МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка

до дипломної роботи
бакалавр
(освітньо-кваліфікаційний рівень)

на тему «Розробка програмного комплексу для пошуку й оцінки уразливостей інформаційних систем»

Виконав: студент 4 курсу, групи ІПЗ-15д
спеціальності 121 „Інженерія програмного забезпечення”
_____ Кирилов В.І.
(підпис)

Керівник,
доцент, д.т.н._____ Лифар В.О.
(підпис)

Рецензент,
доцент, к.т.н._____ Митрохін С.О.
(підпис)

СЄВЄРОДОНЕЦЬК
2019 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 „Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
«___» _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ КИРИЛОВУ В.І.

1. Тема роботи Розробка програмного комплексу для пошуку й оцінки уразливостей інформаційних систем.

керівник роботи Лифар В.О.

затверджені наказом вищого навчального закладу від “__”__2019 року №___
2. Строк подання студентом роботи 20 травня 2019 р.
3. Вихідні дані до роботи

Об'єктом даної роботи є модель обліку уразливостей інформаційних систем.

3.1 Літературні джерела:

Скляров Д.В. Мистецтво захисту й злому інформації / Д.В. Скляров. – Спб.: БхвбПетербург, 2004. – 288 с.

Стивен Норткат, Джуди Новачок. Виявлення вторгнень у мережу. Настільна книга фахівця із системного аналізу. — М.: "Лорі", 2001. – 384 стор. — ISBN 5-8459-0526-5, 0-7357-1265-4

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

4.1 Вступ

4.2 Аналіз предметної галузі (огляд літератури), з висвітленням наступних питань:

Аналіз сучасних мережевих сканерів безпеки.

Інформаційна безпека й стандарти ISO/ IEC.

4.3 Основна частина, в якої висвітлити:

Аналіз уразливості й оцінки ризиків.

Критерії прийняття рішень при невизначеності.

Розробка модуля сканування й аналізу уразливостей.

4.4 Висновки

4.4 Перелік використаних джерел

5. Перелік графічного матеріалу немає

6. Дата видачі завдання 4 лютого 2019 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Приміт ка
1	Одержання завдання на виконання роботи	04.02.19	
2	Укладання і погодження з керівником плану і етапів виконання роботи	20.02.19	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	1.03.19	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	10.03.19	
3	Проектування інфологічної моделі задачі що реалізується.	01.04.19	
5	Укладання та тестування програмного продукту	20.04.19	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	10.05.19	
7	Здача готової пояснювальної записки на кафедру	20.05.19	
8	Укладання доповіді і презентації	01.06.19	

Студент _____
(підпис)

Керівник роботи _____
(підпис)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-15д Кирилов В.І.

Науковий керівник

Доцент, д.т.н. _____ Лифар В.О.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

РЕФЕРАТ

Текст – 53, мал. – 14, додатків –1, літературних джерел –13

У ході виконання даної дипломної роботи була розроблена модель обліку уразливостей інформаційних систем і мереж.

Метою роботи є розробка розробка програмного комплексу, що дозволяє аналізувати параметри інформаційних систем і давати оцінку ступеня їх уразливостей.

В інтегрованім середовищі розробки Embarcadero RAD Studio XE8 розроблений програмний сканер, що одержує ряд параметрів мережного встаткування й операційних систем від аналізованих мережних вузлів. Також розроблений модуль аналізатора, який використовується для оцінки захисту програм і процесів

Отримані результату роботи й програмний засіб аналізу й оцінки уразливостей інформаційних систем можуть бути практично використані для прийняття своєчасних контрзаходів по захисту мереж і інформаційних систем від несанкціонованого доступу й шкідливих дій програм.

Ключові слова: ІНФОРМАЦІЙНА БЕЗПЕКА. СКАНЕР УРАЗЛИВОСТЕЙ. ІНФОРМАЦІЙНА СИСТЕМА. КРИТЕРІЇ ПРИЙНЯТТЯ РІШЕНЬ. ОЦІНКИ РИЗИКІВ.

ЗМІСТ

ВСТУП

Актуальність досліджень. Можливості бізнесу на теперішній час значно розширюються шляхом впровадження інноваційних інформаційних технологій. Однак, нові можливості завжди пов'язані з новими ризиками. Чим більше стає залежність основної діяльності компанії від інформаційних технологій, тим більше зростає ризик здійснення щодо неї різних погроз: наприклад, фінансового шахрайства або розкрадання конфіденційної інформації. У випадку, якщо можливий збиток від потенційних погроз досить великий, необхідно впроваджувати адекватні й економічно виправдані заходи захисту.

Таким чином, у сьогоднішній ситуації підприємствам доцільно мати грамотну політику інформаційної безпеки, яка повинна ґрунтуватися на комплексному підході, здійснювати контроль усіх параметрів і мати підготовку до майбутнього.

Безпека інформації є однієї з найважливіших деталей існування будь-якої організації, але забезпечити абсолютну безпеку даних і надійну роботу автоматизованої інформаційної системи організації не можливо. Однак, створення комплексної, продуманої концепції безпеки, з урахуванням специфіки задач конкретної організації, допоможе звести ризик втрати коштовної інформації до мінімуму.

Об'єкт досліджень: модель обліку уразливостей інформаційних систем.

Предмет досліджень: мережні протоколи TCP, UDP, параметри операційних систем і критерії прийняття рішень.

Ціль дослідження: розробка програмного комплексу, що дозволяє аналізувати параметри інформаційних систем і давати оцінку ступеня їх уразливостей.

Завдання дослідження:

a) розробити модель обліку уразливостей інформаційних систем;
b) розробити програмний сканер, що одержує ряд параметрів мережного встаткування й операційних систем від аналізованих мережних вузлів;

c) розробити програмний аналізатор, використовуваний для оцінки захисту програм і процесів.

Методологічна й теоретична основа дослідження: засобу захисту програмного забезпечення і інформаційна безпека стандарту ISO/ IEC

Практичне значення отриманих результатів. Отримані результату роботи й програмний засіб аналізу й оцінки уразливості інформаційних систем можуть бути практично використані для прийняття своєчасних контрзаходів по захисту мереж і інформаційних систем від несанкціонованого доступу й шкідливих дій програм.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ЗАСОБІВ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз сучасних мережевих сканерів безпеки

Одним з етапів забезпечення інформаційної безпеки є ідентифікація потенційних ризиків і уразливості. Це спеціалізоване апаратне або програмне забезпечення, яке сканує інформаційну систему або мережу і її обладнання на предмет виявлення слабких місць у системі безпеки. Мережні сканери виконують функції:

1. ідентифікація й аналіз уразливостей;
2. інвентаризація ресурсів і обладнання мережі;
3. формування звітів, що містять опис уразливостей.

Сканер локальної мережі — засіб, що забезпечує безпека зберігання й обробки баз даних, конфіденційної інформації.
Сканери мережі необхідні організаціям у сферах оборони, науки, медицини, торгівлі, IT, фінансів, реклами, виробництва, для органів влади й диспетчерських служб — де небажаний витік інформації, є бази персональних даних клієнтів.

Сканери уразливостей мережі при використовують зондування — механізм активного аналізу, який запускає імітації атак, тим самим перевіряючи уразливість. При зондуванні застосовуються методи реалізації атак, які допомагають підтвердити наявність уразливості й виявити сховані проблеми. Другий механізм — сканування — більш швидкий, але дає менш точні результати. Це пасивний аналіз, при якім сканер шукає уразливість без підтвердження її наявності, використовуючи непрямі ознаки.
За допомогою сканування визначаються відкриті порти й збираються пов'язані з ними заголовки. Вони надалі рівняються з таблицею правил визначення мережних обладнань, ОС і можливих проблем мережі. Після порівняння мережний сканер безпеки повідомляє про наявність або відсутність уразливості.

Алгоритм роботи сканерів наступний:

1. Перевірка заголовків. Як перший крок сканування — це оптимальне рішення, що не приводить до порушення роботи мережі.

2. Активні зондувальні перевірки (active probing check). Це сканування, при якім рівняється хэш фрагмента програми із сигнатурою уразливості. Досить швидкий метод, з більшим коефіцієнтом надійності.

3. Імітація атак (exploit check). Це тестування, яке експлуатує дефекти в програмнім забезпеченні. Подається спеціальний імпульс деяким уразливостям, які не помітні до певного моменту. Ефективний метод, але перед його використанням слід зробити бэкап системних настроювань

Розглянемо схему роботи сканерів безпеки мережі:

- збір інформації про мережу, ідентифікація всіх активних обладнань і сервісів;
- виявлення потенційних уразливостей;
- підтвердження обраних уразливостей, для чого моделюються атаки;
- формування звітів;
- автоматичне усунення уразливостей. Даний етап часто зустрічається в сканерах системних.

Одним з головних вимог до сучасних мережних сканерів уразливостей, це підтримка різних операційних систем. Більшість популярних сканерів — кроссплатформені. Сканери мережі досліджують відразу кілька портів, що пропорційна знижує час на тести. Сканер повинен перевірити програмне забезпечення, акцентую увагу на «хіти зломів» продукти Adobe Flash Player, Outlook, різних браузерів. Сучасні сканери дозволяють задати перелік обладнань, що перевіряються, і типів уразливості, указати дозволені для автоматичного відновлення додатки, установити періодичність перевірки й надання звітів.

Аналізатор мережних протоколів — життєве важлива частина набору інструментів мережного адміністратора. Аналіз мережних протоколів — це пошук проблемних місць у мережних комунікаціях. Щоб довідатися, чому яке-небудь мережне обладнання працює саме так, а не інакше, слід

використовувати аналізатор протоколів, щоб відчути трафік і виділити з нього дані й протоколи, що передаються по мережному кабелю. Аналізатор мережних протоколів може використовуватися для:

- локалізації важкорозв'язних проблем;
- виявлення й ідентифікації несанкціонованого програмного забезпечення;
- одержання такої інформації, як базові моделі трафіка (baseline traffic patterns) і метрики утилізації мережі;
- ідентифікації невикористовуваних протоколів для видалення їх з мережі;
- генерації трафіка для випробування на вторгнення (penetration test) з метою перевірки системи захисту;
- роботи із системами виявлення вторгнень Intrusion Detection System (IDS);
- прослуховування трафіка, тобто локалізації несанкціонованого трафіка з використанням Instant Messaging (IM) або бездротових крапок доступу Access Points — (AP);
- вивчення роботи мережі.

При обслуговуванні мережі необхідно мати аналізатор протоколів. Щоб вибрати аналізатор мережних протоколів, який би відповідав конкретному середовищу роботи, розглянемо спочатку деякі типові функції програмного аналізатора протоколів. Після чого будуть вивчені й зіставлені розглянуті функції популярних аналізаторів мережних протоколів.

Більшість аналізаторів мережних протоколів працюють за схемою, представленої на Рис. 1.1, і відображають у деякому початковому виді, однакову базову інформацію. Аналізатор працює на станції хоста. Коли аналізатор запускається в безладному режимі (promiscuous mode), драйвер мережного адаптера, NIC, перехоплює весь минаючий через нього трафік. Аналізатор протоколів передає перехоплений трафік декодеру пакетів аналізатора (packet-decoder engine), який ідентифікує й розщеплює пакети по відповідних до рівнів ієрархії. Програмне забезпечення протокольного аналізатора вивчає пакети й відображає інформацію про них на екрані хоста у вікні аналізатора. Залежно від можливостей конкретного продукту,

представлена інформація може згодом додатково аналізуватися й відфільтровуватися.

Рисунок 1.1 Аналізатор протоколів виконує моніторинг мережного трафіка

Звичайне вікно протокольного аналізатора полягає їх трьох областей, як, наприклад, показано на екрані 1, що демонструє продукт Ethereal. Верхня область відображає підсумкові дані перехоплених пакетів. Звичайно в цій області відображається мінімум полів, а саме: дата; час (у мілі секундах), коли пакети були перехоплені; вихідні й цільові IP-адреси; вихідні й цільові адреси портів; тип протоколу (мережний, транспортний або прикладного рівня); деяка сумарна інформація про перехоплені дані. У середній області показані логічні урізання пакетів, обраних оператором. І нарешті, у нижній області пакет представлений у шістнадцятирічному виді або в символьній формі — ASCII.

Аналіз декодованих пакетів — основна задача будь-якого аналізатора мережних протоколів. Аналізатор організує перехоплені пакети по рівнях і протоколам. Кращі аналізатори пакетів можуть розпізнати протокол по його

найбільш характерному рівню — верхньому — і відобразити перехоплену інформацію. Цей тип інформації звичайно відображається в другій області вікна аналізатора. Наприклад, будь-який аналізатор протоколів може розпізнавати трафік TCP. Гарний аналізатор помітить, що даний трафік породжений Microsoft Exchange Server, що працюють поверх протоколу вилученого виклику процедур, Remote Procedure Call — RPC, і покаже текст поштового повідомлення. Більшість аналізаторів протоколів розпізнають понад 300 різні протоколи й уміють описувати й декодувати їх по іменам. Чим більше аналізатор у стані декодувати інформації й представити її на екрані, тем менше прийде декодувати вручну.

Заяви виробника про те, що його аналізатор протоколів підтримує більш 4000 декодерів протоколів, повинне викликати підозра; 300-400 декодерів — от найбільш реалістичний діапазон. Більшість продуктів підтримує приблизно однакове число декодерів, незважаючи на рекламні заяви окремих постачальників. Наприклад, один продукт може розчленувати простий процес Ping на кілька протоколів (тобто Internet Control Message Protocol — ICMP, Echo Request, ICMP Echo Reply), тоді як іншої буде декодувати той же процес Ping як один протокол, хоча обоє продукту оцінюють і декодують ту саму інформацію.

Загальна проблема при роботі багатьох аналізаторів - неможливість акуратної ідентифікації (а отже, і декодування) протоколу, що використовує порт, відмінний від порту за замовчуванням. Сьогодні всі добре усвідомлюють важливість проблем безпеки, і запуск відомих додатків на мало використовуваних портах є загальноприйнятою практикою захисту від хакерів. Деякі декодери вміють розпізнавати трафік незалежно від того, через який порт він проходить, тоді як інші — ні, і тому просто будуть визначати протокол по його нижньому рівню (тобто TCP або UDP), а це означає, що декодер не представить більш корисної інформації про поля. Деякі аналізатори дозволяють модифікувати декодер, щоб навчитися розпізнавати більше, чим просто порт за замовчуванням для певних протоколів.

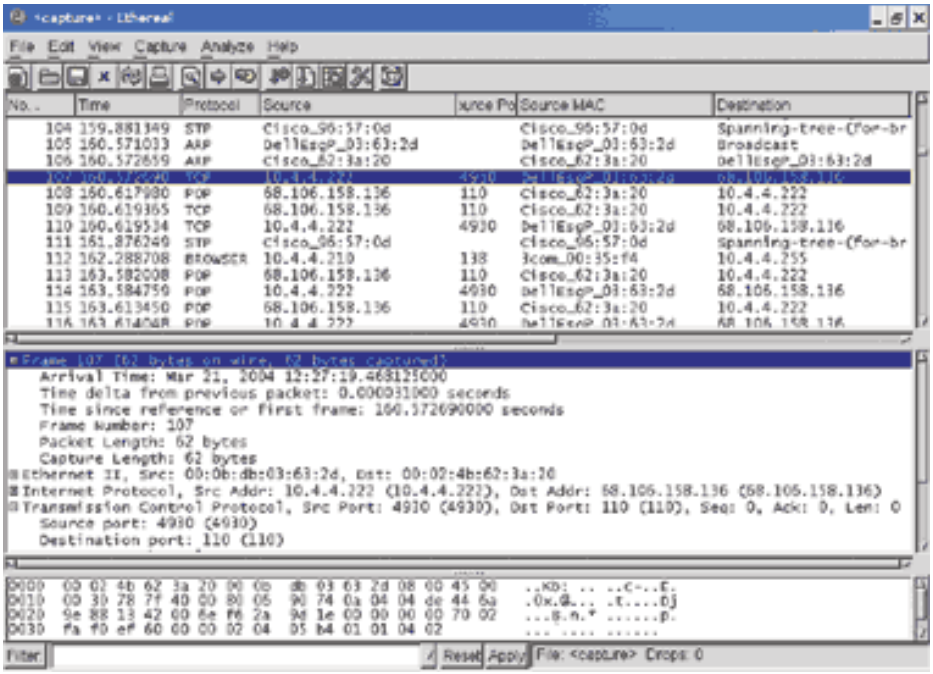

Рисунок 1.2 Приклад вікна аналізатора протоколів

Проведемо порівняння сучасних мережних сканерів безпеки:

1. Mitec Network Scanner — мережний сканер з більшою кількістю різних функцій. Сканує відкриті TCP і UDP порти й показує параметри виявлених мережних обладнань.

2. Advanced IP Scanner — безкоштовний і простій у використанні сканер локальних мереж для Windows.

3. Network Scanner — IP сканер для аналізу мереж. Потужна програма для сканування як великих корпоративних, так і невеликих домашніх мереж з декількома комп'ютерами на наявність усіх доступних загальних ресурсів.

4. Portscan — утиліта для роботи з мережею, дозволяє довідатися МасдАдреса комп'ютера, HTTP, FTP, ім'я хоста, ISMB, ISCSI, SMTP служби.

5. Inssider — сканер Wi-Fi мереж, у режимі реального часу сканує доступні бездротові Wi-Fi мережі в діапазоні від 2,4 до 5 ггц і надає інформацію про них.

6. Advanced Port Scanner — безкоштовний сканер портів на комп'ютері в OS Windows.

7. Find MAC Address — програма для пошуку МасрАдреси на локальних і вилучених комп'ютерах.

8. Lanspy — мережний сканер який допоможе вам довідатися практично все про вилучений комп'ютер — відкриті порти, активні процеси, служби й іншу інформацію.

9. Dipiscan — сканер для відстеження портів і IP адрес у локальній мережі.

10. Nmap — безкоштовний сканер портів, сканує IP мережі в різних режимах і з будь-якою кількістю об'єктів.

11. Softperfect Network Scanner — мультипоточний сканер IP, SNMP, Netbios. Пінгует комп'ютери в локальній мережі, виявляє внутрішні й зовнішні IpeАдреси, апаратний МасдАдреса, відображає які типи ресурсів розподіляються по мережі.

12. Portcheck — консольний додаток для перевірки доступності TCP портів.

13. Domainscan Server Monitoring — мережний монітор для контролю всіх обладнань, доменів і користувачів у вашій мережі.

14. Bping («beeping») — альтернатива утиліті ping в Windows, але зі звуком. Корисний інструмент для системних і мережних адміністраторів, дозволяє перевірити конкретний хост на підключення до мережі TCP / IP.

15. Angry IP Scanner — безкоштовний сканер портів локальної мережі, дозволяє проводити сканування мережі на предмет активних хостов і по заданому діапазону IpcАдрес.

16. IP Tools — набір безкоштовних мережних утиліт для просунутих користувачів.

17. Wi-Fi Scanner — проста програма для пошуку й аналізу бездротових мереж 802.11. Сканер Wi-Fi виявляє бездротові мережі 802.11n і 802.11a/b/g і відображає детальну інформацію про них.

18. Portscanner — безкоштовний сканер який дозволяє сканувати комп'ютер або будь-який хост на наявність відкритих портів по IP адресі або домену.

19. ippulse — програма для моніторингу мережних обладнань, буде корисної мережним адміністраторам.

20. Wifi Hotspot Scanner — сканер для пошуку доступних точок підключення до Wifi.

21. Ethereal розпізнає й декодує такі типи протоколів, як AOL Instant Messenger (AIM), Abstract Syntax Notation One (ASN.1), DNS, FTP, HTTP, Lightweight Directory Access Protocol (LDAP), POP, RPC, Session Initiation Protocol (SIP) і SMTP.

22. Optiview. Optiview — це сімейство продуктів, яке прослуховує трафік у мережах Ethernet, Token Ring і оптоволоконних мережах.

1.2 Інформаційна безпека й стандарти ISO/ IEC

Інформаційна безпека - це захист конфіденційності, цілісності й доступності інформації. Конфіденційність являє собою властивість інформаційних ресурсів, у тому числі інформації, пов'язане з тим, що вони не стануть доступними й не будуть розкриті для неуповноважених осіб. Цілісність є незмінністю інформації в процесі її передачі або зберігання. Доступність – ця властивість інформаційних ресурсів, у тому числі інформації, яка визначає можливість їх одержання й використання на вимогу уповноважених осіб.

Інформаційна безпека - захищеність інформації й підтримуваної інфраструктури від випадкових або навмисних дій природнього або штучного характеру, які можуть нанести неприйнятного збитку суб'єктам інформаційних відносин. Підтримувана інфраструктура - системи електро-, тепло-, вода-, газопостачання і т. і., а також обслуговуючий персонал. Інформаційна безпека організації - стан захищеності інформаційного середовища організації, яке забезпечує її формування, використання й розвиток.

У зв'язку з появою ризиків інформаційної безпеки були прийняті серії міжнародних стандартів (як наприклад серія 27xxx), опубліковані спільно Міжнародною Організацією по Стандартизації (ISO) і Міжнародною Електротехнічною Комісією (IEC). Серії містять кращі практики й рекомендації в області інформаційної безпеки для створення, розвитку й підтримки Системи Менеджменту Інформаційної Безпеки (Рис. 1.3).

Рисунок 1.3 – Модель інформаційної безпеки, заснована на стандартах ISO/IEC (15408, 17799)

1.3 Моделювання, як засіб аналізу погроз і ризиків інформаційної безпеки

Центральним елементом діяльності, яка веде до створення першокласного програмного забезпечення, є моделювання. Моделі дозволяють наочно продемонструвати бажану структуру й поведінка системи. Вони також потрібні для візуалізації й керування її архітектурою. Моделі допомагають досягти кращого розуміння майбутньої системи, привести, по потребі, її спрощення й надають можливість повторного використання. Зрештою, моделі потрібні для мінімізації ризику.

Модель є спрощеною виставою реальності, це схема системи: у неї може входити як детальний план, так і більш абстрактна вистава системи. Гарна модель завжди включає елементи, які істотно впливають на результат, і не включає ті, що мало значать на цьому рівні абстракції. Кожна система може бути описана з різних точок зору. Для цього використовуються різні

моделі, кожна з яких є семантично замкненою абстракцією системи. Модель може бути структурної, що підкреслює організацію системи, або поведінкової, що відображає її динаміку (рис. 1.4).

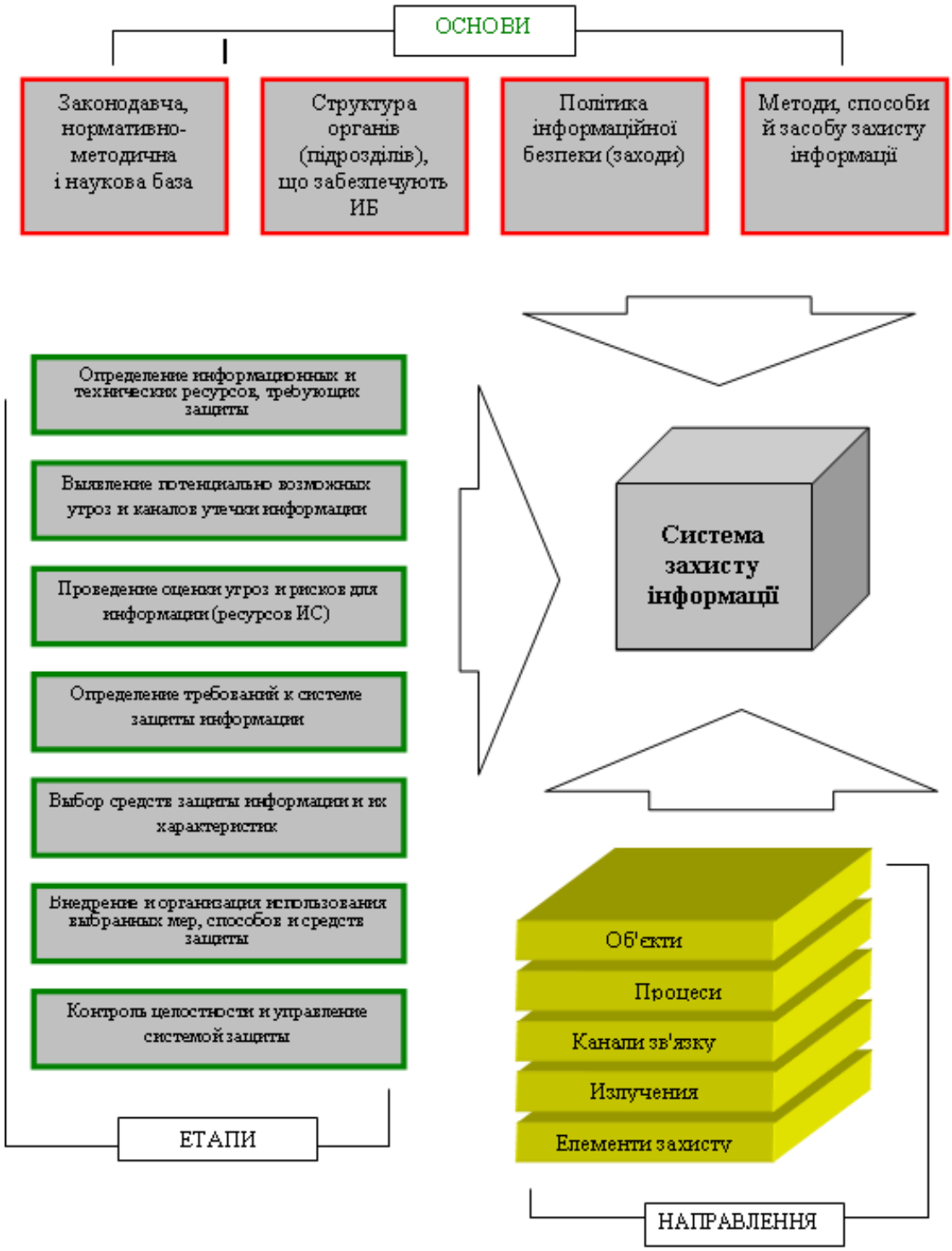

Рисунок 1.4 – Модель побудови системи захисту

При виборі політик інформаційної безпеки, як і при написанні програмного засобу, важливо чітко розуміти структуру організації, для якої

виконується аналіз. Розуміння, у свою чергу, приходить із наочністю, а наочність - це моделювання. Розглянемо як моделювання допоможе захистити інформацію.

Створивши моделі функцій організації, її робочого процесу й безпосередньо структури на рівні відділів (офісів) і т. і., можна наочно побачити їхню взаємодію, зв'язки й таке інше. Саме такі місця, як відомо, і є основними джерелами погроз.

Щоб на практиці підтвердити необхідність моделювання, треба спочатку концептуально розглянути всі переваги й недоліки цього образа.

Становлячи моделі організації для аналізу безпеки інформації, слід пам'ятати, що чим більше сторін проблеми торкнулося з різних точок зору - тем краще. Тому не будемо обмежуватися однієї моделлю.

Після того, як етап моделювання буде завершений - перейдемо до аналізу погроз для кожної, окремо взятої ділянки організації, організації в цілому и створенні політики безпеки для неї.

Тільки після детального розбору цієї методики на прикладі можна приступати до написання програми, яка, у свою чергу, буде аналізувати захід безпеки інформації для більш широкого спектра організацій подібного типу.

РОЗДІЛ 2 ІНФОРМАЦІЙНА МОДЕЛЬ АВТОРИЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЧЕРЕЗ ГЛОБАЛЬНУ МЕРЕЖУ

2.1 Аналіз уразливості й оцінки ризиків

Комп'ютерні системи є недосконалими, вони вразливі для багатьох погроз, а збиток від атак може виявитися значним. Атаки бувають різноманітними: одні порушують конфіденційність або цілісність даних, інші здатні зробити систему недоступної для користувачів. Незважаючи на зниження в порівнянні з минулим роком збитків від атак, величина їх усе ще залишається значної. Згідно даним Computer Security Institute, збитки від вірусів стоять на третьому місці із часток 13%, після крадіжки інформації й атак, що викликають відмову в обслуговуванні (DoS).

Уразливість, як відзначалося вище, асоціюється з порушенням політики безпеки, викликаним неправильно заданим набором правил або помилкою в, що забезпечує безпеку комп'ютера програмі. Варто відзначити, що теоретично всі комп'ютерні системи мають уразливості. Але те, наскільки великий потенційний збиток від вірусної атаки, що використовує уразливість, дозволяє підрозділяти уразливості на активно використовувані й не використовувані зовсім.

MITRE - дослідницька група, фінансована федеральним урядом США, що займається аналізом і дозволом критичних проблем з безпекою. Згідно з термінологією MITRE CVE, уразливість — це стан обчислювальної системи (або декількох систем), яке дозволяє:

- виконувати команди від імені іншого користувача;
- одержувати доступ до інформації, закритої від доступу для даного користувача;
- показувати себе як іншого користувача або ресурс;
- робити атаку типу «відмова в обслуговуванні».

В MITRE уважають, що атака, вироблена внаслідок слабкої або невірно настроєної політики безпеки, краще описується терміном «відкритість» (exposure).

Відкритість — цей стан обчислювальної системи (або декількох систем), яке не є уразливістю, але:

- дозволяє атакуючому робити збір захищеної інформації;
- дозволяє атакуючому приховувати свою діяльність;
- містить можливості, які працюють коректно, але можуть бути легко використані в непорядних цілях;
- є первинною крапкою входу в систему, яку атакуючий може використовувати для одержання доступу або інформації.

Коли зловмисник намагається одержати неавторизований доступ до системи, він робить збір інформації (розслідування) про свій об'єкт, збирає будь-які доступні дані й потім використовує слабість політики безпеки («відкритість») або яку-небудь уразливість.

Аналіз уразливості — необхідний етап у створенні ефективної системи охорони. По його результатах розробляються проектні варіанти технічних комплексів безпеки. Автор пропонує методику створення інженерних моделей і оцінки показників уразливості й ефективності системи захисту. Усі ці відомості допоможуть керівникам вирішувати проблеми забезпечення безпеки об'єктів

Системний підхід — як інструмент оптимізації й зниження ризику помилкових рішень — вимагає, щоб створенню нової або модернізації вже наявної системи передувало обґрунтування проектних і організаційних рішень. Для систем охорони основу такого обґрунтування становить аналіз уразливості об'єкта. Під уразливістю об'єкта розуміється ступінь його незахищеності до впливу порушників. Вона протилежна ефективності охорони (захисту) об'єкта, ступені його захищеності від завдання збитків порушниками.

Аналіз уразливості об'єкта проводиться з метою визначення можливих наслідків впливу порушників на елементи об'єкта, оцінки показників уразливості об'єкта (ефективності охорони), виявлення слабких місць і недоліків існуючої системи охорони або розглянутих проектних варіантів системи, а в підсумку — вибору найкращого варіанта системи охорони для конкретного об'єкта.

Аналіз уразливості об'єкта включає:

- розробку моделі порушників;
- виділення особливо важливих зон об'єкта;
- оцінку показників уразливості;
- визначення слабких місць і недоліків у системі охорони.

2.2 Прийняття рішень в умовах невизначеності

Як відомо, прийняття рішень в умовах ризику характеризується відомим набором значень досліджуваної величини й відповідних їм імовірностей. Невизначеність характеризується тим, що ці ймовірності невідомі, або не можуть бути визначені.

Невизначеність розглядається із двох позицій:

1. як явище невизначеність — це набір нечітких або розмитих ситуацій, взаємовиключної або недостатньої інформації. До явища ставляться й форсомажорні події, які можуть виникнути мимо волі й свідомості конкретного працівника й змінити намічений хід подій.
2. як процес невизначеність — це діяльність некомпетентного працівника, що ухвалює помилкові рішення.

На практиці невизначеність розглядається як єдине ціле, у якім явище створюється процесом, а процес формується явище.

Невизначеності розділяються на дві групи:

Об'єктивні не залежать від керівника або фахівця, що розробляють або реалізують УР, при цьому джерело невизначеностей перебуває поза організацією.

Суб'єктивні виникають через професійні помилки, недогляди, непогодженість. Джерело невизначеностей при цьому перебуває усередині організації.

Предметом теорії рішень в умовах невизначеності є дослідження законів перетворення інформації про стан об'єкта й середовища в інформацію для прийняття рішень, яка характерна для різних об'єктів і суб'єктів (органів) керування.

Основними поняттями (категоріями) теорії прийняття рішень є: система керування, керований об'єкт, суб'єкт керування й прийняття управлінських рішень; інформаційне середовище; стан об'єкта й середовища; рішення, яке ухвалюються; невизначеність і обумовлений нею ризик; функціонал оцінювання (матриця значень функціоналу оцінювання); ситуація прийняття рішень; інформаційна ситуація; джерело інформації, критерії прийняття рішень і таке інше.

Ситуація прийняття рішень характеризується множиною {X, E, F}, де X= {x1,x2,… xm,} - множина рішень об'єкта керування, E = {e1, e2,... en} - множина станів середовища, F= {fij} - функціонал оцінювання (матриця), кожний елемент якої відповідає варіанту ухвалення рішення xi в умовах середовища ej і характеризує ефект (прибуток, дохід, корисність).

У розгорнутій формі ситуація прийняття рішень характеризується матрицею, елементами якої є fij - кількісні оцінки ухваленого рішення xi, за умови, що середовище перебуває в стані ej:

$$F = \begin{pmatrix} & \theta_1 & \dots & \theta_j & \dots & \theta_n \\ x_1 & f_{11} & \dots & f_{1j} & \dots & f_{1n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_k & f_{k1} & \dots & f_{kj} & \dots & f_{kn} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_m & f_{m1} & \dots & f_{mj} & \dots & f_{mn} \end{pmatrix} \quad (1)$$

Творча складова прийняття рішень в умовах невизначеності й складається з таких основних кроків:

1) формування множини рішень X и множини станів середовища E;

2) визначення й формалізація основних показників ефективності й корисності, які входять у функціонал оцінювання F= {fij};

3) визначення інформаційної ситуації, що характеризує альтернативу обігу середовища;

4) вибір критерію прийняття рішень із множини критеріїв, характерних для даної інформаційної ситуації й ЛПР;

5) ухвалення оптимального рішення за вибраним критерієм.

Формальна складова процесу прийняття рішень в умовах невизначеності полягає в проведенні розрахунків по існуючих алгоритмах і показникам ефективності для вибору оптимального варіанта рішення.

У теорії інформації, економіці й менеджменті нерідко доводиться ухвалювати рішення в умовах взаємодії з іншою стороною, яка може переслідувати протилежні або інші мети, використовувати інші шляхи для їхнього досягнення, перешкоджати тими або іншими діями або станами зовнішнього середовища досягненню наміченої мети. Причому дії протилежної сторони можуть носити пасивний або активний характер. Якщо протилежна сторона займає пасивну позицію й активно не протидіє досягненню наміченої мети, то модель називають «грою із природою».

У ситуації, коли протилежна сторона активно протистоїть досягненню наміченої мети, відбувається зіткнення протилежних інтересів, думок, цілей. Такі ситуації називаються конфліктними, і ухвалення рішення кожної зі сторін утрудняється через невизначеність їх поведінки. Незважаючи на таку невизначеність, ухвалювати рішення доводиться кожній стороні.

2.3 Критерії прийняття рішень при невизначеності

Прийняття рішень при невизначеності в припущенні, що ніякі

імовірнісні характеристики не відомі, може базуватися на одному з наступних критеріїв:

• критерій Лапласа, • мінімаксний критерій Вальда,

• критерій Севіджа, • критерій Гурвіца.

Основна відмінність між перерахованими критеріями визначається стратегією поведінки особи, що ухвалює рішення в умовах невизначеності. Ці критерії, незважаючи на їхню кількісну природу, відбивають суб'єктивну оцінку ситуації, у якій доводиться ухвалювати рішення. Не існує загальних правил оцінки застосовності того або іншого критерію, тому що поведінка ЛПР є найбільш важливим фактором при виборі підходящого критерію.

Критерій Лапласа

В основі критерію лежить принцип недостатнього обґрунтування. Це дозволяє розглядати вихідну задачу як задачу ухвалення рішення в умовах ризику, коли всі можливі стани вважаються рівноправними, а їх імовірності рівними один одному, тобто

$$L = \max_i \sum_{j=1}^{n} p_i f_{ij} \qquad p_1 = p_2 = ... = p_n = \frac{1}{n} \tag{2}$$

Максиміний критерій Вальда

Згідно із цим критерієм, у якості оптимальної вибирається та альтернатива, при якій мінімальний виграш максимальний, тобто альтернатива, що гарантує виграш, не менший, ніж максимін:

$$W = \max_i \min_j f_{ij} \tag{3}$$

Даний критерій орієнтує ЛПР на найгірші умови й рекомендує вибирати ту альтернативу, для якої в гірших умовах виграш максимальний. Такий підхід може бути продиктований тільки крайнім песимізмом в оцінці обстановки—«завжди треба розраховувати на гірше». Критерій Вальда часто так і називають-критерій крайнього песимізму.

Критерій мінімаксного ризику Севіджа

Цей критерій рекомендує в умовах невизначеності вибирати ту альтернативу, при якій величина ризику ухвалює найменше значення в самій несприятливій ситуації, тобто таку, яка гарантує мінімум максимального ризику:

$$s = \min_i \max_j r_{ij} \quad r_{ij} = \max_i f_{ij} - f_{ij} \tag{4}$$

Сутність цього критерію в тому, щоб будь-якими шляхами уникнути великого ризику при ухваленні рішення.

Критерій Севіджа, як і критерій Вальда, оце критерій крайнього песимізму, але тільки песимізм тут проявляється в іншому: гіршим уважається не мінімальний виграш, а максимальний ризик — максимальна втрата виграшу в порівнянні з тем, чого можна було б досягти в даних умовах.

Критерій Гурвіца, або критерій « оптимізму-песимізму», на відміну від попередніх, які орієнтують дослідника на найгірші умови, дозволяє зважити як найкращі, так і найгірші умови. Передбачається, що найгірші умови можуть бути з імовірністю а, а найгірші – з імовірність 1-а. Цей критерій має вигляд:

$$H = \max_i [a \min_j f_{ij} + (1-a) \max_j f_{ij}] \tag{5}$$

У критерії Гурвіца ймовірність найгірших умов задається, як правило, суб'єктивно виходячи їх досвіду з урахуванням аналогічних ситуацій.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМИ ПОШУКУ Й ОЦІНКИ УРАЗЛИВОСТЕЙ ІНФОРМАЦІЙНИХ СИТЕМ

3.1 Розробка модуля сканування уразливостей

З метою перевірки настроювання системи безпеки проти неавторизованого вторгнення нами була розроблена програма, що збирає дані для аналізу уразливостей, скріншот якої наведений на Рис. 3.1.

Початковий код створений засобами середовища розробки Embarcadero RAD Studio XE. Середовище допомогло полегшити створення діалогових вікон, а також організувати для програми зручний і зрозумілий для сучасного користувача інтерфейс (Рис. 3.1-3.8).

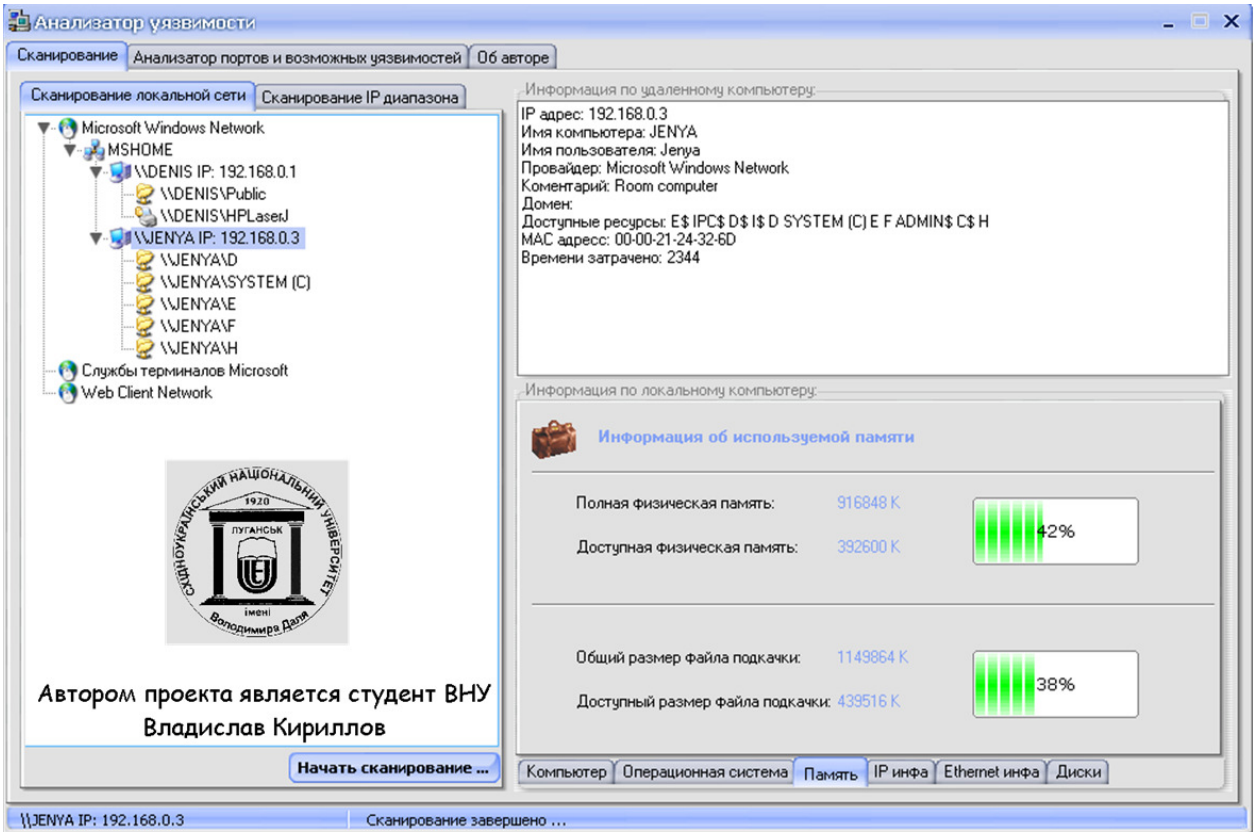

Рисунок 3.1 Головне вікно програми

Програма розроблена, виходячи з вищенаведених моделей і даних, і являє собою мережний сканер, що служить для здійснення діагностики й моніторингу мережних комп'ютерів на предмет виявлення можливих проблем у системі безпеки. Програма виконує перевірку мережі, у пошуках потенційних лазівок, якими може скористатися зловмисник при проведенні атаки й, таким чином, дозволяє довідатися уразливість і відкритість системи до того, як вони будуть виявлені зловмисником.

Програма сканує мережні хости в кілька етапів:

1. сканування локальної мережі на предмет доступних обладнань і дисків;
2. визначення операційної системи й мережних настроювань на хостах мережі;
3. визначення використовуваної оперативної пам'яті на хостах;

4. одержання інформації дискових накопичувачах на хостах;
5. одержання інформації про мережний адаптер і його настроюваннях;
6. одержання інформації про активні служби, сервіси й запущених додатках.

Програма дозволяє переглядати мережні ресурси із загальним доступом, відкриті порти, запущені процеси й модулі, одержувати подробиці про завантаження пам'яті, вилучені адреси й стану з'єднань, імена DNS-серверів, сервіси, пов'язані з тем або іншим з'єднанням і використовуваними ними портами, також дозволяє одержати інформацію про архітектуру й установленої ОС комп'ютера з пакетами виправлень, про мережний адаптер і його настроюваннях.

Розглянемо поетапно особливості модуля програми, який сканує.

1-й етап - сканер структури директорій і ресурсів з відкритим доступом (Рис. 3.2). Виконує сканування всіх IP-адрес у мережі й дозволяє здійснювати пошук і аналіз директорій і ресурсів, доступних для перегляду й запису, дозволяє легко виявити доступні обладнання в локальній мережі користувача. Усі дані витягають автоматично й містять докладну інформацію про комп'ютери й інших обладнаннях, знайдених у локальній мережі й про відкриті порти на цих обладнаннях.

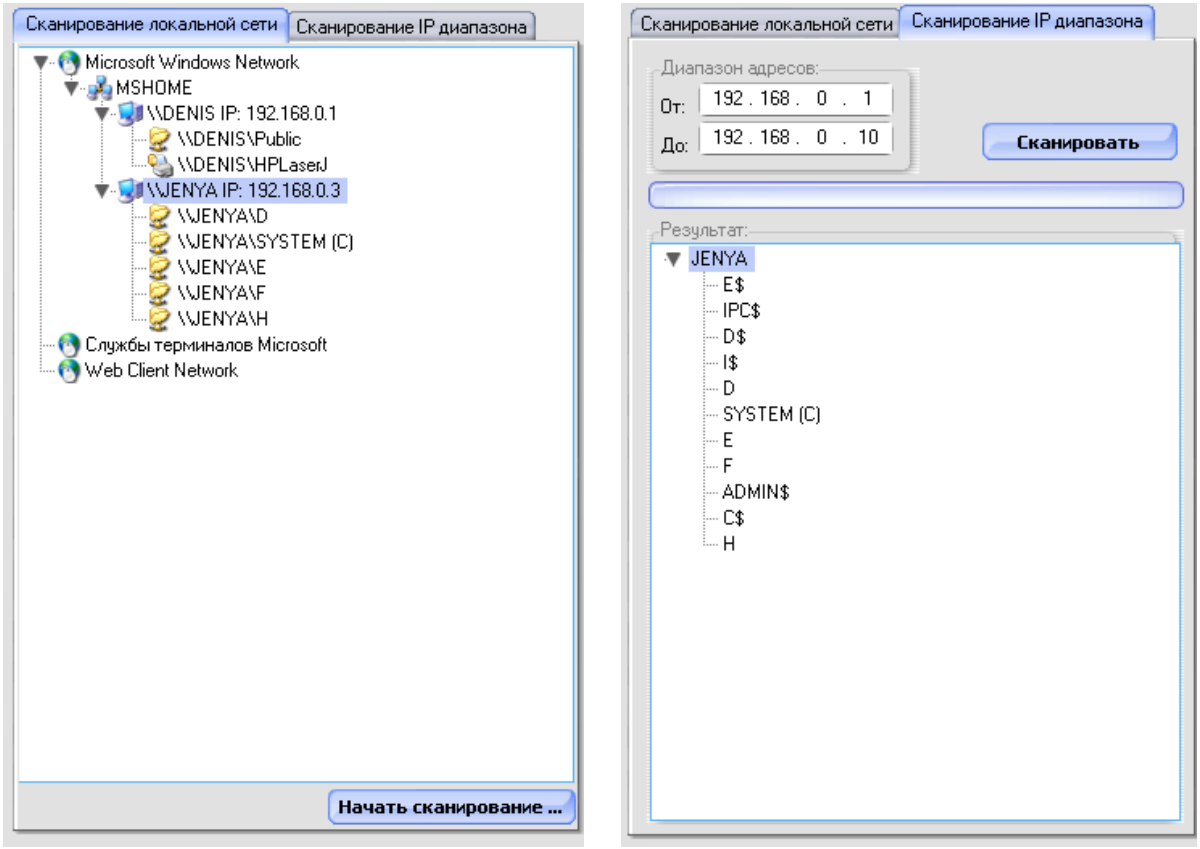

Рисунок 3.2 Вікно сканера локальної мережі й діапазону IP адрес

Сканером можна віддалено перевірити, чи є конкретне обладнання вразливим або ж шкідливим для мережі. Сканери IP-адрес в LAN дозволяють відслідковувати всі обладнання й усувати будь-які підозрілі елементи, які можуть виявитися шкідливими для безпеки мережі й баз даних і тим самим дає можливість знаходити слабкі місця в конфігурації LAN (Рис. 3.3).

Рисунок 3.3 Вікно сканера локальної мережі й діапазону IP адрес

2-й етап - аналізатор операційної системи, і мережних настроювань (Рис.3.4).

Аналізатор операційної системи одержує докладні дані про всі виявлені мережні обладнання:

1. тип ОС і її архітектура, постачальник ПК;
2. відкриті порти і їх статус;
3. використовуваний процесор і його постачальник;
4. список устаткування й програм.

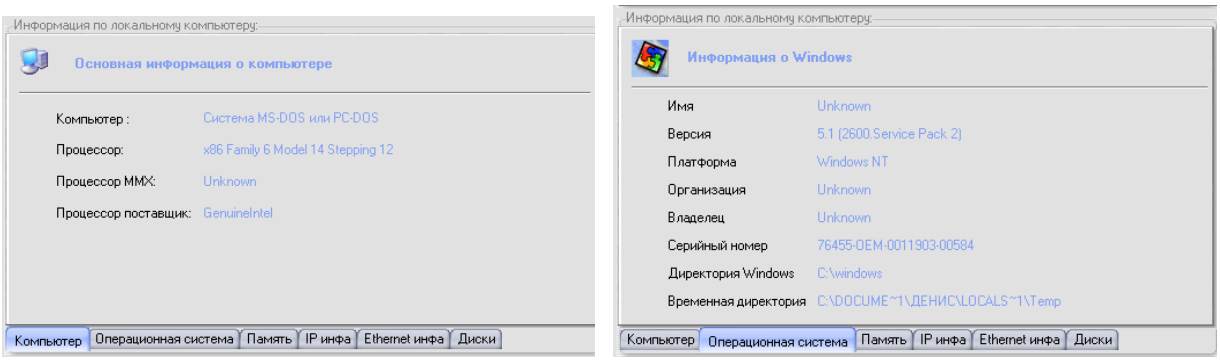

Рисунок 3.4 Одержання інформації про комп'ютер і встановленої ОС

3-й етап - аналізатор фізичної пам'яті (Рис.3.5), що дозволяє одержати дані про доступну фізичну пам'ять, доступний розмір файлу підкачування.

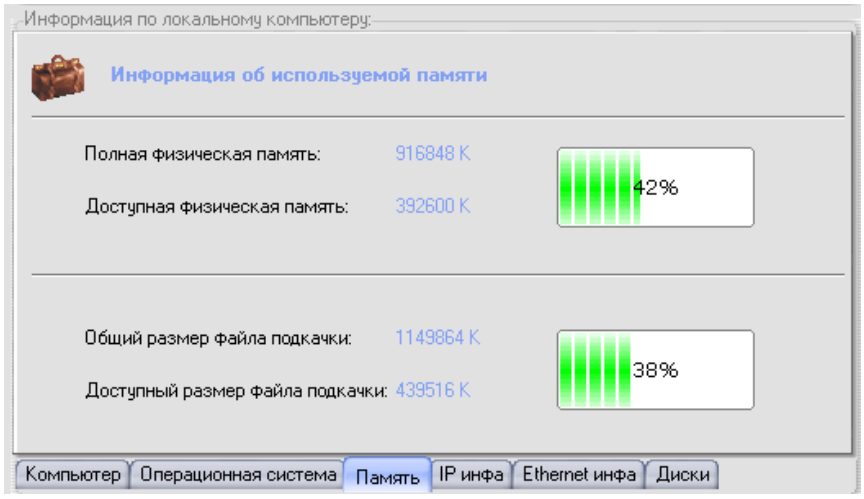

Рисунок 3.5 Одержання інформації про використовувану пам'ять

4-й етап - аналізатор дискових накопичувачів (Рис.3.6), що дозволяє одержати дані про серійні номери дисків, використовуваній файловій системі, параметри секторів у кластері.

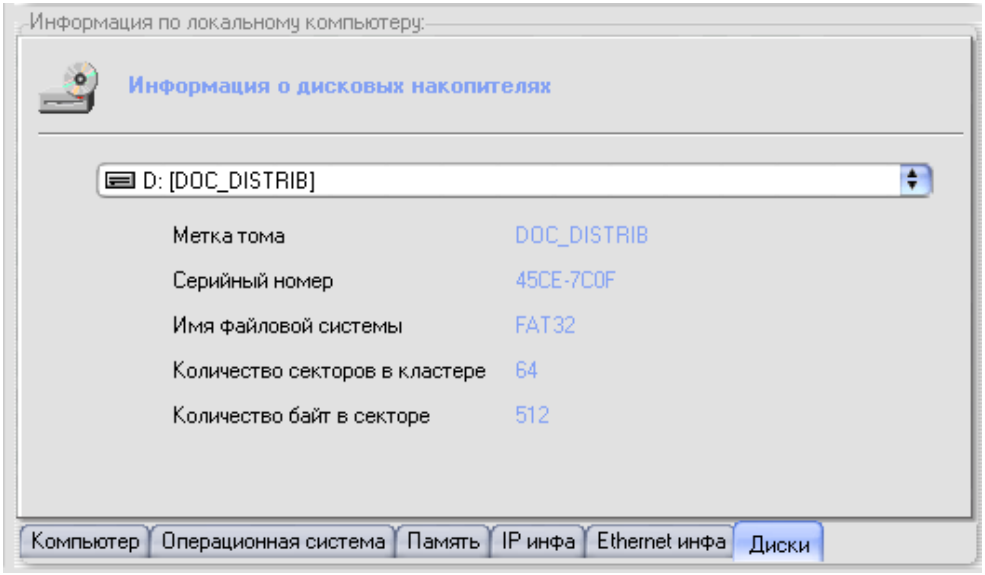

Рисунок 3.6 Одержання інформації про дискові накопичувачі

5-й етап - аналізатор мережного адаптера і його настроювань (Рис.3.7), що дозволяє одержати дані про встановлені мережні інтерфейси, мережні адаптери, одержати MAC адреси всіх адаптерів, IP адреси.

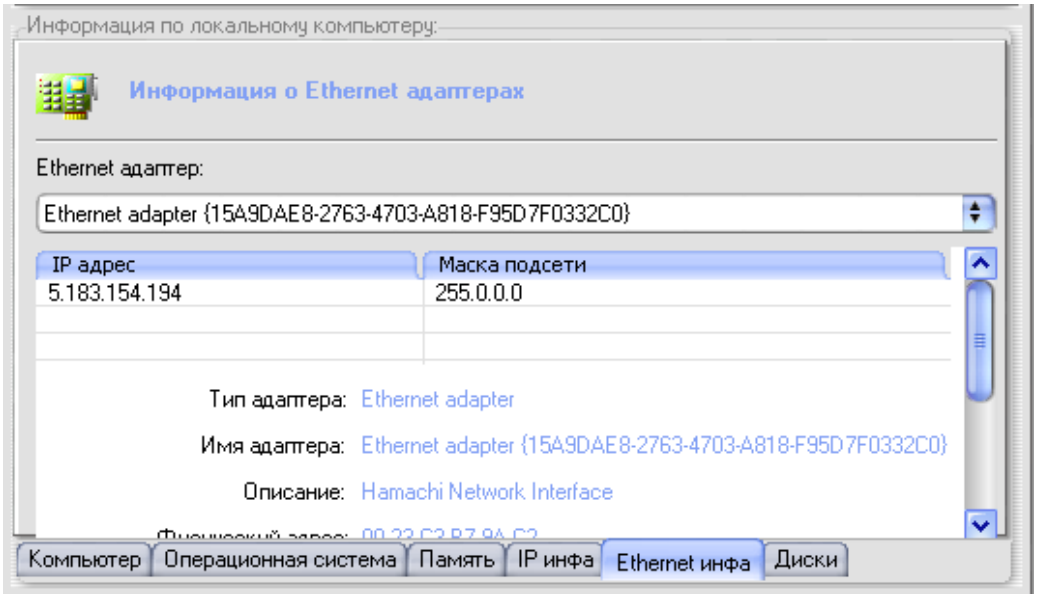

Рисунок 3.7 Одержання інформації про мережний адаптер і його настроюваннях

6-й етап - монітор портів і активних служб (Рис.3.8), що дозволяє одержати дані про активні служби, сервіси й додатках, запущених у мережі, з висновком використовуваних портів й хостів і імені процесу.

Монітор дозволяє проводити аналіз по TCP і UDP протоколам.

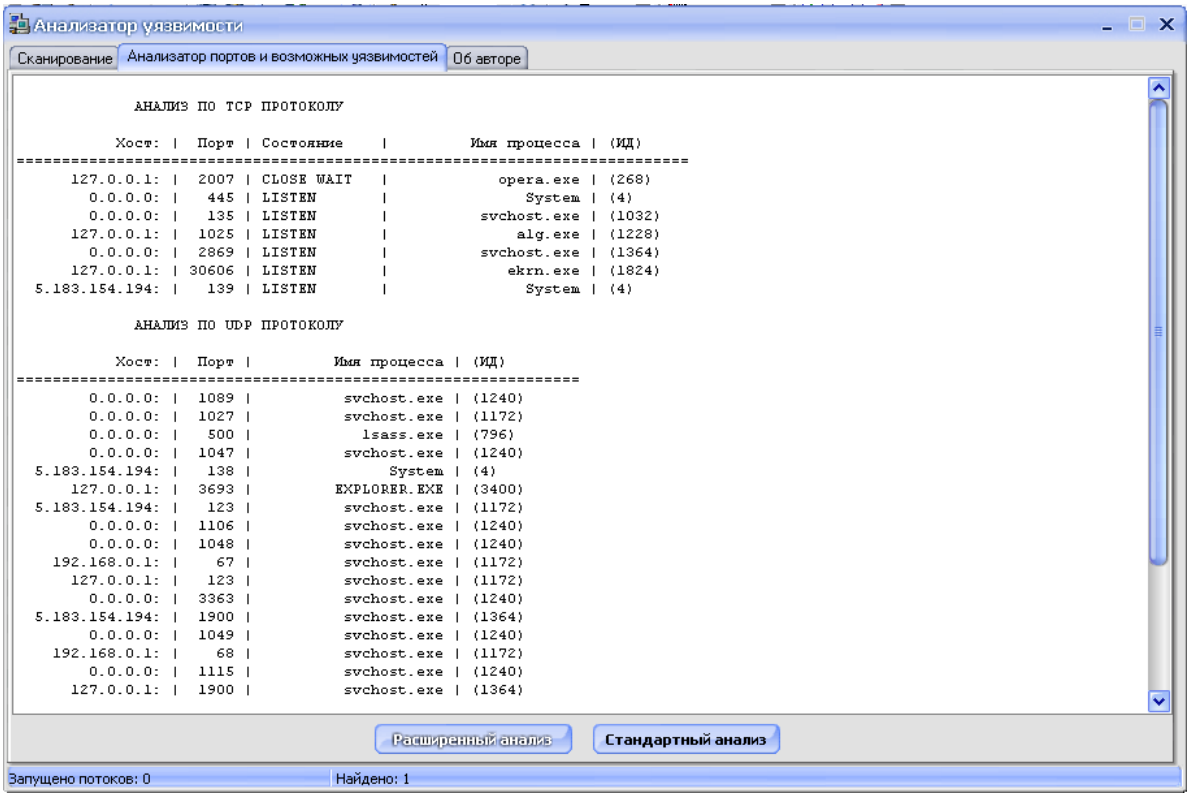

Рисунок 3.8 Одержання інформації про активні служби

Отримана програмою - аналізатором інформація за 6 етапів дозволять прийняти своєчасні контрзаходи по захисту мережі, наприклад, закрити невикористовувані порти, заборонити загальний доступ або встановити останній пакет виправлень.

3.2 Розробка модуля аналізу уразливостей

Критерії прийняття рішень, незважаючи на їхню кількісну природу, відбивають суб'єктивну оцінку ситуації, у якій доводиться ухвалювати рішення. Розглянемо програмну реалізацію критеріїв, використовуваних у програмі .

1. Критерій Лапласа

```
procedure Tform1.Radiobutton1Click(Sender: Tobject);
var   i,j,i_max,i_min:Integer;   ar: array [1..7,1..4] of real;
       str : string;  res1_max,res2_max,res1_min,res2_min:real;
begin
 for i := 1 to 7 do     for j:=1 to 4 do begin
     str:=replace(form1.Advstringgrid4.Cells[j-1,i],'.',',');
     ar[i,j]:=Strtofloat(str);
  end;
 res1_max:=0; res1_min:=1000;   i_min:=0; i_max:=0;
 for i := 1 to 7 do begin
   res2_max:=0; res2_min:=0;
   for j:=1 to 4 do begin
     res2_max:=res2_max+ar[i,j];    res2_min:=res2_min+ar[i,j];
   end;
   if res2_max>=res1_max then begin
    res1_max:=res2_max; i_max:=i;
   end;
   if res2_min<=res1_min then begin   res1_min:=res2_min; i_min:=i;
 end;
 Label50.Caption:=' За критерієм Лапласа визначені ступені захисту:
  'найбільший захист для '+'"'+ Advstringgrid2.Cells[0,i_max]
  'найменший захист для '+'"'+Advstringgrid2.Cells[0,i_min]+'"';
end;
```

2. Максіминний критерій Вальда

```
procedure Tform1.Radiobutton2Click(Sender: Tobject);
var   i,j,i_max,i_min:Integer;  ar: array [1..7,1..4] of real;
       ar1: array [1..7] of real;   str : string;
       res1_max,res2_max,res1_min,res2_min:real;
begin
 for i := 1 to 7 do   for j:=1 to 4 do begin
      str:=replace(form1.Advstringgrid4.Cells[j-1,i],'.',',');
      ar[i,j]:=Strtofloat(str);
    end;
 for i := 1 to 7 do begin
   ar1[i]:=1000;  for j:=1 to 4 do if ar1[i]>ar[i,j]then ar1[i]:=ar[i,j]
 end;
 i_max:=1; i_min:=1;   i_min:=0; i_max:=0;
 res1_max:=0; res1_min:=1000;
 for i := 1 to 7 do begin
   if ar1[i]>=res1_max then  res1_max:=ar1[i];  i_max:=i;
   if ar1[i]<=res1_min then res1_min:=ar1[i];    i_min:=i;
 end;
 if i_max<>i_min then
  Label50.Caption:=' За критерієм Вальда визначені ступені захисту:
       'найбільший захист для '+""+Advstringgrid2.Cells[0,i_max]+
       'найменший захист для '+""+Advstringgrid2.Cells[0,i_min]+""
end;

 3.  Критерій мінімаксного ризику Севиджа
procedure Tform1.Radiobutton3Click(Sender: Tobject);
var    i,j,i_max,i_min:Integer;  ar: array [1..7,1..4] of real;
       ar1: array [1..7] of real;  str : string;
 begin
 for i := 1 to 7 do     for j:=1 to 4 do
      str:=replace(form1.Advstringgrid4.Cells[j-1,i],'.',',');
 for i := 1 to 7 do begin
   ar1[i]:=0;  for j:=1 to 4 do if ar1[i]<ar[i,j]then ar1[i]:=ar[i,j]
```

```
 end;
 for i := 1 to 7 do  for j:=1 to 4 do ar[i,j]:=ar1[i]-ar[i,j];
 for i := 1 to 7 do begin
   ar1[i]:=0;   for j:=1 to 4 do if ar1[i]<ar[i,j]then ar1[i]:=ar[i,j]
 end;
 i_min:=0; i_max:=0;   res1_max:=0; res1_min:=1000;
 for i := 1 to 7 do begin
   if ar1[i]>res1_max then   res1_max:=ar1[i];   i_max:=i;
   if ar1[i]<res1_min then  res1_min:=ar1[i];  i_min:=i;
 end;
 Label50.Caption:=' За критерієм Сэвиджа визначені ступені захисту:
        'найбільший захист для '+""+Advstringgrid2.Cells[0,i_max]+
       'найменший захист для '+""+Advstringgrid2.Cells[0,i_min]+"";
end;
```

4. Критерій Гурвіца

```
procedure Tform1.Radiobutton4Click(Sender: Tobject);
var  i,j,i_max,i_min:Integer;  ar: array [1..7,1..4] of real;
 ar1,ar2,ar3: array [1..7] of real;  str : string;
begin
 for i := 1 to 7 do  for j:=1 to 4 do begin
     str:=replace(form1.Advstringgrid4.Cells[j-1,i],'.',',');
     ar[i,j]:=Strtofloat(str);
  end;
  //max
   for i := 1 to 7 do begin   ar1[i]:=0;
    for j:=1 to 4 do if ar1[i]<ar[i,j]then ar1[i]:=ar[i,j] end;
  //min
  for i := 1 to 7 do begin     ar2[i]:=1000;
    for j:=1 to 4 do if ar2[i]>ar[i,j]then ar2[i]:=ar[i,j]  end;
 for i := 1 to 7 do ar3[i]:=a*ar2[i]+(1-a)*ar1[i];
 i_min:=0; i_max:=0;  res1_max:=0; res1_min:=1000;
 for i := 1 to 7 do begin    if ar3[i]>res1_max then  res1_max:=ar3[i];
```

```
if ar3[i]<res1_min then  res1_min:=ar3[i];  i_min:=i;
end;
Label50.Caption:=' За критерієм Гурвіца визначені ступені захисту:
    'найбільший захист для '+""+Advstringgrid2.Cells[0,i_max]+
    'найменший захист для '+""+Advstringgrid2.Cells[0,i_min]+"";
end;
```

Програмна реалізація критеріїв 1-4 увійшла в модуль оцінки захисту програм і процесів. Для роботи модуля аналізу уразливостей необхідно на формі активувати вкладку «Оцінка захисту програм і процесів» (Рис. 3.9).

Рисунок 3.9 Одержання інформації про активні служби

Програма робить ухвалення оптимального рішення про уразливість і визначає ступінь захисту за обраним критерієм. Такими критеріями можуть бути критерій Лапласа, мінімаксний критерій Вальда, критерій Севіджа, критерій Гурвіца.

Для постановки оцінки обраним критерієм користувачеві необхідно:

1. увести елементи матриці кількісних оцінок ухваленого рішення;
2. увести стани середовища (рівні 1-4);
3. увести коефіцієнти етапів.

Після того, як усі дані будуть уведені користувачем і активована кнопка «Виконати перерахування матриці», матриця рішень буде перелічена з обліків вагових коефіцієнтів рівнів і запропонованих етапів. Вибір опції критерію ухвалення рішення визначає ступінь захисту по заданих етапах (Рис. 3.10).

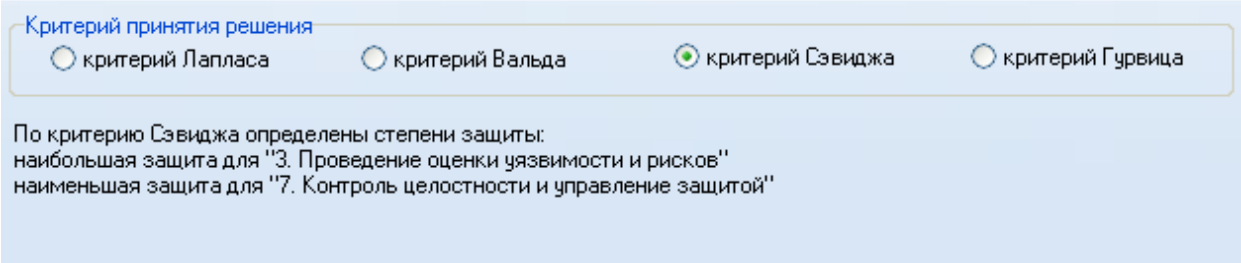

Рисунок 3.10 Одержання інформації про активні служби

Отримана модулем аналізу уразливості інформація дозволять вжити своєчасних заходів по усуненню менш захищених рівнів в інформаційній системі.

ВИСНОВКИ

1. У ході виконання дипломної роботи була розроблена модель обліку уразливостей інформаційних систем і мереж.
2. В інтегрованім середовищі розробки Embarcadero RAD Studio XE8 розроблений програмний сканер, що одержує ряд параметрів мережного встаткування й операційних систем від аналізованих мережних вузлів.
3. Розроблений програмний аналізатор, який використовується для оцінки захисту програм і процесів.
4. Отримані результату роботи й програмний засіб аналізу й оцінки уразливостей інформаційних систем можуть бути практично використані для прийняття своєчасних контрзаходів по захисту мереж і інформаційних систем від несанкціонованого доступу й шкідливих дій програм.

СПИСОК ЛІТЕРАТУРИ

1. Варлатая С.К. Програмно-апаратний захист інформації: учебн. посібник / С.К. Варлатая, М.В. Шаханова. – Владивосток: ИздДУ ДВГТУ, 2007. – 318 с.

2. Скляров Д.В. Мистецтво захисту й злому інформації / Д.В. Скляров. – Спб.: БхвбПетербург, 2004. – 288 с.

3. Віртуальна машина [Електронний ресурс]. – Режим доступу до ресурсу: http://ru.wikipedia.org/wiki/ Віртуальна_машина.

4. Software as a service [Електронний ресурс]. – Режим доступу до ресурсу: http://ru.wikipedia.org/wiki/Software_as_a_service.

5. Молдовян Н.А. Введення в криптосистеми з відкритим ключем / Н.А. Молдовян, А.А. Молдовян. – Спб.: БхвбПетербург, 2005. – 288 с.

6. Буинцев Д.Н. Метод захисту програмних засобів на основі перетворень, що заплутують: дис. .канд. техн. наук : 05.13.19 / Д.Н. Буинцев. – Томськ, 2006. – 121 с.

7. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Мова UML. Посібник користувача = The Unified Modeling Language user guide. — 2-е изд. — М., Спб.: ДМК Пресс, Пітер, 2004. — 432 с. — ISBN 5-94074-260-2

8. Джозеф Шмуллер. Освой самостійно UML 2 за 24 години. Практичне керівництво = Sams Teach Yourself UML in 24 Hours, Complete Starter Kit. — М.: Вільямс, 2005. — 416 с. — ISBN 0-672-32640-X

9. Крэг Ларман. Застосування UML 2.0 і шаблонів проектування = Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development. — 3-е изд. — М.: Вільямс, 2006. — 736 с. — ISBN 0-13-148906-2

10. Стивен Норткат, Джуди Новачок. Виявлення вторгнень у мережу. Настільна книга фахівця із системного аналізу. — М.: "Лорі", 2001. – 384 стор. — ISBN 5-8459-0526-5, 0-7357-1265-4

11. Таненбаум Є. Комп'ютерні сети. 4-е изд. – Спб.: Пітер, 2003. – 992 с. – ISBN 5-8046-0133-4

12. Вільям Столлингс. Основи захисту мереж. Додатка й стандарти. – изд. – Вільямс, 2002. – 432 с. – ISBN 5-8459-0298-3, 0-1301-6093-8

13. Noran, Ovidiu S.. "Business Modelling: UML vs. IDEF". Retrieved 2005-12-28

Додаток А

```
unit Unit1;

interface

uses
  Windows, Messages, Sysutils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Stdctrls, Comctrls, Commctrl, Winsock, Imglist, Shellapi, Xpman,
  jpeg, Extctrls, Filectrl, Advprogressbar, inifiles, registry,  Iphlpapi, Iptypes,
Ipifconst;

resourcestring
  RES_THREADCOUNT = 'Запущене потоків: %d';
  RES_COMPCOUNT = 'Знайдене: %d';
  RES_ERR_RANGE = 'Неприпустимий діапазон';


type

  Tipedit = class
  private
    Fhandle: Thandle;
    FIP: Integer;     Ffont: Integer;
    function Gettext: String;
    procedure Settext(const Value: String);
  public
    constructor Create(Aowner: Twincontrol; Rect: Trect);
    destructor Destroy; override;
    property Text: String read Gettext write Settext;
  end;

    function Portstatetostr(const State: DWORD): String;

    procedure Setthreadcount(const Value: Integer);
    procedure Setcompfound(const Value: Integer);

    procedure Getmemoryinfo;     procedure Getcompinfo;
    procedure Getipinfo;        procedure Getadapterinfo;
    procedure Updatedisk;
  public
    property Threadcount: Integer read Fthreadcount write Setthreadcount;
    property Compfound: Integer read Fcompfound write Setcompfound;
  end;
```

```
LMSTR = LPWSTR;
NET_API_STATUS = DWORD;

Pshareinfo1 = _SHARE_INFO_1;
_SHARE_INFO_1 = record
 shi1_netname: LMSTR;
 shi1_type: DWORD;
 shi1_remark: LMSTR;
end;
Tshareinfo1 = _SHARE_INFO_1;

  function Gettcptable(ptcptable: Ptmibtcptable; var pdwsize: DWORD;
  border: BOOL): DWORD; stdcall; external 'IPHLPAPI.DLL';

function Getudptable(pudptable: Ptmibudptable; var pdwsize: DWORD;
  border: BOOL): DWORD; stdcall; external 'IPHLPAPI.DLL';

//{$IFDEF USES_NATIVE_API}

  function Allocateandgettcpextablefromstack(ptcpextable: Ptmibtcpextable;
   border: BOOL; heap: Thandle; zero: DWORD; flags: DWORD):
DWORD; stdcall;
   external 'IPHLPAPI.DLL';

  function Allocateandgetudpextablefromstack(pudpextable:
Ptmibudpextable;
   border: BOOL; heap: Thandle; zero: DWORD; flags: DWORD):
DWORD; stdcall;
   external 'IPHLPAPI.DLL';

  function Createtoolhelp32Snapshot(dwflags, th32Processid: DWORD):
Thandle;
   stdcall; external 'KERNEL32.DLL';

  function Process32First(hsnapshot: Thandle; var lppe: Tprocessentry32):
BOOL;
   stdcall; external 'KERNEL32.DLL' name 'Process32Firstw';

  function Process32Next(hsnapshot: Thandle; var lppe: Tprocessentry32):
BOOL;
   stdcall; external 'KERNEL32.DLL' name 'Process32Nextw';

//{$ENDIF}

var
```

```
  Form1: Tform1;

implementation

{$R *.dfm}

procedure Tform1.Setcompfound(const Value: Integer);
begin
  Fcompfound := Value;
  Statusbar1.Panels.Items[1].Text := Format(RES_COMPCOUNT, [Value]);
  Application.Processmessages;
end;

procedure Tform1.Setthreadcount(const Value: Integer);
begin
  if Value < Fthreadcount then
    Progressbar.Position := Progressbar.Max - Value;
  Fthreadcount := Value;
  Statusbar1.Panels.Items[0].Text := Format(RES_THREADCOUNT,
[Value]);
  if Value = 0 then
  begin
    Progressbar.Position := 0;
    btnstart.Enabled := True;
  end;
  Application.Processmessages;
end;

procedure Scanlocalnw();
begin
  with form1 do begin
    Tag := Tag + 1;
    if (Tag mod 2) = 1 then begin
      Treeview1.Items.Clear;
      Statusbar1.Panels[1].Text := STR_STARTED;
      Thread := Tdemothread.Create(False);
    end
    else  begin
      Statusbar1.Panels[1].Text := STR_END;
      Thread.Terminate;
    end;
  end;
end;

function Getipaddress(Networkname: String): String;
```

```
var
 Error: DWORD;
 Hostentry: Phostent;
 Data: Wsadata;
 Address: In_Addr;
begin
  Delete(Networkname, 1, 2);
  Error:=Wsastartup(Makeword(1, 1), Data);
  if Error = 0 then
  begin
    Hostentry:=gethostbyname(Pchar(Networkname));
    Error:=Getlasterror;
    if Error = 0 then
    begin
      Address:=Pinaddr(Hostentry.h_addr_list);
      Result:=inet_ntoa(Address);
    end
    else
     Result:='Unknown';
  end
  else
    Result:='Error';
  Wsacleanup;
end;

{ Tform1 }

procedure Tform1.Formcreate(Sender: Tobject);
begin
  application.Title:='Аналізатор уразливості';

  Tag := 0;

  Ipfrom := Tipedit.Create(gbaddrrange, Rect(32, 16, 121, 21));
  Ipfrom.Text := '192.168.0.1';
  Ipto := Tipedit.Create(gbaddrrange, Rect(32, 40, 121, 21));
  Ipto.Text := '192.168.0.10';

   // Задамо первісний IP адреса (це адреса моєї машини)
  IP := MAKEIPADDRESS(192, 168, 0, 3);

  Scanlocalnw;
end;

procedure Tform1.Treeview1Click(Sender: Tobject);
```

```
var
  p1,a,b,c,d,p2:integer;
  st:string;
begin
  if Assigned(Treeview1.Selected) then begin
    Statusbar1.Panels[0].Text := '   ' + Treeview1.Selected.Text;
    st:=Treeview1.Selected.Text;
    p1:=Pos(':',st);
    if p1<>0 then begin
      Ipst:=Copy(st,p1+1,length(st)-p1);
      p2:=Pos('.',st);
      a:=Strtoint(Copy(st, p1+1, p2-p1-1));
      st[p2]:=',';         p1:=p2;
      p2:=Pos('.',st);      b:=Strtoint(Copy(st, p1+1, p2-p1-1));
      st[p2]:=',';          p1:=p2;
      p2:=Pos('.',st);
      c:=Strtoint(Copy(st, p1+1, p2-p1-1));
      st[p2]:=',';
      p1:=p2;
      p2:=length(st)-p1;
      d:=Strtoint(Copy(st, p1+1, p2));
      IP := MAKEIPADDRESS(a,b,c,d);
      Infcomp;
    end;
  end
  else Statusbar1.Panels[0].Text := STR_FIELD;
end;

procedure Tform1.Treeview1Dblclick(Sender: Tobject);
var
  Str: String;
begin
  if Assigned(Treeview1.Selected) then
  begin
    Str := Treeview1.Selected.Text;
    if Copy(Str, 1, 2) <> '\\' then Exit;
    if Pos(' IP:', Str) <> 0 then
      Shellexecute(Handle, 'explore', Pchar(Copy(Str, 1, Pos(' IP:', Str))), nil,
nil, SW_SHOW)
    else
      Shellexecute(Handle, 'explore', Pchar(Str), nil, nil, SW_SHOW);
  end;
end;

procedure Tform1.Pagecontrol2Change(Sender: Tobject);
```

```
begin
  //if form1.Pagecontrol2.Activepageindex=0 then Scanlocalnw;
end;

constructor Tipedit.Create(Aowner: Twincontrol; Rect: Trect);
begin
  Initcommoncontrol(ICC_INTERNET_CLASSES);
  Fhandle:= Createwindow(WC_IPADDRESS, nil, WS_CHILD or
WS_VISIBLE,
    Rect.Left, Rect.Top, Rect.Right, Rect.Bottom, Aowner.Handle, 0,
hinstance, nil);
  Ffont := Createfont(-11, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY,
    DEFAULT_PITCH or FF_DONTCARE, 'MS Sans Serif');
  Sendmessage(Fhandle, WM_SETFONT, Ffont, 0);
  Text := '0.0.0.0';
end;

destructor Tipedit.Destroy;
begin
  Deleteobject(Ffont);
  inherited;
end;

function Tipedit.Gettext: String;
begin
  Sendmessage(Fhandle, IPM_GETADDRESS, 0,
Longint(PDWORD(@FIP)));
  Result := Inttostr(FIRST_IPADDRESS(FIP))+
     '.' + Inttostr(SECOND_IPADDRESS(FIP)) +
     '.' + Inttostr(THIRD_IPADDRESS(FIP)) +
     '.' + Inttostr(FOURTH_IPADDRESS(FIP));
end;

procedure Tipedit.Settext(const Value: String);

  function Makeipaddressex(b1, b2, b3, b4: Char):LPARAM;
  begin
    Result := MAKEIPADDRESS(DWORD(b1), DWORD(b2), DWORD(b3),
DWORD(b4));
  end;

var
  Tmp: Tinaddr;
```

```
begin
  Tmp.S_addr := inet_addr(Pchar(Value));
  if Tmp.S_addr = INADDR_NONE then Exit;
  with Tmp.S_un_b do
    FIP := Makeipaddressex(s_b1, s_b2, s_b3, s_b4);
  Sendmessage(Fhandle, IPM_SETADDRESS, 0, FIP);
end;

{ Tscanthread }

procedure Tscanthread.Deccount;
begin
  Form1.Threadcount := Form1.Threadcount - 1;
end;

procedure Tscanthread.Updatetree;
var
  I: Integer;
  Root: Ttreenode;
begin
  Form1.tvresult.Items.Beginupdate;
  try
    Root := Form1.tvresult.Items.Add(nil, Fres.Strings[0]);
    for I := 1 to Fres.Count - 1 do
      Form1.tvresult.Items.Addchild(Root, Fres.Strings[I]);
      Form1.Compfound := Form1.Compfound + 1;
  finally
    Form1.tvresult.Items.Endupdate;
  end;
end;

procedure Infcomp;
var
  Tmpcompname, Tmpprovider, Tmpgroup, Tmpuser, Tmpserver: String;
  Time: Cardinal;
  Ipstr: String;
begin
 with Form1 do begin
    Time := Gettickcount;  // Засічемо час...

    // Перетворимо цю абракадабру в нормальний "Dotted IP"
    Ipstr := Inttostr(FIRST_IPADDRESS(IP));
    Ipstr := Ipstr + '.' + Inttostr(SECOND_IPADDRESS(IP));
    Ipstr := Ipstr + '.' + Inttostr(THIRD_IPADDRESS(IP));
    Ipstr := Ipstr + '.' + Inttostr(FOURTH_IPADDRESS(IP));
```

```
        // Ну й почнемо працювати...
        with meminfo, meminfo.Lines do begin                    // Висновок інформації
          Clear;                                  // Очищаємо екран
          Refresh;                                  // Ну й обновляємо...
                                                  // ( при виклику першої функції може не обновитися)
          Add(RES_IP + Ipstr);                          // Виводимо IP адреса
          Tmpcompname := Getnamefromip(Ipstr);
          if Tmpcompname = RES_UNKNOWN then Exit;
          Add(RES_CMP + Tmpcompname);                       // Виводимо ім'я комп'ютера
          Tmpuser := Getusers(Ipstr);
          Add(RES_USR + Tmpuser);                       // Виводимо ім'я користувача
          Tmpprovider := Getprovider(Tmpcompname);
          Add(RES_PROV + Tmpprovider);                      // Виводимо провайдера
          Add(RES_COM + Getcomment(Tmpcompname,
            Tmpprovider));                          // Виводимо коментар до ресурсу
          Tmpgroup := Getdomain(Tmpcompname, Tmpprovider);
          Add(RES_DOM + Tmpgroup);                      // Виводимо групу
          Tmpserver := Getdomainserver(Tmpgroup);
          if Tmpserver <> '' then begin
            Add(RES_SER + Tmpserver);                   // Виводимо ім'я сервера
            Add(RES_GRP + Getgroups(Tmpserver, Tmpuser));       // Виводимо групи домена в які входить користувач
          end;
          Add(RES_SHARES + Getshares(Tmpcompname));             // Виводимо список доступних ресурсів
          Add(RES_MAC + Getmacfromip(Ipstr));                   // Виводимо MAC адреса
          Add(RES_TIME + Inttostr(Gettickcount - Time));        // Скільки часу витрачене
        end;
      end;
    end;

    function Tform1.Getnamefromip(const IP: String): String;
    var
      WSA: Twsadata;
      Host: Phostent;
      Addr: Integer;
```

```
  Err: Integer;
begin
  Result := RES_UNKNOWN;
  Err := Wsastartup(WSA_TYPE, WSA);
  if Err <> 0 then  // Краще користуватися такою конструкцією,
  begin             // щоб у випадку помилки можна було побачити її код.
    //Showmessage(Syserrormessage(Getlasterror));
    Exit;
  end;
  try
    Addr := inet_addr(Pchar(IP));
    if Addr = INADDR_NONE then
    begin
      //Showmessage(Syserrormessage(Getlasterror));
      Wsacleanup;
      Exit;
    end;
    Host := gethostbyaddr(@Addr, Sizeof(Addr), PF_INET);
    if Assigned(Host) then  // Обов'язкова перевірка, а якщо ні, то, при
      Result := Host.h_name // відсутності комп'ютера із заданым IP,
одержимо AV
    else
      //Showmessage(Syserrormessage(Getlasterror));
  finally
    Wsacleanup;
  end;
end;

// перерахування доменних груп у які входить користувач
function Tform1.Getgroups(Domainserver: String; Username: String): String;
type
  Tgroupusersinfoarray = array of Tgroupusersinfo0;
var
  Info: Pgroupusersinfo0;
  Sn, Un: Pwidechar;
  entriesread, totalentries: DWORD;
  I, A, B, Size: Integer;
  P: Pointer;
begin
  // нам потрібно тільки ім'я сервера домена
  Sn := Stringtoolestr(Domainserver);
  // і ім'я користувача
  Un := Stringtoolestr(Username);
  // робимо запит
```

```
        if Netusergetgroups(Sn, Un, 0, @Info, DWORD(-1), entriesread,
totalentries) = NO_ERROR  then
        try // і дивимося, що там у нас вийшло
          if entriesread > 0 then
            for I := 0 to entriesread - 1 do
              Result := Result + Tgroupusersinfoarray(@(Info))[I].grui0_name + ' ';
        finally
          Netapibufferfree(Info);
        end;
      end;

      procedure Tform1.Getmemoryinfo;
      var
       Meminfo : Tmemorystatus;
      begin
       Meminfo.dwlength := Sizeof (Meminfo);
       Globalmemorystatus (Meminfo);

       Totalphys.caption:=inttostr(Meminfo.dwtotalphys div 1024) + ' K';
       Availphys.caption:=inttostr(Meminfo.dwavailphys div 1024) + ' K';

       Totalpage.caption:=inttostr(Meminfo.dwtotalpagefile div 1024) + ' K';
       Availpage.caption:=inttostr(Meminfo.dwavailpagefile div 1024) + ' K';

       Advprogressbar1.Position := Meminfo.dwavailphys div
(Meminfo.dwtotalphys div 100);
       Advprogressbar2.Position := Meminfo.dwavailpagefile div
(Meminfo.dwtotalpagefile div 100);
      end;

      procedure Tform1.Button2Click(Sender: Tobject);
      begin
       Pagecontrol3.Activepageindex:=0;
       Getmemoryinfo;
       Getcompinfo;
       Getipinfo;
       Getadapterinfo;
       Updatedisk;
      end;


      procedure Tform1.Drivecombobox1Change(Sender: Tobject);
      begin
        Updatedisk;
      end;
```

```
procedure Tform1.Adaptercbchange(Sender: Tobject);
begin
  Getadapterinfo;
end;

procedure Tform1.Formshow(Sender: Tobject);
begin
  Pagecontrol3.Activepageindex:=0;
  Getmemoryinfo;
  Getcompinfo;
  Getipinfo;
  Getadapterinfo;
  Updatedisk;
end;

// Одержання TCP/UDP статистики за допомогою стандартних методів
procedure Tform1.Button31Click(Sender: Tobject);
var
  Size: DWORD;
  Tcptable: Ptmibtcptable;
  Udptable: Ptmibudptable;
  I: DWORD;
begin
  Memo1.Clear;
  // для успішного одержання статистики спочатку необхідно визначитися
  // скільки пам'яті зажадає дана операція
  // для цього робимо так:
  // Вделяем пам'ять під TCP таблицю ( під один елемент)
  Getmem(Tcptable, Sizeof(Tmibtcptable));
  try
    // Показуємо що пам'яті в нас не виділене
    Size := 0;
    // Виконуємо функцію й після цього змінна Size
    // буде містити кілл у необхідній пам'яті
    if Gettcptable(Tcptable, Size, True) <> ERROR_INSUFFICIENT_BUFFER then Exit;
  finally
    // звільняємо пам'ять зайняту під один елемент
    Freemem(Tcptable);
  end;
  // Тепер виділяємо вже необхідне кіллу пам'яті
  Getmem(Tcptable, Size);
  try
```

```
        // Виконуємо функцію
        if Gettcptable(Tcptable, Size, True) = NO_ERROR then
        begin
          Memo1.Lines.Add('');
          Memo1.Lines.Add('            АНАЛІЗ ПО TCP ПРОТОКОЛУ');
          Memo1.Lines.Add('');
          Memo1.Lines.Add(Format('%15s: | %5s %-12s', ['Хост', 'Порт', 'Стан']));

Memo1.Lines.Add('==========================================
======');
        // і насинаем виводити дані по TCP
        for I := 0 to Tcptable.dwnumentries - 1 do
          Memo1.Lines.Add(Format('%15s:              |              %5d              %s',
[inet_ntoa(in_addr(Tcptable.Table[I].dwlocaladdr)),
            htons(Tcptable.Table[I].dwlocalport),
Portstatetostr(Tcptable.Table[I].dwstate)]));
        end;
      finally
        // Не забуваємо звільнити пам'ять
        Freemem(Tcptable);
      end;

      // За аналогією надходимо й з UDP статистикою
      Getmem(Udptable, Sizeof(Tmibudptable));
      try
        Size := 0;
        if            Getudptable(Udptable,            Size,            True)            <>
ERROR_INSUFFICIENT_BUFFER then Exit;
      finally
        Freemem(Udptable);
      end;
      Getmem(Udptable, Size);
      try
        if Getudptable(Udptable, Size, True) = NO_ERROR then
        begin
          Memo1.Lines.Add('');
          Memo1.Lines.Add('            АНАЛІЗ ПО UDP ПРОТОКОЛУ');
          Memo1.Lines.Add('');
          Memo1.Lines.Add(Format('%15s: | %5s ', ['Хост', 'Порт']));

Memo1.Lines.Add('====================================');
        for I := 0 to Udptable.dwnumentries - 1 do
          Memo1.Lines.Add(Format('%15s:                    |                    %5d',
[inet_ntoa(in_addr(Udptable.Table[I].dwlocaladdr)),
            htons(Udptable.Table[I].dwlocalport)]));
```

```
    end;
  finally
    Freemem(Udptable);
    Memo1.Lines.Delete(0);
    Memo1.Lines.Insert(0,'');
  end;
end;

end.
```