізувати XML, JSON і інші протоколи сучасних порталів і мобільних застосунків. Зокрема, це дозволяє протидіяти більшості методів обходу захисного екрана (HPC, HPP, Verb Tampering і ін).

IP Reputation

Технологія IP-reputation опирається на зовнішні чорні й білі списки ресурсів, і однаково доступна будь-яким периметровим засобам захисту. Але

цінність цього методу трохи перебільшена. Так, у практиці наших фахівців траплялися ситуації, коли великі новинні агентства в силу своїх уразливостей місяцями роздавали malware користувачам, і при цьому не попадали в чорні списки. На жаль, вектора проникнення шкідливого ПЗ дуже різноманітні, і в наші дні джерелом зараження для користувачів можуть бути навіть сайти органів державної влади. А можлива й зворотна проблема, коли по IP блокуються «невинні» ресурси.

Сигнатури атак

Сигнатурний підхід до виявлення атак застосовується всюди, але тільки грамотний препроцесінг трафіку, доступний для WAF, може забезпечити адекватне застосування сигнатур. Недоліки препроцесінга приводять до зайвої «монструозності» сигнатур атак: адміністратори не можуть розібратися в найскладніших регулярних вираженнях, увесь зміст яких у тому, що їх автори, наприклад, усього лише намагалися враховувати можливість передачі параметра як відкритим текстом, так і у формі шістнадцяткового коду з відсотком.

Автоматичне навчання й поведінковий аналіз

Для атак на веб-застосунки зловмисники активно використовують уразливості нульового дня (0-day), що робить пошуками сигнатурні методи аналізу. Замість цього потрібно аналізувати мережний трафік і системні журнали для створення моделі нормального функціонування застосунка, і на основі цієї моделі виявляти аномальну поведінку системи. WAF у силу своєї архітектури може розібрати весь сеанс зв'язку користувача, і тому здатний на більш поглиблений поведінковий аналіз, чому NGFW. Зокрема, це дозволяє виявляти атаки з використанням автоматичних засобів (сканування, добір паролів, DDoS, фрод, залучення в ботнети).

Однак у більшості випадків навчання поведінкової моделі полягає в тому, що оператори звідкись беруть «білий трафік» і «згодовують» його засоби захисту. Але після здачі в експлуатацію поведінка користувачів може мінятися: програмісти дописують інтерфейс по скоректованому технічному завданню,

дизайнери «додають краси», рекламні кампанії міняють напрямок уваги. Не можна раз і назавжди скласти схему поведінки «правильного» відвідувача. При цьому навчатися на реальному, «сірому» трафікі можуть тільки одиниці програмних продуктів – і це тільки WAF'и.

Захист користувачів

Периметрове устаткування, обговорюване в справжній статті, призначене для захисту серверів із веб-застосунками. Однак існує клас атак (наприклад, CSRF) спрямованих на клієнта веб-застосунка. Оскільки трафік атаки не проходить через захисний периметр, на перший погляд захистити користувача виявляється неможливо.

Розглянемо наступний сценарій атаки: користувач заходить на сайт банку, проходить там автентифікацію, і після цього в іншій вкладці браузера відкриває заражений ресурс. Javascript, що завантажився в іншому вікні, може зробити запит на переклад грошей потай від користувача, а браузер при цьому підставить усі необхідні автентифікаційні параметри для здійснення фінансової транзакції, тому що сеанс зв'язки користувача з банком ще не минувся. В описаній ситуації в наявності слабості в алгоритмі автентифікації в банківському ПЗ. Якби для кожної форми, що втримується на сторінці сайту, генерувався унікальний токен, проблеми б не було.

На жаль, розроблювачі ПЗ практично ніколи так не роблять. Однак деякі WAF можуть самостійно впроваджувати подібний захист у веб-форми й захищати, таким чином, клієнта – а вірніше, його запити, дані, URL і cookie-файли.

Взаємодія зі сканерами уразливостей

На периметрове устаткування покладає не тільки завдання захисту веб-застосунків, але й завдання моніторингу атак. При цьому грамотний моніторинг заснований на розумінні слабостей, що захищається ПЗ, що дозволяє відсіяти неактуальні спроби атак і виділити тільки ті, які стосуються реальних уразливостей, наявних у системі.

Кращі зразки WAF мають у своєму розпорядженні інтегровані сканери уразливостей, що працюють у режимі чорного ящика, або динамічного аналізу (DAST). Такий сканер може використовуватися в режимі реального часу для швидкої перевірки тих уразливостей, які «прощупують» зловмисники.

3.2 Розробка структурної схеми

Віртуальний патчинг

Навіть відомі уразливості неможливо усунути відразу: виправлення коду вимагає засобів і часу, а найчастіше й зупинки важливих бізнес-процесів; іноді у випадку використання стороннього ПЗ виправлення неможливо взагалі. Для відбивання таких «часток» погроз у системах IDS/IPS, а в спадщину в UTM/NGFW, застосовуються користувацькі сигнатури. Але проблема в тому, що написання такої сигнатури вимагає від користувача глибокого розуміння механізму атаки. А якщо ні, то користувацька сигнатура може не тільки «пропустити» погрозу, але й породити велика кількість неправильних спрацьовувань.

У найбільш сучасних WAF використовується автоматизований підхід до віртуального патчингу. Для цього використовується аналізатор вихідних кодів застосунка (SAST, IAST), який не просто показує у звіті рядка вразливого коду, але відразу генерує експлойт, тобто виклик з конкретними значеннями для експлуатації виявленої уразливості. Ці експлойти передаються в WAF для автоматичного створення віртуальних патчів, які забезпечують негайне «закриття проломи» ще до виправлення коду.

Кореляції й ланцюжки атак

Традиційний міжмережевий екран дає тисячі спрацьовувань на підозрілі події, у яких необхідно розбиратися вручну, щоб виявити реальну погрозу. Як відзначає Gartner, вендори систем IPS взагалі воліють відключити більшість сигнатур веб-застосунків, щоб знизити ризик виникнення таких проблем.

Сучасний WAF може групувати подібні спрацьовування й виявляти ланцюжок розвитку атаки – від розвідки до крадіжки важливих даних або установки закладок. У результаті замість списку з тисяч підозрілих подій ІБ-фахівці одержують кілька десятків дійсно важливих повідомлень.

Зрозуміло, що застосунки різних вендорів WAF завжди будуть відрізнятися набором функцій. Тому перелічимо тут лише найбільш відомі застосункові фічі сучасних захисних екранів рівня застосунків:

– Робота з SSL-трафіком як застосунковий рівень захисту. На думку аналітиків Gartner, можливість перевірки зашифрованого трафіку є одним з найважливіших відмінностей WAF від звичайних міжмережевих екранів і IPS.

– Служби перевірки дійсності: WAF є єдиною крапкою входу для веб-застосунків або діє в якості брокера перевірки дійсності для застарілих застосунків, механізм автентифікації яких порушений.

– Підтримка політики безпеки контенту (Content Security Policy, CSP) для захисту від XSS і інших атак.

Крім того, експерти Positive Technologies прогнозують наступні перспективні напрямки розвитку міжмережевих екранів рівня застосунків у найближчому майбутньому:

– Нові алгоритми поведінкового аналізу, які дозволять краще розрізняти користувачів для виявлення ботів і зловмисників (UBA).

– Захист застосунків, побудованих на базі HTML5, протоколів на базі XML, з використанням нереляційних БД (NoSQL).

– WAF, орієнтовані на конкретні типи застосунків: банківські (ДБО), системи ERP, застосунка телеком, мас-медіа й ін.

Наостанку варто помітити, що дана стаття присвячена тільки технологічним особливостям WAF. Але на практиці варто враховувати й організаційні – наприклад, відповідність стандартам безпеки або можливість інтеграції WAF з іншими засобами безпеки (антивіруси, DLP і ін.). Моделі

розгортання також можуть бути різними: це може бути апаратне, програмне або віртуальний розв'язок, або хмарний сервіс у моделях Saas, VAS і MSS.

WAF (Web Application Firewall) – міжмережевий екран для веб-застосунків. Це інструмент для фільтрації трафіку, що працює на прикладному рівні й захищаючий веб-застосунка методом аналізу трафіку HTTP/HTTPS і семантики XML/SOAP. WAF може встановлюватися на фізичний або віртуальний сервер і виявляє найрізноманітніші види атак.

Діє міжмережевий екран як проксі-сервер, але через можливість вивчати HTTPS-трафік методом перевірки сертифіката конкретного сервера WAF розрахований на виконання застосункових операцій: балансування навантаження на сервер, термінацію трафіку SSL і т.д. WAF може працювати із кластеризацією і акселерацією застосунків.

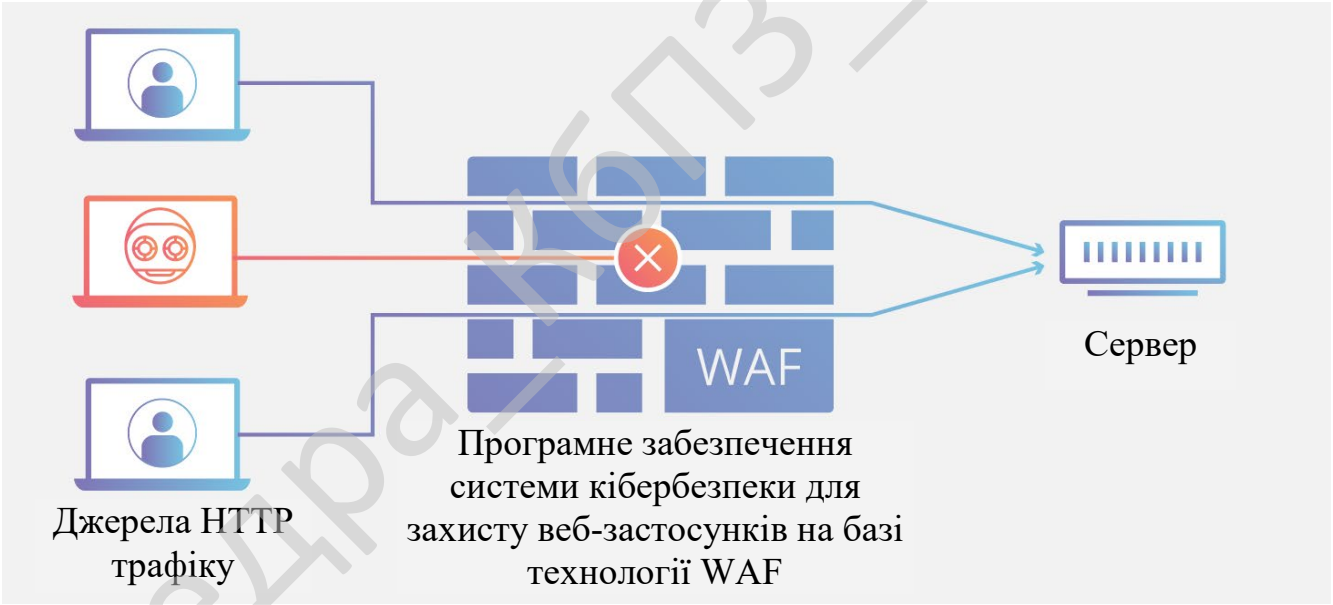

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Тиск на сучасному ринку конкуренції в різних сферах діяльності створило дуже сприйнятливу до навколишнього середовища web-інфраструктуру. Сучасні архітектури розгортання web-застосунків досить складні й вимагають інтеграції багатьох гетерогенних технологій, створюючи потенціал для численних уразливостей. Прискорені цикли розробки й постійні відновлення web-застосунків ще більше збільшують ситуацію. У таких умовах бізнес-ризики компаній занадто великі, щоб ігнорувати дану проблему, тому що уразливості в web-застосунках піддають критично важливі бізнес-операції й конфіденційні дані небезпеки. Значні фінансові втрати можуть виникнути в результаті непередбачених затримок у бізнес-процесах, крадіжки інтелектуальної власності, і втрати довіри клієнтів, а також репутації бренда. У багатьох випадках, безпека web-застосунків також є юридичною вимогою, наприклад, таким як дотримання Payment Card Industry Data Security (PCI DSS) стандарту безпеки даних.

Сучасні тенденції розвитку web-сервісів указують на відсутність єдиних стандартів безпечного програмування web-застосунків, що приводить до помилок у розробці ПЗ й появі серйозних уразливостей в web-сервісах застосунки, що використовують це. Положення ускладнюється тим, що вразливе web-застосунок може бути легко скомпрометоване без використання спеціалізованих засобів, тільки за допомогою браузера. Захист уразливих web-застосунків може здійснюватися або шляхом усунення уразливостей в web-застосунку або з використанням спеціалізованих засобів захисту web-застосунків – Web Application Firewall (WAF).Сьогодні розміщення традиційних брандмауерів, Next Generation Firewalls (NGFW) або Intrusion Prevention Systems (IPS) по периметру мережі або, принаймні, у якості шлюзів для довірених сегментів мережі, є недостатнім чинником для забезпечення повноцінного захисту мережної інфраструктури. Атаки на web-ресурси, які маніпулюють програмним забезпеченням для досягнення шкідливих цілей, як правило, проходять

одночасно із сесіями легітимних користувачів і, по більшому ступеню використовують стандартні HTTP (80) і HTTPS (443) порти. Блокування всього трафіку на рівні порту не є варіантом виходу з даного положення, тому що доступ до web-застосункам буде повністю закритий ззовні або зсередини. Саме така дилема мережної безпеки є відмінною можливістю для пошуку хакерами уразливостей на рівні web-застосунків.

Web Application Firewall (WAF, міжмережевий екран для веб-застосунків) являє собою пристрій безпеки (апаратне або віртуальне), основним завданням якого є захист web-порталів і web-застосунків шляхом перевірки XML/SOAP семантики потокового трафіку, а також перевірки HTTP/HTTPS трафіку з метою виявлення різних атак на рівні застосунків. Міжмережевий екран для веб-застосунків діє як проксі-сервер, але завдяки здатності аналізувати HTTPS трафік (шляхом імпорту сертифіката безпеки цільового сервера), також може виконувати й інші функції, такі як термінація SSL трафіку й балансування навантаження сервера. Крім того WAF підтримує кластеризацію, а також виконує акселерацію web-застосунків.

Web Application Firewall (WAF) підтримує два основні режими розгортання:

- Gateway (bridge mode, transparent proxy, reverse proxy);
- Monitor (режим мережного моніторингу через SPAN порт).

Міжмережевий екран для веб-застосунків (WAF) функціонує на основі двох загальноприйнятих моделей безпеки:

– Negative – негативна модель або чорний список (заперечує те, що є свідомо встановленим). Для надання базового захисту, аналогічного IPS, але з більш високим рівнем оцінки безпеки застосунків, WAF може використовувати як загальновідомі сигнатури для запобігання найпоширеніших атак, так і специфічні сигнатури для атак, що використовують уразливості окремих web-застосунків. Наприклад: заперечувати певний потенційно небезпечний HTTP запит GET і дозволити все інше;

– Positive – позитивна модель або білий список (дозволяє тільки те, що є свідомо встановленим). Для поліпшеного захисту, на застосунокдо сигнатур, використовується ще один тип логіки: правила, які визначають, що явно дозволене. Наприклад: дозволити тільки HTTP GET запити для певного URL і заборонити все інше.

Ключові можливості WAF:

– Підтримка всіх застосовних до web-застосунків PCI DSS Requirements, пов'язаних з компонентами системи в середовищі обробки даних по платіжних картах.

– Оперативна реакція (визначається активною політикою й/або набором правил) на погрози й атаки, визначені, як мінімум, в OWASP Top 10.

– Перевірка вхідного HTTP/HTTPS трафіку й запитів до web-застосунків і вживання захисних заходів на основі активних політик і правил (дозволити, блокувати, попередити).

– Підтримка й дотримання коректного функціонування позитивної й негативної моделі безпеки.

– Вивчення й перевірка web-контенту, створеного за допомогою Hypertext Markup Language (HTML), Dynamic HTML (DHTML), Cascading Style Sheets (CSS) і основних протоколів доставки web-контенту, таких як Hypertext Transport Protocol (HTTP) і Transport Protocol Hypertext over SSL (HTTPS).

– Запобігання витоку даних – перевірка вихідного HTTP/HTTPS трафіку й запитів до web -застосункам і вживання захисних заходів на основі активних політик і правил, а також своєчасний запис що відбувся подій у журнал подій.

– Аналіз повідомлень web-сервісів, особливо публічних. Як правило, включає себе перевірку Simple Object Access Protocol (SOAP) і extensible Markup Language (XML), а також Remote Procedure Call (RPC) орієнтовані моделі взаємодії з web-сервісами, засновані на базі HTTP.

– Перевірка будь-якого протоколу або конструкції даних (пропрієтарних або стандартизованих), які використовуються для передачі даних в/з web-застосунка.

– Захист від погроз, спрямованих безпосередньо на WAF.

– Термінація SSL і/або TLS (розшифрування й перевірка трафіку перед відправленням до web-застосунку).

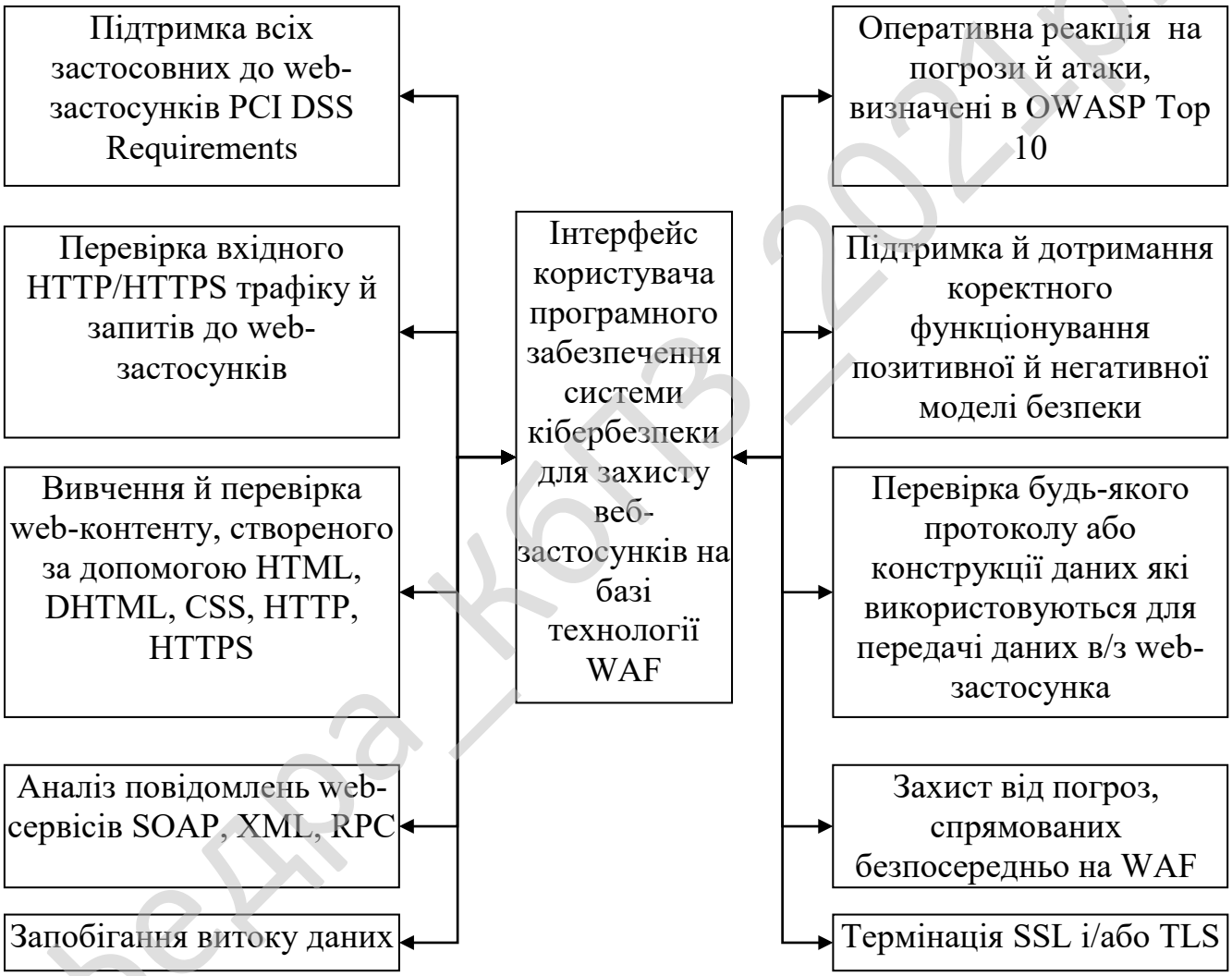

Рисунок 3.2 – Функціональна схема системи

Розглянемо більш докладно PCI DSS Requirements і OWASP Top 10.

PCI DSS Requirements:

– Забезпечити функціонування міжмережевих екранів для захисту даних тримачів платіжних карт.

– Не використовувати паролі й інші системні параметри, задані виробником за замовчуванням.

– Забезпечити безпечне зберігання даних тримачів карт.

– Забезпечити шифрування даних тримачів карт при їхній передачі через мережі загального користування.

– Захищати всі системи від шкідливого ПЗ й регулярно обновляти антивірусне ПЗ.

– Розробляти й підтримувати безпечні системи й застосунки.

– Обмежити доступ до даних тримачів карт у відповідності зі службовою необхідністю.

– Визначати й підтверджувати доступ до системних компонентів.

– Обмежити фізичний доступ до даних тримачів карт.

– Контролювати й відслідковувати будь-який доступ до мережних ресурсів і даним тримачів карт.

– Регулярно виконувати тестування систем і процесів забезпечення безпеки.

– Розробляти й підтримувати політикові інформаційної безпеки для всього персоналу організації.

OWASP Top 10:

– A1: уразливості до впровадження коду (усякого роду ін'єкції, у т.ч. SQL, LDAP і т.д.)

– A2: уразливості XSS (Cross Site Scripting).

– A3: уразливості, що виникають внаслідок некоректної реалізації механізму автентифікації й керування сеансами web-застосунка.

– A4: уразливості, що дозволяють здійснювати неконтрольований доступ до внутрішніх об'єктів web-застосунка прямо. Можливість обходу каталогів (Path Traversal).

– A5: уразливості CSRF (Cross Site Request Forgery).

– A6: уразливості, що виникають внаслідок неправильної конфігурації web-застосунка і його оточення.

– A7: уразливості, пов'язані з некоректною реалізацією криптографічних методів захисту інформації для зберігання критичних даних на стороні сервера.

– A8: уразливості механізму авторизації web-застосунків (можливість прямого доступу до ресурсів по URL).

– A9: недостатня захищеність транспортного протоколу.

– A10: уразливості, що дозволяють використовувати web-застосунок для перенапрямку користувачів на будь-який сайт (Open Redirects).

Недоліки IPS у контексті захисту web-ресурсів

Пристрої IPS призначені для виявлення й блокування атак, з якими не можуть упоратися звичайні міжмережеві екрани. Дані системи аналізують уміст пакетів і порівнюють їх із сигнатурами відомих атак. Крім цього, системи IPS можуть оцінювати й блокувати аномалії протоколів прикладного рівня. Загальне число уразливостей web-застосунків і їх варіацій на кілька порядків вище, чим сумарна кількість сигнатур у базах даних сучасних систем IPS. Тому системи IPS, що функціонують за принципом порівняння пакетів з відомими сигнатурами, не є ефективним застосунком для боротьби з web-атаками. Для надійного захисту web-застосунків потрібне глибоке розуміння їх структури, включаючи URL-параметри, cookie, форми введення даних і ін. На жаль, сучасні системи IPS не в змозі в повному обсязі аналізувати структуру web-застосунків і, відповідно, забезпечувати їхній надійний захист.

Недоліки NGFW у контексті захисту web-ресурсів

Критичною відмінністю між NGFW і WAF є спрямованість даних пристроїв: NGFW, насамперед, орієнтований на контроль зовнішніх застосунків

стосовно підприємства (наприклад, peer-to-peer і Facebook), у той час як WAF забезпечує захист орієнтованих на користувача web-застосунків розміщених на внутрішніх серверах підприємства.

NGFW не контролюють HTTP/HTTPS сесії взагалі або контролює їх частково. Він блокує тільки відомі типи атак і, в основному, використовує «blacklist» підхід до забезпечення web-захисту, а також не компенсує уразливості погано спроектованих web-застосунків. Для захисту більшості web-застосунків NGFW цілком достатньо. Але для застосунків, що вимагають посиленого захисту й постійного моніторингу, WAF забезпечує більш надійний набір елементів керування, які можуть блокувати як відомі типи атак, так і атаки проти яких ще не розроблені захисні механізми (Zero-Day Attacks).

NGFW не оснащуються WAF, тому що це обумовлене необхідністю виділення великої кількості обчислювальної потужності для коректної роботи останнього. Більше того, існує обмежений набір компаній, яким WAF необхідний (в основному це фінансова сфера діяльності). Тому, на даний момент, WAF залишається автономною й незалежною технологією з унікальною й вузьконаправленою функціональністю й високим потенціалом розвитку в найближчому майбутньому.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

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

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Рисунок 3.3 – Діаграма взаємодії процесів

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1.

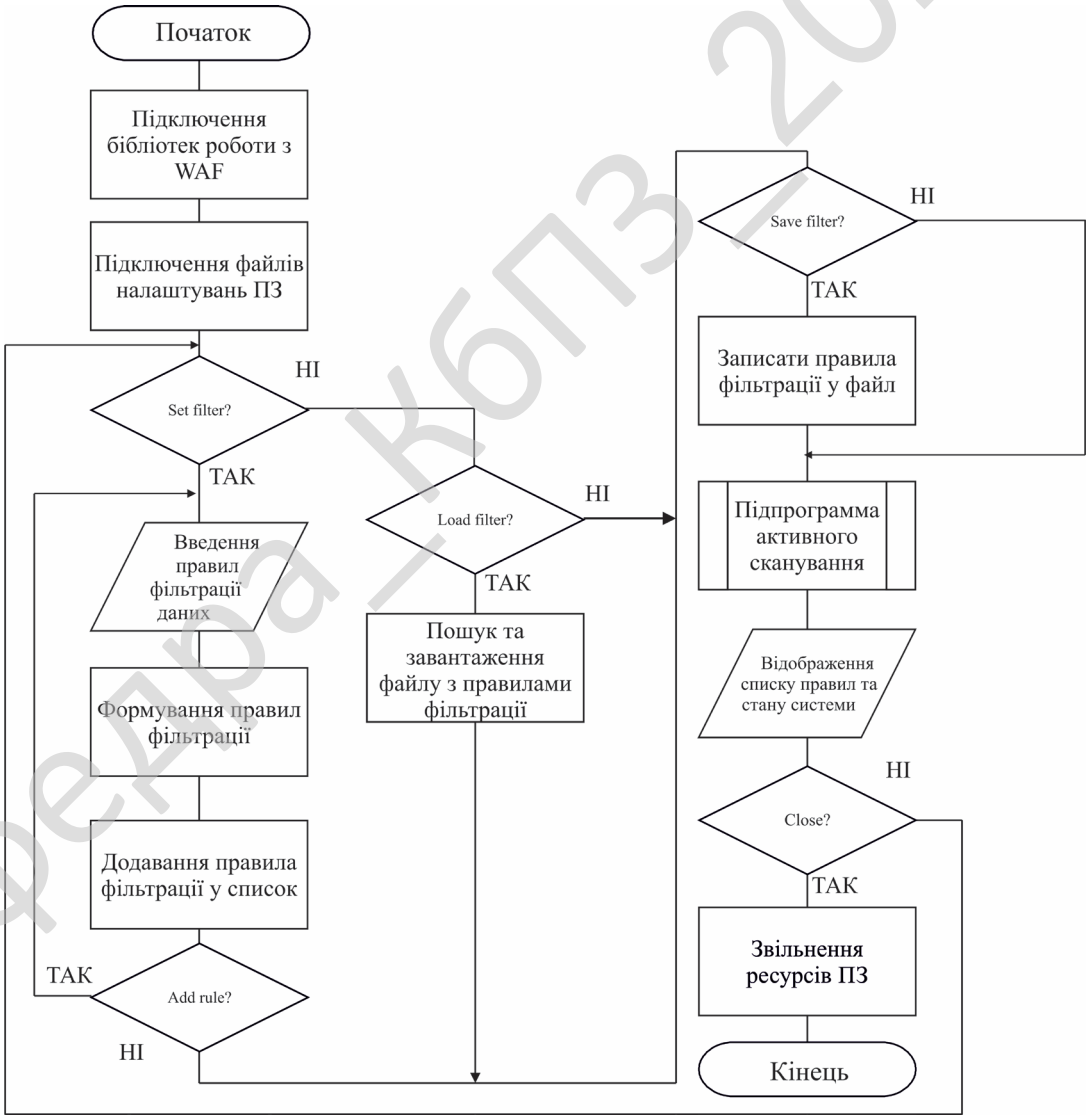

Рисунок 4.1 – Блок-схема основної програми

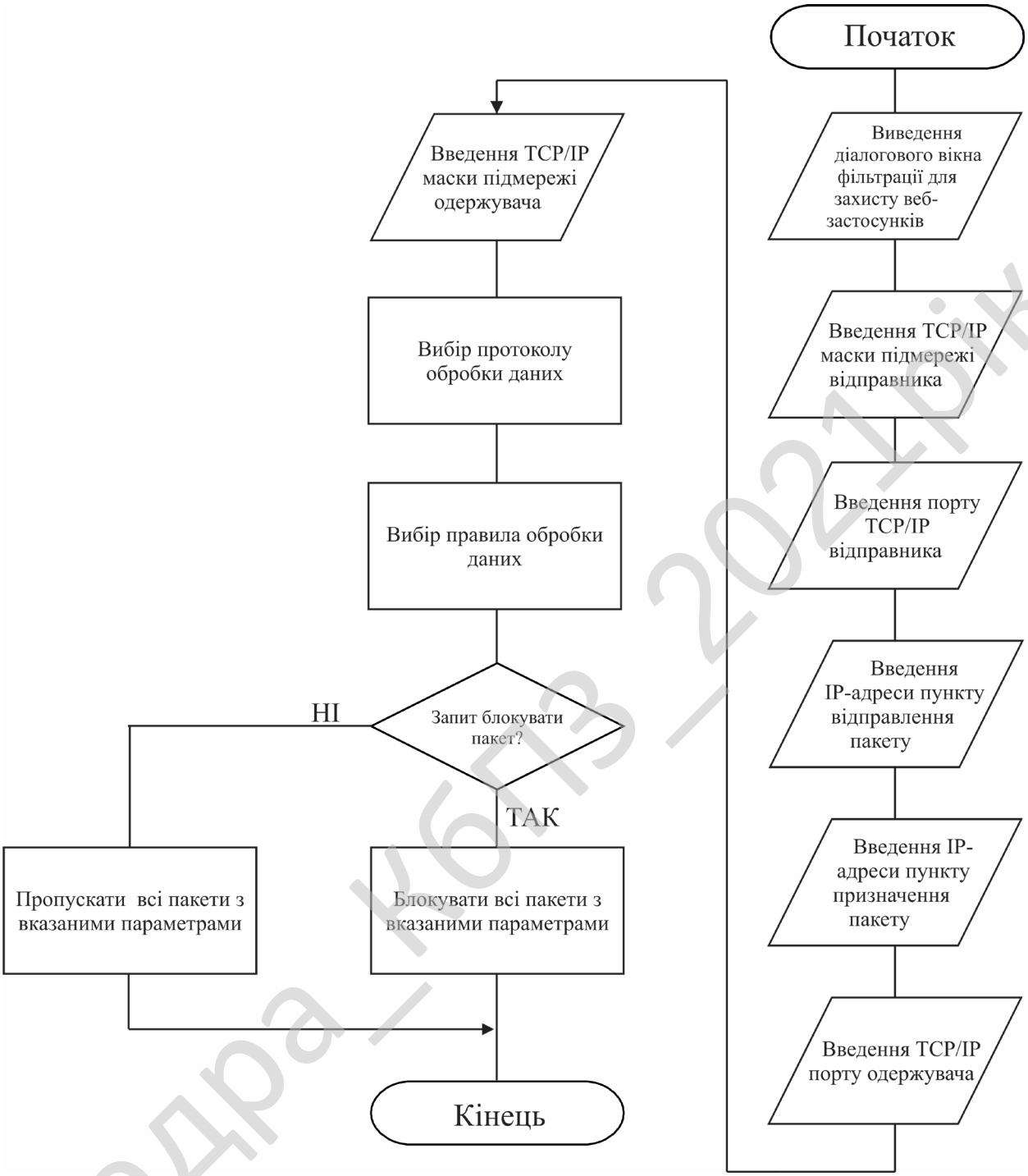

Рисунок 4.2 – Блок-схема роботи підпрограми

З рисунку видно, що після запуску програми спочатку відбувається підключення бібліотек та файлів налаштувань далі через запит відбувається спроба встановлення фільтру з послідуючими діями – введення, формування та додавання правил до списку фільтрації.

Далі проходе запит на додавання правила та встановлення фільтру з реалізацією послідуючих дій.

Після проведених запитів проходе виклик підпрограми активного сканування (детально розглянуто на рис. 4.2) та послідуючого відображення списку правил та поточного стану системи. Останній запит це запит завершення програми наприкінці основного циклу запитів.

Першочергове завдання при реалізації системи кібербезпеки для захисту веб-застосунків на базі технології WAF є створення сканера портів, розглянемо запропоновану реалізацію у вигляді вихідного коду ПЗ, що було реалізовано, а саме модулю сканера:

```
unit Code; // модулю сканера портів

interface // описова частина модулю Code

uses // виклики модулю Code
  Windows, Messages, SysUtils, Classes, Controls, Forms,
  StdCtrls, ExtCtrls, WinSock, AppEvnts, Buttons, Gauges, Spin;

type // об'яви модулю Code
  TForm4 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    SpeedButton1: TSpeedButton;
    ProgressBar1: TGauge;
    ListBox1: TListBox;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    Label3: TLabel;
    Label4: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure ApplicationEvents1Message(var Msg: tagMSG;
      var Handled: Boolean);
    procedure SpeedButton1Click(Sender: TObject);
  end;
```

```
  TMyThread = class(TThread) // потік модулю Code
   protected
    procedure Execute; override;
  end;

  TFSocket = record // структура сокетна модулю Code
   sa: TSockAddr;
   FData: integer;
   TimeOut: integer;
  end;

const // константи модулю Code
 MAX_PORTS=500; // максимальна кількість портів
 Ping=5000; // максимальне чекання

Var // змінні модулю Code
  Form4: TForm4; // екземпляр
  FSocket: array [0..MAX_PORTS - 1] of TFSocket;
  FPort: WORD;
  FInfo: TWSADATA;
  FHost: integer;
  FPStart, FPEnd: WORD;
  SThread: TMyThread;

implementation

{$R *.dfm} // підключення стандартне ресурсів

procedure TForm4.ApplicationEvents1Message(var Msg: tagMSG;
  var Handled: Boolean);
begin
// запит наявності службового сповіщення та сповіщення користувача
 if (Msg.message>=FPStart+WM_USER) and (Msg.message<=FPEnd+WM_USER)
 then
 if WSAGETSELECTERROR(Msg.lParam)=0
 then
  case WSAGETSELECTEVENT(msg.lParam) of
   FD_CONNECT:
      begin
       ListBox1.Items.Add(IntToStr(ListBox1.Count+1)+
                          ') IP: '+inet_ntoa(FSocket[0].sa.sin_addr)+'  |  '+
                          'Port: '+intToStr(Msg.message - WM_USER));
```

```
       Application.ProcessMessages;
      end;
  end;
 Handled:=false;
end;

// дії при створенні форми модулю Code
procedure TForm4.FormCreate(Sender: TObject);
var
 i: integer;
begin
 if WSAStartup(MAKEWORD(2, 0), FInfo)<>0
 then Halt;
 for i:=0 to MAX_PORTS-1 do
 with FSocket[i] do
  begin
   TimeOut := 0;
   FData := Socket(AF_INET, SOCK_STREAM, 0);
   if FData = SOCKET_ERROR
   then
    begin
     WSACleanup;
     Halt;
    end;
  end;
 Application.OnMessage:=Form4.ApplicationEvents1Message;
end;

// дії при виділенні форми модулю Code
procedure TForm4.FormDestroy(Sender: TObject);
var
 i: integer;
begin
 if SThread<>nil
 then
  begin
   SThread.Terminate;
   SThread.WaitFor;
   SThread:=nil;
  end;
 for i:=0 to MAX_PORTS-1 do
  CloseSocket(FSocket[i].FData);
```

```
 WSACleanup;
end;

// ініціалізація сокетів
procedure InitSockets;
var
 i: integer;
begin
 for i:=0 to MAX_PORTS-1 do
  with FSocket[i] do
   if (FPort>=FPEnd) or (SThread.Terminated)
   then break
   else
    if GetTickCount-TimeOut>PING
    then
     begin
      sa.sin_family:=AF_INET;
      sa.sin_addr.S_addr:=FHost;
      sa.sin_port:=htons(FPort);
      WSAAsyncSelect(FData, Application.Handle, WM_USER+FPort, FD_CONNECT);
      connect(FData, FSocket[i].sa, SizeOf(FSocket[i].sa));
      TimeOut:=GetTickCount;
      inc(FPort);
      Form4.Label2.Caption:=Порт: '+intToStr(FPort);
      Form4.ProgressBar1.Progress:=trunc(((FPort-FPStart)*100)/(FPEnd-
                                    FPStart));
      Application.ProcessMessages;
      Sleep(PING div MAX_PORTS);
     end;
end;

// Дії потоку модулю
procedure TMyThread.Execute;
var
 i: integer;
 buf: in_addr;
begin
// беремо параметри з форми
 FPStart:=Form4.SpinEdit1.Value;
 FPEnd:=Form4.SpinEdit2.Value;
 FHost:=inet_addr(PChar(Form4.Edit1.Text));
 if (FPEnd<=FPStart)or (Form4.SpinEdit1.Value>32768) or
```

```
                                (Form4.SpinEdit2.Value>32768)
 then FHost:=SOCKET_ERROR;
 if FHost=SOCKET_ERROR
 then
  begin
   if not Terminated
   then SThread:=nil;
   Form4.SpeedButton1.Caption:='Запуск';
   Exit;
  end;
 buf.S_addr:=FHost;
 Form4.Edit1.Text:=inet_ntoa(buf);
 Form4.ListBox1.Clear;
 FPort:=FPStart;
// основний цикл потоку
 for i:=0 to MAX_PORTS-1 do
  with FSocket[i] do
   begin
    TimeOut:=0;
// робота з сокетом
    FData:=Socket(AF_INET, SOCK_STREAM, 0);
   end;
 //
 while not Terminated do
  begin
   InitSockets;
   if FPort>=FPEnd
   then break;
  end;
 //
 for i:=0 to MAX_PORTS-1 do
  CloseSocket(FSocket[i].FData);
 //
 if not Terminated
 then
  begin
   SThread:=nil;
   Form4.SpeedButton1.Caption:='Запуск';
  end;
end;
```

```
// дії при натисканні кнопки
procedure TForm4.SpeedButton1Click(Sender: TObject);
begin
 if SThread=nil
 then
  begin
   SThread:=TMyThread.Create(false);
   SpeedButton1.Caption:='Стоп';
  end
 else
  begin
   SThread.Terminate;
   SThread.WaitFor;
   SThread:=nil;
   SpeedButton1.Caption:='Запуск';
  end;
end;
end.
```

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

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників.

Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

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

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

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

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

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ.

Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

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

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

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

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

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

XP намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

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

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

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

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

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

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

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

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

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

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

3.4 Стандарт кодування (Coding standard or Coding conventions).

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

NetBIOS (Network Basic Input/Output System) – протокол для роботи в локальних мережах на персональних ЕОМ типу IBM/PC, розроблений у вигляді інтерфейсу, який не залежить від фірми–виробника. Був розроблений фірмою Sytek Corporation за замовленням IBM в 1983 році. Він включає в себе інтерфейс сеансового рівня (англ. NetBIOS interface), в якості транспортних протоколів використовує TCP і UDP.

Особливістю NetBIOS є можливість його роботи поверх різних протоколів, найпоширенішими/відомими з яких є NetBEUI, IPX і стек протоколів TCP/IP; причому якщо старі версії Windows орієнтувалися на більш легкі в реалізації і менш ресурсомісткі NetBEUI і IPX, то сучасні Windows орієнтуються на TCP/IP.

При використанні NetBEUI і IPX NetBIOS сам забезпечує надійність доставки даних (функціональність SPX не використовувати), а при використанні TCP/IP надійність доставки забезпечує TCP, за що удостоївся окремого імені «NBT».

Інтерфейс NetBIOS являє собою типовий інтерфейс взаємодії програм (API) для забезпечення мережевих операцій введення–виведення і управління транспортним протоколом.

Програми, що використовують NetBIOS API інтерфейс, можуть працювати тільки при наявності протоколу, що допускає використання такого інтерфейсу.

NetBIOS також визначає протокол, що функціонує на сеансовому/транспортному рівнях моделі OSI. Цей протокол використовується протоколами нижчих рівнів, такими як NBFP (NetBEUI) і NetBT для виконання мережевих запитів вводу–виводу і операцій, описаних в стандартному интерфейсном наборі команд NetBIOS.

Тобто, NetBIOS сам не підтримує виконання файлових операцій. Ця функція покладається на протоколи нижчих рівнів, а сам NetBIOS забезпечує тільки зв'язок з цими протоколами і NetBIOS API–інтерфейс.

Пакет NETBIOS створений для використання групою ЕОМ. Підтримує як режим сесій (робота через з'єднання), так і режим дейтаграм (без встановлення з'єднання). 16–символьні імена об'єктів в NETBIOS розподіляються динамічно.

NETBIOS має власну DNS, яка може взаємодіяти з інтернетівський. Ім'я об'єкта при роботі з NETBIOS не може починатися з символу *.

Додатки можуть знайти через NETBIOS потрібні їм ресурси, встановити зв'язок і послати або отримати інформацію. NETBIOS використовує для служби імен порт 137, для служби дейтаграм – порт 138, а для сесій – порт 139. Будь–яка

сесія починається з NETBIOS–запиту, завдання IP–адреси і визначення TCP–порту віддаленого об'єкта, далі йде обмін NETBIOS–повідомленнями, після чого сесія закривається.

Сесія здійснює обмін інформацією між двома NETBIOS – додатками. Довжина повідомлення лежить в межах від 0 до 131 071 байт. Припустимо одночасне встановлення декількох сесій між двома об'єктами.

При організації IP–транспорту через NETBIOS IP–дейтаграмма вкладається в NETBIOS–пакет. Інформаційний обмін відбувається в цьому випадку без встановлення зв'язку між об'єктами. Імена NETBIOS повинні містити в собі IP–адреси. Так, частина NETBIOS–адреси може мати вигляд IP.XX.XX. XX. XX, де IP вказує на тип операції (IP через Netbios), а XX. XX. XX.XX – IP– адреса.

Система NETBIOS має власну систему команд (call, listen, hang up, send, receive, session status, reset, cancel, adapter status, unlink, remote program load) і примітивів для роботи з дейтаграммами (send datagram, send broadcast datagram, receive datagram , receive broadcast datagram).

Всі кінцеві вузли NETBIOS діляться на три типи:

– широкомовні («b») вузли;

– вузли точка–точка («p»);

– вузли змішаного типу («m»).

IP–адреса може асоціюватися з одним із зазначених типів. B–вузли встановлюють зв'язок зі своїм партнером за допомогою широкомовних запитів. P– і M–вузли для цієї мети використовують netbios сервер імен (NBNS) і сервер розподілу дейтаграм (NBDD).

NetBIOS забезпечує:

– реєстрацію і перевірку мережевих імен;

– встановлення і розрив з'єднань;

– зв'язок з підтвердженням доставки інформації;

– зв'язок без підтвердження доставки інформації;

– підтримку управління і моніторингу драйвера і мережевої карти.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-1, який заснований на перетвореннях алгоритму SHA-1. SHACAL-1 шифрує 160-бітний блок даних з використанням 512-бітного ключа шифрування. Допускається використання більше коротких ключів шифрування (не коротше 128 біт), які перед виконанням розширення ключа (процедура розширення ключа також успадкована від SHA-1 і буде описана нижче) повинні бути доповнені нульовими бітами для досягнення 512-бітного розміру.

В алгоритмі SHACAL-1 передбачено 80 раундів шифрування. Шифруєме повідомлення представляється у вигляді п'яти 32 -бітних субблоків A, B, C, D і E, над якими в кожному раунді виконуються наступні дії:

$$A_{i+1} = K_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + M_i,$$

$$B_{i+1} = A_i,$$

$$C_{i+1} = B_i <<< 30,$$

$$D_{i+1} = C_i,$$

$$E_{i+1} = D_i,$$

де i – номер раунду (i = 0...79),

K_i – фрагмент розширеного ключа для і-го раунду,

f_i – функція для і-го раунду (див. нижче),

<<< – операція побітового циклічного зрушення вліво,

M_i – константи, що модифікують, визначені в такий спосіб:

Раунди	Значення константи
0...19	5A827999
20...39	6ED9EBA1
40...59	8F1BBCDC
60...79	CA62C1D6

Використовувані в раундах функції f_i визначені так:

Раунди	Функція
0...19	$f(x,y,z)=(x\&y)\|(x'\&z)$
20...39,60…79	$f(x,y,z)=x \oplus y \oplus z$
40...59	$f(x,y,z)=(x \oplus y)\|(x \oplus z)\|(y \oplus z)$

У таблиці символами &, | і $\oplus$ позначені, відповідно, побітові логічні операції «і», «або» й «або, що виключає» (XOR); x' позначає побітовий комплемент до x.

Шифртекстом є конкатенація вмісту змінних A_{80}, B_{80}, C_{80}, D_{80} і E_{80}.

Процедура розширення ключа в алгоритмі SHACAL-1 також досить проста, вона виконується у два етапи:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \ldots K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16} \ldots K_{79}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = (K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}) <<< 1.$$

Раунди розшифрування виконуються у зворотній послідовності; кожний з них має на увазі виконання наступних операцій:

$$A_i = B_{i+1},$$

$$B_i = C_{i+1} <<< 2,$$

$$C_i = D_{i+1},$$

$$D_i = E_{i+1},$$

$$E_i = K'_i + (B_{i+1} <<< 5)' + f_i'(C_{i+1} <<< 2, D_{i+1}, E_{i+1}) + A_{i+1} + M'_i + 4.$$

Тут запис $f'(x)$ позначає побітовий комплемент результату виконання операції $f(x)$.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Функціональних кнопок ПЗ: Додавання правил; Видалення правил; Старт; Стоп.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші: Довідка; Журнал роботи.

– Верхнього меню: Файл; Правила; Довідка.

– Розділу обрання групи правил.

– Розділу виведення результату роботи системи.

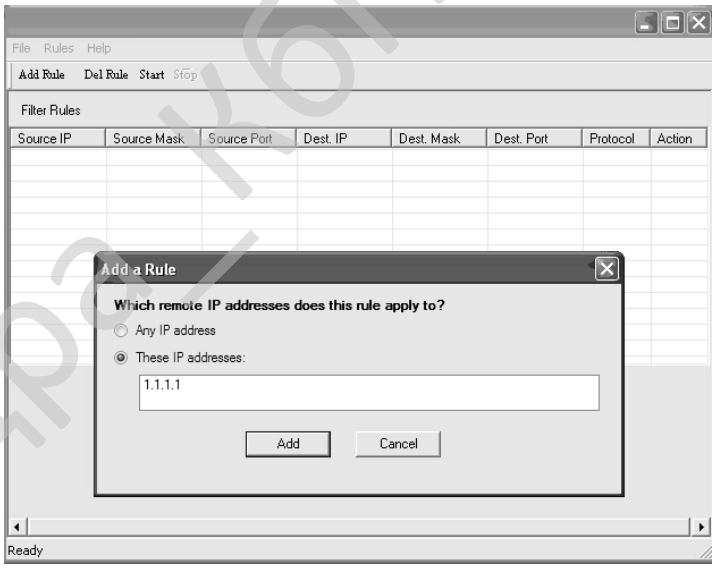

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з

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

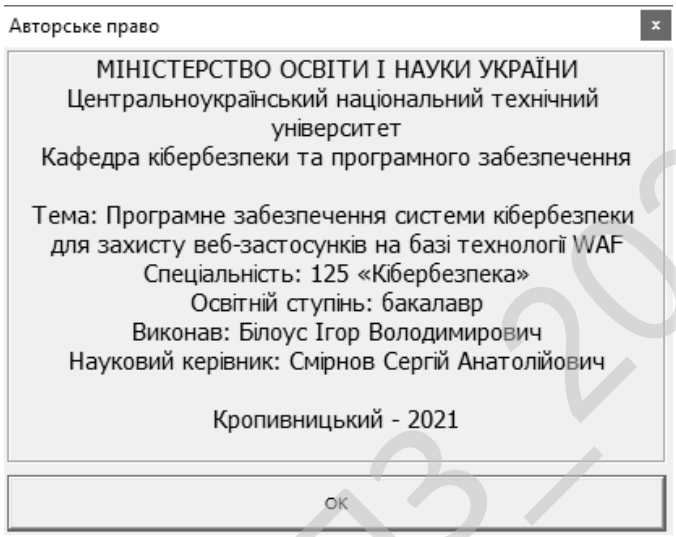

Рисунок 5.2 – Авторське право

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. Таким чином у результаті вищерозглянутого можна стверджувати що розроблено інтерфейс системи у відповідності з вибраною метою роботи. Система містить максимальний необхідний набір функцій придатних для виконання будь-яких дій для забезпечення повноцінної роботи програми. Далі розглянемо висновки та використані літературні джерела.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем кібербезпеки для захисту веб-застосунків на базі технології WAF.

– Досліджена система кібербезпеки для захисту веб-застосунків на базі технології WAF.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання для захисту веб-застосунків на базі технології WAF.

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

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

Програма реалізована на мові високого рівня Visual C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для захисту веб-застосунків на базі технології WAF. 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.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-1.

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. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.: BHV, 2001. – 624 с.

2. Майника Э. Алгоритмы оптимизации на сетях и графах: пер. с англ. / Э. Майника; под ред. Е.К. Масловского. – М.: Мир, 1981. – 321 с.

3. МСЭ-Т Рекомендация G.101. Международные телефонные соединения и цепи – Общие определения //11/2003. [Електронний ресурс]. – Режим доступу до ресурсу: http://www.telecom61.ru/SharedFiles/ Download.aspx? …pageid=106

4. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров АТМ сетей / Алексей Николаевич Назаров. – М.: Горячая линия – Телеком, 2002. – 256 с.

5. Одом Ш. Коммутаторы CISCO / Ш. Одом, Х. Ноттингем – М.: "Кудиц-Образ", 2003. – 528 с.

6. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. – 2-е изд. – СПб.: Питер, 2007. – 958 с.

7. Партыка С.А. Метод ускоренной коррекции SPT с использованием динамических алгоритмов [Електронний ресурс]. – Режим доступу до ресурсу: http://openarchive.nure.ua/bitstream/123456789/936/1/ ASU_158_2012%20%2842-47%29.pdf

8. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

9. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

10. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

11. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

12. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. – Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

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

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

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

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

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

18. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

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

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

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

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

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

24. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

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

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

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

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

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

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

31. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

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

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

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

35. Смирнов С. А. Технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-

технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

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

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

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

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

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

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

42. Смирнов С. А. GERT-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

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

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

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

46. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы /

А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук.-практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

47. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

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

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

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

51. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-

2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХНМІ ДВНЗ «УБС», 2016. – С. 178-182.

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 185-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи 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 кібербезпеки для захисту веб-застосунків на базі технології WAF;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з с умісними 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 XP/Vista/7/8/10.

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 – 60 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

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.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 2.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки для захисту веб-застосунків на базі технології WAF

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 42

Літера: РП

Кропивницький – 2021 року

WAF_App.cpp - головний файл програми

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

#include "MainFrm.h"
#include "WAF_AppDoc.h"
#include "WAF_AppView.h"

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

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

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

/////////////////////////////////////////////////////////////////////////////
CWAF_AppApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CWAF_AppApp initialization

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

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


	SetRegistryKey(_T("WAF_App"));

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

  //Створення  SDI фрейму вікна
```

```
	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CWAF_AppDoc),
		RUNTIME_CLASS(CMainFrame),
		RUNTIME_CLASS(CWAF_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 CWAF_AppApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}

//функція створення й відкриття діалогового вікна допомоги по програмі
void CWAF_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 "WAF_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();
}
```

WAF_AppDoc.cpp – формування правил

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

#include "WAF_AppDoc.h"

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

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

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

/////////////////////////////////////////////////////////////////////////////
// CWAF_AppDoc construction/destruction

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

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

	return TRUE;
}

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

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

```
	CDocument::AssertValid();
}

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

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

WAF_AppView.cpp – Формування вікон

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

#include "WAF_AppDoc.h"
#include "WAF_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

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

BEGIN_MESSAGE_MAP(CWAF_AppView, CFormView)
	//{{AFX_MSG_MAP(CWAF_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()

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

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

//функція зчитування й установки дані вікна
void CWAF_AppView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CWAF_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 CWAF_AppView::PreCreateWindow(CREATESTRUCT& cs)
{


	return CFormView::PreCreateWindow(cs);
}

//Ініціалізація вікна
```

```
void CWAF_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);


}

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

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

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

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

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

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

```
////////////////////////////////////////////////////////////////////////////
// CWAF_AppView message handlers
//опис функції відновлення списку правил в інтерфейсі
void CWAF_AppView::UpdateList()
{
	CWAF_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 CWAF_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 "WAF_App.h"

#include "MainFrm.h"
#include "RuleDlg.h"
#include "DefaultActionDlg.h"
#include "WAF_AppDoc.h"
#include "WAF_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("WAF_");

      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()
{
	CWAF_AppDoc *doc = (CWAF_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()
{
	CWAF_AppDoc *doc = (CWAF_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
				CWAF_AppView *view = (CWAF_AppView *)GetActiveView();

				view->UpdateList();
			}

		}
	}

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

//функція видалення правила
void CMainFrame::OnButtondel()
{
	CWAF_AppView *view = (CWAF_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);

	CWAF_AppDoc *doc = (CWAF_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;

	CWAF_AppDoc *doc = (CWAF_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);
```

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

	view->UpdateList();
}

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

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

                    CWAF_AppView *view = (CWAF_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 створенй, то створюю його.
	if(!interfacesTable.Lookup(CharToIp(localInterfaceIp), interfaceHandle))
	{
		return -1;
	}


	DWORD errorCode;

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

	if(errorCode != NO_ERROR)
		return -1;

	return 0;
}

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

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

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

}

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

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

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

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

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


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

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

              hInterface = NULL;

              return -2;
        }

        return 0;
}

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

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

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

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

              auxCad = strchr(auxCad, '.');

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

              auxCad++;
        }


        return lIp;
}
```

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 "WAF_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;
}
```