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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Кузьмін К.К.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Кузьміну Кирилу Костянтиновичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 136-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів*

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 кібербезпеки в промислову експлуатацію.

6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 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	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Кузьмін К.К.
(прізвище та ініціали)

АНОТАЦІЯ

Кузьмін К.К. Програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

Метою розробки є програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

Результат роботи – програмна реалізація системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

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

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

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: кібербезпека, стелс-вірус

ABSTRACT

Kuzmin K.K. Windows 10/11 cyber security system software against the destabilizing effects of stealth viruses. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of the Windows 10/11 OS against the destabilizing effect of stealth viruses.

The purpose of the development is the Windows 10/11 cyber security system software against the destabilizing effects of stealth viruses.

The result of the work is the software implementation of the Windows 10/11 OS cyber security system against the destabilizing effect of stealth viruses.

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 software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Delphi 10.4 environment.

Keywords: cyber security, stealth virus

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DHCP – протокол, що використовується для динамічного розподілу IP-адрес;

DNS – розподілена служба Інтернет, використовувана для зіставлення логічних (доменних) імен і IP-адрес. DNS використовується для забезпечення можливості роботи зі зрозумілими й іменами, що легко запам'ятовуються, замість IP-адрес у числовому форматі;

ICMP – протокол;

IDS – система виявлення атак;

IP – адресний протокол;

LAN – локальна мережа;

NAT – трансляція IP-адрес із одного адресного простору в IP-адреси іншого адресного простору;

Proxy – програма-посередник, що транслює запити різних протоколів із локальної мережі в зовнішню мережу;

TCP – протокол обміну даними на транспортному рівні;

UDP – протокол;

URL – уніфікований покажчик інформаційного ресурсу (стандартизований рядок символів, що вказує місцезнаходження документа в мережі Інтернет);

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер.

ВСТУП

Актуальність теми. За статистикою 70% персональних комп'ютерів можуть бути заражені внаслідок відсутності або неправильного налаштування антивірусного програмного забезпечення.

Комп’ютерний вірус – це свого роду програма, що містить шкідливий для операційної системи код. У випадку зараження вірусом, комп’ютерові може бути завдано шкоди (операційній системі і файлам) чи (та) іншим комп’ютерам в мережі. Вірус може пошкодити важливі документи, фотографії, бухгалтерські бази даних та інші типи файлів.

Віруси можуть:

– провокувати перезавантаження або вимкнення комп’ютера;

– гальмувати роботу комп’ютера;

– знищувати та пошкоджувати дані (файли);

– розсилати самого себе через Інтернет та в локальній мережі (витрата трафіку, проблеми з провайдером);

– операційна система може припинити працювати;

– велика кількість інших різноманітних проблем.

Одним з найбільш розповсюджених видів вірусів є стелс-віруси.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

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

– Огляд існуючих систем ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

– Дослідження системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

– Програмна реалізація системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів, 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

Система призначена для захисту від стелс-вірусів. Стелс-вірус – вірус, повністю або частково приховує своя присутність у системі, шляхом перехоплення звертань до операційної системи, що здійснюють читання, запис, читання додаткової інформації про заражені об'єкти (завантажувальних секторах, елементах файлової системи, пам'яті й т.д.)

Назва даного типу вірусів походить від назви малопомітних винищувачів «Стелс». Воно підкреслює, що такі віруси можуть залишатися непоміченими для антивірусних програм.

Види Stealth-вірусів:

– Завантажувальний вірус перехоплює функцію ОС, призначену для посекторного доступу до дисків, з метою «показати» користувачеві або програмі-антивірусу оригінальний уміст сектора до зараження.

– Файловий вірус перехоплює функції читання/установки позиції у файлі, читання/запису у файл, читання каталогу й т.д., щоб сховати збільшення розміру заражених програм; перехоплює функції читання/запису/відображення файлу на згадку, щоб сховати факт зміни файлу.

– Макровіруси. Реалізувати стелс-алгоритм у макровірусах досить просто, потрібно заборонити виклик меню File/Templase або Tools/Macro, досягти цього можна видаленням пунктів меню зі списку або їхньою підміною на макроси File Templase і Tools Macro. Також стелс-вірусами можна назвати макровіруси, які свій основний код зберігають не в самому макросі, а в інших областях документа.

До відомих Stealth-вірусам ставляться такі віруси, як Virus.DOS.Stealth.551, Exploit.Macro.Stealth, Exploit.MSWord.Stealth, Brain, Fish#6.

Одним з перших стелс-вірусів прийнято вважати RCE-04096, що був розроблений в Ізраїлі наприкінці 1989 р. Назва «Frodo» указує на наявність бут-сектора вірусу у своєму коді, хоча він не записує своє тіло в бут-сектор.

Способи боротьби з Stealth-вірусами

Для пошуку stealth-вірусів рекомендується здійснити завантаження системи із зовнішнього диска й провести видалення вірусних програм.

Антивіруси-поліфаги ефективні в боротьбі із уже відомими вірусами, тобто чиї методи поводження вже знайомі розроблювачам і є в базі програми. Якщо вірус невідомий, то він залишиться непоміченим. Головне в боротьбі з вірусами якнайчастіше обновляти версії програми й вірусні бази.

1.2 Область застосування

Областю застосування системи є антивірусний захист ОС Windows. Для видалення вірусів існують антивірусні програми (антивірусний захист) – спеціальні програми для захисту від вірусів. Основне призначення цих програм – не видалення чи лікування комп'ютерних вірусів, а недопущення зараження комп'ютера вірусом. Проте, трапляються випадки, коли і антивірусна програма не здатна впоратись зі своїм завданням, і комп'ютер заражується вірусом. У такому випадку антивірусна програма допоможе в лікуванні вірусів. У випадку, якщо лікування неможливе, програма антивірусного захисту допоможе видалити вірус .

Комп'ютерна думка не стоїть на місці і, нажаль, разом з нею не стоїть на місці і фантазія людей, які займаються такою неблагородною справою, як написання вірусів. Іноді автори антивірусних програм не встигають своєчасно відреагувати на появу нового вірусу. У такому випадку антивірусна програма, нажаль, безсила, і в змозі допомогти лише спеціаліст у галузі комп'ютерної безпеки, який проаналізує дії вірусу та розробить комплекс заходів для боротьби з ним.

У випадку, якщо антивірусній програмі не вдалося видалити або вилікувати вірус, радимо звернутися до спеціалістів, які вирішать проблему – проведуть лікування або видалення вірусів на Вашому комп’ютері.

Для лікування комп'ютерних вірусів і запобігання попаданню їх на комп'ютер необхідно реалізувати такі послуги:

– Перевірка комп'ютера на віруси і програми-шпигуни.

– Видалення всіх знайдених вірусів і шпигунів.

– Встановлення антивірусного програмного забезпечення.

– Встановлення анти-шпигунського програмного забезпечення.

– Налаштування автоматичного оновлення всього програмного забезпечення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів, 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

Розглянемо антивірусні програми, приведемо коротку класифікацію, надамо короткий огляд, які на даний момент існують антивірусні програми, виділимо плюси й мінуси популярних і не дуже антивірусів.

Avast Free Antivirus

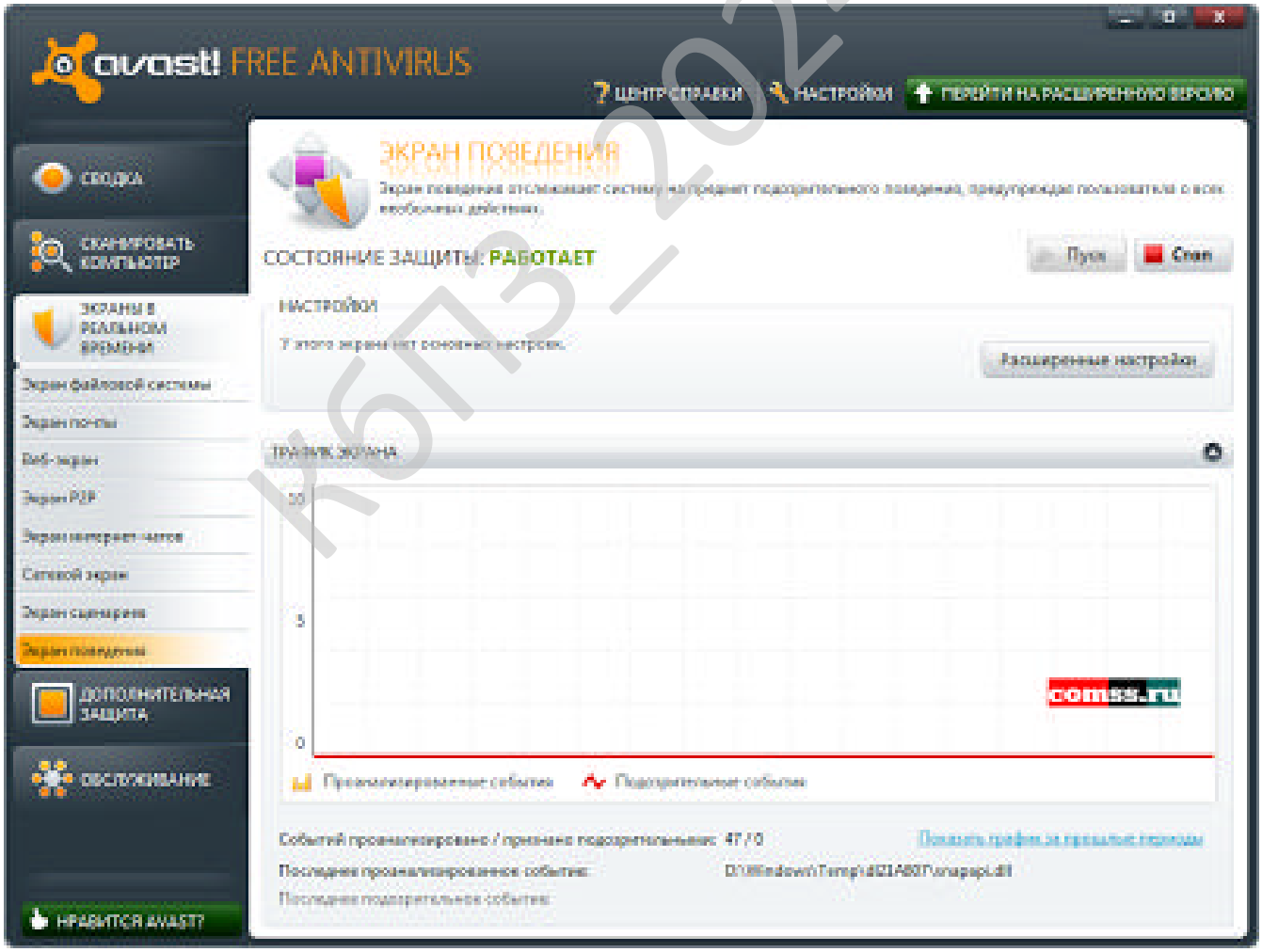

Рисунок 2.1 – Інтерфейс користувача Avast Free Antivirus

Є версія для Android (і не тільки).

Основні функції для того, щоб користувачі почували себе в безпеці:

– Sandbox – віртуальне відособлене середовище, у якій забезпечується повний захист всіх даних, що зберігаються на ПК.

– Антивірус – захист і безпека даних.

– Віддалена допомога – за типом віддаленого помічника в Windows, дуже корисна можливість.

– SafeZone™ – особлива зона, що захищає банківські дані.

– Антиспам – захист від небажаних листів.

– Брандмауер – блокування атак хакерів і зловмисників.

Чим чудовий Avast? Приємний інтерфейс, захист від вірусів, а саме головне – безкоштовний.

Dr.Web

Dr.Web став особливо популярний у першу чергу завдяки тому, що є легкі версії продукту, завдяки яким можна сканувати знімні носії.

Основні функції Dr.Web:

– Сканер.

– SpIDer Guard.

– SpIDer Mail.

– Карантин.

– Центр керування.

– Брандмауер.

– Криптограф.

– Менеджер ліцензій.

– Мій Dr.Web (особисте зберігання даних).

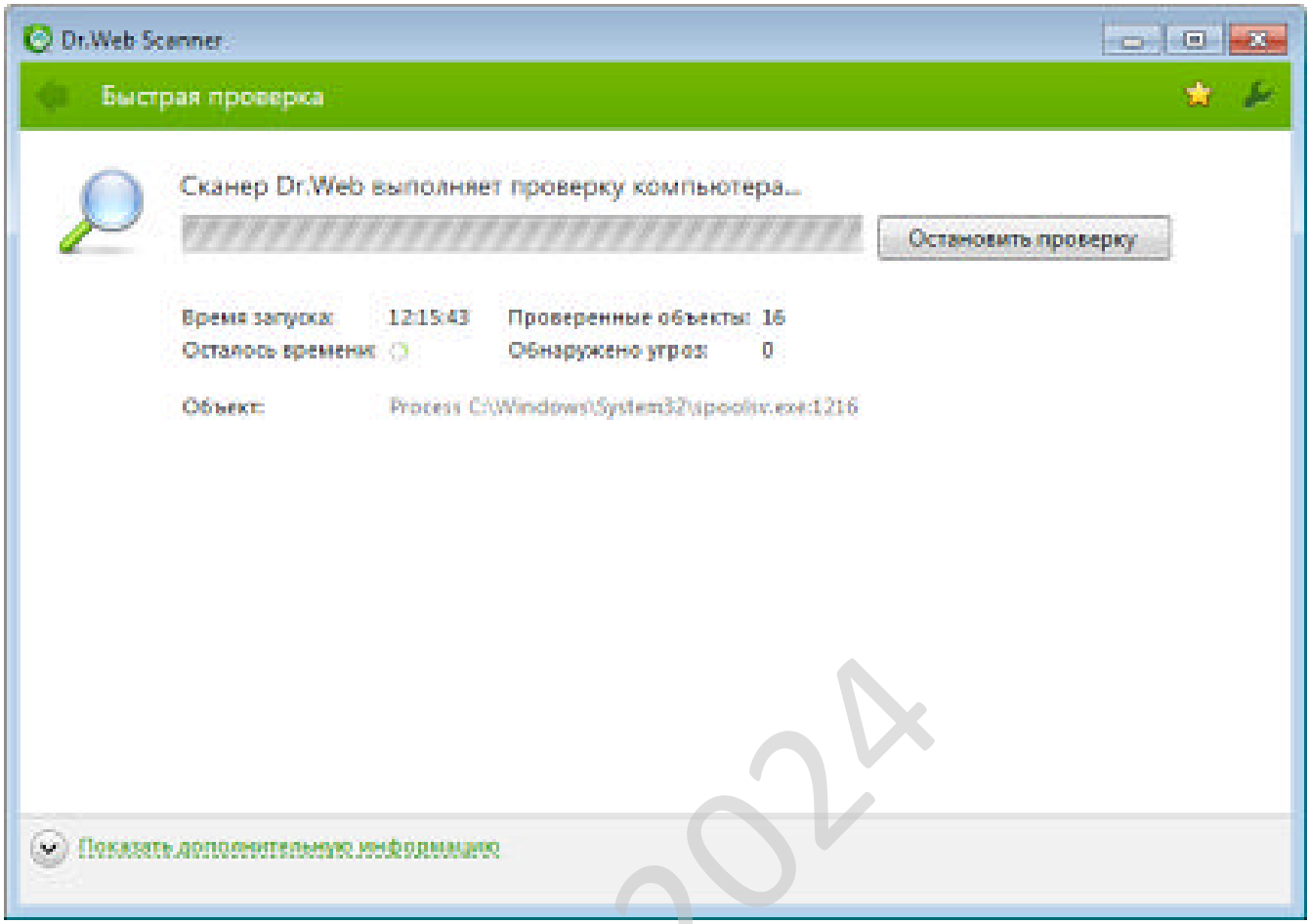

Рисунок 2.2 – Інтерфейс користувача Dr.Web

Іноді бувають помилкові спрацьовування, але це скоріше виключення із правил. Є онлайн-сканування на віруси.

Avira Free Antivirus

Основні функції Avira Free Antivirus:

– Real-Time Protection – містить ПК у чистоті, блокуючи віруси, хробаки, трояни, руткіти й багато чого іншого.

– AntiAd/Spyware – блокує всі спроби шпигувати за вами й рятує від настирливого рекламного ПЗ.

– Browser Tracking Blocker – абсолютний захист від компаній, що намагаються відслідковувати ваші дії в мережі. Включено в Avira Toolbar.

– Website Safety Advisor – оцінює безпеку веб-сайтів у результатах пошуку. Включено в Avira Toolbar.

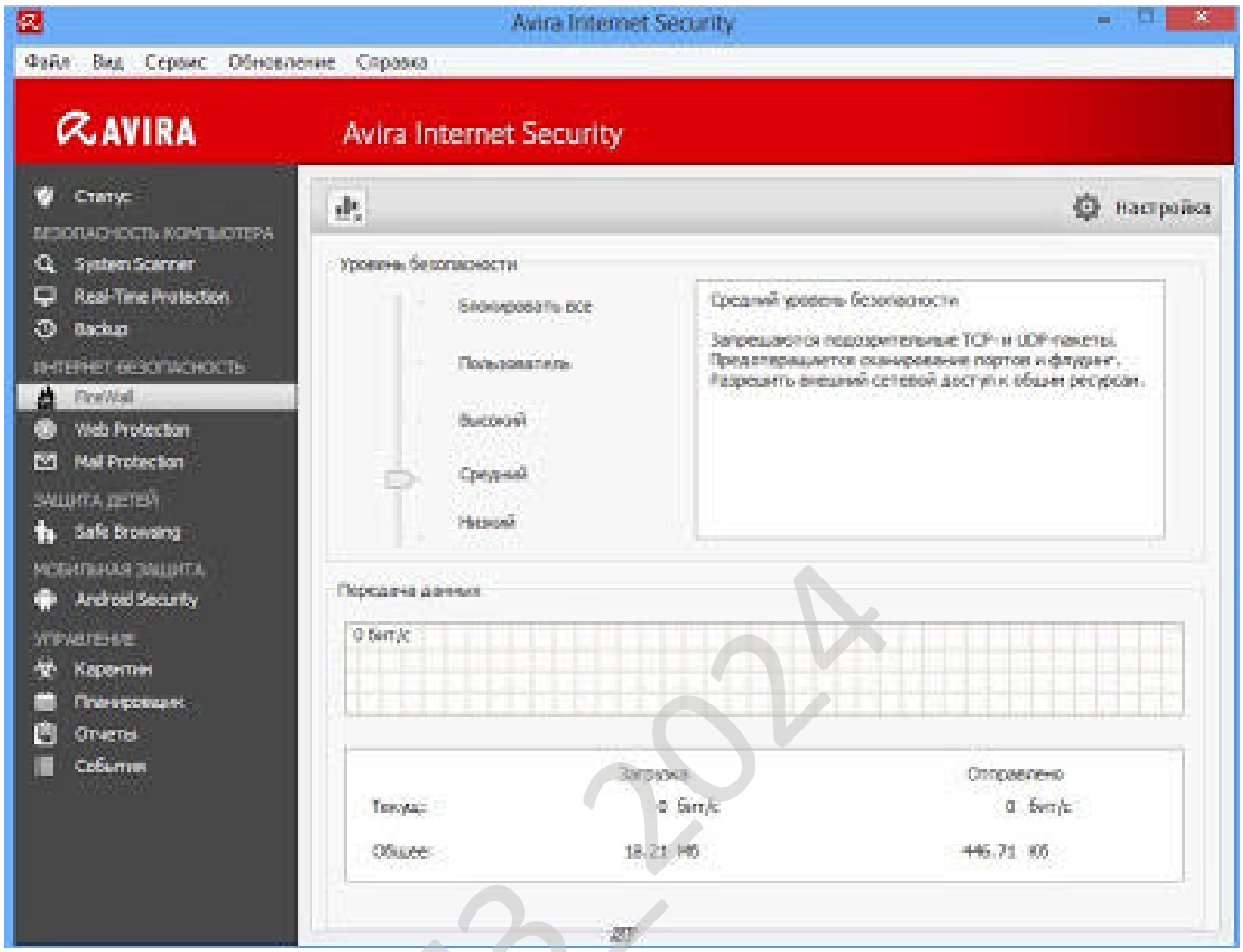

Рисунок 2.3 – Інтерфейс користувача Avira Free Antivirus

Антивірус гарний, але дуже велика кількість помилкових спрацьовувань.

ESET NOD32

ESET NOD32 Антивірус 6 не надав точні функції на сайті (крім Антивірус, Антишпигун, USB-контроль), тому перелічимо основні можливості:

– Блокування шпигунського програмного забезпечення й фішингових сайтів.

– Зупинка дії шпигунських і троянських програм.

– Зниження рівня небезпеки в соціальних мережах.

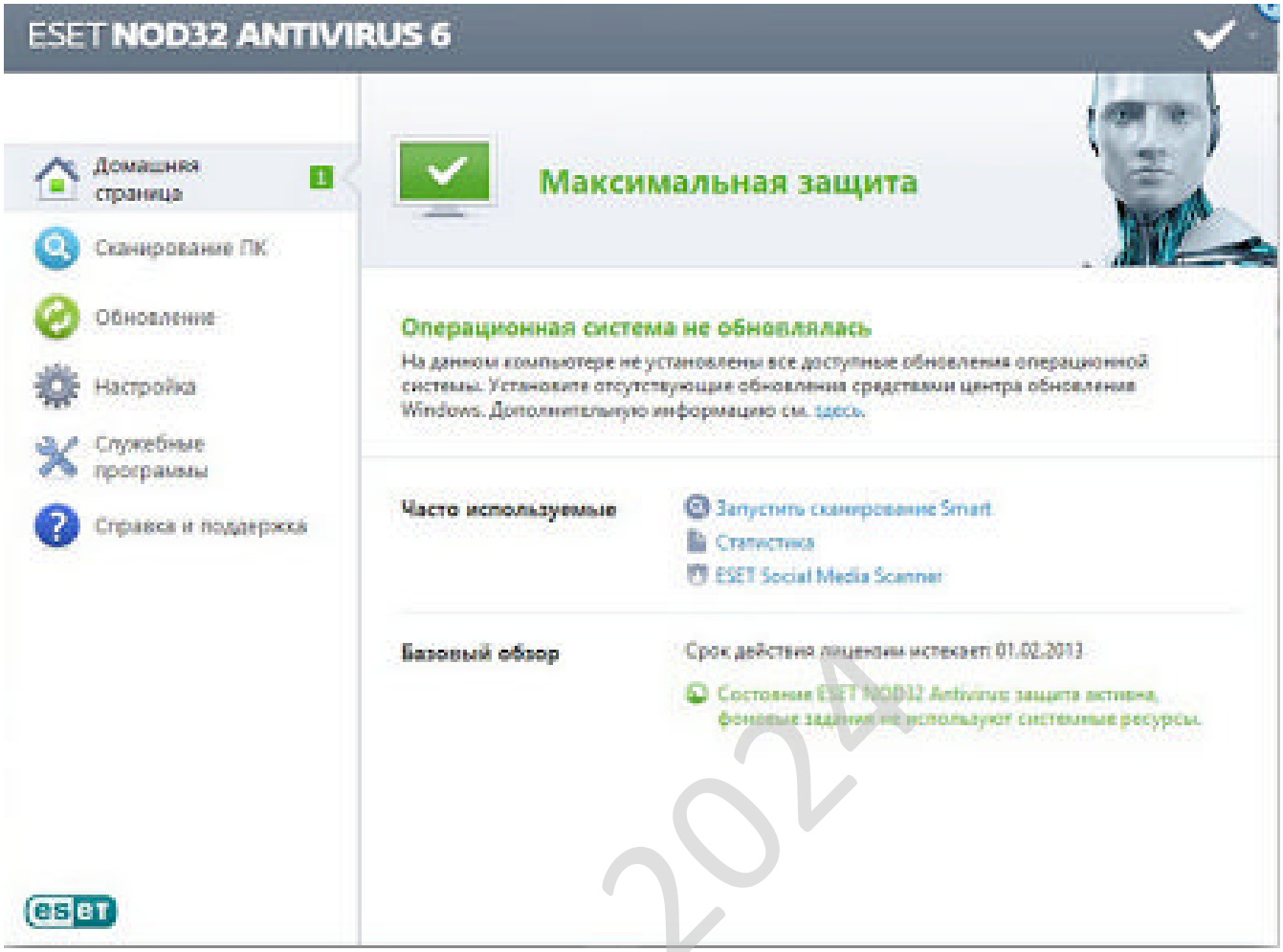

Рисунок 2.4 – Інтерфейс користувача ESET NOD32

Антивірусна програма має дуже гарний потенціал, але моє сугубо особиста думка, якщо нечасто обновляти, то величезна кількість помилкових спрацьовувань. Пам'ятаю як при скануванні в комп'ютерному класі екран просто покривався червоним кольором, причому не все з виявленого було вірусами.

AVG AntiVirus

Раніше продукт був повністю безкоштовним, зараз надають безкоштовну пробну версію.

Основні функції й складові AVG AntiVirus:

– AntiVirus (захист від вірусів).

– AntiMalware (захист від malware).

– AVG Anti-Rootkit (корисний модуль від руткіт-погроз).

– AVG Email Scanner (захист пошти).

– AVG Protective Cloud Technology (хмарні технології).

– AVG LinkScanner® Surf-Shield.

– AVG Community Protection Network (захист мереж),

– AVG Social Networking Protection (додатковий захист).

Рисунок 2.5 – Інтерфейс користувача AVG AntiVirus

AVG антивірус я якось прийняв називати більше не антивірусом, а утилітою, який можна на додаток до встановленого антивірусу просканувати систему. Зараз AVG вийшов на новий рівень, тепер у них не просто утиліта, а повноцінний продукт.

Microsoft Security Essentials

Як виявилося, Захисник Windows (Windows Defender) і Microsoft Security Essentials – це різні речі (Хоча по зовнішньому вигляді ці утиліти схожі). Я зайшов на офіційний сайт Microsoft і скачав Microsoft Security Essentials на свій комп'ютер з Windows 10/11. Комусь може бути корисним, якщо на комп'ютері

вже встановлений Захисник Windows або Microsoft Security Essentials, то краще не пробувати експериментувати в такий спосіб.

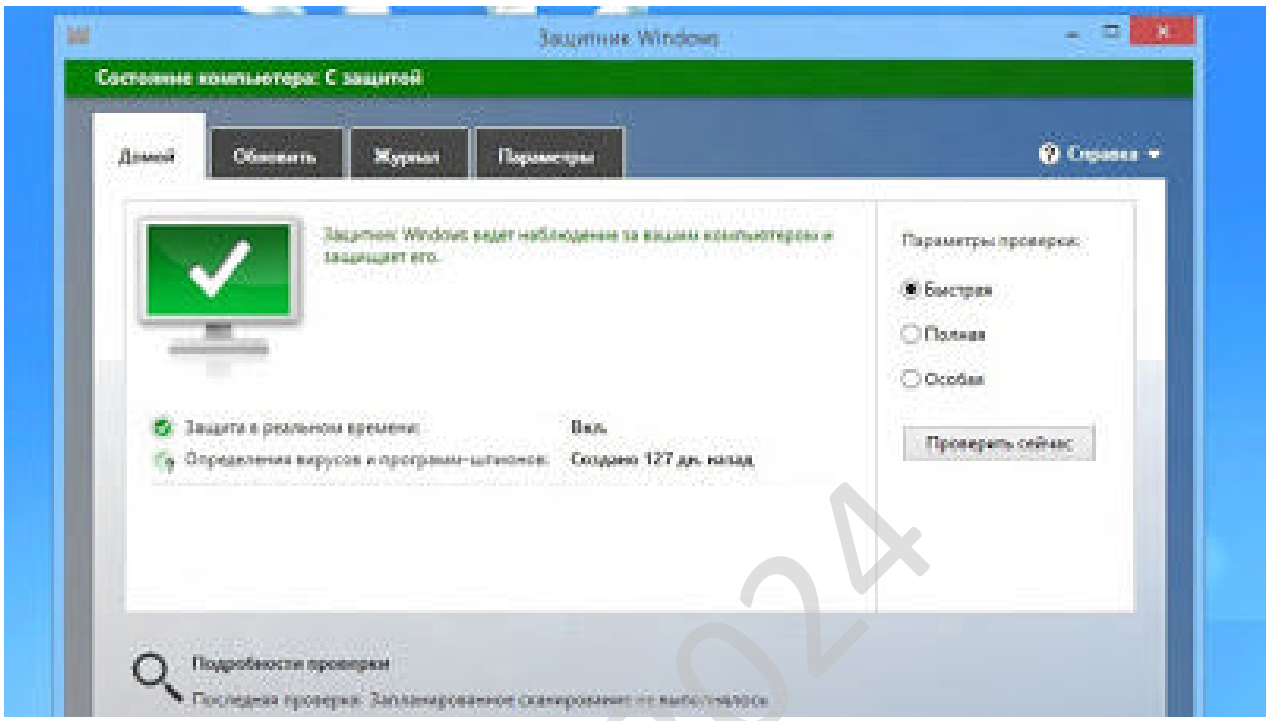

Рисунок 2.6 – Інтерфейс користувача Microsoft Security Essentials

Всі пояснення дає офіційний сайт:

«Захисник Windows для Windows 10/11 і Windows RT забезпечує такий же рівень захисту від шкідливих програм, як і програма Microsoft Security Essentials. Ви не можете використовувати Microsoft Security Essentials з Windows 10/11, але це й не потрібно – Захисник Windows уже встановлений і готовий до роботи».

K7 ULTIMATE SECURITY

K7 Computing – єдина компанія по випуску антивірусних програм в Індії (відповідно до офіційного сайту). Я був здивований, коли знайшов цю програму, стало цікаво, але по відчуттях не можу сказати, так як не пробував неї використовувати.

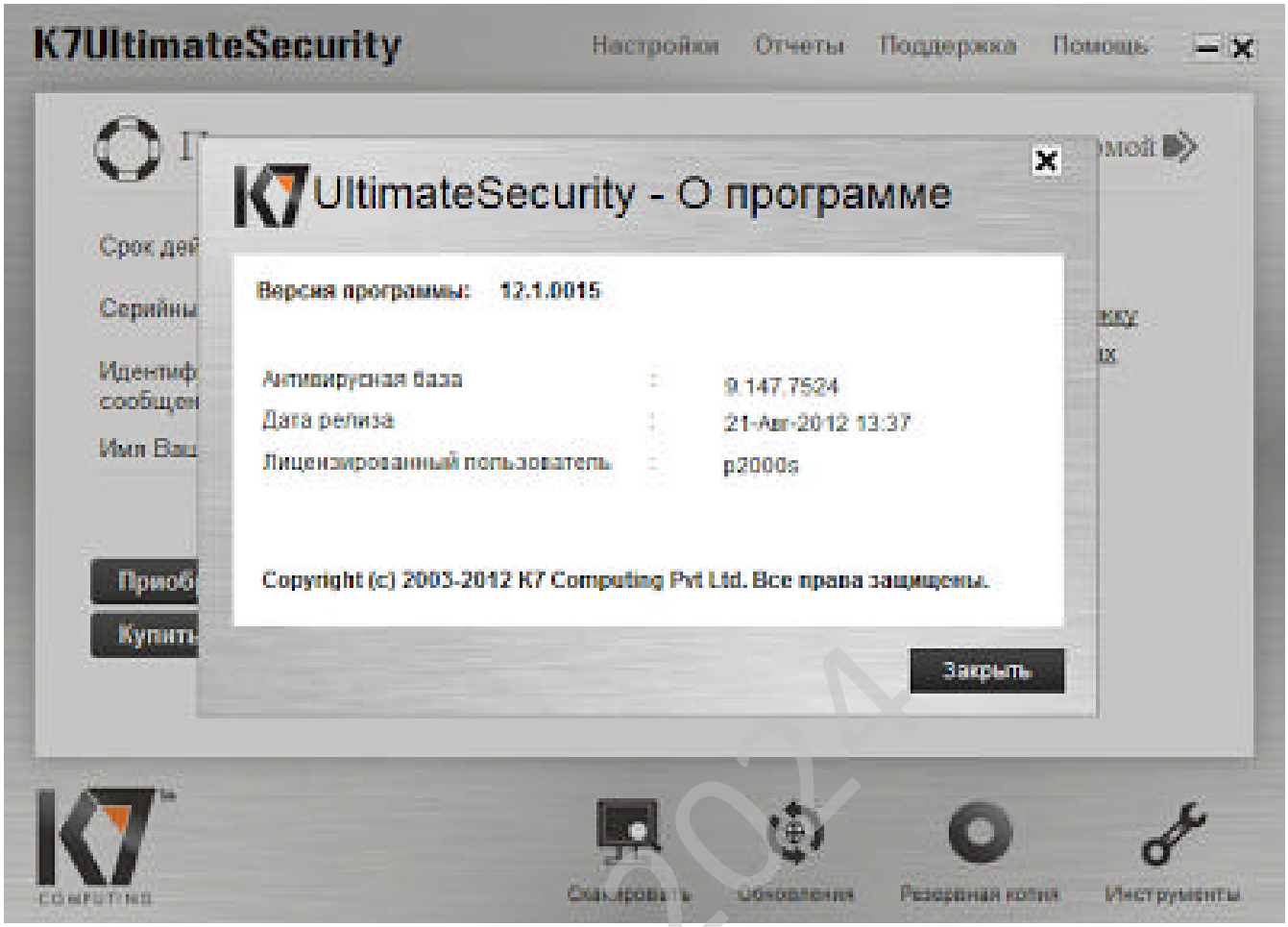

Рисунок 2.7 – Інтерфейс користувача K7 ULTIMATE SECURITY

Основні функції:

– Антивірусний захист (основний захист).

– Резервне копіювання й відновлення.

– Анти шкідливі програми.

– Антируткіт.

– Блокування погроз "нульового дня" (щось новеньке).

– Запобігання вторгнень на ПК (контроль над небезпекою).

– Контроль пристроїв.

– Захист по мережі: Firewall і IDS.

– Антиспам безпека.

– Веб-захист.

– Батьківський контроль.

– І інші утиліти.

Comodo

Варто виділити окремо незвичайний інтерфейс, жаль, що все на англійському.

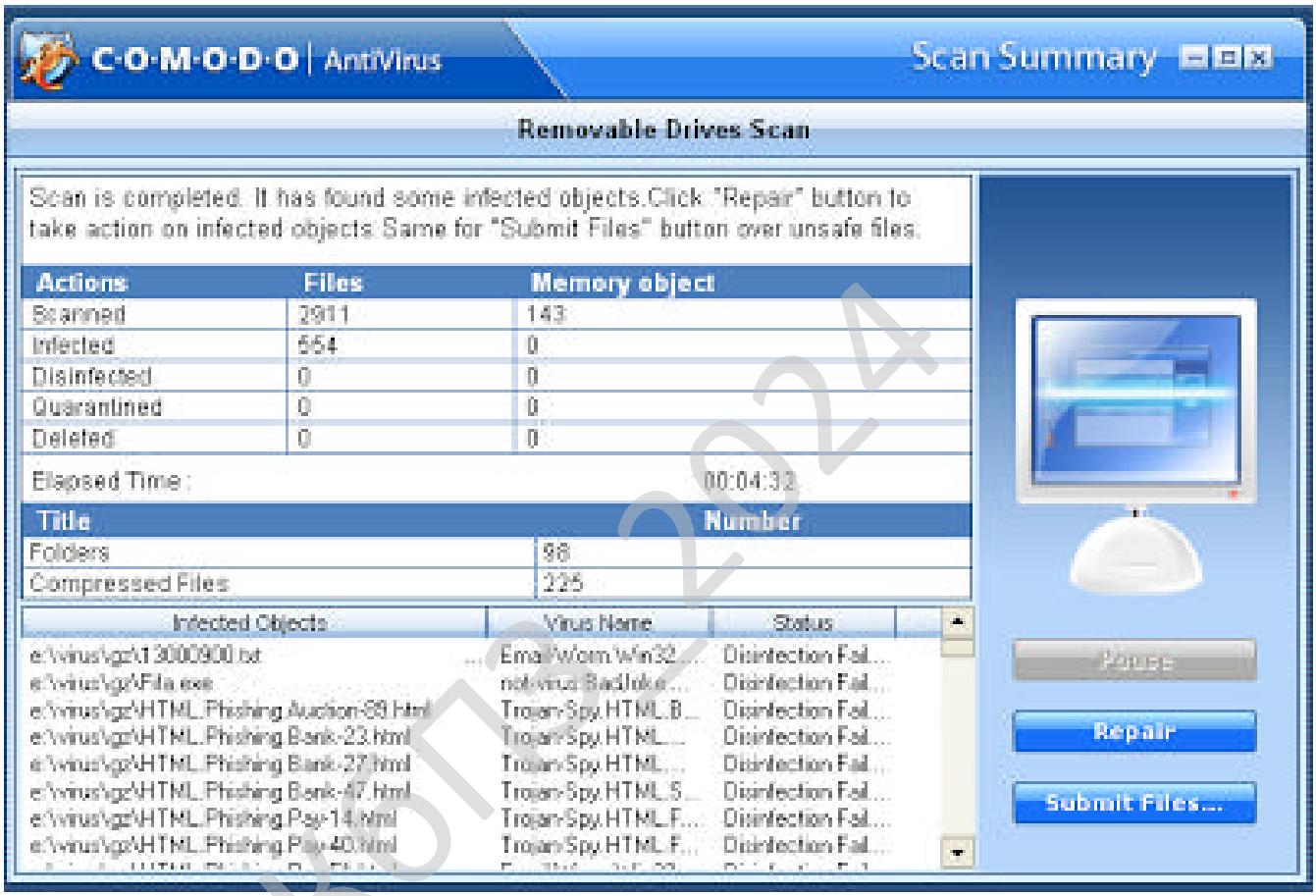

Рисунок 2.8 – Інтерфейс користувача Free Antivirus from Comodo

Основні функції Free Antivirus from Comodo (є версії для різних платформ MAC/Linix/Android):

– Comodo Antivirus Cleans Malware.

– Defense+ Host Intrusion Protection.

– Auto Sandbox Technology™.

– Наявність безкоштовної версії дуже радує.

Dr. ASM

Dr. ASM – оригінальна утиліта, що дозволяє знаходити нові й невідомі раніше антивіруси.

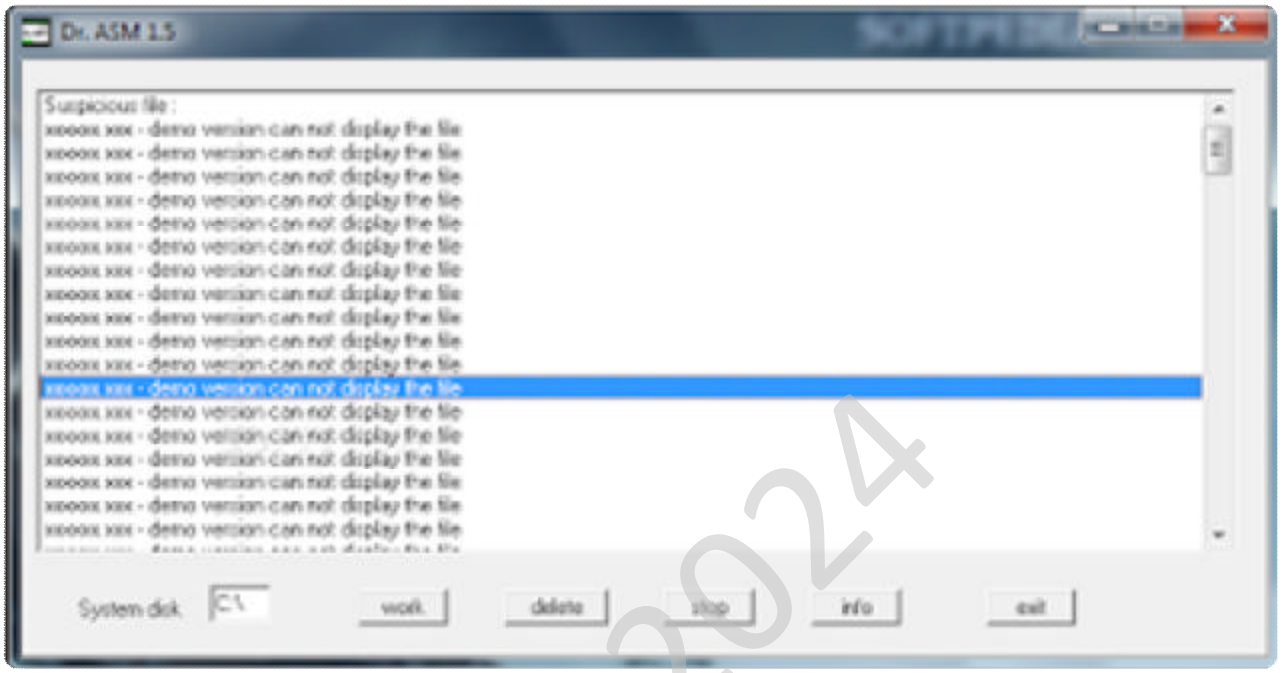

Рисунок 2.9 – Інтерфейс користувача Dr. ASM

Сайту, на жаль, не знайшов. Варто ще додати, що програма важить усього 200 Кб, є демо-версія.

Outpost Antivirus Pro

Основні функції:

– Захист від макро-файлових вірусів.

– Захист пошти.

– Захист від шпигунського пз, захист від keyloggers.

– Захист від dialers і від adware.

– Захист від botware (віддалений захват пк).

– Захист від жартів і хакерских програм.

– Захист від схованих погроз (руткітов).

– Захист від шахраїв.

– Захист банківських карт і від витоку паролів.

Рисунок 2.10 – Інтерфейс користувача Outpost Antivirus Pro

Panda antivirus

Є кілька варіацій для домашньої версії: Panda Global Protection, Panda Internet Security, Panda Antivirus Pro, Cloud Antivirus – PRO Edition, Panda Antivirus for Mac.

Рисунок 2.11 – Інтерфейс користувача Panda antivirus

Також є онлайн-сканер (як у Доктора Веба).

Жаль, що немає безкоштовної версії. З дивного – дизайн.

Bitdefender

Є кілька варіацій, також немає безкоштовної версії, але є 30-денний період. Є навіть окремо спеціальна версія для Windows 10/11, що для мене було трохи дивно. Звичайно антивіруси не пишуться спеціально під Windows 10/11, а їх підбудовують, тобто додатково до основної програми дописують усе, що потрібно, а не випускають зовсім новий продукт.

Рисунок 2.12 – Інтерфейс користувача Bitdefender

Inoculatel

Inoculatel Workgroup Edition – перший антивірус для корпоративних мереж. Радує безкоштовність, щоправда, скачати із сайту не вдалося.

Norton Antivirus

Типове антивірусне програмне забезпечення, 30 днів пробний демо-період, особливих переваг не виявив.

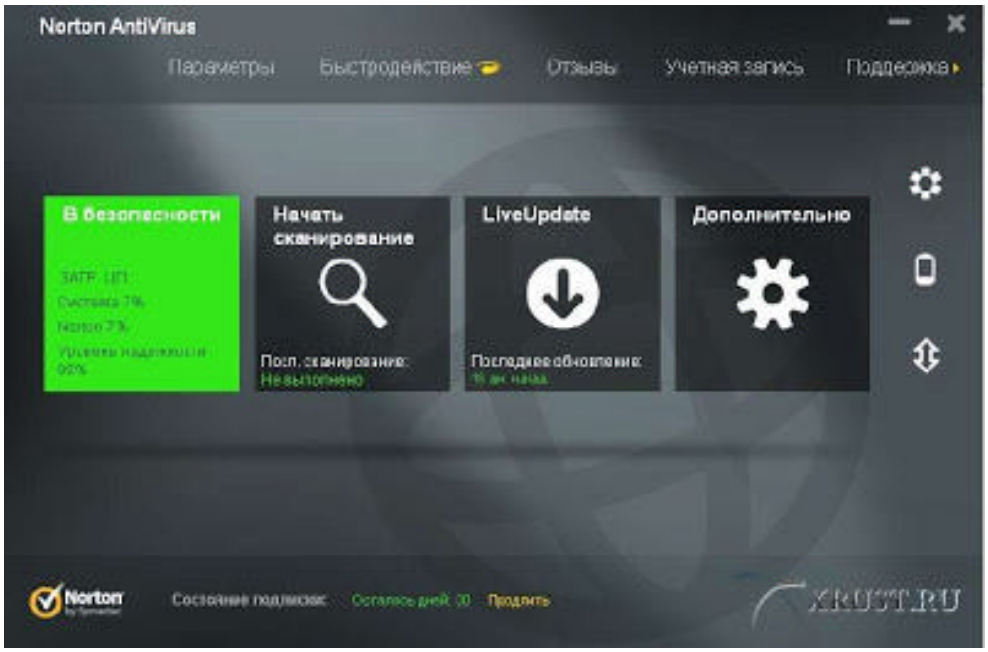

Рисунок 2.13 – Інтерфейс користувача Norton Antivirus

ADinf32

Рисунок 2.14 – Інтерфейс користувача ADinf32

Антивірус ревізор диска – так себе називає даний антивірус. Програма простенькими, створеними розроблювачами-ентузіастами (це видно по скромній рекламі на сайті).

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за

допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

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

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

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL,

включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній

формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

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

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розвиток сучасних комп'ютерних технологій і засобів зв'язку дає можливість зловмисникам використовувати різні джерела поширення погроз. Розглянемо їх докладніше:

Інтернет. Глобальна мережа Інтернет унікальна тим, що не є чиєюсь власністю й не має територіальних границь. Особливості глобальної мережі надають зловмисникам можливість здійснення злочинів в Інтернеті, при цьому ускладнюючи їхнє виявлення й покарання. Зловмисники розміщають шкідливі програми на веб-ресурсах, «маскують» їх під корисне й безкоштовне ПЗ. Крім того, скрипти, запускаються автоматично при відкритті веб-сторінки, можуть виконувати шкідливі дії на ПК. Використовуючи мережні технології, зловмисники реалізують атаки на віддалені приватні комп'ютери й сервери компаній. Результатом таких атак може бути виведення ресурсу з ладу, одержання повного доступу до ресурсу, а, отже, до інформації, що зберігається на ньому, використання ресурсу як частини зомбі-мережі. З появою кредитних карт, електронних грошей і можливістю їхнього використання через Інтернет, інтернет-шахрайство стало одним з найпоширеніших злочинів.

Інтранет. Інтранет – це внутрішня мережа, спеціально розроблена для керування інформацією усередині компанії або приватної домашньої мережі. Якщо який-небудь із комп'ютерів мережі заражений, інші комп'ютери піддаються величезному ризику зараження. Щоб уникнути виникнення таких ситуацій необхідно захищати не тільки периметр мережі, але й кожний окремий ПК.

Електронна пошта. Наявність поштових додатків практично на кожному комп'ютері, а також те, що шкідливі програми повністю використовують вміст електронних адресних книг для виявлення нових жертв, забезпечує сприятливі

умови для поширення шкідливих програм. Користувач зараженого комп'ютера, сам того не підозрюючи, розсилає заражені листи адресатам, які у свою чергу відправляють нові заражені листи й т.д.

З'ємні носії інформації. При запуску файлу, що містить шкідливий код, зі з'ємного носія ви можете ушкодити дані, що зберігаються на вашому комп'ютері, а також поширити вірус на інші диски комп'ютера або комп'ютери мережі.

Хакерські атаки це дії зловмисників або шкідливих програм, спрямовані на захват інформаційних даних віддаленого комп'ютера, виведення системи з ладу або одержання повного контролю над ресурсами комп'ютера.

Деякі види Інтернет-шахрайства

Фішинг – вид Інтернет-шахрайства, що полягає в розсиланні електронних повідомлень із метою крадіжки конфіденційної інформації, як правило, фінансового характеру.

Нав'язлива реклама – це спливаючі вікна й рекламні банери, що відкриваються при роботі з Веб-сайтами.

Спам – це анонімне масове розсилання небажаних поштових повідомлень. Спам істотно збільшує навантаження на поштові сервери й підвищує ризик втрати інформації, важливої для користувача.

Виявлення й блокування даних видів погроз антивірусом здійснюється за допомогою двох методів:

– реактивний метод, заснований на пошуку шкідливих об'єктів за допомогою постійно обновлюваної бази сигнатур погроз. Для реалізації даного методу необхідно хоча б одне зараження, щоб додати сигнатуру погрози в базу й поширити відновлення баз.

– проактивний метод, на відміну від реактивного захисту, будується не на аналізі коду об'єкта, а на аналізі його поводження в системі. Цей метод націлений на виявлення нових погроз, інформації про які ще немає в базах.

Ознаки зараження. Є ряд ознак, що свідчать про зараження комп'ютера. Якщо з комп'ютером відбуваються «дивні» речі, а саме:

– на екран виводяться непередбачені повідомлення, зображення або відтворюються непередбачені звукові сигнали;

– зненацька відкривається й закривається лоток CD/DVD-ROM;

– без вашої участі, на вашому комп'ютері запускаються програми;

– на екран виводяться попередження про спробу якої-небудь із програм вашого комп'ютера вийти в Інтернет, можна припустити, що ваш комп'ютер уражений вірусом.

Є також непрямі ознаки зараження комп'ютера:

– часті зависання й збої в роботі комп'ютера;

– повільна робота комп'ютера при запуску програм;

– неможливість завантаження операційної системи;

– зникнення файлів і каталогів або перекручування їхнього вмісту;

– часте звертання до жорсткого диска;

– веб-браузер «зависає» або поводиться несподіваним образом.

В 90% випадків наявність непрямих симптомів викликано збоєм в апаратному або програмному забезпеченні.

Профілактика зараження. Ніякі самі надійні й розумні міри не зможуть забезпечити стовідсотковий захист від комп'ютерних вірусів і троянських програм. Одним з основних методів боротьби з вірусами є своєчасна профілактика. Комп'ютерна профілактика складається з невеликої кількості правил, дотримання яких значно знижує ймовірність зараження вірусом і втрати яких-небудь даних:

– захист комп'ютера за допомогою антивірусних програм і програм безпечної роботи в Інтернеті.

– обережність під час запису нових даних на комп'ютер;

– недовіра до вірусних містифікацій, програм-жартів, листів про погрози зараження;

– використання сервісу Windows Update і регулярне встановлення відновлення операційної системи Microsoft Windows;

– придбання дистрибутивної копій програмного забезпечення в офіційних продавців;

– обмеження кола людей, допущених до роботи на вашому комп'ютері;

– зменшення ризику неприємних наслідків можливого зараження;

– регулярна перевірка списку установлених програм на вашому ПК.

Таким чином, виходячи з усього вищеперерахованого, можна стверджувати, що розробка програмного забезпечення автоматизованої контрольно-аналітичної системи захисту ПК від деструктивних дій вірусних атак є актуальною задачею, як потребує вирішення. Саме рішенню цієї задачи і присвячено дану роботу.

Система захисту від деструктивних дій вірусів повинна складатися з кількох частин:

– методологічна частина, у якій установлюються правила роботи з програмним забезпеченням, для того, щоб на комп’ютер не попали програмне забезпечення з шкідливим кодом;

– програмна частина, яка складається з двох видів програм: програм захисту локального комп’ютера та програм захисту від вторгнення з мережі Інтернет, або іншої мережі (корпоративної, локальної). У першому випадку потрібне застосування класичних антивірусів, а у другому застосування спеціальних програм для захисту від вірусних атак з мережі – файрволів.

Тільки повноцінне застосування та комбінація приведених вище принципів захисту комп’ютера, забезпечить безпеку інформації, яка зберігається на жорсткому диску. Для того, щоб комп’ютерних вірус не пошкодив інформацію на ПК необхідно перевіряти усі носії інформації, з якими працюєте на комп’ютері.

Визначення вимог до сучасних антивірусів. Головна вимога – це об'єднання й помітне поліпшення поточних функціональних можливостей всіх продуктів в одне комплексне рішення захисту. Додаток забезпечує не тільки антивірусний захист, але й захист від невідомих погроз:

– антивірус повинен захищати не тільки від уже відомих шкідливих програм, але й від тих, що ще не відомо. Наявність компонента проактивного захисту – основна перевага додатка.

– за рахунок використання технологій iChecker TM і iSwift TM можна знизити навантаження на центральний процесор і дискові підсистеми й збільшити швидкість перевірки файлів.

– процес пошуку вірусів повинен підбудовуватись під час роботи.

– перевірка критичних областей, зараження яких може привести до серйозних наслідків, повинна бути представлена окремим завданням.

– антивірус повинен перевіряти на віруси поштовий трафік на наступних протоколах: IMAP, SMTP, POP3, NNTP, MAPI, http.

– повинна бути реалізована перевірка трафіка, переданого через захищене з'єднання по протоколу SSL.

– повинна бути реалізована технологія самозахисту додатка, захисту від вилученого несанкціонованого керування сервісом антивірусу.

– повинна бути реалізована можливість створення диска аварійного відновлення системи.

– відновлення повинне виконуватися з найбільш ефективного джерела.

Захист антивірусу повинен будуватися виходячи із джерел погроз, тобто на кожне джерело передбачений окремий компонент додатка, що забезпечує його контроль і необхідні заходи щодо запобігання шкідливого впливу цього джерела на дані користувача. Антивірус повинен включати:

– Компоненти захисту.

– Завдання пошуку вірусів.

– Сервісні функції.

Захист у реальному часі забезпечується наступними компонентами захисту:

Система захисту Windows від стелс-вірусів – компонентів, що контролює файлову систему.

Поштовий антивірус – компонентів перевірки всіх вхідних і вихідних поштових повідомлень. Він аналізує електронні листи на присутність шкідливих програм.

Веб-антивірус – спеціально розроблений для перехоплення й блокування виконання скрипту, розташованого на сайті, якщо він являє загрозу.

Проактивний захист – заснований на контролі й аналізі поводження всіх програм. На підставі виконуваних дій антивірус приймає рішення: є програма потенційно небезпечною чи ні.

Завдання пошуку вірусів. Крім постійного захисту всіх джерел проникнення шкідливих програм надто важливо періодично проводити перевірку на присутність вірусів. Це необхідно робити для того, щоб виключити можливість поширення шкідливих програм, які не були виявлені компонентами захисту. Для пошуку вірусів до складу антивірусу повинні бути включені три завдання:

Критичні області – перевірка на присутність вірусів всіх критичних областей. До них відносяться: системна пам'ять, об'єкти, що виконуються при старті системи, завантажувальні сектори дисків, системні каталоги *Windows*. Ціль завдання – швидке виявлення в системі активних вірусів, без запуску повної перевірки. Об'єкти автозапуска – перевірка на присутність вірусів об'єктів, завантаження яких здійснюється при старті операційної системи, а також оперативної пам'яті й завантажувальних секторів дисків.

Сервісні функції ПЗ. Антивірус повинен включити ряд сервісних функцій. Вони передбачені для підтримки додатка в актуальному стані, розширення можливостей використання додатка, для надання допомоги в роботі.

Відновлення – відповідає за відновлення баз даних і внутрішніх модулів антивірусу, використовуваних у роботі додатка.

Файли даних – у процесі роботи додатка по кожному компоненту захисту, завданню пошуку вірусів або відновленню додатка формується звіт.

Аварійний диск – диск аварійного відновлення системи.

Підтримка – всі зареєстровані користувачі антивірусу повинні сила скористатися службою технічної підтримки.

Рекомендується провести попереднє налаштування додатка, щоб максимально гнучко підійти до захисту.

Після завершення установки й запуску додатка рекомендуємо виконати наступні дії:

– Оцінити поточний статус захисту.

– Обновити додаток, якщо це не було зроблено.

– Перевірити ПК на присутність вірусів.

Комплексне керування захистом. Антивірус надає можливість управляти своєю роботою:

– Включати, відключати або припиняти роботу додатка.

– Визначати типи небезпечних програм.

– Формувати список виключень із захисту.

– Створювати власні завдання пошуку вірусів і відновлення.

– Налаштовувати запуск завдань по зручному для вас розкладу.

– Налаштовувати параметри продуктивності захисту ПК

Типи контрольованих шкідливих програм. Антивірус пропонує захист від різних видів шкідливого програмного забезпечення. Додаток завжди перевіряє й знешкоджує віруси, троянські програми й хакерські утиліти. Категорії шкідливого ПЗ:

– Віруси, хробаки, троянські й хакерські програми.

– Шпигунське, рекламне ПЗ, програми схованого дозвона.

– Потенційно небезпечне ПЗ (riskware).

Наведені групи регулюють повноту використання сигнатур погроз при перевірці об'єктів у режимі реального часу й при пошуку вірусів на ПК.

Стелс-віруси

Стелс-віруси (Stealth virus) – вірусні програми, що вживають спеціальні дії для маскування своєї діяльності з метою приховання своєї присутності в заражених об'єктах.

Так звана Стелс-технологія може містити в собі:

– утруднення виявлення вірусу в оперативній пам'яті;

– утруднення трасування й дизасемблювання вірусу;

– маскування процесу зараження;

– утруднення виявлення вірусу в зараженій програмі й завантажувальному секторі.

3.2 Розробка структурної схеми

Захист файлової системи комп'ютера від стелс-вірусів – запускається при старті ОС, постійно перебуває в оперативній пам'яті ПК і перевіряє всі файли, що, відкриваються, зберігаються й запускаються. За замовчуванням система захисту Windows від стелс-вірусів перевіряє тільки нові або змінені файли, тобто файли, які додалися або змінилися із часу останнього звертання до них. Процес перевірки виконується за наступним алгоритмом:

1. Обіг користувача або деякої програми до кожного файлу перехоплюється компонентом.

2. Система захисту Windows від стелс-вірусів перевіряє наявність інформації про перехоплений файл у базі. На підставі отриманої інформації приймається рішення про необхідність перевірки файлу.

Процес перевірки включає наступні дії:

1. Файл аналізується на присутність вірусів. Розпізнавання шкідливих об'єктів відбувається на основі сигнатур погроз, які використовуються у роботі. Сигнатури містять опис всіх відомих на даний момент шкідливих програм, погроз, мережних атак і способів їхнього знешкодження.

2. У результаті аналізу можливі наступні варіанти поводження додатка:

– Якщо у файлі виявлений шкідливий код, система захисту Windows від стелс-вірусів блокує файл, поміщає його копію в резервне сховище й намагається

вилікувати файл. У результаті успішного лікування файл стає доступним для роботи, якщо ж лікування зробити не вдалося, файл видаляється.

– Якщо у файлі виявлений код, схожий на шкідливий, файл піддається лікуванню й міститься в спеціальне сховище – карантин.

– Якщо у файлі не виявлено шкідливого коду, він відразу ж стає доступним для роботи.

Опис вибору рівня безпеки файлів. Система захисту Windows від стелс-вірусів повинен забезпечити захист файлів, з якими ви працюєте, на одному з наступних рівнів:

– Високий рівень, на якому здійснюється максимально повний контроль за що відкриваються, що зберігаються й запускаються файлами.

– Що рекомендується – параметри даного рівня передбачають перевірку наступних категорій об'єктів:

– програм і об'єктів по вмісту;

– тільки нових і змінених з моменту останньої перевірки об'єктів;

– вкладених OLE-об'єктів.

– Низький – рівень із параметрами, які дозволяють комфортно працювати з додатками, що вимагають значних ресурсів оперативної пам'яті, оскільки набір файлів, що перевіряються, на даному рівні скорочений.

Опис налаштування захисту файлів. Те, яким чином здійснюється захист файлів на ПК, визначається набором параметрів. Їх можна розбити на наступні групи:

– параметри, що визначають типи файлів, що піддаються аналізу на віруси;

– параметри, що формують область, що захищається;

– параметри, що задають дії над небезпечним об'єктом;

– додаткові параметри роботи файлового антивірусу.

Режимом перевірки об'єктів визначаються умови спрацьовування файлового антивірусу. Можливі наступні варіанти:

– Інтелектуальний режим. Даний режим спрямований на підвищення швидкості обробки об'єктів і надання їхньому користувачеві для роботи. При його виборі рішення про перевірку приймається на підставі аналізу операцій, виконуваних з об'єктом.

– Перевіряти об'єкти файловим антивірусом при відкритті й зміні.

– При відкритті – перевіряти об'єкти тільки при спробі відкриття.

– При виконанні – перевіряти тільки в момент спроби запуску.

Тимчасова зупинка файлового антивірусу може знадобитися для того щоб знизити навантаження й забезпечити швидкий доступ користувача до об'єктів, рекомендується настроїти відключення компонента в певний час або при роботі з певними програмами.

Захист електронної пошти. За замовчуванням захист пошти здійснюється за наступним алгоритмом:

1. Кожний лист, прийнятий або відправлений користувачем, перехоплюється поштовим антивірусом.

2. Поштове повідомлення розбирається на складові його частини: заголовок листа, тіло, вкладення.

3. Тіло й вкладення поштового повідомлення (у тому числі вкладені OLE-об'єкти) перевіряються на присутність у ньому небезпечних об'єктів.

4. У результаті перевірки на віруси можливі такі варіанти поводження:

– Якщо тіло або вкладення листа містить шкідливий код, поштовий антивірус блокує лист, поміщає копію зараженого об'єкта в резервне сховище й намагається знешкодити об'єкт. У результаті успішного лікування лист стає доступним для користувача, якщо ж лікування зробити не вдалося, заражений об'єкт із листа знищується;

– Якщо тіло або вкладення листа містить код, схожий на шкідливий, але стовідсоткової гарантії цього немає, частина листа міститься в карантин;

– Якщо в листі не виявлено шкідливого коду, воно відразу ж стає доступним для користувача.

Сьогодні можна виділити два основних типи стелс-вірусів: руткіт (rootkit) і буткіт (bootkit).

Руткіт

Руткіт – програма або набір програм для приховання слідів присутності зловмисника або шкідливої програми в системі.

Термін Rootkit історично прийшов з миру UNIX, і під цим терміном розуміється набір утиліт або спеціальний модуль ядра, які зломщик установлює на зламаній їм комп'ютерній системі відразу після одержання прав суперкористувача. Цей набір, як правило, містить у собі різноманітні утиліти для «замітання слідів» вторгнення в систему, робить непомітними сніффери, сканери, кейлоггери, троянські програми, що заміщають основні утиліти UNIX (у випадку неядерного руткіту). Rootkit дозволяє зломщикові закріпитися в зламаній системі й сховати сліди своєї діяльності шляхом приховання файлів, процесів, а також самої присутності руткіту в системі.

Класифікація руткітів

За рівнем привілеїв

– Рівень користувача (user-mode).

– Рівень ядра (kernel-mode).

За принципом дії

– алгоритми, що змінюють, виконання системних функцій (Modify execution path);

– системні структури, що змінюють, даних (Direct kernel object manipulation).

Основні методи реалізації

В Microsoft Windows

В Windows через Windows File Protection переписування системних файлів утруднено, тому основні способи впровадження в систему – модифікація пам'яті:

– перехоплення системних функцій Windows API (API hooking) на рівні користувача;

– те ж на рівні ядра (перехоплення Native API);

– зміна системних структур даних;

– модифікація MBR і завантаження до ядра операційної системи – буткіти (наприклад, BackDoor.MaosBoot).

Даний вид шкідливих кодів у середовищі Windows відомий з початку 1990-х років за назвою стелс-вірусів.

В UNIX і linux

– реалізовані підміною основних системних утиліт (дуже легко виявляються засобами контролю цілісності, крім того, легко блокуються засобами мандатного керування доступом типу SELinux або AppArmor;

– реалізовані у вигляді модуля ядра й засновані на патчинзі VFS або перехопленні таблиці системних викликів (sys_call_table);

– засновані на модифікації фізичної пам'яті ядра.

Додаткові можливості

Крім безпосередньо себе, руткіт, як правило, може маскувати присутність у системі будь-яких описаних у його конфігурації каталогів і файлів на диску, ключів у реєстрі. Із цієї причини природно з'явилися «навісні» руткітні бібліотеки. Багато які руткіти встановлюють у систему свої драйвери й служби (вони, природно, також є «невидимими»).

Легальні руткіти

Руткіти можуть «підкидати» не тільки зловмисники. Відомий випадок, коли корпорація Sony вбудовувала подобу руткіту у свої лицензійні аудіодиски. Руткітами по суті є більшість програмних засобів захисту від копіювання (і засобів обходу цих захистів – наприклад, емулятори CD– і DVD-приводів). Від «нелегальних» вони відрізняються тільки тим, що ставляться з відома користувача.

Антируткіти

Це утиліти або резидентні модулі, що виявляють у системі присутність руткітів і (у різній мері) видаляючи їх. Існує безліч конкуруючих засобів для

цього – як платних, так і безкоштовних, але всі вони використовують подібні принципи дії:

Пошук розбіжностей

Проти MEP-руткітів. Та сама інформація виходить декількома способами з використанням API і «прямо» і шукаються розбіжності. Зокрема, звичайно скануються таблиці імпорту, таблиця Native API, файлова система.

У системах Windows під rootkit прийнято мати на увазі програму, що впроваджується в систему й перехоплює системні функції (Windows API). Перехоплення й модифікація низькорівневих API-функцій, у першу чергу, дозволяє такій програмі досить якісно маскувати свою присутність у системі. Крім того, як правило, rootkit може маскувати присутність у системі будь-яких описаних у його конфігурації процесів, каталогів і файлів на диску, ключів у реєстрі. Багато які rootkit встановлюють у систему свої драйвери й служби (вони також є "невидимими").

Список шкідливих програм

Backdoor.Win32.Phanta.a,b; Backdoor.Win32.Sinowal.knf,kmy; Backdoor.Win32.Trup.a,b; Rootkit.Boot.Aeon.a; Rootkit.Boot.Backboot.a; Rootkit.Boot.Batan.a; Rootkit.Boot.Bootkor.a; Rootkit.Boot.Cidox.a,b; Rootkit.Boot.Clones.a; Rootkit.Boot.CPD.a,b; Rootkit.Boot.Fisp.a; Rootkit.Boot.Geth.a; Rootkit.Boot.Goodkit.a; Rootkit.Boot.Harbinger.a; Rootkit.Boot.Krogan.a; Rootkit.Boot.Lapka.a; Rootkit.Boot.MyBios.b; Rootkit.Boot.Nimnul.a; Rootkit.Boot.Pihar.a,b,c; Rootkit.Boot.Plite.a; Rootkit.Boot.Prothean.a; Rootkit.Boot.Qvod.a; Rootkit.Boot.Smitnyl.a; Rootkit.Boot.SST.a,b; Rootkit.Boot.SST.b; Rootkit.Boot.Wistler.a; Rootkit.Boot.Xpaj.a; Rootkit.Boot.Yurn.a; Rootkit.Win32.PMax.gen; Rootkit.Win32.Stoned.d; Rootkit.Win32.TDSS; Rootkit.Win32.TDSS.mbr; Rootkit.Win32.ZAccess.aml,c,e,f,g,h,i,j,k; Trojan-Clicker.Win32.Wistler.a,b,c; Trojan-Dropper.Boot.Niwa.a; Trojan-Ransom.Boot.Mbro.d,e; Trojan-Ransom.Boot.Mbro.f; Trojan-Ransom.Boot.Siob.a; Virus.Win32.Cmoser.a; Virus.Win32.Rloader.a;

Virus.Win32.TDSS.a,b,c,d,e; Virus.Win32.Volus.a; Virus.Win32.ZAccess.k; Virus.Win32.Zhaba.a,b,c.

Останнім часом погроза RootKit стає усе більше актуальною, так як розроблювачі вірусів, троянських програм і шпигунського програмного забезпечення починають вбудовувати RootKit-технології у свої шкідливі програми. Одним із класичних прикладів може служити троянська програма Trojan-Spy.Win32.Qukart, що маскує свою присутність у системі за допомогою RootKit-технології (дана програма цікава тим, що її RootKit-механізм прекрасно працює в Windows 10/11).

Для ефективної боротьби з RootKit необхідне розуміння принципів і механізмів його роботи. Умовно всі RootKit-технології можна розділити на дві категорії – працюючі в режимі користувача (user-mode) і в режимі ядра (kernel-mode). Перша категорія RootKit заснована на перехопленні функцій бібліотек користувальницького режиму, друга – на установці в систему драйвера, що здійснює перехоплення функцій рівня ядра. Далі при описі методів перехоплення функцій опис іде стосовно до RootKit, однак потрібно пам'ятати, описані методики універсальні й застосовуються безліччю корисних програм і утиліт.

Методи перехоплення API функцій у режимі користувача (user mode)

Перехоплення функцій дозволяє RootKit модифікувати результати їхньої роботи – наприклад, перехоплення функції пошуку файлу на диску дозволяє виключити з результатів пошуку маскуємі файли, а перехоплення функцій типу ntdll.ZwQuerySystemInformation дозволяє замаскувати запущені процеси й завантажені бібліотеки.

Принцип виклику API функції

Перед розглядом принципів роботи RootKit користувальницького режиму необхідно коротко й спрощено розглянути принцип виклику функцій, розміщених в DLL.

Відомо два базових способи:

1. Раннє зв'язування (статично імпортовані функції).

Цей метод заснований на тому, що компіляторові відомий перелік імпортованих програмою функцій. Опираючись на цю інформацію, компілятор формує так звану таблицю імпорту EXE файлу. Таблиця імпорту – це особлива структура (її місце розташування й розмір описуються в заголовку EXE файлу), що містить список використовуваних програмою бібліотек і список імпортованих з кожної бібліотеки функцій. Для кожної функції в таблиці є поле для зберігання адреси, але на стадії компіляції адреса не відома. У процесі завантаження EXE файлу система аналізує його таблицю імпорту, завантажує всі перераховані в ній DLL і робить занесення в таблицю імпорту реальних адрес функцій цих DLL. У раннього зв'язування є істотний плюс – на момент запуску програми всі необхідні DLL виявляються завантаженими, таблиця імпорту заповнена – і все це робиться системою, без участі програми. Але відсутність у процесі завантаження зазначеної в його таблиці імпорту DLL (або відсутність в DLL необхідної функції) приведе до помилки завантаження програми. Крім того, дуже часто немає необхідності завантажувати всі використовувані програмою DLL у момент запуску програми.

2. Пізніше зв'язування.

Відрізняється від раннього зв'язування тим, що завантаження DLL виробляється динамічно за допомогою функції API LoadLibrary. Ця функція перебуває в kernel32.dll, тому якщо не прибігати до хакерських прийомів, то kernel32.dll прийдеться завантажувати статично. За допомогою LoadLibrary програма може завантажити її бібліотеку, що цікавить, у будь-який момент часу. Відповідно для одержання адреси функції застосовується функція kernel32.dll GetProcAddress. Щоб не викликати GetProcAddress перед кожним викликом функції з DLL програміст може однократно визначити адреси його функцій, що цікавлять, і зберегти їх у масиві або деяких змінних.

Незалежно від методу зв'язування системі необхідно знати, які функції експортує DLL. Для цього в кожної DLL є таблиця експорту – таблиця, у якій перераховані експортовані DLL функції, їхнього номера (ординали) і відносні адреси функцій (RVA).

1. Модифікація машинного коду прикладної програми.

У цьому випадку модифікується машинний код, що відповідає в прикладній програмі за виклик тої або іншої функції API. Це методика складна в реалізації, так як існує безліч мов програмування, версій компіляторів і програміст може реалізувати виклик API функцій різними методиками. Але теоретично подібне можливо за умови, що впровадження буде йти в заздалегідь задану програму відомої версії. У цьому випадку можна проаналізувати її машинний код і розробити перехоплювач.

2. Модифікація таблиці імпорту.

Дана методика описана в книзі Ріхтера і є однією із класичних. Ідея методу проста – RootKit знаходить у пам'яті таблицю імпорту програми й виправляє адреси його функцій, що цікавлять, на адреси своїх перехоплювачів (природно, він попередньо десь у себе запам'ятовує правильні адреси). У момент виклику API функції програма зчитує її адресу з таблиці імпорту й передає по цій адресі керування. Методика універсальна, але в неї є істотний недолік – перехоплюються тільки статично імпортовані функції. Але є й плюс – методика дуже проста в реалізації і є маса прикладів, що демонструють її реалізацію. Пошук таблиці імпорту в пам'яті не представляє особливої складності, оскільки для цього існують спеціалізовані API функції, що дозволяють працювати з образом програми в пам'яті.

3. Перехоплення функцій LoadLibrary і GetProcAddress.

Перехоплення функцій LoadLibrary і GetProcAddress може бути виконаний будь-яким методом, у класичній реалізації застосовується методика 2 – модифікація таблиці імпорту. Ідея методики проста – якщо перехопити GetProcAddress, те при запиті адреси можна видавати програмі не реальні адреси

її функцій, що цікавлять, а адреси своїх перехоплювачів. Як і в методі 2 програма «не відчує» різниці. При виклику GetProcAddress вона одержує адресу й виконує виклик функції. У даного методу є мінус – він не дозволяє перехопити статично імпортовані функції;

4. Метод, що сполучить методику 2 і 3

У даній методиці модифікується таблиця імпорту, причому в обов'язковому порядку перехоплюються функції LoadLibrary і GetProcAddress бібліотеки kernel32.dll. У цьому випадку при виклику статично імпортованих функцій перекручені адреси беруться з таблиці імпорту, при динамічному визначенні адреси викликається перехоплена функція GetProcAddress, що повертає адреси функцій-перехоплювачів. У результаті в програми не залишається шансів довідатися правильну адресу функції.

5. Модифікація програмного коду API функції.

Даний метод складніший в реалізації, ніж підміна адреси. Методика полягає в тому, що RootKit знаходить у пам'яті машинний код його функцій, що цікавлять, API і модифікує його. При такому методі перехоплення функції вже немає потреби в модифікації таблиці імпорту запущених програм і передачі програмам перекручених адрес при виклику GetProcAddress. З погляду виклику функції все залишається «як є» за одним виключенням – тепер уже по «правильному» адресі усередині «правильної» DLL перебуває машинний код RootKit.

Найчастіше втручання в машинний код функцій, що перехоплюються, мінімально. На початку функції розміщається не більше 2-3 машинних команд, що передають керування основний функції-перехоплювачу. Для виконання викликів модифікованих функцій RootKit повинен зберегти вихідний машинний код для кожної модифікованої їм функції (природно, зберігається не весь машинний код функції, а змінені при перехопленні байти). Саме така методика перехоплення реалізована в широко відомому HackerDefender і бібліотеці AFX Rootkit (www.rootkit.com).

6. Модифікація бібліотек DLL на диску

Дана методика полягає в тому, що системна бібліотека модифікується на диску. Методи модифікації аналогічні описаним вище, тільки зміни виробляються не в пам'яті, а на диску. Подібна методика не одержала широкого поширення.

Перехоплення функцій у режимі ядра (kernel mode)

Для розуміння типової для методики перехоплення функцій у режимі ядра варто розглянути принципи взаємодії бібліотек користувальницького режиму і ядра.

Основна взаємодія з ядром виробляється через ntdll.dll, більшість функцій якої є переходниками, що звертається до ядра через переривання INT 2Eh (варто помітити, що прикладній програмі ніщо не заважає прямо викликати INT 2Eh). Подальше звертання до функцій ядра заснована на структурі, іменованої KeServiceDescriptorTable (або скорочено SDT), розташованої в ntoskrnl.exe. SDT – це таблиця, що містить адреси точок входу сервісів ядра NT. Спрощено можна сказати, що для перехоплення функцій необхідно написати драйвер, що зробить модифікацію таблиці SDT. Перед модифікацією драйверу необхідно зберегти адреси функцій, що перехоплюються, і записати в таблицю SDT адреси своїх оброблювачів. Даний метод чимсь нагадує перехоплення переривань в MSDOS або описану вище методику 2.

Цей метод часто називають перехопленням Native API і природно він працює тільки в NT лінійці Windows. Слід зазначити, що перехоплення Native API здійснюють не тільки руткіти – існує маса корисних програм, що перехоплюють функції за допомогою виправлення SDT – як приклад можуть служити популярна утиліта RegMon від SysInternals або програма Process Guard.

Слід зазначити, що описаний метод є найбільш простим, але далеко не єдиним. Існує ще ряд способів, зокрема створення драйвера-фільтра. Драйвер-фільтр може застосовуватися як для рішення завдань моніторингу (класичний приклад – утиліта FileMon від SysInternals), так і для активного втручання в

роботу системи. Зокрема, драйвер-фільтр може застосовуватися для маскування файлів і папок на диску. Принцип роботи такого драйвера заснований на маніпуляціях з пакетами запиту уведення-виводу (IRP).

Методики виявлення RootKit у системі

Розглянемо базові методики пошуку RootKit:

– Порівняння двох «знімків» системи (наприклад, списку файлів на диску). Перший знімок робиться на системі, що перевіряється, другий – після завантаження з CD або підключення досліджуваного HDD до свідомо чистого комп'ютера. Подібна методика гарантовано дозволить виявити будь-який RootKit, що маскує на диску свої файли.

– Порівняння даних, що повертаються API функціями різного рівня й (або) одержуваних низькорівневими методами (наприклад, прямим читанням диска й аналізом файлів реєстру). Дана методика не вимагає перезавантаження досліджуваного ПК.

– Аналіз у пам'яті функцій основних бібліотек на предмет наявності змін їхнього машинного коду. Даний метод найбільш ефективний для боротьби з RootKit у користувальницькому режимі. Подібна методика дозволяє не тільки виявити перехоплення функцій, але й відновити нормальну роботу ушкоджених функцій. Крім того, порівняння «знімків» системи, отриманих до й після відновлення функцій API у багатьох випадках дозволяє виявити процеси, що маскуються, сервіси й драйвери. Дана методика не вимагає перезавантаження.

– Аналіз і відновлення ServiceDescriptorTable. Дана методика дозволяє боротися з поруч перехоплювачів, що працюють у режимі ядра (властиво, з перехоплювачами, заснованими на виправленні SDT). Однак відновлення SDT вплине на роботу всієї системи й може привести до дуже неприємних наслідків (у найпростішому випадку – повне зависання системи з виходом на BSoD, у гіршому – непередбачене порушення нормальної роботи додатків, що перехоплюють NativeAPI для реалізації своїх функцій).

Буткіт

Буткіт (Bootkit) – це шкідлива програма (так звана MBR-руткіт), що здійснює модифікацію завантажувального сектора MBR (Master Boot Record) – першого фізичного сектора на жорсткому диску. (Відомий представник Backdoor.Win32.Sinowal).

Призначення

Використовується вірусами для одержання максимальних привілеїв в операційних системах. Буткіт може дістати права адміністратора(суперкористувача) і виконувати будь-які шкідливі дії. Наприклад, він може завантажити на згадку деяку динамічну бібліотеку DLL, що взагалі не існує на диску. Таку бібліотеку дуже важко виявити звичайними методами, використовуваними антивірусами.

Спосіб поширення

Через зламані сайти й сайти, з яких можна завантажити піратське ПЗ. При відвідуванні користувачем зараженої сторінки, у нього на комп'ютері починає виконуватися спеціальний скрипт, що на підставі поточної дати, установленої на комп'ютері, генерує ім'я сайту, на який необхідно перенаправляти користувача для одержання «персонального» експлойта.

Зараження

При запуску інсталятор записує зашифроване тіло буткіта в останні сектори жорсткого диска, що перебувають за межами використовуваного операційною системою дискового простору. Для забезпечення автозавантаження буткіт заражає MBR комп'ютера, записуючи в нього свій початковий завантажник, що до старту операційної системи зчитує з диска й розвертає в пам'яті основне тіло руткіту, після чого віддає керування ОС і контролює процес її завантаження. Буткіт можна розглядати як гібрид між вірусом і типом завантажувального сектора.

Виявлення й ліквідація

Сімейство даних шкідливих програм поводиться досить потай, на зараженій системі його не можна виявити штатними засобами, так як при звертанні до заражених об'єктів він «підставляє» оригінальні копії. Крім того, основне тіло шкідливої програми (драйвер рівня ядра) не є присутнім на файловій системі, а розташовано в невикористаній частині диска за кордоном останнього розділу. Шкідлива програма завантажує драйвер самостійно, без допомоги операційної системи. Сама ж операційна система не підозрює про наявність драйвера. Виявлення й лікування даного буткіта є найбільш складним завданням з усіх, з якими доводилося зіштовхуватися фахівцям антивірусної індустрії протягом декількох років. Способом боротьби з буткітами є завантаження системи з будь-якого знімного неінфікованого носія, щоб уникнути основного завантаження вірусу після включення комп'ютера.

В останні кілька років збільшилося поширення шкідливих програм (буткітів), що модифікують завантажувальні сектори в процесі зараження системи. Серед самих видних представників – TDL4, Olmasco і Rovnix. Кожний з них використовує різні способи зараження жорсткого диска, це або модифікація головного завантажувального запису (MBR), або модифікація перших секторів завантажувального розділу, тобто VBR або IPL (перші сектори тому, куди передається керування з MBR – Volume Boot Record/Initial Program Loader).

Існує кілька причин використання буткітів у сучасних погрозах:

– Можливість запуску шкідливого коду раніше коду ОС, що дає незаперечні переваги й дозволяє контролювати процес завантаження ОС.

– Як наслідок першого пункту, дозволяє обходити систему моніторингу цілісності ключових компонентів ядра – PatchGuard (практично єдиний спосіб забезпечити виживаність руткіту в x 64-середовищу).

– Можливість глибоко приховувати свій код і, таким чином, робити його невидимим для AV-сканерів.

– Буткіт має посекторну архітектуру зберігання свого тіла на диску, що дає можливість виносити свій шкідливий код і код корисного навантаження далеко за межі файлової системи й навіть розділів диска, роблячи майже неможливим його виявлення.

– Безпечна установка руткіту в системі.

Дроппер

Почнемо із дроппера – компонента, що є споконвічним носієм коду буткіта й відповідає за його установку в системі. Відповідно до наших спостережень, перша відома версія дроппера була скомпільована у квітні минулого року й містила багато відлагоджувальної інформації, тобто не малася на увазі для масового поширення. Цілком імовірно, що Win32/Gapz почали масово поширювати наприкінці літа або початку вересня минулого року. Для підняття своїх привілеїв у системі Win32/Gapz використовує LPE-експлойти й COM Elevation метод.

У процесі аналізу ми виявили, що зараженню Win32/Gapz піддані: 32-бітні Windows XP SP2 і вище (крім Windows Vista і Vista SP1) і 64-бітного Windows XP SP2 і вище. Обговорювана версія дроппера Win32/Gapz здатна заражати Windows XP і Windows 7, включаючи x64 версії, однак на Windows 10/11 буткіт-частина не працює належним чином і після зараження частина, належна до виконання в режимі ядра, не виконувалася.

Дроппер, що встановлює буткіт у систему, ретельно продуманий і здатний обійти сучасні проактивний захист (HIPS), а також піднімати свої привілеї до рівня системи. Крім того, він містить хитрий метод впровадження коду в адресний простір процесу. Є три експортовані функції, на які варто звернути увагу: start, icmnf і isyspf. Короткий опис:

– start – точка входу в дроппер, здійснює його впровадження в адресний простір довіреного процесу explorer.exe;

– icmnf – відповідає за підвищення (ескалацію) привілеїв;

– isyspf – виконує зараження жертви кодом буткіта.

Код дроппера використовує спеціальну секцію, що спроектована в адресний простір процесу explorer. Через цю секцію він завантажує шелл-код у цей процес і далі, за допомогою спеціально сформованого API-виклику, робить його активацію. Відповідно, після того, як шелл-код активований, він довантажує образ дроппера в адресний простір процесу explorer, викликає функцію підвищення привілеїв і ініціює процедуру зараження кодом буткіта, записуючи його на диск.

Після того, як дроппер заразив систему буткітом, його завдання виконане, і він видаляє свій файл із диска.

Шкідливий код MBR

Ми виявили дві модифікації буткіта Win32/Gapz, які розрізняються методами зараження диска жертви.

Давайте розглянемо більше ранню модифікацію буткіта, що націлена на зараження MBR, докладніше. У цьому випадку, код буткіта можна розбити на кілька частин:

– шкідливий MBR;

– код режиму ядра й корисне навантаження, впроваджуване в процеси.

Шкідливий код зберігає свій код режиму ядра й корисне навантаження або перед найпершим розділом, або після останнього розділу на жорсткому диску. Такий підхід дуже схожий на той, котрий використовувався в буткіті Rovnix, за винятком того, що Rovnix заражає VBR.

Що стосується буткіт частини Win32/Gapz, те в ній немає нічого незвичайного: як тільки код зі шкідливого MBR здійснився, він відновлює оригінальний код у пам'яті й читає наступні сектори жорсткого диска, що містять код для наступного виконання, на який і передається керування. Код буткіта перехоплює оброблювач переривання 0x13, int 13h і відслідковує, таким чином, завантаження нижче перерахованих модулів ОС для установки туди перехоплень:

– ntldr (на системах до Windows Vista);

– bootmgr (на системах Vista+);

– winload.exe (на системах Vista+).

Код буткіта ідентифікує кожний з перерахованих вище модулів, використовуючи спеціальні послідовності байт.

Як тільки шкідливий код виявляє, що конкретний модуль читається з жорсткого диска, він модифікує його таким чином, щоб повернути собі контроль після того, як процесор перемкнеться в захищений режим. Буткіт установлює перехоплення на завантажник ядра ОС: це або ntldr у застарілих системах до Windows Vista, або bootmgr в Vista і вище. У випадку з bootmgr, він також перехоплює функцію OslArchTransferToKernel в winload.exe.

Наступний етап, це установка перехоплення на функцію IoInitSystem, що викликається в процесі ініціалізації ОС. Вона перехоплюється шкідливим кодом або з ntldr, або з winload.exe, залежно від версії ОС.

Після того, як шкідливий код з IoInitSystem був виконаний, буткіт відновлює модифіковані байти в образі ядра ntoskrnl і передає керування оригінальної IoInitSystem. Перед передачею керування оригінальному коду, буткіт перезаписує адреса повернення в стеці на свою функцію, що, відповідно, буде виконана по завершенні виконання IoInitSystem. За допомогою такого трюку шкідливий код одержує керування після того, як ядро ОС буде ініціалізовано. Далі шкідливий код зчитує іншу свою частину з жорсткого диска й створює окремий системний потік, що виконує ці інструкції й у завершенні повертає керування ядру. У цій частині буткіта, що виконується в режимі ядра, реалізується руткіт-функціонал, впровадження корисного навантаження в процеси й взаємодію з C&C сервером.

Шкідливий код VBR

Як ми вже згадували, остання модифікація Win32/Gapz заражає VBR тому, що позначений як активний в MBR (Volume Boot Record – перші сектори тому, у яких прописана службова інформація, а також VBR-код, на який передається керування з MBR і який відповідає за подальше завантаження ОС). Буткіт використовує оригінальний підхід для зараження VBR з наступною передачею

керування своєму коду. Для того щоб бути більше потайливим і непомітним, він модифікує тільки трохи байт оригінальної VBR. Суть такого підходу полягає в тім, що він модифікує значення поля «Hidden Sectors» у поле службової структури VBR, при цьому залишаючи код VBR і код IPL недоторканим! IPL, Initial Program Loader – код, на який передається керування після виконання коду VBR, він відповідає за пошук завантажника в рамках файлової системи тому й передає на нього керування. До складу VBR включені наступні частини:

– Bootstrap-код (VBR-код), відповідальний за завантаження IPL.

– BIOS Parameter Block (BPB) – структура даних, що зберігає блок параметрів NTFS.

– Текстові рядки, відображувані користувачеві у випадку помилки.

– 0xAA55 – стандартна двобайтова сигнатура, маркер службового сектора.

У випадку з Win32/Gapz, найцікавішим місцем для аналізу є BPB і особливе поле «Hidden Sectors». Це поле містить кількість секторів, що передують IPL (тобто зсув до IPL у секторах, за допомогою якого код з VBR визначає, куди передавати керування далі) і зберігаються на NTFS-томі, як показано нижче.

Таким чином, у процесі завантаження на чистій системі, VBR-код зчитує 15 секторів, починаючи зі зсуву, зазначеного в «Hidden Sectors», і передає туди керування. Цим і користується буткіт для передачі керування на себе. Він перезаписує це значення, указуючи зсув у секторах до свого шкідливого коду, що зберігається на диску.

У випадку зараження системи, VBR-код викликає на виконання код буткіта замість легального IPL. Код буткіта, як уже згадувалося, записується або перед найпершим розділом диска, або після останнього. В іншому код буткіта, по суті, нічим не відрізняється від версії з MBR-інфектором.

Шкідливий код режиму ядра

Основне призначення безпосередньо тієї частини, що і називається буткітом, описаної вище, полягає в завантаженні шкідливого коду режиму ядра

або руткіту в системний адресний простір, обходячи обмеження, що накладаються ОС для такого привілейованого коду. Ми вже згадували, що цей що завантажується буткітом код, містить у собі руткіт для приховання своєї присутності, механізм роботи з керуючим сервером C&C, а також корисне навантаження (payload), що призначена для впровадження в процеси.

На відміну від Rovnix, TDL4 і інших розповсюджених буткітів, шкідливий код режиму ядра в Win32/Gapz не має структури що виконується PE-файлу. Замість цього він структурований особливим образом. Код складається з 12 об'єднаних між собою блоків, кожний з яких має заголовок – структуру, що зберігає службову інформацію про нього.

Кожний із блоків реалізує певного функціонала: впровадження корисного навантаження, взаємодія з C&C серверами, самозахист і так далі. Функціонал коду режиму ядра є досить складним і може бути розглянутий окремо.

Bootkits vs. Microsoft ELAM

У цій частині хочемо зупинитися на спеціальному засобі, що Microsoft вирішили використовувати для боротьби з різного роду погрозами, особливо, руткітами й буткітами, які намагаються завантажити себе раніше інших драйверів у системі. Засіб називається ELAM, Early Launch Anti-Malware Module і поставляється в складі ОС, починаючи з Windows 10/11. По суті ELAM – це драйвер, надаваний антивірусним вендором, якому гарантований пріоритет при завантаженні драйверів режиму ядра. З погляду ж ядра ОС, ELAM являє собою API для антивірусних драйверів, а також набір правил, яких такому драйверу варто дотримуватися. Одна з головних можливостей цього засобу полягає в тім, що він гарантовано дозволяє AV-драйверу завантажуватися раніше інших драйверів у системі й, таким чином, виходити за рамки звичайних правил автозавантаження, регламентованих для інших драйверів.

Ми відзначаємо, що ELAM сам по собі не може бути так ефективний для боротьби з буткітами, оскільки це частина ядра ОС, а буткіт одержує керування набагато раніше.

У той же час, слід зазначити, що ELAM є частиною декларуємої Microsoft концепції або схеми «Безпечного завантаження» – Secure Boot. У випадку з Secure Boot, програмне забезпечення, убудоване в UEFI (таке ПЗ одержує керування як тільки комп'ютер почав свою роботу) може контролювати цілісність і легітимність коду, на який передається керування в процесі виконання коду, що передує виконанню основного коду ОС. Концепція Secure Boot, разом з новим стандартом UEFI, покликана запобігти компрометації критичних структур даних ОС/UEFI з боку буткітів.

Структурна схема системи зображена на рисунку 3.1.

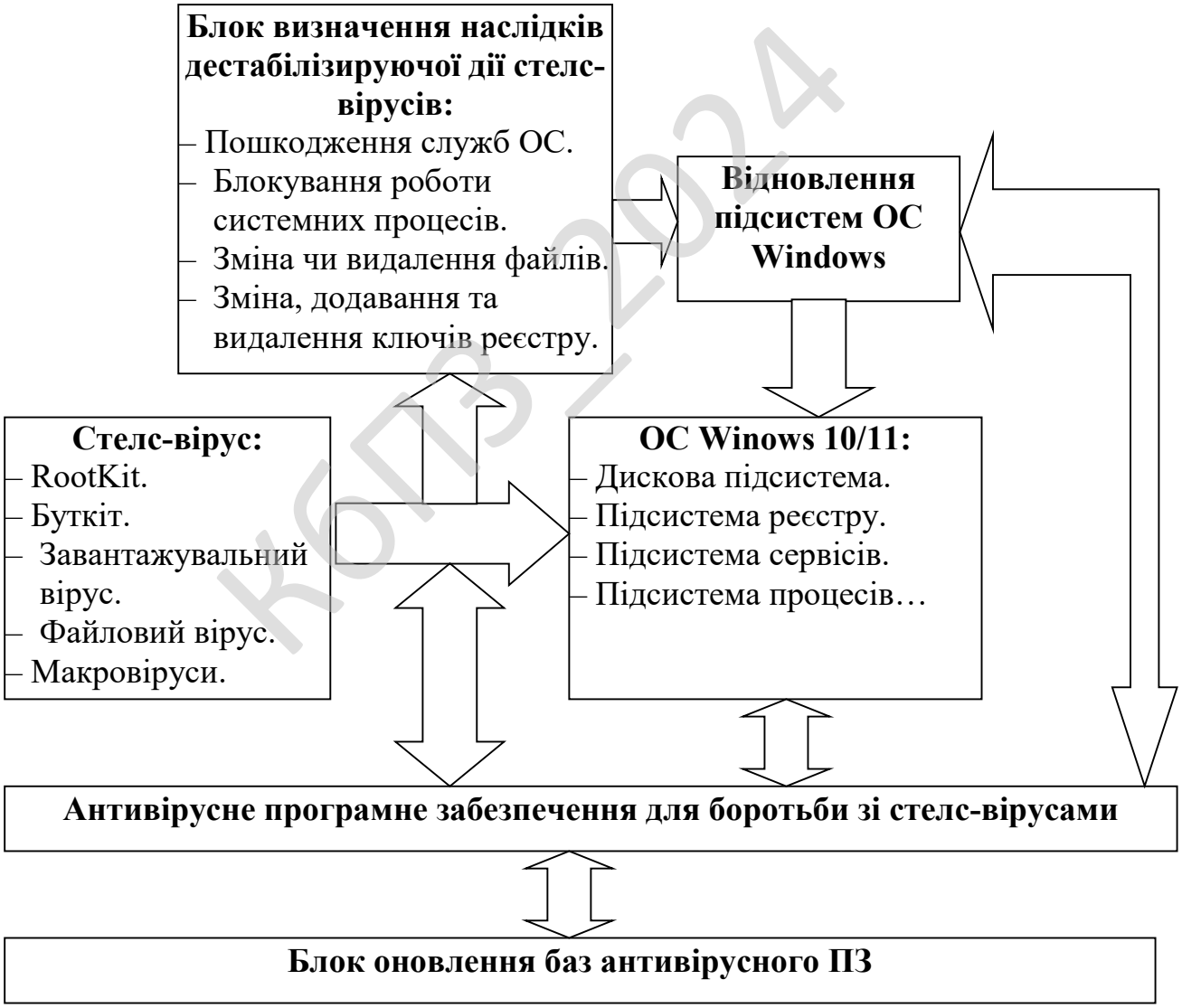

Рисунок 3.1 – Структурна схема системи

Розроблене ПЗ під час роботи проводе:

– Контроль змін реєстру;

– Контроль змін файлової підсистеми;

– Контроль змін системних служб;

– Контроль змін підсистеми процесів.

Структурна схема складається з наступних структурних блоків, більш детальний опис роботи яких наданий вище:

1. Антивірусне програмне забезпечення для боротьби зі стелс-вірусами.

2. Стелс-вірус:

– RootKit.

– Буткіт.

– Завантажувальний вірус.

– Файловий вірус.

– Макровіруси.

3. ОС Winows 10/11:

– Дискова підсистема.

– Підсистема реєстру.

– Підсистема сервісів.

– Підсистема процесів.

4. Блок визначення наслідків дестабілізируючої дії стелс-вірусів:

– Пошкодження служб ОС.

– Блокування роботи системних процесів.

– Зміна чи видалення файлів.

– Зміна, додавання та видалення ключів реєстру.

5. Відновлення підсистем ОС Windows.

6. Блок оновлення баз антивірусного ПЗ.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Система працює під керівництвом ОС Windows 10/11. Розроблене ПЗ проводе сканування підсистем ОС у прихованому режимі та контролює зміни.

Система функціонально визначає наявність стелс-вірусів у ОС.

Вона дозволяє протидіяти наступним діям стелс-вірусів.

Система протидіє наступним методам роботи RootKit:

1. Методи перехоплення API функцій у режимі користувача (user mode):

– Модифікація машинного коду прикладної програми.

– Модифікація таблиці імпорту.

– Перехоплення функцій LoadLibrary і GetProcAddress.

– Метод, що сполучить методику 2 і 3.

– Модифікація програмного коду API функції.

– Модифікація бібліотек DLL на диску.

2. Перехоплення функцій у режимі ядра (kernel mode):

– Перехопленням Native API.

Реалізуються наступні методики виявлення RootKit у ОС:

– Порівняння двох «знімків» системи.

– Порівняння даних, що повертаються API функціями різного рівня.

– Аналіз у пам'яті функцій основних бібліотек на предмет наявності змін їхнього машинного коду.

– Аналіз і відновлення ServiceDescriptorTable.

Реалізується наступна методика виявлення BootKit у системі:

– ELAM.

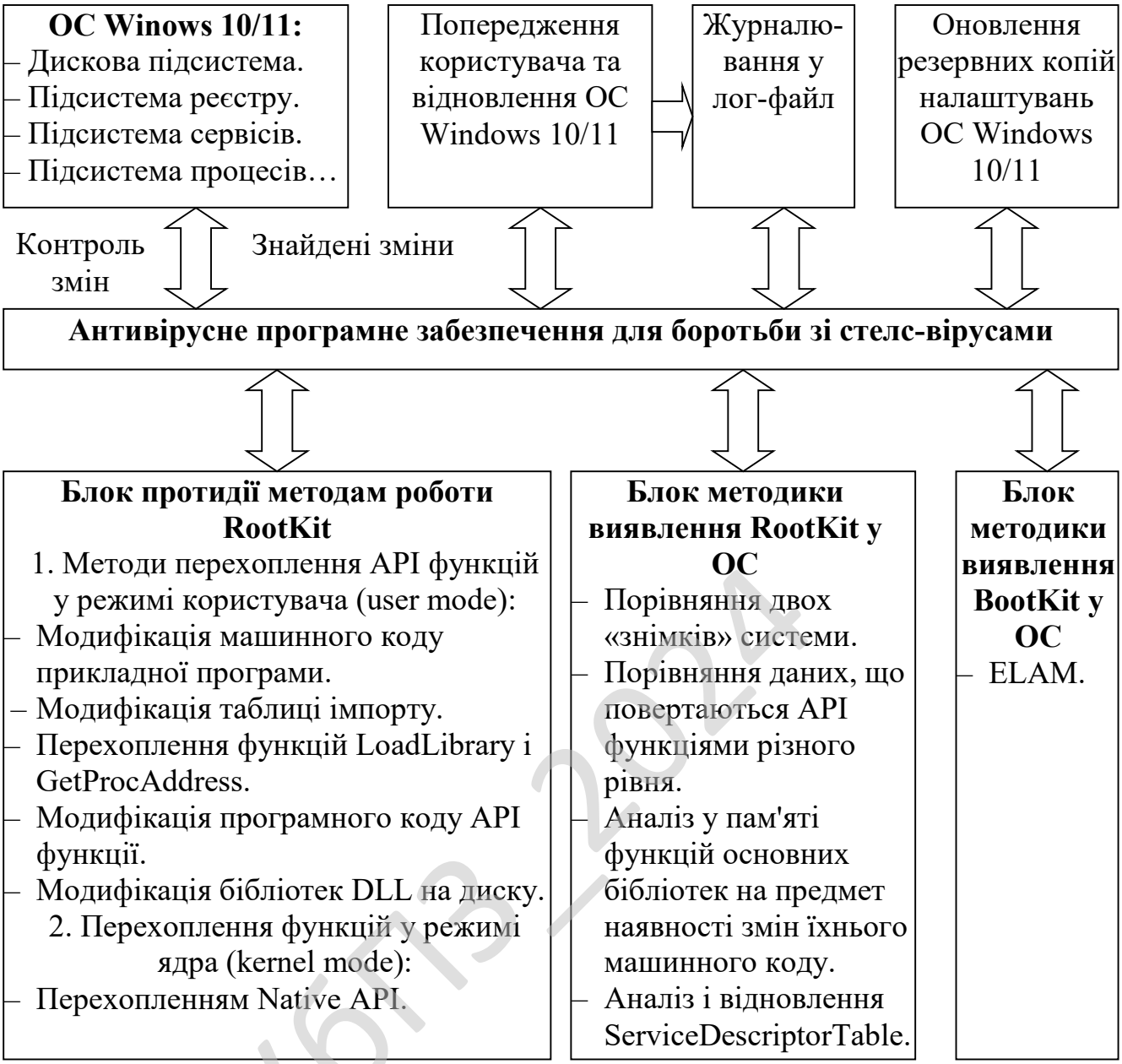

Рисунок 3.2 – Функціональна схема системи

При роботі розробленого програмного забезпечення перевіряються наступні системи:

– Дискова підсистема, а саме системні файли та папки (C:\WINDOWS) та файли користувача (C:\Documents and Settings).

– Підсистема реєстру, особливо гілки HKEY_LOCAL_MACHINE, HKEY_CURRENT_USER.

– Підсистема сервісів з основними службами які підтримують функціювання ОС.

– Підсистема процесів, а саме файли та їх образи у пам’яті основних процесів таких як explorer.exe, lsass.exe, csrss.exe, svchost.exe та інші.

Всі дії додаються до журнал роботи ПЗ. Також програма проводить синхронізація часу через мережу Інтернет. У програмі існує функція зберігання та оновлення резервних копій налаштувань підсистем ОС Windows 10/11, які зберігаються у заданій папці на диску. При запиту користувача розроблене ПЗ проводе активне сканування та при знайдені змін виводить на екран попередження користувача та проводить відновлення підсистем ОС Windows.

Визначення подальших напрямків розробки систем захисту від стелс-вірусів:

1. Комплексне рішення для захисту персонального ПК при роботі в Інтернеті від основних інформаційних погроз: вірусів, хакерів, спаму й шпигунських програм. Функції антивірусного захисту містять у собі: антивірусну перевірку поштового трафіка, перевірку інтернет-трафіка, захист файлової системи, проактивний захист, захист від інтернет-шахрайства.

2. Комплексне рішення для захисту даних, збережених на ПК різних типів. До складу програми необхідно включити антивірусний сканер та антивірусний монітор.

3. Програма повинна здійснювати перевірку, що включає в себе:

– перевірку за вимогою пам'яті ПК, окремої папки або конкретного файлу;

– постійну перевірку: автоматично перевіряються всі вхідні або об'єкти, що змінюються, а також файли при спробі доступу до них;

– перевірку за розкладом інформації в пам'яті ПК.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання дипломного проектування, наведена на рисунку 3.3. Після завантаження та запуску ПЗ ми потрапляємо до головного вікна ПЗ далі можемо переглянути довідкову систему, налаштування ПЗ, через обробник помилок із застосуванням БД сигнатур проводити моніторинг системи, а саме: Сканування файлів; Сканування сервісів; Сканування реєстру; Журналювання роботи ПЗ.

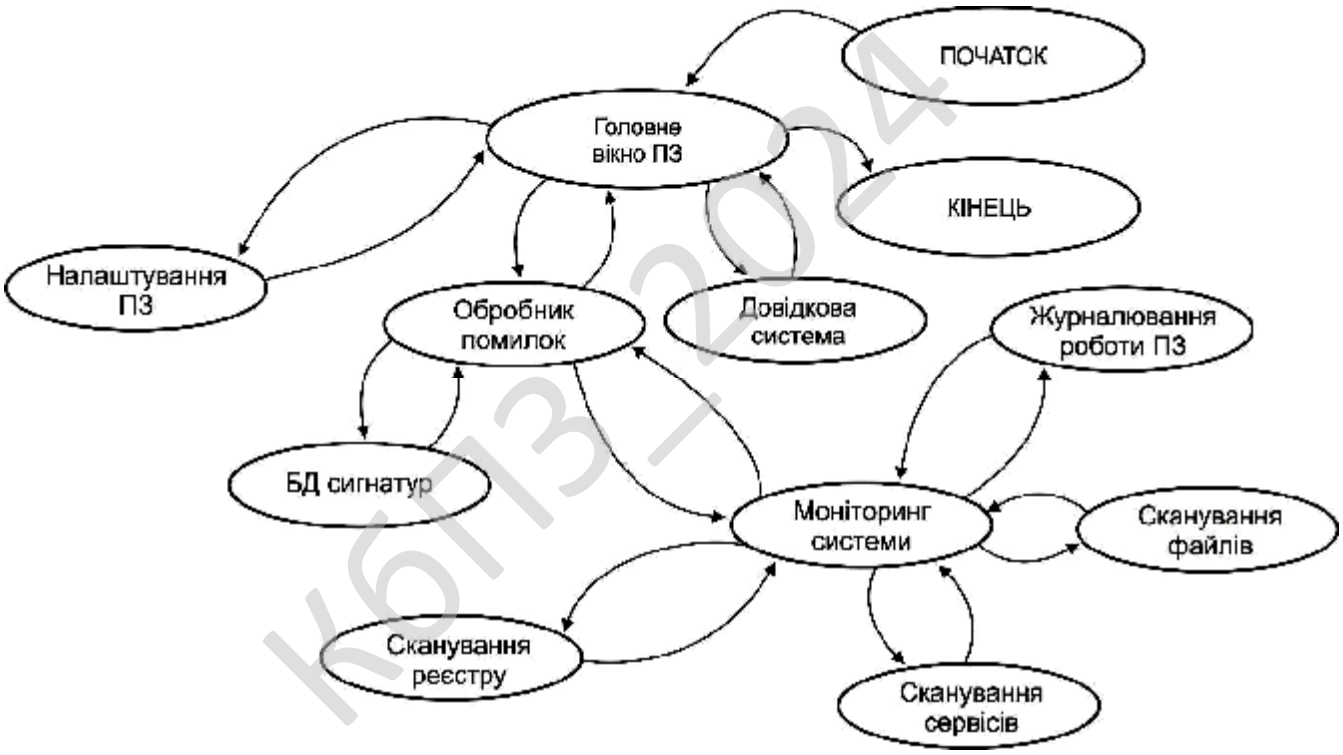

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків:

– Ініціалізація ПЗ.

– Пошук файлу БД сигнатур.

– Запит – знайдено?

– Підключення до БД сигнатур.

– Створення та початкове заповнення БД сигнатур.

– Виведення вікна парольного захисту.

– Запит – отримано пароль?

– Пароль вірний (запит).

– Запис до журналу роботи ПЗ.

– Підпрограма резервування налаштувань підсистем.

– Сканування поточного стану системи.

– Запит – оновлення підсистем?

– Підпрограма оновлення підсистем.

– Запис до журналу роботи ПЗ.

– Запит WM_CLOSE?

– Звільнення ресурсів ПЗ.

На рисунку 4.2 наведено блок-схему підпрограми оновлення підсистем. Її робота складається з виконання наступних кроків:

– Читання поточних налаштувань сканування ПЗ.

– Формування списку сканування підсистем.

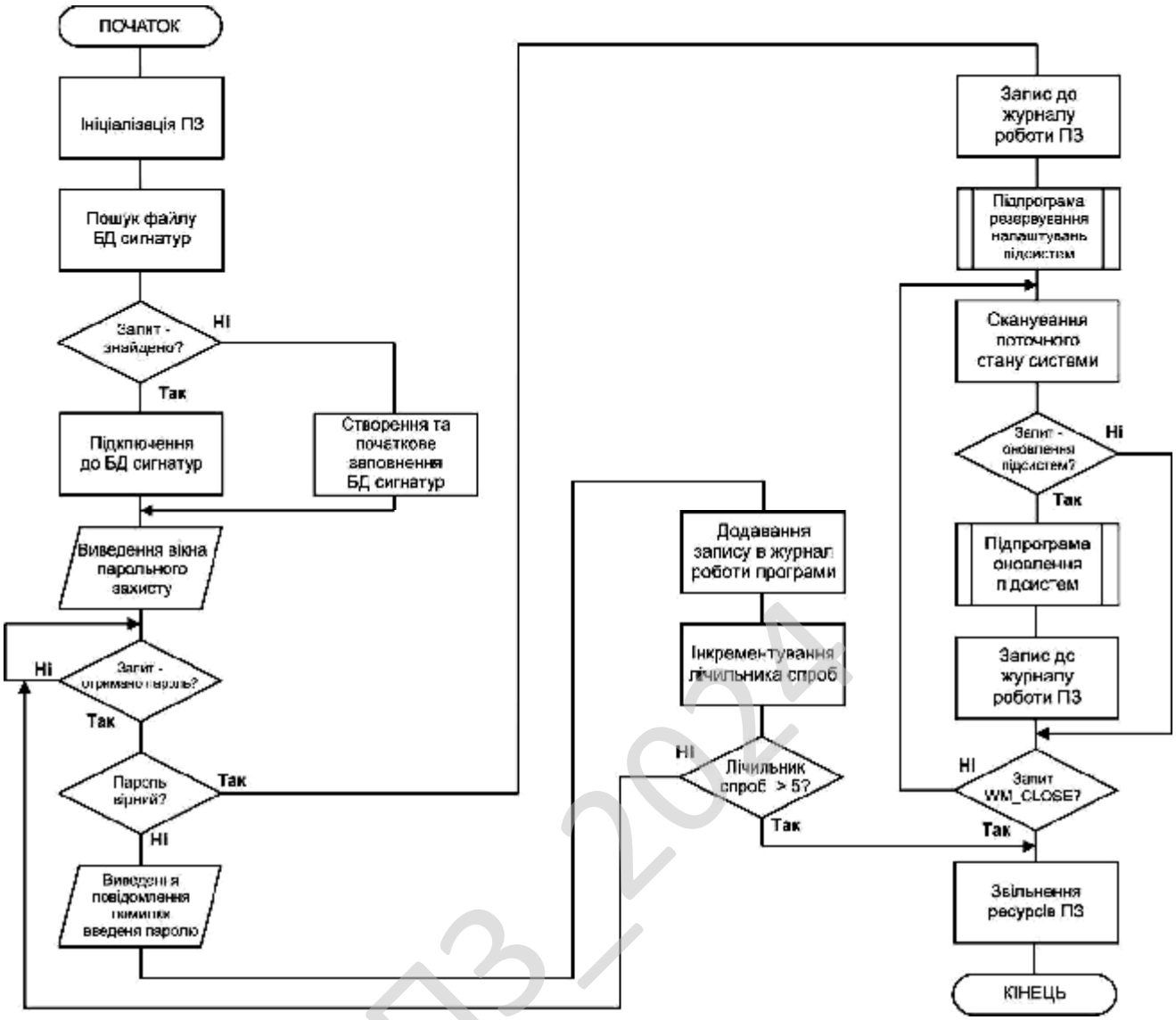

Рисунок 4.1 – Блок-схема основної програми

– Сканування підсистем.

– Знайдені зміни (запит).

– Виведення повідомлення змін.

– Оновлення підсистем.

На рисунку 4.2 наведено блок-схему підпрограми резервування налаштувань підсистем. Її робота складається з виконання наступних кроків:

– Сканування підсистем.

– Резервування служб (запит).

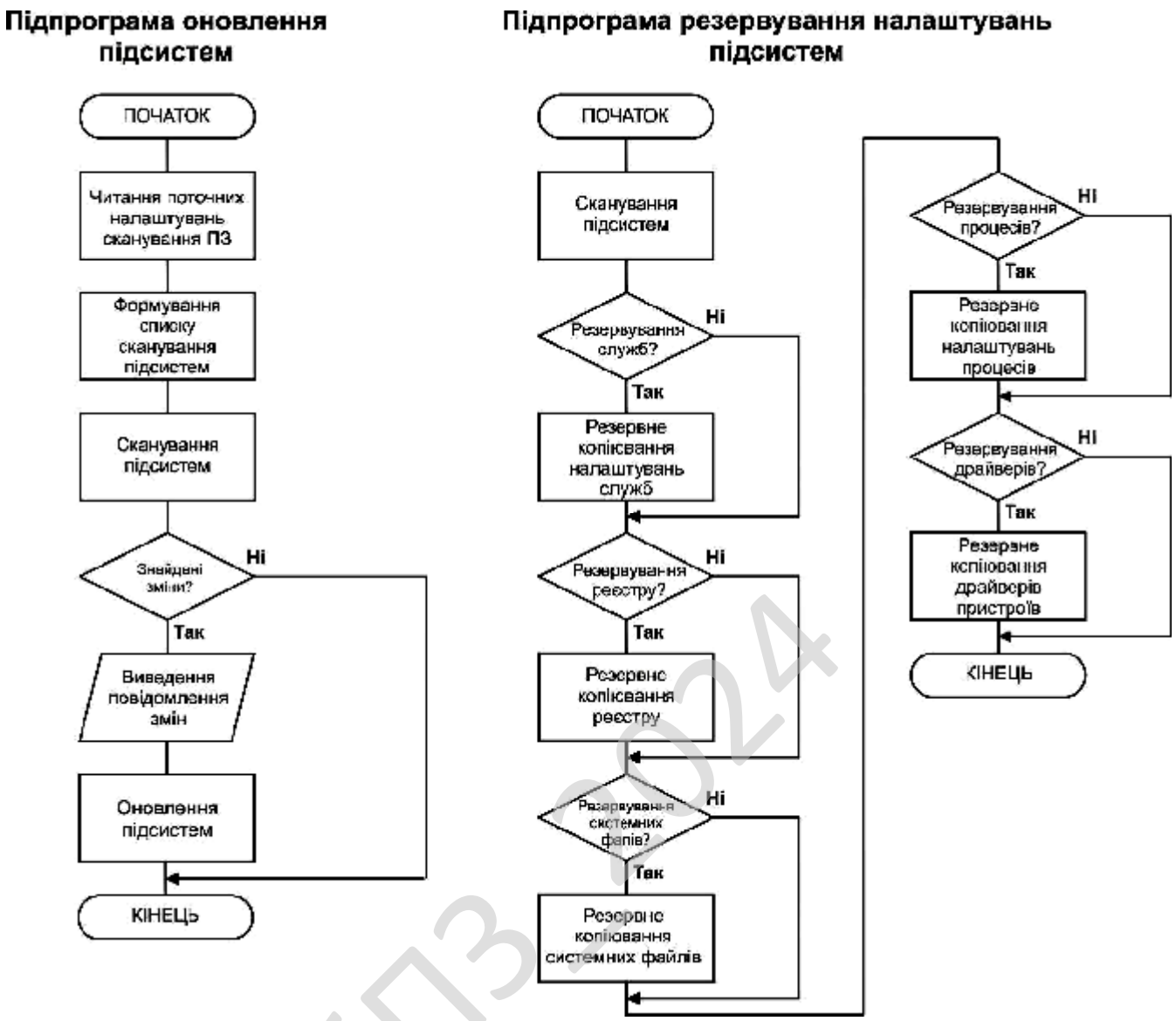

Рисунок 4.2 – Блок-схема підпрограми

– Резервне копіювання налаштувань служб.

– Резервування реєстру (запит).

– Резервне копіювання реєстру.

– Резервування системних фалів (запит).

– Резервне копіювання системних файлів.

– Резервування процесів (запит).

– Резервне копіювання налаштувань процесів.

– Резервування драйверів (запит).

– Резервне копіювання драйверів пристроїв.

Розглянемо розроблені та вдосконалені класи, процедури, функції, типи даних та інше.

Файл MOD_BackupMachine.pas

```
TShellObject = Record
// Запис
DoIt    : boolean;
// це є захист від завантаження додаткових копій
LiteralName   : string[50];
// ім’я об’єкту
SourcePath : string[255];
// шлях до об’єкту
Prefix  : string[20];
// префікс об’єкта
Sufix  : string[20];
//суфікс об’єкта
DestinationFolder : string[255];
// шлях завантаження копії
AttachDateTime : boolean;
end;
TBackupFile = Record
// Запис
GroupName : string[20];
// Літеральне ім’я ПК
DoIt     : Boolean;
// це є захист від завантаження додаткових копій
ShellObject   : TShellObject;
// вище розглянута запис
end;
```

Файл mod_regScanner.pas

```
Hist: array [0..maxhist-1] of record
name: string;
// використовуємое ім’я
foc: pdescriptor;
// дескриптор
dfoc: integer
// дані
end;
```

Файл MOD_RegistryStartup1.pas та MOD_RegistryStartup2.pas

```
procedure RunProgramsClick(Sender: TObject);
// початок сканування.
procedure But_DeleteClick(Sender: TObject);
// видалення ключа реєстру.
procedure But_SaveClick(Sender: TObject);
// збереження ключа реєстру.
procedure CopyrightMouseEnter(Sender: TObject);
// автоматичний запуск при наведені
// на блок форми мишою.
procedure CopyrightMouseLeave(Sender: TObject);
// автоматичний запуск при
// віддаленні миши від блоку форми.
procedure CopyrightClick(Sender: TObject);
// авторські права.
function  GetKey:Int64;
// завантаження ключа реєстру типу Int64.
function  GetKeySt:String;
// завантаження ключа реєстру типу String.
procedure ShowPrograms(Key : Integer; Way : String);
// Виведення на екран форми.
```

Спільні класи розроблених файлів.

Клас TAbstractAllocator.

Оголошення класу: Type AbstractAllocator = class()

Опис: Абстрактне виділення покажчика для захоплення відео потоку.

Клас TASFWriter.

Оголошення класу: type TASFWriter = class(A,B)

Опис: Цей клас розроблений, щоб створити тимчасовий файл ASF (потокової версії).

3) Клас TColorControl

Оголошення класу:

type TColorControl = class()

Опис: Установка і отримання контролю вікна. Клас підтримує апаратні API функції операційної системи. Одночасно можна провести перевірку на здібність доступу програми до API функцій операційної системи.

Клас TFilter.

Оголошення класу: type TFilter = class(A,B)

Опис: Клас пропонує можливість створення і підключення до програми додаткових фільтрів.

Для синхронізації (відновлення) даних будемо використовувати три процедури – Updatelog, Threadstart і Threadstop які будуть використовуватися в методі Synchronize.

```
procedure Tmondirthread.Updatelog;
begin
  fmondirmain.lblog.Items.Add(Timetostr(time)+': зміна!');
  if fmondirmain.lblog.Items.Count > 200 then
// Показуємо тільки 200 записів
    fmondirmain.lblog.Items.Delete(0);
  fmondirmain.lblog.Itemindex := fmondirmain.lblog.Items.Count-1;
end;
procedure Tmondirthread.Threadstart;
begin
  fmondirmain.sbmain.Panels[1].Text := 'Активний';
  fmondirmain.lblog.Items.Add(Timetostr(time)+': монітор запущений');
end;
procedure Tmondirthread.Threadstop;
begin
  fmondirmain.sbmain.Panels[1].Text := 'Відключений';
  fmondirmain.lblog.Items.Add(Timetostr(time)+': монітор зупинений');
  fmondirmain.mmstart.Enabled := True;
// Відключаємо кнопку mmstart
  fmondirmain.mmstop.Enabled := False;
// Включаємо кнопку mmstop
end;
```

Алгоритм роботи процедури Execute. Для початку сканування каталогу викличемо наступну функцію.

```
Findfirstchangenotification
//створює об'єкт, що сигналізує
var
Handlechange: Thandle;
// Handle створюваного об'єкта
begin
  Handlechange :=
```

```
            Findfirstchangenotification(
              Pchar(Fpath),
        // каталог, що перевіряється
              False,
        // Підкаталоги не перевіряються
              FILE_NOTIFY_CHANGE_FILE_NAME +
        // прапор змін
              FILE_NOTIFY_CHANGE_ATTRIBUTES );
        Execute Win32Check(Handlechange <> INVALID_HANDLE_VALUE).
```

У випадку помилки створення об'єкта, видамо повідомлення про помилку й закінчимо процедуру. При успішному створенні синхронізуючого об'єкта викличемо процедуру Threadstart для відновлення головної форми.

Далі, у циклі продовжуємо перевірку змін вмісту каталогу.

```
        Synchronize(Threadstart);
        // Повідомлення про старт потоку

        procedure Tmondirthread.Execute;
        var
        Handlechange: Thandle;
        // Handle створюваного об'єкта для очікування події
        begin
// -- Створюємо об'єкт для очікування події
          Handlechange :=
            Findfirstchangenotification(
              Pchar(Fpath),
        // каталог, що перевіряється
              False,
        // Підкаталоги не перевіряються
              FILE_NOTIFY_CHANGE_FILE_NAME+
        //Перевірка/ створення/ видалення/
              FILE_NOTIFY_CHANGE_ATTRIBUTES);
        // перейменування/ зміни файлів
// -- При помилці Win32Check виводить повідомлення й перериває Execute.
             Win32Check(Handlechange <> INVALID_HANDLE_VALUE);
             Synchronize(Threadstart);
        // Повідомлення про старт потоку
          try
// -- Цикл, поки для потоку не буде видана команда Terminate
             while not Terminated do
             begin
```

```
      case Waitforsingleobject(Handlechange,1000) of
        WAIT_FAILED: Terminate;
// Помилка, завершуємо потік
        WAIT_OBJECT_0: Synchronize(Updatelog);
// Повідомляємо про зміну
      end;
      Findnextchangenotification(Handlechange);
    end;
  finally
    Findclosechangenotification(Handlechange);
  end;
  Synchronize(Threadstop);
// Повідомляємо про завершення потоку
end;
```

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів взаємодії з модулями ПЗ я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою.

Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем. При розробці програмного продукту я вирішував проблеми неправильної класифікації алгоритмів, що використовуються в існуючих розробках, а також нерозуміння або неправильного тлумачення методів їх реалізації, що приводить до чисельних проблем при просуванні продукту на ринок. Далі розглянута загальна класифікація алгоритмів і їх застосування на практиці.

Windows система багатозадачна. Кілька програм можуть працювати одночасно під керуванням ОС. Процес – це екземпляр виконуваного ПЗ. Насправді сам по собі він нічого не виконує, він створюється при запуску ПЗ, містить у собі службову інформацію, через яку система з ним працює, так само йому виділяється необхідна пам'ять під код і дані.

Для того, щоб програма запрацювала, у ньому створюється потік. Будь-який процес містить у собі хоча б один потік, і саме він відповідає за виконання коду й одержує на цей процесорний час.

Цим і досягається уявна паралельність роботи програм, або, як її ще називають, псевдо паралельність.

Процес – це одне з найважливіших понять інформатики. За визначенням це екземпляр виконуваної програми, включаючи змінні та стан програми. Процеси слід відрізняти від нитей виконання – ниті є складовими процесу, так що кожен процес має власний адресний простір, а ниті розподіляють спільний адресний простір процесу. Процес складається щонайменше з одної, головної ниті виконання.

Кожному процесу мають бути виділені такі ресурси: процесор; пам'ять; доступ до пристроїв введення-виведення; файли.

Кожен процес має «батька» (батьківський процес). Він також може мати (але не мусить) «нащадків» (дочірні процеси). Таким чином створюється дерево процесів.

Керування процесами здійснює ядро операційної системи. Під час виконання процес може знаходитися в одному із станів:

– виконання;

– очікування на доступ до ресурсів, які надає операційна система;

– готовності до виконання;

– щойно створений;

– завершений;

– зомбі-процес.

Інформація що міститься в процесі. Процес містить інформацію про (список не виключний):

– повний командний рядок запуску виконуваного процесом завдання;

– інформація про відведений процесу адресний простір, включно зі стеком;

– посилання на поточний робочий каталог і кореневий каталог процесу (останній служить для обмеження доступу процесу до файлової структури);

– таблиця відкритих процесом файлів;

– так зване оточення процесу, тобто перелік заданих для даного процесу змінних з їх поточними значеннями;

– атрибути, що визначають права і привілеї процесу

– таблиця обробників сигналів;

– вказівка на батьківський процес;

– призначена для користувача маска або маска доступу – вказівка на те, які права треба видалити при створенні нового файлу або каталогу із стандартного набору прав, присвоєних файлу (каталогу);

– перелік нитей процесу, якщо операційна система підтримує багатонитевість.

Розглянемо роботу з потоками.

```
program S;
uses
  Kol,
  Messages,
  Windows,
  Kolgif;
{$R *.RES}
{$R pic.res}
var
  forma,
  pb: pcontrol;
  Gif: pgif;
Picstream: Pstream;
// обробка даних
procedure Drawpaint( Dummy: Pointer; Sender: Pcontrol; DC: HDC );
begin
  Gif.Draw(PB.Canvas.Handle, 0, 0);
end;
//"закидаємо" ресурс на диск
procedure Savepic( Dummy : Pointer; Sender: Pcontrol );
begin
  //Getwindowsdir[1] – створюємо файло-потік
  Picstream := Newwritefilestream(Getwindowsdir[1] + ':\logo.gif');
  // переводимо дані з ресурсу в потік
  Resource2Stream(Picstream, Hinstance, 'LOGOTIP', RT_RCDATA);
  Picstream.free;
end;
// показуємо картинку з ресурсу
procedure Showforma( Dummy : Pointer; Sender: Pcontrol );
begin
  Picstream := Newmemorystream;
```

```
//створюємо потік для зображення
  // переводимо дані з ресурсу в потік
  Resource2Stream(Picstream, Hinstance, 'LOGOTIP', RT_RCDATA);
  //Вказуємо з якої позиції в нас будуть читатися дані
  Picstream.Position := 0;
  gif := Newgif;
  gif.Loadfromstream(Picstream);
  PB := Newpaintbox(forma);
  PB.Onpaint := Tonpaint( Makemethod( nil, @Drawpaint ) );
  // зберігаємо дані на диск по запиту
  PB.Onclick := Tonevent( Makemethod( nil, @Savepic ) );
  PB.Width := gif.Width ;
  PB.Height:= gif.Height;
  PB.Centeronparent;
  Picstream.free;
end;
begin
  Applet := Newapplet('Робота з потоком');
  Forma := Newform(Applet, 'Робота з потоком виведення');
  Forma.Setsize(310, 120);
  Forma.Centeronparent;
  Forma.Font.Fontname := 'MS Sans Serif';
  Forma.Font.Fontheight := 9;
  Forma.onshow :=  Tonevent(Makemethod(nil, @Showforma));
  Run(Applet);
end.
```


4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм RC5, який являє собою блоковий шифр із безліччю параметрів: розміром блоку, розміром ключа й числом раундів. В алгоритмі RC5 передбачені три операції: XOR, додавання й циклічні зрушення. На більшості процесорів операції циклічного зрушення виконуються за постійний час, змінні циклічні зрушення являють собою нелінійну функцію. Циклічні зрушення залежать як від ключа, так і від даних.

В RC5 використовується блок змінної довжини, але в приводиться прикладі, що буде розглянутий, 64-бітовий блок даних. Шифрування використовує $2r+2$ залежних від ключа 32-бітових слів – S_0, S_1, S_2,... S_{2r+1} – де r – число раундів. Для шифрування спочатку потрібно розділити блок відкритого тексту на два 32-бітових слова: A и B. (При впакуванні байтів у слова в алгоритмі RC5 дотримується угода про прямий порядок (little-endian) байтів: перший байт займає молодші біти регістра A й т. ін.) Потім:

$$A=A + S_0$$

$$B = B + S_0$$

Для і від 1 до r:

$$A = ((A \oplus \quad B) <<< B) + S_{2i}$$

$$B = ((B \oplus \quad A) <<< A) + S_{2i+1}$$

Вихід перебуває в регістрах A и B.

Розшифрування теж нескладно. Потрібно розбити блок відкритого тексту на два слова, A й B, а потім:

Для i від r до 1 із кроком -1:

$$B = ((B - S_{2i+1}) >>> A) \oplus A$$

$$A = ((A - S_{2i}) >>> B) \oplus B$$

$$B = B - S_i$$

$$A = A - S_0$$

Символом «>>>» позначене циклічне зрушення вправо**.** Звичайно ж**,** всі додавання й вирахування виконуються по модулю 2^{32}.

Створення масиву ключів складніше, але теж прямолінійно. Спочатку байти ключа копіюються в масив L із 32-бітових слів, доповнюючи при необхідності заключне слово нулями. Потім масив S ініціалізується за допомогою лінійного конгруентного генератора по модулю 2^{32}:

$$S_0 = P$$

Для i від 1 до $2(r + 1) – 1$:

$$S_i = (S_{i-1} + Q) \bmod 2^{32}$$

де P = 0xb7e15163 і Q = 0x9e3779b9.

Нарешті, потрібно підставити L в S:

$$i = j = 0$$

$$A = B = 0$$

Виконати $3n$ раз (де n – максимум від $2(r + 1)$ і c):

$$A = S_i = (S_i + A + B) <<< 3$$

$$B = L_i = (L_i + A + B) <<< (A + B)$$

$$i = (i + 1) \bmod 2(r + 1)$$

$$i = (j + 1) \bmod c$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню користувача: Файл; Журнал дій; Налаштування; Довідка.

– Лівого блоку деревовидних параметрів: Реєстр; Сервіси; Файли системи; Драйвери пристроїв; ПЗ що інстальоване до системи; Налаштування системи; Критичні секції.

– Центрального блоку типу сканування: Файлова підсистема; Реєстрова підсистема; Програмна підсистема; Драйверна підсистема; Сервіси та критичні секції; Файлові асоціації.

– Правого верхнього блоку кнопок налаштувань: Шаблони файлів; Головне меню; Захист.

– Правого нижнього блоку кнопок довідкової: Список тем; Авторське право; Написати листа.

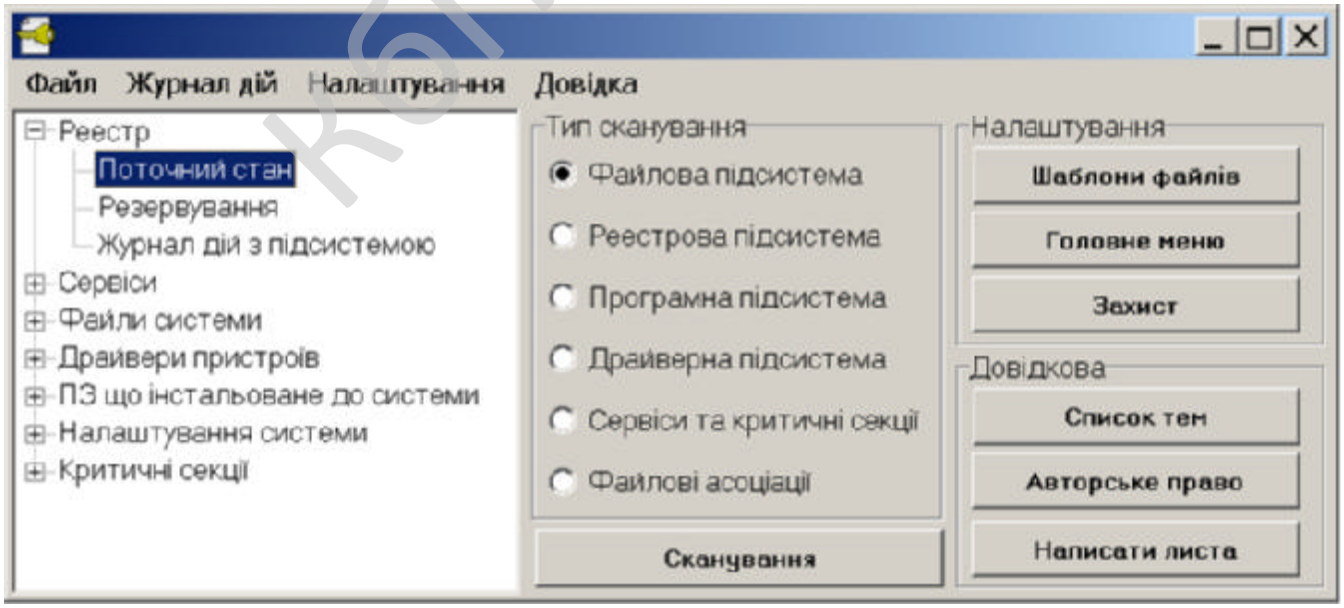

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права.

Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно.

Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (дискету чи CD-ROM). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

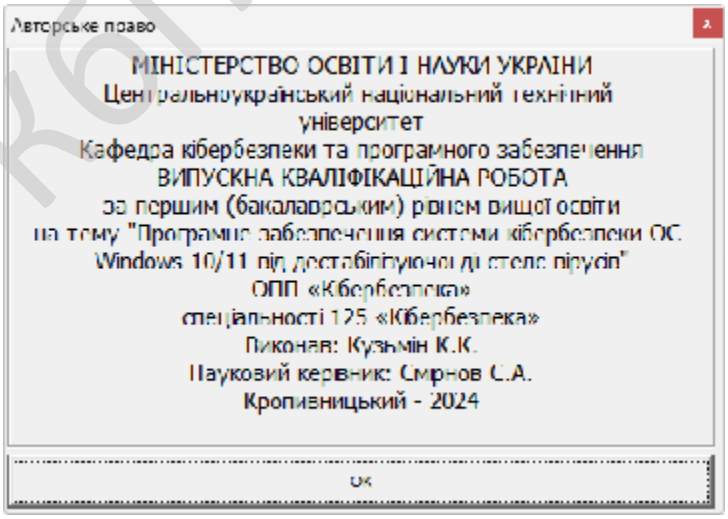

Рисунок 5.2 – Довідка автора

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

– Досліджена система ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів. Це

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

Програма призначена для виконання під управлінням багатозадачної операційної системи кібербезпеки Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RC5.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

2. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

3. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

4. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

5. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

6. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

7. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

8. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

9. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

10. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

11. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

12. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

13. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

14. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

15. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

16. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

17. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

18. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

19. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

20. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

21. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

22. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

23. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

24. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

25. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

26. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

27. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

28. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

30. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

31. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

32. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

33. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

34. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

35. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

36. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

37. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних

систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

38. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

39. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

40. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

41. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

42. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

43. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

44. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

45. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

46. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

47. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

48. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

49. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп’ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

50. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп’ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 136-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 84 аркуші.

8 Етапи розробки

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 ПЗ.

8.6 Віднаходження ПЗ, 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 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2024 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

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 41

Літера: РП

Кропивницький – 2024 року

ФАЙЛ DATA_1.PAS (МОДУЛЬ ФОРМИ СКАНУВАННЯ РЕЄСТРУ ТА КЛЮЧІВ АВТОЗАВАНТАЖЕННЯ.)

```
unit DATA_1;

{ ------------------------------------------------------------------------------
  Центральноукраїнський національний технічний університет
  Програмне забезпечення системи кібербезпеки
  ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів
  Розробив: студент Кузьмін Кирило Костянтинович, КБ-21-3СК
------------------------------------------------------------------------------}

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, Registry, ComCtrls, Buttons, ShellAPI;

type
TForm_1 = class(TForm) class(TForm) // клас який розроблено
RunPrograms: TListBox;
Keys: TTreeView;
But_Delete: TBitBtn;
But_Exit: TBitBtn;
But_Save: TBitBtn;
But_Add: TBitBtn;
SaveFile: TSaveDialog;
Copyright: TLabel;
procedure FormCreate(Sender: TObject);
procedure KeysChange(Sender: TObject; Node: TTreeNode);
procedure But_ExitClick(Sender: TObject);
procedure RunProgramsClick(Sender: TObject);
procedure But_DeleteClick(Sender: TObject);
procedure But_SaveClick(Sender: TObject);
procedure CopyrightMouseEnter(Sender: TObject);
procedure CopyrightMouseLeave(Sender: TObject);
procedure CopyrightClick(Sender: TObject);
function  GetKey:Int64;
function  GetKeySt:String;
function  GetWay:String;
procedure ShowPrograms(Key : Integer; Way : String);
procedure But_AddClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure RunProgramsDblClick(Sender: TObject);
private
procedure PreparingButtons(Active : Boolean);
procedure DeleteItem(Key : Integer; Way : String; ValueName : String);
procedure Reload;
procedure LoadValues;
//зона бачення змінних
public
end;

var // змінні
Form1: TForm_1; // створення екземпляру класу
LOCAL_MACHINE_NUM,
CURRENT_USER_NUM    : Integer;

implementation

uses Unit2;

{$R *.DFM}

// реалізація функції класу TForm_1
function TForm_1.GetKey:Int64;
begin
If Keys.Selected.AbsoluteIndex < LOCAL_MACHINE_NUM
```

```
Then GetKey := HKEY_CURRENT_USER
Else GetKey := HKEY_LOCAL_MACHINE;
end;

// реалізація функції класу TForm_1
function TForm_1.GetKeySt:String;
begin
If Keys.Selected.AbsoluteIndex < LOCAL_MACHINE_NUM
Then GetKeySt := 'HKEY_CURRENT_USER\'
Else GetKeySt := 'HKEY_LOCAL_MACHINE\';
end;

// реалізація функції класу TForm_1
function TForm_1.GetWay:String;
begin
GetWay := 'Software\Microsoft\Windows\CurrentVersion\' + Keys.Selected.Text;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.ShowPrograms(Key : Integer; Way : String);
var // змінні
Reg : TRegistry;
begin
Reg := TRegistry.Create;
Reg.RootKey := Key;
If Reg.OpenKey(Way, False)
Then Begin
Reg.GetValueNames(RunPrograms.Items);
End;
Reg.Free;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.LoadValues;
begin
PreparingButtons(False);
If (Keys.Selected.AbsoluteIndex <> LOCAL_MACHINE_NUM) And
(Keys.Selected.AbsoluteIndex <> CURRENT_USER_NUM)
Then Begin
But_Add.Enabled := True;
But_Save.Enabled := True;
ShowPrograms(GetKey,'Software\Microsoft\Windows\CurrentVersion\' +
Keys.Selected.Text);
End
Else Begin
RunPrograms.Clear;
But_Add.Enabled := False;
But_Save.Enabled := False;
End;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.Reload;
var // змінні
Temp : TStrings;
Reg  : TRegistry;
i    : Integer;
Flag : Integer;
begin
But_Add.Enabled  := False;
But_Save.Enabled := False;
Reg := TRegistry.Create;
Temp := TStringList.Create;
Flag := 0;
Keys.Items.Clear;
Reg.RootKey := HKEY_CURRENT_USER;
If Reg.OpenKey('Software\Microsoft\Windows\CurrentVersion', False)
Then Begin
Temp.Clear;
```

```
Reg.GetKeyNames(Temp);
For i := 0 To (Temp.Count - 1) Do
If (Length(Temp.Strings[i]) >= 3) And
    (UpperCase(Copy(Temp.Strings[i],1,3))='RUN'
Then Begin
If Flag = 0
Then Begin
CURRENT_USER_NUM :=
Keys.Items.Add(Keys.TopItem,'HKEY_CURRENT_USER').AbsoluteIndex;
Keys.Items.AddChild(Keys.Items.Item[CURRENT_USER_NUM],Temp[i]);
Inc(Flag);
End
Else Begin
Keys.Items.AddChild(Keys.Items.Item[CURRENT_USER_NUM],Temp[i]);
End;
End;
Temp.Free;
End;
Reg.CloseKey;
Temp := TStringList.Create;
Flag := 0;
Reg.RootKey := HKEY_LOCAL_MACHINE;
If Reg.OpenKey('Software\Microsoft\Windows\CurrentVersion', False)
Then Begin
Temp.Clear;
Reg.GetKeyNames(Temp);
For i := 0 To (Temp.Count - 1) Do
If (Length(Temp.Strings[i]) >= 3)
    And (UpperCase(Copy(Temp.Strings[i],1,3)) = 'RUN')
Then Begin
If Flag = 0
Then Begin
LOCAL_MACHINE_NUM :=
Keys.Items.Add(Keys.TopItem,'HKEY_LOCAL_MACHINE').AbsoluteIndex;
Keys.Items.AddChild(Keys.Items.Item[LOCAL_MACHINE_NUM],Temp[i]);
Inc(Flag);
End
Else Begin
Keys.Items.AddChild(Keys.Items.Item[LOCAL_MACHINE_NUM],Temp[i]);
End;
End;
Temp.Free;
End;
Reg.Free;
PreparingButtons(False);
end;

// реалізація процедури класу TForm_1
procedure TForm_1.FormCreate(Sender: TObject);
var // змінні
Reg : TRegistry;
begin
Reg := TRegistry.Create;
Reg.RootKey := HKEY_CURRENT_USER;
If Reg.OpenKey('SoftWare\RegistryStartup', False)
Then Try Begin
Form1.Left := Reg.ReadInteger('Left');
Form1.Top  := Reg.ReadInteger('Top');
SaveFile.InitialDir := Reg.ReadString('SaveDialogDir');
End Except End;
Reload;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.KeysChange(Sender: TObject; Node: TTreeNode);
begin
LoadValues;
end;
```

```
// реалізація процедури класу TForm_1
procedure TForm_1.But_ExitClick(Sender: TObject);
begin
Close;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.RunProgramsClick(Sender: TObject);
begin
If (RunPrograms.Items.Count > 0) And (RunPrograms.ItemIndex >= 0)
Then Begin
PreparingButtons(True);
End;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.But_DeleteClick(Sender: TObject);
begin
If Application.MessageBox('Видалити ПЗ з автозапуску?','Removing
Confirmation',MB_OKCANCEL + MB_DEFBUTTON2) = IDOK
Then Begin
DeleteItem(GetKey,'Software\Microsoft\Windows\CurrentVersion\' +
Keys.Selected.Text,RunPrograms.Items[RunPrograms.ItemIndex]);
LoadValues;
End;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.But_SaveClick(Sender: TObject);
begin
SaveFile.Execute;
If SaveFile.FileName <> ''
Then Begin
If UpperCase(Copy(SaveFile.FileName,Length(SaveFile.FileName) - 3,4)) <> '.REG'
Then SaveFile.FileName := SaveFile.FileName + '.reg';
If not FileExists(SaveFile.FileName)
Then Begin
WinExec(PChar('regedit /ea "' + SaveFile.FileName + '" "'
        + GetKeySt + GetWay + '"'),SW_HIDE);
End
Else Begin
WinExec(PChar('regedit /ea "' + SaveFile.FileName + '" "'
       + GetKeySt + GetWay + '"'),SW_HIDE);
End
Else Exit;
End;
End
Else Begin
End;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.CopyrightClick(Sender: TObject);
begin
ShellExecute(Handle,'open',PChar('mailto: KovalenkoEO@mail.ua>'),nil,nil,0);
end;

// реалізація процедури класу TForm_1
procedure TForm_1.But_AddClick(Sender: TObject);
begin
Form2.ExecName.Text := 'консольне вікно';
Form2.CmdLine.Text := 'command.com';
Form2.ShowModal;
end;

// реалізація процедури класу TForm_1
procedure TForm_1.FormClose(Sender: TObject; var Action: TCloseAction);
var // змінні
Reg : TRegistry;
```

```
begin
Reg := TRegistry.Create;
Reg.RootKey := HKEY_CURRENT_USER;
If Reg.OpenKey('SoftWare\RegistryStartup', True)
Then Begin
Reg.WriteInteger('Left',Form1.Left);
Reg.WriteInteger('Top',Form1.Top);
If Form1.SaveFile.FileName <> '' Then
Reg.WriteString('SaveDialogDir',ExtractFilePath(Form1.SaveFile.FileName));
If Form2.AddFile.FileName <> '' Then
Reg.WriteString('OpenDialogDir',ExtractFilePath(Form2.AddFile.FileName));
End;
Form2.AddFile.Free;
Form1.SaveFile.Free;
end;

end.
```

ФАЙЛ ПРОЕКТУ ПРОГРАМИ ANTI_STEATH-VIRUS.DPR (ГЛОВНИЙ ФАЙЛ ПРОЕКТУ)

```
program ANTI_STEATH-VIRUS;
{ ---------------------------------------------------------------------------
  Центральноукраїнський національний технічний університет
  Програмне забезпечення системи кібербезпеки
  ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів
  Розробив: студент Кузьмін Кирило Костянтинович, КБ-21-3СК
---------------------------------------------------------------------------}
uses
  Forms,
  windows,
  VCLUtils,
  MainF in 'MainF.pas' {Form1},
  PAS in 'PAS.pas' {Form2},
  M_BackupMachine in 'M_BackupMachine.pas' {Form3},
  M_Dead_Uninstall_Killer in 'M_Dead_Uninstall_Killer.pas ' {Form4},
  M_RegistryStartup in 'M_RegistryStartup.pas ' {Form5},
  M_Scanner in 'M_Scanner.pas ' {Form6},
  Abox in 'Abox.pas' {AboutBox};

{$R *.res} // Ресурси які використовуються

Begin
// 1 крок - ініціалізація
  Application.Initialize;
// 2 крок – приховання основної форми
  Application.ShowMainForm:=false;
// 3 крок – створення 1 форми ПЗ
  Application.CreateForm(TForm1, Form1);
// 4 крок – створення 2 форми ПЗ
  Application.CreateForm(TForm2, Form2);
// 5 крок – створення 3 форми ПЗ
  Application.CreateForm(TForm2, Form3);
// 6 крок – створення 4 форми ПЗ
  Application.CreateForm(TForm2, Form4);
// 7 крок – створення 5 форми ПЗ
  Application.CreateForm(TForm2, Form5);
// 8 крок – створення 6 форми ПЗ
  Application.CreateForm(TForm2, Form6);
// 9 крок – створення 7 форми ПЗ
  Application.CreateForm(TAboutBox, AboutBox);
// Приховання форми
  ShowWindow(Application.Handle, SW_HIDE);
// Запуск та передача фокусу ПЗ
  Application.Run;
end.
```

ФАЙЛ DATA_2.PAS (МОДУЛЬ ДРУГОЇ ФОРМИ СКАНУВАННЯ)

```
unit DATA_2;

{ ------------------------------------------------------------------------------
  Центральноукраїнський національний технічний університет
  Програмне забезпечення системи кібербезпеки
  ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів
  Розробив: студент Кузьмін Кирило Костянтинович, КБ-21-3СК
------------------------------------------------------------------------------}

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Buttons, Unit1, Registry;

type
TForm2 = class(TForm) // клас який розроблено
ExecName: TEdit;
CmdLine: TEdit;
Label1: TLabel;
Label2: TLabel;
BitBtn1: TBitBtn;
Ok_F2B: TBitBtn;
Cancel_F2B: TBitBtn;
AddFile: TOpenDialog;
procedure BitBtn1Click(Sender: TObject);
procedure Ok_F2BClick(Sender: TObject);
procedure ExecNameChange(Sender: TObject);
procedure CmdLineChange(Sender: TObject);
procedure Cancel_F2BClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
//зона бачення змінних
private
procedure OkCheck;
public
{ Public declarations }
end;

var // змінні
Form2: TForm2; // створення екземпляру класу

implementation

{$R *.dfm}

// реалізація процедури класу TForm2
procedure TForm2.OkCheck;
begin
If ((ExecName.Text = '') or (CmdLine.Text = ''))
Then Ok_F2B.Enabled := False
Else Ok_F2B.Enabled := True;
end;

// реалізація процедури класу TForm2
procedure TForm2.BitBtn1Click(Sender: TObject);
begin
AddFile.FileName := '';
AddFile.Execute;
If AddFile.FileName <> ''
Then Begin
If ((ExecName.Text = '') or (ExecName.Text = 'Консольне вікно')) Then
ExecName.Text := ExtractFileName(AddFile.FileName);
CmdLine.Text := AddFile.FileName
End;
end;
```

```
// реалізація процедури класу TForm2
procedure TForm2.Ok_F2BClick(Sender: TObject);
var // змінні
Reg : TRegistry;
begin
Reg := TRegistry.Create;
Reg.RootKey := Form1.GetKey;
Reg.OpenKey(Form1.GetWay,False);

If AddFile.FileName <> ''
Then Begin
Reg.WriteString(ExtractFileName(AddFile.FileName),AddFile.FileName);
Form1.ShowPrograms(Form1.GetKey,Form1.GetWay);
End;
end;

// реалізація процедури класу TForm2
procedure TForm2.ExecNameChange(Sender: TObject);
begin
OkCheck;
end;

// реалізація процедури класу TForm2
procedure TForm2.CmdLineChange(Sender: TObject);
begin
OkCheck;
end;

// реалізація процедури класу TForm2
procedure TForm2.Cancel_F2BClick(Sender: TObject);
begin
Form2.Close;
end;

// реалізація процедури класу TForm2
procedure TForm2.FormCreate(Sender: TObject);
var // змінні
Reg : TRegistry;
begin
Reg := TRegistry.Create;
Reg.RootKey := HKEY_CURRENT_USER;
If Reg.OpenKey('SoftWare\Jerusalem\RegistryStartup', False)
Then Try
AddFile.InitialDir := Reg.ReadString('OpenDialogDir');
Except End;
end;

end.
```

ФАЙЛ MAINFORM.PAS (ГОЛОВНА ФОРМА ПРОГРАМИ)

```
unit MainForm;

{ ----------------------------------------------------------------------------
  Центральноукраїнський національний технічний університет
  Програмне забезпечення системи кібербезпеки
  ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів
  Розробив: студент Кузьмін Кирило Костянтинович, КБ-21-3СК
----------------------------------------------------------------------------}

interface

uses
SysUtils, Forms, StdCtrls, Windows, Graphics, Dialogs, Controls, Classes,
Buttons, AppEvnts, RXShell, Menus, ExtCtrls;


type
TForm1 = class(TForm) // клас який розроблено
RxTrayIcon1: TRxTrayIcon;
PopupMenu1: TPopupMenu;
N1: TMenuItem;
N2: TMenuItem;
N3: TMenuItem;
Timer1: TTimer;
Panel1: TPanel;
Image1: TImage;
GroupBox1: TGroupBox;
BitBtn1: TBitBtn;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
BitBtn4: TBitBtn;
BitBtn5: TBitBtn;
BitBtn6: TBitBtn;
procedure FormCreate(Sender: TObject);
procedure N3Click(Sender: TObject);
procedure RxTrayIcon1DblClick(Sender: TObject);
procedure N2Click(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Timer1Timer(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
private
procedure ApplicationMinimize(Sender : TObject);
procedure ApplicationRestore(Sender : TObject);
public
//зона бачення змінних
end;

var // змінні
Form1: TForm1; // створення екземпляру класу
implementation

uses PAS, Abox;

{$R *.dfm}

function Crypt(Text,Key: String; Encode: boolean): String;
var // змінні
i, KeyLength: integer;
Sign: ShortInt;
begin
KeyLength:=Length(Key);
if Encode then Sign :=-1 else Sign:=1;
```

```
for i:=1 to Length(Text) do
Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
Result:=Text;
end;

// реалізація процедури класу TForm1
procedure TForm1.ApplicationMinimize(Sender: TObject);
begin
ShowWindow(Application.Handle, SW_HIDE);
end;

// реалізація процедури класу TForm1
procedure TForm1.ApplicationRestore(Sender: TObject);
begin
ShowWindow(Application.Handle, SW_HIDE);
end;

// реалізація процедури класу TForm1
procedure TForm1.FormCreate(Sender: TObject);
begin
Application.OnMinimize := ApplicationMinimize;
Application.OnRestore := ApplicationRestore;
Timer1.Enabled:=true;
end;

// реалізація процедури класу TForm1
procedure TForm1.N3Click(Sender: TObject);
begin
RxTrayIcon1DblClick(Sender);
end;

// реалізація процедури класу TForm1
procedure TForm1.RxTrayIcon1DblClick(Sender: TObject);
begin
if (IsWindowVisible(Form1.Handle)=false) then
begin
Form1.show;
Form1.BringToFront;
end
else
begin
Form1.hide;
end;
end;

// реалізація процедури класу TForm1
procedure TForm1.N2Click(Sender: TObject);
begin
Form1.Close;
end;

// реалізація процедури класу TForm1
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
if MessageDlg('Exit?',mtConfirmation,[mbYes,mbNo],0) = mrYes then
begin
Action:=caFree;
end
else begin
Form1.Repaint;
Action:=caNone;
Form1.Hide;
end;
end;

// реалізація процедури класу TForm1
procedure TForm1.Timer1Timer(Sender: TObject);
begin
Form2.show;
```

```
Timer1.Enabled:=false;
end;

// реалізація процедури класу TForm1
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
WinExec('M_1.exe',SW_SHOW);
end;

// реалізація процедури класу TForm1
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
WinExec('M_2.exe',SW_SHOW);
end;

// реалізація процедури класу TForm1
procedure TForm1.BitBtn3Click(Sender: TObject);
begin
WinExec('M_3.exe',SW_SHOW);
end;

// реалізація процедури класу TForm1
procedure TForm1.BitBtn4Click(Sender: TObject);
begin
WinExec('M_4.exe',SW_SHOW);
end;

// реалізація процедури класу TForm1
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
AboutBox.show;
Form1.Hide;
end;

// реалізація процедури класу TForm1
procedure TForm1.BitBtn6Click(Sender: TObject);
begin
Form1.close;
end;

initialization
ShowWindow(Application.Handle, SW_HIDE);
end.
```

ФАЙЛ RESTORE_SCANNER.PAS (СКАНУВАННЯ ДИСКОВОЇ ПІДСИСТЕМИ)

```
unit Restore_scanner;
{ ----------------------------------------------------------------------------
  Центральноукраїнський національний технічний університет
  Програмне забезпечення системи кібербезпеки
  ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів
  Розробив: студент Кузьмін Кирило Костянтинович, КБ-21-3СК
----------------------------------------------------------------------------}

interface

uses
scan, rescan,

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
ToolWin, Menus, ExtCtrls, ComCtrls, StdCtrls;

Const // визначення констант що використовуються

Device: string= '0000000000000000000000000000';
WM_SCANPROGRESS=WM_USER; //об'ява сповіщень
WM_CHECKSCANEND=WM_USER+1;
WM_RESCANEND=WM_USER+2;

type
TMainForm = class(TForm) // клас який розроблено
Label0: TLabel;
Label1: TLabel;
Label2: TLabel;
Label4: TLabel;
Label7: TLabel;
Bevel2: TBevel;
Bevel3: TBevel;
State1: TProgressBar;
Label5: TLabel;
ToolBar4: TToolBar;
ToolBar5: TToolBar;
Summary: TToolButton;
PopupMenu1: TPopupMenu;
Zoom1: TMenuItem;
Rescan1: TMenuItem;
Mark: TMenuItem;
UpBtn: TButton;
BackBtn: TButton;
UpdateBtn: TButton;
procedure FormPaint(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure FormDestroy(Sender: TObject);
procedure DriveClick(Sender: TObject);
procedure BackBtnClick(Sender: TObject);
procedure UpdateBtnClick(Sender: TObject);
procedure Zoom1Click(Sender: TObject);
procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure Rescan1Click(Sender: TObject);
procedure UpBtnClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure MarkClick(Sender: TObject);
procedure FormResize(Sender: TObject);
private
offscreen: tbitmap;
starting, scanning, rescanning, havedata: boolean;
procedure StartScan;
procedure mainpaint;
procedure setmaxlevel(d: integer; pre: pdescriptor; level: integer);
procedure StopThreads;
```

```
procedure freethreads;
procedure setun;
function uncalc(x: int64): string;
procedure SetSize;
procedure Morphe;
procedure StartRescan(tx: pdescriptor; dtx: integer; txdir: string);
procedure ZoomTo(tx: pdescriptor; dtx: integer; txdir: string);
procedure OnCheckScanEnd(var m: TMessage); Message WM_CHECKSCANEND;
procedure OnRescanEnd(var m: TMessage); Message WM_RESCANEND;
procedure OnScanProgress(var m: TMessage); Message WM_SCANPROGRESS;
end;

var // змінні
MainForm: TMainForm;
visual: set of 0..25;
t: array[0..25] of tscanthread;
sizecluster: array[0..25] of cardinal;
rt: trescanthread;
threadspace, threadscanned: array[0..7] of int64;
state1pos: array[0..7] of integer;

implementation

uses registry;

{$R *.DFM}

Const // визначення констант що використовуються
maxhist=16;
minspace=1.0;
unname: array[0..3] of string=('','k','M','G');
lang:array[0..24] of string=('Сканування...','Bytes','Файлів','Всього','Used
on', 'Free on','У сумі','Recycle Bin','','','','Up','Back','Rescan','',
'Zoom','Rescan','','','','','','','Show','Приховати');

var // змінні
radius, rr, x0, y0, threshold: integer;
drive,nd, un, maxlevel, dfoc, dzoom, histpos: integer;
clBack: TColor;
zoomspace, totalspace: int64;
freecluster, totalcluster: array[0..25] of cardinal;
drivetype: array[0..25] of integer;
start, bytesavailable, bytestotal: array[0..25] of int64;
foc, zoom, prezoom: pdescriptor;
hist: array [0..maxhist-1] of record
name: string;
foc: pdescriptor;
dfoc: integer
end;
focname, zoomname, prename: string;
whitefont, bigfont: tfont;

procedure TMainForm.FormCreate(Sender: TObject);
var // змінні
d: integer;
s: string;
tb: ttoolbutton;
begin
whitefont:=TFont.Create;
whitefont.assign(font);
whitefont.color:=$FFFFFF;
bigfont:=TFont.Create;
bigfont.name:='Times New Roman';
bigfont.Size:=Round(33*72/Font.PixelsPerInch);
bigfont.Style:=[fsBold];
bigfont.color:=$FFFFFF;

label5.caption:=lang[0];
summary.caption:=lang[6];
```

```
UpBtn.Caption:=lang[11];
BackBtn.Caption:=lang[12];
UpdateBtn.Caption:=lang[13];
Zoom1.Caption:=lang[15];
Rescan1.Caption:=lang[16];
un:=0;

offscreen:=tbitmap.create;
clBack:=clBtnFace;

drive:=-1;
for d:=25 downto 0 do
begin
s:=char(d+65)+':\';
drivetype[d]:=GetDriveType(PChar(s));
if drivetype[d] in [DRIVE_FIXED,DRIVE_REMOVABLE,DRIVE_CDROM,DRIVE_REMOTE] then
begin
tb:=ttoolbutton.create(self);
tb.caption:=char(d+65)+':';
tb.tag:=16+d;
tb.wrap:=true;
tb.autosize:=true;
tb.OnClick:=DriveClick;
tb.imageindex:=-1;
tb.parent:=toolbar4;
end;
t[d]:=nil;
end;

SetSize;

havedata:=false;
scanning:=false;
rescanning:=false;
starting:=true;
end;

procedure TMainForm.FormDestroy(Sender: TObject);
begin
freethreads;
offscreen.free;
whitefont.free; bigfont.free;
end;

procedure TMainForm.SetSize;
begin
radius:=(ClientHeight-37) div 2;
if (clientwidth-200) div 2<radius then radius:=(clientwidth-200) div 2;
rr:=radius-10; {Radius }
threshold:=5500 div (radius-42);
offscreen.width:=2*radius+3; offscreen.height:=3*(radius*2 div 3+1)+1;

x0:=clientwidth div 2-radius; y0:=29; {Позиція }
label0.width:=clientwidth-16;
UpdateBtn.left:=clientwidth-88;
UpBtn.left:=clientwidth-88;
BackBtn.left:=clientwidth-88;
label5.left:=(clientwidth-label5.width) div 2;
label5.top:=radius-4;
State1.left:=(clientwidth-State1.width) div 2;
State1.top:=radius+17;
toolbar4.height:=clientheight-121;
toolbar5.top:=clientheight-38;
end;

procedure freetree(pre: pdescriptor);
var // змінні
dx,rem: pdescriptor;
begin
```

```
dx:=pre.sub;
while dx<>nil do
begin
if dx.nf>=0 then freetree(dx);
rem:=dx;
dx:=dx.next;
freemem(rem,16+length(rem.name)+1);
end;
end;

procedure TMainForm.StopThreads;
var // змінні
d: integer;
begin
if scanning then
begin
havedata:=false;
if rescanning then
begin
rt.terminate;
rt.waitfor
end
else
begin
For d:=0 To 25 do if (d in visual) and not t[d].done then t[d].Terminate;
For d:=0 To 25 do if (d in visual) then t[d].WaitFor
end
end;
end;

procedure tmainform.freethreads;
var // змінні
d: integer;
begin
For d:=0 To 25 do if t[d]<>nil then
begin freetree(t[d].tree); freemem(t[d].tree,16+3); t[d].free; t[d]:=nil end;
end;

procedure TMainForm.FormPaint(Sender: TObject);
var // змінні
i: integer;
s: string;
begin
if starting then
begin starting:=false; StartScan; end;

if scanning or not havedata then with canvas do
begin
brush.color:=clBtnFace;
fillrect(rect(0, 0, clientwidth, clientheight));
end
else
begin
bitblt(canvas.handle, x0, y0, radius*2, radius*2, offscreen.canvas.handle, 0, 0,
SRCCOPY);
if zoomspace>0 then with canvas do
begin
brush.style:=bsclear;
s:=uncalc(zoomspace);
font.assign(bigfont);
i:=textheight(s);
textout(x0+radius-textwidth(s) div 2, y0+radius-i div 2, s);
font.assign(whitefont);
if zoom=nil then s:=lang[3] else s:=zoom.name;
textout(x0+radius-textwidth(s) div 2, y0+radius-i div 2-12, s);
s:=unname[un]+lang[1];
textout(x0+radius-textwidth(s) div 2, y0+radius+i div 2-2, s);
end;
end
```

```
end;

procedure tmainform.setun;
begin
un:=0;
if zoomspace=0 then
begin label4.caption:=''; label7.caption:='' end
else
begin
while(zoomspace*threshold div 10000+1 shl((un+1)*10-1)) shr((un+1)*10)>0 do
inc(un);
label4.caption:=unname[un]+lang[1];
label7.caption:=lang[2];
end
end;

function tmainform.uncalc;
begin
if un=0 then result:=inttostr(x)
else result:=inttostr((x+1 shl(un*10-1)) shr(un*10));
if length(result)>3 then insert(',',result,length(result)-2);
end;

procedure tmainform.setmaxlevel(d: integer; pre: pdescriptor; level: integer);
var // змінні
dx: pdescriptor;
begin
dx:=pre.sub;
if not pre.mark then
while dx<>nil do
begin
if (dx.nc*10000.0>=threshold/sizecluster[d]*zoomspace)
and (level>maxlevel) then maxlevel:=level;
if (dx.nf>=0) and (dx<>t[d].recycled) then setmaxlevel(d, dx, level+1);
dx:=dx.next
end
end;

procedure TMainForm.Morphe;
begin
UpdateBtn.visible:=not scanning;
UpBtn.visible:=not scanning and (zoom<>nil);
BackBtn.visible:=not scanning and (histpos>0);
label5.visible:=scanning;
state1.visible:=scanning;
end;



procedure tmainform.mainpaint;
Const // визначення констант що використовуються

vector: array[0..2,0..2] of byte=((0,$80,0),($40,$40,$80),($80,$30,$30));
var // змінні
i, r, d: integer;
a0, a1: single;
lasta1:array[0..63] of single;

procedure sector (ca: tcanvas; x,y,r: integer; a0,a1: single);
begin
if (round(r*sin(a1))=round(r*sin(a0)))
and (round(r*cos(a1))=round(r*cos(a0))) then
if abs(a1-a0)>3.14 then ca.Ellipse(x-r, y-r, x+r, x+r)
else else ca.pie(x-r,y-r,x+r,y+r,x+round(r*sin(a1))-1,y-round(r*cos(a1)),
x+round(r*sin(a0))-1,y-round(r*cos(a0)))
end;

procedure gosub(pre: pdescriptor; level, cl0, icl: integer; a00: single);
var // змінні
```

```
cl, nc0: integer;
a0, a1: single;
dx: pdescriptor;
begin
dx:=pre.sub;
nc0:=0;
if not pre.mark then
while dx<>nil do
begin
a0:=a00+2*pi*nc0/zoomspace*sizecluster[d];
inc(nc0, dx.nc);
a1:=a00+2*pi*nc0/zoomspace*sizecluster[d];
if dx.nc*10000.0>=threshold/sizecluster[d]*zoomspace then
begin
if dx.mark then begin cl:=$E0E0E0; inc(icl) end
else if dx=t[d].recycled then cl:=$00FFFF
else
begin
cl:=cl0+vector[icl mod 3,0] shr(level-2) shl 0
+vector[icl mod 3,1] shr(level-2) shl 8
+vector[icl mod 3,2] shr(level-2) shl 16;
if dx.nf>=0 then gosub(dx, level+1, cl, icl, a0);
inc(icl)
end;
if (a0-lasta1[level])*sqrt((level-1)/maxlevel)*rr<minspace then
a0:=lasta1[level];
lasta1[level]:=a1;
offscreen.canvas.brush.color:=cl;
sector (offscreen.canvas,radius,radius,round(rr*sqrt(level/maxlevel)),a0,a1)
end;
dx:=dx.next
end
end;

begin
r:=round(rr/sqrt(maxlevel));
with offscreen.canvas do
begin
brush.style:=bssolid;
brush.color:=clBack;
fillrect(rect(0, 0, 2*radius+3, 2*radius+3));
if zoomspace=0 then exit;
pen.color:=$000000;
pen.width:=1;
Brush.color:=$FFFFFF;
Ellipse(0, 0, 2*radius, 2*radius);
end;

for i:=0 to 63 do lasta1[i]:=-1;

if zoom<>nil then
begin
d:=dzoom;
gosub(zoom, 2, 0, 0, 0);
with offscreen.canvas do
begin
pen.color:=$000000;
Brush.color:=$907070;
Ellipse(radius-r, radius-r, radius+r, radius+r);
end;
end
else
begin
for d:=0 to 25 do if d in visual then
gosub(t[d].tree, 2, 0, 0, 2*pi*start[d]/zoomspace);
with offscreen.canvas do
begin
pen.width:=2;
Brush.color:=$D0B0D0;
```

```
Ellipse(radius-r, radius-r, radius+r, radius+r);
pen.width:=1;
end;

for d:=0 to 25 do if d in visual then
begin
a0:=2*pi*start[d]/zoomspace;
a1:=a0+2*pi*t[d].tree.nc/zoomspace*sizecluster[d];
with offscreen.canvas do
begin
Brush.color:=$907070;
sector (offscreen.canvas,radius,radius,r-1,a0,a1);
end
end;

if nd>1 then for d:=0 to 25 do if d in visual then
begin
a0:=2*pi*start[d]/zoomspace;
with offscreen.canvas do
begin
moveto(radius-1, radius-1);
lineto(round(radius+r*sin(a0))-1, round(radius-r*cos(a0))-1);
end
end;
end;
end;

procedure TMainForm.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
var // змінні
ax, a0, a1, a2: single;
dselected, d, lx: integer;
selected: pdescriptor;
s,loc: string;

procedure gosub(pre: pdescriptor; level: integer; a00: single);
var // змінні
nc0: integer;
a0, a1: single;
dx: pdescriptor;
begin
nc0:=0;
dx:=pre.sub;
if not pre.mark then
while dx<>nil do
begin
a0:=a00+2*pi*nc0/zoomspace*sizecluster[d];
inc(nc0, dx.nc);
a1:=a00+2*pi*nc0/zoomspace*sizecluster[d];
if dx.nc*10000.0>=threshold/sizecluster[d]*zoomspace then
begin
if (ax>=a0) and (ax<a1) then
begin
if level=lx then begin selected:=dx; loc:=dx.name end
else if (dx.nf>=0) and (dx<>t[d].recycled) then
begin
gosub(dx, level+1, a0);
if selected<>nil then loc:=dx.name+'\'+loc
end;
exit
end;
end;
dx:=dx.next
end
end;

begin
if scanning or not havedata then
begin
```

```
label0.caption:='';
exit;
end;
if nd=0 then
begin label0.caption:=''; label1.caption:=''; label2.caption:=''; exit; end;
if y-y0=radius then if x-x0<radius then ax:=3/2*pi else ax:=pi/2
else
begin
ax:=arctan((x-x0-radius)/(radius-y+y0));
if y-y0>radius then ax:=ax+pi else if x-x0<radius then ax:=ax+2*pi
end;
lx:=trunc((sqr(x-x0-radius)+sqr(y-y0-radius))/sqr(rr)*maxlevel)+1;
loc:='';
selected:=nil;
dselected:=-1;
if zoom<>nil then
if lx=1 then
begin
loc:=zoomname;
selected:=zoom;
dselected:=dzoom;
foc:=nil;
end
else
begin
d:=dzoom;
gosub(zoom, 2, 0);
if selected<>nil then
begin dselected:=dzoom; loc:=zoomname+'\'+loc end;
foc:=selected; dfoc:=dselected;
focname:=loc;
end
else
if lx=1 then
begin
for d:=0 to 25 do if d in visual then
begin
a0:=2*pi*start[d]/zoomspace;
a1:=a0+2*pi*t[d].tree.nc/zoomspace*sizecluster[d];
a2:=a0+2*pi*totalcluster[d]/zoomspace*sizecluster[d];
if ax>=a0 then
if ax<a1 then
begin
selected:=t[d].tree;
loc:=lang[4]+' '+t[d].tree.name;
dselected:=d
end
else if ax<a2 then
begin loc:=lang[5]+' '+t[d].tree.name; dselected:=d end
end;
foc:=nil
end
else
begin
for d:=0 to 25 do if d in visual then
if selected=nil then
begin
gosub(t[d].tree, 2, 2*pi*start[d]/zoomspace);
if selected<>nil then dselected:=d
end;
if selected<>nil then loc:=t[dselected].tree.name+'\'+loc;
foc:=selected; dfoc:=dselected;
focname:=loc;
end;
if (dselected>=0) and (selected=t[dselected].recycled) then loc:=lang[7];
if canvas.TextWidth(loc)>label0.width then
begin
insert('...',loc,4);
while canvas.TextWidth(loc)>label0.width do delete(loc,7,1);
```

```
end;
label0.caption:=loc;
if dselected>=0 then
if selected<>nil then
begin
if selected.nf>=0 then
begin
s:=inttostr(selected.nf);
if length(s)>3 then insert(',',s,length(s)-2);
label1.caption:=s
end
else label1.caption:='';
label2.caption:=uncalc(selected.nc*int64(sizecluster[dselected]));
end
else
begin
label1.caption:='';
label2.caption:=uncalc((totalcluster[dselected]-t[dselected].tree.nc)
*int64(sizecluster[dselected]));
end
else begin label1.caption:=''; label2.caption:='' end;
end;

procedure TMainForm.DriveClick(Sender: TObject);
begin
StopThreads;
drive:=tcomponent(sender).tag-16;
StartScan;
end;

procedure searchprezoom(pre: pdescriptor);
var // змінні
dx: pdescriptor;
begin
dx:=pre.sub;
while dx<>nil do
begin
if dx=zoom then begin prezoom:=pre; prename:='' end;
if prezoom=nil then
begin
if dx.nf>=0 then searchprezoom(dx);
if prezoom<>nil then prename:=dx.name+'\'+prename
end;
dx:=dx.next
end
end;

procedure TMainForm.BackBtnClick(Sender: TObject);
var // змінні
d: integer;
begin
screen.cursor:=crHourGlass;

dec(histpos);
with hist[histpos] do
begin
zoom:=foc;
if zoom=nil then
begin
zoomspace:=totalspace;
setun;
maxlevel:=4;
For d:=0 To 25 do if d in visual then setmaxlevel(d,t[d].tree,2);
end
else
begin
dzoom:=dfoc;
zoom:=foc;
zoomspace:=zoom.nc*int64(sizecluster[dzoom]);
```

```
setun;
zoomname:=name;
maxlevel:=4;
setmaxlevel(dzoom,zoom,2);
prezoom:=nil;
searchprezoom(t[dzoom].tree);
prename:=char(65+dzoom)+':\'+copy(prename,1,length(prename)-1);
end;
end;
mainpaint;
if histpos=0 then BackBtn.Visible:=false;
UpBtn.Visible:= zoom<>nil;
screen.cursor:=crDefault;
Invalidate;
end;

procedure TMainForm.UpdateBtnClick(Sender: TObject);
begin
if zoom=nil then StartScan
else StartRescan(zoom,dzoom,zoomname);
end;

procedure TMainForm.Zoom1Click(Sender: TObject);
begin
ZoomTo(foc,dfoc,focname);
end;

procedure TMainForm.ZoomTo(tx: pdescriptor; dtx: integer; txdir: string);
var // змінні
i,d: integer;
begin
screen.cursor:=crHourGlass;
if histpos=maxhist then
begin
for i:=0 to maxhist-2 do hist[i]:=hist[i+1];
histpos:=maxhist-1
end;
hist[histpos].name:=zoomname;
hist[histpos].foc:=zoom;
hist[histpos].dfoc:=dzoom;
inc(histpos);
if tx=t[dtx].tree then
begin
zoom:=nil;
zoomspace:=totalspace;
setun;
maxlevel:=4;
For d:=0 To 25 do if d in visual then setmaxlevel(d,t[d].tree,2);
end
else
begin
dzoom:=dtx;
zoom:=tx;
zoomspace:=zoom.nc*int64(sizecluster[dzoom]);
setun;
zoomname:=txdir;
maxlevel:=4;
setmaxlevel(dzoom,zoom,2);
prezoom:=nil;
searchprezoom(t[dzoom].tree);
prename:=char(65+dzoom)+':\'+copy(prename,1,length(prename)-1);
end;
mainpaint;
BackBtn.Visible:=true;
UpBtn.Visible:= zoom<>nil;
screen.cursor:=crDefault;
invalidate
end;
```

```
procedure TMainForm.FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
if scanning or not havedata then exit;
FormMouseMove(Sender,Shift,X,Y);
if (foc<>nil) and (foc<>t[dfoc].tree) then
if (button=mbleft) and (foc.nf>=0) and not foc.mark and (foc<>t[dfoc].recycled)
then
ZoomTo(foc,dfoc,focname)
else if button=mbright then
begin
if foc.mark then Mark.Caption:=lang[23]
else Mark.caption:=lang[24];
Mark.Visible:=(foc<>t[dfoc].recycled);
Zoom1.Visible:=(foc.nf>=0) and not foc.mark and (foc<>t[dfoc].recycled);
Rescan1.Visible:=(foc.nf>=0);
popupmenu1.Popup(Left+2+x,Top+2+getsystemmetrics(SM_CYCAPTION)+y);
end
end;

procedure TMainForm.Rescan1Click(Sender: TObject);
begin
StartRescan(foc,dfoc,focname);
end;

procedure TMainForm.UpBtnClick(Sender: TObject);
begin
ZoomTo(prezoom,dzoom,prename);
end;

procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
StopThreads
end;

procedure TMainForm.StartScan;
var // змінні
s: string;
no, d: integer;
bytesfree: TLargeInteger;
sectpcluster, bytespsect: cardinal;
ex: boolean;

begin
freethreads;
totalspace:=0;
nd:=0;
fillchar(totalcluster, sizeof(totalcluster), 0);

for no:=0 to 7 do threadspace[no]:=0;
if drive=-1 then
begin
visual:=[];
for d:=0 to 25 do if drivetype[d]=drive_fixed then
include(visual, d)
end
else visual:=[drive];
for d:=0 to 25 do if d in visual then
begin
s:=char(d+65)+':\';
if GetDiskFreeSpace(PChar(s), sectpcluster, bytespsect, freecluster[d],
totalcluster[d]) then
begin
inc(nd);
sizecluster[d]:=sectpcluster*bytespsect;
t[d]:=tscanthread.create(d);
if drivetype[d]=drive_cdrom then
begin
t[d].usedspace:=int64(sizecluster[d])*totalcluster[d];
```

```
sizecluster[d]:=bytespsect;
end
else
begin
ex:=GetDiskFreeSpaceEx(PChar(s), bytesavailable[d], bytestotal[d], @bytesfree);
if ex then t[d].usedspace:=bytestotal[d]-bytesavailable[d]
else t[d].usedspace:=int64(sizecluster[d])*(totalcluster[d]-freecluster[d]);
end;
threadspace[ord(Device[d+1])-48]:=threadspace[ord(Device[d+1])-
48]+t[d].usedspace;
end
else exclude(visual,d)
end;

scanning:= nd>0;
rescanning:=false;
havedata:= scanning;

label0.caption:='';
state1.position:=0;
Morphe;
if scanning then
begin
for no:=0 to 7 do
begin
state1pos[no]:=maxint; threadscanned[no]:=0;
for d:=0 To 25 do if (d in visual) and (Device[d+1]=char(no+48)) then
begin state1pos[no]:=0; t[d].resume; break end;
end;
end
else
begin
histpos:=0;
BackBtn.Visible:=false;
UpBtn.Visible:=false;
UpdateBtn.visible:=false;
end;
invalidate
end;

procedure TMainForm.OnCheckScanEnd;
var // змінні
d: integer;

procedure setfilenc(pre: pdescriptor);
var // змінні
dx: pdescriptor;
begin
dx:=pre.sub;
while dx<>nil do
begin
if dx.nf>=0 then setfilenc(dx)
else dx.nc:=(dx.bytes+sizecluster[d]-1) div sizecluster[d];
inc(pre.nc,dx.nc);
dx:=dx.next
end
end;

begin
For d:=0 To 25 do if (d in visual) and not t[d].done then
exit;

if havedata then
begin
For d:=0 To 25 do if d in visual then
begin
start[d]:=totalspace;
totalspace:=totalspace+int64(sizecluster[d])*totalcluster[d];
setfilenc(t[d].tree);
```

```
if t[d].recycled<>@norec then
begin
dec(t[d].tree.nc,t[d].recycled.nc);
dec(t[d].tree.nf,t[d].recycled.nf+1);
inc(freecluster[d],t[d].recycled.nc);
end;
end;
state1.position:=128;

maxlevel:=4;
zoomspace:=totalspace;
setun;
For d:=0 To 25 do if d in visual then setmaxlevel(d,t[d].tree,2);
foc:=nil; zoom:=nil;
scanning:=false;
MainPaint;
end
else scanning:=false;

histpos:=0;
BackBtn.Visible:=false;
UpBtn.Visible:=false;
UpdateBtn.visible:=havedata;
Morphe;
invalidate
end;

procedure TMainForm.StartRescan(tx: pdescriptor; dtx: integer; txdir: string);
begin
scanning:=true;
rescanning:=true;
state1.position:=0;
label0.caption:='';
Morphe;
freetree(tx);
rt:=trescanthread.create(dtx,tx,txdir);
rt.resume;
invalidate;
end;

procedure TMainForm.OnRescanEnd;
var // змінні
d: integer;
begin
if havedata then
begin
state1.position:=128;
maxlevel:=4;
if zoom<>nil then
begin
zoomspace:=zoom.nc*int64(sizecluster[dzoom]);
setun;
setmaxlevel(dzoom,zoom,2);
end
else For d:=0 To 25 do if d in visual then setmaxlevel(d,t[d].tree,2);
end;
scanning:=false;
mainpaint;
if histpos>0 then
begin hist[0].foc:=nil; histpos:=1; end;
Morphe;
invalidate
end;

procedure TMainForm.OnScanProgress;
begin
MainForm.State1.position:=m.WParam;
end;
```

```
procedure TMainForm.FormResize(Sender: TObject);
var // змінні
d: integer;
resizechart: boolean;
begin
resizechart:=havedata;
if resizechart then
begin
havedata:=false;
invalidate;
update;
end;
SetSize;
if resizechart then
begin
maxlevel:=4;
if zoom=nil then
begin
For d:=0 To 25 do if d in visual then setmaxlevel(d,t[d].tree,2);
end
else setmaxlevel(dzoom,zoom,2);
havedata:=true;
MainPaint;
end;
Invalidate
end;

procedure TMainForm.MarkClick(Sender: TObject);
var // змінні
d: integer;
begin
screen.cursor:=crHourGlass;
foc.mark:=not foc.mark;
maxlevel:=4;
if zoom<>nil then setmaxlevel(dzoom,zoom,2)
else For d:=0 to 25 do if d in visual then setmaxlevel(d,t[d].tree,2);
mainpaint;
screen.cursor:=crDefault;
invalidate
end;
end.
```

ФАЙЛ DATA_UNINSTALL.PAS (МОДУЛЬ ФОРМИ ВИДАЛЕННЯ УШКОДЖЕНОГО ПЗ)

```
unit Data_uninstall;

{ ------------------------------------------------------------------------------
  Центральноукраїнський національний технічний університет
  Програмне забезпечення системи кібербезпеки
  ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів
  Розробив: студент Кузьмін Кирило Костянтинович, КБ-21-3СК
------------------------------------------------------------------------------}

interface

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

type
TForm_2 = class(TForm) // клас який розроблено
ListBox1: TListBox;
Button1: TButton;
ListBox2: TListBox;
Button2: TButton;
Button3: TButton;
Button4: TButton;
procedure FormCreate(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
//зона бачення змінних
private
public
end;

var // змінні
Form1: TForm_2; // створення екземпляру класу
Reg:TRegistry;

implementation

{$R *.DFM}

procedure TForm_2.FormCreate(Sender: TObject);
begin
listbox1.Items.Clear;
Reg := TRegistry.Create;
Reg.RootKey := HKEY_LOCAL_MACHINE;
if not Reg.OpenKey('\Software\Microsoft\Windows\CurrentVersion\Uninstall',True)
then
Reg.CreateKey('\Software\Microsoft\Windows\CurrentVersion\Security');
try
Reg.RootKey := HKEY_LOCAL_MACHINE;
if Reg.OpenKey('\Software\Microsoft\Windows\CurrentVersion\Uninstall',True)
then
begin
reg.GetKeyNames(listbox1.Items);
end
finally
Reg.CloseKey;
Reg.Free;
inherited;
end;
button1.Click;
end;

procedure TForm_2.Button1Click(Sender: TObject);
```

```
var // змінні
s:string;
i:integer;
begin
listbox2.Items.Clear;
for i:=0 to listbox1.items.count-1 do
begin
Reg := TRegistry.Create;
try
Reg.RootKey := HKEY_LOCAL_MACHINE;
if
Reg.OpenKey('\Software\Microsoft\Windows\CurrentVersion\Uninstall\'+listbox1.ite
            ms[i],True)
then begin
s:=reg.readstring('DisplayName');
if s<>'' then listbox2.items.add(s);
end;
finally
Reg.CloseKey;
Reg.Free;
inherited;
end;
end;
end;

procedure TForm_2.Button2Click(Sender: TObject);
var // змінні
s:string;
i:integer;
begin
for i:=0 to listbox1.items.count-1 do
begin
Reg := TRegistry.Create;
try
Reg.RootKey := HKEY_LOCAL_MACHINE;
Reg.OpenKey('\Software\Microsoft\Windows\CurrentVersion\Uninstall\'+listbox1.ite
            ms[i],True);
s:=reg.readstring('DisplayName');
if s=listbox2.items[listbox2.itemindex] then
if application.messagebox(pchar('Delete '+s),'Видалити? ',MB_YESNO)=IDYES then
reg.DeleteKey('\Software\Microsoft\Windows\CurrentVersion\Uninstall\'+
              listbox1.items[i]);
finally
Reg.CloseKey;
Reg.Free;
inherited;
end;
end;
listbox1.Items.Clear;
Reg := TRegistry.Create;
Reg.RootKey := HKEY_LOCAL_MACHINE;
if not Reg.OpenKey('\Software\Microsoft\Windows\CurrentVersion\Uninstall',True)
then
Reg.CreateKey('\Software\Microsoft\Windows\CurrentVersion\Security');
try
Reg.RootKey := HKEY_LOCAL_MACHINE;
if Reg.OpenKey('\Software\Microsoft\Windows\CurrentVersion\Uninstall',True)
then
begin
reg.GetKeyNames(listbox1.Items);
end
finally
Reg.CloseKey;
Reg.Free;
inherited;
end;
button1.Click;
end;
```

```
procedure TForm_2.Button4Click(Sender: TObject);
begin
close;
end;
end.
```

ФАЙЛ MOD_3.PAS (МОДУЛЬ ФОРМИ РЕЗЕРВНОГО КОПІЮВАННЯ ФАЙЛІВ ТА ПАПОК)

```
unit MOD_3;

{ ------------------------------------------------------------------------------
  Центральноукраїнський національний технічний університет
  Програмне забезпечення системи кібербезпеки
  ОС Windows 10/11 від дестабілізуючої дії стелс-вірусів
  Розробив: студент Кузьмін Кирило Костянтинович, КБ-21-3СК
------------------------------------------------------------------------------}

interface

uses FileCtrl, Windows, SysUtils, Messages, Classes, Graphics, Forms, Controls,
StdCtrls,
Buttons, ComCtrls, ExtCtrls, CheckLst, ImgList, Dialogs, inifiles, ShellAPI,
ShlObj,
AppEvnts;

Const // визначення констант що використовуються

Yes = 0;
No = 1;
keyTAB = #9;

DefaultDestinationsNumber = 5;
type
PShellObject = ^TShellObject;
TShellObject = Record
DoIt    : boolean; //операція
LiteralName   : string[50];
SourcePath : string[255]; //шлях звідкіля
Prefix  : string[20];
Sufix  : string[20];
DestinationFolder : string[255]; // шлях куди
AttachDateTime : boolean;
end;
TBackupFile = Record
GroupName : string[20]; // літеральне ім’я локального ПК
DoIt     : Boolean;
ShellObject   : TShellObject;
end;


type
TfrMain = class(TForm) // клас який розроблено
Panel1: TPanel;
Panel2: TPanel;
Page: TPageControl;
tsAbout: TTabSheet;
tsOdredista: TTabSheet;
btnExit: TBitBtn;
btnEngage: TBitBtn;
Panel3: TPanel;
memoO: TMemo;
Image1: TImage;
tree: TTreeView;
treeImages: TImageList;
sbtnCollapse: TSpeedButton;
sbtnExpand: TSpeedButton;
Label1: TLabel;
tsStatus: TTabSheet;
pbarGroups: TProgressBar;
cdMemo: TMemo;
pnlLed: TPanel;
ledStazaDo: TLabeledEdit;
btnNewGroup: TBitBtn;
btnDelGroup: TBitBtn;
```

```
btnNewObject: TBitBtn;
btnDelObject: TBitBtn;
ledPrefix: TLabeledEdit;
pbarObjects: TProgressBar;
stext: TStaticText;
Animate: TAnimate;
memoErrors: TMemo;
AppEvents: TApplicationEvents;
ledSufix: TLabeledEdit;
btnUpdateObject: TBitBtn;
ledDestFolder: TLabeledEdit;
ledLiteralName: TLabeledEdit;
lblItemSelected: TStaticText;
btnSelectObject: TSpeedButton;
btnSelectDest: TSpeedButton;
lblResultObject: TStaticText;
chkAttachDateTime: TCheckBox;
procedure btnExitClick(Sender: TObject);
procedure sbtnCollapseClick(Sender: TObject);
procedure sbtnExpandClick(Sender: TObject);
procedure treeDblClick(Sender: TObject);
procedure treeClick(Sender: TObject);
procedure btnDelGroupClick(Sender: TObject);
procedure btnNewGroupClick(Sender: TObject);
procedure btnDelObjectClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormCreate(Sender: TObject);
procedure treeChanging(Sender: TObject; Node: TTreeNode;
var AllowChange: Boolean);
procedure treeExit(Sender: TObject);
procedure btnEngageClick(Sender: TObject);
procedure treeChange(Sender: TObject; Node: TTreeNode);
procedure AppEventsActivate(Sender: TObject);
procedure btnUpdateObjectClick(Sender: TObject);
procedure btnNewObjectClick(Sender: TObject);
procedure btnSelectObjectClick(Sender: TObject);
procedure btnSelectDestClick(Sender: TObject);
private
fLastSelected:TTreeNode;

function CtrlDown : Boolean;

function DoTransfer(const sDateTime:string; const backupFile: TBackupFile) :
integer;

procedure UpdateChecked(Node: TTreeNode);
procedure SaveData;
procedure LoadData;

{ Private declarations }
public
//зона бачення змінних
end;


var // змінні
frMain: TfrMain;

implementation
{$R *.dfm}

function BrowseDialogCallBack
(Wnd: HWND; uMsg: UINT; lParam, lpData: LPARAM):
integer stdcall;
var // змінні
wa, rect : TRect;
dialogPT : TPoint;
begin
//початок
```

```
if uMsg = BFFM_INITIALIZED then
begin
wa := Screen.WorkAreaRect;
GetWindowRect(Wnd, Rect);
dialogPT.X := ((wa.Right-wa.Left) div 2) -
       ((rect.Right-rect.Left) div 2);
dialogPT.Y := ((wa.Bottom-wa.Top) div 2) -
       ((rect.Bottom-rect.Top) div 2);
MoveWindow(Wnd,
    dialogPT.X,
    dialogPT.Y,
    Rect.Right - Rect.Left,
    Rect.Bottom - Rect.Top,
    True);
end;

Result := 0;
end; // опис BrowseDialogCallBack

function BrowseDialog (const Title: string; const Flag: integer): string;
var // змінні
lpItemID : PItemIDList;
BrowseInfo : TBrowseInfo;
DisplayName : array[0..MAX_PATH] of char;
TempPath : array[0..MAX_PATH] of char;
begin
Result:='';
FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
with BrowseInfo do begin
hwndOwner := Application.Handle;
pszDisplayName := @DisplayName;
lpszTitle := PChar(Title);
ulFlags := Flag;
lpfn := BrowseDialogCallBack;
end;
lpItemID := SHBrowseForFolder(BrowseInfo);
if lpItemId <> nil then begin
SHGetPathFromIDList(lpItemID, TempPath);
Result := TempPath;
GlobalFreePtr(lpItemID);
end;
end;

function HasAttr(const FileName: string; Attr: Word): Boolean;
var FileAttr: Integer;
begin
FileAttr := FileGetAttr(FileName);
if FileAttr = -1 then FileAttr := 0;
Result := (FileAttr and Attr) = Attr;
end;



procedure TfrMain.UpdateChecked(Node: TTreeNode);
begin
if Node=nil then
begin
Exit;
end;

if Node.Parent=nil then
begin
Exit;
end;

PShellObject(Node.Data)^.DoIt := Node.SelectedIndex = Yes;
end;
```

```
procedure TfrMain.SaveData;
var // змінні
backupFile : TBackupFile;
F        : file of TBackupFile;
tn, tnc  : TTreeNode;
begin
//збереження
try
AssignFile(F,ChangeFileExt(Application.ExeName,'.dat'));
Rewrite(F);

tn:=tree.Items.GetFirstNode;
while tn <> nil do
begin
backupFile.GroupName:=tn.Text;
backupFile.DoIt := tn.SelectedIndex = Yes;
tnc:= tn.getFirstChild;
while tnc <> nil do
begin
backupFile.ShellObject.DoIt := tnc.SelectedIndex=Yes;
backupFile.ShellObject.LiteralName := PShellObject(tnc.Data)^.LiteralName;
backupFile.ShellObject.SourcePath := PShellObject(tnc.Data)^.SourcePath;
backupFile.ShellObject.Prefix := PShellObject(tnc.Data)^.Prefix;
backupFile.ShellObject.Sufix := PShellObject(tnc.Data)^.Sufix;
backupFile.ShellObject.DestinationFolder :=
PShellObject(tnc.Data)^.DestinationFolder;
backupFile.ShellObject.AttachDateTime := PShellObject(tnc.Data)^.AttachDateTime;

Write (F, backupFile);

//очищення
Dispose(PShellObject(tnc.Data));

tnc:=tnc.getNextSibling;
end;
 tn:=tn.getNextSibling;
end;
 finally
CloseFile(F);
end;//спроба

end;

procedure TfrMain.LoadData;
var // змінні
backupFile : TBackupFile;
F: file of TBackupFile;
ShellObjectPtr: PShellObject;
RacName:string;
tn, tnc:TTreeNode;
begin
//читання
If not FileExists(ChangeFileExt(Application.ExeName,'.dat')) then Exit;

AssignFile(F,ChangeFileExt(Application.ExeName,'.dat'));
Reset(F);

try
while not Eof(F) do
begin
Read(F, backupFile);

RacName:=backupFile.GroupName;

tn:=tree.Items.AddChild(nil,backupFile.GroupName);
if backupFile.DoIt = True then
begin
tn.SelectedIndex := Yes; tn.ImageIndex := Yes;
end
```

```
else
begin
tn.SelectedIndex := No; tn.ImageIndex := No;
end;

while RacName=backupFile.GroupName do
begin
New(ShellObjectPtr);

ShellObjectPtr^.LiteralName := backupFile.ShellObject.LiteralName;
ShellObjectPtr^.SourcePath := backupFile.ShellObject.SourcePath;
ShellObjectPtr^.Prefix := backupFile.ShellObject.Prefix;
ShellObjectPtr^.Sufix := backupFile.ShellObject.Sufix;
ShellObjectPtr^.DestinationFolder := backupFile.ShellObject.DestinationFolder;
ShellObjectPtr^.AttachDateTime := backupFile.ShellObject.AttachDateTime;

tnc:=tree.Items.AddChildObject(tn,backupFile.ShellObject.LiteralName,ShellObject
Ptr);

if backupFile.ShellObject.DoIt = True then
begin
tnc.SelectedIndex := Yes; tnc.ImageIndex := Yes;
end
else
begin
tnc.SelectedIndex := No; tnc.ImageIndex := No;
end;

if not Eof(F) then
begin
Read(F, backupFile);
if RacName<>backupFile.GroupName then Seek(F,FilePos(F)-1);
end
else Break;
end;//якщо RacName=backupFile.GroupName
end; //while eof
tree.AlphaSort(True);
finally
CloseFile(F);
end;//спроба

end;


function TfrMain.DoTransfer(const sDateTime:string; const backupFile:
TBackupFile) : integer;
var // змінні
fileOp                : TShFileOpStruct;
fromF, toF            : string;
FileName, Destination : string;
tempName              : string;
begin

if (backupFile.ShellObject.SourcePath = '') or
(backupFile.ShellObject.DestinationFolder = '') then
begin
Result:=-1;
Exit;
end;

if HasAttr(backupFile.ShellObject.SourcePath, faDirectory) AND (NOT
DirectoryExists(backupFile.ShellObject.SourcePath)) then
begin
Result:=-1;
Exit;
end;

if HasAttr(backupFile.ShellObject.SourcePath, faArchive) AND (NOT
FileExists(backupFile.ShellObject.SourcePath)) then
```

```
begin
Result:=-1;
Exit;
end;

//старт
FileName := backupFile.ShellObject.SourcePath;
tempName := ExtractFileName(FileName);
tempName :=  StringReplace(tempName, ExtractFileExt(FileName),
backupFile.ShellObject.Sufix, [rfReplaceAll, rfIgnoreCase]) +
ExtractFileExt(FileName);
Destination :=
IncludeTrailingPathDelimiter(backupFile.ShellObject.DestinationFolder +
sDateTime) + backupFile.ShellObject.Prefix + tempName;
Destination := ExpandUNCFileName(Destination);
//стоп

ForceDirectories(ExtractFilePath(Destination));

if HasAttr(backupFile.ShellObject.SourcePath, faDirectory) then
begin
fromF := IncludeTrailingPathDelimiter(backupFile.ShellObject.SourcePath) +
'*.*'#0;
toF   := IncludeTrailingPathDelimiter(Destination) + #0;
end
else begin
fromF := backupFile.ShellObject.SourcePath + #0;
toF   := Destination + #0;
end;

FillChar(fileOp, Sizeof(TShFileOpStruct), 0);
with fileOp do begin
wnd     := Handle;
wfunc   := FO_COPY;
pFrom   := PChar(fromF);
pTo     := PChar(toF);
fFlags  := FOF_NOCONFIRMATION OR FOF_NOCONFIRMMKDIR OR FOF_SILENT;
fAnyOperationsAborted := False;
hNameMappings         := nil;
lpszProgressTitle     := nil;
end;

Result := SHFileOperation(fileOp);
end;


procedure TfrMain.btnExitClick(Sender: TObject);
begin
Close;
end;

procedure TfrMain.sbtnCollapseClick(Sender: TObject);
begin
Tree.FullCollapse;
end;

procedure TfrMain.sbtnExpandClick(Sender: TObject);
begin
tree.FullExpand;
end;

function TfrMain.CtrlDown : Boolean;
var // змінні
State : TKeyboardState;
begin
GetKeyboardState(State);
Result := ((State[vk_Control] And 128) <> 0);
end;
```

```
procedure TfrMain.treeDblClick(Sender: TObject);
var // змінні
tn : TTreeNode;
tnc : TTreeNode; begin
tn:=tree.Selected;
if tn = nil then Exit;

if CtrlDown then
begin
if tn.ImageIndex = Yes then
begin
tnc := tn.getFirstChild;
while tnc <> nil do
begin
tnc.ImageIndex := No;
tnc.SelectedIndex:=No;
PShellObject(tnc.Data)^.DoIt := False;

tnc:=tnc.getNextSibling;
end;
end
else
begin
tnc := tn.getFirstChild;
while tnc <> nil do
begin
tnc.ImageIndex := Yes;
tnc.SelectedIndex:=Yes;
PShellObject(tnc.Data)^.DoIt := True;

tnc:=tnc.getNextSibling;
end;
end;
end;

if tn.ImageIndex = Yes then
begin
tn.ImageIndex := No;
tn.SelectedIndex:=No;
if tn.Parent <> nil then
begin
PShellObject(tn.Data)^.DoIt := False;
end
else tn.Collapse(True);
end
else
begin
tn.ImageIndex := Yes;
tn.SelectedIndex := Yes;
if tn.Parent <> nil then
begin
PShellObject(tn.Data)^.DoIt := True;
end
else tn.Expand(True);
end;

tree.refresh;
end;

procedure TfrMain.treeClick(Sender: TObject);
var // змінні
tn:TTreeNode;
FileName,tempName, Destination : string;
sDateTime : string;
begin
tn:=tree.Selected;
if fLastSelected = tn then UpdateChecked(fLastSelected);
lblItemSelected.caption:='';
```

```
if tn = nil then Exit;
if tn.Parent=nil then //якщо група
begin
lblItemSelected.caption:=tn.Text;
pnlLed.Color:=clSilver;
ledLiteralName.Text := '';
ledStazaDo.Text := '';
ledPrefix.Text := '';
ledSufix.Text := '';
ledDestFolder.Text := '';
lblResultObject.Caption:='';
chkAttachDateTime.State:=cbGrayed;
end
else //якщо об'єкт
begin
try
pnlLed.Color:=clBtnFace;
lblItemSelected.caption:=tn.Parent.Text + ' >> ';
ledLiteralName.Text := PShellObject(tree.Selected.Data)^.LiteralName;
ledStazaDo.Text := PShellObject(tree.Selected.Data)^.SourcePath;
ledPrefix.Text := PShellObject(tree.Selected.Data)^.Prefix;
ledSufix.Text := PShellObject(tree.Selected.Data)^.Sufix;
ledDestFolder.Text := PShellObject(tree.Selected.Data)^.DestinationFolder;
chkAttachDateTime.Checked:=PShellObject(tree.Selected.Data)^.AttachDateTime;

FileName := ledStazaDo.Text;
tempName := ExtractFileName(FileName);
tempName :=  StringReplace(tempName, ExtractFileExt(FileName), ledSufix.Text,
[rfReplaceAll, rfIgnoreCase]) + ExtractFileExt(FileName);
if chkAttachDateTime.Checked then
begin
sDateTime:= StringReplace(DateTimeToStr(Now),DateSeparator,'-', [rfReplaceAll,
rfIgnoreCase]);
sDateTime:= ' ' + StringReplace(sDateTime,TimeSeparator,'-', [rfReplaceAll,
          rfIgnoreCase]);
end;
Destination := IncludeTrailingPathDelimiter(ledDestFolder.Text + sDateTime) +
                ledPrefix.Text + tempName + ledSufix.Text;
lblResultObject.Caption := MinimizeName(ExpandUNCFileName(Destination), canvas,
                           lblResultObject.ClientWidth);
except
end;
end;

fLastSelected:=tree.Selected;

end;

procedure TfrMain.btnDelGroupClick(Sender: TObject);
begin
if tree.Selected = nil then
begin
MessageDlg('Виділіть групу зі списку!', mtWarning,[mbOK],0);
Exit;
end;

if tree.Selected.Parent <> nil then
begin
Exit;
end;

if MessageDlg('Виділити групу ' +  tree.Selected.Text, mtConfirmation,
             mbOKCancel,0) = mrOK then
begin
tree.Selected.Delete;
end;

frMain.treeClick(Sender);
end;
```

```
procedure TfrMain.btnNewGroupClick(Sender: TObject);
var // змінні
racName: string;
tn:TTreeNode;
begin
racName:=InputBox('Нова група,'Ім’я групи (опис):','');
if racName<>'' then
begin
tn:=tree.Items.AddChild(nil,Copy(racName,0,20));
tn.Selected:=True;
end;
end;


procedure TfrMain.btnDelObjectClick(Sender: TObject);
begin
if tree.Selected=nil then
begin
Exit;
end;

if tree.Selected.Parent=nil then
begin
Exit;
end;

if MessageDlg('Видалити ' +  tree.Selected.Text + ' in ' +
tree.Selected.Parent.Text, mtConfirmation, mbOKCancel,0) = mrOK then
begin
Dispose(PShellObject(tree.Selected.Data));
tree.Selected.Delete;
end;


end;

procedure TfrMain.FormClose(Sender: TObject; var Action: TCloseAction);
begin
SaveData;
end;

procedure TfrMain.FormCreate(Sender: TObject);
var // змінні
tabs: Array [0..9] of Integer;
i:integer;
begin
Page.ActivePage:=tsAbout;

LoadData;

lblItemSelected.Caption:='';
lblResultObject.Caption:='';

self.pbarGroups.Position:=0;
self.pbarObjects.Position:=0;
for i:=0 to 9 do
begin
tabs[i] := i * 3 * 4;
end;
cdMemo.Clear;
cdMemo.Perform(EM_SETTABSTOPS, 10, LongInt(@tabs));
cdMemo.Refresh;

memoErrors.Clear;
end;

procedure TfrMain.treeChanging(Sender: TObject; Node: TTreeNode;
var AllowChange: Boolean);
```

```
begin
UpdateChecked(tree.Selected);
end;

procedure TfrMain.treeExit(Sender: TObject);
begin
UpdateChecked(tree.Selected);
end;

procedure TfrMain.btnEngageClick(Sender: TObject);
var // змінні
backupFile:TBackupFile;
tn,tnc:TTreeNode;
NrOf, errCount: integer;
TransferResult:integer;
sDateTime : string;
begin

sDateTime:= StringReplace(DateTimeToStr(Now),DateSeparator,'-', [rfReplaceAll,
rfIgnoreCase]);
sDateTime:= ' ' + StringReplace(sDateTime,TimeSeparator,'-', [rfReplaceAll,
rfIgnoreCase]);

errCount:=0;

UpdateChecked(tree.Selected);

Page.ActivePage:=tsStatus;
cdMemo.Clear;
Refresh;

btnEngage.Enabled:=False;

tn:=tree.Items.GetFirstNode;
NrOf:=0;
while tn <> nil do
begin
Inc(NrOf);
tn:=tn.getNextSibling;
end;
pbarGroups.Min:=0;
pbarGroups.Max:=NrOf;
pbarGroups.Position:=pbarGroups.Min;


// початок копіювання
Animate.Active:=True;
memoErrors.Lines.Clear;
tn:=tree.Items.GetFirstNode;
while tn <> nil do
begin
pbarGroups.StepBy(1);

backupFile.GroupName := tn.Text;
backupFile.DoIt := tn.SelectedIndex = Yes;
if NOT backupFile.DoIt then
begin
cdMemo.Lines.Add(keyTAB + 'HI!')
end
else
begin

 NrOf:=0;
 tnc := tn.getFirstChild;
 while tnc <> nil do
 begin
   Inc(NrOf);
   tnc:=tnc.getNextSibling;
 end;
```

```
 pbarObjects.Min:=0;
 pbarObjects.Max:=NrOf;
 pbarObjects.Position:=pbarObjects.Min;

tnc := tn.getFirstChild;
while tnc <> nil do
begin
Refresh;
pbarObjects.StepBy(1);

backupFile.ShellObject.DoIt := tnc.SelectedIndex=Yes;
backupFile.ShellObject.LiteralName := PShellObject(tnc.Data)^.LiteralName;

if NOT backupFile.ShellObject.DoIt then
begin
 cdMemo.Lines.Add(keyTAB + 'Hi!' + keyTAB + 'Об’єкт: ' +
backupFile.ShellObject.LiteralName)
end
else
begin
    backupFile.ShellObject.SourcePath := PShellObject(tnc.Data)^.SourcePath;
    backupFile.ShellObject.Prefix := PShellObject(tnc.Data)^.Prefix;
    backupFile.ShellObject.Sufix := PShellObject(tnc.Data)^.Sufix;
    backupFile.ShellObject.DestinationFolder :=
PShellObject(tnc.Data)^.DestinationFolder;
    backupFile.ShellObject.AttachDateTime :=
PShellObject(tnc.Data)^.AttachDateTime;

    stext.Caption := tn.text + keyTAB +  ' Об’єкт: ' +
backupFile.ShellObject.LiteralName;
    stext.Refresh;

    if backupFile.ShellObject.AttachDateTime then
       TransferResult := DoTransfer(sDateTime, backupFile)
    else
       TransferResult := DoTransfer('', backupFile);

    if TransferResult = 0 then
      cdMemo.Lines.Add(keyTAB + 'OK! ' + keyTAB + 'OBJECT: ' +
                       backupFile.ShellObject.LiteralName)
    else
    begin
      cdMemo.Lines.Add(keyTAB + 'Помилка! ' + stext.Caption);
      memoErrors.Lines.Add('ERRORS: ' + IntToStr(TransferResult) + ' ' +
stext.Caption);
      Inc(errCount);
    end;
    Refresh;
end;
tnc:=tnc.getNextSibling;
end;
end;
tn:=tn.getNextSibling;
end;

Animate.Active:=False;
btnEngage.Enabled:=True;
stext.Caption:='Transfer complete. Error count : ' + IntToStr(errCount);
end;

procedure TfrMain.treeChange(Sender: TObject; Node: TTreeNode);
begin
frMain.treeClick(Sender);
end;

procedure TfrMain.AppEventsActivate(Sender: TObject);
begin
frMain.Refresh;
end;
```

```
procedure TfrMain.btnUpdateObjectClick(Sender: TObject);
begin

if (NOT DirectoryExists(ledStazaDo.Text)) and (NOT FileExists(ledStazaDo.Text))
then
begin
ledStazaDo.EditLabel.Caption,mtWarning,[mbOK],0);
ledStazaDo.SetFocus;
Exit;
end;
tree.Selected.Text:= ledLiteralName.Text;
PShellObject(tree.Selected.Data)^.LiteralName := ledLiteralName.Text;
PShellObject(tree.Selected.Data)^.SourcePath := ledStazaDo.Text;
PShellObject(tree.Selected.Data)^.Prefix := ledPrefix.Text;
PShellObject(tree.Selected.Data)^.Sufix := ledSufix.Text;
PShellObject(tree.Selected.Data)^.DestinationFolder := ledDestFolder.Text;
PShellObject(tree.Selected.Data)^.AttachDateTime := chkAttachDateTime.Checked;
treeClick(Sender);
tree.AlphaSort(True);
end;

end.
```