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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Перепелиця А.В.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.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 *125 “*Кібербезпека*”*

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

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

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

Перепелиці Антону Васильовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN*

2. Керівник роботи *Дрєєва Ганна Миколаївна, доктор філософії (PhD)*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN*

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

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

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Дрєєва Г.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Перепелиця А.В.
(прізвище та ініціали)

АНОТАЦІЯ

Перепелиця А.В. Програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

Метою розробки є програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

Результат роботи – програмна реалізація системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

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 Python.

Ключові слова: кібербезпека, Cloud-система, VPN

ABSTRACT

Perepelytsia A.V. Software for a cybersecurity system for protecting information in Cloud systems using VPN. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a cybersecurity system for protecting information in Cloud systems using VPN.

The purpose of the development is to develop software for a cybersecurity system for protecting information in Cloud systems using VPN.

The result of the work is a software implementation of a cybersecurity system for protecting information in Cloud systems using VPN.

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 is developed in Python.

Keywords: cybersecurity, Cloud system, VPN

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AH	–	автентифікуючий заголовок
CA	–	сертифікаційне співтовариство
DES	–	Data Encription Standard
DoS	–	атака "Відмова в обслуговуванні"
DOI	–	область інтерпретації
ESP	–	Інкапсуляція зашифрованих даних
HTTPS	–	зашифрований http
IAB	–	координаційна рада мережі Internet
IDS	–	система, яка автоматизує процес перегляду подій
IETF	–	проблемна група проектування Internet
IKE	–	протокол обміну ключами за замовчуванням для ISAKMP
IKMP	–	протоколу керування ключами прикладного рівня
IPsec	–	комплект протоколів захисту інформації по IP
ISAKMP	–	механізми узгодження атрибутів використовуваних протоколів
ISP	–	постачальник послуг Internet
MAC	–	коди на перевірку цілісності
MD5	–	дайджест повідомлення
Oakley	–	сесійні ключі на комп'ютери мережі Інтернет
PFS	–	ідеальна пряма безпека
PRF	–	псевдовипадкова функція
SA	–	Security Association
SKIP	–	команда підготовки наступної команди
SPI	–	індекс параметрів безпеки
SPD	–	база даних політики безпеки
SSL	–	протокол захищених сокетів
TCP	–	транспортний протокол
VPN	–	віртуальні приватні мережі

ВСТУП

Актуальність теми. Згідно з оптимістичними прогнозами, уже через кілька років багато вітчизняних організацій будуть здобувати послуги з моделі «IT як сервіс», однак поки на хмарні сервіси доводиться лише мала частка українського ринку IT. Як показують опитування, в усьому світі одним з головних перешкод для їхнього широкого використання залишаються проблеми, пов'язані із забезпеченням інформаційної безпеки: дотепер більшість керівників великих компаній упевнені в недостатній захищеності хмарних сервісів.

Однак, згідно даним торішнього дослідження Symantec про поширеність хмарних технологій серед українських підприємств малого й середнього бізнесу, 81% співробітників уже так чи інакше використовує хмари в робочих цілях. Така ситуація змушує задуматися про прийняття певних правил, спрямованих на захист корпоративної інформації. Багато хто із цих співробітників у рамках своєї професійної діяльності звертаються до популярних безкоштовних хмарних сервісів, а деякі – до комерційних хмарним бізнес-додаткам. Зокрема, 44% опитаних використовують хмарні сховища файлів, а 41% – розміщені в хмарі додатки.

Серед головних причин переходу на хмарні рішення називаються зниження витрат по експлуатації, підвищення мобільності персоналу, поліпшення захисту даних (резервне копіювання/відновлення), гнучкість і зручність роботи. У той же час респонденти виражають занепокоєння щодо захищеності даних, їхньої недоторканності, а також відсутності контролю. Лише чверть опитаних упевнені в безпеці інформації, розміщеної в хмарі.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

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

– Огляд існуючих систем для захисту інформації у Cloud-системах з використанням VPN.

– Дослідження системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

– Програмна реалізація системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для захисту інформації у Cloud-системах з використанням VPN.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN, 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

Інформаційна безпека – це насамперед процес керування ризиками. Абсолютна захищеність неможлива в принципі, тому потрібно оцінити, наскільки тих або інших факторів ризику, у тому числі пов'язані із хмарними сервісами, впливають на життєдіяльність конкретної компанії. У цілому ряді випадків переваги хмарної моделі значно переважують пов'язані з нею ризики, у тому числі й погрози безпеки. Так, наприклад, завдяки хмарам стартап-компанії можуть динамічно розвивати свій бізнес, уникати великих витрат на створення власної інфраструктури IT і швидко реагувати на нові вимоги ринку.

Сам по собі переклад сервісів і додатків вторинний. Головне – можливість ефективного використання сервісів і додатків у процесі роботи. Оскільки хмарна модель дуже перспективна, насамперед з економічної точки зору, породжувані нею проблеми безпеки будуть в остаточному підсумку дозволені. У будь-якій компанії є політика безпеки, що при переході до хмарних сервісів відповідним чином видозмінюється. Своя політика є й у провайдеру, і нею варто поцікавитися. Якщо адаптувати до хмарної моделі всі існуючі правила, то не буде причин боятися чогось невідомого.

Завжди виникає питання: про які саме ризики мова йде? Ще шість-сім років тому основна увага фокусувалась на забезпеченні конфіденційності даних, на початку року багато говорилося про їхню доступність, а тепер знову широко обговорюється конфіденційність. Ми переходимо від парадигми серверів до парадигми хмар, однак багато хто навіть не можуть точно сформулювати, що саме викликає їхнього побоювання при переході до хмарного середовища.

Схожі проблеми забезпечення інформаційної безпеки виникають при впровадженні моделі BYOD і підтримці мобільної роботи співробітників.

Периметр корпоративної мережі зараз розсунувся аж до смартфонів і планшетів, але лише деякі компанії подбали про те, щоб на особистих мобільних пристроях були встановлені засоби безпеки, тим більше сертифіковані. Причому вони не ускладнюють роботу користувачів з корпоративними ресурсами. За прогнозом VMware, у даному році близько 23% українських співробітників стануть повністю «мобільними». Протягом наступних п'яти років цей показник досягне 40%, а в більше довгостроковій перспективі – 56%. У конфлікті бізнесу й безпеки перемагає те, що сприяє досягненню більшої ефективності. Це твердження справедливо й відносно хмарної моделі в цілому.

Навіть якщо вимоги замовників відносно безпеки, надійності й доступності хмар виконуються не повною мірою, у хмарного сервісу однаково будуть клієнти, оскільки він дозволяє домагатися бажаних результатів. Наприклад, навіть найбільші компанії, такі як HP і Cisco, ведуть облік своїх комерційних справ за допомогою сервісу Salesforce.com. Саме завдяки підвищенню ефективності ведення бізнесу й досить високому рівню безпеки (включаючи засобу автентифікації) цьому хмарному провайдеру вдалося залучити настільки великі компанії, які до того ж конкурують між собою на ринку. Таким чином, усі визначають потреби бізнесу.

1.2 Область застосування

Сьогодні технологія VPN (Virtual Private Network – віртуальна приватна мережа) завоювала загальне визнання й будь-якого адміністратора вважає своїм обов'язком організувати VPN-Канали для співробітників, що працюють поза офісом являє собою об'єднання окремих машин або локальних мереж у віртуальній мережі, що забезпечує цілісність і безпеку переданих даних. Вона має властивості виділеної приватної мережі й дозволяє передавати дані між двома комп'ютерами через проміжну мережу (internetwork), наприклад Internet.

VPN відрізняється рядом економічних переваг у порівнянні з іншими методами віддаленого доступу. По-перше, користувачі можуть звертатися до корпоративної мережі, не встановлюючи с її з'єднанням, що комутирується, таким чином, відпадає потреба у використанні модемів. По-друге, можна обійтися без виділених ліній.

Маючи доступ в Інтернет, будь-який користувач може без проблем підключитися до мережі офісу своєї фірми. Варто помітити, що загальнодоступність даних зовсім не означає їхня незахищеність. Система безпеки VPN – це броня, що захищає всю корпоративну інформацію від несанкціонованого доступу. Насамперед, інформація передається в зашифрованому виді. Прочитати отримані дані може лише власник ключа до шифру. Найбільше часто використовуваним алгоритмом кодування є Triple DES, що забезпечує потрійне шифрування (168 розрядів) з використанням трьох різних ключів. Підтвердження дійсності містить у собі перевірку цілісності даних і ідентифікацію користувачів, задіяних в VPN. Перша гарантує, що дані дійшли до адресата саме в тім виді, у якому минулому послані. Самі популярні алгоритми перевірки цілісності даних – MD5 і SHA1. Далі система перевіряє, чи не були змінені дані під час рухи по мережах, помилково або зловмисно. Таким чином, побудова VPN припускає створення захищених від стороннього доступу тунелів між декількома локальними мережами або віддаленими користувачами.

Для побудови VPN необхідно мати на обох кінцях лінії зв'язку програми шифрування вихідного й дешифрування вхідного трафіків. Вони можуть працювати як на спеціалізованих апаратних пристроях, так і на ПК із такими операційними системами як Windows, Linux або NetWare. Керування доступом, автентифікація й шифрування – найважливіші елементи захищеного з'єднання.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN, 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

CyberGhost

Першим у нашім огляді фігурує німецький сервіс CyberGhost, в активі якого представлені 349 розподілених VPN-серверів в 24 країнах світу й клієнтські додатки для платформ Windows, Mac OS X, iOS, Android, що забезпечують швидке налаштування віртуальної приватної мережі. Рішення підтримує різні протоколи для побудови VPN-тунелю, використовує алгоритм шифрування AES з 256-бітним ключем і оснащено системою Anti Fingerprinting, що запобігає збір різної інформації про комп'ютер користувача й забезпечує тим самим ще більшу анонімність в Інтернеті. Що стосується тарифних планів, те їх в CyberGhost усього три – Free, Premium ($7/місяці) і Premium Plus ($11/місяць). Безкоштовним передплатникам сервіс пропонує обмежений набір VPN-площадок з автоматичним розривом з'єднання кожні 5 годин. Крім того, користувачі тарифного плану Free не мають змоги використовувати мобільний iOS-клієнт, змушені миритися з рекламними баннерами й нетривалими паузами при підключенні до VPN-серверів. Оплатити послуги сервісу CyberGhost можна за допомогою різних платіжних систем, у тому числі віртуальною валютою Bitcoin.

Інтерфейс клієнтського додатка CyberGhost лаконічний і простий у використанні. Щоб приступитися до роботи, досить вибрати підходящий VPN-сервер і активувати з'єднання клацанням миші по круглій жовтій клавіші. Для зручності користувачів передбачена вибірка по IP-адресах, а також карта, що наочно відображає прив'язку конкретного шлюзу до місцевості. На випадок утруднень припасена значних розмірів веб-документація, що докладно роз'ясняє

будь-які питання технічного характеру аж до підключення до мережі CyberGhost комп'ютерів під керуванням Linux і Chrome OS, роутерів і іншої обчислювальної техніки з убудованою підтримкою VPN.

Рисунок 2.1 – Інтерфейс клієнтського додатка CyberGhost

Spotflux

Американський сервіс, доступний для платформ Windows, Mac OS X, iOS, Android і – крім організації VPN-Каналу – ряд, що пропонує, послуг із забезпечення безпеки при роботі в Мережі. Spotflux уміє в автоматичному режимі перевіряти завантажувані з Інтернету файли на предмет наявності в них шкідливого коду, різати баннери на сторінках, що переглядаються в браузері, блокувати фішінгові ресурси, стискати передані через шлюз дані для економії трафіку й робити веб-серфінг максимально анонімним. Про кількіість VPN-серверів і їх географічної розподіленості нічого на сайті сервісу не говориться,

зате є інформація про інші тарифні плани – Free, Mobile Only ($6/рік) і Premium ($30/рік). Безкоштовний аккаунт істотно урізаний по частині пропонованих можливостей, припускає захист тільки веб-трафіку й обмеження швидкості передачі даних.

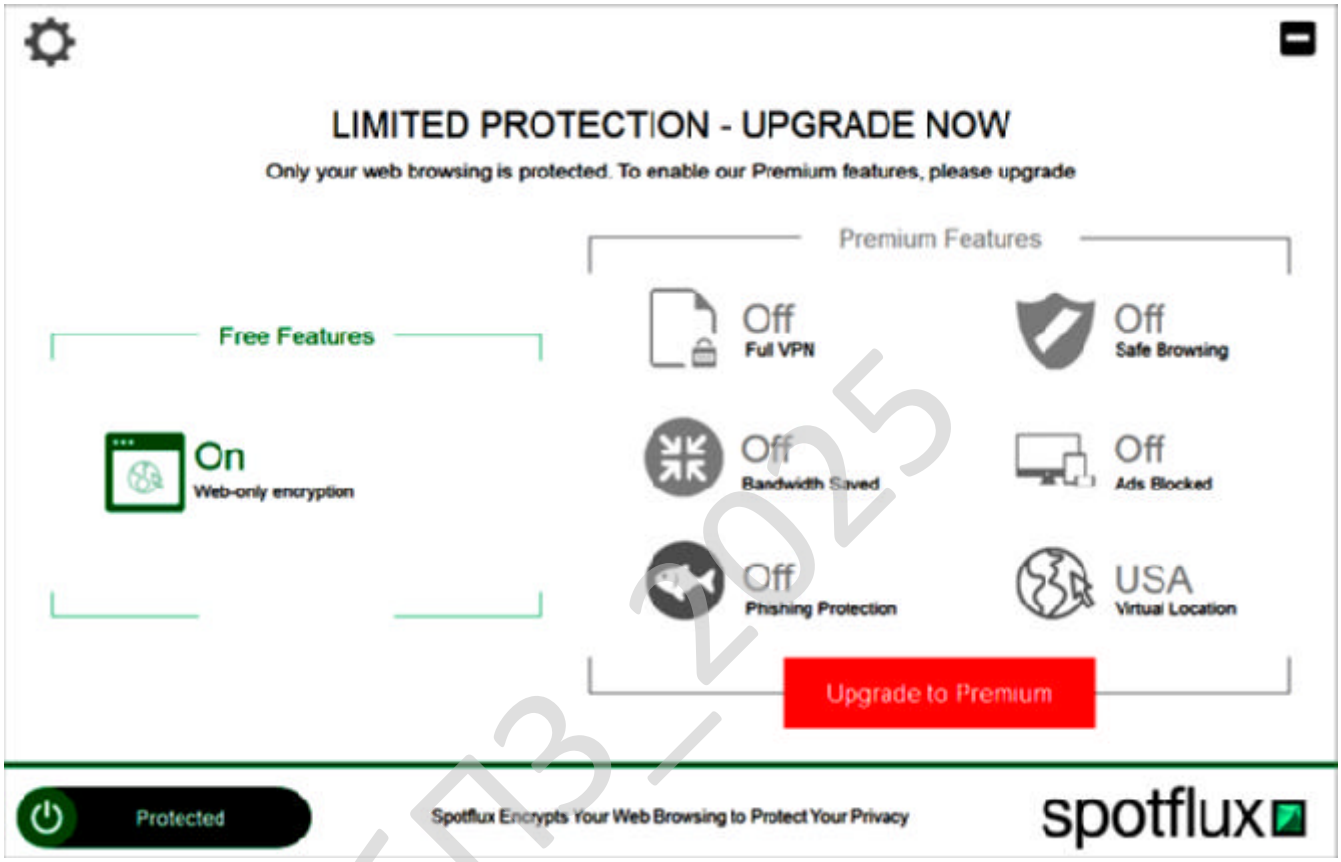

Рисунок 2.2 – Інтерфейс клієнтського додатка Spotflux

Яких-небудь премудростей в інтерфейсі клієнтського додатка Spotflux шукати не доводитися, особливо користувачам free-передплати на сервіс, яким доступна тільки клавіша активації VPN так ряд немудрих налаштувань. Для доступу до всіх елементів керування необхідний Premium-аккаунт, що дозволяє вибирати сервери VPN для підключення, регулювати рівень захисту та інші параметри програми. З метою оцінки всіх можливостей Spotflux розроблювачами надається безкоштовний триденний тест-драйв повнофункціональної версії сервісу.

Private Tunnel

Сервіс американської компанії OpenVPN Technologies, широко відомої своїми розробками в сфері захисту даних. До послуг користувачів Private Tunnel доступні вісім VPN-серверів у шести країнах і набір клієнтських додатків для Windows, Mac OS X, iOS і Android. Всі гранично просто. Лінійка тарифних планів така ж нехитра – вона складається з пакетів з різними обсягами включеного в них трафіку. Для безкоштовного аккаунту ліміт трафіку становить скромні 100 Мбайт, для оплачених облікових записів обсяг трансльованих через VPN-шлюз даних може становити 50 Гбайт ($12), 100 Гбайт ($20) або 500 Гбайт ($50). При цьому до одному аккаунту можна прив'язати скільки завгодно багато пристроїв, а придбані пакети є безстроковими, тобто не згоряють із часом.

Рисунок 2.3 – Інтерфейс клієнтського додатка Private Tunnel

Зі списку доступних для завантажування клієнтських додатків Private Tunnel окремого згадування заслуговують версії для Windows і Mac OS X, написані на Java і, як наслідок, що вимагають для своєї роботи встановлену на комп'ютері виконавче середовище Java Runtime Environment (JRE). Наявність останньої істотно знижує захищеність системи через значне число «дір» і проломів в Java, рік у рік утримуючий титул платформи з найбільшою кількістю уразливостей. Кількість атак на Java останнім часом перевищує всі мислимі межі. Тому при використанні Private Tunnel потрібно бути гранично уважним – щоб уникнути проблем з інформаційною безпекою варто регулярно обновляти встановлену на комп'ютері платформу Java.

Your Freedom

Другий у нашім огляді VPN-сервіс німецьких розроблювачів, «ахілесовою п'ятої» якого також є популярна серед зловмисників платформа Java, відповідальна за роботу клієнтських додатків для Windows і Mac OS X. Your Freedom пропонує на вибір кілька десятків VPN-серверів у різних країнах світу, підтримує різні протоколи для побудови захищеної мережі, дозволяє настроїти маршрутизацію будь-якого типу трафіку й має у своєму составі клієнтську програму для мобільної ОС Android. Робота із сервісом можлива в рамках чотирьох тарифних планів – Free, Basic (€4/місяць), Enhanced (€10/місяць) і Total (€20/місяць), що відрізняються швидкістю передачі даних і кількістю одночасно підтримуваних з'єднань. Безкоштовний варіант Your Freedom урізаний донезмоги й практично не прийнятний для веб-серфінгу: швидкість обмежена до модемних 64 кбіт/с, а тривалість VPN-З'єднання – двома годинниками в день. При випадкових обривах зв'язку щораз доводиться проходити CAPTCHA-тест і доводити свою приналежність до людської раси.

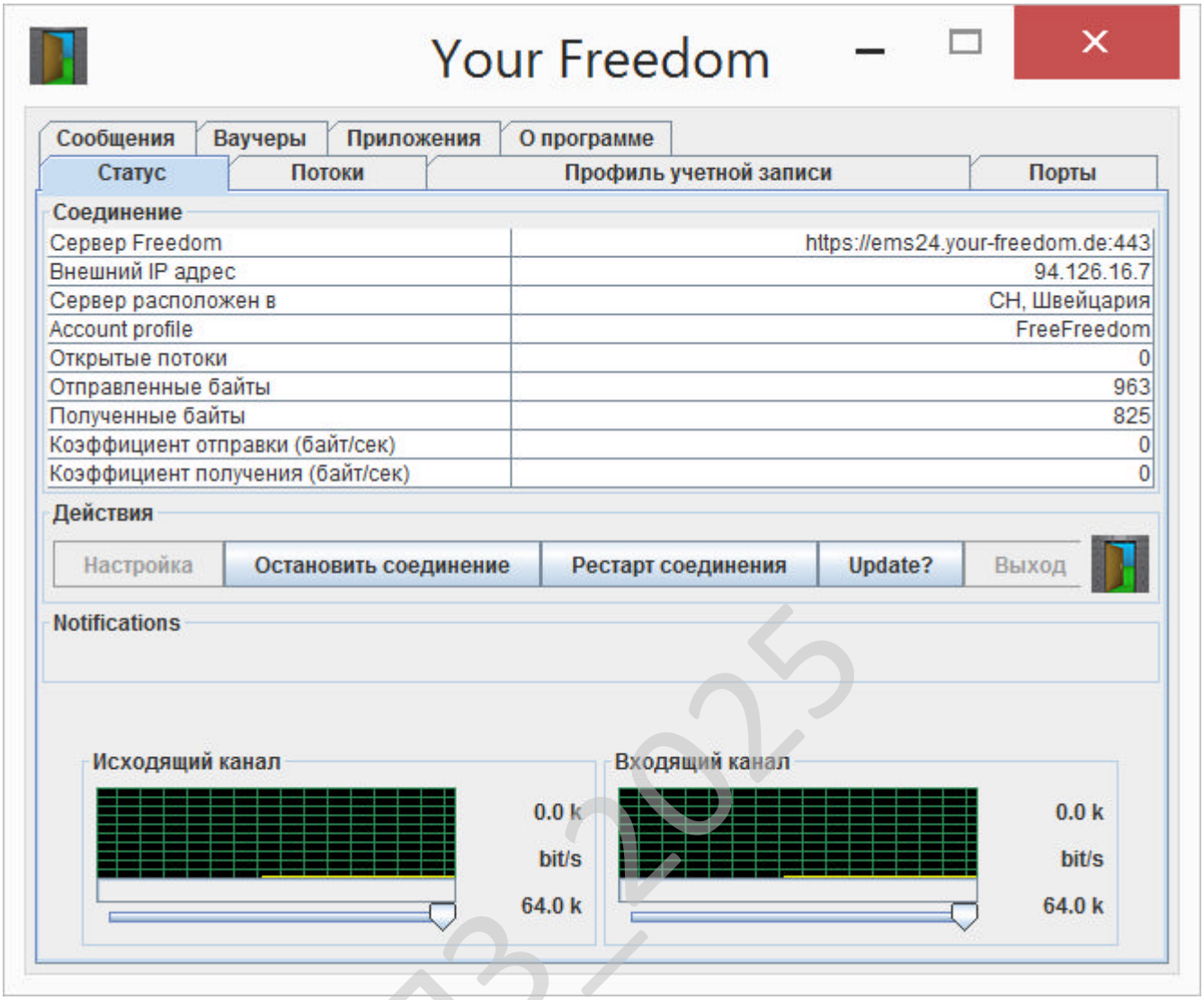

Рисунок 2.4 – Інтерфейс клієнтського додатка Your Freedom

Інтерфейс клієнтських додатків Your Freedom буквально кишить численними налаштуваннями, здатними віджахнути малодосвідченого користувача. Положення почасти рятують грамотно проведена русифікація й майстер налаштування, що поетапно роз'ясняє ключові моменти роботи із програмою.

proXPN

Сервіс нідерландської компанії, що підтримує Windows, Mac OS X, iOS і Android і, що має у своєму активі розподілену мережу VPN-серверів, розташованих у США, Лондоні, Амстердамі й Сінгапурі. Значну географію

proXPN доповнюють потужні засоби шифрування й розвинені функціональні можливості сервісу, повною мірою скористатися якими можна при наявності Premium-аккаунту ($6,3/місяць). Для аматорів заощаджувати передбачена безліч обмежень у вигляді швидкості передачі даних (до 300 кбіт/с), одного американського VPN-шлюзу, типу трафіку (тільки веб) і можливості використання сервісу винятково на комп'ютері. Крім того, у безкоштовних аккаунтах proXPN нарочито підсуває користувачеві рекламні баннери, у тому числі при завантаженні файлів і перегляді сайтів. Миритися з таким поводженням сервісу можна, але нерви потрібно мати міцні.

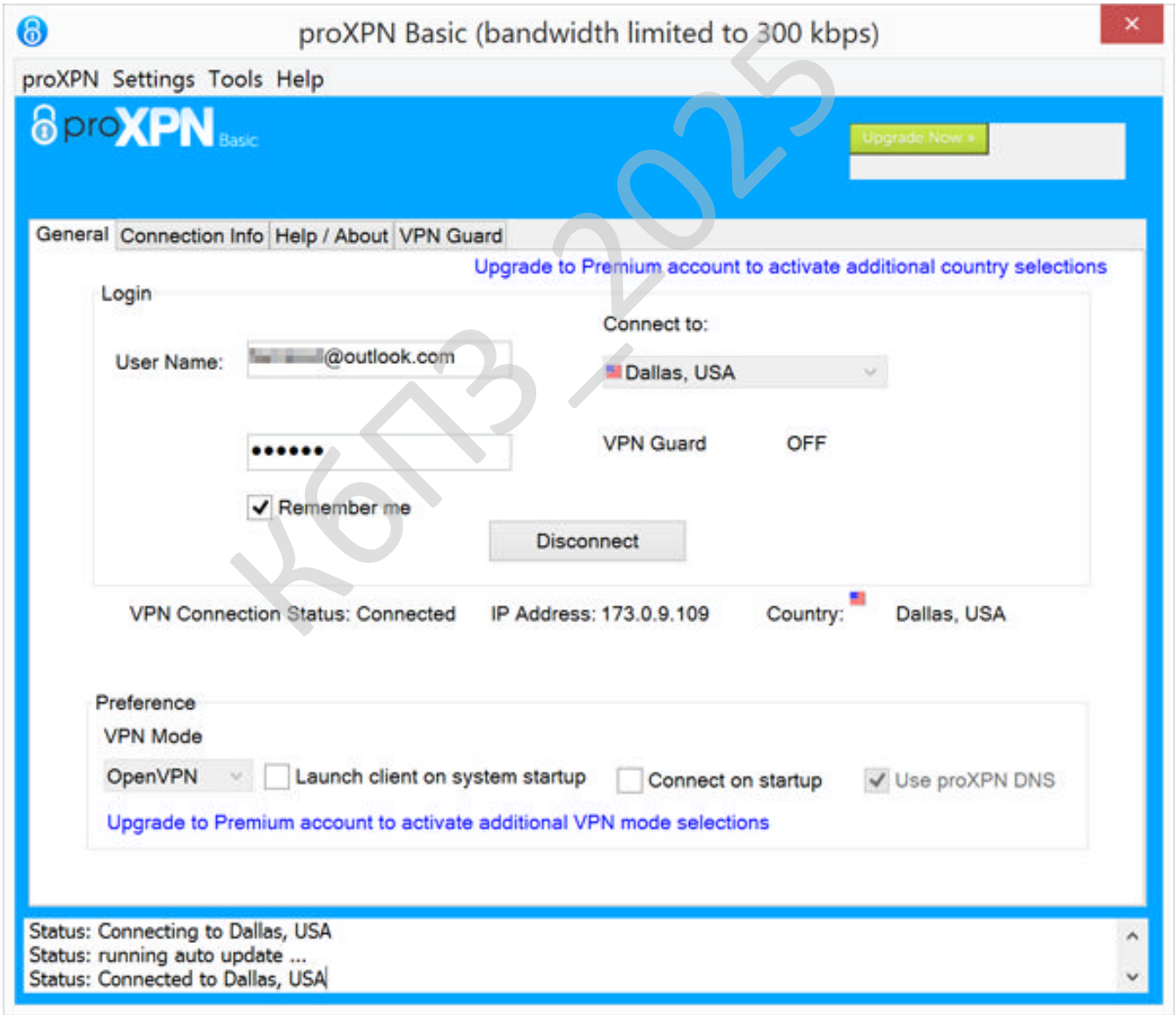

Рисунок 2.5 – Інтерфейс клієнтського додатка proXPN

Не менш дратівливий proXPN і на етапі реєстрації користувальницького аккаунту, для активації якого кров з носу необхідний номер мобільного телефону. Щоб зазрячи не «світити» контактні дані, можна скористатися безкоштовними сервісами receive-sms-online.com і onlinesim.ru, що надають можливість прийому SMS через Інтернет у режимі онлайн. Загалом, сервіс на аматора.

Hotspot Shield

Ще один «американець», заточений для роботи із платформами Windows, Mac OS X, iOS, Android і, крім серверів на території США, що має VPN-площадки у Великобританії, Австралії, Японії, Канаді й Німеччині. Крім організації захищеного мережного з'єднання, Hotspot Shield може сканувати трафік на предмет шкідливого коду й стискати передані через VPN-шлюз дані в мобільних пристроях. Вартість послуг цілком демократична – 2,5 долари на місяць. Без оформлення підписки Hotspot Shield «радує» користувача баннерами, які нерідко виводяться поверх перегляда_ користувачем веб-ресурсів.

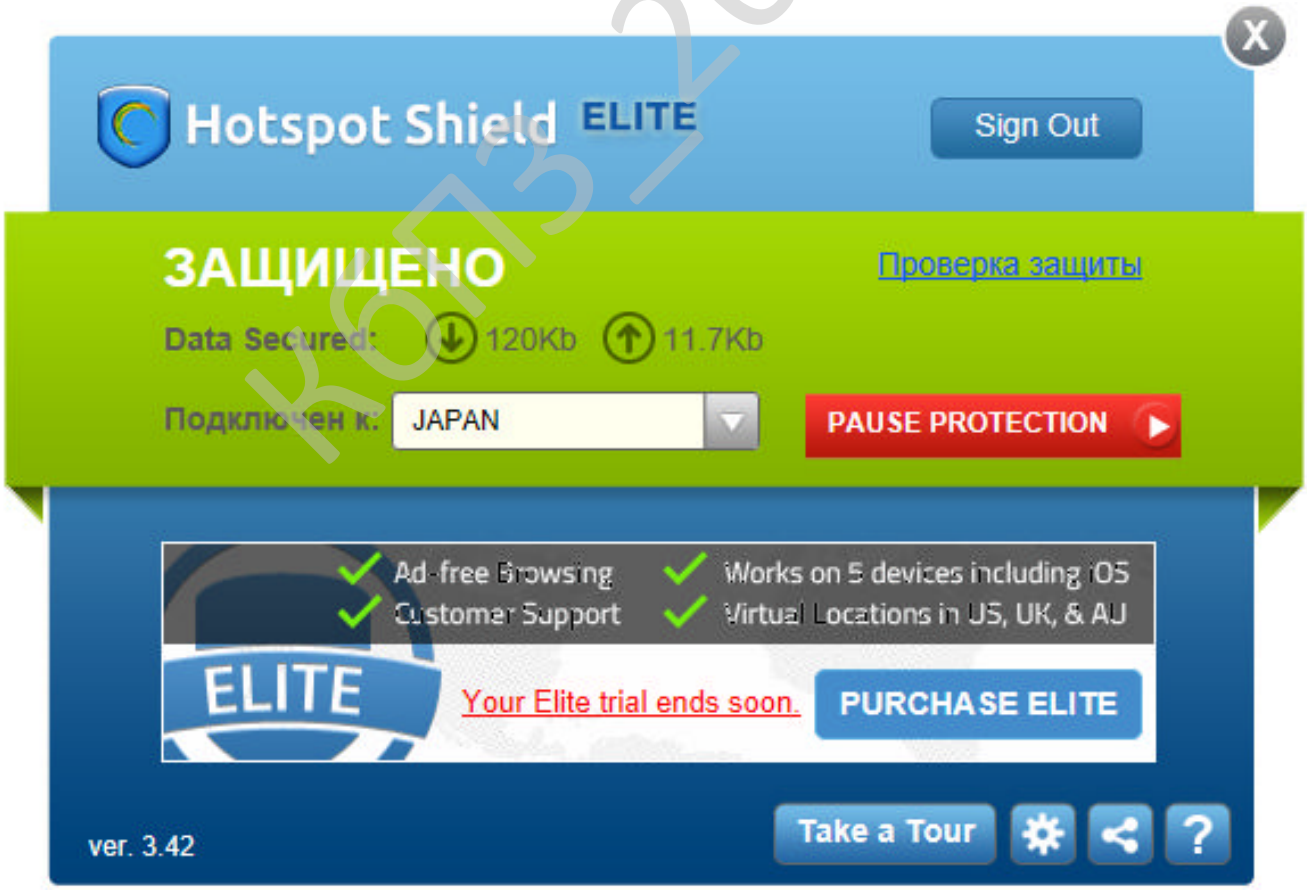

Рисунок 2.6 – Інтерфейс клієнтського додатка Hotspot Shield

TunnelBear

Канадський інструмент для аматорів анонімного веб-серфінгу, представлений у версіях для операційних систем Windows, Mac OS X, iOS і Android і дозволяючий користувачеві вибирати одне з восьми місць свого віртуального місцезнаходження.

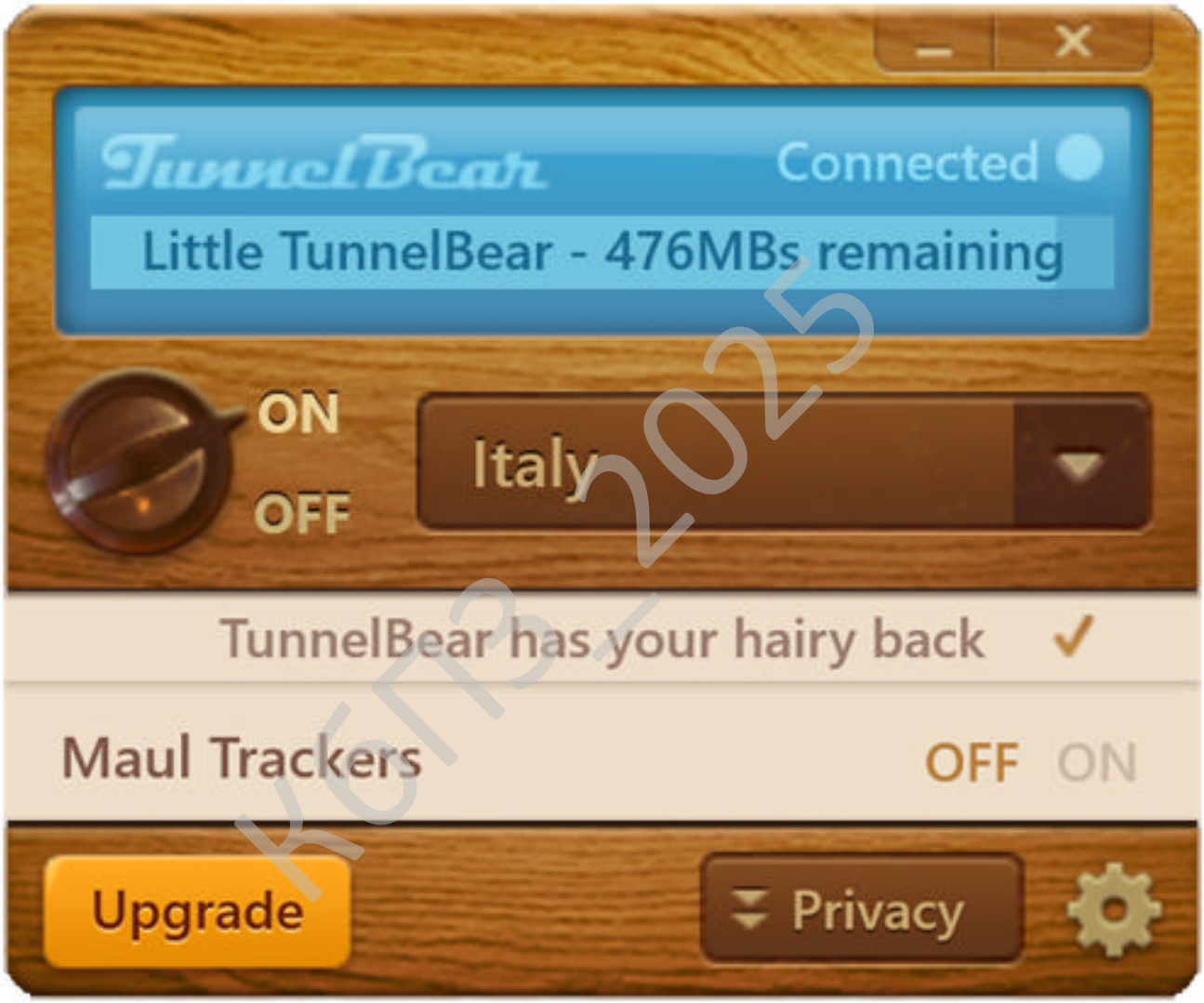

Рисунок 2.7 – Інтерфейс клієнтського додатка TunnelBear

Особливістю TunnelBear є оригінальний інтерфейс клієнтських додатків, що запам'ятовується особливістю якого є архаїчний тумблер ON/OFF, що перемикає режим роботи програми. Для захисту інформації використовується

протокол OpenVPN з 128-бітним шифруванням Blowfish, для забезпечення користувальницької конфіденційності – фірмова система Maul Trackers (блокує дані, які могли б ідентифікувати власника комп'ютера або смартфону), а для більшої безпеки задіяна функція IntelliBear (дозволяє задати список ресурсів, звертання до яких буде вироблятися тільки через VPN). Як і всі розглянуті в огляді сервіси, TunnelBear охочий до дзвінких монет, але при цьому абсолютно толерантний до ощадливої аудиторії, не дратує баннерами й напрочуд швидко працює навіть при використанні безкоштовних аккаунтів. Free-версія сервісу має ліміт по трафіку в розмірі 500 Мбайт на місяць, позбавлена механізму IntelliBear і підтримки мобільних пристроїв. Зняття всіх обмежень обійдеться в 5 доларів щомісяця або в $50 за рік.

USAIP

Єдиний у нашім огляді сервіс, що має в арсеналі клієнтський додаток тільки для Windows. Для інших платформ – Mac OS X, iOS, Android, Linux – на сайті USAIP представлена довідкова документація, у подробицях пояснює процес ручного підключення VPN за допомогою протоколів PPTP і L2TP, підтримуваних багатьма сучасними системами й роутерами. В активі сервісу: 35 VPN-серверів в 20 країнах, включаючи Росію, універсальність, відсутність обмежень по обсязі й типу трафіку (деякі VPN-сервіси обмежують доступ до відеоконтенту й торрентам). Сильною стороною USAIP навіть є гнучка тарифна політика, що допускає можливість одноденної оплати послуг сервісу ($1,5), що є відмінним варіантом для тих, кому необхідний разовий доступ до захищеної мережі. Тижневе користування USAIP коштує $4, місячне – $8, а річне – 75 доларів США. У безкоштовному варіанті сервіс часто рве з'єднання й обмежує швидкість передачі даних до 200 кбіт/с.

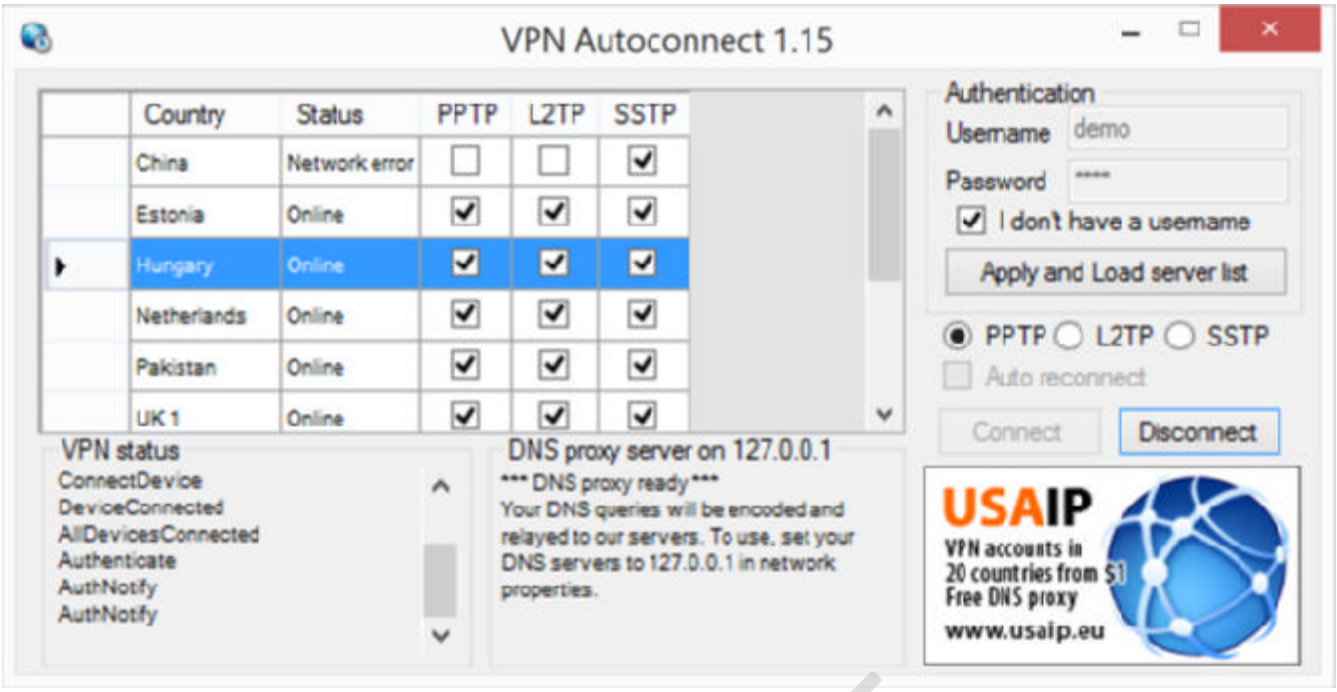

Рисунок 2.8 – Інтерфейс клієнтського додатка USAIP

VPN Gate

Відритий проект, розроблений співробітниками японського університету Цукуби й об'єднуючий понад 7 тисяч публічних VPN-площадок, розгорнутих добровольцями по усьому світі. VPN Gate характеризується широкою підтримкою всіляких програмних платформ, VPN-протоколів і способів розгортання віртуальних приватних мереж, наявністю різноманітних клієнтських додатків і величезною армією волонтерів, що приймають активну участь у розвитку не знаючих територіальних границь сервісу. Щоб почати роботу з ним, не потрібно реєструватися: досить вибрати зі списку менш завантажений сервер і встановити з'єднання. Учасники проекту переконані, що у вільному світі без границь Інтернет повинен бути вільним і відкритим. Хто дотримується аналогічної думки, може завантажити спеціальне програмне забезпечення й надати мережної аудиторії свій власний комп'ютер як сервер VPN. Відповідні інструкції можна знайти на сайті VPN Gate.

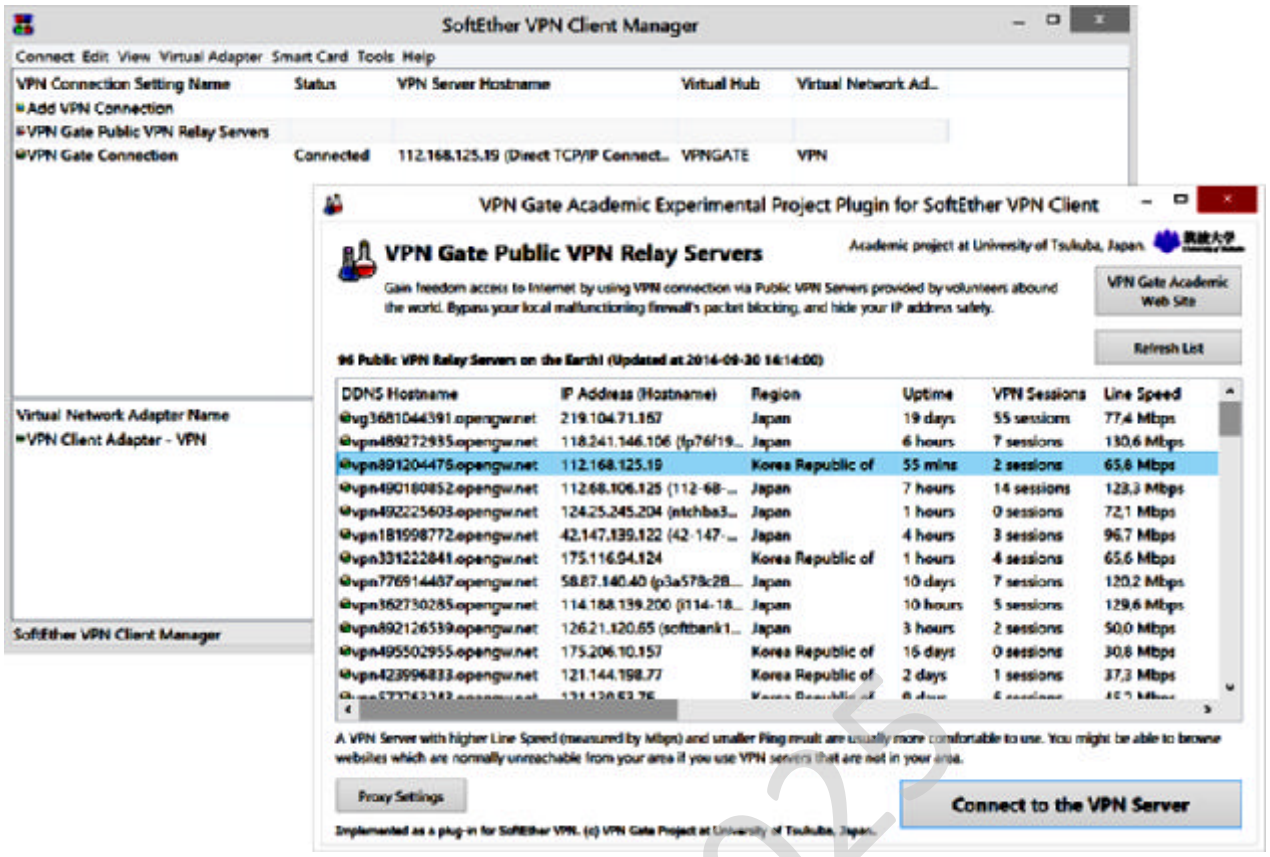

Рисунок 2.9 – Інтерфейс клієнтського додатка VPN Gate

SecurityKISS

Останній у нашім огляді – ірландський сервіс SecurityKISS, що пропонує до послуг користувальницької аудиторії 46 VPN-серверів і можливість оперативного налаштування захищеного мережного оточення за допомогою клієнтських додатків для Windows і Mac OS X (для інших ОС можлива ручне налаштування за допомогою протоколів OpenVPN, PPTP, L2TP). Рішення підтримує шифрування з 1024-бітним ключем, не вимагає створення облікового запису й оснащено фірмовою функцією Exclusive Tunneling, що забезпечує роботу VPN при нестабільному мережному з'єднанні. На вибір розроблювачами SecurityKISS пропонується аж п'ять тарифних планів, один із яких безкоштовний, має ліміт добового трафіку (300 Мбайт) та інші обмеження. Комерційні пропозиції варіюються від 3 євро на місяць до 90 євро в рік. При оплаті електронною валютою Bitcoin можна одержати 50-процентну знижку.

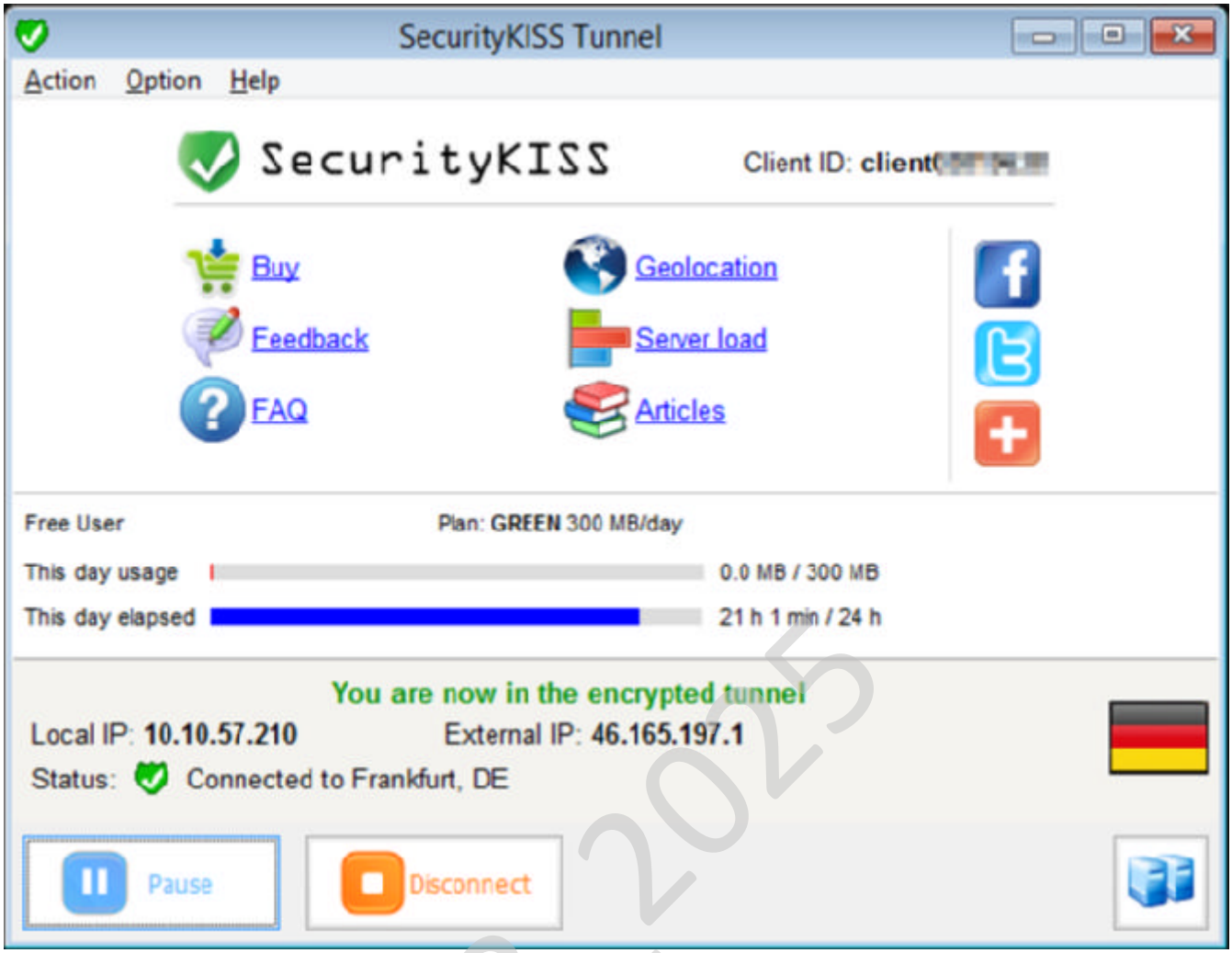

Рисунок 2.10 – Інтерфейс клієнтського додатка SecurityKISS

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за

інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У сегменті SMB хмарна модель буде активно розвиватися просто тому, що більшість малих і середніх компаній не хочуть витрачати зусилля на рішення численних завдань, пов'язаних з IT, оскільки ті мають лише непряме відношення до їхньої діяльності. Уже зараз це можна бачити на прикладі систем електронної пошти: користуватися сервісом, що пропонує провайдер, набагато зручніше й надійніше, ніж власними поштовими серверами.

Однак у кожному разі хмарні сервіси – це численні ризики, які не можна не враховувати. До них ставляться втрата замовником власних компетенцій і залежність від зовнішнього постачальника послуг, відсутність контролю над діями віддалених адміністраторів і потоками інформації за межами корпоративної мережі, невідповідність рівня ІБ на площадці провайдеру корпоративній політиці й вимогам регуляторів.

Компаніям (особливо великим) варто ретельно зважувати ризики, ураховувати надійність хмар і оцінювати потенційні збитки від можливих простоїв. Зовсім необов'язково виносити в хмари відразу геть усе. Спочатку можна обмежитися певною часткою інформації й бізнес-процесів. Нерідко класифікація даних показує, що більшу їхню частину можна безбоязно розмістити в хмарі, де рівень ІБ може бути помітно вище, ніж у корпоративному середовищі. Справа в тому, що в хмарі дані «знеособлені», а зламати добре захищений комерційний ЦОД значно складніше й дорожче, ніж проникнути в корпоративну мережу.

Безпека хмар забезпечується насамперед за рахунок керування ризиками й відповідності вимогам законодавства. Ми надаємо обчислювальні потужності в середовищі VMware і захищені канали VPN. У завершальній стадії перебуває

проект створення сегмента хмарного середовища, що відповідає вимогам по зберіганню персональних даних. При керуванні ризиками ІБ ми керуємося здоровим глуздом і тими ж практиками, які вже зарекомендували себе при наданні традиційних сервісів ЦОД. Забезпечення безпеки хмарних додатків – у першу чергу завдання власника додатків. Як правило, їм є клієнт комерційного ЦОД. Ми, як провайдер хмарної обчислювальної інфраструктури, забезпечуємо безпеку на своєму рівні (ізоляція мереж, протоколювання дій адміністраторів, резервне копіювання й т.д.), але до самих додатків доступу звичайно не маємо. Якщо останній використовує платформу IaaS, то він в основному й відповідає за її безпеку, а у випадку SaaS безпека повинен свідомо гарантувати провайдер сервісу.

Сьогодні хмарні провайдери охоче обговорюють із потенційними клієнтами питання ІБ і, більше того, самі піднімають цю тему, намагаючись аргументовано переконати замовників у тім, що рівень безпеки в хмарі істотно вище, ніж у корпоративному ЦОД, де застосовуються й постійно обновляються найсучасніші захисні засоби. До того ж, наприклад, витік цілого ряду категорій даних, збережених у хмарі, не нанесе серйозного збитку бізнесу компанії, а в порівнянні з ризиками втрат з вини инсайдера її ймовірність набагато нижче.

Атаки хакерів є причиною лише декількох відсотків інцидентів, пов'язаних з порушенням ІБ. Набагато частіше такі проблеми виникають внаслідок «людського фактора. Якщо підрахувати, скільки співробітників мають доступ до корпоративної інформації, то виявиться, що й варіантів витоків дуже багато. Крім великих компаній, лише відносно невелике число організацій мають власну політику ІБ. Крім того, її реалізація на практиці залишає бажати кращого, а внутрішня політика безпеки й захисту даних не вибудувана належним чином – у такому випадку перехід у хмару не збільшить, а тільки поліпшить ситуацію. На відміну від системного адміністратора, хмарний провайдер несе хоч якусь фінансову відповідальність, і в контракті з ним можна зафіксувати ті або інші умови, що стосуються ІБ.

Переважна більшість проблем викликана діями співробітників компаній при їхній роботі із внутрішніми інформаційними ресурсами. Це прямі втрати. «Рівень безпеки й надійності у великих ЦОД свідомо вище, ніж в SMB. Центри обробки даних у стані підтримувати складні й дорогі системи ІБ і сертифікувати їх. Нарешті, між провайдером і клієнтом підписується «угода про нерозголошення» (Non-Disclosure Agreement, NDA), де деталізується, зокрема, як і який доступ до даних клієнта провайдер може одержати в тій або іншій ситуації, як цей факт реєструється, які застосовуються штрафні санкції у випадку порушень. Практика такої відповідальності провайдеру, безсумнівно, буде поступово напрацьовуватися.

Сконцентрувати дані в хмарі й організувати їхній захист – виходить, одержати більше надійне, безпечне й контрольоване середовище в порівнянні з офісної, коли вони розподілені по різним ПК і філіях. Разом з тим існують і певні вимоги українського законодавства. Замовники користуються хмарами заради підвищення ефективності бізнесу, але це може суперечити нормативним вимогам.

На жаль, забезпечити в хмарі захист інформації, доступ до якої обмежений (наприклад, даних держустанов) відповідно до законодавства, поки непросто.

По-перше, потрібний довірений транспорт від клієнта до хмари. Його забезпечує тільки шифрування, причому сертифіковане. Програмних клієнтів для мобільних пристроїв з такою сертифікацією небагато, а сертифікованих реалізацій SSL практично немає.

По-друге, для даних потрібно забезпечити захист усередині ЦОД. Стандартні засоби для цього – міжмережні екрани, IPS та інші інструменти. Але як сертифікувати віртуалізовані міжмережні екрани, якщо в гіпервізорах повно дір? Ці проблеми (сертифікація по ISO 15408, «Загальні критерії») не вирішені й за рубежем. Ми виявляємося в ситуації, коли обмежене кількість міжмережних екранів сертифіковано на відповідність конкретним хмарним середовищам, причому з досить низьким класом.

Говорити про сертифікацію шлюзів VPN поки теж передчасно. Тому провайдери змушені встановлювати виділені апаратні засоби безпеки, сертифіковані на відповідність нормативним вимогам, але такий підхід ламає всю хмарну модель сервісів. Замість хмарної структури все рішення знову виявляється прив'язано до конкретного ЦОД.

Таким чином, доступність і безпеку можна забезпечити наявними на ринку засобами, але, якщо не порушувати ідеологію хмари, це буде погано сполучатися з українською нормативною базою. Хмара – ефективний механізм надання ресурсів, однак прив'язка до конкретних центрів обробки даних негативно впливає на ефективність.

І цим список фундаментальних проблем не вичерпується. Як уважають експерти, традиційні моделі розмежування прав доступу не мають необхідний ступінь надійності, навіть у так званих довірених системах (Trusted Systems). Це стосується й хмарної моделі. Зараз безпека гарантується (і те з певними застереженнями) тільки при фізичному поділі систем. Фахівцям з ІБ варто визнати, що ступінь довіри до «довірених систем» недостатня для того, щоб переходити в хмари. Так у банках служба безпеки виступає навіть проти впровадження віртуалізації на власних площадках банку, оскільки в цьому випадку губиться здатність контролювати системи й бачити, що в них відбувається. У цій ситуації може бути корисна реструктуризація систем – поділ їх на фізично ізольовані зони, кожна з яких побудована по хмарних принципах.

Але життя не стоїть на місці. Хоча, як показав недавнє опитування, багато українські (близько 30%) замовники не користуються послугами комерційних ЦОД з міркувань безпеки, для клієнтів ЦОД ці проблеми не так актуальні – ними стурбовані лише 12% респондентів. Для 17% опитані побоювання щодо безпеки їхніх даних стають причиною звертання до сервісів закордонних ЦОД. Інакше кажучи, українські замовники переносять дані в хмару в тому числі й з метою підвищення їхньої конфіденційності й, як наслідок, забезпечення безпеки свого бізнесу, що непевно почуває себе в нашій країні.

Те, що за рубіж витікають не тільки капітали, але й інформація, не залишається без уваги. Про регулювання хмар замислюється держава.

Крім того, для забезпечення конфіденційності даних при їхній трансграничній передачі потрібне шифрування. Однак закордонні криптоалгоритми не розглядаються як засоби захисту, а українські дуже складно експортувати. Якщо хмара перебуває в країнах, що підписали європейську конвенцію про захист даних, то винос туди персональних даних допускається, оскільки ці країни вважаються благонадійними. Однак вивід даних за рубіж не скасовує вимог по захисту даних, а цим повинен займатися їхній власник. Таким чином, перекласти відповідальність за забезпечення ІБ на провайдеру не вдасться.

Якщо за рубіж переміщаються дані, що не підпадають під дію законодавства, то компанія може це робити без обмежень. Однак у кожному разі українські замовники, особливо із сегмента SMB, хочуть укласти зрозумілий їм договір у рамках звичного правового поля, що буде сприяти створенню локальних ЦОД, уважають експерти.

Для зниження ризиків ІБ як компенсаційні механізми контролю можуть застосовуватися наступні міри: висновок угод NDA і визначення порядку знищення конфіденційної інформації із закінченні дії сервісного контракту, включення в контракт права на періодичного аудита ІБ за погодженими критеріями, чіткий поділ зон відповідальності по розслідуванню й діям сторін при можливих інцидентах безпеки, впровадження рішень для контролю дій віддалених адміністраторів і ін. Нарешті, необхідно ретельно продумувати всі умови угод SLA.

Забезпечення безпеки (у тому числі інформаційної) корпоративних даних, які перебувають у хмарі, і послуги інформаційної безпеки, надавані із хмари. Другий напрямок зараз активно розвивається. Моніторинг безпеки, облік інцидентів і надання звітів компанії – споживачеві послуг є частиною такого сервісу.

Поки лише невелике число компаній (звичайно це великі організації й оператори зв'язку) впроваджують засобу моніторингу інформаційних систем у цілому й моніторингу ІБ зокрема (наприклад, Security Operation Center, SOC). Інші навіть не зможуть довідатися, наприклад, про, що відбулася в них витоку даних. Що ж стосується хмарних провайдерів, те цілодобовий моніторинг найчастіше пропонується в якості одного з додаткових сервісів, що дозволяє підвищити рівень інформаційної безпеки підприємства.

Якщо хмара розглядається як платформа для додатків, то для нього потрібно забезпечити мінімальний рівень безпеки – наприклад, гарантувати схоронність даних. Якщо ж воно стає платформою для сервісів безпеки, то набір послуг буде зовсім іншим.

Тепер з'явилася можливість одержувати сервіси інформаційної безпеки із хмари – «ІБ як послуга» (Security as a Service, SECaaS). Ми почали уживати певних заходів у даному напрямку й пропонуємо, наприклад, Trend Micro Deeep Security as a Service. Зокрема, замовниками SECaaS можуть стати компанії, яким занадто дорого або складно забезпечувати свою власну інформаційну безпеку й простіше доручити це завдання провайдеру, опираючись на SLA і NDA.

Українські організації виявляють цікавість до аутсорсингу послуг центра моніторингу й керування подіями безпеки (Security Operations Center, SOC). По оцінках експертів, при типовому бюджеті на ІБ в 5-10% від загального Ит-Бюджету аутсорсинг SOC обійдеться вдвічі дешевше придбання власних рішень (хоча через 4,5 роки витрати зрівняються) і до того ж дозволить скористатися експертизою й досвідом провайдеру. До аутсорсингу ІБ звертаються через недостачу власного персоналу, з метою організації безперебійної роботи сервісів (24x7), а також для забезпечення високого рівня доступності, швидкого запуску сервісів, зниження залежності від своїх фахівців і скорочення управлінських витрат. На аутсорсинг можна віддавати керування ризиками ІБ, контроль за виконанням нормативних вимог, забезпечення працездатності систем ІБ, їхнє адміністрування, моніторинг і реагування на інциденти.

В Україні хмарні сервіси безпеки ще тільки починають розвиватися, однак дуже часто все впирається в неможливість складання адекватних угод SLA. Тим часом у світі такі сервіси активно використовуються, наприклад, для фільтрації доступу до різних ресурсів. До цієї сервісної хмари можуть підключатися, наприклад, пристрої, що обслуговують школи. Крім того, у хмарі вже працює велика кількість антивірусних движків. Цікава технологія, що добре реалізується в хмарі, – перевірочне виконання недовіреного програмного коду в ізольованому середовищі (sandboxing) перед пропуском його до користувача.

Для малих і середніх підприємств хмарні сервіси безпеки набагато ефективніше з економічної й технологічної точок зору й надають більше широкі можливості для захисту бізнесу, чим власна система ІБ. Через два-три роки ця тема буде активно обговорюватися.

Не перший рік в Україні застосовуються системи Web-фільтрації й контентної фільтрації, установлені в провайдера. У хмари давно виносяться системи антиспаму й електронна пошта з відповідними механізмами захисту. Інші ж сервіси тільки починають розвиватися, у тому числі SOC. Поки не ясно, наскільки активно вони будуть використовуватися через рік-два. Однак базові хмарні сервіси ІБ уже досить зрілі. Кожної компанії залишається тільки зрівняти, яке сполучення засобів хмарної й «локальної» безпеки буде для неї оптимальним по фінансових витратах, мінімізації ризиків і можливості реалізації.

Насправді часто мова йде про використання якихось певних ресурсів із хмари, наприклад баз URL. При цьому сам сервіс працює локально. Сервіси безпеки в хмарі ми поки розміщати не вміємо. Ще рідкі випадки, коли весь сервіс ІБ перебуває в хмарі й туди перенаправляється трафік користувача. Таких комерційних послуг у нашій країні ще дуже мало.

Замовники нерідко сумніваються не тільки в безпеці хмар і хмарних сервісів, але й у їхній надійності. Наріжним каменем залишається довіра до провайдеру. Приваблювані більше високою надійністю сервісу, передбачуваністю контрагентів, виразністю SLA і законодавства, меншими

ризиками, деякі підприємства віддають перевагу великим закордонним хмарним провайдерам, які вже зараз несуть відповідальність, наприклад, за забезпечення необхідного рівня доступності даних.

Тим часом навіть українські хмарні провайдери здатні забезпечити більшу надійність і схоронність даних, чим багато підприємств. В угоді про рівень сервісу передбачаються не тільки параметри доступності й продуктивності, але також обмовляється політика резервного копіювання. За втрату даних, що перебувають у хмарі, передбачене стягнення штрафу в розмірі повного місячного платежу при неприступності сервісу. За замовчуванням всі дані резервуються. У випадку особливих вимог замовника можлива зміна періодичності й технології резервного копіювання або використання таких рішень, як метрокластер – географічно розподілений кластер віртуалізації з дублюванням всіх компонентів (у тому числі задіяних ліній зв'язку) шляхом рознесення їх по двох різних площадках і синхронізації даних на рівні СЗД по FC SAN. У такий спосіб забезпечується катастрофостійкість із RTO не більше 15 хв. для віртуальної машини й зберігається вся функціональність «живої» міграції й гнучкості в масштабуванні віртуального ресурсного пула.. Звичайно, ці послуги надаються за додаткову плату.

Безумовно, провайдер може нести відповідальність, передбачену умовами SLA, але навряд чи штрафні санкції у вигляді знижок або пільг улаштують клієнта. Метрокластер захищає від фізичного збою СЗД із рівнем готовності «п'ять дев'яток», однак системи зберігання корпоративного класу й так мають високу надійність. Крім того, метрокластер не захищає від «логічних збоїв»: наприклад, якщо адміністратор помилково відформатує диск (або це зробить зловмисник), його вміст буде автоматично репліковано на віддалену площадку. Серйозна проблема полягає в тому, що в Україні практично немає хмарних архітекторів, здатних грамотно спроектувати й побудувати надійне й безпечне хмарне середовище.

Широке поширення ЦОД і хмарних технологій означає новий виток розвитку в технологіях ІБ. У нових умовах доводиться говорити не про захист периметра, а про захист даних. Безпечну роботу потрібно забезпечити не тільки для корпоративного ЦОД і співробітників підприємства, але й для мобільних користувачів. Активний розвиток хмарних сервісів – ще один стимул для росту ринку засобів ІБ, що дає можливість заробити цілому ряду гравців, незважаючи на кризи й спади. Причому хмарні сервіси безпеки будуть постійно вдосконалюватися. Це одна з тих тенденцій, які визначають розвиток ринку на найближчі два-три роки.

Основи туннелювання

Туннелювання (tunneling), або інкапсуляція (encapsulation), – це спосіб передачі корисної інформації через проміжну мережу. Такою інформацією можуть бути кадри (або пакети) іншого протоколу. При інкапсуляції кадр не передається в згенерованому вузлом-відправником виді, а забезпечується додатковим заголовком, що містить інформацію про маршрут, що дозволяє інкапсульованим пакетам проходити через проміжну мережу (Internet). На кінці тунелю кадри деінкапсулюються й передаються одержувачеві.

Цей процес (який включає інкапсуляцію й передачу пакетів) і є туннелювання. Логічний шлях пересування інкапсульованих пакетів у транзитній мережі називається тунелем.

VPN працює на основі протоколу PPP(Point-to-Point Protocol). Протокол PPP розроблений для передачі даних по телефонних лініях і виділених з'єднаннях « точка-точка». PPP інкапсулює пакети IP, IPX і NetBIOS у кадри PPP і передає їх по каналі «точка-точка». Протокол PPP може використовуватися маршрутизаторами, з'єднаними виділеним каналом, або клієнтом і сервером RAS, з'єднаними віддаленим підключенням.

Основні компоненти PPP:

– Інкапсуляція – забезпечує мультиплексування декількох транспортних протоколів по одному каналі

– Протокол LCP – PPP задає гнучкий LCP для установки, налаштування й перевірки каналу зв'язку. LCP забезпечує узгодження формату інкапсуляції, розміру пакета, параметри установки й розриву з'єднання, а також параметри автентифікації. Як протоколи автентифікації можуть використовуватися PAP, CHAP і ін.

– Протоколи керування мережею – надають специфічні конфігураційні параметри для відповідних транспортних протоколів. Наприклад, IPCP протокол керування IP.

– Для формування тунелів VPN використовуються протоколи PPTP, L2TP, IPsec, IP-IP.

– Протокол PPTP – дозволяє інкапсулювати IP-, IPX- і NetBEUI-трафік у заголовки IP для передачі по IP-Мережі, наприклад Internet.

– Протокол L2TP – дозволяє шифрувати й передавати IP-трафік з використанням будь-яких протоколів, що підтримують режим « точка-точка» доставки дейтаграмм. Наприклад, до них ставляться протокол IP, ретрансляція кадрів і асинхронний режим передачі (ATM).

– Протокол IPsec – дозволяє шифрувати й інкапсулювати корисну інформацію протоколу IP у заголовки IP для передачі по IP-Мережах.

– Протокол IP-IP – IP-дейтаграмма інкапсулюється за допомогою додаткового заголовка IP. Головне призначення IP-IP – туннелюванние багатоадресного трафіку в частинах мережі, не підтримуючих багатоадресну маршрутизацію.

Для технічної реалізації VPN, крім стандартного мережного встаткування, знадобиться шлюз VPN, що виконує всі функції по формуванню тунелів, захисту інформації, контролю трафіку, а нерідко й функції централізованого керування.

На сьогоднішній день VPN – це економічне, надійне й загальнодоступне рішення організації віддаленого доступу. Яким би не була відстань, VPN забезпечить з'єднання з будь-якою точкою світу й схоронність передачі найважливіших даних.

3.2 Розробка структурної схеми

Технологія VPN (Virtual Private Network – віртуальна приватна мережа) – не єдиний спосіб захисту мереж і переданих по них даних. Але я вважаю, що вона досить ефективна, і її повсюдне впровадження – це не тільки данина моді, досить прихильної до VPN в останні пару років.

Оскільки інформація, що циркулює в Інтернеті, являє собою безліч пакетів протоколу IP, VPN-агенти працюють саме з ними.

Перед відправленням IP-пакета VPN-агент діє в такий спосіб:

– З декількох підтримуваних їм алгоритмів шифрування й ЕЦП по IP-адресі одержувача вибирається потрібний для захисту даного пакета, а також ключі. Якщо ж у його налаштуваннях такого одержувача ні, то інформація не відправляється.

– Визначає й додає в пакет ЕЦП відправника або імітовставку.

– Шифрує пакет (цілком, включаючи заголовок).

– Проводить інкапсуляцію, тобто формує новий заголовок, де вказується адреса зовсім не одержувача, а його VPN-агента. Ця корисна додаткова функція дозволяє представити обмін між двома мережами як обмін всього-на-всього між двома комп'ютерами, на яких установлені VPN-агенти. Усяка корисна для темних цілей зловмисника інформація, наприклад внутрішні IP-адреси, йому вже недоступна.

При одержанні IP-пакета виконуються зворотні дії:

– Заголовок містить відомості про VPN-агента відправника. Якщо такий не входить у список дозволених у налаштуваннях, то інформація просто відкидається. Те ж саме відбувається при прийманні пакета з навмисно або випадково ушкодженим заголовком.

– Відповідно до налаштувань вибираються алгоритми шифрування й ЕЦП, а також необхідні криптографічні ключі.

– Пакет розшифровується, потім перевіряється його цілісність. Якщо ЕЦП невірна, то він викидається.

– І, нарешті, пакет у його вихідному виді відправляється справжньому адресатові по внутрішній мережі.

Всі операції виконуються автоматично. Складним в технології VPN є тільки налаштування VPN-агентів, що, втім, цілком під силу досвідченому користувачеві.

VPN-агент може перебувати безпосередньо на ПК, який захищається, що корисно для мобільних користувачів, що підключаються до Інтернет. У цьому випадку він забезпечить обмін даними тільки того комп'ютера, на якому встановлений.

Можливе сполучення VPN-агента з маршрутизатором (у цьому випадку його називають криптографічним) IP-пакетів. До речі, що ведуть світові виробники останнім часом випускають маршрутизатори з вбудованою підтримкою VPN, наприклад Express VPN від Intel, що шифрує всі пакети, які проходять, за алгоритмом Triple DES.

Як видно з опису, VPN-агенти створюють канали між мережами, що захищаються, які звичайно називають “тунелями”. І дійсно, вони “прориті” через Інтернет від однієї мережі до іншої, циркулююча усередині інформація захована від чужих очей.

Крім того, всі пакети “фільтруються” відповідно до налаштувань. Таким чином, всі дії VPN-агентів можна звести до двох механізмів: створення тунелів і фільтрація минаючих пакетів.

Сукупність правил створення тунелів, що називається “політикою безпеки”, записується в налаштуваннях VPN-агентів. IP-пакети направляються в той або інший тунель або відкидаються після того, як будуть перевірені:

– IP-адреса джерела (для вихідного пакета – адреса конкретного комп'ютера мережі, що захищається);

– IP-адреса призначення;

– протокол більш високого рівня, якому належить даний пакет (наприклад, TCP або UDP);

– номер порту, з якого або на який відправлена інформація (наприклад, 1080).

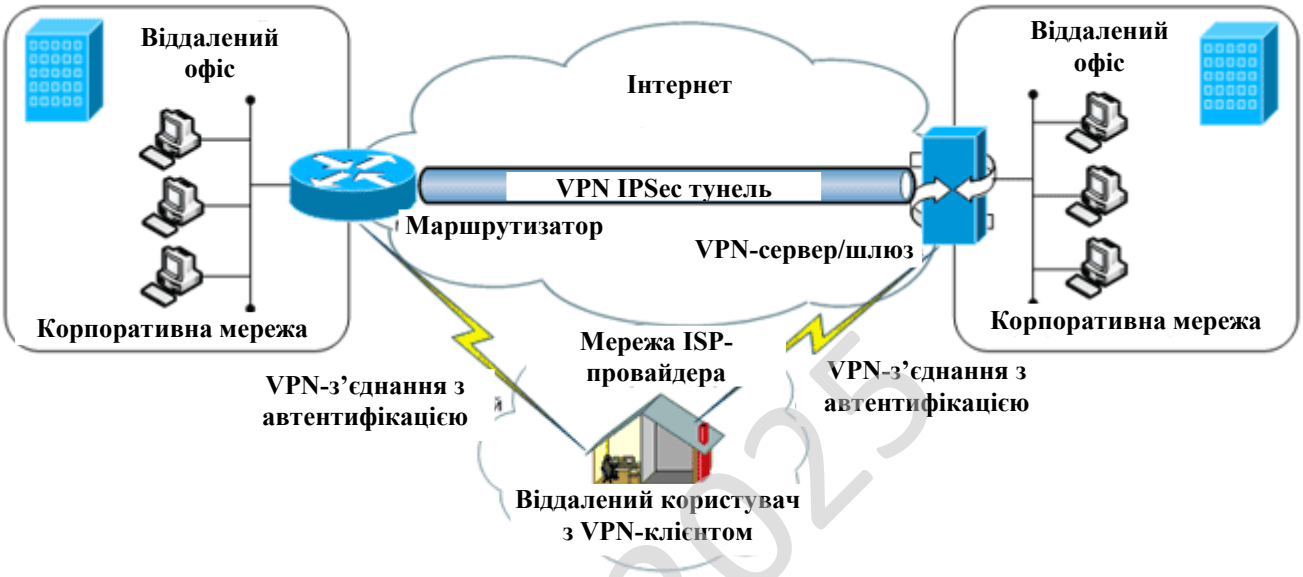

Рисунок 3.1 – Структурна схема системи з використанням VPN

У системі використовуються наступні протоколи VPN:

– IPSec (IP security) – часто використовується поверх IPv4.

– PPTP (point-to-point tunneling protocol) – розроблявся спільними зусиллями декількох компаній, включаючи Microsoft.

– PPPoE (PPP (Point-to-Point Protocol) over Ethernet).

– L2TP (Layer 2 Tunnelling Protocol) – використовується в продуктах компаній Microsoft і Cisco.

– L2TPv3 (Layer 2 Tunnelling Protocol version 3).

– OpenVPN SSL VPN з відкритим вихідним кодом, підтримує режими PPP, bridge, point-to-point, multi-client server.

Суть VPN полягає в наступному:

– На всі комп'ютери, що мають вихід в Інтернет, установлюється засіб, що реалізує VPN (VPN-агент). Не повинно залишитися жодного незахищеного.

– VPN-агенти автоматично шифрують всю вихідну інформацію (і відповідно розшифровують всю вхідну). Вони також стежать за її цілісністю за допомогою ЕЦП або імітовставок (криптографічна контрольна сума, розрахована з використанням ключа шифрування).

3.3 Розробка функціональної схеми

На рисунку 3.3 зображена функціональна схема системи. Нижче розглянемо її більш докладно. У системі використовуються наступні протоколи. В основу програмного забезпечення захисту інформації у Cloud-системах з використанням VPN покладені протоколи забезпечення безпеки інформації IPsec та SSL.

Заголовок AH

Автентифікуючий заголовок (AH) є звичайним опціональним заголовком і, як правило, розташовується між основним заголовком пакета IP і полем даних. Наявність AH ніяк не впливає на процес передачі інформації транспортного й більш високого рівнів. Основним і єдиним призначенням AH є забезпечення захисту від атак, пов'язаних з несанкціонованою зміною вмісту пакета, і в тому числі від підміни вихідної адреси мережного рівня. Протоколи більш високого рівня повинні бути модифіковані з метою здійснення перевірки автентичності отриманих даних.

Формат AH досить простий і складається з 96-бітового заголовка й даних змінної довжини, що складаються з 32-бітових слів. Назви полів досить ясно відбивають їхній зміст: Next Header указує на наступний заголовок, Payload Len представляє довжину пакета, SPI є покажчиком на контекст безпеки й Sequence Number Field містить послідовний номер пакета.

Наступний заголовок	Довжина навантаження	Зарезервовано
Індекс параметрів безпеки (SPI)		
Поле послідовного номеру		
Дані аутентифікації (перемінної довжини)		

Рисунок 3.2 – Формат заголовка AH

Послідовний номер пакета був введений в AH в 1997 році в ході процесу перегляду специфікації IPsec. Значення цього поля формується відправником і служить для захисту від атак, пов'язаних з повторним використанням даних процесу автентифікації. Оскільки мережа Інтернет не гарантує порядок доставки пакетів, одержувач повинен зберігати інформацію про максимальний послідовний номер пакета, що пройшов успішну автентифікацію, і про одержання деякого числа пакетів, що містять попередні послідовні номери (звичайно це число дорівнює 64). На відміну від алгоритмів обчислення контрольної суми, застосовуваних у протоколах передачі інформації з линиям зв'язку, що комутуються або по каналах локальних мереж і орієнтованих на виправлення випадкових помилок середовища передачі, механізми забезпечення цілісності даних у відкритих телекомунікаційних мережах повинні мати засоби захисту від внесення цілеспрямованих змін. Одним з таких механізмів є спеціальне застосування алгоритму MD5: у процесі формування AH послідовно обчислюється хеш-функція від об'єднання самого пакета й деякого попередньо погодженого ключа, а потім від об'єднання отриманого результату й перетвореного ключа. Даний механізм застосовується за замовчуванням з метою забезпечення всіх реалізацій IPv6, принаймні, одним загальним алгоритмом, не підданим експортним обмеженням.

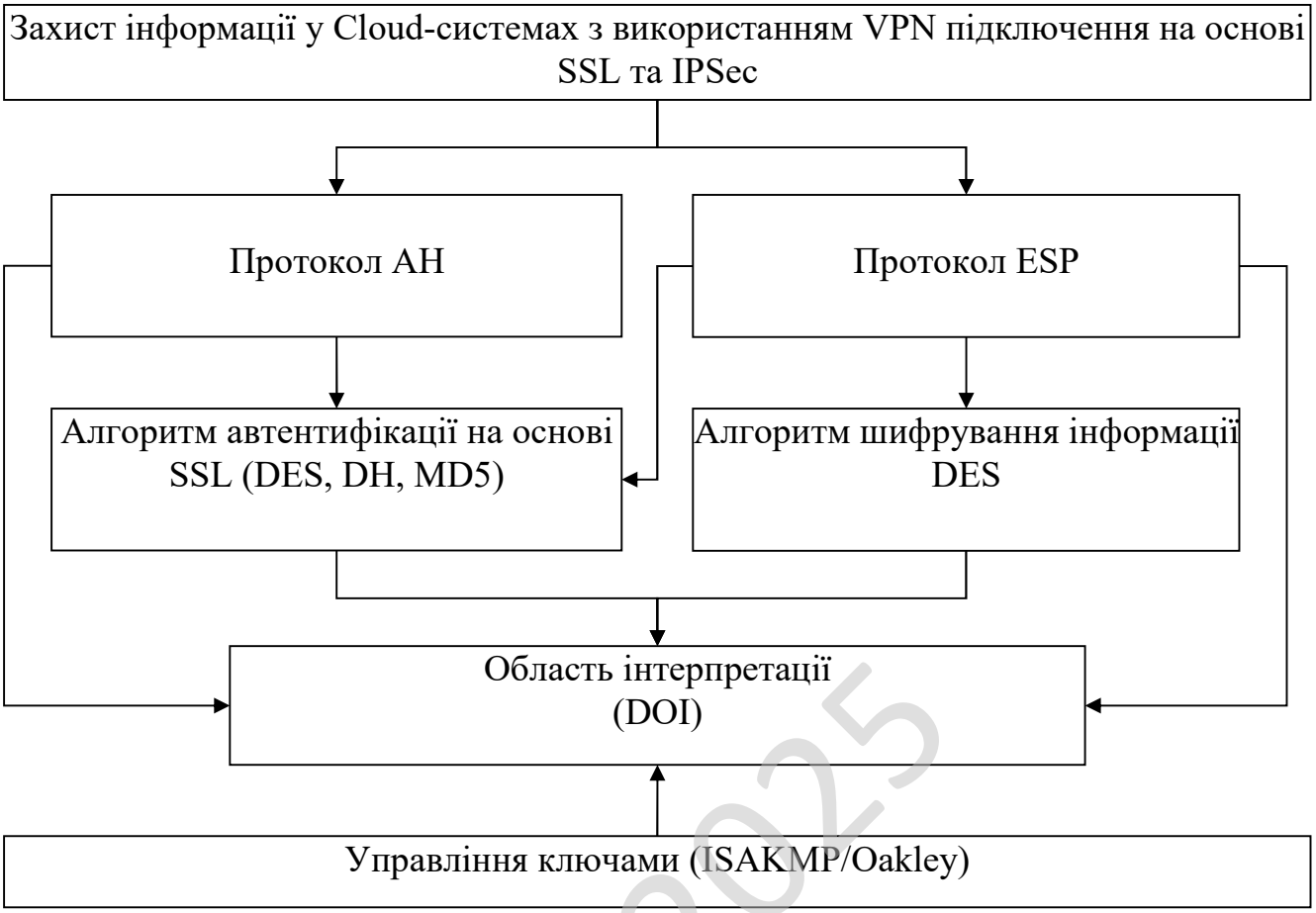

Рисунок 3.3 – Функціональна схема системи

Заголовок ESP – інкапсуляція зашифрованих даних

У випадку використання інкапсуляції зашифрованих даних заголовок ESP є останнім у ряді опціональних заголовків, "видимих" у пакеті. Оскільки основною метою ESP є забезпечення конфіденційності даних, різні види інформації можуть вимагати застосування істотно різних алгоритмів шифрування. Отже, формат ESP може перетерплювати значні зміни залежно від використовуваних криптографічних алгоритмів. Проте, можна виділити наступні обов'язкові поля: SPI (SPI – Secutity Parameter Index – індекс параметра безпеки), що вказує на контекст безпеки, поле порядкового номера, що містить послідовний номер пакета, і контрольна сума, призначена для захисту від атак на цілісність зашифрованих даних. Крім цього, як правило, у тілі ESP присутні параметри (наприклад, режим використання) і дані (наприклад, вектор

ініціалізації) застосовуваного алгоритму шифрування. Частина ESP заголовка може бути зашифрована на відкритому ключі одержувача або на спільному ключі пари відправник-одержувач. Одержувач пакета ESP розшифровує ESP заголовок і використовує параметри й дані застосовуваного алгоритму шифрування для декодування інформації транспортного рівня.

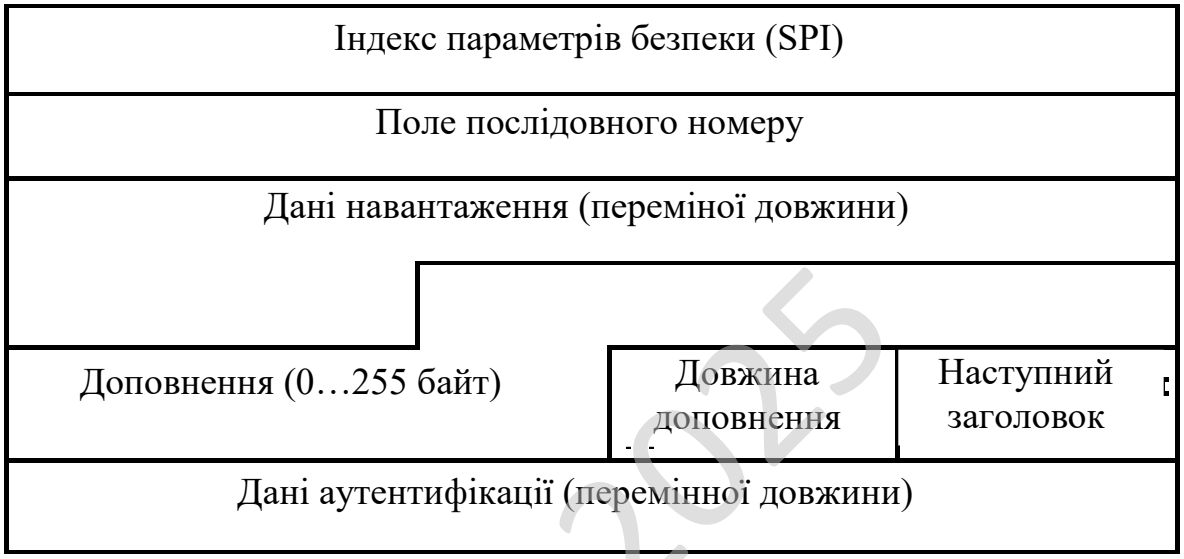

Рисунок 3.4 – Формат заголовка ESP

Розрізняють два режими застосування ESP – транспортний і тунельний.

Транспортний режим – використовується для шифрування поля даних IP пакета, що містить протоколи транспортного рівня (TCP, UDP, ICMP), який, у свою чергу, містить інформацію прикладних служб. Прикладом застосування транспортного режиму є передача електронної пошти. Всі проміжні вузли на маршруті пакета від відправника до одержувача використовують тільки відкриту інформацію мережного рівня й, можливо, деякі опціональні заголовки пакета (в IPv6). Недоліком транспортного режиму є відсутність механізмів приховання конкретних відправника й одержувача пакета, а також можливість проведення аналізу трафіку. Результатом такого аналізу може стати інформація про об'єми й напрямки передачі інформації, області інтересів абонентів, розташування

керівників.

Тунельний режим – припускає шифрування всього пакета, включаючи заголовок мережного рівня. Тунельний режим застосовується якщо буде потреба приховання інформаційного обміну організації із зовнішнім миром. При цьому, адресні поля заголовка мережного рівня пакета, що використовує тунельний режим, заповнюються міжмережним екраном організації й не містять інформації про конкретного відправника пакета. При передачі інформації із зовнішнього світу в локальну мережу організації як адреса призначення використовується мережна адреса міжмережного екрана. Після дешифрування міжмережним екраном початкового заголовка мережного рівня пакет направляється одержувачеві.

Протокол ISAKMP/Oakley

Завдання алгоритмів IPsec – справа непроста, для цього потрібен протокол керування сеансом. Протокол ISAKMP (Internet Security Association Key Management Protocol) є рамковою основою для такого протоколу, а протокол Oakley – це вже конкретна реалізація його на цій основі, призначена для спільного використання з IPsec.

Протокол Oakley має більш широкий набір функціональних можливостей, ніж необхідно для керування IPsec-сеансами. Реалізація ISAKMP/Oakley являє собою функціональну підмножину, достатню, щоб забезпечити безпечний спосіб повідомлення автентифікованих даних для генерації ключів і SA-параметрів. Обмін по протоколу ISAKMP/Oakley відбувається у двох режимах (фазах): основному й швидкому. Відповідно до протоколу Oakley, обмін починається в основному й триває у швидкому режимі. У першому режимі встановлюються угоди SA для обміну даними по протоколу Oakley, а в другому – по протоколу IPsec.

На один обмін в основному режимі може доводитися кілька обмінів у швидкому, так як час існування SA-угоди для протоколу Oakley може бути більш тривалим, ніж для протоколу IPsec. Завдяки обмеженому строку існування SA-угод комбінування в сеансі основного й швидкого режимів забезпечує дуже

потужний захисний механізм обміну ключами.

Обмін ключами в основному режимі здійснюється по методу Діффі-Хелмана (DH), що вимагає інтенсивного використання обчислювальних ресурсів. Цей метод є механізмом розподілу відкритих ключів для безпечного обміну секретною інформацією без застосування якої-небудь інформації, заздалегідь відомим обом сторонам. Тому ним активно користуються для встановлення безпечних сеансів зв'язку в тих випадках, коли необхідний динамічний захист і коли кіцеві системи не належать однієй й тій же системі адміністративного керування. Наприклад, метод DH можна використовувати в електронній комерції при встановленні з'єднання для передачі транзакцій між двома компаніями.

Хоча цей метод і вимагає більших обчислювальних ресурсів, при його застосуванні можливий компроміс між криптостійкістю алгоритму (при використанні менш довгих відкритих ключів) і необхідним об'ємом обчислень. Обмін ключами у швидкому режимі не вимагає великого об'єму обчислень, так як тут використовується набір простих математичних операцій. Існує обмеження припустимого числа швидких фаз, перевищення якого веде до того, що ключі, згенеровані в основній фазі, а потім використовувані у швидких фазах, виявляться під погрозою розкриття. На сьогоднішній день немає твердого правила, що визначає число швидких фаз на одну основну фазу; криптографи діють, керуючись загальними міркуваннями й з огляду на оперативну обстановку.

В основному режимі обоє учасника обміну встановлюють SA-угоди для безпечного спілкування один з одним по протоколу Oakley. У швидкому режимі SA-угоди встановлюються вже "від імені" протоколу IPsec або будь-якої іншої служби, який необхідні дані для генерації ключів або узгодження параметрів. Протокол Oakley розроблений таким чином, що він ніяк не пов'язаний з IPsec. Наприклад, для підвищення безпеки процесу встановлення сеансів його цілком можна використовувати разом із протоколом SSL (Secure Sockets Layer) версії 4.0 замість механізму обміну ключами SSL 3.0.

DOI – область інтерпретації

Протокол ISAKMP/Oakley не був спеціально розроблений для спільного використання із протоколом IPsec, тому виникає необхідність у так званій області інтерпретації (Domain Of Interpretation – DOI), що забезпечила б спільну роботу протоколів IPsec і ISAKMP/Oakley. Щоб інші протоколи також могли використовувати ISAKMP/Oakley, вони повинні мати власні DOI-області. У даний момент таких областей для інших протоколів не існує, але ситуація може змінитися на черговій конференції групи IETF або в тому випадку, якщо приватний розроблювач, наприклад фірма Netscape, вирішить використовувати цей механізм. Більш докладно про це можна прочитати в документі "The Internet Key Exchange (IKE)", розробленому робочою групою IP Security Protocol Working Group (ftp://ftp.ietf.org/ internet-draft/draft-ietf-IPsec-isakmp-oakley-06.txt).

В основному режимі між сторонами погоджуються методи шифрування, хешування, автентифікації й так звана група DH (їх усього чотири), що визначає криптографічну стійкість алгоритму відкритого розподілу ключів. Перша група DH характеризується високою стійкістю й дозволяє використовувати стандарт DES, у той час як для другої й третьої груп варто застосовувати Triple DES. Оскільки в основному режимі іноді потрібно передавати до шести пакетів, то, наприклад, при використанні космічного сегмента з великою тимчасовою затримкою, DES краще застосовувати з більш сильною групою DH. Тоді перед виконанням чергового основного режиму, сполученого з інтенсивними обчисленнями й обміном пакетами, вам удасться виконати більше обмінів у швидкому режимі.

Коли SA-угода для обміну по протоколу Oakley установлюється в основному режимі, створюється ланцюжок випадкових біт, що використовують для генерації ключів. Також визначається тривалість (за часом або кількістю переданих даних) "життя" SA-угоди Oakley і дані для генерації ключів до того, як буде потрібно наступний обмін в основному режимі.

Швидкий режим простіше основного, і узгодження SA для IPsec здійснюється за допомогою трьох пакетів. IPsec-ключі створюються за допомогою простих операцій піднесення в ступінь переданих в основному режимі даних. У швидкому режимі погодяться також алгоритми шифрування й строки існування SA для IPsec-сеансів.

Згідно із цими строками визначається, як незабаром, залежно від часу або об'єму переданих даних, буде потрібно нове узгодження у швидкому режимі. Помітьте, є два різних строки існування SA-угоди. Основний режим задає його для протоколу Oakley, а швидкий – для обміну по протоколу IPsec. Як приклад пропонуємо значення цих параметрів для шифрування IPsec-сеансів за допомогою алгоритму DES: 15 хв або 10 Мбайт для швидкого режиму, і 60 хв або 40 Мбайт для основного. Ці числа варто збільшити для Triple DES і зменшити для ARCFour (в ARCFour застосовується 40-бітний, а в TripleDES – 112-бітний ключ). Такий підхід дозволяє збалансувати криптографічну стійкість сервісів IPsec і вартість накладних витрат на передачу пакетів ISAKMP/Oakley.

При генерації ключів в основному режимі сеанс можна примусово перервати на підставі відкликання сертифіката. Сертифікати кінцевих вузлів використовуються тільки під час основного режиму. Таким чином, при анулюванні одного із сертифікатів обмін перерветься тільки в основному режимі. Тимчасові обмеження, погоджені в основному й швидкому режимах, значно відрізняються друг від друга й залежать від типу даних і транзакцій, що використовують IPsec-з'єднання. Для правильного визначення цих обмежень із обліком, з одного боку, об'єму обчислень і навантаження на мережу, а з іншого боку – імовірності порушення захисту даних, потрібно деякий аналіз.

Сполучення різних IPsec-механізмів забезпечує цілком безпечні з'єднання як між мережами, так і між кінцевими станціями. Оскільки практично всі постачальники підтримують ці стандарти, рано або пізно це приведе до виникнення середовища для реалізації безпечних з'єднань через Інтернет. Таким чином, протокол IPsec стане основним для безпечної е-комерції в Інтернет.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.5. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Рисунок 3.5 – Діаграма взаємодії процесів

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

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми. При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захисту інформації у Cloud-системах з використанням VPN.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML:

– діаграма діяльності (діаграми поведінки типу);

– діаграма прецедентів (діаграми поведінки типу);

– діаграма класів;

– діаграма компонент;

– діаграма об'єктів;

– діаграма розгортання.

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

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання

діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

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

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

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

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

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

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

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

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

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

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

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

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

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

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

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

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

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

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

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

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

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на

одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

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

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

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

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент.

Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

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.

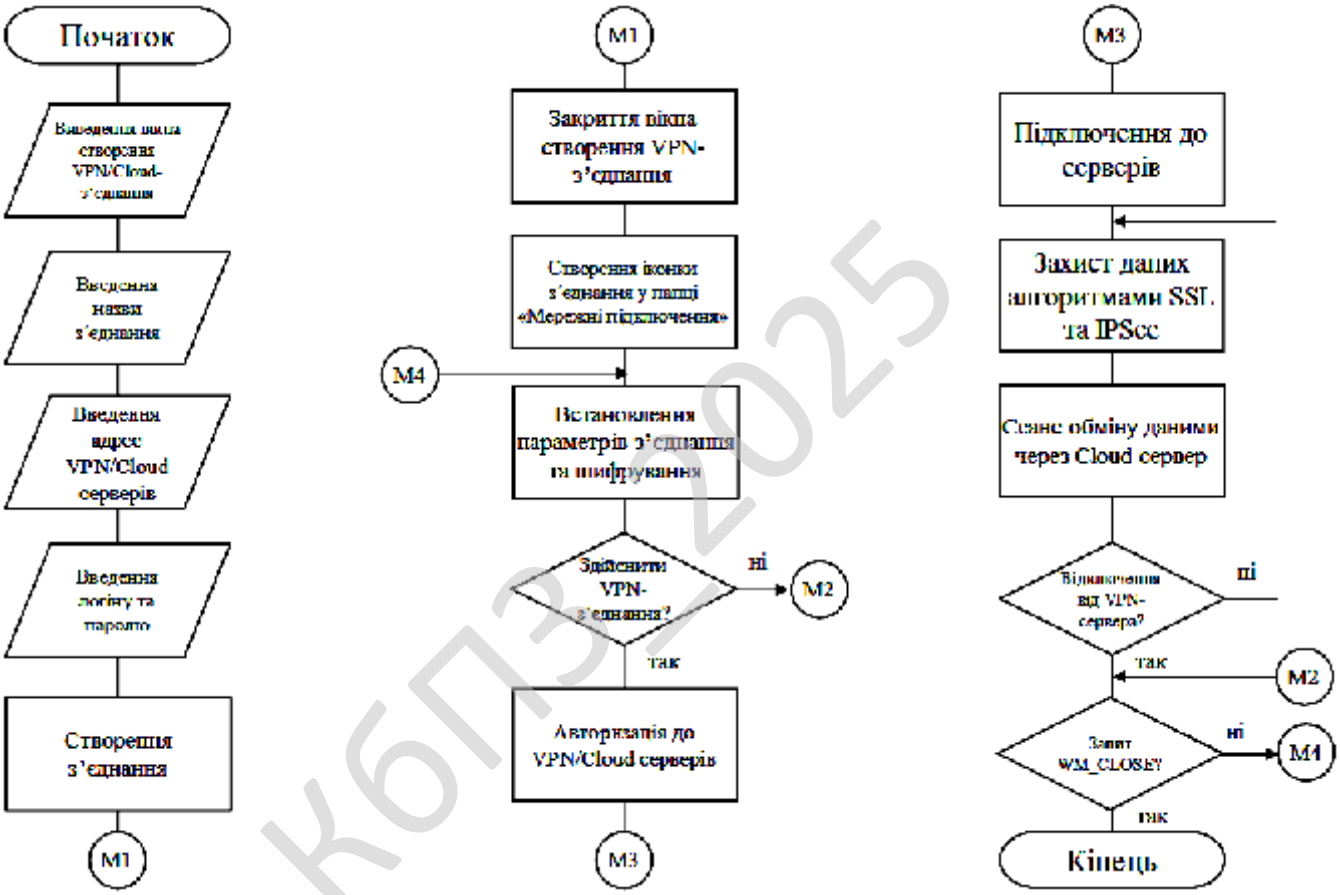

Рисунок 4.1 – Блок-схема основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Розповсюджені помилки VPN підключення.

Якщо при запуску VPN-з'єднання виникла помилка, то необхідно запам'ятати її тризначний код. Як правило помилки мають код 6XX (помилки, пов'язані з некоректним введенням пароля, ім'я або вибору типу протоколу зв'язку й говорять про те, що мережа працює, але необхідно перевірити дані, що вводяться), 7XX (помилки, пов'язані з некоректними налаштуваннями з'єднання й говорять про те, що мережа працює, але необхідно перевірити налаштування) або 8XX (помилки, пов'язані із проблемами в налаштуванні мережі або відсутністю підключення до мережі.

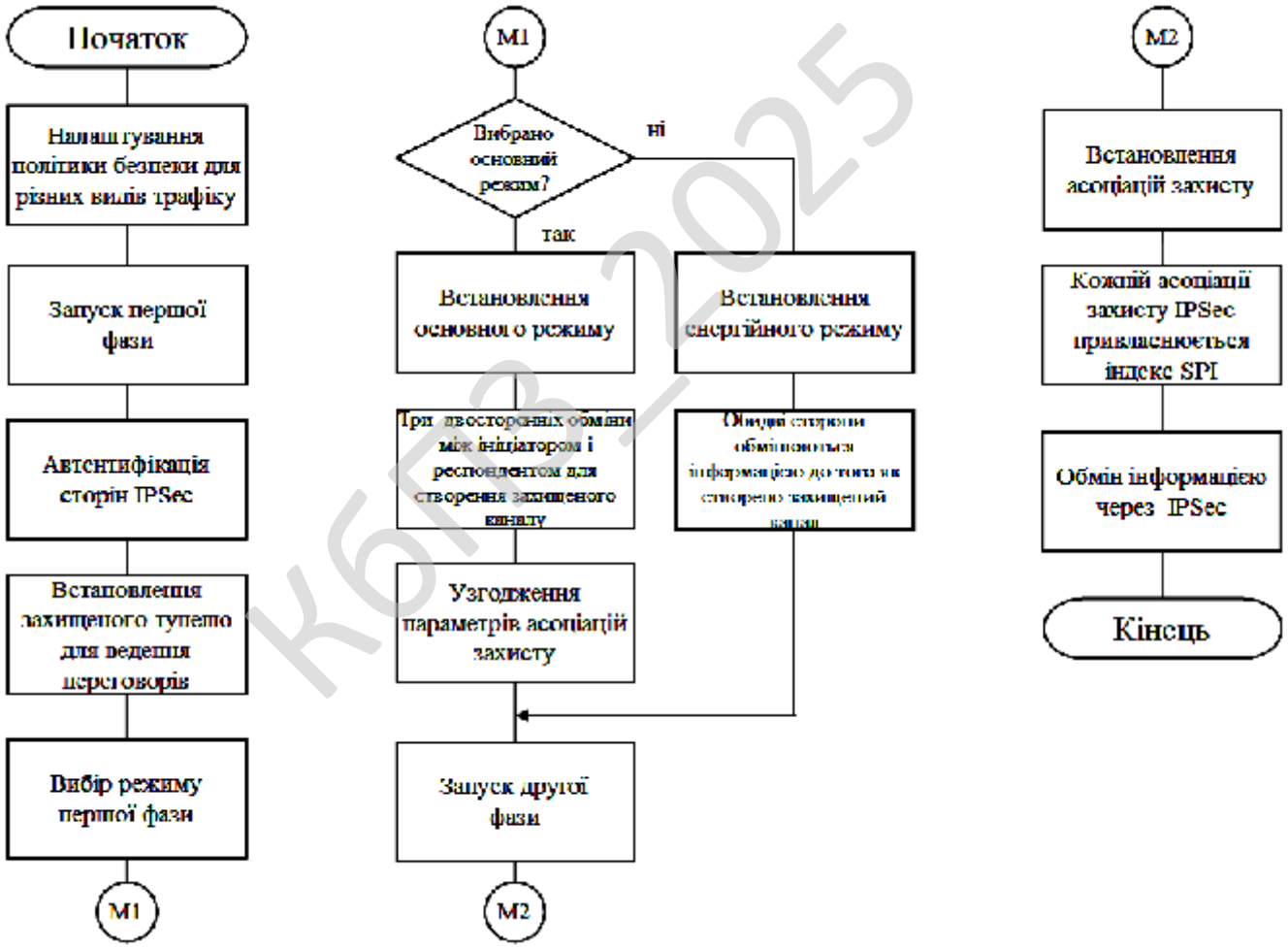

Рисунок 4.2 – Блок-схема роботи підпрограми

У випадку виникнення помилок цієї серії зв'яжіться з нашою службою підтримки). Нижче наведені розшифровки деяких кодів:

А тепер докладні пояснення по основних видах помилок:

400 Bad Request. Означає, що сервер виявив у запиті клієнта синтаксичну помилку.

1. Відключити міжмережний екран (брандмауер) або програми, що впливають на мережну активність і безпеку.

2. Спробувати обновити поточний браузер або використовувати іншої.

3. Поставити налаштування браузера по-умовчанню й обнулити всі збережені файли, сертифікати та інше, тобто видалити всі що пише браузер на диск.

Помилка 624, проблема при підключенні до VPN:

1. Відсутній файл RASPHONE.PBK в {папка з Windows}/system32/RAS або ... Documents and Settings/{ваш юзер або всі юзери}/Application Data/Microsoft/Network/Connections/Pbk, а якщо навіть і є, то перейменувати його в щось інше, наприклад RASPHONE.OLD, ще можна запустити rasphone.exe, може допомогти.

2. Є заборона на створення підключення. – Дозволити користувачам створювати нові підключення. Пуск-виконати-gpedit.msc і там розбиратися.

3. Заборонено доступ до папки Documents and Settings/All Users/Application Data/Microsoft/Network/Connections/Pbk – Відкрити доступ до папки. Але швидше за все проблему вдасться вирішити в такий спосіб: зняти галку з пункту "Тільки для читання" у властивостях файлу Rasphone.pbk.

Помилка 691, проблема при підключенні до VPN:

1. Немає грошей.

2. Неправильний логін або пароль.

3. Неправильно зазначений у налаштуваннях сервер VPN.

4. Є користувальницьке або адміністративне блокування в системі статистики.

Помилка 800, проблема при підключенні до VPN:

Можливі причини:

1. Брандмауер блокує вихідні запити на VPN з'єднання, що в принципі трапляється, але не так часто.

2. Запит з якої-небудь причини не доходить до сервера, тобто можливо шлюз сегмента не пропускає запит у силу виниклого навантаження або збою. Що теж може траплятися, але дуже-дуже рідко.

3. Сервер відправляє відповідь про неможливість підключитися тому що в цей момент спостерігається велика кількість одночасних спроб з'єднання.

Можливі виправлення:

1. Перевірити чи працює локальна мережа в цей момент часу.

2. Перевірити проходження сигналу командою ping до шлюзу, а потім до сервера авторизації.

Натисніть кнопку "Пуск"->"Виконати". У рядку введіть команду.

Приклад:

ping 172.22.0.1

ping 172.22.0.254

де в цьому випадку 172.22.0.254 – сервер VPN доступу, а 172.22.0.1 – адреса локального сервера. Свій сервер доступу й адреса шлюзу Ви можете довідатися в "анкеті абонента" або через службу тех. підтримки.

Якщо у вікні що відкрилося побачите щось начебто "Заданий вузол недоступний" або "Перевищений інтервал очікування для запиту", то можливо проблеми на комп'ютері, або на лінії. Якщо піде відповідь від вузла із зазначеною швидкістю (звичайно це відбувається швидко й віконце швидко зникає), то швидше за вся проблема у комп’ютері. У кожному разі перевірте перший пункт (1) і виконаєте дії наступного пункту (3).

3. Перезавантажити свій комп'ютер.

Помилка 650. "Сервер віддаленого доступу не відповідає". Недоступний сервер доступу в Інтернет. Перевірте, чи включене "Підключення по локальній

мережі", чи справна мережна карта, чи справний мережний кабель, чи не виставлений у налаштуваннях IP-з'єднання якась певна IP-адреса.

Помилка 735. "Запитана адреса була відкинута сервером". Ви неправильно настроїли VPN-з'єднання для доступу в мережу Інтернет. Швидше за все в налаштуваннях TCP/IP з'єднання VPN прописали якусь IP-адресу, а він повинен виділяться автоматично.

Помилка 789, "Обраний невірний тип VPN з'єднання". Зайдіть у налаштування VPN з'єднання й на вкладці "Мережа" зі списку "Тип VPN" виберіть "Автоматично". Спробуйте повторно підключитися.

Помилки 741 – 743, "Невірно настроєні параметри шифрування". Зайдіть у налаштування VPN з'єднання, і у вкладці "Безпеку" відключите пункт "шифрування даних".

Помилки 600, 601, 603, 606, 607, 610, 613, 614, 616, 618, 632, 635, 637, 638, 645. Перезавантажте комп'ютер. Якщо помилка не зникає, спробуйте переустановити VPN-з'єднання. Перевірте комп'ютер на предмет наявності вірусів. Видалите недавно встановлені програми, або скасуєте недавні зміни в налаштуваннях Windows, якщо такі були.

Помилки 604, 605, 608, 609, 615, 620, "Файл телефонної книги підсистеми віддаленого доступу Windows і поточна конфігурація Віддаленого Доступу до Мережі несумісні один з одним". Перезавантажте комп'ютер. Якщо помилка не зникає, то видаліть й заново створіть VPN з'єднання.

Помилки 611, 612, "Внутрішня конфігурація мережі Windows некоректно настроєна". Перезавантажте комп'ютер і переконайтесь, що локальна мережа нормально функціонує на комп'ютері. Якщо помилка не зникає, то зверніться в службу технічної підтримки. Крім того, помилка може бути викликана недостачею ресурсів (пам'яті) на комп'ютері. Спробуйте закрити запущені програми.

Помилка 617, "Windows перебуває в процесі підключення до Інтернету, або відбулася внутрішня помилка Windows". Почекайте кілька хвилин. Якщо

підключення не встановилося, і при повторному підключенні помилка повторюється, то перезавантажте комп'ютер.

Помилка 619, "Неправильно настроєні параметри безпеки VPN з'єднання, VPN-трафік блокується на шляху до шлюзу, або налаштування VPN не вступили в дію". У властивостях VPN з'єднання відкрийте вкладку "Безпека" – повинне бути обране "Звичайні (параметри, що рекомендуються)" і повинна бути знята галочка "Потрібно шифрування даних (інакше відключатися)". Перезавантажте комп'ютер і спробуйте підключитися ще раз. Перевірте налаштування брандмауера, і, якщо не впевнені в їхній правильності, відключіть його.

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 КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Розроблене програмне забезпечення захист інформації у Cloud-системах з використанням VPN складається з наступних функціональних блоків:

– Навігаційне меню: Налаштування Cloud-системи; Налаштування VPN; Довідка.

– Розділу вікна поточної інформації.

– Розділу вікна введення інформації: Назва Cloud-системи що використовується; Логін від Cloud-системи; Пароль від Cloud-системи; Назва VPN; Адреса VPN сервера; Логін від VPN; Пароль від VPN.

– Навігаційного меню.

– Функціональних кнопок ПЗ: Підключити; Відключити; Налаштувати; Журнал подій.

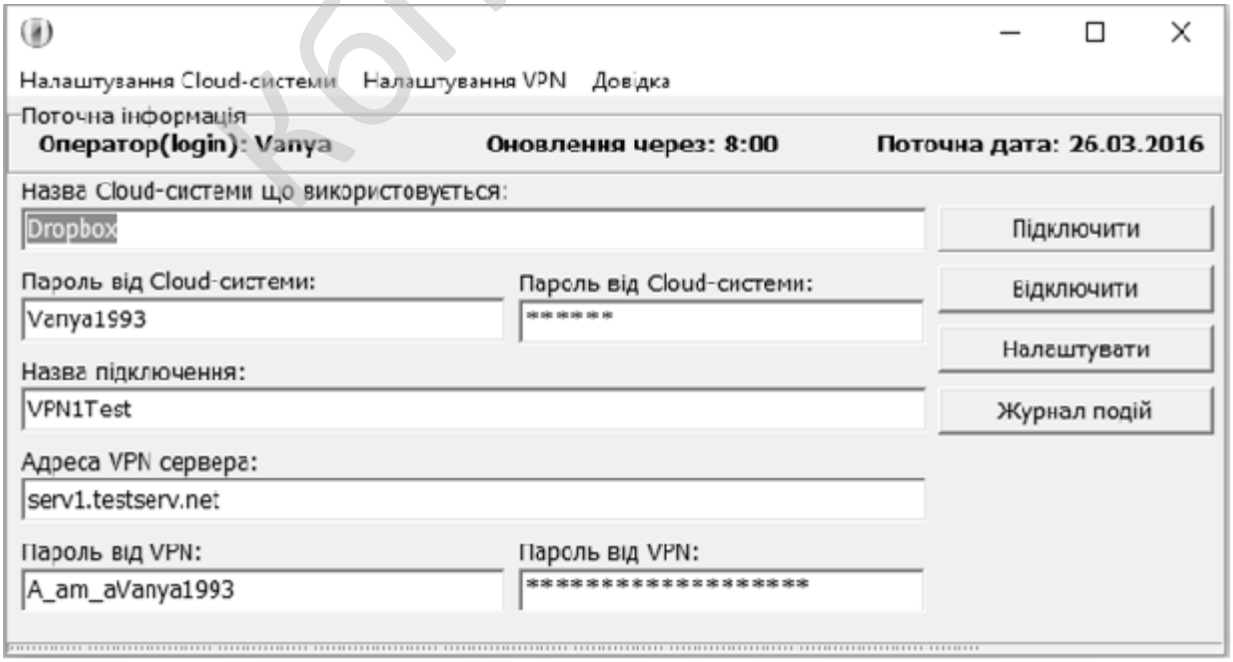

Рисунок 5.1 – Головне вікно розробленого ПЗ

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

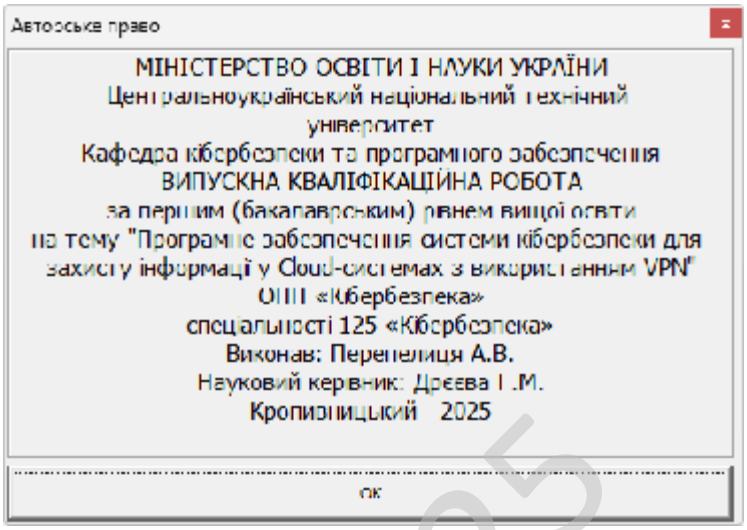

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Shareware.

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

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для захисту інформації у Cloud-системах з використанням VPN.

– Досліджена система для захисту інформації у Cloud-системах з використанням VPN.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для захисту інформації у Cloud-системах з використанням VPN.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для захисту інформації у Cloud-системах з використанням

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

Програма призначена для виконання під управлінням багатозадачної операційної системи 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. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

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

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

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

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

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

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

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

9. 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.

10. 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.

11. 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.

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

13. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

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

15. 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.

16. 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.

17. 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,

18. 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.

19. 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.

20. 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

21. 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.

22. 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.

23. 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.*

24. 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.

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

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

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

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

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

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

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

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

39. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

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

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

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

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

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

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

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

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

48. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 51-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 74 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

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

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Дрєєва Г.М.

Програмне забезпечення системи кібербезпеки для захисту інформації у Cloud-системах з використанням VPN

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import socket
import threading
import ssl
import time
import os
import json
import base64
import random
import string
import hashlib
import sqlite3
import logging
from datetime import datetime

# System configuration parameters
CONFIG = {
'HOST': '127.0.0.1',
'PORT': 9090,
'SSL_CERT': 'server.crt',
'SSL_KEY': 'server.key',
'DB_FILE': 'users.db',
'LOG_FILE': 'system.log',
'ENCRYPTION_KEY': 'supersecretkey98765',
'CLOUD_STORAGE_PATH': 'cloud_storage'
}

# This class provides simple encryption and decryption using XOR cipher with
base64 encoding
class SimpleEncryptor:
    def __init__(self, key):
        self.key = key
    def encrypt(self, data):
#Convert input data to bytes if it is a string
        if isinstance(data, str):
            data = data.encode('utf-8')
#Encode the key as bytes
        key_bytes = self.key.encode('utf-8')
#Create a bytearray for the encrypted data
        encrypted = bytearray()
#Perform XOR encryption for each byte in the data
        for i in range(len(data)):
            encrypted.append(data[i] ^ key_bytes[i % len(key_bytes)])
#Encode the encrypted bytes using base64 and return as string
        return base64.b64encode(encrypted).decode('utf-8')
    def decrypt(self, data):
#Decode the base64 encoded data to get the encrypted bytes
        encrypted_data = base64.b64decode(data.encode('utf-8'))
#Encode the key as bytes
        key_bytes = self.key.encode('utf-8')
#Create a bytearray for the decrypted data
        decrypted = bytearray()
#Perform XOR decryption for each byte in the encrypted data
        for i in range(len(encrypted_data)):
            decrypted.append(encrypted_data[i] ^ key_bytes[i % len(key_bytes)])
#Return the decrypted data as a utf-8 string
        return decrypted.decode('utf-8')

# This class is responsible for logging events, warnings, and errors in the
system
class SystemLogger:
    def __init__(self, log_file):
        self.log_file = log_file
#Configure the logging module with file, level, and format
        logging.basicConfig(filename=log_file, level=logging.DEBUG,
format='%(asctime)s - %(levelname)s - %(message)s')
    def log_event(self, message):
#Log a general event message
```

```
        logging.info(message)
    def log_warning(self, message):
#Log a warning message indicating potential issues
        logging.warning(message)
    def log_error(self, message):
#Log an error message indicating a problem occurred
        logging.error(message)
    def log_debug(self, message):
#Log a debug message for detailed troubleshooting
        logging.debug(message)

# This class manages user authentication using a SQLite database
class UserAuthenticator:
    def __init__(self, db_file):
        self.db_file = db_file
#Initialize the user database
        self._initialize_database()
    def _initialize_database(self):
#Connect to the SQLite database
        conn = sqlite3.connect(self.db_file)
        cursor = conn.cursor()
#Create a table for users if it does not exist
        cursor.execute('''CREATE TABLE IF NOT EXISTS users (username TEXT
PRIMARY KEY, password_hash TEXT)''')
        conn.commit()
        conn.close()
    def register_user(self, username, password):
#Create a SHA-256 hash of the user's password
        password_hash = hashlib.sha256(password.encode('utf-8')).hexdigest()
#Connect to the SQLite database to register the user
        conn = sqlite3.connect(self.db_file)
        cursor = conn.cursor()
        try:
#Insert the new user into the users table
            cursor.execute("INSERT INTO users (username, password_hash) VALUES
(?, ?)", (username, password_hash))
            conn.commit()
            result = True
        except sqlite3.IntegrityError:
#If the user already exists, registration fails
            result = False
        conn.close()
        return result
    def authenticate_user(self, username, password):
#Verify user credentials by comparing password hashes
        password_hash = hashlib.sha256(password.encode('utf-8')).hexdigest()
        conn = sqlite3.connect(self.db_file)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username=? AND
password_hash=?", (username, password_hash))
        user = cursor.fetchone()
        conn.close()
        return user is not None

# This class simulates cloud storage operations such as upload, download, and
listing files
class CloudStorage:
    def __init__(self, storage_path):
        self.storage_path = storage_path
#Create the storage directory if it does not exist
        if not os.path.exists(storage_path):
            os.makedirs(storage_path)
    def upload_file(self, filename, data):
#Construct the full file path for uploading
        file_path = os.path.join(self.storage_path, filename)
        try:
#Open the file in write mode and save the data
            with open(file_path, 'w') as f:
                f.write(data)
```

```
            return True
        except Exception as e:
            return False
    def download_file(self, filename):
#Construct the full file path for downloading
        file_path = os.path.join(self.storage_path, filename)
        if os.path.exists(file_path):
            try:
#Open the file in read mode and return its contents
                with open(file_path, 'r') as f:
                    content = f.read()
                return content
            except Exception as e:
                return None
        else:
            return None
    def list_files(self):
#Return a list of all files in the storage directory
        try:
            return os.listdir(self.storage_path)
        except Exception as e:
            return []

# This class simulates an intrusion detection system by scanning input data for
suspicious keywords
class IntrusionDetectionSystem:
    def __init__(self):
        self.activity_log = []
    def detect(self, data):
#Define a list of suspicious keywords to monitor
        suspicious_keywords = ['attack', 'malware', 'breach', 'unauthorized',
'exploit', 'injection']
#Check if any suspicious keyword is present in the input data
        for keyword in suspicious_keywords:
            if keyword in data.lower():
#Log the suspicious activity with a timestamp
                self.activity_log.append((datetime.now().isoformat(), data))
                return True
        return False
    def get_logs(self):
#Return the log of suspicious activities
        return self.activity_log

# This class manages VPN sessions, including session token generation and
validation
class SessionManager:
    def __init__(self):
        self.sessions = {}
    def generate_session_token(self, username):
#Generate a random session token using letters and digits
        token = ''.join(random.choice(string.ascii_letters + string.digits) for
_ in range(32))
#Associate the session token with the username and current timestamp
        self.sessions[token] = {'username': username, 'timestamp': time.time()}
        return token
    def validate_session(self, token):
#Validate if the session token exists and is not expired (simulate expiration
after 3600 seconds)
        session = self.sessions.get(token)
        if session and (time.time() - session['timestamp'] < 3600):
            return True
        else:
            return False
    def invalidate_session(self, token):
#Remove the session token from the session manager
        if token in self.sessions:
            del self.sessions[token]
```

```
# This class represents the VPN server which handles client connections,
authentication, and command processing
class VPNServer:
    def __init__(self, host, port, encryptor, authenticator, cloud_storage,
intrusion_system, session_manager, logger):
        self.host = host
        self.port = port
        self.encryptor = encryptor
        self.authenticator = authenticator
        self.cloud_storage = cloud_storage
        self.intrusion_system = intrusion_system
        self.session_manager = session_manager
        self.logger = logger
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server_socket.bind((host, port))
        self.server_socket.listen(5)
        self.clients = []
        self.running = True
    def start(self):
#Start the server to accept incoming client connections in a new thread
        threading.Thread(target=self._accept_clients, daemon=True).start()
        self.logger.log_event("VPN Server started on {}:{}".format(self.host,
self.port))
        threading.Thread(target=self._monitor_connections, daemon=True).start()
    def _accept_clients(self):
#Continuously accept new client connections while the server is running
        while self.running:
            try:
                client_socket, addr = self.server_socket.accept()
                self.logger.log_event("Accepted connection from
{}:{}".format(addr[0], addr[1]))
                client_thread = threading.Thread(target=self._handle_client,
args=(client_socket, addr), daemon=True)
                client_thread.start()
                self.clients.append((client_socket, addr))
            except Exception as e:
                self.logger.log_error("Error accepting clients:
{}".format(str(e)))
    def _handle_client(self, client_socket, addr):
#Handle the communication with an individual client
        try:
#Send username prompt to the client
            client_socket.send(b'Username: ')
#Receive the username from the client
            username = client_socket.recv(1024).decode('utf-8').strip()
#Send password prompt to the client
            client_socket.send(b'Password: ')
#Receive the password from the client
            password = client_socket.recv(1024).decode('utf-8').strip()
#Authenticate the user credentials
            if not self.authenticator.authenticate_user(username, password):
                client_socket.send(b'Authentication Failed.')
                self.logger.log_warning("Authentication failed for
{}:{}".format(addr[0], addr[1]))
                client_socket.close()
                return
#Generate a session token for the authenticated user
            session_token =
self.session_manager.generate_session_token(username)
#Send a successful authentication message along with the session token
            client_socket.send(("Authentication Successful. Session Token: " +
session_token + "\n").encode('utf-8'))
            self.logger.log_event("User {} authenticated from
{}:{}".format(username, addr[0], addr[1]))
            while True:
#Prompt the client for a command
                client_socket.send(b'Enter Command: ')
#Receive an encrypted command from the client
```

```
                encrypted_command = client_socket.recv(4096).decode('utf-
8').strip()
                if not encrypted_command:
                    break
#Decrypt the received command
                command = self.encryptor.decrypt(encrypted_command)
                self.logger.log_event("Received command from {}:
{}".format(username, command))
#Check if the command contains suspicious content using the intrusion detection
system
                if self.intrusion_system.detect(command):
                    client_socket.send(b'Intrusion Detected. Connection
Terminated.')
                    self.logger.log_warning("Intrusion detected from
{}:{}".format(addr[0], addr[1]))
                    break
#Process the command and generate a response
                response = self._process_command(command)
#Encrypt the response before sending it back to the client
                encrypted_response = self.encryptor.encrypt(response)
                client_socket.send(encrypted_response.encode('utf-8'))
        except Exception as e:
            self.logger.log_error("Error handling client {}: {}".format(addr,
str(e)))
        finally:
            client_socket.close()
            self.logger.log_event("Connection closed for {}:{}".format(addr[0],
addr[1]))
    def _process_command(self, command):
#Split the command into parts to identify the operation and its arguments
        parts = command.split()
        if len(parts) == 0:
            return "No command entered."
        cmd = parts[0].lower()
#Process the 'upload' command to store data in cloud storage
        if cmd == 'upload':
            if len(parts) < 3:
                return "Usage: upload <filename> <data>"
            filename = parts[1]
            data = " ".join(parts[2:])
            if self.cloud_storage.upload_file(filename, data):
                return "File uploaded successfully."
            else:
                return "File upload failed."
#Process the 'download' command to retrieve data from cloud storage
        elif cmd == 'download':
            if len(parts) < 2:
                return "Usage: download <filename>"
            filename = parts[1]
            file_data = self.cloud_storage.download_file(filename)
            if file_data is not None:
                return "File data: " + file_data
            else:
                return "File not found."
#Process the 'list' command to list all files in cloud storage
        elif cmd == 'list':
            files = self.cloud_storage.list_files()
            return "Files: " + ", ".join(files)
#Process the 'ping' command to check connectivity
        elif cmd == 'ping':
            return "pong"
#Process the 'status' command to get system status information
        elif cmd == 'status':
            return "Server running. Active connections: " +
str(len(self.clients))
#Process unknown commands with an error message
        else:
            return "Unknown command."
    def _monitor_connections(self):
```

```
#Periodically monitor active client connections and log the count
        while self.running:
            active_count = len(self.clients)
            self.logger.log_debug("Monitoring connections. Active clients: " +
str(active_count))
            time.sleep(5)
    def broadcast_message(self, message):
#Broadcast a message to all connected clients
        for client_socket, addr in self.clients:
            try:
                encrypted_message = self.encryptor.encrypt("Broadcast: " +
message)
                client_socket.send(encrypted_message.encode('utf-8'))
            except Exception as e:
                self.logger.log_error("Error broadcasting to {}:
{}".format(addr, str(e)))
    def shutdown(self):
#Shutdown the VPN server and close all connections
        self.running = False
        self.server_socket.close()
        for client_socket, addr in self.clients:
            try:
                client_socket.close()
            except:
                pass
        self.logger.log_event("VPN Server shutdown.")

# This class represents the VPN client which connects to the VPN server and
sends commands
class VPNClient:
    def __init__(self, host, port, encryptor, logger):
        self.host = host
        self.port = port
        self.encryptor = encryptor
        self.logger = logger
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.session_token = None
    def connect(self):
#Attempt to connect to the VPN server
        try:
            self.client_socket.connect((self.host, self.port))
            self.logger.log_event("Connected to VPN Server at
{}:{}".format(self.host, self.port))
#Handle authentication process upon connection
            self._authenticate()
        except Exception as e:
            self.logger.log_error("Failed to connect to VPN Server: " + str(e))
    def _authenticate(self):
#Receive username prompt from the server
        prompt = self.client_socket.recv(1024).decode('utf-8')
#If server requests username, send the username
        if "Username:" in prompt:
            self.client_socket.send(b'admin\n')
#Receive password prompt from the server
            prompt = self.client_socket.recv(1024).decode('utf-8')
            if "Password:" in prompt:
                self.client_socket.send(b'adminpass\n')
#Receive authentication confirmation along with session token
            response = self.client_socket.recv(1024).decode('utf-8')
            if "Authentication Successful" in response:
                parts = response.split("Session Token:")
                if len(parts) > 1:
                    self.session_token = parts[1].strip()
                self.logger.log_event("Authenticated successfully with session
token: " + str(self.session_token))
            else:
                self.logger.log_warning("Authentication failed on client side.")
    def send_command(self, command):
#Wait for command prompt from the server
```

```
        try:
            prompt = self.client_socket.recv(1024).decode('utf-8')
#Encrypt the command using the encryptor
            encrypted_command = self.encryptor.encrypt(command)
#Send the encrypted command to the server
            self.client_socket.send(encrypted_command.encode('utf-8'))
#Receive the encrypted response from the server
            response = self.client_socket.recv(4096).decode('utf-8')
#Decrypt the response to get the plain text
            decrypted_response = self.encryptor.decrypt(response)
            self.logger.log_event("Received response: " + decrypted_response)
            return decrypted_response
        except Exception as e:
            self.logger.log_error("Error sending command: " + str(e))
            return None
    def disconnect(self):
#Close the client socket connection
        self.client_socket.close()
        self.logger.log_event("Disconnected from VPN Server.")

# Additional utility functions for miscellaneous operations in the system
def generate_random_data(length=64):
#Generate a random string of specified length using letters and digits
    return ''.join(random.choice(string.ascii_letters + string.digits) for _ in
range(length))
def simulate_network_latency():
#Simulate network latency by sleeping for a random short duration
    time.sleep(random.uniform(0.1, 0.5))
def dummy_query_processor(query):
#Simulate processing of a query and return a dummy result
    simulate_network_latency()
    return "Result for query: " + query
def perform_bulk_upload(cloud_storage, num_files=10):
#Simulate bulk uploading of random files to the cloud storage
    for i in range(num_files):
        filename = "file_" + str(i) + ".txt"
        data = generate_random_data(128)
        cloud_storage.upload_file(filename, data)
        time.sleep(0.1)
def perform_bulk_download(cloud_storage):
#Simulate bulk downloading of all files from cloud storage
    files = cloud_storage.list_files()
    results = {}
    for file in files:
        data = cloud_storage.download_file(file)
        results[file] = data
        time.sleep(0.1)
    return results

# Main function to initialize and run the complete VPN and cloud security system
def main():
#Initialize system logger
    logger = SystemLogger(CONFIG['LOG_FILE'])
#Log the start of the system initialization process
    logger.log_event("Initializing Cybersecurity Cloud VPN System")
#Initialize the encryption module with the specified encryption key
    encryptor = SimpleEncryptor(CONFIG['ENCRYPTION_KEY'])
#Initialize the user authentication system with the database file
    authenticator = UserAuthenticator(CONFIG['DB_FILE'])
#Register the default administrative user with credentials
    authenticator.register_user('admin', 'adminpass')
#Initialize cloud storage module for file operations
    cloud_storage = CloudStorage(CONFIG['CLOUD_STORAGE_PATH'])
#Perform a bulk upload of dummy files to populate the cloud storage
    perform_bulk_upload(cloud_storage, num_files=5)
#Initialize the intrusion detection system for monitoring suspicious activities
    intrusion_system = IntrusionDetectionSystem()
#Initialize the session manager for VPN sessions
    session_manager = SessionManager()
```

```
#Initialize the VPN server with all required components
    vpn_server = VPNServer(CONFIG['HOST'], CONFIG['PORT'], encryptor,
authenticator, cloud_storage, intrusion_system, session_manager, logger)
#Start the VPN server to listen for incoming connections
    vpn_server.start()
#Wait a moment to ensure the server is fully started
    time.sleep(2)
#Initialize the VPN client to simulate a client connection to the server
    client = VPNClient(CONFIG['HOST'], CONFIG['PORT'], encryptor, logger)
#Connect the client to the VPN server
    client.connect()
#Wait briefly for the connection and authentication to complete
    time.sleep(1)
#Define a list of commands to be executed by the client
    commands = [
        'ping',
        'upload sample.txt This is a sample file uploaded by the client.',
        'list',
        'download sample.txt',
        'status',
        'upload alert.txt malware detected in system'
    ]
#Iterate over each command, send it to the server, and log the response
    for cmd in commands:
        result = client.send_command(cmd)
        time.sleep(1)
#Simulate additional queries processed by the client
    for i in range(3):
        query = "SELECT * FROM dummy_table WHERE id=" + str(i)
        result = dummy_query_processor(query)
        logger.log_event("Processed query result: " + result)
        time.sleep(0.5)
#Perform a bulk download of files from the cloud storage and log the results
    downloaded_files = perform_bulk_download(cloud_storage)
    logger.log_event("Bulk download completed with files: " +
str(list(downloaded_files.keys())))
#Disconnect the VPN client from the server
    client.disconnect()
#Shutdown the VPN server gracefully after operations
    vpn_server.shutdown()
#Log the completion of the system operations
    logger.log_event("Cybersecurity Cloud VPN System operations completed
successfully")
#End of main function

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

Файл ExtendedLogger.py

```
import os
import time
import hmac
import hashlib
import base64
import struct
import random
import logging
import sqlite3
import threading
import socket
from flask import Flask, request, redirect, url_for, render_template_string,
session
from datetime import datetime
from functools import wraps

class TwoFactorAuth:
    def __init__(self):
        self.interval = 30
    def generate_secret(self, length=16):
        chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ234567'
        return ''.join(random.choice(chars) for _ in range(length))
    def get_totp_token(self, secret, for_time=None, digits=6):
        if for_time is None:
            for_time = int(time.time())
        counter = int(for_time / self.interval)
        key = base64.b32decode(secret, True)
        msg = struct.pack(">Q", counter)
        hmac_hash = hmac.new(key, msg, hashlib.sha1).digest()
        offset = hmac_hash[-1] & 0x0F
        code = (struct.unpack(">I", hmac_hash[offset:offset+4])[0] & 0x7FFFFFFF)
% (10 ** digits)
        return str(code).zfill(digits)
    def verify_token(self, secret, token, allowed_drift=1, digits=6):
        current_time = int(time.time())
        for drift in range(-allowed_drift, allowed_drift+1):
            calc_time = current_time + (drift * self.interval)
            if self.get_totp_token(secret, for_time=calc_time, digits=digits) ==
token:
                return True
        return False

class ExtendedLogger:
    def __init__(self, log_file, db_file):
        self.log_file = log_file
        self.db_file = db_file
        logging.basicConfig(filename=self.log_file, level=logging.DEBUG,
format='%(asctime)s %(levelname)s %(message)s')
        self.conn = sqlite3.connect(self.db_file, check_same_thread=False)
        self._init_db()
        self.lock = threading.Lock()
```

```
    def _init_db(self):
        cur = self.conn.cursor()
        cur.execute("CREATE TABLE IF NOT EXISTS logs (id INTEGER PRIMARY KEY
AUTOINCREMENT, timestamp TEXT, level TEXT, message TEXT)")
        self.conn.commit()
    def log_event(self, message):
        self._log("INFO", message)
    def log_warning(self, message):
        self._log("WARNING", message)
    def log_error(self, message):
        self._log("ERROR", message)
    def _log(self, level, message):
        timestamp = datetime.now().isoformat()
        log_line = "{} {} {}\n".format(timestamp, level, message)
        with self.lock:
            with open(self.log_file, "a") as f:
                f.write(log_line)
            cur = self.conn.cursor()
            cur.execute("INSERT INTO logs (timestamp, level, message) VALUES (?,
?, ?)", (timestamp, level, message))
            self.conn.commit()
        if level == "INFO":
            logging.info(message)
        elif level == "WARNING":
            logging.warning(message)
        elif level == "ERROR":
            logging.error(message)
    def get_logs(self):
        cur = self.conn.cursor()
        cur.execute("SELECT timestamp, level, message FROM logs ORDER BY id
DESC")
        return cur.fetchall()

class NetworkTrafficMonitor:
    def __init__(self, logger):
        self.logger = logger
        self.running = False
        self.thread = None
        self.suspicious_keywords = ["attack", "breach", "malware", "exploit",
"injection"]
    def start_monitoring(self):
        self.running = True
        self.thread = threading.Thread(target=self._monitor_loop)
        self.thread.start()
    def stop_monitoring(self):
        self.running = False
        if self.thread:
            self.thread.join()
    def _monitor_loop(self):
        while self.running:
            packet = self._capture_packet()
            self._process_packet(packet)
            time.sleep(1)
```

```
    def _capture_packet(self):
        sample_packets = [
            "Normal traffic packet data",
            "This packet indicates a possible attack",
            "Routine data transfer",
            "Suspicious breach detected in packet",
            "Malware payload detected",
            "Regular update packet"
        ]
        return random.choice(sample_packets)
    def _process_packet(self, packet):
        for keyword in self.suspicious_keywords:
            if keyword in packet.lower():
                self.logger.log_warning("Suspicious packet detected: " + packet)
                break

class VulnerabilityScanner:
    def __init__(self, logger):
        self.logger = logger
    def scan_host(self, host, port_range):
        open_ports = []
        for port in port_range:
            try:
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                s.settimeout(0.5)
                result = s.connect_ex((host, port))
                if result == 0:
                    open_ports.append(port)
                s.close()
            except Exception as e:
                self.logger.log_error("Error scanning port {}: {}".format(port,
str(e)))
        self.logger.log_event("Scan complete on host {}: Open ports:
{}".format(host, open_ports))
        return open_ports

def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if not session.get("logged_in"):
            return redirect(url_for("login"))
        return f(*args, **kwargs)
    return decorated_function

class AdminInterface:
    def __init__(self, logger, scanner, tfa):
        self.logger = logger
        self.scanner = scanner
        self.tfa = tfa
        self.app = Flask(__name__)
        self.app.secret_key = os.urandom(24)
        self.admin_user = "admin"
        self.admin_pass = "password123"
```

```
        self.tfa_secret = self.tfa.generate_secret()
        self._setup_routes()
    def _setup_routes(self):
        self.app.add_url_rule("/", "index", self.index)
        self.app.add_url_rule("/login", "login", self.login, methods=["GET",
"POST"])
        self.app.add_url_rule("/dashboard", "dashboard", self.dashboard)
        self.app.add_url_rule("/scan", "scan", self.scan, methods=["GET",
"POST"])
    def index(self):
        if session.get("logged_in"):
            return redirect(url_for("dashboard"))
        return redirect(url_for("login"))
    def login(self):
        if request.method == "POST":
            username = request.form.get("username")
            password = request.form.get("password")
            token = request.form.get("token")
            if username == self.admin_user and password == self.admin_pass and
self.tfa.verify_token(self.tfa_secret, token):
                session["logged_in"] = True
                self.logger.log_event("Admin logged in successfully")
                return redirect(url_for("dashboard"))
            else:
                self.logger.log_warning("Failed admin login attempt")
                return "Login Failed"
        return render_template_string("""
<html>
<head>
<title>Admin Login</title>
</head>
<body>
<form method="post">
Username: <input type="text" name="username"/><br/>
Password: <input type="password" name="password"/><br/>
Token: <input type="text" name="token"/><br/>
<input type="submit" value="Login"/>
</form>
</body>
</html>
""")
    @login_required
    def dashboard(self):
        logs = self.logger.get_logs()
        log_entries = ""
        for log in logs:
            log_entries += "{} {} {}<br/>".format(log[0], log[1], log[2])
        return render_template_string("""
<html>
<head>
<title>Dashboard</title>
</head>
<body>
```

```
<h1>Admin Dashboard</h1>
<h2>System Logs</h2>
<div>{}</div>
<a href="/scan">Run Vulnerability Scan</a><br/>
<a href="/logout">Logout</a>
</body>
</html>
""".format(log_entries))
    @login_required
    def scan(self):
        if request.method == "POST":
            host = request.form.get("host")
            start_port = int(request.form.get("start_port"))
            end_port = int(request.form.get("end_port"))
            open_ports = self.scanner.scan_host(host, range(start_port,
end_port+1))
            result = "Open ports on {}: {}".format(host, open_ports)
            return render_template_string("""
<html>
<head>
<title>Scan Results</title>
</head>
<body>
<h1>Scan Results</h1>
<div>{}</div>
<a href="/dashboard">Back to Dashboard</a>
</body>
</html>
""".format(result))
        return render_template_string("""
<html>
<head>
<title>Vulnerability Scan</title>
</head>
<body>
<h1>Vulnerability Scanner</h1>
<form method="post">
Host: <input type="text" name="host"/><br/>
Start Port: <input type="text" name="start_port"/><br/>
End Port: <input type="text" name="end_port"/><br/>
<input type="submit" value="Scan"/>
</form>
<a href="/dashboard">Back to Dashboard</a>
</body>
</html>
""")
    def run(self, port=5000):
        self.app.add_url_rule("/logout", "logout", self.logout)
        self.app.run(port=port)
    def logout(self):
        session.clear()
        return redirect(url_for("login"))
```

```
if __name__ == "__main__":
    logger = ExtendedLogger("extended_system.log", "extended_logs.db")
    tfa = TwoFactorAuth()
    scanner = VulnerabilityScanner(logger)
    net_monitor = NetworkTrafficMonitor(logger)
    admin_interface = AdminInterface(logger, scanner, tfa)
    net_monitor.start_monitoring()
    admin_thread = threading.Thread(target=lambda:
admin_interface.run(port=5000))
    admin_thread.start()
    while True:
        time.sleep(10)
```

Файл SessionRecoveryManager.py

```
import os
import time
import threading
import random
import json
import shutil
import logging
from datetime import datetime

class LDAPConnector:
    def __init__(self, server, port):
        self.server = server
        self.port = port
        self.connected = False
        self.users = {}
        self.lock = threading.Lock()
    def connect(self):
        time.sleep(0.5)
        self.connected = True
        return self.connected
    def disconnect(self):
        self.connected = False
    def add_user(self, dn, attributes):
        with self.lock:
            if dn in self.users:
                return False
            self.users[dn] = attributes
            return True
    def delete_user(self, dn):
        with self.lock:
            if dn in self.users:
                del self.users[dn]
                return True
            return False
    def update_user(self, dn, attributes):
        with self.lock:
            if dn in self.users:
                self.users[dn].update(attributes)
                return True
            return False
    def search_user(self, search_filter):
        results = []
        with self.lock:
            for dn, attrs in self.users.items():
                match = True
                for key, value in search_filter.items():
                    if key not in attrs or attrs[key] != value:
                        match = False
                        break
                if match:
                    results.append((dn, attrs))
```

```
        return results
    def authenticate(self, dn, password):
        with self.lock:
            if dn in self.users and 'password' in self.users[dn]:
                return self.users[dn]['password'] == password
            return False

class AutoScaler:
    def __init__(self):
        self.servers = []
        self.lock = threading.Lock()
        self.cpu_threshold = 70
        self.scale_factor = 2
        self.monitor_interval = 2
        self.scaling_thread = threading.Thread(target=self.scaling_loop)
        self.scaling_thread.daemon = True
        self.running = False
    def add_server(self, server_id):
        with self.lock:
            self.servers.append({'id': server_id, 'cpu_usage':
random.randint(10, 50)})
    def remove_server(self):
        with self.lock:
            if self.servers:
                self.servers.pop()
    def simulate_load(self):
        with self.lock:
            for server in self.servers:
                server['cpu_usage'] = random.randint(10, 100)
    def get_average_cpu(self):
        with self.lock:
            if not self.servers:
                return 0
            total = sum(s['cpu_usage'] for s in self.servers)
            return total / len(self.servers)
    def scale_up(self):
        with self.lock:
            current_count = len(self.servers)
            for i in range(self.scale_factor):
                self.servers.append({'id': 'server_{}'.format(current_count + i
+ 1), 'cpu_usage': random.randint(10, 50)})
    def scale_down(self):
        with self.lock:
            if len(self.servers) > self.scale_factor:
                for i in range(self.scale_factor):
                    self.servers.pop()
    def scaling_loop(self):
        while self.running:
            self.simulate_load()
            avg_cpu = self.get_average_cpu()
            if avg_cpu > self.cpu_threshold:
                self.scale_up()
            elif avg_cpu < 30 and len(self.servers) > 1:
```

```
                self.scale_down()
            time.sleep(self.monitor_interval)
    def start(self):
        self.running = True
        self.scaling_thread.start()
    def stop(self):
        self.running = False
        self.scaling_thread.join()
    def get_server_status(self):
        with self.lock:
            return [(s['id'], s['cpu_usage']) for s in self.servers]

class SessionRecoveryManager:
    def __init__(self, recovery_file):
        self.recovery_file = recovery_file
        self.sessions = {}
        self.lock = threading.Lock()
        self.recovery_interval = 3
        self.recovery_thread = threading.Thread(target=self.recovery_loop)
        self.recovery_thread.daemon = True
        self.running = False
    def add_session(self, session_id, data):
        with self.lock:
            self.sessions[session_id] = data
    def remove_session(self, session_id):
        with self.lock:
            if session_id in self.sessions:
                del self.sessions[session_id]
    def get_sessions(self):
        with self.lock:
            return dict(self.sessions)
    def save_sessions(self):
        with self.lock:
            with open(self.recovery_file, 'w') as f:
                json.dump(self.sessions, f)
    def load_sessions(self):
        if os.path.exists(self.recovery_file):
            with open(self.recovery_file, 'r') as f:
                try:
                    self.sessions = json.load(f)
                except:
                    self.sessions = {}
    def recovery_loop(self):
        while self.running:
            self.save_sessions()
            time.sleep(self.recovery_interval)
    def start(self):
        self.load_sessions()
        self.running = True
        self.recovery_thread.start()
    def stop(self):
        self.running = False
        self.recovery_thread.join()
```

```
        self.save_sessions()

class BackupManager:
    def __init__(self, source_dir, backup_dir):
        self.source_dir = source_dir
        self.backup_dir = backup_dir
        self.backup_interval = 5
        self.backup_thread = threading.Thread(target=self.backup_loop)
        self.backup_thread.daemon = True
        self.running = False
    def perform_backup(self):
        if not os.path.exists(self.backup_dir):
            os.makedirs(self.backup_dir)
        for root, dirs, files in os.walk(self.source_dir):
            rel_path = os.path.relpath(root, self.source_dir)
            dest_dir = os.path.join(self.backup_dir, rel_path)
            if not os.path.exists(dest_dir):
                os.makedirs(dest_dir)
            for file in files:
                src_file = os.path.join(root, file)
                dest_file = os.path.join(dest_dir, file)
                try:
                    shutil.copy2(src_file, dest_file)
                except:
                    pass
    def backup_loop(self):
        while self.running:
            self.perform_backup()
            time.sleep(self.backup_interval)
    def start(self):
        self.running = True
        self.backup_thread.start()
    def stop(self):
        self.running = False
        self.backup_thread.join()
    def restore_backup(self, restore_dir):
        if os.path.exists(self.backup_dir):
            for root, dirs, files in os.walk(self.backup_dir):
                rel_path = os.path.relpath(root, self.backup_dir)
                dest_dir = os.path.join(restore_dir, rel_path)
                if not os.path.exists(dest_dir):
                    os.makedirs(dest_dir)
                for file in files:
                    src_file = os.path.join(root, file)
                    dest_file = os.path.join(dest_dir, file)
                    try:
                        shutil.copy2(src_file, dest_file)
                    except:
                        pass

class AccessPolicyEngine:
    def __init__(self):
        self.roles = {}
```

```
        self.user_roles = {}
        self.lock = threading.Lock()
    def add_role(self, role, permissions):
        with self.lock:
            self.roles[role] = set(permissions)
    def remove_role(self, role):
        with self.lock:
            if role in self.roles:
                del self.roles[role]
            for user, roles in self.user_roles.items():
                if role in roles:
                    roles.remove(role)
    def assign_role_to_user(self, user, role):
        with self.lock:
            if user not in self.user_roles:
                self.user_roles[user] = set()
            if role in self.roles:
                self.user_roles[user].add(role)
    def revoke_role_from_user(self, user, role):
        with self.lock:
            if user in self.user_roles and role in self.user_roles[user]:
                self.user_roles[user].remove(role)
    def add_permission_to_role(self, role, permission):
        with self.lock:
            if role in self.roles:
                self.roles[role].add(permission)
    def remove_permission_from_role(self, role, permission):
        with self.lock:
            if role in self.roles and permission in self.roles[role]:
                self.roles[role].remove(permission)
    def check_permission(self, user, permission):
        with self.lock:
            if user not in self.user_roles:
                return False
            for role in self.user_roles[user]:
                if permission in self.roles.get(role, []):
                    return True
            return False
    def get_user_permissions(self, user):
        with self.lock:
            perms = set()
            for role in self.user_roles.get(user, []):
                perms = perms.union(self.roles.get(role, set()))
            return perms
    def list_roles(self):
        with self.lock:
            return dict(self.roles)
    def list_user_roles(self):
        with self.lock:
            return dict(self.user_roles)

def main():
    ldap = LDAPConnector("ldap.example.com", 389)
```

```
    ldap.connect()
    ldap.add_user("cn=admin,dc=example,dc=com", {"password": "adminpass",
"email": "admin@example.com"})
    ldap.add_user("cn=user1,dc=example,dc=com", {"password": "user1pass",
"email": "user1@example.com"})
    result = ldap.search_user({"email": "user1@example.com"})
    autoscaler = AutoScaler()
    autoscaler.add_server("server_1")
    autoscaler.add_server("server_2")
    autoscaler.start()
    session_manager = SessionRecoveryManager("sessions.json")
    session_manager.start()
    session_manager.add_session("sess1", {"user": "admin", "created":
datetime.now().isoformat()})
    backup_manager = BackupManager("data_source", "data_backup")
    backup_manager.start()
    access_policy = AccessPolicyEngine()
    access_policy.add_role("admin", ["read", "write", "delete"])
    access_policy.add_role("user", ["read"])
    access_policy.assign_role_to_user("admin", "admin")
    access_policy.assign_role_to_user("user1", "user")
    for i in range(10):
        session_manager.add_session("sess" + str(i+2), {"user": "user1",
"created": datetime.now().isoformat()})
        time.sleep(0.2)
    for i in range(5):
        autoscaler.simulate_load()
        time.sleep(1)
    sessions = session_manager.get_sessions()
    server_status = autoscaler.get_server_status()
    policies = access_policy.list_user_roles()
    autoscaler.stop()
    session_manager.stop()
    backup_manager.stop()

if __name__ == "__main__":
    logging.basicConfig(filename="system_extra.log", level=logging.DEBUG,
format="%(asctime)s %(levelname)s %(message)s")
    main()
```