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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Маліна О.С.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2025 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 2025 року

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

Маліні Олександру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 51-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях*

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 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

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

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Маліна О.С.
(прізвище та ініціали)

АНОТАЦІЯ

Маліна О.С. Програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

Метою розробки є програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

Результат роботи – програмна реалізація системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

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 мобільних пристроях під керуванням ОС Android.

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, захист додатків та даних

ABSTRACT

Malina O.S. Software for a cybersecurity system to protect applications and data on mobile corporate devices. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a cybersecurity system to protect applications and data on mobile corporate devices.

The purpose of the development is to develop software for a cybersecurity system to protect applications and data on mobile corporate devices.

The result of the work is a software implementation of a cybersecurity system to protect applications and data on mobile corporate devices.

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 are provided.

The program can be used on mobile devices running the OS Android.

The program is developed in Python.

Keywords: cybersecurity, application and data protection

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AP	–	точка доступу
DSSS	–	технологія Direct Sequence Spread Spectrum
EAP	–	протокол розширеної автентифікації Extensible Authentication Protocol
MIC	–	криптографічна контрольна сума
RADIUS	–	сервер доступу Remote Access Dial-in User Server
RC4	–	алгоритм шифрування
TKIP	–	протокол генерації генерація ключів WPA шифрування даних Temporal Key Integrity Protocol
TLS	–	протокол захисту транспортного рівня Transport Layer Security
SSID	–	ідентифікатор мережі який передає точка доступу
WEP	–	Wired Equivalent Privacy – протокол безпеки
WPA	–	стандарт безпеки Wi-Fi Protected Access

ВСТУП

Актуальність теми. Мобільні пристрої все частіше використовуються в корпоративному середовищі, і це створює проблеми для фахівців, що відповідають в організації за інформаційну безпеку. В умовах «мобілізації» бізнес-процесів доводиться шукати баланс між комфортною роботою співробітників, з одного боку, і інформаційною безпекою бізнесу, з іншої. За прогнозами Ericsson, з 2024 по 2027 рік кількість використовуваних у світі мобільних пристроїв виросте з 6,7 до 9,3 млрд. Як очікують аналітики IDC, в 2025 році продажу смартфонів у світі перевищать 1 млрд штук, з яких більше 400 млн будуть використовуватися в бізнес-середовищу, причому більшою мірою з ініціативи співробітників компаній, а не внаслідок вимог корпоративних стандартів.

Концепція використання власного пристрою в рамках корпоративної IT-інфраструктури (Bring Your Own Device, BYOD) набирає популярність, хоча й викликає в IT-керівників чимало питань, насамперед пов'язаних із забезпеченням безпеки, контролю, адміністрування й підтримки. Аналітики Gartner називають використання співробітниками власних мобільних пристроїв однієї з основних проблем ІБ. Мобільність міняє фундаментальні принципи доставки додатків користувачам і захисту їхніх даних у частині авторизації, автентифікації, контролю доступу й функцій керування. В Gartner вважають, що усе більше завдань безпеки буде покладати на кінцевих користувачів. У них з'являться нові обов'язки по захисту даних і дотриманню прийнятих компаніями вимог. Для цього необхідно організовувати навчання користувачів і здійснювати моніторинг їхніх дій.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

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

– Огляд існуючих систем для захисту додатків та даних на мобільних корпоративних пристроях.

– Дослідження системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

– Програмна реалізація системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для захисту додатків та даних на мобільних корпоративних пристроях.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях, 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

Оскільки перехід на мобільні й хмарні технології ставить перед фахівцями з ІБ складні завдання, витрати на IT-безпеку не тільки не скорочуються, але й продовжують рости значно швидше ринку в цілому. Аналітики IDC переконані, що в найближчі роки ця тенденція збережеться.

За даними опитувань, дві третини закордонних компаній уже практикують підключення мобільних пристроїв до корпоративної інфраструктури. Якщо в 2014 році лише кожний п'ятий співробітник корпоративних клієнтів використовував для роботи смартфон, то в 2024 році вже кожний четвертий, а в поточному, згідно із прогнозами, таким буде кожний третій.

По визнанню експертів, незважаючи на те що модель BYOD різко збільшує кількість інцидентів ІБ, вона стає усе популярніше – неї використовують як великі, так і невеликі підприємства з різних галузей економіки. За даними дослідження Dimensional Research, проведеного в 2024 році за замовленням Check Point, 67% закордонних компаній дозволяють співробітникам підключатися до корпоративних мереж з мобільних пристроїв, причому в 45% таких організацій число особистих пристроїв збільшилося за останні два роки в п'ять разів.

При цьому 63% організацій ніяк не управляють корпоративною інформацією, що перебуває на особистих пристроях, 93% зіштовхуються зі складностями при впровадженні політики безпеки, а 67% уважають захист корпоративної інформації головною проблемою BYOD. 60% опитаних думають, що безтурботні співробітники представляють більший ризик для IT-безпеки, чим кіберзлочинці. Обсяги конфіденційних даних і корпоративної інформації, що зберігаються на мобільних пристроях, постійно ростуть. Її втрата або витік може

мати серйозні наслідки. Торік 79% респондентів зштовхнулися хоча б з одним інцидентом мобільної безпеки.

Підтримка безлічі різних мобільних пристроїв, тисячі неконтрольованих додатків, віддалена робота, розмивання границь корпоративної мережі й хмарних сервісів, керовані зовнішніми провайдерами, – все це ускладнює завдання забезпечення ІБ настільки, що аналітики не виключають навіть спаду інтересу до BYOD. Доцільність впровадження BYOD залежить від здатності компаній забезпечити безпека й контроль доступу до корпоративних ресурсів не тільки з офісу, але й з будь-якої точки, де може перебувати співробітник. Нарешті, хоча модель BYOD дозволяє компаніям скоротити витрати на покупку встаткування, вартість обслуговування ИТ збільшується.

У березні Oracle опублікувала звіт «Oracle European BYOD Index Report», що розкриває відношення представників бізнесу в Європі до концепції BYOD. За даними дослідження, 44% компаній не схвалюють цю концепцію або допускають її застосування лише у виняткових обставинах. В 29% організацій пристрою BYOD використовуються тільки керівниками, в 22% категорично заборонене розміщення корпоративної інформації на пристроях BYOD, а в 20% не прийнято ніяких правил. Більше половини підприємств не управляють смартфонами в рамках BYOD.

Безпека інформації – сама серйозна проблема BYOD. По даним Oracle, респонденти стурбовані захистом пристроїв (45%), додатків (53%) і даних (63%). Однак багато які із цих побоювань пов'язані з недостатньою поінформованістю про можливості сучасних рішень: 37% опитаних ніколи не чули про контейнеризацію (поділ корпоративних і персональних даних), майже третина не використовують технології для керування мобільними пристроями (Mobile Device Management, MDM), 22% не знайомі з технологіями керування мобільними додатками (Mobile Application Management, MAM). Проблема забезпечення безпеки змушує багато організацій по всій Європі відмовлятися від BYOD. У той же час лише 21% прихильників BYOD серйозно турбує захист

інформації. Вони добре інформовані про доступні технології – наприклад, близько 80% використовують які-небудь засоби керування мобільними додатками, і лише 7% не управляють захистом даних або зберігають їх на пристроях у незашифрованому виді.

Регламенти побудови мобільної інфраструктури – рідке явище. В основному все покладаються на особистий досвід співробітників і їхнє подання про безпеку й ризики. Іноді компанії намагаються орієнтуватися на стандарти побудови систем керування ІБ, оскільки відповідні норми для мобільного сектора відсутні. Бар'єрами на шляху до реалізації «корпоративної мобільності» уважають велика розмаїтість пристроїв і ПЗ, швидку еволюцію пристроїв, при цьому відзначається, що використання додатків в особистих цілях створює ризики витоку даних і втрати корпоративної інформації.

IDC рекомендує застосовувати централізовані системи керування ІБ, рішення MDM, розглянути можливість розгортання систем керування доступом (Identity and Access Management, IAM), у тому числі мережним (Network Access Control, NAC), і шифрування на клієнтських пристроях. В умовах розмитого периметра потрібна грамотна сегментація мережі. У великих організаціях зростає роль рішень керування подіями безпеки (Security Information and Event Management, SIEM), запобігання витоку даних (Data Leak Prevention, DLP), міжмережних екранів нового покоління (NGFW) з ідентифікацією користувачів, додатків і контролем даних. Використання VPN забезпечить захищене підключення до мережі організації.

1.2 Область застосування

«Корпоративна мобільність» складається з багатьох рішень, і одним з основних є MDM. Ця система керування дозволяє контролювати локальні додатки на пристроях, надавати надійний парольний захист пристрою й т.п. Вона реалізує такі функції, як віддалене відновлення правил безпеки, поширення

додатків і даних, а також керування конфігурацією. Рішення MDM забезпечують видалення даних із пристрою у випадку його втрати або крадіжки, шифрування убудованої пам'яті й карт зберігання даних, створення захищених контейнерів для даних додатків. Однак це не дає 100-процентної гарантії від злому пристрою, випадкового витоку даних або від дій інсайдерів.

Рішення MDM розраховані на певний сегмент компаній, що вирішують завдання забезпечення ІБ мобільних пристроїв. Однак потенційні клієнти найчастіше не інформовані про сучасні технічні й програмні засоби, а також можливостях MDM. Крім того, через те, що на ринку не сформувалося серйозних історій успіху, вибір рішень MDM визначається випадковими факторами. Кожна компанія фактично самостійно проводить тестування й пробує різні варіанти використання MDM. У результаті цей процес затягається, а компанії, що навіть використовують мобільні технології, фактично не управляють своєю інфраструктурою.

Дуже багато чого залежить від галузевого сегмента й вимог до безпеки. MDM – це серйозна система зі своєю логікою, і роботі з нею потрібно навчатися. А оскільки нові версії ОС виходять дуже швидко, вона вимагає постійного відновлення. При виборі рішення MDM дуже важливим фактором є наявність україномовної підтримки. Провідні рішення MDM схожі по функціональних можливостях, однак лише деякі вендори готові спілкуватися із клієнтами в Україні.

Це рішення стало основою для розгортання хмарного сервісу на базі системи SAP. Послуга MDM дозволяє управляти планшетами й смартфонами з Web-інтерфейсу: налаштовувати пристрої (паролі, пошту, контакти, сертифікати, доступ у корпоративну мережу), установлювати й контролювати додатка (чорний і білий списки), вести статистику по пристроях і додаткам, управляти політикою безпеки (наприклад, блокувати камеру), видаляти дані. Однак перелік доступних функцій різний для різних мобільних платформ.

Особливості продукту – моментальне підключення сервісу (не потрібно встановлювати встаткування або ПЗ), оплата «за використання», єдиний інтерфейс керування смартфонами й планшетами всіх популярних ОС, більше 70 політик безпеки. За даними оператора, щодня це рішення підключає для себе ще одна компанія.

При виборі рішення MDM у першу чергу треба звернути увагу на те, якою мірою система відповідає стратегічним цілям компанії й чи не заважає нововведення виконанню ключових бізнес-процесів. Якщо є можливість інтегрувати MDM з іншими корпоративними системами, особливо в частині керування користувачами й генерації звітності, процес його експлуатації буде простіше. Що стосується безпеки, то не зайвими будуть такі функції, як керування установкою додатків на мобільні пристрої за принципом білих і чорних списків або наявність власного корпоративного магазина додатків. Можливість поділу персональної й корпоративної частин інтерфейсу мобільного пристрою спрощує інтеграцію рішення в корпоративну IT-інфраструктуру й підвищує рівень безпеки при роботі з конфіденційними даними.

Системи класу MDM не можуть вирішити всіх проблем, пов'язаних з BYOD і керуванням різними аспектами корпоративної мобільності, але дозволяють досягти максимальної ефективності в застосуванні пристроїв. Вони дозволяють точечно впливати на всі фактори експлуатації парку мобільних пристроїв і додатків, включаючи поширення додатків і керування їхніми настроюваннями, передачу ключової інформації для шифрування даних, централізоване ведення журналів і діагностику. Рішення, що надають зазначені функції, а також інструменти підтримки розроблювачів поступово з'являються на ринку й уже зараз можуть бути використані для ефективного керування мобільними пристроями.

Аналітики Gartner відносять до лідерів ринку MDM компанії MobileIron, Airwatch (VMware), Fiberlink, Zenprize і Good Technology. Популярність одержали вже більше десятка вендорів таких рішень.

Особливу роль здобуває захист мобільних пристроїв від шкідливих програм. Компанія McAfee в 2024 році зібрала 2,47 млн нових зразків таких програм для мобільних пристроїв. На кінець минулого року їх налічувалося вже 3,73 млн, що на 197% більше, ніж до кінця 2014 року. Шкідливі програми можуть потрапити на мобільний пристрій у результаті завантаження додатків, відвідування шкідливих Web-сайтів, одержання спама й SMS, завантаження рекламних баннерів. Все більше поширення одержують мобільні додатки для збору користувальницьких даних і телеметричної інформації про мобільний пристрій.

Програма, установлена на телефоні генерального директора компанії й яка відслідковує місцезнаходження, може насправді виявитися шпигуном, що надає інформацію конкурентам, постачальникам, фінансовим аналітикам, шантажистам і навіть тим, хто збирається заподіяти фізичну шкоду. McAfee (Intel Security Group) веде базу даних про репутацію мобільних додатків, а для захисту найбільш уразливих мобільних платформ на базі Android випущений безкоштовний антивірус.

Для кожної компанії стратегія повинна враховувати її специфіку, але пари особливостей варто виділити. Насамперед, не варто робити вигляд, що проблем BYOD можна уникнути. Офісні співробітники й, що дуже важливо, керівники, використовують для роботи ті пристрої, який, незважаючи на приписання, найбільше зручно користуватися в даній конкретній ситуації. Крім того, «стратегія BYOD» – це погоня за постійно, що віддаляється метою, тому немає можливості визначити фронт робіт і реалізувати всі контрзаходи. Спектр пристроїв буде постійно розширюватися, а користувачі стануть вишукувати способи застосовувати особисті пристрої для виконання своїх посадових обов'язків». Що ж змушує організації йти на ризик?

Переваги BYOD

У звіті «Oracle European BYOD Index» затверджується, що в організаціях, що впроваджують BYOD, скорочуються IT-витрати й підвищується ефективність

роботи користувачів. У цей час існують рішення, що пропонують засоби контролю безпеки для корпоративних і персональних пристроїв. З їхньою допомогою організації можуть гнучко визначати права доступу до корпоративної інформації, здійснювати більше детальний контроль завдяки ізоляції корпоративних і персональних даних, надавати захищений доступ до корпоративних додатків і керування даними.

В Oracle вважають, що прихильники концепції BYOD у стані впоратися з багатьма проблемами забезпечення безпеки й готові до подальшого розвитку цієї концепції. Такі технології, як контейнеризація, шифрування й керування пристроями й додатками, інтегровані з єдиним корпоративним сховищем ідентифікаційної інформації, здатні захистити середовища BYOD.

«Мобілізація» бізнесу надає переваги всім організаціям, поза залежністю від їхнього розміру й галузевої специфіки, упевнені в Softline. Кількість компаній, що вже оцінила вигоди від мобільності співробітників, продовжує збільшуватися. Віддалений доступ до всіх корпоративних ресурсів (пошті, програмам «1С», порталу, CRM- і ERP-системам і т. буд.) з особистого або робочого ноутбука, планшета або смартфона стає нормою в сучасній організації. Це вимагає вибудовування життєвого циклу використання мобільних пристроїв, що регламентує всі процеси – від інвентаризації до віддаленої технічної підтримки.

Поряд з BYOD з'явилися й більше гнучкі варіанти застосування мобільних пристроїв у корпоративному середовищі. Оскільки багато компаній з небажанням дозволяють використання мобільних пристроїв, та й користувачі не завжди палко бажають застосовувати свої смартфони або планшети в робочих цілях, компанії іноді надають співробітникам пристрій по їхньому виборі із уже оформленим договором на надання послуг зв'язку. Така модель одержала назву «вибери свій пристрій» (Choose Your Own Device, CYOD). Іноді цей мобільний пристрій дозволяється використовувати в особистих цілях (Corporate Owned, Personally Enabled, COPE), при цьому співробітникам роздають однакові мобільні пристрої

із установленим на них стандартним ПЗ. Моделі COPE і BYOD розраховані на різні групи користувачів, можуть застосовуватися одночасно й, таким чином, допомагають стандартизувати керування й забезпечення безпеки.

Для зниження ризиків, пов'язаних з BYOD, у периметр безпеки компанії варто включити мобільні пристрої за аналогією з тим, як контролюються зараз робочі станції й сервери. Проблема в тому, що число моделей смартфонів і версій мобільних ОС дуже велико. Таким чином, якщо співробітникам дозволяється працювати з корпоративною інформацією зі своїх пристроїв, то на них необхідно примусово встановити системи забезпечення ІБ для відповідного варіанта операційних систем і моделей смартфонів, що на практиці трапляється рідко. Виходом може стати надання співробітникам обмеженого числа корпоративних мобільних пристроїв на єдиній платформі. При масовій закупівлі вони обійдуться відносно недорого, а схема керування ними буде зрозуміла й прозора. Та й з юридичної точки зору набагато простіше управляти пристроєм, що є власністю підприємства.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях, 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

З розвитком всесвітньої павутини, багато користувачів зштовхнулися з необхідністю зберігати незліченна безліч паролів від аккаунтів до різних інтернет ресурсів. І якщо ще на початку двохтисячних можна було обійтися всього парою облікових записів, те тепер практично в кожного з нас є кілька електронних ящиків, сторінок у соціальних мережах, облікових записів на форумах ну й звичайно пластикових або віртуальних карт Visa і MasterCard. Однак, щоб запам'ятати дані всього цього багатства, необхідно мати мозок нобелівського лауреата, ну або звичайно ризикнути й використовувати скрізь той самий захисний код. Обидва варіанти по своєму гарні й саме завдяки цьому з'явилися додатки-сховища для надійного захисту даних в одному місці й з окремим майстер-кодом. Про їх-те ми сьогодні й поговоримо.

У нашу добірку ввійшли універсальні сховища даних для Android. Деякі з них орієнтуються на простий захист інформації, деякі мають додаткові функції, такими як хмарна синхронізація або прив'язка фото й відео.

Ewallet

Це, напевно, найвідоміше рішення для збереження інформації на різних пристроях. Розроблювач eWallet у свій час випустив версії для всіх самих популярних платформ (Android, iOS, Windows Mobile, Symbian, PC), чим заслужив репутацію доступного й надійного рішення для зберігання конфіденційної інформації. На Android додаток уміє створювати окремі гаманці як для одного користувача, так і для декількох, з різними майстрами-кодами. А зручна функція генерації паролів позбавить вас від зайвих турбот. Безпосередньо

в гаманці можна створювати, змінювати, видаляти категорії й підкатегорії. До записів можна прив'язувати стікери й зображення, завдяки чому ваше сховище придбає більше естетичний вид, не на шкоду, звичайно, зручності розподілу записів. Занесені в eWallet дані шифруються по 256-бітному AES-алгоритму, що став останнім часом уже стандартом для захисту даних. Єдиний мінус додатка – відсутність повноцінного резервного копіювання у файл. Розроблювачі звичайно передбачили синхронізацію з PC версією, але такий своєрідний "повідець" багатьом може не сподобатися.

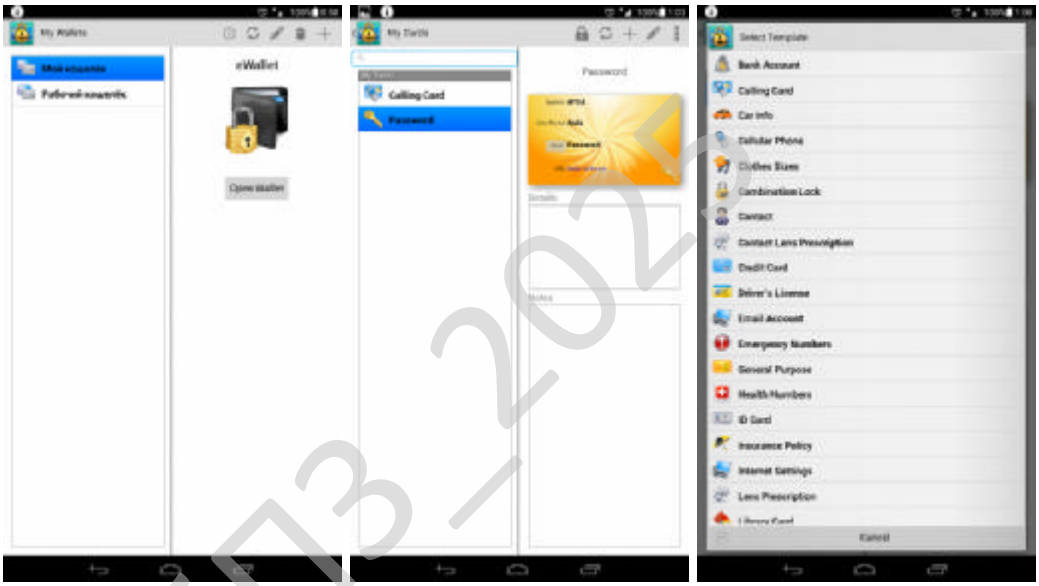

Рисунок 2.1 – Інтерфейс користувача eWallet

Переваги:

– кілька гаманців для різних користувачів;

– прив'язка зображень і стікеров;

– список категорій, що налаштовується, і записів.

Недоліки:

– відсутність повноцінного резервування бази даних.

Password safe

Просте й досить зручне додатки для зберігання даних. В Password Safe можна створювати категорії для карт, документів, паролів і аккаунтів, для більш зручного зберігання й каталогізації даних. Для шифрування використовується 256-бітний алгоритм AES, що робить злом даних практично неможливим. Додаткова система захисту включає приховання додатка із системного аркуша останніх запущених, а також автоматичне блокування по таймері й блокування скриншотів робочих областей додатка. Для початку використання користувачеві знадобиться задати базовий майстер-код, надалі ніяких додаткових маніпуляцій не знадобиться. Розроблювачі вмонтували можливість резервного копіювання й імпорту даних через ".CSV-файл" – правда ці функції стануть доступні тільки при покупці повної версії. В іншому безкоштовний варіант нічим не відрізняється від повнофункціонального, у ньому навіть немає реклами, за що авторам можна сказати тільки спасибі.

Рисунок 2.2 – Інтерфейс користувача Password Safe

Переваги:

– категорії для записів;

– додатковий захист додатка;

– ні реклами.

Недоліки:

– можливість резервування бази даних тільки за гроші.

Awallet Cloud Password Manager / awallet Password Manager

Універсальний менеджер паролів і облікових записів. Для шифрування даних додаток використовує 128-, 256- і 192-бітний AES-алгоритм, 112- і 168-бітний DES-алгоритм, а також кілька режимів їхньої роботи, при цьому додатково захищаючи сховище автоблокуванням (при роботі в тлі й по таймері) і захищаючи додаток від зняття скриншотів. На головному екрані виведений редактор категорій, загальний список категорій і вибране. Передвстановлені каталоги можна редагувати, видаляти й заміняти своїми власними. Для редагування доступні як шаблони, так і чисті бланки. Шаблони у свою чергу, можна налаштовувати для уведення різного типу даних: числової, текстової, символьні й дати. Якщо буде потреба, aWallet Password Manager може створити резервну копію даних на карті пам'яті пристрою. Варто також відзначити функції платної версії. Так при придбанні aWallet Cloud Password Manager користувач одержить можливість імпорту й експорту резервних копій через.CSV-файл, хмарну синхронізацію (Dropbox, Google Drive) і доступ до генератора паролів.

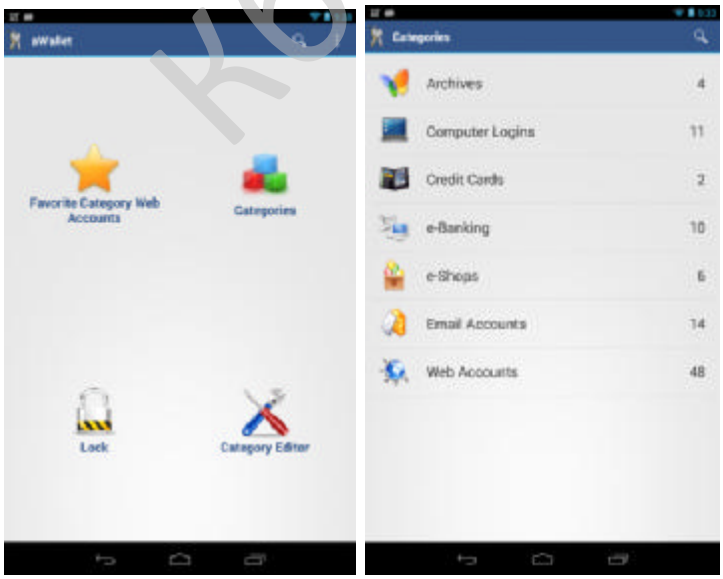

Рисунок 2.3 – Інтерфейс користувача aWallet Cloud Password Manager

Переваги:

– широкий вибір алгоритмів шифрування;

– каталогізація даних по категоріях;

– зручні шаблони даних;

– синхронізація й резервування даних (хмара, локальний носій).

Недоліки:

– хмарна синхронізація доступна тільки в повній версії.

Сейф / сейф +

Сейф – зручне сховище будь-яких даних від звичайної текстової замітки, до конкретних файлів з убудованого або зовнішнього носія пристрою. Шифрування інформації виробляється по алгоритму AES з 256-бітним ключем. Такий захист дозволить надійно сховати конфіденційну інформацію від сторонніх очей (вух і пустотливих рук). У самому додатку дані сортуються по папках і категоріям, які можна редагувати. Додатковий захист включає майстер-код, автоблокування по таймері, блокування при роботі в тлі. Повнофункціональна версія (Сейф +) містить у собі синхронізацію із хмарними сховищами (Google Drive, Яндекс і ін.), синхронізацію з PC версією, імпорт даних з інших сервісів і локальних додатків-сховищ (SafeWallet, eWallet, SPBWallet і ін.), створення незалежних карток і файлів даних, а також поліпшення інтерфейсу й спрощення використання базових функцій (графічний ключ, переглядач медіафайлів, керування файлами з контекстного меню операцій). Додатково варто відзначити повну російську локалізацію й відсутність реклами як для платної, так і для звичайної версії.

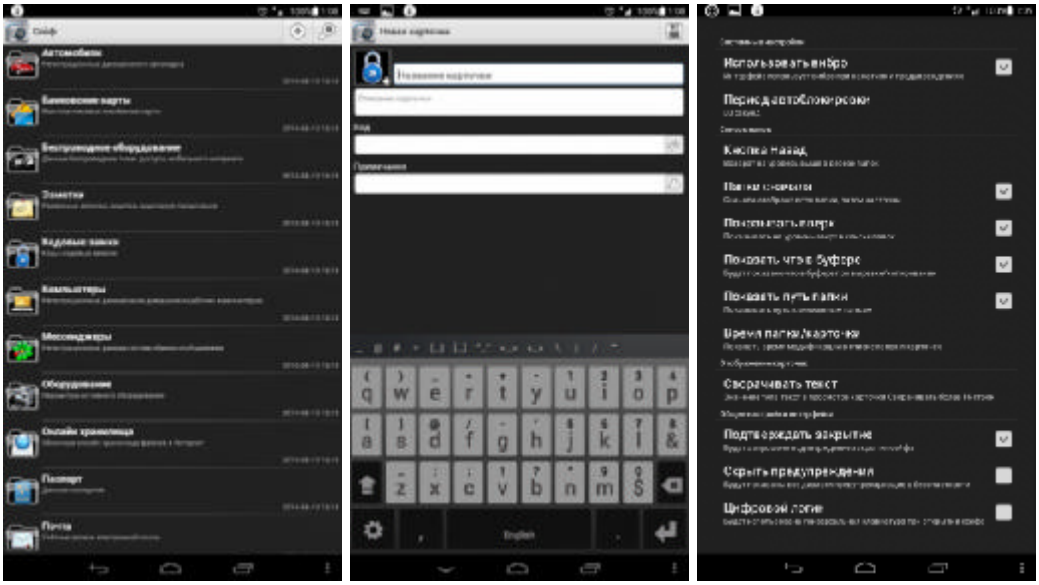

Рисунок 2.4 – Інтерфейс користувача Сейф +

Переваги:

– сортування даних по каталогах і категоріям;

– додатковий захист додатка;

– робота із хмарними сервісами.

Недоліки:

– немає об'єктивних мінусів.

Safe in cloud password manager

У цьому додатку є напевно все, що потрібно для збереження й захисту важливої інформації. Зручний інтерфейс із можливістю розподілу всіх записів по категоріях і фільтрам, гнучка система настроювання як самих категорій, так і конкретних полів у картках, 256-бітний AES алгоритм і система автоблокування з автоматичним вивантаженням із тла. Зручна система інтеграції з убудованим буфером обміну допоможе використовувати швидку вставку дані авторизації зі збереженої картки. Автор передбачив можливість резервного копіювання як через локальне збереження зашифрованих даних на носій, так і через хмарне сховище XML-файлом. Додаток підтримує синхронізацію з Google Drive, DropBox, OneDrive (SkyDrive). Окремо варто згадати кроссплатформенность Safe

In Cloud Password Manager. Даний менеджер зберігання підтримує Android, iOS, Mac і PC, при цьому, купивши мобільну версію, ви одержуєте безкоштовно доступ і до десктопному клієнта без обмежень функціональності й реклами.

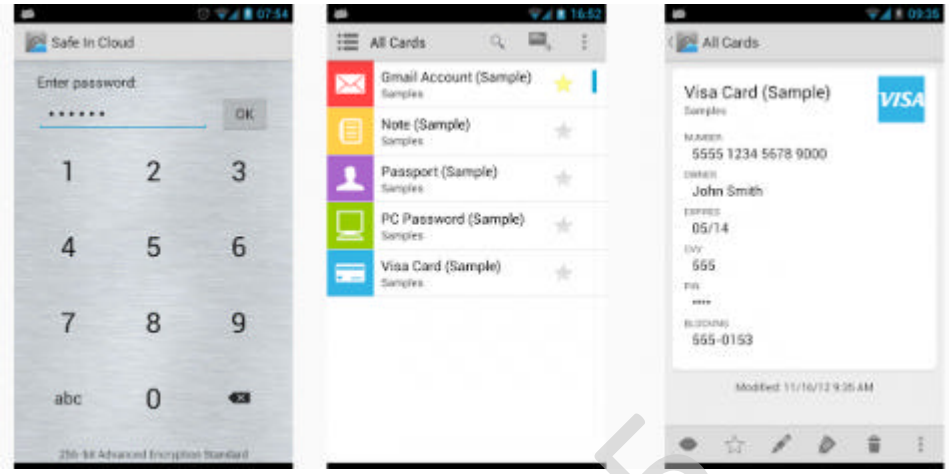

Рисунок 2.5 – Safe In Cloud Password Manager

Переваги:

– сортування даних;

– кроссплатформеність;

– додатковий захист додатка;

– безкоштовний додаток для PC;

– робота із хмарними сервісами.

Недоліки:

– немає безкоштовної (пробної) мобільної версії.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність

розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети). Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями. Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною язикових властивостей) виходять приблизно раз у два з половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython. Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android. При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть

бути написаними на Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net.

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючої коми, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

Додати новий тип можна або написавши клас (class), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиночне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень.

Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться int, long, complex і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. Кортеж у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх. Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий. Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

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 для захисту додатків та даних на мобільних корпоративних пристроях.

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

У динамічному цифровому робочому місці, де гнучкість має першочергове значення, віддалена робота стала новою нормою. Хоча ця зміна відкрила численні можливості для гнучкості та продуктивності, вона також пов’язана зі значними проблемами, зокрема щодо кібербезпеки.

Оскільки співробітники мають доступ до конфіденційних корпоративних даних із різних пристроїв і місць, захист цієї інформації став першочерговим. Ось тут і вступає в гру керування мобільними пристроями (MDM). Рішення безпеки MDM надають підприємствам інструменти для віддаленого моніторингу, захисту та керування пристроями, забезпечуючи їх захист незалежно від того, де працюють співробітники.

У цьому блозі ми дослідимо роль MDM у підвищенні кібербезпеки для віддалених команд, його основні переваги та те, як вибір правильного рішення MDM може захистити ваші корпоративні дані від потенційних загроз.

Що таке кібербезпека керування мобільними пристроями (MDM)?

Управління мобільними пристроями (MDM) у сфері кібербезпеки – це рішення, яке дозволяє IT-адміністраторам дистанційно керувати, контролювати та захищати мобільні пристрої, такі як смартфони, планшети та ноутбуки, якими користуються співробітники в бізнес-середовищі. Безпека MDM зосереджена на тому, щоб мобільні пристрої, які отримують доступ до корпоративних мереж, відповідали політикам безпеки компанії, забезпечуючи контроль над цими пристроями з центральної платформи.

У контексті кібербезпеки MDM відіграє життєво важливу роль, дозволяючи організаціям, наприклад:

– Впроваджуйте політики безпеки на всіх пристроях.

– Відстежуйте використання пристрою та доступ до даних у реальному часі.

– Застосуйте шифрування, захист паролем і багатофакторну автентифікацію.

– Дистанційно блокуйте, стирайте або блокуйте пристрої, якщо вони втрачені або зламані.

Чому кібербезпека MDM має вирішальне значення для віддалених команд?

Зростання віддаленої роботи стерло межі між особистим і професійним використанням пристроїв. Співробітники часто використовують особисті пристрої для доступу до робочої електронної пошти, документів та інших конфіденційних даних. Це ставить перед підприємствами унікальні проблеми щодо підтримки безпеки, оскільки ці пристрої можуть не мати такого ж захисту, як пристрої в традиційному офісному середовищі.

Без рішення MDM віддалені команди дуже вразливі до таких кіберзагроз, як:

– Фішингові атаки: віддалені працівники є основними цілями для фішингових шахрайств. Без моніторингу MDM важче запобігти випадковому натисканню співробітниками шкідливих посилань або завантаженню шкідливих файлів.

– Порушення даних: незахищені пристрої, які отримують доступ до мереж компанії, можуть надати конфіденційну інформацію хакерам.

– Крадіжка або втрата пристрою: віддалені співробітники, швидше за все, втратять або викрадуть свої пристрої, що збільшує ризик несанкціонованого доступу до даних компанії.

MDM надає підприємствам видимість і контроль, необхідні для вирішення цих проблем. Впроваджуючи рішення безпеки MDM, компанії можуть гарантувати, що лише авторизовані пристрої та користувачі зможуть отримати доступ до конфіденційних даних і що будь-які потенційні ризики безпеки будуть

негайно усунені. Крім того, MDM забезпечує віддалений моніторинг і здатність забезпечити відповідність усіх пристроїв у полі, що робить його важливим компонентом будь-якої стратегії віддаленої роботи.

Переваги рішень для керування мобільними пристроями для віддалених команд

Оскільки віддалена робота стає постійною частиною сучасного бізнес-ландшафту, переваги MDM для віддалених команд стають все більш очевидними. Давайте детальніше розглянемо деякі основні переваги:

1. Покращена безпека

Однією з найважливіших переваг MDM є здатність захищати віддалені пристрої від кіберзагроз. Безпека MDM дозволяє компаніям застосовувати суворі заходи безпеки, такі як шифрування та безпечні протоколи доступу, на кожному пристрої, який підключається до корпоративної мережі. Це гарантує захист конфіденційної інформації, навіть якщо пристрій зламано.

Кібербезпека MDM також включає такі функції, як віддалене стирання, що дозволяє IT-адміністраторам стерти всі дані з втраченого чи викраденого пристрою. Це зменшує ризик потрапляння конфіденційних даних у чужі руки. Крім того, інструменти моніторингу MDM можуть ідентифікувати та позначати підозрілу активність на пристроях, що дозволяє IT-командам швидко вживати заходів.

2. Централізоване керування пристроями

За допомогою MDM компанії можуть керувати та контролювати всі пристрої з однієї платформи. Цей централізований контроль полегшує застосування політик безпеки, керування оновленнями програмного забезпечення та відстеження продуктивності пристрою. IT-адміністратори можуть дистанційно блокувати або стирати пристрої, надсилати оновлення та гарантувати, що всі пристрої відповідають протоколам безпеки компанії, не потребуючи фізичного доступу до кожного пристрою.

Це централізоване керування особливо корисно для віддалених команд,

оскільки воно зменшує потребу в ручному втручанні та гарантує, що пристрої завжди оновлюються з останніми виправленнями безпеки.

3. Підвищення продуктивності

Рішення MDM надають віддаленим командам плавний доступ до корпоративних програм і даних без шкоди для безпеки. Співробітники можуть використовувати свої пристрої для доступу до робочих файлів, спільної роботи над проектами та безпечного спілкування, дотримуючись політики компанії.

Спрощуючи доступ до основних інструментів і гарантуючи, що пристрої залишаються в безпеці, MDM може підвищити продуктивність для віддалених команд, дозволяючи їм зосередитися на своїй роботі, не турбуючись про безпеку пристрою чи проблеми з відповідністю.

4. Дотримання Правил безпеки

Багато галузей мають суворі нормативні вимоги щодо безпеки даних, наприклад GDPR, HIPAA та інші. Недотримання цих правил може призвести до великих штрафів і завдати шкоди репутації компанії. Рішення MDM полегшують підприємствам дотримання цих правил, надаючи необхідні інструменти для моніторингу використання пристрою та застосування політик безпеки.

Функції безпеки MDM, такі як шифрування, віддалене стирання та контроль доступу, гарантують, що конфіденційна інформація захищена та обробляється відповідно до галузевих стандартів. Це дає компаніям спокій, знаючи, що вони виконують свої зобов’язання щодо відповідності навіть у віддаленому робочому середовищі.

5. Економічна безпека

Впровадження безпеки MDM є економічно ефективним способом захисту віддалених команд. За допомогою єдиного рішення підприємства можуть керувати та захищати всі мобільні пристрої без необхідності виконання дорогих і трудомістких ручних процесів. Знижуючи ризик витоку даних, втрати пристроїв і штрафів за недотримання вимог, MDM забезпечує значну економію коштів у довгостроковій перспективі.

Усунення ключової вразливості в рішеннях MDM

Незважаючи на широке застосування, традиційні рішення MDM мають критичну слабкість: хоча вони захищають пристрій і ідентифікаційні дані користувача, вони не можуть захистити фактичні дані після їх завантаження на мобільний пристрій. Коли співробітники отримують доступ до корпоративних мереж зі своїх телефонів, конфіденційна інформація зберігається локально – фактично це означає, що завантажені дані залишають безпеку вашого безпечного мережевого середовища. Це створює привабливу мішень для хакерів, які все більше зосереджуються на компрометації пристроїв окремих співробітників, а не на мережевій інфраструктурі.

Оскільки компанії розширюють свою віддалену робочу силу, використання платформи віртуальних мобільних пристроїв (VMD) пропонує надійне рішення для захисту корпоративних даних, надаючи віртуалізоване середовище для керування та захисту доступу мобільних пристроїв до даних.

Платформа VMD пропонує кілька ключових переваг:

– Масштабованість: Платформа VMD може підтримувати зростаючу кількість віддалених пристроїв, що робить її ідеальною для підприємств із розширенням віддалених команд.

– Безпека: віртуальні пристрої захищені розширеними функціями безпеки, включаючи шифрування та багатофакторну автентифікацію, що знижує ризик кіберзагроз.

– Гнучкість: співробітники можуть отримати доступ до платформи VMD з будь-якого місця, забезпечуючи продуктивність і зв’язок, дотримуючись протоколів безпеки компанії.

Вирішення проблем мобільної безпеки за допомогою VMD

Тож ми дізналися, що традиційні рішення MDM зосереджені на захисті окремих пристроїв, залишаючи прогалину в безпеці, коли конфіденційні дані переміщуються за межі захищеної корпоративної мережі та зберігаються на мобільних пристроях.

Щоб вирішити цю проблему, розроблено рішення MDM з нульовою довірою, яке зміщує фокус з безпеки пристрою на захист самих даних. Перетворюючи всі мобільні пристрої на безпечні віртуальні розширення корпоративної мережі, гарантуємо, що конфіденційна інформація залишається захищеною в межах периметру організації.

Підхід використовує VMD, які безпечно залишаються в мережі, дозволяючи користувачам отримувати доступ до даних через однорангову зашифровану потокову передачу. Це означає, що дані ніколи не зберігаються на зовнішніх пристроях, що зменшує ризик злому завдяки дотриманню принципу «без даних у спокої». Користувачі можуть безпечно переглядати та взаємодіяти з даними без необхідності переносити їх на свої фізичні пристрої.

Це інноваційне рішення пропонує плавний перехід до середовища нульової довіри без необхідності капітального ремонту існуючої інфраструктури. Використовуючи VMD, компанії можуть підтримувати відповідність вимогам, слідувати IT-протоколам і забезпечувати найвищий рівень безпеки, одночасно надаючи можливість безперебійній роботі віддалених команд. У світі, де співробітники та дані все частіше працюють за межами традиційних рамок, технологія VMD втілює основні принципи нульової довіри, захищаючи вашу організацію без шкоди для продуктивності.

Висновок: оптимізація безпеки керування мобільними пристроями

У сучасному мінливому середовищі віддаленої роботи забезпечення безпеки мобільних пристроїв і конфіденційних даних, до яких вони мають доступ, є надзвичайно важливим, ніж будь-коли. Кібербезпека MDM відіграє ключову роль в управлінні та захисті цих пристроїв, пропонуючи компаніям інструменти для моніторингу, захисту та оптимізації парку мобільних пристроїв.

Однак традиційні рішення безпеки, орієнтовані на пристрій, часто роблять дані вразливими, коли вони виходять за межі корпоративної мережі. Впровадивши інноваційне мобільне рішення MDM із нульовою довірою, компанії можуть усунути цю прогалину. Завдяки VMD, що забезпечує безпечний

зашифрований доступ без зберігання даних на зовнішніх пристроях, компанії можуть застосувати комплексний підхід до захисту даних, забезпечуючи безпеку конфіденційної інформації в межах периметра мережі.

3.2 Розробка структурної схеми

Системні інтегратори поступово накопичують досвід створення комплексних рішень для забезпечення корпоративної мобільності, настроювання й адаптації таких продуктів під існуючу інфраструктуру замовників і сценарії використання, технічної підтримки й навчання співробітників IT-Департаментів. При необхідності розробляються спеціалізовані продукти й додатки.

Розроблена система використовує в таких проектах системи керування мобільними пристроями (MDM) і додатками (MAM). У сполученні з корпоративними засобами захисту даних вони здатні забезпечувати необхідний рівень безпеки, причому функції MAM вендори всі частіше включають в MDM.

Існуючі рішення BYOD гарантують розмежування особистого й робітничого середовища, використання захищених комунікаційних каналів, стандартизацію й застосування політики безпеки. У випадку втрати пристрої передбачені засоби видалення з його корпоративних даних. Для реалізації особливих мір захисту, наприклад забезпечення безпеки персональних даних, розроблені сертифіковані рішення.

У випадку BYOD звичайно доводиться мати справа з мультивендорним середовищем. Зручний варіант – розгортання на всіх клієнтах Citrix XenMobile з технологією контейнеризації й можливостями керування політикою безпеки. «Цього вистачить, щоб забезпечити керівникам компанії необхідний рівень технічної підтримки й базовий захист даних на пристроях. Далі за допомогою Citrix XenMobile на мобільні пристрої співробітників доставляються програмні агенти. Така методика дозволяє привести наявну в клієнта інфраструктуру у відповідність із регламентами ІБ не прибігаючи до значних витрат».

Із всіх можливих підходів до BYOD самий інноваційного – віртуалізація мобільних додатків з розміщенням їх у ЦОД і наданням захищеного віддаленого доступу до них з будь-яких мобільних платформ.

Важлива роль приділяється автентифікації користувачів. У розробленій системі забезпечуються двох- і трифакторна автентифікація, посилена кваліфікований електронний підпис і захист від погроз недовіреного середовища. JaCarta підтримується на всіх сучасних мобільних платформах – Apple iOS, Android, Linux, Mac OS. Використання смарт-карт і токенів сприяє впорядкуванню технологій і механізмів захисту інформації для всього різноманіття пристроїв.

Застосування електронного підпису в рішенні дозволяє організувати безпечний доступ до систем і сервісів компанії з мобільних пристроїв і гарантує дійсність підписаних документів. На мобільних пристроях будуть доступні такі функції, як стругаючи взаємна двухфакторна автентифікація, формування посиленого кваліфікованого електронного підпису, безпечне зберігання ключів і цифрових сертифікатів на відчужуваному модулі безпеки (смарт-карті або токені Secure MicroSD). За допомогою розробленої системи можна надати співробітникам, партнерам і клієнтам безпечний доступ до сервісів компанії – корпоративній пошті, корпоративним ресурсам і системам усередині компанії.

Для створення повноцінного й ефективного рішення по запобіганню витоків даних з мобільних пристроїв системи MDM повинні взаємодіяти з резидентними системами DLP. Спеціалізовані рішення DLP допомагають виявити важливі для бізнесу дані й визначити правила роботи, зберігання й передачі інформації за межі компанії. Технології DLP корисні, коли потрібно швидко інтегрувати персональні мобільні пристрої в бізнес-процеси підприємства, однак для цього необхідно чітко виділити критичні для організації дані й політику доступу до них, визначити правила зберігання, переміщення й передачі, а також використання даних.

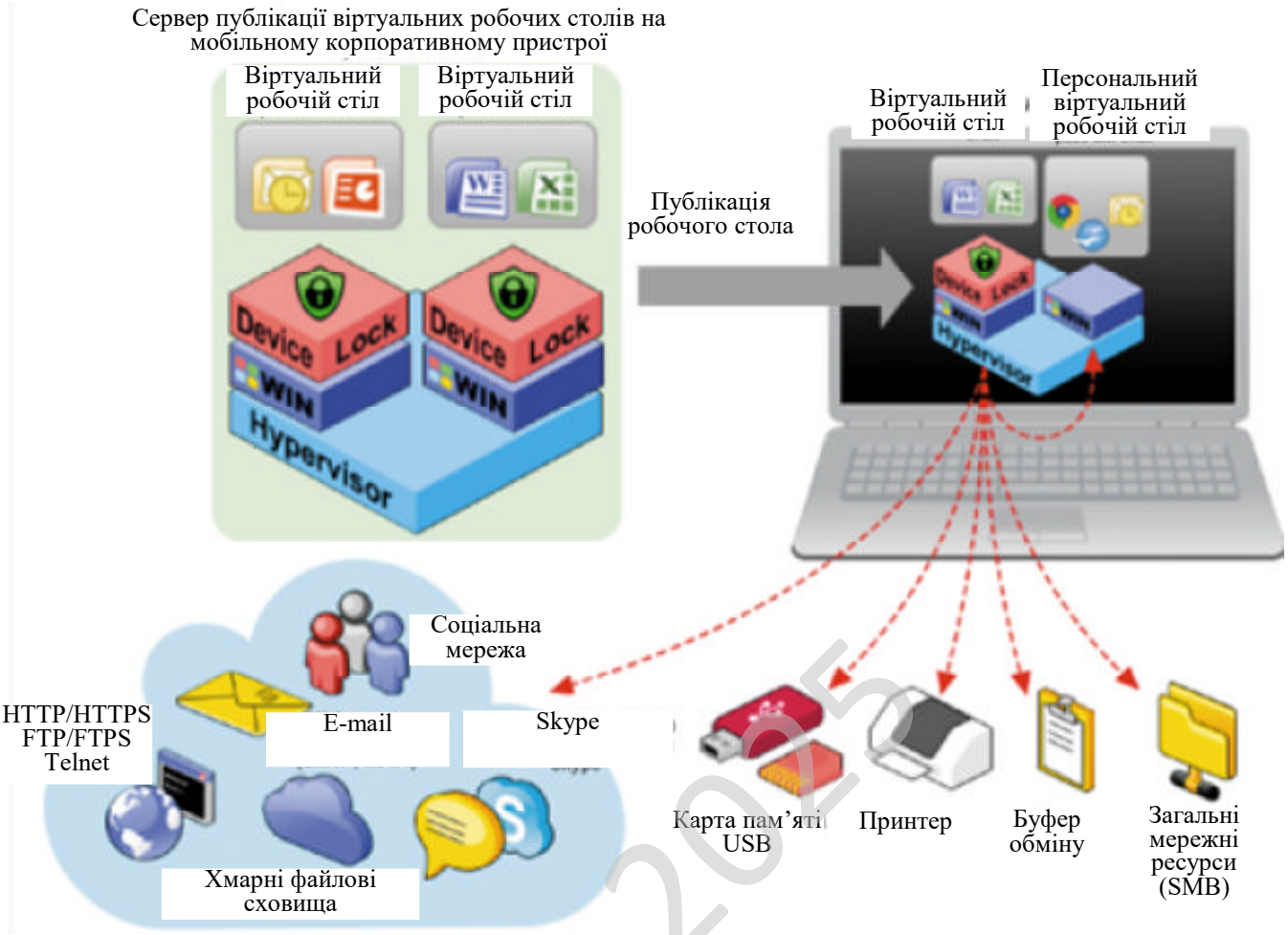

Рисунок 3.1 – Структурна схема системи

Всеосяжна й оптимальна стратегія безпеки BYOD повинна містити в собі MDM, DLP, VPN і засоби віртуалізації. Причому DLP – це система, інтегрована у віртуальне середовище Windows і яка забезпечує контроль доступних каналів передачі даних для запобігання витоків даних із пристрою BYOD.

Складність або простота наступної підтримки мобільних систем цілком залежить від правильності архітектурних рішень, прийнятих на етапі проектування. Як правило, що відповідають питання зачіпають два аспекти: технологічну платформу мобільного додатка й корпоративний мобільний шлюз. Ці рішення приймаються виходячи із цілей і пріоритетів IT-підрозділу. Вони можуть відрізнятися ергономікою для кінцевого користувача й вартістю первинної розробки.

Корпоративні мобільні шлюзи дозволяють централізовано управляти парком мобільних додатків і їх трафіком, допомагають діагностувати й усувати аварійні ситуації, що в умовах далекості користувачів надто важливо. Зараз існує безліч різновидів програмних продуктів такого класу.

Рецепти BYOD

Кожне рішення класу MDM, IAM або NAC ефективно у своїй досить вузькій ніші. При відсутності продукту з більше широкою функціональністю їх краще використовувати в комплексі.«Жодне із сімейств продуктів не вирішує «проблеми BYOD», оскільки складність полягає не в появі нової платформи, для якого потрібно забезпечити контроль, керування й безпека, а в тому, що підрозділу IT/ІБ не можуть контролювати частина інформаційної системи. Якщо ж ризики використання мобільних пристроїв розглядати в розрізі застосування технічних контрзаходів, то в першу чергу треба звернути увагу на рішення, орієнтовані на безпеку, а не на керування. Причина в тому, що погрози розвиваються стрімко, тому тільки постачальник, що спеціалізується на захисті інформації, зможе дати надію на те, що види, що з'являються, погроз будуть відбиті. Крім того, практика показує, що вкрай непросто розгорнути кілька засобів безпеки на персональних пристроях, тому варто вибрати одне, наявність якого й буде «пропуском у мережу» для кожного співробітника. Це самий прагматичний підхід до BYOD.

В BYOD необхідно збалансовано сполучати організаційні й технічні міри – проводити навчання співробітників, використовувати різні системи класу MDM і ін. Потрібна комбінація засобів, але набір рішень буде залежати від того, як і яку конфіденційну інформацію обробляє мобільний користувач на своєму пристрої. Мова йде про централізоване керування парком пристроїв в організації, про використання рішень для обмеження доступу до даних на пристрої (при його крадіжці або втраті), шифрування переданих даних (при роботі з відкритих каналів зв'язку), забезпечення довіри до створеного на ньому документам і зробленим операціям (завдяки строгій автентифікації й електронному підпису).

Стратегія BYOD залежить від безлічі факторів і внутрішніх корпоративних правил, але, узагальнюючи накопичений досвід, можна говорити про необхідність поділу особистого й робочого простору на мобільному пристрої. Така стратегія дозволить зберегти зручність і ергономіку мобільних пристроїв, використовуваних для особистих цілей, і забезпечити захист корпоративних даних. Завдяки даному підходу можна домогтися підвищення продуктивності праці співробітників за рахунок віддаленої, але комфортної роботи. Однак не всі мобільні платформи дозволяють організувати такий підхід без застосування спеціалізованих програмних і інструментальних засобів.

Різні технології керування й забезпечення безпеки реалізують концепції BYOD у різному ступені. У першу чергу варто визначити, яку мету переслідують IT-підрозділу, що бажають впровадити у своїх організаціях BYOD, і виходячи із цього вибрати необхідні інструментальні засоби. Як правило, насамперед треба подбати про захист корпоративних даних, оброблюваних у зв'язуванні мобільний пристрій – бізнес-системи: мова йде про безліч компонентів, включаючи канали зв'язку, сховище даних на мобільному пристрої, підсистеми автентифікації/авторизації й навіть самі бізнес-системи.

На жаль, єдиного рецепта для BYOD немає. Вирішити дане завдання дозволить тільки комплексний підхід із застосуванням великої кількості інструментів і технічних-організаційно-технічних мір, що забезпечить конфіденційність і цілісність оброблюваних даних.

3.3 Розробка функціональної схеми

На рисунку 3.2 представлена функціональна схема системи у вигляді процесу перевірки дійсності 802.1X. 802.1x – це затверджений IEEE стандарт контролю доступу до середовища передачі, що дозволяє дозволити або заборонити доступ до мережі, контролювати доступ до віртуальних локальних

мереж, застосовувати політики керування трафіком на основі ідентифікаторів користувача або машини.

WPA і **WPA2** (**Wi-Fi Protected Access**) – являє собою оновлену програму сертифікації пристроїв бездротового зв'язку. Технологія WPA прийшла на заміну технології захисту бездротових мереж WEP. Плюсами WPA є посилена безпека даних і більш жорсткий контроль доступу до бездротових мереж. Немаловажною характеристикою є сумісність між безліччю бездротових пристроїв як на апаратному рівні, так і на програмному. На даний момент WPA і WPA2 розробляються й просуваються організацією Wi-Fi Alliance.

Основні поняття

В WPA забезпечена підтримка стандартів 802.1X, а так само протоколу EAP (Extensible Authentication Protocol, розширюваний протокол автентифікації). Варто помітити, що в WPA підтримується шифрування у відповідності зі стандартом AES (Advanced Encryption Standard, удосконалений стандарт шифрування), що має ряд переваг над використовуваним в WEP RC4, наприклад набагато більше стійкий криптоалгоритм.

Великим плюсом при впровадженні WPA є можливість роботи технології на існуючому апаратному забезпеченні Wi-Fi.

Деякі відмінні риси WPA:

– удосконалена схема шифрування RC4;

– обов'язкова автентифікація з використанням EAP;

– система централізованого керування безпекою, можливість використання в діючих корпоративних політиках безпеки.

Автентифікація користувачів

Wi-Fi Alliance дає наступну формулу для визначення суті WPA:

$$WPA = 802.1X + EAP + TKIP + MIC.$$

Видно, що WPA, по суті, є сумою декількох технологій.

Як згадано вище, у стандарті WPA використовується Розширюваний протокол автентифікації (EAP) як основа для механізму автентифікації

користувачів. Неодмінною умовою автентифікації є, пред'явлення користувачем свідчення (інакше називають мандатом), що підтверджує його право на доступ у мережу. Для цього права користувач проходить перевірку за спеціальною базою зареєстрованих користувачів. Без автентифікації робота в мережі для користувача буде заборонена.

База зареєстрованих користувачів і система перевірки в великих мережах найчастіше розташовані на спеціальному сервері (найчастіше RADIUS). Але слід зазначити, що WPA має спрощений режим.

Цей режим одержав назву Pre-Shared Key (WPA-PSK). При застосуванні режиму PSK необхідно ввести один пароль для кожного окремого вузла бездротової мережі (бездротові маршрутизатори, точки доступу, мости, клієнтські адаптери). Якщо паролі збігаються із записами в базі, користувач одержить дозвіл на доступ у мережу.

Шифрування

Навіть не приймаючи в уваги той факт що WEP, попередник WPA, не має які-небудь механізми автентифікації користувачів як такі, його ненадійність складається, насамперед, у криптографічній слабості алгоритму шифрування. Ключова проблема WEP полягає у використанні занадто схожих ключів для різних пакетів даних.

TKIP, MIC і 802.1X (частини рівняння WPA) внесли свою лепту в посилення шифрування даних мереж, що використовують WPA.

TKIP відповідає за збільшення розміру ключа з 40 до 128 біт, а також за заміну одного статичного ключа WEP ключами, які автоматично генеруються й розсилаються сервером автентифікації. Крім того, в TKIP використовується спеціальна ієрархія ключів і методологія керування ключами, що забирає зайву передбачуваність, що використовувалася для несанкціонованого зняття захисту WEP ключів.

Сервер автентифікації, після одержання сертифіката від користувача, використовує 802.1X для генерації унікального базового ключа для сеансу

зв'язку. TKIP здійснює передачу згенерованого ключа користувачеві й точки доступу, після чого вишиковує ієрархію ключів плюс систему керування. Для цього використовується двосторонній ключ для динамічної генерації ключів шифрування даних, які у свою чергу використовуються для шифрування кожного пакета даних. Подібна ієрархія ключів TKIP заміняє один ключ WEP (статичний) на 500 мільярдів можливих ключів, які будуть використані для шифрування даного пакета даних.

Іншим важливим механізмом є перевірка цілісності повідомлень (Message Integrity Check, MIC). Її використовують для запобігання перехоплення пакетів даних, зміст яких може бути змінено, а модифікований пакет знову переданий по мережі. MIC побудована на основі потужної математичної функції, що застосовується на стороні відправника й одержувача, після чого рівняється результат. Якщо перевірка показує на розбіжність результатів обчислень, дані вважаються помилковими й пакет відкидається.

При цьому механізми шифрування, які використовуються для WPA і WPA-PSK, є ідентичними. Єдина відмінність WPA-PSK полягає в тому, що автентифікація відбувається з використанням пароля, а не по сертифікату користувача.

WPA2

WPA2 визначається стандартом IEEE 802.11i і покликаний замінити WPA. У ньому реалізоване CCMP і шифрування AES, за рахунок чого WPA2 став більше захищеним, ніж свій попередник.

Уразливість

Необхідно помітити, що з'єднання WPA, що використовують більше захищений стандарт шифрування ключа AES, а також WPA 2-з'єднання не піддані атакам на алгоритм шифрування.

Тепер обрій бездротової безпеки придбав чіткі обриси, давши IT-адміністраторам упевненість, необхідну для розгортання мереж WLAN. На сьогоднішній день у рамках концепції Cisco Unified Wireless Network компанія

Cisco пропонує систему безпеки для корпоративних мереж WLAN, що надає наступні можливості для бездротових продуктів Cisco, продуктів Cisco Aironet® і сумісних з Cisco клієнтських пристроїв WLAN.

– Підтримка стандарту IEEE 802.11i.

– Підтримка сертифікатів безпеки Wi-Fi Alliance – Wi-Fi Protected Access (WPA) і Wi-Fi Protected Access 2 (WPA2).

– Потужна двостороння автентифікація й керування динамічними ключами шифрування завдяки підтримці IEEE 802.1X.

– Шифрування даних за допомогою алгоритмів Advanced Encryption Standard (AES) або Temporal Key Integrity Protocol (TKIP).

– Сама широка на ринку підтримка типів автентифікації 802.1X, клієнтських пристроїв і клієнтських операційних систем.

– Придушення активних і пасивних мережних атак.

– Інтеграція з рішеннями Cisco Self-Defending Network і Network Admission Control (NAC).

– Можливості системи запобігання мережних вторгнень (Intrusion Prevention System, IPS) і спостереження за переміщенням абонента – прозоре подання мережі в реальному часі.

– Конвергенція безпеки внутрішньої й зовнішньої мереж Wi-Fi завдяки рішенню для повнозв'язаної бездротової мережі Cisco.

Компанія Cisco, лідер і рушійна сила розвитку бездротових мереж, уже дозволила мережним адміністраторам дати користувачам той ступінь волі, що вони заслуговують – без шкоди для мережної безпеки, що їм необхідна.

Мережним адміністраторам необхідні мережі WLAN, здатні забезпечити той же рівень захищеності, масштабованості, надійності, зручності експлуатації й керування, що вони звикли очікувати від діючих провідних локальних мереж. Перевірка політики безпеки повинна вироблятися на регулярній основі. Рішення мережної безпеки повинні з легкістю розгортатися як на декількох точках доступу, так і на сотнях і навіть тисячах. Необхідно проводити заходи щодо

виявлення несанкціонованих точок доступу, установлюваних співробітниками або зловмисниками.

Cisco Unified Wireless Network являє собою корпоративне рішення, що відповідає стандартам, WLAN для забезпечення безпеки, що дає мережним адміністраторам упевненість у тім, що дані залишаться конфіденційними й захищеними при використанні бездротових продуктів Cisco, продуктів серії Cisco Aironet, сумісних розширень Cisco (Compatible Extensions), а також клієнтських пристроїв WLAN, сертифікованих на відповідність вимогам Wi-Fi. Це рішення бездротової безпеки класу підприємства підтримує надійні сервіси безпеки бездротових локальних мереж, строго ідентичні мірам безпеки, доступним для провідних локальних мереж. Рішення повністю задовольняє вимоги до сумісної, надійної й захищеної мобільної мережі за рахунок надання провідних рішень в області безпеки мереж WLAN. Рішення зм'якшує ризик сучасних пасивних і активних атак на бездротову локальну мережу, взаємодіє з рядом клієнтських пристроїв і надає надійні, масштабовані й централізовані заходи щодо забезпечення й керуванню безпекою. Cisco Unified Wireless Network дозволяє мережним адміністраторам розгортати широкомасштабні корпоративні бездротові локальні мережі з масштабованим і простим адмініструванням безпеки, не ускладнюючи при цьому роботу IT-співробітників.

Cisco Unified Wireless Network надає різноманітні сучасні вдосконалення Cisco і підтримує Wi-Fi Protected Access (WPA) і Wi-Fi Protected Access 2 (WPA2), що забезпечує керування доступом за допомогою двосторонньої автентифікації кожного користувача й кожної сесії, а також конфіденційність даних завдяки потужним алгоритмам динамічного шифрування. Для підтримки різноманітних асортиментів корпоративних додатків у дане рішення інтегровані можливості керування якістю обслуговування (QoS) і мобільності.

Cisco Unified Wireless Network надає:

– Безпечне підключення до мереж WLAN – потужні динамічні ключі шифрування автоматично змінюються відповідно до настроювань для забезпечення конфіденційності переданих даних.

– Шифрування WPA-TKIP розширюється такими функціями, як перевірка цілісності повідомлення (MIC), хешування ключів (для кожного пакета), зміна вектора ініціалізації (IV) і ротація широкомовних ключів.

– WPA 2-AES.

– Довірчі відносини й ідентифікацію в мережах WLAN. Надійне керування доступом до WLAN допомагає забезпечити підключення вповноважених клієнтів тільки до довірених точок доступу, крім неавторизованих точок доступу. Це забезпечується за допомогою двосторонньої автентифікації кожного користувача й кожної сесії із застосуванням IEEE 802.1X, різноманітних типів розширюваного протоколу автентифікації (Extensible Authentication Protocol, EAP) і сервера автентифікації RADIUS (Remote Authentication Dial-In User Service) або сервера автентифікації, авторизації й обліку AAA (Authentication, Authorization and Accounting).

– Саму широку на ринку підтримку типів автентифікації 802.1X, клієнтських пристроїв і клієнтських операційних систем.

– Підтримку записів біллінгу протоколу RADIUS для всіх спроб автентифікації.

– Захист від атак на мережі WLAN. Виявлення несанкціонованого доступу, мережних атак і несанкціонованих точок доступу за допомогою надійних засобів запобігання вторгнень IPS, WLAN NAC і розширених сервісів виявлення місця розташування. Cisco надає засоби запобігання мережних вторгнень (IPS) класу підприємства, що дозволяють IT-менеджерам безупинно сканувати радіодіапазон, виявляти несанкціоновані точки доступу та інші несанкціоновані події, при цьому одночасно відслідковуючи тисячі пристроїв і зм'якшуючи можливість мережних атак. Протокол NAC, Network Admission Control, був спеціально розроблений для того, щоб допомогти забезпечити адекватний захист всіх провідних і бездротових кінцевих пристроїв (таких як персональні комп'ютери, ноутбуки, сервери й КПК), що звертаються до мережних ресурсів, від погроз безпеки. Використання протоколу NAC дозволяє

організаціям аналізувати й контролювати всі пристрої, що підключаються до мережі.

Cisco Unified Wireless Network – це єдине в галузі рішення, що сполучить у собі провідні й бездротові компоненти, що дозволяє підприємствам без зайвих витрат вирішувати питання, пов'язані із забезпеченням безпеки WLAN, її розгортання й керування. Поєднуючи в собі кращі якості провідних і бездротових мереж, це потужне рішення дозволяє створювати масштабовані, керовані й захищені мережі WLAN з низькою сукупною вартістю володіння. У ньому реалізовані інноваційні радіочастотні можливості, які дозволяють забезпечити доступ у реальному часі до ключового бізнес-додаткам і організувати захищені підключення до мережі корпоративного класу. Cisco Unified Wireless Network являє собою інтегроване, всеосяжне рішення, що охоплює всі рівні WLAN, починаючи від клієнтських пристроїв і точок доступу й закінчуючи мережною інфраструктурою, інструментами мережного адміністрування, засобами інтеграції сучасних бездротових сервісів.

Підтримка WPA і WPA2

Cisco Unified Wireless Network включає підтримку сертифікованих Альянсом Wi-Fi механізмів WPA і WPA2. Всі продукти, сертифіковані Wi-Fi на відповідність вимогам WPA2, обов'язково можуть взаємодіяти із продуктами, сертифікованими Wi-Fi на відповідність вимогам WPA.

WPA і WPA2 надають кінцевим користувачам і мережним адміністраторам високий рівень упевненості в тім, що їхні дані залишаться конфіденційними, а доступ до їхніх мереж буде надаватися тільки санкціонованим користувачам. Обидва стандарти мають персональний і корпоративний режими роботи, що відповідають окремим вимогам цих двох сегментів ринку. Корпоративний режим використовує для автентифікації IEEE 802.1X і EAP. Персональний режим кожного використовує для автентифікації загальні ключі (PSK). Cisco не рекомендує застосовувати персональний режим при розгортанні комерційних або державних рішень через використання

загальних PSK-ключів при автентифікації користувачів. PSK-ключі не вважаються досить надійною мірою для впровадження на підприємстві.

WPA дозволяє закрити всі відомі уразливості WEP вихідного стандарту безпеки IEEE 802.11 і являє собою швидке рішення для забезпечення безпеки мереж WLAN як для підприємств, так і для невеликих компаній або домашніх систем. WPA використовує алгоритм шифрування TKIP.

WPA2 – це наступне покоління безпеки Wi-Fi. WPA2 являє собою запропонований Wi-Fi Alliance варіант ратифікованого стандарту IEEE 802.11i. У його состав входить рекомендований Національним інститутом стандартів і технологій (NIST) алгоритм шифрування AES, що використовує протокол CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol). WPA2 забезпечує відповідність вимогам урядового стандарту FIPS 140-2.

WPA-шифрування – Temporal Key Integrity Protocol, протокол цілісності тимчасових ключів

Cisco Unified Wireless Network підтримує протокол TKIP, одну зі складових WPA і стандарту IEEE 802.11i. TKIP являє собою наступне покоління стандарту забезпечення безпеки WEP. Як і WEP, TKIP використовує метод шифрування, розроблений інженером Роном Райвестом і відомий як алгоритм шифрування Ron's Code 4 (RC4). Однак TKIP поліпшує WEP за рахунок ліквідації відомих уразливостей WEP і додавання таких функцій, як хешування ключа кожного пакета, MIC і ротація широкомовних ключів.

TKIP реалізує RC4-кодування потоку 128-бітними ключами для шифрування й 64-бітними ключами для автентифікації. За рахунок шифрування даних ключем, що може бути використаний тільки запропонованим користувачем цих даних, TKIP дозволяє гарантувати одержання переданих даних у відкритому виді тільки тими, для кого вони призначені. Шифрування TKIP приводить до 280 трильйонів можливих комбінацій ключів для кожного окремого пакета даних.

У рамках Cisco Unified Wireless Network реалізовані алгоритми Cisco TKIP і WPA TKIP для автономних точок доступу Cisco Aironet, пристроїв Cisco Aironet

і сумісних з Cisco клієнтських пристроїв для роботи з бездротовою локальною мережею. Незважаючи на те, що Cisco TKIP і WPA TKIP не можуть взаємодіяти один з одним, автономні точки доступу серії Cisco Aironet можуть працювати одночасно в цих режимах при використанні декількох VLAN. Системним адміністраторам потрібно вибрати один набір TKIP-алгоритмів для активації на клієнтських пристроях підприємства, оскільки клієнти не можуть підтримувати обидва набори TKIP-алгоритмів одночасно. Cisco рекомендує по можливості використовувати для клієнтських пристроїв і точок доступу алгоритм WPA TKIP. Контролери бездротової локальної мережі Cisco і прості точки доступу Cisco Aironet підтримують тільки WPA TKIP. Хешування ключів для кожного пакета з метою зниження ризику атак типу "Слабкий вектор ініціалізації" ("Weak IV").

При використанні WEP-ключа для (де-)шифрування переданих даних кожний пакет включає вектор ініціалізації (IV), що представляє собою 24-бітне поле, що міняється з кожним пакетом. Алгоритм відновлення ключів TKIP RC4 генерує вектор на базі основного WEP-ключа. Уразливість у реалізації WEP-алгоритму RC4 дозволяє створювати "слабкі" вектори, що дають можливість злому основного ключа. За допомогою таких інструментів, як AirSnort, зломщик може скористатися даною уразливістю шляхом збору пакетів, зашифрованих одним ключем і підстановки слабких векторів ініціалізації для знаходження основного ключа. TKIP містить засоби хешування ключів, або створення ключів для кожного пакета, з метою зниження ризику атак з використанням слабких векторів ініціалізації. При впровадженні підтримки хешування ключів як на точці доступу, так і на всіх асоційованих клієнтських пристроях, відправник даних хешує базовий ключ за допомогою вектора ініціалізації для створення нового ключа для кожного пакета. Забезпечуючи шифрування кожного пакета своїм ключем, хешування ключа знімає ймовірність визначення WEP-ключа за допомогою уразливості векторів ініціалізації.

Message Integrity Check, перевірка цілісності повідомлення для захисту від активних мережних атак

Використання MIC дозволяє уникнути активних мережних атак, націлених на пошук ключа шифрування, застосовуваного для шифрування перехоплених пакетів. Активні атаки є комбінацією атаки методом "жонглювання бітами" і злому захисту шляхом заміщення оригіналу. При впровадженні MIC як на точці доступу, так і на всіх асоційованих клієнтських пристроях відправник пакета даних додає трохи байт (для перевірки цілісності повідомлення) до пакета перед його шифруванням і відправленням. При одержанні пакета одержувач дешифрує його й перевіряє байти MIC. Якщо байти MIC пакета відповідають розрахунковим даним (розраховується з функції MIC), одержувач приймає пакет; у противному випадку одержувач знищує пакет. За допомогою MIC стає можливим відкидати пакети, змінені зловмисниками. Зломщики не можуть скористатися ні атакою методом "жонглювання бітами", ні активним зломом захисту шляхом заміщення оригіналу для обману мережі з метою автентифікації, оскільки продукти Cisco Aironet оснащені MIC-захистом і здатні ідентифікувати й знищувати змінені пакети.

Ротація широкомовних ключів

TKIP дозволяє мережним адміністраторам здійснювати ротацію як привласнених конкретному пристрою, так і широкомовних ключів, використовуваних для шифрування широкомовних повідомлень. Мережні адміністратори можуть конфігурувати політики ротації широкомовних ключів для точок доступу. Оскільки статичний широкомовний ключ піддається тим же атакам, що й привласнені конкретному пристрою або статичні WEP-ключі, підтримується ротація значень ключа для широкомовних ключів, що дозволяє закрити цю уразливість.

WPA 2-шифрування – Advanced Encryption Standard, поліпшений стандарт шифрування

Cisco Unified Wireless Network підтримує WPA2, що використовує з метою забезпечення конфіденційності й цілісності даних алгоритм шифрування AES.

AES – алгоритм шифрування, використовуваний як альтернатива стандарту RC4 протоколів TKIP і WEP. AES не підданий відомим атакам і пропонує більше високий рівень шифрування, чим TKIP і WEP. AES – украй захищений криптографічний алгоритм – поточні дослідження показують, що для взлому AES-ключа потрібно 2^120 операцій, і це вище сучасних можливостей.

AES являє собою алгоритм блокового кодування, що є одним з методів кодування симетричним ключем, який використовує той самий ключ як для шифрування, так і для дешифрування. Алгоритм використовує групи біт фіксованої довжини, називані блоками. На відміну від WEP, що використовує потік ключів для шифрування вхідного потоку текстових даних, AES шифрує біти в блоках тексту, які розраховуються незалежно друг від друга. Стандарт AES визначає розмір блоку AES, рівним 128 біт, і дозволяє вибрати один із трьох варіантів довжини ключа – 128, 192 і 256 біт. Ключі довжиною 128 біт використовуються для WPA2/802.11i. Один цикл алгоритму WPA2/802.11i AES містить у собі до чотирьох проходів шифрування. Для алгоритму WPA2/802.11i кожний цикл повторюється десятикратно.

Для забезпечення як конфіденційності даних, так і дійсності з алгоритмом AES використовується новий режим складання даних, називаний Counter-Mode/CBC-Mac (CCM). CCM використовує AES у режимі лічильника (Counter mode, CTR) для забезпечення конфіденційності даних, а AES використовує Cipher Block Chaining Message Authentication Code (CBC-MAC) для забезпечення цілісності даних. Даний тип конструкції, що використовує один ключ для двох режимів (CTR і CBC-MAC) є "новою" конструкцією, схваленої NIST (спеціальна публікація 800-38C) і співтовариством формування стандартів (IETF RFC-3610).

Для CCM застосовується 48-бітний вектор ініціалізації. Як і TKIP, AES використовує вектор ініціалізації способом, відмінним від методів шифрування WEP. Для CCM вектор ініціалізації використовується в якості вхідних даних для процесів шифрування й дешифрування з метою зниження атак шляхом заміщення оригіналу. Також, через вектор ініціалізації, розширеного до 48 біт, час, що

вимагається для колізії вектора ініціалізації, збільшується експоненційно, забезпечуючи більше високий рівень захисту даних.

Рекомендується використовувати апаратні можливості (де-)шифрування по алгоритму AES у зв'язку з підвищеним обчислювальним навантаженням, викликаним обробкою AES. Бездротові продукти Cisco використовують апаратне шифрування по алгоритму AES. Шифрування по алгоритму AES програмними засобами для декількох клієнтів одночасно приводить до підвищених вимог до апаратного забезпечення, наприклад, наявності процесора рівня Pentium 2,5 Ггц для ноутбука. Якщо точка доступу виконує (де-)шифрування по алгоритму AES програмними засобами й обслуговує безліч підключених клієнтів, швидше за все, точка доступу зштовхнеться з падінням продуктивності – особливо, якщо точка доступу не обладнана потужним процесором і великим обсягом оперативної й постійної пам'яті.

Розгортання WPA і WPA2

Cisco рекомендує замовникам використовувати WPA2 для клієнтських пристроїв, що надають підтримку WPA2. Незважаючи на те, що WPA дотепер вважається надійним методом, а TKIP дотепер не був зламаний, Cisco рекомендує замовникам по можливості переходити на WPA2. Оскільки WPA2 вимагає внесення конфігураційних змін як для точки доступу, так і для клієнтських пристроїв, розгортання WPA2 повинне плануватися заздалегідь. Рекомендується переводити на новий алгоритм по можливості більшу кількість клієнтських пристроїв і точок доступу одночасно для мінімізації простоїв мережі. Зручним часом для розгортання WPA2 є періоди розгортання, відновлення й розширення бездротової мережі.

З метою спрощення переходу на WPA і WPA2 автономні точки доступу Cisco Aironet підтримують як режим WPA-міграції (WPA Migration Mode), так і змішаний режим WPA2 (WPA2 Mixed Mode). Режим WPA-міграції – це автономне настроювання точки доступу Cisco і що дозволяє як WPA-, так і не WPA-клієнтам асоціюватися із точкою доступу з використанням того самого

ідентифікатора SSID. Режим WPA-міграції повинен бути використаний у якості тимчасового перехідного заходу щодо причини автентифікації WEP-клієнтів і, відповідно, зниженого рівня безпеки. Змішаний режим WPA2 дозволяє WPA– і WPA 2-клієнтам співіснувати на загальному SSID. Змішаний режим WPA2 є сертифікованої Wi-Fi функцією. Змішаний режим WPA2 уважається надійним, тому що використовує для шифрування як TKIP, так і AES.

Спеціалізовані клієнтські пристрої WLAN можуть не підтримувати AES і не підтримувати можливість відновлення до AES (і WPA2). Із цієї причини Cisco рекомендує підприємствам продовжувати використовувати й розгортати WPA для таких пристроїв, якщо це доречно. Всі мережі повинні забезпечувати як мінімум WPA-захист.

AP – точка доступу.

Процес перевірки виконується в такий спосіб:

– Клієнт формує відповідність 802.11 і AP.

– AP посилає клієнтові “ідентифікаційний запит” EAP.

– Клієнт відповідає, посилаючи свої “облікові дані” EAP до AP. Облікові дані складаються з імені користувача й домену.

– AP пересилає даний запит на сервер RADIUS через неконтрольований порт.

– Після одержання “облікових даних” EAP клієнта сервер RADIUS запитує сертифікат користувача домену, що відповідає тільки що отриманим обліковим даним, і посилає сертифікат сервера клієнтові.

– AP пересилає запит сертифіката клієнтові.

– Клієнт підтверджує отриманий сертифікат сервера й посилає свій сертифікат користувача домену назад до AP. (Примітка: клієнтові необхідно попередньо одержати сертифікат через мережне з'єднання Ethernet або якими-небудь іншими засобами підключення, що не використовують 802.1x.)

– AP пересилає сертифікат серверу RADIUS.

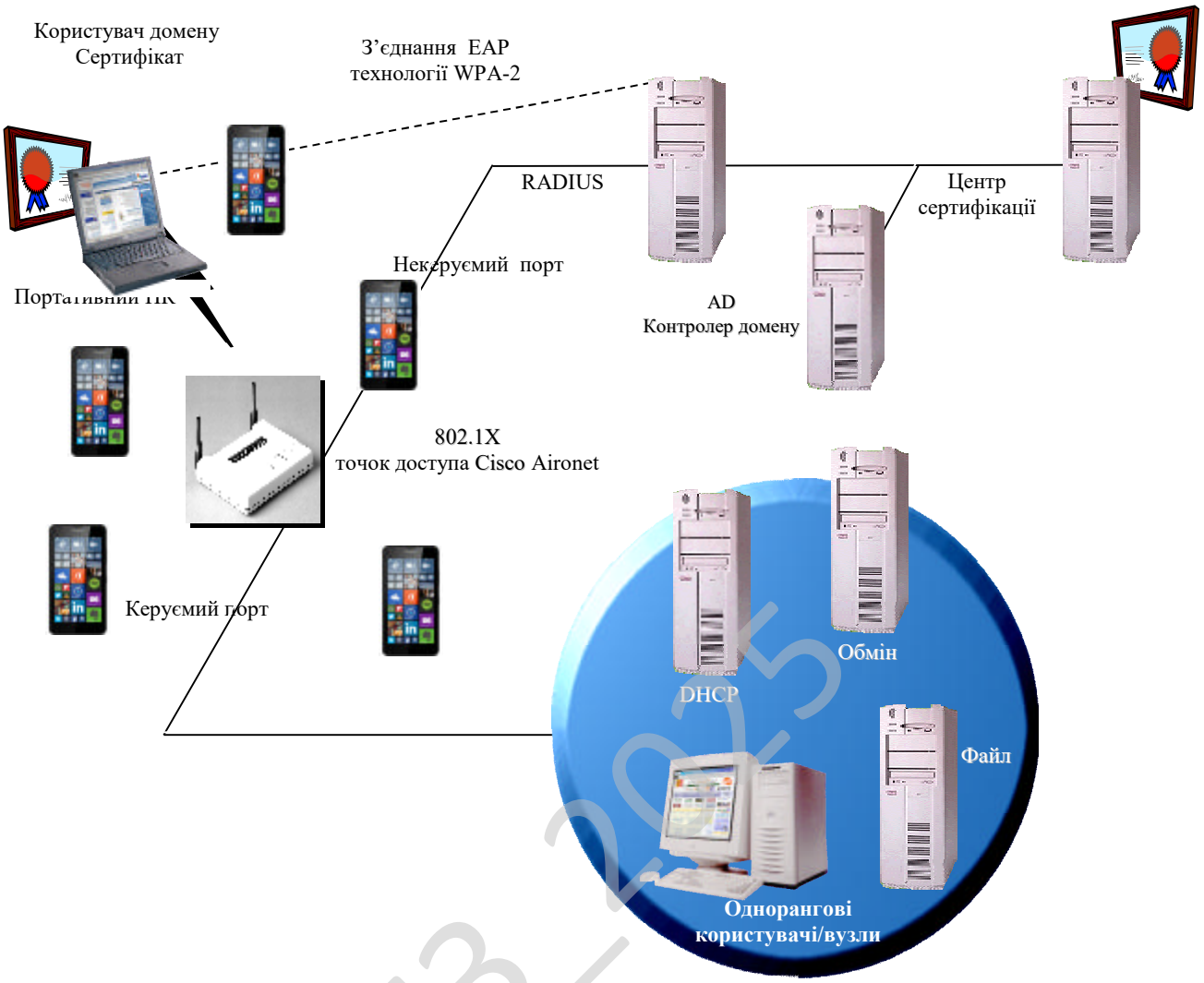

Рисунок 3.2 – Функціональна схема системи

– Сервер RADIUS перевіряє його на відповідність за допомогою контролера домену Active Directory і Центра сертифікації, щоб гарантувати, що інформація облікового запису користувача домену в пакеті "облікових даних" відповідає сертифікату користувача домену, отриманому від клієнта. Якщо такої відповідності ні, сервер RADIUS посилає AP повідомлення про помилку перевірки дійсності.

– При успішній перевірці дійсності сервер RADIUS посилає AP повідомлення про успішну перевірку дійсності разом із придатним для використання ключем WPA-2 для сеансу.

– AP пересилає ключ WPA-2 клієнтові. Даний ключ WPA-2 використовується протягом усього сеансу, під час якого клієнт зіставлений AP.

– AP відкриває контрольований порт, щоб надати клієнтові доступ до мережних ресурсів.

– Клієнт використовує ключ WPA-2 для шифрування безпечного з'єднання 802.11 з AP і запускає DHCP, щоб одержати припустиму IP-адресу.

– Після успішного одержання IP-адреси із сервера DHCP клієнт виконує звичайний вхід у домен і може починати користуватися мережею.

DHCP – Dynamic Host Configuration Protocol – протокол динамічної конфігурації вузла) – це мережний протокол, що дозволяє комп'ютерам автоматично одержувати IP-адресу й інші параметри, необхідні для роботи в мережі TCP/IP. Даний протокол працює по моделі «клієнт-сервер». Для автоматичної конфігурації комп'ютер-клієнт на етапі конфігурації мережного пристрою звертається до т.зв. сервера DHCP, і одержує від нього потрібні параметри. Мережний адміністратор може задати діапазон адрес, що розподіляються сервером серед комп'ютерів. Це дозволяє уникнути ручного настроювання комп'ютерів мережі й зменшує кількість помилок. Протокол DHCP використовується в більшості великих (і не дуже) мереж TCP/IP.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним

спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

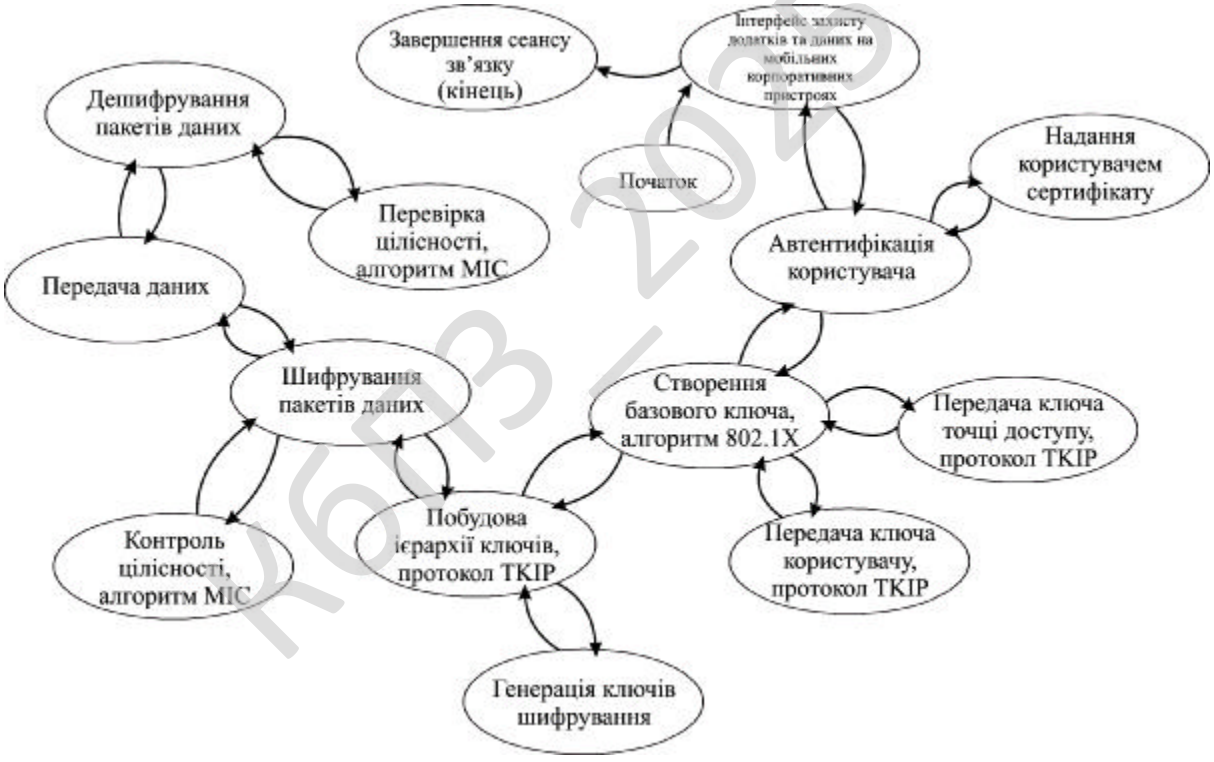

Рисунок 3.3 – Діаграма взаємодії процесів

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

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

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

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

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захисту додатків та даних на мобільних корпоративних пристроях.

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

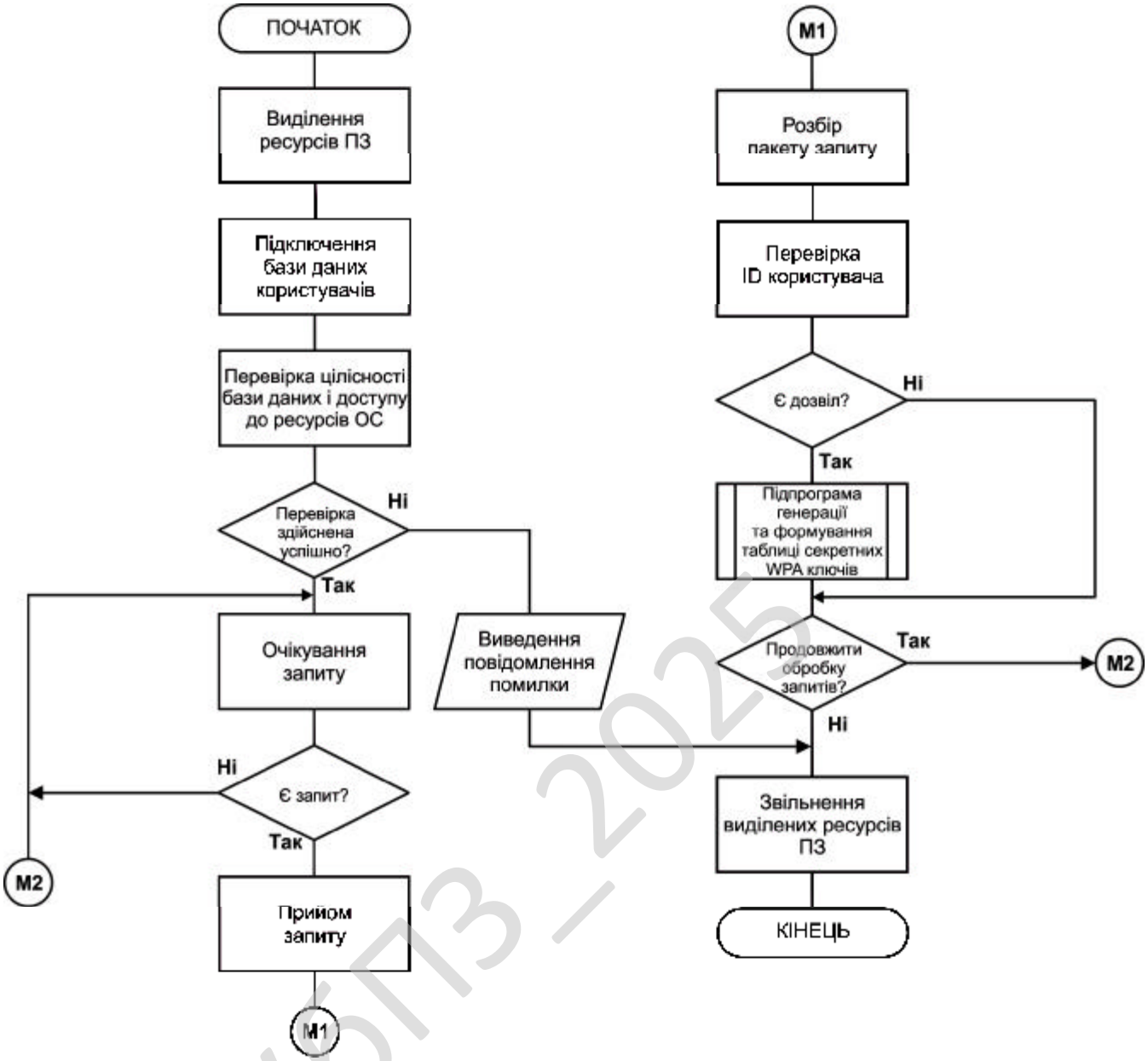

Рисунок 4.1 – Блок-схема основної програми

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

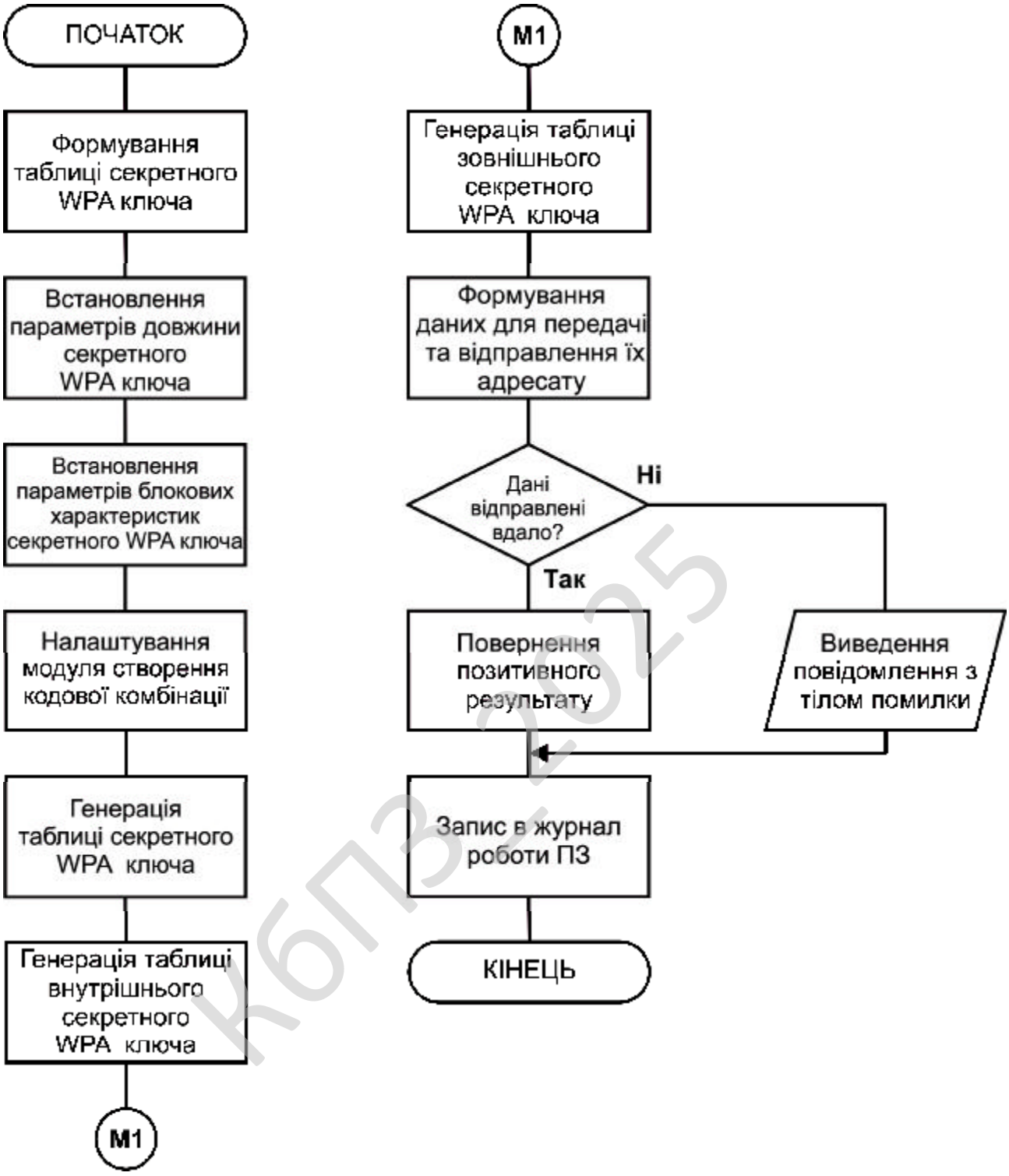

Рисунок 4.2 – Блок-схема роботи підпрограми

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому

місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Розглянемо реалізацію методу обробки запитів користувачів. Для реалізації задачі обробки запитів від кінцевих користувачів необхідно використовувати мережні компоненти.

Розглянемо використаний компонент TNMHTTP (NetMasters HTTP), який знаходиться на вкладці FastNet палітри компонентів Delphi та розглянемо коди підпрограм які дозволяють реалізувати поставлені задачі (винесені у різні програми для зменшення коду програми):

```
{... ПОЧАТОК.
Йде заголовок файлу і визначення форми (TForm1) і її екземпляра Form1.
У формі є кнопка TButton і одне поле TEdit.
При натисненні на кнопку викликається обробник події OnClick – Button1Click. }
procedure Button1Click(Sender: TObject);
begin
 {Намагаємося отримати заголовок}
 NMHTTP1.Head(Edit1.Text);
 {Якщо адрес невірний, то тут вискочить помилка}
end;
       //Викачування вказаної URL в папку розробленої
       //програми
{... ПОЧАТОК.
Йде заголовок файлу і визначення форми TForm1 і її екземпляра Form1.
 У формі є кнопка TButton і три поля TEdit.
При натисненні на кнопку викликається обробник події OnClick – Button1Click.
Перед цим в перший TEdit введен  адресу URL, в другій – ім'я файлу для заголовка,
а в третій – ім'я файлу для тіла html}
procedure Button1Click(Sender: TObject);
begin
    {Намагаємося отримати http-документ}
    {Результат треба записати у файли}
    NMHTTP1.InputFileMode := True;
    { указуємо в які саме файли}
    NMHTTP1.Header := Edit2.Text;
    NMHTTP1.Body := Edit3.Text;
    NMHTTP1.Get(Edit1.Text);
end;
```

Розгляд викачування відразу декількох URL одночасно необхідно розглянути більш детально. У Delphi дуже легко створювати окремі, підлеглі процеси за допомогою базового класу TThread.

Розглянемо приклад реалізований метод – одночасного викачування вказаних URL в заданий каталог.

….

```
{... ПОЧАТОК.
Йде заголовок файлу і визначення форми TForm1 і її екземпляра Form1,опис класу
окремого процесу}
Type
    THTTPThread = class(TThread)
    private
{Для кожного процесу – створюємо свій компонент TNMHTTP}
        FHTTP: TNMHTTP;
    Protected
{Execute викликається при запуску процесу;
override – замінюємо існуючу процедуру базового класу TThread}
procedure Execute; override;
{DoWork – створена нами функція, виконання якої синхронізується в Execute}
procedure DoWork;
    public
{URL – створена нами рядок, вказуючий процесу, який URL йому потрібно викачати}
        URL: string;
    end;
{У форму потрібно помістити три кнопки TButton, одне поле TEdit і один список
TListBox.
При натисненні на кнопку Button1 викликається обробник події OnClick –
 Button1Click.
 Перед цим в TEdit потрібно ввести шлях до каталога, в якому зберігатимуться
викачані файли, а ListBox1 потрібно заповнити списком URL-ов для викачування (за
допомогою кнопок Add (Button2) і Delete (Button3))}
procedure TForm1.Button3Click(Sender: TObject);
begin
    {Видалення виділеного URL із списку}
    if ListBox1.ItemIndex >= 0 then
        ListBox1.Items.Delete(ListBox1.ItemIndex);
end;
procedure TForm1.Button2Click(Sender: TObject);
    var s: string;
begin
    {Додавання URL в список}
    s := InputBox('Input','Введення URL:','');
    if s <> '' then ListBox1.Items.Add(s);
end;
procedure TForm1.Button1Click(Sender: TObject);
  var i: Integer;
begin
    {Перевірка на існування каталога}
```

```
    if Length(Edit1.Text)> 0 then
        if not DirectoryExists(Edit1.Text) then
            MkDir(Edit1.Text);
 {Далі йде створення для кожного URL в списку свого процесу}
    for i := 0 to ListBox1.Items.Count-1 do begin
        with THTTPThread.Create(True) do begin
{Створюємо припинену задачу, указуємо їй її URL і запускаємо її}
            URL := ListBox1.Items[i];
            Resume;
        end;
    end;
end;
{Оператори процесу THTTPThread}
procedure THTTPThread.Execute;
begin
{Робимо так, щоб кожний процес виконувався одночасно з іншими (синхронізація)}
    Synchronize(DoWork);
end;
procedure THTTPThread.DoWork;
    var i: Integer;
begin
    {Створюємо компонент TNMHTTP}
    FHTTP := TNMHTTP.Create(Form1);
    {Результат записуємо у файли}
    FHTTP.InputFileMode := True;
    {Підбираємо імена для файлів}
    i:= 1;
    while FileExists(Form1.Edit1.Text+'\page'+IntToStr(i)+'.htm') do
        Inc(i);
    {Указуємо, в які саме файли класти результат}
    FHTTP.Body := Form1.Edit1.Text+'\body'+IntToStr(i)+'.htm';
    FHTTP.Header := Form1.Edit1.Text+'\header'+IntToStr(i)+'.txt';
    {Намагаємося послати запит}
    FHTTP.Get(URL);
    FHTTP.Free;
end;
```

Розглянемо реалізацію програмного опитування користувачів. При роботі розробленого програмного продукту в мережі в деяких випадках необхідно знати поточний стан як локального, так і видалених хостів (чи має локальний хост в

даний момент можливість виходу в мережу Інтеренет, чи доступний якийсь видалений хост і т.д.)

Загальновідомо, що для вказаної мети використовується утиліта ping. Принцип роботи ping-а заснований на використовуванні протоколу ICMP – Internet Control Message Protocol (протокол керівників, або контрольних, повідомлень).

За допомогою ICMP хост в мережі обмінюються різною службовою інформацією (інформацією про зміну маршруту, зменшення швидкості передачі, неприступність якої-небудь адреси і т.д.)

В основі протоколу ICMP лежить поняття повідомлень. Повідомлення ICMP протоколу, як правило, оповіщають про помилки, що виникають при обробці датаграмм. ICMP використовує основні властивості протоколу IP, неначебто він був протоколом більш високого рівня. На самій же справі ICMP є складовою частиною IP.

Одним з типів повідомлень протоколу є т.н. "ехо-запит". Отримавши "ехо-запит" хост зобов'язаний відповісти тому, що послав "ехо-відповіддю".

По суті, "ехо-запит" та "ехо-відповідь" відрізняються лише адресами відправника і одержувача і кодом типу повідомлення (тип 8 – "лунає-запит, тип 0 – "лунає-відповідь).

Реалізації утиліти ping на різних платформах істотно відрізняються. Так, в Ос UNIX використовуються RAW sockets (необроблені, "сирі" сокети), а в Ос Windows всіх версій – т.з. ICMP API.

На практиці, у всіх версіях Windows є і використовується бібліотека icmp.dll. Отже, на даний момент, можна її використовувати що було зроблено у роботі.

Визначення типів і прототипи функцій для Microsoft icmp.dll:

```
type
// Інформація заголовка IP (Наповнення)
ip_option_information = packed record
Ttl : byte; // Час життя (використовується traceroute-ом)
Tos : byte; // Тип обслуговування, звичайно 0
```

```
Flags : byte; // Прапори заголовка IP, звичайно 0
// Розмір даних в заголовку, звичайно 0, максимум 40
OptionsSize : byte;
OptionsData : Pointer; // Покажчик на дані
end;
icmp_echo_reply = packed record
        Address : u_long; // Адреса відповідаючого
        Status : u_long; // IP_STATUS
        // Час між луна-запитом і луна-відповіддю
        RTTime : u_long; // в мілісекундах
        DataSize : u_short; // Розмір повернених даних
        Reserved : u_short; // Зарезервовано
        Data : Pointer; // Покажчик на повернені дані
        // Інформація із заголовка IP
        Options : ip_option_information;
    end;
    PIPINFO = ^ip_option_information;
    PVOID = Pointer;
function IcmpCreateFile():THandle;stdcall;external 'ICMP.DLL';
function IcmpCloseHandle(IcmpHandle : THandle): BOOL; stdcall; external 'ICMP.DLL'
name 'IcmpCloseHandle';
function IcmpSendEcho(IcmpHandle : THandle;
// Адреса одержувача (в мережному порядку)
DestAddress : u_long;
RequestData : PVOID; // Покажчик на посилані дані
RequestSize : Word; // Розмір посиланих даних
RequestOptns : PIPINFO;// Покажчик на посилану структуру
//ip_option_information (може бути nil)
//Покажчик на буфер, що містить відповіді.
ReplyBuffer:PVOID;
ReplySize : DWORD;
//Розмір буфера відповідей
Timeout : DWORD
//Час очікування відповіді в мілісекундах
) : DWORD; stdcall; external 'ICMP.DLL' name 'IcmpSendEcho';
```

Функція IcmpSendEcho() посилає ICMP exo-запит за заданою IP адресою і поміщає всі відповіді, отримані за час заданого таймауту, в буфер відповідей (ReplyBuffer). Перед використовуванням функцій Icmp.dll необхідно викликати функцію WSAStartup() з Winsock.

Якщо цього не зробити, то перший же виклик функції IcmpSendEcho() приведе до помилки 10091 (WSASYSNOTREADY).

Слід помітити, що відповіді, поміщені в буфер, необов'язково будуть повідомленнями ехо-відповідь. Можливо, що у відповідь на ехо-запит прийдуть повідомлення ICMP про виниклі помилки.

Природно, ці повідомлення так само будуть поміщені в ReplyBuffer.

Розмір буфера відповідей повинен бути достатнім для прийому хоча б однієї відповіді, будь то ехо-відповідь або повідомлення про помилку. Звідси, вказаний розмір складається з розміру самої структури icmp_echo_reply плюс Max(RequestSize, 8), оскільки повідомлення ICMP про помилку складає 8 байт.

Функція IcmpSendEcho() повертає кількість відповідей (або структур icmp_echo_reply) в масиві ReplyBuffer. Якщо функція повернула 0, то для більш детальної діагностики виниклої помилки використовується функція GetLastError().

Нижче приведений базовий фрагмент коду який необхідний для однократного посилання ехо-запиту:

```
procedure TForm1.Button1Click(Sender: TObject);
var
    hIP : THandle;
    pingBuffer : array [0..31] Char;
    pIpe : ^icmp_echo_reply;
    pHostEn : PHostEnt;
    wVersionRequested : WORD;
    lwsaData : WSAData;
    error : DWORD;
    destAddress : In_Addr;
begin
    // Створюємо handle
    hIP := IcmpCreateFile();
GetMem( pIpe, sizeof(icmp_echo_reply)+ sizeof(pingBuffer));
    pIpe.Data := @pingBuffer;
    pIpe.DataSize := sizeof(pingBuffer);
    wVersionRequested := MakeWord(1,1);
    error := WSAStartup(wVersionRequested,lwsaData);
    if (error <> 0) then
```

```
    begin
Memo1.SetTextBuf('Помилка у виклику  '+'WSAStartup().');
Memo1.Lines.Add('Код помилки: '+IntToStr(error));
    Exit;
    end;
        pHostEn := gethostbyname;
    error := GetLastError();
    if (error <> 0) then
    begin
Memo1.SetTextBuf('Помилка у виклику '+'gethostbyname().');
Memo1.Lines.Add('Код помилки: '+IntToStr(error));
    Exit;
    end;
destAddress := PInAddr(pHostEn^.h_addr_list^)^;
// Посилаємо ping-пакет
Memo1.Lines.Add('Pinging'+pHostEn^.h_name+' ['+
    inet_ntoa(destAddress)+'] '+ ' with '+
    IntToStr(sizeof(pingBuffer))+
'bytes data:');
IcmpSendEcho(hIPdestAddress.S_addr, pingBuffer                    sizeof(pingBuffer),
Nil, pIpe sizeof(icmp_echo_reply)+ sizeof(pingBuffer), 5000);
error := GetLastError();
    if (error <> 0) then
    begin
Memo1.SetTextBuf('Помилка у виклику  '+'IcmpSendEcho()');
Memo1.Lines.Add('Код помилки: '+IntToStr(error));
    Exit;
    end;
     // Дивимося деякі з даних, що повернулися
Memo1.Lines.Add('Відповідь з '+ IntToStr(LoByte(LoWord(pIpe^.Address)))+'.'+
IntToStr(HiByte(LoWord(pIpe^.Address)))+'.'+
IntToStr(LoByte(HiWord(pIpe^.Address)))+'.'+
IntToStr(HiByte(HiWord(pIpe^.Address))));
Memo1.Lines.Add('Час відповіді:'+IntToStr(pIpe.RTTime)+' ms');
    IcmpCloseHandle(hIP);
    WSACleanup();
    FreeMem(pIpe);
end;
```

Для роботи цього фрагмента коду необхідно:

– створити форму TForm1;

– включити в розділ Uses юнит WinSock;

– помістити на форму компонент Memo1:TMemo і кнопку Button1:TButton;

– в розділі Implementation помістити описи типів і функцій Icmp.dll вище);

– зіставити події OnClick кнопки приведену функцію.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням. Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8 × 8-бітних S-блоку із вхідними й вихідними афінними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузія використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) уважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу

NESSIE і Японським CRYPTREC проект. Японський шифр має рівні безпеки й можливості обробки, порівнянні із шифром AES/Rijndael.

Camellia – це блоковий шифр, який може бути повністю визначені мінімальними системами багатомірних багаточленів:

– Камелія (а також AES) S-блоки можуть бути описані системою 23 квадратних рівнянь в 80 членах.

– Розклад ключів можна описати 1120 рівняннями в 768 змінні з використанням 3328 лінійних і квадратичних членів.

– Увесь блоковий шифр можна описати 5104 рівняннями в 2816 змінні з використанням 14 592 лінійних і квадратичних членів.

– Усього потрібно 6224 рівняння з 3584 змінними з використанням 17 920 лінійних і квадратичних членів.

– Кількість вільних членів становить 11 696, що приблизно таке ж число, що й для AES.

Теоретично, такі властивості можуть дозволити зламати Camellia (і AES) за допомогою алгебраїчної атаки, такий як розширена розріджена лінеаризація, у т Майбутнє за умови, що атака стане можливою.

Хоча Camellia запатентована, вона доступна за безоплатною ліцензією. Це дозволило шифру Camellia стати частиною проекту OpenSSL під ліцензією з відкритим вихідним кодом з листопада 2006 року. Це також дозволило йому стати частиною Mozilla Модуль NSS (Служби мережної безпеки).

Підтримка Camellia була додана в остаточний випуск Mozilla Firefox 3 в 2008 році (за замовчуванням відключене починаючи з Firefox 33 в 2014 році в дусі «Пропозиції по зміні стандартних наборів шифрів TLS, пропонованих браузерами», який був виключено з версії 37 в 2015 році). Pale Moon, відгалуження Mozilla / Firefox, продовжує пропонувати Camellia і розширив свою підтримку, включивши в нього набори Galois / Counter mode (GCM) із шифром, але вилучив GCM знову у випуску 27.2.0, пославшись на очевидну відсутність інтересу до них.

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Неможлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення захисту додатків та даних на мобільних корпоративних пристроях складається з: блоку обрання параметрів роботи; функціональних кнопок (налаштування робочого столу, параметри захисту додатків, параметри даних, параметри моніторингу); блоку запуску моніторингу системи.

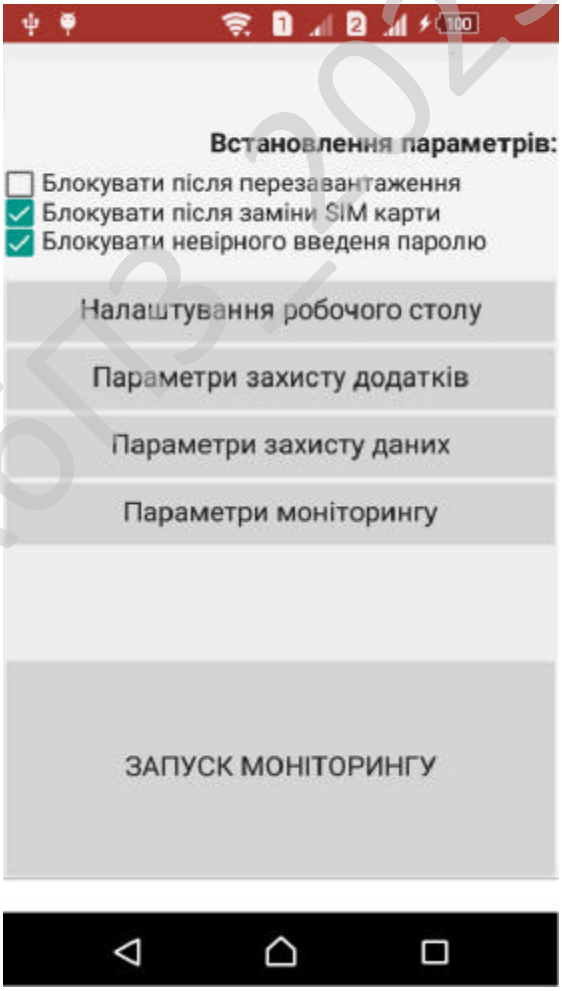

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

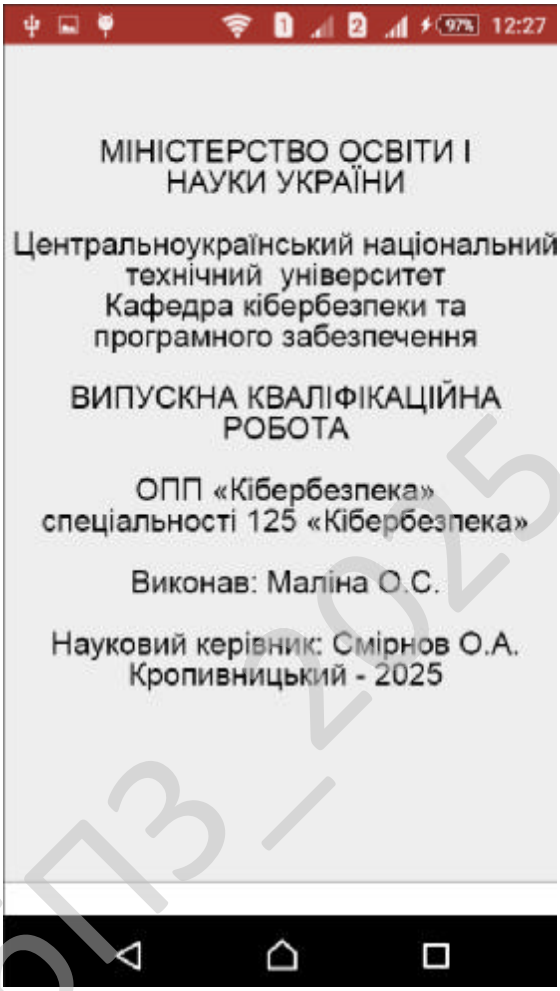

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів. Проводилось тестування форматом чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють: Як виконуються функції програми; Як приймаються вихідні дані; Як виробляються результати; Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТ).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок: Некоректних чи відсутніх функцій; Помилок інтерфейсу; Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних; Помилок характеристик (необхідна ємність пам'яті і т.д.); Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються. Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для захисту додатків та даних на мобільних корпоративних пристроях.

– Досліджена система для захисту додатків та даних на мобільних корпоративних пристроях.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для захисту додатків та даних на мобільних корпоративних пристроях.

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 Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для захисту додатків та даних на мобільних корпоративних

пристроях. 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 Android.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Camellia.

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. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

2. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

3. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

4. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

5. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

6. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

7. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

8. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

9. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

10. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

11. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

12. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

13. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

14. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

15. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

16. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

17. 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.

18. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

19. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

20. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

21. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

22. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

23. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

24. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

25. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

26. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

28. 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.

29. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

30. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

31. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

32. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

33. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

34. 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.

35. 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.

36. 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.

37. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

38. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

39. 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.

40. 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.

41. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

42. 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.

43. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

44. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

45. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

46. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

47. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

48. 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.

49. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

50. 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.

51. 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.

52. 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.

53. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

54. Smirnov A.A., Kuznetsov 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

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. № 51-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях.

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 для захисту додатків та даних на мобільних корпоративних пристроях;

– 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 мобільних пристроях під керуванням ОС Android 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 Android.

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 Python.

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 – 82 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 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Смірнов О.А.

Програмне забезпечення системи кібербезпеки для захисту додатків та даних на мобільних корпоративних пристроях

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 19

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#import os module for interacting with the operating system
import os
#import sys module for system-specific parameters and functions
import sys
#import time module for time-related functions
import time
#import hashlib for cryptographic hashing functions
import hashlib
#import logging for logging events and errors
import logging
#import sqlite3 for database operations
import sqlite3
#import random for random number generation
import random
#import datetime for date and time operations
import datetime
#import threading for concurrent operations
import threading
#import socket for network communications
import socket
#import json for JSON parsing and formatting
import json

#Configure logging with DEBUG level and specific format
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')
#Global variable for database file path
DB_PATH = 'security_system.db'
#Global variable for encryption key used in data protection
ENCRYPTION_KEY = 'my_secret_key_12345'
#Global variables for threat intelligence server details
THREAT_INTEL_SERVER = '127.0.0.1'
THREAT_INTEL_PORT = 8080

#Initialize the SQLite database and create necessary tables
def init_database():
#Connect to SQLite database using the specified path
    conn = sqlite3.connect(DB_PATH)
#Create a cursor object to execute SQL statements
    cursor = conn.cursor()
#Create table for mobile devices if it does not already exist
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS devices (
            device_id TEXT PRIMARY KEY,
            owner TEXT,
            os_version TEXT,
            is_encrypted INTEGER,
            last_checked TIMESTAMP
        )
    ''')
#Create table for system users if it does not already exist
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            username TEXT PRIMARY KEY,
            password_hash TEXT,
            role TEXT
        )
    ''')
#Create table for security logs to record events
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS security_logs (
            log_id INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp TIMESTAMP,
            event TEXT
        )
    ''')
#Commit changes to the database
```

```
    conn.commit()
#Close the database connection
    conn.close()
#Return True to indicate successful initialization
    return True

#Define a class to represent a mobile device in the corporate network
class MobileDevice:
    def __init__(self, device_id, owner, os_version, is_encrypted):
#Assign device ID from input
        self.device_id = device_id
#Assign owner email or identifier
        self.owner = owner
#Assign operating system version installed on the device
        self.os_version = os_version
#Assign encryption status (True if encrypted, False otherwise)
        self.is_encrypted = is_encrypted
#Initialize an empty list to store applications installed on the device
        self.apps = []
#Store the last time a security scan was performed on the device
        self.last_scan = None

    def display_info(self):
#Format and return a string containing device information
        info = "Device ID: {} | Owner: {} | OS: {} | Encrypted: {}".format(
            self.device_id, self.owner, self.os_version, self.is_encrypted
        )
        return info

    def add_app(self, app):
#Add a mobile application object to the device's application list
        self.apps.append(app)

    def get_apps(self):
#Return the list of mobile applications installed on the device
        return self.apps

#Define a class to represent an application installed on a mobile device
class MobileApp:
    def __init__(self, app_id, name, version, permissions):
#Assign application ID from input
        self.app_id = app_id
#Assign application name from input
        self.name = name
#Assign application version from input
        self.version = version
#Assign a list of permissions requested by the application
        self.permissions = permissions
#Initialize an empty list to store any detected vulnerabilities
        self.vulnerabilities = []

    def check_for_vulnerabilities(self):
#Simulate vulnerability scanning by randomly deciding if vulnerabilities exist
        check = random.choice([True, False])
        if check:
#Append a dummy vulnerability description if one is detected
            self.vulnerabilities.append("Critical vulnerability detected in
module XYZ")
        return self.vulnerabilities

    def update_version(self, new_version):
#Update the application's version to the new version provided
        self.version = new_version

#Define a class to represent a system user
class User:
    def __init__(self, username, password, role):
#Assign username for the user
        self.username = username
```

```
#Hash the provided password using the hash_password method
        self.password_hash = self.hash_password(password)
#Assign the role of the user (e.g., admin, user)
        self.role = role

    def hash_password(self, password):
#Create a SHA-256 hash object for the password
        hash_obj = hashlib.sha256(password.encode())
#Return the hexadecimal digest of the hash
        return hash_obj.hexdigest()

    def verify_password(self, password):
#Verify the provided password by comparing its hash to the stored hash
        return self.hash_password(password) == self.password_hash

#Define a class to manage user authentication processes
class AuthenticationManager:
    def __init__(self):
#Initialize a dictionary to store logged-in users
        self.logged_in_users = {}
#Initialize a dictionary to store all users loaded from the database
        self.users = {}
#Load user information from the database into the users dictionary
        self.load_users()

    def load_users(self):
#Attempt to load users from the SQLite database
        try:
            conn = sqlite3.connect(DB_PATH)
            cursor = conn.cursor()
            cursor.execute("SELECT username, password_hash, role FROM users")
            rows = cursor.fetchall()
            for row in rows:
                username, password_hash, role = row
#Create a User object with an empty password and set the password hash manually
                user = User(username, "", role)
                user.password_hash = password_hash
                self.users[username] = user
            conn.close()
        except Exception as e:
            logging.error("Error loading users: " + str(e))

    def login(self, username, password):
#Handle user login by verifying the username and password
        if username in self.users:
            user = self.users[username]
            if user.verify_password(password):
#Mark the user as logged in in the dictionary
                self.logged_in_users[username] = True
#Log the successful login event using the static logging method
                LogManager.log_event_static("User {} logged in
successfully".format(username))
                return True
            else:
#Log the failed login attempt due to incorrect password
                LogManager.log_event_static("Failed login attempt for user
{}".format(username))
                return False
        else:
#Log an event for a login attempt with an unknown username
            LogManager.log_event_static("Login attempt with unknown username:
{}".format(username))
            return False

    def logout(self, username):
#Handle user logout by removing the user from the logged-in dictionary
        if username in self.logged_in_users:
            del self.logged_in_users[username]
```

```
            LogManager.log_event_static("User {} logged out
successfully".format(username))
            return True
        return False

#Define a class to manage encryption and decryption of sensitive data
class EncryptionManager:
    def __init__(self, key):
#Store the provided encryption key for use in data protection
        self.key = key

    def encrypt_data(self, data):
#Encrypt data using a simple XOR cipher for demonstration purposes
        encrypted = []
        for i, char in enumerate(data):
            key_c = self.key[i % len(self.key)]
            encrypted_char = chr(ord(char) ^ ord(key_c))
            encrypted.append(encrypted_char)
        encrypted_data = ''.join(encrypted)
        return encrypted_data

    def decrypt_data(self, encrypted_data):
#Decrypt data that was encrypted using the XOR cipher method
        decrypted = []
        for i, char in enumerate(encrypted_data):
            key_c = self.key[i % len(self.key)]
            decrypted_char = chr(ord(char) ^ ord(key_c))
            decrypted.append(decrypted_char)
        decrypted_data = ''.join(decrypted)
        return decrypted_data

#Define a class to handle threat detection and vulnerability scanning
class ThreatDetection:
    def __init__(self):
#Initialize threat detection by loading predefined threat signatures
        self.threat_signatures = []
        self.load_threat_signatures()

    def load_threat_signatures(self):
#Load a list of dummy threat signatures for simulation
        self.threat_signatures = [
            "malware_signature_001",
            "trojan_signature_002",
            "ransomware_signature_003",
            "spyware_signature_004"
        ]

    def scan_for_threats(self, device):
#Scan a given mobile device for security threats and vulnerabilities
        detected_threats = []
        if not device.is_encrypted:
#Flag the device if it is not encrypted as a potential risk
            detected_threats.append("Device {} is not
encrypted".format(device.device_id))
        for app in device.get_apps():
            vulnerabilities = app.check_for_vulnerabilities()
            if vulnerabilities:
                detected_threats.extend(vulnerabilities)
        return detected_threats

    def query_threat_intelligence(self):
#Query a remote threat intelligence server for the latest threat signatures
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.connect((THREAT_INTEL_SERVER, THREAT_INTEL_PORT))
            request_data = json.dumps({"action": "get_latest_signatures"})
            sock.sendall(request_data.encode())
            response = sock.recv(4096).decode()
            sock.close()
```

```
            new_signatures = json.loads(response).get("signatures", [])
            if new_signatures:
                self.threat_signatures.extend(new_signatures)
                LogManager.log_event_static("Threat intelligence updated with
new signatures")
            return new_signatures
        except Exception as e:
            LogManager.log_event_static("Failed to query threat intelligence: "
+ str(e))
            return []

#Define a class to manage security logging operations
class LogManager:
    def __init__(self, db_connection):
#Store the database connection for logging events
        self.conn = db_connection

    def log_event(self, event):
#Insert a log event into the security_logs table with a timestamp
        try:
            cursor = self.conn.cursor()
            cursor.execute("INSERT INTO security_logs (timestamp, event) VALUES
(?, ?)",
                           (datetime.datetime.now(), event))
            self.conn.commit()
        except Exception as e:
            logging.error("Error logging event: " + str(e))

    @staticmethod
    def log_event_static(event):
#Static method to log an event directly to the database without an instance
        try:
            conn = sqlite3.connect(DB_PATH)
            cursor = conn.cursor()
            cursor.execute("INSERT INTO security_logs (timestamp, event) VALUES
(?, ?)",
                           (datetime.datetime.now(), event))
            conn.commit()
            conn.close()
        except Exception as e:
            logging.error("Static log error: " + str(e))

    def get_logs(self):
#Retrieve all log entries from the security_logs table
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT * FROM security_logs")
            logs = cursor.fetchall()
            return logs
        except Exception as e:
            logging.error("Error retrieving logs: " + str(e))
            return []

#Define a class representing the mobile security agent operating on a device
class MobileSecurityAgent:
    def __init__(self, device, auth_manager, encryption_manager,
threat_detector, log_manager):
#Store the mobile device to be protected
        self.device = device
#Store the authentication manager for handling user access
        self.auth_manager = auth_manager
#Store the encryption manager for protecting data
        self.encryption_manager = encryption_manager
#Store the threat detection system for scanning vulnerabilities
        self.threat_detector = threat_detector
#Store the log manager for recording events
        self.log_manager = log_manager
#Initialize the running flag to control the security check loop
        self.running = False
```

```
    def run_security_checks(self):
#Start periodic security checks on the mobile device
        self.running = True
        while self.running:
            self.log_manager.log_event("Starting security check for device " +
self.device.device_id)
            threats = self.threat_detector.scan_for_threats(self.device)
            if threats:
                for threat in threats:
                    self.log_manager.log_event("Threat detected on device {}:
{}".format(self.device.device_id, threat))
            else:
                self.log_manager.log_event("No threats detected on device " +
self.device.device_id)
            self.device.last_scan = datetime.datetime.now()
            time.sleep(10)

    def stop_security_checks(self):
#Stop the periodic security check loop
        self.running = False

    def update_security_policies(self):
#Update security policies by querying the latest threat intelligence
        new_signatures = self.threat_detector.query_threat_intelligence()
        if new_signatures:
            self.log_manager.log_event("Security policies updated with new
threat signatures")
        else:
            self.log_manager.log_event("No new threat signatures found during
update")

#Define a class to manage data protection functions such as encryption and
decryption
class DataProtection:
    def __init__(self, encryption_manager):
#Store the encryption manager for use in data protection
        self.encryption_manager = encryption_manager

    def protect_data(self, data):
#Encrypt the provided data using the encryption manager
        encrypted = self.encryption_manager.encrypt_data(data)
        return encrypted

    def recover_data(self, encrypted_data):
#Decrypt the provided encrypted data to recover original information
        decrypted = self.encryption_manager.decrypt_data(encrypted_data)
        return decrypted

#Simulate network queries to a threat intelligence server for security
operations
def simulate_network_queries():
    try:
        for i in range(5):
#Create a socket connection for each query iteration
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.connect((THREAT_INTEL_SERVER, THREAT_INTEL_PORT))
#Prepare a JSON payload with a unique query ID and action
            payload = json.dumps({"query_id": i, "action": "status_check"})
            sock.sendall(payload.encode())
#Receive the server's response
            response = sock.recv(2048).decode()
#Log the response from the network query
            LogManager.log_event_static("Network query {} response:
{}".format(i, response))
            sock.close()
            time.sleep(2)
    except Exception as e:
        LogManager.log_event_static("Error during network queries: " + str(e))
```

```
#Periodically query the security logs from the database to monitor events
def periodic_log_query():
    while True:
        try:
            conn = sqlite3.connect(DB_PATH)
            cursor = conn.cursor()
            cursor.execute("SELECT * FROM security_logs ORDER BY timestamp DESC
LIMIT 10")
            logs = cursor.fetchall()
            for log in logs:
                logging.info("Log Entry: " + str(log))
            conn.close()
            time.sleep(15)
        except Exception as e:
            logging.error("Error in periodic log query: " + str(e))
            time.sleep(15)

#Main function to initialize and run the cybersecurity system simulation
def main():
#Initialize the database and create required tables
    init_database()
#Establish a database connection for the log manager
    db_conn = sqlite3.connect(DB_PATH)
#Instantiate the log manager with the database connection
    log_manager = LogManager(db_conn)
#Instantiate the authentication manager and load users
    auth_manager = AuthenticationManager()
#Instantiate the encryption manager with the global encryption key
    encryption_manager = EncryptionManager(ENCRYPTION_KEY)
#Instantiate the threat detection system to handle vulnerability scanning
    threat_detector = ThreatDetection()
#Create a sample mobile device for simulation with dummy data
    device = MobileDevice("DEV123456", "john.doe@corporate.com", "Android 11",
False)
#Create sample mobile applications and add them to the mobile device
    app1 = MobileApp("APP001", "Email Client", "1.0", ["internet", "contacts"])
    app2 = MobileApp("APP002", "Corporate Messenger", "2.5", ["internet",
"microphone"])
    device.add_app(app1)
    device.add_app(app2)
#Instantiate the data protection system for encrypting sensitive information
    data_protection = DataProtection(encryption_manager)
#Define some dummy sensitive data for testing data protection
    sample_data = "Confidential Corporate Data: Top Secret Information"
#Encrypt the sensitive data using the data protection system
    encrypted_data = data_protection.protect_data(sample_data)
#Decrypt the data back to verify encryption and decryption processes
    recovered_data = data_protection.recover_data(encrypted_data)
#Log the data protection operation for auditing purposes
    log_manager.log_event("Data encrypted and recovered for verification on
device " + device.device_id)
#Instantiate the mobile security agent with all required components
    security_agent = MobileSecurityAgent(device, auth_manager,
encryption_manager, threat_detector, log_manager)
#Start the security agent in a separate daemon thread for continuous monitoring
    agent_thread = threading.Thread(target=security_agent.run_security_checks)
    agent_thread.daemon = True
    agent_thread.start()
#Start a separate daemon thread to periodically query the security logs
    log_query_thread = threading.Thread(target=periodic_log_query)
    log_query_thread.daemon = True
    log_query_thread.start()
#Start a separate daemon thread to simulate network queries to threat
intelligence server
    network_query_thread = threading.Thread(target=simulate_network_queries)
    network_query_thread.daemon = True
    network_query_thread.start()
#Simulate an admin user login for system maintenance operations
```

```
    login_status = auth_manager.login("admin", "admin123")
    if login_status:
        log_manager.log_event("Admin user logged in for system maintenance")
    else:
        log_manager.log_event("Admin user failed to log in during system
maintenance")
#Run the main simulation loop for a fixed duration to demonstrate system
operation
    for _ in range(3):
        security_agent.update_security_policies()
        time.sleep(20)
#Stop the security agent after simulation is complete
    security_agent.stop_security_checks()
#Log the graceful termination of the simulation
    log_manager.log_event("System simulation terminated gracefully")
#Keep the main thread alive briefly to allow background threads to finish
logging
    time.sleep(5)
#Close the database connection before exiting the application
    db_conn.close()

#Execute the main function when the script is run directly
if __name__ == "__main__":
    main()
```

Файл MultiFactorAuthentication.py

```
import time
import hashlib
import sqlite3
import random
import socket
import json
import threading
import datetime
import os
import sys
import base64
import hmac

class MultiFactorAuthentication:
    def __init__(self):
        self.totp_secrets = {}
        self.sms_codes = {}
    def register_user(self, username, secret):
        self.totp_secrets[username] = secret
    def generate_totp(self, secret, interval=30, digits=6):
        t = int(time.time() // interval)
        key = base64.b32decode(secret, True)
        msg = t.to_bytes(8, 'big')
        h = hmac.new(key, msg, hashlib.sha1).digest()
        o = h[19] & 15
        code = (int.from_bytes(h[o:o+4], 'big') & 0x7fffffff) % (10 ** digits)
        return str(code).zfill(digits)
    def send_sms_code(self, username):
        code = random.randint(100000, 999999)
        self.sms_codes[username] = str(code)
        return str(code)
    def verify_totp(self, username, code):
        if username not in self.totp_secrets:
            return False
        secret = self.totp_secrets[username]
        expected = self.generate_totp(secret)
        return expected == code
    def verify_sms(self, username, code):
        if username not in self.sms_codes:
            return False
        expected = self.sms_codes[username]
        return expected == code
    def verify_mfa(self, username, totp_code, sms_code):
        return self.verify_totp(username, totp_code) and
self.verify_sms(username, sms_code)

class NetworkTrafficMonitor:
    def __init__(self, interface='lo'):
        self.interface = interface
        self.running = False
        self.captured_packets = []
```

```
    def start_monitoring(self):
        self.running = True
        self.thread = threading.Thread(target=self._monitor)
        self.thread.daemon = True
        self.thread.start()
    def _monitor(self):
        while self.running:
            packet = self._simulate_packet()
            self.captured_packets.append(packet)
            time.sleep(1)
    def _simulate_packet(self):
        src_ip = "192.168.{}.{}".format(random.randint(0,255),
random.randint(1,254))
        dst_ip = "10.0.{}.{}".format(random.randint(0,255),
random.randint(1,254))
        protocol = random.choice(["TCP", "UDP", "ICMP"])
        length = random.randint(20,1500)
        packet = {"src_ip": src_ip, "dst_ip": dst_ip, "protocol": protocol,
"length": length, "timestamp": datetime.datetime.now().isoformat()}
        return packet
    def stop_monitoring(self):
        self.running = False
        self.thread.join()
    def get_captured_packets(self):
        return self.captured_packets
    def analyze_traffic(self):
        analysis = {}
        for packet in self.captured_packets:
            proto = packet["protocol"]
            analysis[proto] = analysis.get(proto, 0) + 1
        return analysis

class AntivirusModule:
    def __init__(self):
        self.signatures = ["virus_signature_1", "malware_pattern_A",
"trojan_marker_X", "worm_identifier_9"]
        self.scan_results = []
    def scan_file(self, file_path):
        result = {"file": file_path, "infected": False, "details": ""}
        try:
            with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
                content = f.read()
            for sig in self.signatures:
                if sig in content:
                    result["infected"] = True
                    result["details"] += "Detected " + sig + "; "
            if result["infected"]:
                self.scan_results.append(result)
            return result
        except Exception as e:
            result["details"] = "Error scanning file"
            return result
    def scan_directory(self, directory):
```

```
        results = []
        for root, dirs, files in os.walk(directory):
            for file in files:
                file_path = os.path.join(root, file)
                res = self.scan_file(file_path)
                results.append(res)
        return results
    def update_signatures(self):
        new_sigs = ["new_virus_" + str(random.randint(1,100))]
        self.signatures.extend(new_sigs)
        return new_sigs

class AnomalyDetectionSystem:
    def __init__(self):
        self.metrics = []
        self.anomalies = []
    def record_metric(self, value):
        timestamp = datetime.datetime.now().timestamp()
        self.metrics.append({"timestamp": timestamp, "value": value})
    def analyze_metrics(self):
        if not self.metrics:
            return []
        values = [m["value"] for m in self.metrics]
        avg = sum(values) / len(values)
        variance = sum((x - avg) ** 2 for x in values) / len(values)
        std_dev = variance ** 0.5
        threshold = std_dev * 2
        self.anomalies = []
        for metric in self.metrics:
            if abs(metric["value"] - avg) > threshold:
                self.anomalies.append(metric)
        return self.anomalies
    def simulate_metric_generation(self, count=50):
        for _ in range(count):
            value = random.uniform(0,100)
            self.record_metric(value)
            time.sleep(0.1)
    def reset_metrics(self):
        self.metrics = []
        self.anomalies = []

class SecurityUpdatesManager:
    def __init__(self):
        self.available_updates = []
        self.installed_updates = []
    def check_for_updates(self):
        self.available_updates = ["patch_" + str(i) for i in range(1,
random.randint(2,5))]
        return self.available_updates
    def download_update(self, update):
        time.sleep(random.uniform(0.5, 1.5))
        return True
    def install_update(self, update):
```

```
        success = self.download_update(update)
        if success:
            self.installed_updates.append(update)
        return success
    def schedule_updates(self):
        updates = self.check_for_updates()
        for update in updates:
            installed = self.install_update(update)
            if installed:
                print("Installed update: " + update)
            else:
                print("Failed to install update: " + update)
    def get_update_status(self):
        status = {"available": self.available_updates, "installed":
self.installed_updates}
        return status

def main():
    mfa = MultiFactorAuthentication()
    mfa.register_user("user1", "JBSWY3DPEHPK3PXP")
    totp_code = mfa.generate_totp("JBSWY3DPEHPK3PXP")
    sms_code = mfa.send_sms_code("user1")
    mfa_result = mfa.verify_mfa("user1", totp_code, sms_code)
    print("MFA verification result:", mfa_result)
    ntm = NetworkTrafficMonitor()
    ntm.start_monitoring()
    time.sleep(5)
    ntm.stop_monitoring()
    packets = ntm.get_captured_packets()
    traffic_analysis = ntm.analyze_traffic()
    print("Traffic analysis:", traffic_analysis)
    av = AntivirusModule()
    test_dir = "test_directory"
    if not os.path.exists(test_dir):
        os.makedirs(test_dir)
    with open(os.path.join(test_dir, "file1.txt"), "w") as f:
        f.write("This is a harmless file.")
    with open(os.path.join(test_dir, "file2.txt"), "w") as f:
        f.write("This file contains virus_signature_1 in it.")
    scan_results = av.scan_directory(test_dir)
    print("Antivirus scan results:")
    for res in scan_results:
        print(res)
    av_updates = av.update_signatures()
    print("Updated signatures:", av_updates)
    ads = AnomalyDetectionSystem()
    ads.simulate_metric_generation(30)
    anomalies = ads.analyze_metrics()
    print("Anomalies detected:")
    for anomaly in anomalies:
        print(anomaly)
    ads.reset_metrics()
    sumgr = SecurityUpdatesManager()
```

```
    sumgr.schedule_updates()
    update_status = sumgr.get_update_status()
    print("Update status:", update_status)

if __name__ == "__main__":
    main()
```

Файл AppAccessControl.py

```
import os
import time
import random
import hashlib
import threading
import datetime
import zipfile
import base64
import json
import hmac

class BiometricAuthentication:
    def __init__(self):
        self.user_biometrics = {}
    def register_user(self, username, biometric_data):
        self.user_biometrics[username] = biometric_data
    def simulate_fingerprint_scan(self):
        time.sleep(random.uniform(0.1, 0.3))
        return "fingerprint_hash_" + str(random.randint(1000, 9999))
    def simulate_face_scan(self):
        time.sleep(random.uniform(0.1, 0.3))
        return "face_hash_" + str(random.randint(1000, 9999))
    def verify_fingerprint(self, username, scanned_data):
        stored = self.user_biometrics.get(username, "")
        return stored == scanned_data
    def verify_face(self, username, scanned_data):
        stored = self.user_biometrics.get(username, "")
        return stored == scanned_data
    def verify_biometric(self, username, method="fingerprint"):
        if method == "fingerprint":
            scanned = self.simulate_fingerprint_scan()
            return self.verify_fingerprint(username, scanned)
        elif method == "face":
            scanned = self.simulate_face_scan()
            return self.verify_face(username, scanned)
        return False

class FileEncryptionManager:
    def __init__(self, key):
        self.key = key
    def encrypt_string(self, plaintext):
        encrypted = ""
        for i, c in enumerate(plaintext):
            encrypted += chr(ord(c) ^ ord(self.key[i % len(self.key)]))
        return base64.b64encode(encrypted.encode()).decode()
    def decrypt_string(self, encrypted_text):
        decoded = base64.b64decode(encrypted_text.encode()).decode()
        decrypted = ""
        for i, c in enumerate(decoded):
            decrypted += chr(ord(c) ^ ord(self.key[i % len(self.key)]))
        return decrypted
```

```
    def encrypt_file(self, input_file, output_file):
        with open(input_file, "r", encoding="utf-8", errors="ignore") as f:
            data = f.read()
        encrypted_data = self.encrypt_string(data)
        with open(output_file, "w", encoding="utf-8") as f:
            f.write(encrypted_data)
    def decrypt_file(self, input_file, output_file):
        with open(input_file, "r", encoding="utf-8", errors="ignore") as f:
            encrypted_data = f.read()
        decrypted_data = self.decrypt_string(encrypted_data)
        with open(output_file, "w", encoding="utf-8") as f:
            f.write(decrypted_data)
    def encrypt_directory(self, directory, output_directory):
        if not os.path.exists(output_directory):
            os.makedirs(output_directory)
        for root, dirs, files in os.walk(directory):
            rel_path = os.path.relpath(root, directory)
            target_dir = os.path.join(output_directory, rel_path)
            if not os.path.exists(target_dir):
                os.makedirs(target_dir)
            for file in files:
                input_path = os.path.join(root, file)
                output_path = os.path.join(target_dir, file + ".enc")
                self.encrypt_file(input_path, output_path)
    def decrypt_directory(self, directory, output_directory):
        if not os.path.exists(output_directory):
            os.makedirs(output_directory)
        for root, dirs, files in os.walk(directory):
            rel_path = os.path.relpath(root, directory)
            target_dir = os.path.join(output_directory, rel_path)
            if not os.path.exists(target_dir):
                os.makedirs(target_dir)
            for file in files:
                if file.endswith(".enc"):
                    input_path = os.path.join(root, file)
                    output_path = os.path.join(target_dir, file[:-4])
                    self.decrypt_file(input_path, output_path)

class IntrusionDetectionSystem:
    def __init__(self):
        self.monitored_logs = []
        self.alerts = []
        self.patterns = ["failed login", "unauthorized access", "error 403",
"SQL injection"]
    def add_log_entry(self, entry):
        timestamp = datetime.datetime.now().isoformat()
        self.monitored_logs.append({"timestamp": timestamp, "entry": entry})
    def analyze_logs(self):
        for log in self.monitored_logs:
            for pattern in self.patterns:
                if pattern in log["entry"]:
                    alert = {"timestamp": log["timestamp"], "alert": "Detected
pattern: " + pattern}
```

```
                    self.alerts.append(alert)
    def get_alerts(self):
        return self.alerts
    def clear_logs(self):
        self.monitored_logs = []
    def simulate_log_generation(self, count=20):
        for _ in range(count):
            log_entry = random.choice(["user admin login successful", "failed
login attempt", "page accessed", "SQL injection attempt detected", "normal
operation"])
            self.add_log_entry(log_entry)
            time.sleep(0.05)
    def run_detection(self):
        self.simulate_log_generation(30)
        self.analyze_logs()

class BackupManager:
    def __init__(self, backup_dir):
        self.backup_dir = backup_dir
    def create_backup(self, source_dir):
        if not os.path.exists(self.backup_dir):
            os.makedirs(self.backup_dir)
        backup_name = "backup_" +
datetime.datetime.now().strftime("%Y%m%d%H%M%S") + ".zip"
        backup_path = os.path.join(self.backup_dir, backup_name)
        with zipfile.ZipFile(backup_path, "w", zipfile.ZIP_DEFLATED) as zipf:
            for root, dirs, files in os.walk(source_dir):
                for file in files:
                    file_path = os.path.join(root, file)
                    arcname = os.path.relpath(file_path, source_dir)
                    zipf.write(file_path, arcname)
        return backup_path
    def list_backups(self):
        if not os.path.exists(self.backup_dir):
            return []
        backups = [f for f in os.listdir(self.backup_dir) if f.endswith(".zip")]
        backups.sort()
        return backups
    def restore_backup(self, backup_file, restore_dir):
        backup_path = os.path.join(self.backup_dir, backup_file)
        if not os.path.exists(restore_dir):
            os.makedirs(restore_dir)
        with zipfile.ZipFile(backup_path, "r") as zipf:
            zipf.extractall(restore_dir)
    def delete_backup(self, backup_file):
        backup_path = os.path.join(self.backup_dir, backup_file)
        if os.path.exists(backup_path):
            os.remove(backup_path)
    def scheduled_backup(self, source_dir, interval):
        while True:
            self.create_backup(source_dir)
            time.sleep(interval)
```

```
class AppAccessControl:
    def __init__(self):
        self.app_permissions = {}
    def register_app(self, app_id, permissions):
        self.app_permissions[app_id] = {"permissions": permissions,
"access_granted": False}
    def grant_access(self, app_id):
        if app_id in self.app_permissions:
            self.app_permissions[app_id]["access_granted"] = True
    def revoke_access(self, app_id):
        if app_id in self.app_permissions:
            self.app_permissions[app_id]["access_granted"] = False
    def check_access(self, app_id):
        if app_id in self.app_permissions:
            return self.app_permissions[app_id]["access_granted"]
        return False
    def update_permissions(self, app_id, new_permissions):
        if app_id in self.app_permissions:
            self.app_permissions[app_id]["permissions"] = new_permissions
    def list_apps(self):
        return list(self.app_permissions.keys())
    def simulate_access_requests(self, count=10):
        for _ in range(count):
            app_id = "app_" + str(random.randint(1, 100))
            permission_request = random.choice(["read", "write", "execute",
"delete"])
            if app_id not in self.app_permissions:
                self.register_app(app_id, [permission_request])
            else:
                current = self.app_permissions[app_id]["permissions"]
                if permission_request not in current:
                    current.append(permission_request)
            time.sleep(0.1)
    def get_app_info(self, app_id):
        return self.app_permissions.get(app_id, {})

def main():
    bio_auth = BiometricAuthentication()
    bio_auth.register_user("user_bio", "fingerprint_hash_1234")
    bio_result = bio_auth.verify_biometric("user_bio", method="fingerprint")
    print("Biometric verification:", bio_result)
    file_enc = FileEncryptionManager("my_file_key")
    test_text = "This is a test string for encryption and decryption."
    encrypted_text = file_enc.encrypt_string(test_text)
    decrypted_text = file_enc.decrypt_string(encrypted_text)
    print("Encrypted text:", encrypted_text)
    print("Decrypted text:", decrypted_text)
    with open("test_input.txt", "w") as f:
        f.write(test_text)
    file_enc.encrypt_file("test_input.txt", "test_input.txt.enc")
    file_enc.decrypt_file("test_input.txt.enc", "test_input_decrypted.txt")
    ids = IntrusionDetectionSystem()
    ids.simulate_log_generation(25)
```

```
    ids.run_detection()
    alerts = ids.get_alerts()
    print("IDS alerts:")
    for alert in alerts:
        print(alert)
    backup_manager = BackupManager("backups")
    if not os.path.exists("test_backup_dir"):
        os.makedirs("test_backup_dir")
    with open(os.path.join("test_backup_dir", "backup_test.txt"), "w") as f:
        f.write("Backup test content")
    backup_path = backup_manager.create_backup("test_backup_dir")
    print("Created backup:", backup_path)
    backups = backup_manager.list_backups()
    print("List of backups:", backups)
    restore_dir = "restored_backup"
    backup_manager.restore_backup(backups[-1], restore_dir)
    app_access = AppAccessControl()
    app_access.register_app("app_control_1", ["read", "write"])
    app_access.register_app("app_control_2", ["execute"])
    app_access.grant_access("app_control_1")
    app_access.revoke_access("app_control_2")
    access1 = app_access.check_access("app_control_1")
    access2 = app_access.check_access("app_control_2")
    print("App access control for app_control_1:", access1)
    print("App access control for app_control_2:", access2)
    app_access.simulate_access_requests(15)
    print("Registered apps:", app_access.list_apps())
    for app in app_access.list_apps():
        print("Info for", app, ":", app_access.get_app_info(app))

if __name__ == "__main__":
    main()
```