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 центру генерації, сертифікації та розподілу ключів у корпоративній мережі”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Ільченко С.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Марченко К.М.
«____» _____ 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 *123 “*Комп’ютерна інженерія*”*

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

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

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

Ільченку Святославу Вадимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі*

2. Керівник роботи *Марченко Костянтин Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 47-02 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 центру генерації, сертифікації та розподілу ключів у корпоративній мережі*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*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
_____ Марченко К.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Ільченко С.В.
(прізвище та ініціали)

АНОТАЦІЯ

Ільченко С.В. Програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

Метою розробки є програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

Результат роботи – програмна реалізація центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

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

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

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

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, генерація, сертифікація, розподіл ключів, корпоративна мережа

ABSTRACT

Ilchenko S.V. Software for the center of generation, certification and distribution of keys in a corporate network. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the center of generation, certification and distribution of keys in a corporate network.

The purpose of the development is software for the center of generation, certification and distribution of keys in a corporate network.

The result of the work is the software implementation of the center of generation, certification and distribution of keys in a corporate network.

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 PCs with Windows 10/11.

The program was developed in the Builder C++ environment.

Keywords: computer engineering, generation, certification, key distribution, corporate network

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГОМ	–	глобальна обчислювальна мережа
ЛОМ	–	локальна обчислювальна мережа
AS	–	сервер автентифікації
DSA	–	алгоритм шифрування
HAKDP	–	Hashed Key Distribution Patterns
HAKDS	–	Hashed Key Distribution Scheme
KDC	–	центр розподілу ключів
KDP	–	Key Distribution Patterns
Kerberos	–	протокол автентифікації
NAS	–	мережний сервер додатків
OTPS	–	служба одноразових паролів
PKI	–	інфраструктура відкритих ключів
TA	–	довірений центр
TGS	–	Ticket Granting Server
TGT	–	Ticket Granting Ticket
WAN	–	корпоративна мережа

ВСТУП

Актуальність теми. Уявіть, що ви прокидаєтеся і виявляєте, що найчутливіші дані вашої організації були скомпрометовані за одну ніч. У сучасну цифрову епоху це не просто кошмарний сценарій, а реальна та зростаюча загроза. Захист вашої інфраструктури корпоративних мереж є важливішим, ніж будь-коли, оскільки вона слугує основою захисту вашої компанії від кібератак.

Безпека інфраструктури корпоративних мереж стосується стратегій, політик та практик, розроблених для захисту мережевих компонентів організації, таких як маршрутизатори, комутатори та сервери, від різних кіберзагроз та несанкціонованого доступу, забезпечуючи цілісність та функціональність підключених систем.

Забезпечення надійної безпеки інфраструктури корпоративних мереж є важливим для будь-якого бізнесу. Це допомагає захистити конфіденційні дані, підтримувати цілісність системи, запобігати дороговартісним порушенням та забезпечувати дотримання таких норм, як HIPAA, PCI DSS та GDPR.

Безпека інфраструктури корпоративних мереж необхідна для захисту організації від кіберзагроз, які можуть поставити під загрозу конфіденційні дані, порушити бізнес-операції та зашкодити репутації організації.

У сучасному взаємопов’язаному цифровому середовищі різні типи кіберзагроз становлять серйозні ризики для організацій. Шкідливе програмне забезпечення, таке як програми-вимагачі та шпигунські програми, може проникати в мережі та красти конфіденційну інформацію. Фішингові атаки спрямовані на нічого не підозрюючих співробітників, обманом змушуючи їх розкривати конфіденційні дані.

DDoS-атаки можуть перевантажувати мережеві ресурси, спричиняючи простої системи та порушуючи роботу послуг. Без надійних заходів безпеки

інфраструктури корпоративних мереж організації вразливі до цих загроз, що призводить до фінансових втрат, регуляторних санкцій та втрати довіри клієнтів.

Мета й завдання дослідження. Метою роботи є програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

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

– Огляд існуючих центрів генерації, сертифікації та розподілу ключів у корпоративній мережі.

– Дослідження центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

– Програмна реалізація центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі генерації, сертифікації та розподілу ключів у корпоративній мережі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі, 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

Дійсне забезпечення безпеки вашої мережевої інфраструктури – один із найважливіших рішень, які ваш бізнес може зробити для захисту від загроз. Технології та інструменти, такі як правила брандмауера, керування виправленнями та процедури реагування на інциденти, допомагають захистити ваші конфіденційні дані та програми. Цей процес залежатиме від вашого бізнесу, але існує 13-етапний процес, від оцінки вашої мережі до покращення безпеки з часом, який працює для багатьох організацій.

1. Оцініть свою мережу

Перш ніж впроваджувати будь-які практики чи процедури кібербезпеки, вам спочатку потрібно знати поточний стан вашої мережі, який включає існуючі засоби контролю доступу, стан вашого брандмауера та його правил, а також поточні процедури управління вразливостями.

Аудит усіх елементів керування доступом

Проведіть аудит ваших поточних засобів контролю доступу, включаючи імена користувачів, паролі, коди доступу та будь-яку багатофакторну автентифікацію, яку ви наразі налаштували. Щоб перевірити всі засоби контролю доступу, перегляньте кожну програму або систему, яка вимагає облікових даних для входу або дозволів, і задокументуйте їх, зокрема чи захищені вони менеджерами паролів. Також перегляньте ваші поточні вимоги до паролів; чи змушують вони співробітників встановлювати паролі, які важко вгадати?

Перевірте будь-які існуючі брандмауери та правила брандмауера

Проведіть інвентаризацію всіх існуючих мережевих брандмауерів та правил, які у вас є на даний момент. Перейдіть до панелі керування брандмауером, знайдіть список правил та знайдіть будь-які некорисні або

суперечливі правила. Можливо, одне правило суперечить іншому, або старе правило порушує нові політики безпеки вашого бізнесу. Це також гарний час для проведення початкового аудиту брандмауера; він виявить, як брандмауер працює неналежним чином або чи ваші правила більше не відповідають політикам підприємства.

Документуйте свої методи управління вразливостями

Перш ніж капітально ремонтувати інфраструктуру кібербезпеки вашої мережі, задокументуйте всі наявні інструменти або процедури управління вразливостями. Чи є такі, що не працюють, і які ви можете змінити? Також визначте, чи може ваша команда безпеки легко знайти вразливості та усунути їх, чи це був для них складний процес досі. Ви навіть можете надіслати їм опитування з кількома питаннями щодо процесів управління вразливостями та їх усунення.

2. Визначте лазівки та слабкі місця в безпеці

Виявлення вразливостей йде пліч-о-пліч з оцінкою вашої мережі, тому ви можете виконувати ці кроки одночасно. Щоб знайти лазівки в безпеці вашого бізнесу, ви можете впровадити такі стратегії, як сканування вразливостей та тестування на проникнення.

Виконайте сканування на вразливості

Перевірте архітектуру своєї мережі, щоб знати, де існують проблеми. Тести обсягу трафіку або сканування вразливостей виявляють неправильні конфігурації, незастосоване або неправильно застосоване шифрування, слабкі паролі та інші поширені проблеми, перш ніж хакери зможуть їх використати. Ви також можете використовувати сканування вразливостей для виявлення недбалого управління ключами шифрування. Хоча ви можете сканувати вразливості вручну, я рекомендую використовувати програмне забезпечення, яке є ефективнішим.

Розгляньте можливість проведення тестування на проникнення

Сканування вразливостей може виявити поширені слабкі місця, але

активні тести на проникнення визначають, чи вразливості становлять реальний ризик, чи їх можна пом'якшити за допомогою інших заходів контролю. Тести на проникнення також можуть визначити, чи існуючі засоби контролю достатньою мірою зупинять зловмисників. Ви можете проводити тестування на проникнення за допомогою інструментів, але ви можете отримати точніші результати, якщо залучите зовнішніх експертів. Поговоріть зі своїм керівником команди безпеки про можливість найму пентестера.

3. Впроваджуйте контроль доступу

Успішна безпека включає обмеження доступу до мережевих ресурсів, таких як апаратне та програмне забезпечення для керування. Впроваджуйте відповідні засоби контролю доступу для кожного ресурсу, залежно від того, яким співробітникам потрібен доступ у певний час. До них належать паролі, які важко вгадати, інтеграція з Active Directory, багатофакторна автентифікація, стратегії доступу з мінімальними привілеями та доступ до хмарних платформ.

Створіть вагомі повноваження

Збільште вимоги до надійності пароля для створення складності або забезпечте частішу ротацію паролів для всіх облікових даних співробітників. Менеджери паролів допомагають користувачам відповідати суворішим вимогам і можуть забезпечити централізоване керування. Підприємства також можуть використовувати технології єдиного входу (SSO) для оптимізації доступу до хмарних ресурсів.

Використовуйте Active Directory, якщо потрібно

Найменші організації можуть турбуватися лише про доступ до пристроїв, також відомих як облікові дані для входу – імена користувачів та паролі. Але зі зростанням організації формалізований та централізований контроль за допомогою Active Directory (AD) або еквівалентного інструменту Lightweight Directory Access Protocol (LDAP) економить ваш бізнес-час і дозволяє швидше реагувати на запити на зміни. Впровадження AD або LDAP потребує часу, але є цінним для великих організацій.

Впровадження багатофакторної автентифікації

Організації, що розвиваються, стикаються з підвищеним ризиком порушення безпеки, оскільки потенційні збитки від викрадених облікових даних зростають разом із розміром та репутацією компанії. Щоб зменшити цей ризик, багато хто впроваджує багатофакторну автентифікацію для забезпечення покращеної безпеки порівняно з 2FA, особливо коли додатки або токени замінюють вразливі SMS-повідомлення як фактор. Біометричні та безпарольні рішення можуть бути дорожчими, але їх важко підробити.

Використовуйте принцип найменшого доступу

Впровадження стратегії доступу з мінімальними привілеями означає, що ваші мережеві та безпекові команди надають співробітникам доступ до систем лише за необхідності для виконання своєї роботи. У минулому співробітники мали доступ «про всяк випадок», якщо він їм потрібен. Але це ще більше відкриває двері для зловмисників та потенційних внутрішніх загроз. Переконайтеся, що доступ до програми отримують лише співробітники, яким потрібен доступ, і що вони отримують адміністративний доступ або доступ лише для перегляду залежно від своїх ролей.

Керування доступом до хмарних ресурсів

Навіть менші організації зараз використовують хмарні ресурси, але більшість внутрішніх мережевих елементів керування не поширюються на ресурси, розміщені поза мережею, такі як Office 365, Google Docs або окремі мережі філій. Брокери безпеки доступу до хмари (CASB) та захищені браузерні програми можуть забезпечувати консолідовані рішення для захисту користувачів у хмарі.

Переконайтеся, що кожен користувач хмарного облікового запису має відповідні дозволи, незалежно від того, чи є він адміністратором, чи може лише переглядати документ. Також перевірте, чи всі ваші хмарні екземпляри не підключені до Інтернету.

4. Налаштуйте брандмауер

Процес впровадження брандмауера залежатиме від того, чи вже є він у вашій мережі, але ви все одно можете використовувати його як контрольний список для елементів, які ви ще не виконали. Дотримуйтесь цієї загальної процедури, щоб встановити брандмауер, створити правила та зони, а також протестувати та керувати брандмауером з часом.

Виберіть правильний тип брандмауера

Якщо у вашому бізнесі ще немає брандмауера, вам потрібно вибрати той, який підходить для вашої мережі. Меншим підприємствам може знадобитися відносно невеликий пристрій, але більшим підприємствам може знадобитися брандмауер наступного покоління від одного з провідних постачальників мережевого обладнання. Також розгляньте брандмауери як послугу, якщо у вашому бізнесі наразі немає ресурсів або персоналу для підтримки локального брандмауера.

Захистіть брандмауер

Щоб ваш брандмауер працював належним чином, вам потрібно створити спеціальні правила, які визначають, який трафік брандмауер приймає та блокує. Це залежатиме від потреб вашого бізнесу; ви можете налаштувати списки правил, щоб вони були більш або менш обмежувальними залежно від програм і даних, що знаходяться за брандмауером. Впроваджуйте правила як для вхідного, так і для вихідного трафіку, які обмежують трафік, що входить у вашу мережу, і дані, що виходять з неї.

Створення зон брандмауера та IP-адрес

Тепер розділіть свій брандмауер на зони, які потрібно розділити, та призначте кожній зоні необхідні інтерфейси. Потім призначте ресурси та сервери брандмауера відповідними IP-адресами, якщо вони їх ще не мають.

Створіть список контролю доступу

Список контролю доступу (ACL) визначає, яким ресурсам або користувачам дозволено доступ до мережі. Адміністратори можуть вказати

список для всієї мережі або для певних підмереж. Створіть список контролю доступу разом із правилами брандмауера, щоб ніщо в цьому списку не суперечило іншому; добре мати їх поруч під час розробки правил для прийняття або відхилення пакетів.

Тестові конфігурації

Переконайтеся, що всі мережеві конфігурації працюють. Якщо ви блокуєте трафік з певного веб-сайту, переконайтеся, що брандмауер не пропускає цей трафік. Також перевірте правила, особливо для чорних і білих списків ; ви можете зробити це, підключившись до мережі та спробувавши завантажити заблокований веб-сайт. Якщо він все ще завантажується, ваш чорний список не працює.

Керування брандмауером з часом

Брандмауери потрібно регулярно перевіряти та переналаштовувати. Вам також потрібні члени команди, відповідальні за регулярне обслуговування та ремонт брандмауера, включаючи оновлення правил відповідно до змін у бізнес-політиці. Призначте членів команди для виконання конкретних завдань з управління брандмауером та створіть графік аудиту правил брандмауера. Переконайтеся, що кожен член вашої команди знає, як він повинен дбати про брандмауер, використовуючи чітку та просту документацію.

5. Шифрування передачі даних

Шифрування може захистити активи безпосередньо по всій вашій IT-інфраструктурі. Ви можете захистити кінцеві точки за допомогою повного шифрування диска, бази даних за допомогою налаштувань, а критичні файли – за допомогою шифрування файлів або папок.

Шифрування кінцевих точок

Шифрування всього жорсткого диска або SSD кінцевої точки захищає весь пристрій. Крім того, операційні системи, такі як Windows, пропонують опції зміни налаштувань і вимагають шифрованих з’єднань з певними ресурсами або по всій мережі. Ви можете змінити інші налаштування, щоб запобігти передачі

або зберіганню паролів у звичайному тексті та забезпечити зберігання гешів паролів із зашифрованими паролями.

Шифрування баз даних

Ви можете шифрувати бази даних для всієї програми, для кожного стовпця або через механізм баз даних. Різні реалізації шифрування впливатимуть на швидкість запиту даних у базі даних, тому враховуйте це, перш ніж щось шифрувати.

Шифрування файлів або папок

Ви можете шифрувати дані як на рівні окремого файлу, так і на рівні папки. Шифрування на рівні файлу зазвичай займає більше часу та дозволяє шифрувати окремі файли та не шифрувати інші, якщо це необхідно. Шифрування на рівні папки захищає дані всієї папки одночасно, що корисно, коли потрібно захистити всі папки даних, що зберігаються, одночасно.

6. Логічне сегментування мереж

Організаціям, що розвиваються, потрібно дозволяти різні типи доступу, але вони не повинні дозволяти кожному отримувати доступ до всього в мережі. Сегментація мережі може створювати мережі для гостей, карантинні мережі для незахищених пристроїв і навіть окремі мережі для вразливих IoT, OT та відомих застарілих технологій. Використовуйте віртуальні локальні мережі для створення підмереж та впроваджуйте стратегії нульової довіри, щоб користувачі не мали непотрібного доступу.

Налаштування віртуальних локальних мереж

Віртуальні локальні мережі (VLAN) розділяють мережі на одному апаратному забезпеченні та дозволяють командам розділяти мережі на менші підмережі. Вони корисні, оскільки спрощують процеси керування мережею та забезпечують додаткову безпеку, оскільки не весь трафік спрямовується в одне й те саме місце. Ви можете призначити різні типи трафіку для різних підмереж, залежно від ваших потреб безпеки.

Створення підмереж

Сегментація вашої мережі повинна мати сенс, виходячи зі способу маршрутизації трафіку вашим бізнесом. Наприклад, якщо дві підмережі або підмережі розташовані поруч одна з одною у більшій мережі, вам слід встановити брандмауер між ними, якщо одна підмережа обробляє зовнішній трафік, а сусідня містить конфіденційні дані. Також може бути корисним групувати схожі технологічні ресурси в одній підмережі для більш логічної маршрутизації.

Розгляньте нульову довіру

Я рекомендую впровадити систему нульової довіри для всієї вашої мережевої інфраструктури. Система нульової довіри в поєднанні з сегментацією мережі вимагає від користувачів підтвердження того, що вони мають право доступу до кожного окремого ресурсу в мережі. Система нульової довіри використовує концепцію «Ніколи не довіряй, завжди перевіряй». Користувачі в мережі повинні підтвердити своє право на використання програми або вхід у певну систему, а не мати доступ до всього лише тому, що вони пройшли повз брандмауер.

7. Встановлення систем виявлення та запобігання вторгненням

Система виявлення та запобігання вторгненням (IDPS) є однією з основних функцій мережевої безпеки. Вам потрібно знати, що входить у вашу мережу, що з неї виходить, і чи є якісь очевидні вразливості в апаратному та програмному забезпеченні, що складають мережу. Виявлення та запобігання вторгненням можуть функціонувати дещо окремо, але часто вони поєднуються в пакетах безпеки.

Налаштування систем виявлення вторгнень

Системи виявлення вторгнень (IDS) в першу чергу відповідають за виявлення вразливостей та зловмисників. Вони сповіщають мережевих адміністраторів, коли в системі виявляється шкідливе програмне забезпечення, дивний користувач входить у програмне забезпечення або інтернет-трафік несподівано перевантажує сервер. Вони корисні для виявлення шкідливої

поведінки, але зазвичай не можуть самостійно вирішити проблеми безпеки.

Налаштування систем запобігання вторгненням

Системи запобігання вторгненням не просто виявляють вразливості та атаки – вони відповідають за їх виправлення. Це включає стандартні заходи щодо усунення недоліків, такі як блокування трафіку або видалення шкідливого програмного забезпечення. Виявлення та запобігання вторгненням, як правило, найефективніші, коли вони об'єднані разом, що дозволяє виявляти проблеми та виправляти їх на одній платформі.

8. Створіть політики виявлення активів

Несанкціоновані пристрої можуть перехоплювати або перенаправляти мережевий трафік за допомогою таких атак, як підключення несанкціонованих комп’ютерів до мережі або розгортання сніферів пакетів для перехоплення мережевого трафіку. Аналогічно, підроблені адреси системи доменних імен (DNS) можуть перенаправляти користувачів із легітимних підключень на небезпечні веб-сайти. Щоб захистити свою мережу, блокуйте або кладіть у карантин ресурси за потреби, завжди скануйте ресурси та вимикайте будь-які мережеві функції, які вам не потрібні.

Блокування або карантин пристроїв

Рішення для контролю доступу до мережі (NAC) тестують кінцеві точки на наявність застарілого або вразливого програмного забезпечення та перенаправляють пристрої на карантин до усунення недоліків. Несанкціоновані пристрої можуть бути заблоковані або поміщені на карантин. Ви можете реалізувати деякі можливості NAC, додавши фільтрацію MAC-адрес або білі списки до брандмауерів та серверів, але обслуговування білих списків може займати багато часу.

Безперервне сканування активів

Інструменти управління IT-активами (ITAM) можуть сканувати пристрої, підключені до мережі, та надсилати сповіщення або блокувати незареєстровані пристрої. Організаціям необхідно перевіряти типи активів, які вони намагаються

виявити. Деякі програми, хмарна інфраструктура, мережеве обладнання або пристрої Інтернету речей (IoT) можуть потребувати складнішого ITAM або додаткових інструментів для їх виявлення.

Вимкніть непотрібні функції

Будь-який невикористаний порт доступу в брандмауері, непотрібний віддалений доступ (сховище, принтер, маршрутизатори тощо) та подібні функції часто залишатимуться поза контролем. Хакери намагатимуться знайти та використати ці можливості. Краще просто вимкнути їх, якщо вони непотрібні. З цієї причини організаціям також слід вимкнути можливості Universal Plug and Play (UPnP) після завершення налаштування, оскільки хакери знайшли способи використовувати функції автоматизації для завантаження шкідливого програмного забезпечення.

9. Розробіть процедури управління виправленнями

Захист мережевого обладнання та програмного забезпечення вимагає від команд безпеки постійного оновлення своїх продуктів до найновішої версії. Цей процес включає якомога швидше встановлення виправлень, створення призначень виправлень та ретельний контроль бюлетенів безпеки ваших постачальників.

Негайно виправити

Чим швидше ваш бізнес виправляє апаратне та програмне забезпечення, тим менше часу зловмисники матимуть, щоб скористатися вразливостями в них. Якщо ваш бізнес виявить, що у вас обмежений час для виправлення ресурсів, коригування процесів і завдань, тому виправлення є вищим пріоритетом. Якщо ви впроваджуєте платформи безпеки з вбудованими функціями керування виправленнями, ви отримуватимете нагадування про виправлення, що допоможе вашій команді швидше виправляти вразливості.

Призначення ролей керування виправленнями

Розробіть графік оновлення стандартних версій для всіх ваших мережевих ресурсів. Графік оновлення включає призначення ролей членам команди, щоб

кожен знав, хто відповідає за оновлення кожного пристрою та версії програмного забезпечення. Я рекомендую створити простий документ, який чітко визначає, хто відповідає за оновлення кожного апаратного та програмного забезпечення, а також дні та час їх оновлення.

Новини та релізи щодо вразливостей монітора

Частиною ефективної стратегії управління виправленнями є проактивне вирішення проблем безпеки шляхом моніторингу інформації про вразливості ваших постачальників та світових галузевих новин загалом. Нові вразливості з'являються щотижня, і часто вони проявляються в мережевому обладнанні та операційних системах. Чим швидше ви дізнаєтеся про проблеми, тим швидше ви зможете їх виправити та уникнути експлойту нульового дня або подібної атаки.

10. Моніторинг та ведення журналу мереж

Ви можете не одразу розпізнати мережевий трафік як шкідливий, але його моніторинг за допомогою команди управління інформацією та подіями безпеки (SIEM), центру операцій безпеки (SOC), керованого виявлення та реагування (MDR) або подібної команди може виявити незвичайну поведінку. Ці команди також можуть реагувати на сповіщення та усувати атаки, які уникають автоматичного реагування. Пісочниця також є варіантом, якщо ви хочете додатково проаналізувати дивну поведінку у вашій мережі.

Призначити ресурси та команди моніторингу

Індустрія кібербезпеки пропонує безліч продуктів і послуг, які моніторять мережі, і якщо ваші бізнес-лідери не знають, який саме обрати, уважно ознайомтеся з функціями кожного з них:

– SIEM: Зосереджений на агрегації корпоративних даних та журналів, що часто вимагає значного моніторингу та управління.

– SOC: Керує щоденними операціями безпеки за допомогою команди аналітиків та співробітників служби безпеки, як внутрішніх, так і зовнішніх по відношенню до вашої компанії.

– Виявлення та реагування на кінцеві точки (EDR): знаходить та усуває

загрози безпеці, зокрема на кінцевих пристроях.

– MDR: Надає керовані послуги виявлення та реагування, щоб ваш бізнес міг скористатися технологіями та аналітичними даними зовнішніх аналітиків.

Реагуйте на сповіщення

Реагування на сповіщення безпеки – це або ваша робота, або робота вашого операційного центру чи постачальника керованих послуг. Незалежно від того, хто цим займається, вам потрібно буде точно визначити, який член команди відповідає за яку частину процесу сортування сповіщень. Це покращує швидкість реагування та збільшує ймовірність того, що ваша команда безпеки ефективніше справлятиметься із загрозами.

Використовуйте пісочницю

Якщо ви виявили шкідливе програмне забезпечення у своїй мережі та хочете дізнатися більше про його закономірності, розгляньте можливість використання продукту для ізоляції. Вони допомагають вашій команді спостерігати за тим, як працюють шкідливі програми, у безпечному, контрольованому середовищі. Вони часто входять до складу більших пакетів безпеки, таких як рішення для керованого виявлення та реагування, але ви також можете придбати їх окремо.

11. Розробіть план реагування на інциденти

Вашому бізнесу завжди потрібен план реагування на інциденти, щоб знати, як обробляти події безпеки, незалежно від того, наскільки мала ваша команда безпеки. План реагування на інциденти повинен чітко перераховувати кожен крок, який ваша команда повинна зробити для зменшення загроз. Деякі з найпоширеніших характеристик плану реагування на інциденти включають налаштування, гнучку структуру та належну методологію сповіщень.

Створюйте налаштовувані плани для різних ситуацій

Цілком ймовірно, що вам знадобиться більше однієї ітерації плану реагування на інциденти, а не просто один список кроків, який працює в кожній ситуації. Створення загального шаблону, а потім кількох різних, більш

конкретних планів часто є гарною стратегією для налаштування плану реагування на інциденти для різних типів інцидентів безпеки. Реакції відрізнятимуться залежно від вразливості чи атаки, як і конкретний план для кожного з них.

Будьте гнучкими, коли процес потребує змін

Хоча плани реагування на інциденти потребують упорядкованих кроків, вони також повинні мати певний простір для маневру на випадок, якщо процес потрібно буде змінити в останню хвилину. Це може виглядати як перелік кількох додаткових членів команди, які візьмуть на себе виконання, якщо відповідальна особа за один крок буде у відпустці або захворіє, або як надання різних варіантів пом'якшення загрози безпеці, якщо один з них не спрацює.

Розробіть процедури оповіщення, які мають сенс

Командам безпеки доводиться сортувати багато інформації, і не всі сповіщення є точними. Розробіть процедури для сортування сповіщень та відокремлення хибнопозитивних результатів від проблем, які вам дійсно потрібно розслідувати. Автоматизація тут корисна – ви заощадите свою команду реагування на інциденти, якщо у них буде точне програмне забезпечення, яке вказує їм, яким сповіщенням надавати пріоритет.

12. Навчати співробітників методам кібербезпеки

Користувачі залишаються одним із найпоширеніших джерел порушень безпеки, оскільки всі роблять помилки, а більшість співробітників не є експертами з безпеки. Навчання співробітників та тестування на проникнення – це дві основні тактики, які компанії використовують для того, щоб їхній персонал був в курсі загроз, але щоденні розмови в командах також відіграють ключову роль у захисті компанії.

Навчіть свої команди основам безпеки

Курси навчання з кібербезпеки для підприємств, малого та середнього бізнесу надають фундаментальні інструкції, які дозволяють співробітникам зробити свій внесок у покращення практик безпеки для всієї організації. Вони

висвітлюють такі проблеми, як фішингові атаки, шкідливе програмне забезпечення, небезпечні методи роботи з паролями та скомпрометоване обладнання, таке як USB-накопичувачі. Вони також зменшують ймовірність того, що кібератака застане співробітників зненацька.

Виконайте тестування на проникнення

У сценарії тестування на проникнення внутрішні або зовнішні хакери намагаються проникнути в мережу бізнесу та знайти вразливості в ній. Але деякі стратегії тестування на проникнення можуть також включати соціальну інженерію, яка виявляє, де потрібно навчати співробітників. Будьте обережні, щоб не очорнити співробітників, які роблять помилки, та заохочуйте максимально прозорі обговорення у вашій організації.

Регулярно ведіть розмови

Не применшуйте важливість частих обговорень кібербезпеки. Чим більше співробітники, але особливо керівники, обговорюють зі своїми колегами загрози та вразливості, тим краще ви будете підготовлені до вирішення цих питань. Обговорення безпеки також заважає співробітникам приймати необдумані рішення.

13. Постійно вдосконалюйте мережу

Жодна безпека не є безпомилковою. Вразливості, неправильні конфігурації, помилки та досвідчені зловмисники можуть призвести до порушення мережевої та іншої безпеки. Навіть найнадійніший стек безпеки та найстійкіша мережа розпадуться без обслуговування. Оновлення програмного забезпечення та облікових даних за замовчуванням, вимкнення застарілих протоколів та проведення регулярних аудитів безпеки мережі допоможуть вашій організації бути в курсі вдосконалень мережі.

Автоматичне оновлення систем

Часто можна налаштувати локальні мережеві маршрутизатори, брандмауери та інше обладнання на автоматичне завантаження нових оновлень, щоб пристрої та прошивка не були вразливими. Однак майте на увазі, що збої

живлення під час оновлень – або оновлення з помилками – можуть призвести до виходу обладнання з ладу.

Зміна облікових даних за замовчуванням

Маршрутизатори та інше обладнання зазвичай постачаються з публічно оприлюдненими налаштуваннями та назвами за замовчуванням, але це широко відчинені двері для хакерів. Адміністратори мережі повинні змінити паролі маршрутизаторів за замовчуванням, щоб захиститися від несанкціонованого доступу. Федеральна торгова комісія США (FTC) надає ширші рекомендації щодо захисту домашніх мереж Wi-Fi та інші поради для офісів та споживачів.

Не використовуйте застарілі протоколи

IT-обладнання постачається зі зворотною сумісністю, але це може бути проблематично, оскільки це включає підтримку застарілих та небезпечних опцій. Я рекомендую вимкнути незахищені протоколи та порти, такі як FTP або SMBv1, у всій екосистемі, щоб запобігти майбутнім атакам. Використовуйте консоль керування мережею, щоб вимкнути будь-які застарілі протоколи, які ви більше не хочете, щоб ваша мережа дозволяла.

Аудит вашої мережі

Регулярно проводите аудит усієї вашої мережі, щоб постійно виявляти вразливості та слабкі місця з часом. Аудит повинен охоплювати як апаратне, так і програмне забезпечення, щоб ваші комутатори, маршрутизатори, операційні системи, комп’ютери та сервери тестувалися та перевірялися на безпеку та продуктивність.

Підсумок: Захист мережі – це безперервний процес

Мережі утворюють місток між користувачами та їхніми комп'ютерами з одного боку та ресурсами, до яких їм потрібно дістатися, з іншого. Мережева безпека захищає міст, але для забезпечення безпеки кожен кінець мосту також має бути захищений засобами безпеки для користувачів, програм, даних та ресурсів. Кожен компонент стратегії безпеки посилює та захищає організацію в цілому від збою будь-якого конкретного компонента.

Команди IT-безпеки повинні не лише підтримувати обізнаність про свої поточні та майбутні потреби, але й чітко повідомляти про ці потреби нетехнічним зацікавленим сторонам, щоб отримати бюджети та іншу підтримку.

1.2 Область застосування

Областю застосування розроблювального програмного забезпечення є корпоративна мережа. Недостатній рівень безпеки інфраструктури корпоративних мереж може призвести до серйозних наслідків, таких як витік даних, фінансові втрати, юридичні штрафи та пошкоджена ділова репутація.

Одним із основних наслідків слабкої мережевої безпеки є потенційні значні фінансові втрати через крадіжку критично важливих даних або порушення бізнес-операцій. У сучасному цифровому середовищі клієнти надають пріоритет захисту даних, а порушення безпеки може призвести до повсюдної втрати довіри та лояльності.

Недотримання нормативних вимог може призвести до значних штрафів та юридичних наслідків, що ще більше виснажують фінансові ресурси. Такі стягнення можуть бути каральними, а також можуть зашкодити репутації організації в галузі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі, 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

Витік даних може статися миттєво – чи то через проникнення хакера в мережу, чи то через втрату ноутбука керівником. Без належних заходів безпеки конфіденційна інформація стає вразливою до крадіжки та використання.

Повне шифрування диска пропонує критично важливу першу лінію захисту, захищаючи жорсткі диски, зовнішні сховища та інші системи від несанкціонованого доступу. Як простий, але потужний захід безпеки, воно забезпечує міцну основу для захисту даних і має бути фундаментальним кроком у стратегії кібербезпеки будь-якої організації.

Ось наш вибір найкращих програмних рішень для шифрування дисків на 2025 рік:

– BitLocker**:** найкращий варіант для повного шифрування диска Windows.

– McAfee**:** Найкраще рішення для безпеки корпоративного рівня.

– Trend Micro**:** Найкращий варіант для захисту кількох кінцевих точок.

– Sophos**:** Найкращий варіант для шифрування, керованого IT-фахівцями.

– FileVault**:** найкращий варіант для повного шифрування диска macOS.

– VeraCrypt**:** Найкращий варіант для повного шифрування диска з відкритим кодом.

– Check Point**:** Найкращий варіант для високобезпечної автентифікації.

BitLocker – найкращий варіант для повного шифрування диска Windows

BitLocker – це вбудований інструмент повного шифрування диска від Microsoft, інтегрований у версії Windows Pro та Enterprise. Якщо ви

використовуєте пристрої з ОС Windows, це гарне місце для початку. Він використовує апаратне забезпечення модуля довіреної платформи (TPM) для автоматичного шифрування та підтримує регулярні оновлення Windows без проблем із розшифруванням.

Переваги:

– Без додаткових витрат.

– Централізуйте управління за допомогою інтеграції з Microsoft Intune.

– Підтримка TPM для автоматичного захисту.

Недоліки:

– Ексклюзивно лише для Windows Pro/Enterprise.

– Немає підтримки зашифрованих контейнерів.

– Проблеми прозорості через закритий вихідний код.

Ціноутворення:

– Входить до комплекту версій Windows 10/11 Pro, Enterprise та Education.

– Потрібне корпоративне ліцензування для розгортань у корпоративному середовищі (наприклад, Microsoft 365 E3/E5).

Основні характеристики:

– Автентифікація перед завантаженням за допомогою PIN-коду/ключа запуску для офлайн-атак.

– Розблокування мережі для віддаленого розшифрування без пароля в керованих середовищах.

– Перевірено відповідно до FIPS 140-2 для використання урядом та відповідно до вимог.

– Інтеграція з Azure Active Directory для гібридних хмарних робочих процесів.

McAfee – найкращий засіб для безпеки корпоративного рівня

McAfee Endpoint Encryption виділяється як одне з найкращих програм для шифрування на ринку, оскільки воно розроблене для підприємств, яким потрібно захистити конфіденційні дані на керованих пристроях. Його апаратне

шифрування та попередня автентифікація забезпечують дотримання HIPAA та GDOR, хоча невеликі команди можуть вважати це складним.

Переваги:

– Доступне повне шифрування диска та файлів.

– Підтримка кількох пристроїв.

– Поєднує FDE з інструментами DLP, захисту від крадіжок та віддаленого стирання даних.

Недоліки:

– Для повної функціональності потрібен репозиторій McAfee.

– Крута крива навчання для користувачів, які не є корпоративними користувачами.

– Обмежена безкоштовна версія.

Ціноутворення:

– Базовий: $29.99 (1 пристрій).

– Основний: $39.99 (5 пристроїв).

– Преміум: $49.99 (безлімітний).

– Розширений: $89.99 (без обмежень).

– Ultimate: $199.99 (безлімітний).

Основні характеристики:

– Автентифікація перед завантаженням з інтеграцією Active Directory та SSO.

– Апаратне шифрування AES-256 (підтримка TPM).

– Віддалене відстеження геолокації та видалення даних із втрачених пристроїв.

– Аналітика загроз у режимі реального часу завдяки McAfee Global Threat Intelligence.

Trend Micro – найкращий варіант для захисту кількох кінцевих точок

Шифрування кінцевих точок Trend Micro є частиною Trend Micro Smart Protection Network, хмарної клієнтської інфраструктури безпеки контенту, яка

надає глобальну аналітику загроз. Цей інструмент захищає гібридні середовища Windows/macOS за допомогою повного шифрування диска та захисту знімних носіїв. Окрім шифрування, він пропонує автентифікацію перед завантаженням та детальний контроль політик для запобігання несанкціонованому доступу до USB-накопичувачів, зовнішніх накопичувачів та дисків.

Переваги:

– Кросплатформне FDE + шифрування USB-накопичувача.

– Централізоване управління через Trend Micro control manager.

– Автентифікація перед завантаженням.

Недоліки:

– Може бути складним для невеликих команд.

– Для повноцінної роботи потрібна екосистема Trend Micro.

– Немає вбудованої підтримки Linux.

Ціноутворення:

– Початкова ціна: $45/користувач/рік (орієнтовно для невеликих команд).

– Підприємство: Індивідуальне Ціноутворення: для великих розгортань.

Основні характеристики:

– Автоматичне шифрування для знімних носіїв (USB, зовнішніх накопичувачів).

– Звітність про відповідність HIPAA, PCI DSS та GDPR.

– Контроль доступу на основі ролей та віддалене стирання даних з пристрою.

– Інтеграція з Trend Micro Apex One для виявлення загроз XDR.

Sophos SafeGuard – найкращий варіант для шифрування, керованого IT-підрозділами

Sophos SafeGuard – це рішення для захисту та шифрування даних на різних пристроях, мережах та хмарі. Воно пропонує централізоване повне шифрування диска з детальним контролем політик для підприємств, які керують розподіленими пристроями. Воно інтегрується з платформою Sophos XDR для

єдиного виявлення загроз та моніторингу порушень.

Переваги:

– Централізоване управління політиками для понад 10 000 кінцевих точок.

– Портал самостійного відновлення для скидання паролів.

– Автентифікація перед завантаженням із синхронізацією Active Directory.

Недоліки:

– Для повноцінної роботи потрібна система Sophos.

– Немає версії для особистого використання.

– Немає вбудованого керування macOS.

Ціноутворення:

– Початкова ціна: $44/користувач/рік (орієнтовно для невеликих команд).

– Підприємство: Індивідуальне Ціноутворення: для великих розгортань.

Основні характеристики:

– Повне шифрування диска для Windows та знімних носіїв.

– Контроль доступу на основі ролей та ведення журналу аудиту.

– Інтеграція з Sophos Intercept X для захисту від програм-вимагачів.

– Віддалене стирання та відстеження геолокації пристрою.

FileVault – найкращий варіант для повного шифрування диска macOS

FileVault – це безкоштовний інструмент шифрування, який захищає дані вашого Mac, шифруючи весь диск. Це вбудований інструмент FDE від Apple для macOS, який пропонує користувачам шифрування XTS-AES-128 з відновленням iCloud.

Переваги:

– Вбудовано в macOS.

– Автоматичне резервне копіювання ключа відновлення iCloud.

– Апаратне прискорення шифрування за допомогою Apple Silicon/T2 chios.

Недоліки:

– Обмежено macOS.

– Немає зашифрованого контейнера.

– Без віддаленого керування.

Ціноутворення:

– Безкоштовно входить до комплекту macOS (без додаткової оплати).

Основні характеристики:

– Автентифікація перед завантаженням з відновленням Apple ID.

– Миттєве шифрування без затримки продуктивності на Apple Silicon.

– Інтеграція з macOS Keychain для керування паролями.

– FileVault 2 (поточна версія) шифрує весь системний диск.

VeraCrypt – найкращий варіант для повного шифрування диска з відкритим кодом

VeraCrypt – це безкоштовне програмне забезпечення для шифрування дисків з відкритим кодом для Windows, Mac OSX та Linux. На відміну від інших інструментів, VeraCrypt пропонує повну прозорість та кросплатформну підтримку. Це робить його ідеальним для широкої цільової аудиторії – досвідчених користувачів, організацій, технічних гуру та захисників конфіденційності.

Переваги:

– Відкритий код та можливість перевірки на предмет прозорості.

– Підтримує FDE, контейнери.

– Приховані томи для додаткової безпеки.

Недоліки:

– Крута крива навчання.

– Відсутність централізованого управління для підприємств.

– Ручні оновлення та відсутність офіційної підтримки.

Ціноутворення:

– Безкоштовно: Без витрат (з відкритим вихідним кодом).

Основні характеристики:

– Шифрує весь розділ або пристрій зберігання даних, такий як USB-флеш-накопичувач або жорсткий диск.

– Шифрує розділ або диск, на якому інстальовано Windows (автентифікація перед завантаженням).

– Шифрування відбувається автоматично, у режимі реального часу (на льоту) та прозоро.

– Забезпечення шифрування для Windows/macOS/Linux.

Check Point – найкращий варіант для високобезпечної автентифікації

Check Point забезпечує комплексну безпеку, автоматично шифруючи всі томи жорсткого диска, включаючи системні файли, тимчасові файли, приховані томи та навіть видалені файли. Ключовою особливістю Check Point FDE є його передзавантажувальний захист, який вимагає від користувачів автентифікації перед завантаженням операційної системи.

Переваги:

– Автентифікація перед завантаженням.

– Кросплатформне FDE (Windows, macOS, Linux).

– Детальні журнали аудиту для звітності про відповідність.

Недоліки:

– Високі ціни для некорпоративних користувачів.

– Потрібна екосистема Check Point для розширених функцій.

– Потрібне складне розгортання.

Ціноутворення:

– Підприємство: Індивідуальне Ціноутворення: (ліцензування на основі обсягу).

Основні характеристики:

– Підтверджене FIPS 140-2 шифрування AES-256.

– Централізоване управління через Check Point Security Gateway.

– Віддалене стирання та відстеження геолокації пристрою.

– Інтеграція з Check Point SandBlast для запобігання загрозам нульового дня.

Як вибрати найкраще програмне забезпечення для повного шифрування диска для вашого бізнесу

Вибір правильного програмного забезпечення для повного шифрування диска важливий для захисту конфіденційних даних вашого бізнесу, зберігаючи при цьому зручність використання та ефективність вашого робочого процесу.

Ви повинні оцінити такі функції безпеки, як захист файлів, шифрування хмарного сховища та захист USB-накопичувача. Перш ніж вибрати рішення, врахуйте сумісність пристроїв і запитайте, чи має інструмент відмінні показники безпеки.

Хоча більшість інструментів мають чудові функції шифрування, з часом вони можуть втрачати належний інтеграційний стек та загальну продуктивність. Це означає, що вам потрібно розглянути інструмент з хорошою підтримкою клієнтів. Навіть якщо ми не розглядали це питання безпосередньо, варто перевірити.

Зрештою, для чого ви хочете використовувати це програмне забезпечення? Розгляньте такі інструменти, як **ESET**, **BitLocker** та **FileVault**, якщо вони необхідні для роботи. Однак, якщо вам потрібне рішення корпоративного рівня або високоякісне програмне забезпечення для шифрування, зверніть увагу на такі рішення, як Symantec, McAfee, Veracrypt та Check Point.

Як ми оцінювали програмне забезпечення для повного шифрування диска

Спочатку ми створили чотири зважені категорії, що містять ключові підкритерії, для оцінки найкращого програмного забезпечення для повного шифрування диска. Ми зважили критерії та функції на основі відсотків, зазначених для кожної з них нижче, і ці вагові фактори врахували в загальному балу для кожного продукту. 10 продуктів, які отримали найвищі бали в цій рубриці, потрапили до нашого списку. Однак це не означає, що один із них

автоматично є найкращим вибором для вас або що поза цим списком не існує хороших альтернатив.

Прозорість Ціноутворення:**: 10%**

Ми оцінили, чи був постачальник прозорим щодо Ціноутворення: та чи мав продукт безкоштовну пробну версію, зокрема її тривалість.

Основні функції безпеки: 30%

Щоб визначити рівень безпеки кожного інструменту, ми зосередилися на основних можливостях шифрування, таких як повне шифрування диска, шифрування файлів і папок, а також автентифікація перед завантаженням.

Розширені функції шифрування: 35%

Ми оцінили наявність передових технологій шифрування, включаючи автентифікацію перед завантаженням, приховані томи та багаторівневі підходи до шифрування. Ми також розглянули, чи програмне забезпечення безперешкодно шифрує зовнішні накопичувачі та USB-пристрої, запобігаючи витоку даних під час передачі файлів.

Підтримка кросплатформності: 25%

Оскільки користувачі працюють з різними операційними системами, ми оцінили, наскільки добре кожен інструмент шифрування підтримував платформи Windows, macOS, Linux та мобільні пристрої. Ми вибрали нативні рішення для шифрування всього диска, такі як **FileVault** та **BitLocker**. Ми також перевірили, чи було шифрування однаково ефективним на всіх системах, чи деякі версії мали обмежену функціональність.

Підсумок

Вибір програмного рішення для повного шифрування диска може значно покращити безпеку та контроль над вашими файлами, SaaS та хмарними ресурсами.

Ви повинні ретельно розуміти бюджет, вимоги та досвід персоналу вашої компанії. Після того, як ви визначите всі ці пункти, складіть короткий список з вашими найважливішими потребами.

Повне шифрування диска може бути обмеженим інструментом, але воно є критично важливим у бізнес-середовищі. Низькі зусилля для впровадження FDE та знижений ризик втрати або крадіжки даних більш ніж компенсують потенційні обмеження. FDE пропонує високоцінне рішення, яке всі організації повинні серйозно розглянути щодо додавання до свого стеку безпеки.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++ , тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об’єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм

та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують часові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції додатку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

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 центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

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 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 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

Мережева безпека створює захищений, контрольований та безпечний зв'язок між користувачами та ресурсами. Незважаючи на швидку еволюцію того, що являє собою користувач, ресурси та з'єднання, основи мережевої безпеки залишаються незмінними: блокування зовнішніх загроз, захист внутрішніх мережевих комунікацій, моніторинг мережі на наявність внутрішніх та зовнішніх загроз і забезпечення доступу користувачів лише до авторизованих частин мережі.

Захист зростаючого, розлогого та часом конфліктуючого набору технологій, що складають мережеву безпеку, створює постійні труднощі для фахівців з безпеки.

Щоб зрозуміти зміну вимог і технологій, нам потрібно дослідити контраст між попередніми простими мережами та новішими, складними мережами.

Старіші прості мережі

Спочатку прості мережі з'єднували окремих користувачів та їхні комп'ютери (спочатку термінали) з маршрутизатором або комутатором. Маршрутизатор або комутатор потім сприяв зв'язку між іншими пристроями в мережі, такими як інші сервери, мережеві сховища та принтери. З появою Інтернету до мережі було додано брандмауер для захисту мереж та користувачів під час їхнього підключення до всесвітньої мережі.

У цьому простому середовищі мережева безпека дотримувалася простого протоколу:

– Автентифікувати користувача: за допомогою комп'ютерного входу (ім'я користувача + пароль).

– Перевірте дозволи користувача: за допомогою Active Directory або

подібного протоколу LDAP (Lightweight Directory Access Protocol).

– Увімкнути зв'язок з авторизованими мережевими ресурсами (серверами, принтерами тощо).

– Моніторинг локальних комунікацій: використання пасивних систем виявлення вторгнень (IDS) або активних систем запобігання вторгненням (IPS).

– Моніторинг авторизованих та блокування несанкціонованих з'єднань поза мережею: використання брандмауерів.

Ця структура передбачала, що всі необхідні ресурси знаходяться в мережі, весь зв'язок у мережі авторизований, а всі користувачі автентифіковані. У складній сучасній мережі це припущення руйнується.

Новіші складні мережі

Сучасні мережі охоплюють набагато більше користувачів, ресурсів та типів з'єднань, ніж передбачалося спочатку. Деякі основні кроки залишаються незмінними, але деталі стали набагато складнішими.

Користувачі тепер охоплюють як локальних, так і віддалених співробітників. Постачальники, клієнти та підрядники також часто отримують доступ до ресурсів у мережі або безпосередньо підключених до неї. Програми та служби тепер також виступають у ролі користувачів, оскільки вони активно підключаються до пристроїв, баз даних та програм у мережі безпосередньо або через API.

Пристрої тепер складаються як з корпоративних пристроїв, так і з неконтрольованих пристроїв BYOD, що складаються з комп'ютерів, ноутбуків, планшетів і мобільних телефонів. Як віртуальні, так і фізичні сервери, кінцеві точки та контейнери можуть розташовуватися в локальних центрах обробки даних або віддалено у філіях, або ж розміщуватися в хмарі. Інтернет речей (IoT), операційні технології (OT) та промисловий інтернет речей (IIoT) також тепер підключаються до мереж.

З’єднання все ще охоплюють фізичні комутатори та маршрутизатори, що підключаються до мережі, але тепер також включають бездротові стільникові

мережі, мережі Wi-Fi, віртуальні мережі, хмарні мережі та підключення до Інтернету.

Точки доступу все ще потребують брандмауерів, але часто включають методи для зовнішніх підключень до мережі, такі як віртуальні приватні мережі (VPN), протокол віддаленого робочого столу (RDP) та мережеві контролери для захисту віддалених користувачів за допомогою віддалених ресурсів (безпечні шлюзи, безпечний доступ до служби доступу на межі доступу (SASE) тощо).

Дані зараз існують по всій організації всередині додатків (бази даних, електронна пошта тощо), у хмарних репозиторіях (бази S3, озера даних тощо), у додатках «Програмне забезпечення як послуга» (CRM, сервіси обміну файлами тощо), на мобільних пристроях, домашніх та міжнародних офісах та багато іншого. Засоби контролю та інструменти мережевої безпеки в ідеалі повинні захищати, контролювати та захищати дані в усіх цих середовищах.

Як працює мережева безпека

Мережі з'єднують активи для зв'язку. Мережева безпека забезпечує цілісність і конфіденційність зв'язку, а також перевіряє відповідні з'єднання на основі особи користувача, типу пристрою та класифікації даних.

Засоби контролю безпеки мережі

Мережева безпека виконує ці завдання шляхом впровадження, підтримки та забезпечення дотримання фізичних, технічних та адміністративних засобів контролю для полегшення авторизованого використання та заборони доступу до загроз і неналежного використання мережевих ресурсів.

Більшість постачальників мережевої безпеки зосереджуються на наданні апаратних та програмних рішень для забезпечення технічних засобів контролю, які використовують програми для авторизації, автентифікації, сприяння, захисту та моніторингу мережевого трафіку. Технічні засоби контролю можуть бути впроваджені шляхом:

– Апаратне забезпечення: комутатори, маршрутизатори, брандмауери тощо.

– Програмне забезпечення: віртуальні маршрутизатори, системи запобігання вторгненням (IPS) тощо.

– Конфігурація: закриття портів, що використовуються для трафіку між вузлами, скидання несправних комунікаційних пакетів тощо.

Організації несуть відповідальність за забезпечення того, щоб фізичні засоби контролю забезпечували лише авторизований доступ та зміни до мережевого обладнання, такого як маршрутизатори, кабелі, комутатори, брандмауери та інші мережеві пристрої. Ця вимога може бути впроваджена постачальниками в розміщених або колокаційних середовищах, але організація зберігає відповідальність за перевірку виконання цієї вимоги.

Усі організації впроваджують принаймні рудиментарні адміністративні засоби контролю, додаючи імена користувачів до LDAP або Active Directory. Однак багато організацій можуть покращити можливості мережевої безпеки за допомогою більш детальних політик. Мережева безпека забезпечує дотримання політик за допомогою технічних інструментів, а детальні політики дозволяють здійснювати детальний контроль доступу на основі груп користувачів, потреб користувачів, обмежень за часом або інших правил, які можуть обмежувати автентифікацію та доступ до мережі або дозволяти зміни в мережі.

Для забезпечення безпеки зв'язку мережеве обладнання використовує різноманітні протоколи для перевірки цілісності пакетів та доставки пакетів на правильні пристрої. Шифрування регулярно використовуватиметься для захисту даних від перехоплення. Щоб забезпечити відповідні з'єднання, мережі зіставляють користувачів, пристрої, ресурси та дозволи на основі політик, встановлених в Active Directory або аналогічних рішеннях LDAP.

Мережева безпека виключає будь-який несанкціонований доступ до активів або зв'язку. Краща мережева безпека контролює спроби перевищення дозволів, незвичну поведінку авторизованих користувачів та мережеву активність, яка може свідчити про компрометацію або активність шкідливого програмного забезпечення.

Щоб перевірити ефективне та належне впровадження мережевої безпеки, слід проводити аудит для підтвердження успішного впровадження політик та заходів контролю. Часто аудит проводиться шляхом перегляду журналів мережі, але тестування на проникнення та сканування вразливостей також можуть бути використані для перевірки правильності впровадження та конфігурації.

Мережеві рівні та інкапсуляція даних

Модель Opens Systems Interconnection (OSI) поділяє комунікацію на сім різних рівнів. На практиці ці рівні часто можуть бути розмитими, але модель допомагає в концептуалізації та розробці стандартів. Фахівці з мережевої безпеки розуміють, що комунікація відбувається на всіх рівнях, і рішення безпеки повинні певним чином охоплювати всі рівні.

1. Фізичні: біти, що проходять через дроти або всередині бездротових сигналів

2. Канал передачі даних: Розбиває зв'язок на фрейми даних або менші фрагменти даних, що містять частину передачі даних між двома вузлами: будь-яким пристроєм, що надсилає або приймає дані в мережі, таким як комп'ютери, маршрутизатори, комутатори, шлюзи тощо; менші розміри фреймів забезпечують ефективнішу передачу та перевірку помилок.

3. Мережа: пакети даних, що додають дані адресації, маршрутизації та керування трафіком до кадру даних

4. Транспорт: кадри даних змінного розміру передаються від хоста до хоста через мережу, підтримуючи якість обслуговування.

5. Сеанс: створює, контролює та завершує зв'язок між відправником та одержувачем.

6. Презентація: Встановлює форматування та перетворення даних, включаючи такі функції, як шифрування, стиснення та графічні команди.

7. Застосування: Безпосередня взаємодія з програмним застосунком, який ініціює та отримує зв'язок, таким як браузер, програма для обміну файлами тощо.

Пакет інтернет-протоколів (TCP/IP) використовує спрощену модель, яка

описує інкапсуляцію даних, що передаються протоколом:

1. Застосунок: створює дані для надсилання.

2. Транспорт: здійснює зв'язок між хостами.

3. Інтернет: Бере дані транспортного рівня та інкапсулює їх за допомогою заголовка IP-адреси, який приховує інформацію про підключення до локальної мережі та направляє дані до кінцевого хоста.

4. Зв'язок: Розбиває дані на фрейми та додає заголовки та нижні колонтитули фреймів; керує локальними мережевими з'єднаннями для передачі.

Принцип безпеки: глибоко ешелонований захист

Ешелонований захист застосовується до всіх видів кібербезпеки. Ця концепція визнає, що жоден окремий засіб контролю безпеки навряд чи зможе вирішувати всі загрози постійно. Натомість необхідно впровадити кілька типів засобів контролю, які підсилюють один одного, щоб зменшити ризики, навіть якщо один засіб контролю не спрацює.

У найширшому сенсі, глибоко ешелонований захист використовує:

– Безпека даних: захищає дані в стані спокою та під час передачі, наприклад, шифрування, безпека бази даних, безпека повідомлень тощо.

– Безпека програм: безпека для локальних та віддалено розміщених програм (брандмауери веб-програм, безпека робочого навантаження, локальний контроль доступу тощо).

– Безпека кінцевих точок: захищає кінцеві точки за допомогою антивіруса, інструментів виявлення та реагування на кінцеві точки (EDR) тощо.

– Безпека мережі: контролює доступ між ресурсами та забезпечує безпечне спілкування за допомогою брандмауерів, систем контролю доступу до мережі (NAC), систем виявлення вторгнень тощо.

– Безпека доступу: Контролюйте доступ між активами всередині корпоративної мережі або навіть до хмарних активів за допомогою брандмауерів, віртуальних приватних мереж (VPN), захищених веб-шлюзів тощо.

– Автентифікація користувачів: перевірка користувачів, автентифікація

ідентифікаційних даних та авторизація рівнів доступу за допомогою паролів, біометрії, багатофакторної автентифікації, Active Directory тощо.

– Політики, процедури та навчання користувачів: навчати користувачів, IT-персонал та команди безпеки очікуванням, правилам поведінки, стандартам та методам підтримки безпеки та вирішення інцидентів.

Окрім цих рівнів, глибоко ешелонований захист вимагає перевірки всіх рівнів за допомогою:

– Проектування: створення надійної архітектури для операційної та безпекової стійкості

– Тестування: перевірка правильності роботи всіх компонентів за допомогою тестування на проникнення та сканування на вразливості

– Моніторинг: відстеження незвичайних, несанкціонованих та зловмисних дій за допомогою системи управління інцидентами та подіями безпеки (SIEM), центрів операцій безпеки (SOC) та центрів мережевих операцій (NOC).

– Технічне обслуговування: керування виправленнями, заміна застарілих технологій тощо.

– Навчання: працівників необхідно постійно навчати, щоб вони були в курсі найновіших інструментів, технологій, тактик та засобів контролю.

– Аудит та звітність: перевірка журналів, створення звітів для ключових показників ефективності (KPI) або відповідності вимогам

Принцип безпеки: доступ з найменшими привілеями

В ідеалі, доступ з найменшими привілеями вимагає, щоб користувачі мали доступ лише до даних, пристроїв і програм, необхідних для виконання їхньої роботи, саме тоді, коли ці ресурси потрібні, і з найнижчим рівнем привілеїв, який їм потрібен. Звичайно, такий динамічний і детальний контроль наразі технічно неможливий для більшості організацій.

Однак організації все ще повинні дотримуватися принципу найменшого рівня доступу через групи Active Directory, обмеження папок баз даних і серверів,

елементи керування пристроями, облікові дані доступу тощо. Багато організацій схильні надавати щедрі рівні доступу, щоб уникнути операційних затримок, але з точки зору безпеки слід застосовувати протилежний такт. Організація повинна надавати доступ лише тоді, коли і якщо це необхідно, і навіть тоді наданий доступ має бути мінімальним рівнем доступу, необхідним для виконання конкретного завдання.

Які основні загрози безпеці мережі?

Загрози безпеці мережі ставлять під загрозу операційну цілісність або цілісність безпеки мережі чи її даних. Загрози поділяються на такі основні категорії:

Погані користувачі

До зловмисних користувачів належать неправильно налаштовані програми, хакери та інші неавторизовані користувачі, які навмисно чи ненавмисно намагаються отримати доступ до мережевих ресурсів. Це може статися через викрадені облікові дані, звільнення співробітників з активними обліковими даними, віруси типу "командне керування" на кінцевих точках, викрадені ноутбуки, неправильно налаштовані API-з'єднання та багато іншого.

Погані пристрої

Погані пристрої – це неавторизовані пристрої або пристрої у скомпрометованому стані. Неавторизовані пристрої можуть бути більш нешкідливими, наприклад, співробітник, який підключає неавторизований персональний комп'ютер до мережі Ethernet, відділ маркетингу встановлює телевізор із підключенням до Wi-Fi у конференц-залі, або постачальник промислових насосних двигунів підключає свою останню поставку до стільникової мережі 5G для віддаленого моніторингу.

Набагато небезпечнішими пристроями були б робочі станції, на яких відсутні критичні оновлення безпеки для операційної системи, ноутбук, заражений вірусом, або ноутбук хакера, який підключається до мережі Wi-Fi з громадської парковки. Мережі повинні постійно відстежувати невідомі

з’єднання, незвичайну поведінку та скомпрометовані пристрої.

Шкідливі пристрої також можуть включати атаки, які намагаються викрасти або перенаправити мережевий трафік для підключення до шкідливих ресурсів. Наприклад, хакери можуть використовувати сніфери пакетів або фішингове посилання за допомогою атаки «людина посередині». Інші хакери можуть використовувати підроблену систему доменних імен (DNS) або IP-адреси для перенаправлення користувачів із легітимних підключень (до веб-сайтів, серверів тощо) до ресурсів кінцевих точок, контрольованих зловмисником, з метою викрадення інформації для входу або зараження кінцевої точки шкідливим програмним забезпеченням.

Зловмисний трафік

Мережі мають на меті з’єднати дійсних користувачів із дійсними активами. Однак, коли шкідливе програмне забезпечення або зловмисники намагаються проникнути в організацію, їхній трафік також проходить через мережу і іноді його можна виявити. Інший трафік сам по собі не є шкідливим, а навпаки, свідчить про зловмисні наміри, наприклад, підключення комп’ютера відвантаження складу до сервера відділу кадрів (HR) або спроба бездротової камери безпеки підключитися до хмарної бази даних обробки кредитних карток.

Злісні наміри

Внутрішні загрози виникають, коли легітимні користувачі підключаються до легітимних ресурсів для виконання несанкціонованих дій, використовуючи авторизовані дозволи. Наприклад, віце-президент із продажів має легітимний доступ до CRM відділу продажів та внутрішньої бази даних клієнтів, але завантаження повного списку клієнтів з кожного джерела буде ознакою того, що віце-президент готовий змінити роботу та викрасти інформацію.

Інші користувачі можуть намагатися перевищити межі свого доступу, наприклад, коли стажер-маркетолог намагається отримати доступ до файлового сервера досліджень і розробок та завантажити IP-адресу, що знаходиться в процесі розробки. Не всі внутрішні загрози обов'язково мають бути зловмисними;

деякі можуть бути просто недбалими або простими помилками, наприклад, коли IT-спеціалісти випадково перезаписують білий список веб-сайтів брандмауера та раптово блокують весь інтернет-трафік.

Погане обслуговування

Найкращі інструменти та архітектура безпеки будуть підірвані поганою практикою обслуговування. Організаціям, які відстають у виконанні оновлень програмного забезпечення або заміні застарілих компонентів, буде важко запобігти атакам.

Переривання операцій

Хоча більшість атак спрямовані на кінцеві точки (комп'ютери користувачів, сервери тощо), деякі атаки спрямовані на порушення роботи. Розподілені атаки типу "відмова в обслуговуванні" (DDoS) наразі частіше використовуються проти ресурсів, що піддаються доступу до Інтернету, але також можуть бути використані зловмисником для виведення з ладу мереж і внутрішніх серверів. Ненавмисна або зловмисна неправильна конфігурація сегментації або мережевого обладнання також може призвести до нестійких або непрацюючих мережевих з'єднань, нестабільних потоків трафіку та подібних перебоїв в роботі.

Пов'язані загрози кібербезпеці

Багато кіберзагроз опосередковано впливають на мережі через пов'язані з ними ресурси. Наприклад, шкідливе програмне забезпечення, шпигунське програмне забезпечення, рекламне ПЗ, комп'ютерні черв'яки, ботнети, троянські програми та подібне шкідливе програмне забезпечення зазвичай не впливають на мережеве обладнання (маршрутизатори, брандмауери тощо) або мережевий трафік. Якщо воно не впливає на мережу, воно не є мережевою загрозою. Натомість ці атаки завдають шкоди кінцевим точкам (комп'ютерам користувачів, серверам, контейнерам тощо) та загрожують програмам і даним, розміщеним на цих кінцевих точках.

Хоча служба мережевої безпеки повинна аналізувати мережевий трафік,

щоб виявляти індикатори таких атак і запобігати їм, коли це можливо, основна відповідальність за зупинення цих атак залежить від відповідних заходів безпеки для точки атаки: безпека електронної пошти (фішинг, шкідливі вкладення), безпека програм (міжсайтовий скриптинг, SQL-ін'єкції тощо), безпека кінцевих точок (антивірус, виявлення та реагування кінцевих точок тощо), безпека DNS (перенаправлення IP-адрес тощо) та безпека Active Directory (просування облікових даних, додавання користувачів тощо).

Найкращі практики захисту вашої мережі

Найкращі практики мережевої безпеки безпосередньо протидіють основним загрозам для мережі за допомогою спеціальних технологій та засобів контролю.

Доступ користувачів для контролю зловмисних користувачів

Для захисту від неавторизованих користувачів, мережева безпека повинна впроваджувати ефективний контроль доступу та ефективний контроль віддаленого доступу до мережі. Як мінімум, організація повинна використовувати Active Directory (або еквівалентний LDAP), двофакторну автентифікацію та VPN. Організації також повинні координувати функції IT та HR для одночасної активації та деактивації облікових даних співробітників та статусу зайнятості.

Організації можуть запровадити кращу безпеку шляхом:

– Покращене керування користувачами з детальнішим контролем доступу:

- Управління ідентифікацією та доступом.
- Управління привілейованим доступом.

– Багатофакторна автентифікація.

– Покращений віддалений доступ для контролю доступу до ресурсів, розміщених поза мережею, таких як OneDrive, SaaS-рішення тощо. Безпечні віддалені з’єднання також можуть запобігти певним типам атак перехоплення пакетів та перемаршрутизації пакетів, таким як «людина посередині» або

перехоплення пакетів:

- Хмарні VPN-сервіси.
- Безпечні веб-шлюзи.
- Оновлені локальні VPN-рішення (без використання RDP).
- Безпечні браузери.

Виявлення активів для контролю несправних пристроїв

Для захисту від несанкціонованих або скомпрометованих пристроїв організації повинні щонайменше проводити періодичну інвентаризацію пристроїв, підключених до мережі, встановлювати та підтримувати оновлене антивірусне програмне забезпечення для кінцевих точок, а також контролювати ключові активи (фінансові бази даних, файли відділу кадрів, ключову інтелектуальну власність тощо) на наявність несанкціонованого доступу.

Організації можуть запровадити кращу безпеку шляхом:

– Жорстке управління доступом до активів:

- Безперервне сканування на наявність несанкціонованих пристроїв запобігає несанкціонованому доступу за допомогою інструментів управління IT-активами (ITAM).
- Рішення для контролю доступу до мережі (NAC) тестують кінцеві точки на наявність застарілого або вразливого програмного забезпечення та перенаправляють пристрої на карантин до усунення недоліків.
- Керування мобільними пристроями (MDM) обмежує роботу програм на мобільних пристроях, а також може перевіряти стан пристрою на наявність джейлбрейка, застарілої ОС або шкідливого програмного забезпечення.

– Обмежте доступ до пристроїв і програм:

- Прив’язати пристрої до певних користувачів та певного доступу.
- Створюйте білі списки або списки дозволених ресурсів для певних ресурсів, яким дозволено здійснювати підключення, особливо для критично важливих ресурсів.

Набагато небезпечнішими пристроями були б робочі станції, на яких

відсутні критичні оновлення безпеки для операційної системи, ноутбук, заражений вірусом, або ноутбук хакера, який підключається до мережі Wi-Fi з громадської парковки. Мережі повинні постійно відстежувати невідомі з'єднання, незвичайну поведінку та скомпрометовані пристрої.

Шкідливі пристрої також можуть включати атаки, спрямовані на крадіжку або перенаправлення мережевого трафіку. Наприклад, хакери можуть використовувати сніфери пакетів або фішингове посилання за допомогою атаки «людина посередині» для крадіжки мережевого трафіку. Крім того, підроблені системи доменних імен (DNS) та IP-адреси можуть перенаправляти користувачів із законних з'єднань на небезпечні та шкідливі веб-сайти та кінцеві точки з метою крадіжки інформації або зараження кінцевих точок шкідливим програмним забезпеченням.

Аналіз та моніторинг пакетів для контролю поганого трафіку

Під час атаки мережевий трафік може містити відомі індикатори компрометації (IoC), відомі сигнатури шкідливих файлів або спрямовувати трафік на відомі шкідливі URL-адреси та IP-адреси. Брандмауери, системи виявлення вторгнень (IDS), системи запобігання вторгненням (IPS), системи виявлення та реагування мережі (NDR), розширене виявлення та реагування (XDR) та подібні системи можуть перевіряти пакети для виявлення, блокування або карантину шкідливого трафіку.

В інших випадках сам трафік може не розпізнаватися як шкідливий, але моніторинг за допомогою системи управління інцидентами та подіями безпеки (SIEM), центру операцій безпеки (SOC) або подібного рішення для моніторингу може виявити незвичайні з'єднання. Див. також розділ «Моніторинг активності для контролю злих намірів» нижче.

Моніторинг активності та сегментація для контролю поганих намірів

Зловмисні та випадкові внутрішні загрози можна виявити за допомогою таких інструментів, як запобігання втраті даних (DLP), аналітика сутності та поведінки користувачів (UEBA) або аналітика поведінки на основі штучного

інтелекту, вбудована в брандмауери та рішення IDS/IPS. Технологія обману з використанням принад та подібних хитрощів може викликати проактивні сповіщення як хакерів, так і авторизованих користувачів, які можуть намагатися виконати зловмисні дії.

Несподівані випадкові або зловмисні зміни в мережевих системах будуть виявлені швидше та ефективніше за допомогою контролю змін. Контроль змін відстежує зміни, пов'язані з користувачами та пристроями, щоб гарантувати, що всі зміни авторизовані, схвалені та зареєстровані.

Зі зростанням популярності програмно-визначених периметрів (SDP), програмно-визначених мереж широкого доступу (SD-WAN) та мережевого доступу з нульовою довірою (ZTNA) організації додатково контролюють доступ на детальному рівні, щоб запобігти внутрішнім загрозам. Віртуальні мережі можуть не тільки розширити сферу дії мережі, охопивши хмарні ресурси або ресурси в географічно розподілених місцях, але й створити сегментацію та мікросегментацію на основі груп користувачів (постачальник, консультант, відділ маркетингу тощо), рівнів доступу (базовий користувач, адміністратор тощо) або навіть певних користувачів чи програм. Якщо її належним чином визначити за допомогою інструментів керування ідентифікацією та доступом, мікросегментація може навіть обмежити можливість викрадених облікових даних завдати шкоди мережі або досягти несанкціонованих ресурсів.

Виправлення, оновлення та тестування контролюють погане обслуговування

Щоб запобігти проблемам, пов'язаним з обслуговуванням, організаціям слід регулярно впроваджувати оновлення, оновлення та тестування (сканування на вразливості, тести на проникнення тощо) своєї мережі. Сканування на вразливості та тести на проникнення також повинні виявляти неправильні конфігурації, незастосоване або неправильно застосоване шифрування, недбале управління ключами шифрування, слабкі паролі, відсутність авторизації та інші поширені проблеми.

Моніторинг та стійкість, контроль, перебої в роботі

Перебої в роботі можна виявляти, пом'якшувати або контролювати за допомогою моніторингу мережі за допомогою таких рішень, як центри мережевих операцій (NOC), центри операцій безпеки (SOC) та інструменти управління інцидентами та подіями безпеки (SIEM). Деяке програмне забезпечення для захисту від DDoS-атак та інші засоби забезпечення стійкості також будуть вбудовані в маршрутизатори, брандмауери тощо. Відстеження змін у контролі також може допомогти командам безпеки та операцій швидко аналізувати перебої, що виникають через змінені налаштування мережі.

Окрім програмного забезпечення, фізична безпека та стійка архітектура також відіграють важливу роль у запобіганні збоям у роботі мережі. Стійка архітектура з використанням балансування навантаження та резервування може поглинати початкові об'ємні DDoS-атаки, даючи реагуванню на інциденти можливість пом'якшити атаку без порушення роботи бізнесу.

Для хмарних мереж організації також можуть впроваджувати гіпермасштабовану архітектуру, яка поєднує рівні безпеки, сховища, обчислень та віртуалізації в модульний ресурс, який можна розгортати автоматично та масштабно. Однак гіпермасштабовану архітектуру також слід контролювати, щоб переконатися, що вона не використовує фінансово необґрунтовані ресурси.

3.2 Розробка структурної схеми

Найкращі практики безпеки інфраструктури корпоративних мереж

Впровадження найкращих практик безпеки інфраструктури корпоративних мереж може значно покращити здатність організації захищати свої системи та дані від кіберзагроз, забезпечуючи безперервність бізнес-операцій та дотримання стандартів безпеки.

Регулярно оновлюйте та виправляйте системи

Регулярне оновлення та встановлення виправлень систем має вирішальне

значення для підтримки безпеки інфраструктури корпоративних мереж, оскільки допомагає захиститися від відомих вразливостей та експлойтів. Текст уже належним чином відформатовано з тегами для абзаців та тегами для виділення жирним шрифтом. Подальше форматування не потрібне.

Впроваджуйте політики надійних паролів

Впровадження політик надійних паролів є важливим для забезпечення доступу до мережевих систем лише авторизованих користувачів, тим самим підвищуючи загальну безпеку.

Одним із ключових компонентів надійної політики щодо паролів є вимоги до складності. Завдяки поєднанню великих і малих літер, цифр і спеціальних символів паролі стає важче зламати. Забезпечення регулярних оновлень гарантує швидку заміну скомпрометованих паролів, що зменшує ризик несанкціонованого доступу.

Багатофакторна автентифікація додає додатковий рівень безпеки, вимагаючи від користувачів надання більш ніж однієї форми ідентифікації, що значно ускладнює доступ неавторизованих осіб до конфіденційної інформації.

Використовуйте брандмауери та системи виявлення вторгнень

Брандмауери та системи виявлення вторгнень (IDS) є ключовими компонентами безпеки інфраструктури корпоративних мереж. Вони допомагають контролювати, виявляти та блокувати несанкціонований доступ і кіберзагрози.

Брандмауери діють як бар'єр між довіреною внутрішньою мережею та недовіреними зовнішніми мережами, фільтруючи вхідний та вихідний трафік на основі попередньо визначених правил безпеки. Вони можуть бути апаратними, програмними або хмарними, забезпечуючи різні рівні безпеки.

З іншого боку, системи виявлення вторгнень доповнюють брандмауери, активно моніторячи мережевий трафік на предмет підозрілої активності та потенційних порушень безпеки. Вони аналізують пакети та шаблони, щоб виявляти аномалії, запускати сповіщення або вживати відповідних заходів.

Системи запобігання вторгненням (IPS) йдуть ще далі, не лише виявляючи

загрози, але й активно блокуючи або нейтралізуючи їх, щоб запобігти використанню мережі.

Проводьте регулярні аудити безпеки

Проведення регулярних аудитів безпеки допомагає організаціям виявляти вразливості, забезпечувати дотримання стандартів безпеки та посилювати заходи захисту.

Ці аудити включають ретельну оцінку протоколів безпеки, мереж і систем для виявлення слабких місць, які можуть бути використані кіберзагрозами. Регулярно проводячи ці аудити, організації можуть випереджати нові кіберризики та захищати свої конфіденційні дані від потенційних порушень.

Дотримання таких нормативних актів, як GDPR та PCI DSS, має вирішальне значення для організацій, які обробляють дані клієнтів. Регулярні аудити безпеки допомагають забезпечити дотримання компаніями цих нормативних вимог, уникаючи значних штрафів та шкоди для репутації, які можуть виникнути внаслідок недотримання вимог.

Такі компанії, як PixelPlex, маючи спеціалізований досвід у проведенні аудитів безпеки, можуть надавати комплексні оцінки та рекомендації, адаптовані до унікальних потреб кожної організації. Співпрацюючи з такими фірмами, компанії можуть скористатися першокласними оцінками безпеки та зміцнити свій загальний рівень безпеки.

Стратегії безпеки інфраструктури корпоративних мереж

Ефективні стратегії безпеки інфраструктури корпоративних мереж мають вирішальне значення для захисту систем і даних організації, забезпечення безперервності бізнесу та мінімізації ризику кіберзагроз.

Сегментація мережі

Сегментація мережі передбачає поділ мережі на менші сегменти для підвищення безпеки та ефективнішого управління контролем доступу.

Впроваджуючи сегментацію мережі, організації можуть значно зменшити ризик поширення кібератак по всій своїй мережі. Основна перевага полягає в

ізоляції конфіденційних даних і критично важливих систем, що ускладнює для хакерів подальше проникнення після порушення певного сегмента.

Такі організації, як IBM, пропонують передові рішення для сегментації мережі, які дозволяють детально контролювати мережевий трафік, гарантуючи, що лише авторизовані користувачі та пристрої матимуть доступ до певних сегментів. Такий підхід підвищує безпеку та покращує продуктивність мережі, зменшуючи непотрібні перевантаження трафіку та оптимізуючи використання ресурсів.

Шифрування

Шифрування – це життєво важливий захід безпеки, який захищає конфіденційні дані, перетворюючи їх у нечитабельний формат, забезпечуючи конфіденційність та цілісність.

Існують різні методи шифрування, що використовуються для захисту даних, включаючи симетричне шифрування та асиметричне шифрування. Симетричне шифрування використовує один і той самий ключ як для шифрування, так і для дешифрування, тоді як асиметричне шифрування використовує пару ключів – відкритий та закритий.

Такі компанії, як IBM, пропонують передові рішення для шифрування, які допомагають захистити інформацію в мережах і на пристроях, як-от IBM Cloud Hyper Protect Crypto Services. Шифрування відіграє вирішальну роль у захисті фінансових транзакцій, особистої інформації та конфіденційних повідомлень у сучасну цифрову епоху.

Контроль доступу

Контроль доступу є фундаментальним аспектом безпеки інфраструктури корпоративних мереж. Він гарантує, що лише авторизовані користувачі мають доступ до критично важливих систем і даних.

Одним із поширених методів контролю доступу є контроль доступу на основі ролей (RBAC), де дозволи призначаються на основі ролей, які виконують окремі особи в організації. Це гарантує, що користувачі мають доступ лише до

ресурсів, необхідних для виконання їхніх робочих завдань, мінімізуючи ризик несанкціонованого доступу.

Ще одним важливим підходом є багатофакторна автентифікація (MFA), яка додає додатковий рівень безпеки, вимагаючи від користувачів надання кількох форм ідентифікації перед отриманням доступу. Ці методи допомагають організаціям дотримуватися галузевих стандартів, встановлених такими організаціями, як Агентство національної безпеки (АНБ), забезпечуючи надійні заходи безпеки.

Планування відновлення після аварій та безперервності бізнесу

Планування відновлення після аварій та забезпечення безперервності бізнесу є важливими стратегіями, що забезпечують швидке відновлення організації після кіберінцидентів та підтримку її діяльності.

Ці плани зазвичай включають різні компоненти, такі як процедури резервного копіювання та відновлення даних, альтернативні методи зв'язку, визначення критичних бізнес-функцій та процедури регулярного тестування та оновлення.

Маючи надійний план аварійного відновлення та забезпечення безперервності бізнесу, організації можуть мінімізувати вплив збоїв на свою діяльність, зменшуючи час простою та втрату даних.

Такі компанії, як TierPoint, спеціалізуються на наданні комплексних послуг з аварійного відновлення, щоб допомогти організаціям розробляти та впроваджувати ефективні плани.

Поширені загрози безпеці інфраструктури корпоративних мереж

Розуміння поширених загроз безпеці інфраструктури корпоративних мереж має вирішальне значення для організацій, щоб розробити ефективні заходи захисту та зменшити ризики, що виникають внаслідок кібератак.

Шкідливе програмне забезпечення та віруси

Шкідливе програмне забезпечення та віруси – це шкідливе програмне забезпечення, призначене для проникнення та пошкодження мережевих систем,

що створює значні загрози безпеці організацій.

Ці загрози можуть призвести до витоків даних, фінансових втрат та операційних збоїв, що робить критично важливим для таких компаній, як IBM, пропонування надійних рішень для захисту від шкідливого програмного забезпечення. Без належних заходів захисту можна використовувати вразливості мережі, що призводить до компрометації конфіденційної інформації та втрати довіри користувачів.

Впроваджуючи надійні протоколи кібербезпеки, такі як регулярні оновлення програмного забезпечення, брандмауери та системи виявлення вторгнень, організації можуть зменшити ризики, пов'язані зі шкідливим програмним забезпеченням та вірусами, захищаючи свої мережі та конфіденційні дані від потенційної шкоди.

Фішинг та соціальна інженерія

Фішингові атаки та атаки соціальної інженерії використовують людські вразливості для отримання несанкціонованого доступу до конфіденційної інформації та систем.

Ці зловмисні тактики часто включають замасковані електронні листи, повідомлення або веб-сайти, які обманом змушують людей розголошувати особисті дані, такі як облікові дані для входу, банківські реквізити чи іншу конфіденційну інформацію. Користуючись довірою та цікавістю, кіберзлочинці маніпулюють людьми, змушуючи їх вживати дій, що ставить під загрозу кіберзахист.

– Одна з поширених технік фішингу полягає у створенні фальшивих сторінок входу, що відображають легітимні сайти, з метою обману користувачів та вимагання від них ненавмисного введення облікових даних.

– Такі організації, як Агентство національної безпеки (АНБ), наголошують на необхідності суворих навчальних програм для навчання співробітників виявляти потенційні загрози та реагувати на них з метою підвищення обізнаності.

Внутрішні загрози

Внутрішні загрози включають зловмисні або недбалі дії осіб в організації, які мають доступ до конфіденційних систем і даних.

Ці загрози становлять значний ризик для безпеки мережі, оскільки інсайдери можуть навмисно розголошувати конфіденційну інформацію або несвідомо ставати жертвами фішингових атак, ненавмисно ставлячи під загрозу цілісність даних та репутацію організації.

Контроль доступу має вирішальне значення для запобігання внутрішнім загрозам, гарантуючи, що доступ до конфіденційної інформації матиме лише уповноважений персонал.

Постійний моніторинг активності користувачів та моделей поведінки може допомогти виявити будь-які підозрілі дії та зменшити потенційні ризики до їх загострення. Впровадження надійних протоколів безпеки, проведення регулярних тренінгів з підвищення обізнаності та сприяння культурі безпеки можуть сприяти підвищенню безпеки мережевого середовища.

Як реагувати на порушення безпеки інфраструктури корпоративних мереж

Ефективне реагування на порушення безпеки інфраструктури корпоративних мереж включає ізоляцію уражених систем, проведення ретельного розслідування та впровадження заходів щодо усунення наслідків для пом'якшення наслідків та запобігання майбутнім порушенням.

Ізолюйте уражені системи

Першим кроком у реагуванні на порушення безпеки є ізоляція уражених систем, щоб запобігти подальшому поширенню загрози та захистити інші компоненти мережі.

Швидка ізоляція скомпрометованих систем має вирішальне значення, оскільки вона допомагає обмежити горизонтальне переміщення зловмисника в мережі, запобігаючи поширенню шкідливого програмного забезпечення або несанкціонованому доступу:

1. Ефективної ізоляції можна досягти за допомогою таких інструментів, як брандмауери.

2. Сегментація мережі.

3. Системи виявлення вторгнень для створення бар'єрів навколо скомпрометованих систем.

Посібники реагування на інциденти та платформи реагування на інциденти можуть допомогти командам безпеки ефективно ізолювати уражені системи та виявити будь-які потенційні підключення до інших частин мережі.

Розслідуйте та визначте джерело порушення

Розслідування та визначення джерела порушення має вирішальне значення для розуміння того, як сталася атака, та запобігання подібним інцидентам у майбутньому.

Після виявлення порушення безпеки процес розслідування активізується. Перший крок включає збір доказів з різних джерел, таких як системні журнали, дані мережевого трафіку та потенційно скомпрометовані пристрої. Саме тут команда контролю якості відіграє значну роль, ретельно перевіряючи конфігурації системи та налаштування програм на наявність вразливостей.

Наступний етап полягає в аналізі зібраних журналів та даних для відстеження послідовності подій, що призвели до порушення. Вивчаючи схеми атак та точки входу, слідчі можуть точно визначити вектор атаки, який використовували зловмисники.

Пом'якшити та усунути порушення

Пом'якшення та усунення наслідків порушення включає вжиття заходів для усунення загрози, відновлення уражених систем та посилення заходів безпеки для запобігання майбутнім інцидентам.

Витоки даних можуть завдати шкоди бізнесу та організаціям, спричиняючи фінансові втрати та шкодячи репутації. Для ефективного усунення порушень безпеки першим важливим кроком є оперативне видалення будь-якого шкідливого програмного забезпечення, яке могло проникнути в системи.

Це вимагає ретельного сканування та очищення, щоб гарантувати усунення всіх слідів порушення. Щоб запобігти подібним інцидентам у майбутньому, важливо виправити будь-які вразливості, якими скористалися зловмисники. IBM відіграє значну роль у наданні передових рішень для відновлення, які допомагають організаціям ефективно відновлюватися після порушень безпеки та зміцнювати свій захист від кіберзагроз.

Вживайте запобіжних заходів для майбутніх порушень

Впровадження превентивних заходів є важливим для зменшення ризику майбутніх порушень та підвищення загальної безпеки мережі.

Одним із вирішальних кроків у запобіганні порушенням є проведення регулярних аудитів безпеки для виявлення вразливостей та слабких місць у мережі. Це дозволяє організаціям проактивно усувати будь-які потенційні слабкі місця, перш ніж ними зможуть скористатися кіберзлочинці.

Надання комплексного навчання співробітникам найкращим практикам кібербезпеки є надзвичайно важливим. Навчання персоналу тому, як розпізнавати спроби фішингу, уникати завантаження шкідливого програмного забезпечення та захищати конфіденційну інформацію, може значно зменшити ймовірність успішних порушень.

Також вкрай важливо регулярно оновлювати протоколи безпеки на основі даних з авторитетних джерел, таких як Gartner. Gartner надає цінну галузеву аналітику та рекомендації, які можуть допомогти організаціям випереджати нові загрози та зміцнювати свої захисні механізми від потенційних кібератак.

Структурна схема розробленої системи показана на рисунку 3.1. Для реалізації корпоративної мережі було вибрано технологію приватної мережі на орендованих каналах.

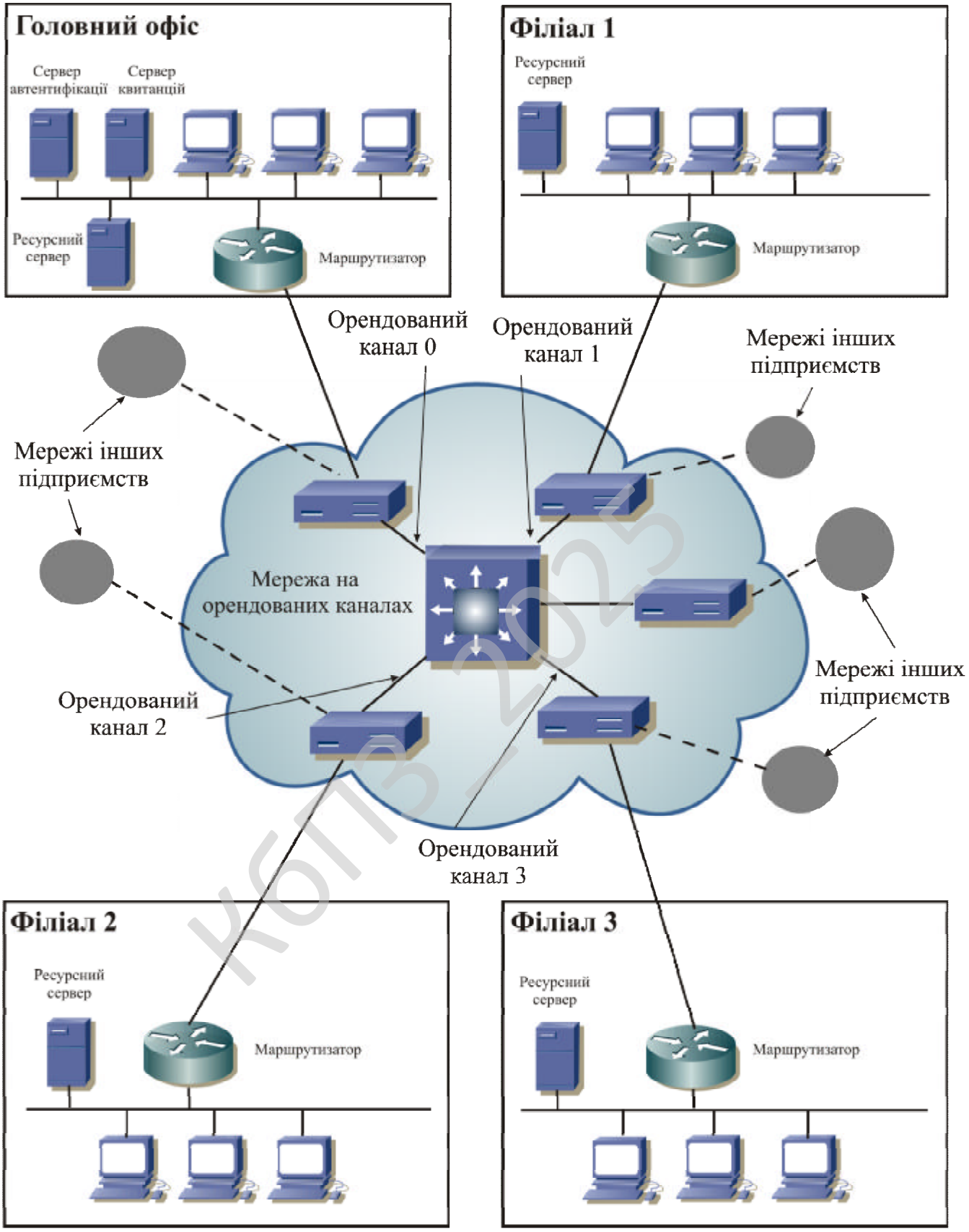

Рисунок 3.1 – Структурна схема розробленої системи

Мережа складається з головного офісу та декількох філіалів. У приміщенні головного офісу знаходяться наступні сервери:

1. Сервер автентифікації.
2. Сервер квитанцій (білетів).
3. Ресурсний сервер.

У приміщеннях філіалів знаходяться:

1. Ресурсні сервери.
2. Робочі станції (хости).

Орендовані територіальні канали прокладаються провайдером транспортних територіальних послуг у його первинній мережі FDM, PDH, SDH або мережі з інтегральними послугами ISDN. При оренді каналу в таких мережах підприємство ділить пропускну здатність магістральних каналів і комутаторів цієї мережі з іншими абонентами даного провайдеру.

На рисунку 3.1 показаний приклад використання орендованих каналів для побудови корпоративної мережі підприємства із трьома філіями.

3.3 Розробка функціональної схеми

Функциональна схема розробленої системи показана на рисунку 3.2.

Розподіл ключів та отримання за допомогою них доступу до ресурсів корпоративної мережі проходить у декілька етапів:

1. Первинна автентифікація.
2. Одержання дозволу на доступ до ресурсного сервера.
3. Одержання доступу до ресурсу.

Первинна автентифікація

Процес доступу користувача до ресурсів включає дві процедури: по-перше, користувач повинен довести свою легальність (автентифікація), а по-друге, він повинен одержати дозвіл на виконання певних операцій з певним ресурсом (авторизація). У розробленій системі користувач один раз автентифікується під час

логічного входу в мережу, а потім проходить процедури автентифікації й авторизації щоразу, коли йому потрібен доступ до нового ресурсного сервера.

Виконуючи логічний вхід у мережу, користувач, а точніше клієнтська програма, установлена на його комп'ютері, посилає автентифікаційному серверу AS ідентифікатор користувача ID (рисунок 3.2).

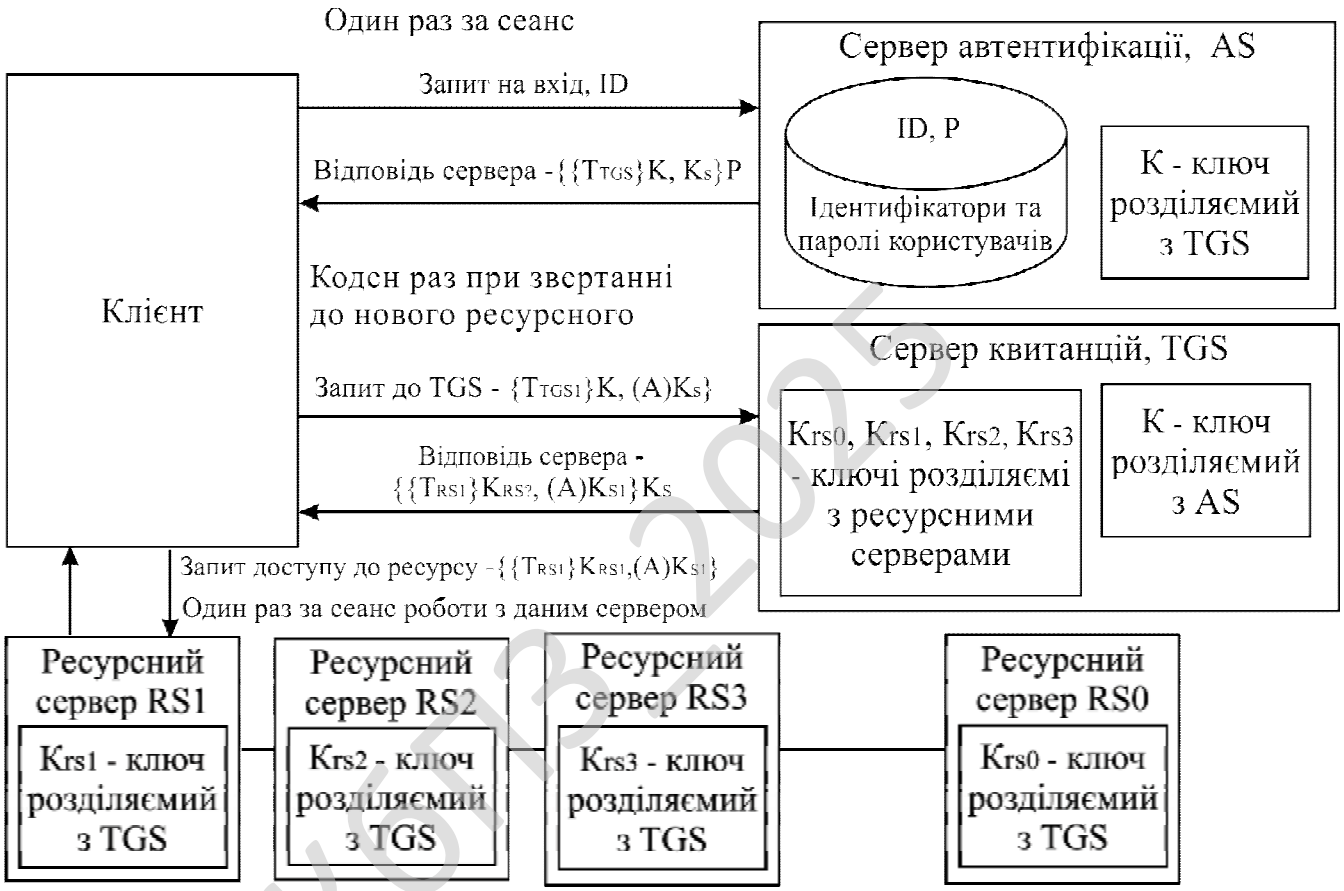

Рисунок 3.2 – Функціональна схема розробленої системи

Спочатку автентифікаційний сервер перевіряє в базі даних, чи є запис про користувача з таким ідентифікатором, потім, якщо такий запис існує, витягає з неї пароль користувача p. Даний пароль буде потрібний для шифрування всієї інформації, що направить автентифікаційний сервер клієнтові як відповідь. А відповідь складається із квитанції T_{TGS} на доступ до сервера квитанцій Kerberos і ключа сеансу K_s. Під сеансом тут розуміється увесь час роботи користувача, від

моменту логічного входу в мережу до моменту логічного виходу. Ключ сеансу буде потрібний для шифрування в процедурах автентифікації протягом усього користувальницького сеансу. Квитанція шифрується за допомогою секретного DES-ключа K, який розділяють автентифікаційний сервер і сервер квитанцій. Все разом – зашифрована квитанція й ключ сеансу – ще раз шифруються за допомогою користувальницького пароля p. Таким чином, квитанція шифрується двічі ключем K и паролем p. У наведених вище позначеннях повідомлення-відповідь, що автентифікаційний сервер посилає клієнтові, виглядає так: $\{\{T_{TGS}\}K, K_s\}p$.

Після того як таке відповідне повідомлення надходить на клієнтську машину, клієнтська програма просить користувача ввести свій пароль. Коли користувач вводить пароль, то клієнтська програма намагається за допомогою пароля розшифрувати повідомлення, що надійшло. Якщо пароль вірний, то з повідомлення витягається квитанція на доступ до сервера квитанцій $\{T_{TG}s\}K$ (у зашифрованому виді) і ключ сеансу K_s (у відкритому виді). Успішна розшифровка повідомлення означає успішну автентифікацію. Помітимо, що автентифікаційний сервер AS автентифікує користувача без передачі пароля по мережі.

Квитанція T_{TGS} на доступ до сервера квитанцій TGS є посвідченням легальності користувача й дозволом йому продовжувати процес одержання доступу до ресурсу. Ця квитанція містить:

– ідентифікатор користувача;

– ідентифікатор сервера квитанцій, на доступ до якого отримана квитанція;

– оцінку про поточний час;

– період часу, протягом якого може тривати сеанс;

– копію ключа сесії KS.

Як уже було сказано, клієнт має квитанцію в зашифрованому виді. Шифрування підвищує впевненість у тому, що ніхто, навіть сам клієнт – власник даної квитанції, – не зможе квитанцію підробити, підмінити або змінити. Тільки

сервер TGS, одержавши від клієнта квитанцію, зможе її розшифрувати, тому що в його розпорядженні є ключ шифрування К.

Час дії квитанції обмежений тривалістю сеансу. Дозволена тривалість сеансу користувача, що втримується у квитанції на доступ до сервера квитанцій, задається адміністратором і може змінюватися залежно від вимог до захищеності мережі. У мережах із твердими вимогами до безпеки час сеансу може бути обмежено 30 хвилинами, в інших умовах цей час може скласти 8 годин. Інформація, що утримується у квитанції, визначає її строк придатності. Надання квитанції на цілком певний час захищає її від неавторизованого користувача, що міг би її перехопити й використовувати в майбутньому.

Одержання дозволу на доступ до ресурсного сервера

Наступним етапом для користувача є одержання дозволу на доступ до ресурсного сервера (наприклад, до файлового сервера або сервера додатків). Але для цього треба звернутися до сервера TGS, що видає такі дозволи (квитанції). Щоб одержати доступ до сервера квитанцій, користувач уже отримав квитанцією $\{T_{TGS}\}K$, видану йому сервером AS. Незважаючи на захист паролем і шифрування, користувачу, для того щоб довести серверу квитанцій, що він має право на доступ до ресурсів мережі, потрібно ще дещо, крім квитанції.

Як уже згадувалося, перше повідомлення від автентифікаційного сервера містило не тільки квитанцію, але й секретний ключ сеансу K_s, що розділяється із сервером квитанцій TGS. Клієнт використовує цей ключ для шифрування ще однієї електронної форми, що називається автентифікатором $\{A\}K\$$. Автентифікатор А містить ідентифікатор і мережну адреса користувача, а також власну часову відмітку. На відміну від квитанції $\{T_{TGS}\}K$, що протягом сеансу використовується багаторазово, автентифікатор призначений для одноразового використання й має дуже короткий час життя – звичайно кілька хвилин. Клієнт посилає серверу квитанцій TGS повідомлення-запит, що містить квитанцію й автентифікатор: $\{T_{TGS}\}K$, $\{A\}K_S$.

Сервер квитанцій розшифровує квитанцію наявним у нього ключем К,

перевіряє термін дії квитанції, і витягає з неї ідентифікатор користувача.

Потім сервер TGS розшифровує автентифікатор, використовуючи ключ сеансу користувача K_s, який він витягає із квитанції. Сервер квитанцій порівнює ідентифікатор користувача і його мережну адресу з аналогічними параметрами у квитанції й повідомленні. Якщо вони збігаються, то сервер квитанцій одержує впевненість, що дана квитанція дійсно представлена її законним власником.

Помітимо, що звичайне володіння квитанцією на одержання доступу до сервера квитанцій не доводить ідентичність користувача. Тому що автентифікатор дійсний тільки протягом короткого проміжку часу, то малоймовірно вкрасти одночасно й квитанцію, і автентифікатор і використовувати їх протягом цього часу. Щоразу, коли користувач звертається до сервера квитанцій для одержання нової квитанції на доступ до ресурсу, він посилає багаторазово використовувану квитанцію й новий автентифікатор.

Клієнт звертається до сервера квитанцій за дозволом на доступ до ресурсного сервера, що тут позначений як RS1. Сервер квитанцій, упевнившись у легальності запиту й особистості користувача, відсилає йому відповідь, що містить дві електронні форми: багаторазово використовувану квитанцію на одержання доступу до запитуваного ресурсного сервера T_{RS1} і новий ключ сеансу K_{S1}.

Квитанція на одержання доступу шифрується секретним ключем K_{RS1} поділюваним тільки сервером квитанцій і тим сервером, до якого надається доступ, у цьому випадку – RS1. Сервер квитанцій розділяє унікальні секретні ключі з кожним сервером мережі. Ці ключі розподіляються між серверами мережі фізичним способом або яким-небудь іншим секретним способом при установці системи. Коли сервер квитанцій передає квитанцію на доступ до будь-якого ресурсного сервера, то він шифрує її, так що тільки цей сервер зможе розшифрувати її за допомогою свого унікального ключа.

Новий ключ сеансу K_{S1} утримується не тільки в самому повідомленні, що посилається клієнтові, але й усередині квитанції T_{RS1}. Все повідомлення шифрується старим ключем сеансу клієнта K_s, так що його може прочитати тільки

цей клієнт. Використовуючи введені позначення, відповідь сервера TGS клієнтові можна представити в наступному виді: $\{\{T_{RS1}\}K_{RS1}, K_{S1}\}K_S$.

Одержання доступу до ресурсу

Коли клієнт розшифровує повідомлення, що надійшло, то він відсилає серверу, до якого він хоче одержати доступ, запит, що містить квитанцію на одержання доступу й автентифікатор, зашифрований новим ключем сеансу: $\{TR_{si}\}KR_{si}$, {A}K$1.

Це повідомлення обробляється аналогічно тому, як оброблявся запит клієнта сервером TGS. Спочатку розшифровується квитанція ключем K_{RS1}, потім витягається ключ сеансу K_{si} і розшифровується автентифікатор. Далі порівнюються дані про користувача, що утримуються у квитанції й автентифікаторі. Якщо перевірка проходить успішно, то доступ до мережного ресурсу дозволений.

На цьому етапі клієнт також може захотіти перевірити автентичність сервера перед тим, як почати з ним працювати. Взаємна процедура автентифікації запобігає будь-якій можливості спроби одержання неавторизованим користувачем доступу до секретної інформації від клієнта шляхом підміни сервера.

Автентифікація ресурсного сервера в розробленій системі виконується у відповідності з наступною процедурою. Клієнт звертається до сервера із пропозицією, щоб той надіслав йому повідомлення, у якому повторив часову мітку з автентифікатора клієнта, збільшену на 1. Крім того, потрібно, щоб дане повідомлення було зашифровано ключем сеансу K_{si}. Щоб виконати такий запит клієнта, сервер витягає копію ключа сеансу із квитанції на доступ, використовує цей ключ для розшифровки автентифікатора, нарощує значення часової мітки на 1, заново зашифровує повідомлення, використовуючи ключ сеансу, і повертає повідомлення клієнтові. Клієнт розшифровує це повідомлення, щоб одержати збільшену на одиницю оцінку часу.

При успішному завершенні описаного процесу клієнт і сервер впевнюються у таємності своїх транзакцій. Крім цього, вони одержують ключ сеансу, який можуть використовувати для шифрування майбутніх повідомлень.

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.

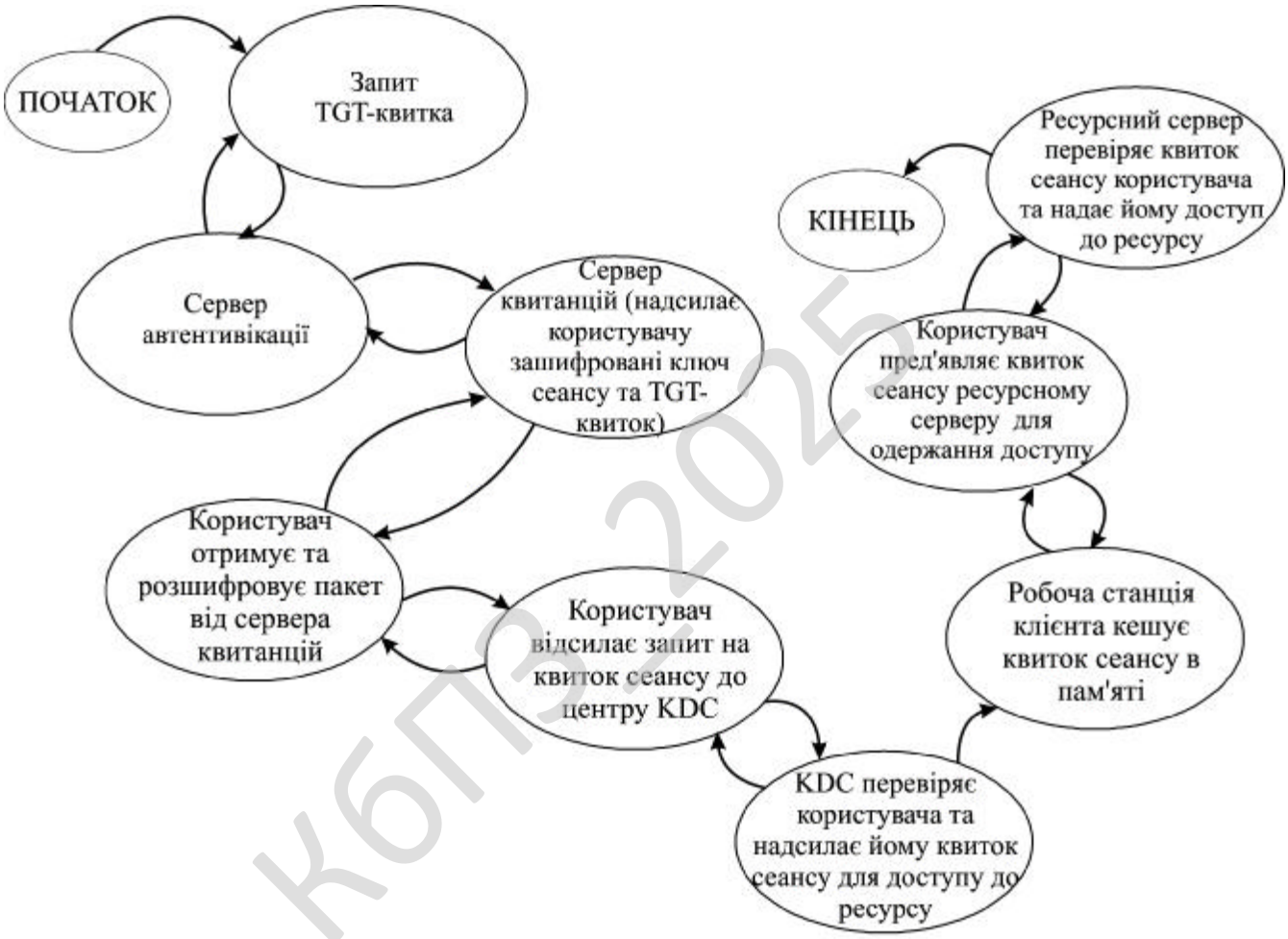

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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 якої 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.

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

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.

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.

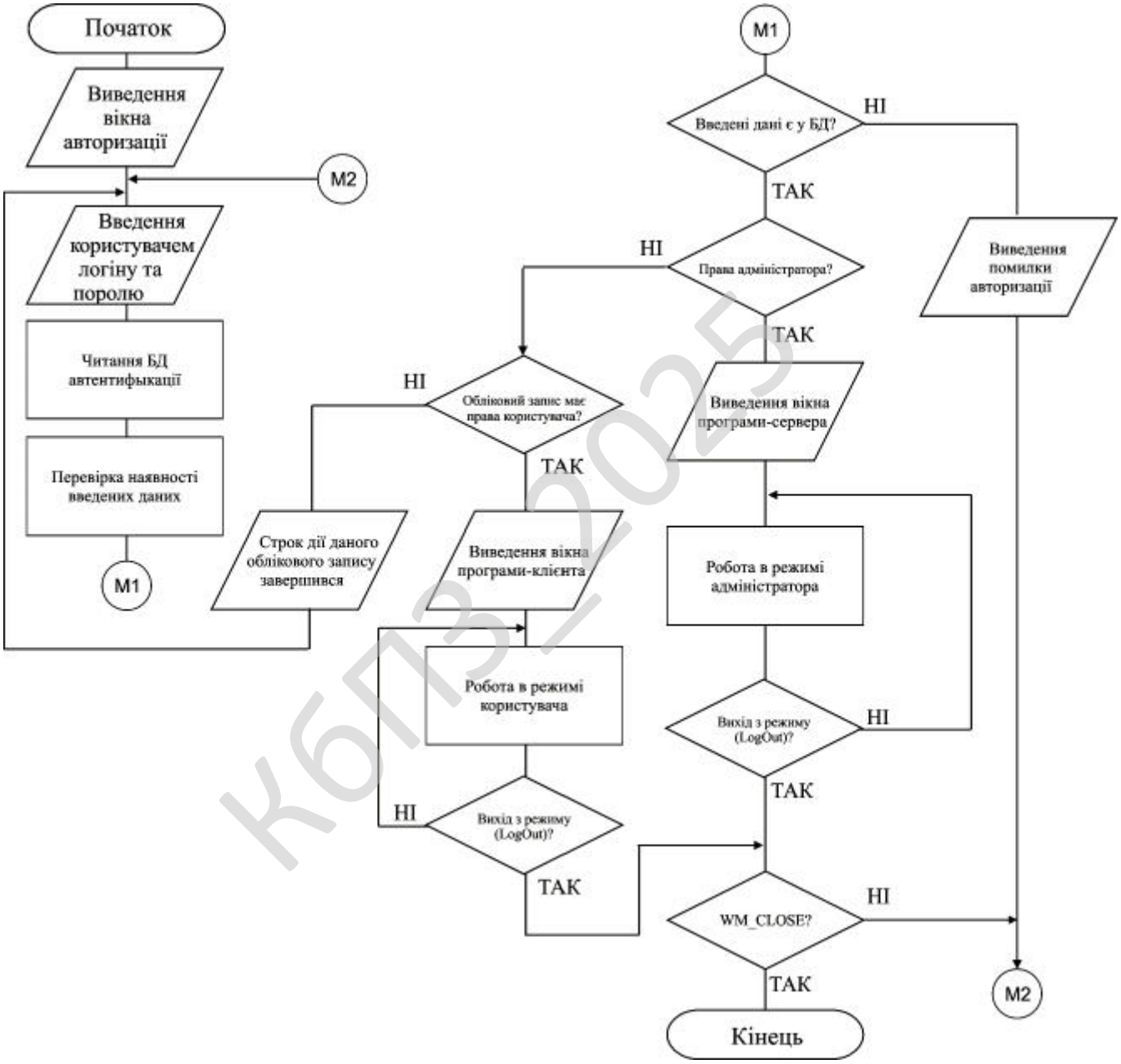

Рисунок 4.1 – Блок-схема основної програми

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.

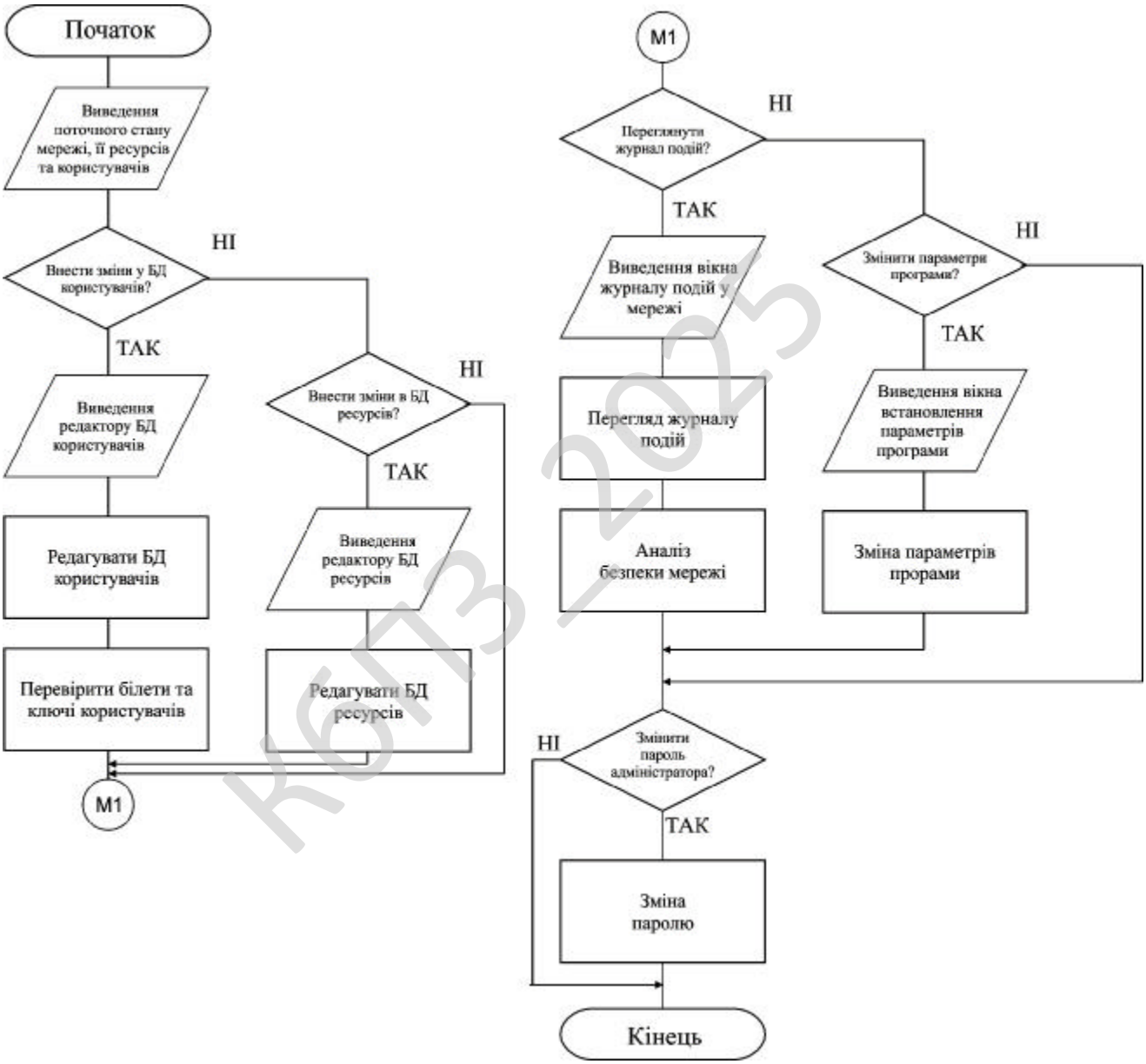

Рисунок 4.2 – Блок-схема роботи підпрограми

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.

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 якдіаграма 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.

Розроблена система забезпечує механізм взаємної автентифікації між

клієнтом і сервером у мережі, у якій пакети, що передаються, можуть бути переглянуті або модифіковані.

Щоб належним чином забезпечити взаємну автентифікацію використовуються симетричні ключі, шифровані об'єкти й служби. Цей підрозділ містить наступні розділи:

– Використання паролів та ключів при автентифікації.

– TGT-квитки.

– Сеансові ключі.

– Сеансові квитки.

– Центр розподілу ключів.

– Область Kerberos.

Концепція поділюваного секрету проста: якщо секрет відомий тільки двом людям, тоді кожна людина зможе перевірити дійсність іншого, упевнившись, що інша людина знає секрет.

Розроблений протокол вирішує проблему з безпечним ключем криптографічно. Замість доступного пароля партнери по з'єднанню створюють криптографічний ключ і використовують знання про цей ключ, щоб перевірити дійсність один одного. Додатково, щоб метод працював, відкриті ключі повинні бути симетричними, тобто один ключ повинен використовуватися й для шифрування й для дешифрування. Одна сторона підтверджує знання про ключ шляхом шифрування частини інформації, інша засвідчує знання про ключ дешифруванням інформації.

Використовується декілька ключів і типів ключів для шифрування. Типи ключів можуть включати довгострокові симетричні ключі, довгострокові асиметричні ключі й короткострокові симетричні ключі. Протокол автентифікації призначений для використання симетричного шифрування, це означає, що один відкритий ключ використовується відправником і одержувачем для шифрування й дешифрування.

Довгостроковий симетричний ключ: користувач, система, служба, і

міжобласні ключі. Довгостроковий симетричний ключ створюється з пароля. Відкритий текст пароля трансформується в криптографічний ключ передачею тексту пароля через криптографічну функцію (всі реалізації протоколу автентифікації Kerberos повинні підтримувати DES-CBC-MD5, інші алгоритми допускаються). Результат криптографічної функції – це ключ.

Користувальницькі ключі. Коли створюється користувач, пароль використовується для створення ключа користувача. У доменах Active Directory ключ користувача зберігається в об'єкті користувача в Active Directory. На робочій станції користувальницький ключ створюється коли користувач входить в ОС.

Системні ключі. Коли робоча станція або сервер входять у домен Windows, вони одержують пароль. Таким же способом, як і у випадку користувальницького акаунта пароль системного акаунта використовується для створення системного ключа.

Ключі служб. Служби використовують ключ, заснований на паролі акаунта, коли вони запускаються. Всі KDC в одній області використовують однаковий ключ служби. Ці ключі засновані на паролі, призначеному акаунту krbtgt. Кожний домен Active Directory має цей вбудований акаунт.

Міжобласні ключі. Додатково до крос-обласної автентифікації KDC повинні створюватися внутрішньообласні ключі. Області повинні довіряти один одному тому що вони мають загальний ключ. Домени Active Directory, що мають батьківські відносини розділяють внутрішньообласний ключ. Цей внутрішньообласний ключ, заснований на відносинах “transitive trusts” в Windows 2000 і Windows Server 2003. Якщо відносини довіри створені, два домени можуть обмінюватися ключами специфічними для їхньої довіри.

Довжини ключів шифрування. Програма підтримує різні типи шифрування й довжини ключів залежно від виконуваного завдання й певних опцій. Незважаючи на те що розмір ключа визначає рівень захисту забезпечуваний ключем, розмір ключа дещо впливає на розмір квитка.

Наступна таблиця перераховує довжини ключів, що підтримують різні типи шифрування.

Таблиця 4.1 – Довжини ключів для різних типів шифрування, що підтримуються розробленою програмою

Алгоритм шифрування	Довжина ключа
RC4-HMAC	128
DES-CBC-CRC	56
DES-CBC-MD5	56

Довгострокові асиметричні ключі – відкритий ключ. Сертифікати з відкритим ключем, що зберегаються на смарт-картах, мають тільки довгострокові асиметричні ключі реалізації Microsoft для автентифікації Kerberos.

Короткострокові симетричні ключі – ключі сесії. Сесійні ключі, що використовуються для квитка на одержання TGT-квитків і квитків сесії на одержання сервісів тимчасово використовуються настільки довго, наскільки сесія або служба працездатні.

Квитки. Основний компонент автентифікації типу Kerberos – це квиток. Повідомлення Kerberos використовуються для запиту або випуску квитків. Є два типи квитків – (TGT) і квитки сесії на одержання сервісу (TGS). Запити квитків. Клієнти посилають запити квитків до Центру сертифікатів.

Таблиця 4.2 – Типи запитів квитків

Тип запитаного квитка	Служба KDC, що надсилає запит
TGT-квиток	Служба автентифікації
Квиток сесії	Служба видачі квитків

Запит квитка включає:

– Властивості запиту (прапори), такі як період перевідновлення квитка.

– Метод шифрування запиту.

Квиток на одержання квитків. KDC відповідає запиту служби автентифікації клієнта поверненням квитка для себе. Це особливий квиток служби, що називається квитком на одержання квитків (TGT). TGT дозволяє службі автентифікації безпечно передавати посвідчення користувача службі видачі квитків.

TGT це:

– Початковий квиток користувача служби автентифікації.

– Використовується, щоб запросити квиток на отримання сервісу.

– Означає тільки використання службою видачі квитків.

TGT шифрується ключем спільним з KDC. Клієнт не може прочитати квиток TGT. Тільки сервер KDC може прочитати TGT для безпечного доступу до користувальницького посвідчення, сесійним ключам та іншій інформації. Подібно звичайному квитку служби TGT містить копію сесійного ключа, що служба (у даному випадку KDC) використовує для зв'язку із клієнтом. TGT зашифрований довгостроковим ключем KDC.

З погляду клієнта TGT це інший квиток. Перед тим як він намагається приєднатися до будь-якого сервісу, клієнт спочатку перевіряє кеш посвідчень на наявність квитка на одержання сервису (TGS) для цього сервісу. Якщо його немає клієнт перевіряє кеш знову на наявність TGT. Якщо він знаходить TGT то дістає відповідний TGS сесійний ключ із кешу, і використовує цей ключ для підготовки автентифікатору та посилає автентифікатор і TGT в KDC.

З боку KDC TGT дозволяє уникнути виконання зайвих дій по пошуку користувальницького довгострокового ключа щоразу, коли користувач запитує сервіс. KDC дивиться користувальницький довгостроковий ключ один раз, коли він надає початковий TGT.

Для всіх обмінів із цим клієнтом KDC може дешифрувати TGT його власним довгостроковим ключем, витягаючи сесійний ключ і використовувати його для перевірки автентифікатору клієнта.

Квитки на одержання сервісу. Квиток на одержання сервісу TGS дозволяє безпечно передавати посвідчення особи запитуючого до цільового на серверу або службі. KDC відповідає на запит клієнта про доступ до служби, висилаючи 2 копії сесійного ключа клієнтові. Клієнтська копія сесійного ключа зашифрована ключем, що KDC розділяє із клієнтом.

Копія сесійного ключа для одержання сервісу вкладена разом з інформацією про клієнта в структуру даних, що називається квитком на одержання сервісу. Вся структура зашифрована ключем, що KDC розділяє із сервером, який надає сервіс. Обов'язок клієнта тепер управляти контактом із сервером, використовуючи квиток із сесійним ключем на доступ до сервісу. Квиток на одержання сервісу використовується для автентифікації службою й називається цільовою службою.

Квиток на одержання сервісу шифрується сесійним ключем, що є довгостроковим ключем поділюваним KDC і цільовою службою. Таким чином, незважаючи на те, що клієнт управляє квитком на одержання сервісу клієнт не може прочитати його. Тільки KDC і цільова служба можуть читати квиток, одержуючи безпечний доступ до користувальницького посвідчення, сесійного ключа й іншої інформації.

Коли клієнт одержує відповідь KDC до сервера він витягає квиток і клієнтську копію сесійного ключа, розміщює їх у безпечний кеш (розташований в оперативній пам'яті, не на диску) Коли клієнт одержує відповідь KDC він витягає квиток і клієнтську копію сесійного ключа поміщає обидва у безпечний кеш розташований у пам'яті.

Коли клієнт хоче одержати доступ до сервера він висилає серверу повідомлення, що складається із квитка, що усе ще зашифрований секретним ключем сервера, автентифікатор, що зашифрований сесійним ключем. Квиток і автентифікатор разом становлять клієнтське посвідчення до сервера.

Переваги квитків на одержання сервісу. Сервер не зберігає сесійний ключ, що використовується для зв'язку із клієнтом. Це відповідальність клієнта

зберігати квиток сервера в кеші для посвідчень і надавати квиток щоразу, коли йому потрібний доступ до сервера. Щоразу, коли сервер одержує квиток на одержання сервісу від клієнта, сервер може використовувати закритий ключ для розшифрування квитка й добування сесійного ключа. Коли серверу не потрібний сесійний ключ він може видалити його.

Клієнт не має потреби в тому, щоб звертатися до KDC щоразу, коли він хоче одержати доступ до цільового сервера. Для захисту від можливості крадіжки копії квитка квиток на одержання сервісу має час дії, що визначає KDC у структурі даних квитка.

Як довго квиток діє залежить від політики області. Коли користувач виходить із мережі кеш посвідчень очищується й всі квитки на одержання сервісу також як і всі сесійні ключі знищуються.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена

версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа. Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі складається з наступних функціональних блоків:

– Контекстне вспливаюче меню.

– Функціональні кнопки ПЗ: Редагування БД користувачів; Зміна параметрів ПЗ; Зміна паролю; Редагування БД серверу; Зміна прав доступу; Перегляд журналу подій; Перегляд авторського права.

– Вікна БД користувачів, їх ключів та прапорів.

– Вікна БД сервера.

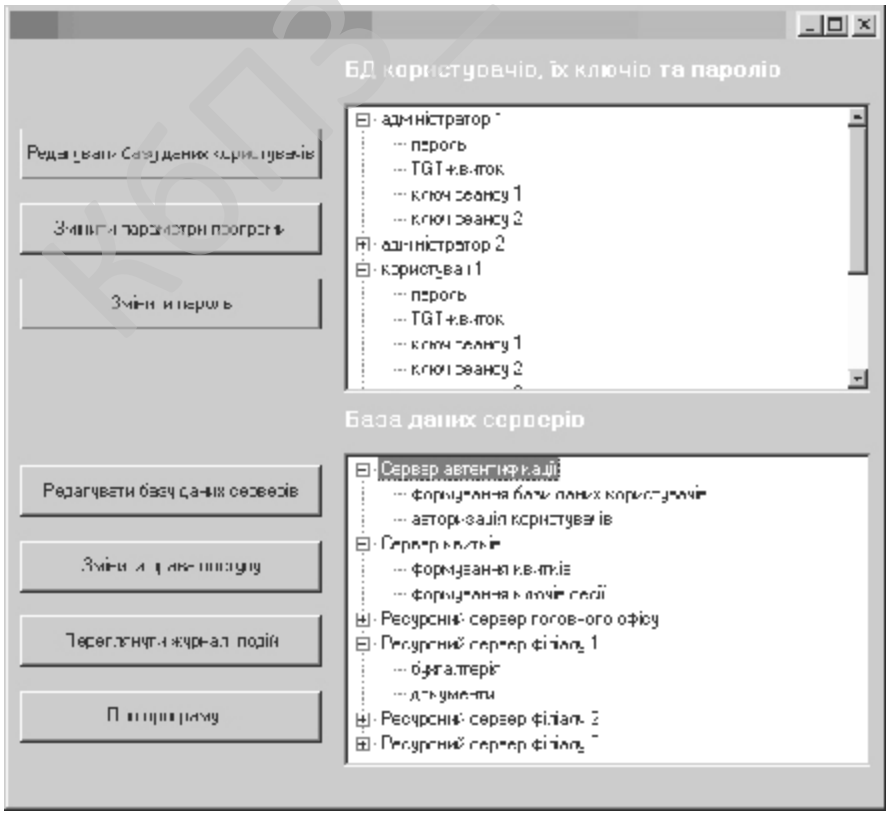

Рисунок 5.1 – Головне вікно розробленого ПЗ

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.

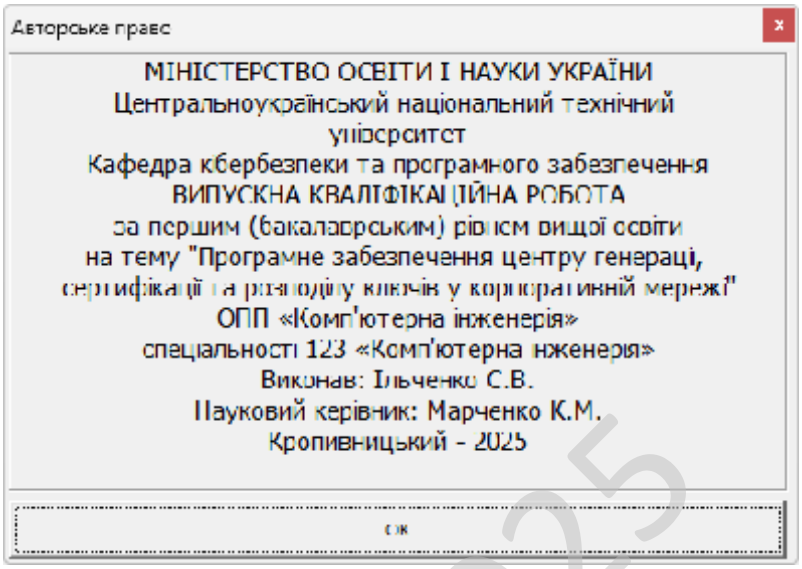

Рисунок 5.2 – Вікно розробника ПЗ

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих центрів генерації, сертифікації та розподілу ключів у корпоративній мережі.

– Досліджена система генерації, сертифікації та розподілу ключів у корпоративній мережі.

– На основі отриманих результатів досліджень створена програмна реалізація центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання генерації, сертифікації та розподілу ключів у корпоративній мережі.

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 Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для центру генерації, сертифікації та розподілу ключів у корпоративній мережі. Це

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

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. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

2. 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.

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

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

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

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

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

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

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

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

11. 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.

12. Kuznetsov, O., Smirnov, O., Mormul, M., Kotukh, Y., Zvieriev, V. «Comparative Research on Cryptocurrency Efficiency: An Objective Analysis of Key Metrics». *International Journal of Computing* 23(4), 2024. pp. 563-573.

13. Kuznetsov O., Frontoni E., Kuznetsova K., Smirnov O., Kostenko V. «Blockchain applications in metaverse environments: new horizons». *Advanced Metaverse Wireless Communication Systems.* pp. 255-293. 2024.

14. Kuznetsov, O., Frontoni, E., Chevardin, V., Smirnov, O., Imoize, A.L. «Advancing metaverse security with cryptographic innovations». *Advanced Metaverse Wireless Communication Systems.* pp 351-386. 2024.

15. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

16. 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.

17. 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.

18. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

19. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

20. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки

програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

28. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

29. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

30. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

31. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

32. 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,

33. 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.

34. 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.

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

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

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

38. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»: навчальний посібник – Кропивницький: ЦНТУ – 2022. – 968 с.

39. Теорія та практика сучасного інформаційно-психологічного протиборства: навчальний посібник / [В.М. Петрик, С.О. Гнатюк, М.М. Присяжнюк та ін.]; за заг. ред. С.О. Гнатюка, В.М. Петрика та О.А Смірнова. – Полтава, 2022. – 334 с.

40. 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

41. 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.

42. 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.

43. 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.*

44. 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.

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

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

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. № 47-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі.

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 з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– центру генерації, сертифікації та розподілу ключів у корпоративній мережі;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Builder C++.

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 – 92 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 9.06.2025 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

_____ Марченко К.М.

Програмне забезпечення центру генерації, сертифікації та розподілу ключів у корпоративній мережі

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 38

Літера: РП

Кропивницький – 2025 року

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//Підключення форм користувача-------------------------------------------------
-------------------------
USEFORM("client.cpp", Form1);
USEFORM("about.cpp", Form2);
USEFORM("avt.cpp", Form3);
USEFORM("server.cpp", Form4);

//Створення форм за допомогою WINAPI ------------------------------------------
--------------------------------

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm3), &Form3);
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->CreateForm(__classid(TForm2), &Form2);
                Application->CreateForm(__classid(TForm4), &Form4);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;
}
//---------------------------------------------------------------------------
```

Файл avt.cpp – авторизація користувача

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "avt.h"
#include "client.h"
#include "server.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button1Click(TObject *Sender)
{

if((Edit1->Text=="1")&(Edit2->Text=="1")) {
Form1->Show();
Form3->Visible=false;
}
if((Edit1->Text=="2")&(Edit2->Text=="2")) {
Form4->Show();
Form3->Visible=false;
}
else   { Label1->Caption="Помилка авторизації";
Label1->Font->Color=clRed;
}
}
//---------------------------------------------------------------------------
```

Файл avt.h – бібліотека для файлу avt.cpp

```
//---------------------------------------------------------------------------

#ifndef avtH
#define avtH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm3 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TEdit *Edit1;
        TEdit *Edit2;
        TButton *Button1;
        TImage *Image1;
        TImage *Image2;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:     // User declarations
        __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```

Файл client.cpp програми-клієнту

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "client.h"
#include "avt.h"
#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
//відкриття вікна "про програму..."
Form2->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
// запит квитка клієнтом

int KFW_request_tickets_for_client(char * user)
{
    krb5_context        ctx = 0;
    krb5_error_code          code;
    krb5_principal      princ = 0;
    krb5_ccache              cc  = 0;

    if (!pkrb5_init_context)
        return 0;

    code = pkrb5_init_context(&ctx);
    if (code) return 1;

    code = pkrb5_parse_name(ctx, user, &princ);
    if (code) goto loop_cleanup;

    code = KFW_get_ccache(ctx, princ, &cc);
    if (code) goto loop_cleanup;

    code = pkrb5_cc_destroy(ctx, cc);
    if (!code) cc = 0;

  loop_cleanup:
    if ( cc ) {
        pkrb5_cc_close(ctx, cc);
        cc = 0;
    }
    if ( princ ) {
        pkrb5_free_principal(ctx, princ);
        princ = 0;
    }

    pkrb5_free_context(ctx);
    return 0;
}
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
//доступ до ресурсу клієнтом

//Опис змінних
      char *ignore_str = "--ignore=";
      int ignore_len;
      char *cp, tmp[80];
      char *win_flag;
      char wflags[1024];

#ifdef _WIN32
      win_flag = win32_flag;
#else
      win_flag = "UNIX##";
#endif

      wflags[0] = 0;

//Створення списку ресурсів
      ignore_len = strlen(ignore_str);
      argc--; argv++;
      while (*argv && *argv[0] == '-') {
            wflags[sizeof(wflags) - 1] = '\0';
            if (strlen (wflags) + 1 + strlen (*argv) > sizeof (wflags) - 1) {
                  fprintf (stderr,
                         "wconfig: Дуже багато змінних (internal limit %d)",
                         sizeof (wflags));
                  exit (1);
            }
            if (wflags[0])
                  strcat(wflags, " ");
            strcat(wflags, *argv);

            if (!strcmp(*argv, "--mit")) {
                  mit_specific = 1;
                  argc--; argv++;
                  continue;
            }
            if (!strcmp(*argv, "--win16")) {
                  win_flag = win16_flag;
                  argc--; argv++;
                  continue;
            }
            if (!strcmp(*argv, "--win32")) {
                  win_flag = win32_flag;
                  argc--; argv++;
                  continue;
            }
            if (!strncmp(*argv, "--enable-", 9)) {
                  sprintf(tmp, "%s##", (*argv)+ignore_len);
                  for (cp = tmp; *cp; cp++) {
                        if (islower(*cp))
                              *cp = toupper(*cp);
                  }
                  cp = malloc(strlen(tmp)+1);
                  if (!cp) {
                        fprintf(stderr,
                              "wconfig: malloc failed!\n");
                        exit(1);
                  }
                  strcpy(cp, tmp);
                  add_ignore_list(cp);
                  argc--; argv++;
                  continue;
            }
            if (!strncmp(*argv, ignore_str, ignore_len)) {
```

```
                add_ignore_list((*argv)+ignore_len);
                argc--; argv++;
                continue;
            }
            fprintf(stderr, "Помилка в опціях: %s\n", *argv);
            exit(1);
      }
// Створення списку ресурсів, які заборонені
      if (win_flag)
            add_ignore_list(win_flag);

      if (mit_specific)
            add_ignore_list("MIT##");

      if (wflags[0] && (argc > 0))
            printf("WCONFIG_FLAGS=%s\n", wflags);

      if (argc > 0)
            copy_file (*argv, "win-pre.in");

      copy_file("", "-");

      if (argc > 0)
            copy_file (*argv, "win-post.in");

      return 0;
}

char *ignore_list[64] = {
      "DOS##",
      "DOS",
      };

/*
 * Додавання нових значень до листа ігнорування
 */
void add_ignore_list(char *str)
{
      char **cpp;

      for (cpp = ignore_list; *cpp; cpp++)
            ;
      *cpp = str;
}


/*
 *
 * копіювання файлу доступу
 *
 * копіювання файлу 'path\fname' to stdout.
 *
 */
static int
copy_file (char *path, char *fname)
{
    FILE *fin;
    char buf[1024];
    char **cpp, *ptr;
    int len;

    if (strcmp(fname, "-") == 0) {
          fin = stdin;
    } else {
#ifdef _WIN32
          sprintf(buf, "%s\\%s", path, fname);
#else
          sprintf(buf, "%s/%s", path, fname);
#endif
```

```
          fin = fopen (buf, "r");                    /* File to read */
          if (fin == NULL) {
                fprintf(stderr, "wconfig: файл доступу не відкрито %s\n", buf);
                return 1;
          }
    }


    while (fgets (buf, sizeof(buf), fin) != NULL) { /* копіювання файлу доступу
після завершення роботи */
          if (buf[0] == '@') {
                fputs("\n", stdout);
                continue;
          }
          if (buf[0] != '#' || buf[1] != '#') {
                fputs(buf, stdout);
                continue;
          }
          ptr = buf;
          for (cpp = ignore_list; *cpp; cpp++) {
                len = strlen(*cpp);
                if (memcmp (*cpp, buf+2, len) == 0) {
                      ptr += 2+len;
                      break;
                }
          }
          fputs(ptr, stdout);
    }

    fclose (fin);

    return 0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
//зміна пароля клієнтом
code = pkrb5_change_password_client(ctx,
                                    &my_creds,
                                    me,
                                    password, // пароль
                                    NULL,     //
                                    hParent, //
                                    0, // час початку
                                    0, // ім'я сервісу
                                    &options);
    if (code)
      goto cleanup;

    code = pkrb5_cc_initialize(ctx, cc, me);
    if (code)
      goto cleanup;

    code = pkrb5_cc_store_cred(ctx, cc, &my_creds);
    if (code)
      goto cleanup;
// очищення паролю клієнта
 cleanup:
    if ( addrs ) {
        for ( i=0;i<addr_count;i++ ) {
            if ( addrs[i] ) {
                if ( addrs[i]->contents )
                    free(addrs[i]->contents);
                free(addrs[i]);
            }
        }
    }
//запис нового паролю який генерується системою
    if (my_creds.client == me)
```

```
      my_creds.client = 0;
    pkrb5_free_cred_contents(ctx, &my_creds);
    if (name)
        pkrb5_free_unparsed_name(ctx, name);
    if (me)
        pkrb5_free_principal(ctx, me);
    if (cc && (cc != alt_cc))
        pkrb5_cc_close(ctx, cc);
    if (ctx && (ctx != alt_ctx))
        pkrb5_free_context(ctx);
    return(code);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
Form3->Close();
}
```

Файл client.h – бібліотека для файлу client.cpp

```
//---------------------------------------------------------------------------

#ifndef clientH
#define clientH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TButton *Button2;
        TButton *Button3;
        TTreeView *TreeView1;
        TButton *Button1;
        TButton *Button4;
        TImage *Image1;
        TLabel *Label1;
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
private: // User declarations
public:     // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Файл server.cpp програми-серверу

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "server.h"
#include "avt.h"
#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::FormClose(TObject *Sender, TCloseAction &Action)
{
 Form3->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button1Click(TObject *Sender)
{
//редагування бази даних користувачів
//опис зміних
int argc;
    char *argv[];
{
    extern char *optarg;
    int optchar, i, n;
    char tmp[4096], tmp2[BUFSIZ], *str_newprinc;

    krb5_error_code retval;
    char *dbname = 0;
    int enctypedone = 0;
    extern krb5_kt_ops krb5_ktf_writable_ops;
    int num_to_create;
    char principal_string[BUFSIZ];
    char *suffix = 0;
    int depth;

    krb5_init_context(&test_context);

    if (strrchr(argv[0], '/'))
      argv[0] = strrchr(argv[0], '/')+1;

    progname = argv[0];

    memset(principal_string, 0, sizeof(principal_string));
    num_to_create = 0;
    depth = 1;

    while ((optchar = getopt(argc, argv, "D:P:p:n:d:r:k:M:e:m")) != -1) {
      switch(optchar) {
      case 'D':
          depth = atoi(optarg);
          break;
        case 'P':
          mkey_password = optarg;
          break;
      case 'p':                         /* посилання на ім’я користувача, що
створюється */
```

```
        strncpy(principal_string, optarg, sizeof(principal_string) - 1);
        principal_string[sizeof(principal_string) - 1] = '\0';
        suffix = principal_string + strlen(principal_string);
        break;
    case 'n':                          /* дуже багато користувачів */
        num_to_create = atoi(optarg);
        break;
    case 'd':               /* установка імені бази даних користувачів */
        dbname = optarg;
        break;
    case 'r':
        cur_realm = optarg;
        break;
    case 'k':
        master_keyblock.enctype = atoi(optarg);
        enctypedone++;
        break;
    case 'M':               /* майстер ключ бази даних користувачів */
        mkey_name = optarg;
        break;
    case 'm':
        manual_mkey = TRUE;
        break;
    case '?':
    default:
        usage(progname, 1);
          }
}

if (!(num_to_create && suffix)) usage(progname, 1);


if ((retval = krb5_kt_register(test_context, &krb5_ktf_writable_ops))) {
    if (retval != KRB5_KT_TYPE_EXISTS) {
    com_err(progname, retval,
        "Реєстрація ключових таблиць бази даних користувачів ");
    exit(1);
  }
}

if (!enctypedone)
  master_keyblock.enctype = DEFAULT_KDC_ENCTYPE;

if (!krb5_c_valid_enctype(master_keyblock.enctype)) {
  com_err(progname, KRB5_PROG_ETYPE_NOSUPP,
        "встановлення типів %d", master_keyblock.enctype);
  exit(1);
}

if (!dbname)
  dbname = DEFAULT_KDB_FILE;

if (!cur_realm) {
  if ((retval = krb5_get_default_realm(test_context, &cur_realm))) {
      com_err(progname, retval, "пошук імені користувача у базі");
      exit(1);
  }
}
if ((retval = set_dbname_help(progname, dbname)))
  exit(retval);

for (n = 1; n <= num_to_create; n++) {
  /* Побудова нового імені користувача */
  /* для початку генеруємо нові імена*/
  (void) sprintf(suffix, "%d", n);
  (void) sprintf(tmp, "%s-DEPTH-1", principal_string);
  tmp[sizeof(tmp) - 1] = '\0';
  str_newprinc = tmp;
  add_princ(test_context, str_newprinc);
```

```
      for (i = 2; i <= depth; i++) {
      (void) sprintf(tmp2, "/%s-DEPTH-%d", principal_string, i);
      tmp2[sizeof(tmp2) - 1] = '\0';
      strncat(tmp, tmp2, sizeof(tmp) - 1 - strlen(tmp));
      str_newprinc = tmp;
      add_princ(test_context, str_newprinc);
      }
    }

    retval = krb5_db_fini(test_context);
    memset((char *)master_keyblock.contents, 0,
         (size_t) master_keyblock.length);
    if (retval && retval != KRB5_KDB_DBNOTINITED) {
      com_err(progname, retval, "Кінець наповнення бази даних користувачів");
      exit(1);
    }
    if (master_princ_set) {
      krb5_free_principal(test_context, master_princ);
    }
    krb5_free_context(test_context);
    exit(0);
}

void
add_princ(context, str_newprinc)
    krb5_context    context;
    char          * str_newprinc;
{
    krb5_error_code       retval;
    krb5_principal        newprinc;
    krb5_db_entry         newentry;
    char            princ_name[4096];

    memset((char *)&newentry, 0, sizeof(newentry));
    sprintf(princ_name, "%s@%s", str_newprinc, cur_realm);
    if ((retval = krb5_parse_name(context, princ_name, &newprinc))) {
      com_err(progname, retval, "аналіз імені '%s'", princ_name);
      return;
    }

    /* добавляємо дані на імя користувача */
    newentry.len = KRB5_KDB_V1_BASE_LENGTH;
    newentry.attributes = mblock.flags;
    newentry.max_life = mblock.max_life;
    newentry.max_renewable_life = mblock.max_rlife;
    newentry.expiration = mblock.expiration;
    newentry.pw_expiration = mblock.expiration;

    /* Додаємо характеристики в базу даних */
    if ((retval = krb5_copy_principal(context, newprinc, &newentry.princ))) {
            com_err(progname, retval, "кодуємо дані при запису до бази даних
користувача'%s'",
              princ_name);
      krb5_free_principal(context, newprinc);
      goto error;
    }

    {
      /* Реалізуємо обробку даних */
      krb5_int32 now;

      retval = krb5_timeofday(context, &now);
      if (retval) {
          com_err(progname, retval, "при виборі даних");
          krb5_free_principal(context, newprinc);
          goto error;
      }
      retval = krb5_dbe_update_mod_princ_data(context, &newentry, now,
```

```
                                        master_princ);
      if (retval) {
          com_err(progname, retval, "кодуємо дані при додавані до бази даних");
          krb5_free_principal(context, newprinc);
          goto error;
      }
    }

    {   /* додаємо ключ користувача до бази даних */
        krb5_data pwd, salt;
        krb5_keyblock key;

        if ((retval = krb5_principal2salt(context, newprinc, &salt))) {
          com_err(progname, retval, "перетворюємо ключ'%s'",
                princ_name);
          krb5_free_principal(context, newprinc);
          goto error;
        }

      krb5_free_principal(context, newprinc);

      pwd.length = strlen(princ_name);
      pwd.data = princ_name;  /* повинно буди регенеруватися */
      if ((retval = krb5_c_string_to_key(context, master_keyblock.enctype,
                                &pwd, &salt, &key))) {
          com_err(progname,retval,"конвертуємо парол до ключа'%s'",
                princ_name);
          krb5_free_data_contents(context, &salt);
          goto error;
      }
      krb5_free_data_contents(context, &salt);

      if ((retval = krb5_dbe_create_key_data(context, &newentry))) {
          com_err(progname, retval, "Створюємо ключові дані користувача '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }

        if ((retval = krb5_dbekd_encrypt_key_data(context,&master_keyblock,
                                    &key, NULL, 1,
                                    newentry.key_data))) {
          com_err(progname, retval, "кодуємо ключові дані для '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }
        free(key.contents);
    }

    {
      int one = 1;

      if ((retval = krb5_db_put_principal(context, &newentry, &one))) {
          com_err(progname, retval, "Запам’ятовуємо дані");
          goto error;
      }
      if (one != 1) {
         com_err(progname,0,"Запис не знайдено в базі користувачів (невідома
помилка)");
          goto error;
      }
    }

    fprintf(stdout, "Додано %s до бази користувачів\n", princ_name);

error: /* Очищуємо змінні */
#if 0
    krb5_dbe_free_contents(context, &newentry);
```

```
#endif
    krb5_db_free_principal(context, &newentry, 1);
    return;
}

int
set_dbname_help(pname, dbname)
char *pname;
char *dbname;
{
    krb5_error_code retval;
    int nentries;
    krb5_boolean more;
    krb5_data pwd, scratch;
    char *args[2];

    /* створюємо й перевіряємо майстер ключ користувача */

    if ((retval = krb5_db_setup_mkey_name(test_context, mkey_name, cur_realm,
                                0, &master_princ))) {
      com_err(pname, retval, "установка імені майстер ключа користувача");
      return(1);
    }
    master_princ_set = 1;
    if (mkey_password) {
      pwd.data = mkey_password;
      pwd.length = strlen(mkey_password);
      retval = krb5_principal2salt(test_context, master_princ, &scratch);
      if (retval) {
          com_err(pname, retval, "розраховуємо майстер ключ");
          return(1);
      }
      if ((retval = krb5_c_string_to_key(test_context,
                                master_keyblock.enctype,
                                &pwd, &scratch,
                                &master_keyblock))) {
          com_err(pname, retval,
                "перетворюємо майстер ключ до паролю користуваача");
          return(1);
      }
      free(scratch.data);
    } else {
      if ((retval = krb5_db_fetch_mkey(test_context, master_princ,
                               master_keyblock.enctype, manual_mkey,
                               FALSE, 0, NULL, &master_keyblock))) {
          com_err(pname, retval, "while reading master key");
          return(1);
      }
    }

    if ((retval = krb5_set_default_realm(test_context, cur_realm))) {
      com_err(pname, retval, "встановлюємо встроєну область");
      return 1;
    }
    /* Pathname is passed to db2 via 'args' parameter.  */
    args[1] = NULL;
    args[0] = malloc(sizeof("dbname=") + strlen(dbname));
    if (args[0] == NULL) {
      com_err(pname, errno, "Встановлюємо параметри бази даних користувача");
      return 1;
    }
    sprintf(args[0], "dbname=%s", dbname);

    if ((retval = krb5_db_open(test_context, args, KRB5_KDB_OPEN_RO))) {
      com_err(pname, retval, "ініціалізуємо базу даних користувача");
      return(1);
    }
    free(args[0]);
```

```
    if ((retval = krb5_db_verify_master_key(test_context, master_princ,
                                  &master_keyblock))){
      com_err(pname, retval, "перевіряєм майстер ключ користувача");
      (void) krb5_db_fini(test_context);
      return(1);
    }
    nentries = 1;
    if ((retval = krb5_db_get_principal(test_context, master_princ,
                               &master_entry, &nentries, &more))) {
      com_err(pname, retval, "Пошук основного входу");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (more) {
      com_err(pname, KRB5KDC_ERR_PRINCIPAL_NOT_UNIQUE,
            " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (!nentries) {
      com_err(pname, KRB5_KDB_NOENTRY, " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    }

    mblock.max_life = master_entry.max_life;
    mblock.max_rlife = master_entry.max_renewable_life;
    mblock.expiration = master_entry.expiration;

    /* не змінюємо прапорці */
    mblock.mkvno = master_entry.key_data[0].key_data_kvno;

    krb5_db_free_principal(test_context, &master_entry, nentries);
    return 0;
} }
//---------------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
//зміна параметрів програми
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button4Click(TObject *Sender)
{
//зміна паролю адміністратора


code = pkrb5_change_password_admin(ctx,
                    &my_creds,
                    me,
                    password, // пароль адміністратора
                    NULL,     //
                    hParent, //
                    0, // час початку
                    0, // ім'я сервісу
                    &options);
    if (code)
      goto cleanup;

    code = pkrb5_cc_initialize(ctx, cc, me);
    if (code)
      goto cleanup;

    code = pkrb5_cc_store_cred(ctx, cc, &my_creds);
    if (code)
      goto cleanup;
// очищення паролю адміністратора
 cleanup:
    if ( addrs ) {
        for ( i=0;i<addr_count;i++ ) {
```

```
            if ( addrs[i] ) {
                if ( addrs[i]->contents )
                    free(addrs[i]->contents);
                free(addrs[i]);
            }
        }
    }
//запис нового паролю адміністратора який генерується системою
    if (my_creds.client == me)
      my_creds.client = 0;
    pkrb5_free_cred_contents(ctx, &my_creds);
    if (name)
        pkrb5_free_unparsed_name(ctx, name);
    if (me)
        pkrb5_free_principal(ctx, me);
    if (cc && (cc != alt_cc))
        pkrb5_cc_close(ctx, cc);
    if (ctx && (ctx != alt_ctx))
        pkrb5_free_context(ctx);
    return(code);
}

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button2Click(TObject *Sender)
{
//редагування бази даних серверів

//опис зміних
int argc;
    char *argv[];
{
    extern char *optarg;
    int optchar, i, n;
    char tmp[4096], tmp2[BUFSIZ], *str_newprinc;

    krb5_error_code retval;
    char *dbname = 0;
    int enctypedone = 0;
    extern krb5_kt_ops krb5_ktf_writable_ops;
    int num_to_create;
    char principal_string[BUFSIZ];
    char *suffix = 0;
    int depth;

    krb5_init_context(&test_context);

    if (strrchr(argv[0], '/'))
      argv[0] = strrchr(argv[0], '/')+1;

    progname = argv[0];

    memset(principal_string, 0, sizeof(principal_string));
    num_to_create = 0;
    depth = 1;

    while ((optchar = getopt(argc, argv, "D:P:p:n:d:r:k:M:e:m")) != -1) {
      switch(optchar) {
      case 'D':
          depth = atoi(optarg);
          break;
        case 'P':
          mkey_password = optarg;
          break;
      case 'p':                       /* посилання на ім’я серверу, що
створюється */
          strncpy(principal_string, optarg, sizeof(principal_string) - 1);
          principal_string[sizeof(principal_string) - 1] = '\0';
```

```
        suffix = principal_string + strlen(principal_string);
        break;
    case 'n':                          /* дуже багато серверів */
        num_to_create = atoi(optarg);
        break;
    case 'd':               /* установка імені бази даних серверів */
        dbname = optarg;
        break;
    case 'r':
        cur_realm = optarg;
        break;
    case 'k':
        master_keyblock.enctype = atoi(optarg);
        enctypedone++;
        break;
    case 'M':               /* майстер ключ бази даних серверів */
        mkey_name = optarg;
        break;
    case 'm':
        manual_mkey = TRUE;
        break;
    case '?':
    default:
        usage(progname, 1);
          }
}

if (!(num_to_create && suffix)) usage(progname, 1);


if ((retval = krb5_kt_register(test_context, &krb5_ktf_writable_ops))) {
    if (retval != KRB5_KT_TYPE_EXISTS) {
    com_err(progname, retval,
        "Реєстрація ключових таблиць бази даних серверів ");
    exit(1);
  }
}

if (!enctypedone)
  master_keyblock.enctype = DEFAULT_KDC_ENCTYPE;

if (!krb5_c_valid_enctype(master_keyblock.enctype)) {
  com_err(progname, KRB5_PROG_ETYPE_NOSUPP,
        "встановлення типів %d", master_keyblock.enctype);
  exit(1);
}

if (!dbname)
  dbname = DEFAULT_KDB_FILE;

if (!cur_realm) {
  if ((retval = krb5_get_default_realm(test_context, &cur_realm))) {
      com_err(progname, retval, "пошук імені сервера у базі");
      exit(1);
  }
}
if ((retval = set_dbname_help(progname, dbname)))
  exit(retval);

for (n = 1; n <= num_to_create; n++) {
  /* Побудова нового імені сервера */
  /* для початку генеруємо нові імена*/
  (void) sprintf(suffix, "%d", n);
  (void) sprintf(tmp, "%s-DEPTH-1", principal_string);
  tmp[sizeof(tmp) - 1] = '\0';
  str_newprinc = tmp;
  add_princ(test_context, str_newprinc);

  for (i = 2; i <= depth; i++) {
```

```
      (void) sprintf(tmp2, "/%s-DEPTH-%d", principal_string, i);
      tmp2[sizeof(tmp2) - 1] = '\0';
      strncat(tmp, tmp2, sizeof(tmp) - 1 - strlen(tmp));
      str_newprinc = tmp;
      add_princ(test_context, str_newprinc);
      }
    }

    retval = krb5_db_fini(test_context);
    memset((char *)master_keyblock.contents, 0,
         (size_t) master_keyblock.length);
    if (retval && retval != KRB5_KDB_DBNOTINITED) {
      com_err(progname, retval, "Кінець наповнення бази даних серверів");
      exit(1);
    }
    if (master_princ_set) {
      krb5_free_principal(test_context, master_princ);
    }
    krb5_free_context(test_context);
    exit(0);
}

void
add_princ(context, str_newprinc)
    krb5_context    context;
    char          * str_newprinc;
{
    krb5_error_code       retval;
    krb5_principal        newprinc;
    krb5_db_entry         newentry;
    char            princ_name[4096];

    memset((char *)&newentry, 0, sizeof(newentry));
    sprintf(princ_name, "%s@%s", str_newprinc, cur_realm);
    if ((retval = krb5_parse_name(context, princ_name, &newprinc))) {
      com_err(progname, retval, "аналіз імені '%s'", princ_name);
      return;
    }

    /* добавляємо дані на імя сервера */
    newentry.len = KRB5_KDB_V1_BASE_LENGTH;
    newentry.attributes = mblock.flags;
    newentry.max_life = mblock.max_life;
    newentry.max_renewable_life = mblock.max_rlife;
    newentry.expiration = mblock.expiration;
    newentry.pw_expiration = mblock.expiration;

    /* Додаємо характеристики в базу даних */
    if ((retval = krb5_copy_principal(context, newprinc, &newentry.princ))) {
            com_err(progname, retval, "кодуємо дані при запису до бази даних
сервера'%s'",
              princ_name);
      krb5_free_principal(context, newprinc);
      goto error;
    }

    {
      /* Реалізуємо обробку даних */
      krb5_int32 now;

      retval = krb5_timeofday(context, &now);
      if (retval) {
          com_err(progname, retval, "при виборі даних");
          krb5_free_principal(context, newprinc);
          goto error;
      }
      retval = krb5_dbe_update_mod_princ_data(context, &newentry, now,
                                      master_princ);
      if (retval) {
```

```
          com_err(progname, retval, "кодуємо дані при додавані до бази даних");
          krb5_free_principal(context, newprinc);
          goto error;
      }
    }

    {   /* додаємо ключ сервера до бази даних */
        krb5_data pwd, salt;
        krb5_keyblock key;

        if ((retval = krb5_principal2salt(context, newprinc, &salt))) {
          com_err(progname, retval, "перетворюємо ключ'%s'",
                princ_name);
          krb5_free_principal(context, newprinc);
          goto error;
        }

      krb5_free_principal(context, newprinc);

      pwd.length = strlen(princ_name);
      pwd.data = princ_name;  /* повинно буди регенеруватися */
      if ((retval = krb5_c_string_to_key(context, master_keyblock.enctype,
                                &pwd, &salt, &key))) {
          com_err(progname,retval,"конвертуємо парол до ключа'%s'",
                princ_name);
          krb5_free_data_contents(context, &salt);
          goto error;
      }
      krb5_free_data_contents(context, &salt);

      if ((retval = krb5_dbe_create_key_data(context, &newentry))) {
          com_err(progname, retval, "Створюємо ключові дані сервера '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }

        if ((retval = krb5_dbekd_encrypt_key_data(context,&master_keyblock,
                                    &key, NULL, 1,
                                    newentry.key_data))) {
          com_err(progname, retval, "кодуємо ключові дані для '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }
        free(key.contents);
    }

    {
      int one = 1;

      if ((retval = krb5_db_put_principal(context, &newentry, &one))) {
          com_err(progname, retval, "Запам’ятовуємо дані");
          goto error;
      }
      if (one != 1) {
         com_err(progname,0,"Запис не знайдено в базі серверів (невідома
помилка)");
          goto error;
      }
    }

    fprintf(stdout, "Додано %s до бази серверів\n", princ_name);

error: /* Очищуємо змінні */
#if 0
    krb5_dbe_free_contents(context, &newentry);
#endif
    krb5_db_free_principal(context, &newentry, 1);
```

```
    return;
}

int
set_dbname_help(pname, dbname)
char *pname;
char *dbname;
{
    krb5_error_code retval;
    int nentries;
    krb5_boolean more;
    krb5_data pwd, scratch;
    char *args[2];

    /* створюємо й перевіряємо майстер ключ сервера */

    if ((retval = krb5_db_setup_mkey_name(test_context, mkey_name, cur_realm,
                                0, &master_princ))) {
      com_err(pname, retval, "установка імені майстер ключа сервера");
      return(1);
    }
    master_princ_set = 1;
    if (mkey_password) {
      pwd.data = mkey_password;
      pwd.length = strlen(mkey_password);
      retval = krb5_principal2salt(test_context, master_princ, &scratch);
      if (retval) {
          com_err(pname, retval, "розраховуємо майстер ключ");
          return(1);
      }
      if ((retval = krb5_c_string_to_key(test_context,
                                master_keyblock.enctype,
                                &pwd, &scratch,
                                &master_keyblock))) {
          com_err(pname, retval,
                "перетворюємо майстер ключ до паролю користуваача");
          return(1);
      }
      free(scratch.data);
    } else {
      if ((retval = krb5_db_fetch_mkey(test_context, master_princ,
                               master_keyblock.enctype, manual_mkey,
                               FALSE, 0, NULL, &master_keyblock))) {
          com_err(pname, retval, "while reading master key");
          return(1);
      }
    }

    if ((retval = krb5_set_default_realm(test_context, cur_realm))) {
      com_err(pname, retval, "встановлюємо встроєну область");
      return 1;
    }
    /* Pathname is passed to db2 via 'args' parameter.  */
    args[1] = NULL;
    args[0] = malloc(sizeof("dbname=") + strlen(dbname));
    if (args[0] == NULL) {
      com_err(pname, errno, "Встановлюємо параметри бази даних сервера");
      return 1;
    }
    sprintf(args[0], "dbname=%s", dbname);

    if ((retval = krb5_db_open(test_context, args, KRB5_KDB_OPEN_RO))) {
      com_err(pname, retval, "ініціалізуємо базу даних сервера");
      return(1);
    }
    /* Done with args */
    free(args[0]);

    if ((retval = krb5_db_verify_master_key(test_context, master_princ,
```

```
                                    &master_keyblock))){
      com_err(pname, retval, "перевіряєм майстер ключ сервера");
      (void) krb5_db_fini(test_context);
      return(1);
    }
    nentries = 1;
    if ((retval = krb5_db_get_principal(test_context, master_princ,
                                &master_entry, &nentries, &more))) {
      com_err(pname, retval, "Пошук основного входу");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (more) {
      com_err(pname, KRB5KDC_ERR_PRINCIPAL_NOT_UNIQUE,
            " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (!nentries) {
      com_err(pname, KRB5_KDB_NOENTRY, " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    }

    mblock.max_life = master_entry.max_life;
    mblock.max_rlife = master_entry.max_renewable_life;
    mblock.expiration = master_entry.expiration;

    /* не змінюємо прапорці */
    mblock.mkvno = master_entry.key_data[0].key_data_kvno;

    krb5_db_free_principal(test_context, &master_entry, nentries);
    return 0;
}
}

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button5Click(TObject *Sender)
{
//зміна прав доступу

//опис змінних
extern char *optarg;
    int optchar, i, n;
    char tmp[4096], tmp2[BUFSIZ], *str_princ;

    krb5_context context;
    krb5_error_code retval;
    char *dbname = 0;
    int enctypedone = 0;
    int num_to_check;
    char principal_string[BUFSIZ];
    char *suffix = 0;
    int depth, errors;

    krb5_init_context(&context);

    if (strrchr(argv[0], '/'))
      argv[0] = strrchr(argv[0], '/')+1;

    progname = argv[0];

    memset(principal_string, 0, sizeof(principal_string));
    num_to_check = 0;
    depth = 1;
// вибір параметрів
    while ((optchar = getopt(argc, argv, "D:P:p:n:d:r:R:k:M:e:m")) != -1) {
      switch(optchar) {
      case 'D':
```

```
            depth = atoi(optarg);
            break;
          case 'P':
            mkey_password = optarg;
            break;
        case 'p':                          /* префікс імені для перевірки */
            strncpy(principal_string, optarg, sizeof(principal_string) - 1);
            principal_string[sizeof(principal_string) - 1] = '\0';
            suffix = principal_string + strlen(principal_string);
            break;
         case 'n':                           /* як багато для перевірки */
            num_to_check = atoi(optarg);
            break;
        case 'd':                /* установка імені бази даних */
            dbname = optarg;
            break;
        case 'r':
            cur_realm = optarg;
            break;
        case 'k':
            master_keyblock.enctype = atoi(optarg);
            enctypedone++;
            break;
        case 'M':                /* мастер ключ імені в базі даних */
            mkey_name = optarg;
            break;
        case 'm':
            manual_mkey = TRUE;
            break;
        case '?':
        default:
            usage(progname, 1);
            /*Не досягнено бажаного результату*/
        }
      }

      if (!(num_to_check && suffix)) usage(progname, 1);

      if (!enctypedone)
        master_keyblock.enctype = DEFAULT_KDC_ENCTYPE;
// Перевіка валідності ключа користувача
      if (!krb5_c_valid_enctype(master_keyblock.enctype)) {
        com_err(progname, KRB5_PROG_ETYPE_NOSUPP,
              "while setting up enctype %d", master_keyblock.enctype);
        exit(1);
      }

      krb5_use_enctype(context, &master_encblock, master_keyblock.enctype);
// Перевікаа по базі даних
      if (!dbname)
        dbname = DEFAULT_KDB_FILE;

      if (!cur_realm) {
        if ((retval = krb5_get_default_realm(context, &cur_realm))) {
            com_err(progname, retval, "while retrieving default realm name");
            exit(1);
        }
      }
      if ((retval = set_dbname_help(context, progname, dbname)))
        exit(retval);

      errors = 0;

      fprintf(stdout, "\nПеревірка ");

      for (n = 1; n <= num_to_check; n++) {
        /* Змінюємо параметри */
        /* Не вибираємо будь-які імена, тому, що потрібно генерувати імена, які є
в базі даних та до яких є ключ */
```

```
      (void) sprintf(suffix, "%d", n);
      (void) sprintf(tmp, "%s-DEPTH-1", principal_string);
      str_princ = tmp;
      if (check_princ(context, str_princ)) errors++;

      for (i = 2; i <= depth; i++) {
      (void) sprintf(tmp2, "/%s-DEPTH-%d", principal_string, i);
      tmp2[sizeof(tmp2) - 1] = '\0';
      strncat(tmp, tmp2, sizeof(tmp) - 1 - strlen(tmp));
      str_princ = tmp;
      if (check_princ(context, str_princ)) errors++;
      }
    }

    if (errors)
      fprintf(stdout, "\n%d Помилка.\n", errors);
    else
      fprintf(stdout, "\n Помилки немає.\n");

    krb5_finish_random_key(context, &master_encblock, &master_random);
    krb5_finish_key(context, &master_encblock);

    retval = krb5_db_fini(context);
    memset((char *)master_keyblock.contents, 0, (size_t)
master_keyblock.length);
    if (retval && retval != KRB5_KDB_DBNOTINITED) {
      com_err(progname, retval, "При закритті бази даних");
      exit(1);
    }

    if (str_master_princ) {
      krb5_free_unparsed_name(context, str_master_princ);
    }
    krb5_free_principal(context, master_princ);
    krb5_free_context(context);
    exit(0);
}

int
check_princ(context, str_princ)
    krb5_context context;
    char * str_princ;
{
    krb5_error_code retval;
    krb5_db_entry kdbe;
    krb5_keyblock pwd_key, db_key;
    krb5_data pwd, salt;
    krb5_principal princ;
    krb5_boolean more;
    int nprincs = 1;
    /* char *str_mod_name; */
    char princ_name[4096];

    sprintf(princ_name, "%s@%s", str_princ, cur_realm);

    fprintf(stderr, "\t%s ...\n", princ_name);

    if ((retval = krb5_parse_name(context, princ_name, &princ))) {
      com_err(progname, retval, "while parsing '%s'", princ_name);
      goto out;
    }

    pwd.data = princ_name;  /* Повинно бути в змозі змінюватися */
    pwd.length = strlen(princ_name);

    if ((retval = krb5_principal2salt(context, princ, &salt))) {
      com_err(progname, retval, "При перетворенні принципових параметрів
розподілу ключів'%s'", princ_name);
      krb5_free_principal(context, princ);
```

```
      goto out;
    }

    if ((retval = krb5_string_to_key(context, &master_encblock,
                            &pwd_key, &pwd, &salt))) {
      com_err(progname, retval, "при перетворенні паролю в ключ для '%s'",
            princ_name);
      krb5_free_data_contents(context, &salt);
      krb5_free_principal(context, princ);
      goto out;
    }
    krb5_free_data_contents(context, &salt);

    if ((retval = krb5_db_get_principal(context, princ, &kdbe,
                              &nprincs, &more))) {
      com_err(progname, retval, "Перевіка існування параметрів та прав
доступу");
      krb5_free_principal(context, princ);
      goto out;
    }
    krb5_free_principal(context, princ);

    if (nprincs != 1) {
      com_err(progname, 0, "Знайдено %d входів в базу даних %s.\n", nprincs,
            princ_name);
      goto errout;
    }

    if ((retval = krb5_dbekd_decrypt_key_data(context, &master_keyblock,
                                        kdbe.key_data, &db_key, NULL))) {
      com_err(progname, retval, "Ключ декодується для '%s'", princ_name);
      goto errout;
    }

    if ((pwd_key.enctype != db_key.enctype) ||
      (pwd_key.length != db_key.length)) {
      fprintf (stderr, "\t Ключові типи не співпадають (%d expected, %d from
db)\n",
             pwd_key.enctype, db_key.enctype);
errout:
      krb5_db_free_principal(context, &kdbe, nprincs);
      return(-1);
    }
    else {
      if (memcmp((char *)pwd_key.contents, (char *) db_key.contents,
             (size_t) pwd_key.length)) {
      fprintf(stderr, "\t Ключ не відповідає збереженій величині для %s\n",
            princ_name);
      goto errout;
      }
    }

    free((char *)pwd_key.contents);
    free((char *)db_key.contents);

    if (kdbe.key_data[0].key_data_kvno != 1) {
      fprintf(stderr,"\t kvno не відповідає збереженій величині для %s.\n",
princ_name);
      goto errout;
    }

    if (kdbe.max_life != mblock.max_life) {
      fprintf(stderr, "\tmax не відповідає збереженій величині для %s.\n",
            princ_name);
      goto errout;
    }

    if (kdbe.max_renewable_life != mblock.max_rlife) {
      fprintf(stderr,
```

```
            "\tmax не відповідає збереженій величині для %s.\n",
            princ_name);
      goto errout;
    }

    if (kdbe.expiration != mblock.expiration) {
      fprintf(stderr, "\Час texpiration не відповідає збереженій величині для
%s.\n",
            princ_name);
      goto errout;
    }

/*
    if ((retval = krb5_unparse_name(context, kdbe.mod_name, &str_mod_name)))
      com_err(progname, retval, "для непарсингу імені режиму");
    else {
      if (strcmp(str_mod_name, str_master_princ) != 0) {
      fprintf(stderr, "\tmod не є визначеним іменем  (%s not %s).\n",
            str_mod_name, str_master_princ);
      free(str_mod_name);
      goto errout;
      }
      else free(str_mod_name);
    }
*/

    if (kdbe.attributes != mblock.flags) {
      fprintf(stderr, "\tАтрибути не відповідають збереженій величині для
%s.\n",
            princ_name);
      goto errout;
    }

    out:
    krb5_db_free_principal(context, &kdbe, nprincs);

    return(0);
}

int
set_dbname_help(context, pname, dbname)
    krb5_context context;
    char *pname;
    char *dbname;
{
    krb5_error_code retval;
    int nentries;
    krb5_boolean more;
    krb5_data pwd, scratch;
    char *args[2];

    /* створення нового ключа */

    if ((retval = krb5_db_setup_mkey_name(context, mkey_name, cur_realm, 0,
                                &master_princ))) {
      com_err(pname, retval, "при встановленні імені ключа");
      return(1);
    }
    if (mkey_password) {
      pwd.data = mkey_password;
      pwd.length = strlen(mkey_password);
      retval = krb5_principal2salt(context, master_princ, &scratch);
      if (retval) {
          com_err(pname, retval, "розраховується майстер ключ");
          return(1);
      }
      if ((retval = krb5_string_to_key(context, &master_encblock,
                            &master_keyblock, &pwd, &scratch))) {
          com_err(pname, retval,
```

```
            "перетворюємо майстер ключ в пароль");
        return(1);
    }
    free(scratch.data);
} else {
    if ((retval = krb5_db_fetch_mkey(context, master_princ,
                            master_keyblock.enctype,
                           manual_mkey, FALSE, (char *) NULL, 0,
                           &master_keyblock))) {
        com_err(pname, retval, "Зчитуємо майсер ключ");
        return(1);
    }
}

/* Поточний інтерфейс вимагає, щоб default_realm
   Поле було встановлено в  krb5_context.  */
if ((retval = krb5_set_default_realm(context, cur_realm))) {
    com_err(pname, retval, "Відповідні установки");
    return 1;
}
/* Захищений путь до  db2 з 'args' параметрами.  */
args[1] = NULL;
args[0] = malloc(sizeof("dbname=") + strlen(dbname));
if (args[0] == NULL) {
    com_err(pname, errno, "Установлюємо параметри бази даних");
    return 1;
}
sprintf(args[0], "dbname=%s", dbname);

if ((retval = krb5_db_open(context, args, KRB5_KDB_OPEN_RO))) {
    com_err(pname, retval, "Ініціалізуємо базу даних");
    return(1);
}
if ((retval = krb5_db_verify_master_key(context, master_princ,
                                &master_keyblock))) {
    com_err(pname, retval, "Перевіряємо майстер ключ");
    (void) krb5_db_fini(context);
    return(1);
}
nentries = 1;
if ((retval = krb5_db_get_principal(context, master_princ, &master_entry,
                                &nentries, &more))) {
    com_err(pname, retval, "Шукаємо текст ");
    (void) krb5_db_fini(context);
    return(1);
} else if (more) {
    com_err(pname, KRB5KDC_ERR_PRINCIPAL_NOT_UNIQUE,
          "Пошук основного входу");
    (void) krb5_db_fini(context);
    return(1);
} else if (!nentries) {
    com_err(pname, KRB5_KDB_NOENTRY, " Пошук основного входу ");
    (void) krb5_db_fini(context);
    return(1);
}

if ((retval = krb5_unparse_name(context, master_princ,
                          &str_master_princ))) {
    com_err(pname, retval, "Помилка пошуку");
    krb5_db_fini(context);
    return(1);
}

if ((retval = krb5_process_key(context,
                       &master_encblock, &master_keyblock))) {
    com_err(pname, retval, "Обробка майстер ключа");
    (void) krb5_db_fini(context);
    return(1);
}
```

```
    if ((retval = krb5_init_random_key(context,
                              &master_encblock, &master_keyblock,
                              &master_random))) {
      com_err(pname, retval, "ініціалізуємо генератор випадкових ключів");
      krb5_finish_key(context, &master_encblock);
      (void) krb5_db_fini(context);
      return(1);
    }
    mblock.max_life = master_entry.max_life;
    mblock.max_rlife = master_entry.max_renewable_life;
    mblock.expiration = master_entry.expiration;
    /* Не треба встановлювати інші флаги */
    mblock.mkvno = master_entry.key_data[0].key_data_kvno;

    krb5_db_free_principal(context, &master_entry, nentries);
    return 0; }
//---------------------------------------------------------------------------

void __fastcall TForm4::Button6Click(TObject *Sender)
{
//перегляд журналу подій
void
kmqint_dump_publisher(FILE * f) {

    int n_free = 0;
    int n_active = 0;
    kmq_message * m;

    EnterCriticalSection(&cs_kmq_msg);

    fprintf(f, "qp0\t*** Події ***\n");
    fprintf(f, "qp1\t Адреса\n");

    m = msg_free;
    while(m) {
        n_free++;

        fprintf(f, "qp2\t0x%p\n", m);

        m = LNEXT(m);
    }

    fprintf(f, "qp3\t Всього подій : %d\n", n_free);

    fprintf(f, "qp4\t*** Активних подій ***\n");
    fprintf(f,
"qp5\tAddress\tType\tSubtype\tuParam\tvParam\tnSent\tnCompleted\tnFailed\twait_o
\trefcount\n");

    m = msg_active;
    while(m) {

        n_active++;

        fprintf(f, "qp6\t0x%p\t%d\t%d\t0x%x\t0x%p\t%d\t%d\t%d\t0x%p\t%d\n",
                m,
                (int) m->type,
                (int) m->subtype,
                (unsigned int) m->uparam,
                m->vparam,
                (int) m->nSent,
                (int) m->nCompleted,
                (int) m->nFailed,
                (void *) m->wait_o,
                (int) m->refcount);

        m = LNEXT(m);
    }
```

```
    fprintf(f, "qp7\t Кількість активних подій = %d\n", n_active);

    fprintf(f, "qp8\t--- End ---\n");

    LeaveCriticalSection(&cs_kmq_msg);

}

#endif

/*! Кратке резюме дій об'єкту події */
kmq_message *
kmqint_get_message(void) {
    kmq_message * m;

    LPOP(&msg_free,&m);
    if(!m) {
        m = PMALLOC(sizeof(kmq_message));
    }
    ZeroMemory((void*)m, sizeof(kmq_message));

    LPUSH(&msg_active, m);

    return m;
}

void
kmqint_put_message(kmq_message *m) {
    int queued;
    /* Ми повинні звільнити місце под подію. У іншому випадку ми повинні чекати
поки подія не буде завершена */
    if(m->refcount == 0) {
        LDELETE(&msg_active, m);
        LeaveCriticalSection(&cs_kmq_msg);
        queued = kmqint_notify_msg_completion(m);
        EnterCriticalSection(&cs_kmq_msg);
        if (!queued) {
            if(m->err_ctx) {
                kherr_release_context(m->err_ctx);
                m->err_ctx = NULL;
            }
            if(m->wait_o) {
                CloseHandle(m->wait_o);
                m->wait_o = NULL;
            }
            LPUSH(&msg_free,m);
        }
    } else if(m->wait_o) {
        SetEvent(m->wait_o);
    }
}

/*! Отримуємо ::cs_kmq_msg, ::cs_kmq_types, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_send_message(khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * blob) {
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmqint_post_message_ex(type, subtype, uparam, blob, &c, TRUE);
    if(KHM_FAILED(rv))
        return rv;

    rv = kmq_wait(c, INFINITE);
    if(KHM_SUCCEEDED(rv) && c->nFailed > 0)
        rv = KHM_ERROR_PARTIAL;

    kmq_free_call(c);
```

```
    return rv;
}

/*! \Отримуємо ::cs_kmq_msg, ::cs_kmq_types, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_post_message(khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * blob) {
    return kmqint_post_message_ex(type, subtype, uparam, blob, NULL, FALSE);
}

/*! \Отримуємо ::cs_kmq_msg
    */
KHMEXP khm_int32 KHMAPI
kmq_free_call(kmq_call call) {
    kmq_message * m;

    m = call;

    EnterCriticalSection(&cs_kmq_msg);
    m->refcount--;
    if(!m->refcount) {
        kmqint_put_message(m);
    }
    LeaveCriticalSection(&cs_kmq_msg);

    return KHM_ERROR_SUCCESS;
}

/*! \Отримуємо ::cs_kmq_msg, ::cs_kmq_types, ::cs_kmq_msg_ref, kmq_queue::cs
    */
khm_int32
kmqint_post_message_ex(khm_int32 type, khm_int32 subtype, khm_ui_4 uparam,
    void * blob, kmq_call * call, khm_boolean try_send)
{
    kmq_message * m;
    kherr_context * ctx;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = type;
    m->subtype = subtype;
    m->uparam = uparam;
    m->vparam = blob;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    ctx = kherr_peek_context();
    if (ctx) {
        if (ctx->flags & KHERR_CF_TRANSITIVE) {
            m->err_ctx = ctx;
        } else {
            kherr_release_context(ctx);
        }
    }

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    kmqint_msg_publish(m, try_send);
```

```
    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_post_message_ex(khm_int32 type, khm_int32 subtype,
                    khm_ui_4 uparam, void * blob, kmq_call * call)
{
    return kmqint_post_message_ex(type, subtype, uparam, blob, call, FALSE);
}

KHMEXP khm_int32 KHMAPI
kmq_abort_call(kmq_call call)
{
    return KHM_ERROR_NOT_IMPLEMENTED;
}

KHMEXP khm_int32 KHMAPI
kmq_post_sub_msg(khm_handle sub, khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * vparam)
{
    return kmq_post_sub_msg_ex(sub, type, subtype, uparam, vparam, NULL);
}

khm_int32
kmqint_post_sub_msg_ex(khm_handle sub, khm_int32 type, khm_int32 subtype,
                       khm_ui_4 uparam, void * vparam,
                       kmq_call * call, khm_boolean try_send)
{
    kmq_message * m;
    kherr_context * ctx;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = type;
    m->subtype = subtype;
    m->uparam = uparam;
    m->vparam = vparam;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    ctx = kherr_peek_context();
    if (ctx) {
        if (ctx->flags & KHERR_CF_TRANSITIVE) {
            m->err_ctx = ctx;
            /* leave it held */
        } else {
            kherr_release_context(ctx);
        }
    }

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    if (try_send)
        EnterCriticalSection(&cs_kmq_types);
    EnterCriticalSection(&cs_kmq_msg);
    kmqint_post((kmq_msg_subscription *) sub, m, try_send);

    if(m->nCompleted + m->nFailed == m->nSent) {
        kmqint_put_message(m);
    }
    LeaveCriticalSection(&cs_kmq_msg);
```

```
    if (try_send)
        LeaveCriticalSection(&cs_kmq_types);

    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_post_sub_msg_ex(khm_handle sub, khm_int32 type, khm_int32 subtype,
                    khm_ui_4 uparam, void * vparam, kmq_call * call)
{
    return kmqint_post_sub_msg_ex(sub, type, subtype,
                                  uparam, vparam, call, FALSE);
}

khm_int32
kmqint_post_subs_msg_ex(khm_handle * subs, khm_size   n_subs, khm_int32 type,
                        khm_int32 subtype, khm_ui_4 uparam, void * vparam,
                        kmq_call * call, khm_boolean try_send)
{
    kmq_message * m;
    kherr_context * ctx;
    khm_size i;

    if(n_subs == 0)
        return KHM_ERROR_SUCCESS;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = type;
    m->subtype = subtype;
    m->uparam = uparam;
    m->vparam = vparam;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    ctx = kherr_peek_context();
    if (ctx) {
        if (ctx->flags & KHERR_CF_TRANSITIVE) {
            m->err_ctx = ctx;
            /* leave it held */
        } else {
            kherr_release_context(ctx);
        }
    }

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    if (try_send)
        EnterCriticalSection(&cs_kmq_types);
    EnterCriticalSection(&cs_kmq_msg);
    for(i=0;i<n_subs;i++) {
        kmqint_post((kmq_msg_subscription *) subs[i], m, try_send);
    }

    if(m->nCompleted + m->nFailed == m->nSent) {
        kmqint_put_message(m);
    }
    LeaveCriticalSection(&cs_kmq_msg);
    if (try_send)
        EnterCriticalSection(&cs_kmq_types);
```

```
    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_post_subs_msg(khm_handle * subs,
                  khm_size   n_subs,
                  khm_int32 type,
                  khm_int32 subtype,
                  khm_ui_4 uparam,
                  void * vparam)
{
    return kmqint_post_subs_msg_ex(subs,
                                   n_subs,
                                   type,
                                   subtype,
                                   uparam,
                                   vparam,
                                   NULL,
                                   FALSE);
}

KHMEXP khm_int32 KHMAPI
kmq_post_subs_msg_ex(khm_handle * subs,
                     khm_int32 n_subs,
                     khm_int32 type,
                     khm_int32 subtype,
                     khm_ui_4 uparam,
                     void * vparam,
                     kmq_call * call)
{
    return kmqint_post_subs_msg_ex(subs, n_subs, type, subtype,
                                   uparam, vparam, call, FALSE);
}

KHMEXP khm_int32 KHMAPI
kmq_send_subs_msg(khm_handle *subs,
                  khm_int32 n_subs,
                  khm_int32 type,
                  khm_int32 subtype,
                  khm_ui_4 uparam,
                  void * vparam)
{
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmqint_post_subs_msg_ex(subs, n_subs, type, subtype,
                                 uparam, vparam, &c, TRUE);
    if(KHM_FAILED(rv))
        return rv;

    rv = kmq_wait(c, INFINITE);
    if(KHM_SUCCEEDED(rv) && c->nFailed > 0)
        rv = KHM_ERROR_PARTIAL;

    kmq_free_call(c);

    return rv;
}

KHMEXP khm_int32 KHMAPI
kmq_send_sub_msg(khm_handle sub, khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * vparam)
{
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmqint_post_sub_msg_ex(sub, type, subtype, uparam, vparam, &c, TRUE);
    if(KHM_FAILED(rv))
        return rv;
```

```
    rv = kmq_wait(c, INFINITE);
    if(KHM_SUCCEEDED(rv) && c->nFailed > 0)
        rv = KHM_ERROR_PARTIAL;

    kmq_free_call(c);

    return rv;
}

/*! Отримуємо ::cs_kmq_global, ::cs_kmq_msg, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_send_thread_quit_message(kmq_thread_id thread, khm_ui_4 uparam) {
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmq_post_thread_quit_message(thread, uparam, &c);
    if(KHM_FAILED(rv))
        return rv;

    rv = kmq_wait(c, INFINITE);

    kmq_free_call(c);

    return rv;
}

/*! Отримуємо ::cs_kmq_global, ::cs_kmq_msg, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_post_thread_quit_message(kmq_thread_id thread,
                             khm_ui_4 uparam, kmq_call * call) {
    kmq_message * m;
    kmq_queue * q;

    EnterCriticalSection(&cs_kmq_global);
    q = queues;
    while(q) {
        if(q->thread == thread)
            break;
        q = LNEXT(q);
    }
    LeaveCriticalSection(&cs_kmq_global);

    if(!q)
        return KHM_ERROR_NOT_FOUND;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = KMSG_SYSTEM;
    m->subtype = KMSG_SYSTEM_EXIT;
    m->uparam = uparam;
    m->vparam = NULL;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    kmqint_post_queue(q, m);
```

```
    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_get_next_response(kmq_call call, void ** resp) {
    return 0;
}

KHMEXP khm_boolean KHMAPI
kmq_has_completed(kmq_call call) {
    khm_boolean completed;

    EnterCriticalSection(&cs_kmq_msg);
    completed = (call->nCompleted + call->nFailed == call->nSent);
    LeaveCriticalSection(&cs_kmq_msg);

    return completed;
}

KHMEXP khm_int32 KHMAPI
kmq_wait(kmq_call call, kmq_timer timeout) {
    kmq_message * m = call;
    DWORD rv;

    if(m && m->wait_o) {
        rv = WaitForSingleObject(m->wait_o, timeout);
        if(rv == WAIT_OBJECT_0)
            return KHM_ERROR_SUCCESS;
        else
            return KHM_ERROR_TIMEOUT;
    } else
        return KHM_ERROR_INVALID_PARAM;
}

/*! \Отримуємо ::cs_kmq_types
    */
KHMEXP khm_int32 KHMAPI
kmq_set_completion_handler(khm_int32 type,
                           kmq_msg_completion_handler handler) {
    return kmqint_msg_type_set_handler(type, handler);
}

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

void __fastcall TForm4::Button7Click(TObject *Sender)
{
//відкриття вікна "про програму..."
Form2->Show();
}
//---------------------------------------------------------------------------
```

Файл server.h – бібліотека для файлу server.cpp

```
//---------------------------------------------------------------------------

#ifndef serverH
#define serverH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm4 : public TForm
{
__published:     // IDE-managed Components
        TTreeView *TreeView1;
        TLabel *Label1;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TTreeView *TreeView2;
        TLabel *Label2;
        TButton *Button5;
        TButton *Button6;
        TImage *Image1;
        TImage *Image3;
        TButton *Button7;
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
private:         // User declarations
public:          // User declarations
        __fastcall TForm4(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm4 *Form4;
//---------------------------------------------------------------------------
#endif
```

Файл about.cpp – вікно "Про програму..."

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
Form2->Close();
}
//---------------------------------------------------------------------------
```

Файл about.h – бібліотека для файлу about.cpp

```
//---------------------------------------------------------------------------

#ifndef aboutH
#define aboutH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:      // IDE-managed Components
        TImage *Image1;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:     // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```