МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Бурлаченко Владислав Юрійович

Програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood

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

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

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

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

Бурлаченку Владиславу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood*

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

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

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

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood*

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

Бурлаченко В.Ю. Програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

Метою розробки є програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

Результат роботи – програмна реалізація системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

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 10.4.1.

Ключові слова: кібербезпека, TCP SYN Flood

ABSTRACT

Burlachenko V.Yu. Cybersecurity system software for modeling and detecting DoS attacks such as TCP SYN Flood. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software which is intended for system of cybersecurity of modeling and definition of DoS of attack like TCP SYN Flood is developed.

The purpose of development is the software of cybersecurity system of modeling and definition of DoS attack like TCP SYN Flood.

The result is a software implementation of a cybersecurity system modeling and detection of DoS attacks such as TCP SYN Flood.

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 10.4.1.

Keywords: cybersecurity, TCP SYN Flood

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ACK	–	Підтвердження (ACKnowledge) успішності отримання TCP-сегменту
DNS	–	Domain Name System, DNS – ієрархічна розподілена система перетворення імені хоста (комп'ютера або іншого мережевого пристрою) в IP-адресу
DoS	–	Denial of Service, Атаки «відмови в обслуговуванні»
DDoS	–	Distributed Denial-of-Service, атака відбувається одночасно з великої кількості IP-адрес
HTTP	–	HyperText Transfer Protocol, протокол передачі гіпертекстових документів
ICMP	–	Internet Control Message Protocol, міжмережевий протокол керуючих повідомлень
SYN	–	Синхронізація (Synchronize) використовується для встановлення з'єднання між хостами при так званому триходовому рукостисканні
TCP	–	Протоко́л керува́ння передачею – разом із протоколом IP є стрижневим протоколом Інтернету, який дав назву моделі TCP/IP
TCP SYN Flood	–	атака за допомогою переповнення SYN-пакетами, коли зловмисники використовують тристороннє рукостискання за протоколом TCP, щоб викликати збої в роботі мережі й сервісів
UDP	–	Протокол датаграм користувача – один із протоколів в стеку TCP/IP

ВСТУП

Актуальність теми. У рамках даної роботи розберемося в суті атаки TCP SYN Flood. Крім того змоделюємо дану DoS-атаку зловмисників для тестових цілей за допомогою передвстановленої програми-генератора пакетів hping3 дистрибутива Kali Linux, а також як правильно й швидко ідентифікуємо атаку TCP SYN Flood, використовуючи аналізатор мережних протоколів розробленого в даній роботі програмного забезпечення системи кібербезпеки моделювання й визначення DoS атаки типу TCP SYN Flood.

Атаки «відмови в обслуговуванні» (Denial of Service), сумно відомі також як DoS-атаки, досить прості в проведенні, далеко не завжди очевидні й здатні стати причиною серйозних збоїв у роботі обчислювальної системи, що неминуче приведе до збільшення часу простою ваших системних ресурсів. При атаці за допомогою переповнення SYN-пакетами (TCP SYN Flood) зловмисники використовують тристороннє рукостискання за протоколом TCP, щоб викликати збої в роботі мережі й сервісів. Атаки такого типу можуть легко застати вас зненацька, тому що найчастіше системним адміністраторам буває складно їх швидко ідентифікувати. На щастя, такі інструменти, як розроблене в даній роботі програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типу TCP SYN Flood, спрощують захват і перевірку будь-яких підозрілих активностей, які можуть виявитися DoS-атакою.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

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

– Огляд існуючих систем кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

– Дослідження системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

– Програмна реалізація системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моделювання й визначення DoS атаки типа TCP SYN Flood.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

DoS-атака (від англ. Denial of Service – «відмова в обслуговуванні») – атака, яка використовується для виведення з л аду й злому обчислювальної техніки й створення технічних і економічних труднощів у цілі. Здійснюється за допомогою створення великої кількості запитів і серйозного навантаження на техніку, найчастіше на великі сервера.

Чому використовуються DoS-атаки

Популярність DoS-атак обумовлена тим, що визначити виконавця вкрай складно – він створює велике навантаження через множину комп'ютерів у мережі. Найчастіше такі атаки використовуються тоді, коли зламати систему або сервер не виходить. Навіть якщо за допомогою DoS-атаки не вдається одержати доступ, техніка виходить із ладу або втрачає продуктивність.

Жертвами таких атак найчастіше стають урядові сайти, великі портали, онлайн-змі, сервери онлайн-ігор, інтернет-магазини, корпоративні сайти фінансового сектору. Мотивація атакуючого також залежить від сфери. Найчастіше атака відбувається із причин політичних протестів, несумлінної конкуренції, вимагань і шантажу, особистої ворожості, а також з метою розваги.

Класифікація DoS-атак

– Насичення інтернет-каналу пошуком трафіком (флудом) створює велика кількість запитів до системи або сервера, за рахунок цього відбувається швидке вичерпання його ресурсів, що приведе до відключення або некоректній роботи. Флуд буває різних типів:

– HTTP-флуд і ping-флуд – одні з найпростіших і доступних видів атак: за допомогою ping-запитів на комп'ютер «жертви» з меншим каналом інтернету. HTTP-флуд застосовується для серверів, відправляється HTTP-пакет, на який

сервер шле відповідні пакети, що перевищують розміри, за рахунок чого пропускні здатності сервера знижуються.

– ICMP-флуд – атака за допомогою ICMP-пакета, який через посилюючу мережу здатний вивести з ладу будь-який комп'ютер, сервер, якщо розмір мережі нараховує велика кількість комп'ютерів.

– UDP-флуд – аналог ICMP-атаки, тільки в цьому випадку використовується UDP-пакет і створюються echo-запити на сьомий порт жертви. За рахунок атаки виникає насичення смуги пропущення.

– Проблеми системи квотування – якщо на сервері погано настроєне квотування, те можна одержати доступ до CGI і задіяти скрипт, який буде використовувати велику кількість ресурсів комп'ютерної системи.

– Помилки програмування – якщо в програмному коді є помилки, професіонали в сфері DoS-атак використовують саме їх для того, щоб змусити систему виконати команду з помилкою, що приведе до аварійного вимикання.

– Атаки на DNS-сервера – вид атаки, який використовує множину комп'ютерів-зомбі й шляхом захвата системних ресурсів або насичення смуги виводить із ладу оброблювач доменного імені на IP адресу. Це робить сторінку в інтернеті недоступної для користувачів.

Захист від DoS-атак

Наявність DoS-атаки неможливо не помітити по навантаженню на сервер і супутнім проблемам і збоям. Але потрібно знати хоча б основні приймання по захисту від подібних атак:

– Для захисту від HTTP-флуда збільшується кількість одночасних підключень. Робиться це за допомогою установки продуктивного веб-сервера Nginx, кешуючого запити.

– ICMP-флуд можна запобігти, відключивши на комп'ютерній системі відповіді на запити ICMP ECHO.

– Якщо обмежити кількість з'єднань до DNS-серверу й відключити від зовнішнього виходу UDP-сервіси, можна уникнути атаки з UDP-флудом.

– Якщо відключити черга «напіввідчинених» портів TCP-з'єднань, можна захистити систему від SYN-флуда.

Існують також універсальні способи захисту від різного роду DoS-атак:

– Включати й набудовувати брандмауер для мережних сервісів.

– Використовувати сервіси захисту від даних атак.

– Збільшувати ресурси системи, сервера.

1.2 Область застосування

TCP SYN Flood

Клієнт генерує SYN-пакет, запитуючи нову сесію в сервера. Оскільки TCP сесія відкрита (алгоритм “триетапного рукостискання TCP” виконаний), хост буде відслідковувати й обробляти кожну користувацьку сесію, поки вона не буде закрита. Під час SYN Flood, що атакується сервер з великою швидкістю одержує підроблені SYN-запити, що містять підроблену IP-адресу джерела. SYN-флуд вражає сервер, займаючи всю пам'яті таблиці з'єднань (Transmission Control Block (TCB) table), звичайно використовувану для зберігання й обробки вхідних пакетів. Це викликає критичне падіння продуктивності й, як підсумки, відмова в роботі сервера.

Існує кілька механізмів захисту, які можуть частково убезпечити від SYN Flood:

– Обмеження мікро-блоків: будучи адміністратором, Ви можете обмежити розмір пам'яті сервера для кожного, що водить SYN-пакета;

– SYN-куки (SYN-cookie): використовуючи криптографічне гешування, сервер відправляє SYN-ACK відповідь із номером послідовності, яка складена з IP-адреси клієнта, номера порту й іншої унікальної інформації, що ідентифікує клієнта. Коли клієнт відповідає, цей геш уже включений в ACK-пакет. Далі сервер перевіряє ACK і, у випадку успішної перевірки, йому залишається тільки

виділити пам'ять для з'єднання. Для використання цього механізму необхідно, щоб його підтримували все сервера, що брати участь в обміні трафіком;

– Налаштування стека: у якості однієї з часових заходів можна настроїти стек TCP, зменшивши тайм-аут звільнення пам'яті, виділеної для з'єднання, а також тайм-аут блокування вхідних з'єднань. Але в цих налаштувань можуть бути побічні ефекти у вигляді втрати частини легітимних з'єднань через затримки й нестабільних каналів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood, 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

Solarwinds Real-Time NetFlow Traffic Analyzer

Free NetFlow Traffic Analyzer є одним з найбільш популярних інструментів, доступних для безкоштовного завантаження. Він дає можливість сортувати, позначати й відображати дані різними способами. Це дозволяє зручно візуалізувати і аналізувати мережний трафік. Інструмент відмінно підходить для моніторингу мережного трафіку по типах і періодам часу. А також виконання тестів для визначення того, скільки трафіку споживають різні застосунки.

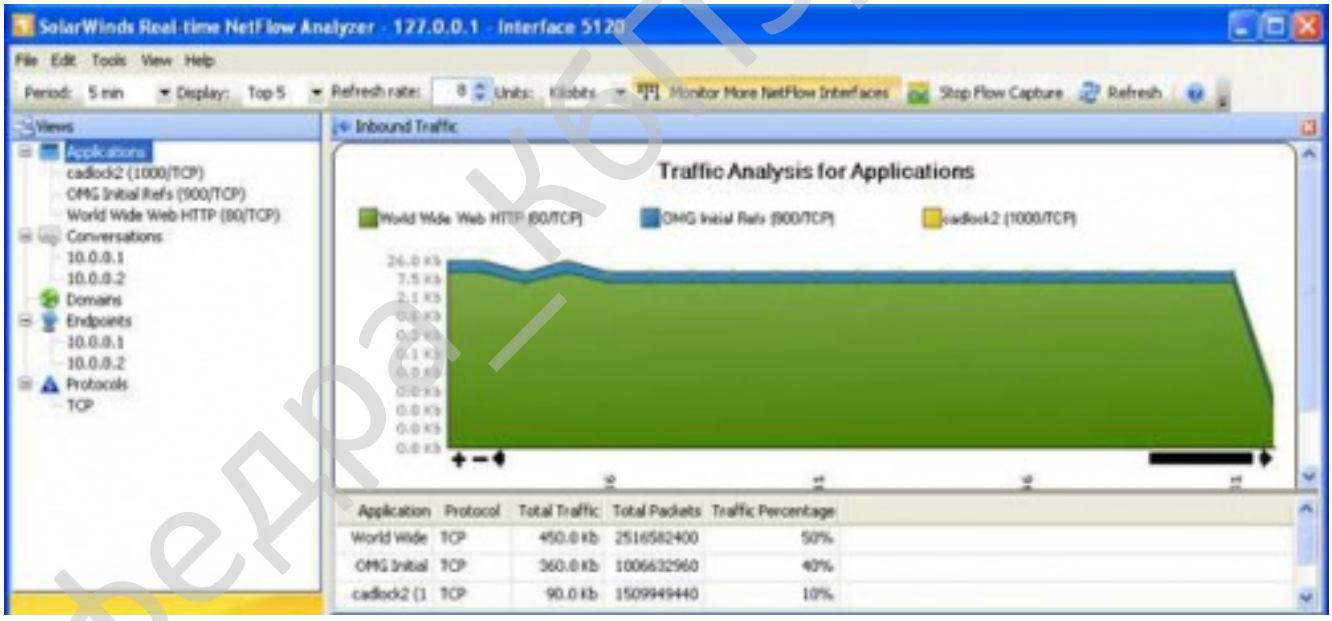

Рисунок 2.1 – Інтерфейс користувача NetFlow Traffic Analyzer

Цей безкоштовний інструмент обмежено одним інтерфейсом моніторингу NetFlow і зберігає тільки 60 хвилин даних. Даний NetFlow аналізатор є потужним інструментом, який коштує того, щоб його застосувати.

Colasoft Capsa Free

Цей безкоштовний аналізатор трафіку локальної мережі дозволяє ідентифікувати й відслідковувати більш 300 мережних протоколів, і дозволяє створювати звіти, що налаштовуються. Він містить у собі моніторинг електронної пошти й діаграми послідовності TCP-синхронізації, усе це зібране в одній панелі, що настроюється.

Рисунок 2.2 – Інтерфейс користувача Colasoft Capsa Free

Інші функції містять у собі аналіз безпеки мережі. Наприклад, відстеження DoS/DDoS-атак, активності хробаків і виявлення ARP-атак. А також декодування пакетів і відображення інформації, статистичні дані про кожний хост в мережі, контроль обміну пакетами й реконструкція потоку. Capsa Free підтримує всі 32-бітні й 64-бітні версії Windows XP.

Мінімальні системні вимоги для установки: 2 Гб оперативної пам'яті й процесор 2,8 Ггц. У вас також повинне бути з'єднання з інтернет по мережі Ethernet (сумісної з NDIS 3 або вище), Fast Ethernet або Gigabit із драйвером зі змішаним режимом. Він дозволяє пасивно фіксувати всі пакети, передані по Ethernet-кабелю.

Angry IP Scanner

Це аналізатор трафіку Windows з відкритим вихідним кодом, швидкий і простій у застосуванні. Він не вимагає установки й може бути використаний на Linux, Windows і MAC OSX. Даний інструмент працює через просте пінгування кожного IP-адреси й може визначати MAC-адреси, сканувати порти, надавати Netbios-інформацію, визначати авторизованого користувача в системах Windows, виявляти веб-сервери й багато чого іншого. Його можливості розширюються за допомогою Java-плагінів. Дані сканування можуть бути збережені у файли форматів CSV, TXT, XML.

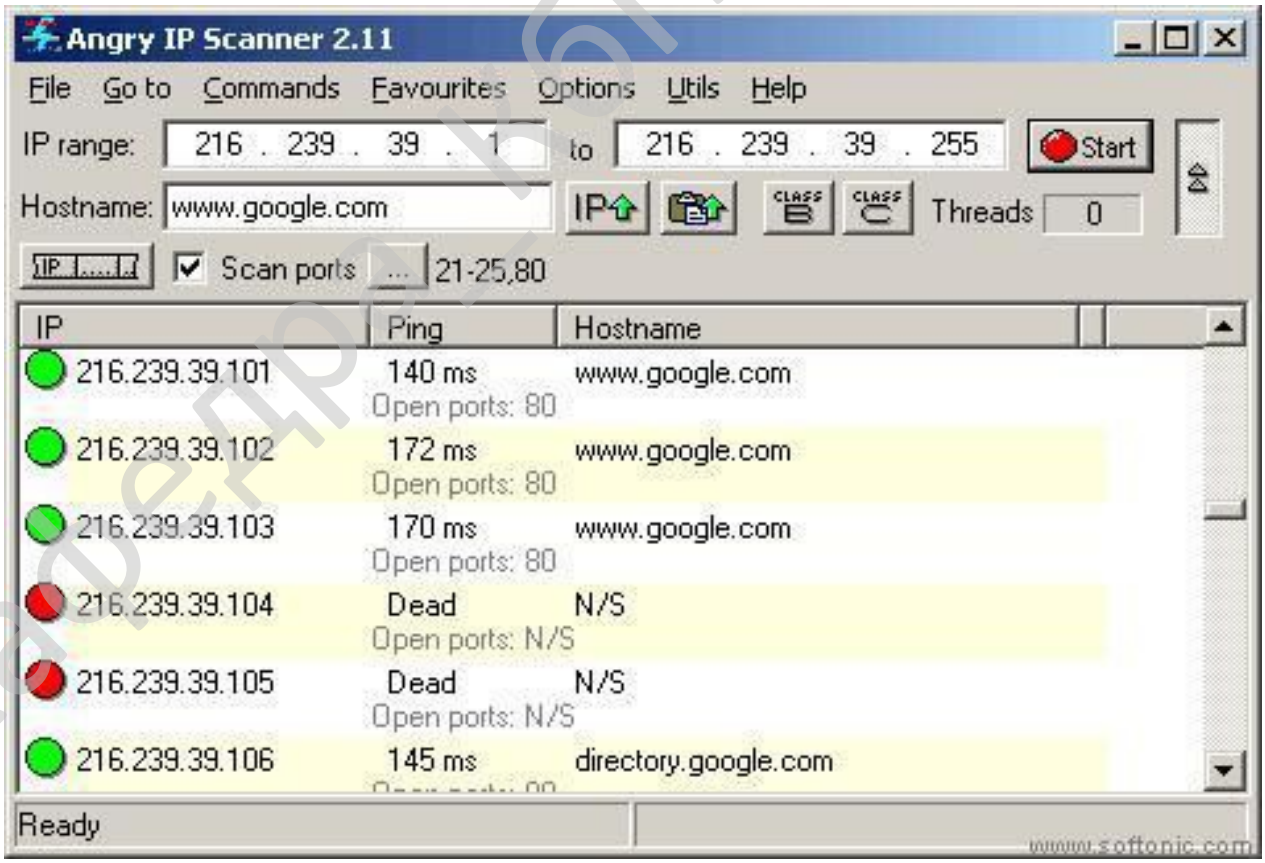

Рисунок 2.3 – Інтерфейс користувача Angry IP Scanner

Manageengine NetFlow Analyzer Professional

Повнофункціональна версія програмного забезпечення NetFlow від ManageEngines. Це потужне програмне забезпечення з повним набором функцій для аналізу й збору даних: моніторинг пропускної здатності каналу в режимі реального часу й оповіщення про досягнення граничних значень, що дозволяє оперативно адмініструвати процеси. Крім цього передбачений вивід зведених даних по використанню ресурсів, моніторинг застосунків і протоколів і багато чого іншого.

Рисунок 2.4 – Інтерфейс користувача Manageengine NetFlow Analyzer Professional

Безкоштовна версія аналізатора трафіку Linux дозволяє необмежено використовувати продукт протягом 30 днів, після чого можна робити моніторинг тільки двох інтерфейсів. Системні вимоги для NetFlow Analyzer Manageengine

залежать від швидкості потоку. Рекомендовані вимоги для мінімальної швидкості потоку від 0 до 3000 потоків у секунду: двоядерний процесор 2,4 Ггц, 2 Гб оперативної пам'ятей і 250 Гб вільного простору на жорсткому диску. У міру збільшення швидкості потоку, який потрібно відслідковувати, вимоги також зростають.

The Dude

Цей застосунок являє собою популярний мережний монітор, розроблений Mikrotik. Він автоматично сканує всі пристрої й відтворює карту мережі. The Dude контролює сервери, що працюють на різних пристроях, і попереджає у випадку виникнення проблем.

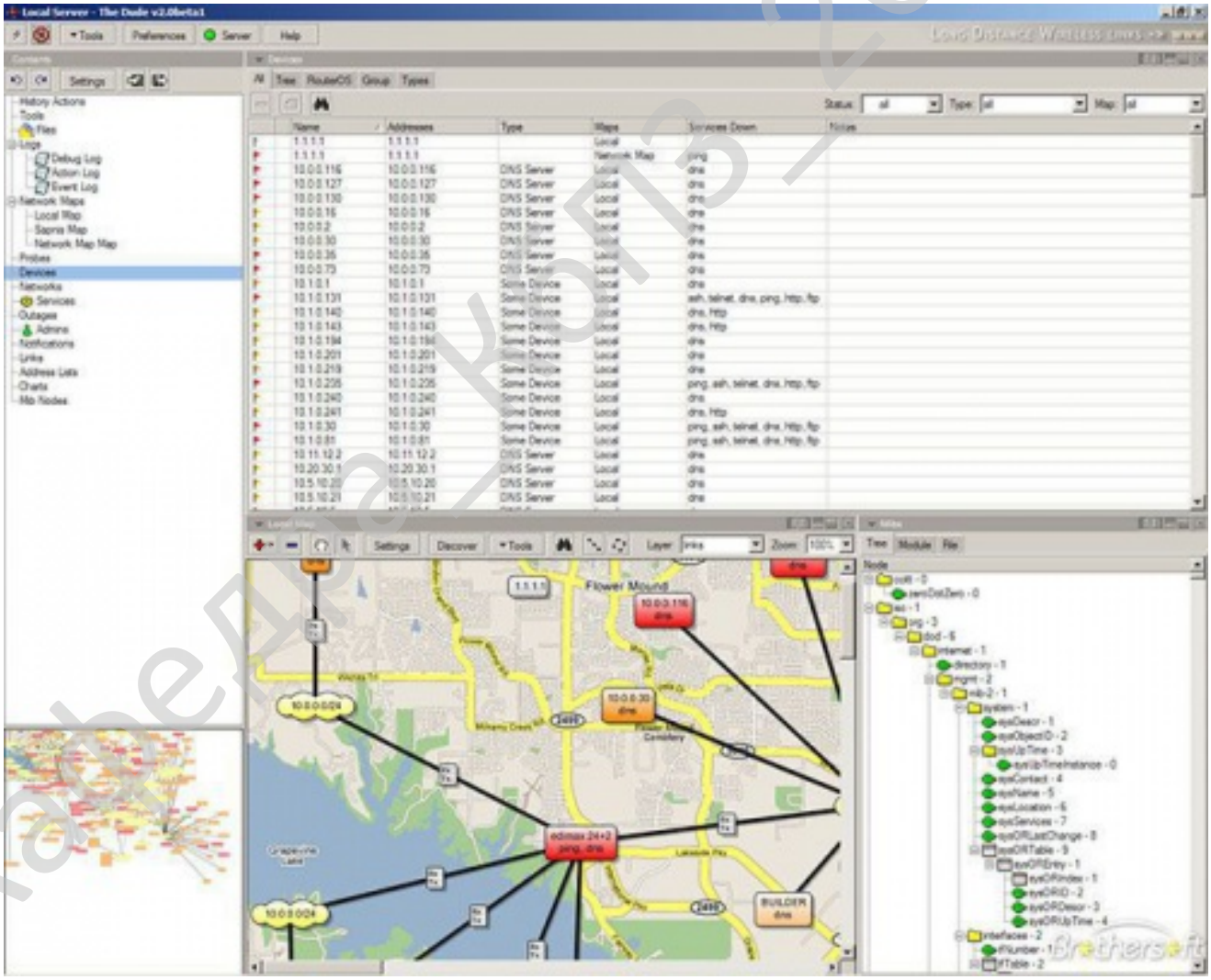

Рисунок 2.5 – Інтерфейс користувача The Dude

Інші функції містять у собі автоматичне виявлення й відображення нових пристроїв, можливість створювати власні карти, доступ до інструментів для віддаленого керування пристроями й багато чого іншого. Він працює на Windows, Linux Wine і MacOS Darwine.

JDSU Network Analyzer Fast Ethernet

Ця програма аналізатор трафіку дозволяє швидко збирати й переглядати дані по мережі. Інструмент надає можливість переглядати зареєстрованих користувачів, визначати рівень використання пропускної здатності мережі окремими пристроями, швидко знаходити й усувати помилки. А також захоплювати дані в режимі реального часу й аналізувати їх.

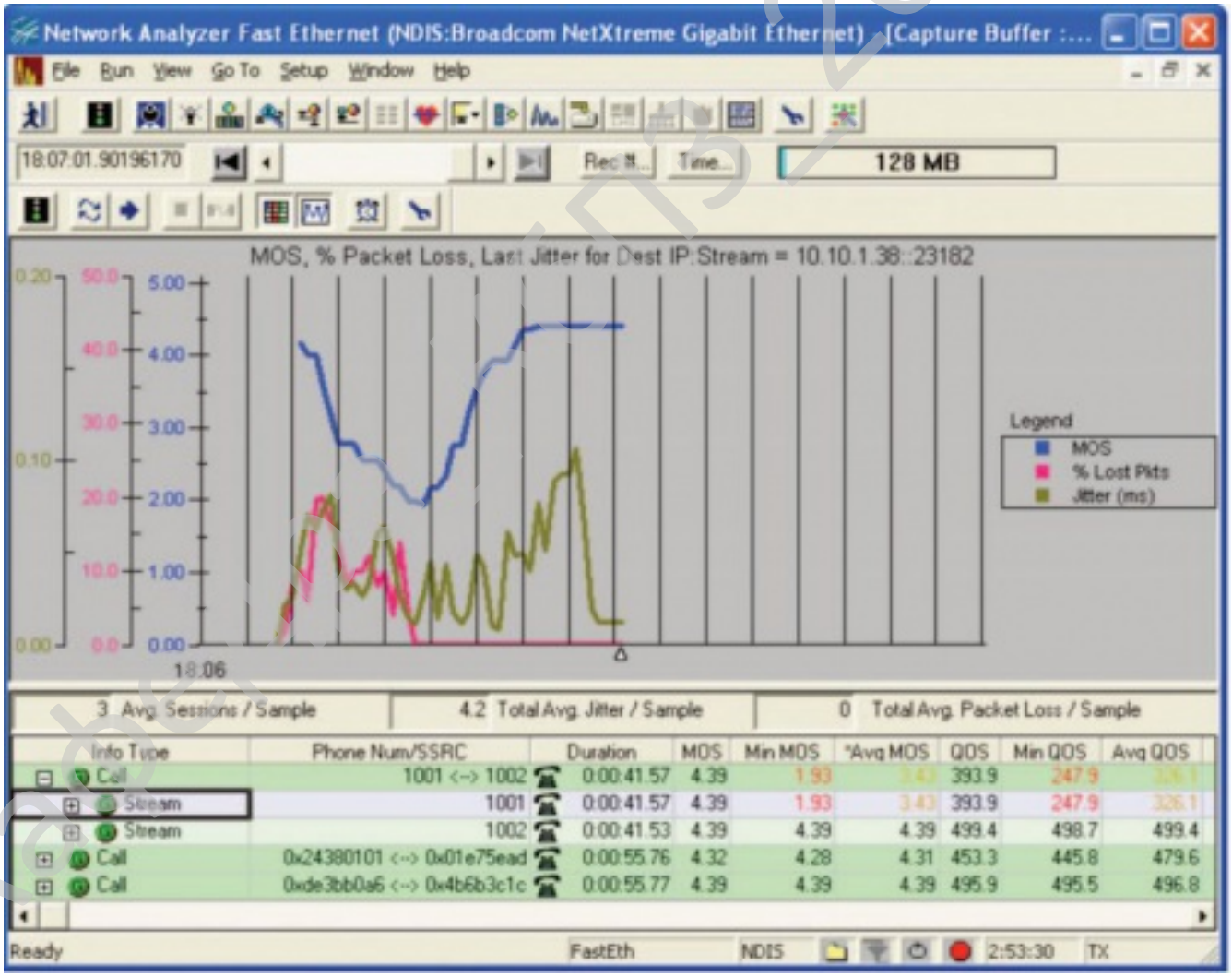

Рисунок 2.6 – Інтерфейс користувача JDSU Network Analyzer Fast Ethernet

Застосунок підтримує створення графіків і таблиць із високою деталізацією, які дозволяють адміністраторам відслідковувати аномалії трафіку, фільтрувати дані, щоб просівати більші обсяги даних, і багато чого іншого. Цей інструмент для фахівців початкового рівня, а також для досвідчених адміністраторів, дозволяє повністю брати мережу під контроль.

Plixer Scrutinizer

Цей аналізатор мережного трафіку дозволяє зібрати й всебічно проаналізувати мережний трафік, а також швидко знайти й виправити помилки.

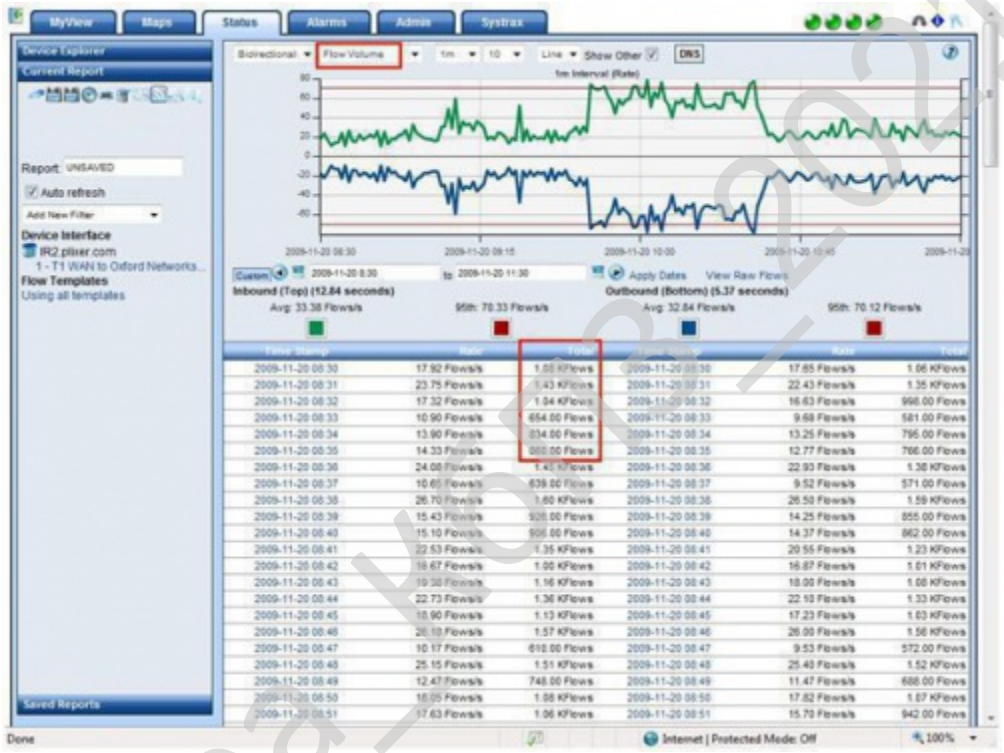

Рисунок 2.7 – Інтерфейс користувача Plixer Scrutinizer

За допомогою Scrutinizer можна відсортувати дані різними способами, у тому числі по часових інтервалах, хостам, застосункам, протоколам і т.д. Безкоштовна версія дозволяє контролювати необмежена кількість інтерфейсів і зберігати дані по 24 годинникові активності.

Wireshark

Wireshark – це потужний мережний аналізатор може працювати на Linux, Windows, MacOS X, Solaris і інших платформах. Wireshark дозволяє переглядати захоплені дані за допомогою графічного інтерфейсу, або використовувати утиліти TTY-mode Tshark. Його функції містять у собі збір і аналіз трафіку VoIP, відображення в режимі реального часу даних Ethernet, IEEE 802.11, Bluetooth, USB, Frame Relay, вивід даних в XML, Postscript, CSV, підтримку дешифрування й багато чого іншого.

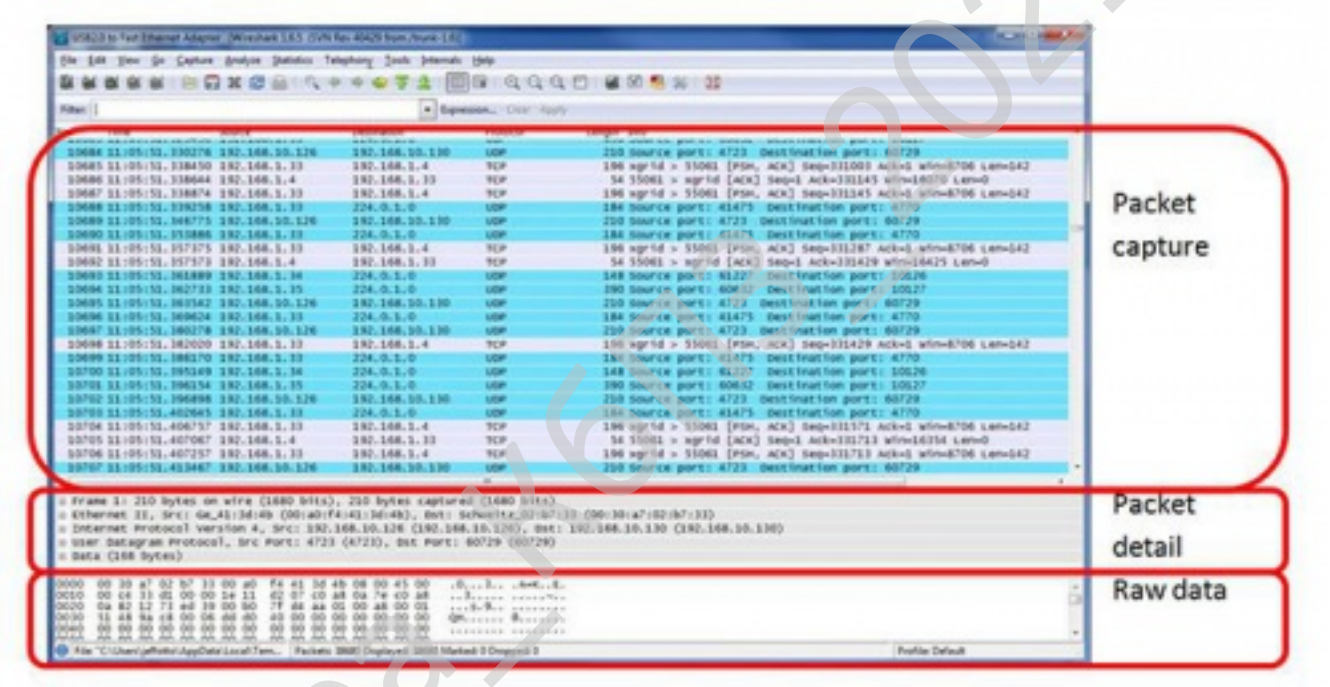

Рисунок 2.8 – Інтерфейс користувача Wireshark

Системні вимоги: Windows XP і вище, будь-який сучасний 64/ 32-бітний процесор, 400 Mb оперативної пам'яті й 300 Mb вільного дискового простору. Wireshark NetFlow Analyzer – це потужний інструмент, який може суттєво спростити роботу будь-якому адміністраторові мережі.

Paessler PRTG

Цей аналізатор трафіку надає користувачам множину корисних функцій: підтримку моніторингу LAN, WAN, VPN, застосунків, віртуального сервера, Qos і середовища. Також підтримується моніторинг декількох сайтів. PRTG використовує SNMP, WMI, NetFlow, SFlow, JFlow і аналіз пакетів, а також моніторинг часу безперебійної роботи/простою й підтримку Ipv6.

Рисунок 2.9 – Інтерфейс користувача Paessler PRTG

Безкоштовна версія дає можливість використовувати необмежену кількість датчиків протягом 30 днів, після чого можна безкоштовно використовувати тільки до 100 штук.

nProbe

Це повнофункціональний застосунок з відкритим вихідним кодом для відстеження й аналізу NetFlow.

nProbe підтримує Ipv4 і Ipv6, Cisco NetFlow v9 / IPFIX, NetFlow-lite, містить функції аналізу VoIP трафіку, вибірки потоків і пакетів, генерації балок, Mysql/Oracle і DNS-активності, а також багато чого іншого. Застосунок є

безкоштовним, якщо ви аналізатор трафіку завантажуєте й компілюєте на Linux або Windows. файл, що виконується, установки обмежує обсяг захвата до 2000 пакетів. nProbe є повністю безкоштовним для освітніх установ, а також некомерційних і наукових організацій. Даний інструмент буде працювати на 64-бітних версіях операційних систем Linux і Windows.

Рисунок 2.10 – Інтерфейс користувача nProbe

Цей список з 10 безкоштовних аналізаторів трафіку й колекторів NetFlow допоможе вам приступитися до моніторингу й усуненню несправностей у невеликій офісній мережі або великий, що охоплює кілька сайтів, корпоративної WAN-мережі.

Кожне представлене в цьому розділі застосунок дає можливість контролювати й аналізувати трафік у мережі, виявляти незначні збої, визначати аномалії пропускного каналу, які можуть свідчити про погрози безпеки. А також візуалізувати інформацію про мережу, трафік і багато чого іншого. Адміністратори мереж обов'язково повинні мати у своєму арсеналі подібні інструменти.

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 – використовуючи 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 множину 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 використовуються 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 – використовуючи 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 множину 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 використовують 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 моделювання й визначення DoS атаки типа TCP SYN Flood.

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

Інтернет стрімко перетворюється в ринок послуг з дуже великим обігом коштів. Збій або неприступність інформаційного сервера тягне величезні фінансові втрати. Останнім часом стала особливо актуальної проблема DoS/DDoS-атак. Їхня потужність збільшилася майже в 3.5 рази [1] тільки за четвертий квартал 2020 року. Для забезпечення інформаційної безпеки в організаціях необхідно вміти відбивати або запобігати DoS/DDoS-атаки, але на сьогоднішній день немає ефективного застосунку.

DoS (Denial of Service – відмова в обслуговуванні) – хакерська атака на обчислювальну систему з метою довести її до відмови, тобто створення таких умов, при яких легальні користувачі системи не можуть одержати доступ до надаваних системних ресурсів (серверам), або цей доступ утруднений. DDoS (Distributed Denial of Service – розподілена атака типу «відмова в обслуговуванні») – це атака, що виконується одночасно з великої кількості комп'ютерів. Ціль атакуючих – зробити в Інтернеті недоступним той або інший ресурс. У цей час DoS і DDoS-атаки найбільш популярні, тому що дозволяють довести до відмови практично будь-яку систему, не залишаючи юридично значимих доказів [2].

На даний момент існує множина методів боротьби з DoS – атаками, але окремо друг від друга їх застосування неефективне, тому що при виникненні DoS – атаки незрозуміло, до якого типу вона ставиться. Універсальних рецептів протидії DoS/DDoS атакам немає. Але вже зараз можна почати ряд заходів щодо зниження ймовірності реалізації таких атак, ґрунтуючись на їхній класифікації. Саме вона допомагає швидко виявити початок DoS-атаки й використовувати відомі методи боротьби для її запобігання.

Класифікація DoS-атак [3]

На даний момент виділяють 4 типу DoS-атак:

- атака на насичення смуги пропущення;
- атака, що приводить до недоліку системних ресурсів;
- атака, що використовує помилки програмування;
- атака на DNS-сервери.

Кожний із цих типів має свої особливості.

Атака на насичення смуги пропущення

Звичайно зловмисники для атаки на цільову систему використовують флуд (flood – «повінь», «переповнення») – атака, пов'язана з більшою кількістю звичайно безглуздих запитів до комп'ютерної системи або мережного встаткування, що має своєю метою або, що приводить до відмови в роботі системи через насичення смуги пропущення каналу зв'язки. Є кілька різновидів флуда: HTTP-флуд, ICMP-флуд, UDP-флуд, SYN-флуд і ін.

Методи боротьби

За замовчуванням усі сервери повинні бути оновлені до останньої версії патча захисту, якщо така є. Також повинні бути відключені всі сервіси, які не використовуються в мережі.

Один з методів боротьби з DDoS-атаками, розрахованими на насичення смуги пропущення каналу зв'язки, є постійна зміна IP-адреси комп'ютера-жертви. Даний вид захисту зветься «оборонної цілі, що рухається, ». Після зміна IP-адреси маршрутизатори скинуть шкідливі пакети, які надходили на конкретний IP-адресу.

Відключення широкомовної передачі повідомлень допомагає в боротьбі проти ICMP-флуда. У цьому випадку хости більше не будуть використовуватися в якості підсилювачів DDoS-атаки.

Так само створюється спеціальна база маршрутів, по яких звичайно йдуть пакети в штатному режимі роботи системи. Це дозволяє захиститися від UDP, TCP і ICMP-флудів.

Атака, що приводить до недоліку системних ресурсів

У цьому випадку DoS – атака спрямована на захвата системних ресурсів комп'ютера, не зачіпаючи канал зв'язку. Процесор сервера може не впоратися зі складними обчисленнями. Відбудеться збій через те, що центральний процесор або оперативна пам'ять виявляться недоступні. Прикладами таких атак є: відправлення «важких пакетів», переповнення сервера балка-файлами, атака на погано організовану систему квотування або на відсутність перевірки даних користувача.

Методи боротьби

Існують різні системи виявлення вторгнень, які дозволяють виявляти DoS-атаки по відомих сигнатурах, розпізнаючи аномальну поведінку системи.

Атака, що використовує помилки програмування

Ще даний вид атаки називають атаками на застосунки. Цей вид атак найнебезпечніший і спричиняє важкі наслідки для сервера, тому що плече атаки (відношення ресурсів необхідних на стороні додатка до ресурсів на атакуючій стороні) у цьому випадку максимально. Більш просунуті хакери, повністю розібравшись у структурі системи або додатка, пишуть спеціальні програми – експлойти, які допомагають атакувати комерційні підприємства. Прикладом таких атак є просте переповнення буфера. Часто хакери використовують недоліки в програмному коді.

Методи боротьби

Створюється база шаблонів відомих експлойтів і проводиться моніторинг входжень даних шаблонів у саму систему.

Аналіз журналу подій після й під час DoS-атаки допомагає виявити нові типи атак, або модифікації старих.

Атака на DNS-сервери

Даний вид атаки самий дорогий і ресурсозатратний. Хакери, використовуючи великі ботнети, атакують урядові організації, дуже великі компанії, або відомі інтернет-магазини. Сервер-жертва може перебувати в

простої досить тривалий час. Через цей компанія зазнає величезних збитків [4]. В 2012 році хакери групи Anonymous прагли вивести з ладу глобальну мережу, шляхом атаки на 13 кореневих серверів DNS [5]. У цьому ж році найбільший у світі хост – провайдер Go Daddy і разом з ним більш 33 мільйонів доменов по усьому світу постраждали від DDoS-атаки [6]. Однією із самих потужних атак в історії вважається атака на компанію Spamhaus в 2012 році. Вона протривала кілька днів і досяглася своєї пікової потужності в 300 Гб/с [7].

Методи боротьби

Одним з ефективних методів боротьби є перенапрямок трафіку на майданчик могутнішого провайдера. За рахунок цього збільшується смуга пропущення каналу зв'язки, трафік фільтрується й уже легітимний трафік іде до користувача.

Захист від DoS/DDoS-атак

Від будь-якої атаки можна захиститися, але в цей час немає універсального способу захисту. Повної узагальненої класифікації ні, у тому числі й у тій, що представлена, змішані різні рівні абстракції. Одні компанії використовують статичні засоби захисту до початку самої атаки, вишиковуючи спеціальні фільтри. Вони у свою чергу навчаються на маршрутах сайту, по яких ходять звичайні користувачі. Відбувається моніторинг Інтернет-трафіку в реальному часі. Інші – зберігають базу даних сигнатур уже відомих DoS-атак. Роблячи при цьому моніторинг усієї системи по сигнатурах, можна зафіксувати початок DoS-атаки. Але, як і у випадку з вірусами, даний спосіб не так ефективний, тому що з появою який -небудь нової DoS-атаки дана база сигнатур пошук. Ще один спосіб виявлення початку DoS-атаки надає собою виявлення аномальної поведінки в системі. Поточний стан системи в реальному часі рівняється з її нормальним станом, тобто тем, коли спостерігається звичайна динаміка трафіку й продуктивності самого сервера [8]. Усі ці способи допомагають захиститися лише від деяких видів DoS-атак. Хакери, націлені «упустити» той або інший сервер, аналізують захист і придумують інші способи

впливу на неї, підсилюючи атаку, або зовсім прибігаючи до іншого методу. Коштує так само відзначити той факт, що протидії різним видам DoS/DDoS-атак є частиною засобів і методів інформаційної безпеки в загальному значенні. Тому коштує актуальне завдання інтеграції наведених вище застосунків боротьби з DoS-атаками у вже відомі методи захисту інформації, до DoS-атакам не стосовні.

Таким чином, хочеться одержати срібну кулю, яка зможе вразити не тільки всі комп'ютери-зомбі, але ліквідувати експлойти, флуд і іншу погань.

Приватні підходи занадто ресурсозатратні й у силу в еличезної різноманітності DoS-атак, не забезпечують систему повноцінним захистом. Але правильна класифікація – це перший крок до створення комплексного розширюваного апарата, здатного детектувати, вживати необхідних заходів на початку атак і, у деяких випадках, повністю захищати систему від них. Також варто враховувати той факт, що за останній час збільшилася потужність DoS-атак, а разом із цим збільшилися й витрати на її реалізацію. Тому, не маючи 100% захисту від них, але ґрунтуючись на існуючій класифікації DoS-атак можна вибудувати захист своєї системи так, що дані атаки стали б просто економічно невигідними. Для цього, у міру можливості, на озброєнні треба мати всі відомі способи захисту системи від атак.

DDoS-атаки (Distributed Denial of Service)

DDoS-атака (Distributed Denial of Service) – хакерська атака на веб-сайт, головне завдання якої – привести до відмови в обслуговуванні, при якому взаємодія користувачів із сервісами й сайтами буде утруднено або неможливе. Її відмінність від DoS-атаки – у тому, що вона проводиться відразу з множині пристроїв і адрес. Для DDoS-атак хакери збирають ботнети із заражених шкідливими програмами комп'ютерів-зомбі.

Ціль проведення DDoS-атаки – вимагання грошей за її припинення й за відновлення доступу до веб-сайту. Найчастіше зловмисники піддають таким нападам мережні ресурси електронної комерції, онлайн-банки, системи

бронювання, букмекерські контори, інформаційні сервіси, ЗМІ й інші організації, що ведуть свій бізнес у Мережі.

Існують спеціальні шкідливі програми (боти), які дозволяють формувати ботнети. Наприклад, інтернет-хробак Mirai заразив більш 500 000 пристроїв, підключених до інтернету, з яких був сформований однойменний ботнет для рекордно потужних на той момент DDoS-атак.

DDoS-атаки знайшли першу популярність в 1999 році, коли зловмисники атакували веб-сайти найбільших компаній (Yahoo, ebay, Amazon, E-Trade, CNN і багатьох інших). Через рік після атак на великі корпорації була усвідомлена необхідність вжити термінових заходів для боротьби із проблемою, що з'явився.

Класифікація DDoS-атак

Основні способи DDoS-атак:

– HTTP-флуд. Найпоширеніший спосіб, чия основна ідея – відправлення серверу такого пакета, відповіддю на який буде пакет набагато більшого розміру. У спеціально сформованому запиті до сервера зловмисник заміняє свій IP-адресу на мережний ідентифікатор машини усередині сітки-жертви.

– ICMP-флуд. При цьому типі DDoS-атаки хакер відправляє ICMP-пакет (часто – за допомогою утиліти ping) посилюючої мережі. IP-адресу зловмисника в цьому випадку також заміняється цільовим, і на сервер-жертву приходить відповідь на команду, збільшений у стільки раз, скільки машин містить посилююча мережа. Також подібна атака може відбуватися за допомогою UDP-пакетів.

– SYN-флуд. Для обміну даними комп'ютерним системам потрібна установка з'єднання, і при цьому на саме з'єднання теж виділяються комп'ютерні ресурси – на які й націлений даний вид атак. Відправляючи неправильні запити, можна використовувати всі ресурси комп'ютерної системи, які зарезервовані на установку з'єднань.

– «Важкі пакети». Для реалізації цього методу атаки зловмисник за допомогою ботнета відправляє серверу важкі для обробки пакети даних, які не

переповняють канал зв'язку, але віднімають ресурси процесора, що може привести до його перегріву або перевантаження.

Об'єкт впливу

Ціль DDoS-атаки – вивід з ладу або неприступність веб-сайту. Однак буває так, що метою DDoS виявляється DNS-сервер. Також метою може стати вразливий веб-застосунок. Окремі DDoS-атаки організовуються для розваги або в знак політичного протесту. Часто DDoS проводять для шантажу або вимагання. Від цього щорічно страждає величезна кількість компаній і приватних осіб, адже через атаки їх сайти стають недоступними клієнтам і не приносять доходу. Мережні ресурси державних установ, сайти ЗМІ, інтернет-магазини й онлайн-банки, портали комерційних і некомерційних організацій – усі вони є потенційними цілями DDoS-атак.

У середині 2010-х років мало місце деякий затишок, але згідно зі звітом компанії Qrator Labs, в 2016 році DDoS-атаки почали знову тривожити корпоративних користувачів. Незважаючи на те, що багатьом провайдерам легко нейтралізувати атаки потужністю до 300 Гбіт/с, проблеми все-таки залишилися. Зокрема, зловмисники почали використовувати заражені сервери відеозапису, веб-камери, пристрої інтернету речей, у яких є уразливості. Через поширеність таких пристроїв атаки стали ще більш масштабними.

На думку Qrator Labs, технічним фахівцям необхідно знову звертати увагу на захист від D DoS-атак. Якщо колись спостерігалося лінійне збільшення потужності останніх, то в 2016 році ситуація різко змінилася. Сьогодні атаки можуть досягати таких масштабів, що їм під силу накривати цілі регіони земної кулі, а це прямо загрожує функціонуванню роботи великих провайдерів.

Найбільшу «увагу» кіберзлочинці приділили наступним галузям:

- купонні сервіси,
- платіжні системи,
- інформаційні агрегатори,
- електронна комерція,

– гри й ігрові майданчики.

Джерело DDoS-атак

Джерел нападів, спрямованих на відмову в обслуговуванні, множину: конкуренти, недоброзичливці, хактивисти і т.д. Найчастіше DDoS-атакам зазнає великий бізнес (20%).

Аналіз ризику

Протистояння DDoS-атакам – складне завдання, запити до сайту надходять із багатьох напрямків. Від слабких DDoS-атак можна захиститися: наприклад, від HTTP-флуда допоможе установка ліміту підключень, від ICMP-флуда – відключення відповідей на всі запити ECHO або правильно настроєний WAF, від UDP-флуда – відключення від інтернету UDP-сервісів і установка ліміту звертань до DNS-серверу. Однак проти більшості атак, організованих професійними кіберзлочинцями й націлених на максимально можливий обсяг трафіку, налаштування веб-сервера нічого не дасть, тому що буде «забитий» сам канал зв'язку. У цьому випадку зможуть допомогти тільки спеціальні сервіси захисту.

3.2 Розробка структурної схеми

У рамках сервісу використовується просунутий метод перевірки й керування мережним трафіком, який виявляє, ідентифікує, класифікує, перенаправляє або блокує пакети з конкретними даними або корисним навантаженням, які звичайна фільтрація пакетів (перевіряє тільки заголовки пакетів) не може виявити.

Крім вивчення пакетів по якихось стандартних правилах, що дозволяють однозначно визначити приналежність пакета до конкретного додатка, скажемо, по формату заголовків, номерам портів і т.п., використовувана система здійснює й так званий поведінковий аналіз трафіку. Ц аналіз дозволяє розпізнавати

застосунок, що не задіють для обміну даними заздалегідь відомі заголовки і структури даних

Використання глибокого аналізу мережного трафіку дозволяє спростити процеси підтримки й адміністрування мереж і сайтів, підвищує захищеність останніх від різного роду атак і розширює область застосовності технологій аналізу трафіку.

Розв'язувані завдання:

– Фільтрація трафіку за певними критеріями, включаючи доменні імена й протоколи (наприклад, складнорозрахунковий трафік P2P, використання Skype або торрент-клієнтів).

– Застосування як статистичних політик, вибираючи: кому, що й коли можна й з якими пріоритетами, так і динамічними, тобто абонентами можуть бути не тільки звичайні користувачі фіксованих мереж, але також, приміром, і бездротових, з усіма властивими ним атрибутами (можуть відслідковуватися APN, номера телефонів, спосіб доступу – GERAN, UTRAN і ін.).

– Запобігання атак (відносно мережних атак ззовні, таких як DDoS, сканування портів і ін.).

– Детектування атак зсередини й збір найрізноманітнішої статистики.

– Завдяки дзеркалюванню й перенапрямку трафіку з'являються додаткові можливості, як перевірка пошти на предмет спама або трафіку з веб-сайтів на віруси.

Глибока перевірка пакетів є коштовною практикою для багатьох цілей, починаючи від керування продуктивністю й закінчуючи аналітикою мережі, експертизою й безпекою підприємства.

Сценарій реалізації

Ми надаємо Вам можливість керування мережним трафіком, класифікованим по різних ознаках: IP-адресу, протокол або застосунок, часовий інтервал, тип інкапсуляції трафіку (VLAN, GRE) та ін.

Склад послуги:

– Підключення функціонала послуги для всієї доступної смуги пропущення основної послуги «Доступ у мережу Інтернет» з параметрами за замовчуванням.

– Автоматичне відновлення сигнатур із сервера відновлення постачальника сигнатур (можлива окрема розробка й додавання приватних сигнатур).

– Надання доступу до веб-інтерфейсу керування послугою через захищене VPN-з'єднання (дозволяє створювати й управляти віртуальними каналами в рамках потоку, а також формувати звіти).

– Надання бази стандартних політик (об'єктів) обробки класифікованого трафіку.

У рамках глибокого аналізу трафіку пропонується до використання ряд інших додаткових технологій, таких як VPN, аналіз шкідливих програм, антиспам-фільтрація, фільтрація URL-адрес і інші технології, що забезпечують більш комплексний захист мережі.

Для захисту від розподілених атак надаємо хмарний сервіс очищення трафіку. Функціонал застосунку дозволяє моделювати профілі роботи IT-систем у штатному режимі й відслідковувати будь-які відхилення в мережі з можливістю блокування атак і виявлення їх джерел. Режим захисту активується автоматично, згідно з настроєними політиками детектування атак.

У рамках послуги із захисту від DDoS-атак аналіз мережного трафіку проводиться в режимі 24/7. У результаті використання нашого сервісу Ви одержуєте тільки чистий, відфільтрований трафік. Таким чином, Ваші користувачі можуть і не довідатися, що на них була почата атака.

Результатом використання сервісів захисту від DDoS-атак буде своєчасне виявлення й запобігання DDoS-атак, безперервність функціонування інтернет-ресурсів і їх постійна доступність для користувачів, включаючи мінімізацію

фінансових і репутаційних втрат від простоїв корпоративного сайту або веб-порталу.

Ми здійснюємо моніторинг мережі в режимі 24/7 і застосовуємо спеціальні засоби для зм'якшення наслідків DDoS-атак.

Склад послуги:

– Підключення – активація захисту від DDoS-атак на інтернет-каналі з типовими параметрами.

– Захист смуги пропущення інтернет-каналу (виявлення атаки, очищення трафіку відповідно до встановлених політиків фільтрації й керування трафіком).

– Надання щомісячного статистичного звіту по фактах відбиття DDoS-атак.

Принцип роботи атаки TCP SYN Flood

Коли клієнт намагається підключитися до сервера з використанням протоколу TCP (наприклад, при встановленні HTTP- або HTTPs-з'єднання), він відразу повинен пройти процедуру тристороннього рукостискання, перш ніж обмін даними між клієнтом і сервером стане можливим. Оскільки ініціатором тристороннього рукостискання TCP завжди є клієнт, то він першим відправляє пакет із прапором SYN серверу.

Одержавши такий SYN-пакет, сервер відповідає, підтверджуючи одержання запиту й відправляючи при цьому свій власний запит SYN – пакет із установленими прапорами SYN і ACK. Клієнт, у свою чергу, одержавши пакет з підтвердженням і запитом від сервера, відправляє серверу пакет із установленим прапором ACK, який підтверджує, що обоє хости згодні створити з'єднання. Після такого «обміну рукостисканнями» з'єднання вважається встановленим, і дані можуть передаватися між хостами.

При проведенні TCP SYN Flood атаки зловмисники інтенсивно відправляють серверу велика кількість SYN-пакетів з підробленими IP-адресами. Це змушує сервер реагувати, відправляючи у відповідь на кожний такий неправильний запит пакет SYN-ACK, виділяючи частину ресурсів і залишаючи

свої порти «напіввідчиненими» чекаючи численних відповідей (пакетів із установленим прапором ACK) від хостів, яких насправді не існує, і пі дтверджень вони, відповідно, відправляти не будуть.

Структура DoS-атаки

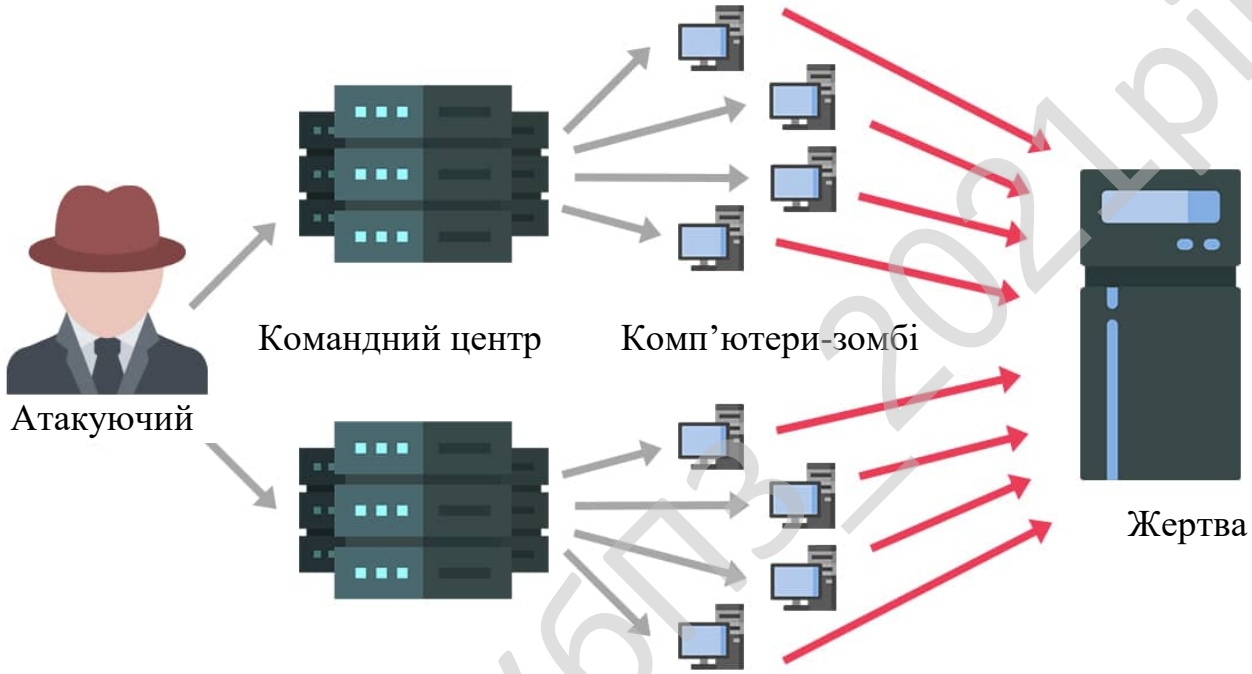

Рисунок 3.1 – Структурна схема системи

При проведенні більш простої версії атаки TCP SYN Flood (прямій атаки без підміни IP-адреси) зловмисники просто використовують налаштування брандмауера, щоб заблокувати одержання зворотних пакетів із установленими прапорами SYN і ACK від сервера жертви ще до того, як ці відправлені у відповідь запити досягнуть їх системи. Завалюючи свою мету SYN-пакетами й не відповідаючи на запити (не відправляючи у відповідь пакети ACK), що атакують можуть легко вичерпати ресурси цілі, при цьому не занадто перевантажуючи свої ресурси. Адже в цей час сервер жертви « щосили» намагається впорається з різко зрослим трафіком, що, як наслідок, серйозно збільшує використання ЦП і пам'яті. Зрештою, це може привести до вичерпання всіх його ресурсів (ЦП і ОЗП), і

сервер жертви більше не зможе обслуговувати будь-які клієнтські запити (у тому числі й від сумлінних користувачів), тобто не надавати їм доступ до системних ресурсів і сервісам.

Використання Kali Linux & hping3 для моделювання атаки TCP SYN Flood

Однак, для того, щоб провести аудита безпеки й перевірити, чи можете ви виявити цей тип DoS-атаки, насамперед ви повинні навчитися її самі проводити. Мабуть, найпростіший спосіб досягнення цієї цілі базується на використанні дистрибутива Kali Linux, а точніше – програми hping3, популярного TCP-інструмента тестування проникнення, включеного в набір Kali Linux.

Користувачі Linux у якості альтернативної можливості можуть установити інструментарій hping3 у свій існуючий дистрибутив Linux за допомогою наступної команди:

«# sudo apt-get install hping3»

Тому що в більшості випадків зловмисники будуть використовувати генератор пакетів hping або інший інструментарій з подібною функціональністю для підміни реальних IP-адрес випадковими, ми також оборотний фокус нашої уваги на цей момент. Наступний рядок дозволяє нам почати й направити атаку TCP SYN Flood на нашу мету (192.168.1.159):

«# hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source 192.168.1.159»

А тепер давайте детально розберемо наведену ледве вище команду. Ми відправляємо 15000 пакетів («-c 15000») розміром 120 байт («-d 120») кожний. Ми також указуємо, що прапор SYN («-S») повинен бути включений, а розмір TCP-Вікна має значення 64 («-w 64»). Щоб направити атаку на HTTP-веб-сервер нашої жертви, ми вказуємо порт 80 («-p 80») і використовуємо прапор («--flood») для максимально швидкого відправлення пакетів. Як ви, напевно, уже зрозуміли, прапор («--rand-source») використовується для генерування підроблених IP-адрес, щоб замаскувати реальне джерело й уникнути виявлення, а

також у те ж саме час не дати системі зловмисника одержувати відповідні пакети із установленими прапорами SYN і ACK від сервера жертви.

Використання розробленого в даній роботі програмного забезпечення системи кібербезпеки моделювання й визначення DoS атаки типу TCP SYN Flood для ідентифікації атаки TCP SYN Flood

Тепер, коли ми навчилися моделювати атаку зловмисників, ми можемо спробувати виявити її. Розроблене в даній роботі програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типу TCP SYN Flood може здатися досить складним інструментом. Однак він має рядом унікальних переваг, яких немає в жодного іншого інструментарію, який ви можете використовувати в якості альтернативи для застосунку цієї проблеми. Його функціональність підходить для застосунку величезної кількості завдань, він повністю безкоштовний, з відкритим вихідним кодом і доступний на багатьох платформах.

Для кращого ілюстрування нашого керівництва ми створили тестове середовище, у якому ми використовували ноутбук із установленим дистрибутивом Kali Linux для проведення атаки через мережний комутатор на стаціонарний комп'ютер під керуванням ОС Windows 10. Незважаючи на те, що подібна структура захищена набагато гірше, чим багато корпоративних мереж, для нашого тестового випробування це не має значення, тому що зловмисники можуть реалізувати подібні атаки після одержання несанкціонованого доступу й проникнення в мережу. Також як ви могли бачити з наведеної вище команди hping3, ми використовували випадкові IP-адреси, тому що саме цей метод атаки TCP SYN Flood будуть використовувати досвідчені зловмисники.

Здійснювану атаку TCP SYN Flood досить легко виявити, якщо ви знаєте, що шукаєте. Її початок адміністратори відразу ж повинні ідентифікувати по різко зрослому потоці TCP-трафіку. Як і слід було сподіватися, основною ознакою саме цієї атаки є величезну кількість пакетів із прапором SYN, одержуваних сервером, що атакуються (у нашій тестовій демонстрації – ПК із Windows 10).

При аналізі трафіку для їхньої ідентифікації нам будуть потрібні певні фільтри. Так, ми можемо відфільтрувати трафік по отриманих пакетах із установленим прапором SYN без наступного підтвердження (одержання пакетів із установленим прапором ACK), використовуючи наступний фільтр:

«TCP.flags.SYN == 1 and TCP.flags.ACK == 0»

Ми виявили велику кількість TCP-пакетів із установленим прапором SYN без наступного підтвердження на запит сервера, отриманих з дуже малою різницею в часі. Джерела кожного такого SYN-пакета відрізняються (усі пакети відправлені з різних IP-адрес), але всі вони мають ідентичний порт призначення 80 (HTTP), ідентичну довжину (120) і розмір TCP вікна (64). Якщо ми задамо фільтр «TCP.flags.SYN == 1 and TCP.flags.ACK == 1», те побачимо, що кількість отриманих пар пакетів від клієнтів (відразу із установленим прапором SYN, а потім – із прапором ACK) відносно невелике. Це вірна ознака атаки TCP SYN Flood на вашу систему.

Ми також можемо подивитися графіки розроблене в даній роботі програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типу TCP SYN Flood для візуальної вистави про ріст трафіку. Графік отриманих/відправлених пакетів можна одержати за допомогою вибору меню «Statistics —> I/O Graph». Для нашого тестового прикладу цей графік демонструє величезний сплеск кількості пакетів від 0 до приблизно 2400 пакетів у секунду.

Вилучивши наш фільтр і відкривши статистичні дані ієрархії протоколів («Protocol hierarchy statistics»), ми також можемо переконатися, що нами був зафіксований незвичайно великий обсяг TCP-пакетів.

Усі наведені нами в рамках даної роботи спостереження за різкою зміною показників з великою часткою ймовірності вказують саме на виявлення атаки TCP SYN Flood, залишаючи дуже мізерні шанси для іншої інтерпретації. Таким чином, використовуючи розроблене в даній роботі програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типу TCP SYN Flood, ми можемо переконатися в здійсненні протиправних дій з використанням

тристороннього рукостискання за протоколом TCP з боку зловмисників проти наших мережних ресурсів, і вчасно вжити заходів для виправлення ситуації.

У рамках даної роботи ми показали, як у тестових цілях змоделювати атаку TCP SYN Flood за допомогою дистрибутива Kali Linux (передвстановленого інструментарію hping3), а також як виявити цю DoS-атаку, використовуючи фільтри й графіки аналізатора мережних протоколів розроблене в даній роботі програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типу TCP SYN Flood, і вчасно прийняти захід для її нейтралізації.

3.3 Розробка функціональної схеми

Програмне забезпечення системи кібербезпеки моделювання й визначення DoS – це потужний мережний аналізатор, який може використовуватися для аналізу трафіку, що проходить через мережний інтерфейс комп'ютера. Він може знадобитися для виявлення й вирішення проблем з мережею, налагодження веб-застосунків, мережних програм або сайтів.

Програмне забезпечення системи кібербезпеки моделювання й визначення DoS дозволяє повністю переглядати вміст пакета на всіх рівнях: так ви зможете краще зрозуміти як працює мережа на низькому рівні.

Програмне забезпечення системи кібербезпеки моделювання й визначення DoS працює з переважною більшістю відомих протоколів, має зрозумілий і логічний графічний інтерфейс на основі GTK+ і наймогутнішу систему фільтрів.

Програмне забезпечення системи кібербезпеки моделювання й визначення DoS є кроссплатформенним, працює на Linux, Solaris, Freebsd, Netbsd, Openbsd, MAC OS, і Windows.

Основні можливості програмного забезпечення системи кібербезпеки моделювання й визначення DoS

Список основних можливостей програми:

– Захват пакетів у реальному часі із провідного або будь-якого іншого типу мережних інтерфейсів, а також читання з файлу.

– Підтримуються такі інтерфейси захвата: Ethernet, IEEE 802.11, PPP і локальні віртуальні інтерфейси.

– Пакети можна відсівати по множині параметрів за допомогою фільтрів.

– Усі відомі протоколи підсвічуються в списку різними кольорами, наприклад TCP, HTTP, FTP, DNS, ICMP і так далі.

– Підтримка захвата трафіку VoIP-дзвінків.

– Підтримується розшифрування HTTPs-трафіку при наявності сертифіката.

– Розшифрування WEP-, WPA-трафіку бездротових мереж при наявності ключа й handshake.

– Відображення статистики навантаження на мережу.

– Перегляд умісту пакетів для всіх мережних рівнів.

– Відображення часу відправлення й одержання пакетів.

Аналіз мережного трафіку

На головному екрані відображається список доступних для аналізу мережних інтерфейсів. Виберемо інтерфейс “Бездротова мережа” і відкриється вікно:

Це вікно теж розділене на кілька частин:

– Верхня частина – це меню й панелі з різними кнопками.

– Список пакетів – далі відображається потік мережних пакетів, які ви будете аналізувати.

– Уміст пакета – трохи нижче розташований уміст обраного пакета, воно розбите по категоріях залежно від транспортного рівня.

– Реальне представлення – аж унизу відображається вміст пакета в реальному виді, а також у вигляді HEX.

Рисунок 3.2 – Функціональна схема системи

Можна кликнути по будь-якому пакету, щоб проаналізувати його вміст.

Фільтри програмного забезпечення системи кібербезпеки моделювання й визначення DoS

Для введення фільтрів під меню є спеціальний рядок. Розглянемо основні фільтри:

- IP.dst – цільова IP-адреса;
- IP.src – IP-адресу відправника;
- IP.addr – IP відправника або одержувача;
- IP.proto – протокол;
- TCP.dstport – порт призначення;

- TCP.srcport – порт відправника;
- IP.ttl – фільтр по ttl, визначає мережна відстань;
- HTTP.request_uri – запитувана адресу сайту.

Для вказівки відносини між полем і значенням у фільтрі можна використовувати такі оператори:

- == – рівно;
- != – не рівно;
- < – менше;
- > – більше;
- <= – менше або рівно;
- >= – більше або рівно;
- matches – регулярне вираження;
- contains – містить.
- Для об'єднання декількох виражень можна застосовувати:
- && – обоє вираження повинні бути вірними для пакета;
- || – може бути вірним одне з виражень.

Програмне забезпечення системи кібербезпеки моделювання й визначення DoS майже завжди позиціонується як мережний інструмент для аналізу, правда полягає в тому, що він може аналізувати й інші пристрої, такі як USB-трафік і навіть Unix-сокети між застосунками.

Використовуючи програмне забезпечення системи кібербезпеки моделювання й визначення DoS і володіючи необхідними знаннями можна досить ефективно знаходити й діагностувати різноманітні проблеми, що виникають у мережі.

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.

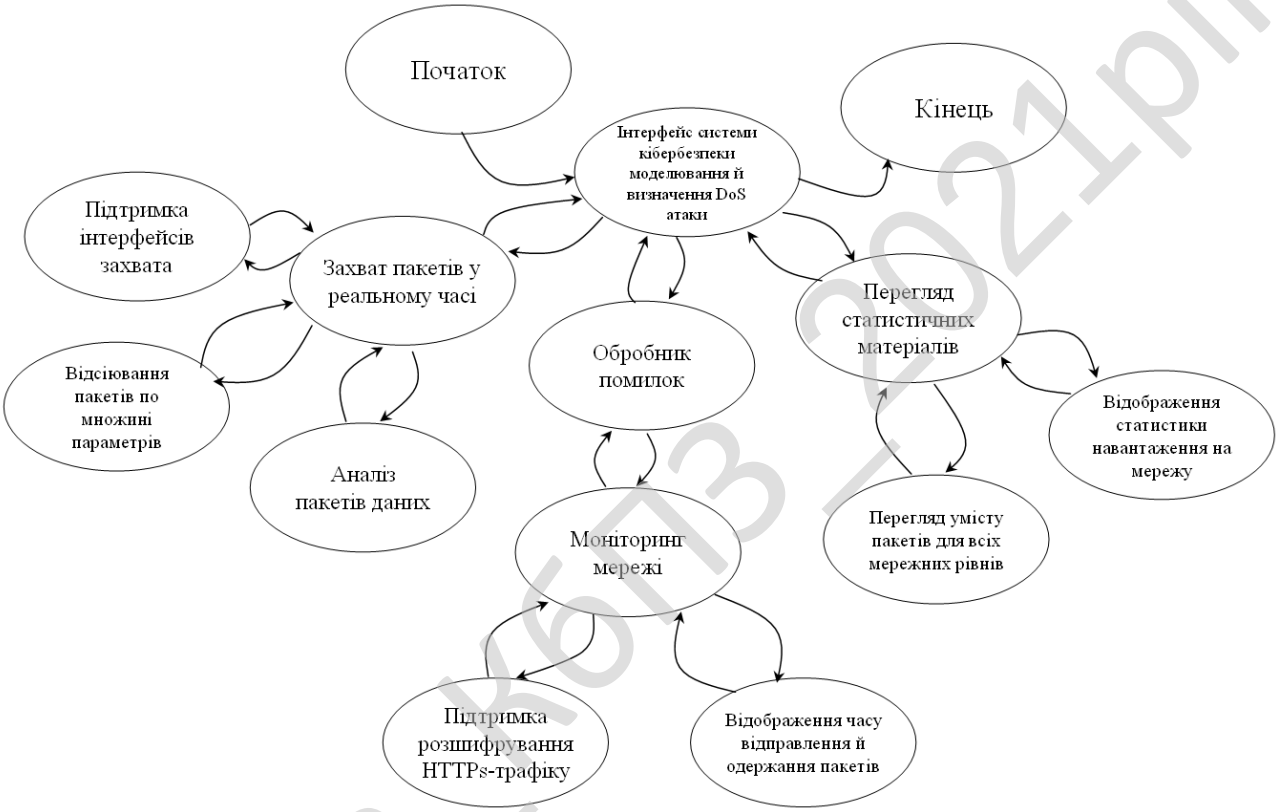

Рисунок 3.3 – Діаграма взаємодії процесів

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

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, поті сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

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

– Потоки даних гібридні між елементами трьох попередніх типів.

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

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо послідовність дій та викликів підпрограм в загальному алгоритмі роботи основної програми що зображено на рисунку 4.1. у вигляді блок-схеми:

– Виведення вікна кібербезпеки моделювання й визначення DoS атаки;

– Тест доступу до ГМ;

– Захват пакетів у реальному часі, формування пакету даних;

– Виведення отриманої інформації;

– Виведення статистики навантаження на мережу;

– Встановлення параметрів ПЗ;

– Налаштування пріоритетів різним типам трафіку (TCP, HTTP, FTP, DNS, ICMP і так далі);

– Встановлення смуги пропускання для різних класів трафіку;

– Моделювання завантаженістю мережі;

– Встановлення параметрів обслуговування черг;

– Вибір та запуск механізму обслуговування;

– Запит моніторинг мережі;

– Відсіювання пакетів по множині параметрів за допомогою фільтрів;

– Розшифрування частини трафіку;

– Підпрограма обробки пакету даних;

– Виведення даних;

– Запит WM_CLOSE – запит завершення ПЗ;

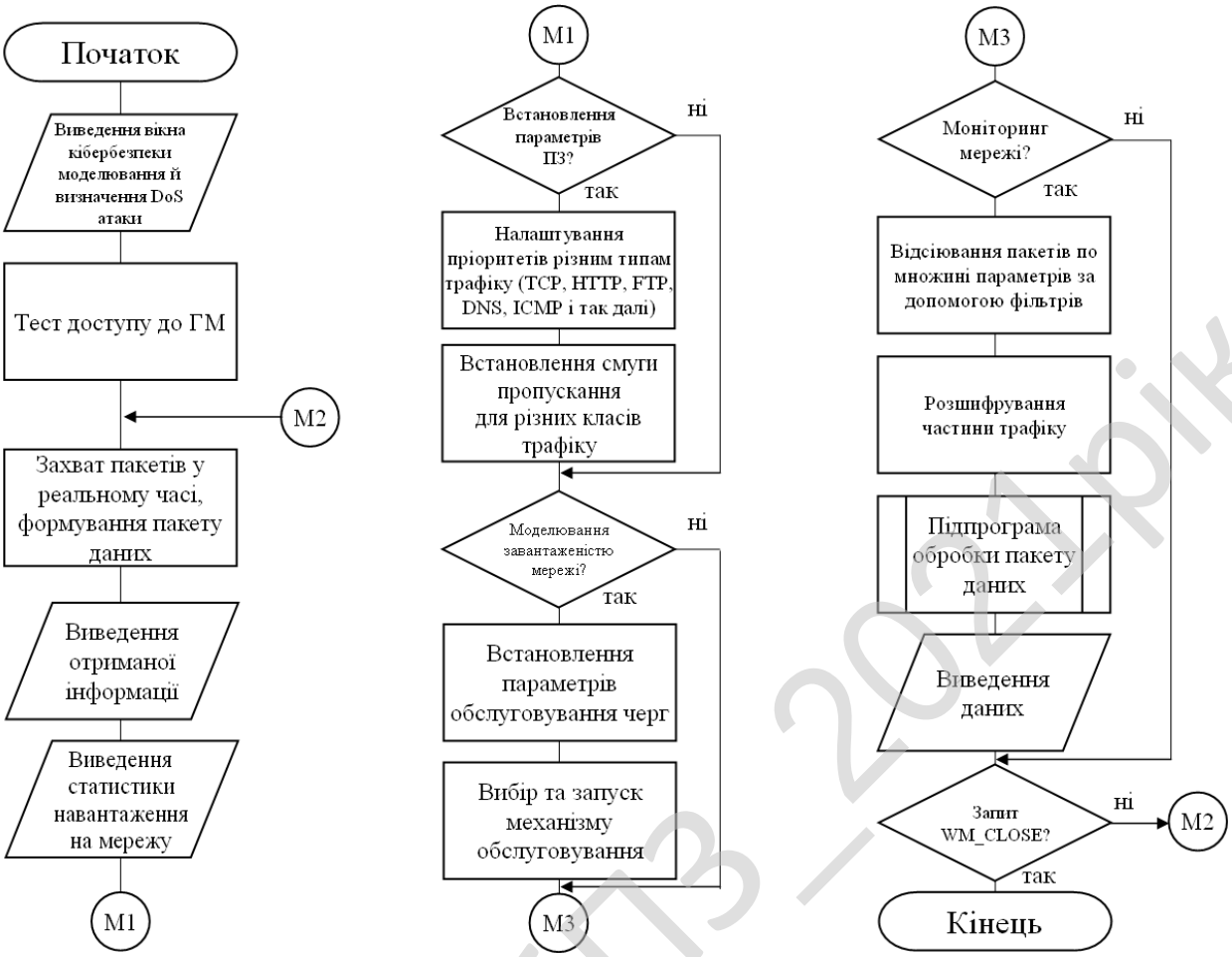

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображено роботу підпрограми з реалізацією наступних дій:

– Обчислення кількості відкинутих пакетів;

– Обчислення ймовірності відкидання пакетів;

– Обчислення значень середньозважених черг;

– Обчислення середнього часу доставки пакетів;

– Обчислення варіації часу доставки пакетів;

– Порівняння отриманих значень з допустимими;

– Виведення часу відправлення й одержання пакетів;

– Виведення списку обробленого потоку даних;

– Виведення результатів обробки;

– Збереження звіту обробки пакетів даних;

– Збереження звіту обробки пакетів у файлі (*.dt7);

– Запис до журналу роботи ПЗ.

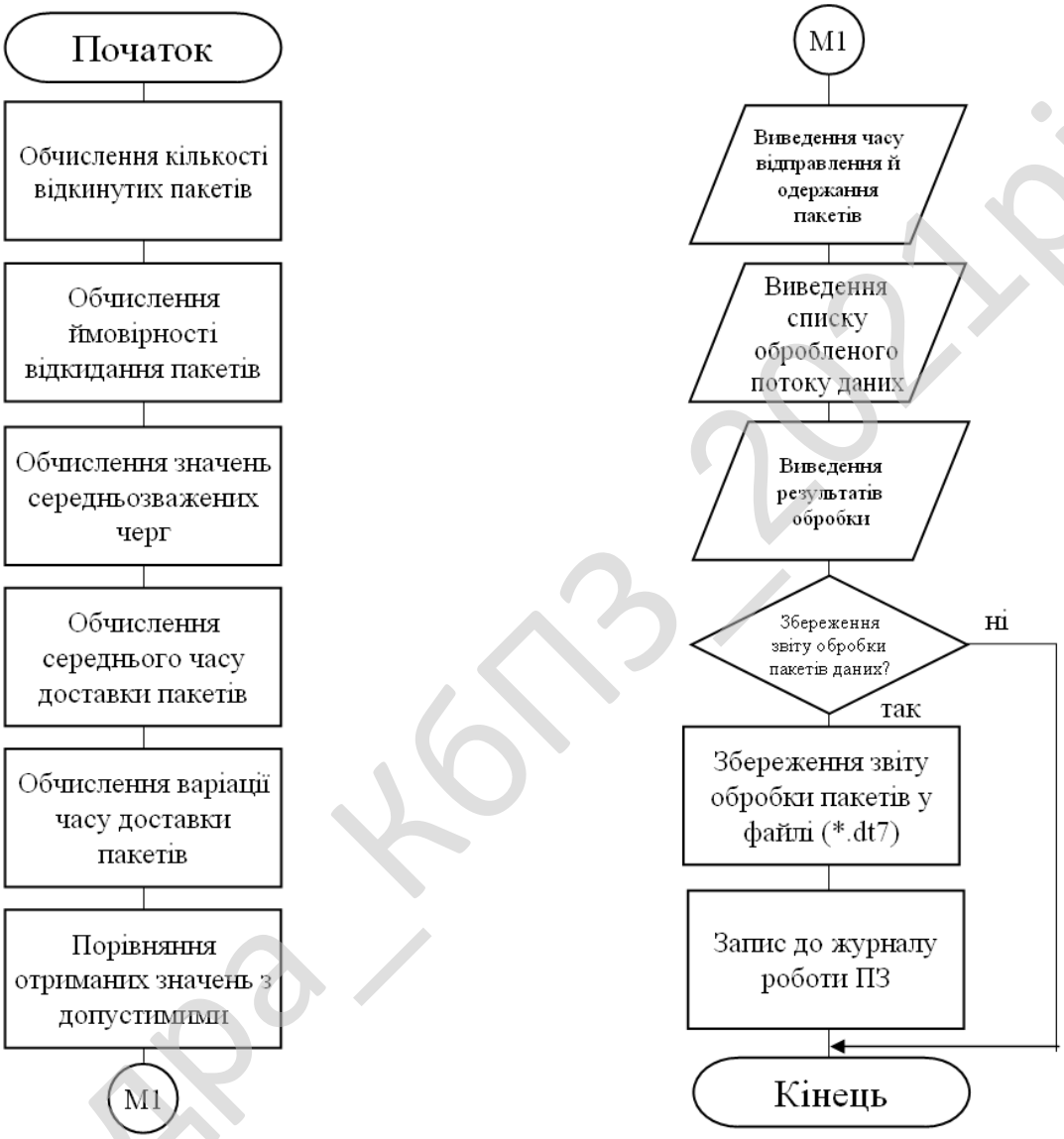

Рисунок 4.2 – Блок-схема роботи підпрограми

При розробці ПЗ було використано V-Model (або VEE модель) є моделлю розробки інформаційних систем (ІС), спрямованої на спрощення розуміння складнощів, пов'язаних з розробкою систем. Вона використовується для

визначення єдиної процедури розробки програмного забезпечення, апаратного забезпечення та людино-машинного інтерфейсу.

Концепція V-подібної моделі була розроблена Німеччиною та США в кінці 1980-х років незалежно один від одного:

– Німецька V-модель була розроблена аерокосмічної компанією IABG в Оттобрунні поряд з Мюнхеном у сприянні з Федеральним департаментом з закупівлі озброєнь в Кобленці, для Міністерства оборони Німеччини. Модель була прийнята німецькою федеральною адміністрацією для цивільних потреб влітку 1992.

– Американська V-Model (VEE) була розроблена національною радою з системної інженерії (міжнародна – з 1995 року) для супутникових систем, включаючи обладнання, програмне забезпечення та взаємодію з користувачами.

Сучасною версією V-Model є V-Model XT, яка була затверджена в лютому 2005 року. V-модель використовується для управління процесом розробки програмного забезпечення для німецької федеральної адміністрації.

Зараз вона є стандартом для німецьких урядових і оборонних проектів, а також для виробників ПЗ в Німеччині. V-Model являє собою скоріше набір стандартів у галузі проектів, що стосуються розробки нових продуктів. Ця модель багато в чому схожа з Prince2 і описує методи як для проектного управління, так і для системного розвитку.

Основні принципи

Основний принцип V-подібної моделі полягає в тому, що деталізація проекту зростає при русі зліва направо, одночасно з плином часу, і ні те, ні інше не може повернути назад. Ітерації в проекті виробляються по горизонталі, між лівою і правою сторонами літери.

Стосовно до розробки інформаційних систем V-Model – варіація каскадної моделі, в якій завдання розробки йдуть зверху вниз по лівій стороні букви V, а завдання тестування – вгору по правій стороні букви V. Усередині V проводяться

горизонтальні лінії, що показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній із фаз тестування.

Модель базується на тому, що приймально-здавальні випробування ґрунтуються, насамперед, на вимогах, системне тестування – на вимогах та архітектури, комплексне тестування – на вимогах, архітектурі та інтерфейсах, а компонентне тестування – на вимогах, архітектурі, інтерфейсах та алгоритмах

Цілі

V-модель забезпечує підтримку у плануванні та реалізації проекту. В ході проекту ставляться такі завдання:

1. Мінімізація ризиків: V-подібна модель робить проект більш прозорим і підвищує якість контролю проекту шляхом стандартизації проміжних цілей і опису відповідних їм результатів та відповідальних осіб. Це дозволяє виявляти відхилення в проекті і ризики на ранніх стадіях і покращує якість управління проектом.

2. Підвищення та гарантії якості: V-Model – стандартизована модель розробки, що дозволяє домогтися від проекту результатів бажаної якості. Проміжні результати можуть бути перевірені на ранніх стадіях. Універсальне документування полегшує читаність, зрозумілість та контрольованість.

3. Зменшення загальної вартості проекту: Ресурси на розробку, виробництво, управління і підтримку можуть бути заздалегідь прораховані та проконтрольовані. Отримувані результати також універсальні і легко прогнозуються. Це зменшує витрати на подальші стадії та проекти.

4. Підвищення якості комунікації між учасниками проекту: Універсальний опис усіх елементів та умов полегшує взаєморозуміння всіх учасників проекту. Таким чином, зменшуються неточності у розумінні між користувачем, покупцем, постачальником і розробником.

Переваги:

– Користувачі V-Model беруть участь у розробці та підтримці V-моделі. Комітет з контролю за змінами підтримує проект і збирається раз на рік для обробки всіх отриманих запитів на внесення змін до V-Model.

– На старті будь-якого проекту V-подібна модель може бути адаптована під цей проект, так як ця модель не залежить від типів організацій та проектів.

– V-model дозволяє розбити діяльність на окремі кроки, кожен з яких буде включати в себе необхідні для нього дії, інструкції до них, рекомендації та докладне пояснення діяльності.

Обмеження. Наступні моменти не враховуються в V-моделі, але можуть бути розглянуті окремо, або можливо адаптувати модель під них:

– Не регулюється розміщення контрактів на обслуговування.

– Організація і виконання управління, обслуговування, ремонту та утилізації системи не враховуються в V-моделі. Однак, планування і підготовка до цих операцій моделлю розглядаються.

– V-подібна модель більше стосується розробки програмного забезпечення в проекті, ніж всієї організації процесу.

– У моделі особливе значення надається плануванню, спрямованому на верифікацію та атестацію розроблювального продукту на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації.

– У моделі передбачені атестація та верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту.

– У V-подібної моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПО – перед розробкою компонентів.

– Модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню.

– Завдяки моделі менеджери проекту можуть відслідковувати хід процесу розробки, так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою.

Недоліки:

– Модель не передбачає роботу з паралельними подіями.

– У моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу.

– Тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести змін, не вплинувши при цьому на графік виконання проекту.

– У модель не входять дії, спрямовані на аналіз ризиків.

– Деякий результат можна отримати тільки при досягненні низу букви V.

Розглянемо формат що використовується, а саме JSON (JavaScript Object Notation, укр. запис об'єктів JavaScript, вимовляється джейсон) – це текстовий формат обміну даними між комп'ютерами.

JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією). Розробив і популяризував формат Дуглас Крокфорд.

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

JSON з'явився через необхідність обміну даними з сервером у реальному часі без використання плагінів для браузерів, flash-додатків або Java-аплетів, які використовувались скрізь на початку 2000-х років. Дуглас Крокфорд був тим, хто активно просував новий на той час формат. Він з колегами хотів створити технологію, яка використовувала б можливості звичайного браузера давала б веб-розробникам можливість створювати веб-застосунки із постійним двостороннім зв'язком із веб-сервером.

JSON вперше був використаний в проекті в Communities.com для Cartoon Network, він дозволяв обмінюватись повідомленнями і одночасно маніпулювати DHTML-елементами.

Веб-сайт JSON.org було запущено 2002 року. У грудні 2005-го року Yahoo! почав переводити деякі зі своїх веб-сервісів на роботу з JSON. Google взялася до роботи з технологією у своєму веб-протоколі GData у грудні 2006-го року.

Використання

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Якщо говорити про веб-застосунки, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси).

Формат JSON так само добре підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші.

Синтаксис

JSON будується на двох структурах:

1. Набір пар назва/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом.

2. Впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі їхні форми:

1. Об'єкт – це послідовність пар назва/значення. Об'єкт починається з символу { і закінчується символом } . Кожне значення слідує за : і пари назва/значення відділяються комами.

2. Масив – це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.

3. Значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.

4. Рядок – це послідовність з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються зі зворотної косої риски \. Символи представляються простим рядком. Тип Рядок (String) дуже схожий на String в мовах C і Java. Число теж дуже схоже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

Використання JSON в AJAX

Наступний фрагмент коду JavaScript показує, як клієнт може використати XMLHttpRequest для запиту об'єктів в форматі JSON з серверу. (Серверна частина коду пропущена, вона просто повертає на запит URL рядок у JSON форматі).

Треба відзначити, що тут використання XMLHttpRequest не є кросс-браузерним (за деталями звертайтеся на сторінку XMLHttpRequest), і код зазнаватиме незначних модифікацій в різних версіях оглядачів Internet Explorer, Opera, Safari або Mozilla. Використання запиту XMLHttpRequest обмежено

правилом одного джерела (same origin policy): URL, що відповідає на запит, має посилатися на той же сайт, що обслуговує сторінку, що ініціювала запит.

Оглядачі можуть також використовувати тег <iframe> для асинхронного запиту JSON-даних в кросс-браузерному варіанті, або використати просте перенаправлення <form action="url_to_cgi_script" target="name_of_hidden_iframe">. Такий підхід був поширений до приходу популярного нині запиту XMLHttpRequest.

Динамічний тег <script> також можна використати для підвантаження JSON-даних. Ця техніка можлива, щоб обійти надто суворе правило одного джерела, але вона не є безпечною. Запит JSONRequest пропонується, як безпечніша альтернатива.

Питання безпеки

Хоча JSON призначений для передачі даних в серіалізованому вигляді, його синтаксис відповідає синтаксису JavaScript і це створює низку проблем безпеки. Часто для обробки даних, отриманих від зовнішнього джерела у форматі JSON, до них застосовується функція eval() без якої-небудь попередньої перевірки.

JavaScript eval()

Оскільки JSON представляється синтаксично правильним фрагментом коду JavaScript, природним способом розбору JSON-даних в JavaScript-програмі є використання вбудованої в JavaScript функції eval(), яка призначена для обчислення JavaScript-виразів. При цьому підході відпадає необхідність у використанні додаткових парсерів.

Техніка використання eval() робить систему вразливою, якщо джерело JSON-даних, що використовуються, не відноситься до надійних. Такими даними може виступати шкідливий JavaScript-код для атак за допомогою ін'єкції коду. За допомогою цієї вразливості можливо здійснювати крадіжку даних, підробку автентифікації. Проте, вразливість можна усунути за рахунок використання додаткових засобів перевірки даних на коректність. Наприклад, до виконання

eval() отримані від зовнішнього джерела дані можуть перевірятися за допомогою регулярних виразів.

Вбудований JSON

Останні версії веб-браузерів мають вбудовану підтримку JSON і здатні його обробляти без виклику функції eval(), що призводить до описаної проблеми. Обробка JSON у такому разі зазвичай здійснюється швидше. Так у червні 2009 року вбудовану підтримку JSON мали такі браузери: Mozilla Firefox 3.5+; SeaMonkey 2; Thunderbird 3; Microsoft Internet Explorer 8; Opera 10.5; Браузери, засновані на WebKit (наприклад, Google Chrome, Apple Safari).

Принаймні дві популярні бібліотеки JavaScript використовують вбудований JSON у разі його доступності: jQuery; Dojo.

Підробка крос-доменного запиту

Непродумане використання JSON робить сайти вразливими до підробки міжсайтових запитів (CSRF або XSRF). Оскільки тег <script> допускає використання джерела, що не належить до того ж домену, що і використовуваний ресурс, це дозволяє виконувати код даних, представлених у форматі JSON, в контексті довільної сторінки, що робить можливою компрометацію паролів або іншої конфіденційної інформації користувачів, що пройшли авторизацію на іншому сайті.

Це є проблемою тільки у разі вмісту в JSON-даних конфіденційної інформації, яка може бути компрометована третьою стороною і якщо сервер розраховує на політику одного джерела, блокуючи доступ до даних при виявленні зовнішнього запиту. Це не є проблемою, якщо сервер визначає допустимість запиту, надаючи дані тільки у разі його коректності. HTTP cookie не можна використовувати для визначення цього. Виключне використання HTTP cookie використовується підробкою міжсайтових запитів.

Довідатися трафік можна не використовуючи проксі-сервер. Для цього досить використовувати всього лише одну функцію бібліотеки iphlpapi.dll, що поставляється з усіма версіями Windows.

Оголошення функції (всі версії Windows):

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

Параметри:

– pIfTable – повинен містити покажчик на структуру;

– pdwSize – повинен містити розмір структури;

– bOrder – указує, чи потрібна сортування в масиві, що повертається.

Як перший параметр функція використовує покажчик на структуру, от сам опис структури

```
type
 TDSIfTable = packed record
   dwNumEntries : DWORD;
   Table        : TDSIfArray;
 end;
 PDSIfTable = ^ TDSIfTable;
```

Поля:

– dwNumEntries – визначає розмірність масиву представленого другим параметром;

– Table – є масивом структур.

Структура сама по собі вкрай неінформативна, нас цікавить друге її поле, що також представляє собою структуру, даю її опис:

```
type
  TDSIfRow = 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;
            dwOutOoctets        : DWORD;
            dwOutUCastPkts      : DWORD;
            dwOutNUCastPkts     : DWORD;
            dwOutDiscards       : DWORD;
            dwOutErrors         : DWORD;
            dwOutQLen           : DWORD;
            dwDescrLen          : DWORD;
            bDescr              : array[0..255] of Char;
          end;
          TDSIfArray = array [0..512] of TDSIfRow;
          PDSIfRow = ^TDSIfRow;
          PDSIfArray = ^TDSIfArray;
        (*
        Поля:
        wszName – Покажчик на рядок утримуючий ім’я інтерфейсу
        dwIndex – Визначає індекс інтерфейсу
        dwType – Визначає тип інтерфейсу
        dwMtu – Визначає максимальну швидкість передачі
        dwSpeed – Визначає поточну швидкість передачі в бітах у секунду
        dwPhysAddrLen – Визначає довжину адреси втримується в bPhysAddr
        bPhysAddr – Містить фізичну адресу інтерфейсу (якщо простіше то його,
ненабагато видозмінену, MAC адресу)
        dwAdminStatus – Визначає активність інтерфейсу
        dwOperStatus – Містить поточний статус інтерфейсу
        dwLastChange – Містить останній змінений статус
        dwInOctets – Містить кількість байт прийнятих через інтерфейс
        dwInUcastPkts – Містить кількість спрямованих пакетів прийнятих інтерфейсом
        dwInNUCastPkts – Містить кількість ненаправлених пакетів прийнятих
інтерфейсом (включаючи броадкаст і т.п.)
        dwInDiscards – Містить кількість забракованих вхідних пакетів (навіть якщо
вони не містили помилки)
        dwInErrors – Містить кількість вхідних пакетів утримуючі помилки
        dwInUnknownProtos – Містить кількість забракованих вхідних пакетів зі
структурою невідомого протоколу
        dwOutOctets – Містить кількість байт відправлених інтерфейсом
        dwOutUCastPkts – Містить кількість спрямованих пакетів відправлених
інтерфейсом
```

```
        dwOutNUCastPkts- Містить кількість ненаправлених пакетів відправлених
інтерфейсом (включаючи броадкаст і т.п.)
        dwOutDiscards- Містить кількість забракованих вихідних пакетів (навіть якщо
вони не містили помилки)
        dwOutErrors- Містить кількість вихідних пакетів утримуючі помилки
        dwOutQLen - Містить довжину черги даних
        dwDescrLen - Містить розмір масиву bDescr
        bDescr - Містить опис інтерфейсу
        *)
```

В цій структурі втримується багато інформації, що ми й будемо використовувати. Інтерфейсом є не обов'язково якийсь фізичний пристрій (наприклад, мережна карта).

Отже, додаємо на форму ListView, назвемо його lvTraffic, створемо у ній чотири колонки з наступними іменами (Caption) – bDescr, bPhysAddr (MAC), dwInOctets і dwOutOctets.

У них ми будемо виводити найменування інтерфейсу, його MAC адресу, загальну кількість прийнятих і відправлених байт. Додайте опису структур і функції в інтерфейсну частину модуля.

TDSIfRow повинна бути оголошена першою. Тепер додаємо на форму таймер, він буде відповідати за щосекундне відновлення інформації про трафік, назвемо його tmrTraffic. От сам код визначення поточного вхідного/вихідного трафіка:

```
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         : TDSIfTable;
  i             : Integer;
  Size          : Integer;
begin
  tmrTraffic.Enabled := False;
//Припиняємо про всякий випадок 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
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets));
//Усього відправлено байт
      end;
    end;
  lvTraffic.Items.EndUpdate;
```

```
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := True;
//Не забуваємо активувати таймер
end;
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-2, який шифрує дані 256-бітними блоками з використанням 512-бітного ключа. Допускається використання ключів менших розмірів (не менш 128 біт), які доповнюються бітовими нулями до 512 біт.

Шифруємий блок даних ділиться на 8 фрагментів по 32 біта (які позначені буквами *A*...*H*). Алгоритм виконує 64 раунду перетворень, у кожному з яких дані фрагменти обробляються в такий спосіб:

$$T = H_i + S_1(E_i) + Ch(E_i, F_i, G_i) + M_i + K_i,$$

$$H_{i+1} = G_i,$$

$$G_{i+1} = F_i,$$

$$F_{i+1} = E_i,$$

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

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

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

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

$$A_{i+1} = T + S_0(A_i) + Maj(A_i, B_i, C_i).$$

де T – часова змінна.

Використовувані функції визначені в такий спосіб:

$$S_0(x) = (x >>> 2) \oplus (x >>> 13) \oplus (x >>> 22),$$

$$S_1(x) = (x >>> 6) \oplus (x >>> 11) \oplus (x >>> 25),$$

$$Ch(x, y, z) = (x \,\&\, y) \oplus (x' \,\&\, z),$$

$$Maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z),$$

де >>> – операція побітового циклічного зрушення вправо.

Константи, що модифікують, M_i (i = 0…63) наведено нижче (одна за одною від M_0 до M_{63}):

428A2F98	71374491	B5C0FBCF	E9B5DBA5
3956C25B	59F111F1	923F82A4	AB1C5ED5
D807AA98	12835B01	243185BE	550C7DC3
72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC
2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7
C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13
650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3
D192E819	D6990624	F40E3585	106AA070
19A4C116	1E376C08	2748774C	34B0BCB5
391C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
748F82EE	78A5636F	84C87814	8CC70208
90BEFFFA	A4506CEB	BEF9A3F7	C67178F2

Фрагменти розширеного ключа $K_0…K_{63}$ обчислюються в процесі процедури розширення ключа, що виконується в такий спосіб:

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

Етап 2. Інші фрагменти розширеного ключа $K_{16}…K_{63}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = O_1(K_{i-2}) + K_{i-7} + O_0(K_{i-15}) + K_{i-16},$$

де функції O_0 і O_1 визначені так:

$$O_0(x) = (x >>> 7) \oplus (x >>> 18) \oplus (x >> 3),$$

$$O_1(x) = (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10),$$

де >> – операція побітового зрушення (не циклічного) вправо.

Раунди розшифрування алгоритму виконуються у зворотній послідовності:

$$T = A_{i+1} + S_0'(B_{i+1}) + Maj'(B_{i+1}, C_{i+1}, D_{i+1}) + 2,$$

$$H_i = T + S_1'(F_{i+1}) + Ch'(F_{i+1}, G_{i+1}, H_{i+1}) + M_i' + K_i' + 4,$$

$$G_i = H_{i+1},$$

$$F_i = G_{i+1},$$

$$E_i = F_{i+1},$$

$$D_i = E_{i+1} + T' + 1,$$

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

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

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

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: Додаткові параметри; Оновити.

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

– Верхнього меню: Файл; Дії; Параметри; Довідка.

– Розділу обрання групи мережевих даних.

– Розділу виведення результату роботи системи.

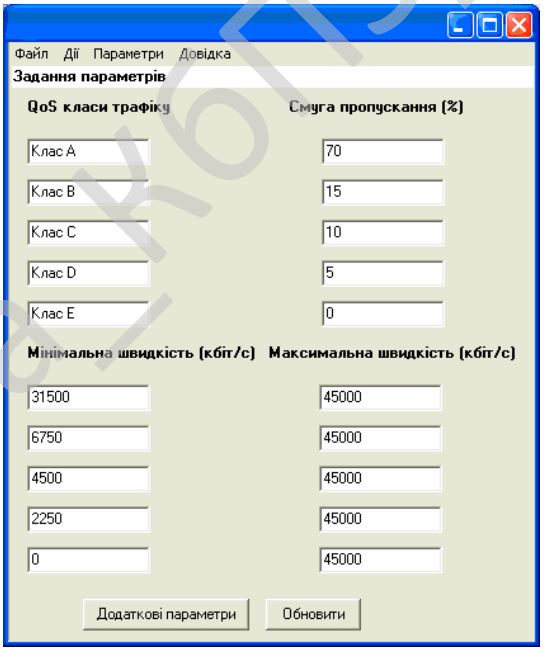

Рисунок 5.1 – Вікно введення параметрів ПЗ

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.

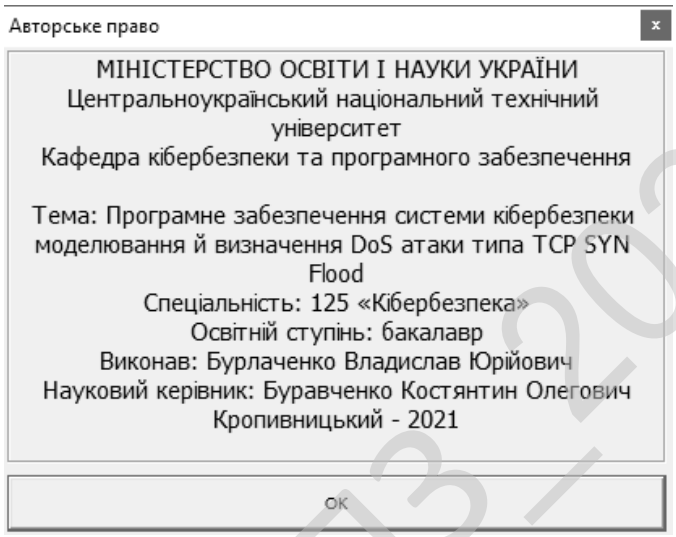

Рисунок 5.2 – Авторське право

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 моделювання й визначення DoS атаки типа TCP SYN Flood.

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 моделювання й визначення DoS атаки типа TCP SYN Flood.

– Досліджена система кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання моделювання й визначення DoS атаки типа TCP SYN Flood.

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 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood. Це дозволило мінімізувати строк розробки програмного

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-2.

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. Qrator Labs отмечает массовое распространение инструментов для DDoS. //БИТ [Електронний ресурс]..

2. *J. Yuan, K. Mills*. Monitoring the Macroscopic Effect of DDoS Flooding AttACKs. IEEE Transactions on dependable and secure computing. , Senior Member, IEEE. M., 2005 – 12 p.

3. 3. *Douligeris, A. Mitrokotsa*. DDoS AttACKs and defense mechanisms: a classification. P., 2003 – 24 p.

4. *N. Quinn.* In Flawed, Epic Anonymous Book, the Abyss Gazes BACK.N.Y. *Wired*.2012 – 3 p.

5. 6. Perlroth. КонецформиGoDaddy Goes Down, and a HACKer Takes Credit. // Bits. 2012.

6. M. Prince. The DDoS That Knocked Spamhaus Offline (And How We Mitigated It). //CloudFlare blog.

7. M. J. Hashmi, M. Saxena, Dr. R. Saini. Classification of DDoS AttACKs and their Defense Techniques using Intrusion Prevention System. Research Scholar, Singhania University, Pacheri Bari, Jhujhunu, R. 2013. – 8 p.

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

9. 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.

10. 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

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

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

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

27. 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

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

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

30. Ершов В.А.. Мультисервисные телекоммуникационные сети /

В.А. Ершов, Н.А. Кузнецов. – М.: МГТУ им. Н.Э. Баумана, 2003. – 432 с.

31. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

32. 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

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

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

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

35. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

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

37. 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

38. 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/

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

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

41. 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.

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

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

44. 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

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

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

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

48. Наиболее опасная страна. Попытка заражения ПК. [Электронный ресурс]. – Режим доступа к ресурсу:: http://sooweb.ru/news/naibolee_opasnaja_strana_popytka_zarazhenija_pk/ 2012-01-23-526

49. НД ТЗІ Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csk.kfc.in.ua/documents/nakaz-web.doc

50. НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в

комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Электронный ресурс]. – Режим доступа к ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

51. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2012. – 943 с.

52. Поповский В.В. Защита информации в телекоммуникационных системах / В.В. Поповский, А.В. Персиков. – Х.: ООО "Компания СМИТ", 2006. Т2 – 292 с.

53. Постанова Кабінета Міністрів України від 29.03.2006 №373 «Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon2.rada.gov.ua/laws/show/373-2006-п.

54. Розробка методів підвищення оперативності передачі та захисту інформації у телекомунікаційних системах: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0113U003086

55. Розробка стеганографічних засобів вбудовування інформації в нерухливі та рухливі зображення: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U002599

56. Розробка методів підвищення безпеки телекомунікаційних мереж: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U006630

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood.

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 моделювання й визначення DoS атаки типа TCP SYN Flood;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.1.

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 – 75 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 2.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Буравченко К.О.

Програмне забезпечення системи кібербезпеки моделювання й визначення DoS атаки типа TCP SYN Flood

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 35

Літера: РП

Кропивницький – 2021 року

Основна програма

unit MainFormUnit – Запуск основної форми програми;

```
interface

uses
  Windows, Graphics, ExtCtrls, Controls, StdCtrls, Buttons, Tabs,
  ComCtrls, Classes, SysUtils, Forms, dialogs,
  TrafficUnit, IPHelper, IPHLPAPI, ShellAPI;

type
  TMainForm = class(TForm)
    pnlMain: TPanel;
    pnlBottom: TPanel;
    pc: TPageControl;
    tsAbout: TTabSheet;
    tsTraffic: TTabSheet;
    ExitButton: TButton;
    TrafficTabs: TTabSet;
    GroupBox: TGroupBox;
    ledAdapterDescription: TLabeledEdit;
    UnFreezeButton: TBitBtn;
    FreezeButton: TBitBtn;
    ClearCountersButton: TBitBtn;
    ledMACAddress: TLabeledEdit;
    gbIN: TGroupBox;
    ledOctInSec: TLabeledEdit;
    ledAvgINSec: TLabeledEdit;
    ledPeakINSec: TLabeledEdit;
    ledTotalIN: TLabeledEdit;
    gbOUT: TGroupBox;
    ledOctOUTSec: TLabeledEdit;
    ledAvgOUTSec: TLabeledEdit;
    ledPeakOUTSec: TLabeledEdit;
    ledTotalOUT: TLabeledEdit;
    Timer: TTimer;
    gbTime: TGroupBox;
    ledStartedAt: TLabeledEdit;
    ledActiveFor: TLabeledEdit;
    RemoveInactiveButton: TBitBtn;
    StatusText: TStaticText;
    cbOnTop: TCheckBox;
    Panel3: TPanel;
    ProductName: TLabel;
    lblURL: TLabel;
    Label3: TLabel;
    ProgramIcon: TImage;
    StaticText1: TStaticText;
    ledSpeed: TLabeledEdit;
    procedure TimerTimer(Sender: TObject);
    procedure ClearCountersButtonClick(Sender: TObject);
    procedure cbOnTopClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure TrafficTabsChange(Sender: TObject; NewTab: Integer;
      var AllowChange: Boolean);
    procedure ExitButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FreezeButtonClick(Sender: TObject);
    procedure UnFreezeButtonClick(Sender: TObject);
    procedure RemoveInactiveButtonClick(Sender: TObject);
    procedure lblURLClick(Sender: TObject);
    procedure StaticText1Click(Sender: TObject);
    procedure pcChange(Sender: TObject);
    procedure ledAdapterDescriptionChange(Sender: TObject);
  private
    procedure HandleNewAdapter(ATraffic : TTraffic);
```

```
    procedure HandleFreeze(ATraffic : TTraffic);
    procedure HandleUnFreeze(ATraffic : TTraffic);
    function LocateTraffic(AdapterIndex : DWord) : TTraffic;
    procedure ProcessMIBData;
    procedure ClearDisplay;
    procedure RefreshDisplay;
  public
    { }
  end;

var
  MainForm: TMainForm;
  ActiveTraffic : TTraffic;

implementation
{$R *.dfm}

procedure TMainForm.ClearDisplay;
var
 j:integer;
begin
 TrafficTabs.Tabs.Clear;
 StatusText.Caption:='';
 for j:=0 to GroupBox.ControlCount-1 do
  begin
   if GroupBox.Controls[j] is TCustomEdit
   then TCustomEdit(GroupBox.Controls[j]).Text:='';
  end;
end;

procedure TMainForm.TimerTimer(Sender: TObject);
begin
 Timer.Enabled:=false;
 ProcessMIBData;
 Timer.Enabled:=true;
end;

procedure TMainForm.ClearCountersButtonClick(Sender: TObject);
begin
 ActiveTraffic.Reset;
 RefreshDisplay;
end;

procedure TMainForm.cbOnTopClick(Sender: TObject);
begin
 if cbOnTop.Checked=true
 then FormStyle:=fsSTAYONTOP
 else FormStyle:=fsNORMAL;
end;
procedure TMainForm.FormDestroy(Sender: TObject);
var
 i: integer;
begin
 Timer.OnTimer:=nil;
 ActiveTraffic:=nil;
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  TrafficTabs.Tabs.Objects[i].Free;
end;

procedure TMainForm.TrafficTabsChange(Sender: TObject; NewTab: Integer; var
AllowChange: Boolean);
begin
 if NewTab=-1
 then ActiveTraffic:=nil
 else ActiveTraffic:=TTraffic(TrafficTabs.Tabs.Objects[NewTab]);
 RefreshDisplay;
end;

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

```
begin
 Close;
end;

procedure TMainForm.FormCreate(Sender: TObject);
begin
  Timer.Interval:=1000; // усі розрахунки за 1 сек.
 //
 ClearDisplay;
 ActiveTraffic:=nil;
 pcChange(Sender);
 Timer.Enabled:=True;
end;

procedure TMainForm.RefreshDisplay;
begin
 if not Assigned(ActiveTraffic)
 then
  begin
   ClearDisplay;
   Exit;
  end;
 with ActiveTraffic do
  begin
   FreezeButton.Visible:=Connected;
   UnFreezeButton.Visible:=Connected;
   ClearCountersButton.Visible:=Connected;
   RemoveInactiveButton.Visible:=not Connected;

   FreezeButton.Enabled:=Running;
   UnFreezeButton.Enabled:=not Running;

   ledAdapterDescription.Text:=Description;
   ledMACAddress.Text:=MAC;

   ledSpeed.Text:=BitsToFriendlyString(Speed);

   ledOctInSec.Text:=BytesToFriendlyString(InPerSec);
   ledPeakInSec.Text:=BytesToFriendlyString(PeakInPerSec);
   ledAvgINSec.Text:=BytesToFriendlyString(AverageInPerSec);
   ledTotalIN.Text:=BytesToFriendlyString(InTotal);

   ledOctOUTSec.Text:=BytesToFriendlyString(OutPerSec);
   ledPeakOUTSec.Text:=BytesToFriendlyString(PeakOutPerSec);
   ledAvgOUTSec.Text:=BytesToFriendlyString(AverageOutPerSec);
   ledTotalOUT.Text:=BytesToFriendlyString(OutTotal);

   self.ledStartedAt.Text:=DateTimeToStr(StartedAt);
   self.ledActiveFor.Text:=FriendlyRunningTime;

   StatusText.Caption:=GetStatus;
  end;
end;

procedure TMainForm.ProcessMIBData;
var
 MibArr: IpHlpAPI.TMIBIfArray;
 i: integer;
 ATraffic: TTraffic;
begin
 Get_IfTableMIB(MibArr);  // Беремо поточні MIB дані
 //Мітку не знайдено, або не підключено
 for i:= 0 to -1 + TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[i]);
   if ATraffic.Connected
   then ATraffic.Found:=false;
  end;
  //процесс
```

```
 if Length(MibArr)>0
 then
  begin
   for i:=Low(MIBArr) to High(MIBArr) do
    begin
     ATraffic:=LocateTraffic(MIBArr[i].dwIndex);
     if Assigned(ATraffic)
     then
      begin
       //заново підключаємось
       ATraffic.NewCycle(MIBArr[i].dwInOctets, MIBArr[i].dwOutOctets,
MIBArr[i].dwSpeed);
      end
     else
      begin
       //новий запис у таблицю!
       ATraffic:=TTraffic.Create(MIBArr[i], HandleNewAdapter);
       ATraffic.Found:=true;
       ATraffic.OnFreeze:=HandleFreeze;
       ATraffic.OnUnFreeze:=HandleUnFreeze;
      end;
    end;
  end;
 //Мітка не знайдена
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  if not TTraffic(TrafficTabs.Tabs.Objects[i]).Found
  then TTraffic(TrafficTabs.Tabs.Objects[i]).MarkDisconnected;
 RefreshDisplay;
end;

function TMainForm.LocateTraffic(AdapterIndex : DWord): TTraffic;
var
 j: cardinal;
 ATraffic: TTraffic;
begin
 Result:=nil;
 if TrafficTabs.Tabs.Count=0
 then Exit;

 for j:= 0 to -1+TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[j]);
   if ATraffic.InterfaceIndex=AdapterIndex
   then
    begin
     Result:=ATraffic;
     Result.Found:=true;
     Break;
    end;
  end;
end;

procedure TMainForm.HandleNewAdapter(ATraffic: TTraffic);
begin
 //додаємо адаптер
 TrafficTabs.Tabs.AddObject(ATraffic.IP, ATraffic);
 // вибираємо
 TrafficTabs.TabIndex:=-1+TrafficTabs.Tabs.Count;
end;

procedure TMainForm.FreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.Freeze;
end;

procedure TMainForm.UnFreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.UnFreeze;
end;
```

```
procedure TMainForm.HandleFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.HandleUnFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.RemoveInactiveButtonClick(Sender: TObject);
begin
 if not ActiveTraffic.Connected
 then //точна перевірка
  begin
   ActiveTraffic.Free;
   ActiveTraffic:=nil;
   TrafficTabs.Tabs.Delete(TrafficTabs.TabIndex);
   TrafficTabs.SelectNext(False);
  end;
 RefreshDisplay;
end;

procedure TMainForm.lblURLClick(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.StaticText1Click(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.pcChange(Sender: TObject);
begin
 pnlBottom.Visible:=pc.ActivePage=tsTraffic;
end;

procedure TMainForm.ledAdapterDescriptionChange(Sender: TObject);
begin
  ledAdapterDescription.Hint:=ledAdapterDescription.Text;

ledAdapterDescription.ShowHint:=Canvas.TextWidth(ledAdapterDescription.Text)>led
AdapterDescription.ClientWidth;
end;

end.
```

Файл TCPSYNFlood.dpr основної програми

```
//Файл основної програми до якого підключаються відповідні бібліотеки

program TCPSYNFlood;

uses
  Forms,
  IPHelper in ' IPHelper.pas' ,
  IPHLPAPI in ' IPHLPAPI.pas' ,
  MainFormUnit in ' MainFormUnit.pas'  {MainForm},
  TrafficUnit in ' TrafficUnit.pas' ;

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.Run;
end.
```

unit IPHelper – файл прогнозування трафіку

```
interface

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

const
  NULL_IP       = '   0.  0.  0.  0' ;

//------перетворення визначених імен портів у сервісні імена------

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

  PTTcpConnStatus = ^TTcpConnStatus;
  TTcpConnStatus = record
    LocalIP      : string;
    LocalPort    : string;
    RemoteIP     : string;
    RemotePort   : string;
    Status       : string;
  end;


const
    // для самих розповсюджених сервісів...
  WellKnownPorts: array[1..29] of TWellKnownPort
  = (
    ( Prt: 0; Srv: ' LOOPBACK' ),
    ( Prt: 7; Srv: ' ECHO'  ),      {Пінгування    }
    ( Prt: 9; Srv: ' DISCRD'  ),    { Відмова}
    ( Prt: 13; Srv: ' DAYTIM'  ),   {Час}
    ( Prt: 17; Srv: ' QOTD'  ),     {Вказник на час}
    ( Prt: 19; Srv: ' CHARGEN'  ),  {Генерація символів}
    ( Prt: 20; Srv: ' FTP '  ),     { File Transfer Protocol}
    ( Prt: 21; Srv: ' FTPC'  ),     { File Transfer Control Protocol}
    ( Prt: 23; Srv: ' TELNET'  ),   {TelNet}
    ( Prt: 25; Srv: ' SMTP'  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv: ' TIME'  ),     { Часовий протокол }
    ( Prt: 43; Srv: ' WHOIS' ),     { WHO IS service  }
    ( Prt: 53; Srv: ' DNS '  ),     { Domain Name Service }
    ( Prt: 67; Srv: ' BOOTPS'  ),   { BOOTP Сервер }
    ( Prt: 68; Srv: ' BOOTPC'  ),   { BOOTP Клієнт }
    ( Prt: 69; Srv: ' TFTP'  ),     { стандартний FTP  }
    ( Prt: 70; Srv: ' GOPHER'  ),   { Gopher       }
    ( Prt: 79; Srv: ' FING'  ),     { Finger       }
    ( Prt: 80; Srv: ' HTTP'  ),     { HTTP         }
    ( Prt: 88; Srv: ' KERB'  ),     { Kerberos     }
    ( Prt: 109; Srv: ' POP2'  ),    { Post Office Protocol Version 2 }
    ( Prt: 110; Srv: ' POP3'  ),    { Post Office Protocol Version 3 }
    ( Prt: 119; Srv: ' NNTP'  ),    { Network News Transfer Protocol }
    ( Prt: 123; Srv: ' NTP '  ),    { Network Time protocol          }
    ( Prt: 135; Srv: ' LOCSVC' ),   { Локальні сервіси             }
    ( Prt: 137; Srv: ' NBNAME'  ),  { NETBIOS Імя сервісу         }
    ( Prt: 138; Srv: ' NBDGRAM'  ), { NETBIOS Сервіс датаграм      }
    ( Prt: 139; Srv: ' NBSESS'  ),  { NETBIOS Сессіійний сервіс        }
    ( Prt: 161; Srv: ' SNMP'  )     { Simple Netw. Management Protocol }
    );


//-----------перетворення ICMP кодів помилок у рядок--------------------------
```

```
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_ERROR' , ' 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 величин до рядку ----------------//

  ARPEntryType  : array[1..4] of string = ( ' Other' , ' Invalid' ,
    ' Dynamic' , ' Static'
    );

  TCPConnState  :      // стани підключення TCP
    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 =  // алгоритми часу TCP
    ( ' Const.Timeout' , ' MIL-STD-1778' ,
    ' Van Jacobson' , ' Other'  );

  IPForwTypes   : array[1..4] of string =   // IP пересилання методів
    ( ' other' , ' invalid' , ' local' , ' remote'  );

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


//---------------експортуємі данні-----------------------------------------------
-

// дані перетворюються у Tstrings для представлення на дісплей
procedure Get_AdaptersInfo( List: TStrings );
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
procedure Get_IfTable( NameList, ItemList: TStrings );
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );



procedure Get_RecentDestIPs( List: TStrings );
```

```
// додаємо функцію
procedure Get_OpenConnections( List: TList );



// утіліти 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 утіліти-----------------------------------------------

{ перетворюємо наступний токен у рядок, потім зчитуємо рядок та видаляємо його }
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;

//------------------------------------------------------------------------------
{ перетворюємо цифрові MAC-адреси у ww-xx-yy-zz строку }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := ' 00-00-00-00-00-00' ;
    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 у крапкову десяткову строку}
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 у 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 номера порту у мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

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

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

//---------------------------------------------------------------------------
{ general,  fixed network parameters }
procedure Get_NetworkParams( List: TStrings );
var
  InfoSize      : Longint;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  InfoSize := 0;
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  GetMem( pBuf, InfoSize );
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  if ErrorCode = ERROR_SUCCESS then
    with PTFixedinfo(pBuf)^ do
    begin
```

```
      List.Add( ' HOSTNAME          : '  + string( Імя хосту ) );
      List.Add( ' DOMAIN            : '  + string( імя домену ) );
      List.Add( ' SCOPE             : '  + string( ScopeID ) );
      List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
      List.Add( ' ROUTING ENABLED   :'  + IntToStr( Дозволена маршрутизація ) );
      List.Add( ' PROXY   ENABLED   :'  + IntToStr( Дозволений Proxy ) );
      List.Add( ' DNS     ENABLED   :'  + IntToHex( Дозволений DNS,8 ) );
    end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  FreeMem(pBuf);
end;

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

//---------------------------------------------------------------------------
//процедура отримання трафіку та занесення у таблицю
procedure Get_IfTable( NameList, ItemList: TStrings );
var
  IfRow         : TMibIfRow;
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  if (not Assigned( NameList ))
  or (not Assigned( ItemList ))  then EXIT;
  NameList.Clear;
  ItemList.Clear;
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

   // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 1 to NumEntries do
      begin
        IfRow := PTMibIfRow( pBuf )^;
        with IfRow do
        begin
          SetLength( sDescr, dwDescrLen );
          move( bDescr, sDescr[1], Length( sDescr ) );
          sDescr := trim( sDescr );
          NameList.Add( sDescr );
          ItemList.Add( Format( ' %0.8x|%2d| %16s| %4d| %8d| %8d| %8d' ,
            [dwIndex, dwType,
            MacAddr2Str( TMacAddress( bPhysAddr ), dwPhysAddrLen )
              , dwMTU, dwSpeed,
```

```
              dwInOctets, dwOutOctets,
              dwOPerStatus] )
              );
        end;
        inc( pBuf, SizeOf( IfRow ) );
      end;
    end
    else  begin
      NameList.Add( ‘ без даних’ );
      ItemList.Add( ‘ немає даних’  );
    end;
  end
  else begin
    NameList.Add( ‘ Oops’ );
    ItemList.Add( SysErrorMessage( GetLastError ) );
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IfRow ) );
  FreeMem( pBuf );
end;


//---------------------------------------------------------------------------
//Занесення даних у таблицю MIB
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
var
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам’ яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

   // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( MIBIfArray, NumEntries );
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 0 to pred(NumEntries) do
      begin
        MIBIfArray[i] := PTMibIfRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIfRow ) );
      end;
    end
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMIBIfRow ) );
  FreeMem( pBuf );
end;



//---------------------------------------------------------------------------
//Заносимо дінні про адреси у таблиці MIB
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
```

```
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;

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



//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
procedure Get_AdaptersInfo( List: TStrings );
var
  Error,
  BufLen      : DWORD;
  P             : Pointer;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  Descr,
  LocalIP,
  GatewayIP,
  DHCPIP      : string;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  BufLen := SizeOf( AdapterInfo^ );
  New( AdapterInfo );
  Error := GetAdaptersInfo( AdapterInfo, @BufLen );
  P := AdapterInfo;
  if Error = NO_ERROR then
  begin
    while P <> nil do
      with TIP_ADAPTER_INFO(P^) do
      begin
        SetLength( Descr, SizeOf( Description ) );
        Descr := Trim( string( Description ) );
        //
        if IPAddressList.IpAddress[1] <> #0 then
          LocalIP := IPAddressList.IpAddress
        else
          LocalIP := NULL_IP;
        //
        if GateWayList.IPAddress[1] <> #0 then
          GateWayIP := GatewayList.IPAddress
        else
```

```
          GateWayIP := NULL_IP;
        //
        if DHCPServer.IPAddress[1] <> #0 then
          DHCPIP := DHCPServer.IPAddress
        else
          DHCPIP := NULL_IP;

        List.Add( Descr );
        List.Add( Format(
          ' %8.8x|%6s|%16s|%2d|%16s|%16s|%16s' ,
          [Index, AdaptTypes[aType],
          MacAddr2Str( TMacAddress( Address ), AddressLength ),
            DHCPEnabled, LocalIP, GatewayIP, DHCPIP] )
            );
        List.Add( '    ' );
        P := Next;  //  TIP_ADAPTER_INFO(P^).Next  points to next entry
      end // with
    end // while
  else
    List.Add( SysErrorMessage( Error ) );
  Dispose( AdapterInfo );
end;



//---------------------------------------------------------------------------
{ записуємо час завантаження трафіка мережі IP }
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; // Розташування BAD_HOST_NAME,й.. т.і.
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;



//---------------------------------------------------------------------------
{ ARP-таблиця записує відношення між IP та MAC-адресам.
}
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: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetIPNetTable( PTMIBIpNetTable( pBuf ), @TableSize, false );
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-cache empty.'  );
    EXIT;
  end;
  // заносимо у таблицю
  GetMem( pBuf, TableSize );
  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 ) ); // записуємо 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 ) );

  // we _must_ restore pointer!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPNetRow ) );
  FreeMem( pBuf );

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: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  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 наступний запис з
навантаженістю мережного трафіку
        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;


//---------------------------------------------------------------------------
// Опитуємо відкриті підключення до мережі
procedure Get_OpenConnections( List: TList );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
  CStat         : PTTcpConnStatus;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  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 наступний запис з
навантаженістю мережного трафіку
        with TCPRow do
          if dwState in [2,5] then   //
          begin
            New( CStat );
            CStat^.LocalIP   := IPAddr2Str( dwLocalAddr );
            CStat^.LocalPort := Port2Svc( Port2Wrd( dwLocalPort ));
            if dwRemoteAddr <> 0 then
            begin
              CStat^.RemoteIP     := IPAddr2Str( dwRemoteAddr );
```

```
            CStat^.RemotePort   := Port2Svc( Port2Wrd( dwRemotePort ));
          end
          else begin
            CStat^.RemoteIP   := ' ...' ;
            CStat^.RemotePort := ' ...' ;
          end;
          CStat^.Status       := TCPConnState[dwState];
          List.Add( CStat );
        end;
      inc( pBuf, SizeOf( TMIBTCPRow ) );
    end;
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;



//-------------------------------------------------------------------------
//Записуємо статистику трафіка по TCP
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі  :'  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мініміальний час             :'  + IntToStr( dwRTOMin ) + '  ms'
);
      List.Add( ' Максимальний час             :'  + IntToStr( dwRTOMax ) + '  ms'
);
      List.Add( ' Максимальна кількість підключень :'  + IntToStr(
dwRTOAlgorithm ) );
      List.Add( ' Активні підключення             :'  + IntToStr( dwActiveOpens
) );
      List.Add( ' Пасивні підключення            :'  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Помилка невдалого відкриття     :'  + IntToStr( dwAttemptFails
) );
      List.Add( ' Скидання встановленого підключення  :'  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.:'  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отриманий сегмент        :'  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлений сегмент            :'  + IntToStr( dwOutSegs ) );
      List.Add( ' Переправлений сегмент   :'  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилка входження         :'  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виходу          :'  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні зв'язки    :'  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );

end;

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

//Запис часу та завантаженості трафіку по UDP
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: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам’ ять. Викликаємо знову
  GetMem( pBuf, TableSize );

  // заносимо у таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  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 наступний запис з
навантаженістю мережного трафіку
        with UDPRow do
          List.Add( Format( ‘ %15s : %-6s’ ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ‘ без даних.’  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------------------------------------------------
procedure Get_IPAddrTable( List: TStrings );

//Запис часу та завантаженості трафіку по IP

var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у 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( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

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


(*
//---------------------------------------------------------------------------
//Запис IP адресів до MIB

procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;
```

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

//---------------------------------------------------------------------------
{ отримуємо данні з таблиць маршрутизації }
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: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize, true
    );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // заносимо у 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
          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
             ] ) );
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

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

// Надання статистики по IP
procedure Get_IPStatistics( List: TStrings );
```

```
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Пересилання дозволено       : '  + ' Так'  )
      else
        List.add( ' Пересилання дозволено       : '  + ' Ні'  );
      List.add( ' Вбудований TTL              : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримана датаграмма       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилки заголовку     (У)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Помилка адреси    (У)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Невідомий протокол  (У)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Відхілені датаграми     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграми встановлені    : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запит             : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрут відхилений      : '  + inttostr( dwRoutingDiscards )
);
      List.add( ' Немає маршруту          (Out): '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебирання часу     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Перебирання запитів      : '  + 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( ' Датаграму фрагментовано     : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут у таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//---------------------------------------------------------------------------
// Надання статистики по UDP

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 (У)     : '  + inttostr( dwInDatagrams ) );
      List.add( ' Датаграми (З)    : '  + inttostr( dwOutDatagrams ) );
      List.add( 'Немає портів           : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилки     (У)     : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
```

```
end;

//---------------------------------------------------------------------------
// Надання статистики по ICMP

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( ' Повідомлення прийнято    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка               : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );

      ICMPIn.Add( ' Запит адреси маски : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор адреси маски   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats^.OutStats do
    begin
      ICMPOut.Add( 'Повідомлення відправлено: '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка               : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Exo повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPOut.Add( 'Запит адреси маски: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Повтор адреси маски   : '  + 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.
```

unit IPHLPAPI – бібліотека API функцій для роботи з IP мережею

```
interface
uses
  Windows, winsock;
const

  VERSION       = '1.3';


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

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

  NETBIOSTypes  : array[0..8] of string[20] =
    ( 'UNKNOWN', 'BROADCAST', 'PEER_TO_PEER', '', 'MIXED', '', '', '', 'HYBRID'
      );

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


  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;

 //------Структура IP-адрес -------------------------------------------------

  PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
  TIP_ADDRESS_STRING = array[0..15] of char; //  IP в xxx.xxx.xxx.xxx рядок
  //
  PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
  TIP_ADDR_STRING = packed record //
    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[0..MAX_HOSTNAME_LEN + 4] of char;
    DomainName: array[0..MAX_DOMAIN_NAME_LEN + 4] of char;
    CurrentDNSServer: PTIP_ADDR_STRING;
    DNSServerList: TIP_ADDR_STRING;
    NodeType: UINT;
    ScopeID: array[0..MAX_SCOPE_ID_LEN + 4] of char;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

//----------INTERFACE структура-------------------------------------------------


  PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] 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[1..MAXLEN_IFDESCR] of char; //byte;
  end;

  TMIBIfArray = array of TMIBIFRow;


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

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

  TTIME_T = array[1..325] of byte; //

  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;
    Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
    AddressLength: UINT;
    Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;
    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: TTIME_T; //??
    LeaseExpires: TTIME_T; //??
  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;

  TMibIPAddrArray = array of TMIBIPAddrRow;


  //
  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;




  //-----------------імпортовано з IPHLPAPI.DLL---------------------------------
  -

function GetAdaptersInfo( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetNetworkParams( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetTcpTable( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

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

```
stdcall; external 'IPHLPAPI.DLL';

function GetUdpTable( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetUdpStatistics( pStats: PTMibUdpStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpStatistics( pStats: PTMibIPStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpNetTable( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpAddrTable( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpForwardTable( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

function GetIcmpStatistics( pStats: PTMibICMPInfo ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetRTTAndHopCount( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL;
stdCall; external 'IPHLPAPI.DLL';
function GetIfTable( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetIfEntry( pIfRow: PTMibIfRow ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

implementation
end.
```

unit TrafficUnit – визначення параметрів трафіку;

```
interface

uses SysUtils, Windows, IPHelper, IPHLPAPI;


type
  TTraffic = Class;

  TNewInstanceEvent = procedure(Sender :TTraffic) of object;
  TFreezeEvent = procedure(Sender :TTraffic) of object;

  TTraffic = Class
  private
    FIP: string;
    FMac: string;
    FInPerSec: Dword;
    FInTotal: Dword;
    FPeakInPerSec: Dword;
    FInterfaceIndex: DWord;
    FActiveCountIn: Dword;
    FSecondsActive: Cardinal;
    FPrevCountIn: DWord;
    FDescription: string;
    FOutTotal: Dword;
    FPeakOutPerSec: Dword;
    FOutPerSec: Dword;
    FPrevCountOut: DWord;
    FActiveCountOut: Dword;
    FAverageInPerSec: Dword;
    FAverageOutPerSec: Dword;
    FStartedAt: TDateTime;
    FRunning: boolean;
    FOnFreeze: TFreezeEvent;
    FOnUnFreeze: TFreezeEvent;
    FConnected: boolean;
    FFound: boolean;
    FSpeed: DWord;

    function GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
  public
    property Found : boolean read FFound write FFound;
    property Connected : boolean read FConnected;
    property Running : boolean read FRunning;
    property InterfaceIndex : DWord read FInterfaceIndex;
    property IP : string read FIP;
    property Mac : string read FMac;
    property Description : string read FDescription;
    property StartedAt : TDateTime read FStartedAt;
    property SecondsActive : Cardinal read FSecondsActive;
    property Speed : DWord read FSpeed;
    property ActiveCountIn : Dword read FActiveCountIn; { }
    property PrevCountIn : DWord read FPrevCountIn; { }
    property InPerSec : Dword read FInPerSec; { байт підраховано в останній
період }
    property AverageInPerSec : Dword read FAverageInPerSec; { У середньому }
    property InTotal : Dword read FInTotal; { загальна кількість байтів }
    property PeakInPerSec : Dword read FPeakInPerSec; { максимальне число
байтів}

    property ActiveCountOut : Dword read FActiveCountOut; { підраховує число
байтів при передачі }
    property PrevCountOut : DWord read FPrevCountOut; { попереднє підрахування
байтів }
    property OutPerSec : Dword read FOutPerSec; { Кількість байтів у останій
період }
    property AverageOutPerSec : Dword read FAverageOutPerSec; { }
```

```
    property OutTotal : Dword read FOutTotal; { Загальна кількість байтів
передано }
    property PeakOutPerSec : Dword read FPeakOutPerSec; { Максимальна кількість
байтів передано }

    procedure NewCycle(const InOctets, OutOctets, TrafficSpeed : Dword);
    procedure Reset;
    procedure Freeze;
    procedure UnFreeze;
    procedure MarkDisconnected;
    function GetStatus : string;
    function FriendlyRunningTime:string;
    constructor Create(const AMibIfRow : TMibIfRow; OnNewInstance :
TNewInstanceEvent);
  published
    property OnFreeze :TFreezeEvent read FOnFreeze write FOnFreeze;
    property OnUnFreeze :TFreezeEvent read FOnUnFreeze write FOnUnFreeze;
  end;

  function BytesToFriendlyString(Value : DWord) : string;
  function BitsToFriendlyString(Value : DWord) : string;

implementation

function BytesToFriendlyString(Value : DWord) : string;
const
 OneKB=1024;
 OneMB=OneKB*1024;
 OneGB=OneMB*1024;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 B',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 KB', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 MB', Value/OneMB)
end;

function BitsToFriendlyString(Value : DWord) : string;
const
 OneKB=1000;
 OneMB=OneKB*1000;
 OneGB=OneMB*1000;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 bps',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 Kbps', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 Mbps', Value/OneMB)
end;

constructor TTraffic.Create(const AMibIfRow: TMibIfRow; OnNewInstance :
TNewInstanceEvent);
var
 Descr: string;
begin
 inherited Create;

 FRunning:=true;
 FConnected:=true;

 self.FInterfaceIndex:=AMibIfRow.dwIndex;
 self.FIP:=GetIPFromIFIndex(self.InterfaceIndex);
```

```
 self.FMac:=MacAddr2Str(TMacAddress(AMibIfRow.bPhysAddr),
AMibIfRow.dwPhysAddrLen);

 SetLength(Descr, Pred(AMibIfRow.dwDescrLen));
 Move(AMibIfRow.bDescr, Descr[1], pred(AMibIfRow.dwDescrLen));
 self.FDescription:=Trim(Descr);

 self.FPrevCountIn:=AMibIfRow.dwInOctets;
 self.FPrevCountOut:=AMibIfRow.dwOutOctets;

 self.FStartedAt:=Now;
 self.FSpeed:=AMibIfRow.dwSpeed;

 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;
  if Assigned(OnNewInstance)
 then OnNewInstance(self);
end;

procedure TTraffic.NewCycle(const InOctets, OutOctets, TrafficSpeed: Dword);
begin
 inc(self.FSecondsActive);
 if not Running
 then Exit;
 FSpeed:=TrafficSpeed;
 // прийнято
 self.FInPerSec:=InOctets-self.PrevCountIn;
 Inc(self.FInTotal, self.InPerSec);
 if InPerSec>0
 then Inc(FActiveCountIn);
 if InPerSec>PeakInPerSec
 then FPeakInPerSec:=InPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageInPerSec:=InTotal div ActiveCountIn
 except
  self.FAverageInPerSec:=0;
 end;
 FPrevCountIn:=InOctets;
 // передано
 self.FOutPerSec:=OutOctets-self.PrevCountOut;
 Inc(self.FOutTotal,self.OutPerSec);
 if OutPerSec>0
 then Inc(FActiveCountOut);
 if OutPerSec>PeakOutPerSec
 then FPeakOutPerSec:=OutPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageOutPerSec:=OutTotal div ActiveCountOut
 except
  self.FAverageOutPerSec:=0;
 end;
 FPrevCountOut:=OutOctets;
end;

function TTraffic.GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
var
 i: integer;
 IPArr: TMIBIPAddrArray;
begin
 Result:='Не знайдено!'; //...
 Get_IPAddrTableMIB(IpArr); // беремо таблицю IP-адрес
 if Length(IPArr)>0
```

```
 then
  for i:=low(IPArr) to High(IPArr) do  // проглядаємо індекси у таблиці...
   if IPArr[i].dwIndex=InterfaceIndex
   then
    begin
     Result:=IPAddr2Str(IParr[i].dwAddr);
     Break;
    end;
end;

procedure TTraffic.Reset;
begin
 self.FPrevCountIn:=InPerSec;
 self.FPrevCountOut:=OutPerSec;
 self.FStartedAt:=Now;
 FSecondsActive:=0;
 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;
end;

procedure TTraffic.Freeze;
begin
 FRunning:=false;
 if Assigned(FOnFreeze)
 then OnFreeze(Self);
end;

procedure TTraffic.UnFreeze;
begin
 FRunning:=true;
 if Assigned(FOnUnFreeze)
 then OnUnFreeze(Self);
end;

procedure TTraffic.MarkDisconnected;
begin
 self.FConnected:=false;
 self.FRunning:=false;
end;
function TTraffic.GetStatus: string;
begin
 if self.Connected
 then Result:='Підключено'
 else Result:='Не підключено';
 if self.Running
 then Result:=Result+', Запущено'
 else Result:=Result+', Не запущено';
end;
function TTraffic.FriendlyRunningTime: string;
var
 H,M,S: string;
 ZH,ZM,ZS: integer;
begin
 ZH:=SecondsActive div 3600;
 ZM:=Integer(SecondsActive) div (60-ZH*60);
 ZS:=Integer(SecondsActive)-(ZH*3600+ZM*60);
 H:=Format('%.2d',[ZH]);
 M:=Format('%.2d',[ZM]);
 S:=Format('%.2d',[ZS]);
 Result:=H+':'+M+':'+S;
end;
end.
```

unit about – підпрограмма про розробників та місце розроблення програми;

```
interface

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

type
  TForm2 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Button1: TButton;
    Label7: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```