Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

До захисту допущено
Завідувач кафедри
комп'ютерної інженерії
к.т.н., доц. О.М.Березький
_____
"___" _____ 20___ р.

ДИПЛОМНА РОБОТА
освітньо-кваліфікаційного рівня "Магістр"
зі спеціальності 8.05010201 "Комп'ютерні системи та мережі"
на тему:

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ВИЯВЛЕННЯ КОМП'ЮТЕРНИХ АТАК НА ОСНОВІ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

Студент групи КСМм - 51
Луцишин В.І.

_____
підпис
Науковий керівник
д.т.н., професор Саченко А.О.

_____
підпис
Консультант з нормоконтролю
Палій І.О.
_____
Прізвище, ініціали Підпис

Тернопіль – 2012

Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

"Затверджую"
Зав. кафедри
комп'ютерної інженерії
к.т.н., доц. О.М. Березький
_____
"____"_____20__ р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

Луцишина Володимира Ігоровича

1. **Тема дипломної роботи** "Інтелектуальна система виявлення комп'ютерних атак на основі імітаційного моделювання" затверджена наказом університету №_____ від „____" _____20__ р

2.**Термін здачі** закінченої дипломної роботи _____

3. **Об'єкт дослідження**: процеси побудови та функціонування інтелектуальної системи виявлення атак.

4. **Предмет дослідження**: системні моделі функціонування системи виявлення атак та алгоритми прийняття рішень, які засновані на використанні динамічних моделей інформаційної системи на базі нечітких когнітивних карт.

5. **Перелік задач, які мають бути вирішені**:

- розробка системних моделей функціонування ІС з використанням SADT методології;
- синтез архітектури і алгоритмів системи прийняття рішень на основі динамічної моделі оцінки ризиків з використанням нечітких когнітивних карт;
- розробка дослідницького прототипу інтелектуальної системи виявлення атак;
- аналіз ефективності функціонування розробленої інтелектуальної системи виявлення атак методом імітаційного моделювання.

6. **Перелік ілюстративного матеріалу:**

- онтологічна модель ІС,
- функціональна модель ІС,
- деталізована функціональна модель ІС,
- функціональна модель системи захисту ІС
- функціональна модель СВА,
- функціональна модель сенсора СВА,
- структура сегменту мережі,
- структурна схема СВА,
- структурна схема інтелектуального сенсора СВА,
- структура центральної бази даних.

7.**Консультанти по роботі**

Розділ	Консультант	Підпис
1		
2		
3	Комар М.П.	

КАЛЕНДАРНИЙ ПЛАН

№	Назва структурних частин ДР	Термін виконання	Примітка
1	Аналіз сучасного стану побудови систем виявлення атак на інформаційні ресурси	15.09.2011 – 5.11.2011	
2	Моделі функціонування інформаційної системи	6.11.2011 – 31.01.2012	
3	Реалізація інтелектуальної системи виявлення атак	1.02.2012 – 23.04.2012	

Завдання прийняв до виконання _____
(підпис)

Керівник дипломної роботи _____
(підпис)

РЕФЕРАТ

Дипломна робота на тему “Інтелектуальна система виявлення комп’ютерних атак на основі імітаційного моделювання” на здобуття освітньо-кваліфікаційного рівня “Магістр” зі спеціальності “Комп’ютерні системи та мережі” написана обсягом 94 сторінки і містить 26 ілюстрацій, 16 таблиць, 1 додаток та 48 джерел за переліком посилань.

Метою роботи є підвищення ефективності виявлення атак на інформаційні ресурси на основі оперативної оцінки ризиків функціонування інформаційної системи з використанням динамічних моделей на основі нечітких когнітивних карт.

Методи досліджень. В процесі дослідження використовувалися методи системного аналізу, теорії ймовірності, нечіткої логіки, теорії Марківських ланцюгів, теорії процесів, нечітких когнітивних карт, теорії нейронних мереж, методи розпізнавання образів, математичної статистики і інформатики.

Розроблений комплекс системних моделей функціонування системи виявлення атак на основі IDEF-технологій.

Запропоновані архітектура системи виявлення атак і алгоритми системи прийняття рішень, які засновані на використанні динамічних моделей інформаційної системи на базі нечітких когнітивних карт, що дозволяє виявляти невідомі атаки.

Запропонований алгоритм навчання нечіткої когнітивної карти на наборі еталонних даних, заснований на алгоритмі зворотного розповсюдження помилки, який дозволяє істотно підвищити точність розпізнавання атак.

Ключові слова: ІНТЕЛЕКТУАЛЬНА СИСТЕМА, ВИЯВЛЕННЯ АТАК, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, ФУНКЦІОНАЛЬНА МОДЕЛЬ, ОНТОЛОГІЧНА МОДЕЛЬ, ДИНАМІЧНА МОДЕЛЬ.

ABSTRACT

Diploma work «Intelligent detection system of computer attacks based on simulations» on acquiring of educationally-qualification «Master» degree, from speciality «Computer systems and networks» with total volume 94 pages that contains 26 illustrations, 16 tables, 1 addition and 48 sources of information according to the list of references.

The objective is to improve attacks detection on information resources based on operational quick estimation of function-risk of information system with the help of dynamic models on the basis of fuzzy cognitive maps.

Research Methods. In the research are used methods of systems analysis, probability theory, fuzzy logic, the theory of Markov chains, the theory of processes, fuzzy cognitive maps, the theory of neural networks, pattern recognition methods, mathematical statistics and computer science methods.

The set of system models functioning of the attacks detection system based on IDEF-technology is developed.

The architecture of attack detection system and algorithms of decision-making systems are proposed that are based on the use of dynamic models of information systems on the basis fuzzy cognitive maps, which can detect unknown attacks.

The proposed learning algorithm of fuzzy cognitive maps to on the set of reference data is based on error backward distribution algorithm, which can significantly improve attacks recognition accuracy.

Keywords: INTELLIGENT SYSTEMS, ATTACK DETECTION, SIMULATION, INFORMATION SYSTEMS, FUNCTIONAL MODEL, ONTOLOGICAL MODEL, DYNAMIC MODELS.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ЛОМ – локальна обчислювальна мережа

ІС – інформаційна система

СВА – системи виявлення атак

НКК – нечітка когнітивна карта

ДЧ – дестабілізуючий чинник

ПЧ – проміжний чинник

ЦЧ – цільовий чинник

КЧ – керуючий чинник

ПЗ – програмне забезпечення

АС – автоматизована система

ГБЗ – глобальна база знань

ОС – операційна система

СУБД – система управління базою даних

ВСТУП

Актуальність роботи. У зв'язку із збільшенням об'ємів інформації, що передаються в локальних обчислювальних мережах (ЛОМ) і розширенням кола задач, які вирішуються за допомогою інформаційних систем (ІС), виникає проблема, пов'язана із зростанням числа загроз і підвищенням вразливості інформаційних ресурсів [1]. Це обумовлено дією багатьох чинників, таких як:

- розширення спектру задач, що вирішуються ІС;
- підвищення складності алгоритмів обробки інформації;
- збільшення об'ємів інформації, що обробляється;
- ускладнення програмних і апаратних компонентів ЛОМ і відповідно – підвищення ймовірності наявності помилок і вразливостей;
- підвищення агресивності зовнішніх джерел даних (глобальних мереж);
- поява нового виду загроз.

Необхідно враховувати, що конкурентоспроможність підприємств, розмір отримуваного ними доходу, їх положення на ринку істотно залежать від коректності функціонування їх інформаційної інфраструктури, цілісності основних інформаційних ресурсів, захищеності конфіденційної інформації від несанкціонованого доступу. Виходячи з цього, зростають вимоги до систем захисту ЛОМ, які повинні забезпечувати не тільки пасивне блокування несанкціонованого доступу до внутрішніх ресурсів мережі підприємства із зовнішніх мереж, але і здійснювати виявлення атак, аналізувати причини виникнення загроз інформаційній безпеці і, в міру можливості, усувати їх в автоматичному режимі.

Однією з основних якостей системи захисту інформації ЛОМ підприємства, що задовільняє перерахованим вимогам, є її адаптивність, тобто здатність аналізувати інформацію, генерувати на її основі знання і автоматично змінювати конфігурацію системи для блокування виявлених загроз інформаційній безпеці [2, 3].

Аналіз існуючих підходів до реалізації систем виявлення атак показує, що більшість програмних продуктів, присутніх в даний час на ринку, орієнтуються на

використання формальних описів системної активності (сигнатур) [4, 5]. Функції виявлення і реєстрації нового виду атак покладаються в подібних системах на розробника, що випускає нові сигнатури. Даний метод захисту є ненадійним, оскільки він ставить захищеність ІС в залежність від дій зовнішнього неконтрольованого джерела.

Не дивлячись на те, що розробка адаптивних систем захисту інформації ведеться вже достатньо тривалий час, жодне подібне рішення не набуло широкого поширення через складність і малоефективність алгоритмів, що використовуються, відсутність в більшості випадків адекватних інструментів їх розгортання і адміністрування, а також користувацької документації.

Аналіз робіт, що ведуться в даній області, показує, що дана проблема вимагає подальшого вивчення як з погляду побудови адекватних математичних моделей предметної області, так і реалізації ефективних алгоритмів виявлення атак і прийняття рішень, що підтверджує актуальність досліджень в даній області.

Мета і завдання дослідження. Метою роботи є підвищення ефективності виявлення атак на інформаційні ресурси на основі оперативної оцінки ризиків функціонування ІС з використанням динамічних моделей на основі нечітких когнітивних карт.

Для досягнення поставленої мети в роботі були поставлені і вирішені наступні завдання:

1. Розробка системних моделей функціонування ІС з використанням SADT методології.
2. Синтез архітектури і алгоритмів системи прийняття рішень на основі динамічної моделі оцінки ризиків з використанням нечітких когнітивних карт.
3. Розробка дослідницького прототипу інтелектуальної системи виявлення атак.
4. Аналіз ефективності функціонування розробленої інтелектуальної системи виявлення атак методом імітаційного моделювання.

Об'єкт дослідження – процеси побудови та функціонування інтелектуальної системи виявлення атак.

Предмет дослідження – системні моделі функціонування системи

виявлення атак та алгоритми прийняття рішень, які засновані на використанні динамічних моделей інформаційної системи на базі нечітких когнітивних карт.

Методи досліджень. В процесі дослідження використовувалися методи системного аналізу, теорії ймовірності, нечіткої логіки, теорії Марківських ланцюгів, теорії процесів, нечітких когнітивних карт, теорії нейронних мереж, методи розпізнавання образів, математичної статистики і інформатики.

Наукова новизна одержаних результатів. Розроблений комплекс системних моделей функціонування системи виявлення атак (СВА) на основі IDEF-технологій, який дозволяє виявити основні джерела загроз, вразливості і ресурси, що захищаються, формулювати вимоги до архітектури СВА. Запропоновані архітектура СВА і алгоритми системи прийняття рішень, які засновані на використанні динамічних моделей інформаційної системи на базі нечітких когнітивних карт, що на відміну від існуючих підходів, дозволяє виявляти невідомі атаки. Запропонований алгоритм навчання нечіткої когнітивної карти на наборі еталонних даних, заснований на алгоритмі зворотного розповсюдження помилки, який дозволяє істотно підвищити точність розпізнавання і блокування атак.

Практичне значення отриманих результатів. Запропонована математична модель і методи оцінки ризиків функціонування ІС можуть використовуватися на ранніх етапах розробки систем захисту інформації для оцінки їх ефективності. Розроблені алгоритми генерації і представлення знань дозволяють підвищити ефективність систем захисту ІС. Використання запропонованої архітектури інтелектуального модуля прийняття рішень дозволяє збільшити ефективність систем виявлення атак і понизити величину ризику функціонування ІС.

Публікації та апробація ДР. Результати дипломної роботи апробовані на студентській науковій конференції «Науково-дослідна робота студентів: формування особистості майбутнього вченого, фахівця високої кваліфікації», ТНЕУ, 2012р.

У першому розділі дипломної роботи показано, що збільшення ролі ІС в задачах збору, аналізу і обробки інформації приводить до необхідності приділяти

активну увагу вирішенню проблем інформаційної безпеки. Вирішення даних проблем можливе шляхом використання різних захисних механізмів, одним з яких є СВА. Використання методології SADT є одним з найбільш зручних способів моделювання елементів ІС на етапі їх проектування, оскільки дозволяє виявити їх структуру, основні функції, залежності між компонентами, основні інформаційні активи, а також формат і зміст структур даних, якими вони обмінюються.

У другому розділі запропонована методика побудови динамічної моделі ІС на основі нечітких когнітивних карт, яка може бути використана надалі як інтелектуальний модуль прийняття рішень в СВА. Проведено імітаційне моделювання з використанням динамічної моделі ІС. Оскільки моделювання проводилося з використанням розробленого прототипу СВА, в якості вхідних даних використовувалася інформація з сенсорів, а не статистична модель атак. Для зменшення ймовірності виникнення помилок I і II роду був запропонований метод коректування ваг концептів і зв'язків, заснований на алгоритмі Back Propagation, суть якого зводиться до навчання нечіткої когнітивної карти на наборі базових еталонів. Використання даного підходу дозволило істотно зменшити кількість помилок розпізнавання атак на інформаційну систему.

У третьому розділі проведений синтез архітектури сенсорів і модулів, розроблена структура бази даних на основі формалізованих моделей IDEF. В якості базової платформи для розробленого прототипу вибрана ОС Microsoft Windows, як найбільш поширена в корпоративному середовищі. Реалізований набір сенсорів, які дозволяють збирати інформацію про активність системних і користувацьких процесів для виявлення і блокування шкідливої активності, а також інструментарій налаштування і тестування системи. Даний модуль дозволяє здійснювати налаштування системи прийняття рішень (структура і конфігурація нечіткої когнітивної карти, склад сенсорів, і т. д.), а також проводити тестування СВА як в режимі збору реальних даних, так і з використанням моделі атак.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПОБУДОВИ СИСТЕМ ВИЯВЛЕННЯ АТАК НА ІНФОРМАЦІЙНІ РЕСУРСИ

1.1 Проблеми інформаційної безпеки систем

1.1.1 Базові поняття в області інформаційної безпеки

Одним з найбільш ефективних методів підвищення ефективності виробництва є активне використання інформаційних технологій, що надають потужні методи організації, обробки і зберігання інформації, тобто відомостей про осіб, предмети, факти, події, явища і процеси, незалежно від форми їх представлення.

Під інформаційними процесами розуміються процеси збору, обробки, накопичення, зберігання, пошуку і розповсюдження інформації.

Інформаційна система – це організаційно впорядкована сукупність документів (масивів документів і інформаційних технологій), зокрема з використанням засобів обчислювальної техніки і зв'язку, що реалізовують інформаційні процеси [6].

Підвищення ефективності інформаційної системи здійснюється в першу чергу за рахунок автоматизації процесів обробки інформації. Під автоматизованою системою при цьому розуміється організована сукупність засобів, методів і заходів, що використовуються для регулярної обробки інформації в процесі вирішення прикладних задач.

Використання автоматизованих систем приводить до збільшення залежності коректного функціонування системи від збереження використовуваних блоків інформації, що, у свою чергу, приводить до необхідності організації спеціальних заходів щодо захисту інформації – діяльності по запобіганню просочуванню інформації, що захищається несанкціонованих і ненавмисних дій на інформацію, що захищається [7, 8].

Під безпекою інформації розуміється стан захищеності інформації, яка обробляється засобами обчислювальної техніки або автоматизованої системи, від внутрішніх або зовнішніх загроз.

Загроза інформаційній безпеці – це подія або сукупність подій, реалізація яких може привести до нанесення збитку інформаційній системі.

Реалізація загроз інформаційній безпеці здійснюється в результаті наявності вразливостей інформаційної системи – набору властивостей або механізмів інформаційної системи, експлуатація яких може привести до порушення безпеки інформації.

Під атакою на інформаційну систему розуміється процес пошуку і експлуатації вразливостей. Необхідно відзначити, що атака на інформаційну систему не завжди має навмисний характер і може відбуватися у ряді випадків в результаті некоректних дій користувачів системи. Тому дане визначення необхідно розширити і розуміти під атакою будь–яку активність в інформаційній системі, в результаті якої може бути порушена безпека інформації.

Для забезпечення коректності функціонування інформаційної системи в умовах дії на неї можливих дестабілізуючих чинників (атак), необхідно використовувати систему захисту ІС, яка є частиною системи управління ІС. Узагальнена структурна схема, що демонструє роль підсистеми захисту ІС, показана на рисунку 1.1.

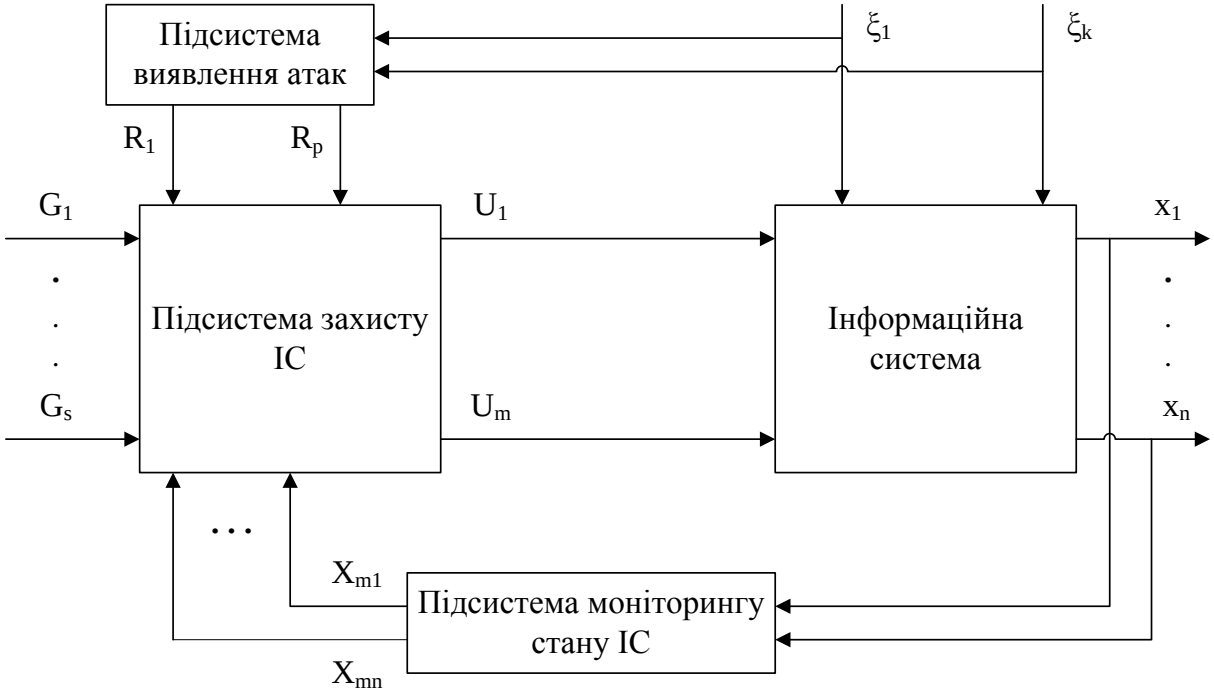

Рисунок 1.1 – Схема взаємодії підсистеми захисту ІС і інформаційної системи

Згідно рисунку 1.1, на автоматизовану інформаційну систему впливають зовнішні дестабілізуючі чинники (атаки) $\xi_1,...,\xi_k$. Дані дії аналізуються підсистемою виявлення атак, яка формує набір керуючих дій $R_1,...,R_p$. Отримані керуючі дії поступають в систему захисту ІС, яка на основі параметрів конфігурації $G_1...G_s$ і даних $X_{m1}...X_{mn}$, отримуваних за допомогою підсистеми моніторингу, формує набір керуючих дій $U_1...U_m$ на ІС, які дозволяють компенсувати дію дестабілізуючих чинників і відновлювати бажаний стан ІС (характеризується змінними стану $x_1...x_n$).

Управління ІС з метою забезпечення заданого рівня інформаційної безпеки є складним недетермінованим процесом внаслідок того, що:

1) зовнішні дії на систему є випадковими, причому в процесі функціонування ІС можуть з'являтися все нові класи дестабілізуючих чинників (атак);

2) ІС є складною гетерогенною системою, що володіє великою кількістю взаємовливаючих компонентів;

3) у зв'язку з великою швидкістю обробки інформації, навіть короткочасна дія дестабілізуючих чинників на систему може привести до великого збитку.

1.1.2 Оцінка рівня безпеки інформаційних ресурсів

Оцінка рівня безпеки інформаційних ресурсів є одним з найважливіших етапів проектування системи захисту інформаційної системи, оскільки вона дозволяє оцінити фактичний рівень захищеності інформації і необхідні фінансові витрати на створення системи захисту. В якості базових критеріїв безпеки інформаційних ресурсів можуть використовуватися [2, 9]:

– ймовірність порушення безпеки інформації за фіксований час $P_H(T)$;

- інші характеристики пов'язані з ймовірністю порушення безпеки інформації $P_H(t)$ (щільність розподілу ймовірності, характеристичні функції, інтегральні показники і т. д.);

- тимчасові оцінки (ймовірний час безпечного існування інформації $T_б$ при заданій довірчій вірогідності P_g);

– потік порушень безпеки інформації, що оцінюється математичним очікуванням інтервалу між сусідніми порушеннями і щільністю розподілу;

– функція відновлення інформації в часі, що оцінюється аналогічним чином.

Математичний аналіз характеристик безпеки інформації може здійснюватися шляхом використання:

– імовірнісних статистичних моделей, байєсівських оцінок ризиків порушення безпеки інформації [10,11];

– казуально–логічних моделей, що є орієнтованими графами, вершини яких відповідають задачам порушника, а дуги – варіантам реалізації загроз [12, 13];

- топологічних моделей, що враховують топологію інформаційної системи (способи взаємодії користувачів, маршрути передачі даних і т. д.) [14, 15];

- ігрових моделей системи безпеки [16].

1.1.3 Підходи до класифікації загроз

Одним з базових етапів проектування системи захисту інформації ІС є побудова таксономії загроз, реалізація яких може привести до атак на ІС. Можна виділити декілька підходів до побудови подібної класифікації [17, 18]:

1. Класифікація загроз по виду збитку, що заподіяний:

– моральний і матеріальний збиток ділової репутації організації;

– моральний, фізичний або матеріальний збиток, пов'язаний з розголошуванням персональних даних окремих осіб;

– матеріальний (фінансовий) збиток від розголошування конфіденційної інформації;

– матеріальний (фінансовий) збиток від необхідності відновлення пошкоджених інформаційних ресурсів;

– матеріальний збиток (втрати) від неможливості виконання взятих на себе зобов'язань перед третьою стороною;

– моральний і матеріальний збиток від дезорганізації діяльності організації;

– матеріальний і моральний збиток від порушення міжнародних зобов'язань.

Даний підхід до класифікації загроз дозволяє ранжирувати їх залежно від ступеня значущості і впливу, що надається на інформаційну систему.

2. Класифікація загроз по впливу на інформацію:

– розкрадання (копіювання) інформації;

– знищення інформації;

– модифікація (спотворення) інформації;

– порушення доступності (блокування) інформації;

– заперечення достовірності інформації;

– нав'язування помилкової інформації.

3. Класифікація загроз по їх джерелу:

– антропогенні;

– техногенні;

– стихійні.

У свою чергу, антропогенні джерела можуть бути внутрішніми і зовнішніми, залежно від того, чи є доступ суб'єкта загрози до інформаційної системи санкціонованим. Зовнішні джерела загроз можуть бути випадковими або навмисними і мати різний рівень кваліфікації. До них можна віднести:

– кримінальні структури;

– потенційних зловмисників (хакерів);

– недобросовісних партнерів;

– технічний персонал постачальників інформаційних послуг;

– представників наглядових організацій і аварійних служб;

– представників силових структур.

Внутрішні джерела антропогенних загроз є легальними користувачами інформаційної системи і, як правило, володіють високою кваліфікацією і знаннями про її будову. До них можна віднести:

– основний персонал;

– представників служби безпеки;
– допоміжний персонал;
– технічний персонал;
– тимчасових користувачів системи.

До техногенних джерел загроз відносять загрози, пов'язані з некоректною роботою технічних компонентів інформаційної системи. До них можна віднести:

– неякісні технічні засоби обробки інформації;
– неякісні програмні засоби обробки інформації;
– допоміжні засоби (охорона, сигналізації, телефонії);
– інші технічні засоби, що використовуються в установі.

Стихійні джерела загроз представляють неконтрольовані обставини, які складають незбориму силу. Подібні загрози мають абсолютний і об'єктивний характер і не піддаються прогнозуванню.

4. Класифікація загроз за типом вразливості. Внаслідок того, що атака на інформаційну систему виявляється як дія зловмисника на існуючі в системі вразливості, даний вид класифікації є одним з найважливіших. Вразливості інформаційної безпеки можуть бути:

– об'єктивними;
– суб'єктивними;
– випадковими.

Об'єктивні уразливості залежать від особливостей побудови і технічних характеристик устаткування, що використовується на об'єкті, що захищається. Повне усунення цих вразливостей неможливе, але вони можуть істотно зменшуватися технічними і інженерно–технічними методами. Суб'єктивні вразливості залежать від дій співробітників і, в основному, усуваються організаційними і програмно–апаратними методами.

Випадкові вразливості залежать від особливостей середовища, що оточує об'єкт, що захищається, і від непередбачених обставин. Ці чинники, як правило, мало передбачені і їх усунення можливе тільки шляхом проведення комплексу організаційних і інженерно–технічних заходів щодо протидії загрозам інформаційної безпеки.

1.1.4 Класифікація атак на ІС

Як вже наголошувалося в параграфі 1.1.1, під атакою на інформаційну систему розуміється процес пошуку і експлуатації вразливості. Таким чином, наявність вразливостей в автоматизованій системі не обов'язково веде до порушення безпеки інформації, негативна дія зловмисників на інформаційну систему виявляється через атаки на неї. Виходячи з цього, побудова таксономії атак на інформаційну систему є наступним кроком після класифікації загроз при її проектуванні. Залежно від способу побудови таксономії, атаки можна класифікувати [18, 19]:

1. По дії на інформацію, що захищається:
- що приводять до порушення конфіденційності;
- що приводять до порушення цілісності;
- що приводять до порушення доступності;
- що надають комплексну негативну дію.

2. По меті атаки:
- отримання привілеїв адміністратора системи;
- отримання привілеїв користувача системи;
- відмова в доступі до інформації (DOS – атака);
- порушення політики безпеки.

3. По прояву атаки:
- атака аутентифікації:
    - підбір пароля;
    - атака на сховища облікових записів;
- атака виконання коду:
    - переповнення буфера;
    - впровадження виконуваного коду;
- атака на функції форматування рядків;
    - впровадження операторів LDAP;
    - виконання команд операційної системи;
    - впровадження запитів SQL;
- атака аналізу конфігурації:

- сканування портів;
- ідентифікація сервісів і додатків;
- виявлення розташування мережевих ресурсів;

4. По рівню протоколу моделі OSI:

– фізичний;
– канальний;
– мережевий;
– транспортний;
– сеансовий;
– представницький;
– прикладний;

5. За типом операційної системи, що атакується.

6. По розташуванню зловмисника:

– фізичний доступ;
– локальна система;
– локальний сегмент мережі;
– міжсегментна атака;
– глобальна мережа (Internet);
– безпровідні мережі;
– комутований доступ (P2P);

7. По виду сервісу, що атакується:

– служба каталогів (LDAP, ADSI);
– Web–сервер (HTTP);
– сервіси обміну файлами (SMB, FTP);
– поштові сервіси (POP3, SMTP, IMAP);
– сервери додатків (DCOM, COM+, Net);
– мережева інфраструктура (DNS, DHCP і т. д.);
– служби маршрутизації (RIP, OSF і т. д.);
– служби віддаленого управління (telnet, RDP);
– служби сертифікатів;
– сервери баз даних;

– інші сервіси;

8. По наявності зворотного зв'язку із зловмисником;

9. За умовами активації атаки:

– по запиту об'єкту, що атакується;

– по певній події об'єкту, що атакується;

– атаки із заданим часом;

– без певних умов;

10. По вигляду взаємодії з об'єктом, що атакується:

– пасивні;

– активні;

11. По ступеню автоматизації:

– автоматизовані;

– частково автоматизовані;

– ручного управління;

12. По топології атаки:

– один до одного;

– один до багатьох;

– багато до одного;

– багато до багатьох;

13. По ступеню прояву:

– прихована атака;

– відкрита атака.

1.2 Аналіз підходів до моделювання інформаційних систем

Для аналізу особливостей функціонування ІС як об'єкту захисту необхідно скласти її формалізований опис у вигляді математичної моделі, яка адекватно описує процес її функціонування. В даний час існує декілька підходів до моделювання ІС, заснованих на:

1) використанні методології SADT;

2) використанні методів теорії систем;

3) використанні марківських моделей [20];
4) застосуванні методів нечіткої логіки [21];
5) застосуванні мереж Петрі [22];
6) побудові нечітких когнітивних карт [23].

1.2.1 Методологія SADT

Методологія SADT – сукупність методів, правив і процедур, призначених для моделювання складних об'єктів будь–якої предметної області. Процес моделювання в SADT включає збір інформації про досліджувану область, документування отриманої інформації і представлення її у вигляді моделі і уточнення моделі за допомогою ітеративного рецензування. Зараз методологія описується наступними стандартами:

– IDEF0 – методологія функціонального моделювання. За допомогою наочної графічної мови IDEF0 представляє систему, що вивчається, у вигляді набору взаємозв'язаних функцій ("функціональних блоків"). Як правило, моделювання засобами IDEF0 є першим етапом вивчення будь–якої системи;

– IDEF1 – методологія моделювання інформаційних потоків всередині системи. Дозволяє відображати і аналізувати їх структуру і взаємозв'язок;

– IDEF1X (IDEF1 Extended) – методологія побудови реляційних структур. IDEFIX відноситься до типу методологій "Суть–взаємозв'язок (ER, Entity–Relationship)" і, як правило, використовується для моделювання реляційних баз даних, що мають відношення до даної системи;

– IDEF2 – методологія динамічного моделювання розвитку систем. Із–за серйозних складнощів, пов'язаних з аналізом динамічних систем, від цього стандарту зараз практично відмовилися і його розвиток припинився на самому початковому етапі. Існуючі алгоритми і їх комп'ютерні реалізації дозволяють перетворювати набір статичних діаграм IDEF0 на динамічні моделі, побудовані на базі "розфарбованих мереж Петрі" (CPN, Color Petri Nets);

– IDEF3 – методологія документування процесів, що відбуваються в системі. За допомогою IDEF3 описуються сценарій і послідовність операцій для кожного процесу. IDEF3 безпосередньо пов'язана з методологією IDEF0: кожна

функція (функціональний блок) може бути представлена засобами IDEF3 у вигляді окремого процесу;

– IDEF4 – методологія побудови об'єктно–орієнтованих систем. Засоби IDEF4 дозволяють наочно відображати структуру об'єктів і принципи їх взаємодії, дозволяючи аналізувати і оптимізувати складні об'єктно–орієнтовані системи;

– IDEF5 – методологія онтологічного дослідження складних систем. За допомогою словника термінів і правил дозволяє описати онтологію системи. У результаті можуть бути сформовані достовірні твердження про стан системи в деякий момент часу, на основі яких робляться висновки про подальший розвиток системи і проводиться її оптимізація.

Методологія SADT може використовуватися для моделювання широкого кола систем, визначення вимог і функцій, а потім для розробки системи, яка задовольняє цим вимогам і реалізує ці функції. Для вже існуючих систем SADT може бути використана для аналізу функцій, що виконуються системою, а також для представлення механізмів, за допомогою яких вони здійснюються.

1.2.2 Методи теорії систем

Інформаційна система підприємства є складним гетерогенним програмно–апаратним середовищем, що реалізовує задачі введення, накопичення, обробки, зберігання і передачі інформації. В загальному вигляді, зміна стану даного середовища може бути описана за допомогою системи диференціальних рівнянь [24]

$$X = f(t, X, U, \xi), \tag{1.1}$$

де $X \in R^n$ – n–мірний фазовий вектор, що описує поточний стан інформаційної системи;

$U \in R^m$ – m–мірна вектор–функція управління, що характеризує дії з боку системи управління (адміністратора, автоматичних коректувань і т. д.), що мають на меті адаптацію ІС до змін умов функціонування;

$\xi \in R^k$ – k–мірний випадковий вектор обурень, що представляє собою атаки на інформаційну систему, відмови, збої її програмних і апаратних компонентів і т. д.;

t – час роботи системи.

Внаслідок того, що програмні і апаратні компоненти ІС, як правило, оперують дискретним часом, систему (1.1) можна переписати таким чином [66]:

$$X_{n+1} = X_n + \tau f(X_n, U_n, \xi_n, t_n) \qquad (1.2)$$

де X_n, U_n, ξ_n – значення векторів змінних X, U, ξ, у момент часу $t=t_n$; t_n – n–й такт функціонування системи (n=0,l,2...,);

$\tau = t_{n+1} - t_n$ – крок інтегрування.

Дану систему рівнянь можна записати в більш загальному вигляді як

$$X_{n+1} = F(X_n, U_n, \xi_n), \qquad (1.3)$$

де F – нелінійна вектор–функція.

1.2.3 Марківські моделі

Марківські моделі відіграють важливу роль при дослідженні ІС. Випадковий процес $y_t, t \geq 0$, заданий на деякому імовірнісному просторі і такий, що приймає значення в числовій множині Y, називається марківським, якщо для будь–якого натурального числа п, будь–яких $y, u, u_n, ..., u_1 \in Y$ і будь–яких $\tau, t, t_n, ..., t_1$ впорядкованих таким чином: $\tau > t > t_n > ... > t_1$ виконується співвідношення [1,55]:

$$P\{y_\tau < y \mid y_t = u, y_{t_n} = u_n, ..., t_{t_1} = u_1\} = P\{y_\tau < y \mid y_t = u\}. \qquad (1.4)$$

Параметр t процесу розглядатимемо як час. Якщо τ – майбутній момент

часу, t – настоящий момент часу, $t_k, (k = 1,...,n)$ – минулі моменти часу, то умову (1.4) можна трактувати таким чином: майбутня поведінка марківського процесу повністю визначається його станом теперішнього часу. Для немарківського процесу майбутня поведінка залежить також від станів процесу в минулому.

У випадку, якщо множина станів Y марківського процесу $y_t, t \geq 0$ є кінцевим, то марківський процес називається ланцюгом Маркова. Якщо параметр t приймає тільки дискретні значення, то ланцюг Маркова називається ланцюгом з дискретним часом. Якщо ж параметр t приймає значення в деякій безперервній множині, то ланцюг Маркова називається ланцюгом з безперервним часом.

Без обмеження спільності вважатимемо, що множина станів ланцюга є множиною не негативних цілих чисел (або його деяка підмножина). Тоді однорідний ланцюг Маркова $S_k, k > 1$ з дискретним часом зважає заданим, якщо:

– заданий початковий розподіл ймовірності станів ланцюга:

$$r_i = P\{S_0 = i\}, i \geq 0; \tag{1.5}$$

– задана матриця P ймовірностей однокрокових переходів ланцюга, що складається з елементів p_{ij}, визначених таким чином:

$$p_{ij} = P\{S_{k+1} = j \mid S_k = i\}, j > 0. \tag{1.6}$$

Матриця P однокрокової перехідної ймовірності p_{ij} є стохастичною, тобто її елементи є не негативними числами і сума елементів будь–якого рядка рівна одиниці. Матриця ймовірності переходів ланцюга за т кроків є m–м ступенем матриці P.

Позначимо через $P_i(k) = P\{S_k = i\}, k \geq 1, i \geq 0$ ймовірність того, що після k–го кроку ланцюг Маркова знаходиться в стані і. Потенційно ймовірності $P_i(k)$, що характеризують нестаціонарну поведінку ланцюга, можуть бути знайдені при вирішенні задачі моделювання динамічного об'єкту, що описується ланцюгом

Маркова. Проте знаходження такої ймовірності на практиці є складним, тому зазвичай аналізують так звану стаціонарну ймовірність станів ланцюга Маркова [27, 28]:

$$\pi_i = \lim_{k\to\infty} P_i(k), i \geq 0. \qquad (1.7)$$

Ці ймовірності називають також граничними (фінальними, ергодичними). Нижче розглядатимуться неперіодичні ланцюги для яких межі (1.7) існують. При цьому перераховані альтернативні назви стаціонарної ймовірності виражають практично одні і ті ж властивості ланцюгів: існування меж, не залежних від початкового розподілу ймовірності її станів, і існування єдиного позитивного вирішення наступної системи лінійних рівнянь (рівнянь рівноваги) алгебри для стаціонарної ймовірності:

$$\pi_i = \sum_{i=0}^{\infty} P_i(k), i \geq 0, \qquad \sum_{i=0}^{\infty} \pi_i = 1 \qquad (1.8)$$

Існує цілий ряд результатів (теореми Феллера, Фостера, Мустафи, Твіді і ін.), що дозволяють по конкретному виду перехідної ймовірності p_{ij} визначити, існує стаціонарна ймовірність чи ні. Якщо в результаті дослідження встановлено, що за деяких умов на параметри ланцюга p_{ij} межі існують, то ці умови вважаються виконаними, після чого система рівнянь рівноваги вирішується, і ланцюг Маркова вважається дослідженим.

1.2.4 Нечітка логіка

Нечіткі множини [29,30] є узагальненням звичайних множин для випадку, коли характеристична функція може приймати будь–які значення з відрізку [0, 1]. У теорії нечітких множин дана характеристична функція називається функцією приналежності, а її значення $\mu_\Lambda(x)$ – ступенем приналежності елементу x нечіткій множині A.

Існують різні типові форми кривих для задання функцій приналежності. Найбільшого поширення набули: трикутна, трапецеїдальна і гауссова функції приналежності.

Ключовими поняттями теорії нечітких множин є поняття нечіткої і лінгвістичної змінних.

Нечітка змінна визначається трійкою $<\alpha, U, \mu>$,

де α – назва змінної;

U – універсальна множина (область визначення α);

$\mu = \mu(x)$ – функція приналежності, що визначена на U і характеризує ступінь впевненості в тому, що x є значенням нечіткої змінної [31].

Лінгвістичною змінною називається набір $<\beta, T, U, G, M>$,

де β – назва лінгвістичної змінної;

T – множина її значень (терм–множина), що є найменуваннями нечітких змінних;

G – синтаксична процедура, що дозволяє оперувати елементами терм–множини T, зокрема, генерувати нові терми (значення);

M – семантична процедура, що дозволяє перетворити кожне нове значення лінгвістичної змінної, що утворюється процедурою G в нечітку змінну.

Нечітким логічним висновком (fuzzy logic inference) називається апроксимація залежності $Y = f(x_1, x_2, ..., x_n)$ вихідної лінгвістичної змінної від вхідних лінгвістичних змінних і отримання висновку у вигляді нечіткої множини, з використанням бази знань, що містить правила вигляду «Якщо ..., то...» [32].

Механізм логічного висновку полягає, в загальному випадку, з наступних етапів:

1) Фаззіфікація – визначення ступенів впевненості, тобто значення кожній з функцій приналежності терма при заданих значеннях вхідних змінних $x_k (k = 1, ..., n)$;

2) Нечіткий висновок – складається з двох етапів:

– визначення рівнів «відсікання» для лівої частини кожного з правил,

тобто значення функцій приналежності для лівих частин кожного правила («передумов»). В більшості випадків, це або максимум, або мінімум із ступенів впевненості термів, обчислених на етапі фаззифікації (логічні «АБО», «І»);

– визначення «відсічених» функцій приналежності. Для цього значення функцій приналежності передумов об'єднуються з відповідними функціями приналежності з правих частин правил за правилом «логічного І»;

3) Нечітка композиція – визначення результуючої функції приналежності всієї сукупності правил, тобто об'єднання отриманих відсічених функцій (зазвичай за правилом «логічного АБО»);

4) Дефаззіфікація – приведення до «чіткості», використовуючи результуючу функцію приналежності. Основним методом деффазифікації є центроїдний (centroid) – знаходження центру тяжіння плоскої фігури, обмеженої осями координат і графіком функції приналежності нечіткої множини.

1.2.5 Мережі Петрі

Мережі Петрі – апарат для моделювання динамічних дискретних систем (переважно, асинхронних паралельних процесів) [22, 33]. Мережа Петрі визначається як $< P,T,I,O >$, де P і T – кінцева множина позицій і переходів, I і O – множина вхідних і вихідних функцій. Іншими словами, мережею Петрі є дводольний орієнтований граф, в якому позиціям відповідають вершини, що зображаються кружками, а переходам – вершини, що зображаються потовщеними рисками; функціям I відповідають дуги, направлені від позицій до переходів, а функціям O – дуги від переходів до позицій [34].

Як і в системах масового обслуговування в мережах Петрі вводяться об'єкти двох типів: динамічні – зображаються мітками (маркерами) всередині позицій і статичні – їм відповідають вершини мережі Петрі.

Розподіл маркерів по позиціях називають маркіровкою. Маркери можуть переміщатися в мережі. Кожну зміну маркіровки називають подією, причому кожна подія пов'язана з певним переходом. Вважається, що події відбуваються миттєво і різночасно при виконанні деяких умов.

Кожній умові в мережі Петрі відповідає певна позиція. Здійсненню події

відповідає спрацьовування (збудження або запуск) переходу, при якому маркери з вхідних позицій цього переходу переміщаються у вихідні позиції. Послідовність подій утворює модельований процес.

Правила спрацьовування переходів конкретизують таким чином: перехід спрацьовує, якщо для кожної з його вхідних позицій виконується умова $N_i \geq K_i$, де N_i – число маркерів в *i*–й вхідній позиції; K_i – число дуг, що йдуть від *i*–ї позиції до переходу. При спрацьовуванні переходу, число маркерів в *i*–й вхідній позиції зменшується на K_i, а в *j*–й вихідній позиції збільшується на M_j, де M_j – число дуг, що пов'язують перехід з *j*–ю позицією.

Можна вводити ряд додаткових правил і умов в алгоритми моделювання, отримуючи той або інший різновид мереж Петрі. Так, перш за все корисно ввести модельний час, щоб моделювати не тільки послідовність подій, але і їх прив'язку до часу. Це здійснюється доданням переходам ваги – тривалості (затримки) спрацьовування, яку можна визначати, використовуючи алгоритм, що задається при цьому. Отриману модель називають тимчасовою мережею Петрі.

Якщо затримки є випадковими величинами, то мережу називають стохастичною мережею Петрі. У стохастичних мережах можливе введення ймовірності спрацьовування збуджених переходів.

Якщо затримки визначаються як функції деяких аргументів, якими можуть бути кількості маркерів в яких–небудь позиціях, стани деяких переходів і т. д., то маємо функціональну мережу Петрі (рисунок 1.2).

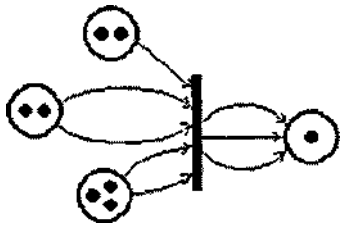

Рисунок – 1.2 – Фрагмент мережі Петрі

У багатьох задачах динамічні об'єкти можуть бути декількох типів, і для кожного типу потрібно вводити свої алгоритми поведінки в мережі. В цьому випадку кожен маркер повинен мати хоч би один параметр, що позначає тип маркера. Такий параметр зазвичай називають кольором; колір можна

використовувати як аргумент у функціональних мережах. Мережу при цьому називають кольоровою мережею Петрі.

1.2.6 Нечіткі когнітивні карти

Як вже наголошувалося, інформаційні системи є складними гетерогенними комплексами, функціонування яких не може бути описане в детермінованому вигляді. Моделювання багатьох аспектів функціонування подібних систем зазвичай зв'язане з використанням знань експертів, описаних в нечіткій формі. Для того, щоб врахувати подібні аспекти ІС, в даній роботі пропонується використовувати метод нечітких когнітивних карт (НКК). Його суть зводиться до виділення сукупності основних чинників (концептів), що позначають різні поняття модельованої предметної області і побудови на їх основі орієнтованого графа, що відображає взаємозв'язки між концептами [35, 36]. Кожному *i*–му концепту нечіткої когнітивної карти ставляться у відповідність змінна стану X_i і вага w_{jj}, що позначає вплив *i*–го концепту на *j*–й концепт. Величина ваги w_{jj} лежить в межах [0;1] і характеризує ступінь значущості (впливу) відповідного концепту. Змінні стани концептів моделі динамічно змінюються в часі відповідно до формули:

$$X_i(k+1) = f\left(X_i(k) + \sum_{\substack{j=1 \\ j\neq 1}}^{n} w_{ij} X_j(k)\right), \tag{1.9}$$

де $X_i(k)$ – змінна стану *i*–го концепту у момент часу *k*;

w_{jj} – вага зв'язку між концептами *i* і *j*;

f(x) – нелінійна функція, яка зазвичай визначається як:

$$f(x) = \frac{1}{1+e^{-\lambda x}}, \tag{1.10}$$

де коефіцієнт λ підбирається емпірично при побудові і відладці

моделі.

Для використання НКК з метою оцінки ризику функціонування ІС виділяються 4 групи концептів, що позначають різні чинники поведінки ІС:

1) дестабілізуючі чинники (ДЧ) – відображають сукупність чинників, що негативно впливають на ІС;

2) проміжні чинники (ПЧ) – сукупність об'єктів, груп об'єктів, або параметрів, що описують стан ІС;

3) цільові чинники (ЦЧ) – описують різні негативні наслідки (збиток), дії ДЧ, що виникають в результаті, на ІС;

4) керуючі чинники (КЧ – моделюють керуючі захисні механізми ІС.

Для визначення впливу i–го ДЧ на j–й ЦЧ розраховується величина повного ефекту T_{ij}, яка знаходиться по наступній формулі:

$$T_{ij} = \max T_{ij}^{l}, \qquad (1.11)$$

де T_{ij}^{l} – мінімальна вага зв'язку l–го шляху між вершинами (концептами) i і j нечіткої когнітивної карти.

Оцінка сукупного впливу всіх ДЧ на певний (i–й) ЦЧ описується величиною повного (комплексного) ефекту:

$$T_i^{l} = \max_{i \le j \le k} (T_{jk}), \qquad (1.12)$$

де k – число ДЧ.

Для оцінки повного ризику функціонування ІС можна скористатися формулою:

$$R = \sum_{i=1}^{N} T_i^{l} r_i v_i, \qquad (1.13)$$

де N – загальне число ЦЧ моделі;

r_i – величина можливого збитку від реалізації i–го ЦЧ;

v_i – значущість i–го ЦЧ.

Використання методології SADT є одним з найбільш зручних способів моделювання елементів ІС на етапі їх проектування, оскільки дозволяє виявити їх структуру, основні функції, залежності між компонентами, основні інформаційні активи, а також формат і зміст структур даних, якими вони обмінюються. Використання моделей IDEF–CPN (або їх аналогів) дозволяє провести первинний аналіз роботи алгоритмів, що реалізовуються, і провести попередню оцінку їх ефективності.

Використання методів теорії систем цілком можливе для моделювання окремих елементів інформаційної системи, проте побудова комплексної моделі з їх допомогою є практично неможливою внаслідок того, що ІС є складною системою, процеси в якій носять стохастичний характер. Даний математичний апарат незручний також тим, що не дозволяє використовувати знання експертів, які, як правило, сформульовані в нечіткому вигляді.

Марківська модель представляє досліджувану систему у вигляді процесу зміни її дискретних станів, що зручно для моделювання ІС, оскільки природа даної системи в своїй суті дискретна. Основним недоліком даного методу є те, що кінцевий результат моделювання представлений у вигляді перехідної і фінальної ймовірності, що не дозволяє проводити комплексну оцінку стану ІС. Набагато зручнішою областю застосування Марківських моделей є моделювання атак і відмов [37] різних компонентів ІС. Отримані імовірнісні характеристики можуть бути використані для розрахунку характеристик дестабілізуючих збурень і можуть застосовуватися в моделі ІС більш високого рівня [26, 38].

Моделі на основі мереж Петрі по суті своїй схожі з Марківськими моделями за тим виключенням, що на виході отримуємо не імовірнісну характеристику об'єкту дослідження, а послідовність зміни стану компонентів моделі (шлях руху фішок по вершинах графа). Подібне моделювання вельми зручне для дослідження роботи алгоритмів різних компонент ІС, проте не зовсім підходить для використання у вигляді базової моделі, оскільки не дає інформації про

ефективність роботи системи.

Математичний апарат нечітких когнітивних карт дозволяє використовувати знання експертів у вигляді нечіткої бази правил і сукупності ваг зв'язків концептів, моделювати динамічну роботу системи шляхом спостереження зміни ваг концептів і оцінювати ризики функціонування моделі. За своєю суттю, даний підхід до моделювання дозволяє узагальнити результати моделювання різних компонент ІС будь–яким з вище перерахованих методів (що генерують в результаті своєї роботи залежність зміни ваги концепту від часу). В даний час існує декілька робіт, що використовують теорію нечітких когнітивних карт для визначення величин ризиків складних систем [35, 36]. Основними недоліками методик оцінки ризиків в даних роботах є наступні:

1) відсутній механізм уточнення експертних знань (зайва нечіткість), що може привести до отримання недостовірних результатів моделювання;

2) величини ризиків не залежать від ваг концептів і визначаються на основі ваг зв'язків;

3) відсутня можливість динамічного моделювання ризиків, у зв'язку з чим відсутня можливість прогнозування стану модельованої системи.

1.3 Методи протидії атакам на інформаційні системи

Під протидією атаці розуміється комплекс заходів, направлених на виявлення, реєстрацію і блокування атаки з метою збереження безпеки інформації [8]. Таким чином, протидія атакам має дві фази – пасивну і активну. В ході пасивної фази система захисту здійснює пошук і ідентифікацію атак, використовуючи наступні методи [39, 40]:

- аналіз сигнатур активності;
- статистичні методи аналізу (аналіз поведінки);
- динамічний (інтелектуальний) аналіз.

Метод аналізу сигнатур має на увазі наявність в системі захисту бази знань сигнатур, що є формалізованими описами можливих видів зловмисної активності. Даний метод виявлення атак є одним з найпоширеніших через відносну простоту

його реалізації і високу швидкість пошуку. Головним недоліком даного методу є неможливість виявлення атак, сигнатури яких відсутні в базі даних.

Статистичні методи аналізу націлені на пошук активності, що відрізняється від звичайної для даної інформаційної системи. Дані методи аналізу дозволяють виявляти атаки, невідомі системі захисту, проте володіють високою ймовірністю помилок першого і другого роду, а також не дозволяють здійснювати пошук атак в режимі реального часу.

Динамічний аналіз орієнтований на пошук активності, схожої на атаки, які відомі системі. Даний метод аналізу вважається інтелектуальним, оскільки він містить алгоритми узагальнення і класифікації даних і дозволяє генерувати нові знання з тих, що існують. Через складність первинної настройки і низьку швидкість донавчання системи захисту, в чистому вигляді даний метод аналізу практично не застосовується. Як правило, методи динамічного аналізу комбінуються з сигнатурними, причому база сигнатур виступає як сховище знань. Даний метод аналізу називається гібридним або комбінованим.

Після виявлення і ідентифікації атаки, система захисту здійснює її реєстрацію шляхом:

- запису інформації про атаку в системний журнал подій;
- генерації повідомлення на консоль адміністратора безпеки;
- розсилки повідомлень по електронній пошті, через служби доставки миттєвих повідомлень (ICQ), SMS і т. д.

В ході активної фази протидії атаці здійснюється її придушення шляхом:

- завершення сесії користувача системи, що здійснює атаку;
- блокування облікового запису зловмисника;
- дезінформації зловмисника;
- зміни конфігурації маршрутизаторів і міжмережевих екранів;
- придушення локальних проявів атаки;
- перенастроювання антивірусних програм;
- атаки у відповідь на систему зловмисника.

1.4 Постановка задачі дослідження

Збільшення ролі ІС в задачах збору, аналізу і обробки інформації приводить до необхідності приділяти активну увагу вирішенню проблем інформаційної безпеки. Вирішення даних проблем можливе шляхом використання різних захисних механізмів, одним з яких є СВА. Основними перевагами даного механізму захисту в порівнянні з традиційними засобами забезпечення інформаційної безпеки є: можливість активного моніторингу загроз всередині демілітаризованої зони ІС ЛОМ в режимі реального часу; протидія широкому спектру атак, як на локальних хостах, так і на мережевих маршрутизаторах; можливість використання даних систем в якості проміжного модуля, що управляє, для інших систем захисту ІС.

Аналіз функціональних можливостей існуючих систем виявлення атак показує, що їм властиві наступні системні недоліки:

1) локальність обхвату – більшість існуючих систем виявлення атак забезпечують захист певного сегменту ІС ЛОМ, повністю ігноруючи всі інші аспекти, що може привести до появи «дірок» в системі захисту;

2) негнучкість архітектури – СВА, що існують не дозволяють враховувати накопичені експертні знання про структуру, архітектуру і призначення ІС, що може привести до низької ефективності їх функціонування;

3) сильна залежність від виробника – більшість СВА здійснюють виявлення атак, використовуючи сигнатурні методи аналізу, що приводить до появи залежності рівня інформаційної безпеки ІС від оперативності оновлення сигнатурних баз фірмою-виробником СВА.

Метою дипломної роботи є підвищення ефективності виявлення атак на ІС на основі оперативної оцінки ризиків функціонування ІС з використанням динамічних моделей на основі нечітких когнітивних карт.

Для досягнення поставленої мети в роботі були поставлені і вирішені наступні завдання:

1. Розробка системних моделей функціонування ІС з використанням SADT методології.

2. Синтез архітектури і алгоритмів системи прийняття рішень на основі динамічної моделі оцінки ризиків з використанням нечітких когнітивних карт.

3. Розробка дослідницького прототипу інтелектуальної системи виявлення атак.

4. Аналіз ефективності функціонування розробленої інтелектуальної системи виявлення атак методом імітаційного моделювання.

2 МОДЕЛІ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Розробка системних моделей ІС

2.1.1 Розробка онтологічної моделі ІС

Онтологічним аналізом предметної області є процес формалізації її об'єктів з метою виділення основних концептуальних класів, властивих їй, їх взаємовідносин і взаимозв’язків. Онтологічна модель інформаційної системи дозволяє виявити її основні компоненти, їх суть і взаємозв'язки і використовувати їх надалі при побудові функціональних і динамічних моделей ІС.

Онтологічний аналіз починається з складання словника термінів, який використовується при обговоренні і дослідженні характеристик об'єктів і процесів, які складають дану систему, а також створення системи точних визначень цих термінів. Крім того, документуються основні логічні взаємозв'язки між відповідними введеними поняттями. Результатом цього аналізу є онтологія системи, або ж сукупність словника термінів, точних визначень взаємозв'язків між ними.

У будь-якій системі існують дві основні категорії предметів сприйняття, такі як самі об'єкти, які складають систему (фізичні і інтелектуальні), і взаємозв'язки між цими об'єктами, що характеризують стан системи. В термінах онтології, поняття взаємозв'язку однозначно описує або, іншими словами, є точним дескриптором залежностей між об'єктами системи в реальному світі, а терміни – є, відповідно, точними дескрипторами самих реальних об'єктів.

На рисунку 2.1 показана онтологічна модель інформаційної системи ЛОМ підприємства у вигляді графа зв'язків.

2.1.2 Розробка функціональної моделі ІС

Для опису ІС скористаємося методологією SADT, описаною в першому розділі роботи. Функціональна модель інформаційної системи показана на рисунку 2.2.

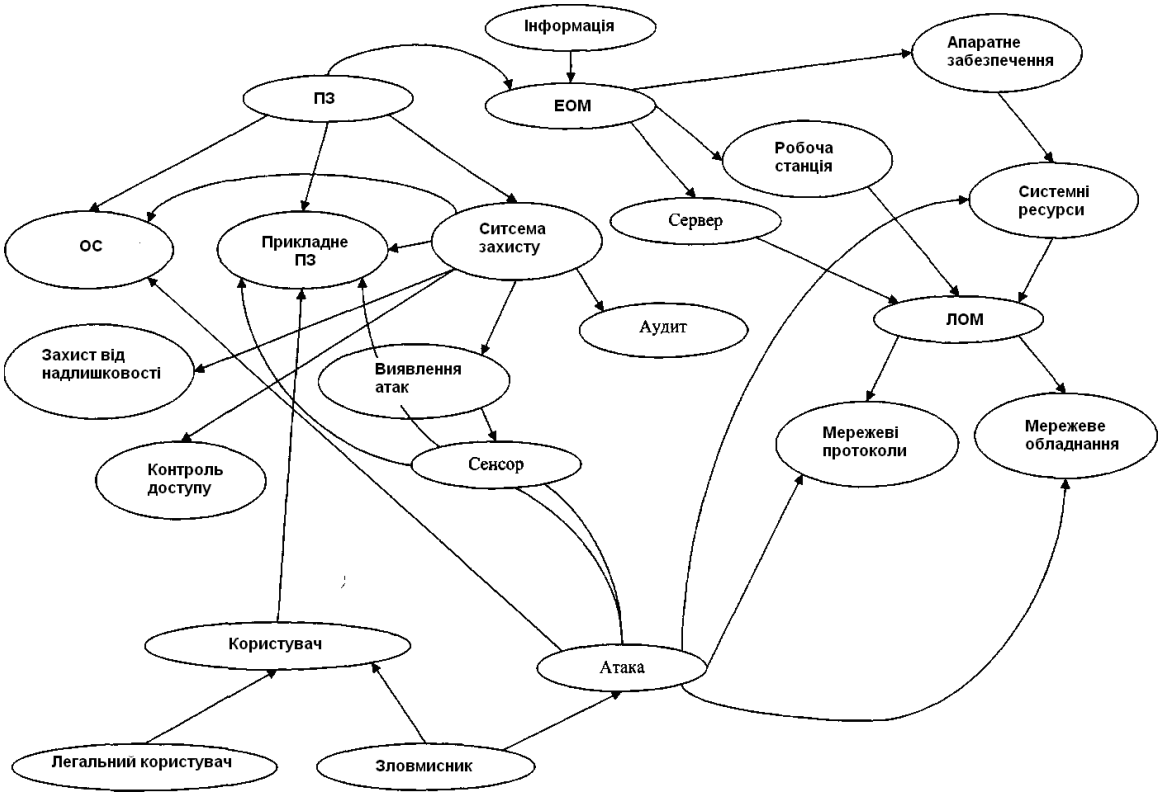

Рисунок 2.1 – Онтологічна модель

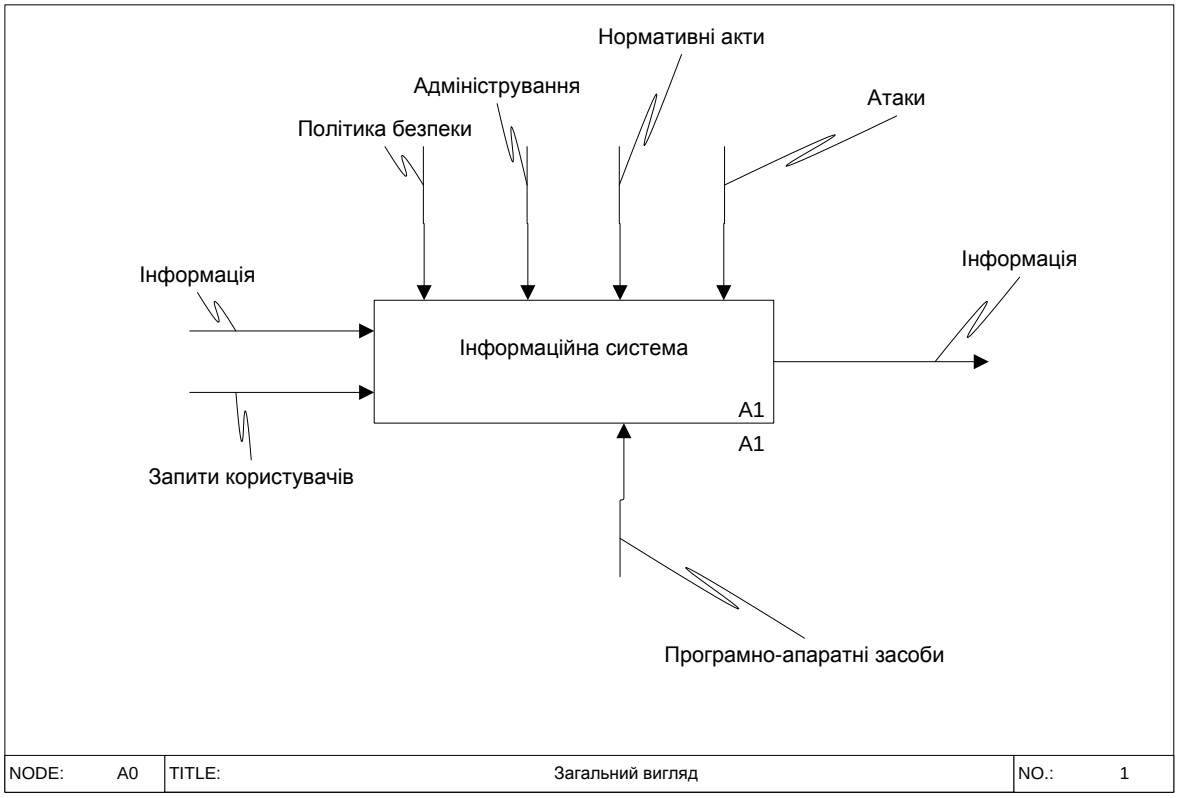

Рисунок 2.2 – Функціональна модель ІС

ІС є «чорним ящиком», на входи якого, поступає інформація і запити користувачів. Матеріально-технічною базою, що забезпечує функціонування ІС є програмно-апаратне середовище підприємства. Управління ІС здійснюється на нормативному (нормативні акти, політика безпеки) і адміністративному рівні (команди адміністраторів і користувачів системи). В процесі функціонування ІС піддається атакам, що є як некоректними командами користувачів ІС, так і зловмисними діями (див. розділ 1). Результатом функціонування ІС є оброблена інформація. Деталізована функціональна модель ІС показана на

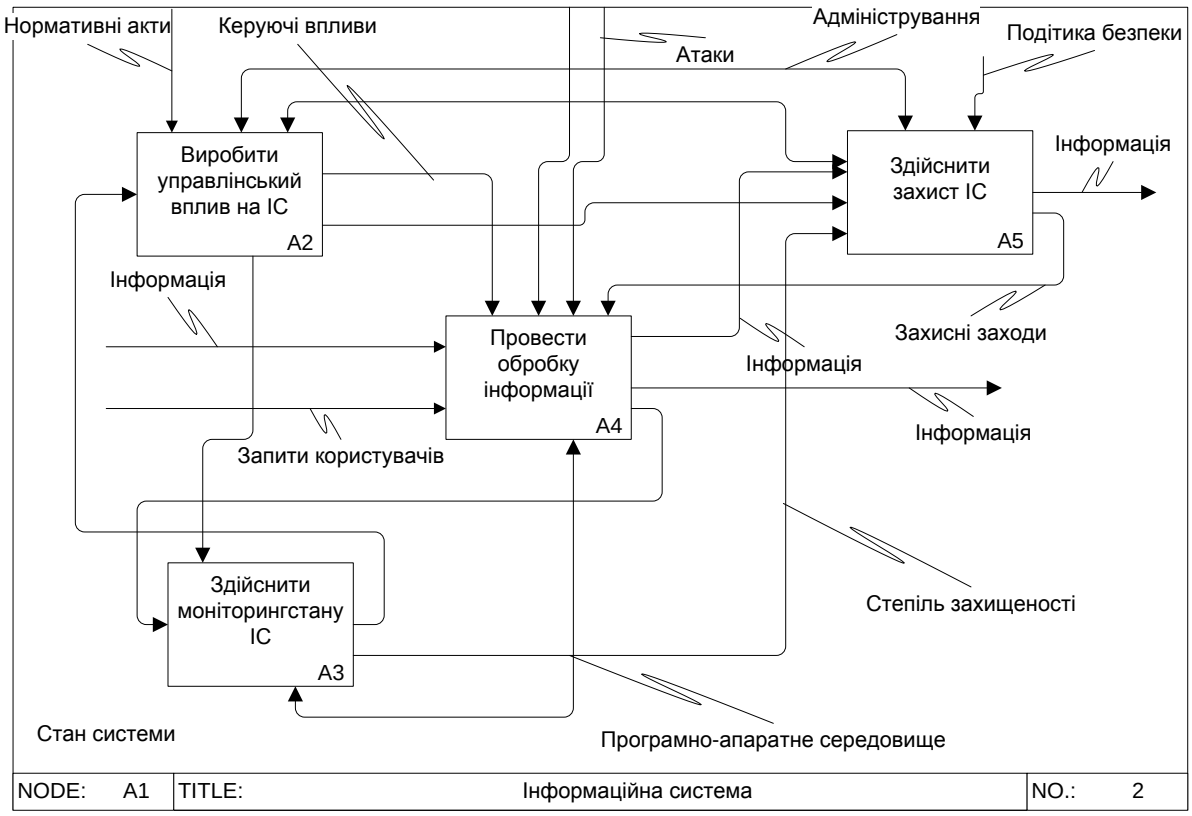

Рисунок 2.3 – Деталізована функціональна модель ІС

Відповідно до рисунку 2.3, ІС складається з чотирьох базових підсистем, що забезпечують її функціонування:

– система управління ІС – здійснює налаштування і управління компонентів ІС;

– система обробки інформації – забезпечує реалізацію функцій ІС – збір, зберігання і обробку інформації;

– система моніторингу ІС – призначена для здійснення перевірки коректності функціонування компонентів ІС;

– система захисту ІС – відповідає за захист і відновлення працездатності компонентів ІС у разі атак і збоїв.

Необхідно відзначити, що система захисту не реагує на атаки безпосередньо, вона отримує інформацію про них від системи моніторингу ІС, що виявляє їх по ознаках некоректної поведінки інформаційної системи. Таким чином, компоненти захисту і моніторингу мають тісний функціональний зв'язок і в більшості реальних систем об'єднуються у складі різних програмно-апаратних засобів захисту.

Функціональна модель системи захисту ІС показана на рисунку 2.4.

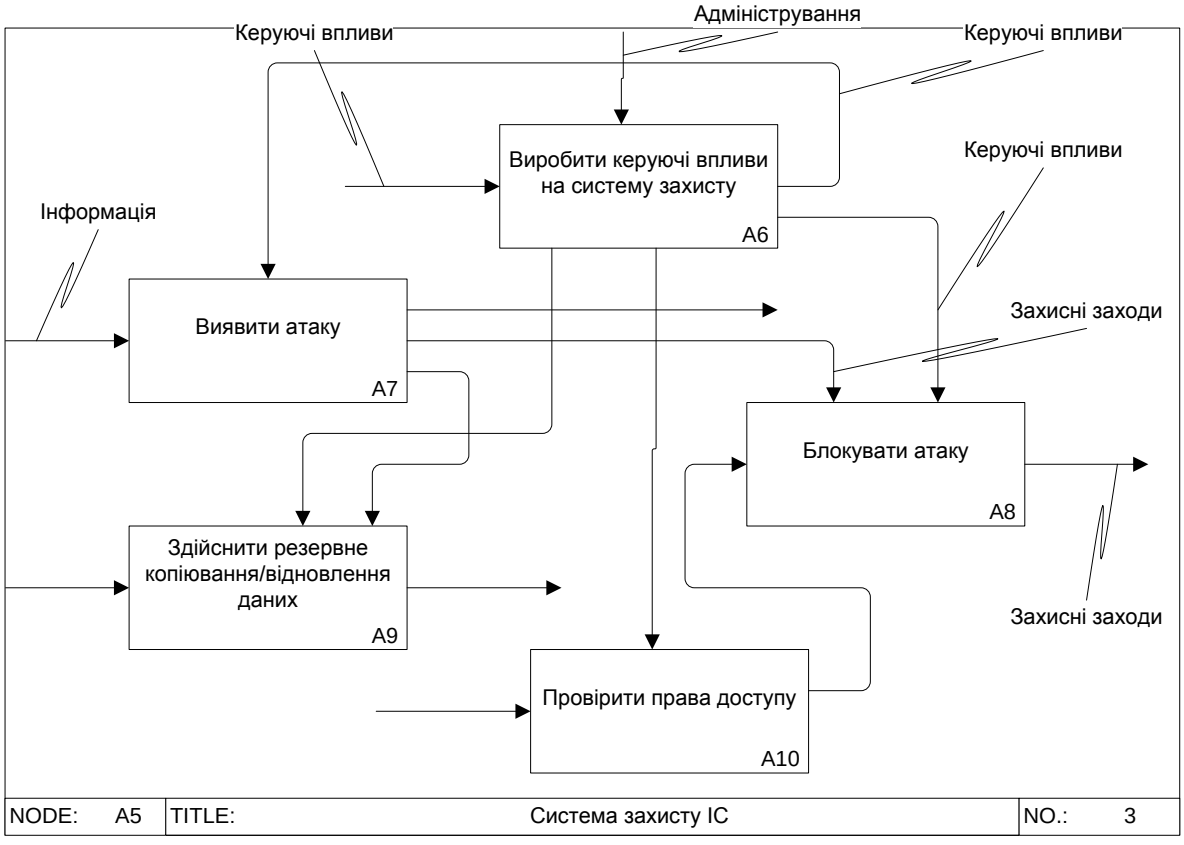

Рисунок 2.4 – Функціональна модель системи захисту ІС

Відповідно до даного рисунку, система захисту ІС містить наступні компоненти:

– система управління – здійснює налаштування і управління компонентами системи захисту ІС;

– система резервного копіювання – призначена для відновлення ІС у разі реалізації загроз інформаційній безпеці;

– система контролю доступу – реалізує функції контролю доступу до інформації і компонентів ІС;

– система виявлення атак – аналізує потік вхідних інструкцій користувача і виявляє атаки;

– система блокування атак – здійснює блокування виявлених атак.

Дана модель має на увазі чітке розділення процедур виявлення атаки (тобто класифікації певних дій користувача як зловмисні) і її блокування (формування списку контрзаходів, що забезпечують блокування атаки). Система резервного копіювання забезпечує можливість усунення збитку у разі реалізації загроз інформаційній безпеці.

Функціональна модель системи виявлення атак показана на рисунку 2.5.

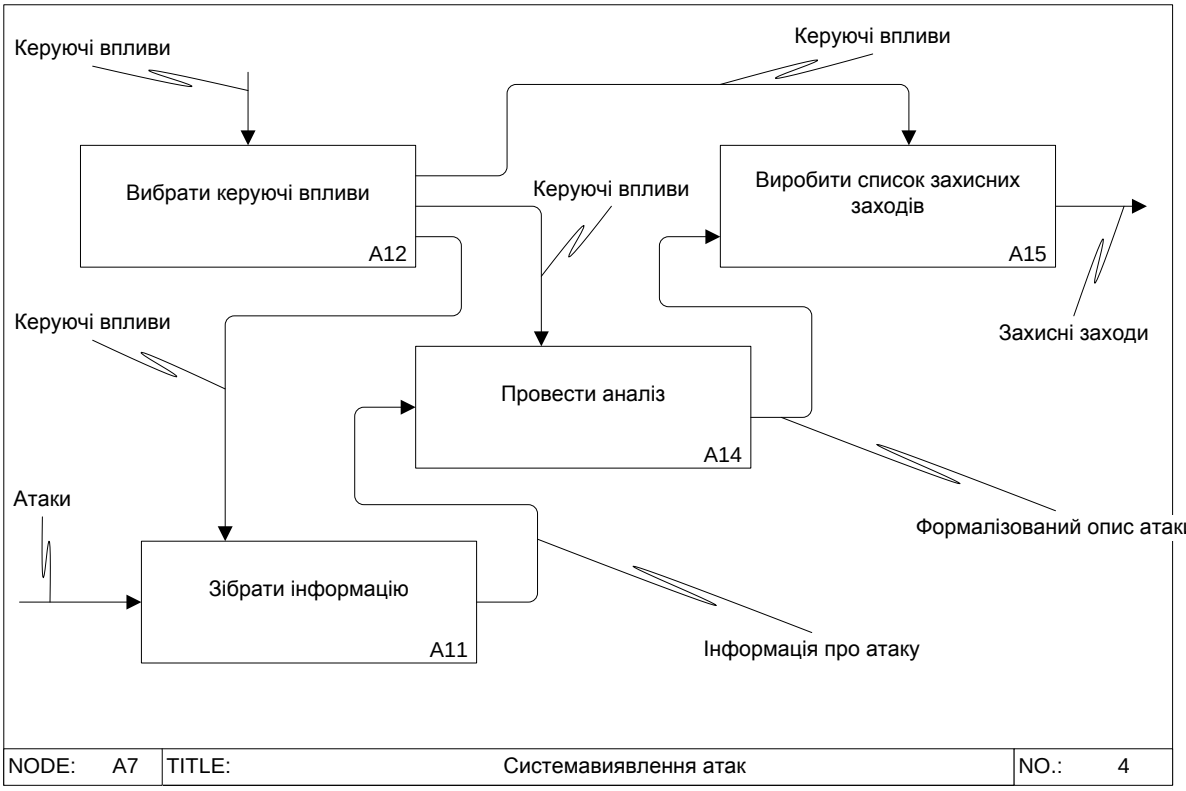

Рисунок 2.5 – Функціональна модель СВА

Дана модель складається з наступних базових компонент:

– сенсори СВА – здійснюють збір інформації про стан об'єкту, що

захищається ІС;

– модуль аналізу даних – проводить формалізацію даних і ухвалює рішення про класифікацію певної активності як атаки;

– модуль ухвалення рішень – формує список захисних заходів, призначених для придушення атаки;

– консоль адміністратора – призначена для налаштування параметрів СВА.

Функціональна модель сенсора СВА показана на рисунку 2.6.

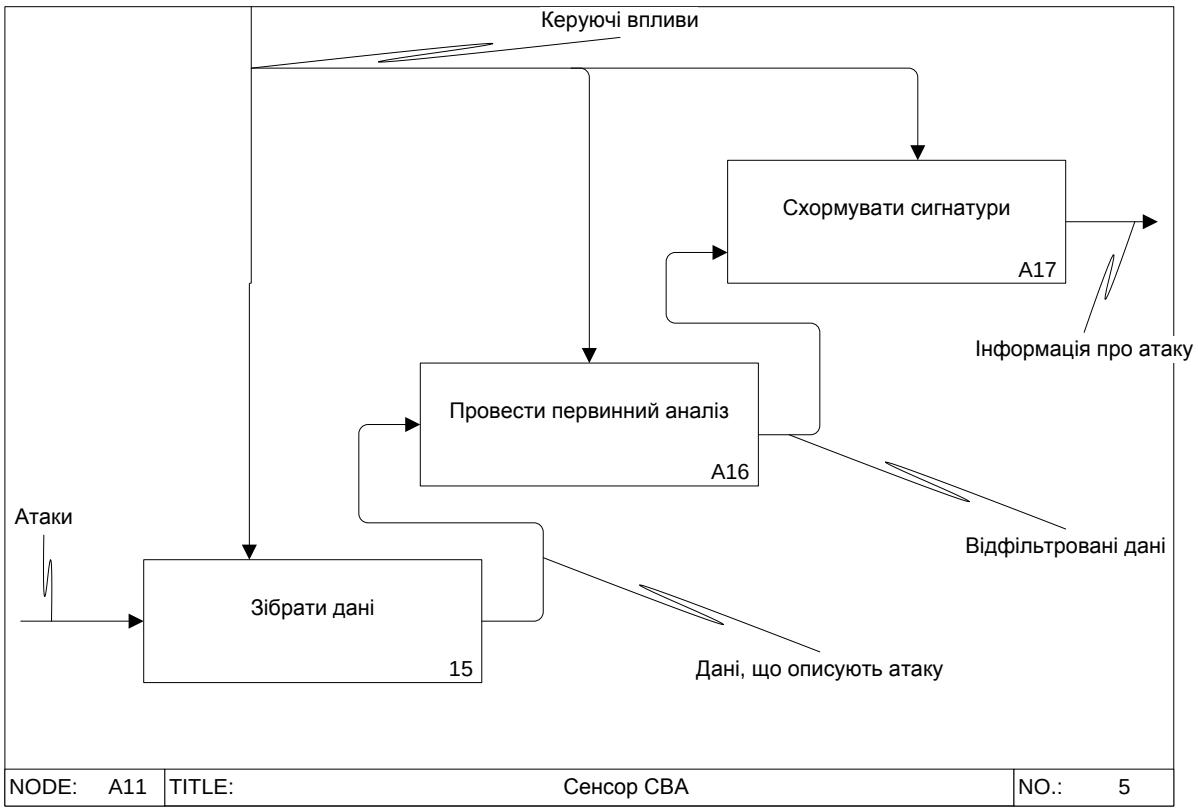

Рисунок 2.6 – Функціональна модель сенсора СВА

2.1.3 Розробка динамічної моделі ІС

Динамічне моделювання ІС є важливим етапом побудови математичної моделі інформаційної системи, оскільки дозволяє формалізувати дію на неї випадкових дестабілізуючих чинників (загроз). Внаслідок того, що функціонування ІС є яскраво вираженим стохастичним процесом через випадковий характер подій (атаки на ЛОМ, виявлення вразливостей в системному ПЗ, порушення політики безпеки і т. д.), що відбуваються в ній, моделювання даної системи зручно здійснювати за допомогою математичного апарату

марківських моделей [1, 26]. В даний час існує декілька підходів до побудови подібних моделей, в числі яких можна назвати:

1) класичні марківські моделі;

2) приховані марківські моделі;

3) марківські моделі *k*-го порядку;

4) марківські моделі, засновані на самоспостереженні [41];

5) напівмарківські моделі для нестаціонарних процесів.

Для оцінки сукупного рихзику функціонування системи скористаємося прихованою марківською моделлю першого порядку з дискретним часом функціонування [42].

Позначимо через *M* множину станів ІС:

$$M = \{S_1, S_2, S_3, S_4, S_5, S_6\}, \quad (2.1)$$

де S_1 – стан, в якому система вважається невразливою, тобто атака на будь-який її компонент в даний момент часу неможлива;

S_2 – стан вразливості ІС, тобто ІС містить компоненти, атака на які може бути реалізована в даний момент часу;

S_3 – стан усунення вразливості, тобто здійснюється процес оновлення програмного забезпечення компонентів ІС з метою усунення виявлених вразливостей;

S_4 – стан зовнішньої атаки на ІС, тобто зловмисник в даний момент часу здійснює атаку на зовнішню систему захисту ІС;

S_5 – стан внутрішньої атаки на ІС, тобто зловмисник здійснює атаку на внутрішні об'єкти ІС;

S_6 – стан відновлення системи після атаки.

Кожен з перерахованих станів S_i системи можна охарактеризувати вектором наслідків реалізації загроз B_i

$$B = \{o_0, o_1, o_2, o_3\}, \quad (2.2)$$

де o_0 – відсутність впливу атаки на АС (невдала атака, або відсутність атаки);

o_1 – порушення конфіденційності інформації;

o_2 – порушення цілісності інформації;

o_3 – порушення доступності інформації.

Ймовірність прояву *j*-го наслідку *i*-го стану S_i задається функцією $b_i(o_j)$, причому $\sum_{j=0}^{3} b_i(o_1) = 1, (i = 1,2,...,6)$.

Прихована марківська модель, що описує можливі переходи ІС з одного стану в інший, має вид направленого графа, представленого на рисунку 2.7. Для спрощення вважатимемо, що елементи (підсистеми) ІС можуть в один і той же час знаходитися тільки в одному з перерахованих вище станів $S_i(i = 1,2,...,6)$, тобто ІС є елементарною.

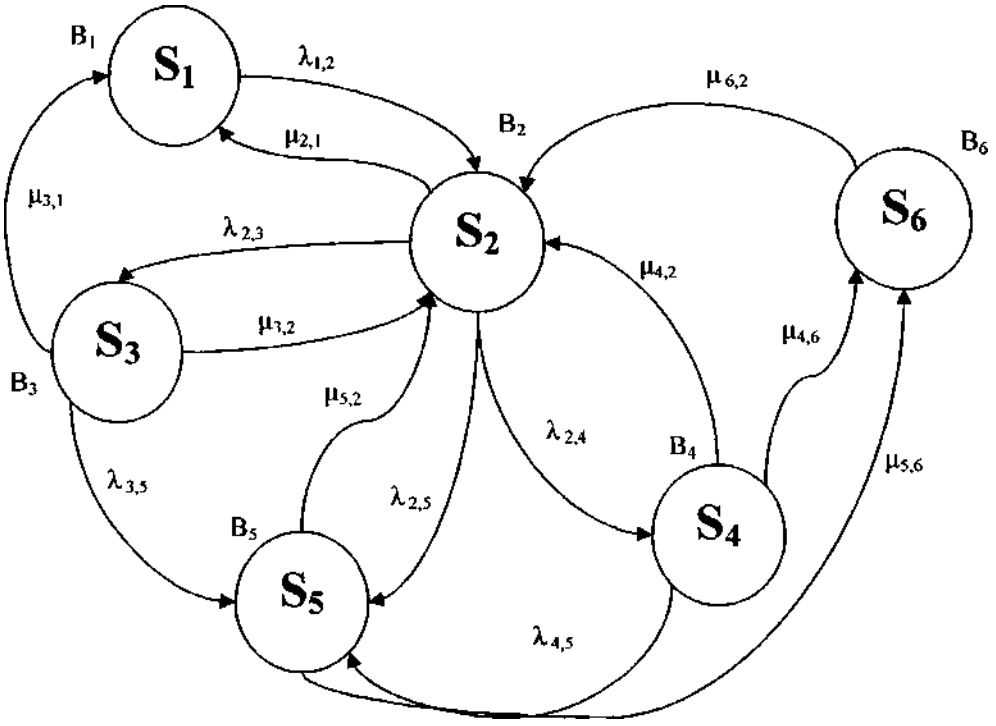

Рисунок – 2.7 Граф переходів між станами

Модель називається «прихованою», оскільки вона не дозволяє безпосередньо відстежувати зміну станів системи, будучи чорним ящиком, на

виході якого формується послідовність спостережень $O = (o_0, o_1, ..., o_t, ..., o_T)$, що містить наслідки реалізації загроз в моменти часу *t=0,1...,T.* Задача моделювання полягає в оцінці показників ризиків функціонування ІС і оптимізації параметрів моделі з метою мінімізації ризиків. Вважатимемо, що матриця інтенсивностей переходів для даної моделі виглядає таким чином:

$$\Lambda = \begin{pmatrix} \sum_1 & \lambda_{1,2} & 0 & 0 & 0 & 0 \\ \mu_{2,1} & \sum_2 & \lambda_{2,3} & \lambda_{2,4} & \lambda_{2,5} & 0 \\ \mu_{3,1} & \mu_{3,2} & \sum_3 & 0 & \lambda_{3,5} & 0 \\ 0 & \mu_{4,2} & 0 & \sum_4 & \lambda_{4,5} & \mu_{4,6} \\ 0 & \mu_{5,2} & 0 & 0 & \sum_5 & \mu_{5,6} \\ 0 & \mu_{6,2} & 0 & 0 & 0 & \sum_6 \end{pmatrix}, \qquad (2.3)$$

де λ_{ij} і μ_{ij} – відповідно інтенсивності прямих і зворотних переходів із стану S_i в стан S_j і навпаки.

$$\sum_i = 1 - \sum_{\substack{j=1 \\ (j \neq 1)}}^{N} \lambda_{i,j}; \qquad (2.4)$$

де N – число різних станів (у нашому прикладі N=6).

Оцінимо ймовірність збереження безпечного стану системи (формування на виході моделі послідовності $O = (o_0, o_0, o_0 ..., o_0)$ за час $t=T$. Введемо пряму змінну $\alpha_t(i)$, що виражає ймовірність того, що до моменту часу t спостерігалася послідовність O і система знаходилася в стані S_i, тобто $\alpha_t(i) = P(O, S_i \mid \Lambda)$. Дану змінну можна визначити таким чином:

$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(i) \Lambda_{i,j} \right] b_i(o_{t+1}). \qquad (2.5)$$

Шукану ймовірність безпечного стану системи можна визначити як суму прямих змінних для кожного стану системи:

$$P(O \mid \Lambda) = \sum_{t=1}^{N} \alpha_t(i). \tag{2.6}$$

Оцінимо ризик функціонування АС за час T таким чином:

$$R(O \mid \Lambda) = 1 - P(O \mid \Lambda) = 1 - \sum_{t=1}^{N} \alpha_t(i) \tag{2.7}$$

Для оптимізації параметрів марківської моделі з метою підвищення ймовірності появи послідовності спостережень O можна скористатися алгоритмом Баума-Уелша [27]. Метою оптимізації при цьому є максимізація цільової функції (2.6) (мінімізація ризиків (2.7)), в якості змінних параметрів виступають інтенсивності переходів λ_{ij} і ймовірності реалізації загроз $b_i(o_j)$. Введемо зворотну змінну $\gamma_t(i)$, яка позначає умовну ймовірність спостереження послідовності O з моменту часу t+1 до моменту T за умови, що у момент t система знаходилася в стані S_i. Приймемо $\gamma_t(i) = 1$, тоді:

$$\gamma_t(i) = \sum_{j=1}^{N} \Lambda_{i,j} b_j(o_{t+1}) \gamma_{t+1}(j). \tag{2.8}$$

Позначимо через $\xi_t(i, j)$ ймовірність того, що при заданій послідовності O система у моменти часу t і t+1 знаходиться в станах S_i і S_j. З врахуванням (2.5) і (2.8), дану ймовірність можна визначити таким чином:

$$\xi_t(i,j) = \frac{\alpha_t(i)\Lambda_{i,j}b_j(o_{t+1})\gamma_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\Lambda_{i,j}b_j(o_{t+1})\gamma_{t+1}(j)} \tag{2.9}$$

Ймовірність того, що у момент часу t при заданій послідовності O система знаходитиметься в стані S_i, рівна

$$\delta_t(i) = \sum_{j=1}^{N}\xi_t(i,j). \tag{2.10}$$

Очевидно, що сума $\sum_{t=1}^{\Gamma-1}\delta_t(i)$ є очікуваним числомпереходів із стану S_i за час T. Аналогічно $\sum_{t=1}^{\Gamma-1}\xi_t(i,j)$ - очікуване число переходів між станами S_i і S_j. Використовуючи отримані залежності, можна обчислити наступні оптимальні значення параметрів марківської моделі:

$$\Lambda_{i,j} = \frac{\sum_{t=1}^{l-1}\xi_t(i,j)}{\sum_{t=1}^{l-1}\delta_t(i)}; \qquad b_j(k) = \frac{\sum_{\substack{t=1\\ o_t=k}}^{l-1}\gamma_t(i,j)}{\sum_{t=1}^{l-1}\delta_t(i)}. \tag{2.11}$$

Як показує аналіз виразів (2.9), в загальному випадку, можливе підвищення ефективності роботи АС за рахунок дії двох чинників:

1) Оптимізація перехідних інтенсивностей матриці Λ. Внаслідок того, що інтенсивності переходів $\lambda_{i,j}$ носять випадковий характер і залежать від чинників, зовнішніх по відношенню до АС, їх оптимізація зазвичай є скрутною. Виграш по захищеності можна отримати за рахунок підвищення інтенсивностей відновлення $\mu_{i,j}$.

2) Оптимізація ймовірності $b_i(o_j)$.

На практиці перехід від початкової моделі до її оптимізованого варіанту забезпечується оптимізацією функціонування СВА в ЛОМ АС. Так, оптимізація ймовірності блокування внутрішньої атаки $b_5(o_0)$, інтенсивностей відновлення $\mu_{4,2}$ і $\mu_{5,2}$ здійснюється за рахунок застосування:

1) розподіленої мережі сенсорів СВА, які забезпечують виявлення і класифікацію шкідливої активності;

2) модулів класифікації невідомої активності, які забезпечують виявлення і блокування невідомих атак СВА.

Ризик функціонування АС можна також зменшити за рахунок підвищення інтенсивностей відновлення $\mu_{4,6}$, $\mu_{5,6}$ і $\mu_{6,2}$ за рахунок використання в СВА підсистеми блокування успішних атак і виявлення прихованих каналів просочування інформації (модуль активного аудиту). Накопичення знань (використання баз знань) в процесі функціонування СВА дозволить забезпечити монотонне зростання цільової функції оптимізації функціонування системи (4) і дозволить мінімізувати загальний ризик функціонування (5).

2.2 Розробка моделі атак на ІС

У даній роботі пропонується здійснювати моделювання атак на компоненти інформаційної системи за допомогою теорії марківських процесів[43].

Нехай у момент часу t_0 на ІС здійснюються атаки і їх число рівне $\xi(t_0) = k$. Кожна *i*-а атака через момент часу *t* породить $\xi_i(t)$ атак на різні компоненти ІС шляхом експлуатації виявлених вразливостей.

Таким чином, загальна кількість атак на інформаційну систему буде рівна:

$$\xi(t + t_0) = \sum_{i=1}^{k} \xi_i(t) \qquad (2.12)$$

Нехай $\xi_1(t) = n_1, ..., \xi_k(t) = n_k$ з відповідними ймовірностями $p_{n1}(t), ..., p_{nk}(t)$. Іншими словами *i*-а атака, за час *t* породить n_i атак з ймовірністю $p_{ni}(t)$.

Таким чином, ймовірність переходу кількості атак $k \to n$ буде рівна:

$$p_{kn} = p(\xi(t+t_0) = n \mid \xi(t_0) = k) = \sum_{n_1+\ldots+n_k=n} p_{n1}(t) \cdot \ldots \cdot p_{nk}(t) \tag{2.13}$$

Крім того, атака може бути пригнічена засобами захисту ІС з ймовірністю $p_0(t)$.

Розглядатимемо процес розповсюдження атаки при $\xi(0) = 1$. Вважатимемо, що процес породження кінцевого числа атак $n \neq 1$ за малий час τ має ймовірність:

$$p_n(\tau) = a_n \tau + o(\tau) \tag{2.14}$$

де $a_n = \lambda_{1n}$ – інтенсивність переходу $1 \to n$.

Виробничі функції для даного процесу розповсюдження атаки виглядатимуть таким чином:

$$F(t,z) = \sum_n p_n(t) z^n, \tag{2.15}$$

$$F_k(t,z) = \sum_n p_{kn}(t) z^n = F(t,z)^k. \tag{2.16}$$

Для функції $F_1(t,z) = F(t,z)$ при будь-якому фіксованому z маємо:

$$\frac{dF(t,z)}{dt} = \sum_n p_n'(t) z^n \tag{2.17}$$

Відповідно до теореми Колмогорова:

$$p_n'(t) = \sum_k a_k p_{kn}(t) \tag{2.18}$$

Підставляючи (2.18) в (2.17) маємо:

$$\frac{dF(t,z)}{dt} = \sum_n a_k F(t,z)^k \qquad (2.19)$$

Позначимо $F(t,z)=x$, тоді:

$$f(x) = \sum_k a_k x^k \Rightarrow \frac{dx}{dt} = f(x) \qquad (2.20)$$

Для знаходження фінальної ймовірності можна скористатися зворотною функцією:

$$\frac{dt}{dx} = \frac{1}{f(x)} \Rightarrow t(x) = \int_z^x \frac{du}{f(u)}, \qquad (2.21)$$

за умови що $f(x) \neq 0$.

Вплив елементарної атаки на якийсь компонент автоматизованої системи, що володіє одним рівнем захисту можна проілюструвати з допомогою рисунку 2.8, де X – інтенсивність успішних, а μ – інтенсивність відбитих атак.

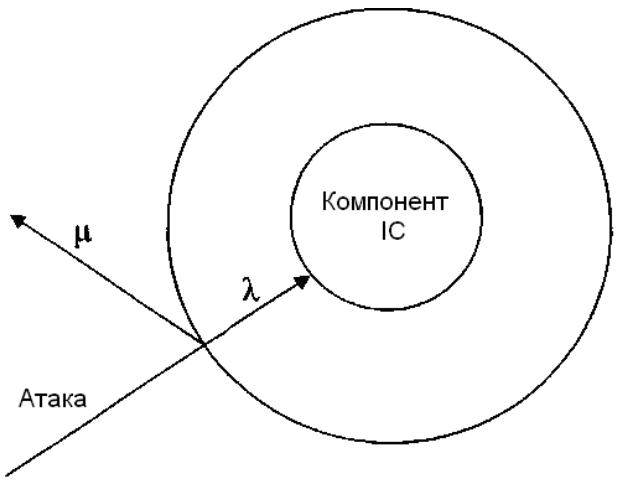

Рисунок 2.8 – Вплив атаки на компонент ІС

Дана модель еквівалентна графові марківського процесу, показаному на рисунку 2.9.

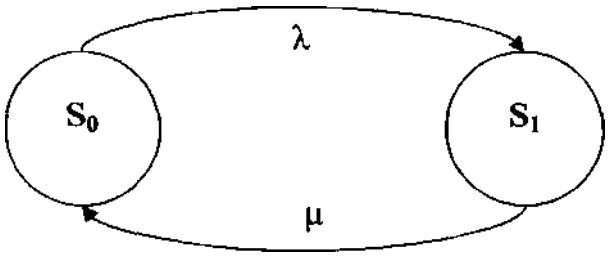

Рисунок 2.9 – Граф марківського процесу

Тут стан S_0 відповідає відсутності впливу атаки на компонент ІС (загасання атаки), стан S_i – нанесенню збитку (успішна атака). Процес функціонування компоненту ІС вважається марківським в силу незалежності інтенсивності переходів λ і μ від історії розвитку процесу. Крім цього, процес зміни станів даної моделі можна вважати виродженим процесом, де число можливих породжених частинок на кожному такті не перевищує 1. Визначимо функцію *f(x)* таким чином:

$$f(x) = \sum_{k=0}^{N-1} a_k x^k \qquad (2.22)$$

де N – число станів моделі (в даному випадку, N=2);

$a_k = \lambda_{1k}$ – відповідні інтенсивності переходів; k=0,1.

Виходячи з рисунку 2.7, $a_0 = \mu, a_1 = -\lambda$ відповідно функція *f(x)* прийматиме наступний вигляд:

$$f(x) = \mu - \lambda x. \qquad (2.23)$$

Задамо зворотну функцію *t(x)* таким чином:

$$\frac{dt(x)}{dx} = \frac{1}{f(x)} \qquad (2.24)$$

Враховуючи (2.23) і (2.24) маємо:

$$t(x) = \int_{z}^{x} \frac{du}{\mu - \lambda u} = -\frac{1}{\lambda}(\ln|-\lambda x + \mu| - \ln|\mu - \lambda z) \tag{2.25}$$

Функція подій графа (див. рисунок 2.7) має вигляд:

$$F(t,z) = \sum_{k=0}^{N-1} p_k(t) z^k \tag{2.26}$$

На основі розкладання функції (2.26) і зворотної функції (2.25) можна знайти залежності перехідної ймовірності станів S_0 і S_1 від часу:

$$p_0(t) = \frac{\mu}{\lambda}(1 - e^{-\lambda t}), \tag{2.27}$$

$$p_1(t) = e^{-\lambda t} \tag{2.28}$$

Перехідна ймовірність $p_1(t)$ тут є ймовірністю дії заданої атаки на компонент ІС.

Додавання додаткових рівнів захисту приведе до істотного ускладнення формули 2.23, що у свою чергу не дозволяє отримати аналітичний вираз інтеграла з формули 2.25. Не дивлячись на це, програмна реалізація моделі (розділ 3) легко дозволяє знайти його значення методом Сімпсона.

2.3 Імітаційне моделювання ІС

2.3.1 Опис процедури моделювання

Імітаційне моделювання здійснювалося з метою перевірки ступеня адекватності запропонованої в другому розділі математичної моделі

операційного середовища реальній обстановці. Процес моделювання зводився до дослідження реакції різних компонентів прототипу СВА на різні збурюючі чинники, що впливають на об'єкт, що захищається. Для моделювання мережевої взаємодії компонентів СВА в якості об'єкта досліджень розглядався сегмент мережі, структура якого показана на рисунку 2.10.

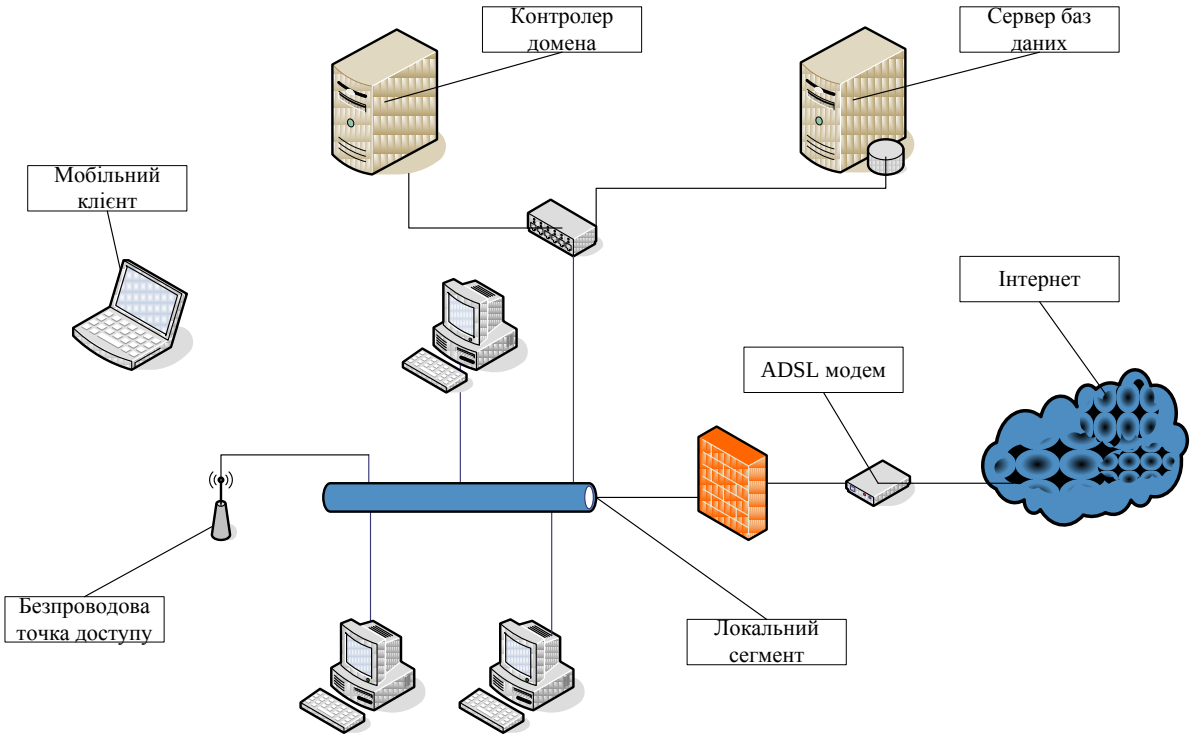

Рисунок 2.10 – Структура сегменту мережі

В ході першої стадії тестування здійснювався набір атак, а також можливих дій, що мають ознаки атаки, проте є легальними. Аналіз реакції системи здійснювався шляхом вимірювання величин дестабілізуючих чинників і ризиків в модулях ухвалення рішення локальних робочих станцій і мережевих сервісів.

Друга стадія тестування полягала в перевірці здатності системи до придушення атак. В ході її зіставлялися величини ризиків до і після реакції системи на виявлену атаку, обчислювалися величини помилок першого і другого роду.

Перед проведенням тестування, сенсори СВА працювали протягом тижня в режимі навчання, формуючи стандартні профілі користувачів, процесів і потоків. Навчання нейронної мережі блоку аналізу потоку команд здійснювалося на попередньому етапі тестування на основі спеціальним чином згенерованої

вибірки шкідливих кодів.

Імітаційне моделювання ІС проводилося за допомогою розробленого прототипу СВА, який детальніше описаний в розділі 3.

В якості дестабілізуючого чинника атаки, що впливає на ІС розглядалося виконання шкідливого коду і впровадження шпигунських програм на локальні хости ІС. Базова структура вимірювального комплексу показана на рисунку 2.11.

Рисунок 2.11 – Структура вимірювального комплексу

Вхідні дані (рівні загроз) поступають з сенсорів СВА і генеруються на основі наступної інформації:

– D1 – інформація про шкідливу активність користувачів, формується на основі обчислення ступеня відхилення поточних дій користувача від його шаблонної поведінки;

– D2 – ступінь шкідливості потоку інструкцій;

– D3 – ступінь підозрілості дій модуля.

Крім цього, в модель системи поступає середній коефіцієнт завантаження системи, що обчислюється на основі наступних даних:

1) Завантаження центрального процесора.

Завантаження центрального процесора локальної робочої станції оцінюється за допомогою двох параметрів: $P_r(t)$ – миттєве завантаження, що

описує поточну завантаженість процесора, і $IP_r(t)$ – інтегральне завантаження процесора за певний період часу $T = t_2 - t_1$:

$$IP_r(t) = \int_{t_1}^{t_2} P_r(t)dt. \qquad (2.29)$$

2) Використання фізичної пам'яті.

Під фізичною пам'яттю в даній роботі розуміється сумарна пам'ять, що утворюється оперативною пам'яттю і файлом підкачки. Для оцінки даного компоненту використовуються наступні параметри:

1. Коефіцієнт зайнятості фізичної пам'яті:

$$M(t) = \frac{M_u(t)}{M_\sum(t)}, \qquad (2.30)$$

де $M_u(t)$ – поточний об'єм фізичної пам'яті, що використовується;

$M_\sum(t)$ – сумарний об'єм фізичної пам'яті.

2. Інтенсивність використання оперативної пам'яті:

$$M_i(T) = \frac{M_r(T) + M_w(T)}{T}, \qquad (2.31)$$

де $M_r(T)$ – число операцій читання за час *T*;

$M_w(T)$ – число операцій запису за час *T*.

3. Швидкість використання оперативної пам'яті:

$$V_m(t) = \frac{dM(t)}{dt} \qquad (2.32)$$

3) Використання жорсткого диска

Оцінка ступеня використання жорсткого диска здійснюється аналогічно оцінці фізичної пам'яті:

1. Коефіцієнт зайнятості жорсткого диска:

$$H(t) = \frac{\sum H_{ui}(t)}{\sum H_{Ti}(t)}, \tag{2.33}$$

де $H_{ui}(t)$ – величина простору, що використовується на *i*-му розділі жорсткого диску;

$H_{Ti}(t)$ – загальний розмір *i*-го розділу жорсткого диску;

2. Інтенсивність використання жорсткого диску:

$$HI(T) = \frac{H_r(T) + H_w(T)}{T}, \tag{2.34}$$

де $H_r(T)$ – число операцій читання жорсткого диску за час *T*;

$H_w(T)$ – число операцій запису на жорсткий диск за час *T*.

Нечітка когнітивна карта, що використовується для оцінки ризиків функціонування ІС, показана на рисунку 2.12, концепти моделі описана в таблиці 2.1.

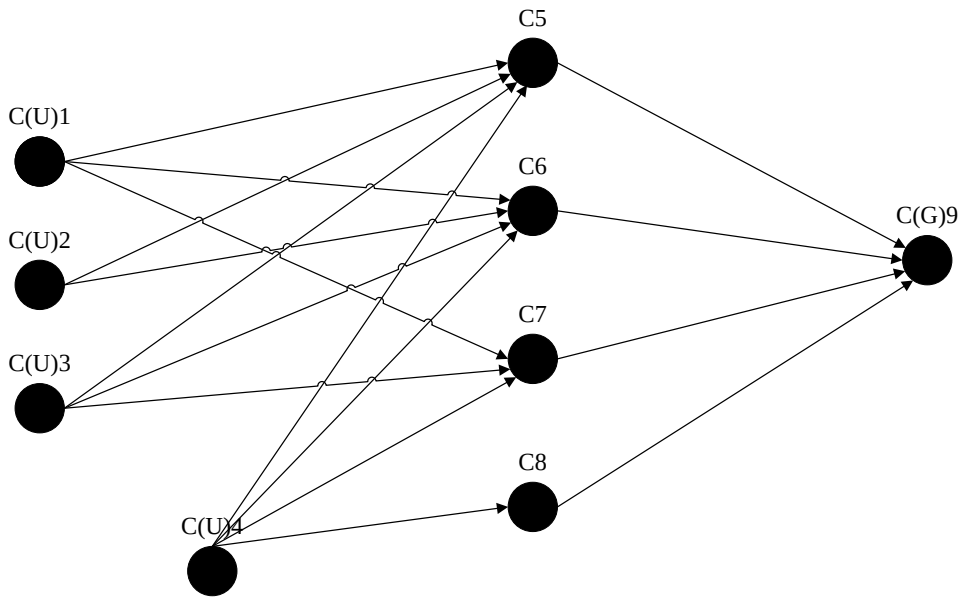

Рисунок 2.12 – Нечітка когнітивна карта ІС

Внаслідок того, що нечітка когнітивна карга є динамічною моделлю з дискретним часом, вимірювання величин рівнів загроз проводилися з частотою 20 Кгц (крок дискретизації 20 мс), вибраної з погляду оптимізації навантаження на модельовану систему. Кожен процес операційної системи оцінювався за власною тимчасовою шкалою використання процесора, часу простою і зміни контексту не враховувалися. Для спрощення процедури вимірювань, використовувалася однопроцесорна система з відключеною технологією HyperThreading.

Таблиця 2.1 – Концепти моделі

Концепт	Тип	Опис
C(U) 1	ДФ	Рівень шкідливої активності користувачів
C(U) 2	ДФ	Ступінь шкідливості потоку інструкцій
C(U) 3	ДФ	Ступінь підозрілості дій модулів
C(U) 4	ДФ	Ступінь завантаження системи
C5	ПФ	Ядро операційної системи
C6	ПФ	Прикладне програмне середовище
C7	ПФ	Програмне забезпечення користувачів
C8	ПФ	Дані користувачів
C(G) 9	ЦФ	Величина ризику функціонування

2.3.2 Оцінка ризиків функціонування ІС

Для оцінки поведінки моделі був сформований набір експериментів, кожен з яких полягав в запуску певного набору програм, причому частина з них була шкідливою, а частина була різними утилітами сторонніх розробників. Атака вважалася зареєстрованою моделлю, якщо рівень ризику перевищував значення 0,6. Список проведених експериментів приведений в таблиці 2.2.

Впровадження rootkil ядра SSDT.

Даний експеримент перевіряв здатність моделі розпізнавати впровадження шпигунського модуля, що реалізований у вигляді драйвера ядра і використовує модифікацію таблиці дескрипторів системних сервісів для приховування своєї присутності [44]. Для проведення експерименту використовувався спеціально

розроблений для цієї мети шпигунський модуль. Графіки залежностей рівнів загроз і ризику від часу показані на рисунку 3.4.

Таблиця 2.2 – Список проведених експериментів

Назва	Є атакою	Опис
E1	Так	Впровадження rootkit ядра SSDT
E2	Ні	Використання утиліти regmon
E3	Так	Впровадження rootkit ядра КОМ
E4	Так	Використання dll-injection
E5	Так	Модифікація IAT
E6	Так	Використання keylogger ядра

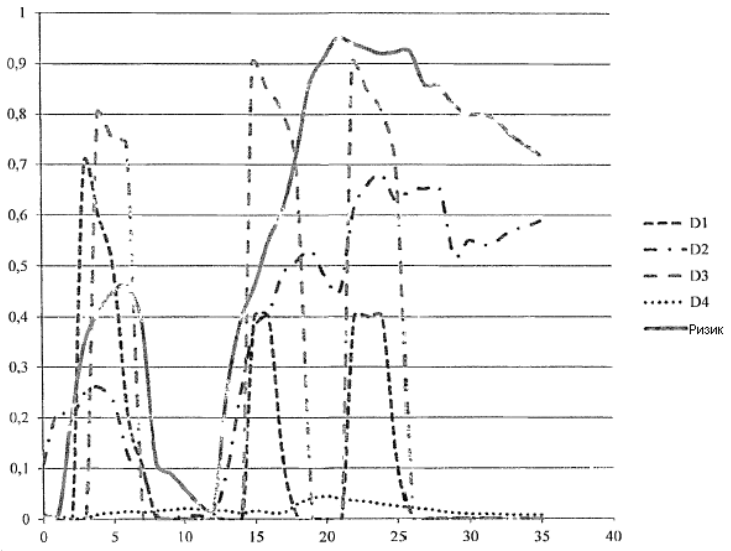

Рисунок 2.13 – Графіки ризику і рівнів загроз E-1

Першим кроком експерименту було завантаження програмою-носієм шпигунського модуля (такти 2-7). На рисунку 2.13 можна побачити різке збільшення рівня загрози D1, оскільки ручне завантаження драйверів ядра не є регулярною дією користувача і не відповідає його профілю поведінки. Модель також реєструє наявність підозрілої активності модуля D3, пов'язаної з використанням функцій SCM програмою-носієм. Високий рівень загрози, що реєструється сенсорами D1 і D3, приводить до підвищення величини ризику функціонування системи, проте він не перевищує величини порогового значення

реєстрації атаки, оскільки подібна ситуація, взагалі кажучи, є штатною.

Другий крок експерименту полягав в зміні значень декількох векторів таблиці SSDT по команді програми-носія (такти 14-18). Спостерігається збільшення рівня загрози D1, пов'язане з використанням функцій IOCTL програмою-носієм (пряма комунікація з драйверами ядра також не входить в поточний шаблон поведінки). Також спостерігається різке збільшення рівня загрози D3, викликане тим, що сенсор виявив в таблиці дескрипторів покажчики на модулі, що належать адресному простору шпигунського драйвера, що є вкрай нетиповою поведінкою для системних модулів. Взагалі кажучи, між командою програми-носія і її виконанням шпигунським драйвером існує певний проміжок часу, проте він свідомо менший кванта тактування, що використовується, тому на графіку ці події відбуваються одночасно. Можна спостерігати також підвищення рівня загрози D2, що отриманий на основі аналізу потоку інструкцій завантажених модулів. Ці події приводять до збільшення рівня ризику і реєстрації атаки на 17-му такті.

Третій крок експерименту полягав в зміні значення, що повертається системної функції ZwQuerySystemlnformation для того, щоб приховати процес програми-носія (такти 20-27). Знову спостерігається збільшення рівня загрози D1, пов'язане з комунікацією програми-носія з драйвером. Підвищення рівня загрози D3 пов'язане з тим, що сенсор виявив спотворення результатів виклику системної функції. Рівень ризику на даному кроці експерименту перевищив 0,9 і на 35-му такті атака була заблокована шляхом відновлення таблиці дескрипторів, вивантаження шпигунського драйвера і завершення процесу програми - носія.

Використання утиліти regmon.

Даний експеримент дуже схожий на попередній, за винятком, що програма, яка використовувалася є широко поширеним інструментом моніторингу системного реєстру. Отримані графіки залежностей рівнів загроз і ризику від часу показані на рисунку 2.14.

Графіки рівнів загроз D1 і D3 на тимчасовому проміжку 0...20 тактів практично співпадають з графіками попереднього експерименту. Це пояснюється тим, що утиліта regmon для моніторингу функцій реєстру використовує той же

самий метод модифікації таблиці дескрипторів системних сервісів, що і шпигунський драйвер, що використовувався в попередньому експерименті. Відсутність різкого збільшення рівня загрози D3 на інтервалі 20...27 тактів пояснюється тим, що утиліта не змінює результати виклику системних функцій, використовуючи їх тільки для створення звіту використання системних функцій роботи з реєстром.

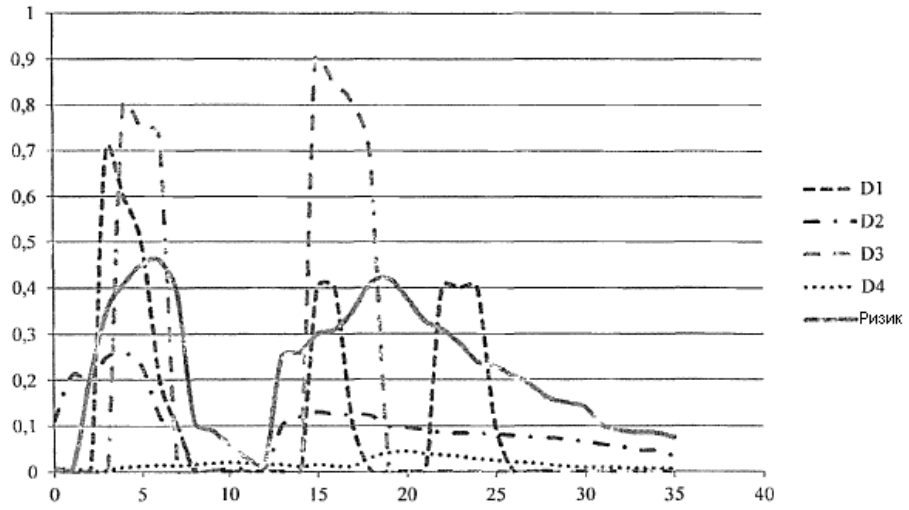

Рисунок 2.14 – Графіки ризику і рівнів загроз Е-2

Не дивлячись на наявність різких піків рівнів загроз, рівень ризику не перевищив порогового значення, сигналізуючи штатний режим роботи системи.

Впровадження rootkit ядра КОМ.

Даний експеримент перевіряв здатність моделі розпізнавати впровадження шпигунського модуля, що реалізований у вигляді драйвера ядра і використовує модифікацію системної структури EPROCESS для приховування своєї присутності. Для проведення експерименту використовувався спеціально розроблений для цієї мети шпигунський модуль. Графіки залежностей рівнів загроз і ризику від часу показані на рисунку 2.15.

Перший крок експерименту полягає в завантаженні шпигунського драйвера і його результати аналогічні першому експерименту (Е-1).

Другий крок експерименту полягає в зміні системної структури EPROCESS для приховування присутності в системі процесу-носія (такти 12-35). Спостерігається різке збільшення рівня загрози D3, пов'язане з виявленням сенсором потоків, що не належать ніякому процесу. Дана ситуація є некоректною,

тому сенсор не знижує рівень загрози до кінця експерименту. Аналіз потоку інструкцій шпигунського драйвера приводить до зростання рівня загрози D2. Ці процеси приводять до монотонного зростання ризику і реєстрації атаки на 16-му такті.

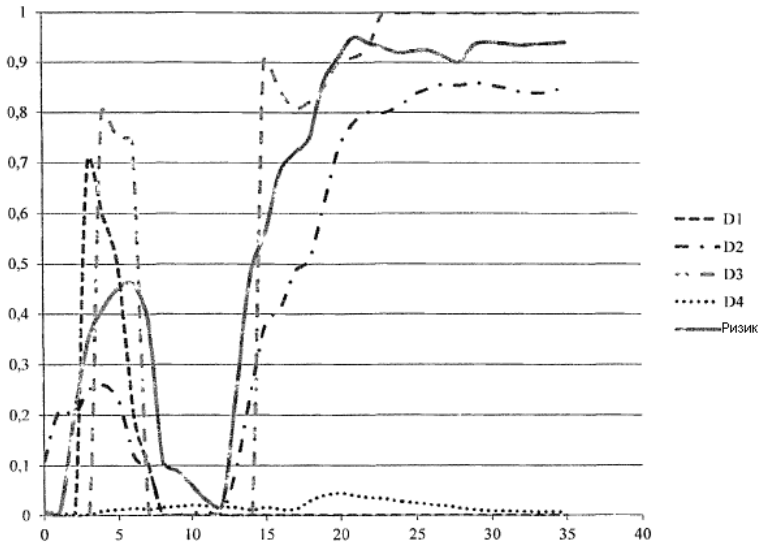

Рисунок 2.15 – Графіки риизку і рівнів загроз Е-3

Використання dll injection.

Даний експеримент перевіряв здатність моделі розпізнавати створення шпигунського потоку в легальному призначеному для користувача процесі шляхом впровадження динамічної бібліотеки (dll injection). Для проведення експерименту використовувався розроблений для цієї мети шпигунський модуль. Графіки залежностей рівнів загроз і ризику від часу показані на рисунку 2.16.

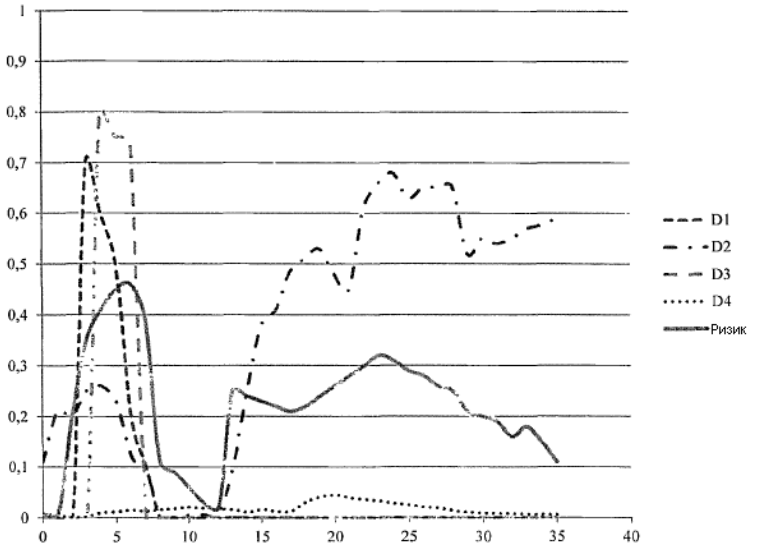

Рисунок 2.16 – Графіки ризику і рівнів загроз Е-4

Перший крок експерименту полягав у віддаленому створенні потоку, в якості адреси якого передавався покажчик на функцію LoadLibrary і подальшому створенні шпигунського потоку в методі DllMain динамічної бібліотеки (такти 1-6).

Спостерігається збільшення рівня загрози D1, пов'язане із запуском програми, що не входить в шаблон поведінки користувача і D3, пов'язане з тим, що сенсор розпізнав спробу впровадження динамічної бібліотеки в адресний простір чужого процесу.

Подальше виконання програми веде до збільшення рівня загроз D2, що обчислюється аналізатором ступеня шкідливості потоку інструкцій, проте рівень ризику не перевищує порогового значення і атака не реєструється. Однією з причин даної поведінки моделі є наявність великого числа невизначеностей в експертних знаннях, що описують модель. Вирішення даної проблеми буде описано в подальших розділах дипломної роботи.

Модифікація IAT.

Даний експеримент перевіряв здатність моделі розпізнавати спроби модифікації системної інформації шляхом заміни покажчиків в адресній таблиці імпорту (IAT) виконуваних модулів. Для проведення експерименту використовувався розроблений для цієї мети шпигунський модуль. Графіки залежностей рівня загроз і ризику від часу показані на рисунку 2.17.

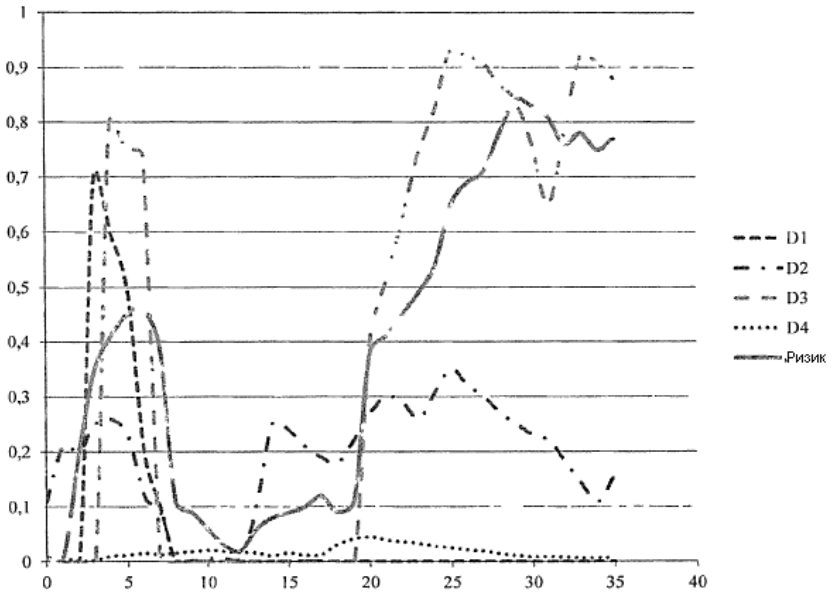

Рисунок 2.17 – Графіки ризиків і рівня загроз E-5

Перший крок експерименту полягає у впровадженні шпигунської динамічної бібліотеки в адресний простір призначеного для користувача процесу і співпадає з попереднім експериментом (Е-4).

На другому кроці експерименту (10–16 такти) шпигунська бібліотека здійснює заміну покажчиків в адресній таблиці імпорту, і, оскільки ця операція полягає в маніпуляції покажчиками у власному адресному просторі процесу, даний процес не реєструється сенсорами.

На третьому кроці експерименту сенсор виявляє зміну результату виклику системної функції, що приводить до зростання величини рівня загрози D3. Не дивлячись на те, що даний експеримент використовує метод атаки, схожий з попереднім експериментом, додаткова інформації сенсора виявляється достатньо для коректної оцінки ризику і реєстрації атаки на 22-му такті.

Використання keylogger ядра.

Даний експеримент перевіряв здатність моделі розпізнавати використання спеціалізованих перехоплювачів системних паролів рівня ядра. Для проведення експерименту використовувався спеціально розроблений для цієї мети шпигунський модуль. Графіки залежностей рівнів загроз і ризику від часу показані на рисунку 2.18.

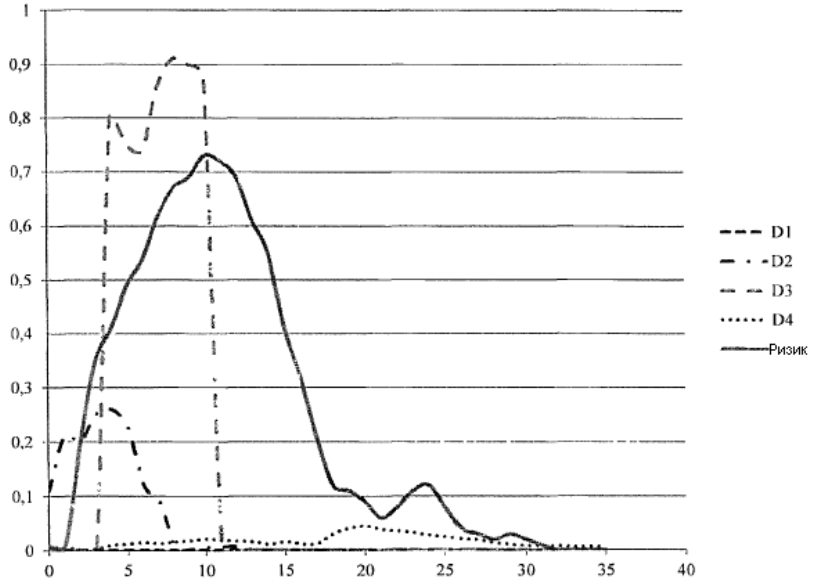

Рисунок 2.18 – Графіки ризику і рівнів загроз Е-6

На першому кроці експерименту (4-11 такти) шпигунський модуль реєструє себе як проміжний драйвер клавіатури, отримуючи таким чином доступ до всієї інформації, що вводиться з клавіатури. Спостерігається різке збільшення рівня загрози D3, пов'язане з тим, що сенсор розпізнав реєстрацію проміжного драйвера пристроїв введення інформації. Всі решта рівні загроз практично рівні нулю, оскільки шпигунський модуль не здійснює ніяких протиправних дій.

Модель реєструє атаку на 9-му такті, проте рівень ризику повертається в допустиму межу на 13-му такті, оскільки шпигунський модуль не здійснює ніяких активних дій в системі. Даний результат є наслідком недосконалості модуля сенсора прототипу СВА і не впливає на якість функціонування моделі.

2.3.3 Оцінка величин помилок

Для оцінки коректності розпізнавання різних атак моделлю ІС проведемо оцінку величини помилок першого і другого роду, використовуючи експерименти попереднього параграфу в якості базових еталонів при формуванні тестуючих вибірок. Моделювання атак здійснювалося на віртуальній машині для забезпечення однакового середовища при кожному експерименті, використовувалися шпигунські модулі, що знаходяться у відкритому доступі і володіють початковими кодами. Тестування кожного шпигунського модуля проводилося протягом трьох разів, результати усереднювалися. Отримані величини помилок першого і другого роду представлені в таблиці 3.3.

Таблиця 2.3 – Величин помилок I і II роду

Тип атаки	Кількість модулів	Помилка I роду	Помилка II роду	Опис
E1	25	0,04	0	SSDT rootkit
E2	16	0	0,188	Системна утиліта
E3	10	0	0	KOM rootkit
E4	30	0,267	0	Dll injection
E5	26	0,038	0	Модифікація IAT
E6	8	0	0	Keylogger

Аналіз отриманих результатів показує, що отримані значення можуть бути пояснені двома причинами: завищеним значенням порогу реєстрації атак і високим рівнем невизначеності експертних знань моделі. Перша проблема вирішується шляхом впровадження в систему прйняття рішень гнучкої системи сигнатурного аналізу, що дозволяє позбавитися від жорсткого порогу активації моделі. Друга проблема вирішується в даній роботі за рахунок додаткового навчання вагів зв'язків нечіткої когнітивної карти на сформованій навчальній вибірці.

2.3.4 Навчання моделі ІС

Описана нечітка когнітивна карта може бути представлена у вигляді нейронної мережі, де концепти карти виконують функції суматорів, а нормуюча функція є функцією активації. Якщо скористатися даним представленням НКК, отрмаємо можливість позбавитися від надмірної невизначеності знань експертів, закладених в неї, шляхом навчання вагів зв'язків НКК на заданій навчальній вибірці з використанням будь-якого алгоритму з вчителем. Суть роботи даного алгоритму зводиться до зменшення величини розузгодження між результатом моделювання і значенням еталону. Для зменшення величини помилок першого і другого роду в даній роботі використовувався алгоритм зворотного розповсюдження помилки (Back Propagation). Формування навчальної вибірки проводилося на основі набору різних шпигунських модулів [44], а також – різних системних утиліт, що активно використовують роботу з системним реєстром, службами операційної системи і що дозволяють модернізувати структури ядра операційної системи (rcgmon, filemon, gflags і т. д.). Після навчання НКК було проведене повторне імітаційне моделювання і обчислені значення помилок першого і другого роду.

Результати навчання представлені в таблиці 2.4.

Дані результати демонструють ефективність запропонованих алгоритмів навчання НКК для усунення невизначеності бази експертних знань, що лежать в основі побудови НКК моделі ІС. Використання даних методів дозволяє істотно підвищити точність розпізнавання атак компонентами СВА, додатково знижуючи

величину ризику функціонування ІС. Істотним недоліком даної методики є неможливість її використання для «калібрування» бази нечітких правил, що лежать в основі методики обчислення ступеня взаємовпливу концептів.

Таблиця 2.4 – Величин помилок I і II роду після навчання СВА

Тип атаки	Кількість модулів	Помилка I роду	Помилка II роду	Опис
E1	25	0	0	SSDT rootkit
E2	16	0	0,063	Системна утиліта
E3	10	0	0	KOM rootkit
E4	30	0,067	0	Dll injection
E5	26	0,038	0	Модифікація IAT
E6	8	0	0	Keylogger

В даному розділі запропонована методика побудови динамічної моделі ІС на основі нечітких когнітивних карт, яка може бути використана надалі як інтелектуальний модуль прийняття рішень в СВА. Запропонована методика включає наступні базові кроки:

1) аналіз суб'єктів і об'єктів ІС, виявлення їх суті і взаємозв'язків, побудова онтологічних, функціональних і динамічних моделей. Для вирішення цієї задачі була використана методологія IDEF, що дозволяє здійснити формалізацію предметної області у вигляді набору стандартизованих, зручних для сприйняття моделей;

2) побудова формалізованої моделі атак на основі моніторингу поточної ситуації, статистичних даних або аналітичних залежностей. Дана задача була вирішене шляхом використання математичного апарату Марківських процесів;

3) формалізація експертних знань у вигляді нечіткої бази правил і ваг концептів і побудова на їх основі нечіткої когнітивної карти.

Проведено імітаційне моделювання з використанням динамічної моделі ІС. Оскільки моделювання проводилося з використанням розробленого прототипу СВА, в якості вхідних даних використовувалася інформація з сенсорів, а не

статистична модель атак. Результати моделювання показують, що:

1) динамічна модель ІС на основі нечітких когнітивних карт може бути успішно використана для вирішення задачі виявлення атак на інформаційну систему;

2) використання експертних знань в нечіткій формі приводить до збільшення ймовірності виникнення помилок I і II роду, причому ускладнення системи може привести до експоненціального зростання даної ймовірності.

Для зменшення ймовірності виникнення помилок I і II роду був запропонований метод коректування ваг концептів і зв'язків, заснований на алгоритмі Back Propagation, суть якого зводиться до навчання нечіткої когнітивної карти на наборі базових еталонів. Використання даного підходу дозволило істотно зменшити кількість помилок розпізнавання атак на інформаційну систему.

3 РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ВИЯВЛЕННЯ АТАК

3.1 Архітектура системи

В основі запропонованої архітектури лежить наділення кожного агента СВА базовою функціональністю системи в цілому, тобто здатністю до аналізу і прийняття рішень в рамках жорсткого поставленого завдання (аналіз мережевого трафіку хоста, аналіз активності системних процесів і т. д.). Таким чином, кожен сенсор СВА є системою виявлення атак в мініатюрі, яка здатна до автономного функціонування при втраті зв'язку з модулями прийняття рішень вищих рівнів.

Сенсори СВА утворюють нижній рівень мережі збору і аналізу даних. Після обробки сенсором, інформація поступає в модулі аналізу і прийняття рішень вищих рівнів через універсальний інтерфейс «Аналізатор - Вирішувач». Даний підхід дозволяє будувати ієрархічне дерево з модулів аналізу і сенсорів, причому для кожного вузла дерева піддерево, що лежить нижче, представлятиметься як єдиний сенсор. Структурна схема СВА, що реалізовує описану архітектуру, показана на рисуноку3.1.

Модулі, що входять в СВА, можна розбити по функціональному призначенню на два основні класи:

1) базові модулі системи;
2) модулі системи виявлення атак (СВА).

Базові модулі забезпечують інтеграцію і взаємодію компонент системи один з одним, дозволяють змінювати конфігурацію СВА і взаємодіяти з ПЗ блокування вторгнень. Модулі СВА дозволяють відстежувати і блокувати атаки в реальному часі. В основі функціонування даної підсистеми лежить метод аналізу сигнатур активності.

Центральним компонентом системи, шо об'єднує решту всіх модулів в єдине ціле є глобальна база знань (ГБЗ), що містить інформацію про всі відомі системі види атак, методи протидії вторгненням, архів подій системи, а також конфігурацію всіх компонент СВА. Для забезпечення необхідного рівня надійності і відмовостійкої, в якості ГБЗ можливе використання сучасної клієнт-

серверної реляційної СУБД, що володіє можливостями кластеризації і прозорої реплікації даних.

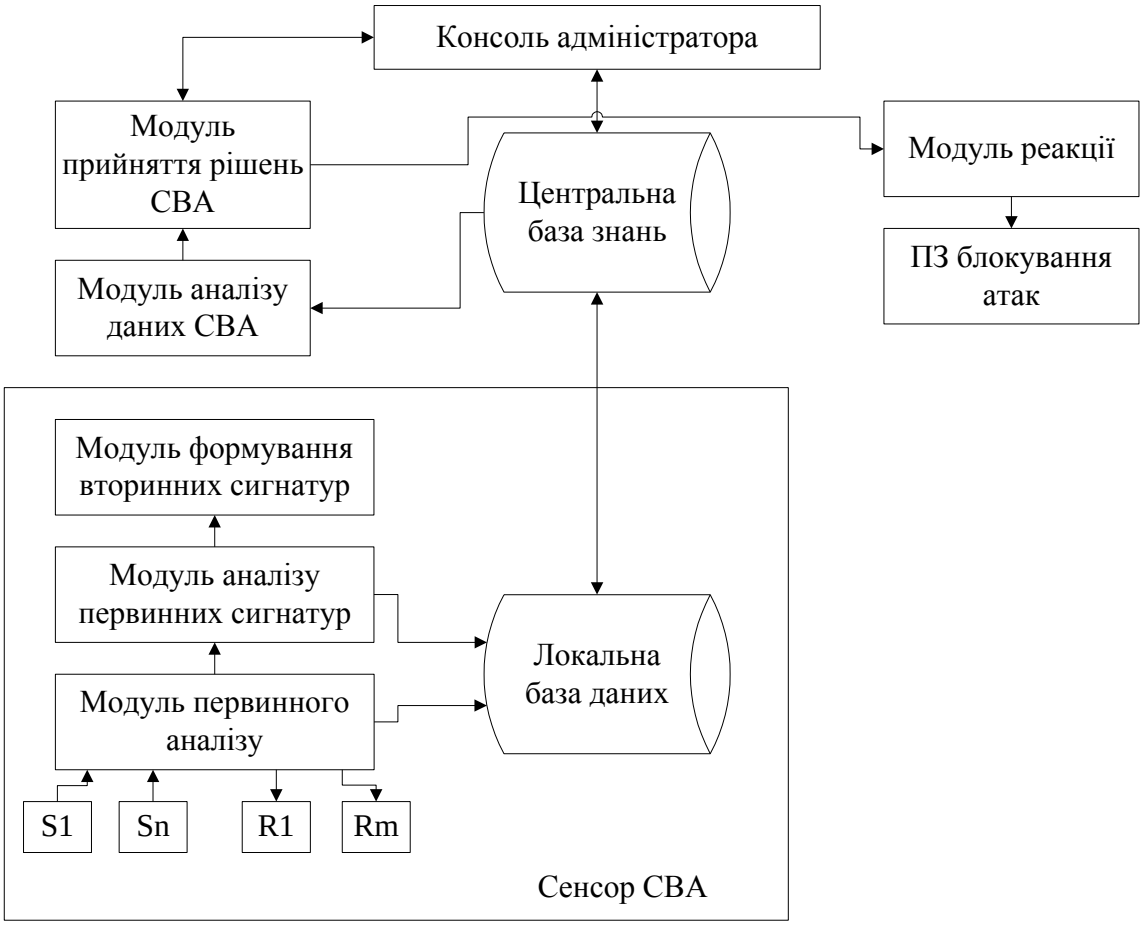

Рисунок 3.1 – Структурна схема СВА

Основним інструментом налаштування і управління СВА є консоль адміністратора, що є комплексом програмних засобів, що забезпечує настройку модулів системи, виведення інформації про виявлені атаки і надає можливість класифікації адміністратором активності, яка не розпізнана модулями СВА. Підключення консолі адміністратора до ГБЗ здійснюється через уніфікований інтерфейс ISysManage, що реалізовується кожним компонентом системи. Даний підхід дозволяє здійснювати налаштування окремих компонент системи на нижчому рівні, що дозволяє розділяти задачі адміністрування СВА між декількома користувачами АС.

Модулем реакції є проміжний інтерфейс між СВА і засобами блокування вторгнень. Даний компонент забезпечує інтеграцію СВА в загальну систему захисту АС і дозволяє їй гнучко змінювати політику безпеки залежно від

зовнішніх умов.

Кожний сенсор представляє собою набір блоків збору даних (Sl...Sn), блоків первинної реакції (Rl...Rn), а також містить блок первинного аналізу даних. Даний модуль здійснює первинний аналіз потоку даних активності об'єктів КІС і формує на його основі сукупність первинних сигнатур. Сукупність первинних сигнатур містить опис активності, що викликала підозру сенсора і повний перелік операцій, виконаних суб'єктом активності (послідовність виклику системних функцій, пакети TCP/IP і т. д.).

Отримані сигнатури обробляються блоком аналізу первинних сигнатур на основі даних локальної бази прецедентів. Класифікація сигнатур проводиться нейро-нечітким когнітивним аналізатором.

У випадку, якщо певна активність класифікується сенсором як атака, здійснюється формування вторинної сигнатури, що є формалізованим описом атаки. Дана сигнатура поступає на вхід модуля аналізу даних, який формує цілісну картину розповсюдження атаки в інформаційній системі. Архитекіура СВА дозволяє організовувати модулі аналізу у вигляді ієрархічного дерева, формуючи тим самим декілька каскадів обробки інформації, що дозволяє гнучко масштабувати СВА залежно від розмірів і структурної організації ІС. Крім формування вторинної сигнатури, сенсор здійснює пошук в локальній базі прецедентів інформації про методи блокування виявленої атаки.

У випадку, якщо дана інформація присутня, блок первинного аналізу здійснює активацію відповідного блоку первинної реакції і, по зміні даних з блоків збору, визначає ступінь ефективності придушення атаки. У випадку, якщо методи локального придушення виявляються неефективними або локальна база не містить інформації про методи блокування даної атаки, ініціюється запит глобального модуля прийняття рішень, який може виконати додаткову настройку систем захисту підприємства, блокувати джерело атаки або видати запит в консоль адміністратора.

3.1.1 Опис архітектури сенсора

Інтелектуальним сенсором є програмний комплекс, що містить сукупність модулів збору інформації, модулів реакції на атаку, модуль прийняття рішень і інтерфейс зв'язку з підсистемами верхнього рівня [45]. Структурна схема інтелектуального сенсора показана на рисунку 3.2.

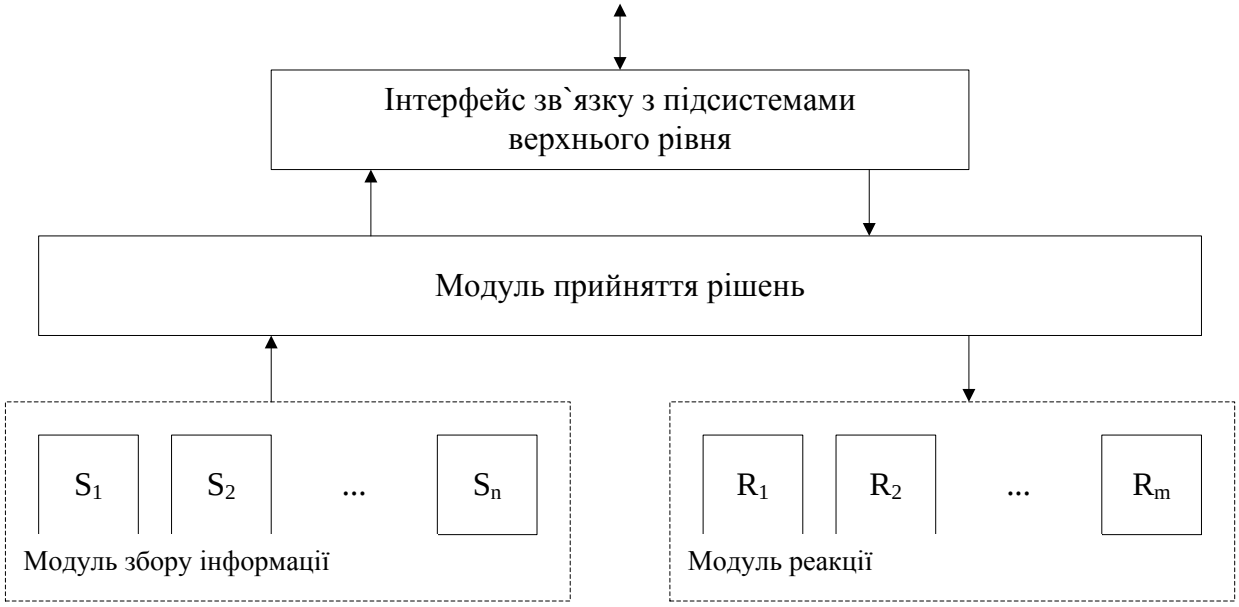

Рисунок 3.2 – Структурна схема інтелектуального сенсора СВА

В якості початкових даних для аналізу активності шкідливого ПЗ сенсор використовує наступну інформацію:

1) статистику викликів системних сервісів операційної системи (ОС);

2) наявність спроб впровадження в адресний простір користувацьких або системних процесів;

3) спроби отримання привілеїв відладки або підключення до налагоджувальних портів процесів;

4) спроби модифікації системних структур ядра ОС;

5) статистичні профілі користувачів і процесів.

Інформація за статистикою викликів системних сервісів збирається окремо для кожного процесу в системі. Графічно її можна представити у вигляді пелюсткової діаграми, що містить 289 осей (число системних сервісів для Windows 2003 SPl). Кожна вісь відповідає певному системному виклику ядра ОС. На осі відкладається відношення кількості викликів даного системного сервісу до

загального процесорного часу, спожитого процесом.

3.1.2 Опис реалізації інтелектуального сенсора

Інтелектуальний нейромережевий сенсор СВА реалізований у вигляді програмного комплексу, структура якого показана на рисунку 3.3.

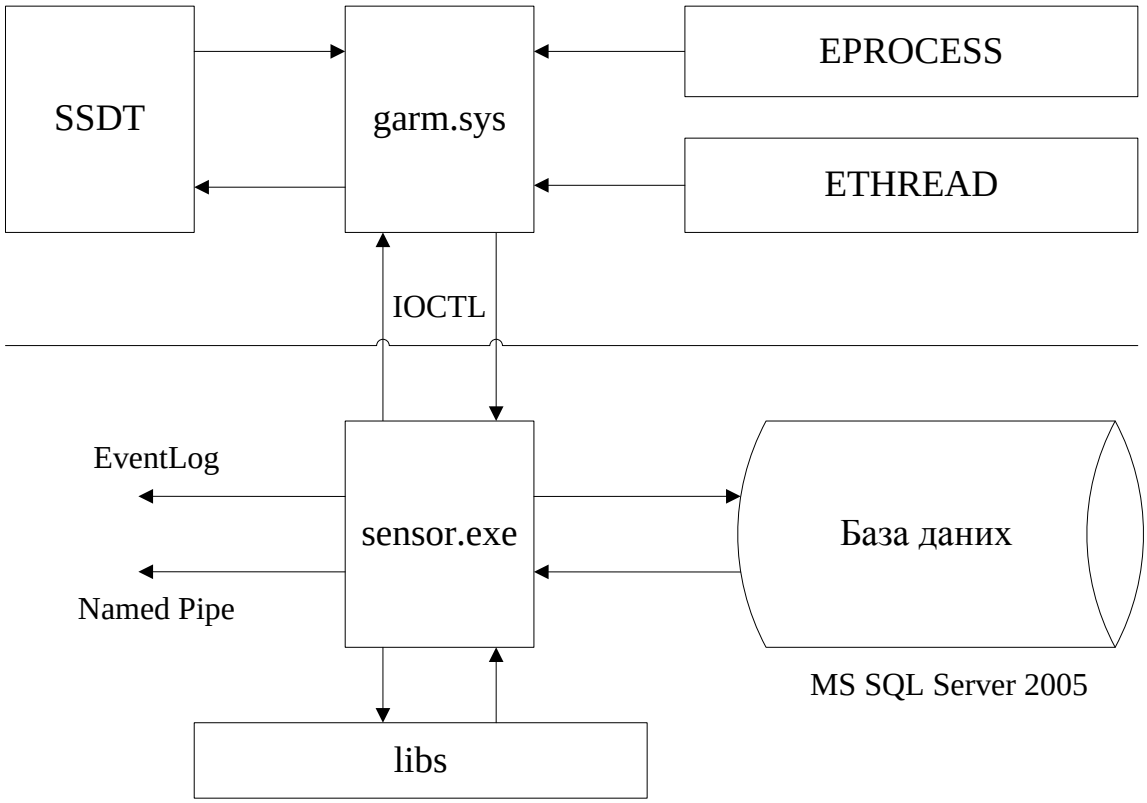

Рисунок 3.3 – Реалізація сенсора СВА

Функції модулів збору даних і модулів реакції на атаки виконує модуль garm.sys, реалізований у вигляді драйвера віртуального пристрою.

Модуль прийняття рішень реалізований у вигляді стандартної служби Windows 2000, що забезпечує можливість його резидентного виконання, незалежно від роботи користувачів комп'ютера. Обмін даними з модулями збору інформації здійснюється за допомогою IOCTL-запитів, направлених віртуальному пристрою, зареєстрованому драйвером. Інтерфейс взаємодії з підсистемами верхнього рівня реалізується окремим потоком, зв'язок здійснюється за допомогою механізму мережевого редиректора Windows (Named Pipes). Сервіс реалізує стандартний механізм запису подій і аварій в окремий системний журнал (EventLog) операційної системи. Блок нечіткого логічного виводу реалізований у

вигляді окремої бібліотеки (DLL), що динамічно підключається, реалізовує стандартний інтерфейс функцій, що експортуються. Подібний підхід дозволяє уніфікувати процедуру розробки, конфігурації і супроводу системи. Підключення бібліотеки здійснюється шляхом додавання запису в таблицю sensor modules локальної бази даних сенсора. Кожен підключений до бази сенсор може створювати свій набір таблиць необхідний йому для зберігання сигнатур шкідливої активності. Для забезпечення можливості прозорої реплікації даних між однотипними сенсорами АС, кожному типу сенсорів привласнюється глобальний унікальний ідентифікатор (GUID). Обмін і акумуляція даних здійснюється через глобальну базу знань.

3.2 Розробка компонентів системи виявлення атак

3.2.1 Реалізація драйвера захисту

Драйвер захисту сенсора СВА garm.sys здійснює автономний збір інформації про статистику викликів системних сервісів з метою виявлення шкідливої активності, а також запобігає спробам приховування шкідливих модулів шляхом модифікації системного списку структур EPROCESS. В основі захисту лежить технологія перехоплення системних викликів в ядрі через модифікацію таблиці дескрипторів системних сервісів (System Services Descriptor Table SSDT).

Модифікація SSDT є практично універсальним способом перехоплення будь-якої функції, бібліотекою ntdll.dll (Native API). Розглянемо приклад, що демонструє можливість подібного перехоплення.

Отримання інформації про запущені процеси і завантажені модулі здійснюється через Too1He1p32 API, що реалізовується бібліотекою psapi31.dll (починаючи з Windows 2000 – kernel32.dll). Всі функції цієї бібліотеки викликають функцію ZwQuerySystemInformation з модуля ntdll.dll, яка записує в регістр eax число 0x97h (Windows 2000 SP4), поміщає в edx покажчик на аргументи в стеку і викликає переривання 0x2eh. При виникненні переривання, операційна система перемикається в режим ядра і викликає диспетчер системних

сервісів KiSystemService. Він копіює аргументи із стека процедури, що викликала, в свій стек, а потім шукає в таблиці диспетчеризації системних сервісів (SSDT) покажчик на обробник, використовуючи значення регістра eax як індекс. При виявленні обробника, KiSystemService передає управління на нього. Покажчик на SSDT експортується ядром операційної системи як KeServiceDescriptorTable, що дозволяє будь-якому модулю ядра отримати доступ до неї.

Таким чином, змінюючи покажчик в SSDT, будь-який модуль ядра може перехоплювати виклики операційної системи, переривання 2eh, що здійснюються через шлюз (інструкцію SYSENTER, починаючи з Windows XP). Схема обробки викликів системних сервісів показана на рисунку 3.4.

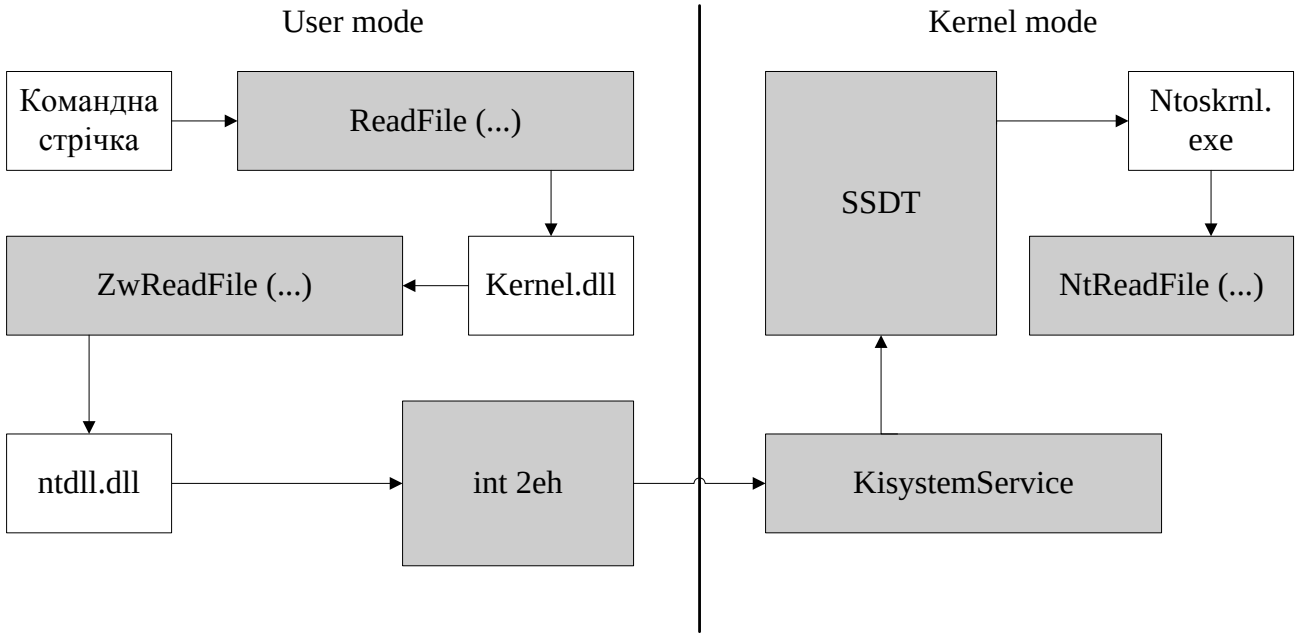

Рисунок 3.4 – Схема обробки викликів системних сервісів

Основною проблемою при організації підміни покажчиків в таблиці SSDT є визначення індексу процедури, що підміняється. Дані числа автоматично генеруються при кожній новій розробці Windows і в різних версіях операційної системи вони відрізняються. У даній роботі пропонується алгоритм визначення індексів методів без прив'язки до конкретної версії операційної системи.

Дізасемблерування бібліотеки ntdll.dll показує, що кожна системна функція Native API має приблизно наступний код:

```
77f8c552 b8a1000000      mov      eax.0xa1
77f8c557 8d542404                 lea      edx[esp+0x4]
```

```
77f8c55b cd2e                   int     2e
77f8c55d c22400                 ret     0x24
```

Можна відмітити, що перші п'ять байт кожного методу містять інструкцію move ax, індекс, причому перший байт b8 – це сама інструкція move eax, а чотири, що залишилися – індекс, який записується в даний регістр. Проста реалізація методу, запропонована Марком Руссиновичем [46, 47], здійснює читання чотирьох байт по зсуву один байт від точки входу в процедуру. На жаль, даний метод неможливо безпосередньо використовувати у разі перехоплення функцій, що не експортуються ядром операційної системи, таких, наприклад, як ZwQuerySystemTime, ZwTerminateThread, ZwListenPort і багато інших.

Запропонований метод припускає аналіз заголовка, списків експорту і розділу коду бібліотеки ntdll.dll, точки входу, що містить, для всіх функцій Native API. Здійснюється динамічне завантаження даної бібліотеки в пам'ять модуля ядра і пошук коду необхідного методу. Технологія визначення індексу процедури в таблиці SSDT аналогічна запропонованою Марком Руссиновичем.

Для отримання доступу до адресного простору процесу, додатки в користувацькому режимі використовують Win32 функції ReadProcessMemory і WriteProcessMemory. Не дивлячись на те, що для доступу до адресного простору «чужого» процесу потрібна наявність привілею на відладку, як правило, це є тільки у адміністраторів системи, зловмисник може отримати її, використовуючи вразливості системних служб Windows. Прообразом функцій Read і WriteProcessMemory є NtReadVirtualMemory і NtWriteVirtualMemory, що експортуються бібліотекою ntdll.dll. Ядро операційної системи не експортує ці функції, тому доступ до них можна отримати тільки через переривання int 2eh, використовуючи в якості індексу значення 0xA4 і 0xF0 відповідно (операційна система Windows 2000 SP4).

Для підключення відладчика до процесу використовується Win32 функція DebugActiveProcess, яка створює і ініціалізує LPC порт, а потім підключає до нього відладжуваний процес. Порт використовується для управління відладжуваним процесом, а також для отримання налагоджувальної інформації. Підключення LPC порту до відладжуваного процесу здійснюється за допомогою

виклику недокументованого сервісу ядра ZwSetlnformationProcess, доступного через SSDT по індексу 0xC6.

При завантаженні, драйвер захисту підміняє покажчики на функції NtReadVirtnalMemory, NtWrite VirtualMemoiy і ZwSetlnformationProcess, підставляючи замість них посилання на власні методи, в яких здійснюється перевірка правомірності доступу. Драйвер зберігає список унікальних ідентифікаторів процесу (PID), на які розповсюджується його захисна дія, і при кожному виклику зіставляє переданий дескриптор процесу з цим списком. У разі виявлення спроби несанкціонованого доступу, інформація про час, ідентифікатор процесу порушника і облікового запису користувача зберігається в спеціальній буферній області підкачуваного пулу пам'яті і, по першому запиту клієнтського додатку реєструється в журналі аудиту.

Спроба несанкціонованого вивантаження драйвера захисту, наприклад, з використанням функцій SCM, призведе до екстреної зупинки системи і виведення так званого «синього екрану смерті» (BSOD) з кодом помилки MANUALLY_INITIATED_CRASH.

3.2.2 Збір статистики викликів системних сервісів

Одним з методів виявлення шпигунських програмних модулів, які використовуються сенсором СВА є аналіз статистики викликів системних сервісів, коли дані про поточний потік викликів зіставляються з існуючими сигнатурами атак з метою виявлення шкідливої активності. Найбільш зручним методом реалізації подібного механізму є використання таблиці SSDT, оскільки вона є єдиним шлюзом, через який проходять всі виклики. При використанні даного підходу, заміщаюча функція здійснює збереження контексту і області аргументів в стеку викликаючого потоку в спеціальний невивантажний (nonpaged) буфер пам'яті, після чого – викликає оригінальний системний сервіс. Основною проблемою, пов'язаною з реалізацією даного методу, є необхідність написання більше 300 функцій-«заглушок» для всіх існуючих системних сервісів, що веде до збільшення початкового коду і ускладнює подальший супровід системи. Вирішенням даної проблеми є використання універсальної функції-«заглушки»,

вперше запропонованою в роботі [47], скорочений лістинг якої приведений нижче.

```
#define HOOK_STUB                       \
_asm push eax                           \
_asm mov eax, offset HookLabel2         \
_asm call eax
//...
_asm
{
        jmp HookLabel3
        ALIGN 8
HookLabell:
}

HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB //0x08
//...
HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB HOOK_STUB //0x170

_asm
{
HookLabel2:
                pop eax
                pushfd
                push ebx
                push ecx
                push edx
                push ebp
                push esi
                push edi

                sub eax, offset Hooklabel1
                mov ecx, SSDT_SYMBOLS_MAX
                mul ecx
                mov ecx, offset Hooklabel2
                sub ecx, offset Hooklabel1
                div ecx
                dec ax

                push eax
                call HookStatisticalRoutine
```

```
                pop edi
                pop esi
                pop ebp
                pop edx
                pop ecx
                pop ebx
                popfd

                xchg eax,[esp]
                ret
HookLabel3:
                mov dwLabell, offset HookLabell
                mov dwLabel2, offset HookLabel2
                }
```

При початковій ініціалізації драйвера захисту, створюється копія таблиці дескрипторів, після чого SSDT заповнюється покажчиками на полі макросів HOOK_STUB, що реалізовують масив однорідних точок входу. При виклику системного сервісу, макрос зберігає значення регістра eax в стеку і здійснює виклик за лінійною адресою vsnrb HookLabel2. Даний код видаляє адресb повернення із стеке, запобігаючи тим самим зворотному переходу до масиву точок входу, і обчислює зсув поточної точки входу відносно початку і кінця масиву, знаходячи тим самим індекс викликаного системного сервісу. Обчислене значення індексу заноситься в стек і відбувається виклик функції збору статистики, яка, після закінчення своєї роботи, повертає покажчик на справжній дескриптор, далі відбувається очищення стека і виклик оригінального сервісу.

3.2.3 Сигнатури атак і реагування

Сигнатура атаки (СА) є текстовим описом, що формалізує її базові аспекти і що дозволяє здійснювати її ефективне розпізнавання. Для уніфікації процедур обробки сигнатур, в якості базової мови опису використовується мова XML. Простий приклад сигнатури для реалізованого прототипу інтелектуального сенсора виглядає таким чином:

```
<?xml version="1.0" encodings "utf-8" ?>
<ids:head ver="1.0" sval="high"type= "local">
<ids:head ver="1.0"sigid="0xFED93">
</ids:head>
```

```
<ids:bodyver="1.0">
<ids:sensor        id="l" name="garm.sys" iotype="IOCTL"  subitem="SSDT_CHECK">
<ids:data type="modification" value="10">
<ids:datarefinstance= "schecksys " routine = "0x7ffed028" status — "2 ">
<ids:callcheck type = "native ">
<ids:routine name- "NtCreateFile " module= "ntoskrnl.exe ">
<ids:arg value= "Oxl 2c "/><ids:arg value= "0x00 "/>
<ids:arg value="0x00 "/>< ids:arg value= "0x7ffe1200"/>
<ids:arg value= "0x00 "/><ids:arg value= "0x00 "/>
<ids:arg value= "0x00 "/><ids:arg value= "0x00 "/>
<ids:arg value= "0x00 "/>
</ids: routine > </ids: callcheck>
</ids:dataref> </ids:data>
</ids:sensor> </ids: body>
```

Дана сигнатура описує спробу заміни покажчика на функцію NtReadFile в модулі ntoskrnl.exe драйвером ядра scheck.sys.

Сигнатура реагування (дій у відповідь) є формалізованим описом списку заходів, які необхідно зробити для придушення певної атаки. На відміну від сигнатур атак, які можуть генеруватися автоматично модулями аналізу СВА, сигнатури дій у відповідь задаються жорстко на етапі розробки або можуть бути додані адміністратором системи. Вони є текстовим XML-стрічками наступного формату:

```
<?xml version= "1.0" encodings "utf-8" ?>
<ids:head ver= "1.0 " sval= "high " type= "local">
<ids:head ver= "1.0 " sigid= "0x0034 ">
</ids:head>
<ids:bodyver="1.0">
<ids:sensor id="l" name="garm.sys" iotype="IOCTL"         subitem ="SSDT_CHECK">
<ids:responce methodJd= "0x2 " method_data= "FULL ">
<ids:response method_id= "0xD " method_data= "NULL ">
</ids:sensor> </ids: body>
```

Приведений приклад сигнатури дій у відповідь описує реакцію на СА, описану в розділі 3.2.3.

3.2.4 Структура центральної бази даних

В якості сховища даних прототип інтелектуальної системи виявлення атак дозволяє використовувати будь-яку реляційну СУБД, що підтримує доступ до

даних за допомогою механізму ADO (ACTIVEX Data Objects). Використання універсального механізму доступу до даних дозволяє забезпечити незалежність системи від виробника конкретної СУБД, що використовується на підприємстві, до тих пір, поки для неї існує коректний OLE DB і задає місцеположення провайдер, що реалізовує підтримку базових запитів і команд мови SQL.

Структура центральної бази даних, що використовується в якості сховища конфігураційної інформації розподіленої системи виявлення атак, показана на рисунку 3.5.

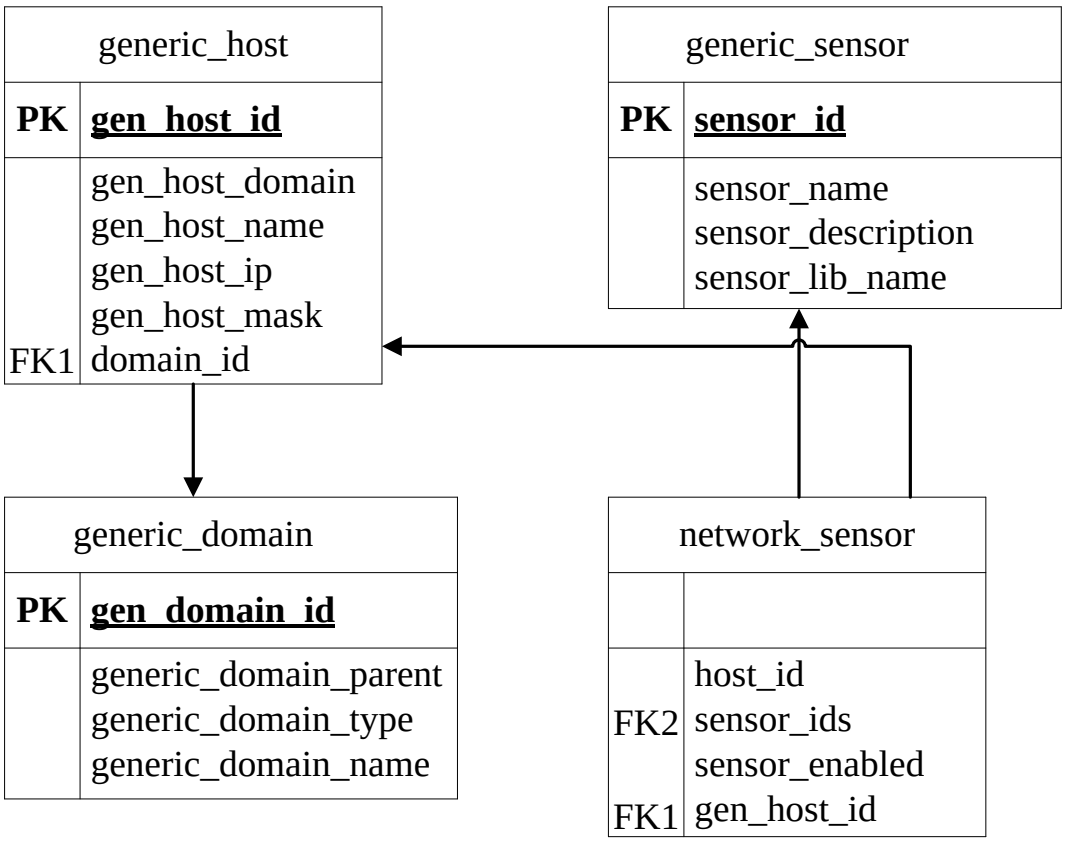

Рисунок 3.5 – Структура центральної бази даних

Інформація, що міститься в центральній базі даних, описує мережеву конфігурацію СВА і задає місцеположення і параметри конфігурації окремих сенсорів і блоків аналізу даних. Для спрощення налаштування і управління СВА, система використовує ієрархічний поділ мережі на сукупність доменів, кожен з яких є логічним сегментом топології мережі (наприклад, підмережа служби безпеки, бухгалтерії і т. д.) і може містити в собі інші домени і окремі хости, що є серверами і робочими станціями організації.

Конфігурація доменів міститься в таблиці generic_domain, структура якої описана в таблиці 3.1.

Таблиця 3.1 – Структура таблиці generic_domain

Ім'я поля	Тип	Опис
domainid	int (PK)	Унікальний ідентифікатор домена
domain_parent	int (FK)	Посилання на батьківський домен (використовується нуль для кореневого домена)
domainname	nvarchar (50)	Ім'я домена
domaindccr	nvarchar (250)	Опис домена

Необхідно відзначити, що доменне розділення використовується виключно в цілях логічної організації мережевого простору і може не відповідати можливій доменній структурі Microsoft Active Directory.

Для зберігання інформації про сервери і робочі станції підприємства використовується таблиця generic_host, структура якої представлена в таблиці 3.2.

Таблиця 3.2 – Структура таблиці generic_host

Ім'я поля	Тип	Опис
gen hostid	int (PK)	Унікальний ідентифікатор хоста
gen host_domain	nvarchar (250)	Ім'я домена Active Directory, до якого належить хост
genhostname	nvarchar (250)	Ім'я хоста
genhostip	int	IPv4 - адреса хоста
gen_host_mask	int	Маска підмережі хоста
domainid	int (FK)	Ідентифікатор логічного домена, до якого належить хост

Базова конфігурація сенсорів СВА зберігається в таблиці generic_sensor, структура якої показана в таблиці 3.3.

Таблиця 3.3 – Структура таблиці generic_sensor

Ім'я поля	Тип	Опис
sensorid	int (PK)	Унікальний дескриптор сенсора
sensor name	nvarchar (50)	Ім'я сенсора
sensor description	nvarchar (100)	Опис сенсора
sensortype	int	Тип сенсора
sensorenabled	bit	Дозвіл на запуск сенсора
sensorhostid	int (FK)	Унікальний ідентифікатор хоста, до якого належить сенсор

Необхідно відзначити, що кожен сенсор володіє набором своїх власних таблиць, що містять його конфігураційну інформацію, таблиця generic_sensor використовується виключно для візуалізації логічної топології системи виявлення атак.

3.2.5 Таблиці конфігурації локального сенсора СВА

Будь-який сенсор СВА може використовувати центральну базу даних для зберігання своєї конфігураційної інформації внаслідок того, що механізм доступу до даних ADO забезпечує прозорий доступ до віддалених джерел даних, проте подібний підхід не є відмовостійким, оскільки не дозволяє забезпечувати автономну роботу сенсорів СВА в умовах недоступності центральної бази даних (DoS-атака, відмова комунікаційного устаткування і т. д). Для забезпечення можливості автономної роботи сенсорів СВА, кожен хост системи містить власну невелику СУБД, яка обслуговує потреби сенсорів даного хоста. В якості такої СУБД можуть використовуватися MSDE, MS Access, MYSQL і ряд інших СУБД вільного доступу. Структура таблиць реалізованого сенсора виявлення шпигунських модулів показана на рисунку 3.6.

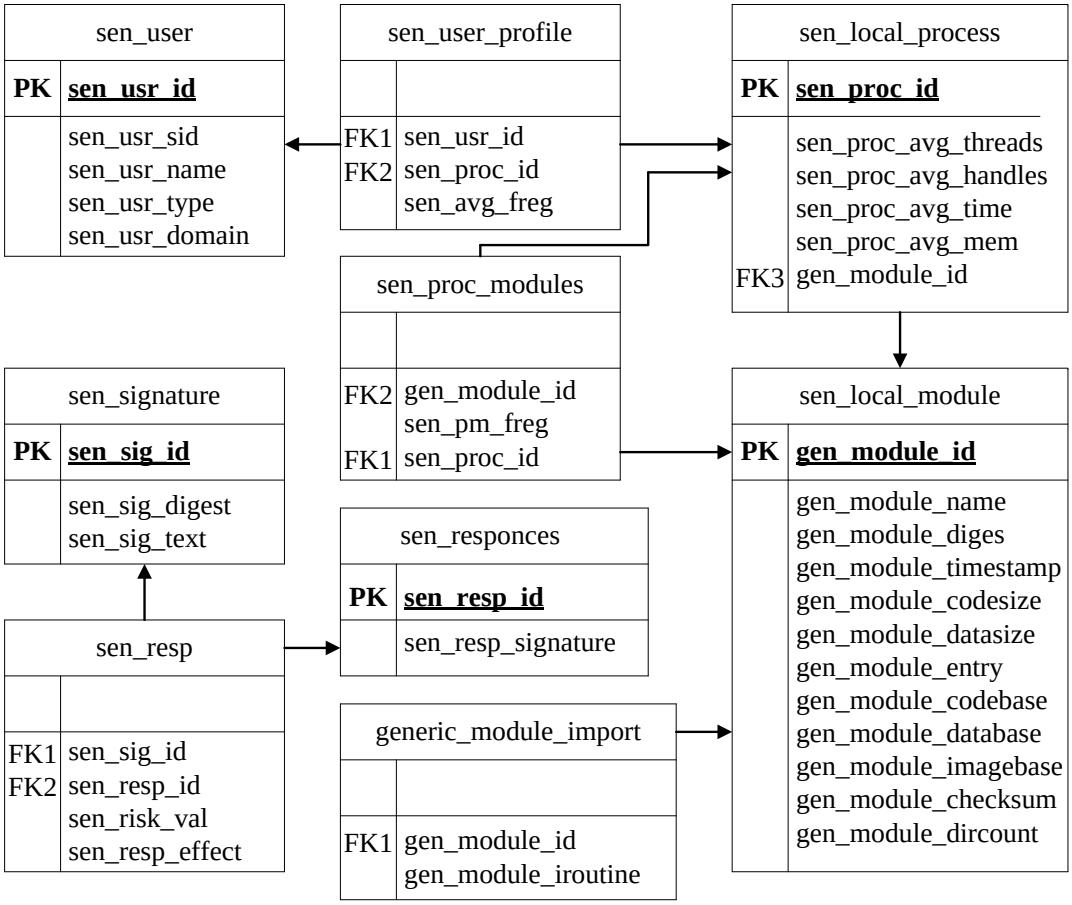

Рисунок 3.6 – Структура таблиць локального сенсора

В процесі роботи сенсор формує профіль користувача, в якого заноситься інформація про його базові параметри (таблиця sen_user), параметри процесів, що запускаються, і завантажувані модулі (таблиці sen_local_process, sen_local_module, generic_module_import, sen_proc_modules і sen_user_profile). Структура даних таблиць описана в таблицях 3.4–3.9.

Таблиця 3.4 – Структура таблиці sen_user

Ім'я поля	Тип	Опис
sen_usr id	int (PK)	Унікальний ідентифікатор користувача
sen_usr_sid	nvarchar (250)	Текстове представлення SID користувача
sen_usr_name	nvarchar (250)	Ім'я користувача
sen_usr_type	bit	Тип користувача (локальний/доменний)
senusrdomain	nvarchar (250)	Домен користувача

Таблиця 3.5 – Структура таблиці sen_local_process

Ім'я поля	Тип	Опис
sen_proc_id	int (PK)	Унікальний ідентифікатор процесу
sen_proc_avg_threads	float	Середнє число потоків процесу
scn_avg_handles	float	Середнє число дескрипторів процесу
sen_avg_time	float	Середній час, що споживається процесором за годину роботи
senavgmem	float	Середній об'єм оперативної пам'яті, що споживається
gen_module_id	int (FK)	Ідентифікатор виконуваного модуля

Таблиця 3.6 – Структура таблиці sen_local_module

Ім'я поля	Тип	Опис
genmoduleid	int (PK)	Унікальний ідентифікатор модуля
gen module name	nvarchar (250)	Ім'я модуля
gen_module_digest	nvarchar (250)	Текстове представлення згортки модуля MD5
gen_module_timestamp	datetime	Мітка часу модуля
gen_module_codesize	int	Розмір секції коду
gen_module_datasize	int	Розмір секції даних
gen_module_entry	int	Адреса точки входу
gen_module_codebase	int	Адресу початку секції коду
gen module_database	int	Адресу початку секції даних
gen module_imagebase	int	Адреса завантаження модуля
gen modulechecksum	int	Контрольна сума модуля
genmoduledircount	int	Кількість віртуальних каталогів образу

Таблиця 3.7 – Структура таблиці generic_module_import

Ім'я поля	Тип	Опис
gen_module_id	int (FK)	Ідентифікатор модуля
gen moduleiroutine	nvarchar (100)	Ім'я функції, що імпортується

Таблиця 3.8 – Структура таблиці sen_proc_modules

Ім'я поля	Тип	Опис
gen moduleid	int (FK)	Ідентифікатор модуля
sen_pm_freq	float	Частота завантаження
sen_proc_id	int (FK)	Ідентифікатор процесу

Таблиця 3.9 – Структура таблиці sen_user_profile

Ім'я поля	Тип	Опис
senusrid	int (FK)	Ідентифікатор користувача
sen_proc_id	int (FK)	Ідентифікатор процесу
sen avg_freq	float	Середня частота завантаження

Сигнатури атак, які згенеровані модулем аналізу даних, зберігаються в таблиці sen_signature (таблиця 3.10), сигнатури дій у відповідь – в таблиці sen_responces (таблиця 3.11).

Таблиця 3.10 – Структура таблиці sen_signature

Ім'я поля	Тип	Опис
sen_sig_id	int (PK)	Унікальний ідентифікатор сигнатури
sen_sig_digest	nvarchar (50)	Згортка MD5 сигнатури
sen_sig_text	text	Текст сигнатури

Зв'язок між сигнатурами атак і сигнатурами дій у відповідь здійснюється шляхом занесення даних в таблицю sen_resp (таблиця 3.12). При кожному

використанні сигнатур дій у відповідь здійснюється обчислення величини зменшення ризику, що характеризує ступінь ефективності контрзаходу, який також заноситься в дану таблицю.

Таблиця 3.11 – Структура таблиці sen_responces

Ім'я поля	Тип	Опис
sen resp_id	int (PK)	Унікальний ідентифікатор міри у відповідь
sen respsignature	text	Сигнатура міри у відповідь

Таблиця 3.12 – Структура таблиці sen_resp

Ім'я поля	Тип	Опис
sensigid	int (FK)	Ідентифікатор сигнатури
sen resp_id	int (FK)	Ідентифікатор міри у відповідь
senriskval	float	Величина ризику
sen_resp_effect	float	Середня ефективність міри

3.2.6 Консоль адміністратора

Консоллю адміністратора є інструмент конфігурації і управління системою виявлення атак. Для уніфікації процедур адміністрування, консоль реалізована у вигляді MMC (Microsoft Management Console), що дозволяє легко інтегрувати її з існуючими механізмами адміністрування ЛОМ.

Реалізовані наступні вузли управління СВА:

1) дерево сенсорів IDS – дозволяє будувати віртуальне представлення мережі на основі ієрархічного дерева доменів, хостів і сенсорів;

2) базові параметри сенсорів – забезпечує налаштування глобальних параметрів, що впливають на всі сенсори мережі;

3) об'єкти операційної системи – містить список довірених об'єктів операційної системи (об'єкти мінімального контролю);

4) параметри безпеки - настройка прав доступу користувачів до СОА, а

також додаткові обмеження політики безпеки мережі.

3.2.7 Редактор моделей

Редактор моделей є графічною утилітою для побудови моделей компонентів ІС, редагування їх параметрів, покрокового моделювання функціонування компонентів ІС і розрахунку ризику.

Програма дозволяє:

1) будувати нечіткі когнітивні карти будь-якої складності із збереженням результатів в базу даних СВА, або у вигляді файлу;

2) моделювати поведінку ІС з використанням як реальних даних з сенсорів СВА, так і моделі атак, описаної в другому розділі;

3) обчислювати величини ризиків.

Отже, даний розділ присвячений опису реалізації дослідницького прототипу інтелектуальної СВА, на основі динамічної моделі, розробленої в другому розділі. Проведений синтез архітектури сенсорів і модулів, розроблена структура бази даних на основі формалізованих моделей IDEF, отриманих в другому розділі. В якості базової платформи для розробленого прототипу вибрана ОС Microsoft Windows, як найбільш поширена в корпоративному середовищі. Реалізований набір сенсорів, які дозволяють збирати інформацію про активність системних і користувацьких процесів для виявлення і блокування шкідливої активності, а також інструментарій налаштування і тестування системи. Даний модуль дозволяє здійснювати налаштування системи прийняття рішень (структура і конфігурація нечіткої когнітивної карти, склад сенсорів, і т. д.), а також проводити тестування СВА як в режимі збору реальних даних, так і з використанням моделі атак, описаної в другому розділі.

ВИСНОВКИ

У дипломній роботі поставлена і вирішена задача підвищення ефективності виявлення атак на інформаційну систему шляхом динамічної оцінки ризиків її функціонування з використанням динамічних моделей на основі нечітких когнітивних карт.

Основні результати і висновки можна сформулювати таким чином:

1. Аналіз систем виявлення атак, представлених в даний час на ринку, показав, що вони володіють низькою ефективністю (високі значення помилок I і II роду) при виявленні невідомих атак. Спроби вирішення даної проблеми з використанням систем на основі штучного інтелекту володіють низькою універсальністю і поки не набули широкого поширення.

2. Розроблений комплекс системних моделей системи виявлення атак на інформаційну систему із застосуванням SADT методології, що дозволили виявити основні процеси, що лежать в основі її функціонування, виявити її компоненти, їх взаємозв'язки і сформулювати вимоги до реалізації системи, виходячи з сучасних вимог забезпечення захищеності ІС.

3. Запропонована архітектура і алгоритми інтелектуальної системи виявлення атак, засновані на використанні нейро-нечітких когнітивних карт, що дозволяє підвищити ефективність системи виявлення атак за рахунок прогнозування і управління ризиками інформаційної системи.

4. Для підвищення зменшення неоднозначності експертних даних, був запропонований алгоритм навчання нечіткої когнітивної карти на наборі еталонних сценаріїв, шляхом використання алгоритму Back Propagation.

5. Розроблений дослідницький прототип системи виявлення атак на основі нейронечітких когнітивних карт. Тестування даного прототипу довело високу ефективність використовуваного підходу при виявленні атак на компоненти ІС. Зокрема, розроблений прототип дозволяє виявити 91,3% атак на компоненти ІС, при цьому помилка другого роду не перевищує 18,8%. Після навчання моделі на наборі еталонних сценаріїв атак, величина помилки другого роду знизилася до 6%.

6. Адекватність розробленої динамічної моделі була підтверджена шляхом імітаційного моделювання атак на ІС з використанням даних з сенсорів СВА в режимі реального часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вишневский В.М. Теоретические основы проектирования компьютерных сетей. - М. : Техносфера, 2003. – С. 76 - 81.

2. Завгородний В.И. Комплексная защита информации в компьютерных системах: Учеб. пособие. - М.: Логос, 2001. – С 57 - 61.

3. Зегжда П.Д. Теория и практика обеспечения информационной безопасности. - М.: Яхтсмен, 1996. – С 92 - 95.

4. Лазарев В.Г. Интеллектуальные цифровые сети. - М.: Финансы и статистика, 1996. - С 74 - 76.

5. Verwoerd T., Hunt R. Intrusion detection techniques and approaches // Computer Communications. – 2002. – №25.

6. Автоматизированные информационные технологии в экономике. Под. ред. Г.А.Титоренко. - М.: Компьютер ЮНИТИ, 1998, - 336 с.

7. НД ТЗІ 1.1-003-99. Системи технічного захисту інформації. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу. – Введ. 01.07.99. – К.: ДСТСЗІ СБ України, 1999. – 24 с.

8. ДСТУ 3396.2-97. Захист інформації. Технічний захист інформації. Терміни та визначення. – Введ. 01.01.98. – К.: Держстандарт України, 1998. – 12 с.

9. Анализ защищенности: сетевой или системный уровень? Руководство по выбору технологии анализа защищенности. Пер.с англ. Лукацкого А.В., Цаплева Ю.Ю. Internet Security Systems, 1999. - С 22-24.

10. Аграновский А.В. Статистические методы обнаружения аномального поведения в СОА. // Информационные технологии, 2005, №1. - С. 61-64.

11. Лникеев М.В., Сидоров И.Д. Обнаружение аномального поведения пользователя в операционной системе Windows на основе анализа работы с приложениями. // Материалы V международной научно - практической конференции «Информационная безопасность», г. Таганрог, № 4(33), 2003, - С. 134-137.

12. Васильев В.И., Катаев Т.Р. Применение нечетких сетей Петри для

анализа безопасности локальной вычислительной сети. // Вычислительная техника и новые информационные технологии. - Уфа: Изд-во У Г АТУ, 2007, -С.166-170.

13. Васильев В.И., Катаев Т.Р., Свечников Л.А. Комплексный подход к построению интеллектуальной системы обнаружения атак. // Системы управления и информационные технологии. - Воронеж: Научная книга, 2007, - №2, - С. 76-82.

14. Васильев В.И., Кудрявцева Р.Т., Савина И.А., Шарипова И.И. Моделирование информационных рисков ВУЗа в среде Matlab с помощью нечеткого когнитивного подхода. // Вычислительная техника и новые информационные технологии. - Уфа: Изд-во УГАТУ, 2007, - С.170-177.

15. Куликов Г.Г. Системное проектирование автоматизированных информационных систем : Учеб.пособие - Уфа: Изд-во УГАТУ, 1999, - С 61 -63.

16. Мельников В. П. Информационная безопасность и защита информации. / под ред. С. А. Клейменова. - М.: Академия, 2006. - С. 46 - 49.

17. Вихорев СВ. Классификация угроз информационной безопасности. http://www.cnews.ra/reviews/free/oldcom/security/elvis_class.shtml.

18. Паулаускас Н., Гаршва Э. Классификация атак компьютерных систем // Электроника и электротехника. - Каунас: Технология, 2006. - № 2(66), - С. 84-87.

19. Махинов Д.В., Рогозин Е.А. Комплексная оценка угроз качеству функционирования эргатических информационно-управляющих систем. // Телекоммуникации 2002. №2, -С. 33-40.

20. Сумин В.И., Рогозин Е.А., Дубровин А.С. Использование полумарковской модели программной системы защиты информации для анализа ее защищенности. // Телекоммуникации 2001. №11,- С. 39-41.

21. Verbruggen H.B., Babuska R. Constructing fuzzy models by product space clustering // Fuzzy model identification. - Berlin: Springer, 1998. - pp. 53-90.

22. Васильев В.В. Сети Петри, параллельные алгоритмы и модели мультипроцессорных систем. - Киев: Наук, думка, 1990, - С 184 - 188.

23. Manikantan Ramadas. Detecting Anomalous Network Traffic with self-organizing maps.2002.

24. Козлов В.Н., Куприянов В.Е., Шашихин В.Н. Вычислительная математика и теория управления. Под ред. Козлова В.Н. - СПб.: Изд-во СПбГТУ,

1996. - С. 97-105.

25. Перегудов Ф.И. Введение в системный анализ. - М.: Высшая школа, 1989, - С. 172-175.

26. Майн Х., Осаки С. Марковские процессы принятия решений. - М.: Наука, 1997, - С. 83-88.

27. Нумеллин Е. Общие неприводные цепи Маркова и неотрицательные операторы. - М.: Мир. - 1989. - С. 94 - 99.

28. Рсвюз Д. Цепи Маркова. / Пер. с англ. Малиновский В. К. - М.: РФФИ, 1997, - С. 69-72.

29. Борисов В. В. Нечеткие модели и сети. - М.: Горячая линия - Телеком, 2007, - С. 75-78.

30. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечеткая логика и искусственные нейронные сети: Учеб. пособие. — М.: Издательство физико-математической литературы. 2001 – С. 33 - 37.

31. Jang J.S., Sun CT., Mizutani E. Neuro-fuzzy and soft computing. - N.Y.: Prentice Hall, 1997.

32. Методы робастного, нейро-нечеткого и адаптивного управления : учебник / Под ред. Ягупова Н. Д. - М.: Изд-во МГТУ им. Баумана, 2001, - С. 91 - 95.

33. Ломазова И.А. Вложенные сети Петри: моделирование и анализ распределенных систем с объектной структурой. - М.: Научный мир, 2004. -С 188-191.

34. Питерсон Дж. Теория сетей Петри и моделирование систем; Пер. М. В. Горбатовой; под ред. В. А. Горбатова. - М.: Мир, 1984. - С. 321 - 325.

35. Васильев В.И., Кудрявцева Р.Т., Савина И.А., Шарипова И.И. Построение нечетких когнитивных карт для анализа и управления информационными рисками ВУЗа. // Вестник УГАТУ. Уфа:Изд-во УГАТУ, 2007, №2(27), - С. 199-209.

36. Куликов Г.Г. Системное проектирование автоматизированных информационных систем : Учеб.пособие - Уфа: Изд-во УГАТУ, 1999, - С 61 -63.

37. Сохен М.Ю., Кузнецов А.С. Модель потока атакующих воздействий на

систему комплексной защиты информационно - вычислительной сети. // Материалы научно - технической конференции «Информационная безопасность автоматизированных систем». Воронеж: Истоки, 1998, - С. 320-325.

38. Колчин В.Ф. Случайные графы. - М.: ФИЗМАТЛИТ, 2004. – С. 264 - 268.

39. Костров Д. Системы обнаружения атак // BYTE. - 2002.

40. Лукацкий А.В. Обнаружение атак. - СПб.:БХВ-Петербург, 2001. - С. 521-523.

41. Chakka R., Harrison P. G. F Markov modulated multi-server queue with negative customers // Acta Informatica. 2001. - V. 37.

42. Ивахненко А.Г. Моделирование сложных систем по экспериментальным данным. - М.: Радио и связь, 1987, - С. 71 - 73.

43. Севастьянов Б.А. Ветвящиеся процессы. - М.: Наука, 1971. - С. 389-391.

44. Колисниченко Д.Н. Rootkits под Windows. - СПб.: Наука и Техника, 2006. – С. 311 -313.

45. Васильев В.И., Свечников Л.А. Подход к реализации нейросетевого сенсора интеллектуальной системы обнаружения атак. // Вычислительная техника и новые информационные технологии. – Уфа: Изд-во УГАТУ, 2007, - С.161-166.

46. Соломон Д., Руссинович М. Внутреннее устройство Microsoft Windows 2000. Мастер-класс. / Пер. с англ. - СПб.: Питер. 2001. - С. 692 - 694.

47. Шрайбер С. Недокументированные возможности Windows 2000. - СПб.: Питер, 2002, - С. 447 -451.

48. Методичні рекомендації до виконання дипломної роботи з освітньо-кваліфікаційного рівня “Магістр”. Спеціальність „Комп’ютерні системи та мережі” / О.М. Березький, Р.Б. Трембач, Г.М. Мельник / Під ред. О.М. Березького – Тернопіль: ТНЕУ, 2012.– 42 с.

Додаток А

Довідка про використання