МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Петров Євген Олександрович

Програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7

Спеціальність: 125 «Кібербезпека»

Освітній ступінь: бакалавр

Науковий керівник:

Смірнов Сергій Анатолійович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Петрову Євгену Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7*

керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 185-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7*

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

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

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

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

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Петров Є.О. Програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи кібербезпеки для протидії атаці на протокол SS7.

Метою розробки є програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7.

Результат роботи – програмна реалізація системи кібербезпеки для протидії атаці на протокол SS7.

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 XP/Vista/7/8/10.

Програму розроблено в середовищі Delphi 10.4.1.

Ключові слова: кібербезпека, SS7

ABSTRACT

Petrov Ye.O. Cybersecurity system software to counter SS7 attack. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

This undergraduate qualification has developed software that is designed for cybersecurity to counter the attack on the SS7 protocol.

The purpose of the development is cybersecurity system software to counter the attack on the SS7 protocol.

The result is a software implementation of a cybersecurity system to counter the attack on the SS7 protocol.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the Delphi 10.4.1 environment.

Keywords: cybersecurity, SS7

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДПФ	–	дискретне перетворення Фур'є
ДПХ	–	дискретне перетворення Хартлі
ЕОМ	–	електрона обчислювальна машина
ЛОМ	–	локальна обчислювальна мережа
НСД	–	несанкціонований доступ
ПЗ	–	програмне забезпечення
СУБД	–	система управління базами даних
СВВ	–	система виявлення вторгнень
ШПФ	–	швидке перетворення Фур'є
ШПХ	–	швидке перетворення Хартлі
QoS	–	Quality of Service, якість обслуговування
SS7	–	сигналізаційна система № 7, відноситься до мережі передачі даних і до ряду технічних протоколів або правил, які регулюють обмін даними ними

ВСТУП

Актуальність теми. Як ми захищаємося на сьогоднішній день від різних атак, націлених на одержання вашого пароля? Хтось робить максимально складні паролі, хтось установлює системи захисту від брутфорса і т.д. Але самі певні у своїй захищеності ті, хто користується популярною сьогодні послугою – двофакторною авторизацією за допомогою SMS. Такі послуги надають банки, відомі мессенджери (viber, telegram і т.д.), соціальні мережі (facebook і т.д.), поштові системи (gmail, ukr.net і т.д.) і велика кількість інших інтернет ресурсів, що вимагають реєстрацію. Не просто так це метод вважається самим безпечним. Адже, апріорі, ваш мобільний телефон завжди при вас, як гаманець і інші особисті речі, за якими ви пильно стежите. До того ж він є другим фактором автентифікації (використовується тільки після введення основного пароля). Але, як відомо, якщо ви забули свій пароль, завжди його можна відновити за допомогою SMS на ваш телефон.

Уже зараз є можливість за невелику суму грошей при наявності середнього рівня майстерності одержати всі ваші SMS так, що ви про цей ніколи не довідаєтеся.

Список ресурсів, куди може одержати доступ зловмисник навіть складно представити (соцмережі, мессенджери, підтвердження банківських операцій і т.д.). Адже в нас «усе життя» прив'язан о до нашого номера телефону. Неспроста було введено виправлення, яке дозволяє переносити номер телефону при зміні оператора. Зараз змінити номер – це як почати нове життя із чистого аркуша.

Трапився такий розрив шаблону завдяки уразливості протоколу SS7 у мережах операторів стільниковому зв'язка.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7.

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

– Огляд існуючих систем кібербезпеки для протидії атаці на протокол SS7.

– Дослідження системи кібербезпеки для протидії атаці на протокол SS7.

– Програмна реалізація системи кібербезпеки для протидії атаці на протокол SS7.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для протидії атаці на протокол SS7.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Лавиночиний ріст ресурсів глобальної мережі Інтернет, тенденція до підвищення мобільності абонентів, конвергенція технологій передачі різних видів трафіку, а також поява мобільних користувацьких застосунків викликають чималий інтерес до питання забезпечення ефективної передачі різнорідного трафіку через бездротові мережі зв'язку. Можливості мобільного доступу користувачів до Інтернет у цей час асоціюються в основному із двома технологіями – це стільникові мережі зв’язку й бездротовий ЛОМ. Багато Інтернет-застосунків як транспортний протокол використовують протокол TCP. Стік протоколів TCP/IP у свій час був розроблений для провідних мереж зв'язку, і його застосування в бездротових мережах часто приводить до значного погіршення характеристик зв’язку. У даній роботі розглянуті особливості мереж мобільному зв'язку, їх вплив на роботу протоколу TCP і основні напрямки подолання існуючих труднощів передачі користувацького трафіку.

Особливості передачі даних у мережах мобільного зв'язку

Основні особливості передачі трафіку в стільникових мережах і бездротових ЛОМ ми розглянемо з урахуванням специфіки стека TCP/IP стосовно до канального й транспортного рівнів еталонної моделі взаємодії відкритих систем (OSI). Можна виділити як загальні, так і специфічні характеристики названих мереж, що впливають на ефективність передачі трафіку.

1.2 Область застосування

Загальні характеристики

На параметри передачі даних по стільникових мережах і бездротовим ЛОМ істотний вплив виявляють затримки й канальні помилки. Затримку доставки кадра канального рівня можна розкласти на наступні основні складові:

1. Час перебування кадра в черзі передавального буфера перед моментом його передачі.

2. Час очікування моменту первісного доступу до каналу із множинним доступом. Певну затримку перед передачею першого або чергового кадра вносять ряд протоколів (наприклад, механізм RTS/CTS у бездротових ЛОМ стандарту IEEE 802.11).

3. Затримки повторної передачі в системах з випадковим множинним доступом. У бездротових ЛОМ механізми Mac-рівня можуть значно збільшувати затримку повторної передачі кадрів, щоб стабілізувати роботу мережі. Багато Mac-протоколи збільшують затримку при росту числа невдалих передач.

4. Властиво затримка передачі кадра, рівна частці від розподілу розміру кадра на швидкість передачі.

5. Затримка поширення сигналу у фізичному каналі, значення якої для стільникових мереж і бездротових ЛОМ суттєво не відрізняється від такої для провідних каналів зв'язку.

6. Час обробки окремого кадра, що включає в себе завадостійке кодування, перемежування, диспетчеризацію потоків з різною якістю обслуговування (Quality of Service – QoS), шифрування й ін.

7. Затримка, обумовлена процесом складання пакета на прийомній стороні.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Розглянемо технології перехоплення інформації в мережах стільникового зв'язку.

Клонування SIM карти

Однією з розповсюджених проблем є клонування SIM карти. В Інтернеті часто можна зустріти оголошення про легкий спосіб клонуванні карти, а також представлена безліч утиліт, наприклад, SIM Card Seizure. У якості цілей клонування звичайно вказують можливість безкоштовно дзвонити за чужий рахунок і можливість прослуховування розмов власника клонованої SIM-карти. У першому варіанті використання у власника клону будуть проблеми з одержанням вхідних дзвінків, а от вихідні можна робити вільно. Основними споживачами є люди, які потім у метро пропонують перехожим дешево подзвонити в будь-яку країну світу. Що стосується прослуховування абонента, то розгляду цього питання присвячений наступний розділ.

Процес перевірки дійсності SIM-карти. Базовими в цьому процесі є параметри IMSI і KI. Для того щоб клон міг пройти автентифікацію в AUC, він повинен знати ці параметри. Довідатися IMSI просто, він може бути записаний на самій карті або додаватися до неї. Його легко можна прочитати з SIM-карти за допомогою пристрою читання смарт-карт. А от з KI усі декілька складніше.

Як ви вже знаєте, KI зберігається всього у двох місцях – у пам'яті SIM-карти й у пам'яті AUC. KI ніколи не передається у відкритому виді при автентифікації, тобто його не можна перехопити при автентифікації. У зловмисників є 4 варіанта одержання KI. Перший варіант це інсайдер у компанії-операторові. Цей варіант переважніше, тому що можна одержати інформацію

відразу по декільком картам. Недоліки цього варіанта полягають у тому, що через значимість KI доступ до їхніх значень строго обмежений і при виявленні масового витоку інсайдер швидко буде обчислений. Крім того, найчастіше в AUC відсутній функціонал для зчитування KI з тих же міркувань безпеки. Другий варіант заснований на викраденні KI відразу після одержання партії SIM-карт від виробника. Проблеми тут ті ж що й у попередньому варіанті: кількість людей, що мають потрібні доступи, обчислюється одиницями.

Третій варіант: зчитати KI з пам'яті SIM-карти. Почнемо з того що необхідно одержати фізичний доступ до карти (вийняти її з телефону жертви під якимось приводом, знати PIN код). Важливий недолік: в SIM-карти немає інтерфейсу, по якому можна безпосередньо вважати або змінити KI.

І нарешті, останній варіант: обчислити KI. Зловмисник повинен мати відомості про використовуваний оператором алгоритмі A3. У цьому випадку можна спробувати обчислити KI, спостерігаючи за результатами перетворення RAND в SRES. Для цього вручну формують RAND, викликають алгоритм шифрування й передають йому RAND. Цей процес автоматизують такі програми як SIMscan і Woronscan.

Саме в такий спосіб були отримані перші клони SIM-карт. Це стало доступно через витік відомостей про алгоритм A3, називаної COMP128 у мережу. В алгоритмі була виявлена уразливість, яка дозволяла підбирати KI за прийнятну кількість спроб. Після виявлення уразливості більшість операторів замінила його чимсь більш стійким. На теперішній момент існує три версії COMP128. І хоча в мережі присутні програми, що декларують можливість злому цих версій, на перевірку завжди виявляється, що їх ціль – змусити користувача скачати «трояна».

Якщо ж зловмисник не має відомостей про реалізацію A3, то він може спробувати підібрати KI шляхом перебору (brute force). Тут виникає ще одна перешкода: кількість спроб для добору KI обмежене. В SIM-карти є вбудований лічильник кількості викликів A3, і при перевищенні певного порога (65535) карта

блокується й перестає відповідати на запити реєстрації (хоча інші функції працюють, наприклад, телефонна книга). У звичайних умовах експлуатації, коли А3 викликається при кожній реєстрації SIM-карти в мережі (при включенні телефону), подібні обмеження не заважають абонентові. А от для одержання KI може знадобитися більша кількість спроб.

Якщо ж зловмисникові вдалося підібрати KI, то він одержує можливість дзвонити за чужий рахунок. Але отут є декілька обмежуючих факторів. По-перше, тому що гроші на рахунку почнуть витрачатися швидше, чим звичайно, досить імовірно, що власник SIM-карти може це помітити. У детальній роздруківці відразу виявляться «зайві» дзвінки. Це стосується й «безлімітних» тарифів, тому що в них теж є обмеження, зокрема при дзвінках за кордон. Тому, зловмисники прагнуть якомога швидше виговорити весь доступний баланс і позбутися клону. По-друге, якщо обидві карти зареєстровані в мережі, те вхідні дзвінки будуть приходити на карту, яка остання авторизувалася, або з якої був зроблений останній вихідний дзвінок. Відповідно, легітимний користувач може помітити, що йому перестануть приходити очікувані дзвінки. Зловмисникам з метою конспірації взагалі протипоказане знімати трубку. Інакше кореспонденти користувача відразу виявлять шахрайство. По-третє, оператор може обчислювати SIM-карти, які реєструються в мережі в географічно рознесених місцях протягом обмеженого часу. При підозрах у клонуванні карти оператор заблокує карту й видасть абонентові нову.

Резюмуючи, можна сказати, що клонувати SIM-карти можливо, але досить важко. Якщо оператор вчасно модернізував реалізацію А3, а його співробітники лояльні й непідкупні, то абонентам не варто боятися появи клонів своєї SIM-карти. Крім того, актуальність такого шахрайства спадає, тому що попит на дешеві дзвінки за кордон компенсується можливістю дзвінків в Skype, а також пропозиціями від легальних операторів.

Перехоплення розмов у мережі GSM

Переходимо до розгляду злому GSM. Статті про уразливості в A5/1 з'явилися близько 15 років тому, але публічної демонстрації злому A5/1 в умовах реального світу дотепер не було. Більше того, як видне з опису роботи мережі треба розуміти, що крім злому самого алгоритму шифрування потрібно застосувати ще ряд сугубо інженерних проблем, які звичайно завжди опускаються з розгляду (у тому числі на публічних демонстраціях).

Більшість статей по злому GSM опираються на статтю Елі Баркана в 2006 році й дослідження Карстена Нола (Karsten Noh).

У своїй статті Баркан зі співавторами показав, що тому що в GSM корекція помилок іде до шифрування (а треба б навпаки), можливо певне зменшення простору пошуку для добору КС і реалізація known-ciphertext атаки (при повністю пасивному прослуховуванні ефіру) за прийнятний час за допомогою попередньо обчислених даних.

Самі автори статті говорять, що при прийманні без перешкод для злому протягом 2 хвилин потрібно 50 терабайт передрозрахованих даних. У тій же статті (у розділі про A5/2) вказується, що сигнал з ефіру завжди йде з перешкодами, які ускладнюють добору ключа. Для A5/2 наведений змінений алгоритм, який здатний ураховувати перешкоди, але при цьому вимагає вдвічі більшого обсягу передрозрахованих даних і, відповідно, час злому збільшує у два раз. Для A5/1 зазначена можливість побудови аналогічного алгоритму, але сам він не наведений. Можна припустити, що в цьому випадку також потрібно побільшати обсяг передрозрахованих даних удвічі.

Процес добору ключа A5/1 є імовірнісним і залежить від часу, тобто чому довше йде прослуховування, тим більше ймовірність підібрати КС. Таким чином, заявлені в статті 2 хвилини – це зразкове, а не гарантований час добору КС.

Карстен Нол розробляє найвідоміший проект по злому GSM мереж. Його фірма, що займається проблемами комп'ютерної безпеки, збиралася викласти у

відкритий доступ райдужні таблиці сесійних ключів алгоритму A5/1, який використовується для шифрування мови в мережах GSM.

Свій демарш проти A5/1 Карстен Нол пояснює бажанням привернути увагу громадськості до існуючої проблеми й змусити операторів зв'язку переходити на більш досконалі технології. Наприклад, технологія UMTS припускає використання 128 бітного алгоритму A5/3, стійкість якого така, що ніякими доступними засобами на сьогоднішній день зламати його не вдасться.

По розрахунках Карстена, повна таблиця ключів A5/1 в упакованому виді буде займати 128 петабайт і розподілено зберігатися на безлічі комп'ютерів у мережі. Для її розрахунків буде потрібно близько 80 комп'ютерів і 2-3 місяця роботи. Істотне зменшення часу обчислень повинне виявити використання сучасних CUDA графічних карт і програмувальних МАСивів Xilinx Virtex. Зокрема багато шуму наробило його виступ на 26C3 (Chaos Communication Congress) у грудні 2009 року. Коротко сформулювати суть виступу можна так: незабаром можна чекати поява бюджетних системи для онлайн декодування A5/1.

Переходимо до інженерних проблем. Як одержати дані з ефіру? Для перехоплення розмов треба мати повноцінний сканер, який повинен уміти розбиратися, які базові віщають навколо, на яких частотах, яким операторам вони належать, які телефони з якими TMSI у даний момент активні. Сканер повинен уміти стежити за розмовою із зазначеного телефону, коректно обробляти переходи на інші частоти й базові станції.

В інтернеті є пропозиції по придбанню подібного сканера без дешифратора за 40-50 тис. доларів. Це не можна назвати бюджетним пристроєм.

Таким чином, для створення приладу, який після нескладних маніпуляцій міг починати прослуховувати розмова по телефону, необхідно:

а) реалізувати частина, яка працює з ефіром. Зокрема, дозволяє вказати, який з TMSI відповідає шуканому телефону або за допомогою активних атак змусити телефони «виявити» свої реальні IMSI і MSISDN;

б) реалізувати алгоритм добору Kc для A5/1, що добре працює на реальних даних (з перешкодами/помилками, пропусками й т.п.);

в) розрахувати для нього «райдужні таблиці» (rainbow tables);

г) об'єднати всі ці пункти в закінчений працюючий застосунок.

Карстен і інші дослідники в основному вирішують пункт «в». Зокрема він його колеги пропонують використовувати Openbts, airdump і Wireshark для створення перехоплювача IMSI (IMSI catcher). Докладніше про пристрій і перехопленні з його допомогою дзвінків написане нижче в розділі «Атака людині-по-середині в GSM». Поки можна сказати, що цей пристрій емулює базову станцію й вбудовується між MS і справжньою базовою станцією.

Доповідачі затверджують, що SIM-карта легко може заборонити телефону показувати, що він працює в режимі шифрування A5/0 (тобто без шифрування взагалі) і що більшість SIM-карт в оберті саме такі. Це дійсно можливо. В GSM 02.07, написане (Normative Annex B.1.26), що SIM-карта містить спеціальний біт OFM у поле Administrative, який при значенні рівному одиниці, приведе до заборони індикації шифрування з'єднання (у вигляді комірного замочка). В GSM 11.11 зазначені наступні права доступу до цього поля: читання доступне завжди, а права на запис описані як «ADM». Конкретний набір прав, що регулюють запис у це поле, задається оператор на етапі створення SIM-карт. Таким чином, доповідачі сподіваються, що більша частина карт випускається із установленим битому й телефони в них дійсно не показують індикацію відсутності шифрування. Це дійсно суттєво полегшує роботу IMSI catcher-а тому що власник телефону не може виявити відсутність шифрування й щось запідозрити.

Цікава деталь. Дослідники зіштовхнулися з тим, що прошивання телефонів тестуються на відповідність специфікаціям GSM і не тестуються на обробку позаштатних ситуацій, тому, у випадку некоректної роботи базової станції (наприклад, «підставна» Openbts, яка використовувалася для перехоплення) телефони найчастіше зависають.

Найбільший резонанс викликала заява, що всього за $1500 можна з USRP, Openbts, Asterisk і airprobe зібрати готовий комплект для прослуховування розмов. Ця інформація широко розійшлася по Інтернеті, тільки автори цих новин і похідних від них статей забули згадати, що самі доповідачі деталей не надали, а демонстрація не відбулася.

У грудні 2010 року Карстен і Мунот (Sylvain Munaut) знову виступив на конференції 27C3 з доповіддю про перехоплення розмов в GSM мережах. Цього разу вони представили більш повний сценарій, але в ньому є присутнім безліч «тепличних» умов.

Для виявлення місця розташування вони використовують інтернет-сервіси, які дають можливість вкидати в мережу SS7 запити «send routing info». SSV – це мережа/стік протоколів, які використовуються для спілкування телефонних операторів (GSM і «наземних») один з одним і для спілкування компонентів мережі GSM один з одним.

Далі автори роблять посилання на реалізацію мобільного зв'язку в Німеччині. Там отримане в результаті запиту RAND добре корелює з кодом регіону (area code/zip code). Тому такі запити там дають можливість визначити з точністю до міста або навіть частини міста, де в Німеччині перебуває цей абонент. Але так робити оператор не зобов'язаний.

Тепер дослідники знають місто. Після цього вони беруть сніффер, їдуть у знайдене раніше місто й починають відвідувати всі його LAC. Приїхавши на територію, яка входить у якийсь LAC, вони посилають жертві SMS і слухають, чи іде пейджинг (paging) телефону жертви (це відбувається по незашифрованому каналу, у всіх базові відразу). Якщо виклик є, то вони одержують відомості про TMSI, який був виданий абонентові. Якщо немає – їдуть перевіряти наступний LAC.

Необхідно помітити, що тому що IMSI при пейджинзі не передається (і дослідники його не знають), а передається тільки TMSI (який вони й пра гнуть довідатися), те проводиться «атака за часом» (timing attack). Вони посилають

декілька SMS з паузами між ними, і дивляться, для яких TMSI проводиться пейджинг, повторюючи процедуру доти, поки в списку «підозрілих» TMSI не залишиться тільки один (або жодного).

Щоб жертва не помітила такого «промацування», посилає такий SMS, який не буде показаний абонентові. Це або спеціально створений flash SMS, або невірний (битий) SMS, який телефон обробить і вилучить, при цьому користувачеві нічого показане не буде.

З'ясувавши LAC, вони починають відвідувати весь стільник цього LAC, посилати SMS-ки й слухати відгуки на пейджинг. Якщо є відповідь, то жертва перебуває от у цьому стільнику, і можна починати зламувати її сесійний ключ (KC) і слухати її розмови.

Перед цим необхідно записати ефір. Тут дослідники пропонують наступне:

1) існують вироблені на замовлення FPGA-плати, які здатні одночасно записувати всі канали або uplink (канал зв'язку від абонента (телефону або модему) до базової станції стільникового оператора), або downlink (канал зв'язку від базової станції до абонента) частот GSM (890-915 і 935-960 Мгц відповідно). Як уже було відзначено, коштує таке устаткування 4050 тис. доларів, тому доступність такого устаткування для простого дослідника безпеки сумнівна;

2) можна брати менш потужне й більш дешеве устаткування й слухати частина частот на кожному з них. Такий варіант коштує приблизно 3,5 тис. євро з застосунком на базі USRP2;

3) можна спочатку зламати сесійний ключ, і потім декодувати трафік «на льоту» і іти за зміною частоти (frequency hopping) за допомогою чотирьох телефонів, у яких замість рідного прошивання коштує альтернативне прошивання Osmocombb. Ролі телефонів: 1-й телефон використовується для пейджинга й контролю відповідей, 2-й телефон виділений абонентові для розмови. При цьому кожний телефон повинен писати й приймання й передачу. Це дуже важливий пункт. До цього моменту Osmocombb фактично не працював і за рік (з 26C3 до

27C3) Osmocombb був дороблений до придатного до використання стану, тобто до кінця 2010 року не було практичного працюючого застосунку.

Злом сесійного ключа. Перебуваючи в одному стільнику з жертвою, вони посилають їй SMS, записують спілкування жертви з базової, і зламують ключ, користуючись тим, що під час установлення сесії (session setup) відбувається обмін безліччю напівпорожніх пакетів або з передбачуваним умістом. Для прискорення злому використовуються райдужні таблиці. На момент проведення 26C3 ці таблиці були не так добре заповнені й злом робився зовсім не за хвилини й навіть не за десятки хвилин (автори згадують годину). Тобто, до 27C3 навіть у Карстена (основного дослідника в цій області) не був застосунку, який дозволяв зламати КС за прийнятний час (у плині якого, швидше за все, не відбудеться зміна сессионого ключа (rekeying)).

Потім дослідники користуються тим, що зміна ключа рідко робиться після кожного дзвінка або SMS і сесійний ключ, який вони довідалися, не буде мінятися протягом якогось часу. Тепер, знаючи ключ, вони можуть декодувати зашифрований трафік до/від жертви в режимі реального часу, і робити зміну частоти (frequency hopping) одночасно з жертвою. Для захвата ефіру в цьому випадку реально досить чотирьох перепрошитих телефонів, тому що не потрібно писати всі частоти й усі таймслоти. Дослідники продемонстрували цю технологію в роботі. Правда «жертва» сиділа на місці й обслуговувалася однієї сотої.

Підводячи проміжний підсумок можна ствердно відповістити на запитання про можливість перехоплення й розшифруванню на льоту GSM розмов. При цьому треба мати пам'ятати наступне:

1. Технологія, описана вище не існує у вигляді, доступному для будь-якого бажаючого (у т.ч. script kiddies). Це навіть не конструктор, а заготовка для деталей конструктора, які треба дороблять до придатного до використання стану. Дослідники неодноразово зауважують, що в них немає чітких планів по викладанню в загальний доступ конкретики реалізації. Це означає, що на підставі

цих наробітків виробники Близького Сходу не виготовляють МАСово пристрою за 100 доларів, які можуть слухати все.

2. Osmocombb підтримує тільки одне сімейство чипів (хоча й найпоширеніше).

3. Спосіб визначення місця розташування по запитах до HLR і перебору LAC працює скоріше в теорії, чому на практиці. На практиці зловмисник або знає, де перебуває жертва фізично, або не може потрапити в той же стільник що й жертва. Якщо зловмисник не можете послухати той же стільник, у якій перебувати жертва, то спосіб не працює.

На відміну від демонстрації, у реальності в середній по навантаженню LA присутні тисячі пейджингових повідомлень. Більше того, пейджинг працює не в момент відправлення, а в певні часові вікна й пачками (по пейджинг-групам зі своїми чергами, номер якої є залишок від розподілу IMSI на кількість каналів, яка в кожної стільнику може бути своє), що знову ускладнює реалізацію.

4. Допустимо, LA знайдений. Тепер треба “намацати” відповідь абонента. Передавач телефону має потужність 1 -2 вата. Відповідно, просканувати його з відстані декількох десятків метрів теж є завданням (не простій). Виходить парадокс: LA накриває, наприклад, цілу область (місто). У ній, наприклад, 50 стільників, у деяких з яких радіус дії доходить до 30 км. Ми намагаємося піймати й розшифрувати на ненаправлену антену випромінювання. Для реалізації цього завдання в такому варіанті потрібно багато устаткування. Якщо ж виходити з передумови, при якій жертва перебувати в прямої видимості, тобто відстані, на якому перехоплення виглядає більш реалістичним, набагато ефективніше й простіше спрямований мікрофон. Треба відзначити, що на демонстрації дослідники перехоплюють свої телефони на відстані 2-х метрів.

5. Переміщення жертви між стільником також викликає проблеми, тому що вам також треба переміщатися разом з нею.

6. Телефони, використовувані в демонстрації, вимагають апаратної модифікації, у них потрібно забрати фільтр із антени, а якщо ні, то телефони

«чужий» uplink не «побачать». Фільтр у телефоні потрібний для того що б «слухати» не всі частоти, а тільки «свою».

7. Якщо в мережі регулярно відбувається зміна ключа (rekeying) або міняються TMSI (жоден з дослідників не враховував це), то це спосіб не працює взагалі або працює дуже погано (час розшифрування може виявитися більше чому час розмови).

8. Прослуховувати всю мережу не вийде, треба знати номер телефону.

Захист від перехоплювання трафіку

Як правильно зауважують автори, захиститися від цього конкретного способу досить легко:

1. Замість константного байта використовувати для пейджинга порожніх GSM-повідомлень випадкові значення.

2. Міняти КС після кожного дзвінка.

3. Міняти TMSI якнайчастіше.

Пункти 2 і 3 можна застосунати простою переконфігурацією елементів мережі провайдера й не вимагають відновлення прошивань або устаткування.

Крім цього, на ринку представлені різні модифіковані телефони, наприклад, крипто смарт телефон «Cancort», який забезпечує роботу на лініях зв'язку стандарту GSM 900/1800 у двох режимах:

Відкритий режим (звичайний режим GSM);

Режим шифрування з гарантованим від злому шифруванням інформації.

Cancort виконує наступні функції:

– шифрування/розшифрування голосової інформації;

– шифрування/розшифрування коротких повідомлень (послуга SMS);

– шифрування/розшифрування даних (послуга BS26 і GPRS);

– шифрування/розшифрування електронної пошти;

– шифрування/розшифрування інформації всіх телефонних директорій (SIM PB);

– шифрування/розшифрування інформації MMS.

Також для захисту можна використовувати скремблери, які добре зарекомендували себе при захисті звичайних телефонних мереж. Як приклад можна привести GUARD GSM. Даний пристрій (як і аналоги) з'єднується зі стільниковим телефоном по провідній гарнітурі й має невеликі розміри. Скремблер GUARD GSM має тридцять два режими скремблювання.

Принцип роботи даного скремблера заснований на первісному руйнуванні й часової перестановки звуку на передавальній стороні з його наступним відновленням на стороні, що ухвалює. Цей процес двосторонній. Часова перестановка відрізків мовного сигналу й відновлення їх послідовності на прийманні займають деякий інтервал часу. Тому обов'язковою властивістю такої апаратури є невелика затримка сигналу на прийомній стороні. Початок розмови, як правило, починається у відкритому режимі й далі по обопільній команді пристрою перемикаються в режим скремблювання. При веденні переговорів прилад виконує одночасно дві функції скремблювання й дескремблювання. Тобто вимовлена одним з абонентів мова шифрується з його боку, а другий скремблер, що перебуває в другого абонента розшифровує дану мову. І теж саме відбувається у зворотному напрямку, коли починає говорити другий абонент.

Технічні характеристики:

1. Розбірливість мови не менш 95%.
2. Тип з'єднання повний дуплекс.
3. Затримка сигналу в лінії не більш 100 мс.
4. Рівень захищеності лінійного сигналу часовий.
5. Використання в мережах стандарту GSM 900/1800.
6. Тип підключення до стільникового телефону провідна гарнітура 7. Габаритні розміри 80х45х16 мм

Атака “людині-по-середині” в GSM

Розглянута раніше атака активно використовувала пристрій за назвою IMSI-catcher. У цьому розділі розглядається принцип роботи подібного пристрою і його обмеження.

В Інтернет можна зустріти безліч пропозицій із продажу спеціальних пристроїв, які можуть емулювати базові станції. У подібних оголошеннях декларується, що такі емулятори дозволяють потай прослуховувати будь-які розмови, не ставлячи в популярність оператора й навіть не знаючи номера телефону, що прослуховується людину.

Пристрою з подібним функціоналом дійсно існують (наприклад, вироблений компанією Rohde & Schwarz комплекс RA 900), але вони мають далеко не настільки вражаючі можливості:

1. Потай можна тільки встановити, перебуває чи в зоні покриття телефон, у який вставлена SIM-карта із зазначеним IMSI, або одержати список IMSI/IMEI але не номерів телефонів у зоні покриття «псевдобазовою». Це має на увазі, що зловмисникові відомий IMSI.

2. Можна прослуховувати вихідні розмови з конкретного телефону, але в абонента при цьому буде відключене шифрування сигналу. Крім того, номер абонента, що дзвонить, буде змінений або схований. При цьому сам абонент може це виявити й установити факт прослуховування (або запідозрити).

3. При безпосередньому прослуховуванні вхідні дзвінки не можуть бути доставлені абонентові й, відповідно, не можуть бути прослухані. Для інших абонентів мережі абонент, що прослуховується, перебуває «поза зоною покриття».

Як видне, функціонал припускає наявності певних відомостей про жертву.

Принципи роботи IMSI-catcher

IMSI-catcher являє собою пристрій, який, з одного боку, поводиться як базова станція мережі GSM, а з іншого сторони містить у собі SIM-карту або якісь інші технічні засоби для з'єднання з комунікаційними мережами. Використовується воно в такий спосіб:

1. Пристрій розміщається недалеко від мобільного телефону жертви. Дальність визначається виходячи з рівня потужності реальної базової станції.

2. При роботі пристрій представляється звичайною станцією. Природно що вона повинна видавати себе за станцію того оператора, до якого належить жертва. У стандарті GSM не потрібно, щоб базова станція підтверджувала свою автентичність телефону (на відміну від мереж UMTS, наприклад), тому зробити це досить легко. Частота й потужність сигналу підробленої бази підбираються так, щоб реальні базові станції всіх сусідніх мереж не створювали їй перешкод у роботі.

3. Телефон жертви змушують вибрати підроблену базу в якості найкращої доступної базової станції через її гарний і потужний сигнал. Принцип вибору був описаний раніше. У результаті, зловмисник може визначити IMEI жертви.

4. Для прослуховування розмов при реєстрації підроблена база повідомляє телефон про необхідність переходу в режим шифрування A5/0, тобто без шифрування взагалі. Телефон по стандарту GSM не може відмовитися.

5. Після цього всі вихідні дзвінки жертви проходять через підроблену станцію у відкритому виді й можуть бути там записані/прослухані. Пристрій при цьому виступає в ролі проксі, самостійно з'єднуючись із набраним номером і прозоро транслюючи крізь себе голос в обидва боки.

Обмеження IMSI-catcher:

1. При підключенні до підробленої станції жертва стає недоступною для вхідних дзвінків. Для забезпечення підтримки вхідних дзвінків пристрій повинний обслуговуватися мережею оператора так само як інші базові станції. Для цього треба підключитися до якогось контролера базових станцій (BSC) і прописатися в його таблицях маршрутизації. Але якщо в зловмисника є доступ до мережі оператора на рівні, що дозволяє підключати й конфігурувати нові базові станції, то в цьому випадку ефективніше використовувати COPM. Якщо крім жертви в зону покриття пристрою потраплять інші мобільні телефони, розташовані поруч із жертвою, то вони будуть показувати наявність покриття, але ні вхідні, ні вихідні дзвінки обслуговуватися не будуть. Це може викликати підозри.

2. Більшість сучасних телефонів мають індикацію шифрування (у вигляді замочка) і жертва може насторожитися, якщо побачить, що з'єднання не шифрується.

3. Для трансляції вихідних дзвінків, пристрою потрібний вихід у телефонну мережу. Якщо для цього використовується власний GSM-модуль із SIM-картою, то вихідні дзвінки від підробленої станції будуть відбуватися з номером, відмінним від номера жертви. Для приховання цього можна використовувати послугу «приховання номера, що дзвонить» (calling line identification restriction, CLIR), що також може насторожити одержувачів дзвінка й вони можуть сповістити про це жертві. Як варіант, при використанні WiFi+VoIP можна підмінити номер підробленої станції на правильний, але це ускладнює конструкцію.

Для більш точної підміни необхідно щоб пристрій використовував SIM-карту того ж оператора, яким користується жертва, у цьому випадку в зловмисника буде можливість транслювати дзвінки жертви на службові й короткі номери.

4. Якщо жертва рухається, то може легко вийти із зони покриття пристрою, це приведе до того, що процес треба буде починати спочатку.

Перераховані недоліки показують, застосування подібного пристрою обмежується короткостроковим перехопленням розмов і практично не підходить для тривалого прослуховування.

Таким чином, основна користь від подібного пристрою може бути в тому, щоб ідентифікувати ШЗШМШ жертви, про яку точно відомо тільки її місце розташування, а потім уже використовувати відомості про Ш5І для проведення звичайного прослуховування засобами СОРМ.

Перехоплення повідомлень в ОБМ мережах можливий. Але, враховуючи умови, необхідні для реалізації перехоплення, можна сказати, що ОБМ захищений набагато краще, чим це показано у фільмах і Інтернеті.

2.2 Обґрунтування вибору засобів для побудови системи 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 декілька 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 додаткової 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 кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для протидії атаці на протокол SS7.

В процесі розробки кваліфікаційної бакалаврської роботи необхідно виконати наступний обсяг роботи:

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

На відміну від провідних ліній зв'язку мережам мобільному зв'язку властивий високий рівень канальних помилок. Основною причиною їх виникнення в бездротових ЛОМ є завмирання сигналу, викликане його багатопроменевим поширенням.

У стільникових мережах до високого рівня помилок приводять також атмосферні явища у вигляді дощу, грози й ін. В умовах постійного переміщення абонентів параметри потоку помилок постійно міняються й погано піддаються прогнозуванню.

Для передачі TCP/IP-трафіку канальний рівень повинен інкапсулювати IP-дейтаграмми (пакети мережного рівня) у свої кадри, у такий спосіб ізолюючи більш високі рівні від специфіки реалізації нижчележачих (канального й фізичного) рівнів. Однак звичайних можливостей канального рівня часто може виявитися недостатньо для успішного обміну даними. З обліком того, що більшість Інтернет-застосунків досить чутливі до виникаючих помилок, актуальних є вимога забезпечення додаткового захисту від них.

Логікою побудови стека TCP/IP передбачене виконання таких функцій, як керування потоком даних і виправлення помилок високорівневим транспортним протоколом. Якщо рівень канальних помилок високий, такий підхід приводить до різкого погіршення параметрів передачі користувацьких даних. У бездротових мережах реалізація додаткових функцій боротьби з помилками на канальному рівні виявляється більш ефективною. Для чутливих до помилок застосунків у стільникових мережах передбачений так званий непрозорий (nontransparent) режим передачі, який на відміну від прозорого (transparent) режиму передбачає додаткові процедури боротьби з помилками на канальному рівні. Непрозорий

режим, однак, не є панацеєю в боротьбі з помилками й втратами кадрів, оскільки кожному застосунку може знадобитися різний рівень безпомилковості передачі даних. Більше того, транспортний протокол зі своїм механізмом захисту від помилок часом несприятливо взаємодіє з аналогічними механізмами канального рівня, здійснюючи повторну передачу затриманих або загублених пакетів поряд з їхньою повторною передачею канальним протоколом. Це приводить до непродуктивних витрат пропускної здатності радіолінії.

Основні випадки й причини втрати кадрів канального рівня при їхній передачі по радіолініях наступні:

– Виникнення канальних помилок. Останні можуть бути й результатом погіршення ряду системних параметрів мобільному зв'язку, наприклад енергетики радіолінії.

– Вплив ненавмисних перешкод, які є випадковими, структурованими й іноді періодичними.

– Розрив фізичної лінії зв'язку. Така подія достатня типово для стільникових мереж під час процедури хендовера (handover) – переходу мобільного термінала з одного стільника в інший.

Існують і інші випадки втрати кадрів, що безпосередньо не є наслідком умов радіозв'язку. Це:

– Втрата переданих кадрів через колізії в мережах на основі MAC-протоколу змагального типу. Така ситуація типова для бездротових ЛОМ стандарту IEEE 802.11.

– Викликаюча втрату кадрів дія механізмів запобігання перевантаження (керування потоком) – наприклад, скидання пакетів у результаті переповнення буфера приймача.

– Кадри губляться й у результаті так званих помилок виконання, до яких ставляться апаратні помилки й помилки програмного забезпечення. На прийомній стороні вони сприймаються як втрата пакетів або кадрів.

Боротьба з помилками на канальному рівні здійснюється засобами механізму автоматичного запиту повторення (Automatic Repeat Request – ARQ), реалізованого в рамках відповідного протоколу. Робота цього механізму заснована на тому або іншому способі визначення факту втрати або викривлення переданого кадра і його повторній передачі. Факт викривлення кадра визначається шляхом його попереднього кодування завадостійким (як правило, циклічним) кодом і подальшого декодування на прийомній стороні в режимі виявлення помилок.

Застосовується цілий ряд варіантів механізму ARQ. Кожний правильно прийнятий кадр підтверджується окремим спеціальним кадром, або ознака підтвердження вставляється в керуюче поле інформаційних кадрів, переданих у зворотному напрямку. В останньому випадку також використовуються спеціальні кадри підтвердження, оскільки в потрібний момент інформаційного кадра може не виявитися. Існують два типи підтвердження про приймання – позитивне й негативне. Передавальна сторона, що не одержала позитивного підтвердження протягом заданого проміжку часу після передачі (тайм-ауту), удруге посилає відповідний кадр. Щоб виконувати повторну передачу сторона, що передає, зберігає кадри в накопичувачі до одержання підтвердження правильності передачі.

Існують три основні методи контролю й забезпечення доставки кадрів:

– Так званий стартостопний метод, або передача із зупинкою й очікуванням (Stop And Wait – SAW), часто його називають блоковим. У цьому випадку без підтвердження може бути переданий тільки один кадр. Після передачі кожного кадра передавальна сторона чекає підтвердження. Якщо надходить негативне підтвердження або відбувається перевищення тайм-ауту, кадр передається повторно. Він скидається (стирається) з буфера передавача лише після одержання позитивного підтвердження. Цей метод зручно використовувати при напівдуплексному зв'язку, коли сторони передають дані по черзі. Однак він неефективний у випадку організації дуплексному зв'язку,

особливо якщо час поширення сигналу по каналу значно перевершує час передачі кадра, що типово для високошвидкісних радіоканалів. Однак при невеликій довжині каналу або через низьку швидкість передачі метод SAW не приводить до помітного зниження продуктивності радіоканалу.

– Потоковий метод передачі, або метод з поверненням на N кадрів (Go Back N – GBN). Метод GBN часто називають ARQ типу REJ – за назвою службових кадрів Reject, що переносять підтвердження від приймача до передавача. У випадку застосування методу GBN кадри передаються безупинно без очікування підтвердження приймання кожного з них. При одержанні негативного підтвердження або після закінчення встановленого часу очікування непідтверджений кадр і всі наступні кадри передаються повторно.

– Метод вибіркового, або селективного, повтору (Selective Repeat – SR). Його ще йменують SREJ – так називаються службові кадри Selective Reject, що переносять підтвердження про селективне неприймання від приймача до передавача. Згідно із цим методом, повторно передається тільки той кадр, на який зробило негативне підтвердження (або минув час очікування підтвердження). У порівнянні з методами SAW і GBN метод SR суттєво підвищує ефективність роботи каналу. Але для передачі й приймання кадрів не один по одному їх номерів на прийомній стороні повинен перебувати буферний накопичувач із довільним доступом. Зі збільшенням затримки поширення сигналу в каналі зв'язку необхідно збільшувати буферну пам'ять. Реалізація методу SR є більш складної й дорогою. Тому він довго не міг знайти широкого застосування.

Крім перерахованих методів, у мережах мобільного зв'язку можуть знайти широке застосування так звані гібридні методи ARQ (Hybrid ARQ). Їхньою особливістю є використання завадостійкого коду в режимі виправлення помилок. Виправлення перекручених кадрів на прийомній стороні може суттєво скоротити число повторних передач на канальному рівні. Прикладом реалізації гібридного ARQ можуть служити схеми кодування CS-1, CS-2, CS-3 і CS-4 у радіоінтерфейсі

GPRS мереж GSM, що передбачають кодування згорточним кодом зі швидкостями 1/2, 2/3, 3/4 і 1 відповідно.

Конкретний варіант реалізації методу повторного пересилання кадрів впливає на значення затримки їх передачі. У кожному разі застосування ARQ приводить до того, що перераховані вище складові 2, 3, 4, 5 і 6 підсумкової затримки можуть бути повторено кілька раз.

Особливості мереж стільникового зв'язку

Сучасні цифрові мережі стільникового зв'язку покоління характеризуються відносно невисокими значеннями швидкості передачі даних, малими розмірами кадрів і перевагою режиму комутації каналів. У них реалізований часовий TDMA (GSM і D-AMPS) або кодовий множинний доступ CDMA (IS-95). Кадри протоколу канального рівня цих технологій переносять або кодовану мову, або дані високорівневих застосунків. У результаті обмеженої енергетики радіоканалу й завмирання сигналу, затінення від природніх і штучних перешкод рівень втрати кадрів (Frame Error Rate – FER) досягає 1-2%. Для мовного трафіку такий рівень уважається припустимим, оскільки при цьому не відбувається істотного погіршення якості мовному зв'язку, але він неприйнятний у випадку передачі трафіку даних.

Стільникові мережі підключаються до інших мереж за допомогою так званої міжмережевої служби IWF (Interworking Function), яка забезпечує цифроаналогове перетворення при з'єднанні із ТфОП або виконує функції адаптації швидкості передачі й перетворення формату кадрів при підключенні до цифрових мереж (наприклад, ISDN або Інтернет). Підтримка спільної роботи з пакетними мережами передачі даних у службі IWF забезпечується за допомогою протоколу RLP (Radio Link Protocol), який використовується для зв'язку з мобільними станціями. Цей протокол підтримує можливість сегментації й складання IP-дейтаграмм, забезпечуючи прозорий зв'язок з Інтернет, і виправлення помилок у радіоканалі, крім у такий спосіб їх вплив на Інтернет-трафік.

Мережі GSM надають абонентам можливість передавати дані зі швидкістю до 9,6 кбіт/с. Протокол RLP, працюючи в непрозорому режимі, використовує кадри довжиною 240 біт і реалізує метод SR. У результаті ймовірність викривлення біта знижується з 10 -3 до 10-8, але це відбувається за рахунок внесення додаткової затримки передачі й збільшення її дисперсії.

У мережах D-AMPS (IS-136) максимальна швидкість передачі даних теж рівна 9,6 кбіт/с. У них протокол RLP реалізує метод GBN і працює з кадрами довжиною 256 біт у непрозорому режимі. Кожний переданий кадр може підтверджувати одержання безлічі послідовних кадрів.

В CDMA-мережах (IS-95) забезпечується швидкість передачі даних до 8,6 кбіт/с, а протокол RLP працює в непрозорому режимі з кадрами довжиною 172 біт.

При передачі Інтернет-трафіку пакети мережного рівня (IP-пакети) спочатку інкапсулюються в кадри змінної довжини протоколу PPP (Point-to-Point Protocol), а потім сегментуються в RLP-кадри фіксованого розміру. Така комбінація перетворень зручна для забезпечення передачі IP-пакетів змінної довжини й ефективного захисту від помилок на основі використання RLP-кадрів фіксованого розміру. З метою зменшення службового заголовка кадрів RLP реалізована передача тільки негативних підтверджень. Кадри, не прийняті після декількох повторних спроб передачі, відкидаються. Таким чином, відбувається деяке зниження надійності передачі в ім'я обмеження дисперсії затримки. Результуюча ймовірність втрати кадра забезпечується на рівні 10-4.

Особливості бездротових ЛОМ

У цей час в області бездротових ЛОМ найбільше поширення одержали мережі стандартів 802.11 і 802.11b. Перший з них був розроблено для діапазону 2,4 ГГц. У ньому передбачалися максимальні швидкості передачі даних, рівні 1 і 2 Мбіт/с, і розширення спектра сигналів по методу прямої послідовності (Direct Sequence Spread Spectrum – DSSS) або шляхом псевдовипадкової перебудови частоти. Пізніше були створені стандарти 802.11a і 802.11b. Стандарт 802.11a

забезпечує передачу даних на швидкостях до 54 Мбіт/с у діапазоні частот 5 ГГц із застосуванням на фізичному рівні методу модуляції OFDM (Orthogonal Frequency-Division Multiplex). Стандарт 802.11b орієнтований на збільшення швидкості передачі при роботі в тому ж самому (що й у стандарті 802.11) діапазоні частот 2,4 ГГц шляхом використання методу модуляції CCK (Complementary Code Keying) разом з DSSS. Устаткування стандарту 802.11b підтримує максимальні швидкості передачі, рівні 5,5 і 11 Мбіт/с. У цей час розробляється технологія Hiperlan/2 і ряд інших технологій бездротових ЛОМ.

Характерна для протоколів бездротових ЛОМ інформаційна й процедурна надмірність суттєво обмежує швидкість передачі користувацького трафіку навіть у випадку застосування високошвидкісних технологій. У таблиці представлені результати виміру швидкості передачі на IP-рівні для мережі стандарту 802.11b при довжині максимального блоку передачі (Maximum Transfer Unit – MTU), рівної 1500 байт. Наведені значення можуть зчитатися верхньою оцінкою досяжної швидкості передачі IP-трафіку, тому що отримані в умовах відсутності канальних помилок. Вплив канальних помилок і деяких особливостей функціонування транспортних протоколів приводять до ще більшого погіршення швидкісних і інших якісних показників доставки користувацького трафіку.

Характеристики передачі TCP-трафіку по стільникових мережах зв'язку

Протокол TCP є самим популярним транспортним протоколом Інтернет. Він гарантує безпомилкову доставку даних від одного хоста до іншого. Крім того, TCP виконує прозору сегментацію й складання користувацьких даних, а також керування потоком і запобігання перевантаження. Протокол TCP “розглядає” усі, що мають місце втрати як результат перевантажень у мережі. Однак це в корені не вірно відносно бездротових мереж, у яких переважну роль відіграють канальні помилки.

Характеристики TCP-трафіку декілька різняться при прозорій і непрозорій передачі на канальному рівні стільникової мережі. У прозорому режимі роботи

припустимий рівень втрат кадрів приводить до наступного результату. Для передачі по повношвидкісному каналу в мережі CDMA (IS-95) 1400-байт IP-дейтаграмма сегментується на 68 кадрів. Якщо виходити з того, що викривлення кадрів має незалежний характер, імовірність успішної передачі пакета складе всього лише 50,49% при FER, рівному 1%. Зменшенням розміру IP-дейтаграмм можна добитися зниження ймовірності їх викривлення, однак при цьому через незмінність розміру їх заголовків росте відносна частка службової інформації (overhead). Зменшити розмір IP-заголовків до 3-5 байт цілком реально, використовуючи відомі алгоритми їх компресії. Однак це несумісне із процедурами шифрування мережного рівня й може привести до нових помилок (які виявляє протокол TCP) і перевстановленню TCP-з'єднання із втратою переданих у ковзному вікні даних. З одного боку, кодування даних і перемежування бітів у стільникових мережах дозволяють зменшити рівень помилок і ефект їх групування, але, з іншого боку – збільшують затримку обробки (на час виконання цих операцій).

У непрозорому режимі основні помилки усуваються протоколом RLP ще до моменту перевищення значення таймерів затримок. Виміру, проведені в типовий GSM-мережі великого міста показали, що 95% значень часу обігу TCP-сегмента (Round Trip Time – RTT) перебувають у районі 600 мс із дисперсією близько 20 мс. Однак іноді значення RTT може бути значно більше й досягати 6-12 с. Такі випадки рідкі й відбуваються в основному через те, що протокол RLP не завжди забезпечує абсолютно надійну передачу даних. Після декількох невдалих спроб повторної передачі (за замовчуванням їх число рівне шести) протокол RLP відмовляється від подальших передач перекрученого кадра, після чого передавач і приймач переустановлюють свої лічильники послідовних номерів і очищають буфери приймання й передачі. Для зменшення числа переустановок параметрів на етапі встановлення з'єднання можна побільшати максимальне число повторних передач протоколу RLP. Збільшенням розміру

MTU можна добитися відносного скорочення накладних витрат на службові заголовки, але при цьому збільшиться й час відповіді.

Пропускна здатність з'єднання збільшується також шляхом оптимізації розміру кадрів RLP. Хоча малі розміри кадрів спрощують роботу протоколу RLP і роблять його більш надійним при роботі на поганих каналах, вибір оптимального розміру кадра для конкретних умов здатний привести до помітного збільшення швидкості передачі.

Якщо з'єднання містить у собі кілька бездротових ділянок, відбувається акумуляція втрат, що сприяє ще більш значному зменшенню ефективності передачі. Це також приводить до непродуктивного використання бездротових ліній стільникової мережі в режимі комутації каналів. У випадку викривлення TCP-пакета після проходження ряду бездротових ділянок він повторно передається зі своєї початкової точки по всіх ділянках TCP-з'єднання, при цьому результуюча пропускна здатність з'єднання сильно знижується. Втрати мають більш серйозні наслідки у випадку значних міжкінцевих затримок передачі, що вимагають від протоколу TCP установлення великого розміру вікна підтвердження. У протяжних з'єднаннях протокол TCP установлює більші значення тайм-аутів очікування підтверджень із боку приймача. Крім того, стільникові мережі самі розривають з'єднання в процесі процедури хендовера, вносячи, таким чином, додаткові випадкові затримки. Інша проблема полягає в переупорядкуванні пакетів у процесі їх передачі по стільникових мережах із протоколом RLP.

Шляхи підвищення ефективності передачі даних у мобільних мережах

На цей момент розроблене й досліджується велика кількість методів поліпшення характеристик передачі користувацького трафіку в мережах мобільного зв'язку. Усі їх в основному можна розділити на дві більші групи застосунків – транспортного або канального рівня.

Застосунки транспортного рівня

Погіршення ефективності роботи протоколу TCP у мережах мобільного зв'язку є в основному наслідком того, що, як ми вже відзначали, протокол помилково “уважає” усі втрати обумовленими перевантаженням мережі. Більшість запропонованих способів поліпшення протоколу TCP спрямовані саме на боротьбу із цим його “оманою”.

Уведення й використання ознак мобільності. Під час хендовера передані пакети можуть затриматися або навіть згубитися. Процес їх відновлення повинен починатися відразу після закінчення процедури хендовера, до того моменту, коли почнуть спрацьовувати більші таймери TCP. Протокол TCP може “довідатися” про факт х ендовера шляхом одержання спеціальних сигналів від нижчележачих рівнів. Роль таких сигналів виконують так звані ознаки (hints) мобільності, які використовуються для визначення факту втрати даних у результаті хендовера. Для боротьби з такими втрата ми на рівні TCP також можна зменшувати поріг повільного старту, скорочуючи в такий спосіб часові втрати протягом фази боротьби з перевантаженням. Інший варіант використання додаткової інформації, отриманої на основі введених ознак, придатний для кінцевих точок бездротових каналів і полягає в ініціюванні зупинки передачі TCP-трафіку шляхом прозорого закриття встановленого вікна приймання. У цьому випадку передавач “заморожує” усі таймери й починає періодичну апробацію вікна приймання. Скорочення розміру встановленого вікна, однак, порушує концептуальні принципи, на яких побудований протокол TCP.

Поділ TCP-з'єднання на кілька складених частин. Після процедури хендовера механізм боротьби з перевантаженням намагається знову визначити пропускну здатність нового каналу. Тому що повторна передача між кінцевими точками транспортного з'єднання є занадто повільної, TCP-з'єднання можна розділити на кілька послідовних з'єднань. При цьому в якості точок поділу повинні використовуватися маршрутизатори, що перебувають на стику провідних і бездротових сегментів мережі. У результаті вихідне TCP-з'єднання

декомпозується на окремі провідні й бездротові з'єднання. Схема поділу вихідного з'єднання порушує семантику TCP-протоколу, тому що в цьому випадку підтвердження можуть досягтися передавач ще до моменту одержання відповідної до порції даних приймачем. Для збереження вихідної семантики потрібно ввести застосункову затримку для підтверджень, що, у свою чергу, знизить швидкість з'єднання. Уведення точок поділу приведе до істотного збільшення надмірності як через застосункову обробку переданих пакетів на рівні TCP-протоколу, так і через необхідність уведення додаткової службової інформації для організації нестандартного транспортного з'єднання.

Використання часових міток. Модифікація Ейфеля дозволяє позбутися випадкових тайм-аутів і швидких повторних передач TCP-протоколу при хендовері й затримках, ініційованих повторною передачею на канальному рівні. Оскільки ці проблеми з'явилися наслідком нездатності протоколу TCP розрізняти підтвердження, обумовлені вихідною передачею пакетів і повторними передачами, у рамках модифікації протоколу запропоновано додавати часові мітки в, що відсилаються TCP-пакети. Часові мітки повторюються в пакетах-підтвердженнях, дозволяючи, таким чином, легко знаходити причину випадкових тайм-аутів без порушення семантики TCP.

Варіанти модернізації протоколу TCP можуть бути привабливими лише в тому випадку, якщо зажадають модифікації тільки кінцевих вузлів транспортного з'єднання. На практиці часто потрібні додаткові зміни, такі, як забезпечення сигналізації про факти хендовера від протоколів нижніх рівнів, інсталяція додаткового програмного забезпечення в точках поділу, розробка альтернативних TCP-сумісних прикладних протоколів.

Застосунки канального рівня

Замість модифікації властиво протоколу TCP застосунку канального рівня орієнтовані на те, щоб сховати від нього втрати, що відбуваються на нижчележачих рівнях.

Використання непрозорого режиму RLP-протоколу стільникових мереж. Застосування протоколу RLP у стільникових мережах дозволяє уникнути порушення рівневой логіки роботи системи зв'язку, характерної для деяких інших способів підвищення ефективності передачі, але в той же час може привести до повторних пересилань даних і на канальному й на транспортному рівні. Такі ситуації виникають рідко, однак зовсім їх не враховувати не можна.

Локальне виправлення помилок на IP-рівні. Такий спосіб, відомий за назвою Snoop TCP, є скоріше застосунком канально-мережного рівня, чому строго канального. Snoop-Агент установлюється в точці поділу TCP-з'єднання й відслідковує минаючі через нього TCP-дані й підтвердження. Він також кешує непідтверджені TCP-пакети, виявляє їхню втрату шляхом аналізу повторно переданих підтверджень і стану локальних таймерів. На основі отриманій у такий спосіб інформації агент організує прозору повторну передачу загублених даних. Реалізація такого підходу приховує повторні передачі підтверджень, індицируючі втрати даних у бездротовому сегменті TCP-з'єднання, запобігаючи в такий спосіб зайві спроби відновлення даних на транспортному рівні. Snoop-Агент використовує наявну в TCP-пакетах інформацію для того, щоб уникнути введення додаткової надмірності в заголовки кадрів канального рівня. Даний застосунок являє собою схему з поділом вихідного TCP-з'єднання але без порушення його семантики й дозволяє також уникнути конфлікту повторних передач на канальному й транспортному рівнях шляхом придушення повторних TCP-підтверджень.

При використанні описаного способу потрібно, щоб TCP-приймач перебував відразу за точкою поділу. Якщо бездротової хост передає дані вилученому приймачу, то підтвердження протоколу TCP можуть зробити від нього занадто пізно для здійснення ефективної боротьби із втратами, і в цьому випадку не виключений ріст втрат, обумовлених перевантаженням. Для ідентифікації втрат, що виникають через перевантаження мережі й канальних помилок, запропоноване явно повідомляти про втрати (Explicit Loss Notification –

ELN). Якщо snoop-агент визначає втрати, не обумовлені перевантаженнями, він установлює біт ELN у заголовку TCP-пакета й відправляє його далі, до приймача. TCP-приймач повертає встановлений біт назад передавачу. Snoop-Агент може також використовувати інформацію про розміри черг для розмежування причин наявних втрат даних. При одержанні Eln-повідомлення TCP-передавач повторно передає загублений пакет, не задіючі при цьому свої механізми боротьби з перевантаженнями. Хоча застосування Eln-повідомлень можливо для більшості мереж, такий підхід вимагає зміни алгоритму роботи маршрутизаторів. Потрібно також ураховувати, що повторну передачу не можна здійснити раніше витікання RTT, тому що для цього потрібно одержати підтвердження із установленим битому ELN.

Застосунку, про яких мова йде, функціонально обмежені рівнем реалізації, але характеризуються низьким значенням кругової затримки, що дозволяє здійснювати значно більш швидке відновлення даних на відміну від варіантів з модифікацією протоколу TCP. Обмеженість застосунків канального рівня полягає в тому, що вони намагаються відновлювати загублені дані в рамках одного-двох рівнів, "не знаючи" про вимоги до передачі конкретних даних, що може виявитися неприйнятним при роботі з багатьма високорівневими протоколами й застосунками.

Уведення проміжного рівня адаптації. Для підвищення ефективності роботи транспортних протоколів шляхом приховання від них особливостей конкретної реалізації бездротової мережі в рівневу архітектуру бездротового хоста (між IP-протоколом і нижчележачими протоколами бездротової мережної інфраструктури) можна ввести проміжний рівень адаптації бездротовому зв'язку (Wireless Adaptation Layer – WAL). Вхідний до складу WAL-рівня координатор організує взаємозалежну роботу всіх модулів рівня, здійснює класифікацію оброблюваних потоків даних, генерує WAL-заголовок нового пакета й віддає його транслятору (Logical Link Control Translator – LLCT) для подальшої передачі на канальний рівень бездротової мережі.

Гнучкість WAL-рівня обумовлена можливістю включення до його складу великого числа різноманітних цільових модулів. Прикладами завдань, виконуваних окремими модулями, можуть бути:

– кодування з виправленням помилок (Forward Error Correction – FEC);

– автоматичний запит повторення (ARQ);

– сегментація й складання пакетів;

– керування трафіком з метою забезпечення необхідного рівня QoS;

– стиск заголовків і властиво даних.

Модулі можуть виконувати й інші функції.

Розглянемо приклад використання Fec-модуля на WAL-рівні устаткування бездротової ЛОМ. Вибір конкретного завадостійкого коду і його початкових параметрів може здійснюватися на етапі встановлення зв’язку між мобільною станцією й точкою доступу й надалі коректуватися залежно від мінливих умов зв'язку. На мал. 3 представлена послідовність обробки IP-пакета на WAL-рівні за допомогою Fec-модуля, який додає до вихідної IP-дейтаграммі свій заголовок, WAL-координатор – свій, а драйвер бездротової мережі остаточно формує переданий пакет.

Реалізація Fec-модуля можлива також і на фізичному рівні в складі протоколу радіоінтерфейса, що вже підтримує застосування завадостійкого кодування в режимі з виправленням помилок. Уведення такої схеми адаптації кодування в радіоінтерфейс GPRS (CS-1, CS-2, CS-3 і CS-4) може підвищити його ефективність від 10 до 80% для різного рівня помилок у каналі (BER).

Сучасні реалізації рівневої архітектури бездротових хостів далеко не завжди забезпечують ефективну передачу даних в умовах мобільного зв'язку. У роботі проаналізовані причини зниження ефективності роботи протоколу TCP при передачі даних через стільникові й локальні радіомережі, а також розглянуті основні застосунки транспортного й канального рівнів, що дозволяють у значній мірі справлятися з виникаючими проблемами.

3.2 Розробка структурної схеми

SS7

Протокол SS7, також відомий як Сигналізаційна система № 7, відноситься до мережі передачі даних і до ряду технічних протоколів або правил, які регулюють обмін даними ними. Він був розроблений в 1970-х роках для відстеження й підключення викликів у різних мережах операторів зв’язку, але тепер він звичайно використовується для розрахунків біллінгу стільниковому зв'язка й відправлення текстових повідомлень на додаток до маршрутизації мобільних і стаціонарних викликів між операторами й регіональними комутаційними центрами.

Як не дивно, але стало відомо про наявність цієї уразливості ще в далекі часи. Ще всім відомий Стив Джобс у своєму гаражі здійснив перші атаки, зробивши, так званий, безкоштовний стільниковий зв'язок, використовуючи уразливості. Відтоді частина недоліків була закрита, але по колишньому SS7 піддана вдалим атакам.

Тільки в грудні 2014 року телекомунікаційні компанії почали розглядати інструменти припинення атак SS7. Саме тоді Карстен Нол з Берлінських дослідницьких лабораторій по безпеці й незалежний дослідник по імені Тобиас Енгель виступили з повідомленнями про SS7 на Конгресі зв'язку Хаосу в Німеччині. Енгель продемонстрував SS7-метод для відстеження телефонів в 2008 році, але цей метод не був настільки «кричущим», як ті, які він і Нол описали в 2014 році. Останнє спонукало регулювальні органі в Північній Європі зажадати, щоб оператори почали застосовувати заходи щодо запобігання атак SS7 до кінця 2015 року.

Алгоритм атаки на SS7:

1. Зловмисник підключається до сигнальної мережі SS7 і відправляє службову команду Send Routing Info для SM (SRI4SM) у мережний канал, указуючи номер телефону, що атакується абонента як параметра. Домашня

абонентська мережа відправляє у відповідь наступну технічну інформацію: IMSI (International Mobile Subscriber Identity) і адресу MSC, по якому в цей час надає послуги передплатникові.

2. Після цього зловмисник змінює адресу біллінгової системи в профілі передплатника на адресу своєї власної псевдобіллінгової системи (наприклад, повідомляє, що абонент прилетів на відпочинок і в роумінзі зареєструвався на новій біллінговій системі). Як відомо, ніяку перевірку така процедура не проходить. Далі атакуючий уводить оновлений профіль у базу даних VLR через повідомлення «Insert Subscriber Data» (ISD).

3. Коли абонент, який атакується, робить вихідний дзвінок, його комутатор звертається до системи зловмисника замість фактичної біллінгової системи. Система зловмисника відправляє комутатору команду, що дозволяє перенаправляти виклик третій стороні, контрольованій зловмисником.

4. У сторонньому місці встановлюється конференц-зв'язок із трьома передплатниками, два з них є реальними (викликаючий абонент А і викликуваний В), а третій уводиться зловмисником незаконно й здатний прослуховувати й записувати розмову.

5. Відповідним чином одержуємо й SMS, того хто атакується. Маючи доступ до псевдобіллінгової системи, на яку вже зареєструвався наш абонент, можна одержати будь-яку інформацію, яка приходить або йде з його телефону.

Як одержати нелегальний доступ до SS7 мережі?

Як ми побачили, маючи доступ до SS7 мережі, зробити атаку не складається праці. Як же одержати нелегальний доступ до мережі?

Доступ продають у даркнеті, а при бажанні можна знайти й безкоштовно. Така доступність обумовлена тим, що в мало розвинених державах одержати статус оператора дуже просто, відповідно й одержати доступ до SS7 хабам. Так само присутні несумлінні працівники в операторів.

Рисунок 3.1 – Структурна схема системи

Що вживають провайдери й інтернет ресурси з SMS верифікацією?

У цей момент оператори повільно закривають численні діри в SS7, незважаючи на те, що деякі навіть не визнають їхню наявність. Хтось

обвинувачує в розпаленні паніки навколо цієї проблеми, хтось утримується від коментарів, хтось говорить обтічні фрази, типу: «Безпека наших клієнтів коштує в нас у пріоритеті». Проте, на сьогоднішній день цьому питанню не приділяється винної уваги з боку операторів, а доступність експлуатації цієї уразливості, як ніколи велика.

Як захиститися від перехоплення SMS і прослуховування дзвінків?

Є й гарні новини. Представники інтернет послуг, які насправді турбують про своїх клієнтів, уже або мають, або спішно готовлять альтернативи SMS верифікації.

Компанія Apple готовить новий механізм двофакторної авторизації для захисту Apple ID. Нагадаємо, що заволодівши обліковими даними Apple ID можна ні багато ні мало заблокувати й повністю стерти всі ваші Apple пристрої віддаленно. На жаль, на даний момент пароль можна відновити через SMS, яке у свою чергу, як ви довідалися, можна перехопити.

Хотілося б виділити важлива рада, яка рідко можна знайти на просторах інтернету. Він дуже простий, але діючий.

Враховуючи, що псевдобіллінгові системи зловмисників у більшості випадків представляються іноземними операторами (повідомляють, що ви в міжнародному роумінзі), те досить просто відключити на телефоні можливість міжнародного роумінгу. Це можна зробити у вашого оператора стільникового зв'язку безкоштовно. Якщо вам немає гострої необхідності саме із цього телефону спілкуватися в роумінзі, то цей застосунок сильний убезпечить вас від тих неприємностей, які були описані в цій роботі.

Як же складаються подібні ситуації? Протокол SS7 був створений багато років тому. Його безпеки просто не приділяли винну увагу протягом усього цього часу. У теж час у цей момент лідери багатьох розвинених країн наголошують на «кібер озброєння». Розвиток кібершпіонажа розвивається величезними темпами. Деякі з «кібер інструментів» попадають у руки зловмисників і тоді страждають звичайні люди.

Потрібно не забувати про це й підтримувати інструменти масового використання на актуальному рівні безпеки.

Ви будете шоковані, але навіть у таких критично важливих місцях, як авіація, дотепер використовується небезпечні способи спілкування між літаками й наземними станціями. Будь-який хакер середнього рівня може фальсифікувати сигнали від диспетчера або борту літака.

3.3 Розробка функціональної схеми

Мережі п'ятого покоління будуть одночасно й схожі на будь-яке попереднє покоління мобільних мереж, і при цьому помітно відрізнятися від них – і цьому є цілий ряд пояснень, які стають очевидніше, якщо задуматися про те, яким образом ці зміни впливають на принципи забезпечення безпеки користувачів і устаткування в екосистемі мереж п'ятого покоління.

5G будується навколо наступних 8 вимог:

– Швидкість передачі даних до 10 Гб/с, що перевершує швидкості мереж 4G і 4.5G в 10-100 раз.

– Затримки на рівні 1 мілісекунди.

– В 1000 раз більше пропускної здатності на одиницю площі.

– До 100 раз більше підключених пристроїв на одиницю площі (у порівнянні з 4G LTE).

– Доступність сервісу 99.999%.

– Покриття 100%.

– Зниження енергоспоживання мережних пристроїв на 90%.

– До 10 років роботи від батареї для пристроїв IoT з низьким енергоспоживанням.

На відміну від існуючих застосунків, які змушено жертвувати продуктивністю при використанні поточних бездротових технологій (3G, 4G, WiFi, Bluetooth, Zigbee і т.д.), мережі 5G створюються для реального масового

впровадження «інтернету речей» і інших вимогливих до швидкості мережі й доступності сервісів.

У даній роботі розглянуто 5 основних моментів у забезпеченні безпеки, у відношенні яких операторам мобільних мереж слід переглянути свій підхід у найближчі місяці й роки, напередодні комерційного запуску мобільних мереж 5G:

1. Мережі 5G мають більшу поверхню атаки по порівнянню з нинішніми мережами стільникового зв'язку. Ми поступово йдемо від тієї моделі, коли оператори мобільних мереж одержували від перевірених постачальників відразу комплекс апаратного й програмного забезпечення, і попадаємо в новий світ віртуалізованої інфраструктури, де використовується цілий «зоопарк» програмного забезпечення, побудованого як на базі відкритих вихідних кодів, так і на базі закритих технологій. А це означає, що потенційна «поверхня атаки» у мобільних мережах нового покоління буде мати набагато більше подібності з такий у класичному підприємстві, оскільки стандартні віртуалізовані технології більш доступні й краще відомі зловмисникам, чому пропрієтарні мережні технології, які характерні для нинішніх мереж стільникового зв'язку.

2. Розвиток мобільної периферії приводить до змін у периметрі безпеки. Мережі п'ятого покоління забезпечують набагато більший ступінь волі у використанні «периферійних» ресурсів, які дозволяють зняти частина навантаження з «ядра» мережі – і це дійсно важливо з обліком того, що в стільникових мережах з'являється усе більше застосунків, що припускають мінімальну мережну затримку, починаючи з ігор з високим дозволом і закінчуючи такими критично важливими застосунками як безпілотні автомобілі, де на коні коштують людські життя. Так, мережі 5G дозволяють кешувати контент локально – і якщо вам захочеться подивитися фільм по запиту, застосунок може забирати потік з локального кешу, а не передавати його по ядру мережі прямо с серверів контент-провайдера… що у свою чергу вимагає зміни підходів до забезпечення безпеки даних і стільникових комунікацій.

3. Поява нових можливостей для створення додаткових рівнів безпеки в каналах стільникового зв'язку. У цей час основним завданням при забезпеченні безпеки в стільниковому зв'язку є захист від прослуховування. Однак у майбутньому світі смартфонів і інтернету речей, в оточеннях з великою кількістю механізмів, імовірність прослуховування, як видно, відійде на другий план. Замість цього нам доведеться задуматися про такі речі як атаки з маніпуляцією даними, які, приміром, можуть використовуватися для того, щоб віддавати механізмам команди на виконання певних дій (наприклад, відкрити двері або перехопити керування безпілотним автомобілем). В операторів мобільних мереж, як і у виробників побутової електроніки, з'явиться можливість пропонувати «безпека у вигляді сервісу», у результаті чого постачальники застосунків зможуть при передачі окремих видів даних застосовувати додаткові рівні безпеки поверх уже існуючих захищених каналів мережі стільниковому зв'язку.

4. Застосування різних механізмів безпеки залежно від типу даних. Сьогодні відповідальність щодо шифрування й захисту даних лежить на постачальнику застосунків. У мережах 5G певна роль у цьому процесі буде приділятися й операторам мереж – особливо з появою IoT пристроїв, що не володіють достатньою обчислювальною потужністю для достатнього шифрування комунікацій. Постачальники безпеки у вигляді сервісу можуть запропонувати три різні опції захисту на додаток до того, що мережа забезпечує за замовчуванням:

– передачу трафіку без шифрування (для даних, що не мають великої цінності, які, у випадку їх перехоплення, будуть фактично пошук для зловмисників, наприклад, дані ping-запитів);

– середній рівень захисту (як варіант – для датчиків IoT, дані з яких можна використовувати для атак з підтасуванням цих даних, наприклад, коли зловмисник може спотворити такі дані, показуючи рівень води в заплаві нижче реального значення й у такий спосіб відключити спрацьовування захисту);

– високий рівень безпеки (де потрібно більш висока конфіденційність і захист персональної інформації, які актуальні для високоціних даних, наприклад, при передачі даних кредитних карт у торговельних транзакціях).

Усе це дозволить розроблювачам застосунків не обмежуватися одними лише платформами смартфонів, і при необхідності вони зможуть реалізувати додатковий рівень безпеки.

5. Забезпечення безпеки в умовах росту M2M комунікацій. Очікуваний ріст числа пристроїв, які будуть підключені до мережі в найближче десятиліття, вражає уяву. У цих умовах велике значення має забезпечення конфіденційності й цілісності даних, переданих між пристроями, оскільки подібні оточення допускають нові види уразливостей. Недавно ми вже бачили успішні атаки з маніпуляцією даними, у результаті яких дослідникам Keen Security Lab у лабораторних умовах удалося перехопити керування безпілотним автомобілем Tesla. Слід задуматися про те, як захистити такі дані від подібних атак у всіх фрагментах ланцюга, як у мережі стільникового зв'язку, так і за її межами. Навіть при наявності захищених каналів зв’язку вам може знадобитися забезпечити шифрування переданих через публічну IP мережа даних, коли ці дані залишають мобільну мережу й ідуть за межі операторського шлюзу.

Перш ніж ми побачимо комерційний запуск перших мереж п'ятого покоління, нам ще має бути багато чого визначити в області технологій 5G, і має бути чимало обговорень проблем безпеки. Починаючи із цього моменту, замість Масштабних і дорогих модернізацій на рівні ядра мережі для одержання мереж 6G і більш нових технологій, імовірно, ми будемо мати справа із програмно-обумовленим оточенням, у якому застосовні дуже різні правила. Операторам має бути багато чого переосмислити, і ми з нетерпінням чекаємо можливості взаємодіяти з ними з метою виявлення й визначення тих нових погроз – а також нових можливостей – які можуть запропонувати нам мережі п'ятого покоління.

Для коректного відображення протоколів GSM у розроболеному у даній роботі програмному забезпеченні системи кібербезпеки для протидії атаці на

протокол SS7 – SS7attack не потрібно встановлювати ніяких додаткових діссекторів.

Єдине, що потрібно виставити настроювання для протоколу LAPD – Use GSM SAPI values.

GSM протоколи будуть інкапсулюватися в UDP пакети із заголовком GSMTAP при передачі через Um інтерфейс або в TCP пакети із заголовками OML, RSL при трасуванні A-bis інтерфейсу.

Запустимо SS7attack на прослуховування loopback-інтерфейсу й задамо фільтри, щоб бачити тільки GSM Um трафік.

Застосунки, здатні генерувати GSMTAP трафік для SS7attack звичайно діють у такий спосіб:

– Одержують Um фрейм по радіо інтерфейсу.

– Додають GSMTAP заголовок.

– Відправляють усе це на зазначену IP адресу в UDP пакетах (у нашому випадку на loopback).

Крім фільтра GSMTAP, можна використовувати інші фільтри, що починаються з GSM, наприклад GSM_SMS, для пошуку SMS-повідомлень у трафікі. Але використання фільтра GSMTAP дозволить переглядати весь GSM Um трафік, оскільки всі інші заголовки вкладені в GSMTAP.

Захват трафіку

Для вивчення GSM трафіку його потрібно спочатку якось записати. Використовуючи SDR пристрій як приймач, Ви зможете вивчати дані, передані на загальнодоступних каналах CCCH, якщо тільки вам не відомий Kc – сесійний ключ шифрування.

Однак при використанні Osmocombb, телефону з SIM-картою й застосунка mobile Ви знаєте свій ключ шифрування Kc і зможете переглядати весь свій трафік у відкритому виді, переданий через Um інтерфейс (радіо інтерфейс) в SS7attack, і вивчати як працюють реальні стільникові мережі.

Рисунок 3.2 – Функціональна схема системи

Якщо Ви запустили свою GSM мережу на основі двох Osmocombb-сумісних телефонів, Ви можете передавати GSMTAP пакети з Osmobts в SS7attack і в такий спосіб вивчати роботу мережі.

Справа в тому, що обоє чипа rtl2832u і r820t2, звичайно використовувані RTL-SDR сумісними пристроями, дуже сильно гріються при роботі, а ніякого охолодження дешеві приймачі не передбачають. По-друге, немає ніякого захисту від статики й можна запросто спалити приймач при неакуратному обігу. Є й інші проблеми. Тому, я рекомендую придбати офіційний RTL-SDR приймач останньої версії в алюмінієвому корпусі. Він коштує всього 1500 рублів, але в комплекті Ви одержите 2 антени, пасивне охолодження, захист від статики, стабільну роботу, поліпшену фільтрацію, і багато іншого.

Аналіз трафіку

Я не буду тут розглядати роботу з gr-GSM і RTL-SDR, тому відразу переходимо до Osmocom. Ви можете працювати з будь-якою комерційною GSM мережею використовуючи Open Source імплементацію стека протоколів GSM від

Osmocom. Інакше кажучи, Ви можете підключити телефон на чипсету Calypso до комп'ютера, запустити застосунок з пакета Osmocombb і дзвонити, відправляти SMS і робити все те, що вміє робити звичайний 2G телефон, плюс одержувати доступ до будь-якої технічної інформації про поточний сеанс зв'язку.

Застосунок, який нам потрібно для цього, називається mobile і перебуває тут src/host/layer23/src/mobile

Mobile – це застосунок, який запускається на стороні комп'ютера, але Ви також повинні завантажити в телефон прошивання layer1

У базовому виді Вам потрібно:

– Підключити телефон до комп'ютера через USB-TTL конвертер.

– Перевірити, що з'явився пристрій /dev/ttyusbx, завантажити в телефон прошивання layer1 через osmocon.

– Запустити mobile і вказати інтерфейс для відправлення GSMTAP, щоб ми могли прослухати цей інтерфейс в SS7attack.

Щоб запустити mobile, перейдіть до директорію з застосунком і виконаєте наступну команду

./mobile -i 127.0.0.1

При першому запуску Ви можете побачити повідомлення про відсутність конфігураційного файлу. Створіть його порожнім у директорії, зазначеної в повідомленні про помилку. (У мене це /root/.osmocom/bb).

Якщо Ви все зробили правильно, то побачите в консолі балка роботи mobile.

Тепер ви можете підключитися до термінала mobile і управляти телефоном так

telnet localhost 4247

Ви потрапите в інтуїтивно зрозумілий cisco-подібний VTY інтерфейс. Використовуйте команду help, TAB для авто доповнення або SHIFT+?

Команди show ms, show subscriber, show ba і show cell допоможуть одержати інформацію про поточне підключення. Для переходу в привілейований режим використовуйте enable і для запису змін у файл write file.

SS7attack у нас уже запущений, і Ви можете бачити GSMTAP пакети в списку. Поки телефон перебуває в режимі очікування, а нам ніхто не надсилає SMS і не дзвонить, вивід буде складатися з відкритого трафіку з керуючих каналів.

Ви можете відправити SMS зі свого особистого номера на телефон mobile або відправити SMS самому собі через VTY інтерфейс. Тоді, якщо пошукати в SS7attack трафік по фільтру GSM_SMS, ми побачимо.

Є й інші утиліти, що дозволяють захоплювати різного роду GSM трафік, їх Ви можете знайти отут src/host/layer23/src/misc

Зокрема, cell_log дозволяє визначити, які базові станції перебувають навколо Вас і вміє відправляти GSMTAP пакети в SS7attack.

ccch_scan дозволяє прослуховувати повідомлення на CCCH каналах, у тому числі SMS, якщо використовувати модифіковану версію з osmocombb вітки sylvain/burst_ind, яка дозволяє задати ключ шифрування Kc.

Однак у цьому випадку Вам доведеться ще прошити ваш USB-TTL конвертер для роботи на нестандартних швидкостях.

Що стосується захвата голосових викликів, те тут усе стає складніше, тому що реальний оператор може використовувати Frequency Hopping, що «розмазує» burst-и по різних частотах і нам потрібно іти за зміною частоти телефону, або записувати весь трафік на всіх можливих TCH частотах стільник, що у свою чергу може не дозволяти зробити ваш приймач через фізичні обмеження.

Якщо Ви досліджуєте свою власну мережу, то вам не потрібний Frequency Hopping, однак Osmocom не надає утиліт для сніффінгу голосових викликів за допомогою osmocombb-сумісних телефонів.

Аналіз трафіку інтерфейсу A-bis

A-bis – інтерфейс обміну повідомленнями між BTS і BSC. У нас немає доступу до досліджень такого роду трафіку комерційних мереж. Але Ви можете вивчати подібні речі на прикладі власної GSM мережі.

Щоб переглянути RSL повідомлення, потрібно почати прослуховувати loopback інтерфейс (зверніть увагу, що тут використовується TCP, а не UDP) і ви побачите повідомлення начебто цих:

Для RSL можна використовувати фільтр GSM_abis_rsl.

І для OML. Можна скористатися фільтром GSM_abis_oml або фільтрувати по портах 3002 і 3003.

У той же час SMS повідомлення будуть вкладені в RSL пакети, а не в GSMTAP, як у випадку з передачею через Um інтерфейс.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

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

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників.

Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

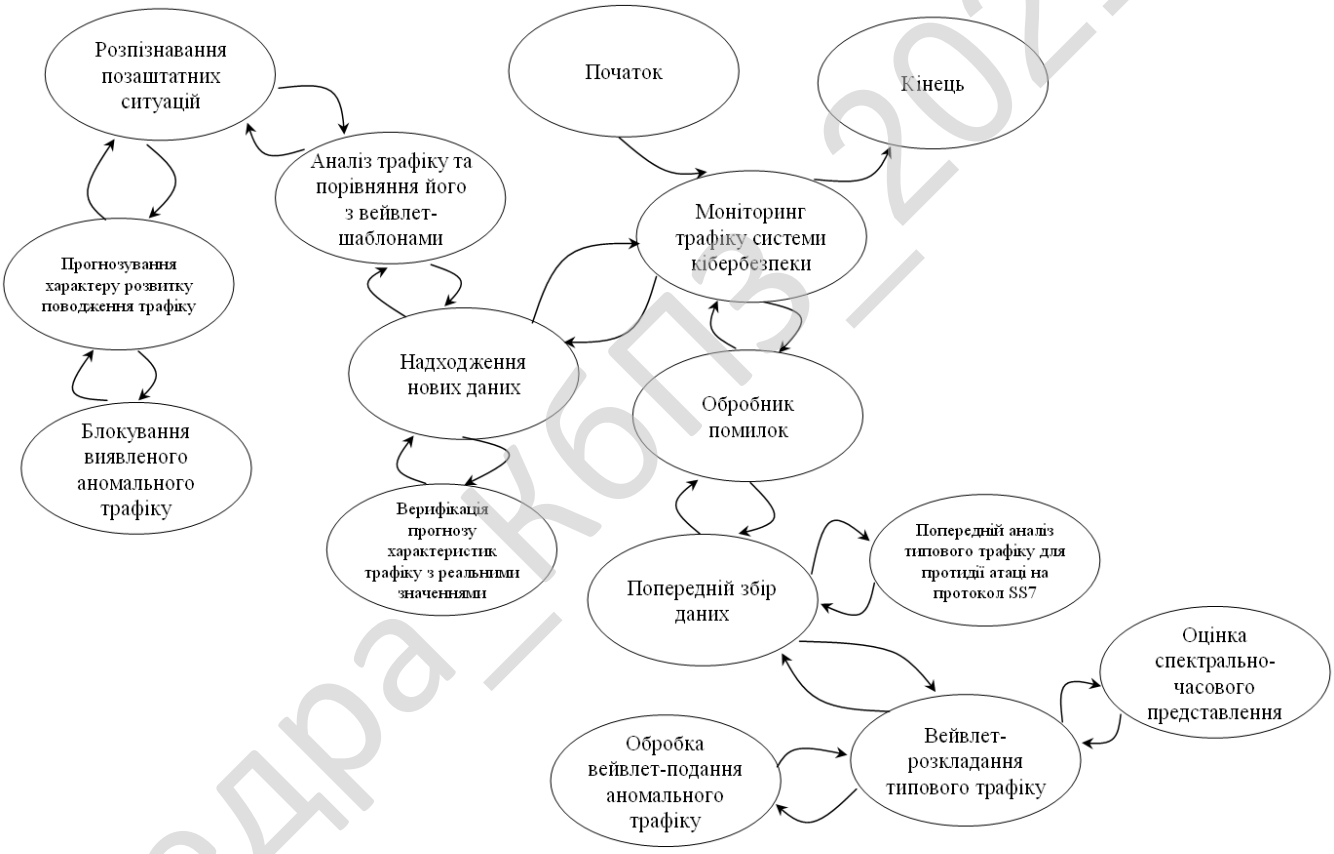

Рисунок 3.3 – Діаграма взаємодії процесів

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому.

Використовується модель проектування, графічне представлення

«потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду 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

Розглянемо послідовність дій та викликів підпрограм в загальному алгоритмі роботи основної програми що зображено на рисунку 4.1. у вигляді блок-схеми:

– Виведення вікна системи кібербезпеки для протидії атаці на протокол SS7.

– Запит створити базу образів.

– Попередній збір даних для протидії атаці на протокол SS7.

– Налаштування параметрів протидії та формату звітування.

– Виклик підпрограми моніторингу трафіку.

– Формування шаблонів обробки.

– Збереження шаблонів обробки в базі даних.

– Сканування та збір даних.

– Відображення інформації про ЛМ.

– Відображення отриманих даних.

– Виклик підпрограми розпізнавання вторгнень (рис. 4.2).

– Запит виявлено аномальний трафік.

– Виведення інформації про виявлення аномального трафіку.

– Блокування аномального трафіку.

– Вихід – кінець чергового циклу роботи програмного забезпечення якщо запит буду позитивний ПЗ буде завершено якщо ні проходе наступний цикл.

На рисунку 4.2 зображено роботу підпрограми з реалізацією наступних дій:

– Аналіз трафіку та порівняння його з шаблонами SS7.

– Запит знайдено спів падіння.

– Встановлення прапору аномального трафіку.

– Розпізнавання підтипу атаки.

– Формування звіту роботи та блокування атаки.

– Виявлення зв'язків між аномаліями, що відбулися в різних точках спостереження.

– Запис до журналу роботи ПЗ.

– Виведення звіту роботи системи.

У розділі 4.2 розглянуто теоретичні основи застосованого алгоритму захисту DES. Зараз розглянемо практичну реалізацію алгоритму у вигляді вихідного коду:

```
unit Uni_DES; // модуль реалізації алгоритму захисту DES
interface // інтерфейс
uses // підключені бібліотеки
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ImgList, ActnMan, ActnColorMaps, ToolWin, ActnCtrls,
ActnMenus, XPStyleActnCtrls, ActnList, ExtCtrls;
type
TF1 = class(TForm)
    ActionManager1: TActionManager;
    ActionMainMenuBar1: TActionMainMenuBar;
    XPColorMap1: TXPColorMap;
    Edit1: TEdit;
    Action1: TAction;
    Action2: TAction;
    Action3: TAction;
    Action4: TAction;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    OpenDialog2: TOpenDialog;
    SaveDialog2: TSaveDialog;
    ImageList1: TImageList;
```

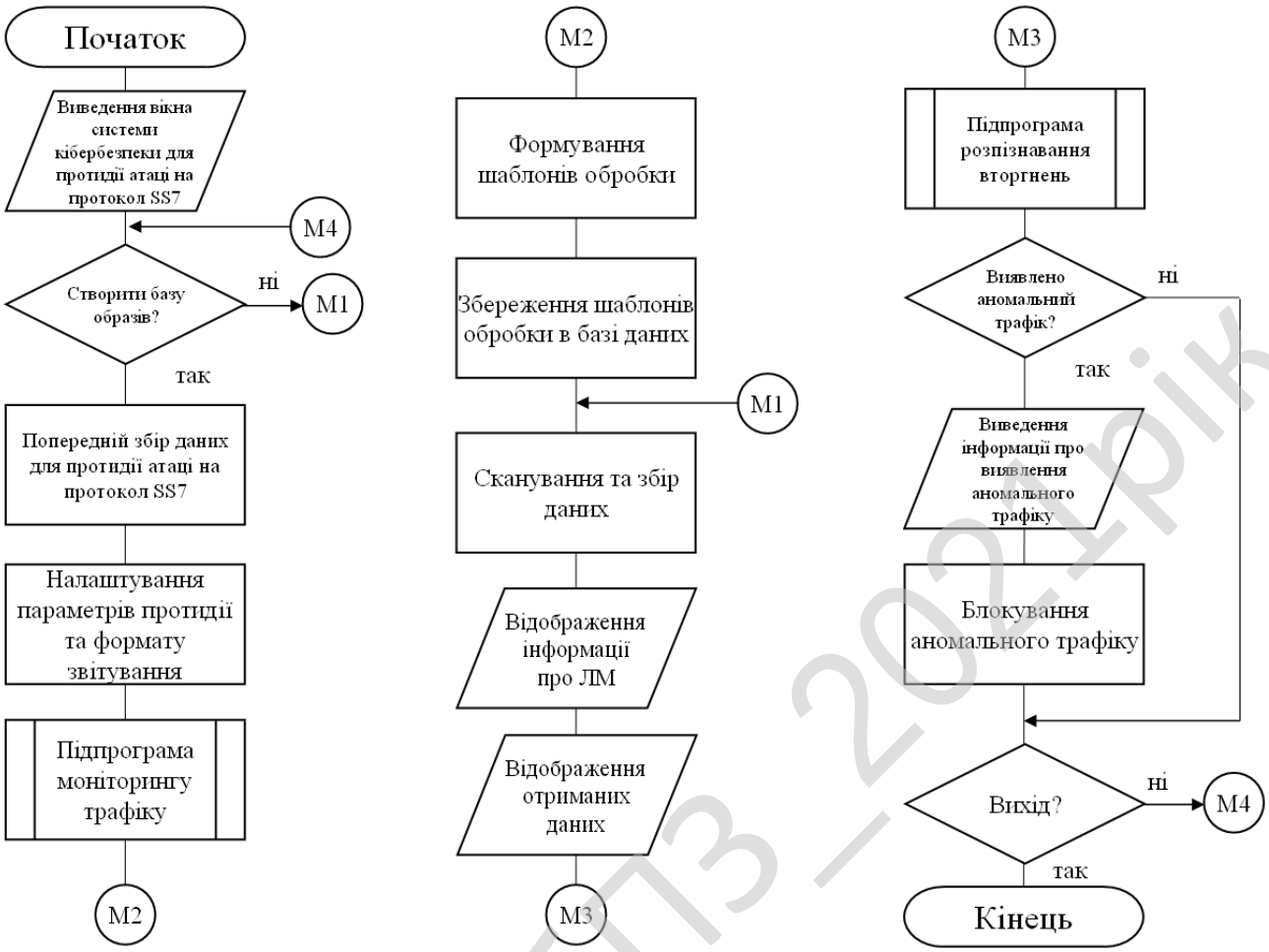

Рисунок 4.1 – Блок-схема основної програми

```
Image1: TImage;
Button1: TButton;
Button2: TButton;
Memo1: TMemo;
procedure Action1Execute(Sender: TObject);
procedure Action2Execute(Sender: TObject);
procedure Action3Execute(Sender: TObject);
procedure Action4Execute(Sender: TObject);
  private
  public
  end;

var
  F_1: TF1;
  fp1,fp2:file;
```

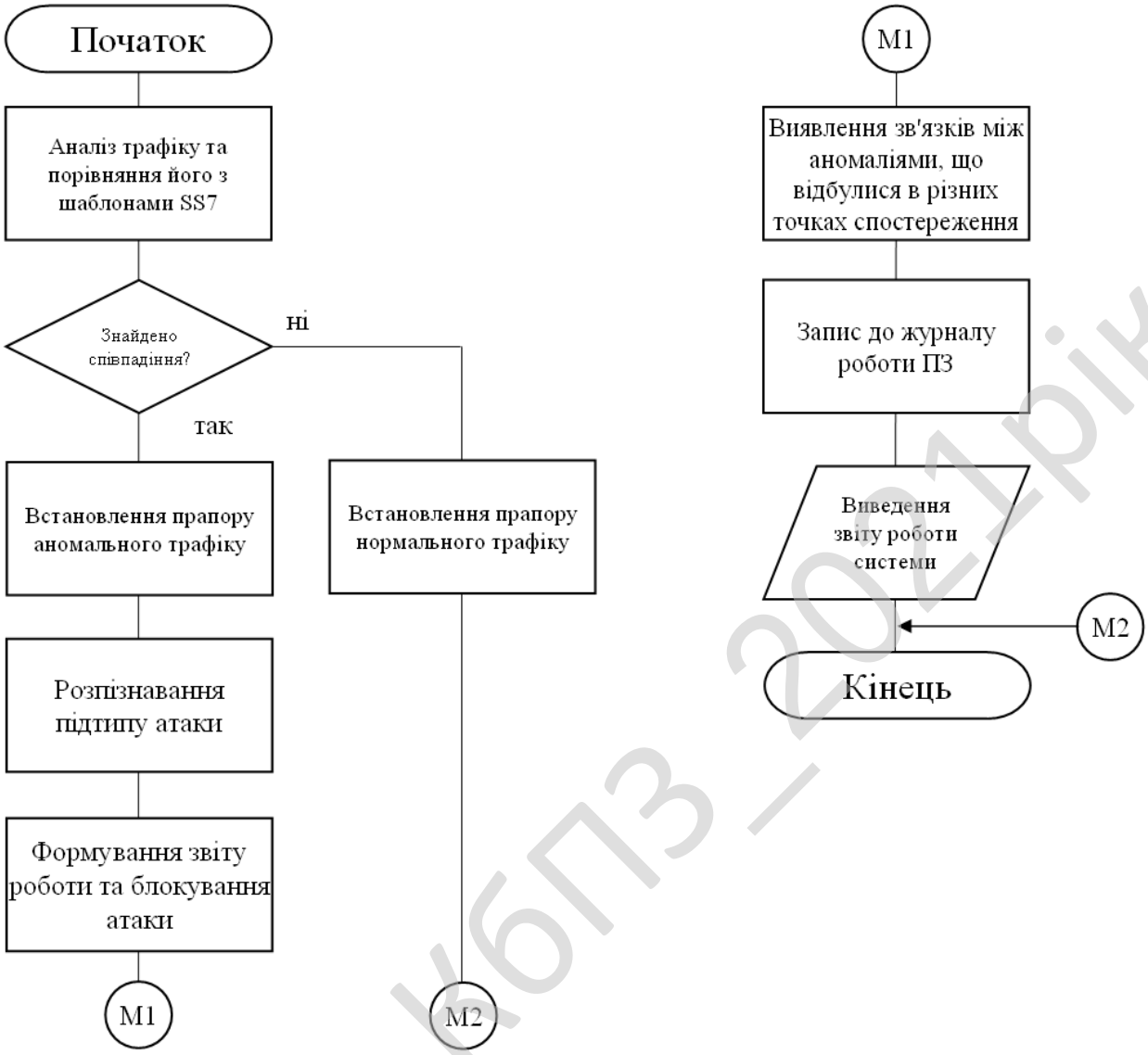

Рисунок 4.2 – Блок-схема роботи підпрограми

```
ip1: array[1..64] of integer=(58,50,42,34,26,18,10,2,
                              60,52,44,36,28,20,12,4,
                              62,54,46,38,30,22,14,6,
                              64,56,48,40,32,24,16,8,
                              57,49,41,33,25,17,9,1,
                              59,51,43,35,27,19,11,3,
                              61,53,45,37,29,21,13,5,
                              63,55,47,39,31,23,15,7);
ip2: array[1..64] of integer=(40,8,48,16,56,24,64,32,
                              39,7,47,15,55,23,63,31,
                              38,6,46,14,54,22,62,30,
                              37,5,45,13,53,21,61,29,
```

```
                                36,4,44,12,52,20,60,28,
                                35,3,43,11,51,19,59,27,
                                34,2,42,10,50,18,58,26,
                                33,1,41,9,49,17,57,25);
  i,j,k,flag:integer;
  t: array[1..64] of byte;
  l: array[1..32] of byte;
  r: array[1..32] of byte;
  tmp: array[1..64] of byte;
  pass: array[1..32] of byte;
  temp: byte;

implementation // реалізація

uses Unit2;

{$R *.dfm}

procedure TF1.Action1Execute(Sender: TObject);
begin
F_1.Close;
end;

procedure TF1.Action2Execute(Sender: TObject);
begin
Form2.ShowModal;
end;
// основний алгоритм
procedure TF1.Action3Execute(Sender: TObject);
begin
if Length(Edit1.Text)>1 then
  begin
   i:=1;
   flag:=0;
   while i<33 do
     begin
       for j:=1 to Length(Edit1.Text) do
         begin
           for k:=0 to 7 do
             begin
               pass[i]:=((byte(Edit1.Text[j]) shl k) and $80) shr 7;
               inc(i);
```

```
                if i>32 then
                  break;
              end;
            if i>32 then
              break;
          end;
      end;
// діалогове вікно
    OpenDialog1.FileName:='';
    OpenDialog1.InitialDir:=GetCurrentDir;
    if OpenDialog1.Execute then
      begin
        AssignFile(fp1,OpenDialog1.FileName);
        Reset(fp1, SizeOf(byte));
        SaveDialog1.FileName:=OpenDialog1.FileName+'.des';
        SaveDialog1.InitialDir:=GetCurrentDir;
        if SaveDialog1.Execute then
          begin
// підключення зазначеного вище файлу
            AssignFile(fp2,SaveDialog1.FileName);
            Rewrite(fp2, SizeOf(byte));
// початок вкладених циклів
            while not eof(fp1) do
              begin
                for i:=1 to 8 do
                  begin
                    if not eof(fp1) then
                      BlockRead(fp1, temp, SizeOf(byte))
                    else
                      begin
                        temp:=0;
                        inc(flag);
                      end;
                    for k:=0 to 7 do
                      begin
                        tmp[(i-1)*8+k+1]:=((temp shl k) and $80) shr 7;
                      end;
                  end;
                for i:=1 to 64 do
                  t[i]:=tmp[ip1[i]];
                for i:=1 to 32 do
                  begin
```

```
                    l[i]:=t[i];
                    r[i]:=t[i+32];
                  end;
                for i:=1 to 16 do
                  begin
                    for j:=1 to 32 do
                      begin
                        tmp[j]:=l[j];
                        l[j]:=r[j];
                      end;
                    for j:=1 to 32 do
                      begin
                        r[j]:=l[j] xor pass[j];
                        r[j]:=r[j] xor tmp[j];
                      end;
                  end;
                for i:=1 to 32 do
                  begin
                    t[i]:=l[i];
                    t[i+32]:=r[i];
                  end;
                for i:=1 to 8 do
                  begin
                    for k:=0 to 7 do
                      begin
                        temp:=temp shl 1;
                        temp:=temp+(t[ip2[(i-1)*8+k+1]] and $1);
                      end;
                    BlockWrite(fp2, temp, SizeOf(byte));
                  end;
              end;
            BlockWrite(fp2, flag, SizeOf(byte));
            CloseFile(fp2);
          end;
        CloseFile(fp1);
      end;
  end;
end;

procedure TF1.Action4Execute(Sender: TObject);
begin
if Length(Edit1.Text)>1 then
```

```
  begin
    i:=1;
    flag:=0;
    while i<33 do
      begin
        for j:=1 to Length(Edit1.Text) do
          begin
            for k:=0 to 7 do
              begin
                pass[i]:=((byte(Edit1.Text[j]) shl k) and $80) shr 7;
                inc(i);
                if i>32 then
                  break;
              end;
            if i>32 then
              break;
          end;
      end;
    OpenDialog2.FileName:='';
    OpenDialog2.InitialDir:=GetCurrentDir;
// діалогове вікно 2
    if OpenDialog2.Execute then
      begin
        AssignFile(fp1,OpenDialog2.FileName);
        Reset(fp1, SizeOf(byte));
        SaveDialog2.FileName:=OpenDialog2.FileName;
        SaveDialog2.InitialDir:=GetCurrentDir;
        if SaveDialog2.Execute then
          begin
            AssignFile(fp2,SaveDialog2.FileName);
            Rewrite(fp2, SizeOf(byte));
// початок вкладених циклів обробки
            while not eof(fp1) do
              begin
                for i:=1 to 8 do
                  begin
                    if not eof(fp1) then
                      BlockRead(fp1, temp, SizeOf(byte))
                    else
                      begin
                        flag:=temp;
                      end;
```

```
        for k:=0 to 7 do
          begin
            tmp[(i-1)*8+k+1]:=((temp shl k) and $80) shr 7;
          end;
      end;
    for i:=1 to 64 do
      t[ip2[i]]:=tmp[i];
    for i:=1 to 32 do
      begin
        l[i]:=t[i];
        r[i]:=t[i+32];
      end;
    for i:=1 to 16 do
      begin
        for j:=1 to 32 do
          begin
            tmp[j]:=r[j];
            r[j]:=l[j];
          end;
        for j:=1 to 32 do
          begin
            l[j]:=r[j] xor pass[j];
            l[j]:=l[j] xor tmp[j];
          end;
      end;
    for i:=1 to 32 do
      begin
        t[i]:=l[i];
        t[i+32]:=r[i];
      end;
    for i:=1 to 64 do
      tmp[ip1[i]]:=t[i];
    for i:=1 to 8 do
      begin
        for k:=0 to 7 do
          begin
            temp:=temp shl 1;
            temp:=temp+(tmp[(i-1)*8+k+1] and $1);
          end;
        BlockWrite(fp2, temp, SizeOf(byte));
      end;
  end;
```

```
                Seek(fp2,FilePos(fp2)-flag-8);
                Truncate(fp2);
                CloseFile(fp2);
              end;
            CloseFile(fp1);
          end;
      end;
    end;

    end.
```

Розглянемо БД що була використана, але спочатку розберемо що таке система керування базами даних (далі СКБД).

Це набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Першим поколінням СКБД прийнято вважати ієрархічні й мережеві системи. Ці системи отримали широке поширення в 1970-х роках, а першою комерційною системою цього типу була система IMS компанії IBM.

У 1980-х роках ці системи були витіснені системами другого покоління – повсюдно використовуваними і донині реляційними СКБД. У цих системах використовувалися непроцедурні мови управління даними (SQL) і передбачався значний ступінь незалежності даних. Реляційні системи внесли значні удосконалення в управління даними: графічний користувацький інтерфейс (GUI), клієнт-серверні застосунки, розподілені бази даних, паралельний пошук даних та інтелектуальний аналіз даних.

Але вже до кінця 1980-х років існуюча тоді реляційна модель перестала задовольняти розробників в силу низки обмежень. Відповіддю на зростаючу складність програм баз даних стали два нових напрямки розвитку СКБД: об'єктно-орієнтовані СКБД і об'єктно-реляційні СКБД.

У 1991 був утворений консорціум ODMG, основною метою якого стало вироблення промислового стандарту об'єктно-орієнтованих баз даних. Між 1993 та 2001 роками ODMG опублікувала п'ять ревізій своїх специфікацій. Остання

версія стандарту має індекс 3.0, після чого група розпустилася. До кінця 1990-х існувало близько десяти компаній, що виробляли комерційні продукти, що позиціонуються на ринку як ООСКБД. Найбільш відомими системами даного класу стали Objectivity, Versant виробництва однойменних компаній, а також СКБД Jasmine, випущена компанією CA. Незважаючи на переваги, що дозволяють ефективніше вирішувати певний ряд завдань, об'єктно-орієнтовані системи так і не змогли завоювати значущу частку ринку СКБД, залишившись «нішевим» продуктом.

Постачальниками традиційних реляційних СКБД також була проведена значна робота з об'єднання об'єктно-орієнтованих і реляційних систем. Розробники постаралися розширити мову SQL, щоб включити в неї концепції об'єктно-орієнтованого підходу, зберігаючи переваги реляційної моделі (об'єктні розширення мови SQL були зафіксовані в стандарті SQL:1999).

Основний принцип – це еволюційний розвиток можливостей СКБД без поломки попередніх підходів та зі збереженням наступності з системами попереднього покоління.

Поняття СКБД третього покоління, якими, власне кажучи, і є об'єктно-реляційні СКБД, з'явилося після опублікування групою відомих фахівців в області баз даних «Маніфесту систем баз даних третього покоління». Основні принципи СКБД третього покоління, позначені в маніфесті:

Крім традиційних послуг з управління даними, СКБД третього покоління повинні забезпечити підтримку розвиненіших структур об'єктів і правил. Розвинутіша структура об'єктів характеризує засоби, необхідні для зберігання і маніпулювання нетрадиційними елементами даних (тексти, просторові дані, мультимедіа).

СКБД третього покоління повинні включити в себе СКБД другого покоління. Системи другого покоління внесли вирішальний вклад у двох областях – непроцедурний доступ за допомогою мови запитів SQL і незалежність

даних. Ці досягнення обов'язково повинні враховуватися в системах третього покоління.

СКБД третього покоління повинні бути відкриті для інших підсистем. Це включає оснащення різноманітними інструментами підтримки прийняття рішень, доступом з багатьох мов програмування, інтерфейсами до існуючих популярних систем і бізнес-застосунків, можливістю запуску програм з бази даних на іншій машині і розподілені СКБД. Весь набір інструментів і СКБД має ефективно функціонувати на різноманітних апаратних платформах з різними операційними системами.

Крім того, СКБД, що розраховує на широку сферу застосування, повинна бути оснащена мовою четвертого покоління (4GL).

У середині 1990 років було лише кілька дослідних прототипів СКБД, які поєднали найкращі риси реляційних і об'єктно-орієнтованих СКБД. Першим комерційним продуктом, якому були властиві об'єктно-реляційні риси, став Universal Server компанії Informix (згодом була поглинена IBM). В даний час більшість цих ідей вже втілено в реальних комерційних рішеннях, в тому числі і в продуктах основних постачальників СКБД (Oracle Database і IBM DB2).

Розвиток індустрії систем керування базами даних базується на значних фундаментальних наукових дослідженнях. Найчастіше, між самими дослідженнями та їхньою конкретною реалізацією в прикладних рішеннях минають роки, а іноді й десятиліття. Роботу в області управління даними проводять як університетські дослідницькі групи (MIT, Berkeley), так і центри розробок основних постачальників СКБД (Oracle, IBM, Microsoft). Інвестування в управління даними – це довгострокове, і разом з тим, вигідне вкладення коштів. В даний час дослідники мають у своєму розпорядженні засоби, що дозволяють ефективно реалізувати найскладніші запити, що маніпулюють терабайтами й петабайтами різних даних.

Основними тенденціями, які дали привід для проведення різних масштабних досліджень в області баз даних стали:

– Експонентний ріст даних. Обсяг даних, у тому числі синтетичних, що генеруються автоматизованими системами, значно зріс. Збільшилося і число прикладних областей, в яких вимагається обробка великих обсягів даних. До таких областей тепер відносяться не тільки традиційні корпоративні програми, пошук у в еб, але також і наукові дослідження, обробка природних мов, аналіз соціальних мереж тощо.

– Значне ускладнення структур використовуваних даних. Прості види даних у вигляді чисел і символьних рядків стали доповняться численною мультимедійною інформацією, просторовими, процедурними даними та великою кількістю інших складних форматів.

– Широке поширення дешевих високопродуктивних апаратних засобів. Щорічно ми спостерігаємо зростання обчислювальних можливостей мікропроцесорів, збільшення ємності і зниження вартості доступних і зручних в експлуатації пристроїв дискової оперативної пам'яті.

– Активний розвиток засобів комунікації та «всесвітньої павутини» World Wide Web. WWW стає єдиним інформаційним середовищем, що пронизує весь світ і об'єднує величезне число користувачів та електронних пристроїв.

– Поява нових важливих областей застосування СКБД. У першу чергу, це пов'язано з інтелектуальним аналізом даних, сховищами даних, а останнім часом – з паралельними обчисленнями і хмарними технологіями.

Основні характеристики СКБД

1. Контроль за надлишковістю даних.
2. Несуперечливість даних.
3. Підтримка цілісності бази даних (коректність та несуперечливість).
4. Цілісність описується за допомогою обмежень.
5. Незалежність прикладних програм від даних.
6. Спільне використання даних.
7. Підвищений рівень безпеки.

Можливості СКБД

1. Дозволяється створювати БД (здійснюється за допомогою мови визначення даних DDL (Data Definition Language)).

2. Дозволяється додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними DML, яку часто називають мовою запитів)

3. Можна надавати контрольований доступ до БД за допомогою: Системи забезпечення захисту, яка запобігає несанкціонованому доступу до БД; Системи керування паралельною роботою прикладних програм, яка контролює процеси спільного доступу до БД; Система відновлення – дозволяє відновлювати БД до попереднього несуперечливого стану, що був порушений в результаті збою апаратного або програмного забезпечення.

Основні компоненти середовища СКБД

1. Апаратне забезпечення.

2. Програмне забезпечення.

3. Дані.

4. Процедури – інструкції та правила, які повинні враховуватись при проектуванні та використанні БД.

5. Користувачі: адміністратори даних (керування даними, проектування БД, розробка алгоритмів, процедур) та БД (фізичне проектування, відповідальність за безпеку та цілісність даних); розробники БД; прикладні програмісти; кінцеві користувачі.

Архітектура СКБД

Існує трирівнева система організації СКБД ANSI-SPARC, при якій існує незалежний рівень для ізоляції програми від особливостей представлення даних на нижчому рівні.

Рівні:

1. Зовнішній – представлення БД з точки зору користувача.

2. Концептуальний – узагальнене представлення БД, описує які дані зберігаються в БД і зв'язки між ними. Підтримує зовнішні представлення, підтримується внутрішнім рівнем.

3. Внутрішній – фізичне представлення БД в комп'ютері.

Логічна незалежність – повна захищеність зовнішніх моделей від змін, що вносяться в концептуальну модель.

Фізична незалежність – захищеність концептуальної моделі від змін, які вносяться у внутрішню модель.

Використано SQLite – це полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92.

Сирцевий код SQLite поширюється як суспільне надбання (public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

З 2018р SQLite, як й JSON та CSV, рекомендований Бібліотекою Конгресу США формат зберігання структурованого набору даних. У 2005 році проект отримав нагороду Google-O'Reilly Open Source Awards.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми.

Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму.

SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок.

Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі рушія можливо використовувати SQlite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

Особливості SQLite:

1. Транзакції атомарні, послідовні, ізольовані, і міцні (ACID) навіть після збоїв системи і збоїв живлення.

2. Встановлення без конфігурації – не потребує ані установки, ані адміністрування.

3. Реалізує значну частину стандарту SQL92.

4. База даних зберігається в одному крос-платформовому файлі на диску.

5. Підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків і BLOBів.

6. Малий розмір коду: менше ніж 350KB повністю налаштований, і менш 200KB з опущеними додатковими функціями.

7. Швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій.

8. Простий, легкий у використанні API.

9. Написана в ANSI C, включена прив'язка до TCL; доступні також прив'язки для десятків інших мов.

10. Добре прокоментований сирцевий код зі 100% тестовий покриттям гілок.

11. Доступний як єдиний файл сирцевого коду на ANSI C, який можна легко вставити в інший проект.

12. Автономність: немає зовнішніх залежностей.

13. Крос-платформовість: з коробки підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE). Легко переноситься на інші системи.

14. Сирці перебувають в суспільному надбанні.

15. Поставляється з автономним клієнтом інтерфейсу командного рядка, який може бути використаний для управління базами даних SQLite.

Інструменти створення та обслуговування БД

Створення та обслуговування БД можуть здійснюватись через текстову консоль SQL-командами або через спеціальні інструменти, у тому числі – з графічним інтерфейсом користувача.

Технології, що підтримують SQLite та мови програмування

Сама бібліотека SQLite написана мовою С. Проте є реалізація бібліотеки на JavaScript sql.js, яка дозволяє обробляти файли БД безпосередньо в браузері.

Для інших мов програмування розроблено механізм підключення й роботи з БД через цю бібліотеку: C++, Java, Python, Perl, PHP, Ruby, Haskell, Scheme, Smalltalk, Lua тощо. Засоби для роботи з Tcl включені в комплект постачання SQLite. Повний список наявних засобів можна знайти на сторінці проекту.

Web-інструментарії

У ряді інструментаріїв присутня можливість використання SQLite як бази даних, наприклад: Django; Java; PHP; Ruby on Rails; Trac; Symfony; Parser.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму DES.

DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ.

Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною таблицею.

Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями.

Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

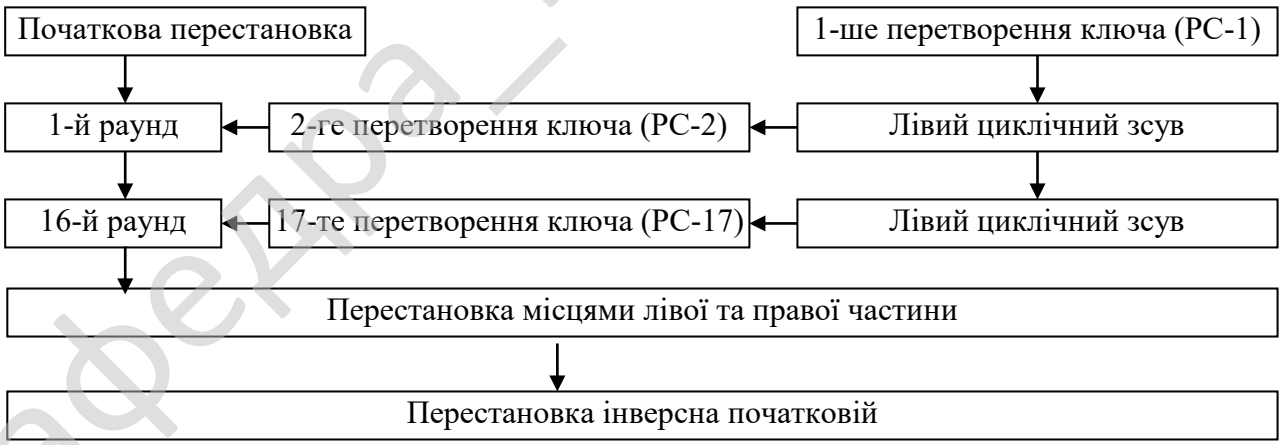

Рисунок 4.3 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші: Налаштування; Довідка; Журнал роботи.

– Верхнього меню: Файл; Аналізатор; Блокувальник; Журнал подій; Параметри; Довідка.

– Розділу обрання групи обзору у вигляді деревообразної структури.

– Розділу виведення результату роботи системи.

– Діаграм навантаження.

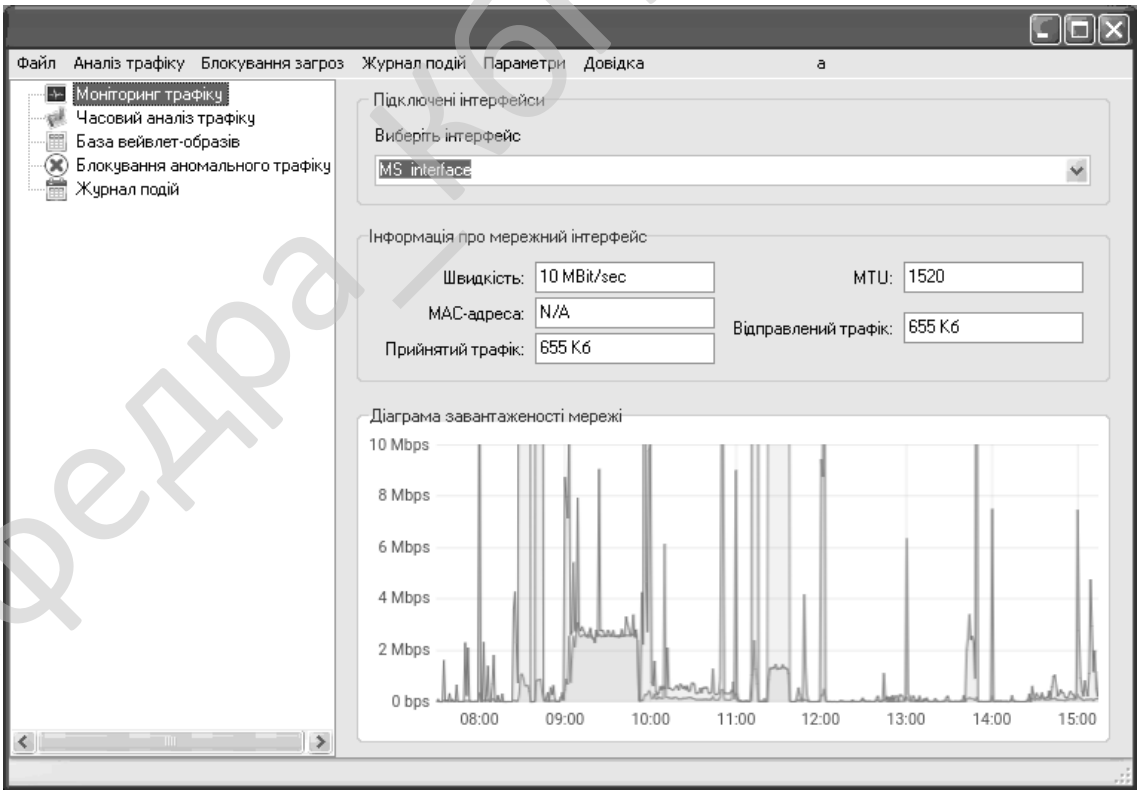

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

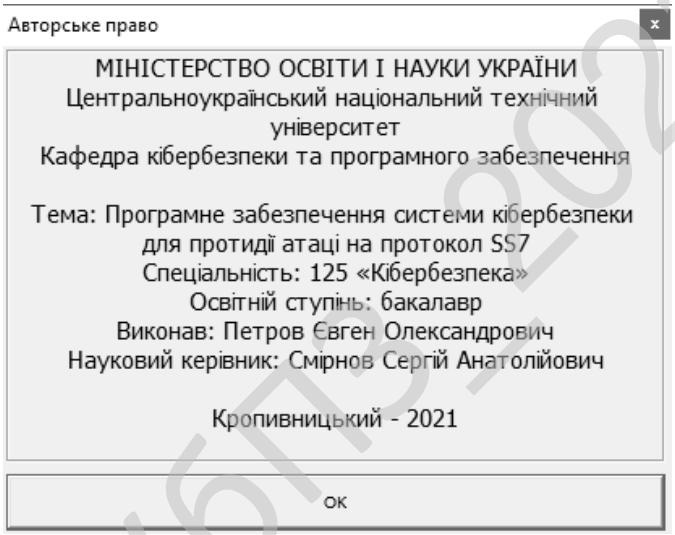

Рисунок 5.2 – Авторське право

Процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення. Таким чином у результаті вищерозглянутого можна стверджувати що розроблено інтерфейс системи у відповідності з вибраною метою роботи. Система містить максимальний необхідний набір функцій придатних для виконання будь-яких дій для забезпечення повноцінної роботи програми. Далі розглянемо висновки та використані літературні джерела.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи кібербезпеки для протидії атаці на протокол SS7.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем кібербезпеки для протидії атаці на протокол SS7.

– Досліджена система кібербезпеки для протидії атаці на протокол SS7.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для протидії атаці на протокол SS7.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання для протидії атаці на протокол SS7.

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 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для протидії атаці на протокол SS7. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DES.

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. Вегешна Ш. Качество обслуживания в сетях IP / Ш. Вегешна; пер.с англ. – М.: Издательский дом «Вильямс», 2003. – 386 с.

2. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский . – М.: Техносфера, 2003. – 512 с.

3. Горбенко І.Д. Прикладна криптологія. Теорія. Практика. Застосування / І.Д. Горбенко, Ю.І. Горбенко. – Х.:Форт, 2012. – 870 с.

4. Гордейчик С.В. Безопасность беспроводных сетей. / С.В. Гордейчик, В.В. Дубровин– М.: Горячая линия-Телеком, 2008. – 288 с.

5. Гольдштейн Б. Сети связи пост – NGN / Б. Гольдштейн, А. Кучерявый. – СПб.:БХВ-Петербург, 2013. – 160 с.

6. ГОСТ Р 51275-99 Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положення [Электронный ресурс]. – Режим доступа к ресурсу: http://www.gosthelp.ru/text/GOSTR5127599Zashhitainfor.html

7. ГОСТ Р ИСО/МЭК 15408-2-2002 Информационная технология. Методы и средства обеспечения безопасности критерии оценки безопасности информационных технологий. Часть 2 Функциональные требования безопасности [Электронный ресурс]. – Режим доступа к ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_15408-2-2002.htm

8. ГОСТ Р ИСО/МЭК 13335-1-2006 Информационная технология Методы и средства обеспечения безопасности Часть 1 Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий [Электронный ресурс]. – Режим доступа к ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_13335-1-2006.htm

9. ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и

концепции [Электронный ресурс]. – Режим доступа к ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179072

10. ГОСТ Р ИСО/МЭК 27004-2011 Информационная технология. Методы и средства обеспечения безопасности. Менеджмент информационной безопасности. [Электронный ресурс]. – Режим доступа к ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179060

11. Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

12. Смірнов О.А. Дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем / О.О. Кузнецов, О.А. Смірнов, Д.О. Даниленко // Інформаційна та економічна безпека: сучасний стан та тенденції розвитку : монографія за заг. ред. – Х.: ХІБС УБС НБУ – 2014 – С. 82-100.

13. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні системи та мережі / Д.О. Даниленко, О.А. Смірнов, Є.В. Мелешко // Системи озброєння і військова техніка. – Випуск 1(29) – Х.: ХУПС – 2012. – С. 92-100

14. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения. Часть 1. Корреляционный анализ сетевого трафика // А.А.Смирнов, Д.А. Даниленко, Е.В.Мелешко // Науково-технічний журнал «Інформаційно-керуючі системи на залізничному транспорті» – Випуск 4(95). – Х.: УкрДАЗТ – 2012. – С. 8-14.

15. Смирнов А.А. Методы обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 3(101) том 2. – Х.: ХУПС – 2012. – С. 152-155.

16. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, А.В. Коваленко // Системи

управління, навігації та зв'язку. – Випуск 1 (21) том 2. – Київ: ДП «ЦНДІНУ». – 2012. – С. 183-186.

17. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты компьютерных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, И.Г. Кирилов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 70-71.

18. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні мережі для підвищення інформаційної безпеки // Д.О. Даниленко // Збірник тез науково-практичної конференції «Захист інформації в інформаційно-комунікаційних системах». м. Київ. 24-27 квітня 2012 р. – Київ: НАУ. – 2012. – С. 22-25.

19. Смирнов А.А. Исследование систем обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко // Збірник тез доповідей VIII наукової конференції «Новітні технології – для захисту повітряного простору». Харків. 18-19 квітня 2012 р. – м. Харків. ХУПС. – 2012. – С. 45.

20. Смирнов А.А. Исследование методов сигнатурного обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях // Д.А. Даниленко // Збірник тез XIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: КНТУ. – 2012. – С. 43-45.

21. Смирнов А.А. Исследование методов проактивной защиты от вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «інтегровані інтелектуальні робототехнічні комплекси» (ІІРТК-2012). м. Київ. 15-16 травня 2012 р. – Київ: НАУ. – 2012. – С. 314-315.

22. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения на основе корреляционного анализа сетевого трафика / Д.А. Даниленко // Матеріали XII всеукраїнської наукової інтернет-конференції «Наукові дослідження: зв'язок теорії і практики». м. Тернопіль. 29-30 квітня 2012 р. – Тернопіль: ТНЕУ. – 2012. – С. 9-10.

23. Смирнов А.А. Метод детектирования вредоносного трафика в телекоммуникационных сетях на основе использования bds-тестирования / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології» (CSNT-2012). м. Київ. 13-15 червня 2012 р. – Київ: НАУ. – 2012. – С. 121.

24. Смирнов А.А. Обнаружение и предотвращение вторжений в компьютерных сетях на основе статистического анализа сетевого трафика / А.А. Смирнов, Д.А. Даниленко // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 13-14.

25. Смірнов О.А. дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем та мереж / О.А. Смірнов, Д.О. Даниленко // Збірник тез V Всеукраїнської науково-практичної конференції "Інформатика та системні науки". м. Полтава. 13-15 березня 2014 р. – Полтава: ПУЕТ. – 2014. – С. 289-291.

26. Смирнов А.А. Метод дисперсионного анализа сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях/ А.А. Смирнов, Д.А. Даниленко // Збірник тез VI міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 258.

27. Смірнов О.А. метод забезпечення інформаційної безпеки телекомунікаційних систем з використанням дисперсійного аналізу мережного трафіку / О.А. Смірнов, Д.О. Даниленко // Збірник тез міжнародної науково-

практичної конференції «Інформаційна та економічна безпека» (INFECO-2014)». м. Харків. 15-16 травня 2014 р. – Харків: ХІБС УБС НБУ. – 2014. – С. 135-139.

28. Девянин П.Н. Модели безопасности компьютерных систем / П.Н. Девянин. – М.:Издательский центр «Академия», 2005. – 144 с.

29. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты / В.В. Домарев. – К.: ООО "ТИД "ДС", 2002 – 688 с.

30. ДСТУ ISO/IEC TR 13243-2003 Інформаційні технології. Посібник із методів та механізмів якості послуг / [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/informaciini-tehnologiyi.-posibnik-iz-metodiv-ta-mehanizmiv--nor2718.html

31. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.: УкрНДІССІ, 1995. – 23 с.

32. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/docs/tdoc14237.php

33. Ершов В.А.. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов. – М.: МГТУ им. Н.Э. Баумана, 2003. – 432 с.

34. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

35. Информационная война и защита информации. Словарь основных терминов и определений [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csef.ru/files/csef/articles/2176/2176.pdf

36. Казарин О.В. Безопасность программного обеспечения компьютерных систем / О.В. Казарин. – М.:МГУЛ, 2003. – 212 с.

37. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

і. Касперский К. Техника сетевых атак. [Електронний ресурс]. – Режим

доступу до ресурсу: http://rghost.ru/download/43730077/

38. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

39. Касперский К. Техника и философия хакерских атак. / К. Касперский. – М.: Солон-Пресс, 2004 – 272 с.

40. Концепция (основы государственной политики) Национальной безопасности Украины, утвержденная постановлением Верховной Рады Украины от 16 января 1997 г. № 3/97-ВР. [Электронный ресурс]. – Режим доступа к ресурсу: http://search.ligazakon.ua/l_doc2.nsf/link1/ F970003.html

41. Конри-Мюррей Эндрю Выявление вторжений / «Журнал сетевых решений/LAN», № 01, 2001 [Электронный ресурс]. – Режим доступа к ресурсу: http://www.osp.ru/lan/2001/01/131532/

42. Кузнецов О.О. Протоколи захисту інформації у комп'ютерних системах та мережах / О.О. Кузнецов, С.Г. Семенов. – Х.: ХНУРЕ, 2009. – 184 с.

43. Кузнецов О.О. Методи обробки сигналів даних та зображень / О.О. Кузнецов, Г.А. Кучук, С.Г. Семенов. –Х.: НТУ «ХПІ», 2011. – 292 с.

44. Кузнецов А.А. Дисперсионный анализ сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях / А.А. Кузнецов, А.А. Смирнов, Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 2(118). – Х.: ХУПС – 2014. – С. 81-85.

45. Лоскутов А.Ю. Основы теории сложных систем / А.Ю. Лоскутов, А.С. Михайлов. – М.-Ижевск: Институт компьютерных исследований, 2007. – 620 с.

46. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.:BHV – Санкт - Петербург, 2003. – 596 с.

47. Методи підвищення оперативності передачі даних та захисту інформації у телекомунікаційній мережі: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U006631

48. Методология функционального моделирования IDEF0.

Руководящий документ [Электронный ресурс]. – Режим доступа к ресурсу: http://www.scribd.com/doc/76782197/МЕТОДОЛОГИЯ-ФУНКЦИОНАЛЬНОГО-МОДЕЛИРОВАНИЯ-IDEF0

49. Моделирование технических систем с AllFusion Process Modeler (ранее BPwin) [Электронный ресурс]. – Режим доступа к ресурсу:: http://www.sdteam.com/t5506

50. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров сети АТМ / А.Н. Назаров. – М.: Горячая линия – Телеком, 2002. –255 с.

51. Наиболее опасная страна. Попытка заражения ПК. [Электронный ресурс]. – Режим доступа к ресурсу:: http://sooweb.ru/news/naibolee_opasnaja_strana_popytka_zarazhenija_pk/ 2012-01-23-526

52. НД ТЗІ Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csk.kfc.in.ua/documents/nakaz-web.doc

53. НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Электронный ресурс]. – Режим доступа к ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

54. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2012. – 943 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для протидії атаці на протокол SS7.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 185-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для протидії атаці на протокол SS7.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для протидії атаці на протокол SS7;

– 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 XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

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 10.4.1.

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 – 87 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

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 кваліфікаційної бакалаврської роботи на попередній захист 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 7.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки для протидії атаці на протокол SS7

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 35

Літера: РП

Кропивницький – 2021 року

Основна програма

unit MainFormUnit – Запуск основної форми програми;

```
interface

uses
  Windows, Graphics, ExtCtrls, Controls, StdCtrls, Buttons, Tabs,
  ComCtrls, Classes, SysUtils, Forms, dialogs,
  TrafficUnit, IPHelper, IPHLPAPI, ShellAPI;

type
  TMainForm = class(TForm)
    pnlMain: TPanel;
    pnlBottom: TPanel;
    pc: TPageControl;
    tsAbout: TTabSheet;
    tsTraffic: TTabSheet;
    ExitButton: TButton;
    TrafficTabs: TTabSet;
    GroupBox: TGroupBox;
    ledAdapterDescription: TLabeledEdit;
    UnFreezeButton: TBitBtn;
    FreezeButton: TBitBtn;
    ClearCountersButton: TBitBtn;
    ledMACAddress: TLabeledEdit;
    gbIN: TGroupBox;
    ledOctInSec: TLabeledEdit;
    ledAvgINSec: TLabeledEdit;
    ledPeakINSec: TLabeledEdit;
    ledTotalIN: TLabeledEdit;
    gbOUT: TGroupBox;
    ledOctOUTSec: TLabeledEdit;
    ledAvgOUTSec: TLabeledEdit;
    ledPeakOUTSec: TLabeledEdit;
    ledTotalOUT: TLabeledEdit;
    Timer: TTimer;
    gbTime: TGroupBox;
    ledStartedAt: TLabeledEdit;
    ledActiveFor: TLabeledEdit;
    RemoveInactiveButton: TBitBtn;
    StatusText: TStaticText;
    cbOnTop: TCheckBox;
    Panel3: TPanel;
    ProductName: TLabel;
    lblURL: TLabel;
    Label3: TLabel;
    ProgramIcon: TImage;
    StaticText1: TStaticText;
    ledSpeed: TLabeledEdit;
    procedure TimerTimer(Sender: TObject);
    procedure ClearCountersButtonClick(Sender: TObject);
    procedure cbOnTopClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure TrafficTabsChange(Sender: TObject; NewTab: Integer;
      var AllowChange: Boolean);
    procedure ExitButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FreezeButtonClick(Sender: TObject);
    procedure UnFreezeButtonClick(Sender: TObject);
    procedure RemoveInactiveButtonClick(Sender: TObject);
    procedure lblURLClick(Sender: TObject);
    procedure StaticText1Click(Sender: TObject);
    procedure pcChange(Sender: TObject);
    procedure ledAdapterDescriptionChange(Sender: TObject);
  private
```

```
    procedure HandleNewAdapter(ATraffic : TTraffic);
    procedure HandleFreeze(ATraffic : TTraffic);
    procedure HandleUnFreeze(ATraffic : TTraffic);
    function LocateTraffic(AdapterIndex : DWord) : TTraffic;
    procedure ProcessMIBData;
    procedure ClearDisplay;
    procedure RefreshDisplay;
  public
    { }
  end;

var
  MainForm: TMainForm;
  ActiveTraffic : TTraffic;

implementation
{$R *.dfm}

procedure TMainForm.ClearDisplay;
var
 j:integer;
begin
 TrafficTabs.Tabs.Clear;
 StatusText.Caption:='';
 for j:=0 to GroupBox.ControlCount-1 do
  begin
   if GroupBox.Controls[j] is TCustomEdit
   then TCustomEdit(GroupBox.Controls[j]).Text:='';
  end;
end;

procedure TMainForm.TimerTimer(Sender: TObject);
begin
 Timer.Enabled:=false;
 ProcessMIBData;
 Timer.Enabled:=true;
end;

procedure TMainForm.ClearCountersButtonClick(Sender: TObject);
begin
 ActiveTraffic.Reset;
 RefreshDisplay;
end;

procedure TMainForm.cbOnTopClick(Sender: TObject);
begin
 if cbOnTop.Checked=true
 then FormStyle:=fsSTAYONTOP
 else FormStyle:=fsNORMAL;
end;
procedure TMainForm.FormDestroy(Sender: TObject);
var
 i: integer;
begin
 Timer.OnTimer:=nil;
 ActiveTraffic:=nil;
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  TrafficTabs.Tabs.Objects[i].Free;
end;

procedure TMainForm.TrafficTabsChange(Sender: TObject; NewTab: Integer; var
AllowChange: Boolean);
begin
 if NewTab=-1
 then ActiveTraffic:=nil
 else ActiveTraffic:=TTraffic(TrafficTabs.Tabs.Objects[NewTab]);
 RefreshDisplay;
end;
```

```
procedure TMainForm.ExitButtonClick(Sender: TObject);
begin
 Close;
end;

procedure TMainForm.FormCreate(Sender: TObject);
begin
  Timer.Interval:=1000; // усі розрахунки за 1 сек.
 //
 ClearDisplay;
 ActiveTraffic:=nil;
 pcChange(Sender);
 Timer.Enabled:=True;
end;

procedure TMainForm.RefreshDisplay;
begin
 if not Assigned(ActiveTraffic)
 then
  begin
   ClearDisplay;
   Exit;
  end;
 with ActiveTraffic do
  begin
   FreezeButton.Visible:=Connected;
   UnFreezeButton.Visible:=Connected;
   ClearCountersButton.Visible:=Connected;
   RemoveInactiveButton.Visible:=not Connected;

   FreezeButton.Enabled:=Running;
   UnFreezeButton.Enabled:=not Running;

   ledAdapterDescription.Text:=Description;
   ledMACAddress.Text:=MAC;

   ledSpeed.Text:=BitsToFriendlyString(Speed);

   ledOctInSec.Text:=BytesToFriendlyString(InPerSec);
   ledPeakInSec.Text:=BytesToFriendlyString(PeakInPerSec);
   ledAvgINSec.Text:=BytesToFriendlyString(AverageInPerSec);
   ledTotalIN.Text:=BytesToFriendlyString(InTotal);

   ledOctOUTSec.Text:=BytesToFriendlyString(OutPerSec);
   ledPeakOUTSec.Text:=BytesToFriendlyString(PeakOutPerSec);
   ledAvgOUTSec.Text:=BytesToFriendlyString(AverageOutPerSec);
   ledTotalOUT.Text:=BytesToFriendlyString(OutTotal);

   self.ledStartedAt.Text:=DateTimeToStr(StartedAt);
   self.ledActiveFor.Text:=FriendlyRunningTime;

   StatusText.Caption:=GetStatus;
  end;
end;

procedure TMainForm.ProcessMIBData;
var
 MibArr: IpHlpAPI.TMIBIfArray;
 i: integer;
 ATraffic: TTraffic;
begin
 Get_IfTableMIB(MibArr);  // Беремо поточні MIB дані
 //Мітку не знайдено, або не підключено
 for i:= 0 to -1 + TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[i]);
   if ATraffic.Connected
   then ATraffic.Found:=false;
  end;
```

```
  //процесс
 if Length(MibArr)>0
 then
  begin
   for i:=Low(MIBArr) to High(MIBArr) do
    begin
     ATraffic:=LocateTraffic(MIBArr[i].dwIndex);
     if Assigned(ATraffic)
     then
      begin
       //заново підключаємось
       ATraffic.NewCycle(MIBArr[i].dwInOctets, MIBArr[i].dwOutOctets,
MIBArr[i].dwSpeed);
      end
     else
      begin
       //новий запис у таблицю!
       ATraffic:=TTraffic.Create(MIBArr[i], HandleNewAdapter);
       ATraffic.Found:=true;
       ATraffic.OnFreeze:=HandleFreeze;
       ATraffic.OnUnFreeze:=HandleUnFreeze;
      end;
    end;
  end;
 //Мітка не знайдена
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  if not TTraffic(TrafficTabs.Tabs.Objects[i]).Found
  then TTraffic(TrafficTabs.Tabs.Objects[i]).MarkDisconnected;
 RefreshDisplay;
end;

function TMainForm.LocateTraffic(AdapterIndex : DWord): TTraffic;
var
 j: cardinal;
 ATraffic: TTraffic;
begin
 Result:=nil;
 if TrafficTabs.Tabs.Count=0
 then Exit;

 for j:= 0 to -1+TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[j]);
   if ATraffic.InterfaceIndex=AdapterIndex
   then
    begin
     Result:=ATraffic;
     Result.Found:=true;
     Break;
    end;
  end;
end;

procedure TMainForm.HandleNewAdapter(ATraffic: TTraffic);
begin
 //додаємо адаптер
 TrafficTabs.Tabs.AddObject(ATraffic.IP, ATraffic);
 // вибираємо
 TrafficTabs.TabIndex:=-1+TrafficTabs.Tabs.Count;
end;

procedure TMainForm.FreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.Freeze;
end;

procedure TMainForm.UnFreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.UnFreeze;
```

```
end;

procedure TMainForm.HandleFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.HandleUnFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.RemoveInactiveButtonClick(Sender: TObject);
begin
 if not ActiveTraffic.Connected
 then //точна перевірка
  begin
   ActiveTraffic.Free;
   ActiveTraffic:=nil;
   TrafficTabs.Tabs.Delete(TrafficTabs.TabIndex);
   TrafficTabs.SelectNext(False);
  end;
 RefreshDisplay;
end;

procedure TMainForm.lblURLClick(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.StaticText1Click(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.pcChange(Sender: TObject);
begin
 pnlBottom.Visible:=pc.ActivePage=tsTraffic;
end;

procedure TMainForm.ledAdapterDescriptionChange(Sender: TObject);
begin
  ledAdapterDescription.Hint:=ledAdapterDescription.Text;

ledAdapterDescription.ShowHint:=Canvas.TextWidth(ledAdapterDescription.Text)>led
AdapterDescription.ClientWidth;
end;

end.
```

Файл SS7attack.dpr основної програми

```
//Файл основної програми до якого підключаються відповідні бібліотеки

program SS7attack;

uses
  Forms,
  IPHelper in ' IPHelper.pas' ,
  IPHLPAPI in ' IPHLPAPI.pas' ,
  MainFormUnit in ' MainFormUnit.pas'  {MainForm},
  TrafficUnit in ' TrafficUnit.pas' ;

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.Run;
end.
```

unit IPHelper – файл прогнозування системи кібербезпеки для протидії атаці на протокол SS7

```
interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.  0.  0.  0' ;

//------перетворення визначених імен портів у сервісні імена------

type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;

  PTTcpConnStatus = ^TTcpConnStatus;
  TTcpConnStatus = record
    LocalIP      : string;
    LocalPort    : string;
    RemoteIP     : string;
    RemotePort   : string;
    Status       : string;
  end;


const
    // для самих розповсюджених сервісів...
  WellKnownPorts: array[1..29] of TWellKnownPort
  = (
    ( Prt: 0; Srv: ' LOOPBACK' ),
    ( Prt: 7; Srv: ' ECHO'  ),     {Пінгування    }
    ( Prt: 9; Srv: ' DISCRD'  ),    { Відмова}
    ( Prt: 13; Srv: ' DAYTIM'  ),   {Час}
    ( Prt: 17; Srv: ' QOTD'  ),     {Вказник на час}
    ( Prt: 19; Srv: ' CHARGEN' ),  {Генерація символів}
    ( Prt: 20; Srv: ' FTP '  ),     { File Transfer Protocol}
    ( Prt: 21; Srv: ' FTPC'  ),     { File Transfer Control Protocol}
    ( Prt: 23; Srv: ' TELNET'  ),   {TelNet}
    ( Prt: 25; Srv: ' SMTP'  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv: ' TIME'  ),     { Часовий протокол }
    ( Prt: 43; Srv: ' WHOIS' ),     { WHO IS service  }
    ( Prt: 53; Srv: ' DNS '  ),     { Domain Name Service }
    ( Prt: 67; Srv: ' BOOTPS'  ),   { BOOTP Сервер }
    ( Prt: 68; Srv: ' BOOTPC'  ),   { BOOTP Клієнт }
    ( Prt: 69; Srv: ' TFTP'  ),     { стандартний FTP  }
    ( Prt: 70; Srv: ' GOPHER'  ),   { Gopher       }
    ( Prt: 79; Srv: ' FING'  ),     { Finger       }
    ( Prt: 80; Srv: ' HTTP'  ),     { HTTP         }
    ( Prt: 88; Srv: ' KERB'  ),     { Kerberos     }
    ( Prt: 109; Srv: ' POP2'  ),    { Post Office Protocol Version 2 }
    ( Prt: 110; Srv: ' POP3'  ),    { Post Office Protocol Version 3 }
    ( Prt: 119; Srv: ' NNTP'  ),    { Network News Transfer Protocol }
    ( Prt: 123; Srv: ' NTP '  ),    { Network Time protocol          }
    ( Prt: 135; Srv: ' LOCSVC' ),   { Локальні сервіси             }
    ( Prt: 137; Srv: ' NBNAME'  ),  { NETBIOS Імя сервісу         }
    ( Prt: 138; Srv: ' NBDGRAM'  ), { NETBIOS Сервіс датаграм      }
    ( Prt: 139; Srv: ' NBSESS'  ),  { NETBIOS Сессійний сервіс        }
    ( Prt: 161; Srv: ' SNMP'  )     { Simple Netw. Management Protocol }
    );


//-----------перетворення ICMP кодів помилок у рядок-------------------------
```

```
const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ERROR' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


//----------перетворення різних величин до рядку ----------------//

  ARPEntryType  : array[1..4] of string = ( ' Other' , ' Invalid' ,
    ' Dynamic' , ' Static'
    );

  TCPConnState  :      // стани підключення TCP
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

  TCPToAlgo     : array[1..4] of string =  // алгоритми часу TCP
    ( ' Const.Timeout' , ' MIL-STD-1778' ,
    ' Van Jacobson' , ' Other'  );

  IPForwTypes   : array[1..4] of string =   // IP пересилання методів
    ( ' other' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =  // IP пересилання протоколів
    ( ' OTHER' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );


//--------------експортуємі данні-----------------------------------------------
-

// дані перетворюються у Tstrings для представлення на дісплей
procedure Get_AdaptersInfo( List: TStrings );
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
procedure Get_IfTable( NameList, ItemList: TStrings );
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
```

```
procedure Get_RecentDestIPs( List: TStrings );

// додаємо функцію
procedure Get_OpenConnections( List: TList );



// утіліти перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;

//--------------Основні утіліти-----------------------------------------------

{ перетворюємо наступний токен у рядок, потім зчитуємо рядок та видаляємо його }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

//------------------------------------------------------------------------------
{ перетворюємо цифрові MAC-адреси у ww-xx-yy-zz строку }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := ' 00-00-00-00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2 ) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;

//------------------------------------------------------------------------------
{ перетворення IP-адреси в мережний байт типу DWORD у крапкову десяткову строку}
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
```

```
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//---------------------------------------------------------------------------
{ перетворення крапкової десяткової IP-адреси у мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;

//---------------------------------------------------------------------------
{ перетворення номера порту у мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

//---------------------------------------------------------------------------
{ перетворення номера порту у мережний байт типу string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;

//---------------------------------------------------------------------------
{ перетворення номера порту у сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку відсутності порту
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;

//---------------------------------------------------------------------------
{ general,  fixed network parameters }
procedure Get_NetworkParams( List: TStrings );
var
  InfoSize      : Longint;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  InfoSize := 0;
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  GetMem( pBuf, InfoSize );
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  if ErrorCode = ERROR_SUCCESS then
    with PTFixedinfo(pBuf)^ do
```

```
    begin
      List.Add( ' HOSTNAME          : '  + string( Імя хосту ) );
      List.Add( ' DOMAIN            : '  + string( імя домену ) );
      List.Add( ' SCOPE             : '  + string( ScopeID ) );
      List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
      List.Add( ' ROUTING ENABLED   :'  + IntToStr( Дозволена маршрутизація ) );
      List.Add( ' PROXY   ENABLED   :'  + IntToStr( Дозволений Proxy ) );
      List.Add( ' DNS     ENABLED   :'  + IntToHex( Дозволений DNS,8 ) );
    end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  FreeMem(pBuf);
end;

//---------------------------------------------------------------------------
function ICMPErr2Str( ICMPErrCode: DWORD) : string;
var
 i : integer;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;

//---------------------------------------------------------------------------
//процедура отримання системи кібербезпеки для протидії атаці на протокол SS7 та
занесення у таблицю
procedure Get_IfTable( NameList, ItemList: TStrings );
var
  IfRow         : TMibIfRow;
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  if (not Assigned( NameList ))
  or (not Assigned( ItemList ))  then EXIT;
  NameList.Clear;
  ItemList.Clear;
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

  // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 1 to NumEntries do
      begin
        IfRow := PTMibIfRow( pBuf )^;
        with IfRow do
        begin
          SetLength( sDescr, dwDescrLen );
          move( bDescr, sDescr[1], Length( sDescr ) );
          sDescr := trim( sDescr );
          NameList.Add( sDescr );
          ItemList.Add( Format( ' %0.8x|%2d| %16s| %4d| %8d| %8d| %8d' ,
            [dwIndex, dwType,
```

```
            MacAddr2Str( TMacAddress( bPhysAddr ), dwPhysAddrLen )
              , dwMTU, dwSpeed,
              dwInOctets, dwOutOctets,
              dwOPerStatus] )
              );
        end;
        inc( pBuf, SizeOf( IfRow ) );
      end;
    end
    else  begin
      NameList.Add( ' без даних' );
      ItemList.Add( ' немає даних'  );
    end;
  end
  else begin
    NameList.Add( ' Oops' );
    ItemList.Add( SysErrorMessage( GetLastError ) );
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IfRow ) );
  FreeMem( pBuf );
end;


//---------------------------------------------------------------------------
//Занесення даних у таблицю MIB
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
var
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

   // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( MIBIfArray, NumEntries );
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 0 to pred(NumEntries) do
      begin
        MIBIfArray[i] := PTMibIfRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIfRow ) );
      end;
    end
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMIBIfRow ) );
  FreeMem( pBuf );
end;



//---------------------------------------------------------------------------
//Заносимо дінні про адреси у таблиці MIB
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
```

```
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;



//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
procedure Get_AdaptersInfo( List: TStrings );
var
  Error,
  BufLen      : DWORD;
  P           : Pointer;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  Descr,
  LocalIP,
  GatewayIP,
  DHCPIP      : string;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  BufLen := SizeOf( AdapterInfo^ );
  New( AdapterInfo );
  Error := GetAdaptersInfo( AdapterInfo, @BufLen );
  P := AdapterInfo;
  if Error = NO_ERROR then
  begin
    while P <> nil do
      with TIP_ADAPTER_INFO(P^) do
      begin
        SetLength( Descr, SizeOf( Description ) );
        Descr := Trim( string( Description ) );
        //
        if IPAddressList.IpAddress[1] <> #0 then
          LocalIP := IPAddressList.IpAddress
        else
          LocalIP := NULL_IP;
        //
        if GateWayList.IPAddress[1] <> #0 then
```

```
          GateWayIP := GatewayList.IPAddress
        else
          GateWayIP := NULL_IP;
        //
        if DHCPServer.IPAddress[1] <> #0 then
          DHCPIP := DHCPServer.IPAddress
        else
          DHCPIP := NULL_IP;

        List.Add( Descr );
        List.Add( Format(
          ' %8.8x|%6s|%16s|%2d|%16s|%16s|%16s' ,
          [Index, AdaptTypes[aType],
          MacAddr2Str( TMacAddress( Address ), AddressLength ),
            DHCPEnabled, LocalIP, GatewayIP, DHCPIP] )
            );
        List.Add( '    ' );
        P := Next;  //  TIP_ADAPTER_INFO(P^).Next  points to next entry
      end // with
    end // while
  else
    List.Add( SysErrorMessage( Error ) );
  Dispose( AdapterInfo );
end;



//-----------------------------------------------------------------------------
{ записуємо час завантаження трафіка мережі IP }
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT := -1; // Розташування BAD_HOST_NAME,й.. т.і.
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;



//-----------------------------------------------------------------------------
{ ARP-таблиця записує відношення між IP та MAC-адресам.
}
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetIPNetTable( PTMIBIpNetTable( pBuf ), @TableSize, false );
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-cache empty.'  );
    EXIT;
  end;
  // заносимо у таблицю
```

```
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then //
    begin
      inc( pBuf, SizeOf( DWORD ) ); // записуємо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s| %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we _must_ restore pointer!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPNetRow ) );
  FreeMem( pBuf );

end;


//----------------------------------------------------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо наступний запис з
навантаженістю мережного системи кібербезпеки для протидії атаці на протокол SS7
        with TCPRow do
        begin
```

```
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s|%15s : %-7s| %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) = -1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;


//---------------------------------------------------------------------------
// Опитуємо відкриті підключення до мережі
procedure Get_OpenConnections( List: TList );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
  CStat         : PTTcpConnStatus;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо наступний запис з
навантаженістю мережного системи кібербезпеки для протидії атаці на протокол SS7
        with TCPRow do
          if dwState in [2,5] then   //
          begin
            New( CStat );
            CStat^.LocalIP   := IPAddr2Str( dwLocalAddr );
            CStat^.LocalPort := Port2Svc( Port2Wrd( dwLocalPort ));
            if dwRemoteAddr <> 0 then
```

```
            begin
              CStat^.RemoteIP     := IPAddr2Str( dwRemoteAddr );
              CStat^.RemotePort   := Port2Svc( Port2Wrd( dwRemotePort ));
            end
            else begin
              CStat^.RemoteIP   := ' ...' ;
              CStat^.RemotePort := ' ...' ;
            end;
            CStat^.Status       := TCPConnState[dwState];
            List.Add( CStat );
          end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;



//---------------------------------------------------------------------------
//Записуємо статистику трафіка по TCP
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі  :'  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мініміальний час          :'  + IntToStr( dwRTOMin ) + '  ms'
);
      List.Add( ' Максимальний час          :'  + IntToStr( dwRTOMax ) + '  ms'
);
      List.Add( ' Максимальна кількість підключень :'  + IntToStr(
dwRTOAlgorithm ) );
      List.Add( ' Активні підключення             :'  + IntToStr( dwActiveOpens
) );
      List.Add( ' Пасивні підключення            :'  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Помилка невдалого відкриття     :'  + IntToStr( dwAttemptFails
) );
      List.Add( ' Скидання встановленого підключення  :'  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.:'  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отриманий сегмент        :'  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлений сегмент            :'  + IntToStr( dwOutSegs ) );
      List.Add( ' Переправлений сегмент   :'  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилка входження         :'  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виходу          :'  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні зв'язки    :'  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );

end;

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

//Запис часу та завантаженості системи кібербезпеки для протидії атаці на
протокол SS7 по UDP
procedure Get_UDPTable( List: TStrings );
```

```
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );

  // заносимо у таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо наступний запис з
навантаженістю мережного системи кібербезпеки для протидії атаці на протокол SS7
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ' без даних.'  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------------------------------------------------
procedure Get_IPAddrTable( List: TStrings );

//Запис часу та завантаженості системи кібербезпеки для протидії атаці на
протокол SS7 по IP

var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
```

```
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x|%15s|%15s|%15s|%8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;


(*
//---------------------------------------------------------------------------
//Запис IP адресів до MIB

procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
```

```
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;
*)

//----------------------------------------------------------------------------
{ отримуємо данні з таблиць маршрутизації }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize, true
    );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // заносимо у таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize, true
    );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
          List.Add( Format(
            ' %15s|%15s|%15s|%8.8x|%7s|   %5.5d|    %7s|        %2.2d' ,
           [IPAddr2Str( dwForwardDest ),
           IPAddr2Str( dwForwardMask ),
             IPAddr2Str( dwForwardNextHop ),
             dwForwardIFIndex,
             IPForwTypes[dwForwardType],
             dwForwardNextHopAS,
             IPForwProtos[dwForwardProto],
             dwForwardMetric1
             ] ) );
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;
```

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

// Надання статистики по IP
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Пересилання дозволено       : '  + ' Так'  )
      else
        List.add( ' Пересилання дозволено       : '  + ' Ні'  );
      List.add( ' Вбудований TTL              : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримана датаграмма       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилки заголовку     (У)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Помилка адреси    (У)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Невідомий протокол  (У)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Відхілені датаграми     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграми встановлені    : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запит           : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрут відхилений      : '  + inttostr( dwRoutingDiscards )
);
      List.add( ' Немає маршруту          (Out): '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебирання часу     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Перебирання запитів     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація:  '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації    : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграму фрагментовано    : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут у таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//---------------------------------------------------------------------------
// Надання статистики по UDP

procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (У)     : '  + inttostr( dwInDatagrams ) );
      List.add( ' Датаграми (З)    : '  + inttostr( dwOutDatagrams ) );
      List.add( 'Немає портів           : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилки     (У)     : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
```

```
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//---------------------------------------------------------------------------
// Надання статистики по ICMP

procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Повідомлення прийнято    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка               : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами     : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит       : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo повтор      : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );

      ICMPIn.Add( ' Запит адреси маски : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор адреси маски   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats^.OutStats do
    begin
      ICMPOut.Add( 'Повідомлення відправлено: '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка               : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Exo запит       : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Exo повтор      : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPOut.Add( 'Запит адреси маски: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Повтор адреси маски   : '  + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

//---------------------------------------------------------------------------
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
```

```
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

unit IPHLPAPI – бібліотека API функцій для роботи з IP мережею

```
interface
uses
  Windows, winsock;
const

  VERSION       = '1.3';


const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // arb.
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // arb.
  DEFAULT_MINIMUM_ENTITIES = 32; // arb.
  MAX_HOSTNAME_LEN = 128; // arb.
  MAX_DOMAIN_NAME_LEN = 128; // arb.
  MAX_SCOPE_ID_LEN = 256; // arb.

 // Типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] =
    ( 'UNKNOWN', 'BROADCAST', 'PEER_TO_PEER', '', 'MIXED', '', '', '', 'HYBRID'
      );

 // Типи адаптерів
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;
 //
  AdaptTypes    : array[0..6] of string[10] =
    ( 'інший', 'ethernet', 'tokenring', 'FDDI', 'PPP', 'loopback', 'SLIP' );


  MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
  MAXLEN_PHYSADDR = 8; { iprtrmib.h }
  MAXLEN_IFDESCR = 256; { --"---     }

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

type
  TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

 //------Структура IP-адрес -------------------------------------------------

  PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
  TIP_ADDRESS_STRING = array[0..15] of char; //  IP в xxx.xxx.xxx.xxx рядок
  //
  PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
  TIP_ADDR_STRING = packed record //
    Next: PTIP_ADDR_STRING;
    IpAddress: TIP_ADDRESS_STRING;
    IpMask: TIP_ADDRESS_STRING;
    Context: DWORD;
  end;
```

```
//----------Fixed Info структура-----------------------------------------------

  PTFixedInfo = ^TFixedInfo;
  TFixedInfo = packed record
    HostName: array[0..MAX_HOSTNAME_LEN + 4] of char;
    DomainName: array[0..MAX_DOMAIN_NAME_LEN + 4] of char;
    CurrentDNSServer: PTIP_ADDR_STRING;
    DNSServerList: TIP_ADDR_STRING;
    NodeType: UINT;
    ScopeID: array[0..MAX_SCOPE_ID_LEN + 4] of char;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

//----------INTERFACE структура-------------------------------------------------


  PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
    dwAdminStatus: DWORD;
    dwOperStatus: DWORD;
    dwLastChange: DWORD;
    dwInOctets: DWORD;
    dwInUcastPkts: DWORD;
    dwInNUCastPkts: DWORD;
    dwInDiscards: DWORD;
    dwInErrors: DWORD;
    dwInUnknownProtos: DWORD;
    dwOutOctets: DWORD;
    dwOutUCastPkts: DWORD;
    dwOutNUCastPkts: DWORD;
    dwOutDiscards: DWORD;
    dwOutErrors: DWORD;
    dwOutQLen: DWORD;
    dwDescrLen: DWORD;
    bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
  end;

  TMIBIfArray = array of TMIBIFRow;


 //
  PTMibIfTable = ^TMIBIfTable;
  TMibIfTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIfRow;
  end;

//------ADAPTER INFO структура---------------------------------------------------

  TTIME_T = array[1..325] of byte; //

  PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
  TIP_ADAPTER_INFO = packed record
    Next: PTIP_ADAPTER_INFO;
    ComboIndex: DWORD;
    AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;
    Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
    AddressLength: UINT;
    Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;
    Index: DWORD;
```

```
    aType: UINT;
    DHCPEnabled: UINT;
    CurrentIPAddress: PTIP_ADDR_STRING;
    IPAddressList: TIP_ADDR_STRING;
    GatewayList: TIP_ADDR_STRING;
    DHCPServer: TIP_ADDR_STRING;
    HaveWINS: BOOL;
    PrimaryWINSServer: TIP_ADDR_STRING;
    SecondaryWINSServer: TIP_ADDR_STRING;
    LeaseObtained: TTIME_T; //??
    LeaseExpires: TTIME_T; //??
  end;

//----------------TCP структура-----------------------------------------------

  PTMibTCPRow = ^TMibTCPRow;
  TMibTCPRow = packed record
    dwState: DWORD;
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
    dwRemoteAddr: DWORD;
    dwRemotePort: DWORD;
  end;
  //
  PTMibTCPTable = ^TMibTCPTable;
  TMibTCPTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..0] of TMibTCPRow;
  end;
  //
  PTMibTCPStats = ^TMibTCPStats;
  TMibTCPStats = packed record
    dwRTOAlgorithm: DWORD;
    dwRTOMin: DWORD;
    dwRTOMax: DWORD;
    dwMaxConn: DWORD;
    dwActiveOpens: DWORD;
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//---------UDP структура-----------------------------------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE - 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
```

```
    dwNumAddrs: DWORD;
  end;

//-----------IP структура-------------------------------------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
      Unused2: WORD;
  end;

  TMibIPAddrArray = array of TMIBIPAddrRow;


  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPAddrRow;
  end;
```

```
  //
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPForwardRow;
  end;

//--------ICMP-структура-------------------------------------------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;




  //-----------------імпортовано з IPHLPAPI.DLL---------------------------------
  -

function GetAdaptersInfo( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetNetworkParams( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetTcpTable( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetTcpStatistics( pStats: PTMibTCPStats ): DWORD;
```

```
stdcall; external 'IPHLPAPI.DLL';

function GetUdpTable( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetUdpStatistics( pStats: PTMibUdpStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpStatistics( pStats: PTMibIPStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpNetTable( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpAddrTable( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpForwardTable( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

function GetIcmpStatistics( pStats: PTMibICMPInfo ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetRTTAndHopCount( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL;
stdCall; external 'IPHLPAPI.DLL';
function GetIfTable( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetIfEntry( pIfRow: PTMibIfRow ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

implementation
end.
```

unit TrafficUnit – визначення параметрів системи кібербезпеки для протидії атаці на протокол SS7;

```
interface

uses SysUtils, Windows, IPHelper, IPHLPAPI;


type
  TTraffic = Class;

  TNewInstanceEvent = procedure(Sender :TTraffic) of object;
  TFreezeEvent = procedure(Sender :TTraffic) of object;

  TTraffic = Class
  private
    FIP: string;
    FMac: string;
    FInPerSec: Dword;
    FInTotal: Dword;
    FPeakInPerSec: Dword;
    FInterfaceIndex: DWord;
    FActiveCountIn: Dword;
    FSecondsActive: Cardinal;
    FPrevCountIn: DWord;
    FDescription: string;
    FOutTotal: Dword;
    FPeakOutPerSec: Dword;
    FOutPerSec: Dword;
    FPrevCountOut: DWord;
    FActiveCountOut: Dword;
    FAverageInPerSec: Dword;
    FAverageOutPerSec: Dword;
    FStartedAt: TDateTime;
    FRunning: boolean;
    FOnFreeze: TFreezeEvent;
    FOnUnFreeze: TFreezeEvent;
    FConnected: boolean;
    FFound: boolean;
    FSpeed: DWord;

    function GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
  public
    property Found : boolean read FFound write FFound;
    property Connected : boolean read FConnected;
    property Running : boolean read FRunning;
    property InterfaceIndex : DWord read FInterfaceIndex;
    property IP : string read FIP;
    property Mac : string read FMac;
    property Description : string read FDescription;
    property StartedAt : TDateTime read FStartedAt;
    property SecondsActive : Cardinal read FSecondsActive;
    property Speed : DWord read FSpeed;
    property ActiveCountIn : Dword read FActiveCountIn; { }
    property PrevCountIn : DWord read FPrevCountIn; { }
    property InPerSec : Dword read FInPerSec; { байт підраховано в останній
період }
    property AverageInPerSec : Dword read FAverageInPerSec; { У середньому }
    property InTotal : Dword read FInTotal; { загальна кількість байтів }
    property PeakInPerSec : Dword read FPeakInPerSec; { максимальне число
байтів}

    property ActiveCountOut : Dword read FActiveCountOut; { підраховує число
байтів при передачі }
    property PrevCountOut : DWord read FPrevCountOut; { попереднє підрахування
байтів }
    property OutPerSec : Dword read FOutPerSec; { Кількість байтів у останій
період }
    property AverageOutPerSec : Dword read FAverageOutPerSec; { }
```

```
    property OutTotal : Dword read FOutTotal; { Загальна кількість байтів
передано }
    property PeakOutPerSec : Dword read FPeakOutPerSec; { Максимальна кількість
байтів передано }

    procedure NewCycle(const InOctets, OutOctets, TrafficSpeed : Dword);
    procedure Reset;
    procedure Freeze;
    procedure UnFreeze;
    procedure MarkDisconnected;
    function GetStatus : string;
    function FriendlyRunningTime:string;
    constructor Create(const AMibIfRow : TMibIfRow; OnNewInstance :
TNewInstanceEvent);
  published
    property OnFreeze :TFreezeEvent read FOnFreeze write FOnFreeze;
    property OnUnFreeze :TFreezeEvent read FOnUnFreeze write FOnUnFreeze;
  end;

  function BytesToFriendlyString(Value : DWord) : string;
  function BitsToFriendlyString(Value : DWord) : string;

implementation

function BytesToFriendlyString(Value : DWord) : string;
const
 OneKB=1024;
 OneMB=OneKB*1024;
 OneGB=OneMB*1024;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 B',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 KB', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 MB', Value/OneMB)
end;

function BitsToFriendlyString(Value : DWord) : string;
const
 OneKB=1000;
 OneMB=OneKB*1000;
 OneGB=OneMB*1000;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 bps',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 Kbps', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 Mbps', Value/OneMB)
end;

constructor TTraffic.Create(const AMibIfRow: TMibIfRow; OnNewInstance :
TNewInstanceEvent);
var
 Descr: string;
begin
 inherited Create;

 FRunning:=true;
 FConnected:=true;

 self.FInterfaceIndex:=AMibIfRow.dwIndex;
 self.FIP:=GetIPFromIFIndex(self.InterfaceIndex);
```

```
 self.FMac:=MacAddr2Str(TMacAddress(AMibIfRow.bPhysAddr),
AMibIfRow.dwPhysAddrLen);

 SetLength(Descr, Pred(AMibIfRow.dwDescrLen));
 Move(AMibIfRow.bDescr, Descr[1], pred(AMibIfRow.dwDescrLen));
 self.FDescription:=Trim(Descr);

 self.FPrevCountIn:=AMibIfRow.dwInOctets;
 self.FPrevCountOut:=AMibIfRow.dwOutOctets;

 self.FStartedAt:=Now;
 self.FSpeed:=AMibIfRow.dwSpeed;

 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;
  if Assigned(OnNewInstance)
 then OnNewInstance(self);
end;

procedure TTraffic.NewCycle(const InOctets, OutOctets, TrafficSpeed: Dword);
begin
 inc(self.FSecondsActive);
 if not Running
 then Exit;
 FSpeed:=TrafficSpeed;
 // прийнято
 self.FInPerSec:=InOctets-self.PrevCountIn;
 Inc(self.FInTotal, self.InPerSec);
 if InPerSec>0
 then Inc(FActiveCountIn);
 if InPerSec>PeakInPerSec
 then FPeakInPerSec:=InPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageInPerSec:=InTotal div ActiveCountIn
 except
  self.FAverageInPerSec:=0;
 end;
 FPrevCountIn:=InOctets;
 // передано
 self.FOutPerSec:=OutOctets-self.PrevCountOut;
 Inc(self.FOutTotal,self.OutPerSec);
 if OutPerSec>0
 then Inc(FActiveCountOut);
 if OutPerSec>PeakOutPerSec
 then FPeakOutPerSec:=OutPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageOutPerSec:=OutTotal div ActiveCountOut
 except
  self.FAverageOutPerSec:=0;
 end;
 FPrevCountOut:=OutOctets;
end;

function TTraffic.GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
var
 i: integer;
 IPArr: TMIBIPAddrArray;
begin
 Result:='Не знайдено!'; //...
 Get_IPAddrTableMIB(IpArr); // беремо таблицю IP-адрес
 if Length(IPArr)>0
```

```
 then
  for i:=low(IPArr) to High(IPArr) do  // проглядаємо індекси у таблиці...
   if IPArr[i].dwIndex=InterfaceIndex
   then
    begin
     Result:=IPAddr2Str(IParr[i].dwAddr);
     Break;
    end;
end;

procedure TTraffic.Reset;
begin
 self.FPrevCountIn:=InPerSec;
 self.FPrevCountOut:=OutPerSec;
 self.FStartedAt:=Now;
 FSecondsActive:=0;
 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;
end;

procedure TTraffic.Freeze;
begin
 FRunning:=false;
 if Assigned(FOnFreeze)
 then OnFreeze(Self);
end;

procedure TTraffic.UnFreeze;
begin
 FRunning:=true;
 if Assigned(FOnUnFreeze)
 then OnUnFreeze(Self);
end;

procedure TTraffic.MarkDisconnected;
begin
 self.FConnected:=false;
 self.FRunning:=false;
end;
function TTraffic.GetStatus: string;
begin
 if self.Connected
 then Result:='Підключено'
 else Result:='Не підключено';
 if self.Running
 then Result:=Result+', Запущено'
 else Result:=Result+', Не запущено';
end;
function TTraffic.FriendlyRunningTime: string;
var
 H,M,S: string;
 ZH,ZM,ZS: integer;
begin
 ZH:=SecondsActive div 3600;
 ZM:=Integer(SecondsActive) div (60-ZH*60);
 ZS:=Integer(SecondsActive)-(ZH*3600+ZM*60);
 H:=Format('%.2d',[ZH]);
 M:=Format('%.2d',[ZM]);
 S:=Format('%.2d',[ZS]);
 Result:=H+':'+M+':'+S;
end;
end.
```

unit about – підпрограмма про розробників та місце розроблення програми;

```
interface

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

type
  TForm2 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Button1: TButton;
    Label7: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```