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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки побудови VPN-мережі”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Поліщук А.О.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2023 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 2023 року

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

Поліщуку Андрію Олексійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки побудови VPN-мережі*

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

затверджені наказом вищого навчального закладу № 12-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки побудови VPN-мережі*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи 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 2023 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	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

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

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Поліщук А.О.
(прізвище та ініціали)

АНОТАЦІЯ

Поліщук А.О. Програмне забезпечення системи кібербезпеки побудови VPN-мережі. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки побудови VPN-мережі.

Метою розробки є програмне забезпечення системи кібербезпеки побудови VPN-мережі.

Результат роботи – програмна реалізація системи кібербезпеки побудови VPN-мережі.

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

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

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

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: кібербезпека, VPN

ABSTRACT

Polishchuk A.O. VPN network building cyber security system software. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of building a VPN network.

The purpose of the development is the software of the cyber security system for building a VPN network.

The result of the work is the software implementation of the cyber security system for building a VPN network.

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

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10.4 environment.

Keywords: cyber security, 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. Сучасний розвиток інформаційних технологій і, зокрема, мережі Internet, приводить до необхідності захисту інформації, переданої в рамках розподіленої корпоративної мережі, що використовує мережі відкритого доступу. Не кожна компанія може собі дозволити мати власні фізичні канали доступу, і тут допоможе технологія VPN, що забезпечує достатню гнучкість і одночасно високу безпеку мережі, а також істотну економію витрат

Більше кращою альтернативою є створення віртуальної приватної мережі (VPN – Virtual Private Network) на базі загальнодоступної Internet. Одним з головних достоїнств Internet є те, що вона широкодоступна. Однак, зв'язок через Internet має свої недоліки, головним з яких є те, що він підданий потенційним порушенням захисту й конфіденційності. Використовуючи Internet як розширення власної внутрикорпоративної мережі, ви посилаєте інформацію із загальнодоступних каналів, і всякий, хто може встановити на її шляху аналізатор протоколів, має потенційну можливість перехопити вашу інформацію. І тут на допомогу приходить технологія VPN. Віртуальні приватні мережі можуть гарантувати, що направляємий через Internet трафік так само захищений, як і передачі усередині локальної мережі, при збереженні всіх фінансових переваг, які можна одержати, використовуючи Internet. Сучасний стан технології VPN дозволяє забезпечити достатню гнучкість на випадок майбутнього розширення мережі при збереженні високої надійності й безпеки. А головне, віртуальні мережі забезпечують істотну економію витрат у порівнянні зі змістом власної мережі глобального масштабу. Однак при цьому треба пам'ятати, що впровадження рішень на основі VPN може привести до зниження продуктивності й зажадати значних початкових витрат. Тому рішення питання про вибір VPN або альтернативного рішення вимагає ретельного аналізу.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки побудови VPN-мережі.

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

– Огляд існуючих систем побудови VPN-мережі.

– Дослідження системи кібербезпеки побудови VPN-мережі.

– Програмна реалізація системи кібербезпеки побудови VPN-мережі.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі побудови VPN-мережі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки побудови VPN-мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для побудови VPN-мережі. Тепер розглянемо принцип роботи цієї технології. VPN-пристрій розташовується між внутрішньою мережею й Internet на кожному кінці з'єднання. Коли ви передаєте дані через VPN, вони зникають "з поверхні" у точці відправлення й знову з'являються тільки в точці призначення. Цей процес прийнятий називати "тунелювання". Як можна догадатися з назви, це означає створення логічного тунелю в мережі Internet, що з'єднує дві крайні точки. Завдяки тунелюванню приватна інформація стає невидимої для інших користувачів Web. Перш ніж потрапити в Internet-тунель, дані ще й шифруються, що забезпечує їхній додатковий захист. Протоколи шифрування бувають різні. Все залежить від того, який протокол тунелювання підтримується тим або іншим VPN-рішенням. IPsec підтримує самий широкий спектр стандартів шифрування, включаючи DES (Data Encryption Standard) і Triple DES. Ще однією важливою характеристикою VPN-рішень є діапазон підтримуваних протоколів автентифікації. Більшість популярних продуктів працюють зі стандартами, заснованими на використанні відкритого ключа, такими як X.509. Це означає, що, підсиливши свою віртуальну приватну мережу відповідним протоколом автентифікації, ви зможете гарантувати, що доступ до ваших захищених тунелів одержать тільки відомі вам люди.

Віртуальні приватні мережі часто використовуються в сполученні з міжмережевими екранами. Адже VPN забезпечує захист корпоративних даних тільки під час їхнього руху по Internet і не може захистити внутрішню мережу від проникнення зловмисників.

Варіанти побудови VPN

Можна виділити чотири основних варіанти побудови мережі VPN, які використовуються в усьому світі. Дана класифікація пропонується компанією Check Point Software Technologies, що не без підстави вважається законодавцем моди в області VPN.

Варіант "Intranet VPN". Дозволяє об'єднати в єдину захищену мережу кілька розподілених філій однієї організації, взаємодіючих по відкритих каналах зв'язку. Саме цей варіант одержав широке поширення в усьому світі, і саме його в першу чергу реалізують компанії-розроблювачі.

Варіант "Remote Access VPN". Реалізує захищену взаємодію між сегментом корпоративної мережі (центральним офісом або філією) і одиночним користувачем, що підключається до корпоративних ресурсів з будинку (домашній користувач) або через ноутбук (мобільний користувач). Даний варіант відрізняється від першого тем, що віддалений користувач, як правило, не має статичної адреси, і він підключається до захищаємого ресурсу, не через виділений пристрій VPN, а прямо зі свого власного комп'ютера, на якому й установлюється програмне забезпечення, що реалізує функції VPN. Компонент VPN для віддаленого користувача може бути виконаний як у програмному, так і в програмно-апаратному виді. У першому випадку програмне забезпечення може бути як убудованим в операційну систему (наприклад, в Windows 10/11), так і розробленим спеціально. У другому випадку для реалізації VPN використовуються невеликі пристрої класу SOHO, які не вимагають серйозного настроювання й можуть бути використані навіть некваліфікованим персоналом. Такі пристрої одержують зараз широке поширення.

Варіант "Client/Server VPN". Він забезпечує захист переданих даних між двома вузлами (не мережами) корпоративної мережі. Особливість даного варіанта в тому, що VPN будується між вузлами, що перебувають, як правило, в одному сегменті мережі, наприклад, між робочою станцією й сервером. Така необхідність дуже часто виникає в тих випадках, коли в одній фізичній мережі необхідно

створити кілька логічних мереж. Наприклад, коли треба розділити трафік між фінансовим департаментом і відділом кадрів, що звертаються до серверів, що перебуває в одному фізичному сегменті. Цей варіант схожий на технологію VLAN, але замість поділу трафіку, використовується його шифрування.

Останній варіант "Extranet VPN" призначений для тих мереж, до яких підключаються так звані користувачі "з боку" (партнери, замовники, клієнти й т.д.), рівень довіри до яких набагато нижче, ніж до своїх співробітників. Хоча по статистиці найчастіше саме співробітники є причиною комп'ютерних злочинів і зловживань.

1.2 Область застосування

Розглянемо область застосування VPN-мережі. Всі продукти для створення VPN можна умовно розділити на дві категорії: програмні й апаратні. Програмне рішення для VPN – це, як правило, готовий додаток, що встановлюється на підключеному до мережі окремому комп'ютері. Ряд виробників, такі як компанії Axent Technologies, Check Point Software Technologies і NetGuard, поставляють VPN-пакети, які легко інтегруються із програмними міжмережевими екранами й працюють на різних операційних системах, включаючи Windows 10/11, Sun Solaris і Linux. Оскільки для побудови VPN на базі спеціалізованого програмного забезпечення потрібне створення окремої комп'ютерної системи, такі рішення звичайно складніше для розгортання, ніж апаратні. Створення подібної системи передбачає конфігурування сервера для розпізнавання даного комп'ютера і його операційної системи, VPN-пакета, мережевих плат для кожного з'єднання й спеціальних плат для прискорення операцій шифрування. Така робота в ряді випадків може виявитися скрутної навіть для досвідчених фахівців. З іншого боку, програмні рішення для VPN коштують відносно недорого.

На відміну від них апаратні VPN-рішення містять у собі все, що необхідно для з'єднання, – комп'ютер, приватну (як правило) операційну систему й спеціальне програмне забезпечення. Ряд компаній, у тому числі Cisco Systems, NetScreen і Sonic, пропонують цілий спектр рішень, які можуть масштабуватися залежно від кількості одночасних VPN-з'єднань, з якими передбачається працювати, і очікуваного обсягу трафіка. Розгортати апаратні рішення, безумовно, легше. Вони містять у собі все, що необхідно для конкретних умов, тому час, за яке їх можна запустити, обчислюється мінутами або годинниками. Ще однією серйозною перевагою апаратних VPN-рішень є набагато більше висока продуктивність. До мінусів апаратних VPN-рішень можна віднести їхню високу вартість. Ще один недолік таких рішень полягає в тому, що управляються вони окремо від інших рішень по безпеці, що ускладнює завдання адміністрування інфраструктури безпеки, особливо за умови недостачі співробітників відділу захисту інформації.

Існують також інтегровані рішення, у яких функції побудови VPN реалізуються поряд з функцією фільтрації мережевого трафіка, забезпечення якості обслуговування або розподіли смуги пропущення. Основна перевага такого рішення – централізоване керування всіма компонентами з єдиної консолі. Друга перевага – більш низька вартість розраховуючи на кожний компонент у порівнянні із ситуацією, коли такі компоненти здобуваються окремо. Прикладом такого інтегрованого рішення може служити VPN-1 від компанії Check Point Software, що включає в себе крім VPN-модуля, модуль, що реалізує функції міжмережевого екрана, модуль, відповідальний за балансування навантаження, розподіл смуги пропущення й т.д.

Яке рішення найкраще підходить тій або іншій організації? Вибір визначається трьома факторами: розмір мережі, технічні навички, якими володіють співробітники організації, і обсяг трафіка, що планується обробляти. Процес шифрування даних вимагає істотних обчислювальних ресурсів і може перевантажити комп'ютер, коли декілька VPN-з'єднань одночасно беруть участь у

передачі даних. У цьому випадку, щоб розвантажити центральний процесор, можливо, прийде встановити спеціальні пришвидчуючі плати.

Який би шлях не був обраний, однаково прийде зштовхнутися із проблемою керування VPN-пристроями й підтримки погоджених правил безпеки для VPN і міжмережевих екранів у масштабах всієї організації. Якщо співробітники не мають достатні навички в цій області, можна довірити створення віртуальної приватної мережі незалежної компанії, що робить відповідні послуги.

Слід також зазначити, що використання VPN не є приводом для відмови від спеціалізованих засобів безпеки. По статистиці, до 80% всіх інцидентів, пов'язаних з інформаційною безпекою, відбувається з вини авторизованих користувачів, що мають санкціонований доступ у корпоративну мережу, а це значить, що атака або вірус від такого користувача будуть зашифровані й передані нарівні з необразливим трафіком.

І необхідно згадати ще одну особливість VPN – використання цієї технології знижує продуктивність мережі, що обумовлено затримками встановлення захищеного з'єднання між VPN-пристроями, затримками шифрування даних, затримками контролю їхньої цілісності й збільшеним трафіком через використання більше довгих заголовків пакетів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки побудови VPN-мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Розглянемо основні програмні продукти які дозволяють побудувати VPN мережу.

Hamachi

Hamachi – програмне забезпечення, призначене для побудови VPN. Hamachi дозволяє створити власну захищену мережу з комп'ютерів, з'єднаних через інтернет, начебто вони з'єднані однією фізичною локальною мережею.

Hamachi дозволяє створити локальну мережу (LAN) поверх Інтернету. Найчастіше Hamachi-мережі використовуються для з'єднання серверів із сірим IP і клієнтських комп'ютерів. До речі, такий метод значно ускладнює перехоплення пакетів.

Будь-які додатки, які працюють через локальну мережу, можуть працювати через мережі Hamachi, при цьому передані дані будуть захищені, і обмін між ними здійснюється в стилі peer-to-peer.

Hamachi – система організації віртуальних захищених мереж на основі протоколу UDP. У такій мережі вузли для встановлення з'єднання між собою використовують третій вузол, що допомагає їм лише виявити один одного, а передача інформації виробляється безпосередньо між вузлами. При цьому взаємодіючі вузли можуть перебувати за NAT або фаєрволом.

Hamachi використовується геймерами для гри в старих іграх по інтернету через VPN, оскільки офіційні сервери гри в інтернеті закриті (напр. Red Alert 2). Також Hamachi використовується геймерами для гри на піратських копіях ігор,

щоб обходити систему захисту гри, що вимагає унікальний CD-ключ для гри через інтернет.

OpenVPN

OpenVPN – вільна реалізація технології Віртуальної Приватної Мережі (VPN) з відкритим вихідним кодом для створення зашифрованих каналів типу точка-точка або сервери-клієнти між комп'ютерами. Вона дозволяє встановлювати з'єднання між комп'ютерами, що перебувають за NAT-firewall, без необхідності зміни їхніх настроювань. OpenVPN була створена Джеймсом Йонаном (James Yonan) і поширюється під ліцензією GNU GPL.

Для забезпечення безпеки керуючого каналу й потоку даних, OpenVPN використовує бібліотеку OpenSSL. Завдяки цьому задіється весь набір шифрів, доступних у даній бібліотеці. Також може використовуватися пакетна авторизація HMAC, для забезпечення більшої безпеки, і апаратне прискорення для поліпшення продуктивності шифрування. Ця бібліотека використовує OpenSSL, а точніше протоколи SSLv3/TLSv1. OpenVPN використовується на Solaris, OpenBSD, FreeBSD, NetBSD, GNU/Linux, Apple Mac OS X, QNX і Microsoft Windows.

OpenVPN пропонує користувачеві кілька видів автентифікаціи:

– Передвстановлений ключ – найпростіший метод.

– Сертифікатна автентифікація – найбільш гнучкий у настроюваннях метод.

– За допомогою логіна й пароля – може використовуватися без створення клієнтського сертифіката (серверний сертифікат однаково потрібний).

OpenVPN проводить всі мережеві операції через TCP, або UDP порт (при виборі протоколу існують дві точки зору одна Why TCP Over TCP Is A Bad Idea і інша Tcp Over TCP Is Not So Bad-web). Також можлива робота через більшу частину проксі серверів, включаючи HTTP, через NAT і мережеві фільтри. Сервер може бути настроєний на призначення мережевих настроювань клієнтові. Наприклад: IP адреса, настроювання маршрутизації й параметри з'єднання.

OpenVPN пропонує два різних варіанти мережевих інтерфейсів, використовуючи драйвер TUN/TAP. Можливо створити Layer 3-based IP тунель, називаний TUN, і Layer 2-based Ethernet – TAP, здатний передавати Ethernet трафік. Також можливе використання бібліотеки компресії LZO, для стиску потоку даних. Використовуваний порт 1194 виділений Internet Assigned Numbers Authority для роботи даної програми. Версія 2.0 дозволяє контролювати кілька одночасних тунелів, на відміну від версії 1.0, що дозволяла створювати тільки 1 тунель на 1 процес.

Використання в OpenVPN стандартних протоколів TCP і UDP дозволяє йому стати альтернативою IPsec у ситуаціях, коли Інтернет-провайдер блокує деякі VPN протоколи.

Vyatta

Vyatta – мережева операційна система, заснована на Debian GNU/Linux. Працює на встаткуванні x86 і дозволяє використовувати звичайний персональний комп'ютер або сервер як маршрутизатор, міжмережевого екрана або VPN-концентратора. Vyatta також може працювати у віртуальній машині, надаючи традиційні мережеві сервіси для віртуальної інфраструктури (офіційно підтримуються VMware ESX Server і CitrixXenServer, теоретично може працювати в будь-якому гіпервізорі).

Розроблювачем є компанія Vyatta Inc., розташована в місті Белмонт, Каліфорнія. Вона позиціонує Vyatta як конкурента продуктам Cisco рівня ISR 1800 – VXR 7200.

Vyatta Inc. також надає послуги по технічній підтримці своїх продуктів, консультації по їхньому настроюванню й продає апаратне забезпечення із передвстановленою Vyatta.

За заявою розроблювачів, назва продукту походить від санскритського слова «відкритий».

Ключові можливості:

– Статична й динамічна маршрутизація для IPv4 і IPv6 (RIP, OSPF, BGP, RIPng).

– Міжмережеве екранування для IPv4 і IPv6, включаючи фільтрацію p2p-трафіка.

– Трансляція мережевих адрес.

– DHCP-сервер для IPv4 і IPv6.

– Система виявлення вторгнень.

– Балансування навантаження й резервування каналу.

– Резервування маршрутизаторів із синхронізацією таблиці станів з'єднань.

– Віртуальні приватні мережі (IPsec, L2TP/IPsec, PPTP, OpenVPN).

– Облік трафіка (Netflow і sFlow).

– Веб-проксі й фільтрація URL.

У цей час існують дві версії: Vyatta Core, що містить тільки відкриті компоненти й поширюється безкоштовно, і Vyatta Subscription/Vyatta Plus, що містить додаткові (у тому числі пропрієтарні) компоненти й доступна тільки платним передплатникам.

Серед додаткових можливостей Vyatta Subscription:

– Синхронізація настроювань між маршрутизаторами.

– REST API для віддаленого керування.

– Підтримка інтерфейсниих карт PDH (E1, T1/T3) і V.35.

– Комерційний розширений набір правил для фільтрації URL і IPS.

До версії 6.0 Core і Subscription були незалежними періодично синхронізуємими вітками коду, після цієї версії вони були об'єднані, і Subscription являє собою строгу надмножину Core.

Vyatta Core доступна для безкоштовного завантаження із сайті розроблювачів у вигляді LiveCD (з можливістю установки на жорсткий диск), LiveCD для віртуальних машин (містить ядро для Xen), а також готових образів віртуальних машин VMWare і XenServer і хмарного сервісу Riverbed.

Настроювання й керування системою виробляються через єдиний інтерфейс командного рядка, що нагадує інтерфейс Juniper JUNOS. Також доступний веб-інтерфейс; за замовчуванням він відключений.

Інтерфейс командного рядка підтримує автодоповнення команд по натисканню клавіші Tab і вивід довідки по натисканню клавіші «?».

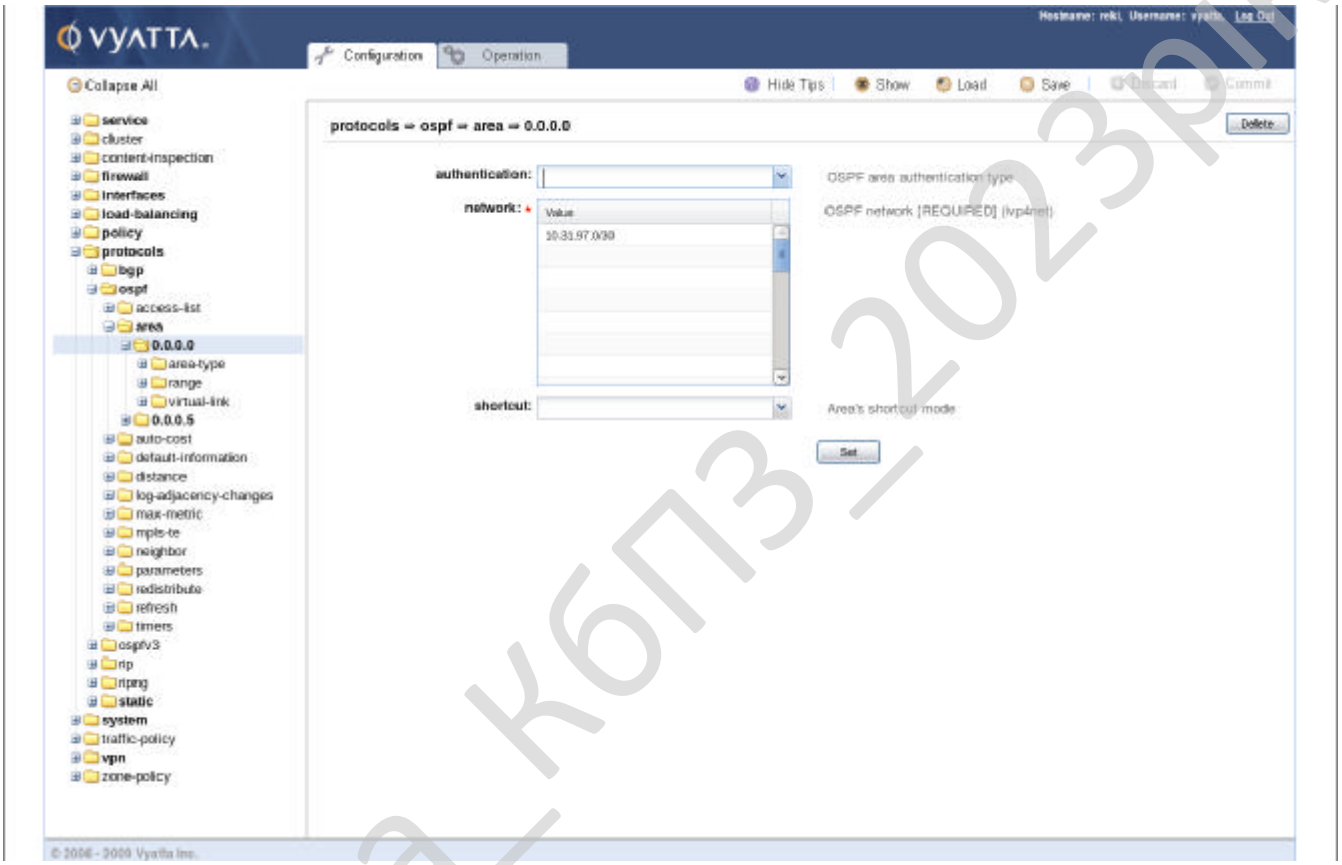

Рисунок 2.1 – Інтерфейс користувача Vyatta

Інтерфейс має два режими: режим операцій (для перегляду параметрів і обслуговування системи) і режим настроювань. Перехід у режим настроювань здійснюється командою configure.

Всі настроювання зберігаються в одному файлі й мають ієрархічну структуру. Рівні вкладеності відзначаються фігурними дужками.

Команди для зміни настроювань містять у собі назва операції (наприклад set або delete), шлях до потрібного вузла конфігурації й значення опції.

Також можна перейти на потрібний рівень вкладеності командою edit (наприклад, «edit policy prefix-list RFC1918») і використовувати відносні шляхи («set rule 20 prefix 192.168.0.0/16»). У цьому режимі можна перейменовувати або копіювати вузли настроювань за допомогою команд rename і copy (наприклад, «rename rule 20 to rule 30»).

Настроювання не застосовуються відразу після уведення команд (на відміну від, наприклад, Cisco IOS). Щоб вони ввійшли в силу, потрібно виконати команду commit. Це дозволяє перевірити внесені зміни й зменшити ймовірність помилки. Відмовитися від внесених змін можна командою discard.

CiscoWorks VPN / Security Management Solution

Складається з наступних програмних компонентів:

– Management Center for Cisco PIX® Firewalls – інструмент для комплексного керування міжмережевыми екранами Cisco PIX Firewall.

– Management Center for IDS Sensors – інструмент для настроювання систем виявлення несанкціонованого доступу Cisco IDS Sensor і Cisco Catalyst 6000 IDS Module.

– Management Center for VPN Routers – інструмент для настроювання й впровадження віртуальних приватних мереж.

– Monitoring Center for Security – універсальний інструмент для збору, перегляду, кореляції й генерації звітності по подіях від систем виявлення несанкціонованого доступу, міжмережевих екранів і маршрутизаторів Cisco.

– IDS Host Sensor Console – консоль керування серверними агентами виявлення несанкціонованого доступу.

– Auto Update Server – сервер, що забезпечує автоматичне відновлення програмного забезпечення на пристроях безпеки.

– Resource Manager Essentials – набір засобів для пошуку й усунення несправностей у мережі, збору деталізованих звітів, централізованого контролю й відновлення програмного забезпечення й конфігурацій пристроїв, керування й конфігурації VPN, міжмережевих екранів.

– CiscoView (Common Services) – графічний засіб керування пристроями.

– CiscoWorks Management Server – загальне керування інтеграцією зі сторонніми системами мережевого керування, контролем адміністративного доступу й сервісами для всього сімейства рішень CiscoWorks.

Характеристики:

– Можливість керування мережею з будь-якої точки в будь-який час через Інтернет з використанням web-інтерфейсу.

– Стандартний браузер як користувальницький інтерфейс.

– Автоматичне складання карти мережі.

– Багаторівневий контроль адміністративного доступу.

– Засоби конфігурування й моніторингу VPN, IPSec, підтримка VPN Concentrator 3000 (VPN c3000).

– Засоби моніторингу й керування міжмережевыми екранами Cisco PIX, IOS Firewall.

– Засоби моніторингу й керування системами виявлення несанкціонованого доступу.

– Засоби моніторингу системи безпеки в цілому з можливістю кореляції подій від різних пристроїв.

Відкритий дизайн CiscoWorks дозволяє швидко обновляти програмні продукти для задоволення нових вимог і потреб користувачів. Додаткові функціональні «вбудовування» (drop-in) модулі, які будуть випускатися в міру необхідності, забезпечать розширення й удосконалювання функціональних можливостей CiscoWorks.

Одна з функцій CiscoWorks, Cisco Management Connection, забезпечує можливість інтеграції засобів мережевого керування інших виробників з використанням web-технологій. Ця можливість широко використовується Cisco і більш ніж 30 компаніями, включаючи Computer Associates, Hewlett-Packard, Sun Microsystems і Tivoli Systems для створення загальних зв'язків між CiscoWorks і ПЗ інших розроблювачів.

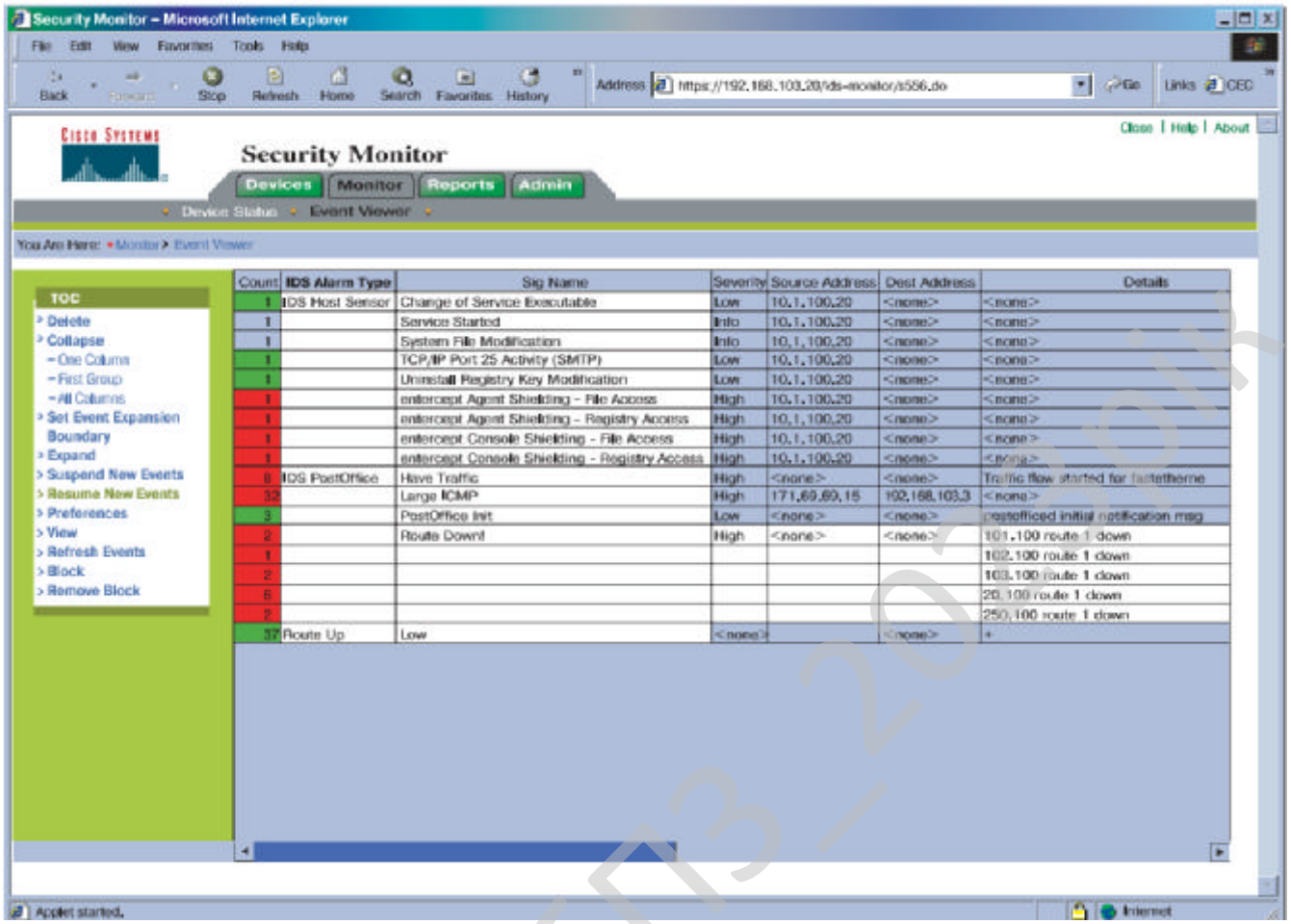

Рисунок 2.2 – Інтерфейс користувача CiscoWorks VPN / Security Management Solution

Огляд устаткування VPN

У даний момент існують десятки фірм, що роблять устаткування для створення VPN мереж. Розглядати всіх виробників ми в даному огляді не будемо. Зупинимося лише на тих, які на наш погляд займають лідируючі позиції на світовому й українському ринку в цій області.

Cisco Systems

Cisco Systems – безумовний лідер у виробництві мережевого й комунікаційного встаткування. Продукція компанії відрізняється винятковою надійністю й багатими можливостями, а як наслідок і висока ціна. Повинен помітити, досить висока ціна на встаткування Cisco цілком виправдана. Зрештою,

Ви платите не тільки за саме встаткування, але й за відсутність проблем з його роботою й за ті додаткові можливості, які воно надає.

Із продуктів, пропонованих компанією для побудови VPN мереж, початковим є Cisco 1700-ї серії. У даний момент пропонується два варіанти маршрутизаторів цієї серії – Cisco 1720 (без голосових портів) і Cisco 1750 (з можливістю підключення до чотирьох голосових портів).

Дана серія маршрутизаторів позиціонується виробником, як спеціально розроблена для тих організацій, які зацікавлені в надійному, з погляду безпеки, підключенні своїх корпоративних мереж до віртуальних приватних мереж (Virtual Private Networks) і до глобальних мереж.

Основні можливості:

– Високопродуктивний RISC процесор, що дозволяє шифрування у відповідності зі стандартною технологією IPSec на швидкості 512 кб/с для пакетів розмірів в 256 байт.

– Продуктивність 8500 (для 1720) і 12500 (для 1750) pps.

– DRAM пам'ять – до 48 mb.

– Flash пам'ять – до 16 mb.

– Вбудований 10/100 BASE-T ethernet порт.

– Два модулі (1720) і три модулі (1750) розширення.

– Повна підтримка всіх протоколів, необхідних для організації безпечної роботи у віртуальних приватних мережах, включаючи міжмережевий екран, IPSec, PAP/CHAP, TACACS+, RADIUS, Layer 2 Tunneling Protocol (L2TP), Layer 2 Forwarding (L2F), трансляцію мережевих адрес (Network Address Translation – NAT) і інші.

– Внутрішній слот розширення для підтримки майбутніх модулів апаратної компресії й шифрування на швидкостях E1.

– Багаті можливості керування якістю сервісу, включаючи Committed Access Rate (CAR), Policy Routing, Weighted Fair Queuing (WFQ), Generic Traffic Shaping (GTS) і Resource Reservation Protocol (RSVP).

– Підтримка повного спектра програмного забезпечення Cisco, а також можливе використання додаткового VPN модуля для апаратного прискорення шифрування.

Крім того, даний маршрутизатор представляє ще й великий інтерес через свою модульну структуру. Ця структура істотно розширює можливий спектр виконуваних завдань, що дозволить компаніям зменшити витрати на модернізацію мережі.

Цікавий і той факт, що основним, якщо не головним ідеологом і розроблювачем технології IPSec (шифрування на мережевому рівні) є саме компанія Cisco Systems. Тому дана функція реалізована в їхніх пристроях як не можна краще. Ну, і, безумовно, всі пристрої підтримують можливість використання серверів RADIUS і TACACS+, як сервери автентифікації.

Разом з тим необхідно помітити, що якщо у Вас кількість VPN-сесій буде великим (понад 10), те більш доцільно використовувати маршрутизатори Cisco 2600-й серії. Створення VPN мереж з використанням устаткування Cisco найбільше популярно, добре налагоджене й уже ефективно використовується в багатьох компаніях.

Allied Telesyn

Пристрої Allied Telesyn останнім часом знайшли своє місце в досить великому сегменті ринку. Якщо ще кілька років назад у Україні були широко поширені лише всілякі адаптери й трансівери, то в даний момент широко застосовуються й комутатори, і маршрутизатори, і інші пристрої.

На ринок VPN пристроїв компанія представила свій маршрутизатор AT-AR410.

Основні можливості:

– Висока продуктивність і гнучкість завдяки модульному доступу в територіальну мережу.

– Один вбудований порт 10/100 BASE-T для з'єднання з WAN.

– Чотирьохпортовий вбудований 10/100 BASE-T комутатор.

– Набудовуються для кожного порту віртуальні локальні мережі.

– Апаратний акселератор для підтримки VPN і IPSEC.

– Продуктивність 8500 pps.

– DRAM пам'ять – 16 mb.

– FLASH пам'ять – 8 mb.

– Повна підтримка всіх протоколів, необхідних для організації безпечної роботи у віртуальних приватних мережах, включаючи міжмережевий екран, IPSec, PAP/CHAP, TACACS+, RADIUS, Layer 2 Tunneling Protocol (L2TP), Layer 2 Forwarding (L2F), трансляцію мережевих адрес (Network Address Translation – NAT) і інші.

– Можливість додаткової установки модуля апаратного прискорення компресії й шифрування.

– Вільний слот для установки додаткових модулів розширення.

– Настроювання через веб інтерфейс.

Як видно з характеристик даного продукту, маршрутизатор має багатий набір можливостей. Тому його можна використовувати і як основу для побудови власної VPN мережі. Особливо коштовним даний продукт робить вбудований чотирьохпортовий світч із підтримкою VLAN. Крім того ICSA-сертифікований файрволл дає можливість створення DMZ у локальній мережі.

При наявності великої кількості VPN з'єднань використовуються вже могутніші моделі маршрутизаторів 700-й серії, що володіють високопродуктивним RISC процесором.

Таким чином, з огляду на ще й ціну даного пристрою, у сполученні із програмним забезпеченням від Allied Telesyn, дане рішення стає досить привабливим для більшості компаній.

Avaya Communication

Компанія Avaya більш широко відома в Росії своїми телефонними станціями Definity і комутаторами Cajun. Однак, вона робить і більшу лінійку пристроїв, призначених саме для організації VPN з'єднань, таких як Avaya VPNet

VSU™ (VPN Service Unit). У нашім огляді розглянемо VSU-100, призначений, за заявою виробника, для організації до 100 тунелів.

Основні можливості:

– Два вбудованих порти 10/100 Base-T.

– Підтримка шифрування DES,3DES.

– Підтримка методів автентифікації MD5, SHA-1.

– Апаратне прискорення шифрування й компресії.

– Вбудована функція файрволла.

Даний пристрій представляє із себе чистий VPN шлюз із можливостями файрволла й повинен установлюватися в мережі за маршрутизатором. Щодо цього він програє двом попереднім пристроям, представленим в огляді. У той же час, саме завдяки цьому VSU-100 у стані обслужити більше число сесій.

Керування пристроєм здійснюється спеціальним програмним забезпеченням VPNManager або за допомогою консольного доступу. Можливо також з'єднання по SSL протоколі.

Lucent Technologies

«Родинна» з Avaya компанія Lucent пропонує свій варіант пристроїв VPN під загальною назвою VPN Firewall Brick. У нашому огляді ми розглянемо продукт VPN Firewall Brick20.

Основні можливості:

– 3 вбудованих порти 10/100 BASE-T.

– Підтримка до 50 одночасних VPN тунелів (за заявою виробника).

– RISC процесор 120 Мгц.

– RAM пам'ять 8 Мб.

– Вбудований ICSA-сертифікований файрволл.

– Стандартна підтримка основних методів і протоколів шифрації.

– Підтримка серверів автентифікації RADIUS і TACACS.

– Має вбудовані порти для підключенню монітора й клавіатури. Тобто представляє із себе практично системний блок, з можливістю керування безпосередньо з консолі.

Даний пристрій поєднує в собі функції файрволла й маршрутизатора, що дозволяє реалізовувати набагато більше можливостей, чим властиво створення VPN з'єднань. Однак, незважаючи на заявлену високу продуктивність, для числа тунелів більше 10 все-таки рекомендується використовувати більше важкі моделі із цієї лінійки Firewall Brick80 або 100.

Наш огляд, звичайно ж, не є вичерпним. У нього не потрапив ряд досить відомих виробників. Наприклад, 3Com, чий продукт OfficeConnect Internet Firewall 25 з VPN upgrade становить великий інтерес у зв'язку з відносно низькою вартістю й тим, що сама марка 3Com добре зарекомендувала себе в українського споживача. Однак, дотепер не вирішена проблема із сертифікацією застосовуваних методів шифрування поки не дозволяє застосовувати його на практиці.

Не розглядали ми й продукцію фірми Nokia, досить відому своїми пристроями захисту даних і файрволлами під керуванням широко відомої системи CheckPoint. Пристрої цього виробника досить дороги й поки не дуже поширені на українському ринку.

Додам ще, що всі виробники для підключення до своїх пристроїв пропонують власні реалізації VPN клієнтів під будь-які операційні системи. Але, як правило, вони працюють і при використанні клієнтської частини від інших виробників. Зокрема від Microsoft або PGPNet.

Процес шифрування трафіка вимагає досить великих потужностей процесора й великого обсягу пам'яті. Реальний пристрій може бути й буде обслуговувати n-число тунелів, однак швидкість передачі даних не дозволить використовувати жодний додаток. Тому дуже важливо, підібрати саме те встаткування, що точно відповідає необхідним завданням.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

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

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

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

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне

забезпечення, яке призначено для системи кібербезпеки побудови VPN-мережі.

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

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

б) вибрати та обґрунтувати методику побудови системи 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

VPN (Virtual Private Network – віртуальна приватна мережа) – узагальнена назва технологій, що дозволяють забезпечити одне або кілька мережевих з'єднань (логічну мережу) поверх іншої мережі (наприклад, Інтернет). Незважаючи на те, що комунікації здійснюються по мережах з меншим невідомим рівнем довіри (наприклад, по публічних мережах), рівень довіри до побудованої логічної мережі не залежить від рівня довіри до базових мереж завдяки використанню засобів криптографії (шифруванню, автентифікації, інфраструктури публічних ключів, засобам для захисту від повторів і зміни переданих по логічній мережі повідомлень).

Залежно від застосовуваних протоколів і призначення, VPN може забезпечувати з'єднання трьох видів: вузол-вузол, вузол-мережа й мережа-мережа.

Звичайно VPN розгортають на рівнях не вище мережевого, тому що застосування криптографії на цих рівнях дозволяє використовувати в незмінному виді транспортні протоколи (такі як TCP, UDP).

Користувачі Microsoft Windows позначають терміном VPN одну з реалізацій віртуальної мережі – PPTP, причому використовувану найчастіше не для створення приватних мереж.

Найчастіше для створення віртуальної мережі використовується інкапсуляція протоколу PPP у який-небудь інший протокол – IP (такий спосіб використовує реалізація PPTP – Point-to-Point Tunneling Protocol) або Ethernet (PPPoE) (хоча й вони мають розходження). Технологія VPN останнім часом використовується не тільки для створення властиво приватних мереж, але й деякими провайдерами «останньої милі» для надання виходу в Інтернет.

При належному рівні реалізації й використанні спеціального програмного

забезпечення мережа VPN може забезпечити високий рівень шифрування переданої інформації. При правильному настроюванні всіх компонентів технологія VPN забезпечує анонімність у Мережі.

VPN складається із двох частин: «внутрішня» (підконтрольна) мережа, якої може бути трохи, і «зовнішня» мережа, по якій проходить інкапсульоване з'єднання (звичайно використовується Інтернет). Можливо також підключення до віртуальної мережі окремого комп'ютера. Підключення віддаленого користувача до VPN виробляється за допомогою сервера доступу, що підключений як до внутрішньої, так і до зовнішньої (загальнодоступної) мережі. При підключенні віддаленого користувача (або при установці з'єднання з іншою захищеною мережею) сервер доступу вимагає проходження процесу ідентифікації, а потім процесу автентифікації. Після успішного проходження обох процесів, віддалений користувач (віддалена мережа) наділяється повноваженнями для роботи в мережі, тобто відбувається процес авторизації.

Класифікувати VPN рішення можна по декількох основних параметрах:

1. За ступенем захищеності використовуваного середовища:

– Захищені. Найпоширеніший варіант віртуальних приватних мереж. З його допомогою можливо створити надійну й захищену на основі ненадійної мережі, як правило, Інтернету. Прикладом захищених VPN є: IPSec, OpenVPN і PPTP.

– Довірчі. Використовуються у випадках, коли передавальне середовище можна вважати надійним й необхідно вирішити лише завдання створення віртуальної підмережі в рамках більшої мережі. Проблеми безпеки стають неактуальними. Прикладами подібних VPN рішень є: Multi-protocol label switching (MPLS) і L2TP (Layer 2 Tunnelling Protocol) (точніше сказати, що ці протоколи перекладають завдання забезпечення безпеки на інші, наприклад L2TP, як правило, використовується в парі з IPSec).

2. За способом реалізації:

– У вигляді спеціального програмно-апаратного забезпечення. Реалізація VPN мережі здійснюється за допомогою спеціального комплексу програмно-

апаратних засобів. Така реалізація забезпечує високу продуктивність і, як правило, високий ступінь захищеності.

– У вигляді програмного рішення. Використовують персональний комп'ютер зі спеціальним програмним забезпеченням, що забезпечує функціональність VPN.

– Інтегроване рішення. Функціональність VPN забезпечує комплекс, що вирішує також завдання фільтрації мережевого трафіку, організації мережевого екрана й забезпечення якості обслуговування.

3. За призначенням:

– Intranet VPN. Використовують для об'єднання в єдину захищену мережу декількох розподілених філій однієї організації, що обмінюються даними по відкритим каналам зв'язку.

– Remote Access VPN. Використовують для створення захищеного каналу між сегментом корпоративної мережі (центральним офісом або філією) і одиночним користувачем, що, працюючи дома, підключається до корпоративних ресурсів з домашнього комп'ютера, корпоративного ноутбука, смартфона або інтернет-кіоску.

– Extranet VPN. Використовують для мереж, до яких підключаються «зовнішні» користувачі (наприклад, замовники або клієнти). Рівень довіри до них набагато нижче, ніж до співробітників компанії, тому потрібне забезпечення спеціальних «рубежів» захисту, що запобігають або обмежують доступ останніх до особливо коштовній, конфіденційній інформації.

– Internet VPN. Використовується для надання доступу до інтернету провайдерами, звичайно у випадку якщо по одному фізичному каналі підключаються декілька користувачів.

– Client/Server VPN. Він забезпечує захист переданих даних між двома вузлами (не мережами) корпоративної мережі. Особливість даного варіанта у тому, що VPN будується між вузлами, що перебувають, як правило, в одному сегменті мережі, наприклад, між робочою станцією й сервером. Така необхідність

дуже часто виникає в тих випадках, коли в одній фізичній мережі необхідно створити кілька логічних мереж. Наприклад, коли треба розділити трафік між фінансовим департаментом і відділом кадрів, що звертаються до серверів, які перебувають в одному фізичному сегменті. Цей варіант схожий на технологію VLAN, але замість поділу трафіку, використовується його шифрування.

4. За типом протоколу. Існують реалізації віртуальних приватних мереж під TCP/IP, IPX і AppleTalk. Але на сьогоднішній день спостерігається тенденція до загального переходу на протокол TCP/IP, і абсолютна більшість VPN рішень підтримує саме його. Адресація в ньому найчастіше вибирається у відповідності зі стандартом RFC5735, з діапазону Приватних мереж TCP/IP.

5. За рівнем мережевого протоколу. За рівнем мережевого протоколу на основі зіставлення з рівнями еталонної мережевої моделі ISO/OSI.

Приклади 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.

Багато великих провайдерів пропонують свої послуги з організації VPN-мереж для бізнесів-клієнтів.

По своїй суті VPN має багато властивостей виділеної лінії, однак розгортається вона в межах загальнодоступної мережі, наприклад Інтернету. За допомогою методики туннелювання пакети даних транслюються через загальнодоступну мережу як по звичайному двоточечному з'єднанню. Між кожною парою « відправник-одержувач даних» установлюється своєрідний

тунель – безпечне логічне з'єднання, що дозволяє інкапсулювати дані одного протоколу в пакети іншого. Дуже важливою властивістю тунелів є можливість диференціації різних типів трафіку й призначення їм необхідних пріоритетів обслуговування.

Основними компонентами тунелю є:

– ініціатор;

– маршрутизуєма мережа;

– тунельний комутатор;

– один або кілька тунельних термінаторів.

Ініціювати й розривати тунель можуть всілякі мережеві пристрої й програмне забезпечення. Наприклад, тунель може бути ініційований ноутбуком мобільного користувача, обладнаним модемом і відповідним програмним забезпеченням для встановлення з'єднань віддаленого доступу. Як ініціатор може виступити також маршрутизатор екстрамережі (локальної мережі), наділений відповідними функціональними можливостями. Тунель звичайно завершується комутатором екстрамережі або шлюзом провайдеру послуг.

Сам по собі принцип роботи VPN не суперечить основним мережевим технологіям і протоколам. Наприклад, при встановленні з'єднання віддаленого доступу клієнт посилає серверу потік пакетів стандартного протоколу PPP. У випадку організації віртуальних виділених ліній між локальними мережами їхні маршрутизатори також обмінюються пакетами PPP. Проте, принципово новим моментом є пересилання пакетів через безпечний тунель, організований у межах загальнодоступної мережі.

Туннелювання дозволяє організувати передачу пакетів одного протоколу в логічному середовищі, що використовує інший протокол. У результаті з'являється можливість вирішити проблеми взаємодії декількох різнотипних мереж, починаючи з необхідності забезпечення цілісності й конфіденційності переданих даних і закінчуючи подоланням невідповідностей зовнішніх протоколів або схем адресації.

Існуюча мережева інфраструктура корпорації може бути підготовлена до використання VPN як за допомогою програмного, так і за допомогою апаратного забезпечення. Організацію віртуальної приватної мережі можна зрівняти із прокладкою кабелю через глобальну мережу. Як правило, безпосереднє з'єднання між віддаленим користувачем і окінцевим пристроєм тунелю встановлюється за протоколом PPP.

Найпоширеніший метод створення тунелів VPN – інкапсуляція мережевих протоколів (IP, IPX, AppleTalk і т.д.) в PPP і наступна інкапсуляція утворених пакетів до протоколу туннелювання. Звичайно в якості останнього виступає IP або (набагато рідше) ATM і Frame Relay. Такий підхід називається туннелюванням другого рівня, оскільки «пасажиром» тут є протокол саме другого рівня.

Альтернативний підхід – інкапсуляція пакетів мережевого протоколу безпосередньо до протоколу туннелювання (наприклад, VTP) називається туннелюванням третього рівня.

Незалежно від того, які протоколи використовуються або які цілі переслідуються при організації тунелю, основна методика залишається практично незмінною. Звичайно один протокол використовується для встановлення з'єднання з віддаленим вузлом, а інший – для інкапсуляції даних і службової інформації з метою передачі через тунель. Як приклад використання тунелю для усунення невідповідностей між протоколами й схемами адресації можна привести технологію Simple Internet Transition (SIT), що повинна з'явитися разом із протоколом IPv6. Це ретельно розроблена групою інженерів (IETF) методологія туннелювання, покликана полегшити перехід від четвертої версії міжмережевого протоколу (IPv4) до шостого (IPv6). Ці версії досить відрізняються, щоб говорити про безпосередню сумісність мереж. Інкапсуляція ж пакетів протоколу IPv6 у пакети IPv4 дозволяє досягти необхідного рівня функціональної сумісності.

Історія появи VPN тісно пов'язана з послугою CENTREX у телефонних мережах. Поняття Centrex з'явилося на рубежі 60-х років у США як загальна назва способу надання послуг ділового зв'язку абонентам декількох компаній на основі спільно використовуваного встаткування однієї станціє для установ PBX (Private Branch Exchange). З початком впровадження в США й Канаді станцій із програмним керуванням термін придбав інший зміст і став означати спосіб надання діловим абонентам додаткових послуг телефонного зв'язку, еквівалентних послугам PBX, на базі модифікованих станцій мережі загального користування. Основна перевага Centrex полягало у тому, що фірми й компанії при створенні виділених корпоративних мереж заощаджували значні засоби, необхідні на покупку, монтаж і експлуатацію власних станцій. Хоча для зв'язку між собою абоненти Centrex використовують ресурси й устаткування мережі загального користування, самі вони утворять так звані замкнуті групи користувачів CUG (Closed Users Group) з обмеженим доступом ззовні, для яких у станціях мережі реалізуються віртуальні PBX.

У прагненні перебороти властиві Centrex обмеження була висунута ідея віртуальної приватної мережі VPN – як об'єднання CUG, що становлять одну корпоративну мережу й перебувають на віддаленні друг від друга. Ресурси VPN (кожна зі своїм планом нумерації) можуть бути розподілені по декількох станціях місцевої мережі, оснащеним функціями Centrex і, що має в зоні свого обслуговування одну або трохи CUG. При цьому в станцію можуть бути включені як PBX, безпосередньо приналежному власникові VPN, так і лінії звичайних індивідуальних абонентів.

Природно, ніяка компанія не хотіла б відкрито передавати в Інтернет фінансову або іншу конфіденційну інформацію. Канали VPN захищені потужними алгоритмами шифрування, закладеними в стандарти протоколу безпеки IPsec. IPsec (Internet Protocol Security – стандарт, обраний міжнародним співтовариством, групою IETF – Internet Engineering Task Force) створює основи безпеки для Інтернет-протоколу (IP), незахищеність якого довгий час була

проблемою. Протокол IPsec забезпечує захист на мережевому рівні й вимагає підтримки стандарту IPsec тільки від пристроїв, що спілкуються між собою, по обох сторони з'єднання. Всі інші пристрої, розташовані між ними, просто забезпечують трафік IP-пакетів.

Спосіб взаємодії осіб, що використовують технологію IPsec, прийнято визначати терміном "захищена асоціація" – Security Association (SA). Захищена асоціація функціонує на основі угоди, укладеного сторонами, які користуються засобами IPsec для захисту переданої один одному інформації. Ця угода регулює кілька параметрів: IP-адреси відправника й одержувача, криптографічний алгоритм, порядок обміну ключами, розміри ключів, термін служби ключів, алгоритм автентифікації.

Інші стандарти включають протокол PPTP (Point to Point Tunneling Protocol), що розвивається Microsoft, L2F (Layer 2 Forwarding), що розвивається Cisco, – обоє для віддаленого доступу. Microsoft і Cisco працюють разом з IETF, щоб з'єднати ці протоколи в єдиний стандарт L2P2 (Layer 2 Tunneling Protocol) з метою використання IPsec для тунельної автентифікації, захисту приватної власності й перевірки цілісності.

Проблема полягає в тому, щоб забезпечити прийнятну швидкодію мережі при обміні шифрованою інформацією. Алгоритми кодування вимагають значних обчислювальних ресурсів процесора, іноді в 100 разів більших, ніж при звичайній IP-маршрутизації. Щоб домогтися необхідної продуктивності, треба подбати про адекватне підвищення швидкодії, як серверів, так і клієнтських ПК. Крім того, є спеціальні шлюзи з особливими схемами, які помітно прискорюють шифрування.

IT-менеджер може вибирати конфігурацію віртуальної приватної мережі залежно від конкретних потреб. Наприклад співробітникові, що працює вдома може бути наданий обмежений доступ до мережі, а менеджерові віддаленого офісу або керівникові компанії – широкі права доступу. Один проект може обмежуватися лише мінімальним (56-розрядним) шифруванням при роботі через

віртуальну мережу, а фінансова й планова інформація компанії вимагає могутніших засобів шифрування – 168-розрядних.

На жаль, доводиться відзначити, що засоби побудови VPN не є повноцінними засобами виявлення й блокування атак. Вони можуть запобігти ряду несанкціонованих дій, але далеко не всі можливості, які можуть використовувати хакери для проникнення в корпоративну мережу. Вони не можуть виявити віруси й атаки типу "відмова в обслуговуванні" (це роблять антивірусні системи й засоби виявлення атак), вони не можуть фільтрувати дані по різних ознаках (це роблять міжмережеві екрани) і т.д. На це можна заперечити, що ці небезпеки не страшні, тому що VPN не прийме незашифрований трафік і відкине його. Однак на практиці це не так. По-перше, у більшості випадків засіб побудови VPN використовується для захисту лише частини трафіку, наприклад, спрямованого у віддалену філію. Інший трафік (наприклад, до публічних Web-серверів) проходить через VPN-пристрій без обробки. А по-друге, статистика стверджує, що до 80% всіх інцидентів, пов'язаних з інформаційною безпекою, відбувається з вини авторизованих користувачів, що мають санкціонований доступ у корпоративну мережу. Із чого слідує вивід, що атака або вірус будуть зашифровані нарівні з безпечним трафіком.

Продуктивність мережі – це досить важливий параметр, і на будь-які засоби, що сприяють його зниженню, у будь-якій організації дивляться з підозрою. Не є виключенням і засоби побудови VPN, які створюють додаткові затримки, пов'язані з обробкою трафіку, що проходить через VPN-пристрій. Всі затримки, що виникають при криптографічній обробці трафіку, можна розділити на три типи:

– Затримки при встановленні захищеного з'єднання між VPN-пристроями.

– Затримки, пов'язані із зашифровуванням і розшифровуванням захищаємих даних, а також з перетвореннями, необхідними для контролю їхньої цілісності.

– Затримки, пов'язані з додаванням нового заголовка до переданих пакетів.

Реалізація першого, другого й четвертого варіантів побудови VPN передбачає встановлення захищених з'єднань не між абонентами мережі, а тільки між VPN-пристроями. З урахуванням криптографічної стійкості використовуваних алгоритмів зміна ключа можлива через досить тривалий інтервал часу. Тому при використанні засобів побудови VPN затримки першого типу практично не впливають на швидкість обміну даними. Зрозуміло, це положення стосується стійких алгоритмів шифрування, що використовують ключі не менш 128 біт (Triple DES, ДСТ28147-89 і т.ін.). Пристрої, що використовують колишній стандарт DES, здатні вносити певні затримки в роботу мережі.

Затримки другого типу починають позначатися тільки при передачі даних по високошвидкісних каналах (від 10 Мбіт/с). У всіх інших випадках швидкодія програмної або апаратної реалізації обраних алгоритмів шифрування й контролю цілісності звичайно досить велика й у ланцюжку операцій «зашифровування пакета – передача пакета в мережу» і «прийом пакетів з мережі – розшифровування пакета» час зашифровування (розшифровування) значно менше часу, необхідного для передачі даного пакета в мережу.

Основна проблема тут пов'язана з додаванням додаткового заголовка до кожного пакета, що пропускається через VPN-пристрій. Як приклад розглянемо систему диспетчерського керування, що у реальному масштабі часу здійснює обмін даними між віддаленими станціями й центральним пунктом. Розмір переданих даних не великий – не більше 25 байтів. Дані порівнянного розміру передаються в банківській сфері (платіжні доручення) і в IP-телефонії. Інтенсивність переданих даних – 50-100 змінних у секунду. Взаємодія між вузлами здійснюється по каналах із пропускною здатністю в 64 Кбіт/с.

Пакет зі значенням однієї змінної процесу має довжину 25 байтів (ім'я змінної – 16 байтів, значення змінної – 8 байт, службовий заголовок – 1 байт). IP-протокол додає до довжини пакета ще 24 байта (заголовок IP-пакета). При використанні як середовище передачі каналів Frame Relay LMI додається ще 10 байтів FR-заголовка. Усього – 59 байтів (472 біта). Таким чином, для передачі 750

значень змінних процесу за 10 секунд (75 пакетів у секунду) необхідна смуга пропущення 75×472 = 34,5 Кбіт/с, що добре вписується в наявні обмеження пропускної здатності в 64 Кбіт/с. Тепер подивимося, як поводиться мережа при включенні в неї засобу побудови VPN. Перший приклад – засобу на основі протоколу SKIP. До 59 байтів даних додається 112 байт додаткового заголовка (для ДСТ28148-89), що складе 171 байт (1368 біт). 75х1368 = 102,6 Кбіт/с, що на 60% перевищує максимальну пропускну здатність наявного каналу зв'язку.

Для протоколу IPsec і вищевказаних параметрів пропускна здатність буде перевищена на 6% (67,8 Кбіт/с). Це за умови, що додатковий заголовок для алгоритму ДСТ28147-89 складе 54 байта. Для протоколу, використовуваного в українському програмно-апаратному комплексі «Континент-К», додатковий заголовок, що додається до кожного пакета, становить усього 36 байтів (або 26 – залежно від режиму роботи), що не викликає ніякого зниження пропускної здатності (57 і 51 Кбіт/с відповідно). Справедливості заради необхідно відзначити, що всі ці викладення вірні лише за умови, що, крім зазначених змінних, у мережі більше нічого не передається.

У цей час найпоширенішим протоколом VPN є протокол двоточечного тунельного зв'язку (Point-to-Point Tunnelling Protocol – PPTP). Розроблений він компаніями 3Com і Microsoft з метою надання безпечного віддаленого доступу до корпоративних мереж через Інтернет. PPTP використовує існуючі відкриті стандарти TCP/IP і багато в чому покладається на застарілий протокол двоточечного зв'язку PPP. На практиці PPP так і залишається комунікаційним протоколом сеансу з'єднання PPTP.

PPTP створює тунель через мережу до NT-сервера одержувача й передає по ньому PPP-пакети віддаленого користувача. Сервер і робоча станція використовують віртуальну приватну мережу й не звертають уваги на те, наскільки безпечною або доступною є глобальна мережа між ними. Завершення сеансу з'єднання з ініціативи сервера (на відміну від спеціалізованих серверів віддаленого доступу) дозволяє адміністраторам локальної мережі не пропускати віддалених користувачів за межі системи безпеки Windows NT Server. У

результаті користувач використовує віртуальну приватну мережу, не наносячи при цьому збитку функціональним можливостям загальнодоступної мережі. Всі служби домену NT, включаючи DHCP, WINS і доступ до Network Neighborhood, без усяких застережень надаються віддаленому користувачеві.

Хоча компетенція протоколу PPTP поширюється тільки на пристрої, що працюють під керуванням Windows, він надає компаніям можливість взаємодіяти з існуючими мережевими інфраструктурами й не завдавати шкоди власній системі безпеки. Таким чином, віддалений користувач може підключитися до Інтернету за допомогою місцевого провайдеру по аналоговій телефонній лінії або каналу ISDN і встановити з'єднання із сервером NT. При цьому компанії не доводиться витрачати великі суми на організацію й обслуговування пула модемів, що надає послуги віддаленого доступу.

У найближчому майбутньому очікується ріст кількості віртуальних приватних мереж, розгорнутих на базі нового протоколу туннелювання другого рівня (Layer 2 Tunneling Protocol – L2TP). Цей протокол дозволяє об'єднати функціонуючі на другому рівні PPTP і L2F (Layer 2 Forwarding – протокол пересилання другого рівня) і розширити їхньої можливості. Однієм з них є багатоточечне туннелювання, що дозволяє користувачам ініціювати створення декількох мереж VPN, наприклад, для одночасного доступу до Інтернету й корпоративної мережі.

Протоколи L2TP і PPTP відрізняються від протоколів туннелювання третього рівня рядом особливостей:

1. Надання корпораціям можливості самостійно вибирати спосіб автентифікації користувачів і перевірки їхніх повноважень – на власній «території» або в провайдеру Інтернет-послуг. Обробляючи туннельовані пакети PPP, сервери корпоративної мережі одержують всю інформацію, необхідну для ідентифікації користувачів.

2. Підтримка комутації тунелів – завершення одного тунелю й ініціювання іншого до одному з безлічі потенційних термінаторів. Комутація тунелів дозволяє

як би продовжити PPP-з'єднання до необхідної кінцевої точки.

3. Надання системним адміністраторам корпоративної мережі можливості реалізації стратегій призначення користувачам прав доступу безпосередньо на брандмауері й внутрішніх серверах. Оскільки термінатори тунелю одержують пакети PPP з відомостями про користувачів, вони в стані застосовувати сформульовані адміністраторами стратегії безпеки до трафіку окремих користувачів. (Туннелювання третього рівня не дозволяє розрізняти вступників від провайдеру пакети, тому фільтри стратегії безпеки доводиться застосовувати на кінцевих робочих станціях і мережевих пристроях.) Крім того, у випадку використання тунельного комутатора з'являється можливість організувати «продовження» тунелю другого рівня для безпосередньої трансляції трафіку окремих користувачів до відповідних внутрішніх серверів. На такі сервери може бути покладене завдання додаткової фільтрації пакетів.

Переваги технології VPN настільки переконливі, що багато компаній починають будувати свою стратегію з урахуванням використання Інтернету як головного засобу передачі інформації, навіть тієї, котра є уразливою. Переваги VPN уже оцінені по достоїнству багатьма підприємствами.

При правильному виборі VPN:

– ми одержуємо захищені канали зв'язку за ціною доступу в Інтернет, що в кілька разів дешевше виділених ліній;

– при установці VPN не потрібно змінювати топологію мереж, переписувати додатки, навчати користувачів – все це значна економія;

– забезпечується масштабування, оскільки VPN не створює проблем росту й зберігає зроблені інвестиції;

– ви незалежні від криптографії й можете використовувати модулі криптографії будь-яких виробників відповідно до національних стандартів тої або іншої країни;

– відкриті інтерфейси дозволяють інтегрувати вашу мережу з іншими програмними продуктами й бізнес-додатками.

До недоліків можна віднести порівняно низьку надійність. У порівнянні з виділеними лініями й мережами на основі Frame relay віртуальні частки мережі менш надійні, однак в 5-10, а іноді й в 20 разів дешевше. На думку західних аналітиків, це не зупинить продаж VPN, оскільки лише п'яти відсоткам користувачів, що торгують, наприклад, на ринку цінних паперів, потрібні такі високі стандарти. Інші 95% не настільки серйозно відносяться до проблем зі зв'язком, а витрати більшої кількості часу на одержання інформації не приводять до колосальних збитків.

У силу того, що послуга VPN надається й підтримується зовнішнім оператором, можуть виникати проблеми зі швидкістю внесення змін у бази доступу, у настроювання firewall, а також з відновленням устаткування, що вийшло з ладу. У цей час проблема вирішується вказівкою в договорах максимального часу на усунення неполадок і внесення змін. Звичайно цей час становить кілька годин, але зустрічаються провайдери, що гарантують усунення неполадок протягом доби.

Ще один істотний недолік – у споживачів немає зручних засобів керування VPN. Хоча останнім часом розробляється устаткування, що дозволяє автоматизувати керування VPN. Серед лідерів цього процесу – компанія Indus River Networks Inc., дочірня компанія MCI WorldCom і Novell. Як говорять аналітики Forester Research, VPN повинні контролюватися користувачами, управлятися компаніями-операторами, а завдання розроблювачів програмного забезпечення – вирішити цю проблему.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема системи побудованої з використанням обладнання Cisco. В основі схеми знаходиться модуль NME-RVPN для маршрутизаторів серії Cisco® 2800 і 3800 Integrated Services Routers. Модуль являє собою інноваційне рішення, інтегроване саме з маршрутизаторами

Cisco і спеціально розроблене для забезпечення українського ринку високотехнологічним сертифікованим рішенням VPN з передовими технологіями Cisco і задовольняючим сучасним вимогам ефективного захисту всіх видів мережевих взаємодій.

Модуль NME-RVPN у складі маршрутизаторів серії Cisco® 2800 і 3800 Integrated Services Routers пропонує українським споживачам унікальний пристрій, що дозволяє забезпечити як ефективну маршрутизацію, так і захист трафіку даних, голосу, відео. При цьому пристрій управляється як єдине ціле, використовуючи інтерфейс Cisco для формування правил маршрутизації й захисту мережевих взаємодій. Подібна глибока інтеграція дозволяє істотно зменшити складність мережі, не пред'являти додаткових вимог до кваліфікації персоналу й, як результат, знизити витрати на розгортання й підтримку, а також строки розгортання підсистеми інформаційної безпеки.

У зв'язку із широкою інтеграцією корпоративних комунікацій з публічними мережами для забезпечення взаємодій компаній з філіями, віддаленими користувачами, замовниками й партнерами першорядного значення набуває питання забезпечення українських користувачів сертифікованим VPN-рішенням у сполученні з передовими технологіями Cisco Systems і задовольняючим сучасним вимогам ефективного захисту всіх видів мережевих взаємодій. При цьому необхідно не тільки вирішити питання захисту зовнішнього обміну даними, але й надати сучасні рішення по захищених бездротових комунікаціях, захисту голосу й відео із забезпеченням якості обслуговування, максимально ефективно захистити взаємодію клієнтів у мережах операторів зв'язку й послуг.

Включення модуля NME-RVPN у маршрутизатори серії Cisco 2800 або 3800 Integrated Services Router дозволяє споживачам одержати єдине рішення, що забезпечує, у тому числі, організацію мережевого захисту, що використовує українську сертифіковану криптографію, розвинену маршрутизацію, якість обслуговування пріоритетного трафіку (QoS), сервіси відео й голосу, комутацію

мереж. Подібні якості разом з керованістю й надійністю технологій Cisco IOS практично повністю закривають потреба сучасного бізнесу в організації й захисті відповідальних, критично важливих мережевих взаємодій.

Програмне забезпечення CSP VPN Gate, що входить до складу модуля NME-RVPN, є ще одним елементом сімейства продуктів CSP VPN Client, CSP VPN Server і масштабованої серії шлюзів безпеки CSP VPN Gate 100/1000/3000/7000/10000.

Продукти CSP VPN забезпечують базову функціональність сучасного VPN-пристрою:

– Шифрування (конфіденційність) і ЕЦП (цілісність, автентифікація) IP-пакетів, цілісність потоку пакетів.

– Маскування топології мережі за рахунок інкапсуляції трафіку в захищений тунель.

– Прозорість для NAT (підтримка інкапсуляції пакета ESP в UDP).

– Автентифікацію вузлів мережі й користувачів, контроль доступу на рівні комп'ютерів, користувачів і додатків, інтегрований міжмережевий екран 4-го класу (CSP VPN Gate задовольняє вимогам до міжмережевого екрану по 4-му класу захищеності).

– Забезпечення надійності з вирівнюванням навантаження в схемі резервування N+1 (Dead Peer Detection protocol).

– Уніфікацію політики безпеки для мобільних і «внутрішніх» користувачів (динамічне конфігурування корпоративних IP-адрес для віддалених користувачів «усередині VPN»).

– Збереження класифікації трафіку для захищених пакетів (мапирование To поверх IPsec), пріоритетну обробку трафіку голосу й відео (підтримка QoS), відсутність втрати пакетів при регенерації сесійних ключів (smooth IKE re-keying).

– Гнучке, централізоване й подійне ведення журналу з можливістю вторинної обробки на основі протоколу Syslog.

Як результат, застосування модуля NME-RVPN у складі маршрутизатора Cisco Integrated Services Router 2800/3800 забезпечує ефективну реалізацію безлічі сценаріїв сертифікованого захисту, включаючи:

– міжмережеву взаємодію;

– захищений доступ віддалених і мобільних користувачів;

– захист бездротових мереж;

– захист мультисервісних мереж (включаючи IP-телефонію й відеоконференцзв’язок);

– захист платіжних систем і систем керування технологічними процесами у виробництві й на транспорті.

Сценарії захисту міжмережевих взаємодій (Site-to-Site VPN) застосовуються для захисту комунікацій територіально розподілених корпоративних мереж через публічні (відкриті, що не заслуговують довіри) мережі/канали зв'язку.

По суті застосування VPN-рішень для цих цілей не повинне приводити до зниження вимог до характеристик безпосередньо каналу передачі даних, таких як підтримка множинності протоколів, висока надійність, більша масштабованість. Навпаки, сучасні VPN-рішення повинні забезпечувати високу цінову ефективність і більшу гнучкість у реалізації таких вимог. Високу цінову ефективність можна одержати, наприклад, за рахунок можливості використовувати публічні канали для передачі інформації, що раніше було недоступно.

Використання для цієї мети маршрутизаторів Cisco ISR повною мірою виконує поставлену вище завдання.

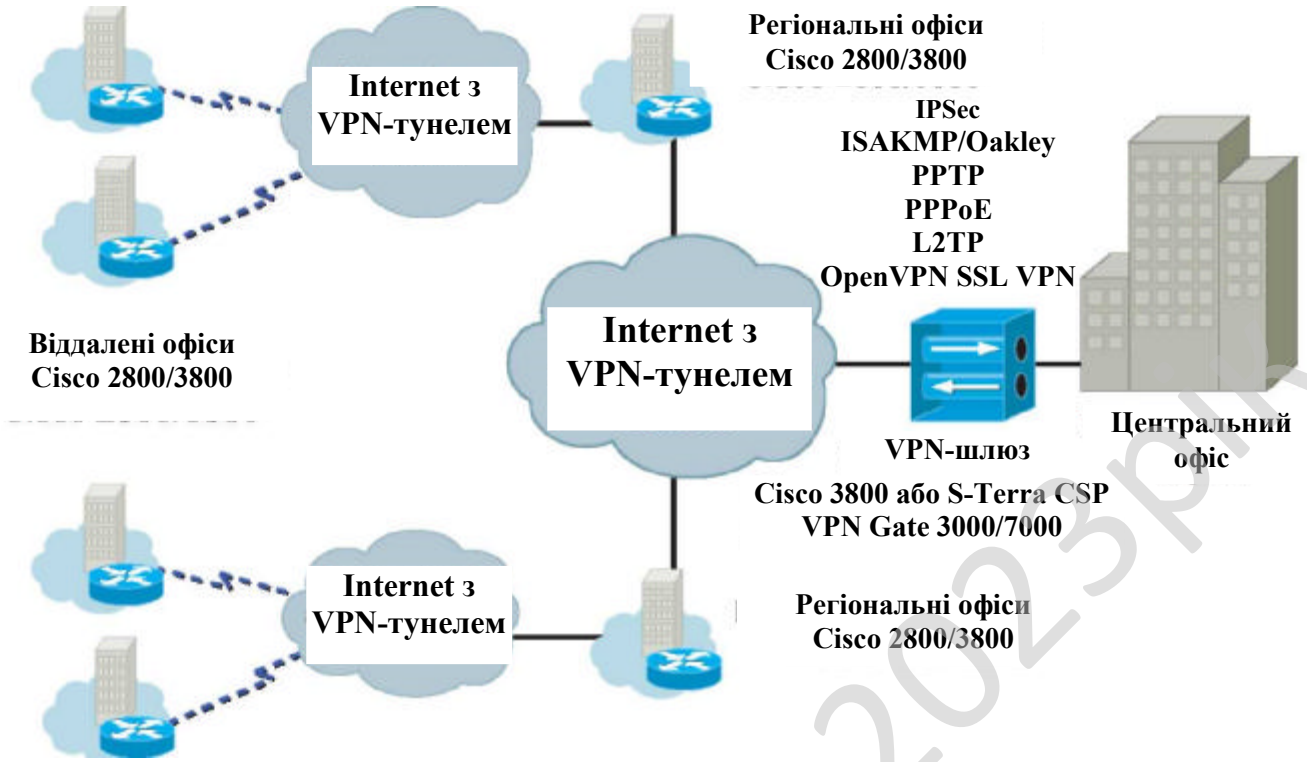

Рисунок 3.1 – Структурна схема системи

Для виконання вимог підвищеної надійності мережевих взаємодій великих мереж (забезпечуючий безперервність бізнес-процесів у них) на додаток до наведеного вище прикладу можуть використовуватися рішення з резервуванням і балансуванням навантаження.

Продукти CSP VPN підтримують сценарії захисту як виділених мультимедійних мереж, так і «змішаних» мереж, забезпечуючи:

– підтримку якості мережевого обслуговування;

– захист якості сервісу в голосовий VPN при перевантаженні трафіку даних.

Модуль NME-RVPN у складі маршрутизаторів Cisco 2800 або Cisco 3800, що забезпечують додаткову функціональність CallManager Express і бездротової точки доступу, надає для віддалених офісів всю необхідну функціональність обробки й захисти бездротових мультимедійних і мультисервісних мереж у єдиному пристрої.

Основним засобом захисту трафіку в бездротовій мережі є IPsec. При цьому забезпечується не тільки автентифікація пристроїв (що робиться на канальному рівні), але й автентифікація користувачів.

Застосування в бездротовій мережі виділеного адресного простору й IPsec VPN забезпечує:

– можливість ізолювати провідний сегмент від відкритого IP-трафіку;

– пропускати усередину провідної корпоративної мережі (до ресурсів локальної мережі) тільки IPsec-трафік, причому тільки в «домашні» мережі.

Сценарії віддаленого доступу користувачів (Remote Access VPN) застосовуються для захисту доступу віддалених або мобільних користувачів у корпоративну мережу через публічні (відкриті, що не заслуговують довіри) мережі або канали зв'язку.

– Політика безпеки клієнта доступу CSP VPN Client визначається тільки системним адміністратором (адміністратором безпеки) і не може бути змінена користувачем.

– Права доступу користувача визначаються в корпоративній мережі, і інформація про права доступу в корпоративній мережі відсутній на клієнті доступу CSP VPN Client.

– Клієнт доступу CSP VPN Client не жадає від користувача ніяких технічних операцій крім установки й введення ключа, наданого адміністратором безпеки.

CSP VPN Client підтримує захищений зв'язок практично з будь-якої точки, де є присутнім який-небудь комунікаційний ресурс. Використовуються спеціальні міри в забезпеченні мобільності користувача:

– адаптивність до адресного простору (IPsec автоматично включається в зонах, де потрібне захищене з'єднання);

– підтримка різних середовищ передачі, у тому числі мобільних (GPRS, CDMA, Wi-Fi, WiMAX і ін.);

– забезпечення прозорої передачі IKE/IPsec трафіку через шлюзи із трансляцією адрес (NAT).

У порівнянні з іншими окремими подібними пристроями модуль NME-RVPN при використанні в мережевій інфраструктурі центрального офісу має ряд переваг:

– Загальний з іншими пристроями інтерфейс керування. Для керування й конфігурування модуля можна застосовувати інтерфейс командного рядка (CLI) з використанням команд, аналогічних Cisco IOS. Модулем можна також управляти за допомогою графічного web-інтерфейсу.

– Зниження споживання й простота комутації. Модуль одержує живлення від маршрутизатора, не має потреби в комутації й не займає місця в стійці з мережевим устаткуванням.

Модуль NME-RVPN можна встановити в маршрутизатори Cisco ISR 2811, 2821, 2851, 3825 і 3845 з версією IOS 12.4(11)T або вище. При цьому модуль NME-RVPN працює незалежно від IOS маршрутизатора, використовуючи програмне забезпечення CSP VPN Gate v 2.1, установлене на компакт-флеш-карті (Compact Flash) модуля. Програмне забезпечення модуля функціонує під керуванням адаптованої OS Linux.

Апаратно модуль NME-RVPN являє собою обчислювальну платформу на базі процесора Intel Celeron-M 1.0 Ггц із 512 МБ оперативної пам'яті й 512 МБ Compact Flash (рисунок 4). Для підключення до локальної мережі модуль має зовнішній інтерфейс Gigabit Ethernet. Аналогічний внутрішній інтерфейс здійснює взаємодію й передачу даних між модулем і маршрутизатором.

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).

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 настроюваннях 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 настроюваннях, то інформація просто

відкидається. Те ж саме відбувається при прийманні пакета з навмисно або випадково ушкодженим заголовком.

– Відповідно до настроювань 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 настроювання 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 настроювань. 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 настроюваннях 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).

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. У системі використовуються наступні протоколи. В основу програмного забезпечення створення VPN підключень покладені протоколи забезпечення безпеки інформації IPsec та SSL.

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).

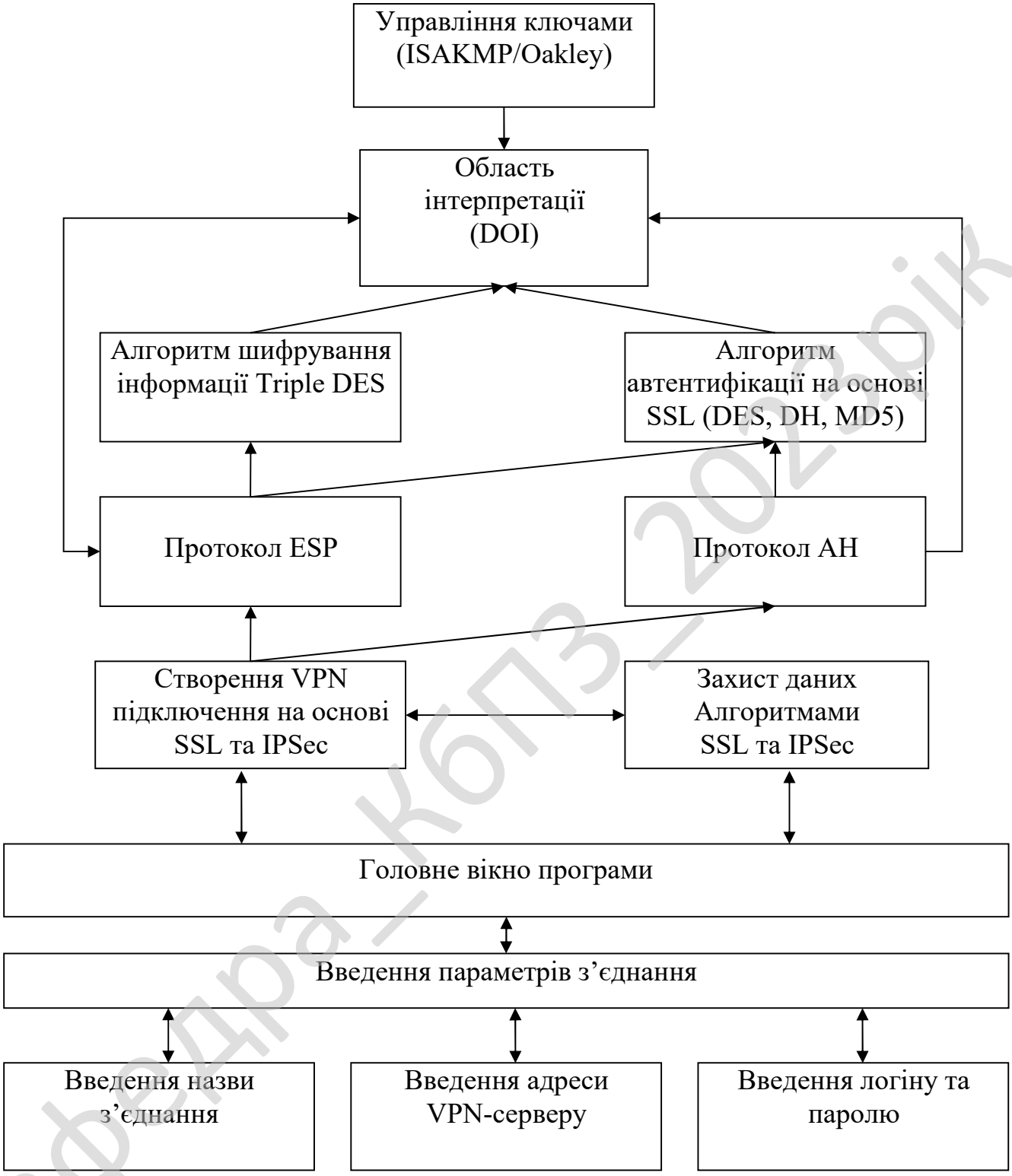

Рисунок 3.2 – Функціональна схема системи

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.

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.

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.

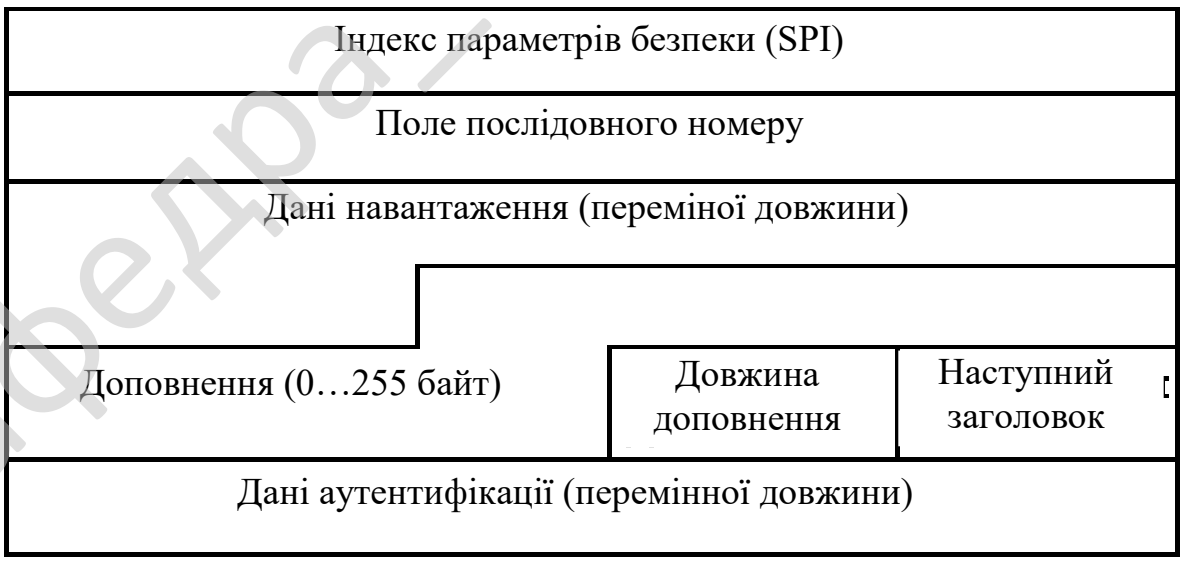

Рисунок 3.3 – Формат заголовка ESP

Розрізняють два режими застосування ESP – транспортний і тонельний.

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 мережевого рівня й, можливо, деякі опціональні заголовки пакета (в IPv6). Недоліком транспортного режиму є відсутність механізмів прихування 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.

Тонельний режим – припускає шифрування всього пакета, включаючи заголовок мережевого рівня. Тонельний режим застосовується якщо буде потреба прихування інформаційного обміну організації із зовнішнім миром. При цьому, адресні поля заголовка мережевого рівня пакета, що використовує тонельний режим, заповнюються міжмережевим екраном організації й не містять інформації про конкретного відправника пакета. При передачі інформації із зовнішнього миру в локальну мережу організації як адреса призначення використовується мережева адреса міжмережевого екрана. Після дешифрування міжмережевим екраном початкового заголовка мережевого рівня пакет направляється одержувачеві.

Заголовок АН

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

Формат АН 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 3.4 – Формат заголовка АН

Послідовний номер пакета був введений в АН 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 лініями 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.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.5. З рисунку видно, що процеси взаємодіють наступним чином.

Спершу запускається процес виведення головного вікна програми. Він взаємодіє з наступними процесами:

– Процес створення VPN-з’єднання.

– Процес вибору VPN-з’єднання зі списку.

Процес створення VPN-з’єднання взаємодіє з наступними процесами:

– Процес введення назви VPN-з’єднання.

– Процес введення адрес VPN-серверу.

– Процес введення параметрів, логіну та паролю VPN-з’єднання.

Процес вибору VPN-з’єднання зі списку взаємодіє з наступними процесами:

– Процес видалення VPN-з’єднання.

– Процес відключення від VPN-з’єднання.

– Процес модифікування VPN-з’єднання.

– Процес підключення VPN-з’єднання.

Процес модифікування VPN-з’єднання взаємодіє з наступними процесами:

– Процес зміни назви VPN-з’єднання.

– Процес зміни адреси VPN-серверу.

– Процес зміни параметрів, логіну та паролю VPN-з’єднання.

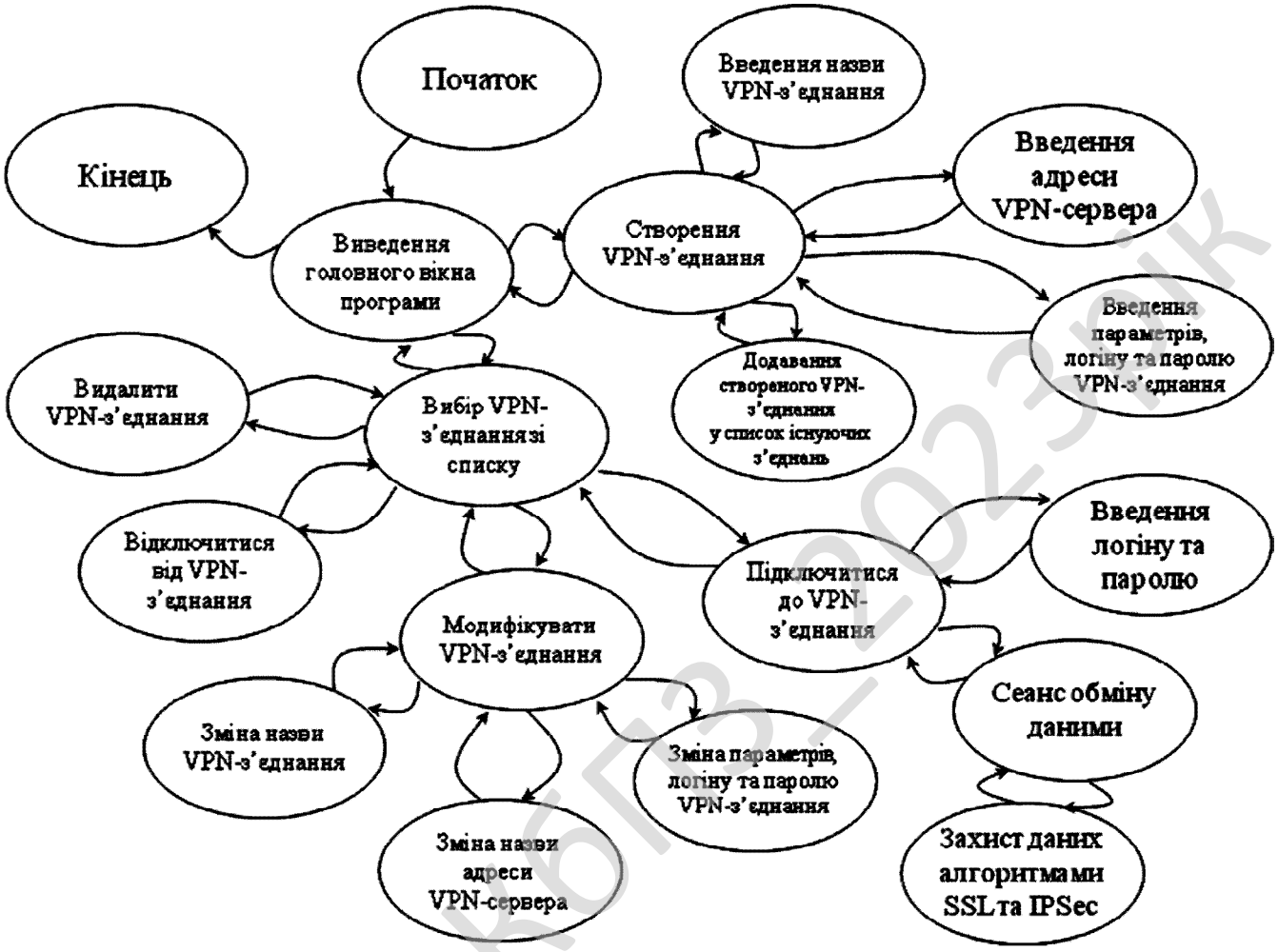

Рисунок 3.5 – Діаграма взаємодії процесів

Процес підключення VPN-з’єднання взаємодіє з наступними процесами:

– Процес введення логіну та паролю.

– Процес сеансу обміну даними, який, у свою чергу, взаємодіє з процесом захисту даних алгоритмами SSL та IPSec.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми. Після цього виводиться список VPN-з’єднань.

Наступним кроком, користувач може обрати одну з дій:

– Створити нове VPN-з’єднання.

– Модифікувати VPN-з’єднання.

– Видалити VPN-з’єднання.

– Підключитися до VPN-з’єднання.

– Відключитися від VPN-з’єднання.

Якщо користувач обирає створити VPN-з’єднання, тоді програма виконує наступні кроки:

– Вводиться адреса VPN-серверу.

– Вводиться логін та пароль.

– Встановлюються параметри з’єднання та шифрування.

– Створюється з’єднання та додається у список існуючих з’єднань.

Якщо користувач обирає модифікування VPN-з’єднання, тоді програма змінює параметри та властивості вказаного VPN-з’єднання.

Якщо користувач обирає видалення VPN-з’єднання, тоді програма видаляє вказане VPN-з’єднання.

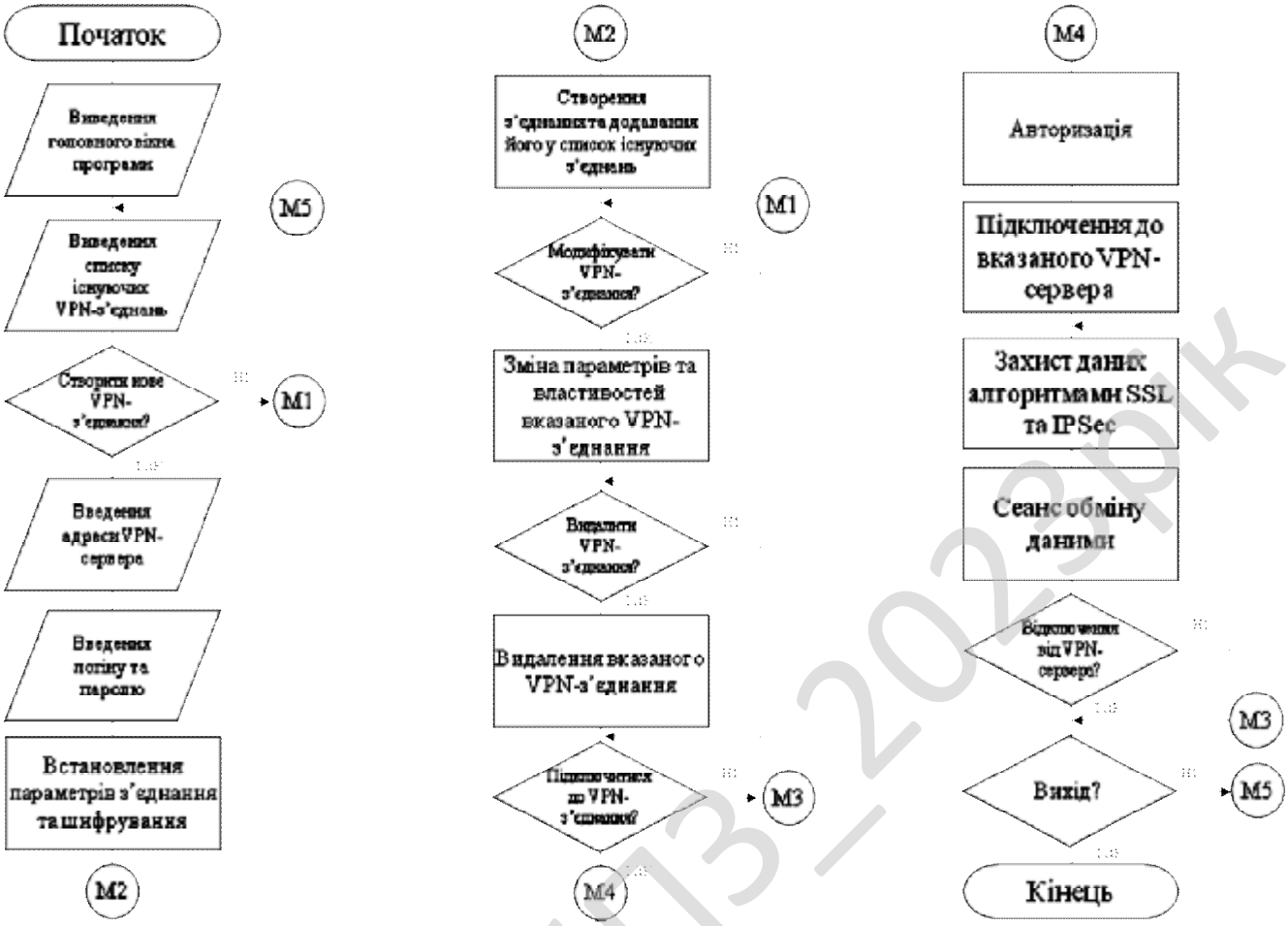

Рисунок 4.1 – Блок-схема основної програми

Якщо користувач обирає підключитися до VPN-з'єднання, тоді програма виконує наступні кроки:

– Відбувається процедура авторизації.

– Відбувається підключення до вказаного VPN-серверу.

– Дані захищаються алгоритмами SSL та IPSec.

– Відбувається сеанс обміну даними.

Якщо користувач обирає відключитися від VPN-з'єднання, то відбувається відключення й програма завершує свою роботу.

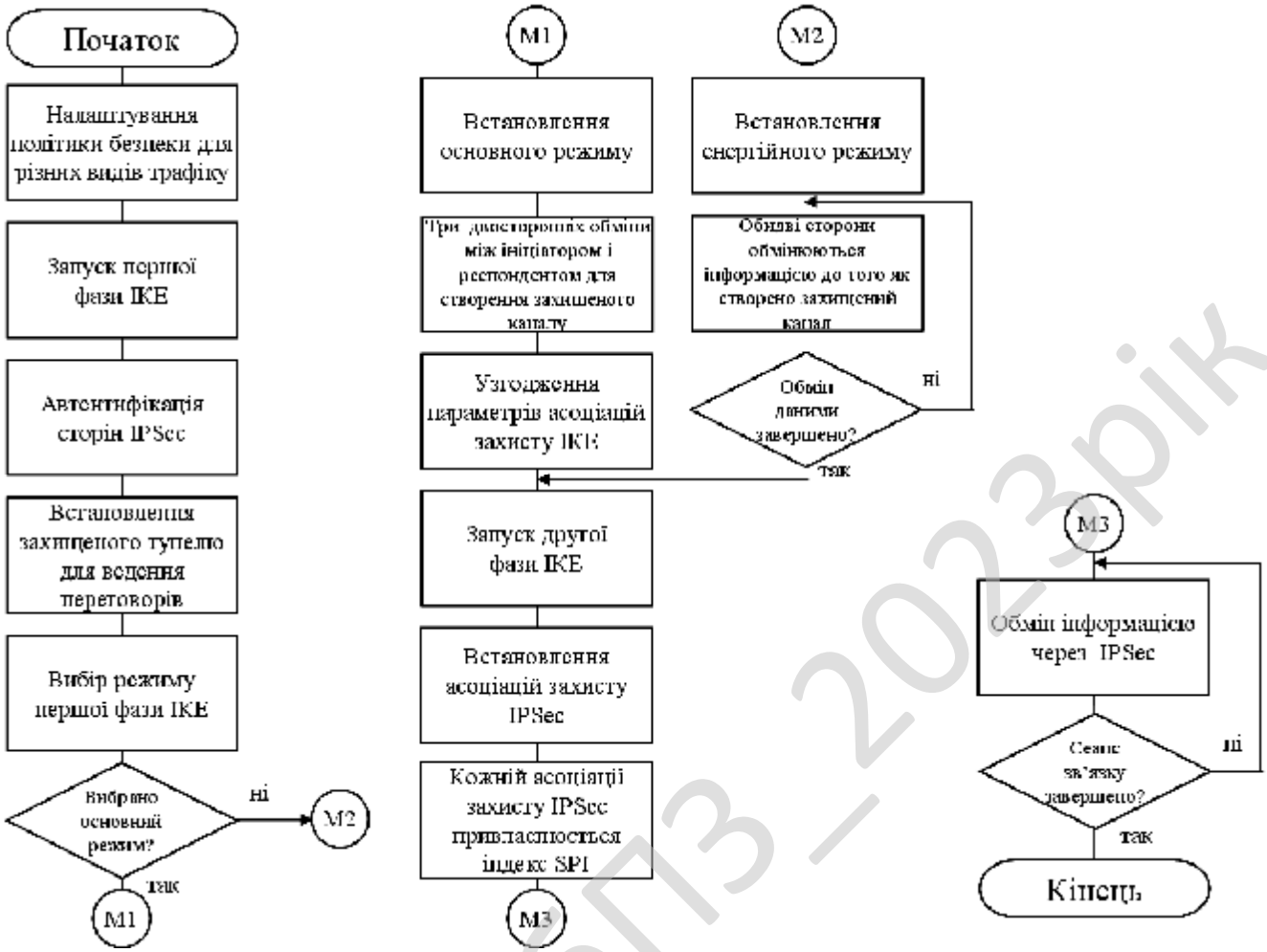

Рисунок 4.2 – Блок-схема підпрограми захисту інформації методом IPsec

На рисунку 4.2 зображено блок-схему підпрограми захисту інформації методом IPsec. Вона працює наступним чином.

Спершу відбувається налаштування політики безпеки для різних видів трафіку.

Далі запускається перша фаза IKE.

Наступним кроком є автентифікація сторін IPsec.

Після цього відбувається встановлення захищеного тунелю для ведення переговорів.

Наступною ітерацією є вибір режиму першої фази IKE.

Якщо обрано основний режим, то виконуються наступні дії:

– Встановлюється основний режим.

– Встановлюється три двосторонні обміни між ініціатором та респондентом для створення захищеного каналу.

– Узгоджуються параметри асоціацій захисту IKE.

У іншому випадку виконуються наступні дії:

– Встановлюється енергійний режим.

– Обидві сторони обмінюються інформацією до того, як створений захищений канал.

Якщо обмін даними завершено то виконуються наступні дії:

– Відбувається запуск другої фази IKE.

– Встановлюються асоціації захисту IPsec.

– Кожній асоціації захисту IPsec присвоюється індекс SPI.

– Відбувається обмін інформацією через IPsec до завершення сеансу зв’язку.

Після цього підпрограма закінчує свою роботу.

Практично всі механізми мережевої безпеки можуть бути реалізовані на третьому рівні еталонної моделі ISO/OSI. Більше того, IP-рівень можна вважати оптимальним для розміщення захисних засобів, оскільки при цьому досягається вдалий компроміс між захищеністю, ефективністю функціонування й прозорістю для додатків.

Стандартизованими механізмами IP езпеки можуть (і повинні)користуватися протоколи більше високих рівнів і, зокрема, що управляють протоколи, протоколи конфігурування й маршрутизації.

Засоби безпеки для IP описуються сімейством специфікацій IPsec, розроблених робочою групою IP Security.

Протоколи IPsec забезпечують керування доступом , цілісність поза з'єднанням, автентифікацію джерела даних, захист від відтворення, конфіденційності, частковий захист від аналізу трафіку.

Архітектура засобів безпеки для IP-рівня специфікована в документі. Це насамперед протоколи забезпечення автентичності (протокол автентифікуючого

заголовка – Authentication Header, AH) і конфіденційності (протокол інкапсулюючий захист вмісту- EncaPsulating Security Payload, ESP), а також механізми керування криптографічними ключами. На більше низькому архітектурному рівні розташовуються конкретні алгоритми шифрування, контролю цілісності й автентичності. Нарешті, роль фундаменту виконує так званих домен інтерпретації (Domain of Interpretation, DOI), що є, по суті, базою даних, що зберігає відомості про алгоритми, їхніх параметрах, протокольних ідентифікаторах і т.п.

Розподіл на рівні важливий для всіх аспектів інформаційних технологій. Там же, де бере участь ще й криптографія, важливість зростає подвійно, оскільки доводиться вважатися не тільки із чисто технічними факторами, але й з особливостями законодавства різних країн, з обмеженнями на експорт і/або імпорт криптозасобів.

Протоколи забезпечення автентичності й конфіденційності в IPsec не залежать від конкретних криптографічних алгоритмів. (Більше того, саме розподіл на автентичність і конфіденційність надає й розроблювачам, і користувачам додатковий ступінь волі в ситуації, коли до криптографічного відносять тільки шифрувальні засоби.) У кожній країні можуть застосовуватися свої алгоритми, що відповідають національним стандартам, але для цього, як мінімум, потрібно подбати про їхню реєстрацію в домені інтерпретації.

Алгоритмічна незалежність протоколів, на жаль, має й зворотний бік, що складається в необхідності попереднього узгодження набору застосовуваних алгоритмів і їхніх параметрів, підтримуваних сторонами, що спілкуються. Іншими словами, сторони повинні виробити загальний контекст безпеки (Security Association, SA) і потім використовувати такі його елементи, як алгоритми і їхні ключі. За формування контекстів безпеки в IPsec відповідає особливе сімейство протоколів, що буде розглянуто в наступних розділах.

Протоколи забезпечення автентичності й конфіденційності можуть застосовуватися у двох режимах: транспортному й тунельному. У першому

випадку захищається тільки вміст пакетів і, бути може, деякі поля заголовків. Як правило, транспортний режим використовується хостами. У тунельному режимі захищається весь пакет – він інкапсулюється в інший IP-пакет. Тунельний режим звичайно реалізують на спеціально виділених захисних шлюзах.

У наступних розділах ми докладно розглянемо основні елементи IPsec.

Контексти безпеки й керування ключами

Формування контекстів безпеки в IPsec розділено на дві фази. Спочатку створюється керуючий контекст, призначення якого – надати довірений канал, тобто автентифікований, захищений канал для вироблення (у рамках другої фази) протокольних контекстів і, зокрема, для формування криптографічних ключів, використовуваних протоколами AH і ESP.

У принципі, для функціонування механізмів IPsec необхідні тільки протокольні контексти; керуючий відіграє допоміжну роль. Більше того, явне виділення двох фаз ускладнює формування ключів, якщо розглядати останнє як однократну дію. Проте, з архітектурних міркувань керуючі контексти не тільки можуть, але й повинні існувати, оскільки обслуговують всі протокольні рівні стека TCP/IP, концентруючи в одному місці необхідну функціональність. Перша фаза починається в ситуації, коли взаємодіючі сторони не мають загальних секретів (загальних ключів) і не впевнені в автентичності один одного. Якщо із самого початку не створити довірений канал, то для виконання кожної керуючої дії із ключами (їхня модифікація, видача діагностичних повідомлень і т.п.) у кожному протоколі (AH, ESP, TLS і т.д.) цей канал прийде формувати заново.

Загальні питання формування контекстів безпеки й керування ключами висвітлюються в специфікації – "Контексти безпеки й керування ключами в Internet" (Internet Security Association and Key Management Protocol, ISAKMP). Тут уводяться дві фази вироблення протокольних ключів, визначаються види керуючих інформаційних обмінів і використовувані формати заголовків і даних. Іншими словами будується протокольно-незалежний каркас.

Існує кілька способів формування керуючого контексту. Вони розрізняються двома показниками:

– використовуваним механізмом вироблення загального секретного ключа;

– ступенем захисту ідентифікаторів сторін, що спілкуються.

У найпростішому випадку секретні ключі задаються заздалегідь (ручний метод розподілу ключів). Для невеликих мереж такий підхід цілком працездатний, але він не є масштабованим. Остання властивість може бути забезпечене при автоматичному вироблення й розподілі секретних ключів у рамках протоколів, заснованих на алгоритмі Діффі-Хелмана. Приклад тому – "Протокол для обміну ключами в Internet" (The Internet Key Exchange, IKE).

При формуванні керуючого контексту ідентифікатори сторін, що спілкуються (наприклад, IP-адреси) можуть передаватися у відкритому виді або шифруватися. Оскільки ISAKMP передбачає функціонування в режимі клієнт/сервер (тобто ISAKMP-сервер може формувати контекст для клієнта), приховання ідентифікаторів деякою мірою підвищує захищеність від пасивного прослуховування мережі. Послідовність переданих повідомлень, що дозволяють сформувати керуючий контекст і забезпечують захист ідентифікаторів, виглядає в такий спосіб.

У першому повідомленні (1) ініціатор направляє пропозиції по наборі захисних алгоритмів і конкретних механізмів їхньої реалізації. Пропозиції впорядковуються по ступені переваги (для ініціатора). У відповідному повідомленні (2) партнер інформує про зроблений вибір – які алгоритми й механізми його влаштовують. Для кожного класу захисних засобів (генерація ключів, автентифікація, шифрування) вибирається тільки один елемент.

У повідомленнях (3) і (4) ініціатор і партнер відправляють свої частини ключового матеріалу, необхідні для вироблення загального секретного ключа (ми опускаємо деталі, специфічні для алгоритму Діффі-Хелмана). Одноразові номери (nonce) являють собою псевдовипадкові величини, що служать для захисту від відтворення повідомлень.

За допомогою повідомлень (5) і (6) відбувається обмін ідентифікаційною інформацією, підписаної (з метою автентифікації) секретним ключем відправника й зашифрованої виробленим на попередніх кроках загальним секретним ключем. Для автентифікації передбачається використання сертифікатів відкритих ключів. Відзначимо, що в число даних, що підписуються, входять одноразові номери.

У представленому виді протокол формування керуючого контексту захищає від атак, вироблених нелегальним посередником, а також від нелегального перехоплення з'єднань. Для захисту від атак на доступність, для яких характерно насамперед нав'язування інтенсивних обчислень, властиві криптографії з відкритим ключем, застосовуються так звані ідентифікуючі ланцюжки (cookies). Ці ланцюжки, формовані ініціатором і його партнером з використанням поточного часу (для захисту від відтворення), насправді присутні у всіх ISAKMP-повідомленнях і в сукупності ідентифікують керуючий контекст (у першому повідомленні, по зрозумілих причинах, фігурує тільки ланцюжок ініціатора). Якщо зловмисник намагається "завалити" кого-небудь запитами на створення керуючого контексту, підробляючи при цьому свою IP-адресу, то в повідомленні (3) він не зможе пред'явити ідентифікуючий ланцюжок партнера, тому до вироблення загального секретного ключа й, тим більше, електронного підпису й повномасштабної перевірки автентичності справа попросту не дійде.

Керуючі контексти є двонаправленими в тому розумінні, що кожна зі сторін, що спілкуються, може ініціювати з їхньою допомогою виробіток нових протокольних контекстів або інші дії. Для передачі ISAKMP-повідомлень використовується будь-який протокол, однак у якості стандартного прийнятий UDP з номером порту 500.

Протокольні контексти й політика безпеки

Системи, що реалізують IPsec, повинні підтримувати дві бази даних:

– базу даних політики безпеки (Security Policy Database, SP);

– базу даних протокольних контекстів безпеки (Security Association Database, SAD).

Всі IP-пакети (вхідні й вихідні) зіставляються з упорядкованим набором правил політики безпеки. При зіставленні використовується селектор, що фігурує в кожному правилі, - сукупність аналізованих полів мережевого рівня й більше високих протокольних рівнів. Перше підходяще правило визначає подальшу долю пакета:

– пакет може бути ліквідований;

– пакет може бути оброблений без участі засобів IPsec;

– пакет повинен бути оброблений засобами IPsec з урахуванням набору протокольних контекстів, асоційованих із правилом.

Таким чином, системи, що реалізують IPsec, функціонують як міжмережеві екрани, фільтруючи й перетворюючи потоки даних на основі попередньо заданої політики безпеки.

Далі детально розглянемо контексти й політику безпеки, а також порядок обробки мережевих пакетів.

Протокольний контекст безпеки в IPsec – це односпрямоване "з'єднання" (від джерела до одержувача), що надає обслуговуються потокам, що, даних набір захисних сервісів у рамках якогось одного протоколу (AH або ESP). У випадку симетричної взаємодії партнерам прийде організувати два контексти (по одному в кожному напрямку). Якщо використовуються й AH, і ESP, буде потрібно чотири контексти.

Елементи бази даних протокольних контекстів містять наступні поля (у кожному конкретному випадку деякі значення полів будуть порожніми):

– використовуваний у протоколі AH алгоритм автентифікації, його ключі й т.п.;

– використовуваний у протоколі ESP алгоритм шифрування, його ключі, початковий вектор і т.п.;

– використовуваний у протоколі ESP алгоритм автентифікації, його ключі й т.п.;

– час життя контексту;

– режим роботи IPsec: транспортний або тунельний;

– максимальний розмір пакетів;

– група полів (лічильник, вікно, прапори) для захисту від відтворення пакетів.

Користувачами протокольних контекстів, як правило, є прикладні процеси. Загалом кажучи, між двома вузлами мережі може існувати довільне число протокольних контекстів, тому що число додатків у вузлах довільно. Відзначимо, що як користувачів керуючих контекстів звичайно виступають вузли мережі (оскільки в цих контекстах бажано зосередити загальну функціональність, необхідну сервісам безпеки всіх протокольних рівнів еталонної моделі для керування криптографічними ключами).

Керуючі контексти – двосторонні, тобто кожний з партнерів може ініціювати новий ключовий обмін. Пара вузлів може одночасно підтримувати кілька активних керуючих контекстів, якщо є додатки з істотно різними криптографічними вимогами. Наприклад, припустимо вироблення частини ключів на основі попередньо розподіленого матеріалу, у той час як інша частина породжується по алгоритму Діффі-Хелмана.

Протокольний контекст для IPsec ідентифікується цільовим IP-адресом, протоколом (AH або ESP), а також додатковою величиною – індексом параметрів безпеки (Security Parameter Index, SP). Остання величина необхідна, оскільки можуть існувати кілька контекстів з однаковими IP-адресами й протоколами. Далі буде показано, як використовуються індекси SP при обробці вхідних пакетів.

IPsec зобов'язує підтримувати ручне й автоматичне керування контекстами безпеки й криптографічних ключів. У першому випадку всі системи заздалегідь забезпечуються ключовим матеріалом і іншими даними, необхідними для захищеної взаємодії з іншими системами. У другому – матеріал і дані виробляються динамічно, на основі певного протоколу – IKE, підтримка якого обов'язкова.

Протокольний контекст створюється на базі керуючого з використанням ключового матеріалу й засобів автентифікації й шифрування останнього.

Коли вироблявся керуючий контекст, для нього було створено три види ключів:

– SKEYID_d – ключовий матеріал, використовуваний для генерації протокольних ключів.

– SKEYID_a – ключовий матеріал для автентифікації.

– SKEYID_e – ключовий матеріал для шифрування.

Всі перераховані види ключів задіяні в обміні. Ключем SKEYID_e шифруються повідомлення. Ключ SKEYID_a служить аргументом хеш-функцій і тим самим автентифікує повідомлення. Нарешті, протокольні ключі – результат застосування псевдовипадкової (хеш) функції до SKEYID_d з додатковими параметрами, у число яких входять одноразові номери ініціатора й партнера. У результаті створення протокольного контексту виявляється автентифікованим, захищеним від несанкціонованого ознайомлення, від відтворення повідомлень і від перехоплення з'єднання.

Повідомлення (1) і (2) можуть нести додаткове навантаження, наприклад, дані для вироблення "зовсім нових" секретних ключів або ідентифікатори клієнтів, від імені яких ISAKMP-сервери формують протокольний контекст. Відповідно до протоколу IKE, за один обмін (щоскладається із трьох повідомлень) формується два односпрямованих контексти – по одному в кожному напрямку. Одержувач контексту задає для нього індекс параметрів безпеки (SP), що допомагає знаходити контекст для обробки прийнятих пакетів IPsec.

Строго говорячи, протокольні контексти відіграють допоміжну роль, будучи лише засобом проведення в життя політики безпеки; вона повинна бути задана для кожного мережевого інтерфейсу із задіяними засобами IPsec і для кожного напрямку потоків даних (вхідні/вихідні). Відповідно до специфікацій IPsec, політика розраховується на безконтекстну (незалежну) обробку IP-пакетів,

у дусі сучасних фільтруючих маршрутизаторів. Зрозуміло, повинні існувати засобу адміністрування бази даних SP, так само, як і засобу адміністрування бази правил міжмережевого екрана, однак цей аспект не входить до числа стандартизуємих.

Із зовнішньої точки зору, база даних політики безпеки (SP) являє собою впорядкований набір правил. Кожне правило задається як пара:

– сукупність селекторів;

– сукупність протокольних контекстів безпеки.

Селектори служать для відбору пакетів, контексти задають необхідну обробку. Якщо правило посилається на неіснуючий контекст, воно повинне містити достатню інформацію для його (контексту) динамічного створення. Очевидно, у цьому випадку потрібна підтримка автоматичного керування контекстами й ключами. У принципі, функціонування системи може починатися із завдання бази SP при порожній базі контекстів (SAD); остання буде наповнюватися в міру необхідності.

Дифференційованість політики безпеки визначається селекторами, ужитими в правилах. Наприклад, пари взаємодіючих хостів може використовувати єдиний набір контекстів, якщо в селекторах фігурують тільки IP-адреси; з іншого боку, набір може бути своїм для кожного додатка, якщо аналізуються номери TCP- і UDP-портів. Аналогічно, два захисних шлюзи здатні організувати єдиний тунель для всіх що обслуговуються хостів або ж розщепити його (шляхом організації різних контекстів) по парах хостів або навіть додатків.

Всі реалізації IPsec повинні підтримувати селекцію наступних елементів:

– вихідна й цільова IP-адреси (адреси можуть бути індивідуальними й груповими, у правилах допускаються діапазони адрес і метасимволи "будь-який";

– ім'я користувача або вузла у форматі DNS або X.500;

– транспортний протокол;

– номера вихідного й цільового портів (тут також можуть використовуватися діапазони й метасимволи).

Обробка вихідні й вхідного трафіку не є симетричною. Для вихідних пакетів проглядається база SP, перебуває підходяще правило, витягають асоційовані з ним протокольні контексти й застосовуються відповідні механізми безпеки. У вхідних пакетах для кожного захисного протоколу вже проставлене значення SP, однозначно визначальний контекст. Перегляд бази SP у такому випадку не потрібно; можна вважати, що політика безпеки враховувалася при формуванні відповідного контексту. (Практично це означає, що ISAKMP-пакети мають потребу в особливому трактуванню, а правила з відповідними селекторами повинні бути включені в SP.)

Відзначена асиметрія, на наш погляд, відбиває певну незавершеність архітектури IPsec. У більше ранньому документі RFC 1825 поняття бази даних політики безпеки й селекторів були відсутні. У новій редакції специфікований перегляд бази SP як обов'язковий для кожного вихідного пакета, але не змінена обробка вхідних пакетів. Звичайно, добування контексту по індексі SP ефективніше, ніж перегляд набору правил, але при такому підході, щонайменше, утрудняється оперативна зміна політики безпеки. Що стосується ефективності перегляду правил, те її можна підвищити методами кешування, широко використовуваними при реалізації IP.

Можливо, ще більш серйозним недоліком є неможливість узагальнення запропонованих процедур формування контекстів (керуючих і протокольних) на багатоадресний випадок. У поточних специфікаціях IPsec змішуються дві різні речі – область дії контексту (зараз це однобічний або двосторонній потік даних) і спосіб його ідентифікації (по індексі SP або парі ідентифікуючих ланцюжків). Виходить, що спосіб ідентифікації (іменування) нав'язує трактування області дії, що представляється невірним. На наш погляд, питання іменування можуть вирішуватися локально, а область дії контексту потенційно повинна поширюватися на довільне число партнерів.

Забезпечення автентичності IP-пакетів

Протокол автентифікуючого заголовка (Authentication Header, AH) служить в IPsec для забезпечення цілісності пакетів і автентифікації джерела даних, а також для захисту від відтворення раніше посланих пакетів. AH захищає дані протоколів більше високих рівнів і ті поля IP-Заголовків, які не міняються на маршруті доставки або міняються передбачуваним образом. (Відзначимо, що число "непередбачених" полів невелике – це Prio. (Traffic Class), Flow Label і Hop Limit. Передбачувано міняється цільова адреса при наявності додаткового заголовка вихідної маршрутизації) .

Пояснимо зміст полів, специфічних для AH:

– індекс параметрів безпеки (SP) – 32-бітне значення, обиране одержувачем пакетів з AH-Заголовками як ідентифікатор протокольного контексту (див. вище розділ "Протокольні контексти й політика безпеки");

– порядковий номер – беззнакове 32-бітне ціле, нарощуване від пакета до пакета. Відправник зобов'язаний підтримувати цей лічильник, у той час як одержувач може (але не зобов'язаний) використовувати його для захисту від відтворення. При формуванні протокольного контексту обидві взаємодіючі сторони роблять свої лічильники нульовими, а потім погодженим образом збільшують їх. Коли значення порядкового номера стає максимально можливим, повинен бути сформований новий контекст безпеки;

– автентифікаційні дані – поле змінної довжини, що містить імітовставку (криптографічну контрольну суму, Integrity Check Value, ICV) пакета; спосіб його обчислення визначається алгоритмом автентифікації.

Для обчислення автентифікованих імітовставок можуть застосовуватися різні алгоритми. Специфікаціями пропонується обов'язкова підтримка двох алгоритмів, заснованих на застосуванні хеш-функцій із секретними ключами:

– HMAC-MD5 (Hashed Message Authentication Code – Message Digest version 5);

– HMAC-SHA-1 (Hashed Message Authentication Code – Secure Hash Algorithm version 1).

Забезпечення конфіденційності мережевого трафіку

Протокол інкапсулюючий захисту вмісту (Encapsulating Security Payload, ESP) надає три види сервісів безпеки:

– забезпечення конфіденційності (шифрування вмісту IP-пакетів, а також частковий захист від аналізу трафіку шляхом застосування тунельного режиму);

– забезпечення цілісності IP-пакетів і автентифікації джерела даних;

– забезпечення захисту від відтворення IP-пакетів.

Можна бачити, що функціональність ESP ширше, ніж в AH (додається шифрування); взаємодія цих протоколів ми докладніше розглянемо пізніше. Тут же відзначимо, що ESP не обов'язково надає всі сервіси, але або конфіденційність, або автентифікація повинні бути задіяні. Формат заголовка ESP виглядає трохи незвичайно. Причина в тім, що це не стільки заголовок, скільки обгортка (інкапсулююча оболонка) для зашифрованого вмісту. Наприклад, посилання на наступний заголовок не можна виносити в початок, у незашифровану частину, тому що вона втратиться конфіденційності.

Поля "Індекс параметрів безпеки (SP)", "Порядковий номер" і "Автентифікационные дані" (останнє є присутнім тільки при включеної автентифікації) мають той же зміст, що й для AH. Правда, ESP автентифікує лише зашифровану частину пакета (плюс два перші поля заголовка).

Застосування протоколу ESP до вихідних пакетів можна уявляти собі в такий спосіб. Назвемо залишком пакета ту його частину, що міститься після передбачуваного місця вставки заголовка ESP. При цьому не важливо, який режим використовується – транспортний або тунельний. Кроки протоколу такі:

– залишок пакета копіюється в буфер;

– до залишку приписуються байти, що доповнюють, їхнє число й номер (тип) першого заголовка залишку, для того щоб номер був притиснутий до границі 32-бітного слова, а розмір буфера задовольняв вимогам алгоритму шифрування;

– поточний уміст буфера шифрується;

– у початок буфера приписуються поля "Індекс параметрів безпеки (SP)" і "Порядковий номер" з відповідними значеннями;

– поповнений уміст буфера автентифікується, у його кінець міститься поле "Автентифікаційні дані";

– у новий пакет листуються початкові заголовки старого пакета й кінцевий уміст буфера.

Таким чином, якщо в ESP включені й шифрування, і автентифікація, те автентифікується зашифрований пакет. Для вхідних пакетів дії виконуються у зворотному порядку, тобто спочатку виробляється автентифікація. Це дозволяє не витрачати ресурси на розшифровку підроблених пакетів, що в якімсь ступені захищає від атак на доступність.

Два захисних протоколи – AH і ESP – можуть комбінуватися різними способами. При виборі транспортного режиму AH повинен використовуватися після ESP (аналогічно тому, як у рамках ESP автентифікація йде слідом за шифруванням). У тунельному режимі AH і ESP застосовуються, строго говорячи, до різного (вкладеним) пакетам, число припустимих комбінацій тут більше (хоча б тому, що можливо багаторазову вкладеність тунелів з різними початковими й/або кінцевими крапками).

Сукупність механізмів, пропонована в рамках IPsec, є досить потужною й гнучкою. IPsec – це основа, на якій може будуватися реалізація віртуальних приватних мереж (VPN), забезпечуватися захищена взаємодія мобільних систем з корпоративною мережею, захист прикладних потоків даних і т.п.

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді

продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В алгоритмі використовуються 8x16 S- блоки, засновані на бент-функції, операції XOR і модулярной арифметиці (медулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт

Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 –

безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256 біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст m_1 ... m_{64}, ключ K = k_1 ... k_{128}.

ВИХІД: зашифрований текст c_1 ... C_{64}.

1. (розгорнення ключа) становить 16 пару підключів {Km_i, Kr_i} отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

2. (L_0, R_0) <- (m_1. .. m_{64}). (Розділяє текст на ліву й праву 32-бітні половини L_0 = m_1 ... m_{32} і R_0 = m_{33} ... m_{64}).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: L_i = R_{i-1}; R_i = L_{i-1} ^ F(R_{i-1},Km_i,Kr_i), де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. c_1 ... c_{64} <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 32-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і

"-" додавання й вирахування по модулю 2 ** 32, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ \ S2[I_b]) - (S3[I_c]))+ S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] - S2[I_b]) + (S3[I_c]))\ \ S4[I_d]$
Раунди	$I = ((Km_i - R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ \ (S3[I_c]))- S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни. Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування / дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K32.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K32 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1, ..., Km_{16} 32-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 32-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми (створення VPN-з’єднань). З рисунку видно, що інтерфейс користувача складається з наступних блоків:

– Блок меню.

– Блок кнопок швидкого доступу до функцій програми.

– Блок закладок.

– Вікно відображення існуючих підключень.

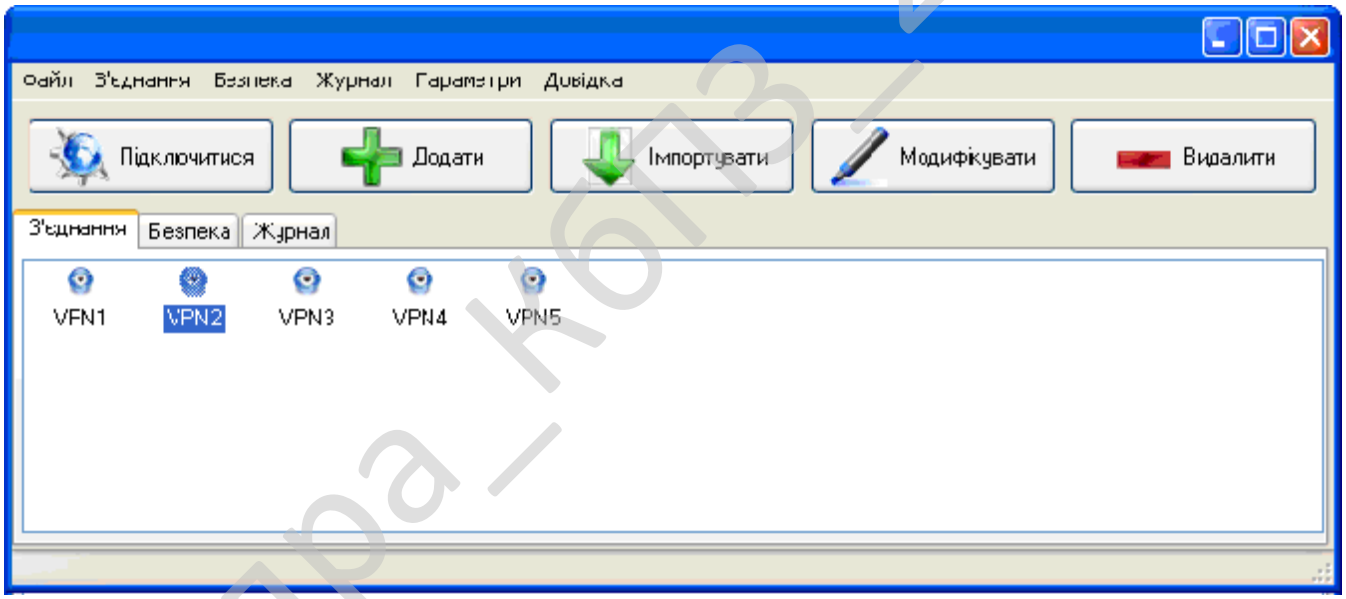

Рисунок 5.1 – Головне вікно програми (створення VPN-з’єднань)

Блок меню складається з наступних елементів:

– Файл.

– З’єднання.

– Безпека.

– Журнал.

– Параметри.

– Довідка.

Блок клавіш швидкого доступу до елементів програми складається з наступних елементів:

– Підключитися.

– Додати.

– Імпортувати.

– Модифікувати.

– Видалити.

Блок закладок складається з наступних елементів:

– З’єднання.

– Безпека.

– Журнал.

У вікні відображення існуючих підключень наведено усі VPN-з’єднання, які існують у мережі для даного VPN-підключення.

На рисунку 5.2 зображено вікно довідки, яке надає наступну інформацію:

– Тема проекту.

– Ким розроблений проект.

– Хто керівник проекту.

– Місце виконання проекту.

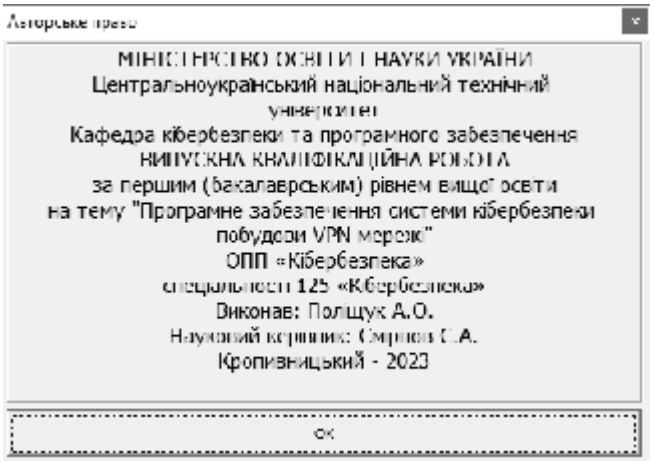

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки побудови VPN-мережі.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем побудови VPN-мережі.

– Досліджена система побудови VPN-мережі.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки побудови VPN-мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання побудови VPN-мережі.

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки побудови VPN-мережі. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CAST-128.

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. 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, pp. 1-12. **(Scopus).**

2. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

3. 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. **(Scopus).**

4. 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. **(Scopus).**

5. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

6. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

7. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

8. 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.* **(Scopus)**.

9. 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. **(Scopus)**.

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

11. Smirnov O.A., 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. **(Scopus)**.

12. 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. **(Scopus)**.

13. 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. **(Scopus)**.

14. 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. **(Scopus)**.

15. 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. **(Scopus)**.

16. 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. **(Scopus)**.

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

18. 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. **(Scopus)**.

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

20. 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., **(Scopus)**.

21. 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. (**Scopus)**.

22. 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. **(Scopus)**.

23. 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. **(Scopus)**.

24. 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. **(Scopus)**.

25. 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. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

27. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures**. Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

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

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

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

31. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

32. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

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

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

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

36. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного

процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

37. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

38. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

39. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

40. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

41. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

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

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

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

45. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

46. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

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

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

49. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

50. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки побудови VPN-мережі.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 12-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки побудови VPN-мережі.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки побудови VPN-мережі;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 95 аркушів.

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.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 14.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки побудови VPN-мережі

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 54

Літера: РП

Кропивницький – 2023 року

VPN_FinsWind.pas - VPN-з'єднання

```
unit VPN_FinsWind;

interface

uses
  Windows, Messages, CommCtrl, VPN_Windows, VPN_Controls, VPN_FileInfo,
VPN_SysUtils,
  VPN_MyMsgBox, VPN_Ole2, VPN_Active, VPN_ShlObj, VPN_RasApi, VPN_Resources;

function FinsDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;

implementation

//

function FinsDlgProc_OnWmInitDialog(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
lParam: LPARAM): LRESULT;
var
  bldfnt: HFONT;
begin

  //

  hApp[2] := hWnd;

  //

  bldfnt := HFONT(SendMessage(GetDlgItem(hApp[0], IDC_STATIC_WELCOME),
    WM_GETFONT, 0, 0));

  if (bldfnt <> 0) then
    SendMessage(GetDlgItem(hApp[2], IDC_STATIC_FINISH), WM_SETFONT,
      Integer(bldfnt), Integer(TRUE));

  //

  Result := 0;

end;

//

function FinsDlgProc_OnWmNotify(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  pnmh   : PNMHdr;
  dwRes  : DWORD;
  pszText: WideString;

  //

  function GetNextItemID(pidl: PItemIDList): PItemIDList;
  var
    cb: DWORD;
  begin
    Result := nil;
    if (pidl = nil) then
      Exit;
    cb := pidl.mkid.cb;
    if (cb = 0) then
      Exit;
    pidl := PItemIDList(Cardinal(pidl) + cb);
    if (pidl.mkid.cb <> 0) then
```

```
    Result := pidl;
end;

//

function GetPIDSize(pidl: PItemIDList): DWORD;
begin
  Result := 0;
  if (pidl <> nil) then
  begin
    Result := SizeOf(pidl.mkid.cb);
    while (pidl <> nil) do
    begin
      Inc(Result, pidl.mkid.cb);
      pidl := GetNextItemID(pidl);
    end;
  end;
end;

//

function IsDesktopFolder(pidl: PItemIDList): Boolean;
begin
  if Assigned(pidl) then
    Result := (pidl.mkid.cb = 0)
  else
    Result := FALSE;
end;

//

function ConcatPIDL(destpidl, srcpidl: PItemIDList): PItemIDList;
var
  cb1: DWORD;
  cb2: DWORD;
  pmc: IMalloc;
  hr : HRESULT;
begin
  Result := nil;
  hr := SHGetMalloc(pmc);
  if SUCCEEDED(hr) then
  begin
    cb1 := 0;
    cb2 := 0;
    if Assigned(destpidl) then
    begin
      if not IsDesktopFolder(destpidl) then
        cb1 := GetPIDSize(destpidl) - SizeOf(destpidl^.mkid.cb);
    end;
    if Assigned(srcpidl) then
      cb2 := GetPIDSize(srcpidl);
    Result := pmc.Alloc(cb1 + cb2);
    if Assigned(Result) then
    begin
      if Assigned(destpidl) then
        CopyMemory(Result, destpidl, cb1);
      if Assigned(srcpidl) then
        CopyMemory(Pointer(DWORD(Result) + cb1), srcpidl, cb2);
    end;
    pmc := nil;
  end;
end;

//

procedure CreateShellVpnLink(pszEntry: WideString);
var
  pMalloc    : IMalloc;
  Desktop    : IShellFolder;
```

```
    pidlDesktop: PItemIDList;
    pszPath    : Array [0..MAX_ PATH-1] of WideChar;
    pidlConnect: PItemIDList;
    Network    : IShellFolder;
    Items      : IEnumIDList;
    pidl2      : PItemIDList;
    dwFetched  : Cardinal;
    Connection : STRRET;
    ObjectName : WideString;
    pfLink     : IUnknown;
    isLink     : IShellLink;
    ipFile     : IPersistFile;
    pidl3      : PItemIDList;
    szFileName : WideString;
  begin
    CoInitialize(nil);
    try
      // визначається оболонка
      if (SHGetMalloc(pMalloc) = S_OK) then
      try
        // визначається простір імен корню директорій
        if (SHGetDesktopFolder(Desktop) = S_OK) then
        try
          if (SHGetSpecialFolderLocation(0, CSIDL_DESKTOP, pidlDesktop) = S_OK)
then
          try
            ZeroMemory(@pszPath, SizeOf(pszPath));
            SHGetPathFromIDList(pidlDesktop, @pszPath);
            if (SHGetSpecialFolderLocation(0, CSIDL_CONNECTIONS, pidlConnect) =
S_OK) then
            try
              Desktop.BindToObject(pidlConnect, nil, IIVPN_IShellFolder,
Network);
              Network.EnumObjects(0, SHCONTVPN_NONFOLDERS, Items);
              while (Items.Next(1, pidl2, dwFetched) = S_OK) do
              try
                if (dwFetched > 0) and Assigned(pidl2) then
                try
                  Network.GetDisplayNameOf(pidl2, SHGDN_NORMAL, Connection);
                  ObjectName := Connection.pOleStr;
                  if (lstrcmpi(@ObjectName[1], @pszEntry[1]) = 0) then
                  try
                    CoCreateInstance(CLSIVPN_ShellLink, nil,
CLSCTX_INPROC_SERVER, IUnknown, pfLink);
                    isLink := pfLink as IShellLink;
                    ipFile := pfLink as IPersistFile;
                    pidl3 := ConcatPIDL(pidlConnect, pidl2);
                    isLink.SetIDList(pidl3);
                    szFileName := Format('%s\%s.lnk',
[ExcludeTrailingPathDelimiter(pszPath), pszEntry]);
                    ipFile.Save(@szFileName[1], FALSE);
                    pMalloc.Free(pidl3);
                  finally
                    {
                    pfLink := nil;
                    isLink := nil;
                    ipFile := nil;
                    }
                  end;
                finally
                  pMalloc.Free(pidl2); // папка версій
                end;
              finally
              end;
            finally
              Network := nil;
              pMalloc.Free(pidlConnect); // папка версій
            end;
          finally
```

```
          pMalloc.Free(pidlDesktop); // папка версій
        end;
      finally
        Desktop := nil; // версії простору імен корню директорій
      end;
    finally
      pMalloc := nil; //
    end;
  finally
    CoUninitialize;
  end;
end;

//

function CreateRasVpnConnection(szEntryName, szPhoneName, szUserName,
szPassword: WideString): LRESULT;
var
  osvi     : TOSVersionInfo;
  rEntry   : RASENTRYW;
  rDial    : RASDIALPARAMSW;
  lpCred   : RASCREDENTIALSW;
  dwSize   : Integer;
  EntrySize: Integer;
  InfoSize : Integer;
  dwFlags  : DWORD;
  dwFlags2 : DWORD;
  dwRes    : DWORD;
begin

  // заповнюємо структуру RASENTRY і довідаємося потрібний розмір для
коректного виклику
  // функції RasSetEntryProperties
  dwSize := SizeOf(RASENTRYW);
  RasGetEntryProperties(nil, nil, nil, EntrySize, nil, InfoSize);
  if (EntrySize < dwSize) then
    dwSize := EntrySize;

  // Задаємо прапорові параметри VPN з'єднання
  dwFlags :=

    // Вкладка 'Параметри', прапор 'Запитувати ім'я, пароль, сертифікат і
т.д.', вмик
    RASEO_PreviewUserPw or

    // Вкладка 'Загальні', прапор 'При підключенні вивести значок в області
повідомлень', вмик
    RASEO_ModemLights or

    // Вкладка 'Загальні', прапор 'Відображати хід підключення', вмик
    RASEO_ShowDialingProgress or

    // Використовувати основний шлюз
    RASEO_RemoteDefaultGateway or

    // Зашифрований пароль буде використовуватися при перевірці дійсності із
сервером
    RASEO_RequireEncryptedPw or

    // Використовувати автоматично логін, пароль і домен з Windows
    RASEO_RequireDataEncryption or

    // Пароль буде зашифрований за схемою Microsoft
    RASEO_RequireMsEncryptedPw;
  dwFlags2 :=

    // Вкладка 'Параметри', прапор 'Погоджувати багатоканальне підключення для
одноканальних', вимк
```

```
      RASEO2_DontNegotiateMultilink or
      // Вкладка 'Параметри', прапор 'Передзвонити при розриві зв'язку', вмик
      RASEO2_ReconnectIfDropped;

    // Заповнюємо структуру RASENTRY
    ZeroMemory(@rEntry, SizeOf(RASENTRYW));
    rEntry.dwSize                   := dwSize;
    rEntry.dwfOptions               := dwFlags;

    // Тип використовуваного протоколу = TCP/IP
    rEntry.dwfNetProtocols          := RASNP_Ip;

    // Тип використовуваного протоколу сервера віддаленого доступу = Point-to-
Point Protocol (PPP)
    rEntry.dwFramingProtocol        := RASFP_Ppp;

    // Тип створюваного підключення - Віртуальна приватна мережа (VPN)
    rEntry.dwType                   := RASET_Vpn;

    // Значення списку, що випадає, 'Тип VPN' = 'Автоматично'
    // Викликається спочатку тільки PPTP, якщо ж спроба закінчується невдачею,
то викликається L2TP
    rEntry.dwVpnStrategy            := VS_Default;
    rEntry.dwfOptions2              := dwFlags2;

    // Вкладка 'Безпека', прапор 'Потрібне шифрування даних', вмик
    // Діалог 'Додаткові параметри безпеки', список 'Шифрування даних' =
'обов'язкове'
    // Тип шифрування даних при підключенні = Шифрування не використовується
    rEntry.dwEncryptionType         := ET_None;

    // Використовуємо з'єднання пристроїв з безліччю «підвходів»
    rEntry.dwDialMode               := RASEDM_DialAll;

    // Вкладка 'Параметри', 'Число повторень набору номера' = 3
    rEntry.dwRedialCount            := 3;

    // Вкладка 'Параметри', 'Інтервал між повтореннями' = 60 секунд
    rEntry.dwRedialPause            := 60;
    lstrcpy(rEntry.szLocalPhoneNumber, @szPhoneName[1]);
    lstrcpy(rEntry.szDeviceType, RASDT_Vpn);

    // Створюємо нове підключення VPN з потрібними параметрами
    dwRes := RasSetEntryProperties(nil, @szEntryName[1], @rEntry, dwSize, nil,
0);
    case dwRes of
      ERROR_SUCCESS:
        Begin

          // виконуємо перевірку версії ОС. починаючи з ОС Win XP і старше,
логін і
          // пароль (фактично це нам і потрібно) можна змінити функцією
          // RasSetCredentials, а в попередніх ОС можна за допомогою функції
          // RasSetEntryDialParams. в Win XP ще можна змінити пароль функцією
          // RasSetEntryDialParams, а от уже в Win Vista і старше не вийде.

          ZeroMemory(@osvi, SizeOf(TOSVersionInfo));
          osvi.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
          VPN_Windows.GetVersionEx(osvi);

          if ((osvi.dwPlatformId = VER_PLATFORM_WIN32_NT) and
            (osvi.dwMajorVersion >= 5) and (osvi.dwMinorVersion >= 1)) then
            Begin

              // Заповнюємо структуру RASCREDENTIALS
              ZeroMemory(@lpCred, SizeOf(RASCREDENTIALSW));
              lpCred.dwMask := RASCM_UserName or RASCM_Password;
              lpCred.dwSize := SizeOf(RASCREDENTIALSW);
              lstrcpy(lpCred.szUserName, @szUserName[1]);
```

```
              lstrcpy(lpCred.szPassword, @szPassword[1]);
              // Змінюємо логін і пароль створеного підключення
              dwRes := RasSetCredentials(nil, @szEntryName[1], lpCred, FALSE);
            end
          else
            begin

              // Заповнюємо структуру RASDIALPARAMS
              ZeroMemory(@rDial, SizeOf(RASDIALPARAMSW));
              rDial.dwSize := SizeOf(RASDIALPARAMSW);
              lstrcpy(rDial.szEntryName, @szEntryName[1]);
              lstrcpy(rDial.szUserName, @szUserName[1]);
              lstrcpy(rDial.szPassword, @szPassword[1]);

              // Змінюємо логін і пароль створеного підключення
              dwRes := RasSetEntryDialParams(nil, @rDial, FALSE);
            end;
        end;
    end;
    Result := dwRes;
  end;

begin

  //

  pnmh := PNMHdr(lParam);

  case pnmh.code of

    //

    PSN_SETACTIVE:
    begin

      pszText := Format(
        LoadStrInst(hInstance, RC_STRING_VPNINFO),
        [
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_ENTRY)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_COMBO_SERVER)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_USER))
        ]
      );

      SendMessage(GetDlgItem(hApp[2], IDC_STATIC_VPNINFO), WM_SETTEXT, 0,
        Integer(@pszText[1]));

      SendMessage(GetParent(hApp[2]), PSM_SETWIZBUTTONS, 0,
        Integer(PSWIZB_BACK or PSWIZB_FINISH));

    end;

    //

    PSN_QUERYCANCEL:
    begin

      dwRes := ExtMessageBox(
        GetParent(hApp[2]),
        MAKEINTRESOURCEW(LoadStrInst(hInstance, RC_STRING_QCANCEL)),
        MAKEINTRESOURCEW(exeInfo.pszProductName),
        MB_YESNO or MB_ICONASTERISK
      );

      SetWindowLong(hApp[2], DWL_MSGRESULT, Integer(dwRes = IDNO));

    end;

    //
```

```
    PSN_WIZFINISH:
    begin

      pszText := Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_ENTRY));

      dwRes := CreateRasVpnConnection(
        pszText,
        Edit_GetText(GetDlgItem(hApp[1], IDC_COMBO_SERVER)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_USER)),
        Edit_GetText(GetDlgItem(hApp[1], IDC_STATIC_PASSW))
      );

      if (dwRes = ERROR_SUCCESS) then
      begin
        dwRes := SendMessage(GetDlgItem(hApp[2], IDC_CHECK_SHORTCUT),
          BM_GETCHECK, 0, 0);
        if (dwRes = BST_CHECKED) then
          CreateShellVpnLink(pszText);
      end;

    end;

  end;

  //

  Result := 1;

end;

//

function FinsDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;
begin

  case uMsg of

    //

    WM_INITDIALOG:
    begin

      Result := BOOL(FinsDlgProc_OnWmInitDialog(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_NOTIFY:
    begin

      Result := BOOL(FinsDlgProc_OnWmNotify(hWnd, uMsg, wParam, lParam));

    end;

  else
    Result := FALSE;
  end;

end;

end.
```

VPN_WinMain.pas - основна програма

```
unit VPN_WinMain;

interface

uses
  Windows, Messages, CommCtrl, VPN_CommCtrl, VPN_SysUtils, VPN_FileInfo,
VPN_Resources,
  VPN_WelcWind, VPN_SettWind, VPN_FinsWind;

function WinMain(hInstance: HINST; hPrevInstance: HINST; lpCmdLine: LPSTR;
nCmdShow: Integer): Integer; stdcall;

implementation

function WinMain(hInstance: HINST; hPrevInstance: HINST; lpCmdLine: LPSTR;
nCmdShow: Integer): Integer; stdcall;
var
  hMutex : THandle;
  pszText: WideString;
  iccex  : TInitCommonControlsEx;
  psh    : TPropSheetHeader;
begin

  // витягаємо інформацію з ресурсу версії й заповнюємо їй підготовлену
  // структуру, що у подальшому будемо використовувати для читання/запису
  // настроювань програми й виводу тексту в заголовку повідомлень.

  ZeroMemory(@exeInfo, SizeOf(TStringFileInfo));
  GetFileInfo(AnsiStringToWide(ParamStr(0), CP_ACP), exeInfo);

  // створюємо Mutex для перевірки запуску копій додатка.

  hMutex := CreateMutex(nil, FALSE, MAKEINTRESOURCEW(exeInfo.pszProductName));
  if (GetLastError = ERROR_ALREADY_EXISTS) then
  begin
    pszText := LoadStrInst(hInstance, RC_STRING_COPYRUN);
    MessageBox(
      GetActiveWindow,
      @pszText[1],
      MAKEINTRESOURCEW(exeInfo.pszProductName),
      MB_OK or MB_ICONEXCLAMATION or MB_SYSTEMMODAL
    );
    Halt;
  end;

  // ініціалізуємо бібліотеку стандартних органів управління.

  iccex.dwSize := SizeOf(TInitCommonControlsEx);
  iccex.dwICC  := ICC_ANIMATE_CLASS or ICC_PROGRESS_CLASS or ICC_TAB_CLASSES or
    ICC_STANDARVPN_CLASSES or ICC_WIN95_CLASSES;
  InitCommonControlsEx(iccex);

  //

  pszText := Format(LoadStrInst(hInstance, RC_STRING_CWINDOW),
    [exeInfo.pszProductName, exeInfo.pszFileVersion]);

  // створюємо й відображаємо сторінки майстра.

  ZeroMemory(@psp, SizeOf(TPropSheetPage));

  psp.dwSize            := SizeOf(TPropSheetPage);
  psp.dwFlags           := PSP_USETITLE or PSP_HIDEHEADER;
  psp.pszTitle          := @pszText[1];
  psp.pfnDlgProc        := @WelcDlgProc;
```

```
  psp.pszTemplate       := MAKEINTRESOURCEW(RC_DIALOG_WELCOME);
  ahpsp[0]              := CreatePropertySheetPage(psp);

  ZeroMemory(@psp, SizeOf(TPropSheetPage));

  psp.dwSize            := SizeOf(TPropSheetPage);
  psp.dwFlags           := PSP_USETITLE or PSP_USEHEADERTITLE or
PSP_USEHEADERSUBTITLE;
  psp.pszTitle          := @pszText[1];
  psp.pszHeaderTitle    := MAKEINTRESOURCEW(LoadStrInst(hInstance,
RC_STRING_THEADER));
  psp.pszHeaderSubTitle := MAKEINTRESOURCEW(LoadStrInst(hInstance,
RC_STRING_SHEADER));
  psp.pszTemplate       := MAKEINTRESOURCEW(RC_DIALOG_SETTINGS);
  psp.pfnDlgProc        := @SettDlgProc;
  ahpsp[1]              := CreatePropertySheetPage(psp);

  ZeroMemory(@psp, SizeOf(TPropSheetPage));

  psp.dwSize            := SizeOf(TPropSheetPage);
  psp.dwFlags           := PSP_USETITLE or PSP_HIDEHEADER;
  psp.pszTitle          := @pszText[1];
  psp.pszTemplate       := MAKEINTRESOURCEW(RC_DIALOG_FINISH);
  psp.pfnDlgProc        := @FinsDlgProc;
  ahpsp[2]              := CreatePropertySheetPage(psp);

  ZeroMemory(@psh, SizeOf(TPropSheetHeader));

  psh.dwSize         := SizeOf(TPropSheetHeader);
  psh.hInstance      := hInstance;
  psh.hwndParent     := 0;
  psh.phpage         := @ahpsp[0];
  psh.nStartPage     := 0;
  psh.nPages         := Length(ahpsp);
  psh.pszbmWatermark := MAKEINTRESOURCEW(RC_BITMAP_WATERMARK);
  psh.pszbmHeader    := MAKEINTRESOURCEW(RC_BITMAP_HEADER);
  psh.dwFlags        := PSH_WIZARD97 or PSH_WATERMARK or PSH_HEADER or
PSH_USEICONID;
  psh.pszIcon        := MAKEINTRESOURCEW(RC_ICONEX_CAPTION);

  PropertySheet(psh);

  // видаляємо іменований об'єкт.

  if (hMutex <> 0) then
  begin
    ReleaseMutex(hMutex);
    CloseHandle(hMutex);
  end;

  //

  Result := 0;

end;

end.
```

VPN_WinSvc.pas - сервіси

```
unit VPN_WinSvc;

interface

uses
  Windows;

type
  LPSERVICE_STATUS = ^SERVICE_STATUS;
  _SERVICE_STATUS = record
    dwServiceType: DWORD;
    dwCurrentState: DWORD;
    dwControlsAccepted: DWORD;
    dwWin32ExitCode: DWORD;
    dwServiceSpecificExitCode: DWORD;
    dwCheckPoint: DWORD;
    dwWaitHint: DWORD;
  end;
  SERVICE_STATUS = _SERVICE_STATUS;
  TServiceStatus = SERVICE_STATUS;
  PServiceStatus = LPSERVICE_STATUS;
  LPSERVICE_STATUS_PROCESS = ^SERVICE_STATUS_PROCESS;
  _SERVICE_STATUS_PROCESS = record
    dwServiceType: DWORD;
    dwCurrentState: DWORD;
    dwControlsAccepted: DWORD;
    dwWin32ExitCode: DWORD;
    dwServiceSpecificExitCode: DWORD;
    dwCheckPoint: DWORD;
    dwWaitHint: DWORD;
    dwProcessId: DWORD;
    dwServiceFlags: DWORD;
  end;
  SERVICE_STATUS_PROCESS = _SERVICE_STATUS_PROCESS;
  TServiceStatusProcess = SERVICE_STATUS_PROCESS;
  PServiceStatusProcess = LPSERVICE_STATUS_PROCESS;


//
// Структура перерахунку статуса послуги
//
  LPENUM_SERVICE_STATUSA = ^ENUM_SERVICE_STATUSA;
  {$EXTERNALSYM LPENUM_SERVICE_STATUSA}
  _ENUM_SERVICE_STATUSA = record
    lpServiceName: LPSTR;
    lpDisplayName: LPSTR;
    ServiceStatus: SERVICE_STATUS;
  end;
  {$EXTERNALSYM _ENUM_SERVICE_STATUSA}
  ENUM_SERVICE_STATUSA = _ENUM_SERVICE_STATUSA;
  {$EXTERNALSYM ENUM_SERVICE_STATUSA}
  TEnumServiceStatus = ENUM_SERVICE_STATUSA;
  PEnumServiceStatus = LPENUM_SERVICE_STATUSA;
  LPENUM_SERVICE_STATUSW = ^ENUM_SERVICE_STATUSW;
  {$EXTERNALSYM LPENUM_SERVICE_STATUSW}
  _ENUM_SERVICE_STATUSW = record
    lpServiceName: LPWSTR;
    lpDisplayName: LPWSTR;
    ServiceStatus: SERVICE_STATUS;
  end;
  {$EXTERNALSYM _ENUM_SERVICE_STATUSW}
  ENUM_SERVICE_STATUSW = _ENUM_SERVICE_STATUSW;
  {$EXTERNALSYM ENUM_SERVICE_STATUSW}
  TEnumServiceStatus = ENUM_SERVICE_STATUSW;
  PEnumServiceStatus = LPENUM_SERVICE_STATUSW;
```

```
  PEnumServiceStatus = PEnumServiceStatus;

  _SC_STATUS_TYPE = (SC_STATUS_PROCESS_INFO);
  SC_STATUS_TYPE = _SC_STATUS_TYPE;

//
// Типи заголовків
//

  {$EXTERNALSYM SC_HANDLE}
  SC_HANDLE = THandle;
  {$EXTERNALSYM LPSC_HANDLE}
  LPSC_HANDLE = ^SC_HANDLE;

const  //
// Стан сервісу – для перерахуємих послуг (бітова маска)
//
  {$EXTERNALSYM SERVICE_ACTIVE}
  SERVICE_ACTIVE                = $00000001;
  {$EXTERNALSYM SERVICE_INACTIVE}
  SERVICE_INACTIVE              = $00000002;
  {$EXTERNALSYM SERVICE_STATE_ALL}
  SERVICE_STATE_ALL             = (SERVICE_ACTIVE   or
                                   SERVICE_INACTIVE);


//
// Менеджер сервісу управління об’єкту типу специфічного доступа
//
  {$EXTERNALSYM SC_MANAGER_CONNECT}
  SC_MANAGER_CONNECT            = $0001;
  {$EXTERNALSYM SC_MANAGER_CREATE_SERVICE}
  SC_MANAGER_CREATE_SERVICE     = $0002;
  {$EXTERNALSYM SC_MANAGER_ENUMERATE_SERVICE}
  SC_MANAGER_ENUMERATE_SERVICE  = $0004;
  {$EXTERNALSYM SC_MANAGER_LOCK}
  SC_MANAGER_LOCK               = $0008;
  {$EXTERNALSYM SC_MANAGER_QUERY_LOCK_STATUS}
  SC_MANAGER_QUERY_LOCK_STATUS  = $0010;
  {$EXTERNALSYM SC_MANAGER_MODIFY_BOOT_CONFIG}
  SC_MANAGER_MODIFY_BOOT_CONFIG = $0020;

  {$EXTERNALSYM SC_MANAGER_ALL_ACCESS}
  SC_MANAGER_ALL_ACCESS         = (STANDARVPN_RIGHTS_REQUIRED or
                                   SC_MANAGER_CONNECT or
                                   SC_MANAGER_CREATE_SERVICE or
                                   SC_MANAGER_ENUMERATE_SERVICE or
                                   SC_MANAGER_LOCK or
                                   SC_MANAGER_QUERY_LOCK_STATUS or
                                   SC_MANAGER_MODIFY_BOOT_CONFIG);

//
// Сервіс  об’єкту типу специфічного доступа
//
  SERVICE_STOP                  = $0020;
  SERVICE_QUERY_STATUS          = $0004;
  SERVICE_ENUMERATE_DEPENDENTS  = $0008;
//
// Стан сервісу – для поточного стану
//
  {$EXTERNALSYM SERVICE_STOPPED}
  SERVICE_STOPPED               = $00000001;
  {$EXTERNALSYM SERVICE_START_PENDING}
  SERVICE_START_PENDING         = $00000002;
  {$EXTERNALSYM SERVICE_STOP_PENDING}
  SERVICE_STOP_PENDING          = $00000003;
  {$EXTERNALSYM SERVICE_RUNNING}
  SERVICE_RUNNING               = $00000004;
  {$EXTERNALSYM SERVICE_CONTINUE_PENDING}
```

```
  SERVICE_CONTINUE_PENDING       = $00000005;
  {$EXTERNALSYM SERVICE_PAUSE_PENDING}
  SERVICE_PAUSE_PENDING          = $00000006;
  {$EXTERNALSYM SERVICE_PAUSED}
  SERVICE_PAUSED                 = $00000007;
//
// Управління
//
  {$EXTERNALSYM SERVICE_CONTROL_STOP}
  SERVICE_CONTROL_STOP           = $00000001;
  {$EXTERNALSYM SERVICE_CONTROL_PAUSE}
  SERVICE_CONTROL_PAUSE          = $00000002;
  {$EXTERNALSYM SERVICE_CONTROL_CONTINUE}
  SERVICE_CONTROL_CONTINUE       = $00000003;
  {$EXTERNALSYM SERVICE_CONTROL_INTERROGATE}
  SERVICE_CONTROL_INTERROGATE    = $00000004;
  {$EXTERNALSYM SERVICE_CONTROL_SHUTDOWN}
  SERVICE_CONTROL_SHUTDOWN       = $00000005;

function OpenSCManager(lpMachineName: LPCWSTR; lpDatabaseName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenSCManager(lpMachineName: LPCWSTR; lpDatabaseName: LPCWSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenSCManager(lpMachineName: LPCWSTR; lpDatabaseName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;

function OpenService(hSCManager: SC_HANDLE; lpServiceName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenService(hSCManager: SC_HANDLE; lpServiceName: LPCWSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;
function OpenService(hSCManager: SC_HANDLE; lpServiceName: LPCSTR;
dwDesiredAccess: DWORD): SC_HANDLE; stdcall;

function CloseServiceHandle(hSCObject: SC_HANDLE): BOOL; stdcall;

function QueryServiceStatusEx(hService: SC_HANDLE; InfoLevel: SC_STATUS_TYPE;
lpBuffer: PByte; cbBufSize: DWORD; var pcbBytesNeeded: DWORD): BOOL; stdcall;

function ControlService(hService: SC_HANDLE; dwControl: DWORD; var
lpServiceStatus: TServiceStatusProcess): BOOL; stdcall;

function EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD;
lpServices: LPENUM_SERVICE_STATUSA; cbBufSize: DWORD; var pcbBytesNeeded,
lpServicesReturned: DWORD): BOOL; stdcall;
function EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD;
lpServices: LPENUM_SERVICE_STATUSW; cbBufSize: DWORD; var pcbBytesNeeded,
lpServicesReturned: DWORD): BOOL; stdcall;
function EnumDependentServices(hService: SC_HANDLE; dwServiceState: DWORD;
lpServices: LPENUM_SERVICE_STATUSA; cbBufSize: DWORD; var pcbBytesNeeded,
lpServicesReturned: DWORD): BOOL; stdcall;

implementation
function OpenSCManager; external advapi32 name 'OpenSCManager';
function OpenSCManager; external advapi32 name 'OpenSCManager';
function OpenSCManager; external advapi32 name 'OpenSCManager';
function OpenService; external advapi32 name 'OpenService';
function OpenService; external advapi32 name 'OpenService';
function OpenService; external advapi32 name 'OpenService';
function CloseServiceHandle; external advapi32 name 'CloseServiceHandle';
function QueryServiceStatusEx; external advapi32 name 'QueryServiceStatusEx';
function ControlService; external advapi32 name 'ControlService';
function EnumDependentServices; external advapi32 name 'EnumDependentServices';
function EnumDependentServices; external advapi32 name 'EnumDependentServices';
function EnumDependentServices; external advapi32 name 'EnumDependentServices';
end.
```

VPN_SysUtils.pas - утиліти

```
unit VPN_SysUtils;

interface

uses
  Windows, Messages;

function LoadStrInst(hInst: HMODULE; I: Integer): WideString;
function Format(szString: WideString; const Params: Array of const): WideString;
function WideStringToAnsi(pszText: WideString; CodePage: WORD): AnsiString;
function AnsiStringToWide(pszText: AnsiString; CodePage: WORD): WideString;
function ExtractFilePath(pszText: WideString): WideString;
function ExcludeTrailingPathDelimiter(szString: WideString): WideString;
function SetCenterDialogPos(hDialog, hParent: HWND; IsParent: Boolean): Boolean;
function GetWindowFontSize(hWnd: HWND; pSize: Integer): Integer;
function GetWindowBoldFont(hWnd: THandle; fntHeight: Integer): HFONT;

implementation

//

function LoadStrInst(hInst: HMODULE; I: Integer): WideString;
var
  lpBuffer: Array [0..MAX_ PATH-1] of WideChar;
begin
  LoadString(hInst, I, lpBuffer, Length(lpBuffer));
  Result := lpBuffer;
end;

//

function Format(szString: WideString; const Params: Array of const): WideString;
var
  lpChar: Array [0..1023] of WideChar;
  lpWord: Array [0..15] of LongWord;
  nIndex: Integer;
begin
  for nIndex := High(Params) downto 0 do
    lpWord[nIndex] := Params[nIndex].VInteger;
  wvsprintf(@lpChar, @szString[1], @lpWord);
  Result := lpChar;
end;

//

function WideStringToAnsi(pszText: WideString; CodePage: WORD): AnsiString;
var
  dwBytes: Integer;
  dwFlags: DWORD;
begin
  if (pszText <> '') then
    begin
      dwFlags := WC_COMPOSITECHECK or WC_DISCARDNS or WC_SEPCHARS or
WC_DEFAULTCHAR;
      dwBytes := WideCharToMultiByte(CodePage, dwFlags, @pszText[1], -1, nil, 0,
        nil, nil);
      SetLength(Result, dwBytes - 1);
      if (dwBytes > 1) then
        WideCharToMultiByte(CodePage, dwFlags, @pszText[1], -1, @Result[1],
          dwBytes - 1, nil, nil);
    end
  else
    Result := '';
end;

//
```

```
function AnsiStringToWide(pszText: AnsiString; CodePage: WORD): WideString;
var
  dwBytes: Integer;
begin
  if (pszText <> '') then
    begin
      dwBytes := MultiByteToWideChar(CodePage, MB_PRECOMPOSED, @pszText[1], -1,
        nil, 0);
      SetLength(Result, dwBytes - 1);
      if (dwBytes > 1) then
        MultiByteToWideChar(CodePage, MB_PRECOMPOSED, @pszText[1], -1,
@Result[1],
          dwBytes - 1);
    end
  else
    Result := '';
end;

//

function ExtractFilePath(pszText: WideString): WideString;
var
  L: Integer;
begin
  Result := '';
  L := Length(pszText);
  while (L > 0) do
    begin
      if (pszText[L] = ':') or (pszText[L] = '\') then
        begin
          Result := Copy(pszText, 1, L);
          Break;
        end;
      Dec(L);
    end;
end;

//

function ExcludeTrailingPathDelimiter(szString: WideString): WideString;
var
  I: Integer;
begin
  Result := szString;
  I := Length(Result);
  while (I > 0) and (Result[I] = '\') do
    Dec(I);
  SetLength(Result, I);
end;

//

function SetCenterDialogPos(hDialog, hParent: HWND; IsParent: Boolean): Boolean;
var
  wRect  : TRect;
  pRect  : TRect;
  wArea  : TRect;
  xLeft  : Integer;
  yTop   : Integer;
  iWidth : Integer;
  iHeight: Integer;
  dwFlags: DWORD;
begin
  case IsParent of
    FALSE:
      begin
        GetWindowRect(hDialog, wRect);
        iWidth  := wRect.Right - wRect.Left;
```

```
        iHeight := wRect.Bottom - wRect.Top;
        xLeft := (GetSystemMetrics(SM_CXSCREEN) - iWidth) div 2;
        yTop  := (GetSystemMetrics(SM_CYSCREEN) - iHeight) div 2;
      end;
    TRUE:
      begin
        GetWindowRect(hDialog, wRect);
        GetWindowRect(hParent, pRect);
        iWidth  := wRect.Right - wRect.Left;
        iHeight := wRect.Bottom - wRect.Top;
        SystemParametersInfo(SPI_GETWORKAREA, 0, @wArea, 0);
        xLeft := pRect.Left + ((pRect.Right - pRect.Left - iWidth) div 2);
        if (xLeft < 0) then
          xLeft := 0
        else
        if ((xLeft + iWidth) > (wArea.Right - wArea.Left)) then
          xLeft := wArea.Right - wArea.Left - iWidth;
        yTop := pRect.Top + ((pRect.Bottom - pRect.Top - iHeight) div 2);
        if (yTop < 0) then
          yTop := 0
        else
        if ((yTop + iHeight) > (wArea.Bottom - wArea.Top)) then
          yTop := wArea.Bottom - wArea.Top - iHeight;
      end;
  end;
  dwFlags := SWP_NOACTIVATE or SWP_NOSIZE or SWP_NOZORDER;
  Result := SetWindowPos(hDialog, 0, xLeft, yTop, 0, 0, dwFlags);
end;

//

function GetWindowFontSize(hWnd: HWND; pSize: Integer): Integer;
var
  dc: HDC;
begin
  dc := GetDC(hWnd);
  Result := -MulDiv(pSize, GetDeviceCaps(dc, LOGPIXELSY), 72);
  ReleaseDC(hWnd, dc);
end;

//

function GetWindowBoldFont(hWnd: THandle; fntHeight: Integer): HFONT;
var
  lf   : TLogFont;
  dwRes: Integer;
  hfnt : HFONT;
begin
  hfnt := HFONT(SendMessage(hWnd, WM_GETFONT, 0, 0));
  ZeroMemory(@lf, SizeOf(TLogFont));
  if (hfnt <> 0) then
    dwRes := GetObject(hfnt, SizeOf(TLogFont), @lf);
  if (dwRes <> 0) then
  begin
    lf.lfHeight := fntHeight;
    lf.lfWeight := FW_BOLD;
    hfnt := CreateFontIndirect(lf);
  end;
  Result := hfnt;
end;

end.
```

VPN_StatAnim.pas – анімація при підключенні VPN

```
unit VPN_StatAnim;

interface

uses
  Windows, Messages, CommCtrl, VPN_Windows;

procedure CreateAnimateStatic(hWnd: HWND);
procedure RemoveAnimateStatic(hWnd: HWND);

const
  SS_SETIMAGELIST   = WM_USER + 101;
  SS_SETELAPSEDTIME = WM_USER + 102;
  SS_GETIMAGELIST   = WM_USER + 111;
  SS_GETELAPSEDTIME = WM_USER + 112;

implementation

const
  IDC_ANIMATETIMER = 101;

type
  TStatWndProc = function(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT; stdcall;

  P_STAT_PRO = ^T_STAT_PRO;
  T_STAT_PRO = packed record
    StatProc  : TStatWndProc;
    rcClient  : TRect;
    //
    hdcMem    : HDC;
    hbmMem    : HBITMAP;
    hbmOld    : HBITMAP;
    //
    himl      : HIMAGELIST;
    //
    imgSize   : Integer;
    imgCount  : Integer;
    imgCurrent: Integer;
    //
    dwElapse  : Integer;
  end;

var
  psp: P_STAT_PRO;

//

function StatWndProc_OnWmSize(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
var
  hdcIn: HDC;
begin

  //

  GetClientRect(hWnd, psp.rcClient);

  //

  if (psp.hdcMem <> 0) then
    begin
      SelectObject(psp.hdcMem, psp.hbmOld);
      DeleteObject(psp.hbmMem);
      DeleteDC(psp.hdcMem);
    end;
```

```
  //

  hdcIn := GetDC(hWnd);
  psp.hdcMem := CreateCompatibleDC(hdcIn);
  psp.hbmMem := CreateCompatibleBitmap(
    hdcIn,
    psp.rcClient.Right - psp.rcClient.Left,
    psp.rcClient.Bottom - psp.rcClient.Top
  );
  psp.hbmOld := SelectObject(psp.hdcMem, psp.hbmMem);
  ReleaseDC(hWnd, hdcIn);

  //

  Result := CallWindowProc(@psp.StatProc, hWnd, uMsg, wParam, lParam);

  //

  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);

end;

//

function StatWndProc_OnWmPaint(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT; stdcall;

var
  hdcIn: HDC;
  ps   : TPaintStruct;
begin

  //

  if (wParam = 0) then
    hdcIn := BeginPaint(hWnd, ps)
  else
    hdcIn := wParam;

  //

  CallWindowProc(@psp.StatProc, hWnd, WM_PRINTCLIENT, psp.hdcMem, PRVPN_CLIENT);

  {
  CallWindowProc(@psp.StatProc, hWnd, WM_ERASEBKGND, psp.hdcMem, 0);
  }

  if (psp.himl <> 0) then
    ImageList_DrawEx(
      psp.himl,
      psp.imgCurrent - 1,
      psp.hdcMem,
      psp.rcClient.Left + ((psp.rcClient.Right - psp.rcClient.Left) div 2) -
(psp.imgSize div 2),
      psp.rcClient.Top + ((psp.rcClient.Bottom - psp.rcClient.Top) div 2) -
(psp.imgSize div 2),
      psp.imgSize,
      psp.imgSize,
      CLR_DEFAULT,
      CLR_DEFAULT,
      ILVPN_NORMAL or ILVPN_TRANSPARENT
    );

  BitBlt(
    hdcIn,
    0,
    0,
    psp.rcClient.Right - psp.rcClient.Left,
```

```
    psp.rcClient.Bottom - psp.rcClient.Top,
    psp.hdcMem,
    0,
    0,
    SRCCOPY
  );

  //

  if (wParam = 0) then
    EndPaint(hWnd, ps);

  //

  Result := 0;

end;

//

function StatWndProc_OnWmTimer(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Inc(psp.imgCurrent);
  if (psp.imgCurrent > psp.imgCount) then
    psp.imgCurrent := 1;

  //

  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);

  //

  Result := 0;

end;

//

function StatWndProc_OnWmEraseBkgnd(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Result := 1;

end;

//

function StatWndProc_OnSetImageList(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  KillTimer(hWnd, IDC_ANIMATETIMER);

  //

  psp.himl := HIMAGELIST(wParam);

  //
```

```
  if (psp.himl <> 0) then
    begin

      ImageList_GetIconSize(psp.himl, psp.imgSize, psp.imgSize);
      psp.imgCount := ImageList_GetImageCount(psp.himl);
      SetTimer(hWnd, IDC_ANIMATETIMER, psp.dwElapse, nil);

    end;

  //

  Result := 0;

end;

//

function StatWndProc_OnSetElapsedTime(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  KillTimer(hWnd, IDC_ANIMATETIMER);

  //

  psp.dwElapse := wParam;

  //

  SetTimer(hWnd, IDC_ANIMATETIMER, psp.dwElapse, nil);


  //

  Result := 0;

end;

//

function StatWndProc_OnGetImageList(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Result := LRESULT(psp.himl);

end;

//

function StatWndProc_OnGetElapsedTime(psp: P_STAT_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin

  //

  Result := LRESULT(psp.dwElapse);

end;

//

function StatWndProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
LRESULT; stdcall;
begin
```

```
psp := P_STAT_PRO(GetWindowLong(hWnd, GWL_USERDATA));

if (psp = nil) then
  begin
    Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
    Exit;
  end;

case uMsg of

  //

  WM_DESTROY:
    begin
      RemoveAnimateStatic(hWnd);
    end;

  //

  WM_SIZE:
    begin
      Result := StatWndProc_OnWmSize(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  WM_PRINTCLIENT,
  WM_PAINT,
  WM_UPDATEUISTATE: // перемальовування вікна без виклику WM_PAINT.
    begin
      Result := StatWndProc_OnWmPaint(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  WM_TIMER:
    begin
      Result := StatWndProc_OnWmTimer(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  WM_ERASEBKGND:
    begin
      Result := StatWndProc_OnWmEraseBkgnd(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  SS_SETIMAGELIST:
    begin
      Result := StatWndProc_OnSetImageList(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  SS_SETELAPSEDTIME:
    begin
      Result := StatWndProc_OnSetElapsedTime(psp, hWnd, uMsg, wParam, lParam);
    end;

  //

  SS_GETIMAGELIST:
    begin
      Result := StatWndProc_OnGetImageList(psp, hWnd, uMsg, wParam, lParam);
    end;
```

```
    //

    SS_GETELAPSEDTIME:
      begin
        Result := StatWndProc_OnGetElapsedTime(psp, hWnd, uMsg, wParam, lParam);
      end;

    else
      Result := CallWindowProc(@psp.StatProc, hWnd, uMsg, wParam, lParam);
  end;

end;

//

procedure CreateAnimateStatic(hWnd: HWND);
begin

  RemoveAnimateStatic(hWnd);

  psp := P_STAT_PRO(HeapAlloc(GetProcessHeap, HEAP_ZERO_MEMORY,
SizeOf(T_STAT_PRO)));
  ZeroMemory(psp, SizeOf(T_STAT_PRO));

  psp.StatProc   := TStatWndProc(Pointer(GetWindowLong(hWnd, GWL_WNDPROC)));
  psp.himl       := 0;
  psp.imgSize    := 0;
  psp.imgCount   := 0;
  psp.imgCurrent := 0;
  psp.dwElapse   := 50;

  KillTimer(hWnd, IDC_ANIMATETIMER);

  SetWindowLong(hWnd, GWL_USERDATA, Longint(psp));

  SetWindowLong(hWnd, GWL_WNDPROC, Longint(@StatWndProc));

  SendMessage(hWnd, WM_SIZE, 0, 0);

end;

//

procedure RemoveAnimateStatic(hWnd: HWND);
begin

  psp := P_STAT_PRO(GetWindowLong(hWnd, GWL_USERDATA));
  if (psp <> nil) then
    begin

      //

      if (psp.hdcMem <> 0) then
        begin
          SelectObject(psp.hdcMem, psp.hbmOld);
          DeleteObject(psp.hbmMem);
          DeleteDC(psp.hdcMem);
        end;

      //
      KillTimer(hWnd, IDC_ANIMATETIMER);
      //
      SetWindowLong(hWnd, GWL_WNDPROC, Longint(@psp.StatProc));
      RedrawWindow(hWnd, @psp.rcClient, 0, RDW_INVALIDATE or RDW_ERASE);
      SetWindowLong(hWnd, GWL_USERDATA, 0);
      HeapFree(GetProcessHeap, 0, psp);
    end;
end;
end.
```

VPN_ShlObj.pas - інтерпретатор команд операційної системи

```
unit VPN_ShlObj;

interface

uses
  Windows, VPN_Active;

{ Ідентифікація об’єкту у просторі імен(ItemID and IDList) }

const
  // Мережні та Dial-up підключення
  CSIDL_CONNECTIONS = $0031;

{ SHGetSpecialFolderLocation constants }

const
  // Десктоп
  CSIDL_DESKTOP = $0000;

{ Інтерфейс IDs }

const
  IIVPN_IShellFolder: TGUID = (D1: $000214E6; D2: $0000; D3: $0000; D4: ($C0,
$00, $00, $00, $00, $00, $00, $46));
  CLSIVPN_ShellLink : TGUID = (D1: $00021401; D2: $0000; D3: $0000; D4: ($C0,
$00, $00, $00, $00, $00, $00, $46));

{ IShellFolder.GetDisplayNameOf/SetNameOf uFlags }

const
  // по замовчуванню (виключно для дисплею)
  SHGDN_NORMAL       = 0;
  // для перегляду по замовчуванню
  SHCONTVPN_NONFOLDERS = 64;

{ Рядкові константи для інтерфейсу IDs }

const
  SIVPN_IShellLink  = '{000214 0000-0000-C 000-000000000046}';
  SIVPN_IShellLink  = '{000214F 0000-0000-C 000-000000000046}';
  SIVPN_IShellFolder = '{000214E 0000-0000-C 000-000000000046}';
  SIVPN_IEnumIDList  = '{000214F 0000-0000-C 000-000000000046}';

{ IShellLink інтерфейс }

type
  IShellLink = interface(IUnknown)
    [SIVPN_IShellLink]
    function GetPath(pszFile: PAnsiChar; cchMaxPath: Integer; var pfd:
TWin32FindData; fFlags: DWORD): HResult; stdcall;
    function GetIDList(var ppidl: PItemIDList): HResult; stdcall;
    function SetIDList(pidl: PItemIDList): HResult; stdcall;
    function GetDescription(pszName: PAnsiChar; cchMaxName: Integer): HResult;
stdcall;
    function SetDescription(pszName: PAnsiChar): HResult; stdcall;
    function GetWorkingDirectory(pszDir: PAnsiChar; cchMaxPath: Integer):
HResult; stdcall;
    function SetWorkingDirectory(pszDir: PAnsiChar): HResult; stdcall;
    function GetArguments(pszArgs: PAnsiChar; cchMaxPath: Integer): HResult;
stdcall;
    function SetArguments(pszArgs: PAnsiChar): HResult; stdcall;
    function GetHotkey(var pwHotkey: Word): HResult; stdcall;
    function SetHotkey(wHotkey: Word): HResult; stdcall;
    function GetShowCmd(out piShowCmd: Integer): HResult; stdcall;
    function SetShowCmd(iShowCmd: Integer): HResult; stdcall;
```

```
    function GetIconLocation(pszIconPath: PAnsiChar; cchIconPath: Integer; out
piIcon: Integer): HResult; stdcall;
    function SetIconLocation(pszIconPath: PAnsiChar; iIcon: Integer): HResult;
stdcall;
    function SetRelativePath(pszPathRel: PAnsiChar; dwReserved: DWORD): HResult;
stdcall;
    function Resolve(Wnd: HWND; fFlags: DWORD): HResult; stdcall;
    function SetPath(pszFile: PAnsiChar): HResult; stdcall;
  end;

  IShellLink = interface(IUnknown)
    [SIVPN_IShellLink]
    function GetPath(pszFile: PWideChar; cchMaxPath: Integer; var pfd:
TWin32FindData; fFlags: DWORD): HResult; stdcall;
    function GetIDList(var ppidl: PItemIDList): HResult; stdcall;
    function SetIDList(pidl: PItemIDList): HResult; stdcall;
    function GetDescription(pszName: PWideChar; cchMaxName: Integer): HResult;
stdcall;
    function SetDescription(pszName: PWideChar): HResult; stdcall;
    function GetWorkingDirectory(pszDir: PWideChar; cchMaxPath: Integer):
HResult; stdcall;
    function SetWorkingDirectory(pszDir: PWideChar): HResult; stdcall;
    function GetArguments(pszArgs: PWideChar; cchMaxPath: Integer): HResult;
stdcall;
    function SetArguments(pszArgs: PWideChar): HResult; stdcall;
    function GetHotkey(var pwHotkey: Word): HResult; stdcall;
    function SetHotkey(wHotkey: Word): HResult; stdcall;
    function GetShowCmd(out piShowCmd: Integer): HResult; stdcall;
    function SetShowCmd(iShowCmd: Integer): HResult; stdcall;
    function GetIconLocation(pszIconPath: PWideChar; cchIconPath: Integer; out
piIcon: Integer): HResult; stdcall;
    function SetIconLocation(pszIconPath: PWideChar; iIcon: Integer): HResult;
stdcall;
    function SetRelativePath(pszPathRel: PWideChar; dwReserved: DWORD): HResult;
stdcall;
    function Resolve(Wnd: HWND; fFlags: DWORD): HResult; stdcall;
    function SetPath(pszFile: PWideChar): HResult; stdcall;
  end;
  IShellLink = IShellLink;

{ IEnumIDList інтерфейс }

type
  IEnumIDList = interface(IUnknown)
    [SIVPN_IEnumIDList]
    function Next(celt: ULONG; out rgelt: PItemIDList; var pceltFetched: ULONG):
HResult; stdcall;
    function Skip(celt: ULONG): HResult; stdcall;
    function Reset: HResult; stdcall;
    function Clone(out ppenum: IEnumIDList): HResult; stdcall;
  end;

{ record for returning strings from IShellFolder member functions }

type
  PSTRRet = ^TStrRet;
  _STRRET = record
     uType: UINT;                                { одне з значень STRRET_* }
     case Integer of
       0: (pOleStr: LPWSTR);                     { повинно бути вільно для
виклику GetDisplayNameOf }
       1: (pStr: LPSTR);                         { НЕ ВИКОРИСТОВУЄТЬСЯ }
       2: (uOffset: UINT);                       { Зсув у SHITEMID (ANSI) }
       3: (cStr: array[0..MAX_ PATH-1] of Char); { Буфер для заповнювання}
    end;
  TStrRet = _STRRET;
  STRRET = _STRRET;

{ structure STRRET for returning strings from IShellFolder member functions }
```

```
const
  STRRET_WSTR = $0000;
  STRRET_CSTR = $0002;

{ IShellFolder інтерфейс }

type
  IShellFolder = interface(IUnknown)
    [SIVPN_IShellFolder]
    function ParseDisplayName(hwndOwner: HWND; pbcReserved: Pointer;
lpszDisplayName: POLESTR; out pchEaten: ULONG; out ppidl: PItemIDList; var
dwAttributes: ULONG): HResult; stdcall;
    function EnumObjects(hwndOwner: HWND; grfFlags: DWORD; out EnumIDList:
IEnumIDList): HResult; stdcall;
    function BindToObject(pidl: PItemIDList; pbcReserved: Pointer; const riid:
TIID; out ppvOut): HResult; stdcall;
    function BindToStorage(pidl: PItemIDList; pbcReserved: Pointer; const riid:
TIID; out ppvObj): HResult; stdcall;
    function CompareIDs(lParam: LPARAM; pidl1, pidl2: PItemIDList): HResult;
stdcall;
    function CreateViewObject(hwndOwner: HWND; const riid: TIID; out ppvOut):
HResult; stdcall;
    function GetAttributesOf(cidl: UINT; var apidl: PItemIDList; var rgfInOut:
UINT): HResult; stdcall;
    function GetUIObjectOf(hwndOwner: HWND; cidl: UINT; var apidl: PItemIDList;
const riid: TIID; prgfInOut: Pointer; out ppvOut): HResult; stdcall;
    function GetDisplayNameOf(pidl: PItemIDList; uFlags: DWORD; var lpName:
STRRET): HResult; stdcall;
    function SetNameOf(hwndOwner: HWND; pidl: PItemIDList; lpszName: POLEStr;
uFlags: DWORD; var ppidlOut: PItemIDList): HResult; stdcall;
  end;

function SHGetMalloc(var ppMalloc: IMalloc): HResult; stdcall;
function SHGetFolderLocation(hWndOwner: HWND; csidl: Integer; hToken: THandle;
dwReserved: DWORD; var pidl: PItemIDList): HResult; stdcall;
function SHGetDesktopFolder(var ppshf: IShellFolder): HResult; stdcall;
function SHGetSpecialFolderLocation(hwndOwner: HWND; nFolder: Integer; var
ppidl: PItemIDList): HResult; stdcall;
function SHGetPathFromIDList(pidl: PItemIDList; pszPath: PChar): BOOL; stdcall;
function SHGetPathFromIDList(pidl: PItemIDList; pszPath: PAnsiChar): BOOL;
stdcall;
function SHGetPathFromIDList(pidl: PItemIDList; pszPath: PWideChar): BOOL;
stdcall;

implementation

const
  shell32 = 'shell32.dll';

function SHGetMalloc;                external shell32 name 'SHGetMalloc';
function SHGetFolderLocation;        external shell32 name
'SHGetFolderLocation';
function SHGetDesktopFolder;         external shell32 name 'SHGetDesktopFolder';
function SHGetSpecialFolderLocation; external shell32 name
'SHGetSpecialFolderLocation';
function SHGetPathFromIDList;        external shell32 name
'SHGetPathFromIDList';
function SHGetPathFromIDList;       external shell32 name 'SHGetPathFromIDList';
function SHGetPathFromIDList;       external shell32 name 'SHGetPathFromIDList';

end.
```

VPN_Resources.pas – ресурси VPN

```
unit VPN_Resources;

interface

uses
  Windows, CommCtrl, VPN_FileInfo;

const

  { ресурси id діалогу}

  RC_DIALOG_WELCOME   = 101;
  RC_DIALOG_SETTINGS  = 102;
  RC_DIALOG_FINISH    = 103;
  RC_DIALOG_UPDATE    = 104;

  { ресурси id вікна}

  RC_ICONEX_CAPTION   = 101;

  { ресурси id бітової площини}

  RC_BITMAP_WATERMARK = 101;
  RC_BITMAP_HEADER    = 102;
  RC_BITMAP_WAITING   = 103;

  { елементи управління діалога#101 }

  IDC_STATIC_WELCOME  = 10101;

  { елементи управління діалога#102 }

  IDC_STATIC_ENTRY    = 10201;
  IDC_STATIC_SERVER   = 10202;
  IDC_COMBO_SERVER    = 10203;
  IDC_STATIC_USER     = 10204;
  IDC_STATIC_PASSW    = 10205;
  IDC_STATIC_WARN     = 10206;

  { елементи управління діалога#103 }

  IDC_STATIC_FINISH   = 10301;
  IDC_STATIC_VPNINFO  = 10302;
  IDC_CHECK_SHORTCUT  = 10303;

  { елементи управління діалога#104 }

  IDC_STATIC_ANIMATE  = 10401;
  IDC_STATIC_ADDRESS  = 10402;

  { Ресурси рядків таблиць }

  RC_STRING_CWINDOW   = 1600;
  RC_STRING_COPYRUN   = 1601;
  RC_STRING_QCANCEL   = 1602;
  RC_STRING_RESMAN    = 1603;

  RC_STRING_THEADER   = 1616;
  RC_STRING_SHEADER   = 1617;

  RC_STRING_VPNINFO   = 1632;

  RC_STRING_IPSERVER  = 1648;
  RC_STRING_IPHOST    = 1649;
```

```
var
  psp    : TPropSheetPage;
  ahpsp  : Array [0..2] of HPropSheetPage;
  hApp   : Array [0..3] of HWND;
  exeInfo: TStringFileInfo;
  pszServ: WideString = 'server.kbpz.kntu.kr.ua';
  hThread: DWORD;

implementation

end.
```

VPN_RasApi.pas - з'єднання VPN

```
  unit VPN_RasApi;

interface

uses
  Windows;

// RASIPADDR структура

type
  PRASIPADDR = ^RASIPADDR;
  RASIPADDR = record
    a: Byte;
    b: Byte;
    c: Byte;
    d: Byte;
  end;

const
  RAS_MaxAreaCode       = 10;
  RAS_MaxPhoneNumber    = 128;
  RAS_MaxDeviceType     = 16;
  RAS_MaxDeviceName     = 128;
  RAS_MaxPadType        = 32;
  RAS_Max25Address     = 200;
  RAS_MaxFacilities     = 200;
  RAS_MaxUserData       = 200;
  RAS_MaxDnsSuffix      = 255;
  RAS_MaxEntryName      = 256;
  RAS_MaxCallbackNumber = RAS_MaxPhoneNumber;
  UNLEN                 = 256; // Максимальна довжина імені користувача
  PWLEN                 = 256; // Максимальна довжина паролю
  CNLEN                 = 15; // Максимальна довжина імені комп'ютера
  DNLEN                 = CNLEN; // Максимальна довжина імені домену

// структура RASCREDENTIALS

type
  RASCREDENTIALSA = record
    dwSize    : DWORD;
    dwMask    : DWORD;
    szUserName: Array [0..UNLEN] of AnsiChar;
    szPassword: Array [0..PWLEN] of AnsiChar;
    szDomain  : Array [0..DNLEN] of AnsiChar;
  end;

  RASCREDENTIALSW = record
    dwSize    : DWORD;
    dwMask    : DWORD;
    szUserName: Array [0..UNLEN] of WideChar;
    szPassword: Array [0..PWLEN] of WideChar;
    szDomain  : Array [0..DNLEN] of WideChar;
  end;

  LPRASCREDENTIALSW = ^RASCREDENTIALSW;
  LPRASCREDENTIALSA = ^RASCREDENTIALSA;
  LPRASCREDENTIALS = ^RASCREDENTIALS;
  RASCREDENTIALS = RASCREDENTIALSA;

const
  // значення RASCREDENTIALS dwMask

  RASCM_UserName = $00000001;
  RASCM_Password = $00000002;
```

```
// структура RASDIALPARAMS

type
  tagRASDIALPARAMSA = record
    dwSize          : DWORD;
    szEntryName     : Array [0..RAS_MaxEntryName] of AnsiChar;
    szPhoneNumber   : Array [0..RAS_MaxPhoneNumber] of AnsiChar;
    szCallbackNumber: Array [0..RAS_MaxCallbackNumber] of AnsiChar;
    szUserName      : Array [0..UNLEN] of AnsiChar;
    szPassword      : Array [0..PWLEN] of AnsiChar;
    szDomain        : Array [0..DNLEN] of AnsiChar;
    // {$IFDEF WINVER_0x401_OR_GREATER}
    dwSubEntry      : DWORD;
    dwCallbackId    : DWORD;
  end;

  tagRASDIALPARAMSW = record
    dwSize          : DWORD;
    szEntryName     : Array [0..RAS_MaxEntryName] of WideChar;
    szPhoneNumber   : Array [0..RAS_MaxPhoneNumber] of WideChar;
    szCallbackNumber: Array [0..RAS_MaxCallbackNumber] of WideChar;
    szUserName      : Array [0..UNLEN] of WideChar;
    szPassword      : Array [0..PWLEN] of WideChar;
    szDomain        : Array [0..DNLEN] of WideChar;
    // {$IFDEF WINVER_0x401_OR_GREATER}
    dwSubEntry      : DWORD;
    dwCallbackId    : DWORD;
  end;

  PRASDIALPARAMSA = ^RASDIALPARAMSA;
  PRASDIALPARAMSW = ^RASDIALPARAMSW;
  PRASDIALPARAMS = PRASDIALPARAMSA;
  tagRASDIALPARAMS = tagRASDIALPARAMSA;
  RASDIALPARAMSA = tagRASDIALPARAMSA;
  RASDIALPARAMSW = tagRASDIALPARAMSW;
  RASDIALPARAMS = RASDIALPARAMSA;

// структура RASENTRY

type
  tagRASENTRYA = record
    dwSize                  : DWORD;
    dwfOptions              : DWORD;

    // Настроювання мережевого номера
    dwCountryID             : DWORD;
    dwCountryCode           : DWORD;
    szAreaCode              : Array [0..RAS_MaxAreaCode] of AnsiChar;
    szLocalPhoneNumber      : Array [0..RAS_MaxPhoneNumber] of AnsiChar;
    dwAlternateOffset       : DWORD;

    // PPP(Протокол Point-to-point)/Ip
    ipaddr                  : RASIPADDR;
    ipaddrDns               : RASIPADDR;
    ipaddrDnsAlt            : RASIPADDR;
    ipaddrWins              : RASIPADDR;
    ipaddrWinsAlt           : RASIPADDR;

    // Протокол
    dwFrameSize             : DWORD;
    dwfNetProtocols         : DWORD;
    dwFramingProtocol       : DWORD;

    // Сценарії
    szScript                : Array [0..MAX_ PATH-1] of AnsiChar;

    // Автодозвон
    szAutodialDll           : Array [0..MAX_ PATH-1] of AnsiChar;
    szAutodialFunc          : Array [0..MAX_ PATH-1] of AnsiChar;
```

```
    // Пристрій
    szDeviceType              : Array [0..RAS_MaxDeviceType] of AnsiChar;
    szDeviceName              : Array [0..RAS_MaxDeviceName]of AnsiChar;

    // X.25
    sz25PadType              : Array [0..RAS_MaxPadType] of AnsiChar;
    sz25Address              : Array [0..RAS_Max25Address] of AnsiChar;
    sz25Facilities           : Array [0..RAS_MaxFacilities] of AnsiChar;
    sz25UserData             : Array [0..RAS_MaxUserData] of AnsiChar;
    dwChannels                : DWORD;

    // Зарезервовано
    dwReserved1               : DWORD;
    dwReserved2               : DWORD;
    //  {$IFDEF WINVER_0x401_OR_GREATER}

    // Підключення з багатьох з'єднань
    dwSubEntries              : DWORD;
    dwDialMode                : DWORD;
    dwDialExtraPercent        : DWORD;
    dwDialExtraSampleSeconds  : DWORD;
    dwHangUpExtraPercent      : DWORD;
    dwHangUpExtraSampleSeconds: DWORD;

    // Час простою до роз'єднання
    dwIdleDisconnectSeconds   : DWORD;
    // {$IFDEF WINVER_0x500_OR_GREATER}
    dwType                    : DWORD;
    dwEncryptionType          : DWORD;
    dwCustomAuthKey           : DWORD;
    guidId                    : TGUID;
    szCustomDialDll           : Array [0..MAX_ PATH-1] of AnsiChar;
    dwVpnStrategy             : DWORD;
    //  {$IFDEF WINVER_0x501_OR_GREATER}
    dwfOptions2               : DWORD;
    dwfOptions3               : DWORD;
    szDnsSuffix               : Array [0..RAS_MaxDnsSuffix] of AnsiChar;
    dwTcpWindowSize           : DWORD;
    szPrerequisitePbk         : Array [0..MAX_ PATH-1] of AnsiChar;
    szPrerequisiteEntry       : Array [0..RAS_MaxEntryName] of AnsiChar;
    dwRedialCount             : DWORD;
    dwRedialPause             : DWORD;
    //  {$IFDEF WINVER_0x600_OR_GREATER}
    //    ipv6addrDns   : RASIPV6ADDR;
    //    ipv6addrDnsAlt: RASIPV6ADDR;
    //  {$ENDIF}
    //    dwIPv4InterfaceMetric: DWORD;
    //    dwIPv6InterfaceMetric: DWORD;
  end;

  tagRASENTRYW = record
    dwSize                    : DWORD;
    dwfOptions                : DWORD;

    // Настроювання мережного номера
    dwCountryID               : DWORD;
    dwCountryCode             : DWORD;
    szAreaCode                : Array [0..RAS_MaxAreaCode] of WideChar;
    szLocalPhoneNumber        : Array [0..RAS_MaxPhoneNumber] of WideChar;
    dwAlternateOffset         : DWORD;

    // PPP(Протокол Point-to-point)/Ip
    ipaddr                    : RASIPADDR;
    ipaddrDns                 : RASIPADDR;
    ipaddrDnsAlt              : RASIPADDR;
    ipaddrWins                : RASIPADDR;
    ipaddrWinsAlt             : RASIPADDR;
```

```
    // Протокол
    dwFrameSize               : DWORD;
    dwfNetProtocols           : DWORD;
    dwFramingProtocol         : DWORD;

    // Сценарії
    szScript                  : Array [0..MAX_ PATH-1] of WideChar;

    // Автодозвон
    szAutodialDll             : Array [0..MAX_ PATH-1] of WideChar;
    szAutodialFunc            : Array [0..MAX_ PATH-1] of WideChar;

    // Пристрій
    szDeviceType              : Array [0..RAS_MaxDeviceType] of WideChar;
    szDeviceName              : Array [0..RAS_MaxDeviceName]of WideChar;

    // X.25
    sz25PadType              : Array [0..RAS_MaxPadType] of WideChar;
    sz25Address              : Array [0..RAS_Max25Address] of WideChar;
    sz25Facilities           : Array [0..RAS_MaxFacilities] of WideChar;
    sz25UserData             : Array [0..RAS_MaxUserData] of WideChar;
    dwChannels                : DWORD;

    // Зарезервовано
    dwReserved1               : DWORD;
    dwReserved2               : DWORD;
    //  {$IFDEF WINVER_0x401_OR_GREATER}

    // Підключення з багатьох з'єднань
    dwSubEntries              : DWORD;
    dwDialMode                : DWORD;
    dwDialExtraPercent        : DWORD;
    dwDialExtraSampleSeconds  : DWORD;
    dwHangUpExtraPercent      : DWORD;
    dwHangUpExtraSampleSeconds: DWORD;

    // Час простою до роз'єднання
    dwIdleDisconnectSeconds   : DWORD;
    // {$IFDEF WINVER_0x500_OR_GREATER}
    dwType                    : DWORD;
    dwEncryptionType          : DWORD;
    dwCustomAuthKey           : DWORD;
    guidId                    : TGUID;
    szCustomDialDll           : Array [0..MAX_ PATH-1] of WideChar;
    dwVpnStrategy             : DWORD;
    //  {$IFDEF WINVER_0x501_OR_GREATER}
    dwfOptions2               : DWORD;
    dwfOptions3               : DWORD;
    szDnsSuffix               : Array [0..RAS_MaxDnsSuffix] of WideChar;
    dwTcpWindowSize           : DWORD;
    szPrerequisitePbk         : Array [0..MAX_ PATH-1] of WideChar;
    szPrerequisiteEntry       : Array [0..RAS_MaxEntryName] of WideChar;
    dwRedialCount             : DWORD;
    dwRedialPause             : DWORD;
    //  {$IFDEF WINVER_0x600_OR_GREATER}
    //    ipv6addrDns   : RASIPV6ADDR;
    //    ipv6addrDnsAlt: RASIPV6ADDR;
    //  {$ENDIF}
    //    dwIPv4InterfaceMetric: DWORD;
    //    dwIPv6InterfaceMetric: DWORD;
  end;

  tagRASENTRY = tagRASENTRYA;
  RASENTRYA = tagRASENTRYA;
  RASENTRYW = tagRASENTRYW;
  RASENTRY = RASENTRYA;

const
// RASENTRY dwfOptions bit flags
```

```
  RASEO_RemoteDefaultGateway      = $00000010;
  RASEO_ModemLights               = $00000100;
  RASEO_RequireEncryptedPw        = $00000400;
  RASEO_RequireMsEncryptedPw      = $00000800;
  RASEO_RequireDataEncryption     = $00001000;
  RASEO_PreviewUserPw             = $01000000;
  RASEO_ShowDialingProgress       = $04000000;

// Біти прапорів RASENTRY dwfOptions

  RASEO2_DontNegotiateMultilink   = $00000004;
  RASEO2_ReconnectIfDropped       = $00000100;

// Біти прапорів RASENTRY dwProtocols

  RASNP_Ip = $00000004;

// Біти прапорів RASENTRY dwFramingProtocols

  RASFP_Ppp = $00000001;

// константи RASENTRY dwIdleDisconnectSeconds

  RASIDS_Disabled = $FFFFFFFF;

// рядок по замовчуванню RASENTRY szDeviceType

  RASDT_Vpn = 'vpn';

// значення RASENTRY dwDialMode

  RASEDM_DialAll = 1;

// Тип входу використаний, для визначення того, які властивості UI повині бути
представлені споживачу

  RASET_Vpn = 2; // VPN
// Немає ніякої різниці між RASCTRYINFOA та RASCTRYINFOW.

  ET_None    = 0; // Без шифрування
  VS_Default = 0; // по замовчуванню (PPTP)

function RasSetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; dwEntrySize: Longint; lpbDeviceInfo: Pointer; dwDeviceInfoSize:
Longint): Longint; stdcall;
function RasSetEntryProperties(lpszPhonebook, szEntry: PWideChar; lpbEntry:
Pointer; dwEntrySize: Longint; lpbDeviceInfo: Pointer; dwDeviceInfoSize:
Longint): Longint; stdcall;
function RasSetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; dwEntrySize: Longint; lpbDeviceInfo: Pointer; dwDeviceInfoSize:
Longint): Longint; stdcall;

function RasGetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; var lpdwEntrySize: Longint; lpbDeviceInfo: Pointer; var
lpdwDeviceInfoSize: Longint): Longint; stdcall;
function RasGetEntryProperties(lpszPhonebook, szEntry: PWideChar; lpbEntry:
Pointer; var lpdwEntrySize: Longint; lpbDeviceInfo: Pointer; var
lpdwDeviceInfoSize: Longint): Longint; stdcall;
function RasGetEntryProperties(lpszPhonebook, szEntry: PAnsiChar; lpbEntry:
Pointer; var lpdwEntrySize: Longint; lpbDeviceInfo: Pointer; var
lpdwDeviceInfoSize: Longint): Longint; stdcall;

function RasSetEntryDialParams(lpszPhonebook: PAnsiChar; lprasdialparams:
PRASDIALPARAMSA; fRemovePassword: BOOL): DWORD; stdcall;
function RasSetEntryDialParams(lpszPhonebook: PWideChar; lprasdialparams:
PRASDIALPARAMSW; fRemovePassword: BOOL): DWORD; stdcall;
function RasSetEntryDialParams(lpszPhonebook: PAnsiChar; lprasdialparams:
PRASDIALPARAMS; fRemovePassword: BOOL): DWORD; stdcall;
```

```
function RasSetCredentials(lpszPhoneBook, lpszEntry: PAnsiChar; var
lpCredentials: RASCREDENTIALSA; fRemovePassword: LongBool): Longint; stdcall;
function RasSetCredentials(lpszPhoneBook, lpszEntry: PWideChar; var
lpCredentials: RASCREDENTIALSW; fRemovePassword: LongBool): Longint; stdcall;
function RasSetCredentials(lpszPhoneBook, lpszEntry: PAnsiChar; var
lpCredentials: RASCREDENTIALS; fRemovePassword: LongBool): Longint; stdcall;

implementation

const
  raslib = 'rasapi32.dll';

function RasSetEntryProperties; external raslib name 'RasSetEntryProperties';
function RasSetEntryProperties; external raslib name 'RasSetEntryProperties';
function RasSetEntryProperties;  external raslib name 'RasSetEntryProperties';

function RasGetEntryProperties; external raslib name 'RasGetEntryProperties';
function RasGetEntryProperties; external raslib name 'RasGetEntryProperties';
function RasGetEntryProperties;  external raslib name 'RasGetEntryProperties';

function RasSetEntryDialParams; external raslib name 'RasSetEntryDialParams';
function RasSetEntryDialParams; external raslib name 'RasSetEntryDialParams';
function RasSetEntryDialParams;  external raslib name 'RasSetEntryDialParams';

function RasSetCredentials;     external raslib name 'RasSetCredentials';
function RasSetCredentials;     external raslib name 'RasSetCredentials';
function RasSetCredentials;      external raslib name 'RasSetCredentials';
end.
```

VPN_MyMsgBox.pas – повідомлення VPN

```
  unit VPN_MyMsgBox;

interface

uses
  Windows, Messages, VPN_SysUtils;

function ExtMessageBox(hWnd: HWND; pszText, pszCaption: PWideChar; dwFlags:
DWORD): Integer;

implementation

var
  hhk: HHOOK;
  ico: HICON;

//

function SysMsgProc(nCode: UINT; wParam: WPARAM; lParam: LPARAM): Integer;
stdcall;
begin
  case nCode of
    HCBT_ACTIVATE:
      begin
        if (ico <> 0) then
          SendMessage(wParam, WM_SETICON, ICON_SMALL, ico);
        SetCenterDialogPos(wParam, GetParent(wParam), TRUE);
        UnhookWindowsHookEx(hhk);
        Result := 0;
      end;
    else
      Result := CallNextHookEx(hhk, nCode, wParam, lParam);
  end;
end;

//

function ExtMessageBox(hWnd: HWND; pszText, pszCaption: PWideChar; dwFlags:
DWORD): Integer;
begin
  ico := GetClassLong(hWnd, GCL_HICON);
  if (ico = 0) then
    ico := SendMessage(hWnd, WM_GETICON, ICON_SMALL, 0);
  hhk := SetWindowsHookEx(WH_CBT, @SysMsgProc, hInstance, 0);
  Result := MessageBox(hWnd, pszText, pszCaption, dwFlags);
end;

end.
```

VPN_LinkStat.pas - інтерфейс користувача

```
unit VPN_LinkStat;

interface

uses
  Windows, Messages, CommCtrl, VPN_Windows;

const
  //
  SCM_EX_SETHOVERCLR  = WM_USER + 101; // установити колір для наведеного стану.
  SCM_EX_SETNORMALCLR = WM_USER + 102; // установити колір для звичайного стану.
  SCM_EX_SETPRESSCLR  = WM_USER + 103; // установити колір для натиснутого
стану.
  SCM_EX_SETBCKGNDCLR = WM_USER + 104; // установити колір для тла тексту.
  SCM_EX_SETTIPTEXT   = WM_USER + 105; // установити текст спливаючої підказки.
  //
  SCM_EX_GETHOVERCLR  = WM_USER + 111; // одержати колір для наведеного стану.
  SCM_EX_GETNORMALCLR = WM_USER + 112; // одержати колір для звичайного стану.
  SCM_EX_GETPRESSCLR  = WM_USER + 113; // одержати колір для натиснутого стану.
  SCM_EX_GETBCKGNDCLR = WM_USER + 114; // одержати колір для тла тексту.
  SCM_EX_GETTIPTEXT   = WM_USER + 115; // одержати текст спливаючої підказки.

// створення елемента управління Hyperlink.

procedure CreateStaticHyperlink(hWnd: HWND);

// видалення елемента управління Hyperlink.

procedure RemoveStaticHyperlink(hWnd: HWND);

implementation

type
  TLinkWndProc = function(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT; stdcall;

  P_LINK_PRO = ^T_LINK_PRO;
  T_LINK_PRO = packed record
    LinkProc  : TLinkWndProc;
    hCursor   : HCURSOR;
    hFont     : HFONT;
    rcClient  : TRect;
    //
    clrHover  : TColorRef;
    clrNormal : TColorRef;
    clrPress  : TColorRef;
    clrBckgnd : TColorRef; // CLR_NONE
    pszText   : Array [0..MAX_ PATH-1] of WideChar;
    //
    bIsHover  : Boolean;
    bIsPress  : Boolean;
    bIsEnabled: Boolean;
    //
    hToolTip  : HWND;
    ti        : TToolInfo;
    pszToolTip: Array [0..MAX_ PATH-1] of WideChar;
    //
    dtStyle   : DWORD;
    //
    hdcMem    : HDC;
    hbmMem    : HBITMAP;
    hbmOld    : HBITMAP;
  end;

var
  plp: P_LINK_PRO;
```

```
//

function LinkWndProc_OnSetHoverClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrHover := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetHoverClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrHover);
end;

//

function LinkWndProc_OnSetNormalClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrNormal := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetNormalClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrNormal);
end;

//

function LinkWndProc_OnSetPressClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrPress := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetPressClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrPress);
end;

//

function LinkWndProc_OnSetBckgdClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.clrBckgnd := TColorRef(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
```

```
  Result := 0;
end;

//

function LinkWndProc_OnGetBckgdClr(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := LRESULT(plp.clrBckgnd);
end;

//

function LinkWndProc_OnSetTipText(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  lstrcpyn(plp.pszToolTip, PWideChar(wParam), wParam);
  //
  Result := 0;
end;

//

function LinkWndProc_OnGetTipText(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  lstrcpyn(PWideChar(lParam), plp.pszToolTip, lstrlen(plp.pszToolTip) + 1);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmSetFont(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.hFont := HFONT(wParam);
  //
  Result := CallWindowProc(@plp.LinkProc, hWnd, uMsg, wParam, lParam);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
end;

//

function LinkWndProc_OnWmSetText(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  ZeroMemory(@plp.pszText, SizeOf(plp.pszText));
  lstrcpyn(plp.pszText, PWideChar(lParam), lParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
end;

//

function LinkWndProc_OnWmEnable(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.bIsEnabled := BOOL(wParam);
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//
```

```
function LinkWndProc_OnWmMouseLeave(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  pt: TPoint;
begin
  if IsWindow(plp.hToolTip) then
    SendMessage(plp.hToolTip, TTM_TRACKACTIVATE, Integer(FALSE), 0);
  //
  GetCursorPos(pt);
  ScreenToClient(hWnd, pt);
  //
  plp.bIsHover := FALSE;
  plp.bIsPress := FALSE;
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmMouseMove(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  tme: Windows.TTrackMouseEvent;
  pt : TPoint;
begin
  //
  GetCursorPos(pt);
  ScreenToClient(hWnd, pt);
  //
  tme.cbSize      := SizeOf(Windows.TTrackMouseEvent);
  tme.dwFlags     := TME_LEAVE;
  tme.hwndTrack   := hWnd;
  tme.dwHoverTime := HOVER_DEFAULT;
  //
  plp.bIsHover := Windows.TrackMouseEvent(tme) and PtInRect(plp.rcClient, pt);
  plp.bIsPress := {(wParam = MK_LBUTTON) and} (GetCapture = hWnd) and
PtInRect(plp.rcClient, pt);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmCaptureChanged(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  plp.bIsPress := FALSE;
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmNcHitTest(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  //
  Result := HTCLIENT;
end;

//

function LinkWndProc_OnWmlButtonDown(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
```

```
  //
  if IsWindow(plp.hToolTip) then
    SendMessage(plp.hToolTip, TTM_TRACKACTIVATE, Integer(FALSE), 0);
  plp.bIsPress := TRUE;
  SetFocus(hWnd);
  SetCapture(hWnd);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmlButtonUp(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  pt: TPoint;
begin
  //
  GetCursorPos(pt);
  ScreenToClient(hWnd, pt);
  if (PtInRect(plp.rcClient, pt) and (GetCapture = hWnd)) then
    SendMessage(GetParent(hWnd), WM_COMMAND, MakeLong(GetDlgCtrlID(hWnd),
STN_CLICKED), 0);
  // plp.bIsPress := FALSE;
  ReleaseCapture;
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmSetCursor(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
var
  pt: TPoint;
begin
  //
  if IsWindow(plp.hToolTip) then
    begin
      SendMessage(plp.hToolTip, TTM_TRACKACTIVATE, Integer(TRUE),
Integer(@plp.ti));
      GetCursorPos(pt);
      SendMessage(plp.hToolTip, TTM_TRACKPOSITION, 0, MakeLong(pt.x, pt.y));
    end;
  //
  if (plp.hCursor <> 0) then
   SetCursor(plp.hCursor);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmSize(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
var
  hdcIn: HDC;
begin
  GetClientRect(hWnd, plp.rcClient);
  //
  if (plp.hdcMem <> 0) then
    begin
      SelectObject(plp.hdcMem, plp.hbmOld);
      DeleteObject(plp.hbmMem);
      DeleteDC(plp.hdcMem);
```

```
    end;
  hdcIn := GetDC(hWnd);
  plp.hdcMem := CreateCompatibleDC(hdcIn);
  plp.hbmMem := CreateCompatibleBitmap(hdcIn, plp.rcClient.Right -
plp.rcClient.Left, plp.rcClient.Bottom - plp.rcClient.Top);
  plp.hbmOld := SelectObject(plp.hdcMem, plp.hbmMem);
  ReleaseDC(hWnd, hdcIn);
  //
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := CallWindowProc(@plp.LinkProc, hWnd, uMsg, wParam, lParam);
end;

//

function LinkWndProc_OnWmPaint(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT; stdcall;
var
  hdcIn : HDC;
  ps    : TPaintStruct;
  hbrNew: HBRUSH;
begin
  if (wParam = 0) then
    hdcIn := BeginPaint(hWnd, ps)
  else
    hdcIn := wParam;

  if (plp.clrBckgnd = CLR_DEFAULT) then
    FillRect(plp.hdcMem, plp.rcClient, HBRUSH(COLOR_BTNFACE + 1))
  else
    begin
      hbrNew := CreateSolidBrush(plp.clrBckgnd);
      FillRect(plp.hdcMem, plp.rcClient, hbrNew);
      DeleteObject(hbrNew);
    end;

  if plp.bIsEnabled then
    begin
      if (plp.bIsHover and plp.bIsPress) then
        SetTextColor(plp.hdcMem, plp.clrPress)
      else
      if (plp.bIsHover and not plp.bIsPress) then
        SetTextColor(plp.hdcMem, plp.clrHover)
      else
        SetTextColor(plp.hdcMem, plp.clrNormal);
    end
  else
    SetTextColor(plp.hdcMem, GetSysColor(COLOR_GRAYTEXT));

  SetBkMode(plp.hdcMem, TRANSPARENT);
  SetBkColor(plp.hdcMem, TRANSPARENT);

  SelectObject(plp.hdcMem, plp.hFont);

  DrawText(plp.hdcMem, plp.pszText, {lstrlen(plp.pszText)}-1, plp.rcClient,
plp.dtStyle);

  BitBlt(hdcIn, 0, 0, plp.rcClient.Right - plp.rcClient.Left,
plp.rcClient.Bottom - plp.rcClient.Top, plp.hdcMem, 0, 0, SRCCOPY);

  if (wParam = 0) then
    EndPaint(hWnd, ps);

  Result := 0;
end;

//
```

```
function LinkWndProc_OnWmEraseBkgnd(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  if (plp.clrBckgnd <> CLR_DEFAULT) then
   begin
     FillRect(HDC(wParam), plp.rcClient, HBRUSH(COLOR_BTNFACE + 1));
     //
     Result := 1;
   end
  else
    Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
end;

//

function LinkWndProc_OnWmSysColorChange(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT;
wParam: WPARAM; lParam: LPARAM): LRESULT;
begin
  RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE or RDW_UPDATENOW or RDW_NOERASE);
  //
  Result := 0;
end;

//

function LinkWndProc_OnWmNotify(plp: P_LINK_PRO; hWnd: HWND; uMsg: UINT; wParam:
WPARAM; lParam: LPARAM): LRESULT;
var
  pnmh: PNMHdr;
  ptit: PToolTipText;
begin
  //
  pnmh := PNMHdr(lParam);
  case pnmh.code of
    TTN_NEEDTEXTW:
      begin
        ptit := PToolTipText(lParam);
        ptit.lpszText := plp.pszToolTip;
      end;
  end;
  //
  Result := 0;
end;

//

function LinkWndProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
LRESULT; stdcall;
begin

  plp := P_LINK_PRO(GetWindowLong(hWnd, GWL_USERDATA));

  if (plp = nil) then
    begin
      Result := DefWindowProc(hWnd, uMsg, wParam, lParam);
      Exit;
    end;

  case uMsg of

    //

    SCM_EX_SETHOVERCLR:
      begin
        Result := LinkWndProc_OnSetHoverClr(plp, hWnd, uMsg, wParam, lParam);
      end;

    //
```

```
SCM_EX_GETHOVERCLR:
  begin
    Result := LinkWndProc_OnGetHoverClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETNORMALCLR:
  begin
    Result := LinkWndProc_OnSetNormalClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETNORMALCLR:
  begin
    Result := LinkWndProc_OnGetNormalClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETPRESSCLR:
  begin
    Result := LinkWndProc_OnSetPressClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETPRESSCLR:
  begin
    Result := LinkWndProc_OnGetPressClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETBCKGNDCLR:
  begin
    Result := LinkWndProc_OnSetBckgdClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETBCKGNDCLR:
  begin
    Result := LinkWndProc_OnGetBckgdClr(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_SETTIPTEXT:
  begin
    Result := LinkWndProc_OnSetTipText(plp, hWnd, uMsg, wParam, lParam);
  end;

//

SCM_EX_GETTIPTEXT:
  begin
    Result := LinkWndProc_OnGetTipText(plp, hWnd, uMsg, wParam, lParam);
  end;

//

WM_DESTROY:
  begin
    RemoveStaticHyperlink(hWnd);
  end;

//
```

```
    WM_SETFONT:
      begin
        Result := LinkWndProc_OnWmSetFont(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_SETTEXT:
      begin
        Result := LinkWndProc_OnWmSetText(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_ENABLE:
      begin
        Result := LinkWndProc_OnWmEnable(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_MOUSELEAVE:
      begin
        Result := LinkWndProc_OnWmMouseLeave(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_MOUSEMOVE:
      begin
        Result := LinkWndProc_OnWmMouseMove(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_CAPTURECHANGED:
      begin
        Result := LinkWndProc_OnWmCaptureChanged(plp, hWnd, uMsg, wParam,
lParam);
      end;

    //

    WM_NCHITTEST:
      begin
        Result := LinkWndProc_OnWmNcHitTest(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_LBUTTONDOWN:
      begin
        Result := LinkWndProc_OnWmlButtonDown(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_LBUTTONUP:
      begin
        Result := LinkWndProc_OnWmlButtonUp(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_SETCURSOR:
      begin
        Result := LinkWndProc_OnWmSetCursor(plp, hWnd, uMsg, wParam, lParam);
      end;
```

```
    //

    WM_SIZE:
      begin
        Result := LinkWndProc_OnWmSize(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_PRINTCLIENT,
    WM_PAINT,
    WM_UPDATEUISTATE: // перемальовування вікна без виклику WM_PAINT.
      begin
        Result := LinkWndProc_OnWmPaint(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_ERASEBKGND:
      begin
        Result := LinkWndProc_OnWmEraseBkgnd(plp, hWnd, uMsg, wParam, lParam);
      end;

    //

    WM_SYSCOLORCHANGE:
      begin
        Result := LinkWndProc_OnWmSysColorChange(plp, hWnd, uMsg, wParam,
lParam);
      end;

    //

    WM_NOTIFY:
      begin
        Result := LinkWndProc_OnWmNotify(plp, hWnd, uMsg, wParam, lParam);
      end;

    else
      Result := CallWindowProc(@plp.LinkProc, hWnd, uMsg, wParam, lParam);
  end;

end;

//

procedure CreateStaticHyperlink(hWnd: HWND);
var
  iccex  : TInitCommonControlsEx;
  dtStyle: DWORD;
  dwLen  : Integer;
begin

  InitCommonControls;
  iccex.dwSize := SizeOf(TInitCommonControlsEx);
  iccex.dwICC  := ICC_BAR_CLASSES;
  InitCommonControlsEx(iccex);

  RemoveStaticHyperlink(hWnd);

  plp := P_LINK_PRO(HeapAlloc(GetProcessHeap, HEAP_ZERO_MEMORY,
SizeOf(T_LINK_PRO)));

  ZeroMemory(plp, SizeOf(plp));
  plp.LinkProc   := TLinkWndProc(Pointer(GetWindowLong(hWnd, GWL_WNDPROC)));
  plp.hCursor    := LoadImage(0, MAKEINTRESOURCEW(IDC_HAND), IMAGE_CURSOR, 0, 0,
LR_SHARED or LR_DEFAULTSIZE);
```

```
  plp.hFont      := SendMessage(hWnd, WM_GETFONT, 0, 0);

  GetClientRect(hWnd, plp.rcClient);

  plp.clrHover   := RGB(255, 0, 0);
  plp.clrNormal  := RGB(0, 0, 255);
  plp.clrPress   := RGB(0, 0, 128);
  plp.clrBckgnd  := CLR_DEFAULT;

  dwLen := SendMessage(hWnd, WM_GETTEXTLENGTH, 0, 0);
  if (dwLen > 0) then
    begin
      ZeroMemory(@plp.pszText, SizeOf(plp.pszText));
      SendMessage(hWnd, WM_GETTEXT, SizeOf(plp.pszText), Integer(@plp.pszText));
    end;

  plp.bIsHover   := FALSE;
  plp.bIsPress   := FALSE;
  plp.bIsEnabled := IsWindowEnabled(hWnd);

  plp.hToolTip   := CreateWindowEx(WS_EX_TOPMOST, TOOLTIPS_CLASS, nil, WS_POPUP
or TTS_NOPREFIX or TTS_ALWAYSTIP, Integer(CW_USEDEFAULT),
Integer(CW_USEDEFAULT), Integer(CW_USEDEFAULT), Integer(CW_USEDEFAULT),
GetParent(hWnd), 0, hInstance, nil);
  if IsWindow(plp.hToolTip) then
    begin
      plp.ti.cbSize   := SizeOf(TToolInfo);
      plp.ti.uFlags   := TTVPN_SUBCLASS or TTVPN_IDISHWND;
      plp.ti.hwnd     := hWnd;
      plp.ti.uId      := hWnd;
      plp.ti.lpszText := LPSTR_TEXTCALLBACKW;
      SetRectEmpty(plp.ti.Rect);
      ZeroMemory(@plp.pszToolTip, SizeOf(plp.pszToolTip));
      SendMessage(plp.hToolTip, TTM_ADDTOOLW, 0, Integer(@plp.ti));
    end;

  dtStyle := GetWindowLong(hWnd, GWL_STYLE);

  case (dtStyle and SS_TYPEMASK) of
    SS_LEFT          : plp.dtStyle := DT_LEFT or DT_EXPANDTABS {or
DT_WORDBREAK};
    SS_CENTER        : plp.dtStyle := DT_CENTER or DT_EXPANDTABS {or
DT_WORDBREAK};
    SS_RIGHT         : plp.dtStyle := DT_RIGHT or DT_EXPANDTABS {or
DT_WORDBREAK};
    SS_SIMPLE        : plp.dtStyle := DT_LEFT or DT_SINGLELINE;
    SS_LEFTNOWORDWRAP: plp.dtStyle := DT_LEFT or DT_EXPANDTABS;
  end;
  if ((dtStyle and SS_CENTERIMAGE) = 0) then
    plp.dtStyle := plp.dtStyle or DT_VCENTER;
  if ((dtStyle and SS_NOTIFY) = 0) then
    SetWindowLong(hWnd, GWL_STYLE, dtStyle or SS_NOTIFY);

  SetWindowLong(hWnd, GWL_USERDATA, Longint(plp));

  SetWindowLong(hWnd, GWL_WNDPROC, Longint(@LinkWndProc));

  // так як ми створюємо hdcMem заново при зміні розмірів вікна елемента
  // управління, то не будемо тут створювати споконвічно контексти, а просто
  // повідомимо елемент управління повідомленням про зміну розмірів.

  SendMessage(hWnd, WM_SIZE, 0, 0); // RedrawWindow(hWnd, nil, 0, RDW_INVALIDATE
or RDW_UPDATENOW or RDW_NOERASE);

end;

//

procedure RemoveStaticHyperlink(hWnd: HWND);
```

```
begin

  plp := P_LINK_PRO(GetWindowLong(hWnd, GWL_USERDATA));
  if (plp <> nil) then
    begin

      if (plp.hCursor <> 0) then
        DestroyCursor(plp.hCursor);

      plp.ti.hwnd := hWnd;
      plp.ti.uId  := hWnd;
      if IsWindow(plp.hToolTip) then
        begin
          SendMessage(plp.hToolTip, TTM_DELTOOLW, 0, Integer(@plp.ti));
          DestroyWindow(plp.hToolTip);
        end;

      if (plp.hdcMem <> 0) then
        begin
          SelectObject(plp.hdcMem, plp.hbmOld);
          DeleteObject(plp.hbmMem);
          DeleteDC(plp.hdcMem);
        end;

      //
      SetWindowLong(hWnd, GWL_WNDPROC, Longint(@plp.LinkProc));
      RedrawWindow(hWnd, @plp.rcClient, 0, RDW_INVALIDATE or RDW_ERASE);

      SetWindowLong(hWnd, GWL_USERDATA, 0);
      HeapFree(GetProcessHeap, 0, plp);
    end;
end;
end.
```

VPN_SettWind.pas - параметри програми роботи з VPN

```
unit VPN_SettWind;

interface

uses
  Windows, Messages, CommCtrl, VPN_FileInfo, VPN_LinkStat, VPN_SysUtils,
VPN_MyMsgBox,
  VPN_Controls, VPN_Resources, VPN_ScanProc;

function SettDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;

implementation

//

function SettDlgProc_OnWmInitDialog(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
lParam: LPARAM): LRESULT;
var
  bldfnt: HFONT;
begin

  //

  hApp[1] := hWnd;

  //

  CreateStaticHyperlink(GetDlgItem(hApp[1], IDC_STATIC_SERVER));

  //

  SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_ADDSTRING, 0,
    Integer(@pszServ[1]));
  SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_SETCURSEL, 0, 0);

  //

  bldfnt := GetWindowBoldFont(hApp[1], GetWindowFontSize(hApp[1], 8));
  if (bldfnt <> 0) then
    SendMessage(GetDlgItem(hApp[1], IDC_STATIC_WARN), WM_SETFONT,
      Integer(bldfnt), Integer(TRUE));

  //

  Result := 0;

end;

//

function SettDlgProc_OnWmCommand(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
const
  dwRes: Array [Boolean] of DWORD = (PSWIZB_BACK, PSWIZB_BACK or PSWIZB_NEXT);
var
  dwEntry: DWORD;
  dwUser : DWORD;
  dwPass : DWORD;
begin

  //

  case HiWord(wParam) of
```

```
      //

      BN_CLICKED:
        case LoWord(wParam) of

          //

          IDC_STATIC_SERVER:
            begin

              DialogBox(hInstance, MAKEINTRESOURCEW(RC_DIALOG_UPDATE), hApp[1],
                @ScanDlgProc);

            end;

        end;

      //

      EN_UPDATE:
        case LoWord(wParam) of
          IDC_STATIC_ENTRY,
          IDC_STATIC_USER,
          IDC_STATIC_PASSW:
          begin

            dwEntry := SendMessage(GetDlgItem(hApp[1], IDC_STATIC_ENTRY),
              WM_GETTEXTLENGTH, 0, 0);
            dwUser := SendMessage(GetDlgItem(hApp[1], IDC_STATIC_USER),
              WM_GETTEXTLENGTH, 0, 0);
            dwPass := SendMessage(GetDlgItem(hApp[1], IDC_STATIC_PASSW),
              WM_GETTEXTLENGTH, 0, 0);

            SendMessage(
              GetParent(hApp[1]),
              PSM_SETWIZBUTTONS,
              0,
              dwRes[(dwEntry > 0) and (dwUser > 0) and (dwPass > 0)]
            );

          end;
        end;

    end;

    //

    Result := 0;

  end;

  //

  function SettDlgProc_OnWmCtlColorStatic(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
  lParam: LPARAM): LRESULT;
  begin

    //

    case GetDlgCtrlId(lParam) of
      IDC_STATIC_WARN:
      begin

        SetBkMode(wParam, TRANSPARENT);
        SetTextColor(wParam, RGB(255, 0, 0));
        Result := GetStockObject(NULL_BRUSH);

      end;
    else
```

```
    Result := 0;

  end;

end;

//

function SettDlgProc_OnWmNotify(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  pnmh : PNMHdr;
  dwRes: DWORD;
begin

  //

  pnmh := PNMHdr(lParam);

  case pnmh.code of

    //

    PSN_WIZNEXT:
    begin

      SendMessage(GetParent(hApp[1]), PSM_SETWIZBUTTONS, 0,
        Integer(PSWIZB_NEXT));

    end;

    //

    PSN_SETACTIVE:
    begin

       SendMessage(hApp[1], WM_COMMAND, MAKELPARAM(IDC_STATIC_ENTRY, EN_UPDATE),
         0);
       SendMessage(hApp[1], WM_COMMAND, MAKELPARAM(IDC_STATIC_USER, EN_UPDATE),
         0);
       SendMessage(hApp[1], WM_COMMAND, MAKELPARAM(IDC_STATIC_PASSW, EN_UPDATE),
         0);

    end;

    //

    PSN_QUERYCANCEL:
    begin

      dwRes := ExtMessageBox(
        GetParent(hApp[1]),
        MAKEINTRESOURCEW(LoadStrInst(hInstance, RC_STRING_QCANCEL)),
        MAKEINTRESOURCEW(exeInfo.pszProductName),
        MB_YESNO or MB_ICONASTERISK
      );

      SetWindowLong(hApp[1], DWL_MSGRESULT, Integer(dwRes = IDNO));

    end;

    //

    PSN_WIZBACK:
    begin

      SendMessage(GetParent(hApp[1]), PSM_SETCURSEL, GetParent(hApp[1]),
        Integer(ahpsp[1]));
```

```
    end;

  end;

  //

  Result := 1;

end;


//

function SettDlgProc_OnWmDestroy(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  bldfnt: HFONT;
begin

  //

  RemoveStaticHyperlink(GetDlgItem(hApp[1], IDC_STATIC_SERVER));

  //

  bldfnt := HFONT(SendMessage(GetDlgItem(hApp[1], IDC_STATIC_WARN),
    WM_GETFONT, 0, 0));
  if (bldfnt <> 0) then
    DeleteObject(bldfnt);

  //

  Result := 0;

end;

//

function SettDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;
begin

  case uMsg of

    //

    WM_INITDIALOG:
    begin

      Result := BOOL(SettDlgProc_OnWmInitDialog(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_COMMAND:
    begin

      Result := BOOL(SettDlgProc_OnWmCommand(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_CTLCOLORSTATIC:
    begin
```

```
      Result := BOOL(SettDlgProc_OnWmCtlColorStatic(hWnd, uMsg, wParam,
lParam));

    end;

    //

    WM_NOTIFY:
    begin

      Result := BOOL(SettDlgProc_OnWmNotify(hWnd, uMsg, wParam, lParam));

    end;

    //

    WM_DESTROY:
    begin

      Result := BOOL(SettDlgProc_OnWmDestroy(hWnd, uMsg, wParam, lParam));

    end;

  else
    Result := FALSE;
  end;

end;

end.
```

VPN_ScanProc.pas - пошук підключень VPN

```
unit VPN_ScanProc;

interface

uses
  Windows, Messages, CommCtrl, WinSock, VPN_SysUtils, VPN_StatAnim,
VPN_Resources;

function ScanDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;

implementation

//

function ThreadCallback(LpParameter: Pointer): DWORD; stdcall;
type
  TaPInAddr = Array [0..MAX_ PATH-1] of PInAddr;
  PaPInAddr = ^TaPInAddr;
var
  pszText: WideString;
  pszUTF8: AnsiString;
  dwErr  : DWORD;
  phe    : PHostEnt;
  addr   : PaPInAddr;
  ws     : TWSAData;
  i      : Integer;
begin

  //

  Result := 0;

  //

  SetThreadPriority(hThread, THREAVPN_PRIORITY_BELOW_NORMAL);

  //

  dwErr := WSAStartup(MAKEWORD(1, 0), ws);
  if (dwErr = NOERROR) then
  try
    pszUTF8 := WideStringToAnsi(pszServ, CP_ACP);
    phe := GetHostByName(@pszUTF8[1]);
    if (phe <> nil) then
    begin
      addr := PaPInAddr(phe^.h_addr_list);
      i := 0;
      SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_RESETCONTENT, 0, 0);
      SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_ADDSTRING, 0,
        Integer(@pszServ[1]));
      while (addr^[I] <> nil) do
      begin
        pszUTF8 := inet_ntoa(addr[I]^);
        pszText := AnsiStringToWide(pszUTF8, CP_ACP);
        SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_ADDSTRING, 0,
          Integer(@pszText[1]));
        pszText := Format(LoadStrInst(hInstance, RC_STRING_IPHOST), [pszText]);
        SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ADDRESS), WM_SETTEXT, 0,
          Integer(@pszText[1]));
        Inc(i);
        Sleep(35);
      end;
      SendMessage(GetDlgItem(hApp[1], IDC_COMBO_SERVER), CB_SETCURSEL, 0, 0);
    end;
```

```
  finally
    WSACleanup;
  end;

  //

  SendMessage(hApp[3], WM_DESTROY, 0, 0);

end;

//

function ScanDlgProc_OnWmInitDialog(hWnd: HWND; uMsg: UINT; wParam: WPARAM;
lParam: LPARAM): LRESULT;
var
  pszText : WideString;
  ThreadID: LongWord;
  himl    : HIMAGELIST;
begin

  //

  hApp[3] := hWnd;

  //

  SetCenterDialogPos(hApp[3], hApp[1], TRUE);

  //

  pszText := Format(LoadStrInst(hInstance, RC_STRING_IPSERVER), [pszServ]);
  SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ADDRESS), WM_SETTEXT, 0,
    Integer(@pszText[1]));

  //

  CreateAnimateStatic(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE));
  himl := ImageList_LoadImage(hInstance, MAKEINTRESOURCEW(RC_BITMAP_WAITING),
    GetSystemMetrics(SM_CXSMICON), 0, CLR_DEFAULT, IMAGE_BITMAP, LR_DEFAULTCOLOR
    or LR_CREATEDIBSECTION);
  if (himl <> 0) then
    SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE), SS_SETIMAGELIST, himl,
      0);

  //

  hThread := CreateThread(nil, 0, @ThreadCallback, nil, 0, ThreadID);
  if (hThread <> 0) then
    begin
      CloseHandle(hThread);
      hThread := 0;
    end;

  //

  Result := 0;

end;

//

function ScanDlgProc_OnWmDestroy(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam:
LPARAM): LRESULT;
var
  himl: HIMAGELIST;
begin

  //
```

```
  if (hThread <> 0) then
    begin
      CloseHandle(hThread);
      hThread := 0;
    end;

  //

  himl := SendMessage(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE), SS_GETIMAGELIST,
    0, 0);
  if (himl <> 0) then
    ImageList_Destroy(himl);
  RemoveAnimateStatic(GetDlgItem(hApp[3], IDC_STATIC_ANIMATE));

  //

  EndDialog(hApp[3], wParam);

  //

  Result := 0;

end;

//

function ScanDlgProc(hWnd: HWND; uMsg: UINT; wParam: WPARAM; lParam: LPARAM):
BOOL; stdcall;
begin

  case uMsg of

    //

    WM_INITDIALOG:
      begin
        Result := BOOL(ScanDlgProc_OnWmInitDialog(hWnd, uMsg, wParam, lParam));
      end;

    //

    WM_DESTROY:
      begin
        Result := BOOL(ScanDlgProc_OnWmDestroy(hWnd, uMsg, wParam, lParam));
      end;

  else
    Result := FALSE;
  end;

end;

end.
```