МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Жупило Микола Миколайович

Програмне забезпечення системи кібербезпеки WIPS для WiFi

Спеціальність: 125 «Кібербезпека»

Освітній ступінь: бакалавр

Науковий керівник:

Буравченко Костянтин Олегович _____

(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

Центр *Заочної та дистанційної освіти*
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

Жупилу Миколі Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки WIPS для WiFi*

керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 135-02 від 24.12.2020 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 WIPS для WiFi*

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

Жупило М.М. Програмне забезпечення системи кібербезпеки WIPS для WiFi. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи кібербезпеки WIPS для WiFi.

Метою розробки є програмне забезпечення системи кібербезпеки WIPS для WiFi.

Результат роботи – програмна реалізація системи кібербезпеки WIPS для WiFi.

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 RAD Studio Delphi 10.4.

Ключові слова: кібербезпека, WIPS

ABSTRACT

Zhupylo M.M. WIPS cybersecurity software for WiFi. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

This bachelor's degree software has been developed for the WIPS cybersecurity system for WiFi.

The purpose of the development is WIPS cybersecurity software for WiFi.

The result is a software implementation of the WIPS cybersecurity system for WiFi.

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 with Windows XP / Vista / 7/8/10.

The program is developed in the environment of RAD Studio Delphi 10.4.

Keywords: cybersecurity, WIPS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 4
1.1 Призначення системи..... 4
1.2 Область застосування 5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми кваліфікаційної бакалаврської роботи..... 9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 19
2.3 Розгорнута постановка завдання 25
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 27
3.1 Опис функціонування системи 27
3.2 Розробка структурної схеми..... 31
3.3 Розробка функціональної схеми 35
3.4 Розробка діаграми процесів 45
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 47
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 47
4.2 Захист розробленого програмного забезпечення..... 64
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 69
6 ОСНОВНІ ВИСНОВКИ 71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 73

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AP	–	Access Point, точка доступу
BYOD	–	Bring Your Own Device, співробітники можуть використовувати для роботи свої особисті мобільні пристрої
CIP	–	технологія Cloud Integration Point (CIP) дозволяє інтегрувати WatchGuard Wi-Fi Cloud зі сторонніми локальними бездротовими контролерами, службами керування й журналом подій
ESM	–	Enterprise Security Management
NAT	–	трансляція мережних адрес
PCI	–	рада по стандартах безпеки
WG	–	WatchGuard
Wifi	–	Wireless Fidelity – бездротова точність
WIPS	–	Wireless Intrusion Prevention Systems, бездротові системи запобігання вторгнень
WPA2	–	алгоритм шифрування у Wi-Fi
WR	–	Wireless Router, бездротовий роутер

ВСТУП

Актуальність теми. Від «атаки посередині» («людей посередині», man-in-the-middle) до глобальної епідемії програмного вимагання – мережним адміністраторам потрібно щось більше, ніж простий посібник з загроз, які можуть виникнути в їхніх мережах. Вони потребують діючого інструментарію, який може не тільки виявляти, але й нівелювати загрози, що з'явилися. Тому не дивно, що бездротові системи запобігання вторгнень (Wireless Intrusion Prevention Systems, WIPS) стають усе більш популярними.

Основна ідея WIPS полягає в тому, що, якщо ви дійсно прагнете не пустити зловмисників у вашу мережу, вам знадобиться аналітичний інструментарій як на рівні точки доступу Wi-Fi, так і на серверному рівні, у тому числі й у хмарі. Тому такі відомі вендори, як NETSCOUT, HP Software (раніше Aruba), Cisco і ряд інших компаній пропонують застосунки, які поєднують аналіз мережних даних з інтелектуальним програмним ядром на серверному рівні.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки WIPS для WiFi.

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

– Огляд існуючих систем кібербезпеки WIPS для WiFi.

– Дослідження системи кібербезпеки WIPS для WiFi.

– Програмна реалізація системи кібербезпеки WIPS для WiFi.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі WIPS для WiFi.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки WIPS для WiFi, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система запобігання вторгнень у бездротову мережу (WIPS) – це спеціальний пристрій безпеки або інтегрований програмний застосунок, який контролює радіочастотний спектр бездротової локальної мережі на предмет несанкціонованих точок доступу й інших загроз бездротової мережі.

WIPS порівнює MAC-адреси всіх точок бездротового доступу в мережі з відомими сигнатурами попередньо авторизованих відомих точок бездротового доступу й попереджає адміністратора при виявленні невідповідності. Щоб обійти підробку MAC-адреси, деякі високопродуктивні WIPS можуть аналізувати унікальні радіочастотні сигнатури, які генерують бездротові пристрої, і блокувати невідомі радіо-відбитки.

Рада по стандартах безпеки PCI рекомендує використовувати WIPS для автоматизації сканування бездротової мережі. Крім забезпечення рівня безпеки для бездротової локальної мережі, WIPS також корисний для моніторингу продуктивності мережі й виявлення точок доступу з помилками конфігурації.

Є три основні способи розгорнути WIPS.

Перший, який в основному зустрічається на нижньому рівні ринку, відомий як квантування часу або поділ часу. У цьому типі розгортання точка бездротового доступу виконує подвійну функцію, забезпечуючи мережний трафік з можливістю бездротового підключення при періодичному скануванні на предмет шахрайських точок доступу.

У другому підході, який відомий як інтегрований WIPS, датчик, вбудований в авторизовану точку доступу, постійно сканує радіочастоти в пошуках неавторизованих точок доступу.

У третьому підході, відомому як накладення WIPS, датчики розміщаються по всьому будинкові для контролю радіочастот. Датчики пересилають зібрані дані на централізований сервер для подальшого аналізу, дій і архівування журналів. Цей підхід більш дорогий, оскільки вимагає спеціального устаткування, але також уважається найбільш ефективним.

Апаратне забезпечення накладення WIPS нагадує стійковий сервер, а відповідні датчики – точки доступу Wi-Fi. Більшість систем накладення WIPS мають ті самі основні компоненти:

– Датчики – контролюють радіочастотний спектр і пересилають журнали назад на центральний сервер керування.

– Сервер керування – одержує інформацію, захоплену датчиками, і на основі цієї інформації вживає відповідних заходів захисту.

– Сервер бази даних – зберігає й систематизує інформацію, отриману датчиками.

– Консоль – надає адміністраторам інтерфейс для налаштування й керування WIPS.

Хоча оверлеї WIPS надають безліч коштовних функцій і засобів захисту, особливо для великих підприємств, які збирають дані про клієнтів, вони можуть бути досить дорогими. З урахуванням устаткування, застосунків, підписок і навчання підприємство з 250 точками доступу може витратити до 100 000 доларів на WIPS.

1.2 Область застосування

Термін Wifi був уведений Wi-Fi Альянсом (Wifi Alliance також відомий як «Wireless Ethernet Compatibility Alliance» – WECA). Споконвічно, термін звучав «IEEE 802.11 b-сумісні», але в Wifi Альянс, вирнішили, що така назва занадто довга й складне для запам'ятовування. Термін Wifi ніяк не розшифровувався й

мав лише співзвучна з Hi-Fi для споживачів назва. Пізніше Wifi став розшифровуватися як Wireless Fidelity – бездротова точність.

Швидше за все секрет такий омани криється в споконвічно неправильній рекламній кампанії постачальників устаткування й платформ, оснащених бездротовими модулями Wi-Fi. За всю історію було досить масових рекламних кампаній, після яких виходило так, що люди називали все, що асоціювалося з рекламованим продуктом, його іменем. Явний приклад – це фірма Xerox. Дотепер усі копіювальні апарати (не важливо, який вони фірми) називають «ксероксами».

Приблизно те ж відбулося й з Wi-Fi мережами. На початку не було Інтернет-провайдерів, що використовували бездротові мережі для надання доступу в Інтернет своїм клієнтам, хоча при цьому виробники й дилери просували свою продукцію під такими гаслами, як «Мобільний Інтернет Wi-Fi у Вашому ноутбуці» та ін. З одного боку, це правильне вираження, тому що якщо, наприклад, у Вас будинку їсти Інтернет і встановлена точка доступу, те мобільний Інтернет Ви все-таки одержите. Але якщо подивитися з іншого боку, то виходить, що покупка ноутбука з наклейкою Wi-Fi не дає Вам гарантії одержання доступу в Інтернет де завгодно.

Зараз, звичайно, такого вже ні, але в головах споживачів чітко закріпилося, що Wi-Fi – це Інтернет. Так і відбувається: коли люди довідаються про сферу нашої діяльності, випливає відповідна реакція: « ПРО, клас, мені саме потрібно підключитися до Wi-Fi!» (припускаючи при цьому необхідність підключення до мережі Інтернет).

Для початку про самий Wi-Fi устаткуванні. На сьогоднішній день існує велика кількість компаній, які роблять бездротове устаткування. Основні типи пристроїв – це «точка доступу» (Access Point) і «бездротовий роутер» (Wireless Router, простіше говорячи, багатофункціональна точка доступу).

Начинка у всіх пристроїв приблизно однакова, в основному вони відрізняються програмним забезпеченням і режимами роботи. Якщо Ви чуєте, що точки доступу якого-небудь виробника працюють нестабільно, часто зависають і

т.д., те, швидше за все, це викликане недоробкою програмного забезпечення, тобто «сирий» або неперевірений у роботі продукт був випущений у продаж.

З іншого боку, є чимало компаній, які роблять дійсно гарні пристрої зі стабільно працюючими прошиваннями. Під час роботи доводилося зустрічатися й тестувати різні пристрої, і якщо брати в приклад клас устаткування для побудови невеликих корпоративних мереж, те можна перелічити кілька надійних виробників: Cisco-Linksys, Asus. Правда не варто кидатися й купувати устаткування з логотипами цих фірм. Для початку почитайте й вивчите, що пишуть про необхідні Вам пристрої (на жаль, і в надійних виробників бувають «помилки»), або звернетеся до професіоналів.

У підсумку точка доступу Wi-Fi – це простий пристрій, і, щоб одержати саме Інтернет через Wi-Fi, Вам, для початку, необхідно підключитися до якого-небудь провайдера й уже потім на це з'єднання встановити бездротове устаткування. У цьому випадку у Вас буде бездротової доступ в Інтернет. Якщо у Вас будинку на ноутбуці відображається безліч мереж, те, швидше за все, це мережі Ваших сусідів, які вже підключилися до мережі Інтернет через кабельного провайдера й установили Wi-Fi устаткування.

Також існують провайдери, які будують мережі Wi-Fi за допомогою установки бездротових точок доступу на свої локальні мережі й надають, таким чином, доступ в Інтернет для своїх клієнтів.

Тепер, коли вже більш-менш зрозуміло, що таке Wi-Fi устаткування, можна, не сильно глибшаючись у технічну сторону, розповісти про те, як усе це працює.

У точці доступу встановлений радіомодуль, який виконує функції приймання-передачі даних. Такий же модуль установлюється в комп'ютер або ноутбук. Ці модулі проводяться різними компаніями на основі різних чипів. Стандарт Wi-Fi ухвалювався спеціально для того, щоб устаткування Wi-Fi різних виробників було сумісно між собою. Тобто щоб Ви, побачивши наклейку Wi-Fi

на коробці від точки доступу, були впевнені, що вона заробить разом з Вашим ноутбуком.

Після до точки доступу підключається кабель від Інтернет-провайдера, і проводиться відповідне налаштування устаткування й комп'ютера. У підсумку Ви одержуєте доступ в Інтернет, тільки вже не через кабель, а через Wi-Fi мережа. Виходить, що точка доступу або роутер служать деяким бездротовим «шлюзом» між Вашим комп'ютером і провайдером.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки WIPS для WiFi, 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

WatchGuard

Число пристроїв, що використовують технологію Wi-Fi, неухильно росте, у зв'язку із чим компанія WatchGuard (WG) також удосконалює свої застосунки для організації бездротової мережі, які можна використовувати як за допомогою контролера Wi-Fi, вбудованого в Firebox UTM, так і як окремий застосунок – WatchGuard Wi-Fi Cloud. При цьому в замовників WG немає необхідності здобувати нові точки для переходу з одного застосунку на інше, а досить просто поміняти ліцензійне сполучення точок доступу:

Таким чином, кожний може вибрати для себе рівень функціонала, який необхідний у компанії, а також комбінувати застосунок по організації бездротової мережі з застосунком по забезпеченню її захисту. Швидка й безпечна бездротова мережа WatchGuard Wi-Fi Cloud надає потужні інструмент організації, керування й захисту Wi-Fi, такі як:

Керування:

– Легко масштабується від однієї точки доступу до необмеженого їхньої кількості, розташованих у декількох місцях (містах і країнах).

– Мобільний веб-застосунок дозволяє управляти бездротовими мережами за допомогою будь-якого мобільного пристрою.

– Ієрархічна архітектура визначення місце розташування дозволяє мережним адміністраторам управляти більшою кількістю місць із єдиної консолі.

– Можливості спектрального аналізу з автоматичною оптимізацією радіочастот, розгортанням мережі за технологією MESH і щоб уникнути перешкод і наведень.

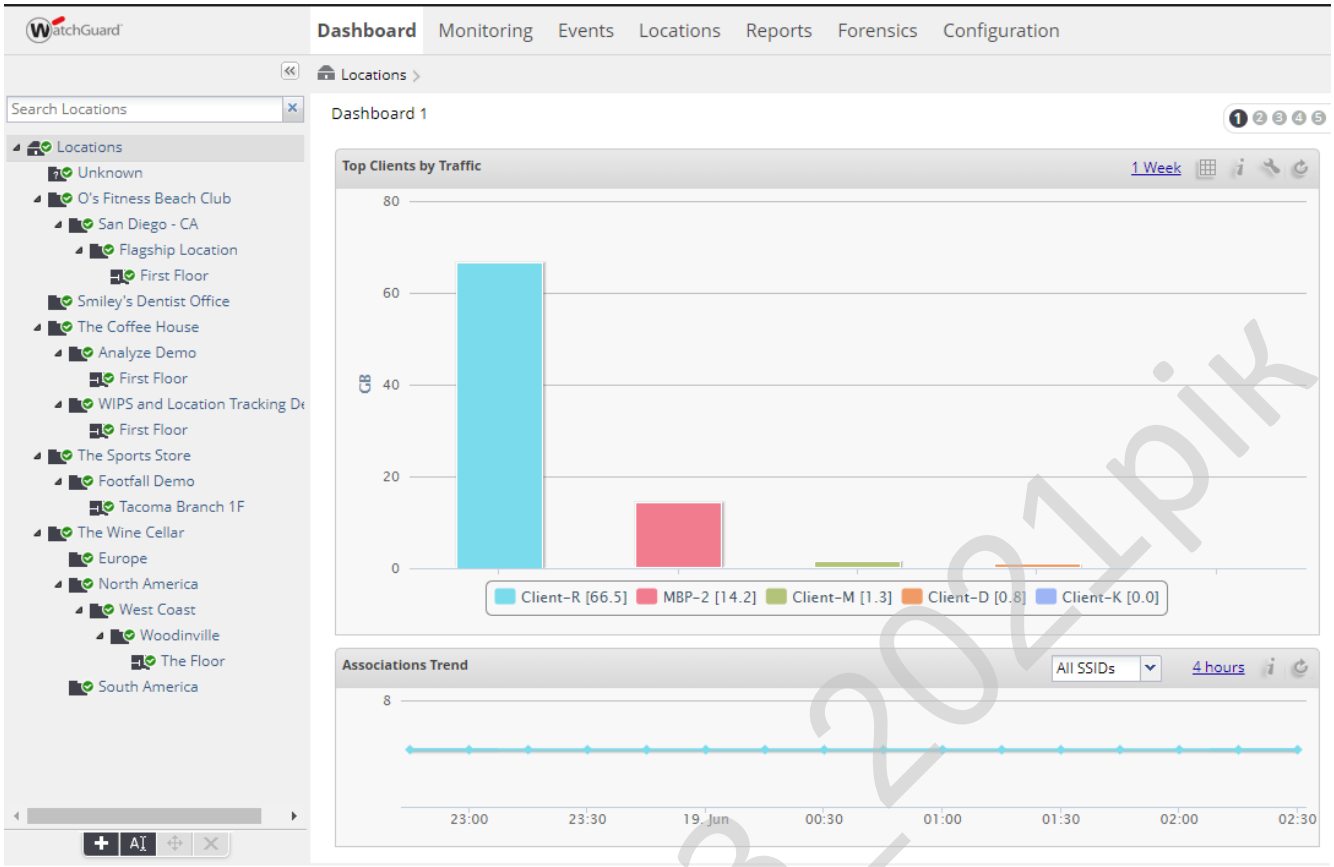

Рисунок 2.1 – Інтерфейс користувача WatchGuard Wi-Fi Cloud

Аналітика:

– Потужний генератор звітності, що включає в себе маркетингові дані, відвідуваність, час затримки, коефіцієнти конверсії й облік повторних відвідувачів.

– Одержання додаткової інформації про аудиторію користувачів вашої точки доступу, що надає публічний Wi-Fi: підлога, вік і лояльність до бренда

Додаткові інструменти;

– Інструменти створення сторінок входу й автентифікації без необхідності навичок веб-дизайну.

– Маркетингові інструменти для організації промоакцій, спецпропозицій і локальних маркетингових компаній.

Хоча клієнти, і співробітники вимагають доступу до швидкого Wi-Fi, часто залишається не у фокусі, який величезний пробіл він залишає у вашій безпеці.

Wireless Intrusion Prevention System (WIPS)

Технологія Wireless Intrusion Prevention виявляє загрози безпеки бездротової мережі, щоб захистити вашу мережу й забезпечити активне запобігання певних типів атак.

Точка доступу може виявити точне місце розташування підозрілих точок доступу й клієнтів і автоматично відключати їх за допомогою технології WIPS

З технологією WIPS і Wi-Fi Cloud від WatchGuard, не обов'язково перебудовувати свою Wi-Fi мережа для її безпеки

Так, у звіті по безпеці незалежної тестової компанії Miercom рівняється ефективність продукту по шести відомими категоріями загроз Wi-Fi і ілюструється схований недолік безпеки з багатьма конкуруючими застосунками Wi-Fi.

Немає необхідності перебудовувати й змінювати мережу, просто потрібно додати WIPS.

Усі точки доступу WG можуть працювати як у якості точки доступу Wi-Fi, так і виділеного датчика безпеки – WIPS-sensor. Це означає, що при розгортанні в якості виділених датчиків, WIPS-пристрою працюють із існуючими точками доступу (Cisco Meraki, Aruba, Ruckus і т.д.) і додають захист бездротової мережі корпоративного рівня. У цьому випадку, замість надання захищеного трафіку Wi-Fi користувачам, точки доступу надають безпрецедентний захист безпеки WIPS, яка на 100% призначена для сканування ефіру й захисту бізнесу від бездротових загроз.

Wi-Fi Cloud від WatchGuard інтегрується зі сторонніми контролерами, що використовують CIP

Технологія Cloud Integration Point (CIP) дозволяє інтегрувати WatchGuard Wi-Fi Cloud зі сторонніми локальними бездротовими контролерами, службами керування й журналом подій, такими як:

- Aruba Mobility Controller.
- Cisco Wireless LAN Controller (WLC).
- HP Multi-Service Mobility (MSM) Controller.
- Arcsight Enterprise Security Management (ESM).
- Syslog server.

Інтеграція зі сторонніми контролерами Wi-Fi (Aruba, Cisco, HP) дозволяє Wi-Fi Cloud одержувати інформацію про пристрої, керовані контролером. Wi-Fi Cloud використовує цю інформацію для класифікації бездротових систем запобігання вторгнень (WIPS) і відстеження місця розташування пристроїв.

Інтеграція із серверами Enterprise Security Management (ESM), такими як Arcsight і Syslog, дозволяє Wi-Fi Cloud відправляти події й повідомлення журналу аудита на ці сервери. Можна використовувати власну існуючу інфраструктуру для керування подіями Wi-Fi Cloud і реєстрації повідомлень.

Інтеграція Wi-Fi Cloud зі сторонніми системами дає наступні ключові переваги безпеки Wi-Fi Cloud, при цьому продовжуючи використовувати існуючу інфраструктуру, не міняючи її:

- Автоматична класифікація WIPS авторизованих пристроїв, керованих контролером.
- Додаткові входи, для відстеження місця розташування клієнтів Wi-Fi
- Можливість відправляти інформацію на центральний сервер журналів для уніфікованого перегляду моніторингу подій і аналізу журналів для усунення неполадок.

Схема роботи Cloud Integration Point (CIP)

Коли Wi-Fi Cloud інтегрується з локальними системами, однієї із ключових проблем є те, що вони звичайно перебувають у приватній мережі за файєрволом. Можна інтегрувати кілька локальних систем з Wi-Fi Cloud, коли використовується пристрій Cloud Integration Point (CIP) у мережі.

Мережне устаткування Aruba

Aruba, a Hewlett Packard Enterprise company – лінійка провідного й бездротового устаткування для мереж передачі даних корпоративного рівня.

Додавання HPE з'явилося в її назві в 2015 році, к оли компанія Aruba Networks, відома до того на ринку 13 років, була куплена гігантом Hewlett Packard Enterprise і стала окремим великим напрямком бізнесу компанії.

Порівняно молодий поки гравець в області корпоративного мережного устаткування на російському ринку вже має міцні позиції в цій сфері за рубежем.

У число лідерів компанія ввійшла завдяки простоті реалізації застосунку, наступного обслуговування й керування розширеним функціоналом мереж Wi-Fi на устаткуванні Aruba.

От основні особливості устаткування Aruba, які актуальні для Україні й дійсно виділяють цю лінійку серед аналогічного устаткування інших виробників:

– Функціональний, налагоджений і масштабований застосунок Aruba Instant – мережа Wi-Fi без виділеного контролера.

– На більшість комутаторів діє довічна гарантія – і вона повноцінно працює в Україні.

– На маршрутизатори й комутатори не потрібно здобувати ліцензії, увесь функціонал доступний відразу.

Точки доступу Aruba

Точки доступу Aruba забезпечують швидку й стабільну бездротову передачу даних у корпоративній мережі. Aruba пропонує як прості пристрої для надання базового функціонала мережі Wi-Fi, так і дуже потужні для мереж з

високою щільністю користувачів і швидкостями до 1,73 Гбіт/с, а також захищені моделі для роботи в суворих умовах, наприклад, під відкритим небом.

Залежно від масштабів проекту можна вибрати:

– Instant-точки доступу – вони не вимагають окремого контролера, а використовують віртуальний контролер, функції якого виконує одна із точок;

– точки доступу, що працюють під керуванням бездротового контролера.

Застосунок також може бути доповнене точками доступу Remote-Access, призначеними для роботи у вилучених філіях і здатними самостійно встановлювати захищене з'єднання із центральним офісом.

Точки доступу прості в інсталяції, можуть працювати одночасно в діапазонах 2,4 і 5 ГГц, використовують технологію MIMO для збільшення швидкості передачі даних і підтримують стандарт 802.11ac Wave2.

Контролери Aruba

Контролери Aruba забезпечують централізоване керування бездротовими точками доступу. На додаток до своєї основної функції вони підтримують WIDS/WIPS, можуть виступати в ролі VPN-концентраторів, а також міжмережевих екранів з підтримкою трансляції мережних адрес (NAT) і фільтрації на рівні застосунків (Layer 7).

Представлені моделі контролерів як для невеликих офісів, що підтримують до 16 точок доступу, так і для великих проектів – до 2000 точок доступу й до 32 тис. клієнтів. Контролери можна поєднувати, збільшуючи загальну ємність точок доступу й клієнтів.

Маршрутизатори HPE

Маршрутизатори для мережних застосунків поставляються під брендом Hewlett Packard Enterprise. Маршрутизатори HPE діляться на дві групи: HSR і MSR.

Лінійка HSR – це високопродуктивні модульні пристрої, що підходять для великих проектів. Топова модель лінійки – маршрутизатор HSR6808, що забезпечує продуктивність до 2 Тбіт/с шасі, що й має, висотою 20 RU.

Лінійка MSR – це маршрутизатори початкового й середнього рівня. Сюди входять компактні моделі MSR9xx, у тому числі з підтримкою Wi-Fi і 3G/4G будинку, що підходять для установки в невеликих філіях організацій і.

Для невеликих і середніх компаній розроблені серії MSR1000, MSR2000, MSR3000 і MSR4000 із продуктивністю від 500 Мбіт/с до 24 Гбіт/с, є й модульні версії. Усі моделі цих серій підтримують широкий набір мережних протоколів і технологій, включаючи динамічну маршрутизацію, NAT, QoS, VPN, міжмережеві екрани й телефонію. Увесь функціонал доступний «у базі», без необхідності додатково здобувати ліцензії. Крім того, є версія, призначена для розгортання в середовищі віртуалізації.

Комутатори Aruba

Великий портфель комутаторів Aruba дозволяє підібрати оптимальне по функціоналу й ціні застосунок для завдання будь-якого масштабу. Є моделі Layer 2 і Layer 3, модульні й у фіксованій конфігурації. Увесь функціонал доступний відразу, не потрібно здобувати додаткові ліцензії.

На роль комутаторів ядра відмінно підходять модульні комутатори 540xr, які залежно від версії шасі підтримують установку до 12 лінійних карт і мають повний функціонал рівня Layer 3.

З комутаторів фіксованої конфігурації можна відзначити комутатор Layer 3 серії 3810. У корпусі 1 RU пропонуються версії з 24 і 48 портами 10/100/1000 Smart Rate multi-gigabit Ethernet з PoE і без, слот для ап лінк-модуля з інтерфейсами SFP+ і QSFP+, а також повністю SFP+ комутатор.

Лінійка Flexnetwork відрізняється розширеною підтримкою протоколів динамічної маршрутизації, IPv6 і MPLS. Крім того, вендор пропонує повністю некеровані комутатори (14xx) і комутатори з веб-інтерфейсом (16xx, 18xx, 19xx).

Програмні застосунки Aruba

Airwave – система керування й моніторингу мережі

Система дозволяє IT-персоналу відслідковувати стан і завантаження компонентів інфраструктури (точок доступу, контролерів, комутаторів), вибирати конфігурацію існуючих і додавати нові пристрої.

Рисунок 2.2 – Інтерфейс користувача Airwave

Airwave також дозволяє одержувати інформацію про підключених до мережі користувачах, збирати статистику про їхні дії й трафікі. Є зручні інструменти візуалізації (відображення інформації із прив'язкою до карти місцевості, різні варіанти графіків і діаграм). Більшість даних можна не тільки переглядати в реальному часі, але й зберігати в балках, щоб потім при необхідності створювати звіти.

Ще одна важлива й корисна функція – планувальник зони покриття мережі Wi-Fi. Маючи плани приміщення з інформацією про розміри й застосовуваних матеріалах, з його допомогою можна вибрати оптимальні місця установки точок доступу й розрахувати зони покриття. Таке планування обов'язкове проводиться на початковому етапі будь-якого проекту по впровадженню Wi-Fi.

Clearpass – сервер керування політиками доступу користувачів у мережу й до конкретних мережних ресурсів

Це стосується як бездротових користувачів (у їхньому випадку контроль доступу завжди особливо актуальний), так і провідних. За допомогою Clearpass адміністратор визначає, хто зможе підключатися до мережі, як будуть проходити автентифікацію ті або інші користувачі: уведуть постійний пароль, одержать разовий пароль по SMS, звернуться до офіс-менеджерові, який на спеціальній сторінці в кілька клацань мишкою згенерує їм пароль, або ж пристрій користувача буде автентифіковано по сертифікату без участі власника.

Після того, як користувач пройшов автентифікацію в мережі, Clearpass авторизує його, причому правила можна набудовувати надзвичайно точно: користувач буде мати доступ строго до тем ресурсам, які дозволені конкретно йому.

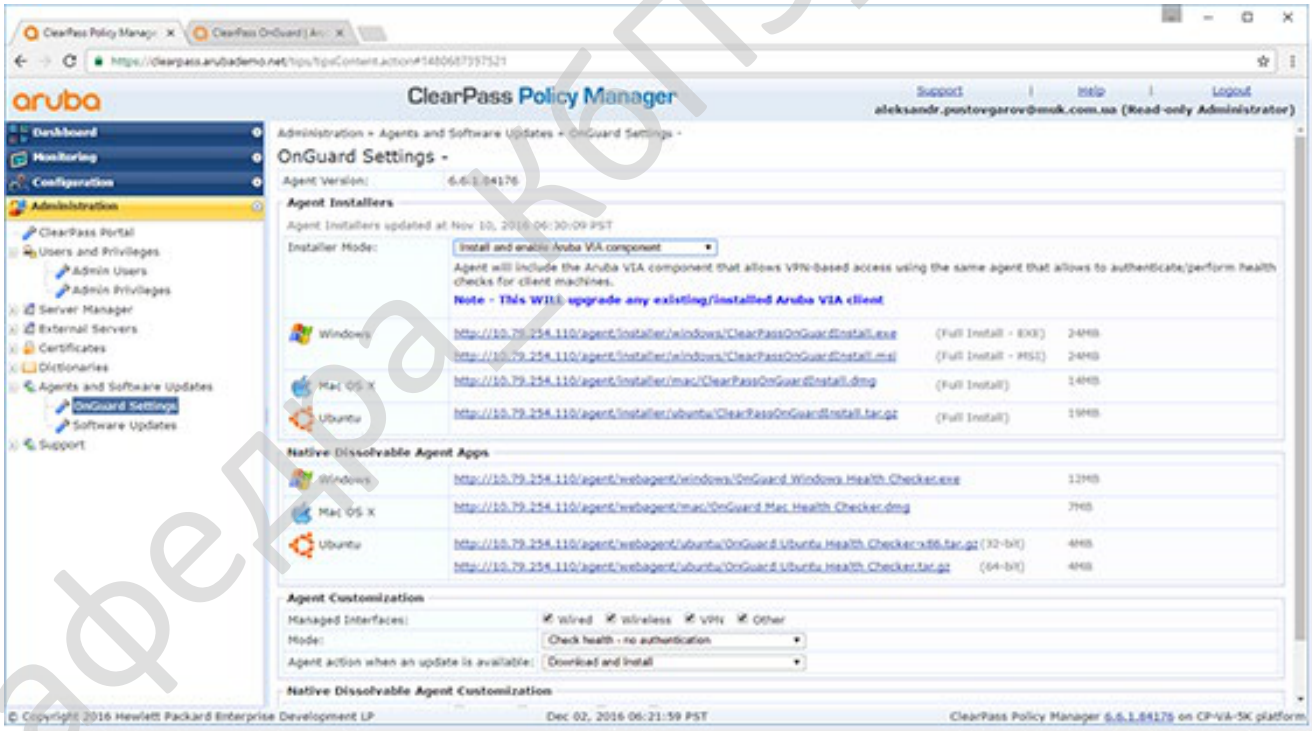

Рисунок 2.3 – Інтерфейс користувача Clearpass

Clearpass дозволяє створювати й легко застосовувати дуже гнучкі політики, що комбінують безліч правил і параметрів: наприклад, автентифікацію користувача й машини, використовувані протоколи, тип і стан клієнтського пристрою, приналежність користувача до тієї або іншій групі й багато чого іншого.

Інша особливість Clearpass – профілювання, яке дозволяє визначати тип і виробника підключеного до мережі пристрою. Дану інформацію можна використовувати і як додатковий параметр у політиках доступу, і для роботи із пристроями, що не підтримують ніякі засоби автентифікації (це, наприклад, багато моделей принтерів). Усе сказане справедливо й для бездротових, і для провідних користувачів.

Ще одна важлива функція Aruba Clearpass – підтримка концепції BYOD (Bring Your Own Device), згідно з якою співробітники можуть використовувати для роботи свої особисті мобільні пристрої. При цьому забезпечується доступ до всіх необхідних мережних ресурсів, а безпека корпоративної мережі не порушується, оскільки доступ користувачеві надається тільки після того, як його пристрій перевірений на відсутність вірусів або jailbreak і наявність усіх актуальних відновлень безпеки.

Як показують проведені дослідження, можливість використовувати для роботи в офісі й за його межами особисті пристрої підвищує лояльність і продуктивність співробітників і до того ж дозволяє компанії заощадити.

Meridian – платформа для визначення місця розташування й навігації в приміщеннях

Застосунок дозволяє визначати місце розташування мобільного пристрою користувача на основі даних мережі Wi-Fi, доповненої маячками Bluetooth Low Energy (BLE).

Aruba Meridian допомагає компаніям, що працюють у сфері обслуговування, набагато ефективніше взаємодіяти зі своєю аудиторією. Використовуючи простий і зручний інструмент, компанія може підготувати й

опублікувати в магазинах застосунків Apple і Google свій власний застосунок для навігації по будинкові. Установивши його на свій смартфон, користувач буде легко орієнтуватися в будинку, бачачи своє поточне положення й одержуючи наочні підказки, як добратися до тієї або іншої точки на карті. Залежно від його місця розташування він також при бажанні буде одержувати через застосунок push-повідомлення, наприклад, про різні знижки й акціях.

Ця технологія стає усе більш популярної у світі й уже знайшла застосування на великих об'єктах, таких як стадіони, університети, торгові центри і т.д.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, MAC OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для MACOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для MACOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

– У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на MACOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion,

навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для MACOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для MACOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO)

варіанти установки. Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки WIPS для WiFi.

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

Хоча в наші дні це часто вважається саме собою, що розуміють, Wi-Fi – це не що інше, як технологічне чудо. Він забезпечує максимальну зручність у мережі і його майже чекають, куди б ви не пішли.

Проте, ця технологія як і раніше викликає особливі побоювання із приводу кібербезпеки, навіть через 20 років після того, як вона стала мейнстримом.

WIPS (бездротові системи запобігання вторгнень) – це застосунок багатьох із цих проблем, тому давайте досліджуємо проблеми мережної безпеки, які переслідують Wi-Fi, і те, як гарний WIPS може від них захиститися.

Коли Wi-Fi став широко поширюватися приблизно в 1999 році, він видався вищим з погляду технологічної зручності. Видалося, майбутнє нарешті настало. Незважаючи на те, що це все ще об'єктивно вражаючий подвиг, за минулі два десятиліття багато чого змінилося, особливо з погляду ландшафту мережних загроз.

Кіберзлочинність за останні 10 років, не говорячи вже про 20, виросла в геометричній прогресії, і хакери постійно знаходять нові й цікаві способи злому мереж, крадіжки даних і, у цілому, руйнування. Однак з погляду безпеки мережі Wi-Fi практично нічого не змінилося. Ми очікуємо якесь з'єднання Wi-Fi, куди б ми не пішли, але в бездротовій мережі немає нічого безпечного.

Деякі можуть заперечити, що шифрування WPA2, яке використовує сучасний Wi-Fi, *є* заходом безпеки. Але ми б посперечалися, що з тих пор, як в 2017 році була виявлена уразливість KRACK, вона не була захищена, що робило шифрування WPA2 практично пошукам.

Але перш ніж ми поглибимося у визначення WIPS, нам необхідно зрозуміти існуючі загрози Wi-Fi.

Кілька сценаріїв безпеки бездротової мережі

Забезпечення Wi-Fi залежить від устаткування доступу, яке відправляє й ухвалює бездротові сигнали, утворюючи майже сферичний діапазон покриття навколо кожної точки доступу (AP).

Розміщення неавторизованої точки доступу в межах цього діапазону, пряме підключення нового обладнання для доступу до вашої фізичної мережі або перекриття покриття Wi-Fi між вами й вашим сусідом може означати початок багатьох проблем кібербезпеки.

Злі подвійні точки доступу

Хакери можуть створювати свої власні точки доступу в межах вашої мережі Wi-Fi, які виглядають як прямий клон вашої власної легітимної точки доступу; називають «злим близнюком».

Член команди, що нічого не підозрює, який не використовує безпечний VPN, або відвідувач може випадково підключитися до цієї мережі – при цьому вони думають, що підключені до вашої мережі, хоча насправді вони підключені до хакерської точки доступу 4/5G.

Таким чином, цей злочинець буде мати повний контроль над даними, переданими через це з'єднання – він може украсти платіжні реквізити, одержати конфіденційні облікові дані, впровадити шкідливий код, установити шкідливе ПЗ й багато чого іншого.

Несанкціоновані або незахищені точки доступу

В іншому сценарії стандартне устаткування для доступу до Wi-Fi може бути підключене до вашої мережі без авторизації. Це може бути зовсім безневинно – команді потрібно краще покриття Wi-Fi, тому вони встановлюють точку бездротового доступу, куплену у великого роздрібного продавця, не думаючи про наслідки для безпеки.

Однак внутрішній суб'єкт із невдоволенням може так само легко настроїти незахищену точку доступу, щоб сформувати незахищений «бекдор» у мережі. Це

може привести до крадіжки даних, поширенню шкідливого ПЗ й може стати вразливим місцем у захисті вашої мережі.

У якості альтернативи технічному фахівцеві так само легко встановити нову точку доступу й забути правильно її настроїти для забезпечення оптимальної безпеки. У будь-якій ситуації хакер може виявити ці слабкі місця, використовувати їх для доступу до вашої мережі й викликати хаос.

Зовнішні точки доступу

Проблеми також можуть виникнути, коли пристрою, які звичайно підключаються до вашої внутрішньої мережі Wi-Fi, використовуються для підключення до сусідніх відкритих мереж у радіусі дії, можливо, для обходу вашої внутрішньої мережі або політик контенту.

Оскільки ви не знаєте, наскільки строгі заходи Wi-Fi вашого сусіда, це може відкрити двері для стеження, атак типу «людей посередині» (MITM) і багато чого іншого. Фактично, будь-який пристрій, який раніше підключався до незахищеної, незахищеній або явно зловмисній точці доступу, може містити всілякі неприємності, включаючи шкідливі програми, програми-здирники й бекдори доступу, що створює реальну загрозу для будь-якої мережі, до якої вони підключаються згодом, включаючи вашу..

Якщо ви думаєте, що все це звучить досить тривожно, ви праві. Але є один застосунок: інвестувати в WIPS або бездротову систему запобігання вторгнень.

Система запобігання вторгнень у бездротові мережі – це система безпеки для бездротових мереж. WIPS контролює радіочастотний спектр у повітряному просторі бездротової мережі на предмет несанкціонованої або несподіваної активності й частот.

Система самостійно визначає небезпечну активність і може автоматично її виключити. Сучасні WIPS часто йдуть далі простого частотного аналізу, включаючи класифікацію відомих бездротових пристроїв, каталогізацію їх

унікальних шаблонів сигналів і, можливо, багато чого іншого залежно від обраного вами застосунку.

Впровадження WIPS не тільки допомагає уникнути дорогих кіберінцидентів, але й допомагає дотримувати таких стандартів і норми, як GDPR, PCI DSS і Cyber Essentials.

Гарні системи WIPS можуть виявляти всі види бездротових загроз і уразливостей, включаючи неавторизоване або неправильно налаштоване устаткування; злі близнюки AP; спроби стеження й багато чого іншого.

Як розгортаються WIPS

Багато сучасних установок WIPS використовують так званий «інтегрований WIPS». Саме тут у ваші точ ки бездротового доступу вбудований постійний моніторинг і запобігання бездротових вторгнень, що забезпечує повне захисне покриття кожного кубічного сантиметра повітряного простору вашої мережі.

Є два інших важливих способу досягнення WIPS. На більш дешевому кінці спектра перебувають системи WIPS з функціями періодичного моніторингу, вбудованими в їхні точки доступу. Кіберзлочинність може завдати удару в будь-який момент, тому періодичне сканування безпеки може залишити вас уразливим.

На більш дорогому кінці спектра є «оверлейні» системи WIPS, які включають розгортання автономних датчиків WIPS – окремо від ваших точок доступу – які постійно контролюють бездротові частоти мережі.

Це додаткове устаткування може мати більш високу вартість, але насправді воно може виявитися досить доступним у порівнянні з копіюванням і заміною існуючого устаткування доступу. Однак також урахуйте, що, коли датчики відділені від точок доступу, існує ризик того, що частина повітряного простору вашої мережі залишиться без контролю.

Маючи це у виді, ми вважаємо, що інтегровані системи WIPS пропонують найкращий баланс вартості й функціональності.

Який би тип бездротової системи запобігання вторгнень ви не вибрали, вона, як правило, відправляє звіти в якийсь інтерфейс керування, щоб ваші технічні фахівці могли відслідковувати зареєстровані порушення, перевіряти попередження й вручну захищати ваші радіохвилі.

WIPS також можуть повідомляти про експлуатаційні показники мережі, таких як використання й продуктивність, допомагаючи вашим технічним фахівцям підтримувати працездатність і час безперебійної роботи мережі поряд з безпекою.

3.2 Розробка структурної схеми

Як випливає з назви, бездротові системи запобігання вторгнень використовують політики й мережні елементи керування для запобігання доступу зловмисників до мережних ресурсів. Коли WIPS виявляє щось схоже на шахрайське устаткування, вона відправляє попередження адміністраторові мережі, який може визначити, чи дійсно це є загрозою, і, якщо так, те допоможе ізолювати цю небезпеку, перш ніж вона нанесе занадто велику шкоду.

Це означає, що WIPS необхідно вбудовувати в широкий діапазон середовищ і, в ідеалі, вимагати мінімальну деталізацію конфігурації устаткування, що обслуговується. Деякі із пропонованих на ринку застосунків WIPS використовують локальний пристрій (або доступ до мережного операційного центру) для моніторингу трафіку й / або роботи як маршрутизатора / міжмережевого екрана / комутатора. Але, коли ви зустрічаєтеся з необхідністю здійснення моніторингу великої кількості бездротових зон, цей спосіб може стати стомлюючим, дорогим і трудомістким. Виконання аналізу загроз і ініціалізація точок доступу в хмарі значно спростить і автоматизує вам цей процес.

Однією з головних причин, по якій мережі, керовані WIPS, стають усе більш популярними, є те, що вони можуть використовувати вашу існуючу інфраструктуру, – вам не потрібно для цього переходити на використання

новітніх технологій, застосовуваних у точках доступу, або чекати, поки ваші системні партнери обновлять свої системи безпеки й автентифікації. Тому, коли згадується про WIPS, те більше ідеться саме про захищеність бездротової мережі, а не про вибір імені бренда для побудови мережної інфраструктури.

Можливості WIPS_Wifi, який розроблений в даній роботі

Архітектура WIPS поєднує точки доступу, які також можуть бути бездротовими маршрутизаторами (з додатковою підтримкою DHCP), із хмарним програмним движком, який використовується як для активації цих точок доступу, так і керування ними. Таким чином, ви можете набудовувати конфігурацію точок доступу, робити моніторинг трафіку, вибирати методи автентифікації, аналізувати продуктивність і т.д.

Застосунок WIPS_Wifi, який розроблений в даній роботі, пропонує всю базову функціональність, яку ви очікуєте від WIPS. Системні адміністратори можуть одночасно управляти багатьма пристроями й одержувати попередження при виконанні певних умов. Вони можуть збирати інформацію про авторизованих відвідувачів і гостях (включаючи дані, які потенційно можуть бути використані для маркетингових цілей). Застосунок дозволяє використовувати одні й ті точки доступу як хост-пристрої для декількох ідентифікаторів мережі SSID з різними рівнями авторизації. І, що найголовніше, точки доступу під управлінням застосунку WIPS_Wifi, який розроблений в даній роботі, можуть ідентифікувати потенційні небезпеки й автоматично їх пригнічувати або залучати до них увага системних адміністраторів.

Точки доступу, як тільки будуть налаштовані, продовжать виконувати свої завдання, навіть якщо будуть відключені від хмарної консолі WIPS_Wifi, який розроблений в даній роботі. Але, поки вони знову не приєднаються до материнської хмарної платформи, ви не зможете скористатися всією вигодою, яку надає аналіз даних на серверному рівні.

Робота з Wi-Fi-мережами за допомогою WIPS здійснюється через хмарну панель моніторингу по двом напрямкам: «Сервіси» (конфігурація, аналіз і

виявлення потенційних загроз, конфігурація WIPS, адміністрування) і «Застосунки» (екрани виявлених шлюзів, аналіз застосунків, додаткового програмного забезпечення і т.д.).

Серед переваг WIPS_Wifi, який розроблений у даній роботі, відзначимо: готовність працювати з багатьма пристроями й технологіями, глибоке розуміння атак, а також дуже просунуті сенсорні можливості. Головним недоліком даного застосунку, на думку фахівців, є єдиний «незграбний» досвід взаємодії й користувацький інтерфейс (User Experience & User Interface, UX/UI).

Як і де влаштовувалася тестова зона

Точки доступу й хмарний застосунок WIPS_Wifi, який розроблений в даній роботі були протестовані в бетонному будинку. У всіх тестуємих зонах була розгорнута точка доступу відповідно до звичайних «правил» для розгортання точок доступу стандарту IEEE 802.11ac. Відкритий майданчик був розбитий на шість зон і, відповідно, розгорнуто шість точок доступу, що використовують для підключення Ethernet-кабель, а також підтримуючих технологію Power Over Ethernet (PoE) –, що забезпечує передачу електричної енергії разом з даними через стандартну кручену пару. Точки доступу були підключені до 8-портовому L2 / L3 комутатору виробництва компанії HP, а потім до волоконно-оптичної магістральної мережі Gigabit Ethernet, що забезпечує доступ до Інтернет.

У зоні дії тестуємо бездротової мережі було виявлено більш 80 інших точок доступу, а також сотні клієнтських пристроїв. Крім того, над обраним для тестування місцем розташовувалися два працюючі передавачі: 250-ватний передавач комерційної Fm-радіостанції і його 900 МГц канал службовому зв'язку. У зоні прямої видимості також працювали ще чотири 150 МГц передавача. Рівень шуму в діапазонах Wi-Fi склав -99 дБм для 2,4 ГГц і -113 дБм для 5 ГГц. Інакше кажучи, було дуже галасливо.

У ході тесту було використано сім «шахрайських» пристроїв для емуляції атак із шахрайською точкою доступу (rogue access point), відмови від асоціації

(disassociation) і відмови в обслуговуванні (Denial of Service, Dos). Використовуване устаткування складалося з декількох комутаторів Linksys і Netgear з перевстановленим прошиванням, які були впроваджені у звичайну конфігурацію Wi-Fi. Крім того, у ряді випадків для проведення більш розширених атак використовувалося програмне забезпечення DD-WRT. Також два ноутбуки на базі операційної системи Linux (Lenovo Yoga і x120e) були перероблені для здійснення атак на стандарти безпеки WEP і WPA. У ході тесту були перевірені можливості здійснення підключення до ноутбуків Lenovo, Samsung і Apple з оновленими версіями операційних систем Windows 10, Linuxmint 18 і MACos 12.12, відповідно.

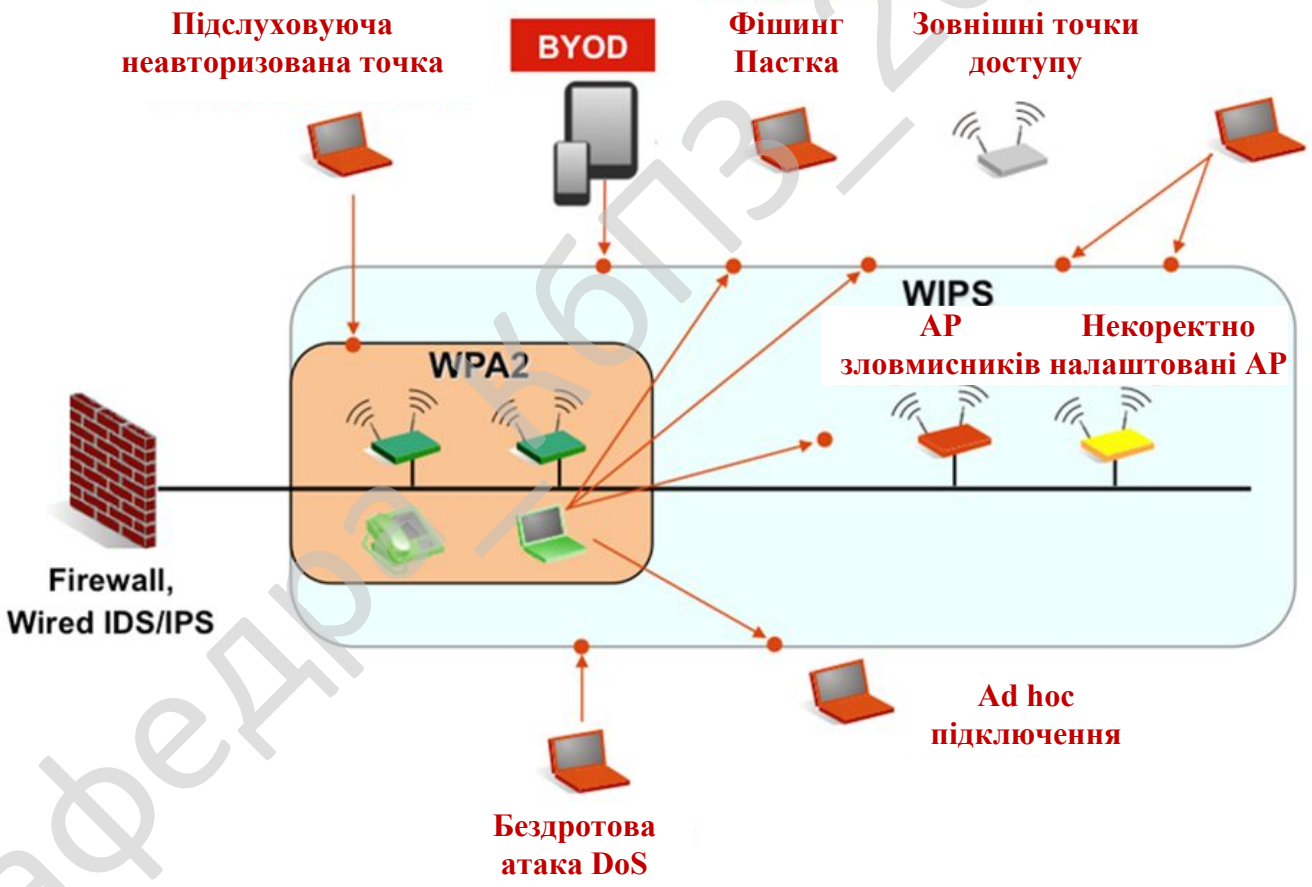

Рисунок 3.1 – Структурна схема системи

П'ять смартфонів (Apple iPhone, Samsung, Asus і Motorola/Lenovo) було використано для перевірки статусу шахрайських клієнтів. За допомогою

телефонів Samsung і Motorola (у тому числі з використанням ноутбуків) були здійснені спроби атак «людей посередині» і відмови від асоціації, а також проведені різні тестові спроби крадіжки мережних облікових даних.

3.3 Розробка функціональної схеми

Кожна точка доступу, використовувана в тестовому експерименті й керована за допомогою хмарного застосунку WIPS_Wifi, який розроблений у даній роботі, підтримувала три режими роботи: два з них безпосередньо обслуговують частотні діапазони Wi-Fi 2,4 і 5 ГГц, відповідно, а третій – служить для спостереження за навколишнім радіоефіром з метою спостереження за трафіком і пошуку аномалій. Третій режим роботи є ключовим елементом у спостереженні за радіочастотним середовищем і, отже, відповідає за здатність запобігати вторгненням. Співробітники компанії WIPS_Wifi, який розроблений в даній роботі називають цей режим «третім оком», тому що він дозволяє зрозуміти дійсні причини збоїв перших двох режимів роботи. Це незвичайна, але не зовсім унікальна функціональність.

Тестуємі точки доступу були приєднані до мережного сегмента з оптоволоконним доступом в Інтернет і опціонально до VLAN. Звідси точки доступу WIPS_Wifi, який розроблений у даній роботі, з підтримкою PoE споконвічно завантажують із сайту WIPS_Wifi, який розроблений у даній роботі, апаратно-програмне забезпечення з політиками, а також інформацією про мережу й автентифікації, призначеними адміністраторами бездротової мережі. Кожна точка доступу може підтримувати кілька ідентифікаторів мережі SSID, кожний з яких має свою власну конфігурацію й політики WLAN. Залежно від методу автентифікації кожний із цих ідентифікаторів мережі може підтримувати різну функціональність, прив'язану до кожного SSID, включаючи, що налаштовується доступ до зовнішніх ресурсів через автентифікацію в соціальних мережах або сторінку реєстрації. Незважаючи на те, що кожній точці доступу можна задати

свої власні детальні налаштування, загальні шаблонові конфігурації підійдуть більшості організацій. Як відзначають автори дослідження, шаблони за замовчуванням дійсно детально пророблені й легко модифікуються.

Як і очікувалося, розгорнута Wi-Fi-мережа підтримувала роботу з усіма стандартами IEEE 802.11 a, b, g,n і ac. Точки доступу під управлінням застосунку WIPS_Wifi, який розроблений у даній роботі, здатні були виявляти точки доступу, а також клієнтські й інші бездротові Wi-Fi-пристрою із силою прийнятого сигналу до -96 дБм, що під час тесту було підтверджено аналізатором компанії NETSCOUT, чий поріг чутливості як тестового інструмента був трохи вище.

Консоль Dashboard (Панель моніторингу)

Налаштування конфігурації точок доступу відбувається через панель моніторингу хмарного порталу WIPS_Wifi, який розроблений у даній роботі. Якщо точка доступу може дотягтися до хмарних ресурсів WIPS_Wifi, який розроблений у даній роботі, вона завантажить задану для неї конфігурацію й перевантажиться вже з використанням цих налаштувань. Цей процес дуже схожий на установку netboot/Pxeboot для прикінцевих точок доступу. У ході тестування все працювало добре, за винятком однієї деталі – установка нових налаштувань і, відповідно, перевантаження точки доступу зайняли значно більше часу, чим очікувалося. Це, мабуть, одна з деяких претензій до функціонування застосунку в цілому, які можна пред'явити розроблювачам.

Панель моніторингу є, що адміністративно налаштовується, а її компоненти моніторингу розділені на чотири розділи:

- «Мережа» (Network).
- «Точки доступу» (Access Points).
- «Клієнти» (Clients).
- WIPS.

Для роботи з Dashboard ми рекомендуємо використовувати монітор з високим дозволом, щоб ви змогли бачити й контролювати більшу частину

необхідної вам інформації. Для цих цілей погано підходить ноутбук, тому що через великий кількості інформації навіть базових налаштувань текст на ньому стає тяжкочитаємим. Для більш розширених установок використання декількох моніторів стане для вас обов'язковою умовою.

Робочою частиною панелі моніторингу є Wireless Services Manager. Це сама більша робоча область адміністративного контролю в Dashboard, де ви проведете більшу частину свого часу при налаштуванні застосунку.

Панель моніторингу дозволяє стежити за «Керованими пристроями» або Managed Devices в англомовному варіанті (а саме – за керованими точками доступу), Wi-Fi (точками доступу, радіоефіром, клієнтами й бездротовими локальними мережами), «Безпекою» або Security (у центрі уваги – точки доступу, клієнти й мережі), «Додатками» або Applications (протоколами, конкретними додатками, такими як, приміром, Facebook, Amazon, XMPP (протокол обміну повідомленнями, раніше відомий як Jabber), FTP і т.д.).

Виявлення бездротових пристроїв

У ході тесту відразу були перевірені керовані пристрої, а також налаштування Wi-Fi і WLAN. Після цього увага була зосереджена на вкладці Security, яка містила багато цікавих даних, включаючи довгий список того, що перебувало поблизу й використовувало радіоефір. Дані пристрої були класифіковані як:

- «Авторизовані» (Authorized).
- «Невірно налаштовані» (Misconfigured).
- «Шахрайські» (Rogue).
- «Зовнішні» (External).
- «Знайдені, але некатегоризовані» (Found but Uncategorized).

Крім шести розгорнутих точок доступу тестуємий застосунок виявив у безпосередній близькості від бездротової зони десятки інших точок доступу, що розташувалися на прилеглій території: студентському містечку коледжу, офісних будинках і торговельних майданчиках, а також навіть у людей, що пересуваються

по вулиці з телефонами або іншими пристроями із включеним режимом вільного доступу. Цікавий, але сумний факт: у прилеглих мережах був виявлений також ряд незахищених Wi-Fi-принтерів. У середньостатистичний день більш 80 точок доступу розташовувалися поруч із іспитовим полігоном, були виявлені численні незахищені пристрої, у тому числі кілька сотень клієнтів, а також величезна кількість минаючих або проїжджаючих повз людей, чиї пристрої перебували в постійній «полюванні» за точками доступу з метою приєднатися за замовчуванням до відкритої мережі.

Безпека

У вкладці «Безпека» користувацького інтерфейсу відображаються різні пристрої, для зручності виділені різними кольорами:

- відомі точки доступу позначаються зеленим;
- клієнтські гостьові пристрої – ясно-зеленим;
- певні шахрайські клієнти або точки доступу – червоним;
- синій колір у виявлених зовнішніх пристроїв;
- білий – у виявлених, але некатегоризованих пристроїв, що очікують оцінки адміністратора.

Ви можете зіставити дані, одержувані від точок доступу, з картою території або кресленням будинку, щоб побачити, де фізично розташовані різні бездротові пристрої, такі як клієнти. Фактично, це дозволяє вам практично в режимі реального часу відслідковувати, приміром, переміщення смартфонів користувачів, коли вони перетинають границю зони покриття ваших точок доступу; і, звичайно ж, ідентифікувати місце розташування шахрайських пристроїв або джерел проведення атак.

Події

Ліквідовані зловмисники відображаються у вкладці «Події» (Events). Як тільки з'являється точка доступу з підозрілою поведінкою – це пристрій попадає в список потенційних шахрайських точок доступу. Неправильні спрацьовування, коли в цей список, приміром, може бути випадково віднесена будь-яка людина зі

смартфоном, що намагаються поділитися своїм підключенням до Інтернету, роблять складніше роботу із запобігання вторгнень і експертний аналіз через «забруднення» списків неправильними спрацьовуваннями. Цей факт може зробити вашу роботу досить стомлюючої. Однак ця надмірна строгість виявляється життєво необхідної організаціям, які опікуються про безпеку своєї інформації.

Строгі правила автентифікації й визначення шахрайських пристроїв дуже важливі, тому що інфраструктура застосунку WIPS_Wifi, який розроблений в даній роботі, намагається виявити інші точки доступу в основній або корпоративній мережі. Якщо вимоги виконуються, ці пристрої вважаються неворожими. Якщо немає – тоді ці точки доступу автоматично вважаються потенційно ворожими, що гарантує вам те, що можливі атаки типу «людей у середині» або інші атаки на вашу бездротову мережу не будуть завершені.

Під час тестування дослідники використовували як програмні емуляції, так і фактичні шахрайські точки доступу, підключаючи їх до мережі, щоб перевірити, як хмарна логіка застосунку бездротової системи запобігання вторгнень WIPS_Wifi, який розроблений у даній роботі, може з ними впорається. Усі атаки були відвернені, і весь процес зайняв менше хвилини, хоча, звичайно, ми б віддали перевагу, щоб це відбувалося миттєво.

При виявленні шахрайських точок доступу й клієнтів подається сигнал тривоги. Як ми вже відзначали раніше, фізичне місце розташування потенційного шахрайського пристрою може бути відображене на карті місцевості.

Моніторинг і аналіз загроз

Контрольовані точки доступу надають різну цікаву інформацію, у тому числі логи подій системного журналу, а також інформацію про надмірну перевантаженість каналів, з якої зустрічаються клієнти деяких точок доступу. Також доступна інформація про трафік, у тому числі його використання різними додатками для кожної точки доступу.

Ви можете здійснювати моніторинг завантаженості Wi-Fi-каналів у зонах дії точок доступу, а також, як у графічному, так і в цифровому виді, відслідковувати інформацію про найбільш слабких прийнятих від клієнтів сигналах, вимірюваних через показник рівня прийнятого сигналу (received signal strength indicator, RSSI) у дБм. Це допоможе виявити самі слабкі сигнали радіозв'язку із клієнтом.

У ході тесту використовувалися три SSID і шість точок доступу, дослідники могли відслідковувати трафік по кожної SSID і способу використовуваної автентифікації. Також було видне, як клієнти розподілені по радіоефіру точок доступу SSID/WLAN. Це, приміром, може допомогти виявити аматорів відео або музики.

Огляд застосунків (Apps Visibility)

Огляд застосунків (Apps Visibility) виявився дуже цікавим інструментом, тому що кожний застосунок (наприклад, Facebook, сторінки SSL/HTTP, icloud і т.д.) мають індекс загрози, які адміністратори можуть перевизначити. Цікаво, що в ході тесту Facebook за замовчуванням був оцінений як дуже висока загроза. За допомогою даного інструментарію у вас буде можливість одержати більш докладний список того, як багато клієнтів (і їх MAC-адрес) звернулися до конкретного додатка, наприклад, Facebook; який обсяг даних вони прийняли й відправили через цей застосунок; яке процентне відношення цих даних до всіх переданих даних цього клієнта; і наскільки часто цей клієнт використовував цей застосунок протягом певного проміжку часу. Таким чином, ви легко можете довідатися, чи користується гіпотетичний Вася соціальною мережею Facebook тільки на перервах, або постійно, з моменту свого приходу й аж до відходу, і наскільки великий обсяг відео, яке він переглядає через соціальні мережі.

Аналітична частина панелі інструментів присвячена виявленню зловмисників. У вас є можливість майже у режим реального часу відслідковувати підозрілі точки доступ (шахрайські, заборонені, Honeypot й некоректно зконфігуровані точки доступ), а також пов'язані з клієнт загроз (неавторизовані,

заборонені, з'єднання, що перериваються, належать бездротовим, що самоорганізуючимся (ad-hoc) мережам, а також ICS / Network. Вони ранжуються за часом і відображаються із прив'язкою до пристроїв або подій.

Крім того, ви можете одержати більш докладну інформацію для кожного елемента в списку, з найдокладнішим описом що відбувся події. От приклад для неавторизованого клієнта:

Unauthorized Client [Apple_9A:00:CF] is connected to Authorized AP. The Client's details are: MAC address [0C:3E:9F:9A:00:CF], user name [--], vendor [Apple], RSSI [-42] dbm. The Ap's details are: MAC address [00:11:74:86:7A:42], protocol [802.11a], channel [44], SSID [WFHB], security setting [802.11i, WPA], vendor [WIPS_Wifi, який розроблений у даній роботі,], RSSI [0] dbm.

Варто відзначити, що тестуємий програмний застосунок дозволяє здійснювати моніторинг Ethernet-кадрів і керуючої інформації набору стандартів 802.11, а не аналізувати корисне навантаження пакетів. Але згенеровані в процесі файли з розрішеням .pcap, якщо це буде необхідно, можуть бути прочитані й оброблені застосунком Wireshark (або будь-яким іншим аналізатором мережі, що використовують бібліотеку Pcap) для усунення неполадок, аналізу спроб злому й виявлення інцидентів, створених сторонніми додатками.

Звіти

Формовані системою виявлення вторгнення WIPS_Wifi, який розроблений в даній роботі звіти дуже сильні й стануть відмінним інструментарієм адміністратора. Вони діляться на:

- «Оцінки» (Assessment).
- «Сумісність» (Compliance).
- «Інциденти» (Incident).
- «Реєстр пристроїв» (Device Inventory).
- «Продуктивність» (Performance).

Умови й розміри тестового майданчика не дозволили дослідникам належним чином протестувати можливості звітів Device Inventory і Performance.

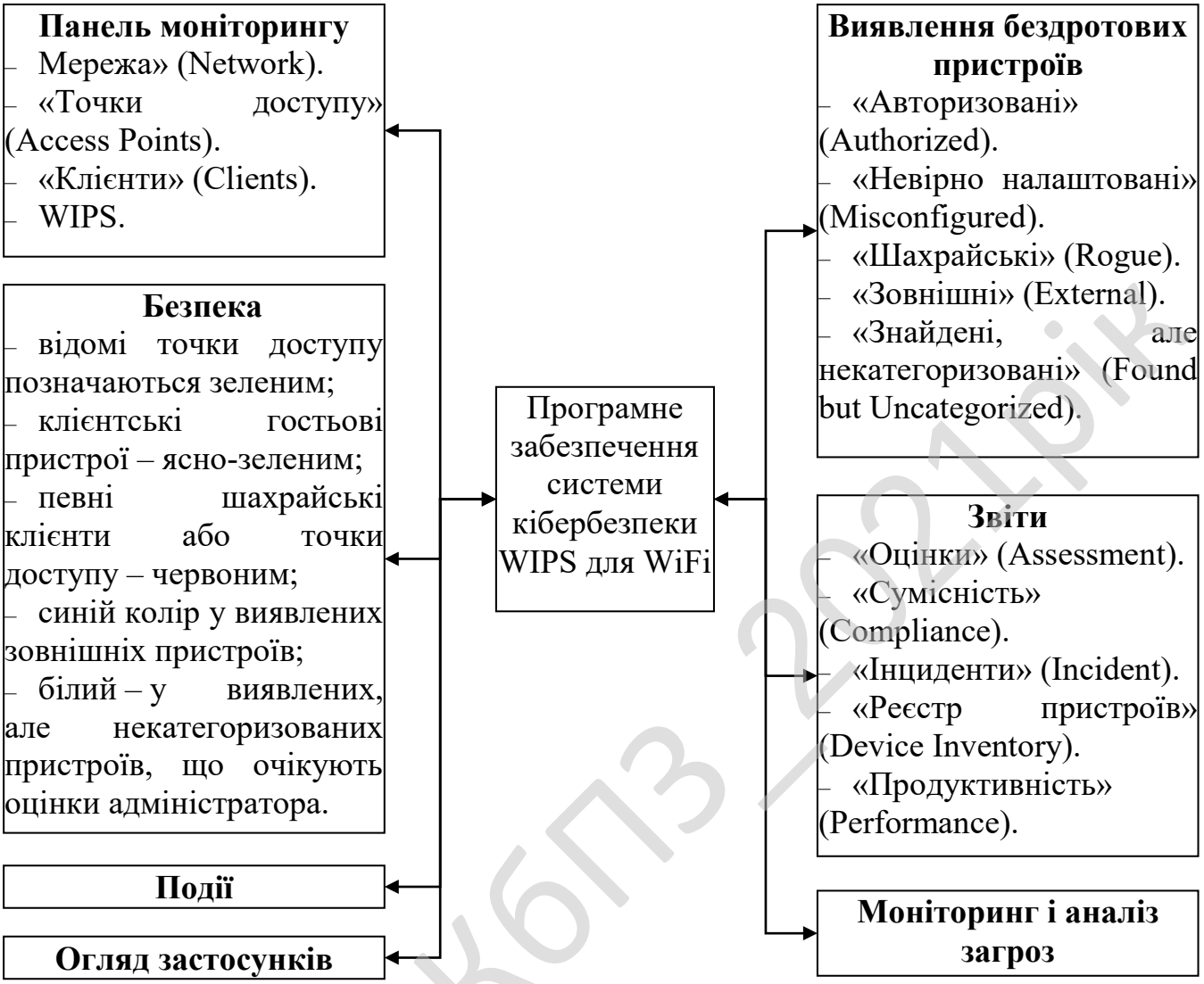

Рисунок 3.2 – Функціональна схема системи

Звіти Assessment включають результати проведення тестового аналізу «Бездротової уразливості» (Wireless Vulnerability), «Ризиків, що виходять від каналів радіозв'язку» (Airspace Risk), а також ряду інших звітів, які можуть бути згенеровані за розкладом, а потім відправлені по електронній пошті.

Звіти Compliance включають Dod Directive 8100.2 Compliance, GLBA Wireless Compliance, HIPAA Wireless Compliance, MITS, PCI DSS 3.1 Wireless Compliance, а також Internal Audit Reports і SOX Compliance Reports.

Ви можете підкоректувати вид усіх звітів (заголовки, поля і т.д.), а потім згенерувати їх відразу або запустити згідно із запланованим розкладом. Це

компенсує час, який довівся б витратити на підтвердження повідомлень, додаючи вручну невеликі примітки до кожного з них.

Недоліки системи

Досвід взаємодії панелі моніторингу WIPS_Wifi, який розроблений у даній роботі, не можна назвати легеням і інтуїтивно зрозумілим. Панель моніторингу містить величезна кількість взаємозалежної інформації, але все-таки це складна ієрархічна система, у якій непросто розібратися.

Через широту, що відкривається, адміністративних можливостей, доступ до аккаунту хмарного сервісу WIPS_Wifi, який розроблений в даній роботі через стандартну авторизацію «ім'я користувача / пароль» видасться вразливим до цілеспрямованих хакерських атак. Автора дослідження також відзначають недостатню, на їхню думку, підтримку застосунком можливостей множинної адресації.

Ще одним недоліком є те, що деякі зміни конфігурації точок доступу можуть зайняти до декількох десятків хвилин від щілінка миші, щоб підтвердити застосування налаштувань, до фактичної їхньої доступності. Як виявилося, більшість змін вимагають перезавантаження точок доступу, яке здійснюється автоматично. І єдиним сигналом про те, що зміни набули чинності, є вказівка про те, що даний сервіс став доступний після невеликого переривання в роботі точки доступу. Як відзначають автори дослідження, одночасне застосування множинних змін конфігурації точок доступу може привести до того, що вся бездротова Wi-Fi-Інфраструктура буде недоступна протягом двадцяти хвилин.

Система виявлення вторгнення WIPS_Wifi, який розроблений у даній роботі, являє собою хмарну систему WIPS, яка добре себе показала при тестуванні. Вона прекрасно впоралася з дуже гучним тестовим середовищем, швидко і якісно зреагувавши на всі можливі загрози безпеки бездротової мережі, штучно створюваними авторами тестового дослідження. По суті, вона робить роботу, яку ми від неї очікуємо, і справляється із цим чудово (за винятком випадків, коли точки доступу довго перевантажуються).

Природно дана система по своєму функціоналу не унікальна, те ж саме може бути реалізоване на Wi-Fi устаткуванні CISCO, ARUBA і ін., при цьому існують системи для забезпечення безпеки Wi-Fi мереж більш високого класу. Це повністю автономні «Wireless IDS/IPS» системи, функціонал яких, як правило, дозволяє більш ретельно відслідковувати загрози в бездротових мережах. Такі системи мають можливість ідентифікувати й визначати сотні різних типів загроз. Але вони обходяться дорожче, через те, що їх потрібно ставити поверх існуючої Wi-Fi мережі.

Показовим прикладом таких більш просунутих систем є, наприклад, Airmagnet Enterprise виробництва компанії NETSCOUT. Це програмно-апаратний комплекс, який не призначений для надання послуг сервісу доступу до бездротової мережі. Грубо говорячи, це не точки доступу, а система, що контролює весь радіоефір на предмет визначення й протидії загрозам корпоративних Wi-Fi мереж. Така система добре справляється із завданням активного придушення Wi-Fi точок доступу й клієнтських пристроїв, не вносячи перешкоди в радіоефір, тому що придушення здійснюється адресно, методом відправлення сервісних пакетів деавторизації на пристрій. Вплинути на роботу такого комплексу зловмисникам практично не можливо через «невидимість» його в мережі, адже він не транслює ніяких SSID по яких його можна виявити, а, отже, і неможливо застосувати до нього Dos атаку або Exploit. Застосунки класу Airmagnet забезпечують більш високий рівень безпеки бездротових мереж організації, тому їх застосовують у першу чергу в компанії, найбільшою мірою, що опікуються про збереження своїх даних: банки, страхові компанії, великі корпорації, держоргани й ін.

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.

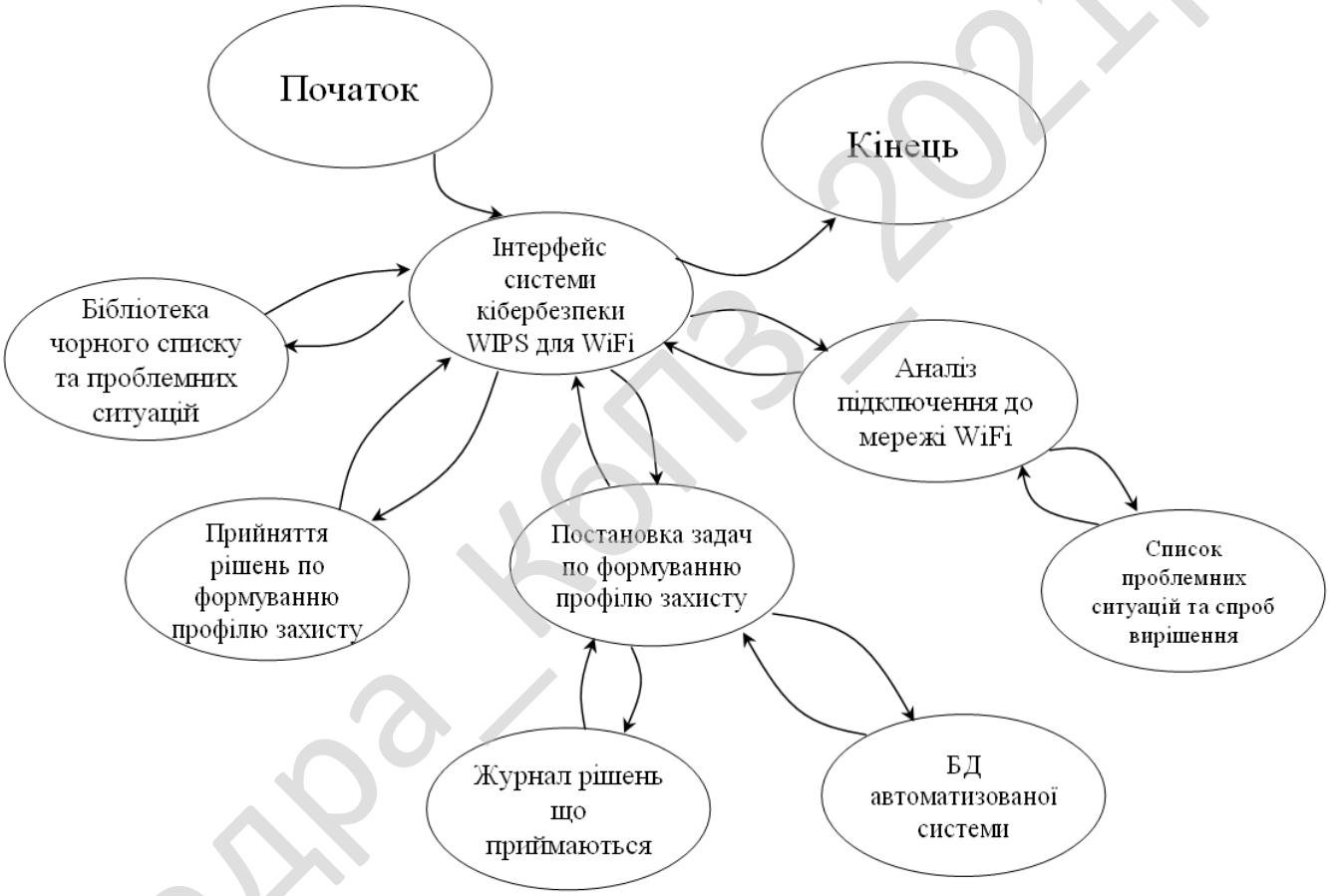

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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.

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

У розділі розглянемо реалізацію бакалаврської дипломної роботи та наведені приклади блок-схем та опис алгоритмів функціонування системи у вигляді частин вихідного коду.

Розглянемо частину вихідного коду, реалізацію що підтверджують вірність прийнятих проектних та програмних рішень, а саме код який показує створення модуля для автоматичної настройки локальної мережі, який дозволяє працювати з маршрутами мережі, пінг, аналіз помилок інтернет-з'єднання, а також отримувати загальні відомості про мережевому обладнанні комп'ютера.

Вихідний код модуля.

```
unit Unit9;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ShellAPI, xpman, WinSock, ComCtrls, ExtCtrls,
  Buttons, Jpeg;

type
  TForm9 = class(TForm)
//…
// Стандартний опис компонентів на форми модуля
//…
  private
    { Private declarations }
  public
    revers1:integer;
    revers2:integer;
```

```
    { Public declarations }
  end;

var
  Form1: TForm9;

implementation
// реалізаційна частина модуля
{$R *.dfm}
// робота з консолью
procedure ExecConsoleApp(CommandLine: AnsiString;
                         Output: TStringList; Errors: TStringList);
var
 sa: TSECURITYATTRIBUTES;
 si: TSTARTUPINFO;
 pi: TPROCESSINFORMATION;
 hPipeOutputRead: THANDLE;
 hPipeOutputWrite: THANDLE;
 hPipeErrorsRead: THANDLE;
 hPipeErrorsWrite: THANDLE;
 Res, bTest: Boolean;
 env: array[0..100] of Char;
 szBuffer: array[0..256] of Char;
 dwNumberOfBytesRead: DWORD;
 Stream: TMemoryStream;
begin
 sa.nLength := sizeof(sa);
 sa.bInheritHandle := true;
 sa.lpSecurityDescriptor := nil;
 CreatePipe(hPipeOutputRead, hPipeOutputWrite, @sa, 0);
 CreatePipe(hPipeErrorsRead, hPipeErrorsWrite, @sa, 0);
 ZeroMemory(@env, SizeOf(env));
 ZeroMemory(@si, SizeOf(si));
 ZeroMemory(@pi, SizeOf(pi));
 si.cb := SizeOf(si);
 si.dwFlags := STARTF_USESHOWWINDOW or STARTF_USESTDHANDLES;
 si.wShowWindow := SW_HIDE;
 si.hStdInput := 0;
 si.hStdOutput := hPipeOutputWrite;
 si.hStdError := hPipeErrorsWrite;
Res := CreateProcess(nil, pchar(CommandLine), nil, nil, true,
  CREATE_NEW_CONSOLE or NORMAL_PRIORITY_CLASS, @env, nil, si, pi);
```

```
// Procedure will exit if CreateProcess fail
if not Res then
begin
 CloseHandle(hPipeOutputRead);
 CloseHandle(hPipeOutputWrite);
 CloseHandle(hPipeErrorsRead);
 CloseHandle(hPipeErrorsWrite);
 Exit;
end;
CloseHandle(hPipeOutputWrite);
CloseHandle(hPipeErrorsWrite);
//Read output pipe
Stream := TMemoryStream.Create;
try
 while true do
 begin
 bTest := ReadFile(hPipeOutputRead, szBuffer, 256, dwNumberOfBytesRead,
 nil);
 if not bTest then
 begin
 break;
 end;
 Stream.Write(szBuffer, dwNumberOfBytesRead);
 end;
 Stream.Position := 0;
 Output.LoadFromStream(Stream);
finally
 Stream.Free;
end;

// Зчитування каналу помилок
 Stream := TMemoryStream.Create;
 try
  while true do
  begin
  bTest := ReadFile(hPipeErrorsRead, szBuffer, 256, dwNumberOfBytesRead,
  nil);
  if not bTest then
  begin
  break;
  end;
```

```
          Stream.Write(szBuffer, dwNumberOfBytesRead);
          end;
          Stream.Position := 0;
          Errors.LoadFromStream(Stream);
         finally
          Stream.Free;
         end;
         WaitForSingleObject(pi.hProcess, INFINITE);
         CloseHandle(pi.hProcess);
         CloseHandle(hPipeOutputRead);
         CloseHandle(hPipeErrorsRead);
        end;

        procedure TForm9.Button2Click(Sender: TObject);
        var adres:string;
        begin
        adres:= Edit1.Text+'.'+Edit2.Text+'.'+Edit3.Text+'.'+Edit4.Text;
        if MessageBox(0, ' прописати роутинг?', 'Пропись роутінга', +mb_YesNo
+MB_ICONQUESTION) = 6 then
        begin
        ShellExecute(0,'open','cmd.exe',PChar('/c route -p add 10.0.0.0 mask
255.0.0.0 10.10.'+ Edit3.Text +'.254'),'C:\Windows\system32\',SW_SHOW);
        ShowMessage('Route на
'+Edit1.Text+'.'+Edit2.Text+'.'+Edit3.Text+'.'+Edit4.Text+' успішно прописаний ');
        end;
        end;

        procedure TForm9.Button3Click(Sender: TObject);
        begin
        if MessageBox (0, 'Ви дійсно хочете видалити роутинг?', 'Вилучення
роутінга', + mb_YesNo + MB_ICONQUESTION) = 6 then
        begin
        ShellExecute(0,'open','cmd.exe',
        '/c route -f','C:\Windows\system32\',SW_SHOW);
        ShowMess age ('Route успішно знищений!');
        end;
        end;

        procedure TForm9.Button4Click(Sender: TObject);
        begin
        ShellExecute(0,'open','cmd.exe',
        '/c ipconfig >ipconfig.txt & ipconfig.txt','C:\Windows\system32\',SW_SHOW);
```

```
end;

procedure TForm9.Button5Click(Sender: TObject);
const WSVer = $101;
var
  wsaData: TWSAData;
  P: PHostEnt;
  Buf: array [0..127] of Char;
  Result: String;
begin
  Result:= '';
  if WSAStartup(WSVer, wsaData) = 0 then begin
    if GetHostName(@Buf, 128) = 0 then begin
      P := GetHostByName(@Buf);
      if P <> nil then Result := iNet_ntoa(PInAddr(p^.h_addr_list^)^);
    end;
    WSACleanup;
    Label2.Caption:=result;
    ShowMessage('Ваш IP-адрес: '+result);
  end;
end;

procedure TForm9.Button6Click(Sender: TObject);
const WSVer = $101;
var
  wsaData: TWSAData;
  P: PHostEnt;
  Buf: array [0..127] of Char;
  Result: String;
begin
  Result:= '';
  if WSAStartup(WSVer, wsaData) = 0 then begin
    if GetHostName(@Buf, 128) = 0 then begin
      P := GetHostByName(@Buf);
      if P <> nil then Result := iNet_ntoa(PInAddr(p^.h_addr_list^)^);
    end;
    WSACleanup;
    Label2.Caption:=result;
  if MessageBox (0, 'Ви дійсно хочете пропінгувати свій IP-адресу?',
'пингования власного IP-адреси', + mb_YesNo + MB_ICONQUESTION) = 6 then
  ShellExecute(0,'open','cmd.exe',
  PChar('/k ping '+ result),'C:\Windows\system32\',SW_SHOW);
```

```
        end;
        end;

        procedure TForm9.Button7Click(Sender: TObject);
        begin
        if MessageBox (0, 'Ви дійсно хочете пропінгувати сервер з IP-адресою:
10.10.1.1?','пингування сервера 10.10.1.1', + mb_YesNo + MB_ICONQUESTION) = 6 then
        ShellExecute (0, 'open', 'cmd.exe',
        PChar('/ k ping 10.10.1.1'), 'C: \ Windows \ system32 \', SW_SHOW);end;

        procedure TForm9.Button8Click(Sender: TObject);
        var ip: string;
            g:string;
        begin
        // Налаштування мережі по введеному IP
        ip:=Edit1.Text+'.'+Edit2.Text+'.'+Edit3.Text+'.'+Edit4.Text;
        g:=Edit1.Text+'.'+Edit2.Text+'.'+Edit3.Text;
        if MessageBox (0, 'Налаштування мережі триває до 2-х хвилин. Що б
приступити до налаштування мережі на введений Вами IP-адрес дочекайтеся закриття
вікна, що з'явилося! Для продовження натисніть "Так"', 'Налаштування мережі зі
статичним IP-адресою', + mb_YesNo + MB_ICONINFORMATION) = 6 then
        // запит на відображення мережевих налаштувань у текстовому файлі
        if MessageBox (0, 'Ви хочете переглянути Ваші мережеві настройки після
завершення установки?', 'Налаштування мережі зі статичним IP-адресою', + mb_YesNo
+ MB_ICONINFORMATION) = 6 then
        begin
        // відображаю приховані елементи візуалізації
        // налаштовую спочатку мережу потім прописую DNS 10.10.1.1
        ShellExecute (0, 'open', 'cmd.exe',
        PChar( '/ c netsh interface ip set address name = "Підключення по локальній
мережі" static addr =' + ip + 'mask = 255.255.255.0 gateway =' + g + ' .254
gwmetric = 10.10.1.1 && netsh interface ip set dns "Підключення по локальній
мережі" static 10.10.1.1 && ipconfig / all> 123.doc & 456.doc '),'
C:\Windows\system32\', SW_SHOW) ;
        timer1.Enabled: = true;
        ProgressBar1.Visible: = true;
        label5.Visible: = true;
        end
        else
        ShellExecute (0, 'open', 'cmd.exe', PChar ( '/ c netsh interface ip set
address name = "Підключення по локальній мережі" static addr =' + ip + 'mask =
255.255.255.0 gateway =' + g + ' .254 gwmetric = 10.10.1.1 && netsh interface ip
```

```
set dns "Підключення по локальній мережі" static 10.10.1.1
'),'C:\Windows\system32\', SW_SHOW);
       end;

       procedure TForm9.Button9Click(Sender: TObject);
       const WSVer = $101;
       var
         wsaData: TWSAData;
         P: PHostEnt;
         Buf: array [0..127] of Char;
         Result: String;
       begin
       if MessageBox (0, 'Ви дійсно хочете налаштувати мережу з автоматичним
присвоєнням IP-адреси дочекайтеся закриття вікна, що з'явилося! Для продовження
натисніть "Так"', 'Налаштування мережі з автоматичним присвоєнням IP-адреси', +
mb_YesNo + MB_ICONINFORMATION) = 6 then begin
       Result:= '';
         if WSAStartup(WSVer, wsaData) = 0 then begin
           if GetHostName(@Buf, 128) = 0 then begin
             P := GetHostByName(@Buf);
             if P <> nil then Result := iNet_ntoa(PInAddr(p^.h_addr_list^)^);
           end;
           WSACleanup;
           Label2.Caption:=result;
           end;
       end;
       end;

       procedure TForm9.Button12Click(Sender: TObject);
       var
        OutP: TStringList;
        ErrorP: TStringList;
       begin
       ShellExecute(0,'open','cmd.exe','/c ipconfig/all >ipconfig_all.txt &
ipconfig_all.txt','C:\Windows\system32\',SW_SHOW);
        OutP := TStringList.Create;
        ErrorP := TstringList.Create;
        //ExecConsoleApp('ipconfig /all', OutP, ErrorP);
        Memo1.Lines.Assign(OutP);
        //ShowMessage(memo1.text);
        OutP.Free;
        ErrorP.Free;
```

```
end;

procedure TForm9.Button14Click(Sender: TObject);
const WSVer = $101;
var
  wsaData: TWSAData;
  P: PHostEnt;
  Buf: array [0..127] of Char;
  Result: String;
begin
  Result:= '';
  if WSAStartup(WSVer, wsaData) = 0 then begin
    if GetHostName(@Buf, 128) = 0 then begin
      P := GetHostByName(@Buf);
      if P <> nil then Result := iNet_ntoa(PInAddr(p^.h_addr_list^)^);
    end;
    WSACleanup;
    Label2.Caption:=result;
if MessageBox (0, 'Комплексна перевірка включає в себе пінгування власного IP-адреси, пінгування VPN.LAN і пінговніе власного шлюзу і займе приблизно 1-2 хвилини! Ви хочете продовжити?', 'Комплексна перевірка втрат', + mb_YesNo + MB_ICONINFORMATION) = 6 then
// пінгуем VPN, шлюз, ip-адреса
ShellExecute(0,'open','cmd.exe',
PChar('/k ping 10.10.'+ edit3.Text+'.254 -t -l 1470 -n 25 &
ping VPN.LAN -t -l 1470 -n 25 & ping '+ result+' -t -l 1470 -n
25'),'C:\Windows\system32\',SW_SHOW);
end;
end;


procedure TForm9.Button16Click(Sender: TObject);
begin
form1.Height:= 500;
if edit5.text='' then
begin
form1.Height:=275;
ShowMessage('Введіть номер помилки ');
edit5.SetFocus;
end
else
if FileExists(ExtractFilePath(Application.ExeName)+
```

```
        '\Errors\'+Edit5.Text+'.rtf') then
        memo2.Lines.LoadFromFile(ExtractFilePath(Application.ExeName)+'\Errors\'+Ed
it5.Text+'.rtf')
        else
        begin
        edit5.SetFocus;
        form1.Height :=500;
        end;
        end;

        end.
```

Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень. Було створено блок-схеми роботи системи. Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

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

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

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

У роботі було використано декілька допоміжних систем, а саме. Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на

мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

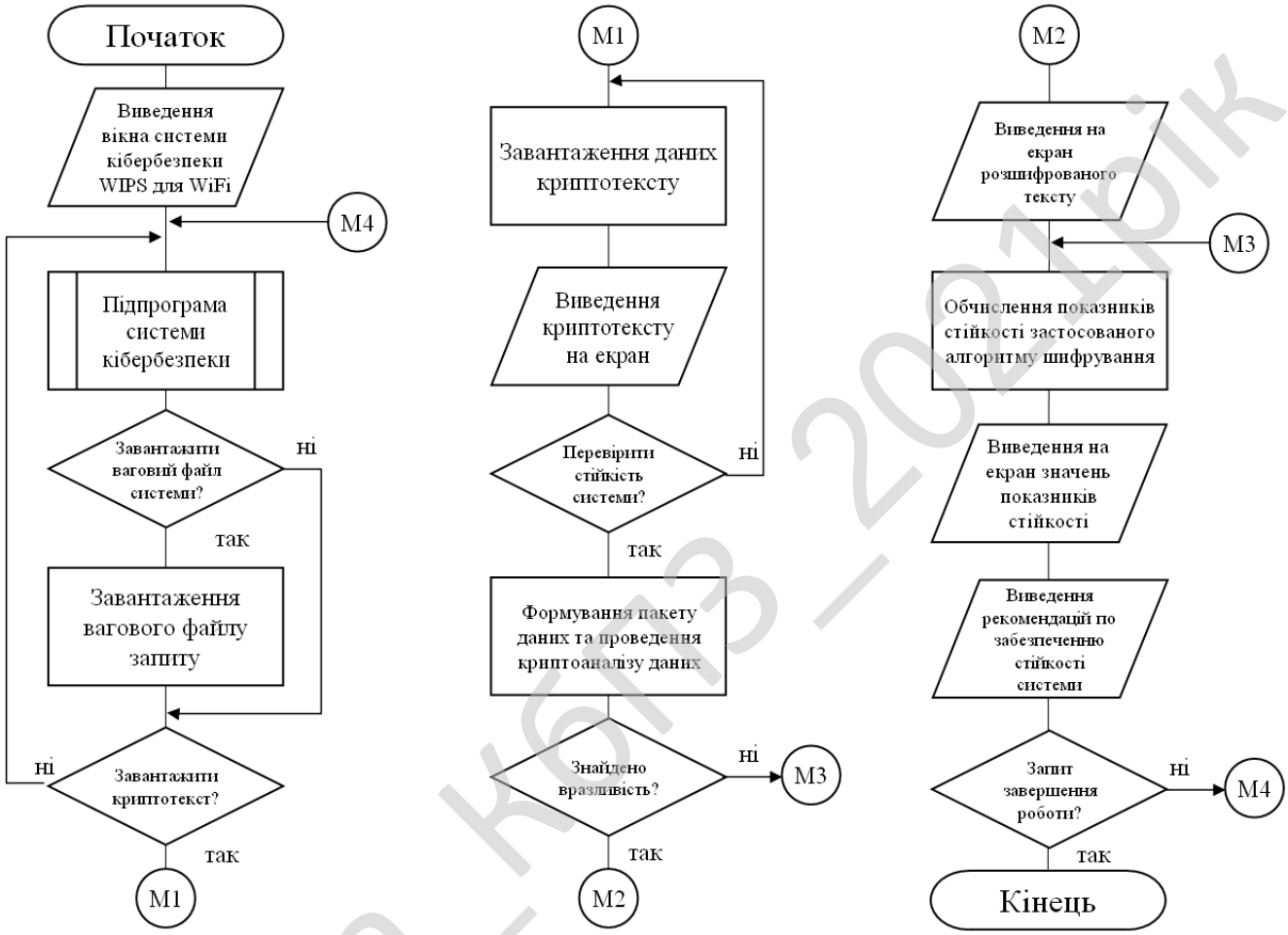

Рисунок 4.1 – Блок-схема основної програми

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

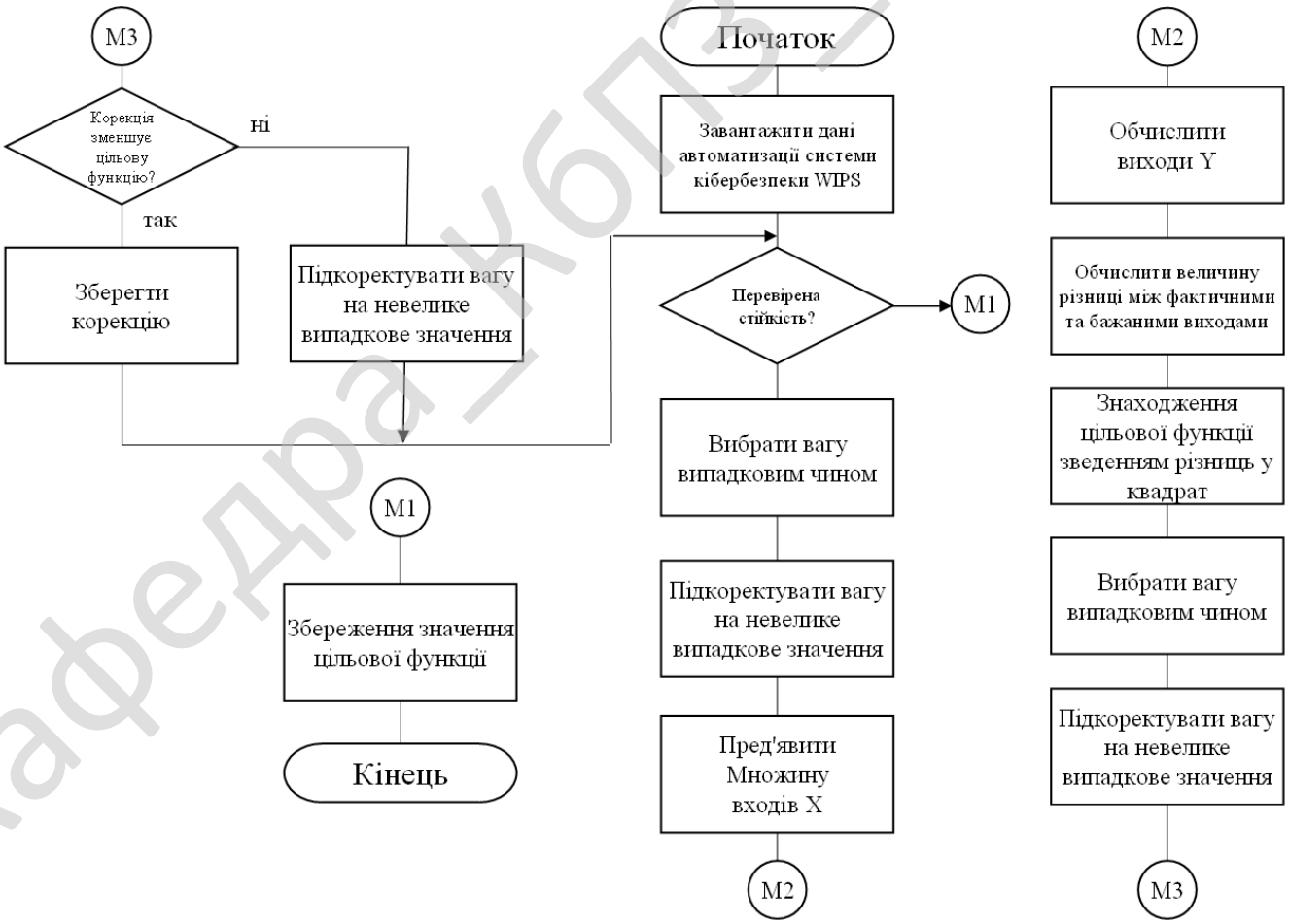

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання. показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Розглянемо визначення API що активно використовувалось під часс рооти. Це прикладни́й програ́мний інтерфе́йс (Application Programming Interface, API) –

набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Спрощено - це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. API є абстрактним поняттям — програмне забезпечення, що пропонує деякий API, часто називають реалізацією (implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим.

В об'єктно-орієнтованих мовах, прикладний програмний інтерфейс зазвичай включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними функціями, які надані або надаватимуться, класами, які реалізуються в методах класу.

Прикладний програмний інтерфейс в даному випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх.

У більш загальному плані можна визначити Прикладний Програмний Інтерфейс як сукупність усіх видів об'єктів, які можна вивести з визначення класу, і пов'язаних з ними можливих варіантів поведінки.

Наприклад: клас, що представляє Stack, може просто виставити публічно два методи Push() (для додавання нового елемента в стек) і Pop() (д ля вилучення останнього пункту, ідеально розташований на вершині стека).

У цьому випадку Прикладний Програмний Інтерфейс може бути інтерпретованим як два методи pop() і push(), або, більш широко, використовується варіант, коли можна використовувати елемент типу Stack, який реалізує поведінку стека, надаючи йому можливість для додавання / видалення елементів з вершини. Друга інтерпретація видається більш доречною в дусі об'єктно-орієнтованого підходу.

Якість документації, пов'язаної з Прикладним Програмним Інтерфейсом, є часто ключовим фактором, що визначає його успішність з точки зору простоти використання.

ППІ, як правило, пов'язаний із бібліотеками програмного забезпечення: ППІ описує і вказує очікувану поведінку в той час, як бібліотека є фактичною реалізацією даного набору правил. Один ППІ може мати декілька реалізацій (або жодної, будучи абстрактним) у вигляді різних бібліотек, які мають такий же інтерфейс.

Прикладний програмний інтерфейс також може бути пов'язаним з платформами програмування: платформа може бути заснована на кількох бібліотеках реалізує декілька інтерфейсів ППІ, але на відміну від звичайного використання ППІ, доступ до поведінки вбудований в платформу опосередкований шляхом розширення його змісту новими класами і вставлений в саму платформу. Крім того, загальний потік управління програми може бути під контролем абонента.

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

Коли ППІ реалізує протокол, він може бути заснованим на проксі-методах віддалених викликів, що засновані на протоколі зв'язку. Роль ППІ може

заключатися саме в тому, щоб приховати деталі транспортного протоколу. Наприклад: RMI є ППІ, який реалізує протокол або JRMP IIOP як RMI-IIOP.

Протоколи, як правило, розподіляються між різними технологіями і зазвичай дозволяють різним технологіям обмінюватися інформацією, діючи як абстракція між двома світами. ППІ, як правило, є специфічним для конкретної технології: звідси, інтерфейси даної мови не можуть бути використані на інших мовах, якщо виклики функції не будуть перетворені з конкретної адаптації бібліотеки.

Деякі мови, серед яких такі, що працюють на віртуальних машинах (наприклад: мови, сумісні з NET CLI середовища CLR і JVM сумісних мов у віртуальній машині Java) можуть ділитися програмними інтерфейсами.

У цьому випадку віртуальна машина дозволяє мові взаємодії завдяки спільному знаменнику віртуальної машини, що абстрагується від конкретної мови, використовувати проміжний байт-код і його мову.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON). У той час як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

Частина цієї тенденції пов'язана з рухом Семантичного веб-ресурсу до Опису Платформ (RDF), Концепції розвитку веб-технологій інженерних онтологій. Прикладні програмні інтерфейси у Web, що дозволяють комбінувати декількома прикладними програмними інтерфейсами в нові додатки називають гібридними.

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою NTRU. NTRUEncrypt (абревіатура Nth-degree TRUncated polynomial ring або Number Theorists aRe Us) – це криптографічна система з відкритим ключем, що раніше називалася NTRU.

Криптосистема NTRUEncrypt, заснована на ґратчастій криптосистемі, створена як альтернатива RSA і криптосистемам на еліптичних кривих (ECC). Стійкість алгоритму забезпечується труднощами пошуку найкоротшого вектора ґрати, що більше стійка до атак, здійснюваним на квантових комп'ютерах. На відміну від своїх конкурентів RSA,ECC, Elgamal, алгоритм використовує операції над кільцем:

$$\mathbb{Z}[X]/(X^N - 1),$$

усічених багаточленів ступеня, що не перевершує $N - 1$:

$$\mathbf{a}(X) = \mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1}.$$

Такий багаточлен можна також представити вектором:

$$\vec{a}(X) = \vec{a} = \sum_{i=0}^{N-1} a_i X^i = [a_0, a_1, a_2, \cdots, a_{N-2}, a_{N-1}].$$

Як і будь-який молодий алгоритм, NTRUEncrypt погано вивчений, хоча й був офіційно затверджений для використання в сфері фінансів комітетом Accredited Standards Committee X9.[1]

Існує реалізації NTRUEncrypt з відкритим вихідним кодом.[2]

NTRUEncrypt, що споконвічно називався NTRU, був винайдений в 1996 році й представлений на конференціях CRYPTO, Конференція RSA, Eurocrypt. Причиною, що послужила початком розробки алгоритму в 1994 році, стала стаття [3], у якій говорилося про легкість злому існуючих алгоритмів на квантових комп'ютерах, які, як показало час, не за горами[4]. У цьому ж році, математики Jeffrey Hoffstein, Jill Pipher і Joseph H. Silverman, що розробили систему разом із засновником компанії NTRU Cryptosystems, Inc. (пізніше перейменованої в

SecurityInnovation), Даніелєм Ліеманом (Daniel Lieman) запатентували свій винахід.[5]

Кільця усічених багаточленів

NTRU оперує над багаточленами ступеня не переважаючої $N-1$:

$$\mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1},$$

де коефіцієнти $a_0, \cdots, a_{N-1}$ – цілі числа. Щодо операцій додавання й множення за модулем багаточлена $X^N - 1$.Такі багаточлени утворять кільце *R*, називане кільцем усічених багаточленів, що ізоморфно кільцю відносин:

$$\mathbb{Z}[X]/(X^N - 1).$$

NTRU використовує кільце усічених багаточленів *R* разом з діленням за модулем на взаємно прості числа *p* і *q* для зменшення коефіцієнтів багаточленів.

У роботі алгоритму також використовуються зворотні багаточлени в кільці усічених багаточленів. Слід зазначити, що не всякий багаточлен має зворотний, але якщо зворотний поліном існує, то його легко обчислити.[6][7]

Генерація відкритого ключа

Для передачі повідомлення від Аліси до Боба необхідні відкритий і закритий ключі. Відкритий знають як Боб, так і Аліса, закритий ключ знає тільки Боб, що він використовує для генерації відкритого ключа. Для цього Боб вибирає два «маленьких» поліноми *f g* з *R*. «Малість» поліномів мається на увазі в тому розумінні, що він маленький щодо довільного полінома за модулем *q*: у довільному поліномі коефіцієнти повинні бути приблизно рівномірно розподілені за модулем *q*, а в малому поліномі вони багато менше *q*. Малість поліномів визначається за допомогою чисел *df* і *dg*:

Поліном *f* має *df* коефіцієнтів рівних «1» і *df*-1коефіцієнтів рівних «-1», а інші – «0». У цьому випадку говорять, що:

$$\mathbf{f} \in \mathcal{L}_f$$

Поліном *g* має *dg* коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0» У цьому випадку говорять, що:

$$\mathbf{g} \in \mathcal{L}_g$$

Причина, по якій поліноми вибираються саме таким чином, полягає в тому, що f, можливо, буде мати зворотний, а g – однозначно немає (g (1) = 0, а нульовий елемент не має зворотного).

Боб повинен зберігати ці полінома в секреті. Далі Боб обчислює зворотні поліноми $\mathbf{f}_p$ й $\mathbf{f}_q$, тобто такі, що:

$$\mathbf{f} \cdot \mathbf{f}_p \equiv 1 \pmod{p} \text{ й } \mathbf{f} \cdot \mathbf{f}_q \equiv 1 \pmod{q}$$

Якщо f не має зворотного полінома, то Боб вибирає інший поліном f.

Секретний ключ – це пари $(\mathbf{f}, \mathbf{f}_p)$, а відкритий ключ h обчислюється за формулою:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod q.$$

Приклад:

Для приклада візьмемо df=4, а dg=3. Тоді як поліноми можна вибрати

$$\mathbf{f} = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$
$$\mathbf{g} = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$$

Далі для полінома f шукаються зворотні поліноми за модулем p=3 иq=32:

$$\mathbf{f}_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9$$
$$\mathbf{f}_q = 5+9X+6X^2+16X^3+4X^4+15X^5+16X^6+22X^7+20X^8+18X^9+30X^{10}$$

Заключним етапом є обчислення самого відкритого ключа h:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod 32 = 8+25X+22X^2+20X^3+12X^4+24X^5+15X^6+19X^7+12X^8+19X^9+16X^{10}.$$

Шифрування

Тепер, коли в Аліси є відкритий ключ, вона може відправити зашифроване повідомлення Бобові. Для цього потрібно повідомлення представити у вигляді полінома m з коефіцієнтами за модулем p, обраними з діапазону (-p/2, p/2]. Тобто m є «малим» поліномом за модулем q. Далі Алісі необхідно вибрати інший «малий» поліном r, що називається «сліпучої», обумовлений за допомогою числа dr:

Поліном r має dr коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0». У цьому випадку говорять, що:

$$\mathbf{r} \in \mathcal{L}_r.$$

Використовуючи ці поліноми, зашифроване повідомлення виходить за формулою:

$$\mathbf{e} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod q.$$

При цьому кожній, хто знає (або може обчислити) осліплюючий поліном r, зможе прочитати повідомлення *m*.

Приклад:

Припустимо, що Аліса хоче послати повідомлення, представлене у вигляді полінома:

$$\mathbf{m} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10},$$

і вибрала «осліплюючий» поліном, для якого *dr*=3:

$$\mathbf{r} = -1 + X^2 + X^3 + X^4 - X^5 - X^7.$$

Тоді шифротекст *e*, готовий для передачі Бобові буде:

$$\mathbf{e} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod 32 = 14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10}.$$

Розшифрування

Тепер, одержавши зашифроване повідомлення *e*, Боб може його розшифрувати, використовуючи свій секретний ключ. Спочатку він одержує новий проміжний поліном:

$$\mathbf{a} = (\mathbf{f} \cdot \mathbf{e}) \bmod q.$$

Якщо розписати шифротекст, то одержимо ланцюжок:

$$\mathbf{a} = (\mathbf{f} \cdot \mathbf{e}) \bmod q = (\mathbf{f} \cdot (\mathbf{r} \cdot \mathbf{h} + \mathbf{m})) \bmod q = (\mathbf{f} \cdot (\mathbf{r} \cdot p\mathbf{f}_q \cdot \mathbf{g} + \mathbf{m})) \bmod q$$

і остаточно:

$$\mathbf{a} = (p\mathbf{r} \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{m}) \bmod q.$$

Після того, як Боб обчислив поліном a за модулем *q*, він повинен вибрати його коефіцієнти з діапазону (-*q*/2, *q*/2] і далі обчислити поліном b, одержуваний з полінома a приведенням за модулем *p*:

$$\mathbf{b} = \mathbf{a} \bmod p = (\mathbf{f} \cdot \mathbf{m}) \bmod p,$$

так як:

$$(p\mathbf{r} \cdot \mathbf{g}) \bmod p = 0.$$

Тепер, використовуючи другу половину секретного ключа й отриманий поліном *b*, Боб може розшифрувати повідомлення:

$$\mathbf{c} = (\mathbf{f}_p \cdot \mathbf{b}) \bmod p.$$

Неважко бачити, що:

$$\mathbf{c} \equiv \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \equiv \mathbf{m} \pmod{p}.$$

У такий спосіб отриманий поліном *c* дійсно є вихідним повідомленням *m*.

Приклад:

Боб одержав від Аліси шифроване повідомлення *e*:

$$\mathbf{e} = 14+11X+26X^2+24X^3+14X^4+16X^5+30X^6+7X^7+25X^8+6X^9+19X^{10}$$

Використовуючи секретний ключ *f* Боб одержує поліном *a*:

$$\mathbf{a} = \mathbf{f}\cdot\mathbf{e} \pmod{32} = 3-7X-10X^2-11X^3+10X^4+7X^5+6X^6+7X^7+5X^8-3X^9-7X^{10} \pmod{32},$$

з коефіцієнтами, що належать проміжку (-*q*/2, *q*/2]. Далі перетворить поліном *a* у поліном *b*, зменшуючи коефіцієнти за модулем *p*.

$$\mathbf{b} = \mathbf{a} \pmod 3 = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10} \pmod 3$$

Заключний крок – перемножування полінома *b* із другою половиною закритого ключа $\mathbf{f}_p$:

$$\mathbf{c} = \mathbf{f}_p \cdot \mathbf{b} = \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \pmod 3 = \mathbf{m} \pmod 3$$

$$\mathbf{c} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

Який є вихідним повідомленням, що передавала Аліса.

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: Архів; перегляд дат; Сформувати звіт; Налаштування параметрів друку; Попередній перегляд звіту та друк.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші з розділами: Налаштуванн;, Авторське право; Довідка; Журнал роботи.

– Верхнього меню: Система; Звіти; Налаштування; Довідка.

– Розділу виведення результату роботи системи: Перегляд архіву; Обрання дії та піддії; Деревовидного обрання підрозділу; Журнал подій.

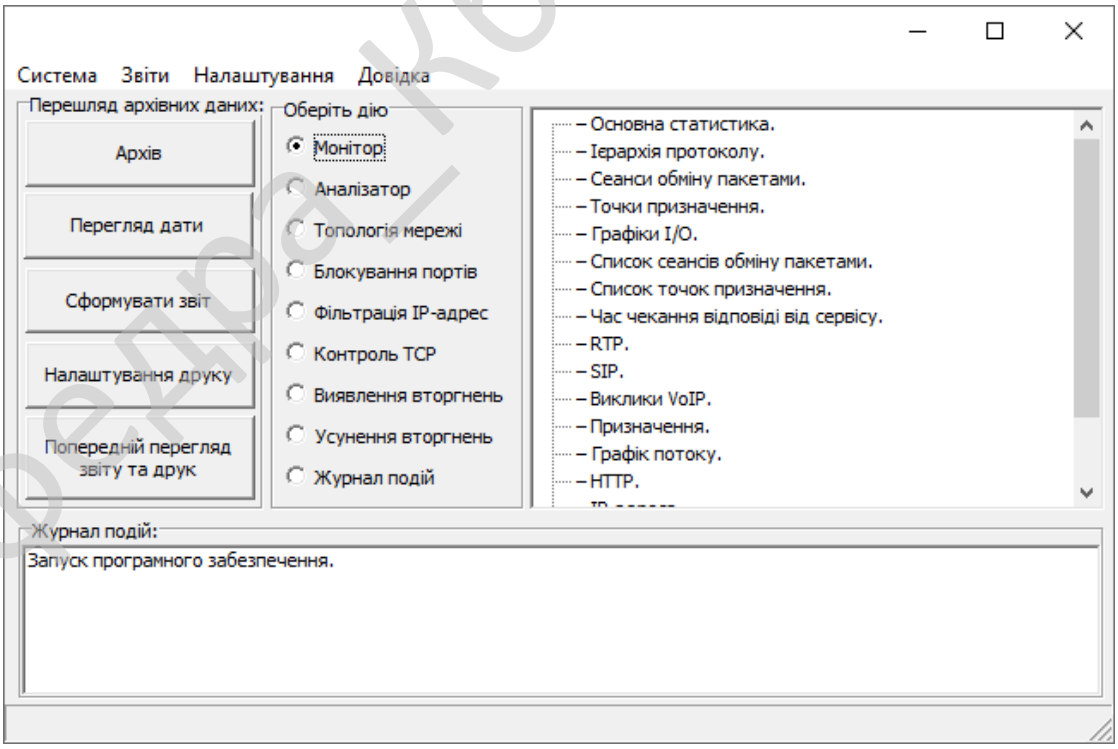

Рисунок 5.1 – Головне вікно ПЗ

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.

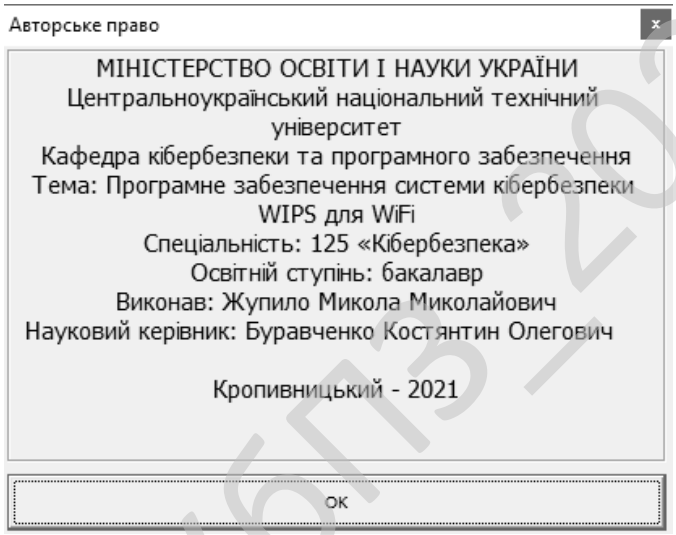

Рисунок 5.2 – Авторське право

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 WIPS для WiFi.

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 WIPS для WiFi.

– Досліджена система кібербезпеки WIPS для WiFi.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки WIPS для WiFi.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання WIPS для WiFi.

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 RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки WIPS для WiFi. 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 XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм NTRU.

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. Бертсекас Д. Сети передачи данных / Д. Бертсекас, Р. Галлагер; пер. с англ. – М.: Мир, 1989. – 544 с.

2. Блинов А.М. Информационная безопасность / А.М. Блинов. – СПб.: Изд-во СПбГУЭФ, 2010. – 96 с.

3. Бокс Д. Анализ временных рядов. Прогноз и управление / Д. Бокс, Г. Дженкинс– М.:Мир, 1974. – 604 с.

4. Василенков С. Кибератаки 2013 года: никто не будет чувствовать себя в безопасности [Электронный ресурс]. – Режим доступа к ресурсу: http://www.peacekeeper.ru/ru/?module=news&action=view&id=16813

5. Вегешна Ш. Качество обслуживания в сетях IP / Ш. Вегешна; пер.с англ. – М.: Издательский дом «Вильямс», 2003. – 386 с.

6. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский . – М.: Техносфера, 2003. – 512 с.

7. Горбенко І.Д. Прикладна криптологія. Теорія. Практика. Застосування / І.Д. Горбенко, Ю.І. Горбенко. – Х.:Форт, 2012. – 870 с.

8. Гордейчик С.В. Безопасность беспроводных сетей. / С.В. Гордейчик, В.В. Дубровин– М.: Горячая линия-Телеком, 2008. – 288 с.

9. Гольдштейн Б. Сети связи пост – NGN / Б. Гольдштейн, А. Кучерявый. – СПб.:БХВ-Петербург, 2013. – 160 с.

10. ГОСТ Р 51275-99 Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положення [Электронный ресурс]. – Режим доступа к ресурсу: http://www.gosthelp.ru/text/GOSTR5127599Zashhitainfor.html

11. ГОСТ Р ИСО/МЭК 15408-2-2002 Информационная технология. Методы и средства обеспечения безопасности критерии оценки безопасности информационных технологий. Часть 2 Функциональные требования безопасности

[Электронный ресурс]. – Режим доступа к ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_15408-2-2002.htm

12. ГОСТ Р ИСО/МЭК 13335-1-2006 Информационная технология Методы и средства обеспечения безопасности Часть 1 Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий [Электронный ресурс]. – Режим доступа к ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_13335-1-2006.htm

13. ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и концепции [Электронный ресурс]. – Режим доступа к ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179072

14. ГОСТ Р ИСО/МЭК 27004-2011 Информационная технология. Методы и средства обеспечения безопасности. Менеджмент информационной безопасности. [Электронный ресурс]. – Режим доступа к ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179060

15. Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

16. Смірнов О.А. Дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем / О.О. Кузнецов, О.А. Смірнов, Д.О. Даниленко // Інформаційна та економічна безпека: сучасний стан та тенденції розвитку : монографія за заг. ред. – Х.: ХІБС УБС НБУ – 2014 – С. 82-100.

17. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні системи та мережі / Д.О. Даниленко, О.А. Смірнов, Є.В. Мелешко // Системи озброєння і військова техніка. – Випуск 1(29) – Х.: ХУПС – 2012. – С. 92-100

18. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения. Часть 1. Корреляционный анализ сетевого трафика // А.А.Смирнов, Д.А. Даниленко, Е.В.Мелешко // Науково-технічний журнал «Інформаційно-

керуючі системи на залізничному транспорті» – Випуск 4(95). – Х.: УкрДАЗТ – 2012. – С. 8-14.

19. Смирнов А.А. Методы обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 3(101) том 2. – Х.: ХУПС – 2012. – С. 152-155.

20. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (21) том 2. – Київ: ДП «ЦНДІНУ». – 2012. – С. 183-186.

21. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты компьютерных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, И.Г. Кирилов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 70-71.

22. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні мережі для підвищення інформаційної безпеки // Д.О. Даниленко // Збірник тез науково-практичної конференції «Захист інформації в інформаційно-комунікаційних системах». м. Київ. 24-27 квітня 2012 р. – Київ: НАУ. – 2012. – С. 22-25.

23. Смирнов А.А. Исследование систем обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко // Збірник тез доповідей VIII наукової конференції «Новітні технології – для захисту повітряного простору». Харків. 18-19 квітня 2012 р. – м. Харків. ХУПС. – 2012. – С. 45.

24. Смирнов А.А. Исследование методов сигнатурного обнаружения

вредоносного программного обеспечения в телекоммуникационных системах и сетях // Д.А. Даниленко // Збірник тез XIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: КНТУ. – 2012. – С. 43-45.

25. Смирнов А.А. Исследование методов проактивной защиты от вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «інтегровані інтелектуальні робототехнічні комплекси» (ІІРТК-2012). м. Київ. 15-16 травня 2012 р. – Київ: НАУ. – 2012. – С. 314-315.

26. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения на основе корреляционного анализа сетевого трафика / Д.А. Даниленко // Матеріали XII всеукраїнської наукової інтернет-конференції «Наукові дослідження: зв'язок теорії і практики». м. Тернопіль. 29-30 квітня 2012 р. – Тернопіль: ТНЕУ. – 2012. – С. 9-10.

27. Смирнов А.А. Метод детектирования вредоносного трафика в телекоммуникационных сетях на основе использования bds-тестирования / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології» (CSNT-2012). м. Київ. 13-15 червня 2012 р. – Київ: НАУ. – 2012. – С. 121.

28. Смирнов А.А. Обнаружение и предотвращение вторжений в компьютерных сетях на основе статистического анализа сетевого трафика / А.А. Смирнов, Д.А. Даниленко // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 13-14.

29. Смірнов О.А. дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем та мереж / О.А. Смірнов, Д.О. Даниленко // Збірник тез V Всеукраїнської науково-практичної конференції "Інформатика та системні науки". м. Полтава. 13-15

березня 2014 р. – Полтава: ПУЕТ. – 2014. – С. 289-291.

30. Смирнов А.А. Метод дисперсионного анализа сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях/ А.А. Смирнов, Д.А. Даниленко // Збірник тез VI міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 258.

31. Смірнов О.А. метод забезпечення інформаційної безпеки телекомунікаційних систем з використанням дисперсійного аналізу мережного трафіку / О.А. Смірнов, Д.О. Даниленко // Збірник тез міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2014)». м. Харків. 15-16 травня 2014 р. – Харків: ХІБС УБС НБУ. – 2014. – С. 135-139.

32. Девянин П.Н. Модели безопасности компьютерных систем / П.Н. Девянин. – М.:Издательский центр «Академия», 2005. – 144 с.

33. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты / В.В. Домарев. – К.: ООО "ТИД "ДС", 2002 – 688 с.

34. ДСТУ ISO/IEC TR 13243-2003 Інформаційні технології. Посібник із методів та механізмів якості послуг / [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/informaciini-tehnologiyi.-posibnik-iz-metodiv-ta-mehanizmiv--nor2718.html

35. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.: УкрНДІССІ, 1995. – 23 с.

36. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/docs/tdoc14237.php

37. Ершов В.А.. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов. – М.: МГТУ им. Н.Э. Баумана, 2003. – 432 с.

38. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к

ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

39. Информационная война и защита информации. Словарь основных терминов и определений [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csef.ru/files/csef/articles/2176/2176.pdf

40. Казарин О.В. Безопасность программного обеспечения компьютерных систем / О.В. Казарин. – М.:МГУЛ, 2003. – 212 с.

41. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

і. Касперский К. Техника сетевых атак. [Електронний ресурс]. – Режим доступу до ресурсу: http://rghost.ru/download/43730077/

42. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

43. Касперский К. Техника и философия хакерских атак. / К. Касперский. – М.: Солон-Пресс, 2004 – 272 с.

44. Концепция (основы государственной политики) Национальной безопасности Украины, утвержденная постановлением Верховной Рады Украины от 16 января 1997 г. № 3/97-ВР. [Электронный ресурс]. – Режим доступа к ресурсу: http://search.ligazakon.ua/l_doc2.nsf/link1/ F970003.html

45. Конри-Мюррей Эндрю Выявление вторжений / «Журнал сетевых решений/LAN», № 01, 2001 [Электронный ресурс]. – Режим доступа к ресурсу: http://www.osp.ru/lan/2001/01/131532/

46. Кузнецов О.О. Протоколи захисту інформації у комп'ютерних системах та мережах / О.О. Кузнецов, С.Г. Семенов. – Х.: ХНУРЕ, 2009. – 184 с.

47. Кузнецов О.О. Методи обробки сигналів даних та зображень / О.О. Кузнецов, Г.А. Кучук, С.Г. Семенов. –Х.: НТУ «ХПІ», 2011. – 292 с.

48. Кузнецов А.А. Дисперсионный анализ сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях / А.А. Кузнецов, А.А. Смирнов, Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 2(118). – Х.: ХУПС – 2014. – С. 81-85.

49. Лоскутов А.Ю. Основы теории сложных систем / А.Ю. Лоскутов, А.С. Михайлов. – М.-Ижевск: Институт компьютерных исследований, 2007. – 620 с.

50. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.:BHV – Санкт - Петербург, 2003. – 596 с.

51. Методи підвищення оперативності передачі даних та захисту інформації у телекомунікаційній мережі: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U006631

52. Методология функционального моделирования IDEF0. Руководящий документ [Электронный ресурс]. – Режим доступа к ресурсу: http://www.scribd.com/doc/76782197/МЕТОДОЛОГИЯ-ФУНКЦИОНАЛЬНОГО-МОДЕЛИРОВАНИЯ-IDEF0

53. Моделирование технических систем с AllFusion Process Modeler (ранее BPwin) [Электронный ресурс]. – Режим доступа к ресурсу:: http://www.sdteam.com/t5506

54. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров сети АТМ / А.Н. Назаров. – М.: Горячая линия – Телеком, 2002. –255 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Перелік документів, що розробляються.....5
8 Етапи розробки.....6
9 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки WIPS для WiFi.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 135-02 від 24.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки WIPS для WiFi.

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 WIPS для WiFi;

– 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 сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 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 RAD Studio Delphi 10.4.

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 – 79 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 11.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Буравченко К.О.

Програмне забезпечення системи кібербезпеки WIPS для WiFi

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2021 року

Файл TCP_IP.pas- монітор TCP/IP з'єднань досліджуємої мережі системи кібербезпеки WIPS для WiFi (WIPS_WiFi)

```
unit TCP_IP;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;
```

```
procedure TForm2.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Відбулася помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( ' Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл Stat.pas- статистика досліджуємої мережі системи кібербезпеки WIPS для WiFi(WIPS_WiFi)

```
unit Stat;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
```

```
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( '  Час запиту '
      + inttostr( rtt ) + '  ms, '
      + inttostr( HopCount )
      + '  hops to : '  + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:'  + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Основна програма

Файл WIPS_WiFi.dpr основної програми

```
program WIPS_WiFi;

uses
  Forms,
  Main in ' Main.pas'  {MainForm},
  About in ' About.pas'  {Form1},
  TCP_IP in ' TCP_IP.pas'  {Form2},
  Stat in ' Stat.pas'  {Form3};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

Файл Main.pas основної програми

```
unit Main;

interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
```

```
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
        ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
  //  function OpenEnum(const NetContainerToOpen: PNetResource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_netname : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
   Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
```

```
    Sesi50_username    : PChar;
    sesi50_key         : Cardinal;
    sesi50_num_conns   : Word;
    sesi50_num_opens   : Word;
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;

type
  TFileInfo3 = packed record
    fi3_id          : DWORD;
    fi3_permissions : DWORD;
    fi3_num_locks   : DWORD;
    fi3_pathname    : PWChar;
    fi3_username    : PWChar;
  end;
  PFileInfo3 = ^TFileInfo3;
  TFileInfo3Array = array[0..512] of TFileInfo3;
  PFileInfo3Array = ^TFileInfo3Array;

type
  TFileInfo50 = packed record
    fi50_id          : Cardinal;
    fi50_permissions : WORD;
    fi50_num_locks   : WORD;
    fi50_pathname    : PChar;
    fi50_username    : PChar;
    fi50_sharename   : PChar;
  end;

type
   TMibIfRow = packed record
     wszName            : array[0..255] of WideChar;
     dwIndex            : DWORD;
     dwType             : DWORD;
     dwMtu              : DWORD;
     dwSpeed            : DWORD;
     dwPhysAddrLen      : DWORD;
     bPhysAddr          : array[0..7] of Byte;
     dwAdminStatus      : DWORD;
     dwOperStatus       : DWORD;
     dwLastChange       : DWORD;
     dwInOctets         : DWORD;
     dwInUcastPkts      : DWORD;
     dwInNUCastPkts     : DWORD;
     dwInDiscards       : DWORD;
     dwInErrors         : DWORD;
     dwInUnknownProtos  : DWORD;
     dwOutOctets        : DWORD;
     dwOutUCastPkts     : DWORD;
     dwOutNUCastPkts    : DWORD;
     dwOutDiscards      : DWORD;
     dwOutErrors        : DWORD;
     dwOutQLen          : DWORD;
     dwDescrLen         : DWORD;
     bDescr             : array[0..255] of Char;
   end;
   TMibIfArray = array [0..512] of TMibIfRow;
   PMibIfRow = ^TMibIfRow;
   PMibIfArray = ^TMibIfArray;

type
  TMibIfTable = packed record
    dwNumEntries     : DWORD;
    Table            : TMibIfArray;
  end;
  PMibIfTable = ^TMibIfTable;
```

```
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

var
NetShareEnum:function ( pszServer   : PChar;
                        sLevel           : Cardinal;
                        pbBuffer    : Pchar;
                        cbBuffer    : Cardinal;
                        pcEntriesRead,
                        pcTotalAvail: Pointer):DWORD; stdcall;

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

var
NetShareAddNT: function(servername: PWideChar;
                        level: DWORD;
                        buf: Pointer;
                        parm_err: LPDWORD): DWORD; stdcall;

var
NetShareAdd: function ( pszServer:Pchar;
                        sLevel:Cardinal;
                        pbBuffer:PChar;
                        cbBuffer:Word):DWORD; stdcall;

Var
NetSessionEnumNT:function(servername,
                        UncClientName,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetSessionEnum:function(pszServer:PChar;
                        sLevel: DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvial:Pointer):integer; stdcall;

var
NetSessionDelNT:function(ServerName,
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
                        sReserved: SmallInt):DWORD; stdcall;

var
```

```
NetFileEnumNT:function( servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetFileEnum:function(     pszServer,
                        pszBasePath:PChar;
                        sLevel:DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvail:pointer):integer; stdcall;

var
NetFileClose:function(  ServerName:PWideChar;
                        FileId:DWORD):DWORD; stdcall;

var
NetFileClose2:function( pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

var
GetIfTable:function(    pIfTable    : PMibIfTable;
                        pdwSize     : PULONG;
                        bOrder      : Boolean ): DWORD; stdcall;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

//////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
// щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
// Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана

  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT тьа вище -
підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
end;
```

```
////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів досліджуємої мережі
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                  @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle); //Не забуваємо вивантажити бібліотеку
end;

////////////////////////////
//
//  Закриття загального ресурсу
//

procedure TMainForm.btnCloseSharesClick(Sender: TObject);
```

```
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;  //Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit; //Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);  //Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i); //Перетворимо в PWideChar
    NetShareDelNT(nil,NameNT,0);   //Видаляємо ресурс
    FreeMem(NameNT);  //Звільняємо пам' ять
  end else begin  //Код для 9 х-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    NetShareDel(nil,@Name9x,0); //Видаляємо ресурс
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Показ діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
```

```
end;

//////////////////////////
//
//  Додавання загального ресурсу
//

procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL =  258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory; //Визначаємо шлях до наступного ресурсу
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' ); //Визначаємо ім'
я під яким він буде видний у мережі
  if TmpDir = ' '  then Exit;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256; //Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength); //Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT; //Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE; //Тип ресурсу
    ShareNT.shi2_remark := ' ' ; //Коментар
    ShareNT.shi2_permissions := ACCESS_ALL; //Доступ
    ShareNT.shi2_max_uses := DWORD(-1); // Кіл-ть максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кіл-ть поточних підкл.

    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT; //Шлях до ресурсу

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
```

```
    Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL; //Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0); //Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Помітьте що активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати Кіл-ть секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

////////////////////////////
//
//  Одержання списку сесій досліджуємої мережі системи кібербезпеки WIPS для
WiFi(WIPS_WiFi)
//

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
```

```
  lvSessions.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname); //Ім' я комп' ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname); //Ім' я комп' ютера
досліджуємої мережі системи кібербезпеки WIPS для WiFi(WIPS_WiFi)
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens)); //Відкритих
ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key; //Унікальний
ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
```

```
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Одержання списку відкритих файлів досліджуємої мережі системи кібербезпеки
WIPS для WiFi(WIPS_WiFi)
//

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
```

```
    end;
    FileInfoNT := nil;
    if NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetFileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
    begin
```

```
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний / вихідний трафік досліджуємої мережі системи
кібербезпеки WIPS для WiFi(WIPS_WiFi)
//

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := '  00-00-00'  else
    begin
      Result := ' ' ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+' -' ;
      Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(' IPHLPAPI.DLL' ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ' GetIfTable' );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //Усього прийнято
байт з досліджуємої мережі системи кібербезпеки WIPS для WiFi(WIPS_WiFi)
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт у досліджуєму мережу системи кібербезпеки WIPS для WiFi(WIPS_WiFi)
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true; //Не забуваємо активувати таймер
end;
```

```
 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                  NetContainerToOpen, hNetEnum))
 then ShowMessage(‘ Помилка!’ )
 else Result:=hNetEnum;
end;

function EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
 function ShowResource(const ParentNode: TTreeNode; Res: TNetResource):
TTreeNode;
  begin
   Result:=MainForm.NetTree.Items.AddChild(ParentNode,
string(Res.lpRemoteName));
  end;

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                              @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає 0 у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES дані в запису на момент
          // попереднього виклику
      Exit;
      else ShowMessage(Помилка!’ );
      Result:=1;
      Exit;
     end;
    end;
   for i:=1 to EntriesToGet do
    begin
     NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
     if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
     then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
     Application.ProcessMessages;
    end;
  end;
end;

procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
```

```
begin
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(NetTree.Items.Add(nil, ' Network Resources' ),
                                ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(NetTree.Selected.Text),' ' ,' ' ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
```

```
end;

procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Файл IPHLPAPI.pas - обробка API функцій

```
unit IPHLPAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // змінна
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // змінна
  DEFAULT_MINIMUM_ENTITIES = 32; // змінна
  MAX_HOSTNAME_LEN = 128; // змінна
  MAX_DOMAIN_NAME_LEN = 128; // змінна
  MAX_SCOPE_ID_LEN = 256; // змінна

 // Вузлові типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] =
    ( ' Невизначений' , ' Передача' , ' Рівень до рівня' , ' ' , '
Змішаний' , ' ' , ' ' , ' ' , ' Гібрід'
      );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

Знайдено у ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet' , ' tokenring' , ' FDDI' , ' PPP' , '
loopback' , ' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
```

```
          ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , '
tokenring' ,
            ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' ' , ' PPP' ,
            ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
      // Кінець змін в типі адаптерів

      //-------для інших MS заготовочних файлів------------------

        MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
        MAXLEN_PHYSADDR = 8; { iprtrmib.h }
        MAXLEN_IFDESCR = 256; {-"--     }

      //-----------------------------------

      type
        TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

      //---IP адресні структури-------------------------

        PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
        TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
        //
        PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
        TIP_ADDR_STRING = packed record // для використання у зв' язних списках
          Next: PTIP_ADDR_STRING;
          IpAddress: TIP_ADDRESS_STRING;
          IpMask: TIP_ADDRESS_STRING;
          Context: DWORD;
        end;

      //-----Fixed Info структура---------------------

        PTFixedInfo = ^TFixedInfo;
        TFixedInfo = packed record
          HostName: array[1..MAX_HOSTNAME_LEN + 4] of char;    // дані
          DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // дані
          CurrentDNSServer: PTIP_ADDR_STRING;
          DNSServerList: TIP_ADDR_STRING;
          NodeType: UINT;
          ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // дані
          EnableRouting: UINT;
          EnableProxy: UINT;
          EnableDNS: UINT;
        end;

      //-----структура мережного інтерфейсу досліджуємої мережі системи
кібербезпеки WIPS для WiFi(WIPS_WiFi)-----------------------

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати дані зразу. Стан >= DISCONNECTED
//
      // може передавати деякі дані. Стан < DISCONNECTED може   //
      // не передавати дані.
//
      // карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
      // причин. Крім невдачі з'єднання.                                  //
      //
//
      // NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не
працює       //
      //                       або не з'єднується з картою.        //
      // UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
```

```
      //                    не з'єднується за потрібний час.
//
      // DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                    не з'єднується.
      //
      // CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
      //                    з сайтом, якого немає.              //
      // CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                    з'єднується.
//
      // OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
      //                    в праці.                                    //
      //
//
      // Усі дії користувачів записуються до  MIB-II значення    //
      // можуть бути використовані                                    //
      //
//
      ////////////////////////

      const
      // дані додані до ipifcons.h
        IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        IF_OPER_STATUS_UNREACHABLE = 1 ;
        IF_OPER_STATUS_DISCONNECTED = 2 ;
        IF_OPER_STATUS_CONNECTING = 3 ;
        IF_OPER_STATUS_CONNECTED = 4 ;
        IF_OPER_STATUS_OPERATIONAL = 5 ;

        MIB_IF_TYPE_OTHER = 1 ;
        MIB_IF_TYPE_ETHERNET = 6 ;
        MIB_IF_TYPE_TOKENRING = 9 ;
        MIB_IF_TYPE_FDDI = 15 ;
        MIB_IF_TYPE_PPP = 23 ;
        MIB_IF_TYPE_LOOPBACK = 24 ;
        MIB_IF_TYPE_SLIP = 28 ;

        MIB_IF_ADMIN_STATUS_UP = 1 ;
        MIB_IF_ADMIN_STATUS_DOWN = 2 ;
        MIB_IF_ADMIN_STATUS_TESTING = 3 ;

        MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
        MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
        MIB_IF_OPER_STATUS_CONNECTING = 3 ;
        MIB_IF_OPER_STATUS_CONNECTED = 4 ;
        MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

      type
        PTMibIfRow = ^TMibIfRow;
        TMibIfRow = packed record
          wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
          dwIndex: DWORD;
          dwType: DWORD;       // дивись MIB_IF_TYPE
          dwMTU: DWORD;
          dwSpeed: DWORD;
          dwPhysAddrLen: DWORD;
          bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
          dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
          dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
          dwLastChange: DWORD;
          dwInOctets: DWORD;
          dwInUcastPkts: DWORD;
          dwInNUCastPkts: DWORD;
          dwInDiscards: DWORD;
          dwInErrors: DWORD;
          dwInUnknownProtos: DWORD;
```

```
          dwOutOctets: DWORD;
          dwOutUCastPkts: DWORD;
          dwOutNUCastPkts: DWORD;
          dwOutDiscards: DWORD;
          dwOutErrors: DWORD;
          dwOutQLen: DWORD;
          dwDescrLen: DWORD;
          bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
        end;

       //
        PTMibIfTable = ^TMIBIfTable;
        TMibIfTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..ANY_SIZE- 1] of TMibIfRow;
        end;

      //---ADAPTER INFO структура-----------------------

        PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
        TIP_ADAPTER_INFO = packed record
          Next: PTIP_ADAPTER_INFO;
          ComboIndex: DWORD;
          AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;       //
дані
          Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
// дані
          AddressLength: UINT;
          Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // дані
          Index: DWORD;
          aType: UINT;
          DHCPEnabled: UINT;
          CurrentIPAddress: PTIP_ADDR_STRING;
          IPAddressList: TIP_ADDR_STRING;
          GatewayList: TIP_ADDR_STRING;
          DHCPServer: TIP_ADDR_STRING;
          HaveWINS: BOOL;
          PrimaryWINSServer: TIP_ADDR_STRING;
          SecondaryWINSServer: TIP_ADDR_STRING;
          LeaseObtained: LongInt ; // UNIX час, секунди з 1970
          LeaseExpires: LongInt;   // UNIX час, секунди з 1970
          SpareStuff: array [1..200] of char ;   // дані- простір для списку IP
адрес досліджуємої мережі системи кібербезпеки WIPS для WiFi(WIPS_WiFi)
        end;

      //--------TCP структура---------------------

        PTMibTCPRow = ^TMibTCPRow;
        TMibTCPRow = packed record
          dwState: DWORD;
          dwLocalAddr: DWORD;
          dwLocalPort: DWORD;
          dwRemoteAddr: DWORD;
          dwRemotePort: DWORD;
        end;
        //
        PTMibTCPTable = ^TMibTCPTable;
        TMibTCPTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..0] of TMibTCPRow;
        end;
        //
        PTMibTCPStats = ^TMibTCPStats;
        TMibTCPStats = packed record
          dwRTOAlgorithm: DWORD;
          dwRTOMin: DWORD;
          dwRTOMax: DWORD;
          dwMaxConn: DWORD;
          dwActiveOpens: DWORD;
```

```
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//-----UDP СТРУКТУРА -------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE- 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
  end;

//------IP СТРУКТУРА ------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
```

```
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

  //
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPForwardRow;
  end;

//---- ICMP-СТРУКТУРА ------------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
```

```
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;

//---------імпорт до IPHLPAPI.DLL------------------

var

GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD; stdcall;

GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
        DWORD; stdcall;

GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD; stdcall;

GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
 bOrder: BOOL ): DWORD; stdcall;

GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD; stdCall;

GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

  // попередження – недокументована функція, можливі помилки при
використанні
  GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    IpHlpDLL = ' IPHLPAPI.DLL' ;
var
    IpHlpModule: THandle;

    function LoadIpHlp: Boolean;

implementation

function LoadIpHlp: Boolean;
begin
    Result := True;
    if IpHlpModule <> 0 then Exit;
```

```
      // відкрити DLL
          IpHlpModule := LoadLibrary (IpHlpDLL);
          if IpHlpModule = 0 then
          begin
              Result := false;
              exit ;
          end ;
          GetAdaptersInfo := GetProcAddress (IpHlpModule, ' GetAdaptersInfo' ) ;
          GetNetworkParams := GetProcAddress (IpHlpModule, ' GetNetworkParams' )
;
          GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
          GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' )
;
          GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
          GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' )
;
          GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
          GetIpNetTable := GetProcAddress (IpHlpModule, ' GetIpNetTable' ) ;
          GetIpAddrTable := GetProcAddress (IpHlpModule, ' GetIpAddrTable' ) ;
          GetIpForwardTable := GetProcAddress (IpHlpModule, ' GetIpForwardTable'
) ;
          GetIcmpStatistics := GetProcAddress (IpHlpModule, ' GetIcmpStatistics'
) ;
          GetRTTAndHopCount := GetProcAddress (IpHlpModule, ' GetRTTAndHopCount'
) ;
          GetIfTable := GetProcAddress (IpHlpModule, ' GetIfTable' ) ;
          GetIfEntry := GetProcAddress (IpHlpModule, ' GetIfEntry' ) ;
          GetFriendlyIfIndex := GetProcAddress (IpHlpModule, '
GetFriendlyIfIndex' ) ;
      end;

      initialization
          IpHlpModule := 0 ;
      finalization
          if IpHlpModule <> 0 then
          begin
              FreeLibrary (IpHlpModule) ;
              IpHlpModule := 0 ;
          end ;

      end.
```

Файл IPHelper.pas- функції роботи з IP-протоколом

```
unit IPHelper;


interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const
    // тільки найбільш популярні сервіси...
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Ping    }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { File Transfer Protocol- дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Часовий протокол }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { Сервіс – Кто це  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP Сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP Кієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { Протокол SUN Remote Procedure Call }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service
}
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій         }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),     { Протокол Simple Netw. Management
Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
```

```
    );


//------перетворення ICMP кодів помилок до рядків-------------

const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ПОМИЛКА' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


//-----Перетворення різних перерахованих величин у рядки--------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , '  MIL-STD-1778' ,
    ' Van Jacobson' , ' інший'  );

  IPForwTypes   : array[1..4] of string =
    ( ' інший' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' інший' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
```

```
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі дані---------------------

function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// функції перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;

//-------Основні функції----------------------

{ отримання наступного "токена" з рядка }
```

```
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

//-----------------------------------
{ перетворення числового MAC-адреса  до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := '  00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення IP-адреси в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення крапкової десяткової IP-адреси в мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;
```

```
//-----------------------------------
{ перетворення номеру порту в мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в мережний байт типу string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку, якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;

//-----------------------------------
{ голона частина, фіксація мережний параметрів досліджуємої мережі системи
кібербезпеки WIPS для WiFi(WIPS_WiFi)}

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен           : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP область        : '  + ScopeID );
        List.Add( ' ROUTING визначено   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-----------------------------------//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
```

```
begin
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                    // дані
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // дані
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // дані
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // дані
    end ;
end;

//-----------------------------------

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


//-----------------------------------

// включення байтів у/з для кожного адаптера

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
```

```
  try
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' данних немає ' )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // конвертуємо Юнікод
до рядка
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
              ' %0.8x |%3d | %16s |%8d |%12d |%2d |%2d |%10d |%10d | %-s| %-s'
,
               [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
               dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
               dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
конвертуємо до 32-біт
               sIfName, sDescr] )  // дані, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;

function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищаємо буфер з  W98 не беремо
  IfRow.dwIndex := Index ;
```

```
  result := GetIfEntry (@IfRow) ;
end ;

//-----------------------------------
{ інформація про інстальовані адаптери }

function IpHlpAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows):
integer ;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                              AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та конвертуємо в IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
```

```
            if Length (AdpRows [AdpTot].IPAddressList) <= I then
            begin
                 SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                 SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

    // беремо список IP адрес для GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.GatewayList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].GatewayList) <= I then
                        SetLength (AdpRows [AdpTot].GatewayList, I -2) ;
        end ;
        AdpRows [AdpTot].GatewayTot := I ;

    // беремо список IP адрес для  GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.DHCPServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].DHCPServer) <= I then
                        SetLength (AdpRows [AdpTot].DHCPServer, I -2) ;
        end ;
        AdpRows [AdpTot].DHCPTot := I ;

    // беремо список IP адрес для PrimaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].PrimWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].PrimWINSTot := I ;

    // беремо список IP адрес для SecondaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].SecWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].SecWINSTot := I ;

        AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
        AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

        inc (AdpTot) ;
        if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot -2) ;  // більше пам' яті
        AdapterInfo := AdapterInfo^.Next;
    end ;
    SetLength (AdpRows, AdpTot) ;
```

```
      end ;
  finally
      FreeMem( pBuf );
  end ;
end ;

procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := IpHlpAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' данних немає ’  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' |’  + Description ); // jpt : не
використовується
            List.Add( Format(' %8.8x | %6s | %16s | %2d | %16s | %16s | %16s’ ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : не використовується
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /’  + IPMaskList [J] + '
| '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '   ' ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//-----------------------------------
{ моніторимо час доступу до IP досліджуємої мережі системи кібербезпеки WIPS для
WiFi(WIPS_WiFi)}
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT :=-1; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
```

```
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   // дані
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( ' ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
    dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNetRow ) );
    FreeMem( pBuf );
  end ;
end;


//-----------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
```

```
  RecentIPs.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо останій запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s | %15s : %-7s | %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) =-1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі : '  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мінімальний час виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу         : '  + IntToStr( dwRTOMax ) +
'  ms'  );
```

```
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних          : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-----------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // виділяємо пам' ять, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
```

```
              ] ) );
          inc( pBuf, SizeOf( TMIBUDPRow ) );
        end;
      end
      else
        List.Add( ‘ немає даних.’  );
    end
    else
      List.Add( SyserrorMessage( ErrorCode ) );
    dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
    FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  // перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ‘ %8.8x | %15s | %15s | %15s | %8.8d’ ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ‘ немає даних.’  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик!
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації досліджуємої мережі системи
кібербезпеки WIPS для WiFi(WIPS_WiFi); }
```

```
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin

  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                   dwForwardType := 1 ;   // дані
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
```

```
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// дані
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос            : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована    : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//-----------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (In)     : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)    : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів          : '  + inttostr( dwNoPorts ) );
```

```
      List.add( ' Помилка    (In)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//-------------------------------------*//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( '  Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка              : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено           : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo відповідь         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка              : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено           : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Exo відповідь         : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
  end
```

```
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

//-----------------------------------
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл About.pas - довідка

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```