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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу”

Виконала здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ А.Г. Савеленко
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Р.М. Минайленко
«____» _____ 2025 р.
Рецензент _____ С.М. Мороз
«____» _____ 2025 р.

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *123 “Комп’ютерна інженерія”*

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

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

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

Савеленко Анастасії Григорівні
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу*

2. Керівник роботи *Минайленко Роман Миколаєвич, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *. .2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*4 аркушів*

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	26.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Минайленко Р.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Савеленко А.Г.
(прізвище та ініціали)

АНОТАЦІЯ

Савеленко А.Г. Програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу. 123 "Комп'ютерна інженерія". Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначене для системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

Метою розробки є програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

Результат роботи – програмна реалізація системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

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

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

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows.

Програму розроблено в середовищі Delphi.

Ключові слова: комп'ютерна інженерія, комп'ютерна мережа, мережева безпека, проактивний захист, поведінковий аналіз.

ABSTRACT

Savelenko A.H. Software of the system of proactive detection and blocking of malicious software in a computer network based on behavioral analysis. 123 "Computer Engineering". Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is designed for a system for proactive detection and blocking of malware in a computer network based on behavioral analysis.

The purpose of the development is to create software for a system of proactive detection and blocking of malware in a computer network based on behavioral analysis.

The result of the work is a software implementation of a system for proactive detection and blocking of malware in a computer network based on behavioral analysis.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with the software are given.

The program can be used on IBM PC architecture PCs with Windows OS.

The program is developed in the Delphi environment.

Keywords: computer engineering, computer network, network security, proactive protection, behavioral analysis.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи..... 6
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 9
2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування..... 24
2.3 Розгорнута постановка завдання 31
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 33
3.1 Опис функціонування системи 33
3.2 Розробка структурної схеми..... 47
3.3 Розробка функціональної схеми 52
3.4 Розробка діаграми процесів..... 60
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 63
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 63
4.2 Захист розробленого програмного забезпечення..... 83
5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 85
6 ОСНОВНІ ВИСНОВКИ..... 90
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 92

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВКБР	- випускна кваліфікаційна бакалаврська робота
ДСТУ	- Державний стандарт України
КМ	- комп’ютерна мережа
МА	- мережева активність
ММА	- моніторинг мережевої активності
МПЗ	- модуль проактивного захисту
ОС	- операційна система
ПЗ	- програмне забезпечення
ПК	- персональний комп’ютер
ТЗ	- технічне завдання

ВСТУП

Актуальність теми. В епоху надшвидкого прогресу цифрових технологій та всебічної інформатизації господарських процесів, проблематика гарантування безпеки мережевих інфраструктур стає критично важливою. Сучасні комп'ютерні мережі перетворюються на привабливі мішені для зловмисників, чиї атаки спричиняють дедалі серйозніші збитки національним інформаційним ресурсам, бізнес-структурам, громадським організаціям та їхнім технологічним екосистемам.

Після повномасштабного військового вторгнення російської федерації на територію України починаючи з 24 лютого 2022 року, наша держава функціонує в умовах запровадженого воєнного стану. Паралельно з фізичною агресією, російські військові формування активно здійснюють ворожі дії у віртуальному просторі України. Статистичні дані демонструють трикратне збільшення інтенсивності кібернетичних нападів на державну інфраструктуру та стратегічно важливі інформаційні об'єкти. Аналітики встановили, що приблизно 90% деструктивних втручань координуються хакерськими підрозділами збройних сил росії та білорусі, діяльність яких безпосередньо фінансується державними структурами цих країн. Постійне підвищення технологічної складності та частоти цифрових нападів потребує розробки й імплементації інноваційних захисних механізмів для забезпечення конфіденційності інформації, збереження її цілісності та мінімізації потенційних загроз.

Випереджувальний захист, що базується на застосуванні поведінкового аналізу, представляє собою комплекс передових методик та інструментів, основною функцією яких є ідентифікація підозрілих активностей програмних продуктів у мережевому середовищі ще до реалізації атаки. На відміну від традиційних сигнатурних підходів, які виявляють лише відомі шкідливі програми після їх потрапляння в систему, випереджувальні технології здатні прогнозувати

та запобігати потенційним загрозам завдяки аналізу нехарактерної поведінки додатків. Такий проактивний підхід дозволяє блокувати небезпечні дії програмного забезпечення ще на етапі їх ініціації.

Сучасні системи поведінкового аналізу використовують алгоритми машинного навчання та штучного інтелекту для постійного вдосконалення захисних механізмів. Вони здатні адаптуватися до нових типів загроз та виявляти аномалії, які можуть свідчити про кібератаку. Впровадження таких систем у комплексну стратегію кібербезпеки дозволяє організаціям значно підвищити рівень захисту своїх інформаційних активів.

У контексті гібридної війни, яку веде росія проти України, особливу увагу слід приділяти захисту об'єктів критичної інфраструктури, таких як енергетичні системи, транспортні мережі, фінансові установи та медичні заклади. Ці об'єкти найчастіше стають мішенями кібератак через їхнє стратегічне значення для функціонування держави та суспільства.

Тож, ефективна організація безпеки комп’ютерних мереж вимагає впровадження багаторівневих підходів, створення інтегрованих систем захисту, які відповідають викликам сучасного кіберсередовища. На противагу традиційному підходу порівняння зі словниковими дефініціями шкідливого коду, методологія виявлення аномальної активності забезпечує ефективний захист від невідомих раніше шкідливих програм, які ще не зафіксовані в жодній базі даних зловмисного програмного забезпечення. Такий поведінковий аналіз дозволяє превентивно ідентифікувати потенційні загрози за характерними ознаками їхнього функціонування, а не за відомими сигнатурами, що суттєво розширює спектр виявлення новітніх кіберзагроз. Системи, що базуються на моніторингу підозрілих дій, здатні розпізнавати та блокувати зловмисне програмне забезпечення нульового дня ще до внесення його характеристик до антивірусних каталогів. Впровадження цих підходів забезпечує не лише поточний захист, але й здатність до адаптації у швидкозмінному технологічному ландшафті.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

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

– огляд існуючих систем проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу;

– дослідження систем проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу;

– програмна реалізація системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі використання технології поведінкового аналізу. Розроблена в процесі виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (ВКРБ) є дієздатною та готовою до впровадження на 100%.

Таким чином, виходячи з вищеозначеного, розробка програмного забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

В процесі виконання ВКРБ розробці підлягає програмне забезпечення (ПЗ) системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі (КМ) на основі поведінкового аналізу (в подальшому - система).

Сьогоднішня військова реальність нашої країни чітко демонструє, що інформація є «зброєю масового знищення». Від початку широкомасштабної відкритої агресії рф інтенсивність кібератак не знижується. Ці атаки координуються з атаками на критичну інфраструктуру і є частиною воєнної агресії рф. Тому існує потреба у забезпеченні захисту національної інформаційної безпеки шляхом створення ефективних, багаторівневих систем захисту комп'ютерних мереж на основі використання сучасних, інноваційних технологій та механізмів захисту.

Безпека комп'ютерної мережі – це заходи, які захищають її від внутрішніх та зовнішніх мережних атак; забезпечують конфіденційність обміну інформацією з будь - якого місця та в будь - який час; контролюють доступ до інформації, ідентифікуючи користувачів та їхні системи; забезпечують надійність КМ; присікають спроби руйнування її компонентів;

Проактивний захист - одна із сучасних технологій, що втілюють якісний стрибок в області протидії вірусним атакам. Суть технології в тому, що аналізу піддаються не дані на носіях інформації, а поводження самих даних – сукупність дій, чинених у системі. Якщо вважати якийсь клас даних шкідливим, то можна визначити й характерні риси поводження.

Технологія аналізу поведінки ґрунтується на перехопленні усіх важливих системних функцій або установці міні-фільтрів, що дозволяє відстежувати усю активність в КМ. Технологія поведінкового аналізу дозволяє оцінювати не лише

одиничну дію, але і ланцюжок дій, що багаторазово підвищує ефективність протидії вірусним загрозам.

Для розробки дійсно корисної та дієздатної системи, доцільно виконати розробку програмного шлюзового сервера, до складу якого ввести наступні компоненти: прямий проксі-сервер (Forward proxy), брандмауер, модуль моніторингу мережевої активності, модуль моніторингу стану відкритих ресурсів, модуль проактивного захисту. Адже у сучасному світі, де цифрові загрози набувають дедалі витонченіших форм, а кібератаки стають цілеспрямованими та руйнівними, безпека інформаційних систем КМ виходить на перший план.

Програмний шлюзовий сервер – це не просто точка входу в КМ, а комплексне рішення безпеки, яке захищає критичну інфраструктуру КМ. Подібно до вартового біля воріт замку, шлюзовий сервер стоїть на сторожі цінних даних, забезпечуючи контрольований доступ до внутрішніх ресурсів та фільтруючи потенційно небезпечний трафік.

Найбільша цінність програмного шлюзового сервера полягає не в окремих компонентах, а в їх синергії. Коли прямий проксі-сервер (Forward proxy), брандмауер, проксі-сервер, системи моніторингу та проактивного захисту працюють як єдиний організм, вони створюють багаторівневу систему захисту, яка значно перевершує суму своїх частин.

Таким чином, основне призначення системи, що підлягає розробці – це проактивний захист на основі поведінкового аналізу КМ від шкідливого ПЗ з зовнішньої мережі, зменшуючи вірогідність підтвердження DdoS, а також контроль мережевої активності додатків та заборона/обмеження їх доступу в мережу. Окрім цього, система допоможе зберегти дорогоцінний трафік: багато програм після установки зразу підключаються до Інтернет для оновлення (Adobe Photoshop CS3, наприклад, може тихенько викачати близько 300 Мб). Зі встановленою системою захисту така активність гарантовано не залишиться непоміченою.

1.2 Область застосування

Розроблена система буде мати надзвичайно широку сферу застосування. Вона може бути впроваджена в будь-якій організації/установі/навчальному закладі/підприємстві, які працюють з персональними комп’ютерами та комп’ютерними мережами.

Розроблені інноваційні інструменти суттєво покращують обізнаність технічних спеціалістів та фахівців із захисту інформації щодо функціональних процесів, які відбуваються всередині мережевих інфраструктур. Запропоновані методологічні підходи можуть ефективно використовуватися для автоматичного формування структурних схем мережі в режимі реального часу, що значно спрощує автоматизацію виявлення та точного визначення розташування потенційних джерел безпекових ризиків, пов'язаних із неавторизованими втручаннями в роботу телекомунікаційної екосистеми організації.

Імплементація даних механізмів дозволяє проактивно моніторити мережевий трафік, аналізувати аномальну поведінку користувацьких сесій та швидко реагувати на підозрілі активності. Візуалізація стану мережевих комунікацій надає професіоналам цілісну картину безпекової ситуації, що критично важливо для оперативного прийняття рішень під час інцидентів інформаційної безпеки.

Окрім цього, система може бути використана в навчальних закладах будь-якого рівня акредитації в якості своєрідного тренажера для підвищення ефективності вивчення курсу дисциплін з основ програмування: системного та прикладного.

Таким чином, виходячи з вищеперерахованого, розробка ПЗ системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Предметом розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу (в подальшому - система).

Захист мережевих інфраструктур виступає фундаментальною складовою загальної системи інформаційної безпеки в контексті сучасних кіберзагроз. Сьогоднішнє середовище комп'ютерних мереж, відкриваючи безпрецедентні перспективи для розширення функціональності, підвищення продуктивності та раціоналізації використання цифрових ресурсів, одночасно стає мішенню для комплексних багаторівневих атак зловмисників.

Протидія таким загрозам потребує всебічного стратегічного підходу до розробки й імплементації багатошарової архітектури захисту, яка враховує особливості постійно змінюваних мережевих структур, впроваджує передові технологічні рішення та фокусується на гнучкості й швидкій адаптації до виникаючих безпекових викликів.

Ефективна система мережевого захисту має інтегрувати проактивні механізми виявлення аномалій, інструменти поведінкового аналізу та автоматизовані процеси реагування на інциденти, забезпечуючи цілісну безпекову екосистему, здатну протистояти різноманітним типам кібератак у режимі реального часу.

Тож, для розробки дійсно корисної системи захисту, розглянемо існуючі на ринку програмних продуктів системи - аналоги. Для проведення

порівняльного аналізу сучасних систем захисту комп’ютерних мереж використаємо ключові критерії, які охоплюють технічні та економічні аспекти: вартість, функціональність, інтеграцію та продуктивність.

Програмні комплекси антивірусного захисту представляють собою найбільш усталений метод гарантування безпеки кінцевих пристроїв інформаційної інфраструктури. Ключові можливості таких систем сконцентровані навколо ідентифікації, превентивного блокування та остаточного усунення ідентифікованого зловмисного програмного коду. У сучасному арсеналі засобів інформаційної безпеки антивірусні рішення займають провідне місце як механізм протидії вже задокументованим шкідливим програмам.

Функціональна архітектура антивірусних продуктів базується на технології зіставлення системних файлів та активних обчислювальних процесів із референтною колекцією цифрових сигнатур, яка містить каталогізовані зразки шкідливих програмних компонентів. Варто зазначити, що передові антивірусні комплекси сьогодні активно запроваджують методики евристичного аналізу, що уможливлює виявлення раніше невідомих типів загроз через ідентифікацію характерних поведінкових патернів, притаманних зловмисному програмному забезпеченню.

Попри свою поширеність, антивірусні рішення мають низку обмежень:

- обмежена ефективність проти невідомих загроз;

- традиційні антивіруси залежать від оновлення бази сигнатур, що залишає прогалини для “нульових днів”;

- відсутність активного моніторингу.

Більшість антивірусів функціонує у форматі “після виявлення”, тобто реагує на загрозу лише після її ідентифікації, що на сьогодення є надзвичайно суттєвим недоліком.

Endpoint Detection and Response (EDR) є еволюційним антивірусним програмним продуктом для захисту робочих станцій комп'ютерної мережі, що суттєво перевершує можливості класичних антивірусних програм. Технологія

EDR зосереджується на оперативному знаходженні підозрілої активності, дослідженні нестандартної поведінки та формуванні детального розбору інцидентів у сфері кібербезпеки. Досягнення цих функцій забезпечується за допомогою таких важливих елементів:

- постійне спостереження за діяльністю: рішення EDR невпинно відстежує різноманітні події на пристроях, охоплюючи запуск додатків, модифікації у файловій структурі й встановлення з'єднань з мережею;

- аналітика поведінки допомагає ідентифікувати в комп'ютерній мережі витончені атаки без специфічних цифрових відбитків;

- взаємодія з додатковими захисними рішеннями: технології EDR зазвичай поєднуються з системами управління інформацією та подіями безпеки для співставлення даних, отриманих від мережевої інфраструктури та кінцевих робочих станцій.

До ключових достоїнств технології EDR варто віднести такі особливості:

- комплексна видимість усіх процесів із детальним відображенням;

- миттєва протидія кіберзагрозам та негайне реагування на підозрілу активність;

- функція автоматичного блокування чи карантину скомпрометованих пристроїв при виявленні небезпеки.

Всебічне дослідження безпекових інцидентів надає можливість адміністративному персоналу комп'ютерних мереж проводити розслідування та впроваджувати захисні механізми проти аналогічних кібератак надалі.

Традиційні антивірусні продукти забезпечують базовий захисний рівень і найкраще відповідають потребам малих підприємств з обмеженим бюджетом. На відміну від них, системи виявлення та реагування спрямовані на захист мережевої інфраструктури або її окремих сегментів для компаній, що потребують всеохоплюючого, випереджаючого підходу до кібербезпеки інформаційного середовища.

Дослідження демонструють, що рішення категорії EDR демонструють набагато вищі показники ефективності виявлення сучасних загроз завдяки впровадженню технологій аналізу поведінкових патернів та синергії з іншими безпековими платформами. Такі захисні механізми гарантують оперативну відповідь на інциденти і легше адаптуються до масштабних мережевих структур, хоча потребують залучення додаткових обчислювальних потужностей. Натомість класичні антивірусні засоби на базі сигнатурних методів виявлення залишаються доцільними для формування початкового захисного бар'єру через їхню інтуїтивність, помірні вимоги до системних ресурсів та економічну доступність.

До суттєвих недоліків слід віднести досить високу щорічну оплату за користування та оновлення програмного коду системи.

Trend Micro 2025 – система, яка є дійсно потужним захистом від вірусів і інших шкідливих програм для серверів, контейнерів, мереж та користувацьких пристроїв. Розробка американо-японської транснаціональної корпорація, що займається розробкою ПЗ у сфері комп'ютерної безпеки Trend Micro. Продукти компанії отримали найкращі оцінки в результаті тестів, проведених великими незалежними лабораторіями.

Trend Micro Titanium Antivirus Plus, це базовий сервіс, який надає основні інструменти для виявлення і видалення шкідливих програм. Коли запускається сканування, він перевіряє усі файли на наявність шкідливого ПЗ та підозрілих процесів в мережі і, якщо знайде що-небудь підозріле, видалить його.

Для випереджаючого захисту комп'ютерних мереж компанія Trend Micro застосовує систему евристичного аналізу разом із платформою раннього сповіщення Outbreak Alert System, яка оперативно інформує про виявлені нові кіберзагрози. Бізнес-рішення Trend Micro OfficeScan Corporate Edition інтегрує спеціалізовані цифрові відбитки загроз із сервісу Outbreak Prevention Service, що забезпечує автоматичне налаштування безпекових політик та створення захисних бар'єрів ще до моменту розповсюдження стандартних оновлень для антивірусних баз даних.

Евристичний аналізатор не розглядає певні процеси, файли або теки, а аналізує їх код. Такий метод допомагає знаходити і усувати будь-які шкідливі програми, які можуть ховатися в, здавалося б, звичайному файлі. Якщо Trend Micro виявить що-небудь підозріле, він завантажить його в хмару компанії для подальшого вивчення командою експертів.

Антивірус Trend Micro включає ще одну цікаву утиліту - Folder Shield, яка дозволяє вибирати певні теки, які потребують захисту від будь-якої атаки і зберігання в недоторканності, адже частенько шкідливе ПЗ проникає в КМ, а відповідно, і на ПК користувачів без їх відома в результаті відвідування сумнівних сайтів або завантаження підозрілих файлів. Ця функція допоможе також оптимізувати налаштування конфіденційності і переконатися, що профілем цікавляться тільки ті люди, які мають на це санкціонований доступ.

Програми-вимагачі (ransomware) становлять серйозну загрозу для цифрових даних, адже зловмисники зашифровують файли користувачів та блокують віддалений доступ до інформації з персонального пристрою, доки не отримають винагороду. Ситуація погіршується тим, що шкідливе програмне забезпечення такого типу перетворилося на один із найпоширеніших інструментів, через який кіберзлочинці вимагають фінансову компенсацію чи інші вигоди від власників інформаційних ресурсів або технологічних систем. Однак рішення від Trend Micro вирізняється своєю ефективністю завдяки комплексній системі багаторівневого захисту проти ransomware-атак.

При виявленні несанкціонованого шифрування файлів захисне рішення Trend Micro негайно активує подвійний механізм безпеки: автоматично створює резервні копії зашифрованих документів, гарантуючи законному власнику повноцінний доступ до важливої інформації, а при фіксації підозрілої активності – рішуче втручається для припинення процесу шифрування даних. Технологічний арсенал Trend Micro дозволяє успішно подолати будь-які бар'єри та блокувальні механізми, встановлені шкідливими програмами-вимагачами.

Вкладка "Конфіденційність Trend Micro" утримує декілька інструментів, які чудово підходять для захисту соціальних мереж, дозволяючи перевіряти значну кількість сайтів соціальних мереж, включаючи Facebook, Twitter, Google+ і інші веб-сайти, що забезпечує оптимізацію налаштування конфіденційності. Щоб усе працювало належним чином, необхідно установити розширення Trend Micro.

Ще одна унікальна і приголомшлива функція - це Data Theft Prevention, дозволяє легко захищати найважливішу інформацію. Необхідно лише ввести набір букв і навіть слів в "список спостереження" програми, і якщо Trend Micro виявить таку послідовність в мережі, вона автоматично заблокує з'єднання.

Trend Micro легко налаштовувати і користуватися ним.

Єдине невеликим недоліком Trend Micro на думку фахівців, є те, що технічна підтримка розділена на підтримку Standard і Premium, і тільки Premium дає цілодобовий доступ до фахівців служби підтримки по телефону. Проте підтримка Standard, як і раніше, доступна з понеділка по п'ятницю, з 5:00 до 20:00 по тихоокеанському стандартному часу.

Ціна системи Trend Micro відносно доступніша, ніж у деяких більших брендових антивірусних компаній, і складає від 830 грн. до 2 080 грн. на рік.

WinPatrol - безкоштовна утиліта із закритим початковим кодом, призначена для моніторингу КМ на предмет несанкціонованих змін в 32-бітових і 64-розрядних операційних систем Windows.

Програмний засіб проводить детальне сканування комп'ютера та створює резервну копію критично важливих елементів користувацької операційної платформи (включаючи реєстр Windows, завантажувальні модулі, каталог системних файлів), які безпосередньо впливають на функціональність та продуктивність обчислювальної системи, після чого забезпечує безперервний нагляд і контроль за їхнім станом.

За допомогою цього інструментального рішення можливо відслідковувати різноманітні шкідливі застосунки, шпигунське програмне забезпечення, рекламні

модулі та мережеві паразитуючі алгоритми, які часто інсталюються без належного дозволу користувача та здатні суттєво порушити нормальне функціонування як окремих додатків, так і цілісної комп'ютерної платформи. При виявленні потенційно небезпечного компонента, утиліта миттєво надсилає сповіщення відповідальному фахівцю та надає можливість оперативно припинити виконання підозрілого процесу. Важливою перевагою даного захисного рішення є його повна сумісність із вже встановленими антивірусними продуктами, що робить його ефективним додатковим шаром кіберзахисту.

Викликавши програму, отримаємо вікно програми з декількома вкладками (рисунок 2.1).

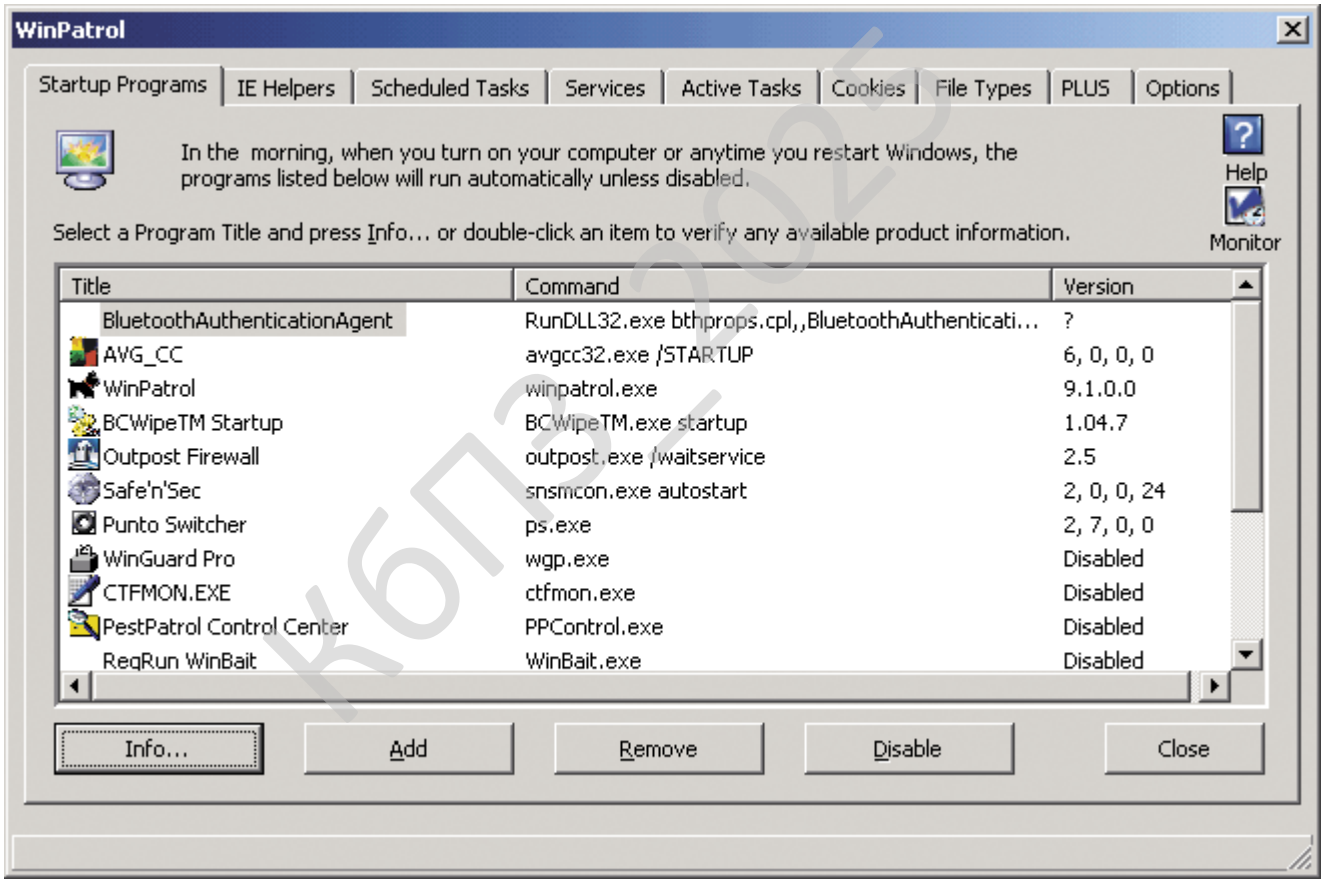

Рисунок 2.1 - Головне вікно WinPatrol

- Startup Programs - дозволяє проглянути усі запущені застосування, зупинити або взагалі видалити з автозапуску непотрібні, отримати інформацію про кожного (ім'я, версія програми, ключ реєстру тощо) або натисненням кнопки

«Full Report» отримати звіт про усі програми за один раз. При першому запуску Startup Programs переглядає список програм, що автоматично запускаються, і при його зміні попереджає про це користувача.

- IE Helpers - контроль за Browser Helper Objects, тобто інформація про наявні об'єкти, запит на установку нових об'єктів, видалення підозрілих об'єктів. Слід сказати, що Browser Helper Objects запускаються кожного разу разом з Internet Explorer. Тому при їх допомозі можна без проблем стежити і за користувачем, що і застосовується в програмах типу SpyWare.

- Scheduled Task Monitoring - відображення запланованих завдань, контроль над додаванням нових і отримання додаткової інформації про плановані роботи.

- Services дозволить відключити або тимчасово зупинити запущені сервіси і отримання додаткової інформації про них.

- View Active Tasks - отримання інформації про запущені на даний момент прцеси, програми і завдання, знищення і призупинення непотрібних і підозрілих. Також за допомогою цього пункту можна розібратися в роботі запущеної системи, тобто дізнатися, для чого призначена та або інша програма.

- Cookies - інформування про появу нових Cookies, відхилення, управління і перегляд інформації записаної в Cookies для ухвалення рішення. Можна також скласти правило фільтрування для Cookies, які завжди повинні видалятися (Cookies with Nuts).

- Options - установка опцій самої програми. Тут же встановлюється контроль за підміною домашньої сторінки у веб-браузері, реакція на зміну файлу hosts або його повне блокування, виведення повного звіту по системі, ведення історії змін.

- File Types - дозволяє переглядати, контролювати і відновлювати змінену асоціацію додатків з розширеннями файлів.

- PLUS – допомога від мережевої бази даних користувачам розібратися в списку незнайомих назв, які використовує утиліта.

Suricata є багатофункціональним та потужним інструментом для проактивного захисту комп'ютерних мереж. Створена під егідою Фонду відкритої інформації безпеки, ця платформа функціонує як багатопотокова система виявлення/запобігання вторгнень, що здійснює детальний аналіз мережевих пакетів, постійний моніторинг та всебічне дослідження протоколів.

Завдяки своїм передовим можливостям Suricata ефективно ідентифікує несанкціоновані втручання та різноманітні шкідливі активності у режимі реального часу, надаючи розширений функціонал для контролю безпеки мережевої інфраструктури.

Програмний комплекс має підтримку численних форматів виведення даних і легко інтегрується з додатковими безпековими рішеннями, що забезпечує надзвичайну гнучкість та результативність при захисті комп'ютерних мереж від загроз. Основні характеристики цієї системи включають:

- багатопоточність: Suricata використовує декілька ядер ЦП для високошвидкісної обробки пакетів, підвищуючи продуктивність і ефективність своєї роботи;

- визначення і аналіз протоколів: автоматичне виявлення і аналіз безлічі протоколів, таких як HTTP, FTP, SMB, DNS, TLS, тощо;

- витягання файлів: система має функцію витягання файлів з мережевого трафіку для подальшого аналізу і перевірки на наявність шкідливого ПЗ;

- просунуте логування: Suricata пропонує великі можливості логування, включаючи журнали у форматі JSON для легкої інтеграції з SIEM - системами;

- правила обробки: підтримує правила, специфічні для Suricata, а також правила, написані для Snort, що полегшує перехід і налаштування;

- Lua - скрипти: дозволяє використовувати Lua - скрипти призначеної для користувача логіки виявлення;

- логування і виявлення TLS: Suricata може логувати і аналізувати TLS - з'єднання для отримання інформації про безпеку мережі.

Система доступна для інсталяції на різноманітних платформах, серед яких Linux, macOS та Windows. Функціонує у двох ключових форматах роботи: режим виявлення несанкціонованих проникнень (IDS) та запобігання вторгнень (IPS). Варто зазначити, що для IPS режиму доступна спеціальна опція «дюйм в секунду» (NFQ), котра забезпечує більш досконале керування мережевими потоками даних.

Режим виявлення вторгнень призначено для пасивного відстеження й детального аналізу трафіку в мережі. При такій конфігурації програмний продукт діє як спеціалізований датчик, що ідентифікує потенційно небезпечну активність та генерує відповідні попередження без безпосереднього втручання у передачу інформації.

Натомість конфігурація запобігання проникненням забезпечує активну протидію загрозам. За цих налаштувань програмне забезпечення не обмежується лише фіксацією підозрілих дій, але й миттєво застосовує захисні механізми для їхнього блокування безпосередньо під час виявлення.

Просунуті функції і використання:

- виявлення і аналіз протоколів: Suricata автоматично виявляє і аналізує різні протоколи, надаючи деталізовану інформацію;

- витяг і аналіз файлів: Suricata може витягати файли з мережевого трафіку для проведення подальшого аналізу на наявність шкідливого ПЗ або конфіденційних даних;

- оптимізація продуктивності: включає налаштування параметрів потоків, прив'язки і інших параметрів для оптимізації продуктивності системи залежно від конкретного мережевого середовища;

- інтеграція з іншими інструментами: Suricata може бути інтегрована з різними інструментами для розширення функціональності.

Як демонструють результати здійсненого дослідження, Suricata становить собою високопродуктивний та надзвичайно адаптивний механізм для контролю захищеності мережевої інфраструктури, котрий виконує різноманітні функції –

від детектування несанкціонованого доступу до комплексного вивчення мережевих протоколів і екстракції файлових даних. Опанувавши принципи налаштування та ключові інструкції керування, професіонали у галузі інформаційної безпеки здатні результативно впроваджувати й ефективно адмініструвати платформу Suricata задля належного убезпечення власних комп'ютерних мереж від сучасних кіберзагроз.

Snort представляє собою інноваційну превентивну систему з відкритим кодом для захисту від несанкціонованих втручань. Ця технологія застосовує методи виявлення на основі сигнатур та аномалій, здобувши популярність завдяки своїй адаптивності та можливостям взаємодії з додатковими елементами безпеки мережевої інфраструктури. Проте варто зауважити, що платформа може зіткнутися з викликами при опрацюванні значних потоків даних, що певним чином звужує її функціональні можливості у великомасштабних комп'ютерних мережах.

Для протидії несанкціонованому доступу до мережевого середовища Snort використовує комплекс спеціалізованих директив, які допомагають ідентифікувати потенційно шкідливу активність. Застосовуючи ці настанови при аналізі пакетної інформації, програмне забезпечення створює відповідні сповіщення для адміністраторів системи.

Маючи понад п'ять мільйонів скачувань та більше ніж 600 тисяч зареєстрованих користувачів у базі, цей інструмент посідає місце серед найбільш широковживаних рішень безпеки від вторгнень глобального масштабу. Втім, необхідно підкреслити, що функціональність Snort найкраще розкривається при впровадженні у невеликих мережевих структурах із помірним трафіком.

Snort має три основні застосування: як аналізатор пакетів, як tcpdump, як реєстратор пакетів - що корисно для налаштування мережевого трафіку або використання в якості повноцінної системи запобігання вторгненням в мережу. Snort можна завантажити і налаштувати як для особистого, так і для ділового використання.

Після завантаження і налаштування, правила Snort поширюються в двох наборах: «Набір правил співтовариства» і «Набір правил передплатника Snort».

Набір правил передплатника Snort розроблений, протестований і схвалений Cisco Talos. Передплатники набору правил Snort отримають набір правил в режимі реального часу у міру їх випуску для клієнтів Cisco. Набір правил співтовариства розроблений співтовариством Snort і перевірений Cisco Talos. Він доступний усім користувачам безкоштовно.

Зіставлення характеристик між системами Snort та Suricata виявляє суттєві відмінності у продуктивності й можливостях масштабування цих захисних інструментів. Snort переважно націлена на обслуговування компактних мережевих інфраструктур з невеликим обсягом інформаційних потоків, тоді як платформа Suricata становить найбільш раціональний вибір для великогабаритних комп'ютерних мереж, де висуваються підвищені вимоги до пропускних можливостей, швидкісної обробки інформації, безпекових стандартів та оперативного реагування на потенційні кібератаки. Особливою перевагою технології Suricata є здатність самостійно блокувати зловмисні дії без необхідності залучення мережевого адміністратора до процесу.

Norton 360 - програмний продукт, який має потужний модуль пошуку шкідливого ПЗ, широкий спектр інструментів інтернет-безпеки, інтуїтивно зрозумілу онлайн-панель управління і добру підтримкою клієнтів, і усе це за ціною, яка менша, ніж у більшості конкурентів.

Його модуль пошуку дій зловмисників використовує машинне навчання, сучасний поведінковий аналіз і каталог шкідливих програм для ідентифікації шкідливих файлів — від простих вірусів і троянів до просунутого шкідливого ПЗ, у тому числі і шпигунського, вимагачів і криптоджекерів.

Застосовує два методи для захисту КМ. Перший метод полягає в порівнянні характеристик відомих загроз з файлами чи сайтами, з якими взаємодіють користувачі кінцевих точок мережі. Другий метод досліджує поведінку програм для виявлення потенційно небезпечної активності до того, як

загроза стане загальновідомою. За допомогою такого двостороннього підходу Norton трохи ефективніше виявляє нові загрози.

Під час сканування мережі Norton в реальному часі забезпечує виявлення усі шкідливих файлів з мінімальним уповільненням системи і без помилкових спрацьовувань. Час сканування в режимі складає «Швидке сканування» 2 хвилини, а в режимі «Повне сканування» - 12 хвилин. Захист КМ в реальному часі від нових загроз зловмисників системою Norton стабільно складає не менше 99,7 % при 100% відсутності помилкових спрацювань.

Архітектура цієї системи захисту складається з наступних компонентів-модулів:

- фаєрвол, доступний в усіх планах після налаштування;
- веб-захист;
- захист від фішингу;
- VPN (віртуальна приватна мережа) - 30 серверних локацій для розблокування стрімінгових сервісів без обмежень трафіку;
- менеджер паролів - необмежене сховище з унікальною функцією автозаміни;
- оптимізація пристрою;
- хмарне резервне копіювання.
- захист веб-камери (тільки для Windows);
- захист від крадіжки персональних даних (тільки для США) і багато чого іншого.

Norton не потребує участі користувача в процесі виявлення шкідливого ПЗ. Шкідливі файли відправляються напряму до карантину, звідки вони не зможуть заразити пристрої кінцевих точок мережі. Користувач зможе в будь-який момент їх видалити або назавжди залишити в карантині.

Norton має один із найдосконаліших фаєрволів серед усіх систем аналогічного спрямування, адже він пропонує розширені функції безпеки.

Закінчивши розгляд та аналіз знайдених в джерелах інформації систем - аналогів, робимо наступний висновок.

Наявні сьогодні механізми захисту інформаційних активів демонструють результативність у протидії вже ідентифікованим загрозам. Швидкість відповіді антивірусних комплексів після виявлення нового шкідливого коду можна охарактеризувати як недостатньо оперативну, що дозволяє ураження величезної кількості обчислювальних пристроїв. Безсумнівно, антивірусне програмне забезпечення, базоване виключно на сигнатурному принципі детектування, без систематичних оновлень поступово втрачає будь-яку практичну цінність. Саме тому більшість виробників антивірусних рішень інтегрують додаткові захисні технології: евристичний аналіз кодів, ізольоване середовище виконання, емуляцію оточення програми, моніторинг незмінності системних компонентів, аналітику поведінкових патернів.

Найвища продуктивність зазначених технологічних підходів досягається через їхнє об'єднання в комплексну систему безпеки, що забезпечує централізований контроль, мінімальне дублювання функціональності та підвищену адаптивність до викликів цифрового середовища. Відповідно до проведеного дослідження аналогічних систем безпеки, передові методології, що спираються на аналіз поведінки та миттєве реагування, набувають визначального значення у боротьбі проти еволюціонуючих кіберзагроз, включаючи цільові довготривалі атаки та вразливості нульового дня.

Особливу цікавість представляють методи поведінкового аналізу або як їх ще називають - методи проактивного захисту. Дійсно, завдяки можливості перехоплення потенційно небезпечних дій програмного забезпечення в реальному режимі часу, методи проактивного захисту мають цілий ряд істотних переваг, а саме:

- можливість виявлення невідомих і поліморфних вірусів;
- блокування небезпечних і потенційно небезпечних дій;
- активна перешкода проникненню вірусів в систему;

- відсутність необхідності в постійних оновленнях;

- можливість автоматичного «відкату» небезпечних дій (захист системи від помилок оператора).

Водночас, вивчені під час аналітичного огляду системи проактивного виявлення шкідливого програмного забезпечення, попри їхню розширену функціональність, нескладну імплементацію та надійну експлуатаційну поведінку, характеризуються низкою істотних обмежень:

- висока вартість ліцензійних програмних комплексів разом із необхідністю щорічних платежів за оновлення придбаних компонентів робить ці потужні захисні механізми недоступними для більшості звичайних користувачів цифрових технологій;

- потреба у безперервному оновленні вірусних сигнатурних баз комплексних захисних рішень, що також потребує додаткових фінансових витрат, задля ефективного убезпечення від новостворених злоякісних програм;

- закритість архітектури цих систем під час використання клієнтом унеможливлює модифікацію програмного середовища без офіційної санкції компанії-розробника навіть допоміжними службовими утилітами (такі послуги теж вимагають окремої оплати);

-процес виявлення небезпечного програмного коду вимагає запуску тривалої процедури сканування, котра споживає значну частку часових ресурсів та обчислювальних потужностей;

-обмежена гнучкість тиражованих версій програмного забезпечення, адже зафіксовано численні конфліктні ситуації при спробах інтеграції придбаних комплексів проактивного захисту до існуючих безпекових інфраструктур.

Тому, враховуючи зазначене вище, доцільно розробити власну систему проактивного захисту, яка буде вільною від зазначених вище недоліків. В основу побудови системи, яка підлягає розробці, покладемо виявлені в процесі аналізу позитивні якості.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

При розробці системи згідно технічного завдання використаємо сучасну методику проактивного захисту на основі технології поведінкового аналізу.

Проактивний захист - одна із сучасних методик, що втілює якісний стрибок в області протидії вірусним атакам. Суть технології в тому, що аналізу піддаються не дані на носіях інформації, а поводження самих даних - сукупність дій (поведінки), чинених у системі. Якщо вважати якийсь клас даних шкідливим, то можна визначити й характерні риси поводження. На відміну від сигнатурних технологій, системи проактивного захисту попереджають, а не виявляють вже відоме зловмисне програмне забезпечення в КМ. При цьому потенційно небезпечна активність програм блокується.

Поведінковий аналіз – це технологія, яка ґрунтується на аналізі дій програм в КМ та виявленні їх незвичайної поведінки, яка може вказувати на наявність шкідливої активності. Вона дозволяє оцінювати не лише одиничну дію, але і ланцюжок дій, що багаторазово підвищує ефективність протидії вірусним загрозам

Для поведінкового аналізу використаємо механізми моніторингу системних викликів, мережевих підключень та інших дій, що виконуються в КМ. Це дасть змогу системі визначити, які дії виконує програма, які ресурси вона використовує, і які зміни вона вносить у систему.

Система, що планується до створення у межах виконання випускної кваліфікаційної бакалаврської роботи, має забезпечувати формування цілісного обчислювального середовища комп'ютерної мережі, відповідаючи таким фундаментальним характеристикам:

- надійність: функціонування загальної архітектури повинно здійснюватися автономно від роботи індивідуальних компонентів мережевої інфраструктури та містити механізми відновлення після можливих збоїв;

- масштабованість: архітектурна концепція має враховувати перспективу збільшення кількісних показників об'єктів захисного периметру;

- відкритість: проектування системного комплексу передбачає можливості розширення та модернізації функціональних можливостей без порушення цілісного функціонування обчислювального середовища;

- сумісність: антивірусне програмне забезпечення повинно підтримувати максимально широкий спектр мережевих ресурсів та володіти інструментарієм комунікації з альтернативними програмними рішеннями;

- уніфікованість (однорідність) архітектурних елементів системи.

Моніторинг безпеки мережевої інфраструктури представляє собою процедуру систематичного контролю інформаційних потоків та технологічної екосистеми для виявлення потенційних безпекових аномалій. Подібні індикатори здатні забезпечити технічні підрозділи цінними аналітичними даними щодо актуального стану захищеності комп'ютерної мережі.

На противагу традиційному відстеженню мережевого трафіку, спеціалізований безпековий контроль фокусується передусім на ідентифікації ознак несанкціонованого втручання в комунікаційні процеси. Підхід до спостереження за поведінковими патернами програмного забезпечення кардинально відрізняється від класичних методик аналізу вмісту файлових структур. Така технологія базується на детальному вивченні операційних характеристик запущених програмних модулів, що можна метафорично порівняти із затриманням правопорушника безпосередньо під час здійснення протиправних дій.

Для реалізації механізму перехоплення потенційно небезпечних дій використаємо технологію перехоплення таблиці системних сервісів SSDT за допомогою додаткового драйвера, працюючого на рівні ядра. Ця технологія має ряд істотних переваг. Працюючи на рівні ядра, модуль перехоплення дозволяє здійснювати перехоплення усіх звернень до файлової підсистеми і реєстру навіть

на ранніх стадіях завантаження операційної системи. При цьому усі перехоплені дії можуть бути своєчасно заблоковані.

При розробці ПЗ системи планується використати програмування під ОС Windows з використанням подій Windows та елементів ActiveX, що дозволить надати програмній моделі більшої оглядовості та зрозумілості для сучасного користувача, який звик працювати в цій операційній системі.

Для створення програмного забезпечення нашої системи обираємо середовище Delphi - одне з найпотужніших візуальних середовищ розробки, що базується на принципах об'єктно-орієнтованого програмування. Це IDE (Integrated Development Environment) дозволяє розробникам створювати різноманітні програмні рішення на сучасному технологічному рівні, починаючи від простих застосунків та закінчуючи складними інформаційними комплексами для функціонування в локальних мережах та глобальному інтернет-просторі.

Розробка в Delphi здійснюється за допомогою однойменної мови високого рівня, яка характеризується імперативним підходом та суворою статичною типізацією. Ця мова програмування надає розробникам потужний інструментарій для щоденної роботи, забезпечуючи можливість проектування широкого спектру програмних продуктів: від елементарних однофреймових додатків до комплексних систем управління розподіленими базами даних. Варто зазначити, що Delphi бере своє походження від мови Pascal, розробленої Ніклаусом Віртом, і була створена компанією Borland під керівництвом Андерса Хейлсберга.

Серед ключових характеристик мови Delphi можна виділити наступні особливості:

- жорстка типізація: кожна змінна на момент оголошення отримує конкретний тип даних, який залишається незмінним протягом усього життєвого циклу програми. Це запобігає багатьом помилкам на етапі компіляції;

- відсутність автоматичного керування пам'яттю: на відміну від більш сучасних мов програмування таких як Java, C# або Python, розробники мають

безпосередній контроль над розподілом та звільненням пам'яті, що дозволяє оптимізувати використання системних ресурсів;

- візуальна парадигма розробки: інтуїтивно зрозумілий графічний інтерфейс допомагає програмістам бачити результат роботи в режимі реального часу, що суттєво прискорює процес розробки та тестування;

- чіткий синтаксис, що зменшує ймовірність виникнення логічних помилок у коді та підвищує загальну надійність програмного забезпечення.

Delphi також має потужні можливості для роботи з базами даних через компоненти BDE, ADO, dbExpress та FireDAC, інтеграцію з веб-технологіями, підтримку COM/DCOM об'єктів та багатопотокове програмування. Середовище включає компілятор, який генерує високооптимізований нативний код, що забезпечує високу продуктивність готових програм.

Мова програмування Delphi вирізняється низкою суттєвих переваг, які роблять її привабливим вибором для розробників програмного забезпечення:

- надзвичайно ефективна система компіляції, що відрізняється від інших підходів унікальним механізмом одноразової компіляції всіх необхідних модулів. Після початкової обробки, вже скомпільовані компоненти автоматично включаються до проєкту без необхідності повторного перекомпілювання. Це дозволяє суттєво економити час розробника, який зазвичай витрачається на очікування переробки використовуваних бібліотек при кожному запуску програми;

- розвинена екосистема готових компонентів становить основу для швидкої розробки додатків. Використовуючи існуючі елементи інтерфейсу та функціональні блоки, програміст може створювати функціонально складні програмні рішення без написання великої кількості власного коду. Наприклад, розробка текстового редактора з функціональністю, подібною до Microsoft Word, значно спрощується завдяки наявності необхідних компонентів на інструментальній палітрі середовища. Delphi також підтримує інтеграцію спеціалізованих підпрограм-експертів, таких як Expert Interface, які розширюють

можливості середовища розробки шляхом додавання нових пунктів меню та діалогових вікон;

- потужний оптимізуючий 32-бітовий компілятор забезпечує не лише виявлення помилок у коді, але й проактивне попередження про потенційні проблеми через систему підказок. Важливою особливістю є здатність відображати всі виявлені помилки одночасно, що дозволяє розробнику ефективніше виправляти проблеми в коді за один підхід, а не послідовно вирішувати кожну окрему помилку;

- інтуїтивно зрозумілий візуальний конструктор інтерфейсів, що реалізує принцип WYSIWYG (What You See Is What You Get), дозволяє розробникам бачити результати своєї роботи безпосередньо під час проєктування;

- зменшення ризику виникнення типових помилок програмування. На відміну від С-подібних мов, де операція присвоєння є виразом і повертає значення змінної зліва, що часто призводить до помилок у новачків, у Delphi така можливість відсутня, оскільки присвоєння реалізовано як операція, яка не повертає жодного значення;

- кросплатформенність розробки дозволяє створювати програмні продукти для різних операційних систем та апаратних платформ. Сучасні версії Delphi (особливо в рамках середовища RAD Studio) підтримують розробку не лише для Windows, але й для macOS, iOS, Android та Linux.

Унікальною особливістю мови Delphi є гармонійне поєднання засобів для високорівневого та низькорівневого програмування. Тоді як більшість сучасних мов схиляються до однієї з крайностей – або забезпечують зручний високорівневий інтерфейс, жертвуючи низькорівневими можливостями, або зосереджуються на низькорівневому програмуванні мережевих комунікацій з обмеженою підтримкою інтерфейсних функцій – Delphi успішно поєднує обидва підходи. Це досягається завдяки поєднанню об'єктно-орієнтованої парадигми з можливістю прямого доступу до системних API та апаратних ресурсів. Система

модульності Delphi дозволяє розробникам створювати добре структуровані, масштабовані програмні рішення з можливістю повторного використання коду.

При розробці ПЗ будуть використані стандартні модулі: Menus- Messages, Dialogs, SysUtils, Classes, Variants, Variant, Graphics, Forms, тощо. Використання стандартних бібліотек дозволить значно полегшити побудову деяких частин програми та скоротити обсяги програмного коду.

У процесі розробки архітектурної концепції нашої системи передбачається імплементація ієрархічної модульної організації: центральний керуючий компонент з диспетчерськими функціями та комплекс функціональних підпрограм двох категорій – універсального та спеціалізованого характеру.

Функціональні блоки універсального характеру забезпечуватимуть реалізацію базових обчислювальних можливостей персонального комп'ютера в контексті потреб системи. Їхній функціонал охоплюватиме: алгоритми інформаційного пошуку в різноманітних структурах даних, механізми кодування та шифрування інформації, алгоритмічні процедури контролю цілісності даних, трансформацію інформаційних потоків між різними форматами, а також генерацію та структуроване представлення результуючих даних для подальшого використання компонентами керування та моніторингу системи.

Блоки спеціалізованого характеру фокусуватимуться на забезпеченні захисту від некоректних дій з боку оператора системи та оптимізації функціонування програмного комплексу в нестандартних операційних ситуаціях. Внутрішня структура цих функціональних блоків формуватиметься з окремих програмних модулів – автономних компонентів з чітко окресленою функціональністю та визначеними інтерфейсами взаємодії. Такий підхід забезпечує високу масштабованість системи та можливість незалежного тестування окремих компонентів.

Важливим компонентом розроблюваної системи стане графічний користувацький інтерфейс, який забезпечуватиме інтерактивну взаємодію між оператором та програмним комплексом. Проектування інтерфейсу

відбуватиметься з урахуванням наступних ергономічних принципів та функціональних вимог:

- збалансованість між функціональною повнотою інтерфейсу та інтуїтивністю його використання, щоб забезпечити комфортну роботу навіть для користувачів з обмеженим досвідом взаємодії з подібними системами. Це досягатиметься через логічну категоризацію функцій, використання уніфікованих елементів керування та візуальних підказок;

- мінімізація надлишкових візуальних елементів та відмова від перевантажених дизайнерських рішень на користь функціональної простоти, що відповідає цільовому призначенню системи. Інтерфейс має фокусуватися на представленні критичної інформації без відволікаючих факторів;

- оптимізація програмного коду, відповідального за функціонування користувацького інтерфейсу, для забезпечення ефективного використання системних ресурсів та миттєвої реакції на дії користувача. Це передбачає використання легковагових компонентів та ефективних алгоритмів відображення даних;

- інтеграція контекстної довідкової системи (HELP), яка надаватиме вичерпні пояснення щодо функціональних можливостей системи як на етапі її запуску, так і в процесі виконання конкретних операцій. Довідкова система включатиме як загальний опис функціональності, так і детальні інструкції щодо виконання специфічних задач.

Додатково варто зазначити, що для оптимізації взаємодії між модулями системи доцільно застосувати патерн "Спостерігач" (Observer), який дозволить забезпечити слабку зв'язність компонентів. Для управління станами системи рекомендується використання патерну "Стан" (State), а для централізованого управління складними операціями – патерн "Команда" (Command).

В якості механізму передачі даних між модулями доцільно використати систему повідомлень (message passing), що забезпечить асинхронну взаємодію

компонентів та підвищить загальну стійкість системи до збоїв. При цьому кожен модуль матиме власний буфер вхідних повідомлень та обробник подій.

Під час виконання ВКБР автором переслідувалася мета – створення дійсно зручної і багатофункціональної системи, яка забезпечить якісне виконання основного завдання з урахуванням певних обмежень і вимог.

2.3 Розгорнута постановка завдання

Згідно ТЗ, розробці підлягає ПЗ системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу. Для створення дійсно працездатної системи, яка ефективно забезпечить безпечне функціонування КМ, необхідно виконати наступний обсяг робіт:

- шляхом критичного аналізу та дослідження наявних на сьогодення на ринку програмних продуктів системам – аналогів щодо теми ВКБР, визначити та обґрунтувати актуальність теми дипломної роботи;

- визначити призначення системи і можливі області її застосування, стисло охарактеризувати її основні параметри і можливості;

- провести: огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми ВКБР з метою виявлення їх позитивних і негативних якостей; виконати аналіз переваг та недоліків існуючих рішень та обґрунтувати необхідності розробки системи за темою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти з урахуванням проведеного аналізу;

- вибрати та обґрунтувати: засоби для побудови системи; основні принципи побудови системи; методику та концепцію проектування; середовище розробки та мову програмування для програмного опису розроблених алгоритмів;

- описати хід теоретичної побудови моделі проекту, визначити та описати компоненти системи, розробити структурну схему системи;

- розробити функціональну схему системи та провести її опис;

- визначити взаємодію процесів в системі, розробити діаграму їх взаємодії;

- навести розрахунки і експериментальні матеріали, які підтверджують вірність рішень, наведених у випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти;

- провести розробку і програмний опис алгоритмів системи, які реалізують її функціональність;

- розробити і описати алгоритм захисту розробленого програмного забезпечення від несанкціонованого доступу;

- розробити і описати методику інтеграції компонентів розробленого програмного забезпечення в існуючу апаратну систему;

- зробити короткі висновки по всіх основних етапах розробки системи, відзначити: ступінь новизни виконаної розробки, методику і засоби реалізації проекту, його основні переваги порівнянні з існуючими аналогічними програмними рішеннями.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

При виконанні ВКБР буде розроблене ПЗ системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу (в подальшому - система).

Питання забезпечення безпеки цифрових мережевих інфраструктур набуває критичної значущості в контексті стрімкого розширення інформаційно-комунікаційних технологій у різноманітні галузі людської активності, соціального функціонування та державного управління. Незважаючи на значний арсенал сучасних технічних рішень та програмних засобів захисту, вразливість комп'ютерних систем до зловмисних втручань демонструє тенденцію до зростання, оскільки еволюція кіберзагроз відбувається паралельно з розвитком захисних механізмів, постійно породжуючи інноваційні вектори атак на обчислювальні ресурси та інформаційні активи.

Сучасна кібербезпекова парадигма стикається з множиною викликів, серед яких особливо варто відзначити зростаючу складність мережевих архітектур, що включають хмарні сервіси, туманні обчислення та периферійні обчислювальні вузли. Ця архітектурна неоднорідність створює додаткові поверхні для потенційних атак та ускладнює впровадження уніфікованих безпекових політик.

Визначимо перелік підозрілих команд, які можуть використовувати зловмисники для проникнення в КМ: копіювання файлів в системну директорію; видалення файлів із системної директорії; створення копій віруса вірусом (наприклад, вірус P2P.Worm.* проводить власне тиражування, використовуючи пірингові мережі); запуск програмних додатків, які не санкціоновані переліком дозволених на запуск процесів, тощо.

Здійснений під час переддипломного практичного дослідження всебічний

аналіз викликів у сфері забезпечення безпеки комп'ютерних мереж дав можливість виокремити чотири фундаментальні стратегічні напрямки та послідовні етапи захисних заходів:

- формування периметрових безпекових бар'єрів, спрямованих на раннє детектування потенційних небезпек шляхом впровадження систем виявлення вторгнень, регулярного моніторингу мережевого трафіку та аналізу аномальної активності. Такі системи функціонують як "цифрові сторожі", що постійно сканують вхідні та вихідні інформаційні потоки;

- розробка дієвої системи контрзаходів та блокувальних механізмів, націлених на припинення поширення кіберзагроз до моменту їх активації в інфраструктурі. Ця стратегія передбачає застосування міжмережевих екранів нового покоління, систем запобігання вторгненням та карантинних зон для підозрілих файлів чи процесів;

- впровадження комплексної методології нейтралізації активних загроз та мінімізації негативних наслідків від атак, які подолали попередні рівні захисту. Даний напрямок включає інструменти відновлення системи після інцидентів, створення резервних копій даних та розробку планів забезпечення безперервності бізнес-процесів;

- реалізація проактивних заходів превентивного характеру на основі глибинного аналізу існуючих та потенційних загроз з подальшим впровадженням профілактичних інструментів захисту. Цей етап охоплює регулярне оновлення програмного забезпечення, навчання персоналу принципам кібергігієни та проведення тестів на проникнення для виявлення прогалин у системі безпеки.

Кожен із запропонованих напрямків становить невід'ємну частину цілісної стратегії багаторівневого захисту, що функціонує за принципом "глибинної оборони" та забезпечує комплексну протидію широкому спектру кіберзагроз сучасності.

Перед початком розробки системи було проведено аналіз засобів реалізації систем проактивного захисту КМ на основі поведінкового аналізу та

вирішено, що найбільш доцільно представити її структуру у вигляді шлюзового сервера, який по суті буде комплексною багаторівневою системою проактивного захисту КМ на основі використання технології поведінкового аналізу.

На сьогодення, коли кіберзагрози стають дедалі складнішими, а їх наслідки - все серйознішими, надійний програмний шлюзовий сервер перетворюється з опції в необхідність для будь-якої КМ, яка цінує безпеку даних своїх користувачів та безперервність бізнес-процесів. Інвестиції в такі системи безпеки на даний час розглядаються не як витрати, а як страхування від потенційно катастрофічних наслідків дій зловмисників. В епоху, коли цифрові активи стають найціннішим ресурсом багатьох суб'єктів будь-якої форми власності, програмний шлюзовий сервер буде надійним вартовим, що стоїть між КМ та хаосом цифрового світу.

Інтегроване рішення, яке об'єднає наступні компоненти системи: міжмережевий екран, прямий проксі-сервер, моніторинг мережевої активності, контроль стану відкритих ресурсів та модуль проактивного захисту, дійсно зможе забезпечити комплексний захист КМ від широкого спектру загроз - від масових фішингових кампаній до цільових атак.

Схема безпеки, реалізована в системі, повинна бути відділена від засобів безпеки самої операційної системи, на якій буде реалізована автоматизована система безпеки, у тому розумінні, що збій або уразливість системи безпеки операційної системи не повинні впливати на роботові автоматизованої системи безпеки.

Система, що підлягає розробці в процесі виконання ВКБР, в ході експлуатації повинна забезпечувати замкнуте збереження й передачу даних, пов'язаних з роботою користувачів КМ (відповідно - модулів інших систем, системних і прикладних даних, тощо) таким чином, щоб:

- неможливо було одержати логічний доступ до зазначених даних поза рамками роботи системи;

- будь - які переміщення даних у системі відбувалися під контролем системи безпеки.

В даний час проактивні технології є важливим і невід'ємним компонентом антивірусного програмного забезпечення. Більш того, як правило, в антивірусних продуктах використовується поєднання відразу декількох технологій проактивного захисту, наприклад евристичний аналіз і емуляція кодів успішно поєднуються з поведінковим аналізом, що дозволяє багато разів підвищити ефективність сучасних антивірусних продуктів проти нових, все більш і більш витончених шкідливих програм.

Технологія аналізу поведінки шкідливого ПЗ ґрунтується на перехопленні всіх важливих системних функцій, що дозволяє відстежувати всю активність в КМ. Технологія поведінкового аналізу дозволяє оцінювати не тільки одиничну дію, але і ланцюжок дій, що в багато разів підвищує ефективність протидії вірусним погрозам.

Модуль проактивного захисту (МПЗ) на основі використання поведінкового аналізу введений до складу шлюзового сервера для випередження потенціальних загроз, які несе шкідливе програмне забезпечення. У світі, де щодня з'являються нові кіберзагрози, реактивного захисту вже недостатньо. Модуль проактивного захисту - це технологічно найскладніший компонент шлюзового сервера, який буде працювати на випередження, запобігаючи атакам ще до їх початку.

Використовуючи технологію, поведінкового аналізу та актуальні дані про загрози з глобальних центрів безпеки, цей модуль здатен виявляти ознаки підготовки до атаки та блокувати підозрілу активність. Проактивний захист включає також автоматичне налаштування правил міжмережевого екрану та налаштування системних параметрів окремих сегментів мережі у відповідь на виявлені загрози. Це дозволить мінімізувати людський фактор у процесі реагування та значно скоротити час від виявлення загрози до її нейтралізації.

МПЗ буде вбудований в програмно реалізований шлюзовий сервер як окремий модуль. Вхідній Internet - трафік для детального аналізу буде надходити в МПЗ. Подальшу роботу системи проактивного захисту можна поділити на етапи:

- вхідний потік даних надходить на шлюзовий сервер;

- використовуючи алгоритми аналізу інформації, МПЗ аналізує вхідний потік даних на наявність погроз;

- усі знайдені погрози надходять до користувацьких бібліотек, де за існуючими правилами буде визначений ступінь погрози вхідних даних;

- після проведеного аналізу пакетів з підозрілою поведінкою модулем проактивного захисту, результат пошуку за правилами і наслідком цього система або перетворить вхідний потік даних у вихідний або проінформує системного адміністратора КМ шляхом створення звіту із занесенням його в журнал про небезпечні вхідні дані.

Крім цього, система повинна забезпечити виведення різноманітної службової інформації про результати обробки системних помилок та мережеві пристрої.

Основні функції в системі, яка підлягає розробці для реалізації політики захисту від МА, буде виконувати брандмауер. Брандмауер забезпечить в системі виконання правил пропуску чи заборони пропуску певних пакетів через мережеву карту/модем/інший пристрій зв'язку. Це буде реалізовано за допомогою мережевої моделі OSI, структуру якої наведена на рисунку 3.1.

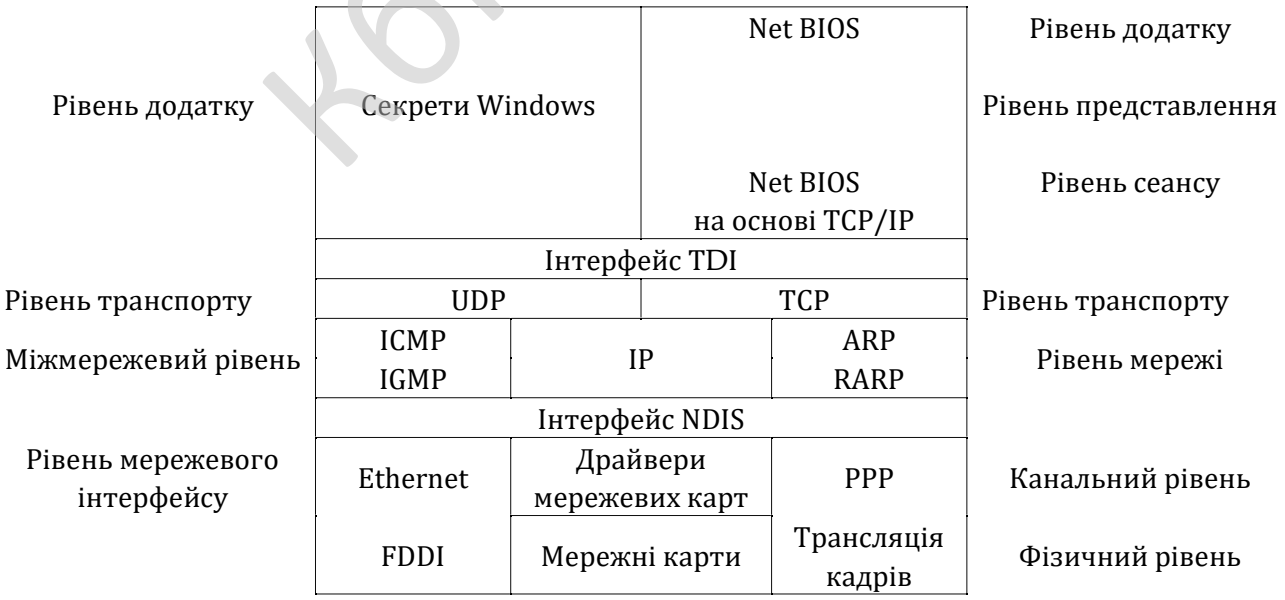

Рисунок 3.1 – Структура мережевої моделі OSI

Кожен пакет формується на рівні додатку при відправці і опускається на рівень мережевого інтерфейсу. При цьому, на кожному рівні до пакету додається свій заголовок. На приймачі здійснюється зворотний процес, і пакет піднімається знову до рівня додатку (тобто, від мережевої карти - до програми).

Реалізація захисту на рівні додатку не раціональна, адже кожна програма тоді повинна буде мати особисту систему захисту, і нема гарантії, що зловмисник не зруйнує таку «стіну». Захищати кожну програму зокрема складно і незручно.

Сенс брандмауера є саме в тому, щоб програма навіть не бачила недозволені пакети або «злі» ПК зловмисників.

Схема надходження пакету на ПК кінцевої точки КМ представлена на рисунку 3.2.

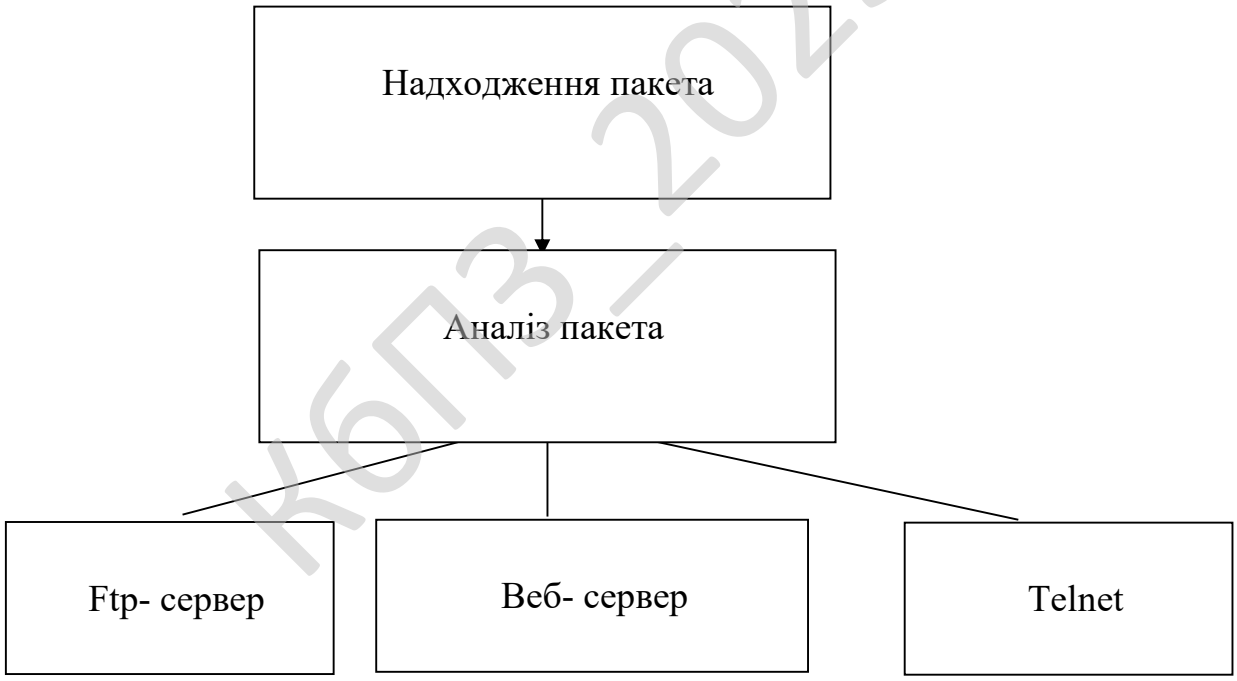

Рисунок 3.2 – Схема надходження пакетів на ПК

Спочатку всі пакети йдуть одночасно і тільки потім розділяються між додатками залежно від порту. Кращий захист від атаки з глобальної мережі реалізується до розгляду пакетів і направлення їх визначеній програмі. Саме до

цього моменту необхідно проаналізувати дані і якщо вони відповідають правилам, надати шанс програмі розпізнати пакет. Коли ж правила забороняють отримання даних від деякого IP, то подальша обробка такого пакету буде призупинена, аде відправником якого є заборонений ПК або порт призначення. Подальше здійснення перевірки після розбірки пакетів на рівні додатку є недоцільним, оскільки від такого захисту користі не буде.

Фільтрація пакетів дозволить кінцевим користувачам КМ виробляти і керувати фільтрами вхідних і витікаючих пакетів. Кожен мережевий інтерфейс буде прив'язаний до одного або декілька фільтрів.

Програмування фільтрації пакетів виконаємо в наступні три етапи:

- створення нового інтерфейсу для додавання або видалення фільтрів до адаптера (слово «інтерфейс» відноситься до фільтрів і не прив’язане до мережевого інтерфейсу);

- встановлення правил інтерфейсу для контролю доступу;

- пов'язання інтерфейсу з IP - адресою.

Дані етапи виконуються за допомогою 3-х функцій: PfCreateInterfase - функція для створення інтерфейсу; PfUnBindInterfase - функція від’єднання інтерфейсу з фільтрами; PfDeletedInterfase - функція видалення інтерфейсу.

Після виконаної програмної реалізації інтерфейсу потрібно додати правила для встановлення заборони чи дозволу на використання певних портів або підключення певних адрес.

Для створення інтерфейсу використаємо функцію PfCreateInterface. Наводимо її опис:

```
function PfCreateInterface( dwName: DWORD;
inActi on: Pfforward_action; OUtAction: PFFORWARD_ACTION; bUseLog: BOOL: bMustBeUmque: BOOL:
var pplnterface: INTERFACE_HANDLE): DWORD; stdcall: external IPHLPAPI name "_PfCreateInterface@24'.
```

Розглянемо параметри функції PfCreate Inter face:

- dwName - ім'я інтерфейсу. Якщо вказати 0, створиться новий унікальний інтерфейс;

- inAction - дія за умовчанням для вхідного пакету. Запис PF_ACTI0N_F0R-WARD означає: пакет буде прийнятий, а PF_ACTI0N_DR0P: пакет буде видалений.

Для серверів мережевий екран буде налаштований так, щоб все, що не дозволене, стало забороненим і за умовчанням видалялось:

- outAction - дії за умовчанням для витікаючого пакету. Тут значення ті ж, що і для параметра nAction, доступ до цих пакетів за умовчанням дозволяється;

- bUseLog - якщо параметр присвоєне значення true, то до інтерфейсу буде прикріплений журнал для легкого визначення активності;

- bMustBeUnique - якщо параметр має значення true, то інтерфейс унікальний і його правила не можуть розділятися з іншими;

- pplnterface - покажчик на створений інтерфейс, тобто вказує на результат виконання функції.

Якщо вона відпрацювала нормально, отримаємо результат N0ERR0R.

Відтак, коли інтерфейс вже реалізовано, долучимо правила для встановлення заборони/дозволу на застосовування певних портів/на підключення з певних адрес. Для цього використаємо функцію PfAddFiltersToInterface, яка описується таким чином:

```
function PfAddFiltersToInterfaceC
ih: INTERFACE_HANDLE:
clnFiIters: DWORD;
pfiltln: PPF_FILTER_DESCRIPTOR;
cOutFiIters: DWORD:
pfiltOut: PPF_FILTER_DESCRIPTOR;
pfHandle: PFILTER_HANDLE
): DWORD: stdcall: external IPHLPAPI name '_PfAddFiltersToInterface@24':
```

Функція PfAddFiltersToInterface має такі параметри:

- і h - покажчик на інтерфейс фільтрів, реалізований на основі функції PfCrea -telnterface;

- clnFi Iters - кількість вхідних правил, занесених в параметр pfiltln;

- pfiltln - покажчик на структуру вхідних правил;

- cOutFi Iters - кількість вихідних правил в параметрі pfiltOut;

- pf і 1 tOut - покажчик на структуру вихідних правил;

- pfHandle – буфер для отримання масиву покажчиків фільтрів. Якщо нетреба встановимо - nil.

При створенні фільтру за допомогою функції PfAddFiltersToInterface, задаємо правила в структурі PFFILTERDESCRIPTOR:

```
_PF_FILTER_DESCRIPTOR = packed record dwFilterFlags: DWORD; dwRule: DWORD;
pfatType: PFADDRESSTYPE: SrcAddr: PByteArray; ' SrcMask:
PByteArray: DstAddr: PByteArray: DstMask: PByteArray: dwProtocol: DWORD;
fLateBound: DWORD: wSrcPort: Word; wDstPort: Word; wSrcPortHighRange: Word;
wDstPortHighRange: Word;
end;
```

Розглянемо основні параметри структури PFFILTERDESCRIPTOR:

- dwFilterFlags - прапорці: підтримується тільки FD_FLAGS_NOSYN;

- dwRule - визначає роль для фільтра;

- pfatType - тип адреси для фільтра, вкажемо PFIPV4 чи PFIPV6;

- SrcAddr, SrcMask і wSrcPort IP - адреса, маска і порт джерела пакету;

- DstAddr, DstMask і wDstPort - IP - адреса, маска і порт одержувача пакету;

- dwProtocol - протокол. вказуємо одне з наступних значень: FILTER_PROTO_ANY - будь-який протокол; FILTER_PROTO_ICMP - протокол ICMP; FILTER_PROTO_TCP - протокол TCP; FILTER_PROTO_UDP - протокол UDP.

Це основні параметри, які слід вказати при створенні нового правила. Решту заповнимо нульовими значеннями.

Створений інтерфейс і правила зв'язуються з мережевим інтерфейсом. Для захисту мережевого з'єднання використаємо функцію PfBindlnterfaceToIPAddress:

```
function PfBindInterfaceToIPAddress(
plnterface: INTERFACE_HANDLE;
pfatLinkType: PFADDRESSTYPE;
IPAddress: PByteArray): DWORD: stdcall: external IPHLPAPI name '
PfBmdInterfaceToIPAddress@12';
```

Розглянемо параметри функції PfBindlnterfaceToIPAddress:

- plnterf acові - покажчик на створений нами інтерфейс;

- pfatLinkType - вказує на тип адреси: PFIPV4B (в даний час використовується 4-байтна IP – адресація) ;

- IPAddress - масив байтів, для визначає IP - адреси інтерфейсу, яку треба захистити.

Щоб відключити захист, потрібно спочатку від'єднати інтерфейс з фільтрами, а потім видалити його. Для від'єднання використовується функція PfUnBindlnterface. У неї тільки один параметр - покажчик на вже створений нами інтерфейс.

Для видалення застосовується функція PfDel etelnterf, яка теж має тільки один параметр - покажчик на інтерфейс, який треба знайти і «знищити».

Для розширення функцій (можливостей) брандмауера, до його складу, в доповнення до ТЗ, необхідно ввести прямий проксі-сервер.

Проксі-сервер, інтегрований у шлюзове рішення, виступає в ролі довіреного посередника між користувачами внутрішньої мережі та зовнішнім світом. Завдяки цьому компоненту користувачі КМ не взаємодіють безпосередньо з інтернет-ресурсами, що суттєво зменшує вектори потенційних атак.

Проксі-сервер в нашій системі забезпечувати виконання наступних функцій:

- забезпечення паралельного доступу користувачів кінцевих точок КМ внутрішньої мережі через виділений канал до зовнішньої мережі;

- маскування IP - адреси ПК.

В зовнішній мережі буде видно тільки адресу сервера, адже запити будуть надаватись тільки від його імені.

На запит клієнта під час роботи, проксі-сервер забезпечить передачу тільки текстового умісту документа. Відтак, коли брандмауер виконає розбірку отриманого документу і знайде картинки, звукові ролики, flach – ролики, тощо, то надасть додатковий запит на отримання цих даних. Отже, брандмауера забезпечить отримання тільки тих даних, які йому потрібні. Це дозволить в процесі експлуатації системи, яка підлягає розробці, значно економити трафік.

Введення до складу системи модуля моніторингу мережевої активності – це дійсно дуже актуальне питання, адже будь-який адміністратор/користувач КМ бажає знати, хто підключиться до його мережі чи ПК, чим займається і де виконує свої зловмисні дії. Таким чином, програмна реалізація цього модуля в процесі виконання ВКБР забезпечить вирішення наступних основних задач з загального функціоналу системи:

- стеження за тим, щоб в системі не було підключень відразу двох адміністраторів якщо такий факт буде виявлений - вислідить та визначить, звідки і хто намагається користуватися такими високими правами на сервері КМ;

- визначення і виведення повідомлення, коли підключення за певними обліковими записами виходять за межі допустимого;

- визначення невідповідності: адреси ПК, який підключився, припустимому в КМ, яка захищається системою.

Для вирішення вищеокреслених задач, необхідно передбачити постійне оновлення інформації про з'єднання і організувати стеження за кількістю підключень з певними правилами (обліковими записами). Ми передбачимо два режими сканування мережі: безперервний та зі встановленим проміжком часу в 1 секунду. На наш погляд, більш доцільним є другий режим, адже якщо хтось встиг за 1 секунду підключитись та вийти з системи, то він, швидше за все, не встиг зробити нічого шкідливого і такі підключення можна ігнорувати.

Тепер виконаємо деталізацію програмної реалізації функції моніторингу мережевої активності. Почнемо з моніторингу підключень. Окреслимо перелік задач та підзадач, виконання яких необхідно при побудові загального алгоритму цієї функції.

Після проведеного дослідження систем-аналогів, було прийнято рішення обмежитись розробкою наступного переліку алгоритмів, які забезпечать в системі:

- відображення імені або адреси ПК, підключеного до КМ, яка захищається системою;

- визначення облікового запису користувача при його вході в КМ;

- визначення кількості файлів, відкритих користувачем;

- визначення кількості часу знаходження користувача в КМ;

- визначення кількості часу знаходження користувача в системі в неактивному стані, коли він підключений, проте не працює з мережею;

- визначення типу клієнта, що підключився до КМ: у цьому полі виводиться ОС клієнта і заповнюється воно тільки для NT-систем;

- визначення протоколу використання.

При підключенні через мережеве оточення візуалізуємо наступне: /Device/NetBiosSmb (заповнюватиметься тільки для NT-систем).

Цього цілком достатньо для отримання з КМ поточних підключень, їх візуалізації та видалення виділеного на екрані монітору адміністратора мережі підключення.

Після проведення моніторингу МА активності щодо КМ виявлення несанкціонованих підключень, часу їх підключення, забезпечимо закриття сесій. Для розробки функції закриття сесій**,** слід визначити та обробити такі параметри:

- ім’я клієнтського комп’ютера КМ, на якому потрібно провести закриття сесії; для локального ПК цей параметр буде мати значення «нуль»;

- ім’я клієнтського комп’ютера КМ, з’єднання з яким необхідно закрити, ураховуючи наступне: для NT-систем ім’я буде мати формат UNC (правила універсального іменування), цебто починатися з подвійного слота; для інших систем ім’я вказується без додавання слотів;

- ім’я користувача.

Окрім списку користувачів, будь-якого системного адміністратора, а ще більше будь-якого зловмисника цікавлять ресурси сервера. Якщо зловмисник привласнив права доступу собі і одержав доступ до секретної інформації, про це можна дізнатись надто пізно. Тому необхідно ввести до складу системи програму, яка б невпинно стежила за певними файлами незалежно від дій

користувача КМ та видавала сигнальне повідомлення в разі виявлення в мережі неправомірних дій.

Одержати список відкритих по мережі ресурсів є досить простою задачею. Для цього створимо нове застосування, помістивши на ньому 2-і кнопки для перегляду відкритих файлів і закриття виділених. Розміщення на застосуванні компонента TListView забезпечить відображення списку. Для компонента TListView розробимо колонки з наступними іменами:

- ID - для відображення ідентифікатора;

- шлях - шлях до відкритого файлу;

- користувач - обліковий запис користувача, який відкрив файл;

- блокування - кількість блокувань.

Для реалізації алгоритму перегляду відкритих ресурсів визначимо наступні операції:

- визначення типу ОС, завантаження бібліотеки і пошук функцій NETFil eEnum і NETSessi on Close, необхідних для рішення поставленої задачі;

- визначення списку відкритих файлів на основі використання функцій NETFil eEnumNT/ NETFilEnum;

- заповнення списку ListView набутих значень.

Наступні параметри списку має визначити системний адміністратор мережі, реагуючи на одержані від системи захисту повідомлення:

- ServerName - ім'я сервера для перегляду відкритих ресурсів;

- BasePath - шлях до ресурсів, усередині яких здійснюється пошук: значення 0 – виведеться перелік усіх відкритих ресурсів; адреса теки - будуть показані відкриті ресурси тільки вказаної теки;

- UserName - якщо задане ім'я облікового запису, то надається тільки список ресурсів, відкриті користувачами; для перегляду всіх ресурсів задаємо 0;

- Level - рівень отримання інформації;

- BufPtr - покажчик на буфер повернення масиву структур з інформацією про відкриті ресурси;

- Prefmaxl en - максимальна кількість повернених записів;

- Entri esRead - повернення дійсної кількісті прочитаних ресурсів;

- Total Entries - отримання зведення про загальну кількість записів в КМ;

- Resume_Handl - потрібно почати перерахування;

- закриття відкритих ресурсів.

Визначимо перелік підозрілих команд, які можуть використовувати зловмисники для проникнення в КМ:

- копіювання файлів в системну директорію;

- видалення файлів із системної директорії;

- створення копій віруса вірусом (наприклад, вірус P2P.Worm.* проводить власне тиражування, використовуючи пірингові мережі);

- запуск програмних додатків, які не санкціоновані переліком дозволених на запуск процесів, тощо.

Кібербезпекова платформа для захисту мережевої інфраструктури, розроблена під час виконання випускної кваліфікаційної бакалаврської роботи, гарантуватиме реалізацію наступних критичних функціональних можливостей при введенні в експлуатацію:

- комплексний захисний механізм проти різноманітних кібернетичних атак як із зовнішніх джерел, так і від внутрішніх зловмисників, включаючи виявлення аномальної активності, блокування підозрілих з'єднань та нейтралізацію шкідливого програмного забезпечення;

- забезпечення приватності інформаційного обміну незалежно від географічного розташування користувацьких терміналів через впровадження сучасних криптографічних алгоритмів та створення захищених каналів комунікації;

- імплементація багаторівневої авторизаційної моделі з використанням різноманітних методів ідентифікації учасників мережевої взаємодії та їхніх обчислювальних систем для гранулярного контролю привілеїв доступу до інформаційних ресурсів;

- високий рівень відмовостійкості архітектури системного рішення з елементами резервування та автоматичного відновлення після потенційних збоїв.

Програмний комплекс, який розроблений відповідно до технічного завдання, характеризується лаконічністю коду та зрозумілою структурою, інтуїтивно зрозумілим інтерфейсом користувача та вбудованою довідковою підсистемою для оперативної консультативної підтримки операторів під час взаємодії з системою. Саме тому створене ергономічне графічне середовище взаємодії, що відповідає сучасним принципам UX/UI дизайну.

Особлива увага приділялась розробці інформативних та візуально структурованих форматів представлення результатів функціонування системи на дисплеях робочих станцій. Додатково створена система запису подій безпеки (SIEM) для забезпечення централізованого збору та аналізу інцидентів, а також можливість генерації аналітичних звітів згідно з регуляторними вимогами галузевих стандартів інформаційної безпеки у лог файли. Передбачена також реалізація механізмів масштабування системи для адаптації до зростаючої інфраструктури захищеної мережі без погіршення продуктивності або рівня захисту.

Отже, визначивши: складові компоненти майбутньої системи; перелік функцій, виконання яких має забезпечити кожен з компонентів системи яка підлягає розробленню; вимоги, обмеження щодо системи вцілому та кожного її компонента зокрема; шляхи реалізації проектних рішень, маємо необхідні дані для розробки структурної і функціональної схем системи.

3.2 Розробка структурної схеми

В процесі виконання ВКБР розробці підлягає комплексна багаторівнева система проактивного захисту КМ на основі використання технології поведінкового аналізу. Найбільш доцільно представити її структуру, як це було визначено та обґрунтовано в підрозділі 3.1 цієї пояснювальної записки) у вигляді шлюзового сервера.

Шлюзовий сервер (Gateway Server) - це критичний елемент мережевої інфраструктури, що виконує роль посередника між внутрішньою мережею організації та зовнішніми мережами, насамперед - інтернетом. До основних функцій шлюзового серверав КМ віднесемо наступне:

- маршрутизація - перенаправлення трафіку між різними мережами з різними протоколами;

- захист - забезпечення безпеки внутрішньої мережі від зовнішніх загроз;

- контроль доступу - управління правами доступу користувачів до зовнішніх ресурсів;

- оптимізація трафіку - кешування, компресія, фільтрація даних.

В процесі виконання ВКБР розробці підлягає програмний шлюзовий сервер, реалізований на стандартному серверному обладнанні. Він забезпечить виконання в системі наступних функцій:

- централізований контроль мережевого трафіку;

- підвищення рівня безпеки;

- оптимізацію використання каналів зв'язку;

- можливість ведення повного аудиту мережевої активності.

До його складу введемо наступні компоненти, які також підлягають програмній реалізації: брандмауер (мережний екран), прямий проксі-сервер (Forward proxy), модуль моніторингу мережевої активності, модуль моніторингу стану відкритих ресурсів, модуль проактивного захисту

Шлюзовий сервер є першою лінією захисту організаційної мережі і критичним компонентом мережевої інфраструктури, а першою лінією оборони - міжмережевий екран (брандмауер). Він виконує функцію розумного фільтра, який аналізує кожен пакет даних, що намагається увійти до захищеної мережі або вийти з неї.

Інтелектуальні правила фільтрації дозволяють адміністратору КМ детально налаштовувати політику безпеки, відкриваючи доступ лише до необхідних ресурсів і закриваючи вразливі порти. Брандмауер буде в системі, що

підлягає розробці, водночас і захисним щитом, і контрольно - пропускним пунктом, забезпечуючи прозорість та безпеку всіх мережевих з'єднань.

Посередницький форвард-проксі механізм, вбудований як компонент шлюзової архітектури, функціонує в якості авторизованого медіатора комунікацій між внутрішніми користувачами корпоративної інфраструктури та глобальним інформаційним простором. Завдяки подібній конфігурації, учасники локальної мережі уникають прямих контактів із ресурсами всесвітньої павутини, що значно мінімізує потенційні можливості для здійснення кібернетичних нападів. Такий форвард-проксі елемент розташовується на межі перетину локальної цифрової інфраструктури з глобальною мережею, здійснюючи транзитне переспрямування клієнтських запитів до зовнішніх ресурсів.

Використання таких посередницьких серверів дозволяє подолати різноманітні обмеження доступу до веб-ресурсів шляхом транформації оригінальної мережевої адреси клієнтського пристрою на альтернативну, яка не підпадає під діючі обмеження. Додатковою функціональністю виступає можливість тимчасового зберігання інформації з частовідвідуваних веб-сторінок, що істотно покращує швидкодію при наступних зверненнях до них. Ця характеристика набуває особливої цінності в контексті ізольованих корпоративних мереж, де співробітники регулярно звертаються до ідентичного набору інформаційних джерел.

Серед ключових переваг такого технологічного рішення варто відзначити: забезпечення конфіденційності мережевої активності користувачів; можливість обходу різноманітних обмежень доступу; суттєве підвищення рівня захищеності при роботі з веб-ресурсами; селективну фільтрацію вхідних інформаційних потоків; оптимізацію розподілу навантаження на канали зв'язку та загальне прискорення взаємодії з веб-сайтами.

Форвард-проксі також забезпечує централізований контроль над політиками безпеки мережі через єдину точку управління, що спрощує адміністрування та моніторинг. Сучасні реалізації таких систем часто включають

функції поглибленої інспекції пакетів (DPI) для виявлення потенційно шкідливого контенту навіть у шифрованому трафіку через технології SSL/TLS інспекції.

Модуль моніторингу мережевої активності - всевидюче око шлюзного серверу. Неможливо захищати те, що ми не бачимо. Саме тому він є критично важливим елементом шлюзового сервера. Цей модуль забезпечує повну видимість усіх процесів, що відбуваються в КМ, дозволяючи адміністраторам виявляти аномалії та потенційні загрози в режимі реального часу. Аналіз трафіку, виявлення незвичайних патернів використання мережі, ідентифікація неавторизованих пристроїв - все це завдання модуля моніторингу.

Модуль моніторингу стану відкритих ресурсів - профілактика важливіша за лікування. Кожен відкритий у мережі ресурс - це потенційна точка вразливості. Модуль моніторингу стану відкритих ресурсів забезпечить в системі постійний нагляд за всіма сервісами та додатками, які мають зовнішні інтерфейси. Він регулярно перевіряє їх доступність, продуктивність і, що найважливіше, безпеку.

Це дозволить оперативно виявляти та усувати вразливості ще до того, як ними скористаються зловмисники. Модуль відстежить оновлення безпеки, виявить неправильні конфігурації та проведе регулярні перевірки на наявність відомих вразливостей. Цей превентивний підхід значно знижує ризик успішної атаки на інфраструктуру КМ.

Модуль проактивного захисту введений до складу шлюзового сервера для випередження потенціальних загроз, які несе шкідливе програмне забезпечення. Це технологічно найскладніший компонент шлюзового сервера, який працює на випередження, запобігаючи атакам ще до їх початку.

Отже, визначивши складові компоненти структури системи та окресливши функції, виконання яких вони повинні забезпечити в процесі експлуатації, маємо всі необхідні дані для розробки структурної схеми системи, яка наведена на рисунку 3.3.

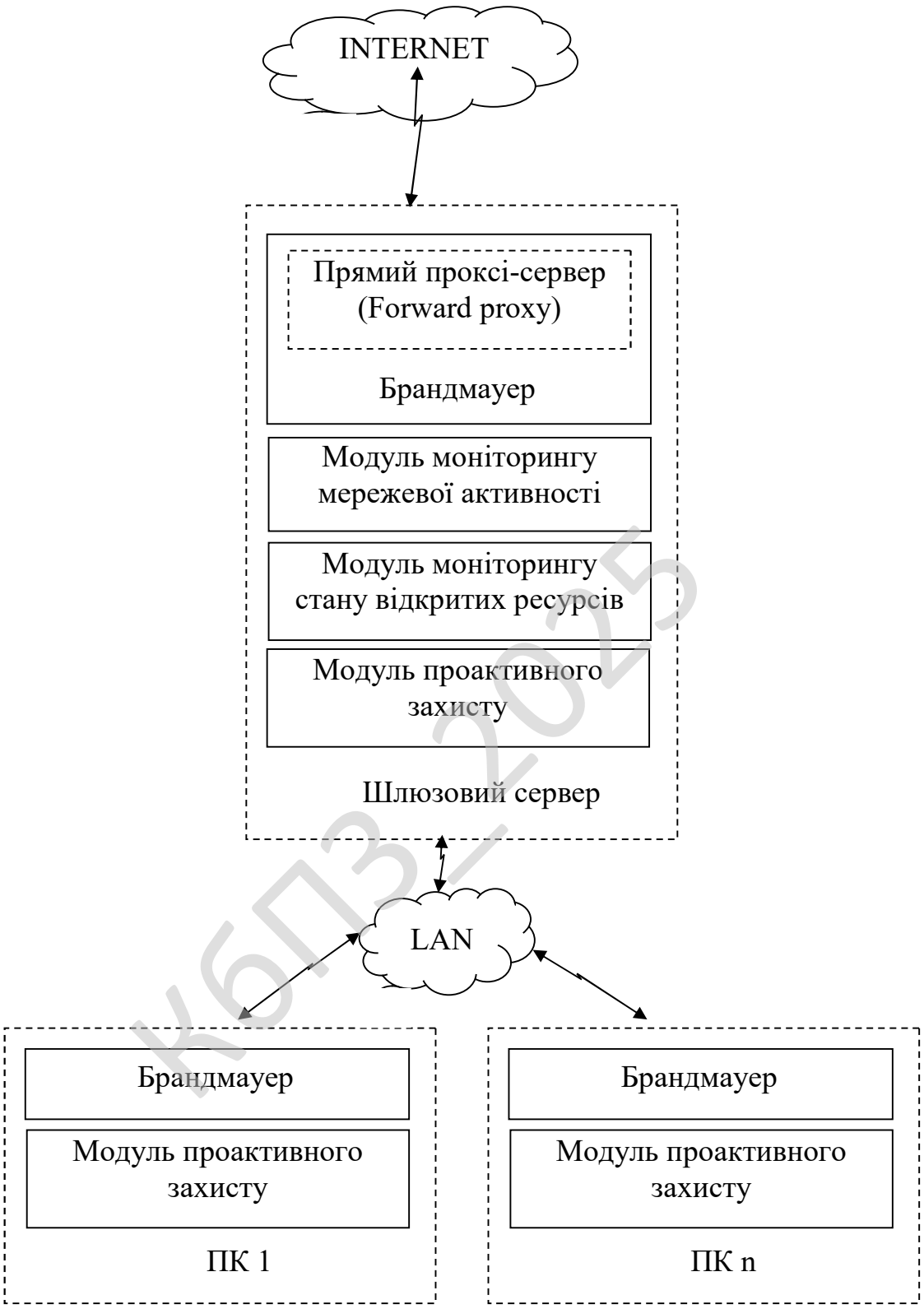

Рисунок 3.3 - Структурна схема системи

При створенні концептуальної основи для побудови майбутньої системи та методів її втілення, головну увагу було зосереджено на швидкодії, інтуїтивній зрозумілості, простоті використання та здатності взаємодіяти з різноманітними локальними операційними системами (LAN), якими користуються клієнти.

Програмний додаток встановлюється на комп'ютер, який буде виконувати функцію шлюзового сервера (повний функціонал) та інші ПК користувачів комп'ютерної мережі (обмежений функціонал).

Відмінність полягає в тому, що на комп'ютері шлюзового сервера задіються всі модулі програми, а також використовуються свої налаштування брандмауера та модуля проактивного захисту. На інших комп'ютерах КМ задіюються модуль брандмауера та модуль проактивного захисту.

Головна перевага розробленої структури полягає не в окремих компонентах, а в їх інтеграції в єдину систему. Коли брандмауер, прямий проксі-сервер, модулі моніторингу та проактивного захисту працюють як єдиний організм, вони створюють багаторівневу систему захисту, яка значно перевершує суму своїх частин.

Наприклад, виявлена системою моніторингу аномалія, може автоматично викликати посилення правил міжмережевого екрану, а підозрілий файл, виявлений проксі-сервером, стає об'єктом поглибленого аналізу модуля проактивного захисту. Така інтегрована реакція забезпечує по-справжньому комплексний захист КМ від сучасних складних кібератак.

3.3 Розробка функціональної схеми

Згідно концепції побудови систем аналогічного класу та спрямування, система, що підлягає розробці, буде побудована по модульному принципу, тобто буде основний, керуючий модуль та підлеглі модулі, кожен з яких виконує певні визначені функції.

Програмне забезпечення, що підлягає розробці, буде виконувати багато різних операцій при роботі з мережею. Тому система повинна мати простий, зрозумілий та дружелюбний інтерфейс. При запуску програми на екран буде виводитись вікно, на якому будуть відображені кнопки з назвами операцій. При натисненні на них, система виконає задані операції та виведе на екран монітора результати своєї роботи: автентифікація користувача; завантаження робочих режимів системи; обробка системних помилок; режим HELP, тощо. Система буде сумісною зі всіма ПК, на яких встановлено ОС Windows

При розробці ПЗ системи плануємо використання API - функцій:

- для роботи з бібліотеками ОС Windows;

- для рішення головної задачі - реалізації в системі функцій проактивного захисту КМ на основі поведінкового аналізу.

Справді для своєчасного виявлення та блокування в КМ розпоряджень на запуск процесів, а не факт появи вже запушених процесів (як це застосовують антивірусні системи сигнатурного аналізу), доцільно здійснювати перехват саме API-функцій, поведінка яких є підозрілою з точки зору нашої системи.

До архітектури системи було інтегровано центральний програмний модуль, який фактично виконує функції системного адміністратора, забезпечуючи протягом всього періоду експлуатаційного використання реалізацію таких важливих завдань:

- створення та підтримка зручного графічного користувацького інтерфейсу для комфортної взаємодії з користувачем;

- комплексне опрацювання виникаючих помилок та збоїв з наданням інформативних повідомлень;

- здійснення запису необхідних конфігураційних даних у системний реєстр для забезпечення коректного автоматичного завантаження компонентів;

- повноцінна ініціалізація та налаштування бібліотеки WinSock для мережевої комунікації.

Для побудови функціональної схеми системи необхідно більш деталізувати функції по кожному компоненту системи та визначити взаємозв’язок між цими компонентами.

Брандмауер забезпечить під час роботи системи виконання наступних функцій:

- фільтрацію пакетів з внутрішньої мережі в зовнішню і навпаки;

- фільтрацію запитів користувачів до зовнішньої мережі та Інтернет - програмних додатків, встановлених на ПК. Правила фільтрації пакетів зберігаються у файлі налаштувань INI-типу і додаються до адаптера в процесі роботи системи, керуючись даними, отриманими з модулів моніторингу (log-файли);

- прямий проксі-сервер маскує IP-адреси ПК внутрішньої мережі.

Спеціалізований модуль контролю мережевої активності безперервно відстежує всі події в комп'ютерній мережі та здійснює детальне протоколювання наступних інформаційних параметрів:

- повна ідентифікаційна інформація щодо облікового запису активного користувача разом з мережевою адресою персонального комп'ютера, який здійснив підключення до системи;

- точна тривалість сеансу перебування кожного користувача в мережевому середовищі та детальний облік кількісних показників відкритих цим користувачем файлових об'єктів;

- тип комунікаційного протоколу, який було задіяно для встановлення з'єднання та передачі інформації;

-автоматична фіксація всіх статистичних даних у спеціалізованому журнальному файлі для подальшого аналізу.

Крім того, система реалізує функціонал моніторингу поточного стану доступних ресурсів через:

- точне встановлення повного навігаційного шляху до кожного відкритого мережевого ресурсу в системі;

- можливість примусового завершення доступу до відкритого ресурсу (за наявності відповідних адміністративних вимог);

- систематичне збереження розгорнутої статистичної інформації у спеціальному журнальному файлі.

Усі зібрані дані та результати функціонування обох моніторингових підсистем зберігаються у структурованих журнальних log-файлах, що забезпечує можливість подальшого глибокого аналізу та аудиту мережевої активності. На основі цих результатів, адміністратор мережі може змінювати або створювати нові правила фільтрації пакетів в брандмауері.

Проактивний захист (моніторинг програмних додатків):

- обробка отриманих повідомлень від користувацької бібліотеки proControl;

- вибір опцій налаштування захисту;

- збереження та читання даних з файлів налаштувань;

- завантаження dll-бібліотек.

Користувачу достатньо означити мітками (на виведеній системою на екран монітору формі) необхідні для роботи параметри, а система в автоматичному режимі з'єднається з реєстром ОС Windows і виконає всі необхідні дії. При цьому помилка встановлення налаштувань фактично буде дорівнювати «0».

Вищезазначені модулі в своїй роботі будуть використовувати дві бібліотеки ОС Windows: Winsock, NetApi 32.dll та дві користувацькі бібліотеки: proControl.dll, proDialog.dll. Користувацькі бібліотеки будуть програмно реалізовані при розробці ПЗ ВКРБ. Визначимо призначення цих бібліотек.

Winsock - це бібліотека, що являє собою інтерфейс мережевого програмування, незалежного від транспортного протоколу. Архітектурно Winsock функціонує як посередницька ланка між прикладним програмним забезпеченням та мережевою інфраструктурою ОС Windows. Концептуально її можна розглядати як інтерпретаційний механізм, що знаходиться між фізичним

мережевим обладнанням комп'ютерної системи та програмними застосунками, які вимагають мережевого з'єднання.

У контексті специфічних функціональних можливостей, Winsock включає такі важливі для нашої розробки елементи:

- функція htonl для трансформації значення типу u_long з формату зберігання байтів, притаманного конкретній обчислювальній системі, у стандартизований мережевий формат TCP/IP.

- функція htons виконання конвертації даних типу u_short з локального формату системи у мережевий порядок байтів, необхідний для коректної передачі через TCP/IP.

WinSock дозволить реалізувати мережеву комунікацію із віддаленими комп'ютерними системами та забезпечити трансфер інформації транспортний протокол TCP для надійного потокового обміну інформацією з підтвердженням отримання

NetApi 32.dll - це спеціальна бібліотека ОС Windows для роботи з мережевими ресурсами. Є важливим компонентом ОС, оскільки містить функції для мережевого адміністрування та забезпечує низькорівневий доступ до мережевих служб Windows.

Основні характеристики та функції NetApi32.dll:

- надає програмний інтерфейс для управління мережевими ресурсами, користувачами, групами та комп'ютерами в КМ;

- зазвичай знаходиться у системній директорії Windows;

- функціональність: управління обліковими записами користувачів та груп; доступ до мережевих ресурсів та керування ними; взаємодія з мережевими серверами та робочими станціями; адміністрування доменів та контролерів доменів; керування спільним доступом до файлів та принтерів.

NetApi32.dll є критичним системним файлом, і його пошкодження або відсутність може призвести до проблем із функціонуванням мережевих служб Windows та програм, які залежать від мережевого API.

Реєстр ОС Windows використаємо для запису даних по автозавантаженню програмного додатку, що дозволить підтримувати у запущеному стані програму на ПК КМ після її першого запуску.

Бібліотека proDialog.dll зберігає наступні графічні форми та формує білий список за допомогою їх інтерфейсу. Дані білого списку записуються у відповідний інформаційний файл, як це показано на рисунку 3.4.

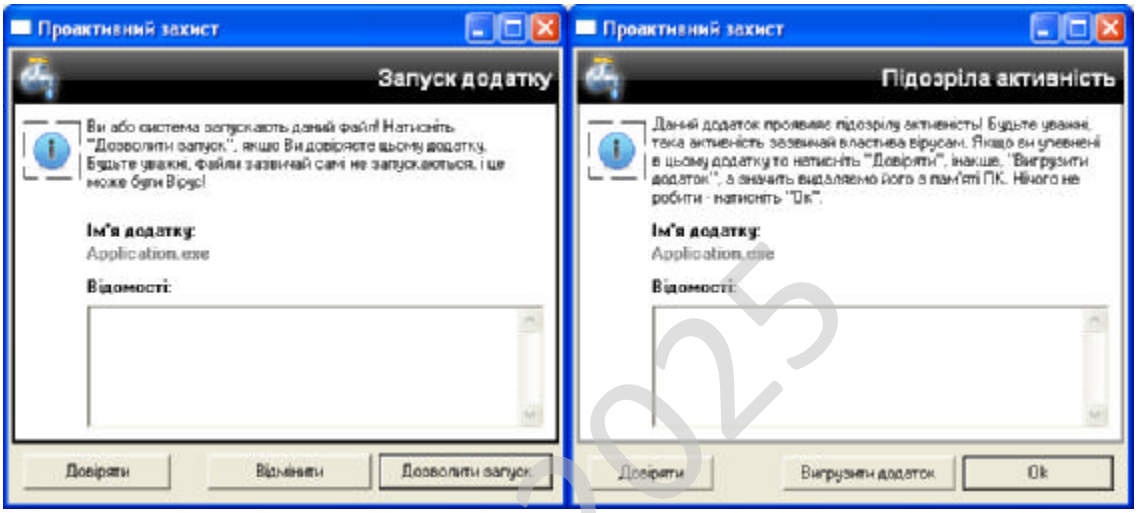

Рисунок 3.4 - Графічного інтерфейсу бібліотеки proDialog.dll

Бібліотека proControl.dll є основним функціональним блоком проактивного захисту. Вона стежить за запуском додатків, копіюванням/видаленням файлів з системної бібліотеки, створенням копій файлів. По суті бібліотека є пасткою системних процесів по вищеописаних подіях з послідуючою їх обробкою і подальшою передачею системі для виконання.

Наводимо перелік подій, які можна відстежити за допомогою бібліотеки proControl.dll:

- створення додатку - завантажника об’єкта через автозапуск;
- створення більше 7 копій програмного додатку;
- створення копії в системній директорії;
- створення підозрілого файлу в системній директорії;
- створення більше 5 файлів, схожих на вірус;

- спроба видалити файл з системної директорії;

- спроба створення файлу, схожого на вірус.

Користувацька бібліотека prodialog.dll реєструється в реєстрі при інсталяції продукта, і завантажується в пам'ять ПК, знаходячись там постійно. Вона виконує функції проактивного захисту.

Графічний інтерфейс програмного додатку спрощує налаштування програми і активність користувацьких бібліотек, обмежуючи їх функції захисту.

Таким чином, маємо всі необхідні дані для побудови функціональної схеми системи, яка надається на рисунку 3.5.

Основні модулі ПЗ системи виконані у вигляді скомпільованих окремих програм і можуть функціонувати окремо одна від одної або разом, доповнюючи функціонал одна одній.

Активація (запуск) модулів може відбуватись в будь-якій послідовності. Модуль моніторингу процесів не потребує додаткових інсталяцій бібліотек і модулів в систему (на відміну від модуля моніторингу програмних додатків.

Для функціонування модуля моніторингу програмних додатків, необхідно встановити користувацьку бібліотеку proControl.dll.

Режим роботи системи залежить від її налаштування. Вона може працювати в трьох режимах: режим налаштування, режим шлюзового сервера, режим клієнта внутрішньої мережі.

Якщо файли налаштувань пусті, то програма завантажується в режимі налаштування системи. В режимі налаштування відбувається налаштування модулів програми (включити/виключити модуль, ввести налаштування по фільтрації пакетів, проксі - серверу, білого списку для проактивного запису тощо), тестування їх роботи (перевірка роботи модулів при вибраних налаштуваннях, при необхідності скидання налаштувань). Всі зміни в налаштуванні системи записуються у файли налаштувань відповідних модулів.

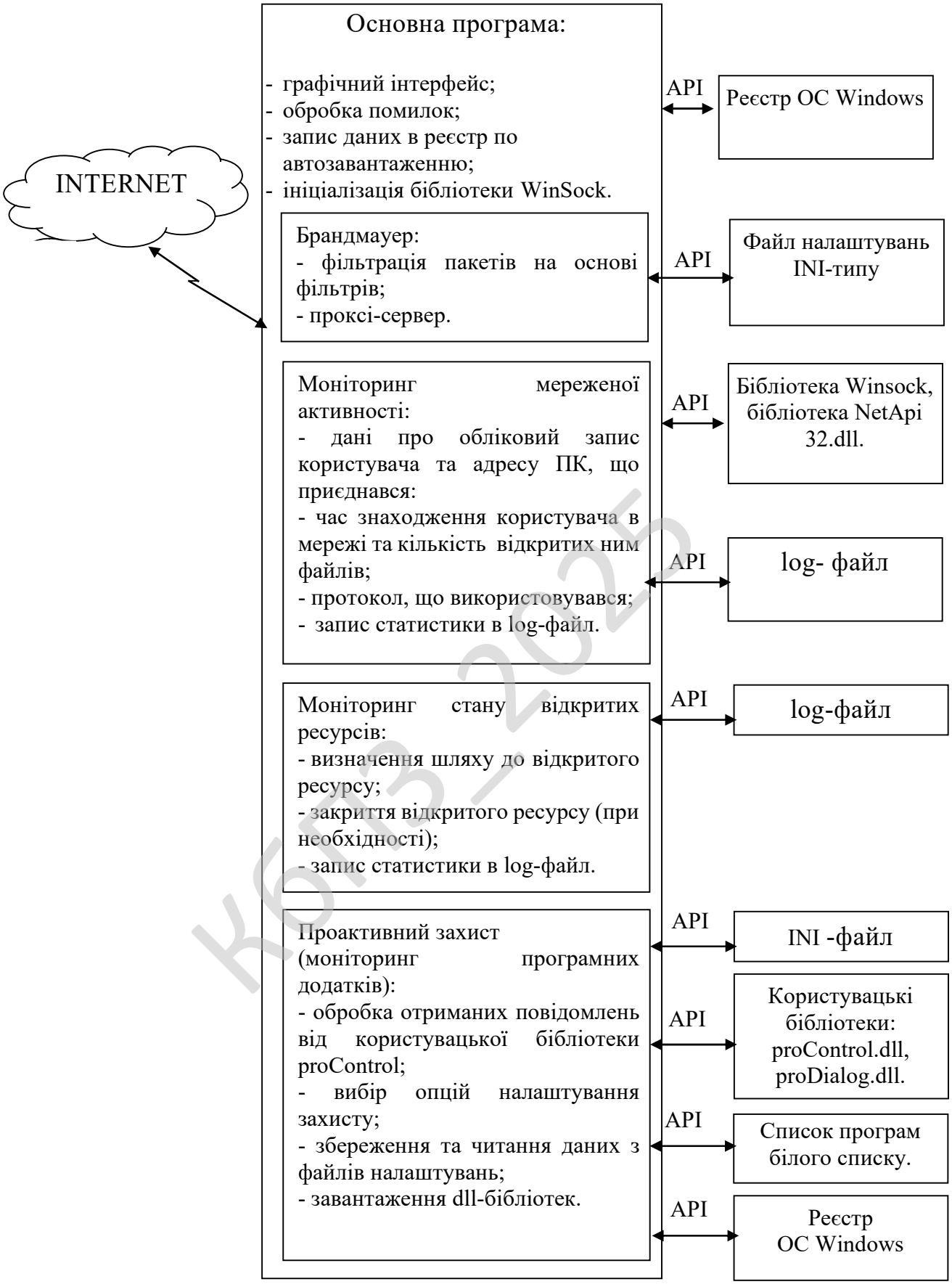

Рисунок 3.5 - Функціональна схема системи

Для налаштування КМ потрібно виконати його як мінімум два рази: перший для шлюзового сервера, другий – для іншого клієнтського ПК мережі. В подальшому можна зберегти файли налаштувань системи і потім їх копіювати на будь-яку кількість ПК внутрішньої мережі. Від них буде залежати, як буде працювати система: як шлюзовий сервер чи як клієнтський ПК мережі.

Після запуску бібліотека proControl.dll прописує себе в реєстрі ОС Windows на автозапуск. Дана бібліотека встановлює програмні пастки на виконання перехоплення API-функцій, створення або відкриття файлу чи каталогу, копіювання файлу з одного місця в інше, видалення файлу.

Обробка кожної з контролюємих функцій виконується наступним чином: спочатку контроль проводиться проактивною системою захисту, а потім передається на виконання її системою, яка підлягає захисту. Тобто, API - функції програмних додатків, що їх викликають, будуть виконуватись, але під контролем нашої проактивної системи захисту.

Якщо дії програмного додатку будуть класифікуватись, як шкідливі, то вони будуть обмежуватись на виконання як в інтерактивному, так і в автоматичному режимах. Даний процес відбувається з повним протоколюванням підозрілих подій, тому можна визначитись з поведінкою додатка і пізніше, переглядаючи його дії з системою.

Отже, розроблені структурна та функціональна схеми системи: визначені основні компоненти та їх призначення; означені функції, виконання яких кожен з них має забезпечити в системі. Переходимо до розгляду взаємодії процесів в системі з наступною розробкою діаграми взаємодії процесів системи.

3.4 Розробка діаграми процесів

Згідно ТЗ на виконання ВКБР проведена розробка діаграми взаємодії процесів в системі (рисунок 3.6).

Спершу завантажується головний процес. Він породжує наступні процеси:

– моніторингу мережевої активності;

– моніторингу стану відкритих ресурсів;

– моніторингу додатків (проактивний захист);

– брандмауера і передає їм керування в системі.

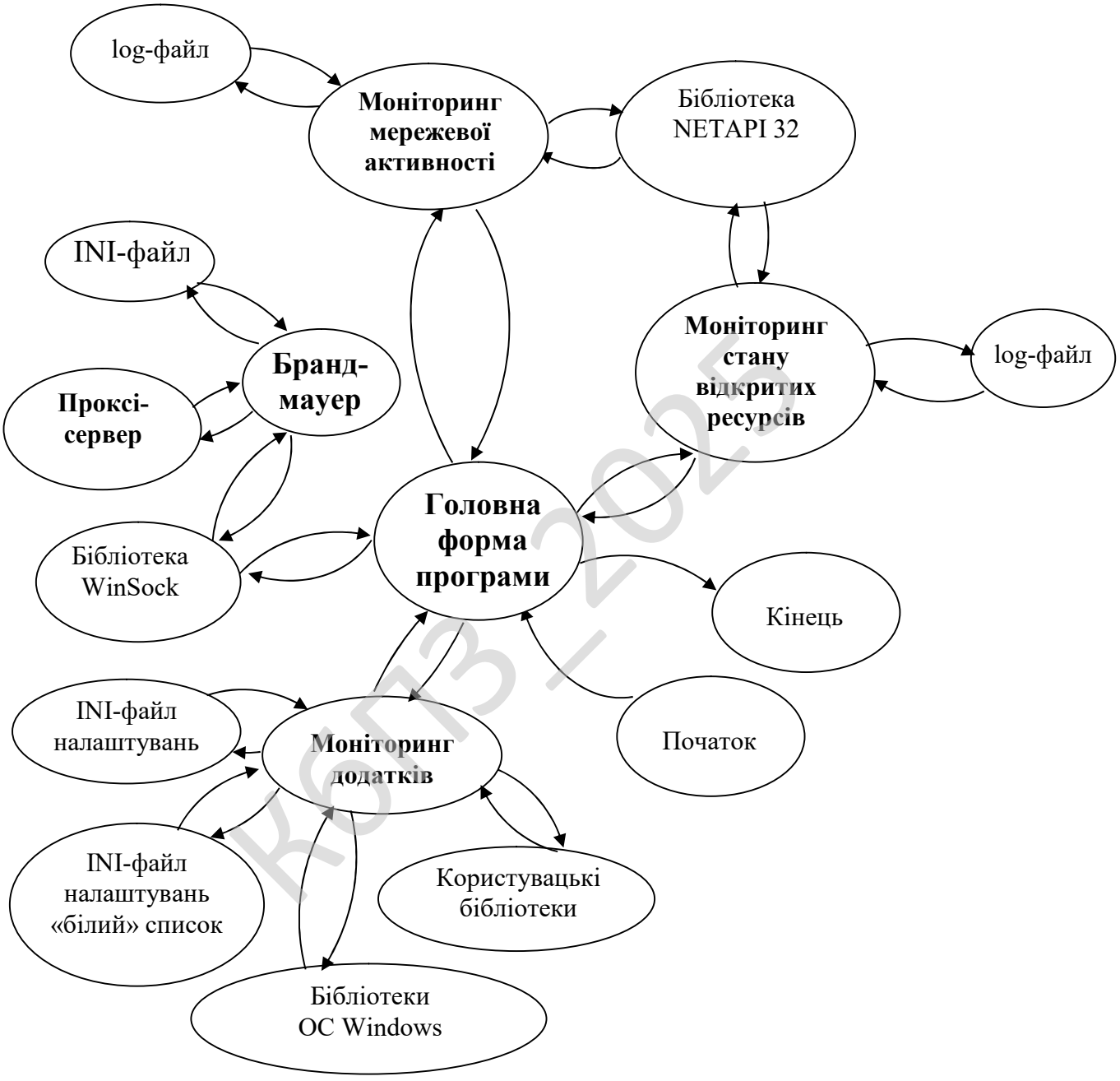

Рисунок 3.6 – Діаграма взаємодії процесів

Процес брандмауера взаємодіє з процесами фільтрації пакетів з внутрішньої мережі в зовнішню (бібліотека WinSock) та процесами прямого

проксі – сервера, забезпечуючи маскування IP-адрес ПК внутрішньої мережі. Після виконання означених функцій, процеси повертаються до головної програми і закриваються.

Процес моніторингу мережевої активності запускає на виконання допоміжні процеси моніторингу мережевої активності (бібліотек NETAPI 32) і після виконання призначених їм функцій повертається до головної програми і закривається. Результати моніторингу заносяться до log - файлу.

Аналогічно процесу моніторингу мереженої активності організований і процес моніторингу стану відкритих ресурсів.

Процес моніторингу програмних додатків (проактивний захист) взаємодіє з допоміжними процесами користувацької бібліотеки proControl.dl, запускаючи їх на встановлення програмних пасток для виконання перехоплення API-функцій: створення процесу, створення або відкриття файлу чи каталогу, копіювання файлу з одного місця в інше, видалення файлу. Після виконання означених функцій, процеси повертаються до головної програми і закриваються.

Активація модулів може здійснюватися в будь-якій черговості.

Узагальнюючи вищезазначене, робимо цілком вмотивований висновок про фактичне завершена теоретичної побудови структури системи. Виконаний наступний обсяг роботи:

- проведений опис функціонування системи;

- визначені стандарті елементи системи та елементи, які підлягають програмній реалізації в процесі розробки ВКБР;

- проведена розробка структурної схеми системи;

- визначені функції компонентів системи, розроблена функціональна схема системи;

- визначені основні та підлеглі їм процеси, їх взаємодія і побудована діаграма взаємодії процесів.

Отже, маємо достатньо інформації для побудови блок-схем алгоритмів та їх опису програмним кодом.

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

Програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу розроблене у відповідності з вимогами побудови систем аналогічного класу та спрямування та є багаторівневою системою реального часу, бо обробка даних ведеться за позначками реального часу.

В процесі промислової експлуатації розроблена система не повинна впливати на роботу других систем захисту комп'ютерної мережі. Недопустимі колізії і під час роботи системи по безпосередньому призначенню з програмними ресурсами мережі. Ці задачі вирішені за рахунок використання при розробці наступних аспектів:

- інтеграція неоднорідних механізмів захисту в єдину систему;
- урахування взаємного впливу окремих механізмів, одночасно працюючих з системою;
- орієнтація на статистику погроз при визначені ключових механізмів моніторингу та проактивного захисту на основі використання поведінкового аналізу;
- ідентифікація та перевірка достовірності суб'єктів доступу при вході в систему по ідентифікатору/паролю/ключу.

4.1 Блок-схеми та опис алгоритмів функціонування системи

При розробці ПЗ системи використане середовище Delphi та програмування під ОС Windows з застосовуванням подій Windows та елементів ActiveX. що надає

програмній моделі більшої наочності та зрозумілості для сучасного користувача, який звик працювати в, цій операційній системі.

Під час роботи системи працює простий, інтуїтивно зрозумілий інтерфейс з користувачем, який в процесі експлуатації системи забезпечить своєчасне виправлення помилок та надасть по запиту користувача, допоміжну інформацію. Інтерфейс також програмно реалізований при виконанні ВКРБ.

Розроблене ПЗ системи забезпечує виконання означених в ТЗ функцій, направлених на виконання основної задачі – забезпечення проактивного захисту на основі поведінкового аналізу від шкідливого програмного забезпечення в КМ.

Програмно реалізовано досить складну багатофункціональну систему. ПЗ системи має модульну структуру побудови. До його складу ввійшли стандартні модулі і модулі, розроблені в процесі розробки ПЗ системи. Програмній реалізації під час розробки ПЗ системи підлягають наступні модулі:

- модуль моніторингу мережевої активності;
- модуль моніторингу стану відкритих ресурсів;
- модуль моніторингу програмних додатків (проактивний захист);
- брандмауер, проксі – сервер.

Керування модулями системи здійснюється основною програмою. Вона за своїм призначенням - диспетчер системи, забезпечуючи виконання наступних функцій: санкціонований доступ до роботи з системою користувачів; завантаження програм-модулів, допоміжних модулів та керування їх роботою: запуск, закриття; ведення режиму допомоги користувачу щодо налагодження режимів її роботи та роботи по прямому призначенню; обробка системних помилок, надання інформаційних повідомлень користувачу.

Основна програма використовує зовнішній графічний інтерфейс. Це забезпечить доступність зовнішнім модулям, які звертаються до неї в процесі роботи, підключених до основної програми модулів, оголошених типів, класів, змінних, констант, функцій і процедур.

Розглянемо блок-схему роботи алгоритму основної програми, яка представлена рисунком 4.1.

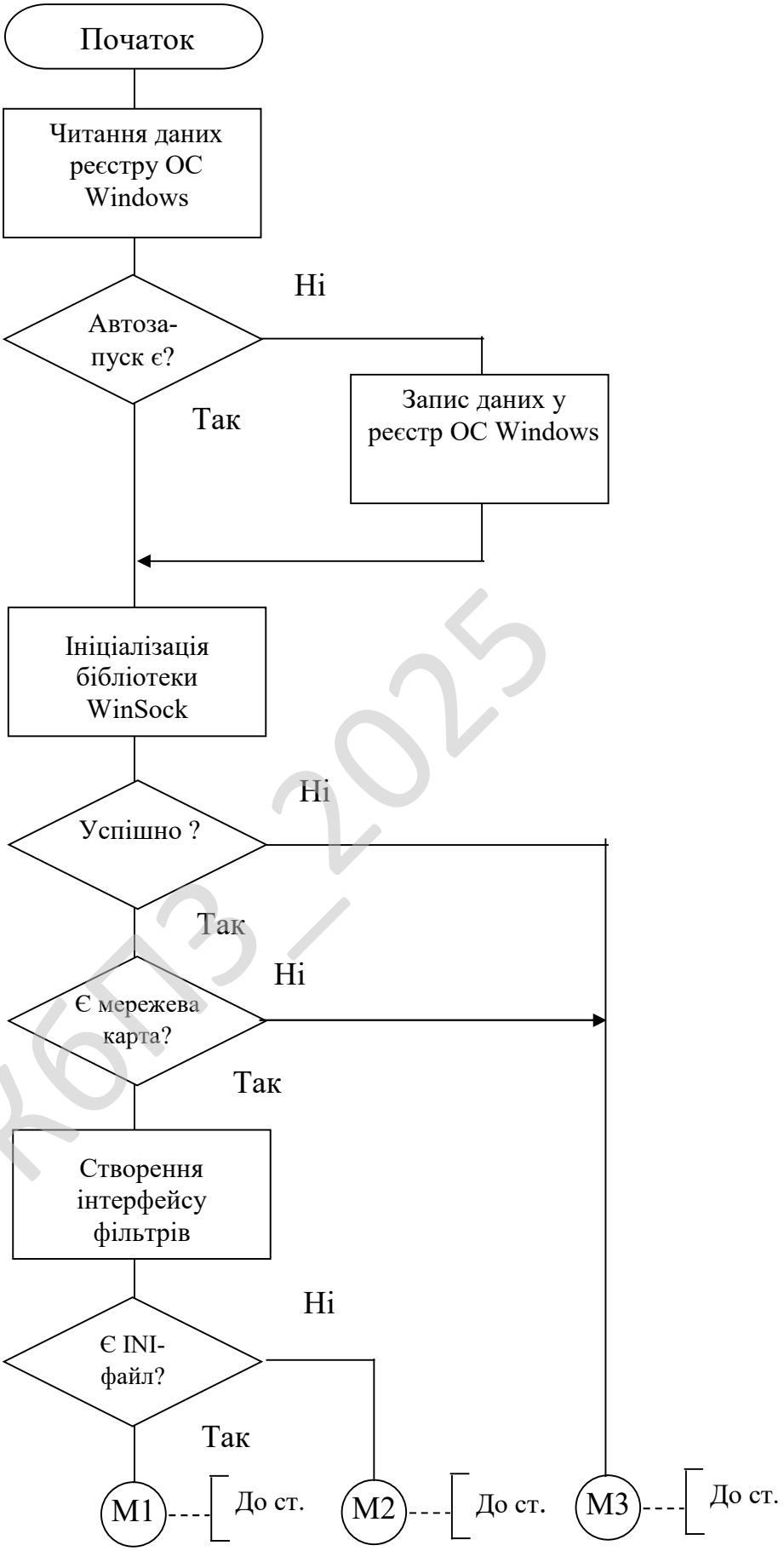

Рисунок 4.1 - Блок-схема алгоритму роботи основної програми

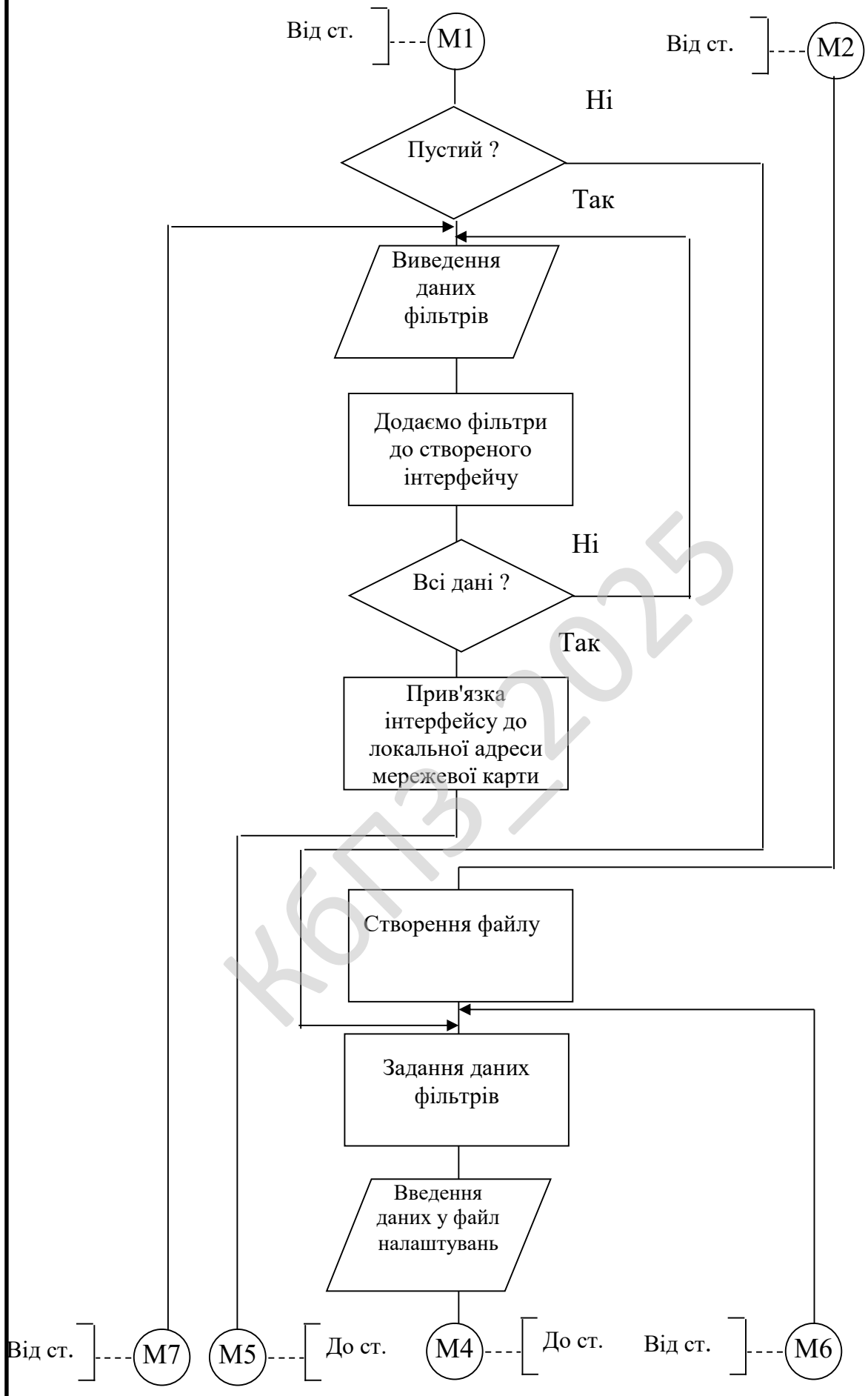

Рисунок 4.1, аркуш 2

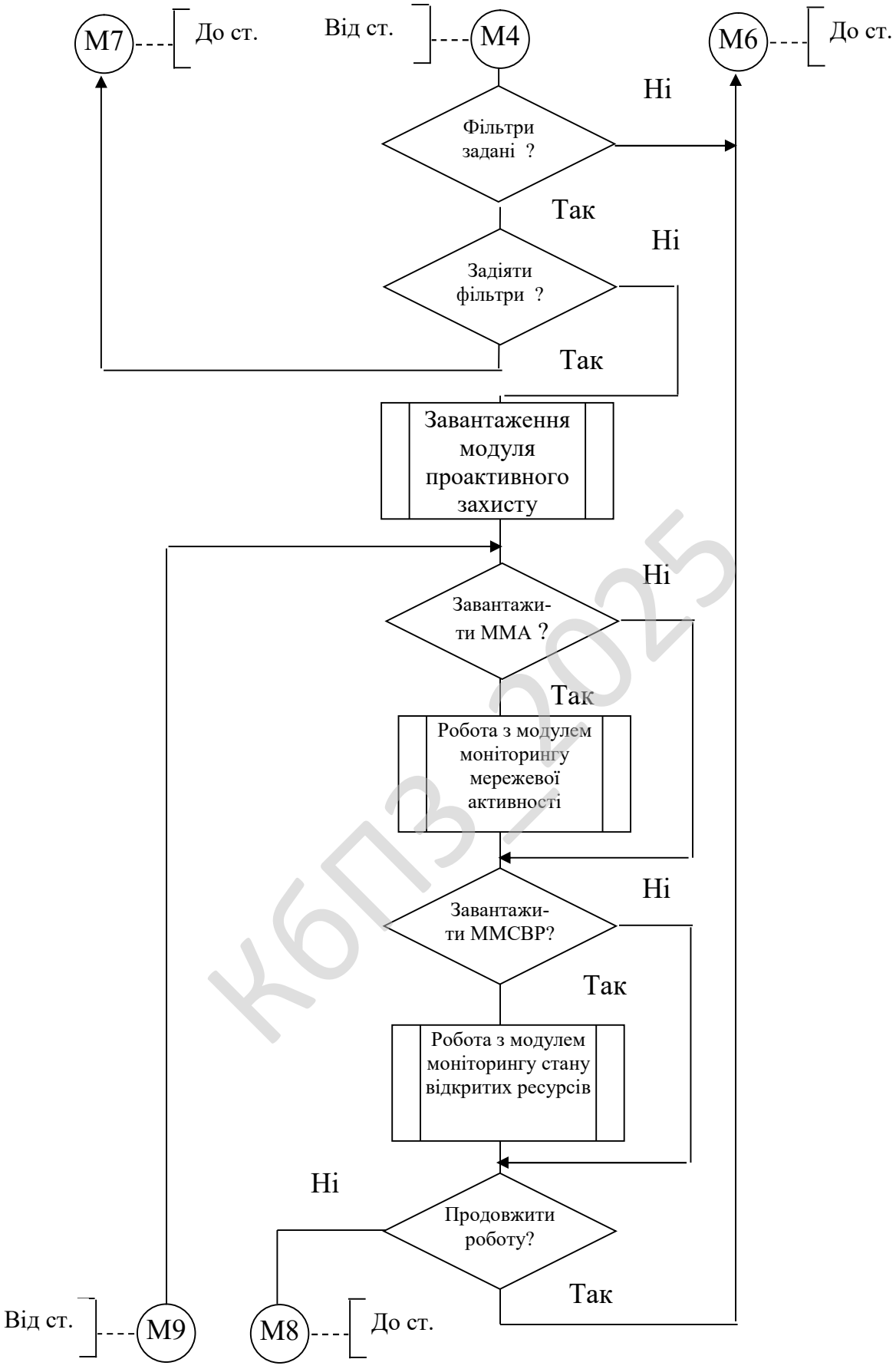

Рисунок 4.1, аркуш 3

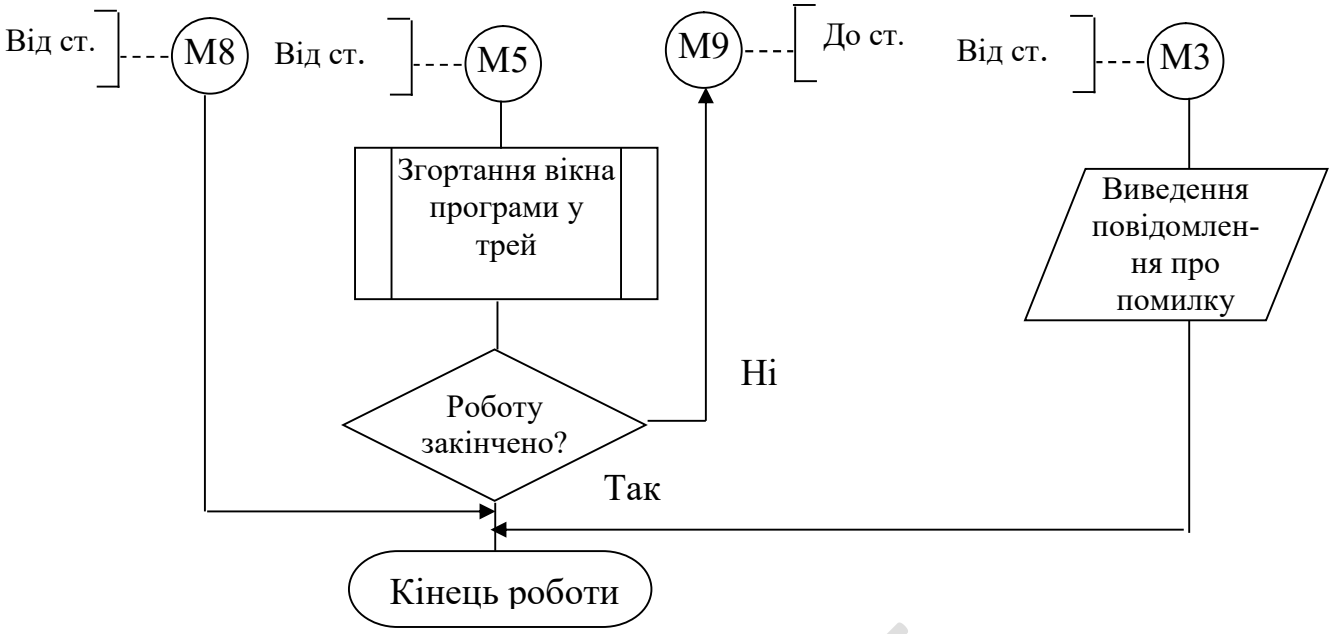

Рисунок 4.1, аркуш 4

Робота основної програми складається з виконання наступних кроків.

Початок роботи програми.

Читання даних реєстру ОС Windows для перевірки наявності факту присутності режиму автозапуску системи. Якщо цей режим відсутній – проводимо запис даних у реєстр ОС Windows і переходимо до ініціалізації бібліотеки WinSock.

Якщо ініціалізація бібліотеки WinSock пройшла успішно – перехід на наступний крок, якщо ні - виведення повідомлення про помилку і завершення роботи системи.

Наступним кроком перевіряємо наявність на ПК мережевої карти. Якщо карта відсутня виведення повідомлення про помилку і перехід на завершення роботи системи.

Створення інтерфейсу фільтрів. Перевірка наявності файла налаштувань типу INI з правилами для брандмауера. Це дозволи на роботу певних портів та програм. Якщо INI-файл є – перехід на наступний крок, якщо ні – завершення роботи системи.

Якщо файл налаштувань типу INI утримує правила для брандмауера - – перехід на наступний крок, якщо ні – задаємо ці правила і вводимо їх до файлу налаштувань.

Читання даних фільтрів, перехід на наступний крок.

Додавання фільтрів до створеного інтерфейсу фільтрів, перехід на наступний крок.

Перевірка даних фільтрів. Якщо всі дані фільтрів, що підлягають приєднанню до інтерфейсу фільтрів, прочитані – перехід на наступний крок. Якщо ні – повернення до створення.

Прив'язка інтерфейсу до локальної адреси мережної карти, перехід на наступний крок.

Якщо перевірка виявила, що INI-файл (правила брандмауера) відсутній, створюємо INI-файл і перехід на наступний крок.

Задаємо дані (правила) фільтрів. Перехід на наступний крок.

Запис даних у файл налаштувань типу INI, перехід на наступний крок.

Перевірка формування даних фільтрів. Якщо вони задані в повному обсязі – перехід на наступний крок, якщо ні – повернення на попередні кроки до їх введення.

Кінець циклу організації режиму автозапуску формування інтерфейсу ланцюжка фільтрів. Якщо фільтри сформовані – запуск і повернення на крок введення даних фільтрів, якщо ні - перехід до завантаження модуля моніторингу активності. Наводимо фрагмент програмного коду щодо реалізації функції ініціалізації бібліотеки WinSock.

```
procedure TFirewallForm.btStartInFilterClick(Sender: TObject);
var
 wsaData:       TWSAData;
 i,j:           Integer;
 F:             TextFile;
 S, S1,
 FlagInOutSt,
 FlagPacketInOutSt:  String;
 IPString:      String;
 PortString:    String;
 FlagInOut:       Boolean;
 FlagPacketInOut: DWORD;

begin
```

```
//ініціалізація бібліотеки winsock
 if (WSAStartup(MakeWord(1, 1), wsaData) <> 0) then
  begin
   ShowMessage('Помилка Winsock');
   exit;
  end;
//отримати локальний IP
 if not GetLocalIPAddr(@ipLocal) then
  exit;

 //Створення інтерфейсу фільтрів
 PfCreateInterface(0, PF_ACTION_FORWARD, PF_ACTION_FORWARD, False,
True, hIF);

 // Додавання фільтрів вхідних пакетів з шаблону
 //відкриваємо файл фільтрів
 AssignFile(F, 'filter.txt');
 Reset(F);
 //поки не досягнуто кінця файлу
 while not Eof(F) do
   begin
     //читаємо рядок
     ReadLn(F, S);
     //видаляємо лишні символи
     S := TrimLeft(S);
     //якщо рядок не пустий
     if S<>'' then
       begin
         j := Pos(',', S);
         //якщо є маркери
         if j > 0 then
           begin
             //визначаємо, який фільтр вхідний чи вихідний
             FlagInOutSt:=Copy(S,0,j-1);
             if AnsiLowerCase(FlagInOutSt)='true' then  FlagInOut:=True;
             if AnsiLowerCase(FlagInOutSt)='false' then  FlagInOut:=False;
             delete(S,1,j);
             //визначаємо машину, трафік якої блокується
             j := Pos(',', S);
             IPString:= Copy(S,0,j-1);
             if AnsiLowerCase(IPString)<>'nil' then  IPString:=''+IPString+';
             delete(S,1,j);
             //визначаємо тип пакетів, який блокується
             j := Pos(',', S);
             FlagPacketInOutSt:= Copy(S,0,j-1);
             if FlagPacketInOutSt='FILTER_PROTO_ANY' then
FlagPacketInOut:=$00
             else
             if FlagPacketInOutSt='FILTER_PROTO_ICMP' then
FlagPacketInOut:=$01
             else
             if FlagPacketInOutSt='FILTER_PROTO_TCP' then
FlagPacketInOut:=$06
             else
             if FlagPacketInOutSt='FILTER_PROTO_UDP' then
FlagPacketInOut:=$11
             else
             if FlagPacketInOutSt='FILTER_TCPUDP_PORT_ANY' then
FlagPacketInOut:=$00
             else  FlagPacketInOut:=$00;
             delete(S,1,j);
             //визначаємо порт, по якому блокується трафік
             j := Pos(',', S);
```

```
                PortString:= Copy(S,0,j-1);
                if AnsiLowerCase(PortString)<>'nil' then
PortString:=''+PortString+'';

                //ввімкнути фільтр згідно вказаних у файлі даних
                if (AnsiLowerCase(IPString)='nil') and
(AnsiLowerCase(PortString)='nil')
                then  AddFilter(FlagInOut, nil, FILTER_PROTO_TCP, nil)
                else
                if (AnsiLowerCase(IPString)='nil') and
(AnsiLowerCase(PortString)<>'nil')
                then  AddFilter(FlagInOut, nil, FlagPacketInOut,
PChar(PortString))
                else
                if (AnsiLowerCase(IPString)<>'nil') and
(AnsiLowerCase(PortString)='nil')
                then  AddFilter(FlagInOut, PChar(IPString), FlagPacketInOut,nil)
                else  AddFilter(FlagInOut, PChar(IPString), FlagPacketInOut,
PChar(PortString));
              end;
        end;
    end;

 // Прив'язати інтерфейс до локального адреса мережевої карти
 PfBindInterfaceToIPAddress(hIF, PF_IPV4, @ipLocal);

 CloseFile(F);
 btStopFilter.Enabled:=true;
end;
```

Продовжуємо опис алгоритму роботи основної програми. Наступним кроком буде завантаження модуля моніторингу мережевої активності. Якщо завантаження відбулось – перехід на наступний крок, якщо ні – перехід на завантаження іншого модуля.

Робота з модулем моніторингу мережевої активності, перехід на наступний крок.

Завантаження модуля моніторингу стану відкритих ресурсів. Якщо успішно – перехід на наступний крок , якщо ні – вихід на кінець роботи.

Робота з модулем моніторингу стану відкритих ресурсів, перехід на наступний крок.

Якщо робота не закінчена – повернення до формування файлу фільтрів, якщо так – перехід на кінець роботи програми.

Згортання вікна у трей, перехід на кінець роботи програми.

Кінець роботи програми.

При необхідності програму моніторингу можна розгорнути за допомогою контекстного меню. При спробі закрити програму-монітор вона згортається в Tray, а повне її закриття можливе лише з контекстного меню в Tray. Наводимо програмну реалізацію фрагментів процедур по роботі з SystemTray.

Зазвичай, в режимі шлюзового сервера задіяні всі модулі програми, а в клієнтському режимі відключаються модулі: моніторингу мережної активності та моніторингу стану відкритих ресурсів. Налаштована основна програма читає налаштування і поступово завантажує налаштовані модулі.

По функціональному призначенню одним із найважливіших модулів розробленої системи є модуль проактивного захисту (рисунок 4.2). Саме він в процесі роботи системи, взаємодіє з допоміжними процесами користувацької бібліотеки proControl.dl, запускаючи їх на встановлення програмних пасток для виконання:

– перехоплення API-функцій,

– створення процесу;

– створення, відкриття або видалення файлу чи каталогу;

– копіювання файлу з одного місця в інше.

Система має графічний інтуїтивно зрозумілий інтерфейс та забезпечує високий ступінь візуалізації. В системі також необхідно забезпечити і роботу невидимих модулів (в першу чергу –модуля проактивного захисту).

Одним з невидимих модулів в системі є робота dll-бібліотек. Бібліотека proControl.dll є основним функціональним блоком проактивного захисту. Вона стежить за запуском додатків, копіюванням/видаленням файлів з системної бібліотеки, створенням копій файлів. По суті бібліотека є пасткою системних процесів, оскільки забезпечує на протязі всього робочого циклу системи виконання функції перехоплення API-функцій системних подій по вищеописаних подіях з послідуючою їх обробкою і подальшою передачею системі для виконання.

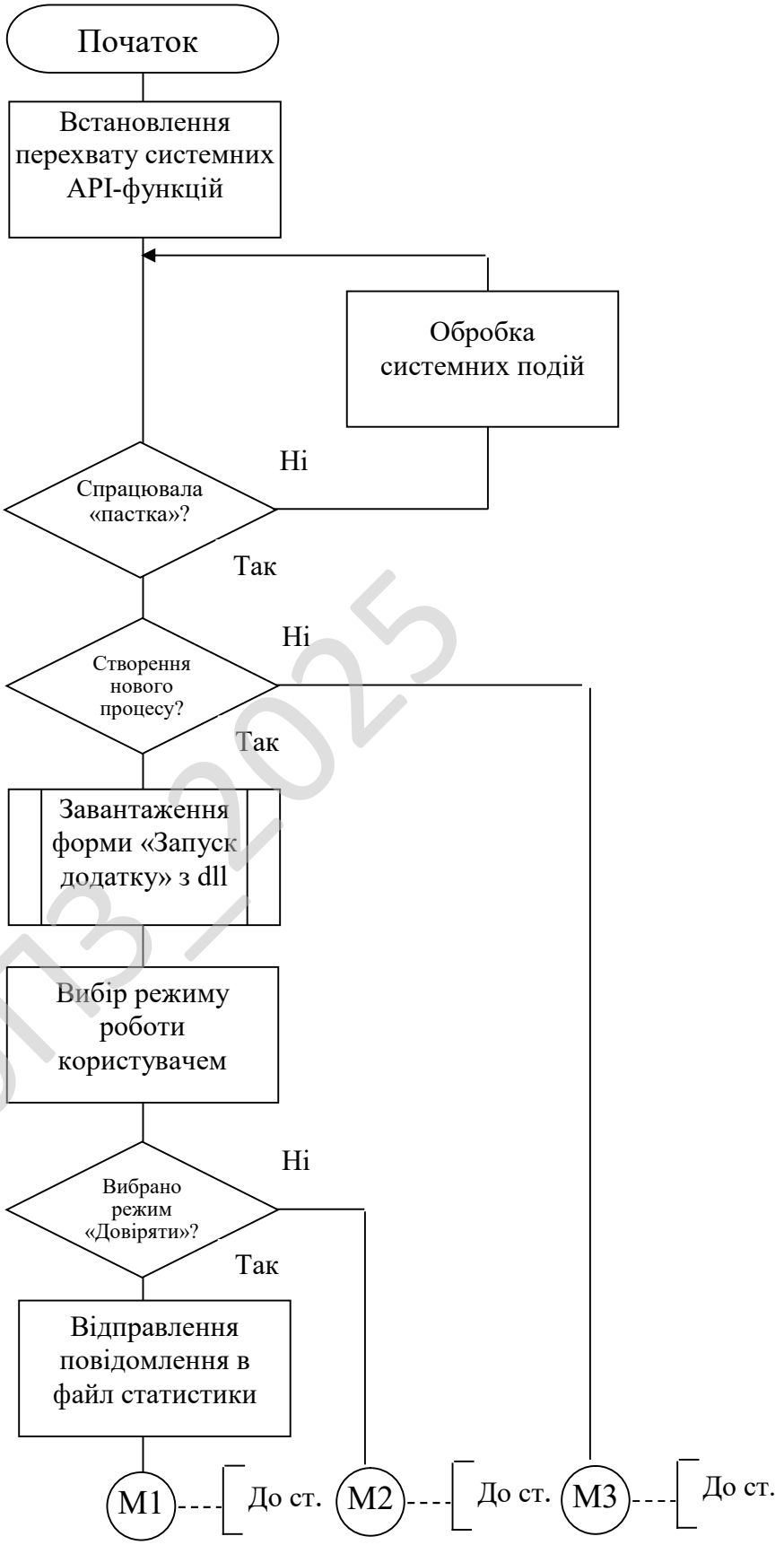

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми

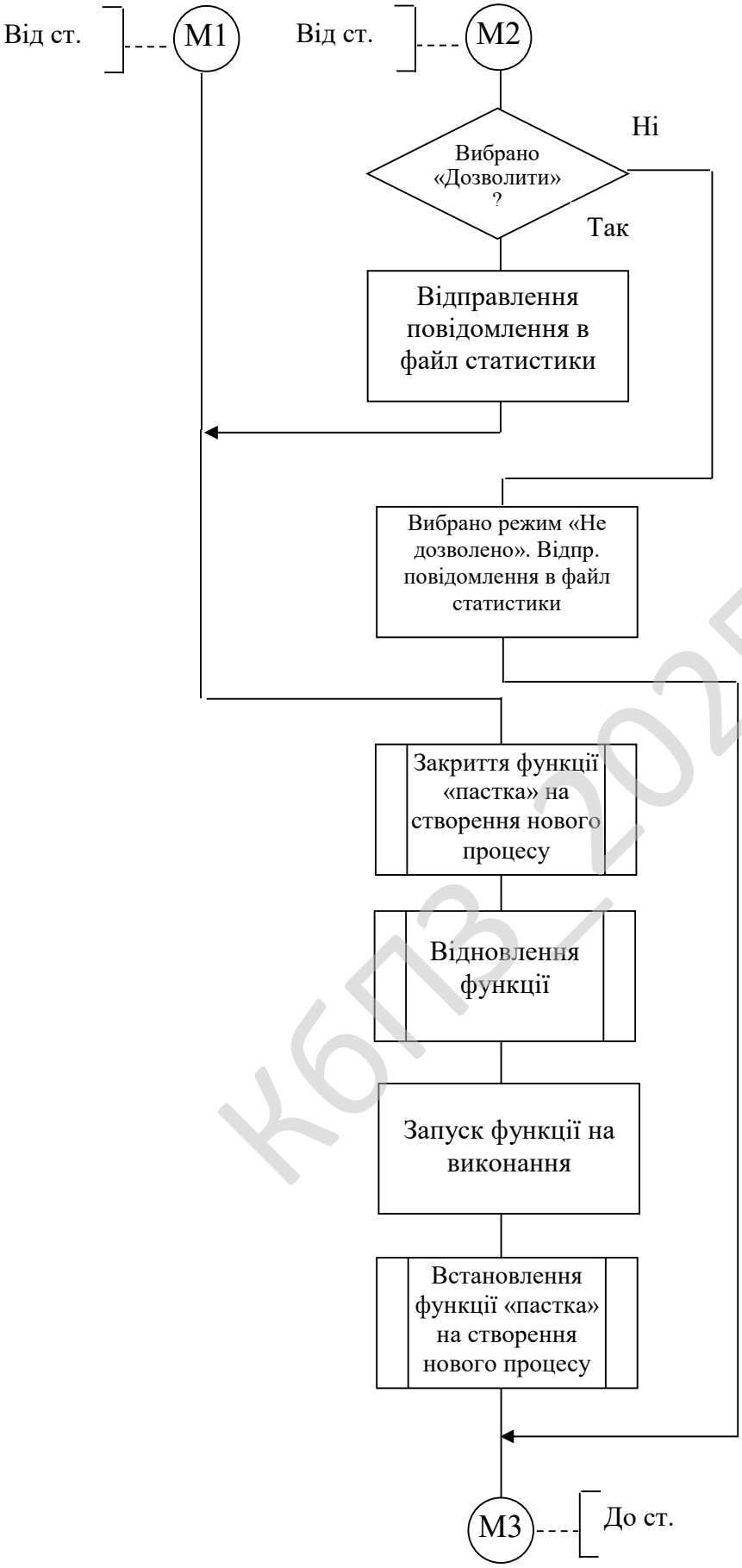

Рисунок 4.2, аркуш 2

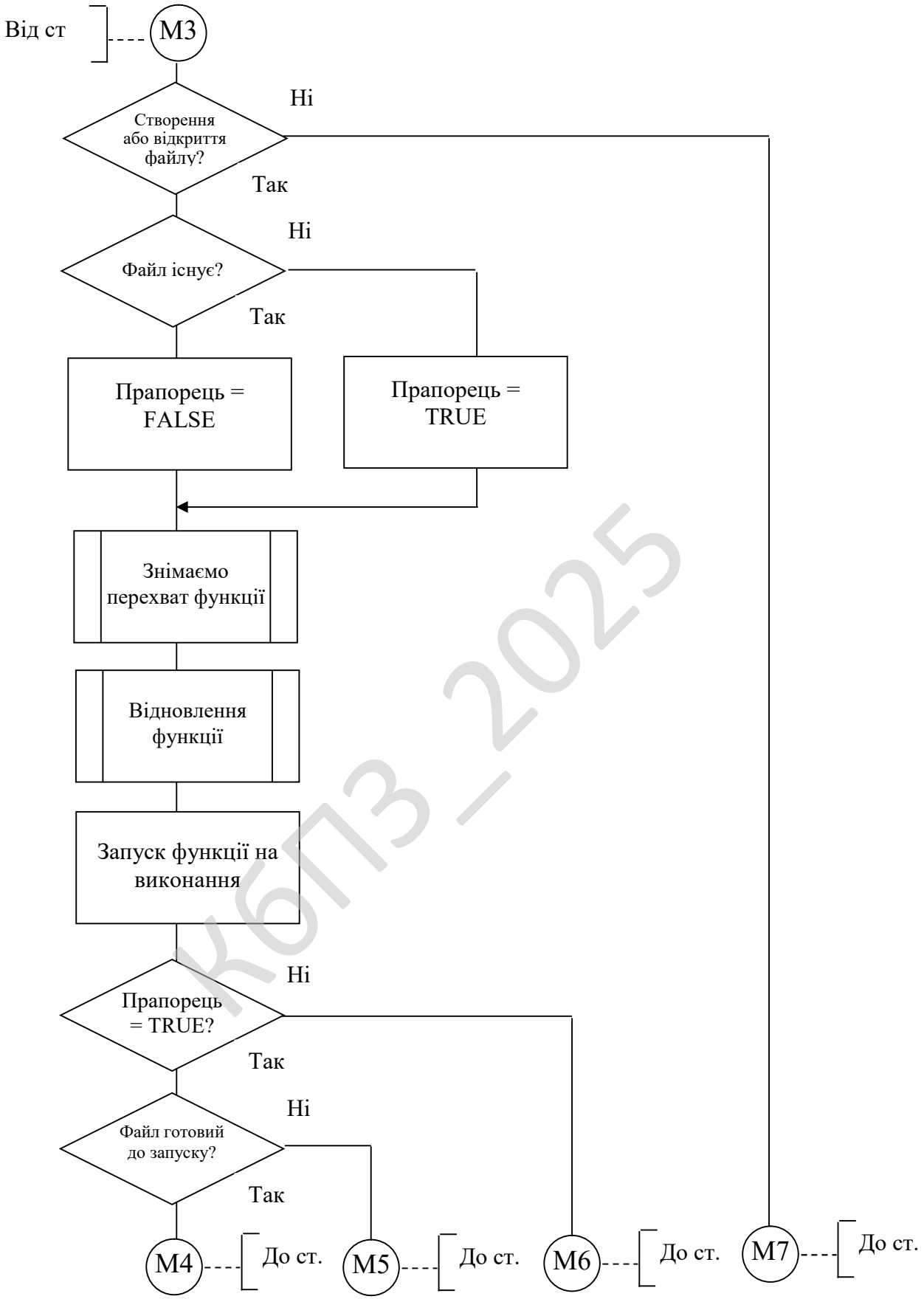

Рисунок 4.2, аркуш 3

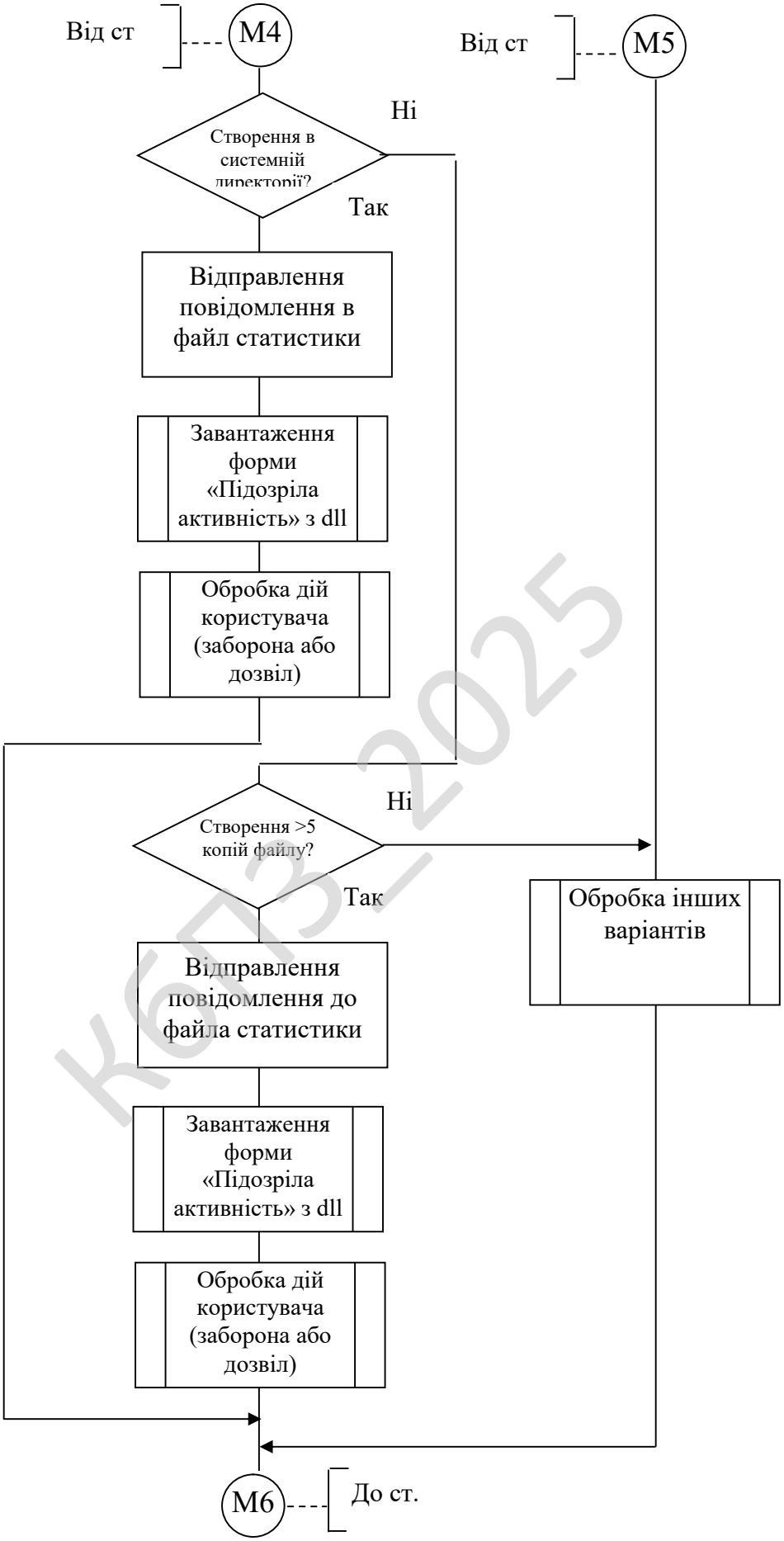

Рисунок 4.2, аркуш 4

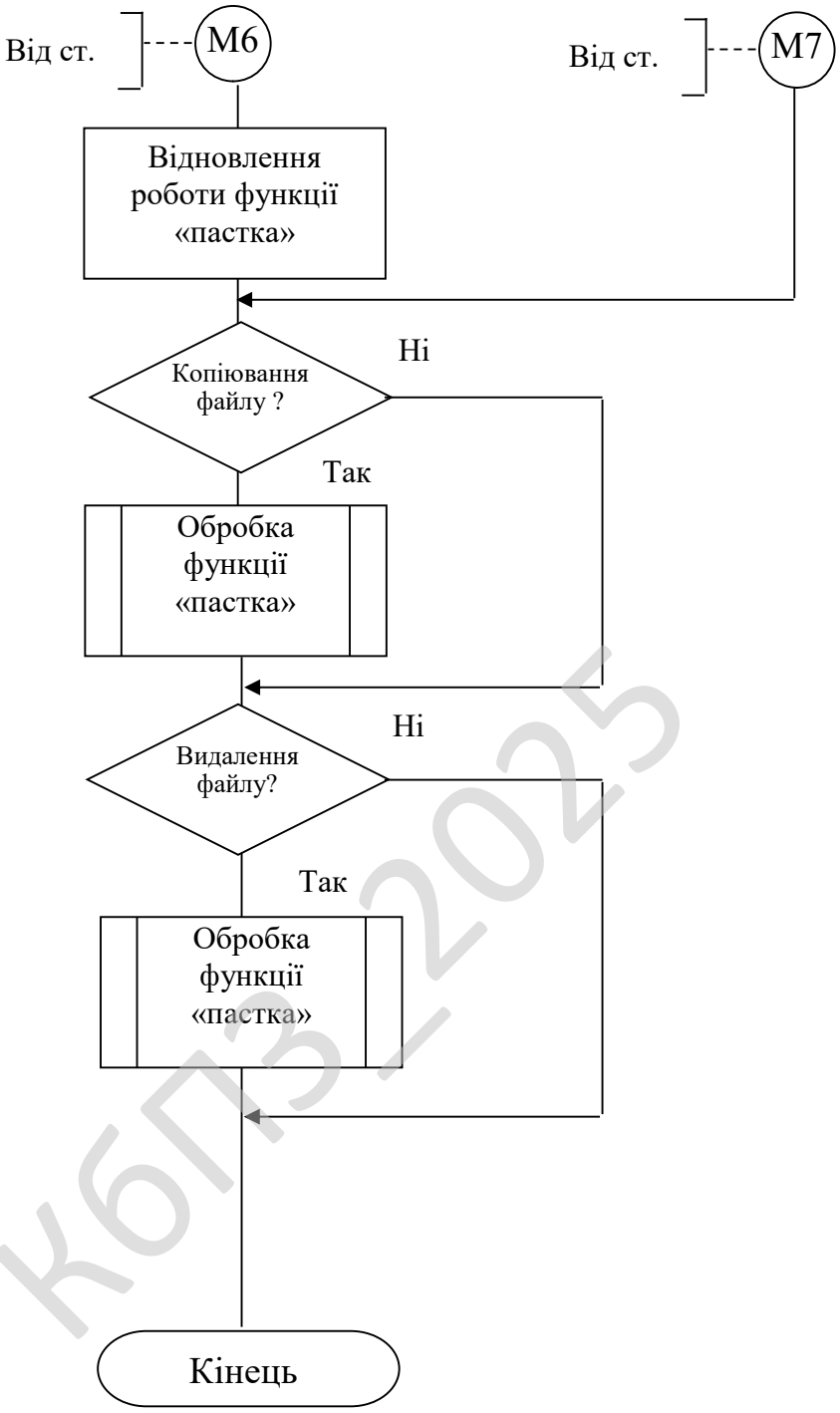

Рисунок 4.2, аркуш 5

Розглянемо алгоритм роботи підпрограми.

На початку роботи встановимо, використавши бібліотеку proControl.dl, режим перехвату системних API-функцій, який працює в фоновому режимі та

очікує надходження сигналу про спрацювання пастки. Перехід на наступний крок.

Якщо пастка спрацювала, визначається тип події. Якщо це створення нового процесу – перехід на виконання наступних кроків:

– завантаження форми «Запуск додатку» з dll;

– вибір режиму роботи користувачем,

Якщо обрано режим «Довіряти» – відправлення повідомлення в файл статистики, Закриття функції «пастка» на створення нового процесу, Відновлення функції та Запуск її на виконання, Встановлення функції «пастка» на очікування створення нового процесу. Якщо ні – відправлення повідомлення в файл статистики і перехід на обробку інших варіантів.

Якщо обрано режим «Дозволити» – відправлення повідомлення в файл статистики, Закриття функції «пастка» на створення нового процесу, Відновлення функції та запуск її на виконання, Встановлення функції «пастка» на очікування створення нового процесу

Якщо обрано режим «Не дозволено» – відправлення повідомлення в файл статистики і перехід на обробку інших варіантів.

Якщо обрано режим «Створення або відкриття файлу», перевірка наявності файлу, якщо ні – перехід на обробку інших варіантів.

Якщо файл не існує – виставляється прапорець TRUE, Файл існує – прапорець FALSE і перехід на наступні кроки:

– знімання перехвату функції;

– відновлення функції;

– запуск функції на виконання і перехід на наступний крок.

Перевірка наявності прапорця = TRUE. Якщо ні – перехід на відновлення роботи функції «пастка». якщо є – перехід на обробку інших варіантів.

Перевірка готовності файл до запуску. Якщо ні – перехід на обробку інших варіантів, якщо файл готовий до запуску – перехід на наступний крок.

Запит на створення файлу в системній директорії; якщо так – перехід на наступні кроки:

– відправлення повідомлення в файл статистики;

– завантаження форми «Підозріла активність» з dll ;

– обробка дій користувача (заборона або дозвіл) і перехід на відновлення роботи функції «пастка». Якщо створення файлу в системній директорії заборонено – перехід на обробку інших варіантів.

В якості ілюстрації наводимо фрагмент програмного коду обробки при створенні або копіюванні файла в системній директорії:

```
var
function CanCreateFile(FileName: String) : boolean;
//обробка при створенні або копіюванні файла в системній директорії
var
  ClearName: String;
  ext      : String;
  DirName  : String;
begin
 //відновлення скорочених імен та розбір шляху
  //файла - відокремлюємо назву додатка і його шлях
  //визначення розширення файлу
  LastFile  := RestoreLongName(FileName);
  Result    := false;
  ClearName := LowerCase(ExtractFileName(FileName));
  Ext       := LowerCase(ExtractFileExt(FileName));
  DirName   := LowerCase(ExtractFilePath(FileName));
  ClearName := GetStrCn(ClearName,length(ClearName)-length(ext));
  delete(dirname,1,(length(DirName)-1)-Length(ClearName));
  //якщо розширення файла відноситься до запускаємого, то
  if (ext = '.exe') or (ext = '.scr') or (ext = '.bat') or (ext = '.cmd') or
(ext = '.com') then
  //якщо системна директорія
  if (LowerCase(ExtractFilePath(FileName)) = LowerCase(WinDir))  or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(TempDir)) or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(SysDir))  then begin
     //відправка повідомлення в файл статистики
       sendstring(proCapt,PChar('#LOG5#'+FileName));
       //виведення діалогу "Підозріла активніть"
       //спроба копіювання в систему
       ShowProDialog(FileName,3);
     end;
   //якщо файли запускаємі
  if (ext = '.exe') or (ext = '.scr') or (ext = '.bat') or (ext = '.cmd') or
(ext = '.com') then
  if (DirName = ClearName+'\') or
(DirectoryExists(ExtractFilePath(FileName)+ClearName)) then begin
    if not ExistsInList(FileName, StrangeList) then begin
      inc(StrangeFiles);
      StrangeList := StrangeList + #13#10 + FileName;
    end;
    //якщо створено більше 5 копій
    if StrangeFiles > 5 then begin
    //відправка повідомлення до файлу статистики
```

```
        sendstring(proCapt,PChar('#LOG4#'+ParamStr(0)));
         //виведення діалогу "Підозріла активніть"
         //можливо P2P вірус
        ShowProDialog(StrangeList,4);
        Result := true;
        Exit;
      end;
    end;
    if Length(DirName) = 3 then
      if LowerCase( ExtractFileName(FileName)) = 'autorun.inf' then begin
     //відправка повідомлення до файлу статистики
        sendstring(proCapt,PChar('#LOG7#'+FileName));
      //виведення діалогу "Підозріла активніть"
      //створення підозрілого файлу
        ShowProDialog(FileName,0);
      end else
      if LowerCase( ExtractFileName(FileName)) = 'explorer.exe' then begin
        //відправка повідомлення до файлу статистики
        sendstring(proCapt,PChar('#LOG7#'+FileName));
        //виведення діалогу "Підозріла активніть"
        //створення підозрілого файлу
        ShowProDialog(FileName,6);
      end;
end;
```

Продовжуємо розгляд роботу алгоритму підпрограми (проактивний захист). Наступними кроками його роботи будуть:

– відновлення роботи функції «пастка»;

– запит на копіювання файлу: якщо так – обробка функції «пастка», якщо ні – перехід на інший варіант;.

Наводимо програмну реалізацію функції обробки при копіюванні додатку:

```
function CanCopySelf(FileName: String) : boolean;
//обробка при копіюванні додатку
var
  ClearName: String;
  ext      : String;
  DirName  : String;
begin
 //відновлення скорочених імен та розбір шляху
  //файла - відокремлюємо назву додатка і його шлях
  //визначення розширення файлу
  LastFile  := RestoreLongName(FileName);
  Result    := false;
  ClearName := LowerCase(ExtractFileName(FileName));
  Ext       := LowerCase(ExtractFileExt(FileName));
  DirName   := LowerCase(ExtractFilePath(FileName));
  ClearName := GetStrCn(ClearName,length(ClearName)-length(ext));
  delete(dirname,1,(length(DirName)-1)-Length(ClearName));
  DeleteInvalidInList(CopyesList);
  if not ExistsInList(FileName,CopyesList) then begin
    CopyesList := CopyesList + FileName + #13#10;
    Inc(CopyesNumber);
  end;
  //якщо більше 7 копій, то
  if CopyesNumber > 7 then begin
```

```
  //відправка повідомлення
    sendstring(proCapt,PChar('#LOG4#'+ParamStr(0)));
     //виведення діалогу "Підозріла активність"
      //можливо P2P вірус
    ShowProDialog(CopyesList,1);
    Exit;
  end;
  if (DirName = ClearName+'\') or
(DirectoryExists(ExtractFilePath(FileName)+ClearName)) then begin
    if not ExistsInList(FileName, StrangeList) then begin
      inc(StrangeFiles);
      StrangeList := StrangeList + FileName + #13#10;
    end;
    //якщо більше 5 копій
    if StrangeFiles > 5 then begin
      //відправка повідомлення в файл статистики
      sendstring(proCapt,PChar('#LOG4#'+ParamStr(0)));
      //виведення діалогу "Підозріла активність"
      //можливо P2P вірус
      ShowProDialog(StrangeList,4);
      Result := true;
      Exit;
    end;
  end;
  //якщо в системну директорію
  if (LowerCase(ExtractFilePath(FileName)) = LowerCase(WinDir))  or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(TempDir)) or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(SysDir))  then begin
       //відправка повідомлення
       sendstring(proCapt,PChar('#LOG5#'+FileName));
      //виведення діалогу "Підозріла активність"
      //спроба копіювання в системну директорію
       ShowProDialog(FileName,2);
     end;
end;
```

Продовжуємо розгляд роботи алгоритму підпрограми (проактивний захист).

Наступним кроком буде запит на видалення файлу: якщо ні – вихід на кінець роботи програми, якщо так – обробка функції «пастка».

Наводимо програмну реалізацію обробки функції «пастка» що виникає при команді видалення файлу:

```
function DeleteFileCallBack(lpFileName: PChar): BOOL; stdcall;
//обробка пастки, що виникає при команді видалення файлу
begin
  //знімаємо пастку
  UnHookCodeHook(@SystemDeleteFileA);
  //обробка системою проактивного захисту
  CanDeleteFile(lpFileName);
  try
    //видалення файлу
    result := Windows.DeleteFile(lpFileName);
  except
  end;
  //відновлення пастки
```

```
SetCodeHook(SystemDeleteFileA.Address,@DeleteFileCallBack,@SystemDeleteFileA);
  end;
  (* ------------------------------------------------------------------------ *)
  //обробка пастки, що виникає при обробці системної функції копіювання файлу
  function CopyFileANext(lpExistingFileName, lpNewFileName: PChar;
  bFailIfExists: BOOL): BOOL; stdcall;
  begin
    //знімаємо пастку
    UnHookCodeHook(@SystemCopyFileA);
    try
      //виконуємо копіювання
      Result := CopyFile(lpExistingFileName,lpNewFileName,bFailIfExists);
    except
    end;
    if LowerCase(RestoreLongName(lpExistingFileName)) = LowerCase(ParamStr(0))
  then
    begin
      //обробка системою проактивного захисту
      CanCopySelf(RestoreLongName(lpNewFileName));
    end else CanCreateFile(RestoreLongName(lpNewFileName));
    //встановлення пастки
    SetCodeHook(SystemCopyFileA.Address,@CopyFileANext,@SystemCopyFileA);
  end;
```

Кінець роботи програми.

В наведених основних модулях програмного забезпечення підтримується принцип інформативності, а саме: надання користувачу інформації про виконання встановлених перевірок та результати роботи окремих програм та підпрограм, надання (в разі необхідності) теоретичних відомостей щодо принципів роботи системи, необхідних для роботи технічних параметрів, тощо. Інформаційні повідомлення в процесі роботи системи виводяться на екран монітора чи роздруковуються (по бажанню користувача).

Таким чином, завдання визначене ТЗ та постановкою задачі, виконане в повному обсязі. Основні етапи розробки ПЗ системи включають:

- розробку структурної і функціональної схем системи для визначення та обґрунтування вибору структурних частин програмної моделі ті їх функцій;

- розробку ПЗ системи відповідно принципам та методиці реалізації систем аналогічного класу та спрямування;

- теоретичне обґрунтування розробки основних рішень побудови програмної моделі та виконання необхідних розрахунків;

- розробка алгоритмів та ПЗ системи.

Розроблене ПЗ має модульну структуру, що забезпечує виконання системою принципів мобільності та адаптивності і цілком дієздатне. Про це свідчать результати дослідної експлуатації. Розроблено досить потужну систему з додержанням усіх вимог до систем даного класу, яка в подальшому може вдосконалюватись шляхом введення нових функцій – сервісних та робочих, елементів експертних систем та штучного інтелекту.

Програмне забезпечення системи працює на платформі ОС Windows. Середовище передачі даних: фізичні лінії, телефоні, виділені чи комутовані канали зв'язку. При цьому вона пред'являє мінімальні вимоги до системних ресурсів, адже нею використовуються тільки ті процедури введення/виведення, які необхідні в роботі.

Для зручності користувача використовуються звичні для нього панелі інструментів і меню та налаштування за допомогою ієрархічного дерева в стилі провідника ОС Windows.

Система працює в автоматичному режимі і може експлуатуватись безперервно чи під керуванням користувача (інтерактивний режим).

Для більш повної ілюстрації роботи системи розроблено та виконано 7 листів графічного матеріалу формату А4, до складу якого ввійшли: структурна схема; функціональна схема; діаграма процесів; блок-схеми роботи алгоритмів основної програми та підпрограми, які найбільш ефективно характеризують та ілюструють розроблене ПЗ.

4.2 Захист розробленого програмного забезпечення

Організація безпеки КМ – непроста задача. Дані в будь-якій мережі (локальній чи глобальній) можуть розподілятись між декількома серверами. Користувачі, яким потрібні ці дані для роботи, повинні мати до них доступ. З іншого боку, забезпечення гнучкого доступу для законних користувачів веде до зниження рівня безпеки мережі.

Було вирішено провести організацію захисту розробленого ПЗ шляхом застосування способу обмеження доступу. Це спосіб заснований на паролі та утримує 2 додаткових ступені захисту: визначеної N-ї кількості разів правильного і неправильного введення паролю.

Використаємо пароль довжиною 8-м символів, щоб значно ускладнити роботу зловмисникам по його злому та в рази знизити можливість несанкціонованого доступу. Еталон паролю розташуємо на сервері. При введені користувачем КМ паролю, який співпав з еталоном, на екран монітора буде виведене повідомлення з пропозицією повторити введення паролю ще раз. Якщо користувач тричі правильно введе пароль – одержить право доступу до завантаження системи та подальшої роботи з даними.

Визначено обмеження в системі і на кількість спроб неправильного введення паролю. Їх закладено дві, якщо після повторного введення паролю він не співпаде з еталоном – на екран монітора буде виведене повідомлення і комп’ютер буде заблоковано.

В процесі експлуатації системи організації-замовнику необхідно безпремінно виконувати наступні рекомендації:

- регулярно змінювати паролі користувачів, періодичність визначає системний адміністратор;

- обов’язково змінювати паролі всіх користувачів при звільненні будь-кого із службовців, відповідального за безпеку мережі;

- призначення паролів користувачам повинен роботи строго системний адміністратор;

- під підпис попередити кожного користувача про неможливість виконання ним наступних дій: повідомлення свого паролю іншим особам; запис паролю в незашифрованому вигляді; повідомлення свого паролю керівництву через телефону мережу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Згідно з ТЗ в процесі роботи над випускною кваліфікаційною роботою за першим (бакалаврським) рівнем вищої освіти було розроблене програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

Впровадження любого побічного інструментарію, в тому числі і додатково захисту (тобто, розробленої системи), супроводжується певними складнощами. Вони пов'язані з тим, що, окрім вивчення користувачами питань безпеки інформації, необхідно розширити коло обов'язків супровідників ПЗ та служби технічної підтримки. Персоналу, що буде супроводжувати ПЗ, необхідно вивчити функції системи, представлені наявною технічною документацією на рівні, що буде достатнім для її безперебійної роботи та забезпечення виконання означених задач інтеграції захисту в новому ПЗ.

Інтеграція системи захисту в існуючу інфраструктуру передбачає сумісність із іншими платформами, наявність API для розширення функціональності, централізоване управління та масштабованість для адаптації до змін у мережі. Відсутність належної інтеграції може призвести до прогалин у безпеці та конфліктів між компонентами.

Інтеграція також вимагає тісної взаємодії з системами управління ідентифікацією та доступом для забезпечення чіткої сегментації прав доступу. Це дозволяє мінімізувати ризики, пов'язані з людським фактором, і підвищити прозорість доступу до критичних активів.

Продуктивність системи визначається її пропускною здатністю, швидкістю реакції на загрози, оптимальним використанням обчислювальних ресурсів і стійкістю до високих навантажень. Чітке врахування цих критеріїв дозволяє

організаціям вибирати рішення, що відповідають їхнім технічним і фінансовим потребам, забезпечуючи при цьому максимальну ефективність у протидії сучасним кіберзагрозам.

Наводимо перелік принципових підходів до налаштування розробленої системи в КМ Замовника.

На шлюзовому сервері зазвичай використовується принцип "все, що не дозволено явно - заборонено", тоді як на персональних комп'ютерах часто застосовується більш ліберальний підхід до вихідного трафіку.

Шлюзовий брандмауер налаштовується централізовано системними адміністраторами, тоді як персональні брандмауери можуть налаштовуватись користувачами або через групові політики.

Налаштування шлюзового брандмауера потребує глибоких технічних знань і розуміння мережевої інфраструктури, тоді як персональні брандмауери зазвичай мають простіший інтерфейс, орієнтований на звичайних користувачів.

На шлюзовому сервері модуль проактивного захисту налаштовується з фокусом на виявлення розподілених атак і загроз, націлених на множину систем. Налаштування часто включає розробку специфічних правил під інфраструктуру конкретної організації.

На персональному комп'ютері акцент робиться на виявленні загроз на кінцевих точках (endpoint detection) та захисті від загроз, націлених на користувачів (фішинг, соціальна інженерія). Налаштування зазвичай більш уніфіковане й орієнтоване на типові сценарії використання.

Шлюзовий модуль проактивного захисту вимагає командної роботи спеціалістів з безпеки та значно більше часу на налаштування й тонке налагодження, тоді як персональний захист часто використовує автоматичні налаштування та рекомендації виробника.

Експлуатацію розробленої системи буде здійснювати персонал підрозділів і відповідних служб організації-замовника. Обслуговування та розвиток системи має здійснювати спеціалізований підрозділ, який необхідно створити в процесі впровадження системи в промислову експлуатацію. Цей підрозділ може бути цілком самостійним (або входити до складу служби, що безпосередньо буде

експлуатувати розроблену систему) і має забезпечити виконання наступних задач:

- технологічне супроводження процесу виконання безпосередніх функціональних задач системою;

- системне забезпечення інформаційно-обчислювальних процесів;

- розробку сервісних програмних компонентів та доробку, в разі необхідності, програмного забезпечення системи з цілю його модифікації;

- супроводження програмних компонентів;

- обслуговування технічних засобів;

- організацію робіт по розвитку системи.

В якості ілюстрації роботи системи по безпосередньому призначенню наводимо скриншоти ПЗ (рисунок 5.1, рисунок 5.2, рисунок 5.3, рисунок 5.4).

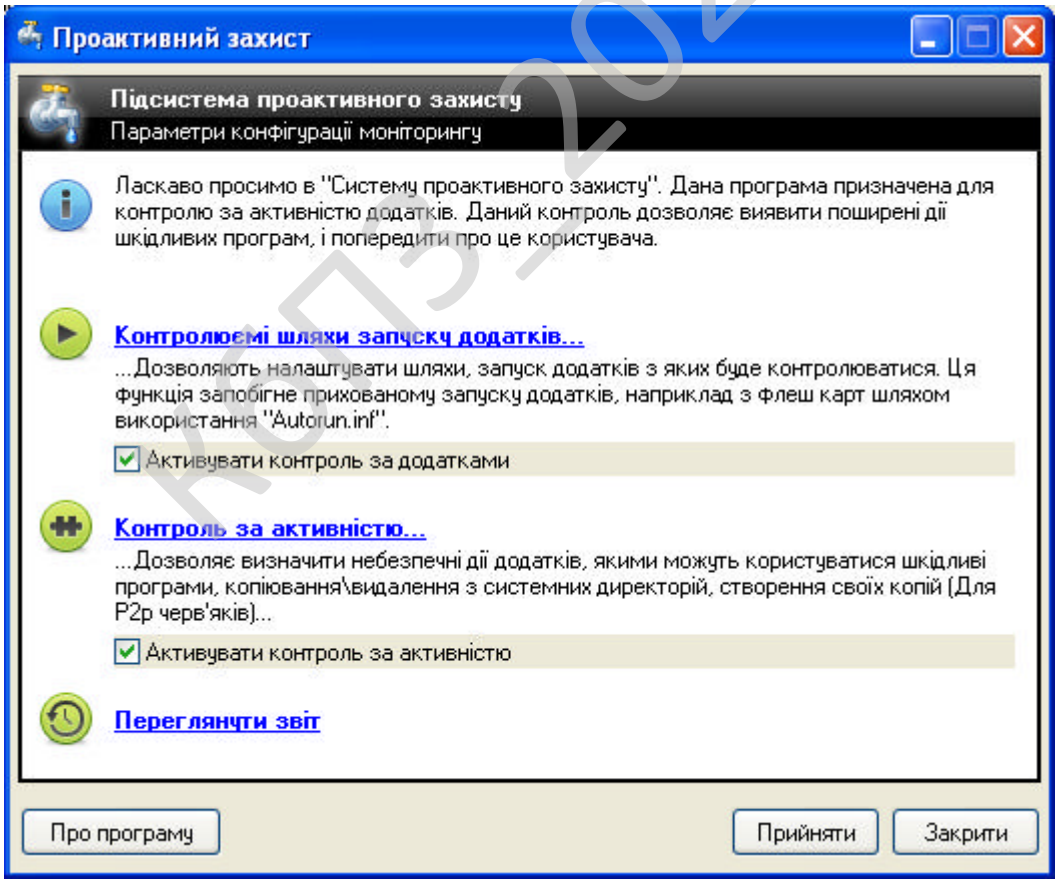

Рисунок 5.1 – Скриншот форми налаштувань проактивної системи захисту

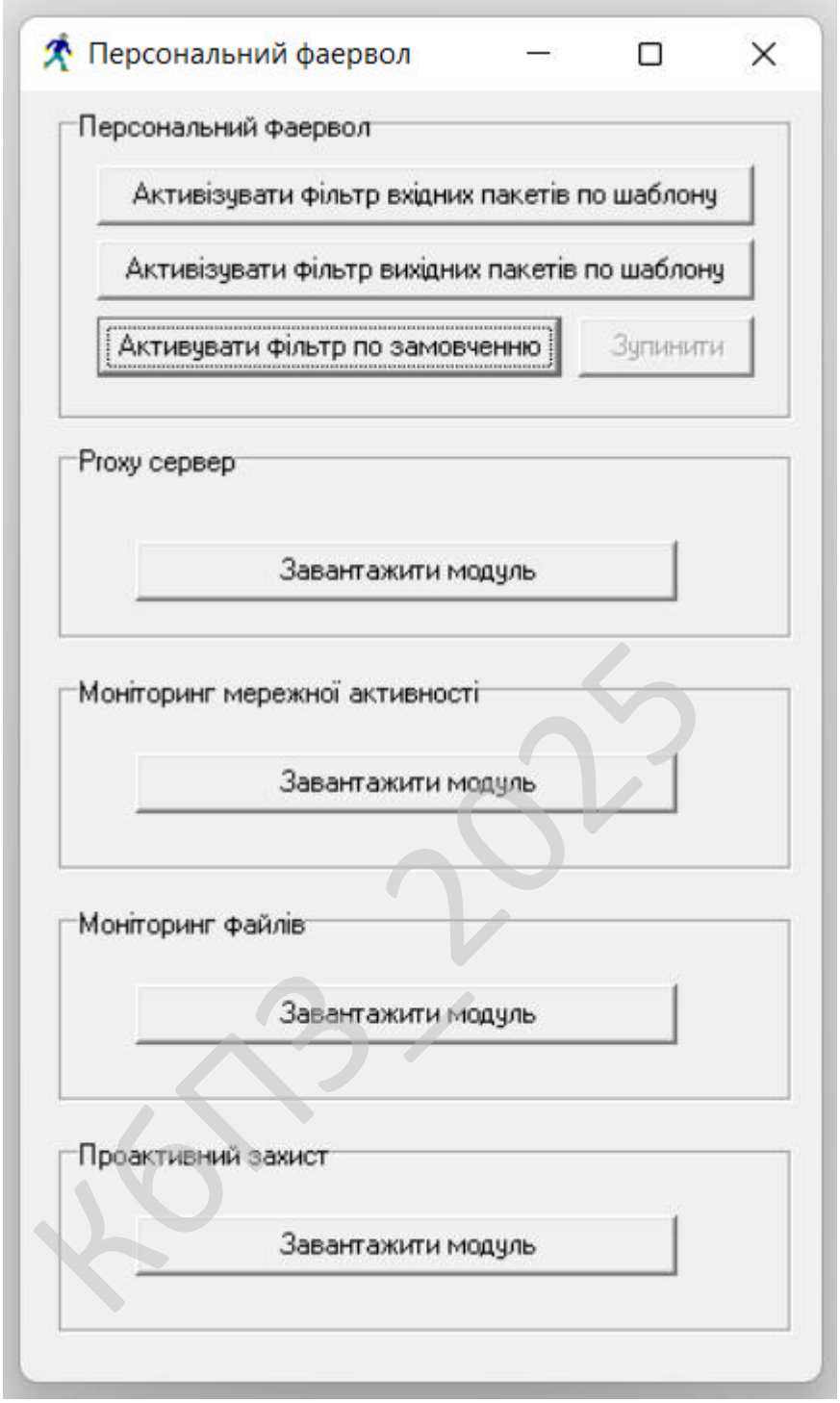

Рисунок 5.2 – Скриншот основної програми

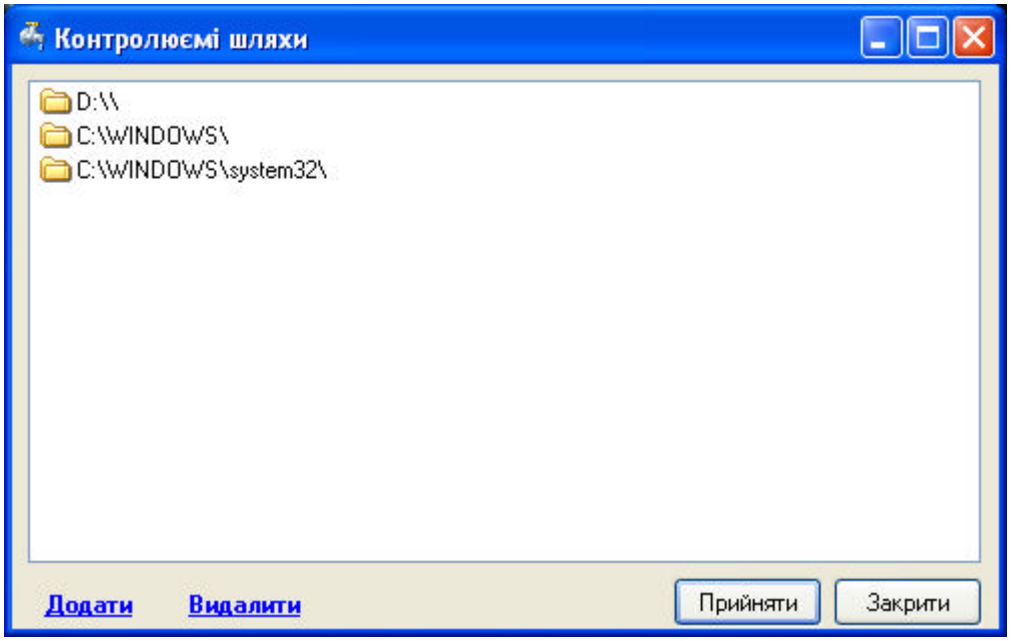

Рисунок 5.3 – Скриншот налаштування контролюємих шляхів

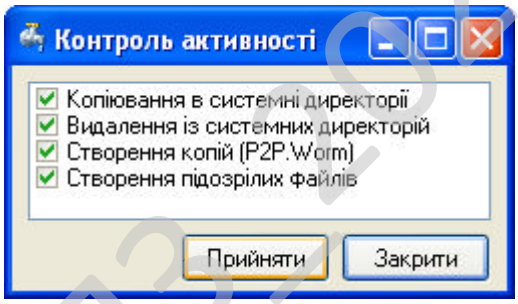

Рисунок 5.4 – Скриншот налаштування контролю активності

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначене для системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

Вітчизняні розробки програмних систем аналогічного класу та спрямування в межах України представлені не в достатній мірі, хоча на сьогодення це питання є надзвичайно актуальним.

Виконання завдання на розробку ПЗ системи полягало у вирішенні наступних задач:

– проведення огляду наявних на світовому ринку програмних продуктів систем – аналогів проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу;

– дослідження механізму проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу;

– виконання на основі отриманих результатів досліджень програмної реалізації системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми, дозволяють успішно вирішувати завдання для проактивного виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі поведінкового аналізу.

Розроблена система:

- є апаратно невибагливою, що дає можливість працювати як на одному, окремо взятому ПК, так і на складній комбінації локальних мереж, що працюють на значній відстані одна від одної;

- дозволить зберегти раніше зроблені капіталовкладення при переході на більш продуктивний варіант апаратної платформи;

- забезпечує можливість роботи в режимі реального часу;

- забезпечує обмін даними з іншими системами КМ;

- надійно захищена від несанкціонованого доступу;

- має прийнятну вартість для мінімальної конфігурації комплексу технічних засобів і можливість використання обладнання, відповідного сучасним технічним досягненням.

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.

В якості середовища розробки ПЗ системи використане середовище швидкої розробки Delphi. Це одне з найбільш потужних візуально об'єктно-орієнтованих середовищ програмування, яке дозволяє на найсучаснішому рівні створювати розгалужені програмні комплекси, призначені для роботи в комп'ютерних мережах та в мережах Інтернет. Дана мова програмування дозволила ефективно обробити дані для системи та мінімізувати терміни розробки ПЗ. Розроблене ПЗ працює під керуванням ОС Windows.

Надаються необхідні рекомендації з установки розробленого ПЗ, проведення дослідної експлуатації та подальшої промислової експлуатації.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Axelrod A. Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects / A. Axelrod. – NY: Apress, 2018. – 588 p

2. Richards M. Fundamentals of Software Architecture: An Engineering Approach / M. Richards, N. Ford. – Sebastopol: O'Reilly Media, 2020. – 432 p.

3. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language / M. Fowler. – Boston: Addison-Wesley Professional, 2018. – 191 p.

4. Rumbaugh J. UML 2.0. Object-Oriented Modeling and Design with UML / J. Rumbaugh, M. Blaha. – London: Pearson, 2012. – 496 p.

5. Larman C. Applying UML and Patterns: An Introduction to ObjectOriented Analysis and Design and Iterative Development / C. Larman. – London: Pearson, 2004. – 736 p.

6. Lamdakkar O., Ameur I., Mbarek Eleyatt M., Carlier F., Ait Ibourek L. Toward a modern secure network based on next-generation firewalls: recommendations and best practices. Procedia Computer Science. Vol. 238. 2024. Pp. 1029–1035. DOI: https: // doi. org / 10.1016 / j. procs. 2024.06.130.

7. . Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 p.

8. Beller M., Gousios G., Panichella A., Proksch S., Amann S., & Zaidman A. (2017). Developer testing in the ide: Patterns, beliefs, and behavior. IEEE Transactions on Software Engineering, 45(3), 261-284.

9. Lamdakkar O., Ameur I., Mbarek Eleyatt M., Carlier F., Ait Ibourek L. Toward a modern secure network based on next-generation firewalls: recommendations and best practices. Procedia Computer Science. Vol. 238. 2024. Pp. 1029–1035. DOI: https: // doi. org / 10.1016 / j. procs. 2024.06.130.

10. IDS vs. IPS: Definitions, Comparisons & Why You Need Both: веб-сайт. URL: https: // www. okta. com / identity – 101 / ids-vs-ips/ (дата звернення: 17.11.2024).

11. Smith R.E. Elementary Information Security, 2nd ed. Jones & Bartlett Learning, 2015.

12. Sichkar M., Pavlova L. A short survey of the capabilities of Next Generation firewalls. Computer Science and Cybersecurity. 2023. №1. С. 28–33.

13. Stamp M. Information Security: Principles and Practice, 2nd ed. Wiley-IEEE Press, 2011.

14. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

15. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

16. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

17. «Про Стратегія інформаційної безпеки України» Указ Президента України: Стратегія від 28.12.2021 № 685/202. URL: https:// zakon.rada.gov.ua/laws/show/685/2021#Text.

18. «Про Стратегію національної безпеки України» Указ Президента України: Стратегія від 14.09.2020 № 392/2020. URL: https:// zakon.rada.gov.ua/laws/show/392/2020#Text.

19. Порядок оновлення антивірусних програмних засобів, які мають позитивний експертний висновок за результатами державної експертизи в сфері технічного захисту інформації, затверджений наказом Адміністрації Держспецзв'язку від 26.03.2007 № 45, зареєстрований в Міністерстві юстиції України 10.04.2007 за № 320/13587.

20. Прищепа, О., & Доценко, О. (2020). Overview of static methods of analysis malicious software. Computer Science and Cybersecurity, (№ 2), 15-24. https://doi.org/10.26565/2519-2310-2020-2-02

21. Ethical Hacking Labs. Шкідливе програмне забезпечення - HackYourMom. HackYourMom. URL: https://hackyourmom.com/servisy/№4-ethical-hacking-labs-shkidlyve-programne-zabezpechennya/ (дата звернення: 07.01.2025).

22. Коробейнікова Т. І., Цар О. О. Аналіз сучасних відкритих систем виявлення та запобігання вторгнень. Грааль науки. 2023. № 27. С. 317–325.

23. Жилін А. В., Шаповал О. М., Успенський О. А. Технології захисту інформації в інформаційнотелекомунікаційних системах: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, Вид-во "Політехніка", 2021. 213 с.

24. Демидов, З. Г. Основні види кібератак на WEB-сайти / З.Г. Демідов // Актуальні питання протидії кіберзлочинності та торгівлі людьми : зб. матеріалів Всеукр. наук.-практ. конф. (м. Харків, 15 листоп. 2017 р.) / МВС України, Харк. нац. ун-т внутр. справ; Координатор проектів ОБСЄ в Україні. – Харків : ХНУВС, 2017. – С. 131-134.

25. Технології забезпечення безпеки мережевої інфраструктури: підручник / В. Л. Бурячок та ін. К.: КУБГ, 2019. 218 с.

26. Безменов М. І. Основи програмування у середовищі Delphi : навч. посіб. – Харків : НТУ «ХПІ», 2010. – 608 с.

27. Короткий курс програмування в середовищі Delphi (для студентів заочної форми навчання спеціальності 7.050201 "Менеджмент організацій" спеціалізації "Інформаційні системи в менеджменті"). Укл. Г.А. Мірошниченко. - Харків: ХНАМГ, 2004. - 81 с.,іл.

28. Бурячок В. Л. Інформаційний та кіберпростори: проблеми безпеки, методи та засоби боротьби : посібник / [В. Л. Бурячок, С. В. Толюпа, В. В. Семко та ін.]. – К. : ДУТ-КНУ, 2016. – 178 с.

29. Сопілко І. М. Інформаційна безпека та кібербезпека: порівняльноправовий аспект// Юридичний вісник Повітряне і космічне право. – Київ: НАУ, 2021. – № 2(59). С. 110-115.

30. Галіпчак В. Інформаційна війна як складова гібридної війни в умовах російської агресії. Вісник Прикарпатського університету, 2023. Вип. 15. 26-32.

31. Чмир Я. Сучасні проблеми інформаційної безпеки України та перспективні напрями їх вирішення». Наукові праці Міжрегіональної Академії управління персоналом. Політичні науки та публічне управління, 2022. №2(62). С. 149-154.

32. Циріна М Сучасні технології захисту й опрацювання конфіденційної докуметної інформації в організаціях і установах різних форм власності. Національна академія керівних кадрів культури і мистецтв. Бібліотекознавство. Документознавство. Інформологія, 2021. №4. С40.

33. .Керусов М. В. Політика інформаційної безпеки інтернет провайдерів. Кібербезпека в сучасному світі : матеріали II Всеукр. наук.-практ. конф. м. Одеса, 20 листоп. 2020 р. Одеса : Видавничий дім «Гельветика», 2020. С.27-30.

34. Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: монографія / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 305 с.

35. Коваленко А.В. Методи якісного аналізу і кількісної оцінки ризиків розробки програмного забезпечення / А.В. Коваленко, А.А. Смирнов // Інформаційні технології в управлінні, освіті, науці і промисловості: монографія / Під редакцією професора В. С. Пономаренко. - Х.: Видавець Рожко С. Г., 2016. - 566 с.

36. Коваленко А.В. Розробка методу управління ризиками розробки програмного забезпечення / А.В. Коваленко, А.А. Смірнов // Інформаційні технології : проблеми та перспективи : монографія / За загальною редакцією В. С. Пономаренка. - Х.: Видавець Рожко С. Г., 2017. - 447 с.

37. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

38. Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141.

39. Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

40. Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

41. Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

42. Жилін А. В., Шаповал О. М., Успенський О. А. Технології захисту інформації в інформаційнотелекомунікаційних системах: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, Вид-во "Політехніка", 2021. 213 с.

43. Шуліка К., Балагура Д., Смірнов А., Непокритов Д., Литвин А. Метод використання сучасних систем захисту кінцевих точок (EDR) для убезпечення від комплексних атак». Сучасний стан наукових досліджень та технологій в промисловості. 2024. №2 (28). С. 182–195.

44. . Скіцько О. Ширшов Р. Система управління інформаційної безпеки як інструмент підвищення захищеності та ефективності об'єктів критичної інфраструктури. International Science Journal of Engineering & Agriculture. 2023. Vol. 2. No. 6. Pp. 12–22.

45. Малькевич Р., Балацька В. Використання SPLUNK для захисту інформації в організаціях. Інформаційна безпека та інформаційні технології: зб. тез доповідей V Всеукр. наук. – практ. конф. молодих учених, студентів і курсантів. С. 66–68.

46. НД ТЗІ 1.1-003-99 Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу.

47. ДСТУ ISO/IEC 27001:2015 Інформаційні технології. Методи захисту системи управління інформаційною безпекою. Вимоги (ISO/IEC 27001:2013; Cor 1:2014, IDT).

48. НД ТЗІ 3.6-004-21 «Порядок впровадження систем безпеки інформації в державних органах, на підприємствах, в організаціях, в інформаційно-комунікаційних системах яких обробляється інформація, вимога щодо захисту якої встановлена законом та не становить державної таємниці».

49. Richards M. Fundamentals of Software Architecture: An Engineering Approach / M. Richards, N. Ford. – Sebastopol: O'Reilly Media, 2020. – 432 p. НД ТЗІ 2.3-025-21 «Методика оцінювання заходів захисту інформації, вимога щодо захисту якої встановлена законом та не становить державної таємниці, для інформаційних систем».

50. НД ТЗІ 1.1-002-99 Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

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

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу.

2 Підстава для розробки

Підставою для розробки системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (наказ № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення. Впровадження нових інформаційних технологій і застосування сучасних засобів програмування дозволить підвищити захищеність та надійність функціонування комп'ютерної мережі за рахунок автоматизації процесу захисту від мережних атак.

4 Джерела розробки

Джерелом розробки цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовні до теми бібліографічні джерела і існуючі на ринку програмних продуктів системи-аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складовими розробки є:

– огляд та аналіз властивостей існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи, що підлягає розробці в процесі виконання ВКРБ;

– вибір і обґрунтування методик побудови системи, що підлягає розробці, та засобів їхньої програмної реалізації;

– розробка структур даних і механізму їхньої взаємодії, основних принципів та методики проектування системи;

– програмна реалізація алгоритмів роботи системи та алгоритмів захисту розробленого програмного забезпечення від несанкціонованого доступу.

5.2 Показники призначення

Система повинна забезпечувати:

– виявлення та блокування шкідливого програмного забезпечення у комп'ютерній мережі на основі його поведінкового аналізу та проактивних технологій;

– цілісність запису даних моніторингу і аналізу параметрів трафіку в поточних файлах інформації;

– простий, інтуїтивно зрозумілий інтерфейс з користувачем.

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, об'єктно-орієнтовані засоби розробки, які на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Компонент повинен використати існуючі угоди по стандартним викликам процедур, функцій, засобів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів системи повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-22 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Система повинна бути реалізована на ЕОМ типу IBM PC, працювати в ОС Windows 10/11 і орієнтований на сумісні з цією платформою зовнішні пристрої, мережне обладнання і прикладне програмне забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Сумісність програмного забезпечення повинна бути забезпечена за рахунок його реалізації засобами об'єктно-орієнтовано програмування та наявності інтерфейсу взаємодії з ОС.

5.8.1 Обладнання

Комп’ютер Intel® N100/8 Gb/240 Gb/SVGA 14” або сумісні з ним.

5.8.2 Мова програмування

Середовище 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 алгоритмів, інструкції користувача, 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-схеми алгоритмів роботи програми – 4 аркуші.
– Пояснювальна записка – 97 аркушів.

8 Етапи розробки

На рівні проекту розробляються (терміни виконання етапів див. в "Завданні на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти":

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу системи.

8.6 Віднаходження ПЗ системи, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 26.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 06.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Минайленко Р.М.

Програмне забезпечення системи проактивного виявлення та блокування шкідливого програмного забезпечення у комп’ютерній мережі на основі поведінкового аналізу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 32

Літера: РП

Кропивницький – 2025 року

Horr_Firewall.dpr

```
program Horr_Firewall;

uses
  Forms,
  MainUnit in 'MainUnit.pas' {FirewallForm};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TFirewallForm, FirewallForm);
  Application.Run;
end.
```

MainUnit.pas

```
unit MainUnit;

interface

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

type
  PIpBytes       =  ^TIpBytes;
  TIpBytes       =  Array [0..3] of Byte;

type
  TFirewallForm = class(TForm)
    GroupBox1: TGroupBox;
    btStartFilter: TButton;
    btStopFilter: TButton;
    GroupBox2: TGroupBox;
    Button1: TButton;
    GroupBox3: TGroupBox;
    GroupBox4: TGroupBox;
    Button2: TButton;
    Button3: TButton;
    procedure btStartFilterClick(Sender: TObject);
    procedure btStopFilterClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    hIF : INTERFACE_HANDLE;
    ipLocal : TIpBytes;
    function StrToIp(lpszIP: PChar; lpipAddr: PIpBytes): PIpBytes;
    function GetLocalIPAddr(lpipAddr: PIpBytes): Boolean;
    procedure AddFilter(inP: Boolean; lpszRemote: PChar; protoType: DWORD;
lpszPort: PChar);
  end;

var
  FirewallForm: TFirewallForm;
  StartInfo : TStartupInfo;
  ProcInfo : TProcessInformation;
  lv_f: File;
  lv_parol: Boolean;
  err: string;
```

```
implementation

{$R *.dfm}

function TFirewallForm.StrToIp(lpszIP: PChar; lpipAddr: PIpBytes): PIpBytes;
//перетворення строки в IP адресу
var
 lpszStr : Array [0..63] of Char;
 dwPos : Integer;
 lpPos : PChar;
begin
 StrLCopy(@lpszStr, lpszIP, SizeOf(lpszStr));
 lpszStr[Pred(SizeOf(lpszStr))]:=#0;

 ZeroMemory(lpipAddr, SizeOf(TIpBytes));

 dwPos:=Pred(SizeOf(TIpBytes));
 lpPos:=StrRScan(lpszStr, '.');
 while Assigned(lpPos) do
  begin
   lpPos^:=#0;
   Inc(lpPos);
   lpipAddr^[dwPos]:=StrToIntDef(lpPos, 0);
   Dec(dwPos);

   if (dwPos = 0) then
    break;

   lpPos:=StrRScan(lpszStr, '.');
  end;
 lpipAddr^[dwPos]:=StrToIntDef(lpszStr, 0);

 result:=lpipAddr;
end;

function TFirewallForm.GetLocalIPAddr(lpipAddr: PIpBytes): Boolean;
//отримати локальний IP
var
 lpszLocal:  Array [0..255] of Char;
 pheAddr:    PHostEnt;
begin
 if (gethostname(lpszLocal, SizeOf(lpszLocal)) = 0) then
  begin
   pheAddr:=gethostbyname(lpszLocal);
   if Assigned(pheAddr) then
    begin
     Move(pheAddr^.h_addr_list^^, lpipAddr^, 4);
     result:=True;
    end
   else
    result:=False;
  end
 else
  result:=False;
end;

procedure TFirewallForm.AddFilter(inP: Boolean;
  lpszRemote: PChar; protoType: DWORD; lpszPort: PChar);
  //додати фільтр
var
 ipFlt : PF_FILTER_DESCRIPTOR;
 dwPort : Integer;
 ipDest : TIpBytes;
 ipSrcMask : TIpBytes;
 ipDstMask :TIpBytes;
begin
 ZeroMemory(@ipFlt, SizeOf(ipFlt));

 ipFlt.dwFilterFlags:=FD_FLAGS_NOSYN;
```

```
 ipFlt.dwRule:=0;
 ipFlt.pfatType:=PF_IPV4;
 ipFlt.fLateBound:=0;

 ipFlt.dwProtocol:=protoType;

 if Assigned(lpszPort) then
  dwPort:=StrToIntDef(lpszPort, FILTER_TCPUDP_PORT_ANY)
 else
  dwPort:=FILTER_TCPUDP_PORT_ANY;

 if inP then
  begin
   ipFlt.wDstPort:=FILTER_TCPUDP_PORT_ANY;
   ipFlt.wDstPortHighRange:=FILTER_TCPUDP_PORT_ANY;
   ipFlt.wSrcPort:=dwPort;
   ipFlt.wSrcPortHighRange:=dwPort;
  end
 else
  begin
   ipFlt.wDstPort:=dwPort;
   ipFlt.wDstPortHighRange:=dwPort;
   ipFlt.wSrcPort:=FILTER_TCPUDP_PORT_ANY;
   ipFlt.wSrcPortHighRange:=FILTER_TCPUDP_PORT_ANY;
  end;

 StrToIP('255.255.255.0', @ipSrcMask);
 StrToIP('255.255.255.0', @ipDstMask);

 if inP then
  begin
   if Assigned(lpszRemote) then
    begin
     ipFlt.SrcAddr:=PByteArray(StrToIp(lpszRemote, @ipDest));
     ipFlt.SrcMask:=@ipSrcMask;
    end
   else
    begin
     ipFlt.SrcAddr:=PByteArray(StrToIp('0.0.0.0', @ipDest));
     StrToIP('0.0.0.0', @ipSrcMask);
     ipFlt.SrcMask:=@ipSrcMask;
    end;
   ipFlt.DstAddr:=@ipLocal;
   ipFlt.DstMask:=@ipDstMask;
   PfAddFiltersToInterface(hIF, 1, @ipFlt, 0, nil, nil);
  end
 else
  begin
   ipFlt.SrcAddr:=@ipLocal;
   ipFlt.SrcMask:=@ipSrcMask;
   if Assigned(lpszRemote) then
    begin
     ipFlt.DstAddr:=PByteArray(StrToIp(lpszRemote, @ipDest));
     ipFlt.DstMask:=@ipDstMask;
    end
   else
    begin
     ipFlt.DstAddr:=PByteArray(StrToIp('0.0.0.0', @ipDest));
     StrToIP('0.0.0.0', @ipDstMask);
     ipFlt.DstMask:=@ipDstMask;
    end;
   PfAddFiltersToInterface(hIF, 0, nil, 1, @ipFlt, nil);
 end;
end;


procedure TFirewallForm.btStartFilterClick(Sender: TObject);
var
 wsaData:       TWSAData;
```

```
begin
//ініціалізація бібліотеки winsock
 if (WSAStartup(MakeWord(1, 1), wsaData) <> 0) then
  begin
   ShowMessage('Помилка Winsock');
   exit;
  end;
//отримати локальний IP
 if not GetLocalIPAddr(@ipLocal) then
  exit;

 //Створення інтерфейсу
 PfCreateInterface(0, PF_ACTION_FORWARD, PF_ACTION_FORWARD, False, True, hIF);

 // Додавання фільтрів
 AddFilter(true, '192.168.1.1', FILTER_PROTO_TCP, nil);
 AddFilter(true, '192.168.8.57', FILTER_PROTO_TCP, '21');
 AddFilter(false, '192.168.1.3', FILTER_PROTO_ANY, '7');
 AddFilter(true, '192.168.1.4', FILTER_PROTO_UDP, '1024');

 // Блокування любих вихідних звертань до 80-го порту
 AddFilter(false, nil, FILTER_PROTO_TCP, '21');

 // Прив'язати інтерфейс до локального адреса
 PfBindInterfaceToIPAddress(hIF, PF_IPV4, @ipLocal);

 btStopFilter.Enabled:=true;
end;

procedure TFirewallForm.btStopFilterClick(Sender: TObject);
//зупинити фільтрацію
begin
 PfUnBindInterface(hIF);
 PfDeleteInterface(hIF);
//відключитись від бібліотеки
 WSACleanup;
 btStopFilter.Enabled:=false;
end;

procedure TFirewallForm.Button1Click(Sender: TObject);
begin
      //Запускаємо модуль HTTP proxy сервера
        FillChar(StartInfo, Sizeof(StartInfo), #0);
        StartInfo.cb := Sizeof(StartInfo);
        StartInfo.dwFlags := STARTF_USESHOWWINDOW;
        StartInfo.wShowWindow := SW_SHOWNORMAL;
        if not CreateProcess(nil,'HTTPProxy.exe', nil,
                             nil, false, CREATE_NEW_CONSOLE or
HIGH_PRIORITY_CLASS,
                             nil, nil, StartInfo, ProcInfo)
        then ShowMessage(IntToStr(GetLastError));

end;

procedure TFirewallForm.Button2Click(Sender: TObject);
//завантаження модуля мережної активності
begin
        FillChar(StartInfo, Sizeof(StartInfo), #0);
        StartInfo.cb := Sizeof(StartInfo);
        StartInfo.dwFlags := STARTF_USESHOWWINDOW;
        StartInfo.wShowWindow := SW_SHOWNORMAL;
        if not CreateProcess(nil,'MonitorSession.exe', nil,
                             nil, false, CREATE_NEW_CONSOLE or
HIGH_PRIORITY_CLASS,
                             nil, nil, StartInfo, ProcInfo)
        then ShowMessage(IntToStr(GetLastError));

end;
```

```
procedure TFirewallForm.Button3Click(Sender: TObject);
//завантаження модуля моніторинг файлів
begin
        FillChar(StartInfo, Sizeof(StartInfo), #0);
        StartInfo.cb := Sizeof(StartInfo);
        StartInfo.dwFlags := STARTF_USESHOWWINDOW;
        StartInfo.wShowWindow := SW_SHOWNORMAL;
        if not CreateProcess(nil,'MonitorResource.exe', nil,
                             nil, false, CREATE_NEW_CONSOLE or
HIGH_PRIORITY_CLASS,
                             nil, nil, StartInfo, ProcInfo)
        then ShowMessage(IntToStr(GetLastError));

end;

end.
```

HTTPProxy.dpr

```
program HTTPProxy;

uses
  Forms,
  MainUnit in 'MainUnit.pas' {HTTPProxyForm},
  ServerThreadUnit in 'ServerThreadUnit.pas',
  ClientThreadUnit in 'ClientThreadUnit.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(THTTPProxyForm, HTTPProxyForm);
  Application.Run;
end.
```

MainUnit.pas

```
unit MainUnit;

interface

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

type
  THTTPProxyForm = class(TForm)
    bnStart: TButton;
    Label1: TLabel;
    edPort: TEdit;
    Label2: TLabel;
    edExtProxyAddr: TEdit;
    Label3: TLabel;
    edExtProxyPort: TEdit;
    procedure bnStartClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  HTTPProxyForm: THTTPProxyForm;

implementation

uses ServerThreadUnit;
```

```
{$R *.dfm}

procedure THTTPProxyForm.bnStartClick(Sender: TObject);
var
 st:TServerThread;
begin
 st:=TServerThread.Create(true);
 st.iLocalPort:=StrToIntDef(edPort.Text, 8088);
 st.iExtProxyPort:=StrToIntDef(edExtProxyPort.Text, 8080);
 st.sExtProxyAddr:=edExtProxyAddr.Text;
 st.Resume;
end;

procedure THTTPProxyForm.FormCreate(Sender: TObject);
var
 wData : WSADATA;
begin
 // Завантаження WinSock
 if WSAStartup(MAKEWORD(1,1), wData) <> 0 then
      begin
   MessageBox(0, 'Бібліотека WinSock не завантажилась', 'Помилка', 0);
   exit;
      end;
end;

procedure THTTPProxyForm.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
var
 st:TServerThread;
//закрити проксі при виході
begin
  st:=TServerThread.Create(False);
end;

end.
```

ServerThreadUnit.pas

```
unit ServerThreadUnit;

interface

uses
  Classes, winsock, windows;

type
  TServerThread = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
  public
   iLocalPort, iExtProxyPort:Integer;
   sExtProxyAddr:String;
  end;

implementation

uses ClientThreadUnit;

procedure TestWinSockError(S:String);
var
 iErr:Integer;
 sFullErr:String;
begin
 sFullErr:='Невідома помилка';
 iErr:=WSAGetLastError();
```

```
 case iErr of
  WSANOTINITIALISED: sFullErr:='Потрібно спочатку викликати функцію WSASturtup,
а потім створювати сокет';
      WSAENETDOWN: sFullErr:='Зв''язок порушений, можливі причини - відійшов
кабель або відключилися від Інтернету ';
      WSAEADDRINUSE: sFullErr:='Вказана адреса вже використовується';
      WSAEFAULT: sFullErr:='Параметри name і namelen не відповідають вибраній
адресації. Параметр namelen може бути менше необхідного значення, а name містити
некоректні дані';
      WSAEINPROGRESS: sFullErr:='Виконується операція в блокуючому режимі. Ви
вже запустили на виконання якусь функцію і потрібно дочекатися завершення її
роботи';
      WSAEINVAL: sFullErr:='Сокет вже пов''язаний з адресою';
      WSAENOBUFS: sFullErr:='Недостатньо буферів, дуже багато з''єднань ';
      WSAENOTSOCK: sFullErr:='Невірний дескриптор сокета';
  WSAEISCONN: sFullErr:='Сокет вже підключений';
  WSAEMFILE: sFullErr:='Немає більше доступних дескрипторів';
 end;

 MessageBox(0, PChar('Помилка у функції '+S+' - '+sFullErr), ' Помилка ', 0);
end;

function TestFuncError(iErr:Integer; FuncName:String):Boolean;
//обробка помилок
begin
 Result:=false;
 if iErr = SOCKET_ERROR then
  begin
   TestWinSockError(FuncName);
   Result:=true;
  end;
end;

procedure TServerThread.Execute;
var
 sServerListen, stClientSocket : TSOCKET;
 localaddr : sockaddr_in;
 ct : TClientThread;
begin
 // Створення сокета
 sServerListen := socket(AF_INET, SOCK_STREAM, 0);
 if sServerListen = INVALID_SOCKET then
  begin
   MessageBox(0, 'Помилка створення сокета ', 'Помилка', 0);
   exit;
  end;

 // Запонення структури адреси
 localaddr.sin_addr.s_addr := htonl(INADDR_ANY);
 localaddr.sin_family := AF_INET;
 localaddr.sin_port := htons(iLocalPort);

 // Пов'язання сокета з локальною адресою
 if TestFuncError(bind(sServerListen, localaddr, sizeof(localaddr)), 'bind')
then
  exit;
 //запустити прослуховування
 if TestFuncError(listen(sServerListen, 4), 'Listen') then
  exit;
 //цикл обробки підключень від клієнта
 while true do
  begin
  //прийняти з'єднання
   stClientSocket := accept(sServerListen, nil, nil);
   if stClientSocket=INVALID_SOCKET then
    continue;
   //створити поток для роботи з кліентом
   ct:=TClientThread.Create(true);
   ct.stClient := stClientSocket;
```

```
   ct.iExtProxyPort := iExtProxyPort;
   ct.sExtProxyAddr := sExtProxyAddr;
   ct.Resume;
  end;
end;

end.



unit ClientThreadUnit;

interface

uses
  Classes, winsock, sysutils, windows;

type
  TClientThread = class(TThread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
  public
    iExtProxyPort:Integer;
    sExtProxyAddr:String;
    stClient:TSocket;
  end;

implementation

{ TClientThread }

function LookupName(name:String): TInAddr;
//функція переводу адреси в необхідний формат
var
 HostEnt: PHostEnt;
 InAddr: TInAddr;
begin
//якщо в текст є "." те введень IP адреса
 if name[4]='.' then
  //перетворюємо у формат
  InAddr.s_addr := inet_addr(PChar(name))
 else
  begin
  //так як точки не має, тому введено ім'я машини
  //отримуємо IP адреса по імені машини
  HostEnt := gethostbyname(PChar(name));

  FillChar(InAddr, SizeOf(InAddr), 0);
  if HostEnt <> nil then
   begin
    with InAddr, HostEnt^ do
     begin
      S_un_b.s_b1 := h_addr^[0];
      S_un_b.s_b2 := h_addr^[1];
      S_un_b.s_b3 := h_addr^[2];
      S_un_b.s_b4 := h_addr^[3];
     end;
   end
  end;
  Result := InAddr;
end;

procedure SendStr(s:TSocket; str:String);
var
 sRecvBuff : array [0..255] of char;
 TempStr : AnsiString;
begin
```

```
 TempStr:=str+#13+#10;
 CopyMemory(@sRecvBuff, PChar(TempStr), Length(TempStr));
 send(s, sRecvBuff, Length(TempStr), 0);
end;

procedure TClientThread.Execute;
var
 Buff: array [0..1024] of char;
 iPort: Integer;
 sRequest, sHost:String;
 server_addr : sockaddr_in;
 sock_server : TSocket;
 iMode, iSize : Integer;
 rfds : TFDSET;
begin
 ////////////////////////////
 //Прочитування заголовка
 Recv(stClient, Buff, 1024, 0);

 sRequest:=String(Buff);

 // Немає заголовка
 if sRequest='' then
  begin
   CloseSocket(stClient);
   exit;
  end;

 ////////////////////////////
 // Видаляємо адресу сервера і порт
 sHost:=Copy(sRequest, Pos('Host: ', sRequest), 255);
 Delete(sHost, Pos(#13, sHost), 255);
 Delete(sHost, 1, 6);
 iPort:=StrToIntDef(Copy(sHost, Pos(':', sHost)+1, 255), 80);
 Delete(sHost, Pos(':', sHost), 255);

 // Якщо не знайдений host то помилка
 if sHost='' then
  begin
   SendStr(stClient, 'HTTP/1.0 400 Invalid header received from browser');
   CloseSocket(stClient);
   exit;
  end;

 // Якщо є зовнішній проксік, то перенаправляємо на нього
 if sExtProxyAddr<>'' then
  begin
    iPort := iExtProxyPort;
    sHost := sExtProxyAddr;
  end;

 sock_server := socket(AF_INET, SOCK_STREAM, 0);

 // Шукаємо сервер/проксік
 server_addr.sin_addr.s_addr := htonl(INADDR_ANY);
 server_addr.sin_family := AF_INET;
 server_addr.sin_port := htons(iPort);
 server_addr.sin_addr := LookupName(sHost);

 //З'єднання з сервером
 if connect(sock_server, server_addr, sizeof(server_addr))=SOCKET_ERROR then
  begin
   SendStr(stClient, '404 Host Not Found');
   exit;
  end;

 iMode:=1;
 setsockopt(sock_server, IPPROTO_TCP, TCP_NODELAY, @iMode, sizeof (integer));
```

```
 // Перенаправляємо запит серверу або іншому проксі
 send(sock_server, buff, strlen(buff),0);

 // Тепер працюємо посередником між клієнтом і сервером
 // передаючи запитані дані
 while true do
  begin
   // Додаємо сокети в набір для очікування
   FD_ZERO(rfds);
   FD_SET(stClient, rfds);
   FD_SET(sock_server, rfds);


   if (select(0, @rfds, nil, nil, nil)< 0) then
    exit;

   // Якщо прийшов запит від клієнта
   // то перенаправляємо серверу
   if(FD_ISSET(stClient, rfds)) then
      begin
      iSize := recv(stClient, buff, sizeof(buff), 0);

      if iSize=-1 then break;

      Send(sock_server, buff, iSize, 0);
      continue;
      end;

   // Якщо прийшли дані від сервера
   // то перенаправляємо клієнтові
   if(FD_ISSET(sock_server, rfds)) then
      begin
      iSize := recv(sock_server, buff, sizeof(buff), 0);

      // Сервер вже все вислав?
      if iSize=0 then
       exit;

      Send(stClient, buff, iSize, 0);
      continue;
      end;
  end;
 CloseSocket(stClient);
 CloseSocket(sock_server);
end;

end.
```

MonitorSession.dpr

```
program MonitorSession;

uses
  Forms,
  MainUnit in 'MainUnit.pas' {MonitorForm};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMonitorForm, MonitorForm);
  Application.Run;
end.



unit MainUnit;

interface
```

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

type
  TMonitorForm = class(TForm)
    lwSessions: TListView;
    bnShow: TButton;
    bnKill: TButton;
    Timer1: TTimer;
    procedure bnShowClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure bnKillClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
    bNT:Boolean;
    fHandleNT, fHandle9x : THandle;
    SessionKeys: array [0..1024] of Integer;
  public
    { Public declarations }
  end;

var
  MonitorForm: TMonitorForm;

implementation

{$R *.dfm}

function TimeToStr(Value: DWORD): String;
var
 d, h, m, s: Integer;
begin
 d:=0;
 h:=0;
 m:=0;
 s:=Value;

 ////////////////////////////////////////////////
 // Перетворення числового значення часу в строку //
 ////////////////////////////////////////////////

 // Обробка секунд
 if s > 59 then
  begin
   m:=s div 60;
   s:=s mod 60;
  end;

 // Обробка хвилин
 if m > 59 then
  begin
   h:=m div 60;
   m:=m mod 60;
  end;

 // Обробка годин
 if h > 23 then
  begin
   d:=h div 24;
   h:=h div 24;
  end;

 Result:='';

 ////////////////////////////////////
 //     Формування строки          //
```

```
 ////////////////////////////////////
 // Дні
 if (d>0) then
  Result:=Result+IntToStr(d)+' д. ';

 // Години
 if (h<10) then
  Result:=Result+'0'+IntToStr(h)+':'
 else
  Result:=Result+IntToStr(h)+':';

 //Хвилини
 if (m<10) then
  Result:=Result+'0'+IntToStr(m)+':'
 else
  Result:=Result+IntToStr(m)+':';

 // Секунди
 if (s<10) then
  Result:=Result+'0'+IntToStr(s)
 else
  Result:=Result+IntToStr(s);
end;

procedure TMonitorForm.bnShowClick(Sender: TObject);
//поновлення інформації про підключених користувачів
var
 SessionInfo50: array [0..512] of TSessionInfo50;
 SessionInfo502 : PSessionInfo502Array;
 TotalEntries,EntriesReadNT: DWORD;
 EntriesRead,TotalAvial: Word;
 i:integer;
begin
//очистка структури відображення
 lwSessions.Items.Clear;
//якщо користувач NT системи
 if bNT then
  begin
   // для NT
   SessionInfo502 := nil;
   //зчитуємо підключення до ПК
   if NetSessionEnumNT(nil, nil, nil, 502, @SessionInfo502,
        DWORD(-1), @entriesreadNT, @totalentries, nil)<>0 then
    exit;
   //заповнюємо структуру відображення даними з структури SessionInfo502
   for i:=0 to EntriesReadNT-1 do
    begin
     with lwSessions.Items.Add do
      begin
       Caption := string(SessionInfo502^[i].sesi502_cname);
       SubItems.Add(SessionInfo502^[i].sesi502_username);
       SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
       SubItems.Add(TimeToStr(SessionInfo502^[i].Sesi502_Time));
       SubItems.Add(TimeToStr(SessionInfo502^[i].sesi502_idle_time));
       SubItems.Add(SessionInfo502^[i].Sesi502_cltype_name);
       SubItems.Add(SessionInfo502^[i].Sesi502_transport);
      end;
    end;
  end
 else
  begin
   //якщо користувач 9x системи
   //отримуємо список підключень
   if NetSessionEnum(nil, 50, @SessionInfo50, SizeOf(SessionInfo50),
          @EntriesRead, @TotalAvial) <> 0 then
    exit;
   //заповнюємо структуру відображення даних з структури  SessionInfo50
   for i:=0 to EntriesRead-1 do
    begin
```

```
     with lwSessions.Items.Add do
      begin
       Caption := string(SessionInfo50[i].Sesi50_cname);
       SubItems.Add(SessionInfo50[i].Sesi50_username);
       SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens));
       SubItems.Add(TimeToStr(SessionInfo50[i].Sesi50_Time));
       SubItems.Add(TimeToStr(SessionInfo50[i].sesi50_idle_time));
       SessionKeys[i]:= SessionInfo50[i].sesi50_key;
      end;
   end;
 end;
end;

procedure TMonitorForm.FormShow(Sender: TObject);
var
 ver: TOSVersionInfo;
begin
//візначення типу платформи
 ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
 GetVersionEx(Ver);
 case Ver.dwPlatformId of
 //якщо NT система
  VER_PLATFORM_WIN32_NT       : bNT := True;
  //якщо 9-х система
  VER_PLATFORM_WIN32_WINDOWS  : bNT := False;
  VER_PLATFORM_WIN32s         : bNT := False
 end;
 //завантаження бібліотеки
 if bNT then
  begin
   fHandleNT := LoadLibrary('NETAPI32.DLL');
   @NetSessionEnumNT := GetProcAddress(fHandleNT, 'NetSessionEnum');
   @NetSessionDelNT := GetProcAddress(fHandleNT, 'NetSessionDel');
  end
 else
  begin
  //завантаження бібліотеки
   fHandle9x := LoadLibrary('SVRAPI.DLL');
   @NetSessionEnum := GetProcAddress(fHandle9x, 'NetSessionEnum');
   @NetSessionDel := GetProcAddress(fHandle9x, 'NetSessionDel');
  end;
end;

procedure TMonitorForm.bnKillClick(Sender: TObject);
//знищити з'єднання
var
 wcNameNT: PWideChar;
begin
 if lwSessions.Selected=nil then
  exit;
 //для NT
 if bNT then
  begin
   wcNameNT :=
PWChar(WideString('\\'+lwSessions.Items.Item[lwSessions.Selected.Index].Caption)
);
   NetSessionDelNT(nil,wcNameNT,nil);
  end
 else
  begin
  //для 9x
   NetSessionDel(
      nil,
      PAnsiChar(lwSessions.Items.Item[lwSessions.Selected.Index].Caption),
      SessionKeys[lwSessions.Selected.Index]
      );
  end;
end;
```

```
procedure TMonitorForm.Timer1Timer(Sender: TObject);
begin
  bnShow.Click;
end;

end.
```

MonitorResource.dpr

```
program MonitorResource;

uses
  Forms,
  MainUnit in 'MainUnit.pas' {MonitorForm};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMonitorForm, MonitorForm);
  Application.Run;
end.
```

MainUnit.pas

```
unit MainUnit;

interface

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

type
  TMonitorForm = class(TForm)
    lwResources: TListView;
    bnShow: TButton;
    bnKill: TButton;
    Timer1: TTimer;
    procedure bnShowClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure bnKillClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
    bNT:Boolean;
    fHandleNT, fHandle9x : THandle;
  public
    { Public declarations }
  end;

var
  MonitorForm: TMonitorForm;

implementation

{$R *.dfm}

procedure TMonitorForm.bnShowClick(Sender: TObject);
var
 FileInfoNT: PFileInfo3Array;
 FileInfo9x: array [0..1024] of TFileInfo502;
 i, iTE, iRE: Integer;
begin
 //очистка структури відображення
 lwResources.Items.Clear;

 if bNT then
  begin
```

```
   // якщо NT
   FileInfoNT := nil;
   if NetFileEnumNT(nil, nil, nil, 3, @FileInfoNT, DWORD(-1),
       @iRE, @iTE, nil)<>0 then
    exit;
   //внесення даних з отриманої структури FileInfoNT
   //в структуру візуалізації даних на формі
   for i:=0 to iRE-1 do
    begin
     with lwResources.Items.Add do
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id));
        SubItems.Add(FileInfoNT^[i].fi3_pathname);
        SubItems.Add(FileInfoNT^[i].fi3_username);
        SubItems.Add(IntToStr(FileInfoNT^[i].fi3_num_locks));
      end;
    end;
  end
 else
  begin
   // для Windows 9x
   if NetFileEnum(nil, nil, 50, @FileInfo9x, SizeOf(FileInfo9x),
       @iRE, @iTE)<> 0 then
     exit;
   //внесення даних з отриманої структури FileInfoNT
   //в структуру візуалізації даних на формі

   for i:=0 to iRE-1 do
    begin
     with lwResources.Items.Add do
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi502_id));
        SubItems.Add(FileInfo9x[i].fi502_pathname);
        SubItems.Add(FileInfo9x[i].fi502_username);
      end;
    end;
 end;
end;

procedure TMonitorForm.FormShow(Sender: TObject);
var
 ver: TOSVersionInfo;
begin
//визначення версії ОС
 ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
 GetVersionEx(Ver);
 case Ver.dwPlatformId of
 //для NT систем
  VER_PLATFORM_WIN32_NT       : bNT := True;
  //для 9x систем
  VER_PLATFORM_WIN32_WINDOWS  : bNT := False;
  VER_PLATFORM_WIN32s         : bNT := False
 end;
 //завантаження бібліотеки в залежності від ОС
 if bNT then
  //для NT
  begin
   fHandleNT := LoadLibrary('NETAPI32.DLL');
   @NetFileEnumNT := GetProcAddress(fHandleNT, 'NetFileEnum');
   @NetFileCloseNT := GetProcAddress(fHandleNT, 'NetSessionClose');
  end
 else
  begin
  //для 9x систем
   fHandle9x := LoadLibrary('SVRAPI.DLL');
   @NetFileEnum := GetProcAddress(fHandle9x, 'NetFileEnum');
   @NetFileClose := GetProcAddress(fHandle9x, 'NetFileClose2');
  end;
end;
```

```
procedure TMonitorForm.bnKillClick(Sender: TObject);
begin
//якщо відкритих немає то вихід
 if lwResources.Selected=nil then exit;
 //закрити ресурс
 if bNT then
  NetFileCloseNT(nil,StrToInt(lwResources.Selected.Caption))
 else
  NetFileClose(nil,StrToInt(lwResources.Selected.Caption));
end;

procedure TMonitorForm.Timer1Timer(Sender: TObject);
//обробка команди показати
//поновлення даних моніторингу на формі через 3 сек
begin
  bnShow.Click;
end;

end.


proControl.dpr

//файл проекту реалізації користувачської бібліотеки
library proControl;

uses
  Windows, pSender;
(* ------------------------------------------------------------------------ *)
const
  THREAD_ALL_ACCESS      = $001F03FF;
  THREAD_SUSPEND_RESUME  = $00000002;
  TH32CS_SNAPTHREAD      = $04;
  MutexName              = '_BLACKPROCESSCONTROLMUTEX';
  proCapt                = 'Проактивний захист';

const
  faReadOnly  = $00000001 platform;
  faHidden    = $00000002 platform;
  faSysFile   = $00000004 platform;
  faVolumeID  = $00000008 platform;
  faDirectory = $00000010;
  faArchive   = $00000020 platform;
  faSymLink   = $00000040 platform;
  faAnyFile   = $0000003F;

type
  PFunctionRestoreData = ^ TFunctionRestoreData;
  TFunctionRestoreData = packed record
    Address:Pointer;
    val1:Byte;
    val2:DWORD;
   end;

  TTHREADENTRY32 = packed record
    dwSize: DWORD;
    cntUsage: DWORD;
    th32ThreadID: DWORD;
    th32OwnerProcessID: DWORD;
    tpBasePri: Longint;
    tpDeltaPri: Longint;
    dwFlags: DWORD;
   end;

  TFileName = type string;
  TSearchRec = record
    Time: Integer;
    Size: Integer;
```

```
    Attr: Integer;
    Name: TFileName;
    ExcludeAttr: Integer;
    FindHandle: THandle  platform;
    FindData: TWin32FindData  platform;
  end;

  LongRec = packed record
    case Integer of
      0: (Lo, Hi: Word);
      1: (Words: array [0..1] of Word);
      2: (Bytes: array [0..3] of Byte);
  end;

Function OpenThread(dwDesiredAccess: dword; bInheritHandle: bool;
                    dwThreadId: dword): dword; stdcall; external 'kernel32.dll';

function CreateToolhelp32Snapshot(dwFlags, th32ProcessID: DWORD): dword;
         stdcall;external 'kernel32.dll';

function Thread32First(hSnapshot: THandle; var lpte: TThreadEntry32): BOOL;
         stdcall;external 'kernel32.dll';

function Thread32Next(hSnapshot: THandle; var lpte: TThreadENtry32): BOOL;
         stdcall;external 'kernel32.dll';

var
  SystemCreateProcW : TFunctionRestoreData;
  SystemCreateProcA : TFunctionRestoreData;
  SystemCreateFileA : TFunctionRestoreData;
  SystemCreateFileW : TFunctionRestoreData;
  SystemCopyFileA   : TFunctionRestoreData;
  SystemDeleteFileA : TFunctionRestoreData;
  {!}
  hDll          : integer;
  LastFile      : string;
  StrangeList   : string;
  CopyesList    : string;
  StrangeFiles  : integer = 0;
  CopyesNumber  : integer = 0;
  ShowDialog    : function(App,Path: PChar; dType: byte): integer; stdcall;
  ShowDialogPro : function (App,Problem: PChar; dType: byte): integer; stdcall;
  CurrentModule : array [0..MAX_PATH] of char;

(* ------------------------------------------------------------------------ *)
procedure InstallMutex;
var
  M:THandle;
begin
//створюємо мьютекс
  m:=CreateMutex(0,false,MutexName);
  //якщо не вдало то вихід
  if m=0 then exit;
end;
(* ------------------------------------------------------------------------ *)
function ExpandFileName(const FileName: string): string;
//расширяємо ім'я файлу
var
  FName: PChar;
  Buffer: array[0..MAX_PATH - 1] of Char;
begin
//змінюємо довжину строки
  SetString(Result, Buffer, GetFullPathName(PChar(FileName), SizeOf(Buffer),
    Buffer, FName));
end;
(* ------------------------------------------------------------------------ *)
procedure FindClose(var F: TSearchRec);
begin
  //якщо структура відкрита
```

```
  if F.FindHandle <> INVALID_HANDLE_VALUE then
  begin
    //закрити структуру
    Windows.FindClose(F.FindHandle);
    F.FindHandle := INVALID_HANDLE_VALUE;
  end;
end;

function FindMatchingFile(var F: TSearchRec): Integer;
//знаходимо відповідність файлу
var
  LocalFileTime: TFileTime;
begin
  with F do
  begin
    while FindData.dwFileAttributes and ExcludeAttr <> 0 do
      //якщо немає наступного файлу
      if not FindNextFile(FindHandle, FindData) then
      begin
      //вихід
        Result := GetLastError;
        Exit;
      end;
    //перетворюємо час файлу
    FileTimeToLocalFileTime(FindData.ftLastWriteTime, LocalFileTime);
    FileTimeToDosDateTime(LocalFileTime, LongRec(Time).Hi,
      LongRec(Time).Lo);
      //записуємо дані файла
    Size := FindData.nFileSizeLow;
    Attr := FindData.dwFileAttributes;
    Name := FindData.cFileName;
  end;
  Result := 0;
end;

function FindFirst(const Path: string; Attr: Integer;
//знайти по масці
  var  F: TSearchRec): Integer;
const
  faSpecial = faHidden or faSysFile or faVolumeID or faDirectory;
begin
//любий атрибут
  F.ExcludeAttr := not Attr and faSpecial;
  //пошук і по імені і по даті
  F.FindHandle := FindFirstFile(PChar(Path), F.FindData);
  if F.FindHandle <> INVALID_HANDLE_VALUE then
  begin
    //якщо знайдений то 0
    Result := FindMatchingFile(F);
    if Result <> 0 then FindClose(F);
  end else
    Result := GetLastError;
end;
(* ---------------------------------------------------------------------------- *)
function StrPas(const Str: PChar): string;
//перетворення в строку
begin
  Result := Str;
end;

function LowCase(ACh:Char): Char;
//якщо заголовочні символи то перетворимо їх на прописні символи
begin
 Result := ACh;
 if (ACh >= 'A') and (ACh <= 'Z') then inc(Result, 32);
end;

function LowerCase(AStr:string): string;
// якщо є заголовочні символи в строці то перетворимо їх на прописні символи
```

```
var
 LI:Integer;
begin
 Result:=AStr;
 for LI := 1 to Length(Result) do Result[LI] := LowCase(Result[LI]);
end;

function FileAgeEx(const FileName: string): Integer;
var
  Handle: THandle;
  FindData: TWin32FindData;
  LocalFileTime: TFileTime;
begin
  //знаходимо хендл файла
  Handle := FindFirstFile(PChar(FileName), FindData);
  //якщо знайдений то
  if Handle <> INVALID_HANDLE_VALUE then
  begin
    //закриваємо хендл
    Windows.FindClose(Handle);
    if (FindData.dwFileAttributes and FILE_ATTRIBUTE_DIRECTORY) = 0 then
    begin
      //переведення часових даних по створенню файла
      FileTimeToLocalFileTime(FindData.ftLastWriteTime, LocalFileTime);
      if FileTimeToDosDateTime(LocalFileTime, LongRec(Result).Hi,
        LongRec(Result).Lo) then Exit;
    end;
  end;
  Result := -1;
end;

function FileExistsEx(const FileName: string): Boolean;
//якщо не -1 то існує
begin
  Result := FileAgeEx(FileName) <> -1;
end;

function DirectoryExists( const Name: String): Boolean;
//якщо true, то знайдений каталог
asm
   PUSH EBX
        PUSH     EAX
   PUSH SEM_NOOPENFILEERRORBOX or SEM_FAILCRITICALERRORS
   CALL SetErrorMode
   XCHG EBX, EAX
        CALL     GetFileAttributes
        INC      EAX
        JZ       @@exit
        DEC      EAX
        {$IFDEF PARANOIA} DB $24, FILE_ATTRIBUTE_DIRECTORY {$ELSE} AND AL,
FILE_ATTRIBUTE_DIRECTORY {$ENDIF}
        SETNZ    AL
@@exit:
   XCHG EAX, EBX
   PUSH EAX
   CALL SetErrorMode
   XCHG EAX, EBX
   POP  EBX
end;
(* ---------------------------------------------------------------------------- *)
function GetCurrentModulePath: String;

begin
  //повний шлях до модуля
  GetModuleFileName(Hinstance, CurrentModule, MAX_PATH);
  Result := CurrentModule;
end;
(* ---------------------------------------------------------------------------- *)
function ExtractFileExt(AFile: String): String;
```

```
//визначаємо розширення файлу з строки повного шляху до нього
var
  I,J: integer;
begin
  if Length(AFile)<>0 then
  begin
    J := 0;
    for I := Length(AFile) downto 1 do
      if AFile[I] = '.' then
      begin
        J := I;
        break;
      end;
    Result:=Copy(AFile,J,MaxInt);
  end else Result:='';
end;

function ExtractFilePath(APath:string):string;
//визначаємо шлях до файлу з строки повного шляху до нього
var
 LI,LJ:Integer;
begin
 if Length(APath)<>0 then
 begin
  LJ:=0;
  for LI:=Length(APath) downto 1 do
   if APath[LI]='\' then
   begin
    LJ:=LI;
    Break;
   end;
  Result:=Copy(APath,1,LJ);
 end else Result:='';
end;

function ExtractFileName(APath:string):string;
//визначаємо ім'я файлу з строки повного шляху до нього
var
 LI,LJ:Integer;
begin
 if Length(APath)<>0 then
 begin
  LJ:=0;
  for LI:=Length(APath) downto 1 do
   if APath[LI]='\' then
   begin
    LJ:=LI;
    Break;
   end;
  Result:=Copy(APath,LJ+1,MaxInt);
 end else Result:='';
end;
(* ---------------------------------------------------------------------------- *)
function GetStrCn(Str: String; Count: integer): string;
// усікання строки на задану кількість символів
var
  i: integer;
begin
  Result := '';
  for i := 1 to Count do
    Result := Result + Str[i];
end;
(* ---------------------------------------------------------------------------- *)
function RestoreLongName(fn: string): string;
//---------------------------------------------------------------------
// Відновлює довгі імена файлів по відомих коротких (8.3)
// Як аргумент приймає повний або неповний (в т.ч. відносний)
// шлях до файлу, наприклад 'C:\windows\ Рабочи~1\ Ітакда~1.LNK' або
// '..\..\COMMON~1\BORLAN~1\BDE\BDEREA~1.TXT'. Розуміє мережеві імена.
```

```
// Повертає повний(!) шлях типу 'C:\windows\ Робочий стол\ і так далі.lnk',
// 'C:\program Files\common Files\borland Shared\bde\bdereadme.txt',
// '\computer\resource\folder with long name\file with long name.ext'
//---------------------------------------------------------------------
  function LookupLongName(const filename: string): string;
    var
    sr: TSearchRec;
  begin
    if FindFirst(filename, faAnyFile, sr) = 0 then
      Result := sr.Name
    else
      Result := ExtractFileName(filename);
    FindClose(sr);
  end;
  function GetNextFN: string;
    var
    i: integer;
  begin
    Result := '';
    if Pos('\\', fn) = 1 then
    begin
      Result := '\\';
      fn := Copy(fn, 3, length(fn) - 2);
      i := Pos('\', fn);
      if i <> 0 then
      begin
        Result := Result + Copy(fn, 1, i);
        fn := Copy(fn, i + 1, length(fn) - i);
      end;
    end;
    i := Pos('\', fn);
    if i <> 0 then
    begin
      Result := Result + Copy(fn, 1, i - 1);
      fn := Copy(fn, i + 1, length(fn) - i);
    end
    else begin
      Result := Result + fn;
      fn := '';
    end;
  end;

  var
  name: string;
begin
  //повертає повне ім'я файлу
  fn := ExpandFileName(fn);
  Result := GetNextFN;
  repeat
    name := GetNextFN;
    Result := Result + '\' + LookupLongName(Result + '\' + name);
  until length(fn) = 0;
end;
(* ------------------------------------------------------------------------- *)
function TempDir: string;
//визначаємо шлях до каталогу тимчасових файлів
var
      buf: packed array [0..4095] of Char;
begin
      GetTempPath(4096,buf);
  //переводимо в строку
      Result := StrPas(buf);
      Result := buf;
  //відновлення довгого імені по заданй строці
  Result := RestoreLongName(Result);
end;

function SysDir: string;
//визначаємо шлях до системного каталогу
```

```
var
      buf: packed array [0..4095] of Char;
begin
      GetWindowsDirectory(buf,4096);
      Result:=StrPas(buf);
      Result:=buf+'\system32\';
end;

function WinDir: string;
//визначаємо шлях до директорії ОС
var
      buf: packed array [0..4095] of Char;
begin
      GetWindowsDirectory(buf,4096);
      Result:=StrPas(buf)+'\';
end;
(* ------------------------------------------------------------------------- *)
Procedure StopThreads;
//зупинити потоки
var
 h, CurrTh, ThrHandle, CurrPr: dword;
 Thread: TThreadEntry32;
begin
//отримуємо "значення" ідентифікатора потока
 CurrTh := GetCurrentThreadId;
 //ідентифікатор потоку потоку
 CurrPr := GetCurrentProcessId;
 //отримуємо знімок процесів
 h := CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD, 0);
 //якщо успішно то
 if h <> INVALID_HANDLE_VALUE then
   begin
    //отримуємо інформацію про перший процес
    Thread.dwSize := SizeOf(TThreadEntry32);
    if Thread32First(h, Thread) then
    repeat
     if (Thread.th32ThreadID <> CurrTh) and (Thread.th32OwnerProcessID = CurrPr)
then
      begin
       //відкрити процес
       ThrHandle := OpenThread(THREAD_SUSPEND_RESUME, false,
Thread.th32ThreadID);
       if ThrHandle>0 then
         begin
         //призупинити процес
          SuspendThread(ThrHandle);
          CloseHandle(ThrHandle);
         end;
       end;
    until not Thread32Next(h, Thread);
   CloseHandle(h);
   end;
end;

Procedure RunThreads;
//відновити процес
var
 h, CurrTh, ThrHandle, CurrPr: dword;
 Thread: TThreadEntry32;
begin
//отримуємо "значення" ідентифікатора потока
 CurrTh := GetCurrentThreadId;
 //ідентифікатор потоку потоку
 CurrPr := GetCurrentProcessId;
  //отримуємо знімок процесів
 h := CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD, 0);
  //якщо успішно то
 if h <> INVALID_HANDLE_VALUE then
   begin
```

```
   //отримуємо інформацію про перший процес
    Thread.dwSize := SizeOf(TThreadEntry32);
    if Thread32First(h, Thread) then
    repeat
     if (Thread.th32ThreadID <> CurrTh) and (Thread.th32OwnerProcessID = CurrPr)
then
      begin
       //відкрити процес
       ThrHandle := OpenThread(THREAD_SUSPEND_RESUME, false,
Thread.th32ThreadID);
       if ThrHandle>0 then
         begin
         //відновити процес
          ResumeThread(ThrHandle);
          CloseHandle(ThrHandle);
         end;
       end;
    until not Thread32Next(h, Thread);
   CloseHandle(h);
   end;
end;

(* Установка перехоплення функції по її показчику

  Procedure: Setcodehook
  Procaddress,                        покажчик на  цільову функцію
  Newprocaddress:pointer;             покажчик на функцію перехоплювач
  Restoredata:pfunctionrestoredata  Покажчик на міст до старої функції
  Result:    boolean                  успішність *)

function SetCodeHook(ProcAddress, NewProcAddress: pointer;
RestoreDATA:PFunctionRestoreData):boolean;
var
  OldProtect, JMPValue:DWORD;
begin
  Result:=False;
  //змінюємо атрибути захисту вказаного регіону
  //вказаного адресного простору
  //PAGE_EXECUTE_READWRITE -  Надає права читання/запису пам'яті і
  //                          виконання коди для вказаного регіону.

  if not VirtualProtect(ProcAddress,5,PAGE_EXECUTE_READWRITE,OldProtect) then
exit;
  JMPValue := DWORD(NewProcAddress) - DWORD(ProcAddress) - 5;
  RestoreDATA^.val1:= Byte(ProcAddress^);
  RestoreDATA^.val2:= DWORD(Pointer(DWORD(ProcAddress)+1)^);
  RestoreDATA^.Address:=ProcAddress;
  byte(ProcAddress^):=$E9;
  DWORD(Pointer(DWORD(ProcAddress)+1)^):=JMPValue;
  Result:=VirtualProtect(ProcAddress,5,OldProtect,OldProtect);
end;

{-------------------------------------------------------------------------------
   Установка перехвата функції по її імені

  Procedure: SetProcedureHook

  ModuleHandle:HMODULE;             Хендл модуля в якому знаходиться цільова
функція
  ProcedureName:PChar;              Ім'я процедури
  NewProcedureAddress:Pointer;      Покажчик на процедуру перехоплювач
  RestoreDATA:PFunctionRestoreData  Покажчик на міст до старої функції

  Result:    Boolean                Успішність установки перехоплення
-------------------------------------------------------------------------------}
function
SetProcedureHook(ModuleHandle:HMODULE;ProcedureName:PChar;NewProcedureAddress:Po
inter;
   RestoreDATA:PFunctionRestoreData):Boolean;
```

```
var
  ProcAddress:Pointer;
begin
  ProcAddress:=GetProcAddress(ModuleHandle,ProcedureName);
  Result:=SetCodeHook(ProcAddress,NewProcedureAddress,RestoreDATA);
end;

(* Зняття перехоплення функції
  Procedure: Unhookcodehook
  Restoredata:pfunctionrestoredata  адреса моста до старої функції
  Result:    Boolean  *)

function UnHookCodeHook(RestoreDATA:PFunctionRestoreData):Boolean;
var
  ProcAddress:Pointer;
  OldProtect,JMPValue:DWORD;
begin
  Result:=False;
  ProcAddress:=RestoreDATA^.Address;
  if not VirtualProtect(ProcAddress,5,PAGE_EXECUTE_READWRITE,OldProtect) then
exit;
  Byte(ProcAddress^):=RestoreDATA^.val1;
  DWORD(Pointer(DWORD(ProcAddress)+1)^):=RestoreDATA^.val2;
  Result:=VirtualProtect(ProcAddress,5,OldProtect,OldProtect);
end;

(* Закрити процес по його PID ідентифікатору *)
procedure TerminateProcessEx(PID: Integer);
var
 hProcess: integer;
begin
  hProcess := OpenProcess(PROCESS_TERMINATE, false, PID);
  if hProcess > 0 then
  begin
    TerminateProcess(hProcess, 0);
    CloseHandle(hProcess);
  end;
end;
(* ------------------------------------------------------------------------- *)
//показ діалогу з dll бібліотеки(завантаження/вигрузка) і
//обробка програмного дадатку, стан якого контролюється

function ShowProDialog(Path: String; dType: byte) : integer;
var
  PID: dWord;
begin
  hDll :=
LoadLibrary(PChar(ExtractFilePath(GetCurrentModulePath)+'proDialog.dll'));
  if hDll <> 0 then begin
    try
    //виведення вікна - підозріла активність
      @ShowDialogPro:=GetProcAddress(hDll, 'ShowDialogPro');
      //читаємо дані вибору користувача
      Result :=
ShowDialogPro(pChar(ExtractFileName(ParamStr(0))),PChar(Path),dType);
      case Result of
      //закрити процес
        3 : begin
              PID := GetCurrentProcessId;
              if PID <> 0 then begin
                TerminateProcessEx(PID);
              end;
            end;
       //відправка повідомлення, для внесення додатку до білого списку додатків
        9 : sendstring(proCapt,PChar('#LOG2#'+ParamStr(0)));
      end;
    finally
    //очищаємо пам'ять
      FreeLibrary(hDll);
```

```
    end;
  end;
end;

function ShowCanRunDialog(Path: String) : integer;
begin
  hDll :=
LoadLibrary(PChar(ExtractFilePath(GetCurrentModulePath)+'proDialog.dll'));
  if hDll <> 0 then begin
    try
    //виведення вікна - запуск додатку
      @ShowDialog:=GetProcAddress(hDll, 'ShowDialog');
      Result := ShowDialog(pChar(ExtractFileName(ParamStr(0))),PChar(Path),0);
    finally
    //очистка пам'яті
      FreeLibrary(hDll);
    end;
  end;
end;
(* ------------------------------------------------------------------------- *)
function CanRun(FName, CmdLine: String): boolean;
  var
  i: integer;
  j: integer;
  S: string;
  cmd : String;
begin
  Result := True;
    case ShowCanRunDialog(FName) of
      -1 : Result := false;
      //білий список
       9 : begin
             Result := true;
             sendstring(proCapt,PChar('#LOG2#'+FName));
             exit;
           end;
       //дозволений одноразовий запуск
       6 : begin
             Result := true;
             sendstring(proCapt,PChar('#LOG1#'+FName));
             exit;
           end;
       //запуск додатку не дозволений
       3 : begin
             Result := False;
             sendstring(proCapt,PChar('#LOG3#'+FName));
             exit;
           end;
    end;

end;
(* ------------------------------------------------------------------------- *)
function ExistsInList(Str: String; List: String) : Boolean;
//перевіряємо присутність шаблону в рядку
//якщо присутній то true
begin
  Result := False;
  if pos(Str,List) <> 0 then Result := true;
end;

function DeleteInvalidInList(var List: String) : Boolean;
//якщо файл існує то true
const
  Return = #13#10;
var
  i: integer;
  tmp,tmp1 : String;
begin
  Result := False;
```

```
  i    := 1;
  tmp1 := List+Return;
  List := '';
  while i <> 0 do begin
    i := pos(Return,tmp1);
    tmp := copy(tmp1,1,i-1);
    delete(tmp1,1,i+1);
    if FileExistsEx(tmp) then List := List + tmp + Return;
  end;
  Result := true;
end;

function CanDeleteFile(FileName: String) : boolean;
//видалення з системної директорії (так чи ні?)
var
  ClearName: String;
  ext      : String;
  DirName  : String;
  S        : String;
begin
  Result := False;
  //відновлення скорочених імен та розбір шляху
  //файла - відокремлюємо назву додатка і його шлях
  //визначення розширення файлу
  LastFile  := RestoreLongName(FileName);
  Result    := false;
  ClearName := LowerCase(ExtractFileName(FileName));
  Ext       := LowerCase(ExtractFileExt(FileName));
  DirName   := LowerCase(ExtractFilePath(FileName));
  ClearName := GetStrCn(ClearName,length(ClearName)-length(ext));
  delete(dirname,1,(length(DirName)-1)-Length(ClearName));
  //якщо системна директорія
  if (LowerCase(ExtractFilePath(FileName)) = LowerCase(WinDir))  or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(TempDir)) or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(SysDir))  then begin
     //відправка повідомлення для статистики
       sendstring(proCapt,PChar('#LOG6#'+FileName));
       //виведення діалогу "Підозріла активність"
       ShowProDialog(FileName,5);
     end;
end;

function CanCreateFile(FileName: String) : boolean;
//обробка при створенні або копіюванні файла в системній директорії
var
  ClearName: String;
  ext      : String;
  DirName  : String;
begin
 //відновлення скорочених імен та розбір шляху
  //файла - відокремлюємо назву додатка і його шлях
  //визначення розширення файлу
  LastFile  := RestoreLongName(FileName);
  Result    := false;
  ClearName := LowerCase(ExtractFileName(FileName));
  Ext       := LowerCase(ExtractFileExt(FileName));
  DirName   := LowerCase(ExtractFilePath(FileName));
  ClearName := GetStrCn(ClearName,length(ClearName)-length(ext));
  delete(dirname,1,(length(DirName)-1)-Length(ClearName));
  //якщо розширення файла відноситься до запускаємого то
  if (ext = '.exe') or (ext = '.scr') or (ext = '.bat') or (ext = '.cmd') or
(ext = '.com') then
  //якщо системна директорія
  if (LowerCase(ExtractFilePath(FileName)) = LowerCase(WinDir))  or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(TempDir)) or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(SysDir))  then begin
     //відправка повідомлення в файл статистики
       sendstring(proCapt,PChar('#LOG5#'+FileName));
       //виведення діалогу "Підозріла активність"
```

```
      //спроба копіювання в систему
      ShowProDialog(FileName,3);
    end;
   //якщо файли запускаємі
  if (ext = '.exe') or (ext = '.scr') or (ext = '.bat') or (ext = '.cmd') or
(ext = '.com') then
  if (DirName = ClearName+'\') or
(DirectoryExists(ExtractFilePath(FileName)+ClearName)) then begin
    if not ExistsInList(FileName, StrangeList) then begin
      inc(StrangeFiles);
      StrangeList := StrangeList + #13#10 + FileName;
    end;
    //якщо створено більше 5 копій
    if StrangeFiles > 5 then begin
    //відправка повідомлення до файлу статистики
      sendstring(proCapt,PChar('#LOG4#'+ParamStr(0)));
       //виведення діалогу "Підозріла активніть"
       //можливо P2P вірус
      ShowProDialog(StrangeList,4);
      Result := true;
      Exit;
    end;
  end;

  if Length(DirName) = 3 then
    if LowerCase( ExtractFileName(FileName)) = 'autorun.inf' then begin
   //відправка повідомлення до файлу статистики
      sendstring(proCapt,PChar('#LOG7#'+FileName));
    //виведення діалогу "Підозріла активніть"
    //створення підозрілого файлу
      ShowProDialog(FileName,0);
    end else
    if LowerCase( ExtractFileName(FileName)) = 'explorer.exe' then begin
      //відправка повідомлення до файлу статистики
      sendstring(proCapt,PChar('#LOG7#'+FileName));
      //виведення діалогу "Підозріла активніть"
      //створення підозрілого файлу
      ShowProDialog(FileName,6);
    end;
end;

function CanCopySelf(FileName: String) : boolean;
//обробка при копіюванні додатку
var
  ClearName: String;
  ext      : String;
  DirName  : String;
begin
 //відновлення скорочених імен та розбір шляху
  //файла - відокремлюємо назву додатка і його шлях
  //визначення розширення файлу
  LastFile  := RestoreLongName(FileName);
  Result    := false;
  ClearName := LowerCase(ExtractFileName(FileName));
  Ext       := LowerCase(ExtractFileExt(FileName));
  DirName   := LowerCase(ExtractFilePath(FileName));
  ClearName := GetStrCn(ClearName,length(ClearName)-length(ext));
  delete(dirname,1,(length(DirName)-1)-Length(ClearName));

  DeleteInvalidInList(CopyesList);
  if not ExistsInList(FileName,CopyesList) then begin
    CopyesList := CopyesList + FileName + #13#10;
    Inc(CopyesNumber);
  end;
  //якщо більше 7 копій то
  if CopyesNumber > 7 then begin
  //відправка повідомлення
    sendstring(proCapt,PChar('#LOG4#'+ParamStr(0)));
     //виведення діалогу "Підозріла активніть"
```

```
      //можливо P2P вірус
    ShowProDialog(CopyesList,1);
    Exit;
  end;

  if (DirName = ClearName+'\') or
(DirectoryExists(ExtractFilePath(FileName)+ClearName)) then begin
    if not ExistsInList(FileName, StrangeList) then begin
      inc(StrangeFiles);
      StrangeList := StrangeList + FileName + #13#10;
    end;
    //якщо більше 5 копій
    if StrangeFiles > 5 then begin
      //відправка повідомлення в файл статистики
      sendstring(proCapt,PChar('#LOG4#'+ParamStr(0)));
      //виведення діалогу "Підозріла активність"
      //можливо P2P вірус
      ShowProDialog(StrangeList,4);
      Result := true;
      Exit;
    end;
  end;
  //якщо в системну директорію
  if (LowerCase(ExtractFilePath(FileName)) = LowerCase(WinDir))  or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(TempDir)) or
     (LowerCase(ExtractFilePath(FileName)) = LowerCase(SysDir))  then begin
       //відправка повідомлення
       sendstring(proCapt,PChar('#LOG5#'+FileName));
      //виведення діалогу "Підозріла активність"
      //спроба копіювання в системну директорію
       ShowProDialog(FileName,2);
     end;
end;
(* ------------------------------------------------------------------------ *)
//обробка пастки на
//відкриття файлу чи каталогу
function CreateFileANext(lpFileName: PAnsiChar; dwDesiredAccess, dwShareMode:
DWORD;
                         lpSecurityAttributes: PSecurityAttributes;
dwCreationDisposition, dwFlagsAndAttributes: DWORD;
                        hTemplateFile: THandle): THandle; stdcall;
var
  isNewFile: Boolean;
begin
  //наявність файла
  if not FileExistsEx(lpFileName) then isNewFile := true else
  begin
    isNewFile := false;
  end;
  //знімаємо перехоплення функції
  UnHookCodeHook(@SystemCreateFileA);

  try
    //виконуємо функцію відкриття
    result := CreateFileA(lpFileName,dwDesiredAccess,dwShareMode,
                          lpSecurityAttributes,dwCreationDisposition,
                          dwFlagsAndAttributes,hTemplateFile);

  except
  end;
   //якщо файл є і він створюється
  if FileExistsEx(lpFileName) and isNewFile then begin
    //обробляємо файл
    CanCreateFile(lpFileName);
    LastFile := RestoreLongName(lpFileName);
  end;
    //ставимо пастку
    SetCodeHook(SystemCreateFileA.Address,@CreateFileANext,@SystemCreateFileA);
end;
```

```
(* ------------------------------------------------------------------------ *)
//обробка пастки на
//відкриття файлу чи каталогу
function CreateFileWNext(lpFileName: PWideChar; dwDesiredAccess, dwShareMode:
DWORD;
                         lpSecurityAttributes: PSecurityAttributes;
dwCreationDisposition, dwFlagsAndAttributes: DWORD;
                         hTemplateFile: THandle): THandle; stdcall;
var
  isNewFile: Boolean;
begin
  //визначаємо створюємо файл чи ні
  if not FileExistsEx(lpFileName) then isNewFile := true else
  begin
    isNewFile := false;
  end;
  //знімаємо пастку
  UnHookCodeHook(@SystemCreateFileW);

  try
  //виконужмо команду
    result := CreateFileW(lpFileName,dwDesiredAccess,dwShareMode,
                          lpSecurityAttributes,dwCreationDisposition,
                          dwFlagsAndAttributes,hTemplateFile);

  except
  end;
  //якщо файл є і він новий то
  if FileExistsEx(lpFileName) and isNewFile then begin
  //обробка події створення через функцію захисту
    CanCreateFile(lpFileName);
    LastFile := RestoreLongName(lpFileName);
  end;
    //відновлюємо пастку
    SetCodeHook(SystemCreateFileW.Address,@CreateFileWNext,@SystemCreateFileW);

end;
(* ------------------------------------------------------------------------ *)
function DeleteFileCallBack(lpFileName: PChar): BOOL; stdcall;
//обробка пастки, що виникає при команді видалення файлу
begin
  //знімаємо пастку
  UnHookCodeHook(@SystemDeleteFileA);
  //обробка системою проактивного захисту
  CanDeleteFile(lpFileName);
  try
    //видалення файлу
    result := Windows.DeleteFile(lpFileName);
  except
  end;
  //відновлення пастки
  SetCodeHook(SystemDeleteFileA.Address,@DeleteFileCallBack,@SystemDeleteFileA);
end;
(* ------------------------------------------------------------------------ *)
//обробка пастки, що виникає при обробці системної функції сопіювання файлу
function CopyFileANext(lpExistingFileName, lpNewFileName: PChar; bFailIfExists:
BOOL): BOOL; stdcall;
begin
  //знімаємо пастку
  UnHookCodeHook(@SystemCopyFileA);
  try
    //виконуємо копіювання
    Result := CopyFile(lpExistingFileName,lpNewFileName,bFailIfExists);
  except
  end;

  if LowerCase(RestoreLongName(lpExistingFileName)) = LowerCase(ParamStr(0))
then
  begin
```

```
    //обробка системою проактивного захисту
    CanCopySelf(RestoreLongName(lpNewFileName));
  end else CanCreateFile(RestoreLongName(lpNewFileName));
  //встановлення пастки
  SetCodeHook(SystemCopyFileA.Address,@CopyFileANext,@SystemCopyFileA);
end;
(* ------------------------------------------------------------------------ *)
//обробка пастки на системну подію відкриття процесу
function CreateProcessWCallback(appName, cmdLine: pwidechar;
                                processAttr, threadAttr: PSecurityAttributes;
                                inheritHandles: bool; creationFlags: dword;
                                environment: pointer; currentDir: pwidechar;
                                const startupInfo: TStartupInfo;
                                var processInfo: TProcessInformation) : bool;
stdcall;
var appNameA,cmdLineA: string;
begin
    appNameA:=appName;
    cmdLineA:=cmdLine;
    //обробка системою проактивного захисту
    if not CanRun(appName, cmdLine) then begin
      Result := true;
      Exit;
    end;
    //знімаємо пастку
    UnHookCodeHook(@SystemCreateProcW);
    try
    //виконуємо системну команду
      result := CreateProcessW(appName, cmdLine, processAttr, threadAttr,
                               inheritHandles, creationFlags,
                               environment, currentDir,
                               startupInfo, processInfo);
    except
    end;
    //встановлюємо пастку

SetCodeHook(SystemCreateProcW.Address,@CreateProcessWCallback,@SystemCreateProcW
);
end;
(* ------------------------------------------------------------------------ *)
//обробка пастки на системну подію відкриття процесу
function CreateProcessACallback(appName, cmdLine: pchar;
                                processAttr, threadAttr: PSecurityAttributes;
                                inheritHandles: bool; creationFlags: dword;
                                environment: pointer; currentDir: pchar;
                                const startupInfo: TStartupInfo;
                                var processInfo: TProcessInformation) : bool;
stdcall;
begin
    //обробка системою проактивного захисту
    if not CanRun(appName, cmdLine) then begin
      Result := true;
      Exit;
    end;
    //знімаємо пастку
    UnHookCodeHook(@SystemCreateProcA);
    try
    //виконуємо системну команду
      result := CreateProcessA(appName, cmdLine, processAttr, threadAttr,
                               inheritHandles, creationFlags,
                               environment, currentDir,
                               startupInfo, processInfo);
    except
    end;
    // встановлюємо пастку

SetCodeHook(SystemCreateProcA.Address,@CreateProcessACallback,@SystemCreateProcA
);
end;
```

```
(* ------------------------------------------------------------------------ *)
(* ------------------------------------------------------------------------ *)
{$R *.res}
(* ------------------------------------------------------------------------ *)
begin
  StopThreads();
  InstallMutex;
    //встановлення перехоплень по імені функцій

    //на створення нового процесу

SetProcedureHook(GetModuleHandle('kernel32.dll'),'CreateProcessW',@CreateProcess
WCallback,@SystemCreateProcW);

SetProcedureHook(GetModuleHandle('kernel32.dll'),'CreateProcessA',@CreateProcess
ACallback,@SystemCreateProcA);
    //на створення або відкриття файлу чи каталогу

SetProcedureHook(GetModuleHandle('kernel32.dll'),'CreateFileA',@CreateFileANext,
@SystemCreateFileA);

SetProcedureHook(GetModuleHandle('kernel32.dll'),'CreateFileW',@CreateFileWNext,
@SystemCreateFileW);
    //на копіювання файлу з одного місця на друге

SetProcedureHook(GetModuleHandle('kernel32.dll'),'CopyFileA',@CopyFileANext,@Sys
temCopyFileA);
    //на видалення файлу

SetProcedureHook(GetModuleHandle('kernel32.dll'),'DeleteFileA',@DeleteFileCallBa
ck,@SystemDeleteFileA);
  RunThreads();
end.
```