МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Слівець Давид Сергійович

Програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей

Спеціальність: 125 «Кібербезпека»

Освітній ступінь: бакалавр

Науковий керівник:

Смірнов Сергій Анатолійович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

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

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

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

Слівецю Давиду Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей*

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

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

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

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей*

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

Слівець Д.С. Програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи кібербезпеки для сканування мережі на наявність вразливостей.

Метою розробки є програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей.

Результат роботи – програмна реалізація системи кібербезпеки для сканування мережі на наявність вразливостей.

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 Embarcadero Delphi.

Ключові слова: кібербезпека, сканування мережі

ABSTRACT

Slivets D.S. Cybersecurity system software to scan the network for vulnerabilities. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

This undergraduate qualification has developed software that is designed for a cybersecurity system to scan the network for vulnerabilities.

The purpose of the development is cybersecurity system software to scan the network for vulnerabilities.

The result is a software implementation of a cybersecurity system to scan the network for vulnerabilities.

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 was developed in the Embarcadero Delphi environment.

Keywords: cybersecurity, network scanning

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електрона обчислювальна машина
ІБ	–	інформаційна безпека
ІТ	–	інформаційні технології
НСД	–	несанкціонований доступ
ПЗ	–	програмне забезпечення
СУБД	–	система управління базами даних
СВВ	–	система виявлення вторгнень
CVE	–	Common Vulnerabilities and Exposures
NVD	–	National Vulnerability Database

ВСТУП

Актуальність теми. Як правило, тест на проникнення починається зі сканування на уразливості. Гарний сканер містить у собі завжди актуальну базу відомих уразливостей і, скануючи вашу мережу, повідомляє про наявність тієї або іншої. Наша подальша робота полягає в тому, щоб перевірити, чи дійсно кожна зі знайдених уразливостей піддається експлуатації, тому що сканери уразливостей часто дають неправильні спрацьовування.

Наявність уразливостей в інформаційних системах, вузлах інфраструктури або елементах комплексу захисту інформації є великою проблемою для підрозділів ІТ і ІБ. Звичайно, пошуком проломів можна займатися вручну, але це буде вкрай працезатратний процес, який займе багато часу при високій імовірності що-небудь не помітити.

Тому найкраще використовувати автоматичні засоби для пошуку уразливостей і слабких місць в інформаційній інфраструктурі підприємства, такі як сканери захищеності або сканери уразливостей. У цей час інструменти подібного класу дозволяють вирішувати завдання широкого профілю. Це – і сканування мережі, і перебір паролів, і пошук невстановлених патчів і небезпечних версій ПЗ, і виявлення паролів і облікових записів, установлених за замовчуванням, і пошук відкритих портів і запущених небезпечних служб, а також відстеження інших елементів, які потенційно несуть погрозу інформаційної безпеки. Також дані інструменти підтримують велика кількість операційних систем, засобів захисту інформації, мережного устаткування й інших об'єктів, щоб максимально охопити інфраструктуру підприємства.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей.

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

– Огляд існуючих систем кібербезпеки для сканування мережі на наявність вразливостей.

– Дослідження системи кібербезпеки для сканування мережі на наявність вразливостей.

– Програмна реалізація системи кібербезпеки для сканування мережі на наявність вразливостей.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для сканування мережі на наявність вразливостей.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Застосування сканерів уразливостей дозволяє вирішувати різні завдання. Такі інструменти використовують не тільки для самостійної перевірки на наявність проломів в інфраструктурі підприємства, але й для виконання вимог регуляторів.

Функціональність сканера уразливостей дає, наприклад, можливість провести інвентаризацію IT-ресурсів і визначити, які застосунки і якої версії встановлені на робочих станціях і серверах. При цьому сканер покаже, яке ПЗ має уразливості, і запропонує встановити патч, обновити версію або зупинити ті або інші служби й відключити протоколи, якщо вони являють собою загрозу інформаційної безпеки. Якщо присутні помилки в скриптах, це буде також виявлене сканером.

Також можна провести сканування мережі, скласти її карту й визначити, які саме мережні пристрої в інфраструктурі підприємства використовуються. Будуть також визначені всі піддомени. Відразу ж можна виявити відкриті порти, запущені мережні сервіси, які являють загрозу для безпеки. На мережних пристроях буде зроблений пошук уразливостей, які можна буде закрити установкою патчів, відновленням або зміною конфігурацій.

Крім того, сканер дозволяє перевірити на стійкість використовувані паролі на сервісах з доступною авторизацією, при цьому будуть виявлені паролі, установлені за замовчуванням. Буде зроблений і брутфорс (повний перебір можливих варіантів) з використанням актуальної бази паролів.

Сканери уразливостей дозволяють також сканувати засобу захисту інформації й визначати, коли можна встановити нові патчі, обновити програмне

забезпечення, змінити конфігурацію й настроювання, а також перевірити актуальність баз сигнатур.

Сучасні сканери уразливостей підтримують практично всі сучасні операційні системи, як серверні, так і користувацькі. Також усе більшу популярність починають набирати хмарні застосунки такого роду.

За результатами перевірки всі сканери дозволяють сформувати звіти різного формату й призначення, де буде відображена вся картина по уразливостях в інфраструктурі, а також дані рекомендації з їхнього усунення. Кожній уразливості буде зіставлений номер з баз CVE (Common Vulnerabilities and Exposures), NVD (National Vulnerability Database). Деякі з інструментів дозволяють робити звіти для надання керівництву.

1.2 Область застосування

Світовий ринок сканерів уразливостей (Vulnerability Management / Assessment) активно розвивається. Дані інструменти стали повноцінними системами для керування уразливостями, які можна вести як проекти. У свою чергу проекти по відстеженню уразливостей перетворюються в процеси, у яких беруть участь представники різних підрозділів. Також виробники сканерів пропонують інтеграцію із системами керування ризиками або патч-менеджменту, платформами по керуванню інцидентами (Incident Response Platform), процесами безпечної розробки (Security Development), не говорячи вже про SIEM.

Ще однієї важливою особливістю є використання сканерів для відповідності вимогам PCI DSS. Деякі вендори, наприклад Tenable, Qualys або Rapid7, є дозволеними постачальниками послуг сканування (ASV) PCI SSC.

Також варто відзначити, що в останні роки намітилася явна тенденція до використання хмарних сканерів уразливостей. Для деяких замовників даний варіант стає кращим, оскільки зникає необхідність виділяти ресурси під розміщення сканера у своїй інфраструктурі.

Виробники пропонують як комерційні версії своїх продуктів, так і безкоштовні. Останні звичайно надаються із сильно обмеженими функціональними можливостями й у вигляді хмарного сервісу.

Дослідницька компанія IDC провела аналіз ринку систем керування уразливостями за 2020 рік і опублікувала свої виводи в матеріалі «Worldwide Device Vulnerability Management Market Shares, 2020: Finding the Transitional Elements Between Device Assessment Scanning and Risk-Based Remediation».

За результатами дослідження, проведеного компанією Geoactive Group, обсяг ринку Vulnerability Management / Assessment досягнеться 2,1 млрд доларів в 2021 році.

Згідно з аналітичним звітом Gartner «Market Guide for Vulnerability Assessment», трьома домінуючими вендорами по розробці сканерів безпеки є Tenable (близько 30 000 замовників), Qualys (близько 16 000 замовників) і Rapid7 (ледве більше 9 000 замовників). На цих гравців доводиться більша частина прибутки в галузі. У тому ж звіті було відзначено, що серед помітних вендорів на цьому ринку також присутні F-Secure, Positive Technologies, Tripwire і Greenbone Networks.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей, 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

Nessus Vulnerability Scanner

Одним з найбільш популярних сканерів уразливостей на ринку є Nessus Vulnerability Scanner. Він став свого роду стандартом для сканерів уразливостей. Споконвічно він стартував як проект із відкритим кодом. Далі його придбала компанія Tenable, і тепер він є комерційним продуктом (версія Professional). Незважаючи на це, в Nessus Scanner як і раніше є «Home» версія, яка поширюється безкоштовно, але має обмеження в 16 IP адрес. Саме цю версію ми й будемо розглядати в даній інструкції із застосування.

Будучи «хакером», після сканування одержуємо повний список уразливостей, для яких необхідно тільки знайти експлойти. На жаль, сканери уразливостей дуже «гучні» і пильні адміністратори можуть виявити їхню роботу. Проте, далеко не всі організації мають таких адміністраторів.

Не варто забувати важливі моменти відносно сканерів уразливостей. Вони не можуть виявити уразливості 0-day. Як і антивірусні програмні продукти, їх бази повинні обновлятися щодня, щоб бути ефективними.

Будь-який поважаючий себе пентестер, повинен бути знаком з Nessus Scanner. Багато досить великі організацій по усьому світу використовують його в комплексі інформаційної безпеки.

З недавнього часу навіть уряд США початок його використовувати для сканування уразливостей. Майже кожний федеральний офіс і військова база США в усьому світі тепер застосовує Nessus.

Давайте розглянемо, що являє собою ця програма в роботі.

Крок 1. Скачати Nessus Scanner безкоштовно

Знайти безкоштовну домашню версію Nessus Home на сайті Tenable непросто.

Для одержання безкоштовної версії потрібна реєстрація, тому вам потрібно буде вказати свій email адреса для одержання коду активації.

Далі вам буде запропоновано вибрати тип і розрядність вашої системи, підтримуються Windows Server, Maxos X, Linux, Freebsd, GPG Keys.

І далі починається саме завантаження дистрибутива. Після завантаження запускаємо установник і проходимо всі кроки. Процес інтуїтивно зрозумілий, тому не будемо його описувати.

Крок 2. Запуск Nessus

Після завершення установки, відкриється браузер за замовчуванням з повідомленням, як показано нижче. Nessus побудований на клієнт-серверній архітектурі. Ви встановили сервер на localhost, а в ролі клієнта виступає браузер.

Ви, швидше за все, одержите повідомлення, у якому говориться: «Your connection is not secure». Натисніть «Advanced».

Потім додайте виключення для підключення Nessus по 8834 портові.

Крок 3. Настроювання Nessus Home

Практично все готове для пошуку уразливостей!

Вам необхідно створити обліковий запис. Саме її потрібно буде вказувати для входу в Nessus.

Після введення вашого логіна й пароля буде необхідно активувати продукт. Знаходимо лист із кодом активації в себе в пошті й уводимо його у відповідне поле на сторінці вашого Nessus.

Коли це буде зроблено, Nessus почне завантажувати всі актуальні відновлення й плагіни, необхідні для пошуку уразливостей у вашій мережі. Процес може зайняти якийсь час.

Крок 4. Запуск сканування уразливостей

Коли Nessus завершить відновлення, вас зустріне от такий екран, як показано нижче. Натисніть "New Scan".

Це відкриє нову сторінку, на якій ви зможете вибрати тип сканування. Звернете уваги, там завжди втримуватися самі актуальні на сьогоднішній день моделі погроз.

Давайте натиснемо на "Basic Network Scan".

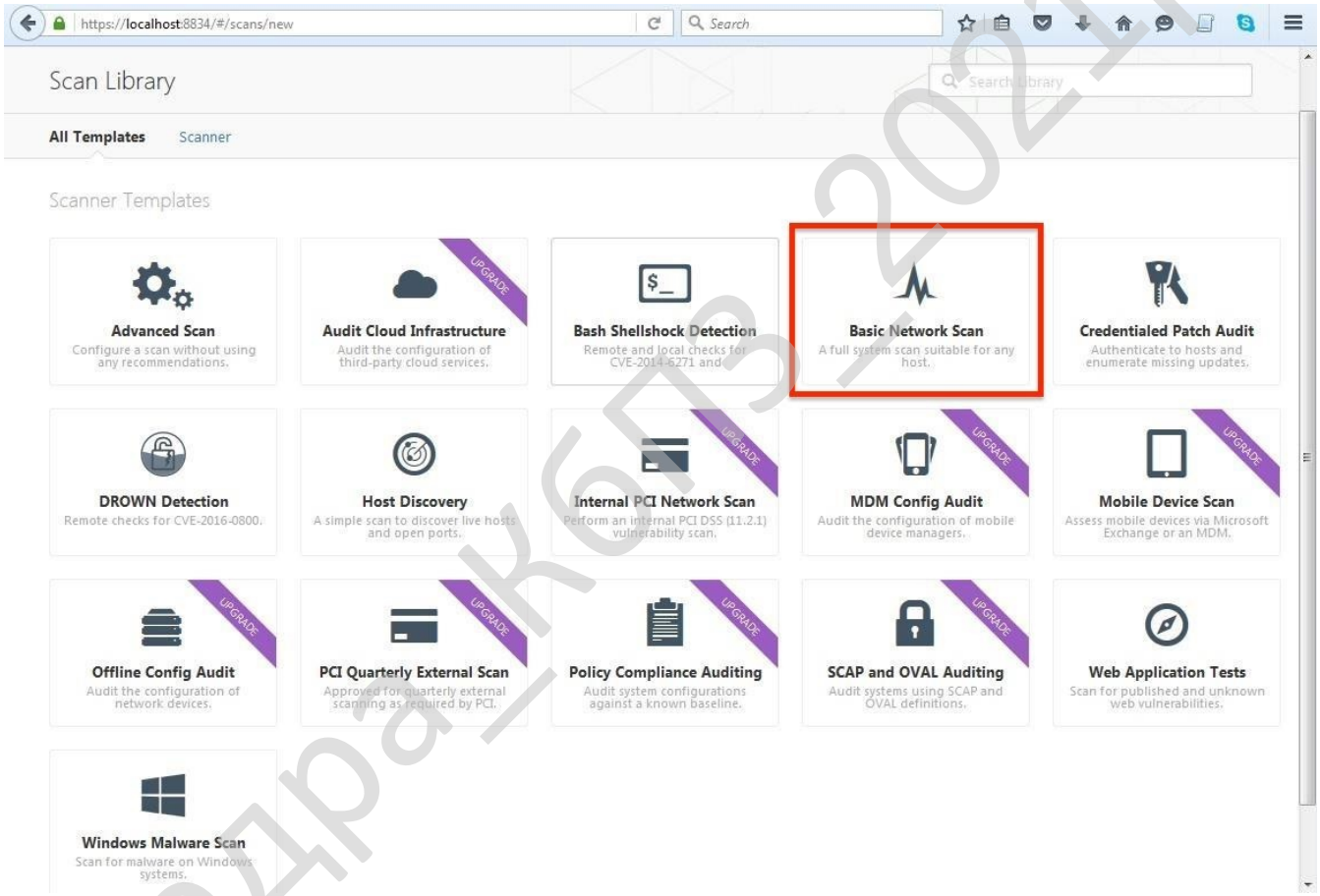

Рисунок 2.1 – Інтерфейс користувача Nessus

Відкриється сторінка, на якій вас попросять указати ім'я вашого сканування (можна вказати будь-яке зрозуміле для вас, наприклад, First Scan). Також необхідно буде вказати вузли, які будемо сканувати. Можна вказати всю підмережу IP адрес 192.168.1.0/24. Натискаємо "Save".

Тепер натискаємо кнопку "Launch", щоб запустити сканування уразливостей.

Крок 5. Перегляд результатів сканування

За результатами сканування ми одержуємо список з IP адресами й пов'язані з ними ризики. Ризики мають колірне кодування.

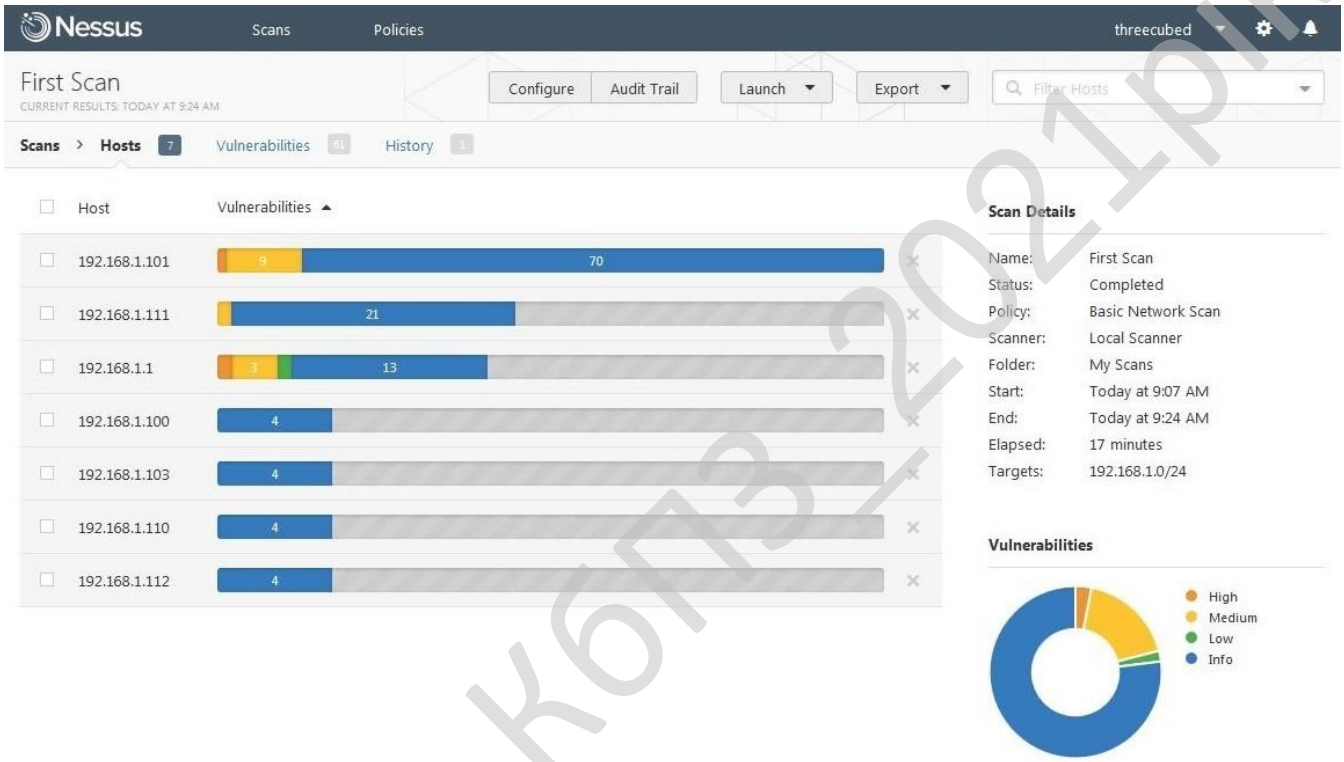

Рисунок 2.2 – Список з IP адресами й пов'язані з ними ризики

Натискаємо "vulnerabilities" у верхньому меню, щоб відобразити всі уразливості, виявлені в мережі.

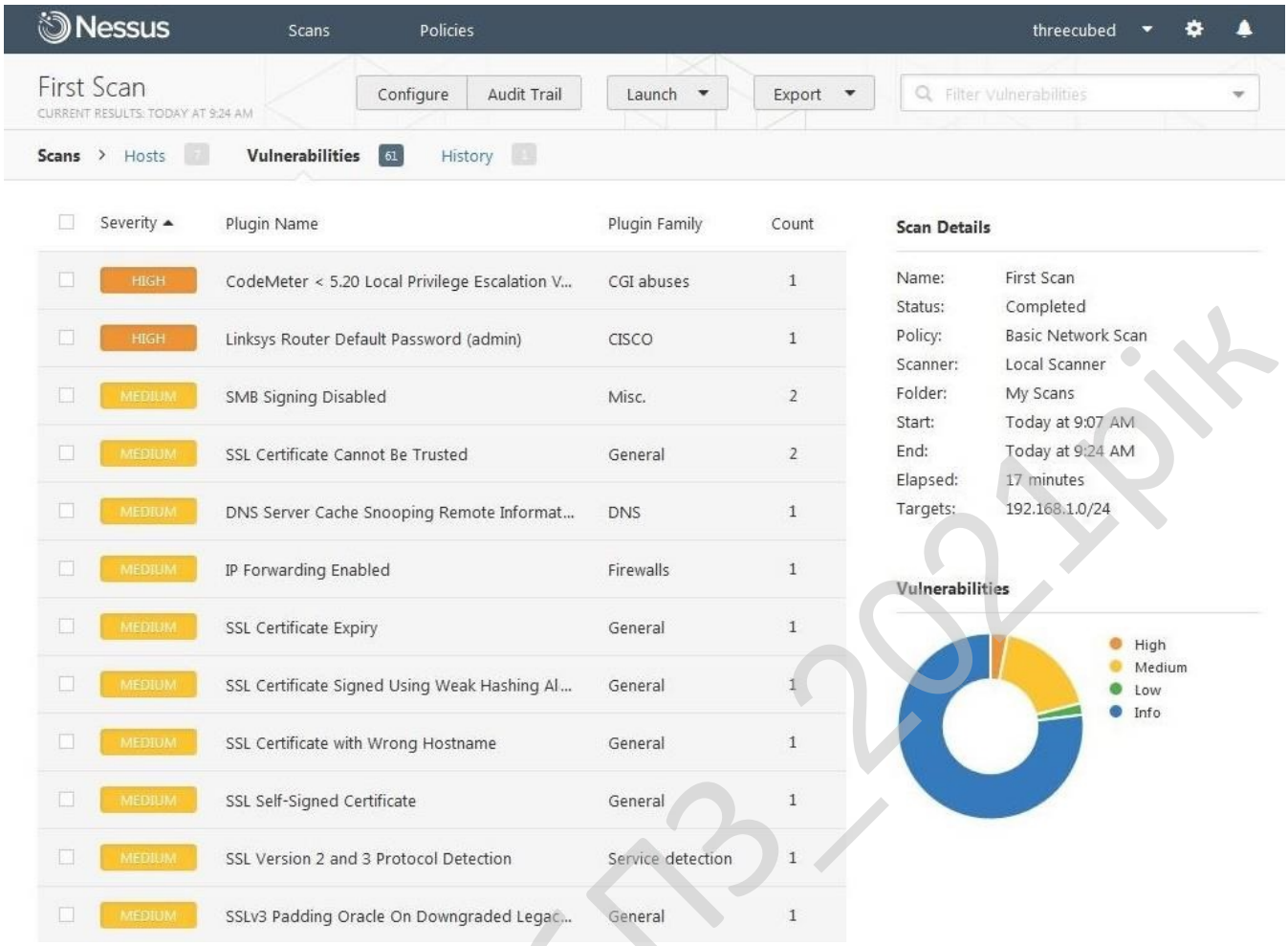

Рисунок 2.3 – Усі уразливості, виявлені в мережі

Якщо кликнути по конкретній уразливості, то ми одержимо більш детальну інформацію. Нижче наведений приклад уразливості "Codemeter".

Важливо відзначити, що крім опису уразливості, у звіті присутній також і спосіб її виправлення й закриття (розділ Solution).

Результати сканування можна зберегти в різних форматах. Відкрийте вкладку "Експорт" виберіть формат файлу звіту.

Nessus Vulnerability Scanner від компанії Tenable навіть у безкоштовній Home версії являє собою досить простій у використанні, але в той же час потужний сканер уразливостей. Головною його гідністю є те, що в ньому завжди можна знайти актуальні моделі погроз, на можливість експлуатації яких він швидко і якісно перевірить вашу мережу.

Помніть, що успіх якісної інформаційної безпеки - це регулярний аудит!

Але не забувайте, що сканування чужих мереж, може мати наслідку у вигляді проблем із законом!

F-Secure Radar

Цей сканер уразливостей є продуктом компанії F-Secure, яка активно працює на ринку антивірусів. Radar – хмарний застосунок, для повноцінної роботи якого необхідна установка агентів. На даний момент підтримується сумісність із ОС сімейств Windows і Linux.

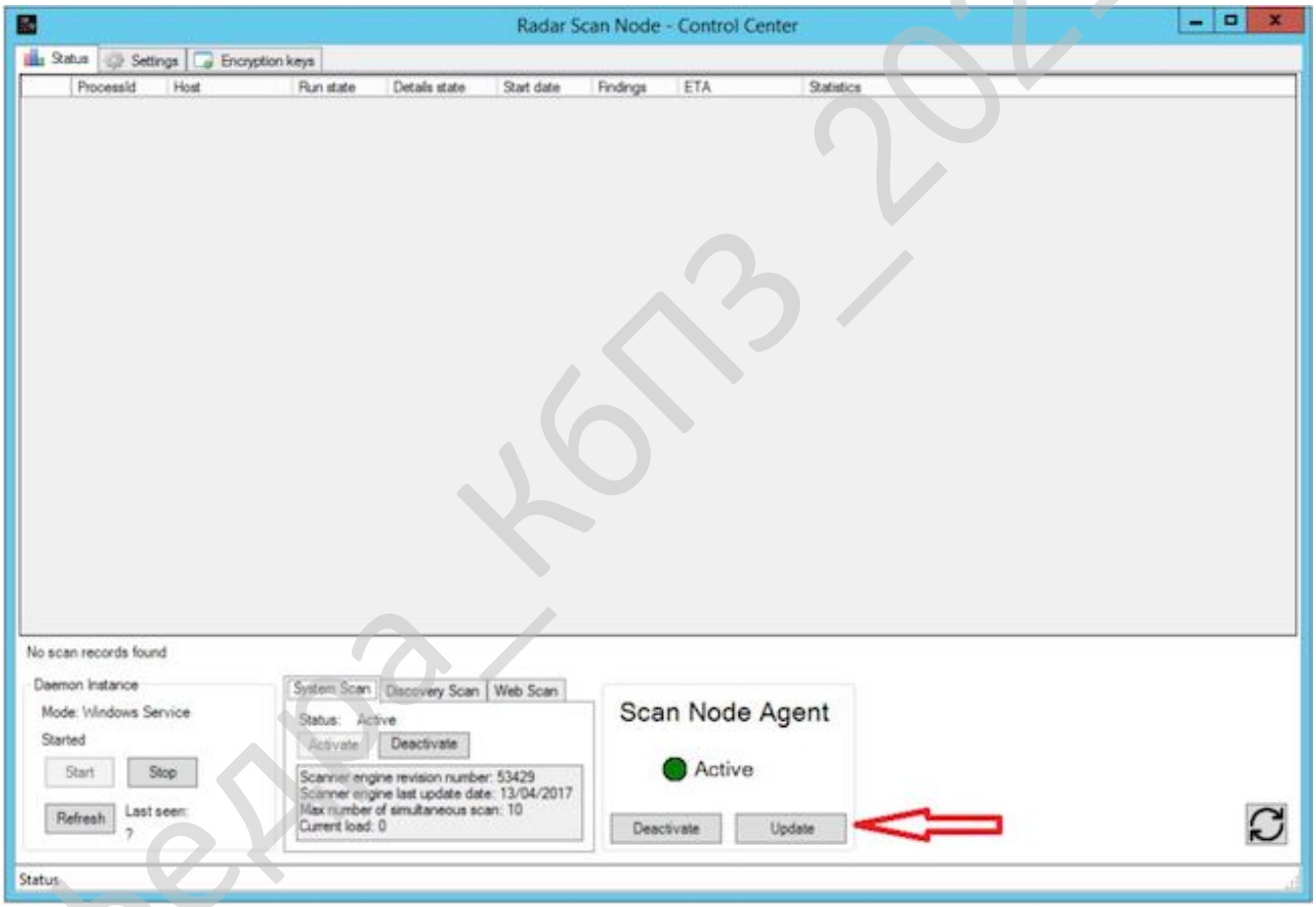

Рисунок 2.4 – Вікно центру керування в F-Secure Radar

F-Secure Radar являє собою не тільки сканер уразливостей, але й платформу для керування уразливостями й активами. Він має можливості по виявленню IT-активів, їх інвентаризації й ідентифікації. Також доступні

інструменти для підготовки звітів про ризики й про дотримання вимог – наприклад, відповідності приписанням PCI і GDPR.

Крім цього Radar пропонує наступні можливості:

– Централізоване керування уразливостями, оповіщеннями по безпеці й розслідуванням інцидентів.

– Виявлення фактів незаконного використання товарного знака й спроб шахрайства під фірмовим найменуванням від третіх осіб.

– Запобігання атак за допомогою виявлення неправильного настроювання програмного забезпечення в службах, операційних системах і мережних пристроях.

– Інвентаризація застосунків на вузлах мережі.

– Відстеження всіх змін в IT-інфраструктурі.

GFI Languard

Мережний сканер безпеки GFI Languard від GFI Software дозволяє сканувати мережа підприємства для виявлення, виявлення й усунення уразливостей.

У тому числі проводиться сканування портів. Кілька готових профілів дозволяють перевірити всі порти або тільки ті, які звичайно використовуються небажаними й шкідливими програмами. GFI Languard забезпечує можливість сканувати кілька вузлів одночасно, заощаджуючи тим самим час, і проводити аналіз того, яке ПЗ які порти використовує.

GFI Languard може також виконувати перевірку наявності останніх відновлень і патчів на вузлах мережі. Сканер аналізує не тільки саму ОС, але й популярне ПЗ, уразливості якого звичайно використовуються для злому: Adobe Acrobat / Reader, Flash Player, Skype, Outlook, браузери, мессенджери і т.д.

Languard проводить кожне сканування після відновлення даних про уразливості. Джерелами інформації про погрози є самі вендори ПЗ, а списки, що також добре зарекомендували себе, SANS і OVAL.

Забезпечується підтримка всіх популярних ОС для робочих станцій і серверів (Windows, macos, дистрибутиви Linux і Unix), а також iOS і Android для смартфонів. GFI Languard дозволяє встановити окремі пари « логін-пароль» для доступу, а також файл ключа для зв'язку по SSH.

Languard здатний створювати звіти відповідно до вимог PCI DSS, HIPAA, SOX, GLB / GLBA і PSN Coco. Також можна додавати власні шаблони в планувальник.

Nexpose Vulnerability Scanner

Nexpose Vulnerability Scanner від Rapid7 пропонується у виконанні «on-premise» для локальної установки на території замовника. Хмарна версія доступна в редакції Rapid7 Insightvm, що пропонує розширені функціональні можливості за ту ж вартість.

Даний продукт обчислює показник реального ризику по шкалі від 1 до 1000, де оцінка CVSS є лише однієї зі складових, що дає більш корисну інформацію. При цьому враховуються такі параметри, як значимість вузла, строк уразливості, наявність загальнодоступних експлойтів / готових шкідливих об'єктів і ін. Nexpose надає контекстну бізнес-аналітикові, акцентуючи увагу на самих значимих ризиках для бізнесу, за допомогою автоматизованої класифікації активів і ризиків.

Відзначимо наступні можливості Nexpose:

– Застосування різних стратегій з адаптацією під різні завдання й інфраструктуру.

– Інтеграція з DHCP Service, VMWare і AWS / Azure для розуміння динаміки середовища й виявлення нових пристроїв і уразливостей.

– Доступ до даних з Project Sonar для виявлення компонентів інфраструктури, найбільш підданих розповсюдженим уразливостям.

– Створення динамічних груп активів з більш ніж 50 фільтрами.

– Виявлення більш ніж 75 000 уразливостей.

– Широкий вибір готових шаблонів звітів і можливість створення власних шаблонів з необхідними параметрами (включаючи таблиці, діаграми, порівняння) з більшими можливостями імпорту.

Функція Adaptive Security дозволяє автоматично виявляти й оцінювати нові пристрої й уразливості в режимі реального часу. У комбінації з підключеннями до VMWare і AWS, а також інтеграцією з дослідницьким проектом Sonar сканер Nexpose забезпечує постійний моніторинг мінливого середовища в IT-інфраструктурі.

Також Nexpose дозволяє проводити аудита політик і конфігурацій. Сканер аналізує політики на відповідність вимогам і рекомендаціям популярних стандартів. Звіти про уразливості містять покрокові інструкції про те, які дії слід почати, щоб усунути проломи й підвищити рівень безпеки.

Nexpose пропонує більші можливості по інтеграції, у тому числі двосторонньої, з інструментом для проведення пентестів Metasploit, а також з іншими технологіями й системами безпеки, у тому числі за допомогою Openapi: SIEM, міжмережевими екранами, системами керування патчами або системами сервісних запитів (тікетів), комплексами технічної підтримки (service desk) і т.д.

Крім цього Nexpose дозволяє вирішувати завдання по відповідності вимогам різних стандартів, зокрема – PCI DSS, NERC CIP, FISMA (USGCB / FDCC), HIPAA / HITECH, Top 20 CSC, DISA STIGS, а також стандартів CIS по оцінці ризиків.

Qualys Vulnerability Management

Особливістю продукту Qualys є поділ процесів збору й обробки інформації:

– інформація з уразливостей збирається або за допомогою сканера (безагентно), який може бути виконаний у віртуальному або «залізному» форматі, або за допомогою хмарних агентів;

– обробка й кореляція інформації, що надходить із сенсорів, відбувається в хмарі Qualys. У такий спосіб не навантажується локальна інфраструктура й

суттєво спрощується робота з даними, які відображаються в інтерфейсі в консолідованому й нормалізованому виді.

Окремо варто відзначити, що великим замовникам доступна можливість розгортання хмари Qualys у локальній мережі.

Хмарні агенти Qualys забезпечують безперервний збір даних і їх передачу на хмарну платформу, яка є свого роду аналітичним центром, де інформація корелюється й розподіляється по пріоритетах для забезпечення видимості всього того, що відбувається на кінцевих точках і в мережі компанії, у режимі реального часу.

Також заслуговують згадування наступні можливості платформи:

- Інвентаризація всієї IT-інфраструктури, включаючи виявлення й класифікацію активів, перевірку типу ліцензій ПЗ (комерційні / з відкритим вихідним кодом) і циклу підтримки (End of Life / End of Support).
- Пошук уразливостей і критичних помилок у конфігураціях відповідно до критеріїв CIS, VMWare, Microsoft, Qualys, NIST, DISA по активах з можливістю виправлення конфігурацій.
- Автоматична пріоритизація погроз на основі інформації про реальні атаки зловмисників, а також даних кіберрозвідка (Threat Intelligence).
- Інвентаризація цифрових сертифікатів SSL / TLS.
- Дотримання вимог PCI DSS.
- Захист кінцевих вузлів від атак з можливістю розслідування інцидентів і пошуку слідів компрометації (EDR).
- Моніторинг цілісності файлів.
- Перевірка на наявність уразливостей протягом усього циклу розробки.
- Захист веб-застосунків із застосуванням віртуальних патчів.
- Сканування контейнерів на всіх етапах.
- Пасивне сканування мережного трафіка й виявлення аномалій.
- Власний патч-менеджмент для ОС і застосунків.
- Сканування хмарних аккаунтів і середовищ Azure, AWS, GCP.

– Сканування зовнішнього периметра з дата-центру Qualys.

– Аналіз безпеки мереж АСУ ТП.

– Відкритий документований API для інтеграції з CMDB, NAC, WAF, SIEM, Service Desk, Skybox, R-Vision і іншими системами.

Tenable.io

На відміну від Nessus Professional інший продукт того ж вендора – Tenable.io – має тільки хмарне виконання. У частині функціональних можливостей є подібність із Nessus. Проте Tenable.io цілком самодостатній, особливо в частині керування всіма даними про активи й уразливостях. Продукт пропонує наступні можливості:

– Одержання оперативних даних про IT-активах за рахунок сканування, використання агентів, пасивного моніторингу, хмарних коннекторів і інтеграції з базами даних керування конфігураціями (CMDB).

– Комбінування відомостей про уразливості з кіберрозвідкою (Threat Intelligence) і дослідженням даних (Data Science) для більш простої оцінки ризиків і розуміння того, які уразливості слід виправляти в першу чергу.

– Угруповання виявлених уразливостей по пріоритетах.

– Широкі можливості по роботі зі звітами й архівними даними.

– Високошвидкісне сканування з мінімальною кількістю неправильних спрацьовувань.

– База Tenable.io містить дані про порядку 60 000 уразливостей.

У ліцензію Tenable.io також входить безлімітна кількість сканерів Nessus Professional.

Образно говорячи, Tenable.io являє собою різноплановий набір сенсорів, який автоматично збирає й аналізує дані про безпеку й уразливості, тим самим показуючи повну інформацію про атаку для будь-якого активу на будь-якій обчислювальній платформі.

Також у компанії Tenable є кілька суміжних продуктів, побудованих на платформі Tenable.io, наприклад спеціалізований інструмент для проведення

сканування відповідно до вимог PCI DSS, сканер для виявлення уразливостей у контейнерах, засіб перевірки веб-застосунків і сервісів.

Tripwire IP360

Tripwire IP360 позиціонується як продукт корпоративного рівня («enterprise») для керування уразливостями й ризиками. Сканер побудований на модульній архітектурі, що дозволяє легко масштабувати дану систему. Можлива поставка версій для локального (on-premise), хмарного або гібридного розгортання. Також IP360 дозволяє розміщати на вузлах агенти для більш ефективного моніторингу мережі й виявлення уразливостей.

Крім цього сканер має наступні можливості:

– Проведення інвентаризації в мережі для виявлення й ідентифікації всіх IT-активів, включаючи локальні, хмарні й контейнерні.

– Відстеження змін в активах.

– Ранжирування уразливостей на основі їх ступеня впливу й віку.

– Керування часом запуску сканування через планувальник.

– Аудит безпеки в контейнерних середовищах з підтримкою інструментальних засобів Devops.

– Централізоване керування через веб-інтерфейс для адміністрування, настроювання, одержання звітів, розпорядження завданнями.

IP360 поставляється у вигляді програмного або апаратного забезпечення, причому для збільшення продуктивності й надійності роботи системи можлива її установка в кластері. Сканер також доступний на торговельних майданчиках AWS і Azure.

Tripwire IP360 надає можливості інтеграції зі сторонніми системами, такими як комплекси технічної підтримки (helpdesk / service desk), застосунку по керуванню активами, SIEM, IDS / IPS і інші засоби забезпечення інформаційної безпеки, за допомогою Api-модуля.

Vulnerability Control

Система Skybox Vulnerability Control дозволяє автоматизувати різні процеси керування уразливостями на єдиній платформі. Продукт здатний збирати дані з різних систем з урахуванням особливостей мережі й виявляти найнебезпечніші погрози.

Vulnerability Control має наступні можливості:

Аналіз даних із систем інвентаризації й патч-менеджменту, а також ряду інших дозволяє збирати відомості в недоступні для сканування областях.

Одержання відомостей про уразливості з різних джерел, таких як мережні пристрої, IPS / IDS, публічні й приватні хмари (Amazon Web Services, Microsoft Azure, Cisco ACI і VMWare NSX), засобу безпеки кінцевих точок (EDR), комплекси патч-менеджменту, системи керування конфігурацією баз даних (CMDB), сканери уразливостей і застосунків, веб-сканери.

Проведення імітації атак на динамічній моделі мережі для виявлення уразливостей, які доступні для експлуатації на критично важливих активах. Також це дозволяє зрозуміти ефективність поточних настроювань компонентів мережі.

Функція моделювання мережі дозволяє визначати альтернативні методи боротьби з погрозами, включаючи зміни правил доступу або сигнатур систем запобігання вторгнень (IPS).

Використання скорінговою моделі, що враховує різні критерії (уразливості, активи, бізнес-сегменти), у тому числі унікальні параметри й атрибути, застосовувані в конкретній інфраструктурі.

Виявлення черговості установки патчів і інших заходів, що компенсують, по усуненню уразливостей на основі рівня погроз і ранжирування їх по ступеню небезпеки.

Vulnerability Control реалізує гнучкий підхід до аналізу уразливостей, що дозволяє краще зрозуміти можливі наслідки. Крім основних даних збираються різноманітні додаткові відомості: настроювання й умови використання ОС і

інших застосунків, матеріали з національної бази даних уразливостей США (NVD), CVSS, бюлетені постачальників, підсумки аналізу проломів і ризиків з інших джерел (IBM X-Force, інші сканери уразливостей і т.д.), історія змін уразливості залежно від ступеня серйозності, експлуатації, доступних патчів, відновлень і т.д.

Skybox Vulnerability Control може інтегруватися з більш ніж 140 різними IT- і ІБ-системами, такими як інструменти пошуку уразливостей, мережні пристрої, ОС, засобу мережного захисту, SIEM і інші.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

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

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для сканування мережі на наявність вразливостей.

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

Для пошуку уразливостей вручну потрібні особливі знання, багатий досвід і рідке чуття. Але як бути новачкам? Як набратися досвіду, якщо не знаєш, із чого починати? На допомогу приходять автоматичні сканери уразливостей. У цій роботі ми подивимося, які вони бувають і як ними користуватися.

Айтишники, як відомо, прагнуть усе автоматизувати, і хакери в цьому не відстають. Існують автоматичні сканери уразливостей. Пошук уразливостей з їхнім використанням зводиться до того, щоб віддати сканеру адреса цілі й нажати велику кнопку Start, ну або Enter, якщо ти аматор термінала.

При цьому зрозуміло, що сканер знайде тільки типові уразливості й, щоб піти далі, потрібно вміти не тільки натискати на кнопку. Але чому б не заощадити небагато сил? У багатьох випадках це цілком виправданно.

Категорії й методи

Універсальних інструментів не існує, і сканери уразливостей не стали виключенням із цього правила. Вони звичайно націлені на уразливості якогось певного роду. У цій роботі ми розглянемо наступні види сканерів.

– WVS (Web Vulnerability Scanner) – сканери вразливостей-веб-уразливостей. У мене це сама численна категорія. Сюди входять як загальновідомі OWASP ZAP і sqlmap, так і менш відомі, але не менш корисні, начебто Vega.

– Аналізатори мобільних застосунків. Отут дуже мало гідних продуктів, і ми зупинимося на самі яскраві з них.

– Напівуніверсальні сканери для локальної мережі підприємства або будинку. Це вже не просто сканери, а цілі комбайни для аналізу й обліку устаткування в мережі. Багато хто з них заодно шукають уразливості.

– Усякі вузькоспеціалізовані сканери типу аналізу вихідного коду, Git/ SVN-репозиторіїв і інших складних для ручної обробки масивів даних.

Сканери бувають із вільною ліцензією й комерційні. Якщо з опенсорсом усі зрозуміло, то для використання комерційних прийдеться викласти досить пристойну суму. На жаль, ні редакція «Хакера», ні автор не настільки багаті, щоб купувати їх для огляду. Тому для всіх комерційних сканерів була використана офіційна пробна версія, якщо не застережене інше.

Саме тестування теж буває різним: Black Box або White Box. При першому типі пентестер або його інструмент повинні працювати із сервісом через ті ж інтерфейси, через які з ним взаємодіють користувачі. Наприклад, якщо для тестування методом Black Box тобі даний сайт, то ти можеш перевіряти його тільки як відвідувач, без якого-небудь спеціального доступу до вихідного коду або привілейованих аккаунтам. Якщо цей додаток, то мається на увазі, що в тебе немає доступу до исходникам: колупай сам, якщо зможеш. Загалом, Black Box виходить, що в тебе немає нічого, чого б не було в усіх.

При тестуванні методом White Box пентестер (або хакер) має доступ до всього потруху цільового об'єкта. Якщо це сайт – у тебе є його код. Якщо це сервер – у тебе є доступ до його нутрощів начебто версії ОС і встановленого софту або до деяких файлів. У цьому випадку можливості куди ширше й ти можеш знайти проблему, яку здатний експлуатувати тільки просунутий зловмисник.

3.2 Розробка структурної схеми

У даному розділі приведемо результати опитування серед фахівців з ІБ. Ціль – довідатися, як організовано керування уразливостями в компаніях в Україні, які проблеми хвилюють фахівців з ІБ і з якими обмеженнями засобів аналізу захищеності вони зустрічаються. Ця інформація потрібна нам як виробникові засобів класу vulnerability management, щоб скласти рекомендації

для фахівців з ІБ – на що звернути увагу при побудові ефективного процесу керування уразливостями.

Ключові результати:

– У половини опитаних найбільше часу йде на те, щоб переконати ІТ-відділ у необхідності встановити відновлення; 11% респондентів розповіли, що їм доводиться обґрунтовувати усунення кожної уразливості. Такий підхід чреватий тим, що при розростанні ІТ-інфраструктури у фахівців просто не залишиться часу на обробку уразливостей.

– 9% фахівців направляють звіт про виявлені уразливості прямо ІТ-фахівцям без попередньої фільтрації й пріоритизації завдань на усунення. У цьому випадку велика небезпека, що, переробляючи великий список уразливостей, ІТ-фахівці не дійдуть до обробки дійсно небезпечних уразливостей. Респондентів не перевіряють, чи усунув ІТ-відділ виявлення уразливостей. Це говорить про те, що у компаній відсутні важливі етап процесу vulnerability management – контроль рівня захищеності.

– Опитування показало, що не все знайомі з патч-менеджментом. Кожний десятий опитаний відповів, що в його компанії критично небезпечні уразливості на важливих активах не усуваються більш напівроки.тверть респондентов не используют в своих компаниях специализированное ПЗ для виявления уразливостей. Получается, специалистам по ІБ необходимо проверять каждий компонент інформаційной системи вручную, что долго и малоэффективно.

Профіль респондентів

Галузь

В опитуванні брали участь в основному фахівці з ІБ з держсектора, кредитно-фінансових организацій, промислових і ІТ-компаний.

Розмір компанії

Серед опитуваних минулого представники малого бізнесу (до 250 працівників) – 24%, середнього (від 250 до 3000 співробітників) – 38%, великого (від 3000 співробітників) – 38%.

Як організований процес керування уразливостями

Працезатрати фахівців з ІБ

Учасники опитування відзначили: найбільше часу в них іде на те, щоб переконати ІТ-відділ у необхідності закрити уразливості (48%), і на аналіз результатів сканування (43%). Також до працемістких завдань 31% фахівців віднесли перевірку усунення уразливостей. Зазначені труднощі характерні як для більших, так і для маленьких компаний. Однак прослідковується тенденція: чому більше компанія, тем сутужніше фахівцям з ІБ домовитися з ІТ-відділом. Якщо для представників малого бізнесу найбільше працеемний процес – аналіз результатів сканування (50%), то для представників середнього й великого бізнесу це узгодження установки патчів (55% і 56% відповідно).

Сканування інфраструктури

Для підтримки захищеності інфраструктури важливо мати актуальні дані про состав мережі (устаткуванні, ПЗ, доступних сервісах) і про виявлені в ней уразливостях.Але, як виявилося, не завжди можливо щодня проводити повне сканування всей ІТ-інфраструктури.Серед причин учасники опитування назвали наступні: 33% респондентів скаржаться на зростаюче навантаження на мережу й на активи при скануванні; 26%, що відповіли невідомий звістка список активів у їхній інфраструктурі, що ставить під сумнів вірогідність оцінки захищеності; а 29% фахівців не можуть регулярно сканувати ІТ-інфраструктуру через необхідність постійно погоджувати ці роботи з ІТ-відділом.

Представники держкомпаній (36%), фінансової галузі (32%) і ІТ-сектору (42%) назвали перешкодою високе навантаження на мережу, представників промисловості (40%) головним чином хвилює необхідність погоджувати кожне сканування с ІТ-відділом. Респонденти з фінансової сфери (32%) відзначили, що в них немає проблем з регулярним скануванням інфраструктури.

Контролювати повноту даних, що збираються, про інфраструктуру можна не тільки за рахунок повного сканування. Добре, якщо це можна зробити, серед іншого, за рахунок вибіркового сканування ІТ-активів, а також за рахунок

імпорту даних про активи із зовнішніх каталогів або з інших засобів захисту (даних аналізу подій в Siem-системах і аналізу трафіка в Nta-системах).

Фільтрація уразливостей

За підсумками сканування фахівці, як правило, одержують великий список уразливостей. На цьому етапі важливо визначити, які уразливості необхідно усунути в першу чергу.

Менше половини опитаних фахівців з ІБ налагодили процес фільтрації виявлених уразливостей по ступеню важливості активів, на яких вони були виявлені.Це можливо, якщо при інвентаризації вони також класифікували активи по ступеню їх впливу на працездатність важливих для бізнесу сервісів. Такий похід дозволяє фокусуватися на усуненні дійсно небезпечних для бізнесу уразливостей. Фільтрацію по ступеню важливості активів підтримують 63% опитаних з держкомпаний і 42% з IT-сфери, по інших галузях відсоток нижче.

Дев'ять відсотків фахівців з ІБ направляють звіт про виявлені уразливості прямо IT- фахівцям без попередньої фільтрації й пріоритизації завдань на усунення. Такий підхід несе в собі багато ризиків: IT-відділ не буде справлятися з усуненням усіх виявлених уразливостей, тому є ризик пропустити найнебезпечніші наслідки, це може привести до того, що без контролю з боку фахівців з ІБ співробітники IT-відділу зовсім не буде приділяти увагу питання безпеки.

Оцінка уразливостей

Щоб коректно оцінити рівень небезпеки уразливості, фахівцеві з ІБ недостатньо орієнтуватися на свій досвід, на тип уразливості й на оцінку по CVSS.Потрібно ще співвідносити шкалу оцінок зі своєю інфраструктурою і оцінювати можливість експлуатації данної уразливості в конкретній системі. Більше половини респондентів для оцінки уразливостей користуються системою CVSS і шукають інформацію про Нові уразливості на тематичних ресурсах. Самостійна оцінка небезпеки уразливостей часто вимагає додаткових ресурсів

для управління уразливостей: 43% респондентів підтвердили, що найбільше часу йде саме на аналіз результатів сканування.

Оцінці вендора довіряють 34% опитаних. Однак незважаючи на те, що в таких оцінках знайдені уразливості ранжирувані по ступеню небезпеки, вони не дають розуміння, скільки уразливостей у мережі організації потрібно усунути, щоб забезпечити достатній рівень захищеності. Краще, якщо вендор на основі своєї експертизи зможе надати обмежений список самих актуальних і небезпечних уразливостей, які досить буде усунути, щоб утруднити зловмисникові проникнення в мережу.

Договоренности с IT-отделом

За усунення уразливостей (відновлення версій ПЗ, установку патчів)і застосування компенсаційних заходів (вимикання сервісів, закриття портів на міжмережевих екранах) відповідає, як правило, IT-підрозділ. Одинадцять відсотків респондентів зустрічаються з тим, що їм доводиться обґрунтовувати усунення кожної уразливості для IT-відділу. При цьому 75% з тих, хто так відповів, – представники великого бізнесу (від 1000 співробітників). У компаніях такого розміру велика кількість мережних вузлів, а виходить, і уразливостей. Тому бюрократичний процес по закладу й обґрунтуванню тікета на кожну уразливість, швидше за все, з'їдає багато часу в співробітників відділів IT і ІБ.

Також 11% опитаних не перевіряють, чи були усунуті виявлені уразливості. Проте 56% з них проголосували, що небезпечні уразливості на критично важливих активах усуваються в їхній компанії швидше чому за тиждень. Ми щиро сподіваємося, що зазначені строки відповідають дійсності, але все-таки радимо перевіряти усунення уразливостей самостійно. Більше половини респондентів перевіряють устранені уразливостей при черговому скануванні системи. Тут важливо, щоб скануючі системи оперувати IP-адресами, а IT-активами. IP- адреси можуть змінюватися від сканування до сканування, тому співвідносити результати двох звітів буде непросто. Якщо доводиться ставити список IP-адрес, то звіт про сканування можуть бути неповними (без обліку

нових вузлів, що з'явилися між двома скануваннями). Більш ефективно, коли засіб керування уразливостями підтримує виявлення й ідентифікацію активів, у цьому випадку ви одержите повні відомості про вузли й системах в інфраструктурі компанії.

Найчастіше в диференціальних звітах, застосовуваних для порівняння результатів декількох сканувань, не враховуються вжиті компенсаційні заходи. Кожний десятий з опитаних (11%) відзначив, що йому траплялося забувати про вжиті компенсаційні заходи через те, що уразливості «маячили» у звіті. Проводить ручну перевірку усунення уразливостей ефективно, якщо їх кількість невелика. Але при розширенні інфраструктурі і, відповідно, збільшення кількості виявляв уразливостей фахівця з ІБ вже потрібно автоматизувати процес їх обробки. Вирішувати це завдання найкраще так: для небезпечних уразливостей (в першу чергу – критично небезпечним уразливостей на важливих активів) необхідно створювати тікети для IT-відділу і контролювати їх виконані в екстрених порядках. Для решти маси уразливостей необхідно узгодити з IT-відділом і контролювати регулярне обновлені ОС і ПЗ. Подобний процес підтримують 25% респондентів.

Швидкість усунення уразливостей

При роботі з уразливостями також важлива швидкість їх усунення, особливо якщо це стосується значимих активів компанії. Тільки 39% опитаних вдається усунути критично небезпечні уразливості на пріоритетні для компанії активах протягом перших двох днів; 9% не закривають такі уразливості протягом напівроку.

Розглянемо залежність швидкості усунення уразливостей від розмірів компанії. У маленьких компаніях (до 250 співробітників) критично небезпечні уразливості на важливих активах устигають усувати за два дні 47% респондентів і тільки 6% – за півроку. Критично небезпечні уразливості на всіх активах в основному встигають усувати за тиждень (44%), а інші уразливості – за місяць (42%).

У середніх по розміру компаніях (від 250 до 3000 співробітників) важливі активи встигають захистити за два дні 45% опитаних – і вже 9% можуть не закривати уразливості по півроку. Критично небезпечні уразливості на звичайних активах усувають за місяць (40%), а інші уразливості – за півроку (38%).

У великих компаніях (від 3000 співробітників) критично небезпечні уразливості на важливих активах усуваються за тиждень в 40% опитаних і тільки в 26% за день-два. Уразливості на звичайних активах усуваються в основному (37%) за місяць, а менш небезпечні уразливості патчать за півроку (44%) або за місяць (38%).

Досвід показує, що компанії затягують із відновленнями, у той час як зловмисники діють швидко й адаптують найновіші експлойти для своїх атак іноді протягом доби.

Інструментарій фахівця з ІБ

На допомогу безпечнику в боротьбі з уразливостями приходять засобу аналізу захищеності – спеціальні системи, які в автоматизованому режимі виявляють відкриті мережні порти й доступні служби, уразливості в ПЗ, а також недоліки конфігурації устаткування, серверів і засобів захисту. Ними користуються 63% опитаних.

Багато фахівців (26%), відповідаючи на запитання про продукти, які вони використовують для керування уразливостями, назвали некомерційні утиліти для сканування мережі. У подібних інструментів є свої переваги, але вважаємо важливим підкреслити: вони дозволяють вирішити завдання сканування портів і служб, але не дають інформації про состав і стан активів, про уразливості ПЗ, застосунків, баз даних, а також не виявляють помилки конфігурацій. Для повноцінного процесу керування уразливостями їх буде недостатньо. Чотири відсотки респондентів указали, що використовують відразу кілька систем аналізу захищеності.

Чверть респондентів не використовують спеціалізоване ПЗ для виявлення уразливостей. Серед них в основному представники малого (35%) і середнього

(38%) бізнесу. Виходить, у таких компаніях фахівцям з ІБ доводиться перевіряти кожний компонент інформаційної системи вручну, що довго й малоефективно. Також ми попросили учасників опитування виділити п’ять самих популярних задач, які вони виконують за допомогою систем аналізу захищеності. Серед них виявились:

- контроль усунення уразливостей (58%);
- оцінка загального рівня захищеності (54%);
- сканування вузлів мережі в режимі чорного ящика (51%);
- сканування вузлів мережі в режимі білого ящика (49%);
- виділення особливо небезпечних уразливостей (48%).

Усі запропоновані нами варіанти завдань виявилися затребувані: як мінімум кожний третій виконує їх за допомогою систем аналізу захищеності. Самим непопулярним варіантом виявилося завдання політик на усунення уразливостей: його виконують тільки 15% респондентів.

Найбільше було цікаво довідатися, які поліпшення в системах керування уразливостями необхідні користувачеві. Більш третини фахівців з ІБ (36%) прагнуть мати можливість ураховувати заходи, що компенсують. Приблизно стільки ж опитаних (34%) скаржаться на довгий час сканування. На думку 32% респондентів, у використовуваних продуктах для vulnerability management не вистачає можливості виявляти активи, не зазначені при скануванні, але існуючі в мережі. Двадцять вісім відсотків фахівців відзначили, що їм допоможе функція перевірки усунення уразливостей.

Інші варіанти теж виявилися затребувані.

Компанії до 250 співробітників стомлюють довгий час сканування (36%) і складне налаштування (30%).Компанії від 250 до 3000 співробітників прагнуть ураховувати заходи, що компенсують (40%), а також перевіряти усунення уразливостей (34%).Великі компанії від 3000 співробітників цікавить зіставлення результатів сканування (33%) і також облік заходів, що компенсують (35%).

Крім того, респонденти скаржилися на погану пріоритизацію уразливостей і необхідність виявляти деякі дані вручну, наприклад дату останнього успішного сканування активу.

Екстрена перевірка

Вибудований процес керування уразливостями повинен бути ефективним як у штатному режимі, так і при проведенні екстреної перевірки. Ми запропонували респондентам представити, що вони будуть робити в першу чергу, якщо довідаються про нову серйозну уразливість в ПЗ. У половині опитаних знадобиться провести додаткове сканування, щоб довідатися про існування в мережі вразливого ПЗ, 19% зможуть відразу вжити заходів по захисту компанії.

Класичні сканери вирішують, є чи на вузлі уразливість, безпосередньо в момент сканування. Відповідно, коли з'явиться нова уразливість, необхідно заново просканувати усі активи. Враховуючи складний процес узгодження повноцінного сканування в багатьох компаніях, оперативно довідатися, чи небезпечна нова уразливість для інфраструктури, буде важко. Ефективніше, якщо система керування уразливостями зберігає інформацію про просканованих активах і може вираховувати застосовність нової уразливості до мережі автоматично на підставі минулого сканування.

За підсумками опитування ми дійшли висновку, що одна із самих більших проблем для фахівців з ІБ – досягтися взаєморозуміння з ІТ-відділом. У процесі керування уразливостями найбільше часу йде на розбір результатів сканування й спроби переконати ІТ-фахівців у необхідності встановити відновлення (48%).

Половина з опитаних зустрічалася з тим, що ІТ-відділ уважав знайденні в інфраструктурі уразливості неексплуатабельними, тому не бачив змісту витрачати час на їхнє усунення.

Також часто (більше 30% голосів) ІТ-фахівці прибігають до наступних аргументів:

- На патчинг немає ресурсів;

– Якщо уразливий сервер скомпрометують, то це не страшно, оскільки він не дуже важний;

– Ми все обновили, це сканер спрацьовує неправильно.

Вічні суперечки між відділами IT і ІБ можуть позначитися на безпеці компанії. Не можна допустити, щоб процес керування уразливостями в компанії залежав від здатності співробітників домовлятися між собою.

Щоб взаємодія відділів IT і ІБ було продуктивнию, необхідно вводити правила обробки уразливостей. Ми рекомендуємо для уразливостей середнього й низького рівня ризику поза критично важливими активами фіксувати певний строк усунення. Необхідний час компанія визначає сама: щотижня, раз на місяць або у два – але строк потрібно встановити й він повинен дотримуватися. Порушення даної політики буде сигналізувати про те, що в процесі щось пішло не так.

IT-підрозділ буде регулярно встановлювати патчі або відновлення ОС і ПЗ на основі заданих політик, не чекаючи інформації про уразливості від служби ІБ. Тоді фахівцям з ІБ залишиться лише контролювати відсутність уразливостей через певну кількість днів після публікації інформації про них. При таких домовленостях у фахівця з ІБ з'явиться час на ручну обробку дійсно небезпечних уразливостей на важливих активах.

Наше опитування показало, що крім самого виявлення уразливостей у процесі vulnerability management багато чого усе ще викликає труднощів у фахівців з ІБ. Тисячі клієнтів теж зустрічалися із проблемами, описаними в роботі, тому застосунали глянути по-новому на керування уразливостями. У даній роботі розробляю продукт, який буде допомагати вишиковувати процеси в компанії, охоплювати всю інфраструктуру, включати регламенти по скануванню й усуненню уразливостей і контролювати загальний рівень захищеності компанії. За результатам опитування в даній роботі сформована структурна схема системи.

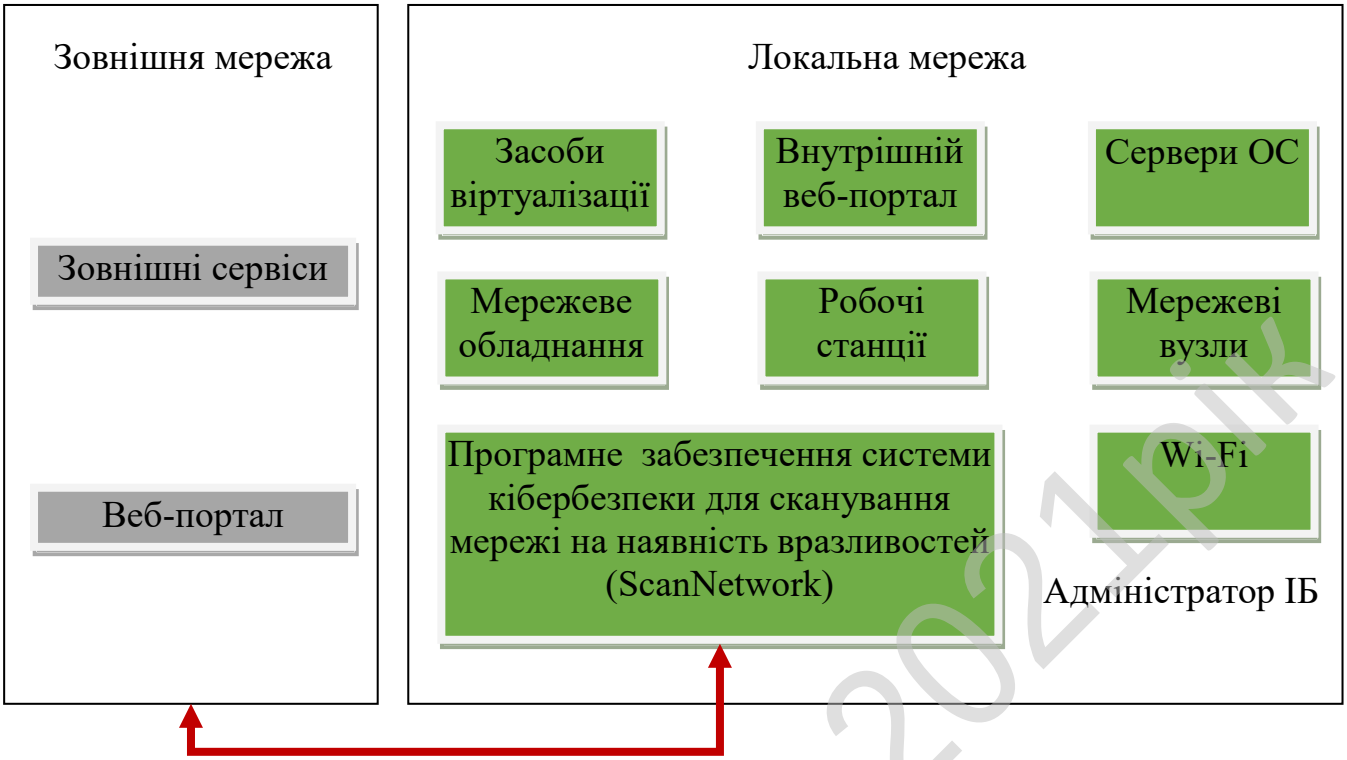

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Розроблений у даній роботі сканер безпеки ScanNetwork має багатий функціонал і покликано вирішувати широке коло завдань у повсякденному циклі керування безпекою IT-інфраструктури підприємства. Крім мережних і системних перевірок на уразливості, його функціональні можливості посилені засобами контролю відповідності, механізмами оцінки захищеності СУБД і систем віртуалізації, засобами патч-менеджменту, контролю цілісності й поруч інших не менш важливих функцій. ScanNetwork створює "моментальний знімок" стану безпеки системи, дозволяє виявляти помилки адміністраторів і виконувати аудит системи для оцінки відповідності політикам і стандартам безпеки.

Функціональні можливості:

– Аудит уразливостей.

– Аудит захищеності СУБД.

– Інвентаризація мережі.

– Перевірка складності паролів.

– Керування відновленнями.

– Аудит безпеки АСУ ТП.

– Аудит серверів застосунків.

– Аудит у режимі «Пентест».

– Аудит мережного устаткування.

– Фіксація й контроль цілісності.

– Створення інтегральних і диференціальних звітів по кожному напрямкові аудита.

– Контроль конфігурацій і оцінка відповідності політикам і стандартам безпеки.

– Детальний аудит платформ віртуалізації (HyperV, VMWare).

Рисунок 3.2 – Функціональна схема системи

Розглянемо більш докладно Функціональні блоки.

Аудит уразливостей

ScanNetwork виконує централізоване й/або локальне сканування вузлів мережі на наявність уразливостей операційних систем, загальносистемного й прикладного ПЗ. Аудит уразливостей може здійснюватися в ручному або автоматичному режимі по сформованих на консолі керування завданням. Сканування виконується або з використанням постійно працюючих керованих служб-агентів, або на основі безагентної технології. Перевірки побудовані на зіставленні стану параметрів системи сигнатурам уразливостей, що втримуються у відкритому репозиторії OVALdb і описаних у форматі SCAP. На сьогоднішній день у базі даних ScanNetwork є описи уразливостей більш ніж для 1000 різних програм

Контроль конфігурацій і оцінка відповідності політикам і стандартам безпеки

Найбільш простий і доступний спосіб зламати "оборону" - це знайти системи, що містять програми, інстальовані з настроюваннями за замовчуванням. Як правило, такі конфігурації надають максимальну функціональність програм, але не гарантують їхню безпеку. Тому здійснення настроювань безпеки й контроль їх незмінності є базової складової підтримки безпеки системи.

ScanNetwork дозволяє автоматизувати процес контролю параметрів безпеки й здійснювати оцінку відповідності інформаційних систем, її окремих компонентів або вузлів стандартам, політикам безпеки, рекомендаціям вендорів і іншим "визнаним практикам" (best practices). Контроль конфігурацій параметрів безпеки може здійснюватися як на рівні комп'ютера, так і на рівні користувача. Користувач може створювати власні конфігурації й включати їх у плани перевірок. ScanNetwork містить ряд готових конфігурацій, розроблених на основі вимог міжнародних стандартів і рекомендацій. Крім оповіщень про проблеми параметрів безпеки у звітах наведені рекомендація з їхнього настроювання. Підтримка стандартизованого формату SCAP дозволяє користувачам

завантажувати сторонні конфігурації, наприклад з Microsoft SCM. Також ми можемо розробити індивідуальні конфігурації (compliance) під ваші вимоги.

Керування відновленнями

Функція "Аудит відновлень" дозволяє виявити невстановлені відновлення безпеки на вузлах мережі й сформувати необхідні посилання для завантаження відсутніх відновлень. Об'єктами аудита є всі актуальні клієнтські й серверні операційні системи Windows, починаючи з Windows XP і Windows Server 2003, Unix\Linux системи, використовувані на об'єктах інформатизації, а також широкий перелік іншого загальносистемного й прикладного ПЗ. Результат аудита відновлень містить: найменування відновлення, відомості про ризики пов'язаних з відсутністю відсутнього відновлення на вузлі, посилання на вендору відновлення, що заявив про вихід, посилання на репозиторій (базу), де зберігається доступне для завантаження відновлення й ін. інформація, що деталізує.

Завдання "Аудит відновлень" може виконуватися як з використанням на вузлах агентів, так і без них. Сканер ScanNetwork є єдиним у своєму класі, який не тільки виявляє, але й самостійно встановлює відсутні відновлення безпеки. Сканер здатний інтегруватися з Microsoft WSUS і за допомогою спеціального Агента здійснювати установку відновлень безпеки як для продуктів Microsoft, так і для програм сторонніх розроблювачів. Доступна установка відновлень практично для всіх продуктів Microsoft і більш 100 популярних програм, що працюють під керуванням Windows (таких як Adobe, Postgresql, Google та ін.). Каталог програм, обновлюваних за допомогою ScanNetwork, постійно розширюється.

Також спеціально для російських користувачів сертифікованих по вимогах безпеки продуктів Microsoft реалізований легітимний механізм керування (установки) відновленнями, що пройшли процедуру інспекційного контролю. Тепер можна за допомогою ScanNetwork і WSUS установлювати відновлення

безпеки, не порушуючи вказівок по експлуатації для сертифікованих версій продуктів Microsoft.

Аудит SCADA-систем

ScanNetwork має окремо ліцензуємий SCADA-модуль для сканування уразливостей елементів АСУ ТП (Iconics, Rockwell Automation, Schneider Electric, Siemens, Simatic).

Сканування здійснюється без привілеїв або використання облікових записів, що дозволяє швидко й ефективно провести аудит захищеності промислових систем АСУ ТП (SCADA), побудувати звіт з рекомендаціями з усунення й привести технологічний сегмент мережі у відповідність із вимогами Регулятора.

Аудит захищеності СУБД

Як правило сервери баз даних містять найбільш чутливу інформацію, втрата або розголошення якої може привести до фінансових втрат, штрафних санкцій з боку регуляторів і т.д. ScanNetwork є ефективним інструментом керування безпекою СУБД, його звіти про сканування можуть служити об'єктивними й аргументованими документами дотримання вимог відомчих і національних стандартів. Крім пошуку уразливостей і критичних невстановлених відновлень, сканер здатний здійснювати перевірки настроювань:

Аудит серверів застосунків

Заставою захищеності серверів застосунків є повсякденна рутинна робота адміністратора, що полягає в уведенні нових вузлів, відключенні невикористовуваних засобів і сервісів, відновленні програм, відстеженні уразливостей, керуванні настроюваннями безпеки. Якщо врахувати, що сервер застосунків може мати тризначна кількість параметрів, застосовуваних на різних рівнях (кореневому; окремого ресурсу або сайту; рівні каталогу і т.д.), а кількість окремих екземплярів (instance) серверів може становити десятки, то не складно представити необхідний рівень кваліфікації й завантаженість адміністраторів. У таких умовах основною погрозою стає людський фактор. Як відомо, людям

властиво помилятися, їхні фізичні можливості непорівнянні з комп'ютерними системами.

Ефективним інструментом контролю безпеки серверів застосунків є ScanNetwork. Сканер забезпечує комплексний аудит захищеності серверів застосунків по наступних напрямках контролю:

- аудит уразливостей;
- аудит відновлень;
- аудит конфігурацій параметрів безпеки;
- інвентаризацію встановленого ПЗ;
- аудит параметрів безпеки СУБД
- контроль цілісності застосунків.

Функція аудита конфігурацій безпеки серверів застосунків підтримує ряд популярних web-серверів і їх компонентів (платформ), таких як Apache, NGINX, Microsoft IIS, Microsoft .NET Framework.

Користувачам сканера доступні конфігурації параметрів безпеки:

- Apache HTTP Server.
- Apache Tomcat.
- IIS і .NET.
- Додаткові сервіси Linux.
- Web-сервера nginx.
- PHP.
- Remote Acces Checklist.

Застосування даних конфігурацій дозволяє контролювати настроювання безпеки найбільш використовуваних ролей і сервісів (HTTP Server, поштові сервера, текстові процесори, програмні платформи й ін.) для різних Linux і Windows систем. Конфігурації створені на основі рекомендацій вендорів і світових визнаних практик.

Аудит у режимі Пентест

Функція «Аудит у режимі Пентест» дозволяє проводити оцінку рівня захищеності інформаційних систем с мінімальними привілеями й знаннями про скануємий хост (методом Чорного ящика). Аудит здійснюється шляхом ідентифікації мережних вузлів, збору інвентаризаційних відомостей (ОС, порти, протоколи, сервіси та ін.) і зіставлення отриманих знань із ознаками (визначеннями) уразливого ПЗ, що зберігаються в базі знань ScanNetwork. Для проведення аудита також використовуються активні методи пошуку уразливостей с допомогою спеціалізованих інструментів і відомостей про експлойтах, арсенал яких постійно розширюється.

Детальний аудит платформ віртуалізації

Сучасні гіпервізори забезпечують високу безпеку, надійність і керованість, але при цьому вимагають такого ж уважного ставлення, як і будь-який фізичний комп'ютер. Віртуальні машини також піддані втраті даних (loss/corruption), вірусам і протиправним діям хакерів. Їм необхідний антивірусний захист, установка відновлень, регулярний моніторинг їх уразливості й конфігурації, резервування даних і середовища віртуалізації, дотримання рекомендацій з настроювання й безпечному використанню (експлуатації).

ScanNetwork є лідером у своєму класі по комплексному аудиті безпеки платформ віртуалізації й дозволяє проводити:

- Аудит уразливостей середовища віртуалізації й центрів керування.
- Аудит відновлень.
- Аудит конфігурацій параметрів безпеки.
- Інвентаризацію віртуальних і фізичних апаратних засобів, состава програмного забезпечення.
- Контроль цілісності конфігураційних файлів гіпервізорів, критичних бібліотек і файлів даних.

– В основі конфігурацій використані рекомендації Security Hardening і інші визнані практики.

Інвентаризація мережі

ScanNetwork дозволяє одержувати детальну інформацію про апаратні й програмних засобах мережі, включаючи: типи й опис заліза, версії й редакції операційних систем, установлені пакети відновлень і виправлень, установлене ПЗ, запущені служби, користувачів і груп, відомості про загальні папки, і багато іншого. Сканер безпеки дозволяє створити "опис" мережі без установки агента програми на комп'ютерах. Глибока деталізація звітів і використання функції "контроль" дозволяє відслідковувати навіть самі незначні зміни в складі програмного й апаратного забезпечення мережі.

Контроль цілісності

Функція "контроль цілісності" дозволяє використовувати ScanNetwork для виявлення й оповіщення про несанкціоновані зміни в конфігураційних файлах, папках, вітках реєстру або важливих файлах даних. Також ця функція може служити доповненням антивірусного захисту для боротьби з уразливостями нульового дня, які здатні підмінювати системні файли або додавати свої. Включення режиму "контроль" дозволяє із заданою періодичністю здійснювати перевірку цілісності "еталонированих" файлів і повідомляти про будь-які, навіть незначних змінах.

У самому сканері передбачене блокування запуску на випадок виявлення фактів порушення цілісності, що виконуються файлів і службових бібліотек, що дозволяє використовувати ScanNetwork в інформаційних системах, до яких пред'являються підвищені вимоги безпеки.

Контроль стану

Функція "Контроль" дозволяє автоматично відслідковувати зміни в системі: виникнення уразливостей, зміна настроювань параметрів безпеки, наявність відновлень, цілісність файлів, состав устаткування, програм і т.д. Адміністраторові необхідно лише зафіксувати "еталонне" стан комп'ютера, а

ScanNetwork візьме на себе обов'язок, періодично контролювати й сповіщати про будь-які навіть незначні зміни.

Підтримка утиліти Nmap

('Network Mapper') дозволяє через інтерфейс ScanNetwork здійснювати ряд мережних перевірок, у тому числі відкриті порти, використовувані протоколи, мережні сервіси та ін. Одержання інформації відбувається при мінімальному рівні привілеїв.

Перевірка складностей пароля

За допомогою програми ScanNetwork можна перевірити стійкість паролів операційних систем і СУБД. Перевірка здійснюється методом добору " по словникові" або парольним хешам.

Оповіщення

У сканері реалізована можливість відправлення на електронну пошту оповіщень про результати сканування.

Документування

Результати сканування зберігаються в "історії" перевірок і можуть бути експортовані у формати PDF і CSV. Звіти формуються в простому, або диференціальному виді, що дає можливість відслідковувати й документувати будь-які зміни в системі. Для швидкого пошуку необхідних подій у сканері реалізовані фільтри, що дозволяють виконувати вибірку по даті, тимчасовому інтервалі, типі перевірок, статусу, найменуванню й номеру хосту.

Створення інтегральних і диференціальних звітів по кожному напрямкові аудита

Результати сканування можуть зберігатися в "історії" перевірок або у вигляді звітів у форматі PDF. Звіти представляються в простому або диференціальному виді, що дозволяє легко відслідковувати будь-які зміни (нові уразливості, несанкціоновано встановлене ПЗ або "залізо").

Для швидкого пошуку необхідних подій у програмі реалізовані фільтри, що дозволяють здійснювати відбори по даті, тимчасовому інтервалі, типі перевірок, статусу, найменуванню або номеру хосту.

Формування звітів у форматі PDF дає можливість експортувати їх у будь-який текстовий формат, підтримуваний Adobe Acrobat або аналогічними програмами.

3.4 Розробка діаграми процесів

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

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

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

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

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників.

Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

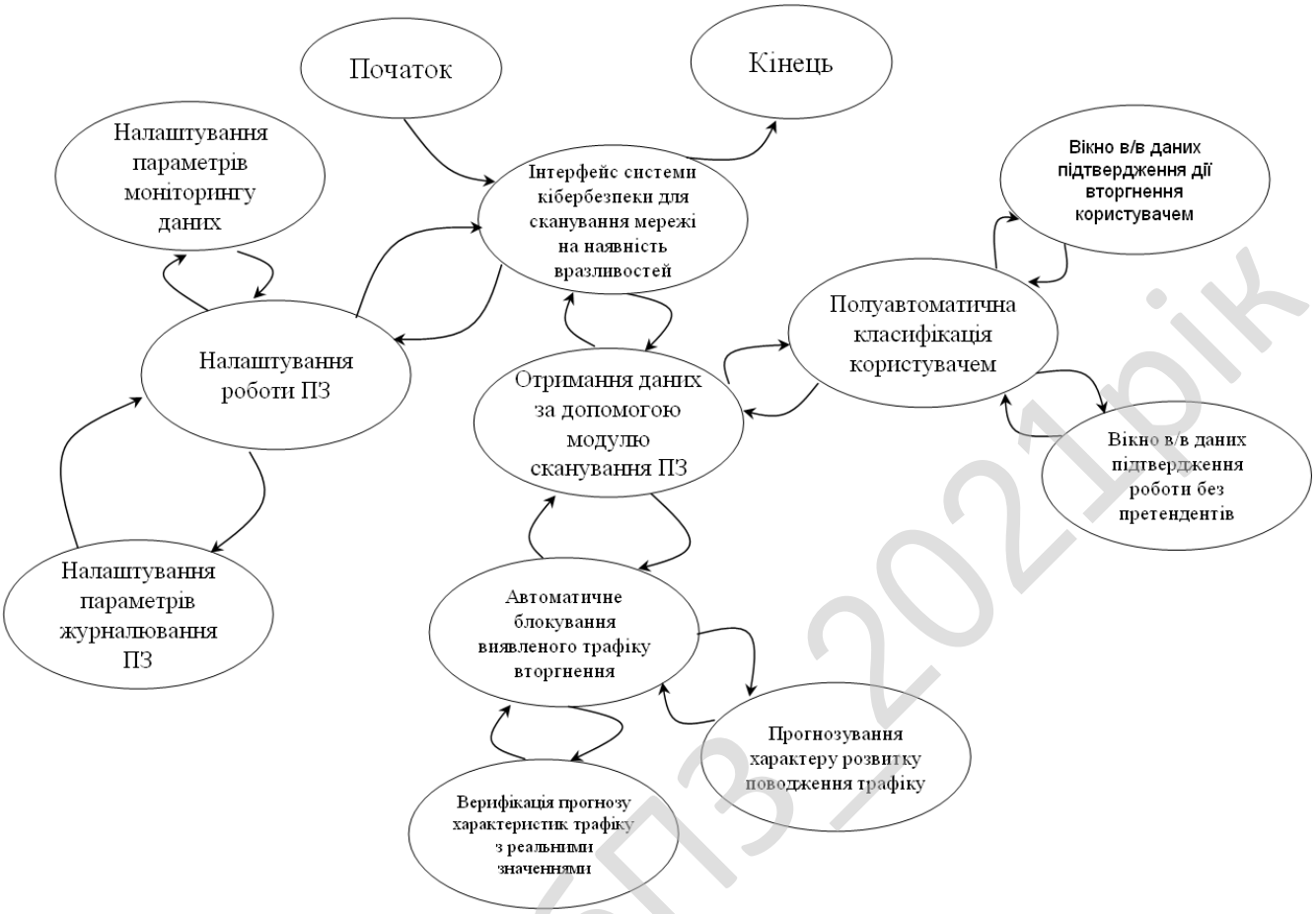

Рисунок 3.3 – Діаграма взаємодії процесів

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

– Виведення вікна сканування мережі на наявність вразливостей.

– Запит створення шаблону.

– Попередній збір та формування даних шаблону.

– Полуавтоматична класифікація даних на нормальний та трафік вторгнення.

– Підпрограма формування даних.

– Формування шаблонів образів системи кібербезпеки.

– Збереження шаблонів образів у БД.

– Сканування мережі та збір даних системи.

– Відображення статистичної інформації.

– Відображення трафіку роботи системи.

– Виклик підпрограми моніторингу більш детально дивись на рисунок 4.2.

– Запит вразливість виявлено.

– Виведення інформації про виявлення вразливості.

– Запис до журналу роботи системи.

– Завершення циклу роботи ПЗ.

На рисунку 4.2 зображено роботу підпрограми з реалізацією наступних дій:

– Аналіз трафіку та порівняння його з шаблонами.

– Запит шаблон співпав.

– Встановлення відмітки що знайдено трафік вразливості.

– Розпізнавання типу вразливості системи.

– Формування рішення рівня загрози.

– Виявлення зв'язків між трафіком вторгнення, що надходить з різних портів TCP/IP.

– Журналювання дії роботи підсистеми ПЗ.

– Прогнозування характеру розвитку поводження трафіку.

– Верифікація прогнозу характеристик трафіку з реальними значеннями.

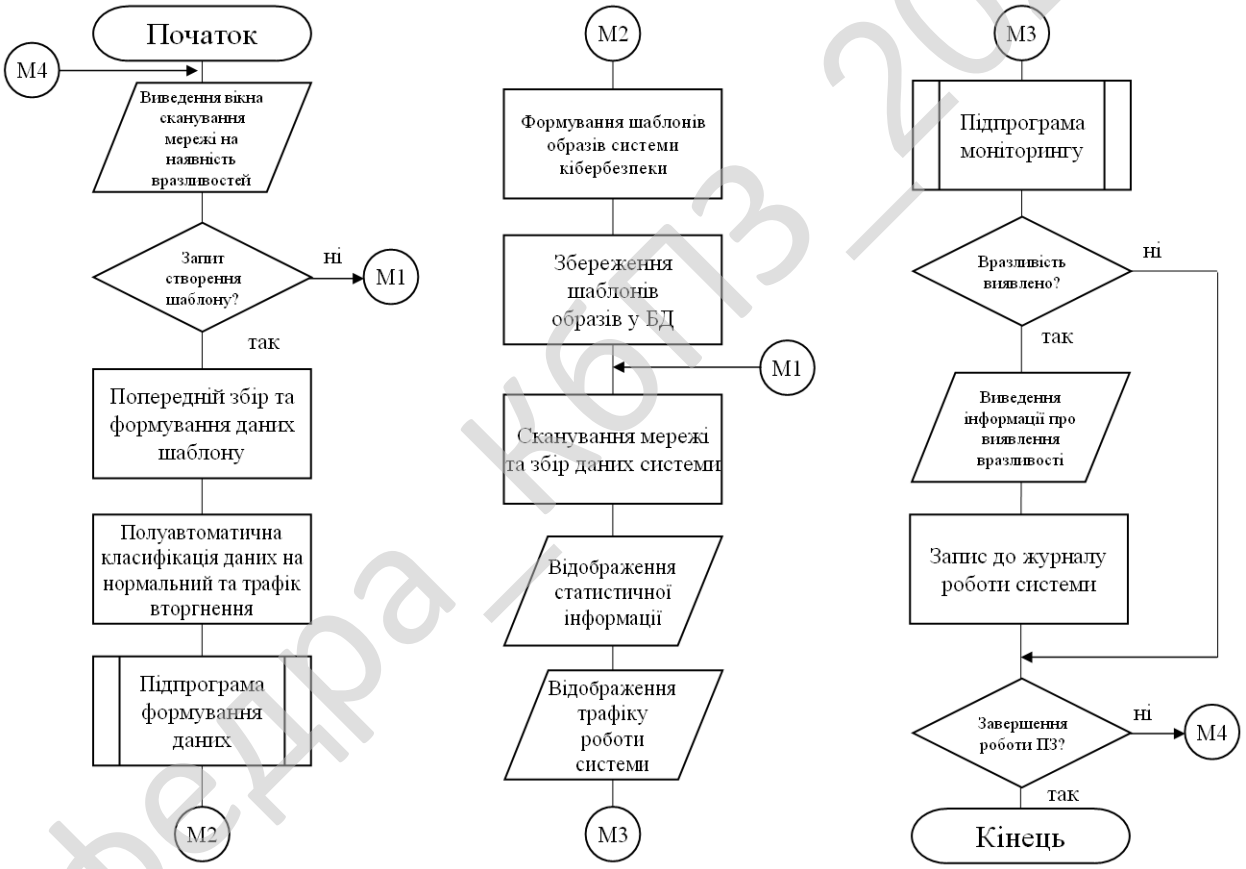

Рисунок 4.1 – Блок-схема основної програми

Наведемо частину коду у вигляді вихідного коду, який реалізує деякі з вищеперерахованих функцій:

```
procedure MY_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
```

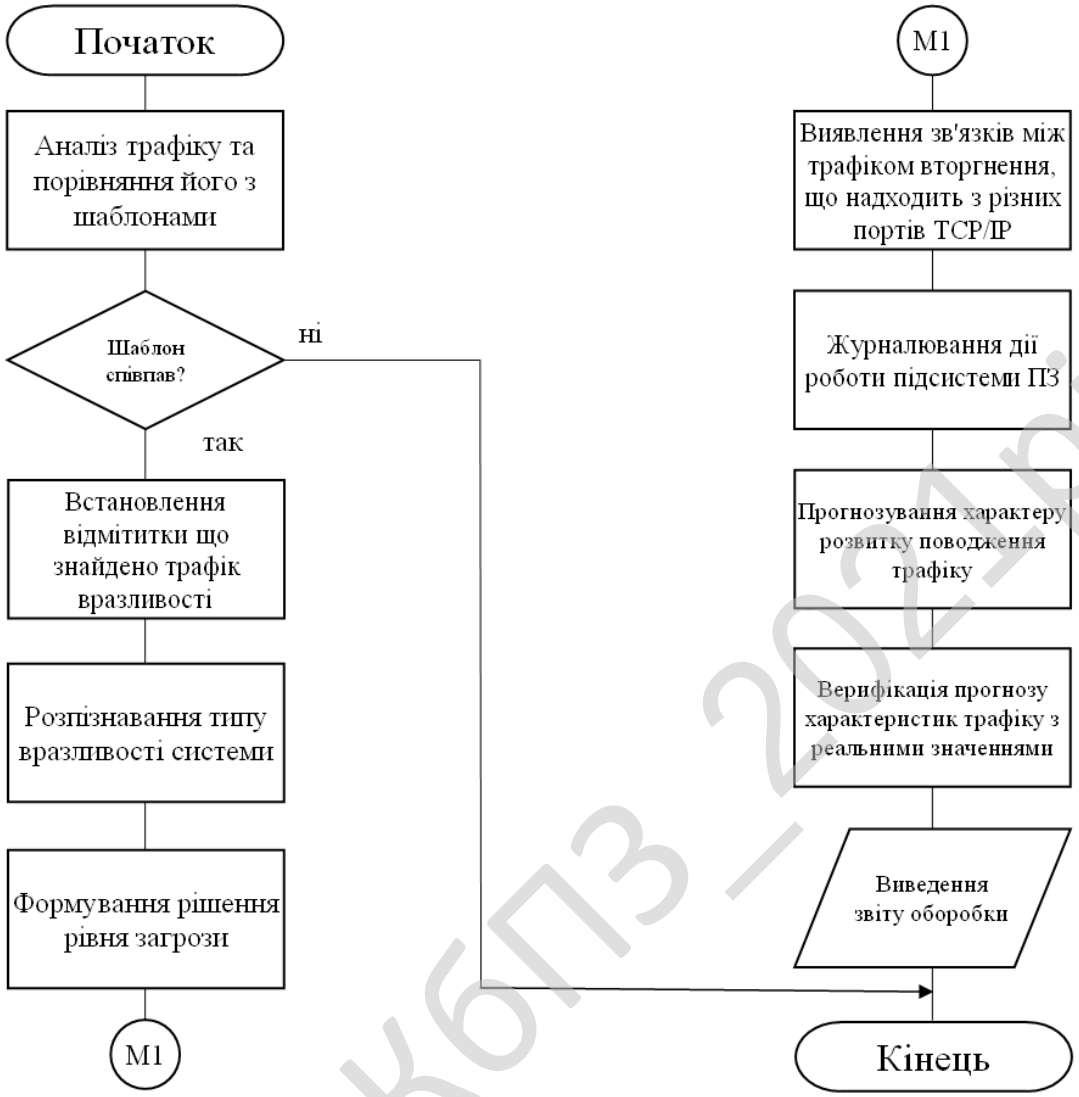

Рисунок 4.2 – Блок-схема роботи підпрограми

```
  else
    List.Add(SysErrorMessage(ErrorCode ));
  // відновлюємо показчик !
  dec(pBuf, SizeOf(DWORD ) + NumEntries * SizeOf(IPAddrRow ));
  FreeMem(pBuf );
end;
// Надання статистики по ICMP
procedure MY_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(' Ехо запит        : '  + IntToStr(dwEchos ));
      ICMPIn.Add(' Ехо 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('Ехо запит        : '  + IntToStr(dwEchos ));
      ICMPOut.Add('Ехо 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;
//Запис часу та завантаженості трафіку по UDP
procedure MY_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;

//Запис IP адресів до MIB
```

```
procedure MY_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 MY_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 MY_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(' 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 MY_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;
procedure MY_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 MY_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 останньої 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;
  end;
  dec(pBuf, SizeOf(DWORD ) + NumEntries * SizeOf(TMibTCPRow ));
  FreeMem(pBuf );
end;
```

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення –звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

Було використано файли формату XML –тандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML.

XML-документ складається із текстових знаків, і придатний до читання людиною. Стандарт XML (Recommendation, перше видання від 10 лютого 1998,

останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів.

Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (Schema), такими як XML Schema (XSD), DTD або RELAX NG. Прикладами мов, заснованих на XML, є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, а також XML Schema.

Основні поняття. Коректність.

Коректний документ (well-formed document) відповідає всім синтаксичним правилам XML. Документ, що не є коректним, не може називатись XML-документом. Сумісний синтаксичний аналізатор (Conforming parser) не повинен обробляти такі документи. Зокрема, коректний XML-документ має:

1. Лише один елемент у корені.

2. Непорожні елементи розмічено початковим та кінцевим тегами (наприклад, <пункт> Пункт 1</пункт>). Порожні елементи можуть позначатися «закритим» тегом, наприклад <IAmEmpty />. Така пара еквівалентна <IAmEmpty></IAmEmpty>.

3. Один елемент не може мати декілька атрибутів з однаковою назвою. Значення атрибутів перебувають або в одинарних ('), або у подвійних (") лапках.

4. Теги можуть бути вкладені, але не можуть перекриватись. Кожен некореневий елемент мусить повністю перебувати в іншому елементі.

5. Документ має складатися тільки з правильно закодованих дозволених символів Юнікоду. Єдиними кодуваннями, які обов'язково має розуміти XML-процесор, є UTF-16 та UTF-8. Фактичне та задекларованe кодування (character encoding) документа мають збігатись. Кодування може бути задекларовано ззовні,

як у заголовку «Content-Type» при передачі по протоколу HTTP, або в самому документі використанням явної розмітки на самому початку документа. У разі відсутності інформації про кодування, документ має бути в кодуванні UTF-8 (або його підмножині ASCII).

Валідність

Документ називається валідним (valid), якщо він є коректним, містить посилання на граматичні правила та повністю відповідає обмеженням, вказаним у цих правилах (DTD або XML Schema або іншому подібному документі).

Синтаксичний аналізатор

Синтаксичним аналізатором (часто парсер, від parser) називається програма або компонент, що читає XML-документ, проводить синтаксичний аналіз та відтворює його структуру. Якщо синтаксичний аналізатор перевіряє документ на валідність, то такий аналізатор називають валідатором (validating).

Назви елементів чутливі до регістру літер. Наприклад, наступна пара елементів правильна: <Step> … </Step> у той час як ця – ні: <Step> … </step>.

Правильний вибір назв для XML-елементів підкреслюватиме значення даних у створеній мові розмітки. Це сприятиме полегшенню роботи людей з такими документами, зберігаючи можливості для комп'ютерної обробки даних.

Вибір змістовних назв передає семантику елементів та атрибутів для людини, без посилання на зовнішню документацію. Однак це може призвести до надмірності розмітки, що ускладнює редагування й збільшує розмір файлів.

Правильний вибір назв для XML-елементів підкреслюватиме значення даних у створеній мові розмітки. Це сприятиме полегшенню роботи людей з такими документами, зберігаючи можливості для комп'ютерної обробки даних. Вибір змістовних назв передає семантику елементів та атрибутів для людини, без посилання на зовнішню документацію. Однак це може призвести до надмірності розмітки, що ускладнює редагування й збільшує розмір файлів. XML-документи мають як фізичну, так і логічну структуру.

Фізична структура

Сутності (Entity). Головною сутністю є зміст документа. Інші можливі сутності вказуються за допомогою. Посилання на сутності (&назва; в самому документі, та, наприклад %назва; у визначені його типу) можуть слугувати в ролі позначень спеціальних символів, посилань на спеціальні символи або окремих документів чи фрагментів тексту.

XML-декларація, в ній вказується версія XML, кодування та інша допоміжна інформація.

Декларація типу документа може застосовуватись для того, аби додавати нові типи сутностей та визначати логічну структуру документа.

Логічна структура

XML-документ має ієрархічну логічну структуру, і може представлятись у вигляді дерева. Вузлами цього дерева можуть бути: елементи, фізична структура яких складається із коректної пари відкриваючого та закриваючого тегів (<Назва-тега>) та (</Назва-тега>), або тега порожнього елемента (<Назва-тега />).

Атрибути, що мають вигляд пар ключ-значення (назва атрибута="значення атрибута") і знаходяться або у відкриваючому, або у порожньому тезі (подібно до метаданих).

XML-документ повинен мати лише один кореневий елемент. Решта елементів є піделементами цього кореневого елемента. Деякі веб-браузери здатні безпосередньо відображати XML-документи. Це може досягатись шляхом застосування таблиці стилів (Stylesheet). Вказані у таблиці стилів операції можуть призводити до перетворення XML-документа в інший, відмінний від XML формат.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму захисту інформації DSA.

Відправник і одержувач електронного документа використовують при обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$. Число Y є відкритим ключем для перевірки підпису відправника. Число Y передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-). У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M, відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана. Потім одержувач обчислює значення $w = 1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$. Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$. Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

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: Аналізатор трафіку; База образів; Журнал дій; Блокиратор; Довідник; Налаштування.

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

– Верхнього меню: Файл; Дані; Додатки; Налаштування; Довідка.

– Розділу виведення результату роботи системи у графічному вигляді.

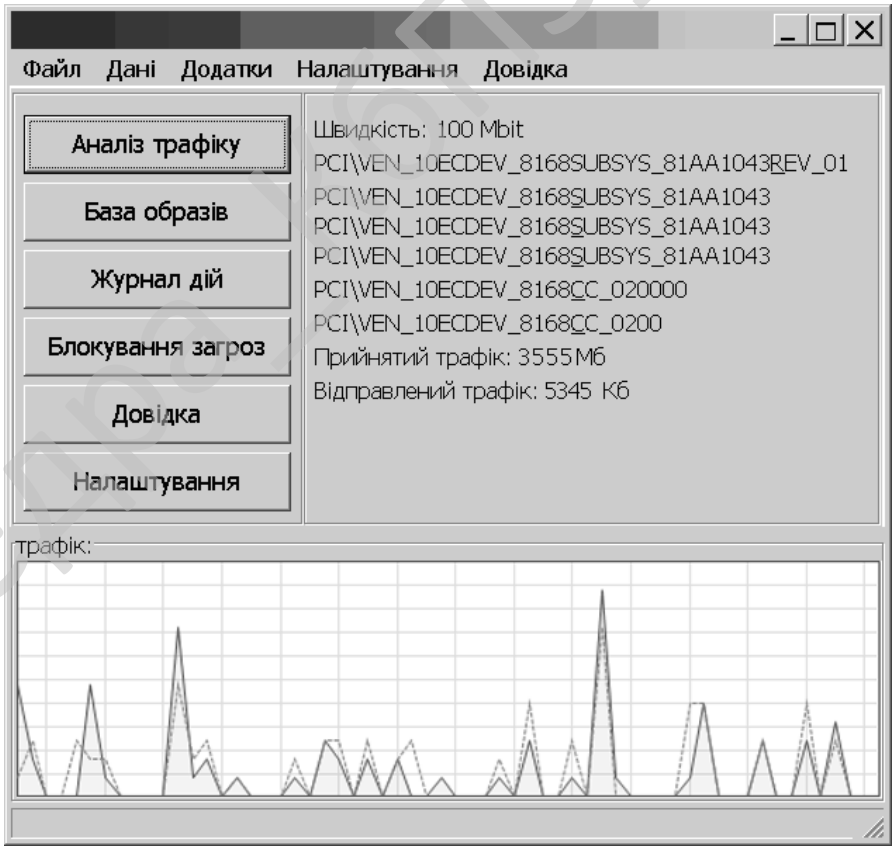

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

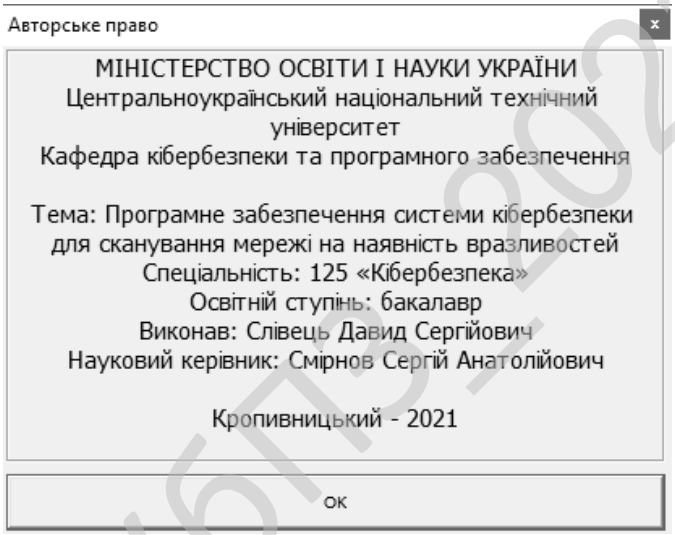

Рисунок 5.2 – Авторське право

Процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення. Таким чином у результаті вищерозглянутого можна стверджувати що розроблено інтерфейс системи у відповідності з вибраною метою роботи. Система містить максимальний необхідний набір функцій придатних для виконання будь-яких дій для забезпечення повноцінної роботи програми. Далі розглянемо висновки та використані літературні джерела.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи кібербезпеки для сканування мережі на наявність вразливостей.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем кібербезпеки для сканування мережі на наявність вразливостей.

– Досліджена система кібербезпеки для сканування мережі на наявність вразливостей.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для сканування мережі на наявність вразливостей.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання для сканування мережі на наявність вразливостей.

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 Embarcadero Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для сканування мережі на наявність вразливостей. 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 DSA.

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

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

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

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

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

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

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

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

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

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

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

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

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

14. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов //

Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

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

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

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

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

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

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

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

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

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

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

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

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

27. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція

«Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

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

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

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

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

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

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

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

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

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

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

38. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення

кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

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

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

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

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

43. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового

забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

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

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для сканування мережі на наявність вразливостей.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для сканування мережі на наявність вразливостей;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Embarcadero Delphi.

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 – 83 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 9.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки для сканування мережі на наявність вразливостей

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 35

Літера: РП

Кропивницький – 2021 року

unit TrafficUnit – визначення параметрів трафіку системи кібербезпеки для сканування мережі на наявність вразливостей;

```
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.
```

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 ScanNetwork.dpr основної програми

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

program ScanNetwork;

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 системи кібербезпеки для сканування мережі на наявність вразливостей

```
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 Сессійний 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 системи кібербезпеки для сканування мережі на
наявність вразливостей 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 системи кібербезпеки для сканування мережі на
наявність вразливостей
        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 системи кібербезпеки для сканування мережі на
наявність вразливостей
        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;
  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 системи кібербезпеки для сканування
мережі на наявність вразливостей 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 системи кібербезпеки для сканування мережі на
наявність вразливостей
        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 системи кібербезпеки для сканування
мережі на наявність вразливостей 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 about – підпрограмма про розробника 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.
```