Хмельницький національний університет
Факультет програмування та комп’ютерних і телекомунікаційних систем
Кафедра комп’ютерної інженерії та системного програмування

ДИПЛОМНА РОБОТА МАГІСТРА

Розподілена система виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах на основі баєсівської мережі

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп’ютерна інженерія

ДРКІСПр.015095.19.02.22 ПЗ

Виконав:
студент 2 курсу, група КІ2м-19-1 _____ О. О. Шевцов
Підпис

Керівник:
д-р.техн.наук, професор _____ О. С. Савенко
Підпис

До захисту допускаю:
Зав. кафедри КІСП д-р.техн.наук, професор _____ Т. О. Говорущенко
Підпис

____ _____ 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ПРОГРАМУВАННЯ ТА КОМП'ЮТЕРНИХ І ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА ПІДГОТОВКИ МАГІСТРА

ЗАТВЕРДЖУЮ

Зав. кафедри КІСП

_____ Т. О. Говорущенко

" 03 " 09 2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Шевцову Олександру Олександровичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Розподілена система виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах на основі баєсівської мережі

Керівник проекту (роботи) Савенко Олег Станіславович, д-р.т.н, професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.09.2020 р. № 118

2. Строк подання студентом проекту (роботи) на кафедру 05.05.2021 р.

3. Вихідні дані до проекту (роботи). Наукові джерела, що стосуються розподілених систем, систем виявлення зловмисного програмного забезпечення та баєсівських мереж

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

Аналіз предметної області та постановка задачі

Архітектура розподіленої системи

Баєсовська мережа у розподіленій системі

Реалізація системи та експерименти

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

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІСП		
Антиплагіат	Нічепорук А.О., доцент кафедри КІСП		

7. Дата видачі завдання «___03___» _вересня_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики ДРМ з керівником	01.09.2020	Виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	20.09.2020	Виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	07.10.2020	Виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	21.11.2020	Виконано
5	Розробка 2 розділу написання ДРМ	12.01.2021	Виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	20.01.2021	Виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	15.02.2021	Виконано
8	Оформлення пояснювальної записки згідно вимог	23.03.2021	Виконано
9	Попередній захист ДРМ	29.04.2021	Виконано
10	Захист ДРМ на засіданні ЕК	04.05.2021	Виконано

Студент  О. О. Шевцов

Підпис Ініціали, прізвище

Керівник проекту (роботи)  О. С. Савенко

Підпис Ініціали, прізвище

РЕФЕРАТ

Тема дипломної роботи: Розподілена система виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах на основі баєсівської мережі.

Автор роботи: Шевцов Олександр Олександрович.

Керівник роботи: Савенко Олег Станіславович.

Пояснювальна записка: 118 с., 13 рис., 8 табл., 3 дод., 80 джерел.

Ключові слова: Баєсівська мережа, машинне навчання, розподілені системи, зловмисне програмне забезпечення, система виявлення вторгнень, система виявлення зловмисного програмного забезпечення, локальні комп’ютерні мережі.

Об’єктом дослідження є: розподілені системи виявлення зловмисного програмного забезпечення.

Предметом дослідження є: процес виявлення зловмисного програмного забезпечення на основі баєсівської мережі засобами розподілених систем.

Метою дипломної роботи є: розробка розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах на основі баєсівської мережі.

Наукова новизна отриманих результатів полягає в тому, що удосконалена розподілена система для виявлення зловмисного програмного забезпечення на основі мереж Баєса, що на відміну відомих спрощує виявлення нових типів вірусних загроз.

На основі проведених досліджень була розроблена розподілена система виявлення вірусної загрози в локальних комп’ютерних мережах на базі мереж Баєса.

Практична значимість отриманих результатів полягає у спрощенні виявлення вірусних загроз.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЗПЗ – зловмисне програмне забезпечення

ОС – операційна система

БМ – Баєсовська мережа

АРС – антивірусна розподілена система

РС – розподілена система

Р2Р – однорангова мережа

ВСТУП

Актуальність роботи. Наш сучасний світ вже неможливо уявити без використання комп’ютерів та інформаційних технологій. Сьогодні вони вже стали невід’ємною частиною нашого світу, вони приймають участь у нашому житті та оточують нас майже повсюди. Ми щодня використовуємо мільйони смартфонів, ноутбуків та різних розумних пристроїв, що допомагають нам, автоматизують та спрощують наші процеси. Бронювання квитків, замовлення в інтернеті, перегляд відео на стрімінгових платформах, спілкування в соціальних мережах, роботизація підприємств – все це надається нам різними сервісами, що використовують тисячі різних технологій. Але разом із розвитком та автоматизацію постає одна серйозна проблема – кіберзлочинність. Технічні засоби кіберзлочинців з кожним днем так само покращуються та розвиваються. Сьогодні під загрозою не тільки великі корпорації або міжнародні банки – сьогодні кожен може бути під загрозою. І кількість цих загроз з кожним днем постійно зростає: вірусні загрози, ботнети, мережеві атаки, крадіжка особистих даних. Вже недостатньо просто мати засоби захисту – потрібно, щоб вони щодня розвивались, оновлювали свої бази та методи детектування загроз. При цьому вже мало просто виявляти існуючі загрози – потрібно вміти розпізнавати ще ті, що не були виявлені та досконально вивчені. Сьогоднішній та майбутній захист має бути не просто комплексним, він має бути розумним. Оскільки будь яку загрозу краще попередити та знешкодити заздалегідь, ніж виявляти результати її діяльності.

Зв'язок роботи з науковими програмами, планами, темами. Представлені у роботі дослідження, проводились в рамках держбюджетних НДР Хмельницького національного університету: № 1Б-2019 «Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем» (номер державної реєстрації 0119U100662); № 1Б-2021 «Самоорганізована розподілена система виявлення зловмисного програмного

забезпечення в комп’ютерних мережах» (номер державної реєстрації 0221U102011).

Метою дипломної роботи є розробка розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах на основі баєсівської мережі.

Задачі дослідження формуються наступним чином**:**

1) розглянути особливості та методи побудови сучасних розподілених інформаційних та обчислювальних систем та антивірусних засобів та рішень;

2) провести аналіз особливостей, переваг та недоліків існуючих рішень;

3) ознайомитись з баєсівською мережею, а також особливостями її використання в антивірусних системах та системах виявлення вторгнень;

4) розробити та реалізувати розподілену обчислювальну антивірусну систему з інтеграцією баєсівської мережі для виявлення зловмисного вірусного програмного забезпечення в локальних мережах;

5) провести аналіз та дослідження ефективності розробленої антивірусної системи у поєднанні з баєсівською мережею;

Об’єкт дослідження – процес функціонування розподілених систем виявлення зловмисного програмного забезпечення у локальних комп’ютерних мережах на основі баєсовської мережі.

Предмет дослідження – методи та способи побудови АРС для виявлення зловмисного програмного забезпечення у локальних комп’ютерних мережах на основі баєсовської мережі.

Методи дослідження. Для виконання визначених задач було використано такий матеріал:

1) теорії розподілених систем;

2) баєсовська мережа, формула Баєса, ациклічний граф, методи побудови БМ;

3) методи побудови РС;

4) теорії комп’ютерних мереж;

Наукова новизна отриманих результатів полягає у:

Удосконаленні розподіленої системи для виявлення зловмисного програмного забезпечення на основі мережі Баєса, що на відміну відомих спрощує виявлення нових типів вірусних загроз;

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій. Наукові положення, висновки і рекомендації дипломної роботи обґрунтовані коректним використанням баєсовської мережі, алгоритмів виявлення ЗПЗ та успішною реалізацією розробленої АРС виявлення зловмисного програмного забезпечення в комп'ютерних мережах.

Практичне значення одержаних результатів. На основі проведеного дослідження розроблена АРС спрощує виявлення вірусних загроз.

Особистий внесок здобувача. Результати дослідження, які надаються до захисту роботи, належать особисто автору.

Публікації. За темою дипломної роботи опублікована одна стаття у збірнику матеріалів конференції «Збірник наукових праць Конференції АПКН-2020».

Структура кваліфікаційної роботи. Дипломна робота магістра складається з реферату, вступу, чотирьох розділів, висновків, списку використаних джерел з 80 на 8 сторінках та двох додатків на 20 сторінках. Загальний обсяг роботи становить 118 сторінок, з них 72 сторінки основного тексту, 13 рисунків, 8 таблиць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Визначення області дослідження

З розвитком технологій, збільшенням рівня автоматизації і потреби у більш серйозних розрахунках та більш потужнішому обладнанні постає і необхідність розпаралелювання та розділення вирішуваних задач. Така необхідність виникла не тільки у спеціалізованих сферах (космонавтика або промисловість), а й у повсякденних задачах (мультимедійні бібліотеки або інтернет-торгівля). Для вирішення цих потреб і були розроблені розподілені системи.

Отже, областю дослідження даної магістерської роботи є розподілена система. Це поняття сформувалось досить давно, але не існує його єдиного та чіткого визначення. Вважається, що розподілена система – це система яка складається з багатьох розділених між собою компонентів проте для кінцевого користувача вона виглядає як єдине ціле. Така система сполучає велику кількість комп’ютерів у єдиний організм з метою виконання певних дій або розрахунків [1].

Найменшим елементом у цій системі виступає вузол, що складається з двох частин: самого комп’ютера з операційною системою та встановленого певного програмного комплексу, який поєднує цей комп’ютер з усією мережею. Саме цей додатковий програмний комплекс і є проміжним зв’язуючим елементом у розподіленій системі [2].

Всім розподіленим системам притаманні наступні характеристики:

1. Всі наявні у системі вузли розподілені у просторі.

2. Кількість вузлів системи може постійно змінюватись протягом роботи без перебудови наново всієї системи.

3. Система асинхронна та незалежна. Всі процеси обміну даними між компонентами є асинхронними та не керуються повністю окремим певним компонентом. При цьому окремо взята компонента не може керувати

процесом комунікації між двома іншими компонентами, що не зв’язані з нею [3].

4. Задачі для компонентів можуть бути паралельними.

РС мають ряд значних переваг у порівнянні зі звичайними програмними додатками та однокомп’терними системами:

1. Інкапсуляція від користувача всіх нюансів роботи, архітектурних рішень та розподілення ресурсів у системі. Користувачі взаємодіють із системою через єдиний спеціально розроблений інтерфейс [4].

2. Можливість дублювання ресурсу, що збільшує надійність системи при відмові вузла з певним ресурсом.

3. Сумісне використання апаратної та програмної частини комп’ютера у системі для обчислень. Це дозволяє не тільки розпаралелювати обрахунки а також будувати системи, що значно потужніші, продуктивніші та гнучкіші за однокомп’ютерні [5].

4. Можливість легкого необмеженого масштабування та додавання нових ресурсів за потреби.

Проте такі системи мають і ряд певних недоліків:

1. Важкість проектування, розробки та обслуговування [6]. Оскільки система може складатись з великої кількості різних комп’ютерів з різним залізом та операційними системами, кожен з яких може мати різні обмеження та різну швидкість зв’язку та доступ до ресурсів, на кожному із етапів роботи можуть виникнути різні проблеми, які потрібно передбачити та усунути. Також необхідно синхронізувати роботу більшості вузлів у випадку зупинки або відмови якогось елемента. Крім того важкість моніторингу стану системи зростає з кількістю її компонентів.

2. Важкість підтримки безпеки. Зі збільшенням кількості комп’ютерів та елементів збільшується і кількість місць де інформаціє може бути перехоплена і потребує захисту [7].

3. Затримка при роботі з даними. Чим більше система – тим більше елементів зв’язку і тим більше затримок може виникнути при обміні даними.

4. Важкість реалізації синхронного обміну компонентів. Ця проблема особливо постає при великій кількості різнопланових компонентів і необхідності серії паралельних обрахунків.

Для підтримки стабільної та ефективної роботи, а також безпроблемного нарощування елементів або перебудови системи до неї висувається ряд вимог:

1. Відкритість. Всі комунікації між всіма наявними у системі компонентами мають базуватись тільки на єдиних та доступних стандартах. Кожна компонента має містити документацію з описом даних та інтерфейсів для взаємодії з нею. Ця вимога особливо потрібна при розробці інших систем на базі вже існуючих або при частковому перенесенні компонентів системи на інші платформи [8].

2. Прозорість. Система має приховувати від користувача всі деталі її реалізації та роботи, а також взаємодію компонентів і ресурсів. Ідеально реалізована система має бути цілою, ілюзорно простою та зрозумілою для користувача [1, 9].

3. Масштабованість. Система має спокійно працювати з новими підключеними елементами та балансувати навантаження без перебудови всієї системи та переробки програмного коду [10].

4. Безпечність. Дані у мережі мають бути захищені від перехоплення, а також взаємодія з кожним компонентом має відбуватись тільки авторизованими користувачами. Помилки у роботі системи не мають призводити до втрати даних.

5. Стійкість. Компонент кожного вузла у мережі має без проблем дублюватись у випадку відмови вузла [11].

6. Однозначність. Всі запити в системі мають або повністю виконуватись або не виконуватись взагалі. Не допускається ситуація часткового опрацювання задачі при неможливості її опрацювання.

Сьогодні розподілені системи набули значного поширення в усіх сферах. Вони не тільки використовуються на підприємствах для серйозних

обчислень та сумісного використання ресурсів, а і у щодених задачах. Майже всі сучасні великі програмні продукти чи системи використовують розподілені технології. Так, компанії Apple, Netflix та Uber використовують розподілені системи баз даних Apache Cassandra [12]. Система моніторингу літаків Flightradar24 базується на даних з розподілених пристроїв, що надсилають інформацію до центру [13]. Подібні технології використовує і платформа Amazon Web Services для своїх хмарних обчислень [14]. Крім того більшість існуючих відомих антивірусних рішень так само є розподіленою клієнт-серверною системою або використовують її у своєму складі, що показує актуальність таких технологій. Це призвело до появи різного програмного забезпечення та технологій для спрощення розробки та побудови розподілених систем [15].

Отже, в даному підрозділі визначена область дослідження даної роботи – а саме розподілені системи. Представлено характерні риси, переваги, недоліки, вимоги до цих систем, а також дані стосовно.

1.2 Аналіз відомих методів та засобів

1.2.1 Методи побудови розподілених систем

Розглянемо основні методи побудови розподілених систем. Досить часто архітектури комбінуються, для покращення ефективності роботи системи [26] [27] [28] та забезпечення автономності [61].

1) клієнт-серверна архітектура надає можливість користувачеві отримати ресурси з віддалених комп'ютерів (серверів) з власного робочого комп'ютера. В такій архітектурі сервером є та складова структура розподіленої системи, що надає (реалізує) деякі сервіси. Клієнт, відповідно, – складова, що робить запит на сервіс [16]. Взаємозв'язок між клієнтом і сервером здійснюється за допомогою протоколів [23] [56] [57]. До таких протоколів можна віднести UDP протокол, TCP протокол тощо. UDP протокол не потребує встановлення зв'язку, що робить його досить ефективним. Основним недоліком його використання є те, що запити можуть губитись або пошкоджуватись, тому він не є безпечним. TCP протокол – це протокол, який потребує встановлення зв'язку, через це страждає його продуктивність, але такий зв'язок є досить безпечним. Коли клієнт відправляє будь який запит на сервер, він повинен встановити з ним зв'язок. Цей зв'язок розривається, як тільки сервер відповість на запит [21]. Існують різні можливості реалізації цієї архітектури [50] [58].

До переваг цієї архітектури можна віднести:

1. Програмне забезпечення серверної частини відокремлене від програмного забезпечення клієнтської частини.

2. Через те, що серверне обладнання має високу вартість, дана архітектура дає можливість зекономити кошти на обладнанні, тому що серверів менше ніж клієнтських машин.

3. Набагато нижче вимоги до клієнтських комп'ютерів.

4. Вся інформація зберігається на сервері, який захищений краще клієнтських машин.

До недоліків можна віднести [55]:

1. При відмові основної складової (сервера) розподіленої системи припиняється робота всієї системи [17].

2. Обслуговуванням серверу, як правило, займається окремий персонал компанії – системний адміністратор.

2) об'єктні розподілені системи. У даній архітектурі об'єктами можна назвати служби та ресурси розподіленої системи. За допомогою об’єктів реалізується прозорість розподілення, тобто вся реалізація системи є прихованою окрім інтерфейсу. Це дозволяє змінювати, модифікувати об’єкти системи не впливаючи на інтерфейс. Широко розповсюдженими об’єктними розподіленими системами є CORBA і DCOM [18].

Поява даної архітектури пов’язана з активним розвитком клієнт-серверної архітектури. Її основними задачами були ізолювати розподілені взаємодії та спростити написання програмного забезпечення розподілених систем відносно класичних клієнт-серверних систем, використовуючи об’єктно-орієнтовані методи та віддалені виклики об’єктів. В даній архітектурі немає розділення на клієнт і сервер. Будь який об’єкт системи може надавати та використовувати послуги інших об’єктів.

Принцип роботи даної архітектури полягає у роботі з об’єктами через їх власні методи. У кожного з об’єктів системи є свій стан, на який можна впливати через методи цього об’єкту. Методи використовуються через віддалені виклики за допомогою зовнішнього інтерфейсу. Для обміну даними по мережі використовуються технології серіалізації та десеріалізації, тобто стан об’єкту перетворюється у бінарний або XML-файл і навпаки, відповідно.

Головними перевагами даної архітектури є:

1. Спрощення розробки програмного забезпечення розподіленої системи.

2. Розділення інтерфейсу і реалізації.

3. Легко розширюється та модифікується.

4. Між собою можуть взаємодіяти різнорідні системи, наприклад ОС тощо.

До недоліків можна віднести складне проектування системи у порівнянні з архітектурою клієнт-сервер.

3) агентні технології. В даній архітектурі інтелектуальний агент це деякий активний процес, який взаємодіє і іншими агентами, здатні змінювати свій стан і поведінку в залежності від ситуації та відповідно до стану і поведінки інших агентів. Таким чином інтелектуальний агент імітує роботу активних елементів у динамічному (фізичному, біологічному, природньому) середовищі. Такі агенти здатні вирішувати проблеми без втручання користувача за рахунок наявності в них деяких знань про навколишнє середовище. Кожен з агентів є автономним (або частково автономним), має уявлення тільки про частину системи, децентралізований, здатен навчатися [19]. Система, в якій є певна множина інтелектуальних агентів, які зв’язані між собою і мають власні унікальні характеристики називають мультиагентною системою. Частіше за все мультиагентні системи є ієрархічними [20].

Перевагами цієї технології є:

1. Мультиагентні системи здатність вирішувати складні завдання, які неможливо вирішити за допомогою одного агента чи монолітної системи [59] [60].

2. Оперативне планування розподілу ресурсів відповідно до потреб, що динамічно змінюються.

До недоліків даної технології можна віднести

1. Складність її практичної реалізації, через складність створення віртуальних середовищ для функціонування інтелектуальних агентів та реалізація самих агентів.

2. Агентні платформи (віртуальне середовище) не є безпечними [39].

4) сервіс-орієнтована архітектура (СОА). Дана технологія з'явилася внаслідок виникнення необхідності поліпшити адаптивність розподілених систем до швидко мінливих вимог і завдань, так як вже існуючі технології не справлялися з цим завданням через дуже швидкий розвиток комп'ютерних та інформаційних технологій [42] [51]. СОА є узагальненням попередніх технологій розробки програмних інформаційних систем корпоративного рівня і вище.

Для побудови СОА системи чітко дотримуються кількох принципів [22] [43]. Є сервіси (компоненти) з власним інтерфейсом (контракти), що не залежать від платформи, мови програмування тощо. Сервіси взаємодіють між собою використовуючи відкриті, часто використовувані стандарти. Кожен сервіс – це окрема функція, задача що повторюється і частиною загальної роботи системи. Сервіси зв’язані між собою, але також можуть бути реалізовані незалежно від інших сервісів системи [52]. Сервіси виконують конкретну задачу, яка може повторюватись безліч разів [41].

Переваги [53] [54]:

1. Легке підключення нових сервісів до системи завдяки використанню відкритих стандартів.

2. Зміни, внесені в один сервіс, не впливатимуть на інші сервіси.

Недоліки:

1. Через незалежність сервісів ускладнюється обмін даними між ними [40].

2. При великій кількості сервісів виникає необхідність посилювати вимоги до розробки сервісів та створенню докладної документації.

5) технології однорангових мереж (P2P). Однорангові мережі є децентралізованими, всі вузли рівноправні. Кожен з вузлів є одночасно і клієнтом і сервером [24] [25]. Апаратні пристрої, що входять до однорангової мережі, використовують спеціалізоване програмне забезпечення для обміну даними по мережі. Всі вузли в мережі є серверами, тоді якщо один з вузлів виступатиме в ролі клієнта і буде скачувати файли, у такому випадку

джерелом будуть усі доступні на даний момент вузли в мережі в яких зберігаються необхідні дані. Після скачування даний вузол також стане джерелом для скачування цих даних.

Переваги:

1. Вихід з ладу одного вузла системи не впливає на працездатність всіх інших її вузлів.

2. Висока стабільність роботи.

3. Е можливість спільного використання обчислювальних потужностей, процесора тощо.

4. Здатність обробляти великі об’єми даних за рахунок розподілення навантаження між вузлами мережі.

Недоліки:

1. Уразливі до віддалених атак та до DoS-атак. Для вирішення цієї проблеми використовують різні методи для підвищення безпеки [29] [30] [31].

2. Можливе впровадження шкідливого коду шляхом зміни файлів в мережі. Для вирішення проблеми уразливості до аналізу трафіка також існують відповідні методи [32].

3. Будь-який з вузлів, що надає необхідні сервіси, може бути відключений в будь-який момент.

6) технології Grid це технологія, яка активно розвивається. Система, що побудована по цій технології передбачає формування простору для спільного використання ресурсів між постійно змінюваними користувачами [35] [36]. Головна ідея в тому, що до системи користувач може підключитися з будь-якого пристрою (комп’ютеру, телефону тощо) і запустити задачу на виконання [37]. Ресурси для виконання цієї задачі будуть автоматично виділені системою на віддалених потужних серверах. Ці ресурси можуть фізично знаходитись у різних місцях. Для керування ресурсами в такій системі використовують стандартні, відкриті, універсальні протоколи та інтерфейси.

Переваги технології:

1. Ефективне використання наявних ресурсів та обчислювальних потужностей системи [33].

2. Можливість вирішувати складні задачі за менший проміжок часу [38] та можливість співпраці з іншими організаціями.

3. Grid система є модульною, тобто якщо один з вузлів вийде з ладу, ресурси для виконання задачі можна виділити з іншого.

4. Оновлення такої системи можна виконувати в процесі її роботи.

Недоліки технології:

1. Технологія все ще розвивається (програмне забезпечення, стандарти тощо).

2. Для ефективної роботи такої системи може знадобитися швидкісне з'єднання між ресурсами.

3. Може виникнути проблема з адмініструванням і керуванням ресурсами [34].

7) хмарні обчислення. Ідея хмарних обчислень у наданні широкому колу користувачів ресурсів з метою користуючись принципом pay-as-you-go (оплата здійснюється тільки за ті ресурси, які були використані для вирішення поставленої задачі) [44]. Ресурсами, у даному випадку, виступають і програмні додатки і апаратне забезпечення [45]. Всі ресурси такої системи віртуалізовані і надаються користувачам за допомогою абстрактних інтерфейсів.

Переваги:

1. Немає необхідності інвестувати у додаткове ліцензійне програмне забезпечення [48].

2. Немає необхідності інвестувати у апаратне забезпечення [62].

3. Можливість використовувати і оплачувати тільки той об'єм ресурсів, що потрібен для конкретної задачі.

4. Легкість масштабування [64].

Недоліки:

1. Використання хмарних систем може бути небезпечним, не гарантується захист конфіденційної інформації [46] [47] [49] [65].

2. Коли розміри такої системи досягають дуже великих розмірів проявляються складності управління ресурсами [63].

3. Не завжди легка інтеграція з існуючими програмними системами.

Отже, в даному підрозділі було розглянуто основні методи побудови розподілених систем, а також проаналізовано їх особливості, переваги та недоліки.

1.2.2 Засоби виявлення вторгнень та антивірусні системи

У більшості випадків в якості архітектури антивірусного програмного забезпечення використовується архітектура клієнт-сервера. Незалежний науково-дослідний інститут IT-безпеки з Німеччини AV-TEST провели порівняння і випробування антивірусних продуктів і надали список кращих антивірусних програм для користувачів Windows 10 на грудень 2020 [66].

Розглянемо найпопулярніші антивіруси та їх можливості [76].

Антивірус Avast розроблений Чеською компанією, що спеціалізується на кібербезпеці [67]. На сьогодні є одним з часто використовуваних антивірусів. Підтримує такі операційні системи як: Windows, MacOS, Android. Не підтримує операційні системи, розраховані на корпоративний сегмент, такі як: Solaris, FreeBSD тощо. Також не підтримує операційну систему Linux та мобільну iOS. В Avast реалізовані функції: сканування за вимогою, сканування при завантаженні, фаервол, безпека електронної пошти, анти-спам та веб-захист. Також реалізований евристичний аналіз, підтримка хмарних технологій, система виявлення вторгнень та Sandbox. Проте недоліком є відсутність система запобігання вторгнень. Антивірус більше орієнтований на використання евристичного аналізу, тому має не найкращі бази сигнатур та можливі помилкові спрацювання. А також підвищене споживання системних

ресурсів при установці максимальних налаштувань безпеки. При скануванні в реальному часі не здатен просканувати упаковані і виконувані файли.

Антивірус ESET розроблений Словацькою компанією по забезпеченню безпеки в інтернеті [68]. Підтримує такі операційні системи як: Windows, Linux, MacOS, Android. Не підтримує операційні системи корпоративного сегменту: Solaris, FreeBSD тощо та ОС iOS. Здатен проводити сканування за вимогою, сканування при завантаженні, надає захист електронної пошти, фаервол, веб-захист та систему анти-спаму. Реалізовано евристичний аналіз, системи виявлення та запобігання вторгнень. Можливі помилкові спрацювання, система запобігання вторгнень HIPS не розвивається. Має недостатній захист від програм-вимагачів (шифраторів).

VirusTotal – представляє собою хмарний антивірус, який використовує для аналізу ЗПЗ комплекс антивірусів. Легко інтегрується в програмні рішення, але має обмежений функціонал – тільки сканування завантажених файлів. Має обмеження на розмір завантаженого файлу у 650Мб [70].

BitDefender – розроблений Румунською компанією, що займається кібербезпекою [69]. Підтримує такі операційні системи як: Windows, MacOS, Android, iOS. Не підтримує операційні системи: Linux, Solaris, FreeBSD. Надає функціонал сканування за вимогою, при завантаженні. Реалізований евристичний аналіз, анти-спам, веб-захист, фаервол, підтримка хмарних технологій, системи виявлення та запобігання вторгнень [75]. Не дуже ефективний проти програм-вимагачів [77].

Антивірус Avira – розроблений Німецькою міжнародною компанією з розробки програмного забезпечення для комп'ютерної безпеки [71]. Підтримує тільки операційні системи Windows, MacOS та Android. Не реалізовано фаервол та системи виявлення та запобігання вторгнень. Надає захист електронної пошти, веб-захист, анти-спам. Має підтримку хмарних технологій та реалізовано евристичний аналіз. Надає можливість сканування за вимогою та при завантаженні.

Антивірус McAfee – розроблений американською компанією з розробки програмного забезпечення для комп'ютерної безпеки [72]. Підтримує тільки операційні системи Windows та Android. Підтримує хмарні технології. Надає захист електронної пошти, веб-захист, анти-спам. Реалізовано евристичний аналіз, системи виявлення та запобігання вторгнень. Надає можливість сканування за вимогою та при завантаженні. На мобільних пристроях надмірне споживання ресурсу батареї [78]. При скануванні у реальному часі впливає на працездатність комп’ютера [79]. Серед антивірусів має не найкращу продуктивність [80].

Антивірус Clam AntiVirus – розроблений компанією Cisco [74]. Це безкоштовне програмне забезпечення, багатоплатформний набір антивірусних програм з відкритим вихідним кодом [73]. Не підтримує мобільні операційні системи. Для операційної системи Windows: Надає можливість сканування за вимогою. Реалізовано захист електронної пошти. Не реалізовано фаервол, системи виявлення та запобігання вторгнень, анти-спам, веб-захист, сканування при завантаженні, евристичний аналіз, є підтримка хмарних технологій. Для операційної системи MacOS: Надає можливість сканування за вимогою. Реалізовано захист електронної пошти. Відсутня підтримка хмарних технологій. Не реалізовано фаервол, системи виявлення та запобігання вторгнень, анти-спам, веб-захист, сканування при завантаженні, евристичний аналіз. Для операційної системи Linux: Надає можливість сканування за вимогою. Інші функції не реалізовані. Для операційної системи Solaris: Надає можливість сканування за вимогою. Інші функції не реалізовані. Для операційної системи FreeBSD: Надає можливість сканування за вимогою. Інші функції не реалізовані.

В даному підрозділі представлено найбільш популярні антивірусні засоби, проведено їх аналіз та виведено переваги та недоліки кожного представленого антивірусного рішення.

1.3 Постановка задачі дослідження

Основуючись на цілі даної магістерської роботи – розробити розподілену систему з виявлення зловмисного програмного забезпечення з використанням баєсівської мережі, було сформовано наступні задачі:

1) розглянути особливості та методи побудови сучасних розподілених інформаційних та обчислювальних систем та антивірусних засобів та рішень;

2) провести аналіз особливостей, переваг та недоліків існуючих рішень;

3) ознайомитись з баєсівською мережею, а також особливостями її використання в антивірусних системах та системах виявлення вторгнень;

4) розробити та реалізувати розподілену обчислювальну антивірусну систему з інтеграцією баєсівської мережі для виявлення зловмисного вірусного програмного забезпечення в локальних мережах;

5) провести аналіз та дослідження ефективності розробленої антивірусної системи у поєднанні з баєсівською мережею;

6) сформувати об’єкт та мету для подальших наукових досліджень, а також оцінити ступінь виконання поставлених завдань;

1.4 Висновки

В даному розділі магістерської роботи було проведено аналіз предметної області дослідження та зроблено постановку задачі. Було проаналізовано розподілені системи, особливості побудови, переваги та недоліки. Також представлена інформація стосовно актуальності таких систем у сучасних інформаційних системах. Зроблено аналіз сучасних методів побудови розподілених систем. Описано особливості кожного методу та представлено його переваги та недоліки. Також було розібрано основні популярні в корпоративному сегменті антивірусні засоби та системи виявлення вторгнень, з аналізом кожного з представлених рішень.

2 АРХІТЕКТУРА РОЗПОДІЛЕНОЇ СИСТЕМИ

2.1 Апаратна та програмна складова розподіленої системи

Виходячи з аналізу розділу 1.1 апаратна та програмна складові є однією з важливих частин будь-якої розподіленої системи в цілому. Саме ці елементи грають досить вагому роль при вирішенні можливості встановлення та безпроблемної роботи того чи іншого програмного забезпечення та системи в цілому. Саме через це існують мінімальні та рекомендовані вимоги до апаратної та програмної частин при роботі з певним додатком чи системою в цілому.

Мінімальними вимогами є такі апаратні характеристики при яких розроблене програмне рішення зможе не тільки бути запущене а і працювати. Відповідно рекомендованими вимогами є такі характеристики – при яких розроблена антивірусна система може бути не тільки бути запущена, а і зможе стабільно та повноцінно виконувати всі закладені в неї задачі.

Зрозуміло, що ці вимоги можуть бути різні для різних компонентів системи в залежності від задач які вони виконують а також від вимог самої системи та додатково встановлених програм. Проте в цілому, враховуючи сферу використання розробленої системи, зрозуміло, що ці вимоги не мають бути надто високими, особливо враховуючи той факт, що дана розподілена система розроблюється для використання на підприємствах, де як правило комп’ютер та його складові оновлюються значно повільніше ніж у домашньому використанні.

Базуючись на аналізі розділу 1.1 та 1.2 в якості архітектури для антивірусної розподіленої системи було обрано клієнт-серверне рішення. Таке рішення було прийнято базуючись на тому, що дана архітектура досить проста в реалізації та обслуговуванні, надає гарні можливості контролю роботи елементів в працюючій мережі а також добре розширюється і просто перебудовується. Крім того в більшості компаній як правило вже наявні

серверні елементи, тому відсутня потреба у додатковому обладнанні. Проте присутній важливий недолік у зупинці роботи всієї працюючої розподіленої мережі у випадку відмови сервера. Це відбувається досить рідко, проте при розробці подібної системи слід передбачити такі ситуації. Відповідно в архітектуру системи було закладено рішення для мінімізації наслідків відмови керуючого компонента у мережі. Також, фактично при такому рішенні весь взаємозв'язок клієнтських компонент буде відбуватись не на пряму, а через серверну компоненту, що надає можливості додаткового контролю. Схему зв'язків компонентів у системі наведено на Рисунку 2.1.

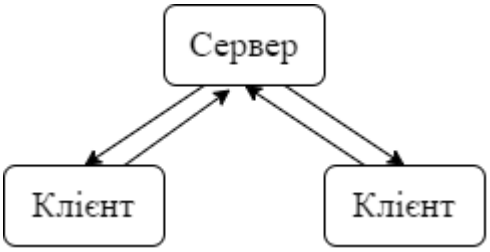


Рисунок 2.1 – Схема зв'язків компонент у системі

Також у випадку відсутності сервера у мережі компанії – центральна керуюча компонента може бути встановлена на одній із робочих машин. В такому випадку вона буде виступати в якості сервера. Проте таке рішення не рекомендується, оскільки у такому випадку не може гарантуватись постійна наявність сервера у мережі, що необхідна для оновлення вірусних баз даних. Також користувач може випадково обмежити в операційній системі доступ клієнтських компонент, що обмежити роботу.

Як вже було сказано в розділі 1.2.1 – розподілена клієнт-серверна система складається з клієнтської та мережевої частини. У даному випадку роль мережевої частини буде виконувати серверна машина у локальній мережі підприємства з її програмно-апаратними ресурсами та додатково встановленим необхідним софтом центру прийняття рішень. Основуючись на цьому серверна частина повинна мати значно більші і кращі показники потужності та продуктивності ніж робочі станції, що підключені до сервера, із

встановленими на них хостовими компонентами. Оскільки серверна частина має не тільки забезпечити одночасне з’єднання та обмін даними між багатьма компонентами самої антивірусної системи, а ще також виконувати покладені на неї задачі.

Крім того необхідно чітко встановити мінімальні вимоги, щоб уникнути ситуації коли одні комп’ютери здатні стабільно працювати із розгорнутою системою, а інші ні. Тобто розроблене програмне забезпечення повинно бути без зайвих проблем і додаткових витрат інтегровано у вже існуючу мережу підприємства з мінімально необхідними доробками та налаштуваннями і при цьому не порушуючи вже існуючій робочій процес.

Отже була приділена суттєва увага формуванню мінімальних та рекомендованих параметрів для серверної частини (центральної компоненти) та комп’ютерів із компонентами ще на етапі розробки, оскільки версії одної програми для різних операційних систем можуть мати різні вимоги, а також уникнути ситуацій, коли для побудови антивірусної розподіленої системи доведеться додатково купувати та встановлювати необхідне обладнання або програмне забезпечення, а також змінювати вже існуючу локальну мережу або спосіб підключення до неї.

Враховуючи все вище наведене а також спираючись на результати тестувань розробленої противірусної системи засобами профайлеру середовища розробки Visual Studio 2019 були встановлені наступні мінімальні вимоги до апаратної частини. Для центральної компоненти при підключенні до 1000 клієнтів потрібен серверний процесор з 4 ядрами та об'єм оперативної пам’яті 4 ГБ (сама серверна компонента при роботі використовує до 300 МБ пам’яті при навантаженні). Щодо об’єму жорсткого диску, мінімальним об’ємом виступає значення 400 МБ (Це з урахуванням самої встановленої програми, баз даних та вірусних сигнатур, а також додаткового місця під майбутні оновлення баз в процесі роботи). Відповідно рекомендованими значеннями для центральної компоненти буде 4-х ядерний серверний процесор, 8 ГБ оперативної пам’яті та 800 МБ жорсткого диску. Відповідно,

що при збільшенні кількості підключених клієнтів кількість необхідної оперативної пам’яті та потужність процесора буде так само зростати.

Щодо клієнтських компонент, тут вимоги трохи менші – оскільки немає необхідності у великій кількості одночасних з’єднань, проте значна частина навантаження йде на сканування самої системи. Відповідно при всіх сценаріях роботи об’єм мінімально необхідної оперативної пам’яті не перевищував 256 МБ. Щодо необхідного об’єму диску для самої програми, то тут мінімальне значення 300 МБ. Рекомендованим же значенням є 600 МБ. Це значення взято із запасом для майбутніх оновлень баз та даних.

Крім того обов’язковою вимогою для побудови та роботи системи є наявність підключення через кабель або бездротовим способом між комп’ютерами з компонентами, оскільки зв’язок компонентів необхідний в процесі роботи самої розподіленої системи під час оновлень та сканувань.

На наступному етапі слід визначитись з програмною складовою – цільовою операційною системою або сімейством систем. Оскільки в корпоративному секторі оновлення операційних систем та компонентів відбувається досить повільно та на деяких підприємствах все ще використовуються застарілі версії операційних систем – цей момент слід також обов’язково передбачити при розробці антивірусної системи. Крім того версії програмного забезпечення розроблені під одну операційну систему можуть бути несумісні з іншою операційною системою, навіть в рамках одного сімейства систем, що може виявитись слабким місцем у використанні такого продукту, особливо враховуючи той факт, що на підприємствах водночас нерідко можуть використовуватись різні операційні системи одного й того ж сімейства.

Отже, спираючись на аналіз найбільш вживаних операційних систем в корпоративному сегменті, що приведено у розділі 1.1, в якості основної операційної системи для роботи розробленої системи було обрано сімейство ОС Windows.

Відповідно до цього для роботи центральної компоненти розробленої системи рекомендуються такі серверні версії Windows: Windows Server 2008, Windows Server 2012, Windows Server 2019, Windows Server 2022.

Щодо клієнтської компоненти – рекомендується Windows 10. Проте наявна підтримка і більш старіших версій Windows для сумісності (7, 8 та 8.1).

Для роботи розробленого програмного антивірусного комплекса з наведеними вище операційними системами, єдиної кодової бази та стабільної роботи було використано фреймворк .NET Framework, а також мову програмування C#.

Це накладає додаткову вимогу до програмної частини сервера та клієнтських комп'ютерів – в системі має бути додатково встановлена версія .NET Framework не менше 4.7.2. Але у випадку використання останніх версій системи цей компонент вже встановлено разом з оновленнями.

Отже, підсумовуючи даний підрозділ варто зазначити, що розроблена антивірусна система є клієнт-серверною та добре інтегрується у вже наявну мережу без її змін. Центральна керуюча компонента може бути встановлена і на робочу машину при необхідності. Крім того антивірусний комплекс розроблявся з мінімальним використанням системних ресурсів комп'ютерів у мережі, а також з врахуванням можливості встановлення на більш старіші версії систем Windows на підприємстві.

2.2 Архітектура центральної керуючої компоненти

Як вже було сказано у розділах 1.1 та 2.1 – розподілена система містить у собі дві основні частини – сервер та клієнти. Серверна частина – вона ж центр прийняття рішень виступає у якості зв'язуючого елемента, що має поєднати велику кількість комп'ютерів у локальній мережі із встановленими на них клієнтськими частинами у єдину систему для вирішення спільних задач.

Центральна компонента має встановлюватись саме на серверній машині у мережі – це необхідно для того, щоб кожній клієнтській компоненті

забезпечувався необхідний до центру доступ у будь-який проміжок часу, а також оскільки серверна машина виконує керуючу функцію у мережі – це надає можливість своєчасного реагування на загрози, а також блокування трафіку або відключення певної робочої станції від мережі підприємства у разі потреби. Крім того це дозволяє також централізовано збирати інформацію та зберігати звіт щодо всіх подій, які відбуваються у системі.

Отже, оскільки центральна компонента встановлюється на мережній серверній машині – у даному випадку вона являє собою Windows-службу, що не надає користувачу графічного інтерфейсу, проте після першого налаштування ця служба автоматично запускається кожного разу при перезавантаженні операційної системи, після чого автоматично будує мережу, основуючись на збережених даних щодо активності та наявності в робочому стані її компонентів, що були записані в базу під час попередньої роботи. При цьому кожен наявний компонент наново опитується для актуалізації даних. Це необхідно для того, щоб система завжди складалась тільки з активних та існуючих компонентів, кількість яких в процесі роботи всієї системи буде постійно змінюватись – оскільки по закінченню роботи працівники компанії будуть вимикати свої робочі комп’ютери, разом з встановленими клієнтськими антивірусними компонентами.

Після того, як система була побудована – відбувається перевірка наявності нової версії сигнатурної бази та оновлення центральної бази серверної компоненти за необхідності. Механізм оновлення центральної бази може відбуватись або через інтернет (для цього потрібно вказати логін, пароль та get-ендпоінти для оновлень при конфігурації до першого включення центральної компоненти) або окремо через файлову систему, скидаючи сумісну вірусну базу від інших антивірусних систем (наприклад, ClamAV) у каталог import_base файлової системи.

Після оновлення центральної вірусної бази всі підключені до центру клієнти вже в автоматичному режимі самі оновлюють стан своїх локальних баз даних. Так само оновлення центральної вірусної бази даних відбувається і

при подальшій періодичній перевірці оновлень, що для зменшення навантажень на саму систему відбувається раз на добу. Крім того при надходженні нової вірусної сигнатури від клієнта, вона записується в центральну вірусну базу зі значенням «host» у полі source, а також відбувається зміна номера версії бази. Саме значення «host» - означає, що сигнатура була знайдена на клієнті засобами евристичного аналізу та попередньо вважається як небезпечна. У майбутньому при подальших оновленнях це значення для конкретної сигнатури в базі може бути змінене на «null», якщо така ж сама сигнатура надійде в офіційному оновленні (тобто достовірно буде встановлено, що ця сигнатура вважається небезпечною).

Відповідно для зручності роботи з базою версія оновлень зберігається у наступній формі: XXXX-YYY, де XXXX – це версія самої антивірусної бази, а YYY – кількість знайдених хостами сигнатур між оновленнями бази з офіційних джерел, при оновленнях цей номер скидається.

По суті мережева частина є автономною та незалежною від втручання користувачів або адміністраторів. Вона здатна сама приймати чіткі рішення та керувати роботою всієї мережі на основі заданих конфігураційних правил при першому налаштуванні перед запуском. Саме тому у мережеву компоненту було закладено наступні задачі, які вона виконує в процесі своєї роботи у побудованій мережі:

1) зберігання та оновлення інформації про стани всіх компонентів, зареєстрованих у мережі;

2) початкова конфігурація та реєстрація клієнта при першому підключенні до центру;

3) підтримка актуальності бази сигнатур та надання єдиної спільної версії всім клієнтам;

4) реєстрація нової сигнатури, що надійшла від хоста в процесі сканування;

5) зміна архітектури мережі при необхідності, а також у випадку зміни станів вузлів;

6) логування основної інформації під час роботи антивірусної мережі;

Варто зазначити, що для більш ефективного виконання наведених вище задач, а також для можливості безпроблемного та простого оновлення програмної логіки у майбутньому – серверна компонента складається з таких основних програмних частин (модулів) (Рисунок 2.2):

1) модуль керування мережею (ManageWebComponent) – цей модуль реєструє та оновлює дані підключених хостових компонентів в процесі роботи, змінює саму структуру мережі при необхідності а також приймає рішення на основі заданих до включення конфігураційних правил про відключення від мережі якогось із компонентів у випадку загроз або його компрометації. Крім того цей модуль надсилає базові налаштування при підключенні нового компоненту до центру;

2) модуль оновлень (UpdatesComponent) – цей модуль відповідає за стан бази даних вірусних сигнатур. Він використовується для перевірки та оновлення вірусної бази, а також додавання у серверну сигнатурну базу нових даних, що надійшли від хоста в процесі його роботи. Також цей модуль надає оновлену базу хостам при запиті на оновлення;

3) модуль логування (LoggerComponent) – ця програмна частина відповідає за логування всіх основних етапів роботи мережі: стан мережі, підключення клієнтів, оновлення бази, результати сканувань хостів, види та сигнатури знайдених загроз в режимі real-time в процесі роботи;

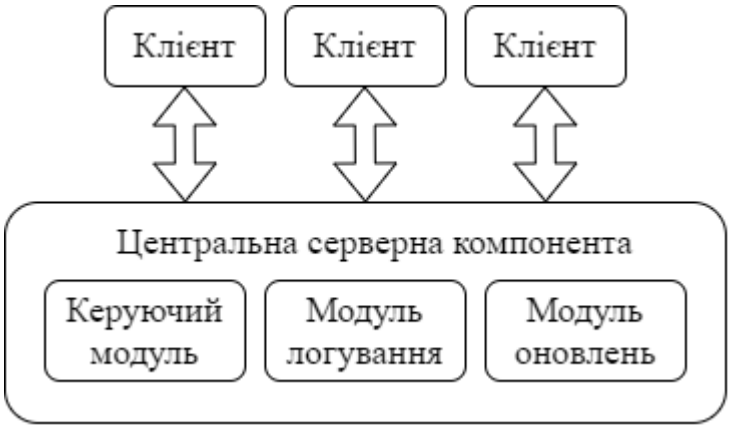


Рисунок 2.2 – Складові модулі центру прийняття рішень

Приведені модулі реалізовано у вигляді окремих бібліотек, які за потреби можна оновити або замінити. Кожен модуль необхідний для реалізації покладеної на нього задачі та в цілому взаємодіє з іншими модулями через програмні інтерфейси-методи. Так, наприклад, при надходженні нової сигнатури та звіту про сканування з хоста до центру – дані потрапляють спочатку до керуючого модуля через який реалізується подальша взаємодія – сигнатура опрацьовується модулем оновлень, звіт про сканування та знайдену загрозу потрапляє до логу через логуючий модуль. Наступним кроком після запису до бази, редагування номеру версій та формування мікро-оновлення з сигнатурою модулем оновлень, керуючий модуль розсилає оновлення до всіх підключених клієнтів.

Як було сказано вище в процесі сканувань та роботи всієї антивірусної мережі створюються логи, що централізуються та оброблюються модулем логування. Цей модуль створює функціонал частково схожий на систему «Журнал аудити» у антивірусній системі ESET Security Management Center. Проте на відміну від нього він не надає графічного інтерфейсу, і використовує значно набагато менше системних ресурсів та записує більш розширену інформацію про події в усій системі у зручному для розуміння текстовому представленні, яке за необхідності можна швидко обробити та візуалізувати сторонніми системами. Також за необхідності записані логи можна відкрити засобами будь-якої операційної системи (Блокнот та йому подібні), це дає значні переваги над системою моніторингу від ESET.

Отже, враховуючи вищенаведений опис архітектури центральної компоненти у даному підрозділі, слід підсумувати, що дана компонента виконує основні керуючі функції у побудованій антивірусній мережі та сполучає між собою усі клієнтські частини, утворюючи єдину антивірусну систему.

2.3 Архітектура клієнтських компонентів

Як вже було сказано у підрозділах 1.1 та 2.1 – клієнтська компонента – це друга основна складова розподіленої системи, а крім того це основна частина по виявленню загроз у системі. Саме засобами цієї компоненти виконується сканування клієнтської системи та виявлення нових можливих сигнатур, що є основою захисту.

Сама по собі окрема хостова компонента є незалежною, тобто здатна працювати та виконувати покладені на неї задачі самостійно без підключення додаткових елементів, при наявності локальної бази сигнатур. Проте наявність центру у поєднанні з такими ж самими хостовими компонентами розкриває увесь закладений у неї потенціал (про сумісне використання компонент в рамках єдиної системи подано інформацію у наступному підрозділі).

На відміну від серверної компоненти – клієнтська має зручний для користувача графічний інтерфейс, що розроблено з використанням WPF технології фреймворку .NET Framework. Завдяки цьому клієнтська компонента більш ефективніше співпрацює з користувачем та надає більш розширений функціонал. Користувач може не тільки встановити заплановану дату й час повного сканування системи, а й провести за потреби вибіркове сканування окремих файлів чи папок. Крім того результат сканування надається у зручній для користувача формі з графіком знайдених загроз за останні 10 днів та з детальним поясненням по кожній виявленій проблемі. Користувач може заблокувати ту чи іншу загрозу або видалити її. Відповідно результат сканування автоматично відправляється до центру для подальшого логування та опрацювання.

Відповідно основними задачами, що закладені у хостову компоненту є:

1) повне сканування системи за розкладом;

2) вибіркове сканування папок чи файлів;

3) надання результатів сканування користувачу з вибором дій у випадку виявлення загрози;

4) відправка результатів сканування до центру;

5) запис знайденої загрози в локальну базу;

6) підтримка в актуальному стані бази з вірусними сигнатурами;

Щодо самої хостової компоненти, то основний робочій процес відбувається наступним чином: при першому запуску компоненти користувачу або адміністратору потрібно ввести логін, пароль та адресу вузла з центральною керуючою компонентою. Після цього відбудеться з'єднання з сервером, отримання базових налаштувань та вірусних сигнатур, що будуть записані в локальну базу хоста. Після чого буде доступний основний функціонал сканування. При скануванні кожна нова знайдена загроза записується в локальну базу та надсилається до центру. Перевірка наявності оновлень та оновлення бази відбувається автоматично кожного дня без втручання користувача.

Користувач в налаштуваннях програми може задати режим сканування: ручний за вимогою або автоматичний за розкладом. У випадку автоматичного сканування клієнтська програма автоматично запускається кожного разу при старті системи, підключається до заданого раніше сервера після цього переходить в режим очікування до настання часу сканування. Незалежно від задання автоматичного режиму сканування, користувач завжди додатково у будь-який час може провести вибіркове ручне сканування. У випадку наявності сканування за розкладом перед виходом з програми, користувач отримає попередження про наявність запланованого сканування та необхідність запущеної на комп'ютері програми. Також додатково для зручності користувача наявний функціонал для згортання програми у системний трей Windows. Це дозволяє не засмічувати панель задач Windows та сприяє більш комфортній роботі з розробленою програмою.

Під час сканування аналіз вірусних сигнатур відбувається за двома методами: сигнатурним та евристичним. При сигнатурному методі відбувається перевірка на співпадіння з базою вже існуючих вірусів. Цей метод ефективний у випадку коли вірусна загроза вже добре відома та була

раніше виявлена і додана у антивірусні бази. Проте у випадку якщо загроза, ще не виявлена – цей метод не ефективний. Саме для цього додатково у випадку не співпадіння з вірусною локальною базою відбувається сканування за евристичним методом при якому відбувається пошук вірусів, схожих на вже відомі з використанням баєсовської мережі. Це поєднання дозволяє ефективно та своєчасно виявляти існуючу вірусну загрозу, та оперативно реагувати на неї. Проте недоліком такої системи є те, що при відсутності бази даних або при малій кількості сигнатур – сканування не дасть результату. Але цей недолік усувається архітектурою реалізованої антивірусної розподіленої системи – при завантаженні хостової компоненти та підключенні до сервера відбувається оновлення або створення локальної вірусної бази. В противному випадку, при відсутності бази даних і неможливості її отримання з серверної керуючої компонети – користувач отримає повідомлення про відсутність бази та неможливість подальшого сканування.

Оскільки розроблена розподілена система є слабозв'язною, то при відмові центральної компоненти – клієнтська компонента продовжує незалежно працювати. Сканування у цьому випадку будуть відбуватись спираючись на локальну робочу базу, що знаходилась на клієнті до відмови серверної компоненти. При подальшому відновленні роботи серверної компоненти бази обох компонент будуть автоматично синхронізовані та оновлені.

Хостова клієнтська компонента як і серверна складається з декількох окремих модулів-бібліотек, кожен з яких надає певний функціонал (Рисунок 2.3):

1) модуль сканування (DetectComponent) – це основний модуль, що використовується для сканування. Він реалізую логіку евристичного та сигнатурного методів, а також логіку для запланованого за часом сканування;

2) модуль роботи з вірусною базою (VirusDBComponent) – цей модуль реалізує логіку роботи з базою, включаючи оновлення бази і реєстрацію нової загрози;

3) модуль користувацького інтерфейсу (UIComponent) – цей модуль містить всі елементи та логіку, що необхідна для відображення інтерфейсу користувача;

4) модуль взаємодії з мережею (WebComponent) – у цьому модулі міститься основна логіка для взаємодіє з мережею та центром;

5) модуль логування (LoggerComponent) – цей модуль відповідає за логування всіх етапів роботи, а також створення звіту з результатами сканувань для відправки до центру;

Рисунок 2.3 – Складові модулі клієнтської компоненти

Кожен з представлених модулів є незалежним та скриває свою реалізацію логіки від інших компонентів, але як і у випадку з серверною компонентою, всі модулі взаємодіють через окремі інтерфейси. Всі модулі при подальшому можуть бути замінено на аналогічні з єдиною вимогою, щоб співпадали інтерфейси для взаємодії. Це означає, що бібліотеку можна оновити без відключення мережі та заміни всіх її компонентів. Наприклад, модуль-бібліотека DetectComponent може бути у подальшому замінений на більш нову версію з більш розширеними можливостями, проте для роботи ця нова версія має містити у собі необхідні базові методи: Scan() та DelayedScan(). Зрозуміло, що ці методи можуть бути розширені. Опис всіх необхідних для роботи базових методів, що складають інтерфейси взаємодії подано у додатку. Також це дозволяє додавати нові бібліотеки, що реалізують інші методи виявлення загроз, що робить розроблену систему універсальною та не потребує значних змін програмного коду.

Проте не зважаючи на закладену гнучкість архітектури клієнтської компоненти та закладену можливість апгрейду та покращення базових функцій, деякі зміни наявних бібліотек можуть призвести до необхідності перезапуску та перебудови всіх компонентів розподіленої антивірусної системи. Особливо, якщо відбудеться зміна організації мережі, наприклад децентралізація. У такому випадку потрібно вносити зміни не лише для однієї клієнтської компоненти, а для всіх наявних. Оскільки для повноцінної роботи необхідні зміни як на відправляючій стороні, так і на приймаючій.

Отже, враховуючи вище наведене можна зробити висновок, що хостова частина виконує основну задачу розробленої антивірусної системи – а саме сканування і виявлення вірусів у мережі. Саме об’єднання багатьох таких однакових встановлених у системі компонент дозволяє швидше виявити будь-яку загрозу, а враховуючи комбінацію з евристичного та сигнатурного сканувань – хостова компонента здатна виявити набагато більшу кількість загроз.

2.4 Взаємодія компонентів

Як вже було сказано у попередніх підрозділах – побудована розподілена антивірусна система має клієнт-серверну архітектуру та складається з керуючої мережної частини, що розташовується на сервері у мережі та клієнтських частин, які встановлюються на комп'ютерах працівників підприємства.

Серверна частина відіграє керуючу роль та об'єднує розрізнені клієнтські компоненти у одну систему. При цьому всі клієнтські компоненти однакові та виконують одні й ті ж самі функції – сканування робочих машин та виявлення нових загроз. Проте саме поєднання масиву клієнтських компонент у єдину мережу робить сканування більш ефективним. Оскільки в цьому випадку хостові частини не тільки працюють кожна на себе, а й кожна для всіх. Так у випадку виявлення нової загрози хостова частина одразу повідомляє про неї у центр, який створює новий запис у центральній базі даних і змінює її версію, а також повідомляє інші підключені хости про наявність нової сигнатури загрози і розсилає її кожному з них. Відповідно після отримання оновлень для локальних баз при подальших скануваннях всі інші хости вже будуть витрачати значно менше часу знаходячи цю сигнатуру в файловій системі – оскільки в цьому випадку для сканування достатньо буде тільки сигнатурного аналізу (так як дана сигнатура вже є у їхній базі) і відповідно евристичним аналізом і затраченим на нього часом у цьому випадку вже можна знехтувати.

Також при першому налаштуванні хостової компоненти перед запуском у якості сервера можна додати декілька елементів. У цьому випадку одна компонента буде одночасно працювати у двох розподілених мережах, відповідно у такому випадку обмін даними бази та виявленими сигнатурами буде відбуватись на дві мережі і одна хостова компонента буде містити у собі одну спільну вірусну базу.

Міжкомпонентна мережева взаємодія у розробленій системі базується на технології WCF (Windows Communication Foundation) фреймворку .NET Framework. Ця технологія набула досить широкого використання у корпоративній сфері та дозволяє будувати сервіс-орієнтовані платформо-незалежні розподілені додатки, абстрагуючись від прив'язки до певної технології. Це дозволяє використовувати у сукупній розробленій системі різні платформи з різними стандартами, а також різні мови, з єдиною умовою, щоб розроблені частини мали спільні технології для взаємодії. Це означає, що у подальшому можна не тільки переписати або модифікувати серверну або клієнтську компоненти з використанням WCF технології, а і реалізувати компоненту на основі іншої технології, мови та платформи без суттєвих змін всієї мережі. Проте з єдиною умовою – спільні інтерфейси та контракти для взаємодії мають співпадати в обох частин для успішної комунікації між одиницями.

У розробленій розподіленій системі серверна компонента працює у режимі сінглтона, тобто єдиного екземпляру для всіх запитів клієнтських компонент. Відповідно цей екземпляр створюється одноразово при запуску служби Windows та піднятті сервера центральної компоненти, до якого вже в процесі роботи підключаються клієнтські компоненти.

Обмін даними між серверною компонентою та клієнтською відбувається на рівні повідомлень та базується на контрактах – наборах певних інтерфейсів, що містять опис всіх типів, операцій та повідомлень, що дозволені між клієнтом та сервером. У якості транспортного каналу WCF використовується протокол TCP з сервісною прив'язкою (спосіб взаємодії WCF сервісу з клієнтом) NetTcpBinding, що підтримує дуплексний (повноцінний двосторонній) зв'язок компонентів. Такий тип прив'язки вважається найбільш надійним та захищеним, оскільки відбувається додаткове двійкове кодування повідомлень. Також за рахунок створення дуплексного двостороннього зв'язку, який побудовано на двосторонньому контракті, клієнт та сервер можуть взаємодіяти один з одним незалежно. Саме це надає можливість

надсилати не тільки повідомлення від клієнта до сервера, а і серверу ініціювати відправку повідомлень клієнту або клієнтам за потреби. Такий механізм в розробленій системі використовується у випадку надходження з клієнта нової сигнатури – після реєстрації її на сервері – сервер розсилає виявлену сигнатуру всім іншим клієнтам. Крім того цей механізм використовується для періодичного опитування підключених хостів з метою оновлення бази наявних у системі компонентів. Такі періодичні опитування відбуваються кожні пів години.

Як вже було сказано в розділах 2.2 та 2.3 кожна компонента має свою робочу вірусну базу даних. При цьому на сервері зберігається основна версія бази, що містить вірусні сигнатури, додані з офіційних джерел, а також знайдені під час сканувань компонентами антивірусної системи. Додатково також до кожної версії йде опис змін у базі. Він необхідний для того, щоб при оновленнях клієнтських баз надсилалась не повністю вся база, а різниця – тобто відсутні дані. Крім центральної бази, кожен клієнт для зменшення навантажень на мережу та сервер містить свою локальну робочу копію серверної бази. Вона містить ті ж самі дані, що і серверна, проте відсутній опис версії.

Для оновлень локальної вірусної бази клієнтів розроблено два основних механізма оновлень: щоденне оновлення та мікро оновлення.

При щоденних оновленнях клієнти починаючи з 12 години дня роблять запит на наявність нової версії бази та у випадку наявності – отримують нову версію від сервера у вигляді відсутніх даних і дописують її у свою локальну робочу базу. Час оновлень було обрано на основі статистики робочого навантаження на локальні мережі – оскільки у цей час робоче навантаження на мережу у більшості компаній мінімальне. Крім того перевірка наявних оновлень та саме оновлення бази відбувається по черзі, щоб не перевантажувати мережу одночасними запитами у разі великої кількості клієнтів.

Щодо мікро оновлень – цей механізм спрацьовує у випадку, коли сервер отримує нову вірусну сигнатуру від хоста, що не міститься в його базі. В такому випадку він додає таку сигнатуру у центральну версію бази, помічає її як знайдену на хості та надсилає кожному підключеному клієнту. Клієнт при отриманні сигнатури додає її в свою локальну вірусну базу для подальшої роботи.

Варто також окремо виділити оновлення бази при створенні нового вузла мережі, тобто при включенні нової клієнтської компоненти та підключенні її до центру. У цьому випадку як і при щоденному оновленні відбувається порівняння версії локальної робочої бази з центральною. Якщо локальна вірусна база відсутня – копія центральної бази завантажується на клієнтську сторону повністю наново, якщо база наявна, проте відрізняються самі версії – робиться оновлення бази через завантаження відсутніх даних, орієнтуючись по опису нової версії, що міститься у центрі.

Крім того на відміну від класичних клієнт-серверних систем, розроблена антивірусна розподілена система не припиняє свою роботу повністю у випадку відмови сервера або відключенні центральної компоненти в процесі роботи, а також при відключенні інтернету. Кожен клієнтський вузол у такому випадку стає незалежним та працює самостійно сам на себе. Тобто працює тільки зі своєю локальною вірусною базою, без використання мережі та без механізма оновлень і надсилання нових виявлених сигнатур до центру. Також не надсилаються звіти з результатами проведених сканувань. Проте знайдені загрози і звіти кешуються та накопичуються на клієнтських машинах. Як тільки серверна частина відновить свою роботу – відбудеться нове підключення клієнтських вузлів до сервера. Відповідно після цього відбудеться надсилання клієнтами нових сигнатур, оновлення бази та відправка звітів сканувань до центру. Такий механізм дозволяє при відмові сервера не тільки продовжувати сканування на вузлах а і не втратити результати при відновленні роботи.

Отже, підсумовуючи даний підрозділ варто зазначити, що незважаючи на той факт, що хоч компоненти розробленої мережі і є незалежними, максимальна продуктивність та найкращій результат у виявленні загроз досягається саме сумісною роботою великої кількості клієнтських компонент у мережі під керуванням з боку серверної компоненти. Тому враховуючи необхідність такої взаємодії та з метою досягнення ефективного обміну інформацією в розподіленій мережі було закладено можливості двостороннього зв’язку клієнтської та серверної компонент, оптимізовано механізми оновлення баз даних та передбачено алгоритм роботи у випадку відмови сервера та втрати зв’язку .

2.5 Висновки

В даному розділі дипломної роботи було розглянуто вимоги до апаратної складової та програмної складової комп’ютерів для побудови розробленої розподіленої антивірусної системи а також архітектуру її компонентів: сервера та клієнтів. Розробка даної системи планувалась з урахуванням можливості встановлення на старіші версії Windows, які все ще використовуються на підприємствах, а також з мінімально-необхідним використанням ресурсів, без зміни існуючого робочого процесу та апаратної частини. В реалізованій системі серверна (центральна) компонента виконує дві основних функції: поєднує комп’ютери в мережі з клієнтськими компонентами в єдину систему та керує цією системою. Клієнтська компонента виконує в мережі основні функції з виявлення загроз про які повідомляє серверу. З’єднання великої кількості клієнтських компонент робить роботу більш ефективною та продуктивною, оскільки в своїй роботі кожна клієнтська компонента при скануванні базується на результатах попередніх сканувань всіх компонент. Для забезпечення своєчасного обміну інформацією в мережі використовується двосторонній зв’язок клієнту з сервером. Саме сервер у випадку виявлення нової загрози за рахунок двостороннього зв’язку негайно повідомляє про неї всіх клієнтів. Також оптимізовано механізми оновлень та реалізовано механізм кешування на випадок відмови сервера або зникнення інтернету

3 БАЄСОВСЬКА МЕРЕЖА У РОЗПОДІЛЕНІЙ СИСТЕМІ

3.1 Баєсовська мережа

Баєсовська мережа (Баєсовська мережа довіри) відноситься до категорії ймовірнісних графічних моделей і представляє собою безліч змінних і їх імовірнісні залежності, що обчислюються по теоремі Баєса. Дана графічна модель є графовою і є орієнтованим ациклічним графом, кожна вершина якого - це випадкова змінна, а дуги (ребра) представляють умовні залежності між цими змінними.

Орієнтований ациклічний граф – це орієнтований граф без спрямованих циклів, тобто, він складається з вершин і ребер (дуг), при цьому кожне ребро йде від однієї вершини до іншої таким чином, щоб не утворювалися замкнуті цикли. З цього випливає, що, почавши шлях з однієї точки буде пройдена не вся схема графа, а тільки її частина. При цьому виключено можливість попасти в одну й ту ж саму вершину більше одного разу.

Кожен вузол графа пов'язаний з функцією ймовірності, яка в якості вхідних даних приймає певний набір значень для батьківських змінних вузла і в якості вихідних даних дає ймовірність змінної, яка представлена вузлом. Функцію ймовірності можна представити у вигляді таблиці, в якій будуть враховуватися всі можливі комбінації і розрахувати для кожної з вершин.

Баєсовські мережі базуються на теорії ймовірності, а також на теоремі Баєса.

Теорема Баєса (формула Баєса) – є одною з основних теорем теорії ймовірностей, і дозволяє визначити ймовірність деякої події за умови, що відбулася інша подія, пов'язана з нею.

Формула Баєса має наступний вигляд представлення:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad (3.1)$$

де

P(A) – це апріорна ймовірність гіпотези A;

P(A|B) – це ймовірність гіпотези A при настанні події B (апостеріорна ймовірність);

P(B|A) – це ймовірність настання події B при істинності гіпотези A;

P(B) – це повна ймовірність настання події B.

В основі теореми Баєса лежить умовна ймовірність.

Умовна ймовірність деякої A події — це чисельне представлення ймовірності того, що подія A відбудеться за умови, що деяка подія B вже відбулась. Можлива також спільна ймовірність – це ймовірність того, що відбудеться і перша, і друга події.

Поняття d-розділеності в мережах Баєса (поняття незалежності).

Змінні A і B вважаються d-розділеними, якщо на будь-якому з шляхів, що їх з'єднують (на графі) є така проміжна змінна C, що

1) з'єднання зі змінною C є послідовним або розбіжним і відомо її значення;

2) з'єднання зі змінною C є збіжним і невідомо її значення і значення її нащадків;

Побудову мереж Баєса можна розділити на два основних кроки.

На першому кроці необхідно встановити структуру мережі. Це означає що необхідно визначити множину змінних, що відносяться до досліджуваної області та множину станів для цих змінних. Після цього необхідно встановити зв’язок між змінними та їх відношення;

На другому кроці побудови необхідно визначитись з параметрами, що встановлюють вагу для відношень. Знаючи про те, що в мережі Баєса існують

два типи змінних, а саме дискретні та неперервні, тоді для дискретних змінних ймовірність визначається за допомогою таблиць ймовірностей. Для кожної дискретної змінної (вузла) існує своя таблиця ймовірностей, що включає в себе всі можливі стани для даної змінної. У випадку неперервних змінних визначаються статистичний розподіл і необхідні параметри.

Для визначення структури мережі необхідно визначити ймовірності та встановити причинно-наслідкові зв’язки між вузлами. Визначити параметри для мережі можуть експерти предметної області, такі параметри можуть бути уточнені за допомогою спеціальних методів аналізу чутливості.

Використовуючи такий підхід, експерт має обрати змінну, що буде досліджуватись, та слідкувати за змінами, що відбуватимуться з нею під час семантичної зміни інших параметрів.

В БС відповідно існують три основні види зв’язку між вузлами:

1) послідовний зв’язок – такий тип зв’язку відповідає причинно-наслідковому зв’язку елементів. Цей зв’язок показує , що крайні змінні є умовно незалежні за умови середньої (Рисунок 3.1);

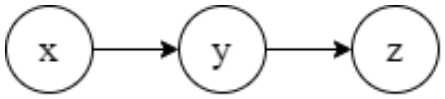


Рисунок 3.1 – Послідовний зв’язок вузлів

2) розбіжний зв’язок – відповідає двом наслідкам з однієї і тієї ж причини. Розбіжний зв’язок демонструє, що наслідки є умовно незалежними за умови загальної причини. Якщо причина відома – наслідки будуть незалежні (Рисунок 3.2);

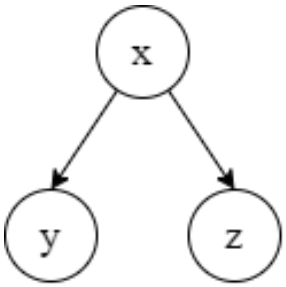


Рисунок 3.2 – Розбіжний зв’язок вузлів

3) збіжний зв’язок – при такому типі зв’язку і перша подія x, і друга подія y разом впливають на подію z. При цьому типі зв’язку обидві причини незалежні, проте тільки до того часу поки значення спільного результату невідомо. Якщо спільний результат отримує значення – причини стають залежними (Рисунок 3.3);

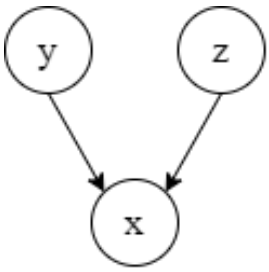


Рисунок 3.3 – Збіжний зв’язок вузлів

Представлення інформації за допомогою графа має ряд переваг для здійснення аналізу даних. Оскільки дана графічна модель оперує залежностями, вона здатна працювати навіть з ситуаціями, де інформація відсутня. Саме тому баєсовські мережі знаходять своє застосування для встановлення причинно-наслідкових зв'язків, наприклад, прогнозування або для отримання розуміння про проблемну область.

Побудовані моделі з використанням баєсовської мережі здатні до їхнього навчання на обробленій інформації, що покращує їхні результати роботи. За рахунок цього ці моделі здатні працювати з неповними або частково помилковими даними.

На сьогоднішній день баєсовська мережа набула досить активного розповсюдження. Баєсовські мережі активно використовуються у багатьох діагностичних, пошукових задачах, а також при моделюванні.

Наприклад, у медицині використовуються системи виявлення та розвитку хвороби за наведеними симптомами у пацієнта, прогнозування епідеміологічної ситуації у країні чи світі або обрахування необхідних препаратів та їхніх доз.

Також вони активно використовуються і у біоінформатиці, генній інженерії та соціології. Вони набули поширення і у автоматизованих системах прогнозування та прийняття складних рішень. Наприклад, при автоматизації керуючих процесів на електростанціях або у системах попередження надзвичайних природних ситуацій чи катаклізмів.

В інформаційних технологіях баєсівські мережі дуже активно використовуються при класифікації та пошуку інформації, обробці медіа даних, машинному навчанні та розпізнаванні інформації.

Варто також окремо виділити, використання цих мереж і в сфері захисту інформації – так майже всі системи виявлення спаму за поведінкою користувача або на основі контексту в сучасних антивірусах використовують саме баєсівські мережі.

Отже, у даному підрозділі було розглянуто що являє собою баєсівська мережа, процес її побудови, а також актуальність та сучасне використання. Також представлено зв’язок з предметною областю.

3.2 Інтеграція баєсовської мережі в існуючу систему

Як вже було сказано у попередньому підрозділі – баєсовська мережа набула досить активного використання у різних сферах діяльності, де відбувається аналіз інформації та прогнозування. IT сфера і інформаційна безпека та захист так само не є виключенням. В даному підрозділі описано інтеграцію баєсівської мережі в розроблене програмне забезпечення по виявленню вірусної загрози та її використання.

За допомогою баєсовської мережі змоделюємо ймовірність того, чи є знайдена сигнатура небезпечною, для цього потрібно провести оцінку виявленої сигнатури. Оцінку сигнатури N представимо у відсотках. Припустимо, що сигнатура N є небезпечною, якщо її оцінка складатиме більше ніж 50% і, відповідно, безпечною, якщо менше 50%. Оцінка сигнатури N зображена на рисунку 3.4

Оцінка складається з:

1. Сигнатурного сканування.
2. Евристичного аналізу.

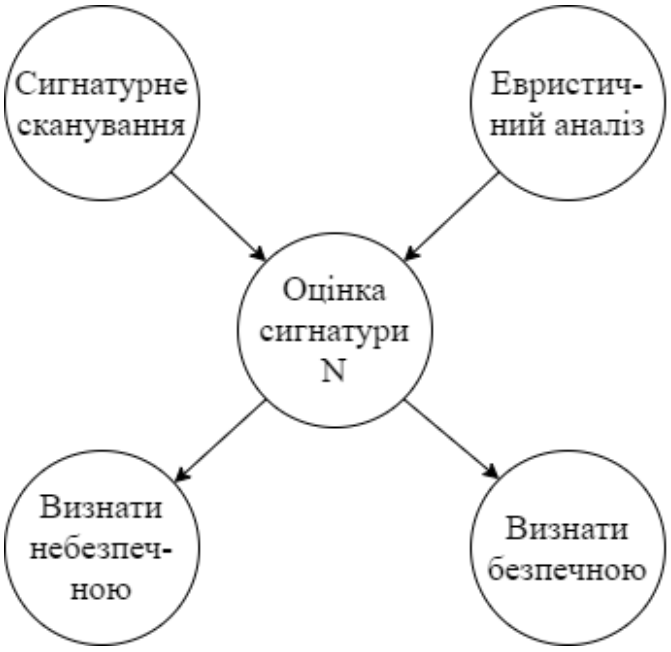


Рисунок 3.4 – Оцінка сигнатури N

Маючи на увазі те, що всі компоненти системи незалежні, мають власні бази сигнатур, тому сигнатурне сканування, евристичне сканування тощо, реалізовано для кожної компоненти окремо. Тобто сканування проводиться в рамках однієї компоненти ніяк не впливаючи на сканування інших компонент.

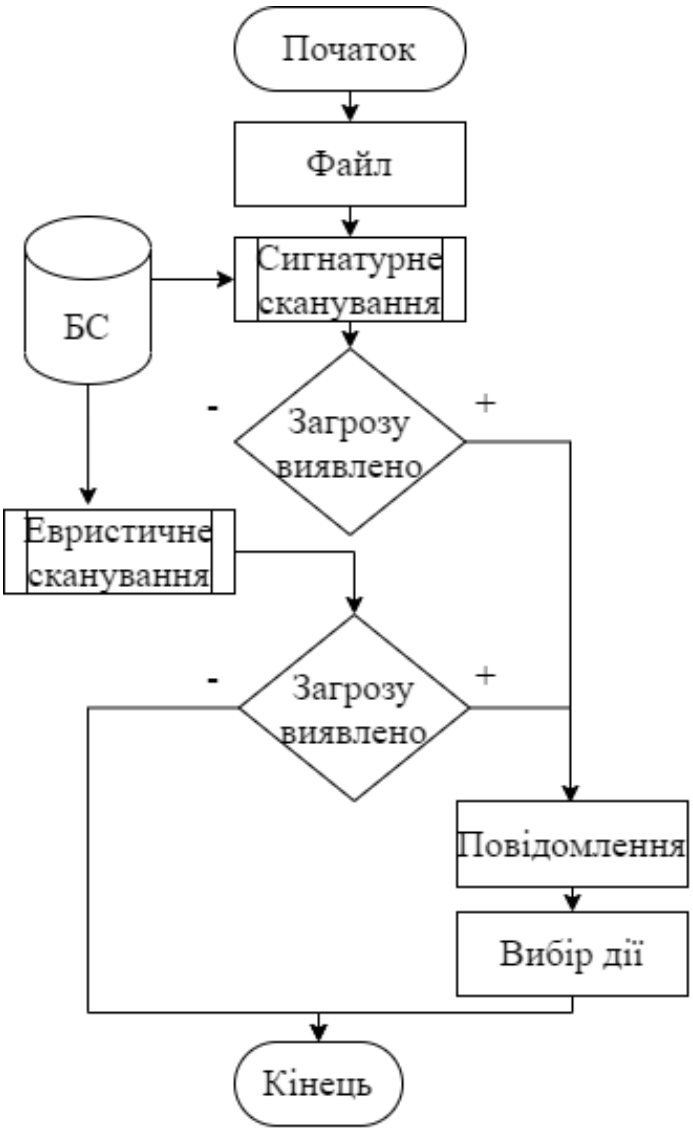


Рисунок 3.5 – Алгоритм роботи сканування з використанням Баєсівської мережі

Розглянемо алгоритм роботи сканування на прикладі однієї клієнтської компоненти. В клієнтській системі обирається файл для сканування. В першу чергу проводиться сигнатурне сканування з використанням бази сигнатур даної клієнтської машини. Якщо сигнатурним скануванням вірусних загроз не було знайдено, тоді цей файл перевіряється евристичним скануванням з використанням клієнтської бази сигнатур. При виявленні вірусної загрози будь яким з методів користувач отримує повідомлення про наявність загрози та має вибрати дію для знешкодження загрози. Отриманні результати сканування, просканована компонента відправляє на сервер, який, в свою чергу, оновлює бази даних всіх інших компонент.

Виявлення, яке ґрунтується на сигнатурах, передбачає, що модуль системи виявлення вірусів переглядає файли з метою виявлення сигнатур вірусів, використовуючи для цього базу з вже відомих сигнатур. Такі системи не здатні виявити нові або модифіковані віруси. Тому при сигнатурному скануванні сигнатура N має містити унікальні рядки, характерні конкретному вірусу і вірус вже має бути детектовано раніше. Для вирішення цієї проблеми застосовується евристичний аналіз знайденої сигнатури. При такому аналізі існують два основних методи виявлення загрози. Перший – це запуск файлу, що сканується на виконання у спеціальному середовищі (емуляція роботи вірусу) , після підраховується контрольна сума і порівнюється з контрольною сумою в базі. Та другий метод – пошук сигнатур вірусів, що подібні до вже відомих та знайдених раніше.

Для початку роботи на початковому етапі потрібно встановити умовні ймовірності для батьківських вершин сигнатурного сканування та евристичного аналізу. Відповідно до цього розрахуємо таблиці умовних ймовірностей для кожного з вузлів.

Таблиця ймовірностей для вузла p(a) представлена у таблиці 3.1

Таблиця 3.1 – Таблиця ймовірностей p(a)

a=0	a=1
0.2	0.8

Таблиця ймовірностей для вузла p(b) представлена у таблиці 3.2.

Таблиця 3.2 – Таблиця ймовірностей p(b)

b =0	b =1
0.3	0.7

При наявності усіх складових оцінки складемо таблицю ймовірностей для оцінки сигнатури N. Таблиця ймовірностей для p(c|a,b) представлена у таблиці 3.

Таблиця 3.3 – Таблиця ймовірностей p(c|a,b)

	c=0	c=1
a=0, b=0	0.6	0.4
a=0, b=1	0.9	0.1
a=1, b=0	0.5	0.5
a=1, b=1	0.8	0.2

Відповідно баєсовська мережа для оцінки сигнатури N графічно відображена на рисунку 3.6.

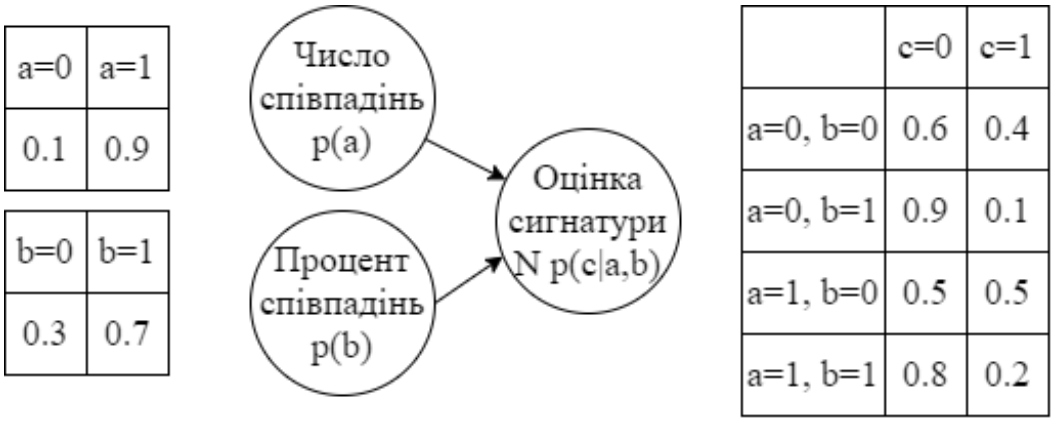


Рисунок 3.6 – Баєсовська мережа для оцінки сигнатури N

Розглянемо детальніше алгоритм роботи евристичного сканування з використанням мереж Баєса. Припустимо що в базі сигнатур клієнтської машини є деякі сигнатури (Таблиця 3.4) і при скануванні файлу евристичним методом було зафіксовано таку сигнатуру:

61 43 9A 7B | 00 0B 11 02

Таблиця 3.4 – Сигнатури вірусів клієнтської бази

№	Сигнатури							
1	69	6D	65	4D	65	74	61	3E
2	54	2C	34	1N	92	0A	00	01
3	66	43	7B	9S	00	0B	23	02

Евристичний метод побітово порівнюватиме сигнатури з бази зі знайденою сигнатурою.

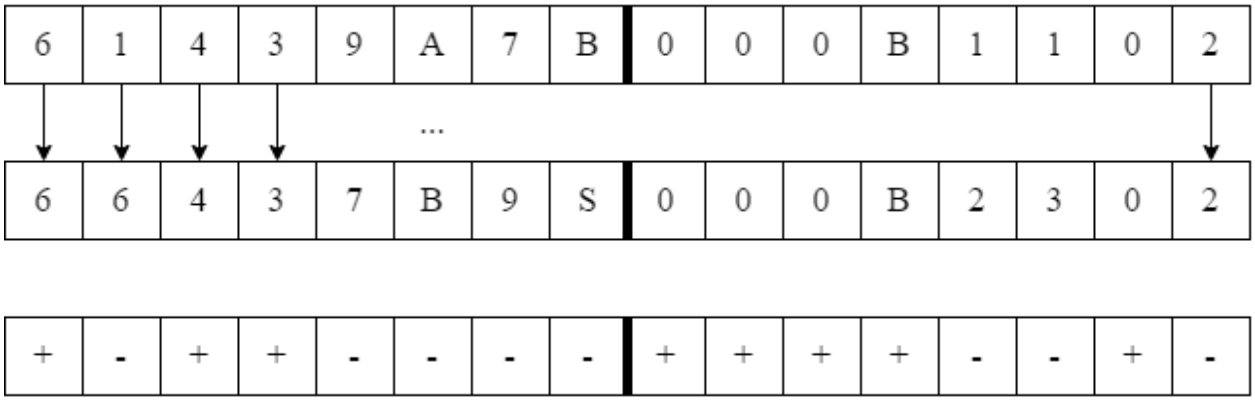


Рисунок 3.7 – Порівняння сигнатур

З 16 біт співпадає 9. У процентному співвідношенні ще перевищує 50%, тому дана сигнатура визнається вірусною.

Наступний етап це залучення баєсовської мережі. Формуємо вибірку вхідних даних. Маємо кількість співпадінь (вузол p(a), A) знайденої сигнатури з сигнатурами з бази та самі сигнатури, а також процент співпадіння сигнатур (вузол p(b), B). В базі сигнатур представлені класи вірусів (черв'яки, трояни тощо), в кожному класі представлена деяка кількість відповідних сигнатур. Отже, маючи число і процент співпадінь, можна зробити прогноз, до якого з «вірусних» класів належить знайдена сигнатура. В залежності від класу загрози користувачу будуть запропоновані різні варіанти вирішення проблеми.

Для покращення результатів роботи моделі та уточнення даних було також реалізовано елемент машинного навчання за алгоритмом випадковий ліс. Нижче приведено основні етапи виконання алгоритму:

1) сформувати початкову вхідну вибірку даних;

2) визначити кількість зразків для кожного класу;

3) вибрати деяку кількість N випадкових одиниць даних з початкової вибірки;

4) використовуючи N випадкових одиниць побудувати дерево рішень;

5) на кожній з вершин

а) обрати критерій для розгалуження вузла;

b) провести навчання;

c) обрати одиницю даних, яка задовольняє критерію;

6) виконати розгалуження на піддерева, відповідно до отриманих даних на попередньому кроці;

7) виконати пункти, починаючи з 2, до закінчення роботи алгоритму;

Розглянемо кроки алгоритму для поставленої задачі;

Отже, формуємо початкову вибірку вхідних даних для Баєсової мережі. Вхідними даними є кількість та процент співпадінь.

Наступний крок – визначення кількості зразків для кожного класу. Припустимо, що є такі класи вірусів, як Черв'яки, Троянські віруси, Віруси-маскувальники, Рекламні віруси та Віруси-шпигуни. Нехай у кожному класі буде представлено по 10 сигнатур.

Далі необхідно вибрати деяку кількість даних з вибірки. Виберемо одну одиницю даних A (число сигнатур з бази, з якими співпадає досліджувана сигнатура) та чотири одиниці даних B (процент співпадіння) (Рисунок 3.8).

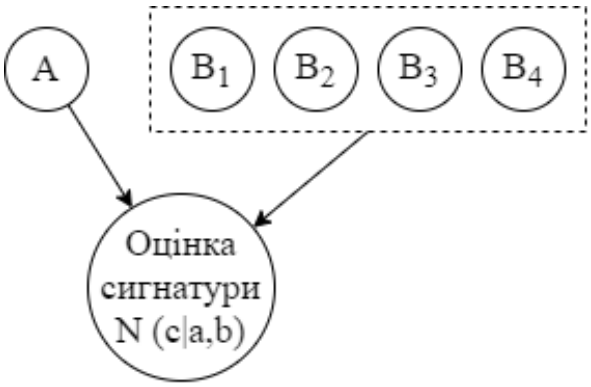


Рисунок 3.8 – Крок № 3 Алгоритму випадковий ліс

Використовуючи обрані дані, будуємо дерево рішень.

В результаті визначимо до якого класу загрози належить сигнатура.

Для оптимізації роботи та зменшені кодової бази для роботи з алгоритмом випадковий ліс використовується готова реалізація RandomForest з фреймворку Accord.NET Framework.

Отже, у даному розділі подано інформацію про використання баєсовської мережі в скануючому модулі клієнтської частини розробленого мережевого антивіруса а також поєднання з сигнатурним скануванням. Також приведено алгоритм навчання мережі для покращення результатів роботи.

3.3 Ефективність методу

Як вже було сказано у попередньому підрозділі – розпізнавання вірусної загрози у системі відбувається за рахунок поєднання двох методів сканування – звичайного порівняння з базою та аналізу сигнатур на основі баєсовської мережі. Це поєднання методів робить процес виявлення вірусів набагато ефективнішим. В цьому випадку виявляються не тільки вже відомі раніше загрози, а і невідомі, але подібні до раніше знайдених.

Проте точність результатів а також ефективність роботи моделі в конкретній області і з конкретними даними безпосередньо залежить від обраного алгоритму навчання або побудови баєсовської мережі. На ефективність також впливає кількість і якість вхідних даних, які доступні для навчання. Крім того, щоб запобігти зайвих помилкових спрацювань при скануванні, кінцевий результат потребує уточнення, тобто має бути реалізовано алгоритм самонавчання – тільки після цього баєсовька мережа стає повноцінною моделлю.

Під самонавчанням розуміється це навчання моделі за рахунок запрограмованих алгоритмів, які здатні аналізувати отримані дані і прогнозувати вихідні дані на основі проведеного аналізу. Отримуючи нові дані, такі алгоритми здатні навчатися і вдосконалювати свою роботу.

Алгоритми машинного навчання для баєсовської мережі можна умовно поділити на категорії:

1. Навчання з вчителем.
2. Навчання з частковим використанням вчителя.
3. Навчання без вчителя.

4. Навчання з підкріпленням.

Під навчанням з вчителем мається на увазі навчання на прикладах. Існує задача, людині (вчителю) відоме рішення. Для навчання моделі готується набір заздалегідь відомих даних, які с свою чергу включають і вхідні і вихідні дані. Алгоритм шукає у наданій множині даних закономірності виникнення конкретних рішень. Після цього модель виконує прогнозування і ці прогнози корегуються вчителем. Таке навчання повторюється до тих пір, поки алгоритм не досягне необхідної точності. До цієї категорії відносять алгоритми класифікації, регресії.

У навчанні з частковим залученням вчителя використовують як готові приклади, такі як при навчанні з вчителем, так і не систематизовані дані. Алгоритм повинен на основі існуючих прикладів навчитися обробляти нерозмічені масиви даних.

У випадку навчання без вчителя, не існує людини, яка би корегувала роботу моделі, а також відсутні приклади з відомими вихідними даними. Алгоритм повинен систематизувати надану множину даних будь-яким доступним йому способом. До цієї категорії належать алгоритми кластеризації.

Навчання з підкріпленням це шлях проб і помилок. Моделі надаються дані, параметри для роботи з ними, набір дій і заздалегідь відомі результати. Модель повинна на кожній новій ітерації своєї роботи виконуючи певні дії отримувати результат і оцінювати його. Якщо результат не є оптимальним, модель змінює підхід до роботи з даними, і знову оцінює результат. Так продовжується до тих пір, доки не буде отриманий оптимальний результат.

Навчальна вибірка повинна демонструвати можливості вірусу так, щоб передавати їх специфічність і взаємозв'язок один з одним. Тобто, при проведенні експериментів слід охопити якомога більше варіантів поведінки кожної зловмисної програми. Для цього потрібно проводити не тільки спостереження за поведінкою програми в нормальних умовах, де вона буде успішно виконана. Необхідно також забезпечити умови, в яких активність

будь-якої функції шкідливої програми можна перервати, не зачіпаючи інших її функцій. Тоді, якщо деякі інші функції залежать від перерваної, вони також не виконаються, що і буде зафіксовано в результатах експерименту. Таким чином, буде виявлена умовна незалежність між перерваною функцією і всіма іншими.

На вибір алгоритму навчання також впливає задача, яку необхідно вирішити. Правильно обраний алгоритм підвищить ефективність методу. Приклади таких задач, що можна вирішити за допомогою баєсовських мереж:

1. Класифікація (по набору медичних характеристик потрібно поставити діагноз або за даними зондування ґрунтів визначити наявність корисних копалин).

2. Регресія (за специфікацією автомобіля та режимом їзди спрогнозувати витрату палива, за характеристикою району його екологічною обстановкою та транспортним зв'язком оцінити вартість житла).

3. Кластеризація (по фізико-географічним і економічним показникам розбити країни світу на групи схожих за економічним становищем держав, за результатами маркетингових досліджень серед безлічі споживачів виділити характерні групи за ступенем інтересу до продукту).

4. Ідентифікація (по камерах спостереження в аеропорті виявити людину та ступінь загрози від неї).

5. Прогнозування (прогноз показників роботи пристрою за даними датчиків з систем його моніторингу, прогнозування зросту чи падіння акцій компанії).

6. Вилучення знань (отримання нових знань про досліджуваний процес).

Задача з виявлення чи є дана сигнатура небезпечною, а також до якого вірусу вона належить є задачею класифікації.

Найпопулярнішими алгоритмами машинного навчання для задач типу класифікації є:

1. Наївний баєсовскій алгоритм.

Цей алгоритм базується на теоремі Баєса з допущенням про незалежність ознак. Моделі, що будуються за допомогою цього класифікатора мають просту структуру і призначені для роботи з великими обсягами даних.

Переваги:

1. Класифікація даних виконується швидко і просто, навіть при наявність великої кількості класів.

2. Краще працює з категорійними ознаками, ніж з безперервними.

Недоліки:

1. У випадку, коли ознака не зустрічалася в наборі раніше, модель визначить для цієї ознаки нульову ймовірність і здійснити прогноз не зможе.

2. Значення прогнозів не завжди є достатньо точними.

3. Допущення про незалежність ознак також є одним з недоліків, тому що набори повністю незалежних ознак зустрічаються нечасто.

Цей алгоритм зарекомендував себе для вирішення таких задач як: виконання класифікації у реальному часі, багатокласова класифікація, класифікація текстів, фільтрація спаму тощо. З цього випливає, що цей алгоритм найкраще справляється з класифікацією текстів.

2. Метод опорних векторів.

Цей алгоритм відносять до бінарних класифікаторів. Основна ідея цього методу полягає в побудові нелінійних площин (поділяючі гіперплощини), які розділятимуть об’єкти вибірки. Чим більша відстань між площиною і об’єктом, тим менша буде середня помилка класифікатора.

Переваги:

1. Є одним з найшвидших методів знаходження рішень.

2. Площина, що розділяє об’єкти є максимальною по ширині, що дає можливість покращити класифікацію.

3. Гарно справляється з класифікацією складних даних невеликого і середнього об’єму.

Недоліки:

1. Є чутливим до шумів в даних.

3. Дерево рішень.

Цей метод має ієрархічну деревовидну структуру, яка складається з умов «якщо» і відповідей «тоді». За умови відповідають гілки (ребра) дерева, а листя це значення цільової функції. У найпростішому випадку дерево працює таким чином: дані потрапляють у вузол (гілку, ребро) і проходять перевірку умовою, після цього діляться на ті що задовольняють умові і на ті, що не задовольняють умові. Тобто початкова множина даних ділиться на дві підмножини, і до кожної підмножини знову застосовують умову. Таким чином перевірка умови проводиться до тих пір, поки не буде досягнуто умови закінчення роботи алгоритму і останній вузол стає листом. Лист представляє собою контейнер даних, які задовольняють всім умовам. До одного листка дерева існує тільки один шлях.

Переваги:

1. Класифікація об’єктів відбувається за чіткими правилами.

2. Здатні досить швидко виконати прогнозування.

3. Алгоритм здатен працювати з великими об’ємами даних.

4. Не потребує підготовки даних.

Недоліки:

1. Присутня проблема «перенавчання» - складні конструкції, що недостатньо точно описують дані.

4. Випадковий ліс.

Випадковий ліс це своєрідне поєднання декількох алгоритмів в один з метою отримання кращої ефективності прогнозування.

Цей алгоритм вважається одним із найточніших і одним з небагатьох універсальних алгоритмів машинного навчання. Його універсальність полягає в тому, що він здатний вирішувати багато завдань, і існують випадкові ліси для вирішення задач класифікації, регресії, розділення по групах, пошуку аномалій та ідентифікації. Крім того існують готові бібліотеки на багатьох мовах для роботи з цим алгоритмом.

Базовою одиницею алгоритму випадковий ліс є дерево рішень. Його можна розглядати як серію питань так / ні про вхідних дані. Ці питання призводять до передбачення певного класу. Рішення приймаються моделлю так само, як і людиною, тобто задаються питання про доступні дані до тих пір, поки не буде прийнято певне рішення. Таким чином дерево рішень формує запити, за допомогою яких алгоритм звертається до даних.

У даному алгоритмі вхідна вибірка ділиться на підвибірки, по яким будуються дерева рішень (для кожного дерева своя підвибірка). Для того, щоб побудувати кожне з розщеплень проглядається деяка кількість випадкових ознак (для кожного нового розщеплення свої випадкові ознаки). Вибираються найкращі ознаки і розщеплення по заздалегідь заданому критерію. Дерево будується до тих пір поки вибірка не закінчиться (поки в листях не залишаться представники тільки одного класу).

Чим більше дерев, тим краще якість, але час налаштування і роботи випадкового лісу також пропорційно збільшуються. Часто при збільшенні кількості дерев якість на навчальній вибірці підвищується. Виходячи з цього є можливість розрахувати кількість необхідних дерев для конкретного випадку.

При збільшенні числа ознак збільшується час побудови лісу, а дерева стають одноманітними. Цей параметр є найважливішим. Він налаштовується в першу чергу (при достатній кількості дерев у лісі).

Чим менше глибина дерев, тим швидше будується і працює випадковий ліс. При збільшенні глибини різко зростає якість навчання і контролю. Найкраще використовувати максимальну глибину (крім випадків, коли об'єктів занадто багато і виходять дуже глибокі дерева, побудова яких займає забагато часу). При використанні неглибоких дерев зміна параметрів, пов'язаних з обмеженням числа об'єктів в листі і для поділу, не призводить до істотного ефекту. Неглибокі дерева рекомендують використовувати в задачах з великим числом шумових об'єктів (викидів).

Переваги:

1. Висока швидкість навчання.

2. Алгоритм має фіксовану кількість ітерацій.

3. Здатен працювати з великими об’ємами вхідних даних.

4. Висока точність вихідних даних.

5. Здатен працювати з непідготовленими даними.

Недоліки:

1. Побудована модель потребує багато пам’яті для зберігання і роботи.

2. Модель, що пройшла навчання, може працювати повільніше інших алгоритмів.

3. Алгоритм не здатний до екстраполяції (екстраполяція передбачає роботу з «невідомими» на основі відомого знання і з «майбутнім» на основі знання минулого та нинішнього).

В якості алгоритму для навчання було обрано алгоритм випадковий ліс. Реалізована система з алгоритмом навчання випадковий ліс здатна коригувати та уточнювати результуюче значення, що призводить до більш точного виявлення чи є дана сигнатура небезпечною, а також відсіювання помилкових спрацювань. Крім того, це значно покращує результати сканування у виявленні вірусів, сигнатури яких невідомі та не містяться в базі даних.

Отже, для покращення результатів сканувань та ефективності роботи моделі було реалізовано елемент самонавчання моделі за алгоритмом випадковий ліс на вже існуючих наборах вірусних сигнатур

3.4 Висновки

У даному розділі дипломної роботи було розглянуто баєсовську мережу та її зв’язок з даною темою, а також алгоритми її навчання для покращення вихідних результатів роботи. Також було розглянуто використання баєсовської мережі в розробленому антивірусному комплексі та її поєднання із іншими методами виявлення вірусних небезпек, а також навчання моделі за алгоритмом випадковий ліс з метою більш точного розпізнавання загрози та покращення ефективності роботи системи виявлення в цілому.

4 РЕАЛІЗАЦІЯ СИСТЕМИ ТА ЕКСПЕРИМЕНТИ

4.1 Реалізація системи

Базуючись на сформованих вимогах, що висуваються до розподілених систем та аналізі існуючих рішень (підрозділи 1.1 – 1.2) було розроблено програмний комплекс, що дозволяє побудувати антивірусну розподілену систему для виявлення вірусної загрози. Як вже було сказано у підрозділі 2.1 – розроблена система має клієнт-серверну сервіс-орієнтовану архітектуру та складається з двох основних компонентів: клієнту та серверу.

Серверна (центральна) компонента не має графічного інтерфейсу користувача та являє собою службу Windows, яка після інсталювання на серверній машині буде автоматично запускатись при перезавантаженні операційної системи. Як вже було сказано у підрозділі 2.2 ця компонента працює в автоматичному режимі та не потребує втручання користувачів.

Клієнтська компонента має зручний графічний інтерфейс, що дозволяє на інтуїтивному рівні працювати з програмою та її функціоналом. Більша частина робочого процесу тут є автоматизованою та не вимагає додаткових дій з боку користувача.

Інтерфейс клієнтської частини для простоти та зручності користувача реалізований у вигляді SWA (Single window application) – тобто єдиного вікна для всіх користувацьких дій та взаємодії з програмою. Він складається з двох основних частин – панелі з назвами розділів меню з лівого боку та відповідно вмісту цих розділів справа. При кліку на назву певного розділу пункту меню відбувається його відкриття у правій частині. що відображається при кліку по назві розділу.

Стартовим розділом при запуску програми за замовченням є вікно «Home». Інтерфейс клієнтської програми з цим відкритим розділом зображено на рисунку 4.1.

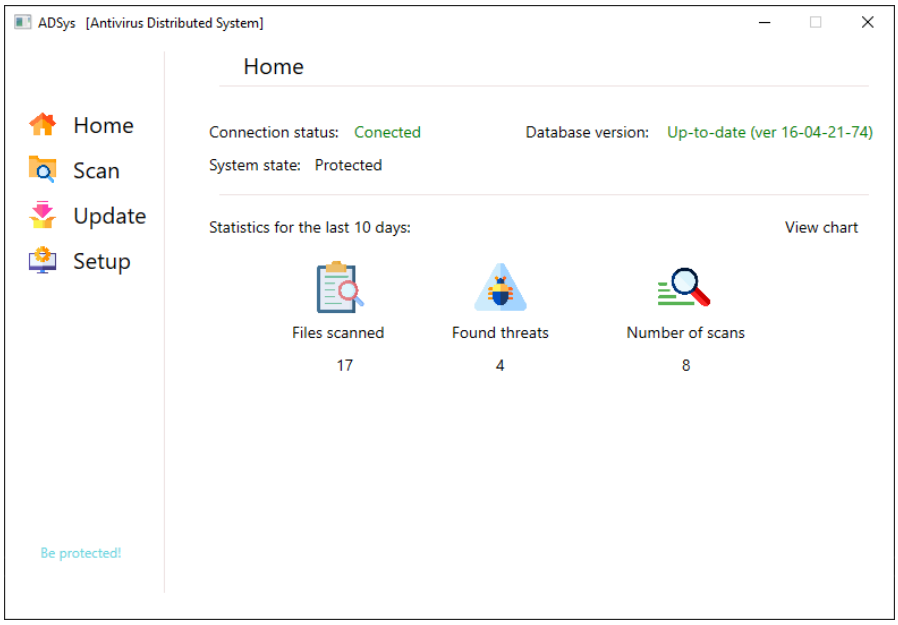


Рисунок 4.1 – Основне вікно програми з відкритим розділом «Home»

Основних розділів чотири, кожен з яких відповідає за певний функціонал:

1. Home – наглядно надає користувачу основну інформацію по роботі з програмою. Тут відображено стан з'єднання з сервером, версію бази та статистику роботи за останні 10 днів.

2. Scan – надає доступ до функціоналу сканування. Можливо обрати тип сканування, а також переглянути результати.

3. Update – цей розділ відповідає за оновлення вірусної бази даних. У ньому відображається версія бази, актуальність та дата останнього оновлення. Наявна також кнопка для примусової перевірки оновлень, у випадку коли користувач не хоче очікувати автоматичного оновлення баз.

4. Setup – цей розділ дозволяє при встановленні клієнтської компоненти задати адресу сервера, а також за потреби вимкнути сканування на основі баєсівської мережі. Проте цей функціонал є тільки експериментальний та не рекомендується для постійної роботи, оскільки призводить до погіршення результатів сканування. Це добре описано у підрозділі 4.2.

Всі основні технічні операції (оновлення, перепідключення після втрати зв’язку, надсилання логів з результатами сканування на сервер, надсилання нових знайдених сигнатур та інші) відбуваються автоматично та приховані від користувача для спрощення алгоритму роботи з програмою та стабільності роботи самої програми. Всі процеси автоматизовано і у крайньому випадку за неможливості виконати ту чи іншу операцію – користувач отримає певне повідомлення.

В реалізованій системі сканування на наявність вірусних загроз у системі відбувається завжди з використанням баєсівської мережі, окрім випадків коли цей функціонал примусово вимкнено. Як вже було сказано у підрозділі 3.2 – баєсівська мережа використовується для евристичного сканування та розпізнавання загроз, схожих на вже відомі. Таке сканування поєднано зі звичайним сигнатурним, що показує високі результати у виявленні загроз (підрозділ 4.3).

Користувачу доступно три варіанти сканування (Рисунок 4.2):

1) за вибором певного файлу чи каталогу у файловій системі;

2) повне сканування;

3) заплановане сканування;

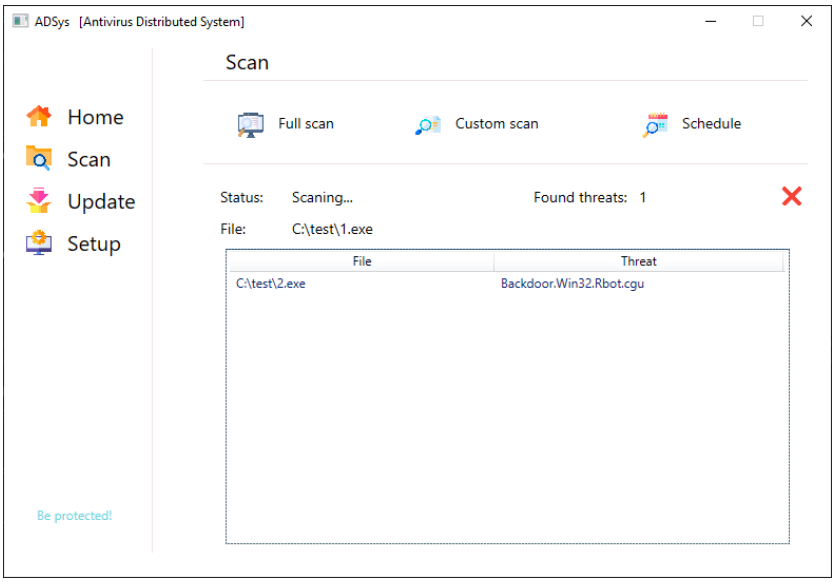


Рисунок 4.2 – Вікно програми із запущеним скануванням

Отже, розподілена антивірусна система реалізована з двох основних частин: сервера та клієнта. Сервер працює незалежно від користувача та не має графічного інтерфейсу як у клієнтської частини, яка має зручне керування всім наявним функціоналом. Сканування за замовченням відбувається з використанням баєсівської мережі, всі додаткові внутрішні робочі процеси програми автоматизовано та не потребують втручання користувача.

4.2 Експерименти

Для визначення ефективності використаного методу, а також системи в цілому необхідно провести експериментальні тестування розробленої антивірусної системи за декількома тест кейсами (випадками). Оскільки у розробленій системі компоненти мають працювати у поєднанні один з одним тестові ситуації також мають моделювати таку ж роботу. Крім того однією з основних задач, що потрібно вирішити під час тестувань – це ефективність сканування саме за допомогою за допомогою баєсівської мережі. Також потрібно перевірити роботу розробленої антивірусної системи з малим об’ємом вірусної бази даних. Додатково також потрібно перевірити роботу мережі при сценарії з відсутністю зв’язку під час роботи та неможливості оновлення бази даних. Базуючись на цьому було сформовано наступні тест кейси:

1. Сканування на одній машині (один клієнт) з малою сигнатурною базою та без використання баєсівської мережі – цей кейс направлено на випадки, коли у базі заздалегідь недостатньо даних для повноцінної оцінки – чи становить знайдена сигнатура загрозу. У клієнтській програмі буде примусово тимчасово вимкнено сканування на основі баєсівської мережі.

2. Сканування на одній машині (один клієнт) з малою сигнатурною базою і використанням баєсівської мережі – цей кейс як і попередній направлено на випадки роботи при недостатніх даних, проте у цьому випадку для аналізу додатково також використовується функціонал на основі баєсівської мережі.

3. Сканування на одній машині з великої сигнатурною базою та без баєсівського сканування – цей кейс направлено на випадки достатньої кількості даних, проте можливо, недостатньої кількості для сигнатурного сканування. Додаткове еврістичне сканування у цьому випадку виконуватись не буде.

4. Сканування на одній машині з великою кількістю даних (велика база) та увімкненим скануванням з використанням мережі Баєса.

5. Сканування на декількох машинах з малою сигнатурною базою та без додаткового використання баєсівської мережі. Цей кейс аналогічний до першого кейсу з єдиною умовою, що сканування одночасно буде виконуватись на 5 комп'ютерах.

6. Сканування з використанням декількох машин та великої вірусної бази без використання баєсівської мережі.

7. Сканування на декількох машинах з використанням малої вірусної бази та баєсівської мережі.

8. Сканування на декількох машинах з великою вірусною базою та баєсівським скануванням.

9. Сканування на всіх машинах з малою базою та лише одна машина з п'яти буде додатково використовувати покращене сканування на основі баєсівської мережі.

10. Сканування на декількох машинах з великою вірусною базою але баєсівським скануванням тільки на одній із них.

11. Сканування на одній машині з великою базою та відсутністю зв'язку з сервером та баєсівським скануванням.

Відповідно тестові кейси з першого по четвертий мають показати ефективність обраного методу (другий та четвертий кейси) у порівнянні зі звичайним сигнатурним скануванням (перший та третій кейси) при роботі на одній машині у мережі.

Кейси з п'ятого по восьмий мають показати ефективність розподіленого сканування та використання масиву з однорідних клієнтських компонентів як при звичайному сигнатурному скануванні так і при скануванні на основі баєсівських мереж.

Дев'ятий та десятий кейси мають показати ефективність роботи одного окремого вузла для всієї розподіленої антивірусної системи в цілому при використанні баєсівської мережі.

Відповідно останній кейс моделює ситуацію, коли під час роботи всієї системи пропадає зв’язок із центральним сервером. При цьому в експерименті приймає участь тільки одна машина, оскільки у ізольованому середовищі поведінка клієнтських компонент однакова, а також після встановлення зв’язку відбудеться обмін даними.

Для сканування на декількох машинах використовувалось п’ять ноутбуків з системними характеристиками, що задовольняють рекомендованим вимогам для роботи клієнтських компонентів розробленої мережі. Серверну компоненту встановлено на окремий комп’ютер, що також відповідає вимогам та виступає сервером у локальній мережі де проводились експерименти з розробленою системою.

Всі машини з встановленими компонентами знаходяться в одній локальній мережі та знаходяться в одній побудованій розподіленій мережі. При тестуванні з участю декількох клієнтів сканування запускалось не одночасно, а з невеликою затримкою для кожного клієнта, швидкість проходження тестів у рамках проведених експериментів не фіксується та не береться до уваги.

Для тестування попередньо з сервера до всіх клієнтів була надіслана та встановлена окрема вірусна база, яка необхідна для певного кейсу. Вірусна база, що надсилалась, була сформована на сервері з записів, обраних випадковим чином. У малій сформованій вірусній базі містилось 1200 сигнатур, велика база містила 22000.

У якості вірусних загроз для тестування був підготовлений окремий спеціальний тестовий набір з файлів, які містять у собі вже заздалегідь відомі віруси, список яких наведено у таблиці 4.1. Файли з вірусами були підібрані таким чином, щоб вони містили різнопланові типи загроз.

Таблиця 4.1 – Тестовий вірусний набір

Назва вірусу
Trojan.KillProc.8107
Net-Worm.Win32.Kido
Trojan.MulDrop4.36607
Backdoor.Win32.OBLIVION.AB
Bloodhound.File.String
Trojan.Downloader.Vbs.Small.M
W32.Spybot.Worm

Тестові сигнатурні вірусні бази даних базувались на вільних антивірусних базах від рішення Clam AntiVirus, які додатково були конвертовані для сумісності з розробленою антивірусною системою.

Під час тестування системи за першим тестовим кейсом з малою базою та тільки з використанням сигнатурного методу на одному клієнті було виявлено лише 2 загрози з 7. Оскільки при цьому нових загроз виявлено не було, а тільки ті, що містяться у базі – до центру було відправлено лише звіт про проходження сканування.

При покращеному скануванні з використанням баєсівської мережі за другим кейсом вже було виявлено 3 загрози. Низький результат у виявленні пояснюється досить малим розміром наданої для цих кейсів вірусної бази, а відповідно і недостатніми даними. Але незважаючи на це при скануванні методом, що базується на використанні баєсівської мережі було виявлено сигнатуру, що не містилась у базі, проте частково схожа до попередніх. У цьому випадку вона була відправлена до центральної компоненти, яка після внесення у базу та перевірки наявності клієнтів у системі не була далі відправлена по причині , що у побудованій системі містився тільки один клієнтський вузол.

При тестуванні системи за третім кейсом вже використовувалась велика база вірусних загроз і відповідно при скануванні на співпадіння було виявлено 4 загрози. До центру так само було відправлено лише звіт.

Сканування з баєсівською мережею у четвертому кейсі показало, що у системі наявні 6 загроз, проте однієї з загроз так і не було виявлено через недостатню кількість наявних у базі сигнатур. Дві нові сигнатури після закінчення сканування разом зі звітом було відправлено до центру.

При наступному – п’ятому кейсі, використовувалось вже 5 машин з малою базою та без баєсівського сканування. В результаті кожна з машин знайшла лише по 2 наявних у системі загрози як і при скануванні за першим тестовим кейсом. При цьому не було виявлено нових сигнатур та обміну між вузлами у розподіленій мережі фактично не було. З вузла до центру відправлявся лише звіт після завершення кожного сканування.

При тестуванні за шостим кейсом всі п’ять машин мали більшу базу, проте баєсівське сканування все ще не використовувалось. Результатом цього стало по 4 знайдених загрози на кожній з машин. Обміну між вузлами так само як і у попередньому випадку не було.

Сьоме та восьме тестування вже проходило з використанням баєсівської мережі та відрізнялося лише розміром бази. При сьомому кожен з вузлів виявив по 3 загрози, при восьмому кейсі по 6 загроз відповідно. Проте у цих випадках вже був активний обмін даними між компонентами. Оскільки вузли які виявили нову загрозу, що за сигнатурою схожа до тих, що містилась в базі, одразу повідомляли про неї в центр, Центральна компонента в свою чергу розсилала зразки до інших клієнтів які виконували сканування. При цьому деякі з клієнтів вже виконували сканування маючи оновлену інформацію про сигнатури. Сканування відповідно у цьому випадку відбувалося швидше оскільки на розпізнавання вірусу у цьому випадку витрачалось менше часу.

Дев’ятий та десятий кейс схожі за умовами. Під час тестування за ними лише перший вузол використовував баєсівське сканування. Розмір бази відповідно у дев’ятому кейсі був малий, у десятому великий. Як результат при

скануванні за дев’ятим кейсом першим вузлом було виявлено 3 вірусних загрози, другий та третій вузли виявили по дві, а четвертий та п’ятий вже по 3 загрози. Відповідно при скануванні за десятим тест кейсом на першому вузлі знайдено 6 загроз, на другому та третьому по 4, а на четвертому та п’ятому вже по 6. Такі значення пояснюються тим, що перший вузол після знаходження загроз одразу повідомляв про них до центру. Центр в свою чергу надсилав їх оновленнями до інших вузлів. І як результат четвертий та п’ятий вузол встигли провести сигнатурне сканування вже маючи оновлені дані, що базувались на результатах від першого вузла. Враховуючи що при реальних сценаріях роботи системи майже відсутні ситуації коли всі вузли будуть працювати одночасно над скануванням однакових даних, можна зробити припущення, що при збільшенні кількості вузлів у системі, а також при неодночасному скануванні кількість загроз, що будуть скоріше виявлятись значно зростає.

Одинадцятий тест проводився на одній машині з великою базою та баєсівським скануванням. У цьому кейсі під час сканування було виявлено 6 загроз та знайдено 2 нових сигнатури, які не містились у базі. Проте через відсутність зв’язку не було можливості їх передати до центру, відповідно вони разом зі звітом про сканування були відправлені пізніше після відновлення зв’язку. Основуючись на цьому можна зробити висновок про таку типову поведінку у майбутньому в усіх машин де буде відсутній зв’язок із центральною компонентою та відновлення нормальної роботи після встановлення з’єднання.

Для більшої інформативності та наглядності опис самих тест кейсів та результатів їх проходження подано у таблиці 4.2, при цьому кількість у кейсах де вузлів було декілька подано через слеш.

Таблиця 4.2 – Тестові кейси та результати

№ кейсу	Кількість вузлів	Розмір бази	Використання баєсівського сканування	Зв’язок з центром	Всього загроз	Загроз виявлено
1	1	мала	ні	так	7	2
2	1	мала	так	так	7	3
3	1	велика	ні	так	7	4
4	1	велика	так	так	7	6
5	5	мала	ні	так	7	2/2/2/2/2
6	5	велика	ні	так	7	4/4/4/4/4
7	5	мала	так	так	7	2/2/2/2/2
8	5	велика	так	так	7	6/6/6/6/6
9	5	мала	Тільки на першому вузлі	так	7	3/2/2/3/3
10	5	велика	Тільки на першому вузлі	так	7	6/4/4/6/6
11	1	велика	так	ні	7	6

Отже, в даному підрозділі описано проведені експерименти, що проводились з метою визначення роботи чи є ефективним обраний метод, а також розподілена мережа, що будувалась з його використанням. Для проведення тестувань було створено всі можливі тест кейси з виключенням тих, які будуть подібні тим, що вже розглянуто. Після цього були проведені відповідні тестування та зібрано інформацію, яка для зручності опрацювання була зібрана у таблицю.

На основі отриманих даних можна робити певні висновки, що наведено у наступному підрозділі (підрозділ 4.3).

4.3 Ефективність методу

Аналізуючи експериментів з підрозділу 4.2 та основуючись на їх результатах можна зробити певні висновки. Отже, метод виявлення на основі баєсівської мережі показує свою ефективність. Його ефективність зростає зі збільшенням об’єму бази сигнатур, тобто зі збільшенням кількості даних для опрацювання та навчання. Це добре видно на результатах другого та четвертого тестових кейсів (Таблиця 4.3), де зі збільшенням кількості вхідних даних збільшилась відповідно і кількість виявлених та розпізнаних загроз.

Таблиця 4.3 – Результати другого та четвертого кейсів

№ кейсу	Всього вірусів	Виявлено вірусів	Кількість записів у базі
2	7	3	1200
4	7	6	22000

Очевидно, що кількість сигнатур, що містилась у тестовій базі була не достатня для виявлення останньої сьомої загрози. Також варто зазначити, що при цих самих вхідних даних звичайний сигнатурний метод показав гірші результати: 2 та 4 сигнатури відповідно.

Варто також окремо зазначити ефективність роботи сканування на основі баєсівської мережі у розподіленій системі. Так у дев’ятому та десятому кейсах тільки один вузол з п’яти використовував такий тип сканування, проте таке сканування виявило більшу кількість загроз за рахунок чого вдалося оперативно попередити інші вузли про можливу загрозу. Тому останні два вузла базуючись на результатах баєсівського сканування змогли виявити більшу кількість загроз ніж при звичайному сигнатурному методі. Ситуація, коли інші два вузла не виявили загрозу пояснюється тим, що на час отримання оновлень вони вже просканували ті файли, що містили надіслані сигнатури. Кількість знайдених загроз при дев’ятому та десятому тестових кейсах

наведена у таблиці 4.4, перший вузол при цьому використовував баєсівське сканування.

Таблиця 4.4 – Кількість загроз, знайдених на вузлах (9-й та 10-й кейси)

Номер вузла	1	2	3	4	5
Кількість знайдених загроз (9 кейс)	3	2	2	3	3
Кількість знайдених загроз (10 кейс)	6	4	4	6	6

Крім того метод показав свою ефективність і у випадках коли всі вузли використовували баєсівські мережі для сканування (сьомий та восьмий кейси). У цих випадках деякі вузли яким вже надійшла інформація про нову наявну загрозу не витрачали час на додаткове сканування, а використовували сигнатурний метод для розпізнавання загрози.

Серед недоліку цього методу можна зазначити деякі помилкові спрацювання. Це вирішується збільшенням кількості даних для аналізу, а також додатковими налаштуваннями виключень.

Отже, підсумовуючи вище наведене варто зазначити, що обраний метод показує свою ефективність, особливо зі збільшенням кількості вхідних даних. А також використання сканування на основі баєсівської мережі у взаємозв’язаних компонентах покращує ефективність роботи всієї антивірусної розподіленої системи в цілому.

4.4 Висновки

В даному розділі наведено інформацію по реалізації компонентів розподіленої мережі, а також описано основні елементи інтерфейсу користувача клієнтської компоненти.

Було також відображено експерименти, що проводились з розробленою розподіленою антивірусною системою. Система тестувалась на різнопланових тестових кейсах. Результати тестувань були зведені у таблицю для подальшого аналізу і висновків стосовно використаного методу, а також самої системи в цілому.

Базуючись на результатах експериментів було встановлено ефективність методу та розробленої мережі. Ефективність методу відповідно зростає зі збільшенням вхідних даних для аналізу. Крім того комбінація з сканування на основі баєсівької мережі дозволяє не тільки розпізнати більшу кількість загроз, а і покращити роботу системи по виявленню в цілому.

ВИСНОВКИ

Отже, у роботі на основі проведених теоретичних, а також практичних досліджень було розроблено РС, що здатна до самоорганізації та призначена для виявлення ЗПЗ в комп’ютерах локальної мережі, що базується на БМ для покращення детектування загроз.

Були отримані наступні результати роботи:

1) Встановлено, що виявлення ЗПЗ у локальних комп’ютерних мережах згідно дослідження може бути виявлено за допомогою АРС з використанням баєсовської мережі.

2) Удосконалено архітектуру РС, в якій на відміну від рішень, що використовуються, покращено внутрішню взаємодію центру та компонентів системи. Як результат, РС, що розроблена за такою архітектурою може швидко нарощувати кількість своїх елементів, а також покращувати функціонал за рахунок використання додаткових модулів в окремих вузлах системи. У системі всі компоненти обмінюються результатами сканувань та знайденими загрозами один з одним з використанням центру, що дозволяє підтримувати всю систему в актуальному стані у будь який проміжок часу.

3) Розроблений метод, що організовує та підтримує цілісність РС, враховує поточні стани компонентів, а також координує її архітектуру і може перебудовувати систему в залежності від ситуацій на вузлах і впливів ЗПЗ.

4) Проведена розробка системи оцінки ефективності використаних рішень і розроблено програмне забезпечення для розробленої АРС з використанням БМ. Проведено експериментальні дослідження з розробленою АРС, що підтвердили ефективність запропонованих рішень.

За темою дипломної роботи опублікована одна стаття у збірнику матеріалів конференції «Збірник наукових праць Конференції АПКН-2020».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЬ

1. Tanenbaum A, van Steen M. Distributed Systems (3rd Edition). Pearson Education, Inc., 2017. p. 18-49

2. Vitillo R. Understanding Distributed Systems: What every developer should know about large distributed applications. Roberto Vitillo, 2021. 252 p.

3. Coulouris G., Dollimore J., Kindberg T., Blair G. Distributed Systems: Concepts and Design 5th Edition. Pearson, 2011. 1080 p.

4. Perry M. The Art of Immutable Architecture: Theory and Practice of Data Management in Distributed Systems. Apress, 2020. 444 p.

5. Mulder J. Multi-Cloud Architecture and Governance: Leverage Azure, AWS, GCP, and VMware vSphere to build effective multi-cloud solutions. Packt Publishing, 2020. 412 p.

6. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 161 p.

7. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley, 2020. 1232 p.

8. Глоба Л. С. Розробка інформаційних ресурсів та систем: підручник. Київ: Політехніка, 2013. 320 с.

9. Raptis D. Distributed Systems for practitioners. Dimos Raptis, 2020 259 p.

10. Adkins H., Beyer B., Blankinship P., Lewandowski P., Oprea A., Stubblefield A. Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems 1st Edition. O'Reilly Media, 2020. 558 p.

11. Cachin C., Guerraoui R., Rodrigues L. Introduction to Reliable and Secure Distributed Programming 2nd edition. Springer, 2011. 386 p.

12. Official site Apache Cassandra. URL: https://cassandra.apache.org/ (дата звернення 19.03.2021)

13. Live Flight Tracker - Real-Time Flight Tracker Map | Flightradar24. URL: https://www.flightradar24.com/how-it-works (дата звернення 19.03.2021)

14. Official site Amazon Web Services | AWS. URL: https://aws.amazon.com/windows/?nc1=h_ls&blog-posts-content-windows.sort-by=item.additionalFields.createdDate&blog-posts-content-windows.sort-order=desc (дата звернення 19.03.2021)

15. Burns B. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media, 2018. 166 p.

16. Chang M. F., Kwong S. An implementation of a high performance client-server system. Singapore ICCS/ISITA `92, 1992. vol.2, p. 517-521

17. Puliafito A., Riccobene S., Scarpa M. Modelling of client-server systems. *MASCOTS '95. Proceedings of the Third International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Durham, NC, 1995, P. 340-344.

18. Thompson D., Watkins D. Comparisons between CORBA and DCOM: architectures for distributed computing. *Technology of Object-Oriented Languages. TOOLS 24*, Beijing, 1997, P. 278-283.

19. Luder A., Klostermeyer A., Peschke J., Bratoukhine A., Sauter T. Distributed automation: PABADIS versus HMS. *Transactions on Industrial Informatics*. 2005 Vol. 1, No. 1, P. 31-38.

20. Stefanova-Stoyanova V., Stankov I. Multi-agent systems (MAS) in the area of IoT and using a model with Distributed Shared Memory system (DSM), *2020 XXIX International Scientific Conference Electronics (ET)*, Sozopol, 2020, P. 1-4.

21. Duong-Ba T., Nguyen T. Distributed client-server assignment. *37th Annual IEEE Conference on Local Computer Networks*, Clearwater Beach, FL, 2012, P. 296-299.

22. Jiangyun Xu and Weichang Du. Software brokers for quality of services in service-oriented distributed systems. *Second Annual Conference on Communication Networks and Services Research*. Fredericton, 2004, P. 341-344.

23. Faiz M., Shanker U. Data synchronization in distributed client-server applications. *2016 IEEE International Conference on Engineering and Technology (ICETECH)*. Coimbatore, 2016, P. 611-616.

24. Samuvelraj G., Nalini N. A survey of self organizing trust method to avoid malicious peers from peer to peer network. *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE).* Coimbatore, 2014. P. 1-4.

25. Schollmeier R. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *First International Conference on Peer-to-Peer Computing, Linköping*. Sweden, 2001 P. 101-102.

26. Quevedo G. P. L., Ocampo R. M. Festin C. A. M. A Hybrid Adaptive Localization Strategy for Peer-to-Peer Video on Demand. *2013 First International Symposium on Computing and Networking.* Matsuyama, 2013 P. 434-440.

27. Jin Y., Yi Y., Kesidis G., Kocak F., Shin J. Hybrid client-server and peer-to-peer caching systems with selfish peers. *2013 Proceedings IEEE INFOCOM*. Turin, 2013. P. 1744-1752.

28. Lin C., Lee J., Kuo L., Yeh Y., Wen J. Hybrid P2P Client-Server Data Transmission Using Dynamic Peer Grouping and Switching. *2012 International Symposium on Computer, Consumer and Control*, Taichung, 2012. P. 92-96.

29. Vadivu G. S., Gomathi C., Priya A. S. Protecting peer-to-peer networks from the denial of service attacks. *2011 3rd International Conference on Electronics Computer Technology*, Kanyakumari, 2011.P. 266-270.

30. Tian D., Liu Y. Li B. Anomaly Intrusion Detection Methods for Peer-to-Peer System. *2007 IFIP International Conference on Network and Parallel Computing Workshops (NPC 2007)*, Dalian, 2007, P. 127-130.

31. Barcellos M. P., Bauermann D., Santanna H., Lehmann M., Mansilha R. Protecting BitTorrent: Design and Evaluation of Effective Countermeasures against DoS Attacks. *Symposium on Reliable Distributed Systems*. Naples, 2008. P. 73-82

32. Saboori E. Abbaspour M. Dual-Path Peer-to-Peer Anonymous Approach. *2010 IEEE 16th International Conference on Parallel and Distributed Systems*, Shanghai, 2010. P. 835-838

33. Xie J., Yang C., Huang Q., Cao Y., Kafatos M. Utilizing Grid Computing to Support Near Real-Time Geospatial Applications. *2008 IEEE International Geoscience and Remote Sensing Symposium*, Boston: MA, 2008. P. II-1290-II-1293

34. Buyya R. Market-Oriented Grid Computing and the Gridbus Middleware. *2008 16th International Conference on Advanced Computing and Communications*. Chennai, 2008. P. 1-1

35. Wilkinson B., Ferner C. Teaching Grid Computing in North Carolina: Part I. *IEEE Distributed Systems Online*, 2006. Vol. 7, No. 6, P. 3-3.

36. Xhafa F. Discrete Event-Based Simulation of Grid Computing Systems. *2010 12th International Conference on Computer Modelling and Simulation*, Cambridge, 2010, P. 9-10.

37. Singh M. An Overview of Grid Computing. *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, 2019. P. 194-198.

38. Congfeng Jiang, Xianghua Xu and Jian Wan. Grid computing based large scale Distributed Cooperative Virtual Environment Simulation. *12th International Conference on Computer Supported Cooperative Work in Design*. Xi'an, 2008. P. 507-512.

39. Wang D., Zheng N., Xu M., Wu Y., Hu Q., Wang G. Resilient Privacy-Preserving Average Consensus for Multi-agent Systems under Attacks. *16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. Shenzhen, 2020. P. 1399-1405.

40. Kovalenko T., N. Tullah. Analysis of service delays in distributed systems with service-oriented architecture. *Proceedings of International Conference on Modern Problem of Radio Engineering, Telecommunications and Computer Science.* Lviv, 2012. P. 333-334.

41. Pasatcha P., Sunat K. A Distributed e-Education System Based on the Service Oriented Architecture. *2008 IEEE International Conference on Web Services*, Beijing, 2008. P. 791-794

42. Oberortner E., Zdun U., Dustdar S., Cavalcante A. B., Tluczek M. Supporting the evolution of model-driven service-oriented systems: A case study on QoS-aware process-driven SOAs. *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. Perth, WA, 2010. P. 1-4.

43. Abuosba K. A., El-Sheikh A. A. Formalizing Service-Oriented Architectures. *IT Professional*, 2008. Vol. 10, No. 4, P. 34-38.

44. Buyya R., Garg S. K., Calheiros R. N. SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. *2011 International Conference on Cloud and Service Computing*, Hong Kong, 2011, P. 1-10

45. Buyya R., Calheiros N., Li X. Autonomic Cloud computing: Open challenges and architectural elements. *2012 Third International Conference on Emerging Applications of Information Technology*, Kolkata, 2012, P. 3-10.

46. Markandey A., Dhamdhere P., Gajmal Y. Data Access Security in Cloud Computing: A Review. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*. Greater Noida, 2018. P. 633-636.

47. Kaufman L. M. Data Security in the World of Cloud Computing. *IEEE Security & Privacy*, 2009. Vol. 7, No. 4, P. 61-64.

48. Dikaiakos M. D., Katsaros D., Mehra P., Pallis G., Vakali A. Cloud Computing: Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing*, 2009. Vol. 13, No. 5, P. 10-13.

49. Henze M., Matzutt R., Hiller J., Mehmer E., Ziegeldorf J. H., Giet J., Wehrle K. Complying with Data Handling Requirements in Cloud Storage Systems. *IEEE Transactions on Cloud Computing*, 2020. P. 1-1

50. Dinuzzo F., Pillonetto G., Nicolao G. Client-Server Multitask Learning From Distributed Datasets. *IEEE Transactions on Neural Networks*, 2011. Vol. 22, No. 2, P. 290-303.

51. Serrano N., Hernantes J., Gallardo G. Service-Oriented Architecture and Legacy Systems. *IEEE Software*, 2014. Vol. 31, No. 5, P. 15-19.

52. Shah S. Y., Szymanski B. K., Zerfos P., Gibson C. Towards Relevancy Aware Service Oriented Systems in WSNs. *IEEE Transactions on Services Computing*, 2016. Vol. 9, No. 2, P. 304-316.

53. Luthria H. Rabhi F. A. Service-Oriented Architectures: Myth or Reality? *IEEE Software*, 2012. Vol. 29, No. 4, P. 46-52.

54. Arsanjani A., Zhang L., Ellis M., Allam A. Channabasavaiah K. S3: A Service-Oriented Reference Architecture. *IT Professional*, 2007. Vol. 9, No. 3, P. 10-17

55. Chengalur-Smith I., Duchessi P. Client-server implementation: some management pointers. *IEEE Transactions on Engineering Management*, 2000. Vol. 47, No. 1, P. 127-145.

56. Chinnappen-Rimer S., Hancke G. P. An XML Model for Use Across Heterogeneous Client–Server Applications. *IEEE Transactions on Instrumentation and Measurement*, 2008. Vol. 57, No. 10, P. 2128-2135.

57. Lim M. C2CFTP: Direct and Indirect File Transfer Protocols Between Clients in Client-Server Architecture. *IEEE Access*, 2020. Vol. 8, P. 102833-102845.

58. Nishida H., Nguyen T., Optimal Client-Server Assignment for Internet Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 2012. Vol. 24, No. 3, P. 565-575.

59. Marik V., McFarlane D. Industrial adoption of agent-based technologies. *IEEE Intelligent Systems*, 2005. Vol. 20, No. 1, P. 27-35.

60. Balaji P. G., Srinivasan D. Multi-Agent System in Urban Traffic Signal Control. *IEEE Computational Intelligence Magazine*, 2010. Vol. 5, No. 4, P. 43-51.

61. Brazier F. M. T., Kephart J. O., Parunak H. V. D., Huhns M. N. Agents and Service-Oriented Computing for Autonomic Computing: A Research Agenda. *IEEE Internet Computing*, 2009. Vol. 13, No. 3, P. 82-87

62. Sadooghi I. Understanding the Performance and Potential of Cloud Computing for Scientific Applications. *IEEE Transactions on Cloud Computing*, 2017. Vol. 5, No. 2, P. 358-371.

63. Linthicum D. S. The Evolution of Cloud Service Governance. *IEEE Cloud Computing*, 2015. Vol. 2, No. 6, P. 86-89.

64. Khan A. u. R., Othman M., Xia F, Khan A. N. Context-Aware Mobile Cloud Computing and Its Challenges. *IEEE Cloud Computing*, 2015. Vol. 2, No. 3, P. 42-49.

65. Juliadotter N. V., Choo K. R. Cloud Attack and Risk Assessment Taxonomy. *IEEE Cloud Computing*, 2015. Vol. 2, No. 1, P. 14-20.

66. Test antivirus software for Windows 10 - February 2021 | AV-TEST. URL: https://www.av-test.org/en/antivirus/home-windows/ (дата звернення 19.03.2021)

67. Порівняння антивірусів Avast. URL: https://www.avast.ua/compare-antivirus#pc (дата звернення 19.03.2021)

68. ESET офіційний сайт Есет в Україні. URL: https://www.eset.com/ua-ru/ (дата звернення 19.03.2021)

69. Bitdefender - Global Leader in Cybersecurity Software. URL: https://www.bitdefender.com/ (дата звернення 19.03.2021)

70. How it works – VirusTotal. URL: https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works (дата звернення 20.03.2021)

71. Download Security Software for Windows, Mac, Android & iOS | Avira Antivirus. URL: https://www.avira.com (дата звернення 20.03.2021)

72. McAfee | Antivirus, VPN, Cloud, Endpoint, & Enterprise Security. URL: https://www.mcafee.com/en-us/index.html (дата звернення 20.03.2021)

73. ClamavNet. URL: https://www.clamav.net/ (дата звернення 20.03.2021)

74. Cisco - Networking, Cloud, and Cybersecurity Solutions. URL: https://www.cisco.com/c/en/us/index.html (дата звернення 20.03.2021)

75. The Best Antivirus Protection for 2021 | PCMag. URL: https://www.pcmag.com/picks/the-best-antivirus-protection (дата звернення 20.03.2021)

76. Antivirus Comparison. URL: https://www.safetydetectives.com/comparison/ (дата звернення 20.03.2021)

77. The best antivirus 2021 | Paid and free antivirus tested | TechRadar. URL: https://www.techradar.com/best/best-antivirus (дата звернення 20.03.2021)

78. Antivirus software 2021: the best programs in comparison - IONOS. URL: https://www.ionos.com/digitalguide/server/security/comparison-of-the-best-antivirus-programs (дата звернення 20.03.2021)

79. The Best Antivirus Protection Software of 2021 | Security.org. URL: https://www.security.org/antivirus/best/ (дата звернення 20.03.2021)

80. Best Antivirus Software 2021: Windows, Mac, iOS & Android | CyberNews. URL: https://cybernews.com/best-antivirus-software/ (дата звернення 20.03.2021)

ДОДАТОК А

(обов'язковий)

ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗРОБЛЕНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ ВИЯВЛЕННЯ ВІРУСНОЇ ЗАГРОЗИ

```
<Window x:Class="ADS.UIComponent.MainWindow"

        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

        xmlns:local="clr-namespace:ADS.UIComponent"

        mc:Ignorable="d"

        Title="ADSys   [Antivirus Distributed System]" Height="550"
Width="800" ResizeMode="CanMinimize" Loaded="Window_Loaded">

    <Grid>

        <Grid HorizontalAlignment="Left" Height="519" VerticalAlignment="Top"
Width="140">

            <Label x:Name="label_mainHome" Content="{Binding Source={x:Static
prop:Resources.Home}}" ToolTip="{Binding Source={x:Static
prop:Resources.tooltipText_Home}}" HorizontalAlignment="Left"
Margin="55,55,0,0" VerticalAlignment="Top" Cursor="Hand" FontSize="20"
MouseLeftButtonUp="Label_mainHome_MouseLeftButtonUp"/>

            <Label x:Name="label_mainScan" Content="{Binding Source={x:Static
prop:Resources.Scan}}" ToolTip="{Binding Source={x:Static
prop:Resources.tooltipText_Scan}}" HorizontalAlignment="Left"
Margin="55,95,0,0" VerticalAlignment="Top" Cursor="Hand" FontSize="20"
MouseLeftButtonUp="Label_mainScan_MouseLeftButtonUp"/>

            <Label x:Name="label_mainUpdate" Content="{Binding
Source={x:Static prop:Resources.Update}}" ToolTip="{Binding Source={x:Static
prop:Resources.tooltipText_Update}}" HorizontalAlignment="Left"
Margin="55,135,0,0" VerticalAlignment="Top" Cursor="Hand" FontSize="20"
MouseLeftButtonUp="Label_mainUpdate_MouseLeftButtonUp"/>

            <Label x:Name="label_mainSetup" Content="{Binding
Source={x:Static prop:Resources.Setup}}" ToolTip="{Binding Source={x:Static
prop:Resources.tooltipText_Setup}}" HorizontalAlignment="Left"
Margin="55,175,0,0" VerticalAlignment="Top" Cursor="Hand" FontSize="20"
MouseLeftButtonUp="Label_mainSetup_MouseLeftButtonUp"/>

            <Image HorizontalAlignment="Left" Height="24" Margin="21,62,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/home-1.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

            <Image HorizontalAlignment="Left" Height="24" Margin="21,102,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/search-1.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>
```

```
            <Image HorizontalAlignment="Left" Height="24" Margin="21,142,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/update.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

            <Image HorizontalAlignment="Left" Height="24" Margin="21,182,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/setting.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

            <Label Content="{Binding Source={x:Static
prop:Resources.BeProtected}}" HorizontalAlignment="Left" Margin="26,439,0,0"
VerticalAlignment="Top" Background="{x:Null}" Foreground="#FF6CD6E0"/>



        </Grid>

        <GridSplitter HorizontalAlignment="Left" Margin="140,10,0,0"
Width="1" Background="#FFEEE3E3" Height="478" VerticalAlignment="Top"
Cursor="None" IsEnabled="False"/>

        <GridSplitter HorizontalAlignment="Left" Margin="475,-246,0,0"
Width="1" Background="#FFEEE3E3" Height="573" VerticalAlignment="Top"
Cursor="None" IsEnabled="False" RenderTransformOrigin="0.5,0.5">

            <GridSplitter.RenderTransform>

                <TransformGroup>

                    <ScaleTransform/>

                    <SkewTransform/>

                    <RotateTransform Angle="90"/>

                    <TranslateTransform/>

                </TransformGroup>

            </GridSplitter.RenderTransform>

        </GridSplitter>

        <Label x:Name="label_mainTitle" Content="{Binding Path=MainTitle,
Mode=TwoWay}" HorizontalAlignment="Left" Margin="205,3,0,0"
VerticalAlignment="Top" FontSize="20"/>



        <Grid x:Name="grid_home" HorizontalAlignment="Left" Height="445"
Margin="155,55,0,0" VerticalAlignment="Top" Width="620" Visibility="Visible">

            <Label Content="{Binding Source={x:Static
prop:Resources.connectionStatus}}" HorizontalAlignment="Left"
Margin="20,11,0,0" VerticalAlignment="Top" FontSize="14"/>

            <Label x:Name="label_home_status" Content="{Binding
Path=ConnectionStatus, Mode=TwoWay}" HorizontalAlignment="Left"
Margin="148,11,0,0" VerticalAlignment="Top" FontSize="14"
Background="{x:Null}" Foreground="Green"/>



            <Label Content="{Binding Source={x:Static
prop:Resources.DBVersion}}" HorizontalAlignment="Left" Margin="299,11,0,0"
VerticalAlignment="Top" FontSize="14"/>
```

```
            <Label x:Name="label_home_dbVersion" Content="{Binding
Path=DBVersion, Mode=TwoWay}" HorizontalAlignment="Left" Margin="424,11,0,0"
VerticalAlignment="Top" FontSize="14" Background="{x:Null}"
Foreground="Green"/>



            <GridSplitter HorizontalAlignment="Left" Margin="320,-205,0,0"
Width="1" Background="#FFEEE3E3" Height="573" VerticalAlignment="Top"
Cursor="None" IsEnabled="False" RenderTransformOrigin="0.5,0.5">

                <GridSplitter.RenderTransform>

                    <TransformGroup>

                        <RotateTransform Angle="90"/>

                    </TransformGroup>

                </GridSplitter.RenderTransform>

            </GridSplitter>



            <Label Content="{Binding Source={x:Static
prop:Resources.protectionStatus}}" HorizontalAlignment="Left"
Margin="20,40,0,0" VerticalAlignment="Top" FontSize="14"/>

            <Label x:Name="label_home_systemState" Content="{Binding
Path=ProtectionStatus, Mode=TwoWay}" HorizontalAlignment="Left"
Margin="113,40,0,0" VerticalAlignment="Top" FontSize="14"
Background="{x:Null}"/>

            <Label Content="{Binding Source={x:Static
prop:Resources.statistics10days}}" HorizontalAlignment="Left"
Margin="20,95,0,0" VerticalAlignment="Top" FontSize="14"/>

            <Image HorizontalAlignment="Left" Height="46"
Margin="118,140,0,0" VerticalAlignment="Top" Width="46"
Source="Resources/search-file.png" RenderTransformOrigin="0.5,-1.053"
Stretch="Fill"/>

            <Label Content="{Binding Source={x:Static
prop:Resources.filesScaned}}" HorizontalAlignment="Left" Margin="93,188,0,0"
VerticalAlignment="Top" FontSize="14"/>

            <Label x:Name="label_home_filesScaned" Content="{Binding
Path=FilesScaned, Mode=TwoWay}" HorizontalAlignment="Left"
Margin="132,217,0,0" VerticalAlignment="Top" FontSize="14"
Background="{x:Null}"/>



            <Image HorizontalAlignment="Left" Height="46"
Margin="259,140,0,0" VerticalAlignment="Top" Width="46"
Source="Resources/virus.png" RenderTransformOrigin="0.5,-1.053"
Stretch="Fill"/>

            <Label Content="{Binding Source={x:Static
prop:Resources.fTreats}}" HorizontalAlignment="Left" Margin="234,188,0,0"
VerticalAlignment="Top" FontSize="14"/>

            <Label x:Name="label_home_foundThreats" Content="{Binding
Path=FTreats, Mode=TwoWay}" HorizontalAlignment="Left" Margin="273,217,0,0"
VerticalAlignment="Top" FontSize="14" Background="{x:Null}"/>
```

```
            <Image HorizontalAlignment="Left" Height="46"
Margin="421,140,0,0" VerticalAlignment="Top" Width="46"
Source="Resources/search-num.png" RenderTransformOrigin="0.5,-1.053"
Stretch="Fill"/>

            <Label Content="Number of scans" HorizontalAlignment="Left"
Margin="388,188,0,0" VerticalAlignment="Top" FontSize="14"/>

            <Label x:Name="label_home_numScans" Content="{Binding
Path=NumbersOfScan, Mode=TwoWay}" HorizontalAlignment="Left"
Margin="437,217,0,0" VerticalAlignment="Top" FontSize="14"
Background="{x:Null}"/>



            <Label Content="View chart" HorizontalAlignment="Left"
Margin="528,95,0,0" VerticalAlignment="Top" FontSize="14" Cursor="Hand"/>

        </Grid>



        <Grid x:Name="grid_scan" HorizontalAlignment="Left" Height="445"
Margin="155,55,0,0" VerticalAlignment="Top" Width="620" Visibility="Hidden">

            <Image HorizontalAlignment="Left" Height="24" Margin="67,16,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/computer.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

            <Label Content="Full scan" HorizontalAlignment="Left"
Margin="100,11,0,0" VerticalAlignment="Top" FontSize="14" Cursor="Hand"
MouseLeftButtonUp="StartFullScan"/>



            <Image HorizontalAlignment="Left" Height="24" Margin="235,16,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/document.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

            <Label Content="Custom scan" HorizontalAlignment="Left"
Margin="268,11,0,0" VerticalAlignment="Top" FontSize="14" Cursor="Hand"
MouseLeftButtonUp="ShowFilesToSelect"/>



            <Image HorizontalAlignment="Left" Height="24" Margin="450,16,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/calendar.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

            <Label Content="Schedule" HorizontalAlignment="Left"
Margin="483,11,0,0" VerticalAlignment="Top" FontSize="14" Cursor="Hand"
MouseLeftButtonUp="ShowSchedule"/>



            <GridSplitter HorizontalAlignment="Left" Margin="320,-222,0,0"
Width="1" Background="#FFEEE3E3" Height="573" VerticalAlignment="Top"
Cursor="None" IsEnabled="False" RenderTransformOrigin="0.5,0.5">

                <GridSplitter.RenderTransform>

                    <TransformGroup>

                        <RotateTransform Angle="90"/>

                    </TransformGroup>
```

```
                </GridSplitter.RenderTransform>

            </GridSplitter>

            <Grid x:Name="grid_scanResult" HorizontalAlignment="Left"
Height="360" Margin="25,75,0,0" VerticalAlignment="Top" Width="585">

                <Label Content="Status:" HorizontalAlignment="Left"
Margin="20,6,0,0" VerticalAlignment="Top" FontSize="14"/>

                <Label x:Name="label_scan_status" Content=" {Binding
Path=LabelScanStatus, Mode=TwoWay}" HorizontalAlignment="Left"
Margin="88,6,0,0" VerticalAlignment="Top" FontSize="14"/>

                <Label Content="File:" HorizontalAlignment="Left"
Margin="20,36,0,0" VerticalAlignment="Top" FontSize="14"/>

                <Label x:Name="label_scan_fileName" Content=" {Binding
Path=ScanedFileLabel, Mode=OneWay}" HorizontalAlignment="Left"
Margin="88,36,0,0" VerticalAlignment="Top" FontSize="14"/>

                <Label Content="Found threats:" HorizontalAlignment="Left"
Margin="317,6,0,0" VerticalAlignment="Top" FontSize="14"/>

                <Label x:Name="label_scan_threats" Content=" {Binding
Path=FoundThreats, Mode=OneWay}" HorizontalAlignment="Left"
Margin="417,6,0,0" VerticalAlignment="Top" FontSize="14"/>

                <Image x:Name="img_close" HorizontalAlignment="Left"
Height="18" Margin="558,11,0,0" VerticalAlignment="Top" Width="18"
Source="Resources/close.png" RenderTransformOrigin="0.5,-1.053"
Stretch="Fill" Cursor="Hand"/>

                <!--<ListBox x:Name="listbox_threats"
HorizontalAlignment="Left" Height="280" Margin="30,70,0,0"
VerticalAlignment="Top" Width="535"/>-->

                <ListView x:Name="listView_threats"
HorizontalAlignment="Left" Height="280" Margin="30,70,0,0"
VerticalAlignment="Top" Width="535">

<ListView.View>

        <GridView>

            <GridViewColumn Header="File" DisplayMemberBinding="{Binding
File}"/>

            <GridViewColumn Header="Threat" DisplayMemberBinding="{Binding
Threat}"/>

        </GridView>

    </ListView.View>

</ListView>

            </Grid>

        </Grid>



        <Grid x:Name="grid_update" HorizontalAlignment="Left" Height="445"
Margin="155,55,0,0" VerticalAlignment="Top" Width="620" Visibility="Hidden">
```

```
            <Label Content="Connection status:" HorizontalAlignment="Left"
Margin="20,11,0,0" VerticalAlignment="Top" FontSize="14"/>

            <Label x:Name="label_update_status" Content=" {Binding
Path=ConnectionStatus, Mode=OneWay}" HorizontalAlignment="Left"
Margin="148,11,0,0" VerticalAlignment="Top" FontSize="14"
Background="{x:Null}" Foreground="Green"/>



            <Label Content="Database version:" HorizontalAlignment="Left"
Margin="299,11,0,0" VerticalAlignment="Top" FontSize="14"/>

            <Label x:Name="label_update_dbVersion" Content=" {Binding
Path=DBVersion, Mode=OneWay}" HorizontalAlignment="Left" Margin="424,11,0,0"
VerticalAlignment="Top" FontSize="14" Background="{x:Null}"
Foreground="Green"/>

            <GridSplitter HorizontalAlignment="Left" Margin="320,-222,0,0"
Width="1" Background="#FFEEE3E3" Height="573" VerticalAlignment="Top"
Cursor="None" IsEnabled="False" RenderTransformOrigin="0.5,0.5">

                <GridSplitter.RenderTransform>

                    <TransformGroup>

                        <RotateTransform Angle="90"/>

                    </TransformGroup>

                </GridSplitter.RenderTransform>

            </GridSplitter>

            <Image HorizontalAlignment="Left" Height="24" Margin="45,95,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/loop-arrow.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

            <Label Content="Update database" HorizontalAlignment="Left"
Margin="78,90,0,0" VerticalAlignment="Top" FontSize="14"
MouseLeftButtonUp="Label_MouseLeftButtonUp_1" Cursor="Hand"
MouseLeftButtonUp="UpdateVirusDB"/>/>

        </Grid>



        <Grid x:Name="grid_setup" HorizontalAlignment="Left" Height="445"
Margin="155,55,0,0" VerticalAlignment="Top" Width="620" Visibility="Hidden">

            <TextBox x:Name="textBox_uri" HorizontalAlignment="Left"
Height="23" Margin="149,20,0,0" TextWrapping="Wrap" Text=""
VerticalAlignment="Top" Width="215" FontSize="14" {Binding Path=ServerAdress,
Mode=TwoWay}/>

            <Label Content="Server adress:" HorizontalAlignment="Left"
Margin="25,16,0,0" VerticalAlignment="Top" FontSize="14"/>



            <Image HorizontalAlignment="Left" Height="24" Margin="490,21,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/mind-map.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>
```

```
            <Label Content="Connect" HorizontalAlignment="Left"
Margin="523,16,0,0" VerticalAlignment="Top" FontSize="14"
MouseLeftButtonUp="ReconnectToServer" Cursor="Hand"/>

        </Grid>


        <Grid x:Name="grid_chart" HorizontalAlignment="Left" Height="445"
Margin="155,55,0,0" VerticalAlignment="Top" Width="620" Visibility="Hidden">

            <Label Content="Server adress:" HorizontalAlignment="Left"
Margin="25,16,0,0" VerticalAlignment="Top" FontSize="14"/>

            <Image HorizontalAlignment="Left" Height="24" Margin="490,21,0,0"
VerticalAlignment="Top" Width="24" Source="Resources/mind-map.png"
RenderTransformOrigin="0.5,-1.053" Stretch="Fill"/>

        </Grid>

        <Grid x:Name="grid_schedule" HorizontalAlignment="Left" Height="445"
Margin="155,55,0,0" VerticalAlignment="Top" Width="620" Visibility="Hidden">

            <Label Content="Server adress:" HorizontalAlignment="Left"
Margin="25,16,0,0" VerticalAlignment="Top" FontSize="14"/>

        </Grid>

    </Grid>

</Window>




using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;


using WC = ADS.WebComponent;

using ADS.VirusDBComponent;

using System.Reflection;


using ADS.ClientApp.Service;
```

```
using ADS.DetectComponent;
using ADS.LoggerComponent;

namespace ADS.ClientApp
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window, IServiceCallback, IBaseModel
    {
        private ServiceClient Client { get; set; }
        private bool IsConnected { get; set; }
        private string ClientID { get; set; }
        private Status ConnectionStatus { get; set; }
        private string ProtectionStatus { get; set; }
        private string DBVersion { get; set; }
        private VirusDBComponent VirusDB { get; set; }
        private WC WebManager { get; set; }
        private DetectComponent Scanner { get; set; }
        private LoggerComponent Logger { get; set; }


        public MainWindow()
        {
            DataContext = this;
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            this.BaseInitialize();
            this.LoadConfigs();
            IsConnected = false;
            this.ConnectHost();
```

```
            this.WebManager.GetBaseConfigs();
        }

        private void ConnectHost()
        {
            if (IsConnected)
            {
                return;
            }

            try
            {
                var Connection = this.WebManager.ConnectToServer(false)

                Client = Connection.Client;
                ClientID = Connection.Client.ClientID;
                ConnectionStatus = Connection.ConnectionStatus;
                this.Logger.Report("Connected", "Connection",
Connection.Report);
            }
            catch (Exception ex)
            {
                this.Logger.Warn("Error", "Connection", ex.Message);
                this.ShowErrorNotification(ex.Message);
                this.IsConnected = false;
            }
            finally
            {
                this.WebManager.UpdateStatus();
            }
            this.IsConnected = true;
        }

        private void DisconnectHost()
```

```
        {
            try
            {
                var Connection = this.WebManager.DisconnectHost()

                Client = null;
                ClientID = "";
                ConnectionStatus = Connection.ConnectionStatus;
                this.Logger.Report("Disconnected", "Connection",
Connection.Report);
            }
            catch (Exception ex)
            {
                this.Logger.Warn("Error", "Disconnection", ex.Message);
                this.ShowErrorNotification(ex.Message);
                this.IsConnected = true;
            }
            finally
            {
                this.WebManager.UpdateStatus();
            }
            this.IsConnected = false;
        }

        private void UpdateBase(bool isFullUpdate = false, bool isForceUpdate
= false)
        {
            try
            {
                string localDbVer = this.VirusDB.GetFullVersion();
                DBSlice totalDbSlice = isFullUpdate ?
this.WebManager.GetCurrentUpdate(localDbVer) :
this.WebManager.GetFullUpdate(localDbVer);
                UpdateLog log this.VirusDB.UpdateDBWithLog(totalDbSlice);
                this.Logger.Report("Update", "Base updated", UpdateLog);
                this.DBVersion = log.Version;
```

```
            }
            catch
            {
                this.Logger.Warn("Error", "Update", ex.Message);
                this.ShowErrorNotification(ex.Message);
            }
            finally
            {
                SystemLog log = this.Logger.Report("Update", "Base updated",
UpdateLog);
                this.WebManager.UpdateStatus(log);
            }
        }

        private void GetBaseConfigs()
        {
            this.WebManager.GetBaseConfigs();
        }

        private void ShowSchedule()
        {
            ScheduleWindow w = new ScheduleWindow();
            w.VirusDB = this.VirusDB;
            w.ClientID = this.ClientID;
            w.ShowDialog();
        }

        private void ShowChart()
        {
            ChartWindow chart = new ChartWindow();

            ChartData data = this.Logger.GetChartData();

            chart.ChartData = data;
```

```
            chart.ShowDialog();
        }

        private void Scan(ScanType type, bool isHide = false, string file =
"")
        {
            try
            {
                bool hasVersion =
string.IsNullOrEmpty(this.VirusDB.GetFullVersion());
                if (!hasVersion)
                {
                    throw new Exception("No database available!");
                }

                ScanLog log;
                switch (type)
                {
                    case ScanType.Full:
                        log = this.Scanner.RunFullScan();
                        break;
                    case ScanType.Custom:
                        log = this.Scanner.RunCustomScan(file);
                        break;
                    case ScanType.File:
                        log = this.Scanner.RunCustomScan(file);
                        break;
                }

                SystemLog log = this.Logger.Report("Scan completed", "Scan",
log.GetShortForm);
                this.WebManager.UpdateStatus(log);
            }
            catch
            {
```

```
                this.Logger.Warn("Error", "Scan", ex.Message);
                this.ShowScanErrorNotification(ex.Message);

            }
            finally
            {

                if (!isHide)
                {
                    this.ShowScannerLog(log);
                }
            }
        }

        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            this.DisconnectHost();
            this.WebManager.SaveConfigs(true, true);
        }

        private void BaseInitialize()
        {
            this.WebManager = new WC();

            this.VirusDB = new VirusDBComponent();
            this.VirusDB.LoadLastSavedBase();
            this.Scanner = DetectComponent(true, true,);
            this.Logger = new LoggerComponent();

            this.WebManager.LoadSettings(null, "");
        }

            // . . .
```

```
    }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Base config-->
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2"
/>
    </startup>
    <system.serviceModel>
        <bindings>
            <netTcpBinding>

                <binding name="NetTcpBinding_IService">
                    <security>
                        <transport sslProtocols="None" />
                    </security>
                </binding>

            </netTcpBinding>
        </bindings>
        <client>

            <endpoint address="net.tcp://192.168.0.189:8302"
binding="netTcpBinding"
                bindingConfiguration="NetTcpBinding_IService"
contract="Service.IService"
                name="NetTcpBinding_IService">
                <identity>
                    <userPrincipalName value="ALEX-LAPTOP\User" />
                </identity>
            </endpoint>
        </client>
    </system.serviceModel>
```

```
</configuration> -


public MyServiceInstaller()
{
     serviceProcessInstaller1 = new ServiceProcessInstaller();
     serviceProcessInstaller1.Account = ServiceAccount.LocalSystem;

     serviceInstaller1 = new ServiceInstaller();
     serviceInstaller1.ServiceName = "ADS";
     serviceInstaller1.DisplayName = "ADS";
     serviceInstaller1.Description = "Antivirus Distributed Servise";
     serviceInstaller1.StartType = ServiceStartMode.Automatic;

     Installers.Add(serviceProcessInstaller1);
}


using System;
using System.ServiceModel;

using CSS = ADS.CentralizedServerService;

namespace ADS.CentralizedServerHostConsoleApp
{
    class HostDebugger
    {
        static void Main(string[] args)
        {
            using(var host = new ServiceHost(typeof(CSS.Service)))
            {
                string date = DateTime.Now.ToShortTimeString();
                host.Open();
```

```
                Console.WriteLine($"{date} : Centralized host has been
started");
                Console.ReadLine();
            }
        }
    }
}



protected override void OnStart(string[] args)
    {
      if(host != null)
        {
            host.Close();
        }

      string addrHTTP = "http://localhost:9001/IService";
      string addrTCP = "net.tcp://localhost:9002/MyService";

      Uri[] addrBase = { new Uri(addrHTTP), new Uri(addrTCP) };
      host = new ServiceHost(typeof(IService), addrBase);

      ServiceMetadataBehavior behavior = new ServiceMetadataBehavior();
      host.Description.Behaviors.Add(behavior);

      BasicHttpBinding bindingHttp = new BasicHttpBinding();
      host.AddServiceEndpoint(typeof(.IService), bindingHttp, addrHTTP);
      host.AddServiceEndpoint(typeof(IMetadataExchange),
MetadataExchangeBindings.CreateMexHttpBinding(), "mex");

      NetTcpBinding bindingTcp = new NetTcpBinding();

      bindingTcp.Security.Mode = SecurityMode.Transport;
      bindingTcp.Security.Transport.ClientCredentialType =
TcpClientCredentialType.Windows;
```

```
      bindingTcp.Security.Message.ClientCredentialType =
MessageCredentialType.Windows;

      bindingTcp.Security.Transport.ProtectionLevel =
System.Net.Security.ProtectionLevel.EncryptAndSign;



      host.AddServiceEndpoint(typeof(IService), bindingTcp, addr_TCP);

      host.AddServiceEndpoint(typeof(IMetadataExchange),
MetadataExchangeBindings.CreateMexTcpBinding(), "mex");



      host.Open();

    }




    protected override void OnStop()

    {

      if (this.host != null)

      {

        this.host.Close();

        this.host = null;

      }

    }



public class VirusData : Virus {

    private string id;

    private string name;

    private string signature;

    private string[] tokens;

    private string tDat;

    private float mark;

    private float markAV;


      // only use for comparator logic

      private string groupID;
```

```
public VirusData(Virus v, string[] tokens, string tDat, MarkData m) {
        this.id = v.id;
    this.name = v.FullName;
    this.signature = v.Sign;

        this.tokens = v.info ? v.info.keys : ["-"];
        this.tDat = tDat;

        this.mark = this.m.value();
        this.markAV = this.m.averageData.mark();

        // only use for comparator logic
        this.groupID = v.group ? v.group.id : "host-detected"
}

  public string ID
  {
        get
        {
              return this.id;
        }
  }

  public string Name
  {
        get
        {
              return this.name;
        }
  }

  public string Signature
  {
        get
```

```
        {
            return this.signature;
        }
    }

    public string[] Tokens
    {
        get
        {
            return this.tokens;
        }
    }

    public string TokensData
    {
        get
        {
            return $"{tDat}/{id}/sample";
        }
    }

    public string Group
    {
        get
        {
            return this.groupID;
        }
    }

    public int CompareTo(float mark) {
      int res = (int) (this.mark - mark);
      return res;
}
```

```
    private void calculateMark_AV(int length){
        if(length > 0)
            {
            this.markAV = this.mark/length;
        }
        else{
            this.markAV = 0;
        }
    }

      // . . .
}
```

ДОДАТОК Б

(обов'язковий)

ТЕЗИ ДО ДИПЛОМНОЇ РОБОТИ

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XII всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2020»

9-10 листопада 2020

Хмельницький 2020

УДК 004.4

Шевцов О. О., Савенко О. С.

Хмельницький національний університет

РОЗПОДІЛЕНА СИСТЕМА ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ МЕРЕЖАХ НА ОСНОВІ БАЄСОВСЬКОЇ МЕРЕЖІ

Розглянуто прикладні аспекти розробки та реалізації розподіленої системи виявлення зловмисного програмного забезпечення в локальній корпоративній мережі на основі Баєсовської мережі. Представлене програмне забезпечення забезпечує ефективне виявлення різного зловмисного програмного забезпечення та використовує алгоритм навчання Баєсовської мережі для покращення результатів розпізнавання загрози.

Applied aspects of development and implementation of a distributed system for detecting malicious software in a local corporate network based on the Bayesian network are considered. The presented software provides effective detection of various malicious software and uses the Bayesian network training algorithm to improve the threat detection results.

У зв'язку з бурхливим розвитком інформаційних технологій та поширенням комп'ютерних систем та мереж у різних сферах, постійно зростає кількість різного зловмисного програмного забезпечення. Воно постійно модифікується, ускладнюється, стає більш прихованим та набуває ефективності обходження антивірусних систем. Все це відповідно вимагає вдосконалення існуючих програмних засобів, методик і алгоритмів детектування а також протидії існуючим кіберзагрозам та зловмисному програмному забезпеченню в цілому. Враховуючи те, що в основному воно націлено на використання в комп'ютерних мережах – що є суттєвою проблемою як для окремих користувачів, так і для великих корпорацій [1]. Слід також приділити належну увагу використанню та інтеграції різних методів машинного навчання, таких як штучні імунні системи, нейронні мережі або Баєсовська мережа, щоб по можливості бути на крок попереду кіберзлочинців.

Отже, метою роботи є розробка програмного комплекса, що реалізує розподілену систему, завданням якої є виявлення зловмисного програмного забезпечення в локальних мережах на основі Баєсовської мережі. Відповідно завданням данного програмного забезпечення буде виявлення зловмисного програмного забезпечення різного плану в локальній корпоративній мережі.

Баєсовська мережа являє собою направлений ациклічний граф, у якого кожній вершині відповідає випадкова змінна. З математичної точки зору – це модель для представлення ймовірностних залежностей або відсутності цих

залежностей. При цьому зв'язок А –> В є наслідком коли подія А є причиною появи В, тобто впливає на В. Використання Баєсовської мережі має наступні переваги:

- Проста побудуви, інтерпритації та аналізу моделі
- Робота з неповними даними
- Можливість навчання в процесі роботи при малих затратах ресурсів [2]

Для навчання Баєсовської мережі існує досить багато різних алгоритмів таких як метод Монте-Карло або метод градієнтного підйому. У даному програмному комплексі використовується ЕМ-алгоритм – через його можливість реалізації різними мовами програмування та лінійне зростання важкості алгоритму при збільшенні об'єму даних, що добре для швидкодії програмного комплекса та використання системних ресурсів.[3] ЕМ-алгоритм використовується для знаходження оцінок максимальної правдоподібності параметрів ймовірнісних моделей, коли модель залежить від деяких прихованих змінних, які раніше були відсутні. Кожна ітерація алгориму скалається з двох кроків:

1) Е-крок (expectation step)
2) М-крок (maximization step)

На Е-кроці алгоритму обчислюється очікуване значення функції правдоподібності, але при цьому приховані змінні не розглядаються як спостережувані. Відповідно на М-кроці обчислюється оцінка максимальної правдоподібності. Таким чином збільшується очікувана правдоподібність, що обчислюється на Е-кроці. Потім це розраховане значення використовується для Е-кроку на наступній ітерації, та алгоритм виконується далі до збіжності. Якщо це виразити математично: у нас є набор даних Y, з них X вже спостережувались, а Z – ні (отже ці дані вважаються прихованими). Відповідно $Y = X\ U\ Zh$ – значення прихованої змінної. Перед самим початком роботи алгоритму вона отримає певне початкове значення. Далі на Е-кроці обраховується умовне математичне очікування $Q(h)$ логарифма правдоподібності набору змінних від h:

$$Q(h) = E\ [ln\ p(Y \mid h) \mid X\ ]$$

Фактично на цьому кроці йде обрахування усіх очікуванних змінних по поточному значенню параметра h, а вже на М-кроці обраховується

$$h_1 = arg\ max_h\ Q(h)$$

Тобто знаходиться наступне приближене значення параметра h при значенні $Q(h)$, що отримоно на Е-кроці. Алгоритм повторюється до тих пір поки h_n не зійдеться.

Таким чином за допомогою ЕМ-алгоритму знаходиться нова оцінка максимальної правдоподібності параметров моделі на основі вибіркових даних. Відповідно чим більше спроб використовується для навчання Баєсовської мережі – тим точніше вона зможе визначити зловмисне програмне забезпечення.

Розроблена розподілена система відповідає усім основним вимогам, таким як: прозорість (приховування свого розподілення процесів та ресурсів від користувача), відкритість (побудована на відкритих стандартах), можливість

ДОДАТОК В

(обов'язковий)

ПРЕЗЕНТАЦІЯ ДО ДИПЛОМНОЇ РОБОТИ

Розподілена система виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах на основі баєсівської мережі

Автор: Шевцов О.О.

Керівник: Савенко О.С.

Метою дипломної роботи є: розробка розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах на основі баєсівської мережі.

Задачі дослідження:

- Розглянути особливості та методи побудови сучасних розподілених інформаційних та обчислювальних систем та антивірусних засобів та рішень;
- Провести аналіз особливостей, переваг та недоліків існуючих рішень;
- Ознайомитись з баєсівською мережею, а також особливостями її використання в антивірусних системах та системах виявлення вторгнень;
- Розробити та реалізувати розподілену обчислювальну антивірусну систему з інтеграцією баєсівської мережі для виявлення зловмисного вірусного програмного забезпечення в локальних мережах;
- Провести аналіз та дослідження ефективності розробленої антивірусної системи у поєднанні з баєсівською мережею;

Наукова новизна отриманих результатів полягає у:

- Удосконаленні розподіленої системи для виявлення зловмисного програмного забезпечення на основі мережі Баєса, що на відміну відомих спрощує виявлення нових типів вірусних загроз

Практичне значення одержаних результатів:

- На основі проведеного дослідження розроблена антивірусна розподілена система, що спрощує виявлення вірусних загроз

Аналіз предметної області та постановка задачі

Переваги розподілених систем

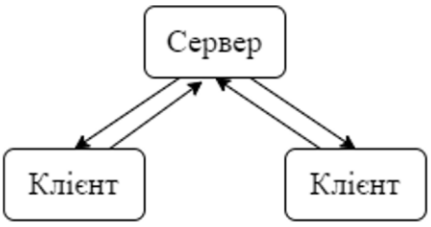
- Інкапсуляція від користувача всіх нюансів роботи
- Можливість дублювання ресурсу
- Сумісне використання апаратної та програмної частини комп'ютера
- Можливість легкого масштабування та додавання нових ресурсів

Недоліки:

- Важкість проектування, розробки та обслуговування

Архітектура розподіленої системи

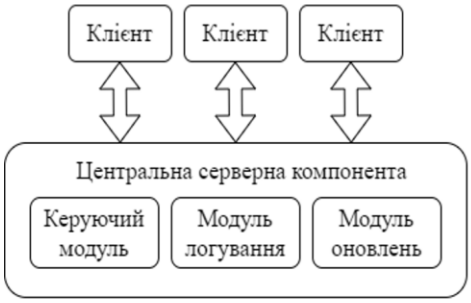
- Розподілена система являє собою багатокомп'ютерну систему, поєднану однією мережею та ПЗ, що об'єднує ці розрізнені комп'ютери в єдину РС, з метою вирішення закладених задач
- Розроблена РС має клієнт-серверну сервіс-орієнтовану архітектуру



Архітектура центральної керуючої компоненти

Серверна компонента складається з таких основних програмних частин:

- Модуль керування мережею
- Модуль оновлень
- Модуль логування



Архітектура клієнтських компонент

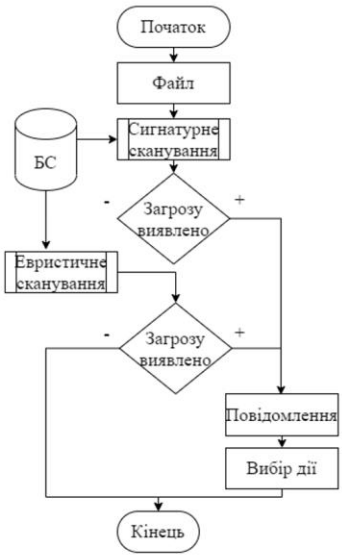
Клієнтська компонента складається з таких програмних частин:

- Модуль сканування
- Модуль роботи з вірусною базою
- Модуль користувацького інтерфейсу
- Модуль взаємодії з мережею
- Модуль логування

Баєсовська мережа у розподіленій системі

Алгоритм роботи сканування з використанням Баєсівської мережі:

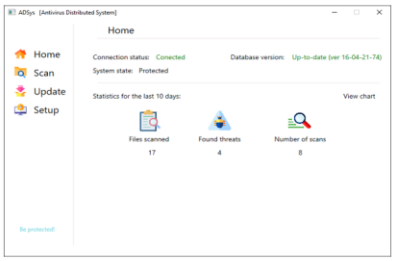
- Обирається файл для сканування
- Проводиться сигнатурне сканування
- Проводиться евристичне сканування
- Надається повідомлення про наявність вірусної загрози
- Результати сканування відправляються на сервер



Реалізація системи

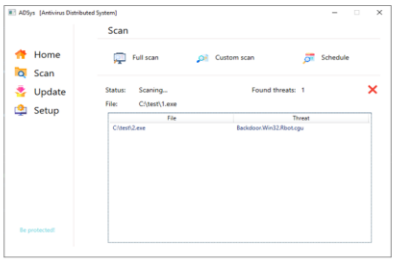
Основні 4 розділи:

- Home
- Scan
- Update
- Setup



Користувачу доступно три варіанти сканування:

- За вибором певного файлу чи каталогу у файловій системі
- Повне сканування
- Заплановане сканування



Експерименти

№ кейсу	Кількість вузлів	Розмір бази	Використання баєсівського сканування	Зв'язок з центром	Всього загроз	Загроз виявлено
1	1	мала	ні	так	7	2
2	1	мала	так	так	7	3
3	1	велика	ні	так	7	4
4	1	велика	так	так	7	6
5	5	мала	ні	так	7	2/2/2/2/2
6	5	велика	ні	так	7	4/4/4/4/4
7	5	мала	так	так	7	2/2/2/2/2
8	5	велика	так	так	7	6/6/6/6/6
9	5	мала	Тільки на першому вузлі	так	7	3/2/2/3/3
10	5	велика	Тільки на першому вузлі	так	7	6/4/4/6/6
11	1	велика	так	ні	7	6

Висновки

Отримані наступні результати роботи:

- Встановлено, що виявлення ЗПЗ у локальних комп'ютерних мережах згідно дослідження може бути виявлено за допомогою АРС з використанням баєсовської мережі
- Удосконалено архітектуру РС, в якій на відміну від рішень, що використовуються, покращено внутрішню взаємодію центру та компонентів системи
- Розроблений метод, що організовує та підтримує цілісність РС, враховує поточні стани компонентів, а також координує її архітектуру і може перебудовувати систему в залежності від ситуацій на вузлах і впливів ЗПЗ
- Проведена розробка системи оцінки ефективності використаних рішень і розроблено програмне забезпечення для розробленої АРС з використанням БМ. Проведено експериментальні дослідження з розробленою АРС, що підтвердили ефективність запропонованих рішень

За темою дипломної роботи опублікована одна стаття у збірнику матеріалів конференції «Збірник наукових праць Конференції АПКН-2020».

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1007498487

Дата перевірки:
23.04.2021 18:45:12 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
23.04.2021 18:46:43 EEST

ID користувача:
100005591

Назва документа: Розподілена система виявлення

Кількість сторінок: 72 Кількість слів: 14436 Кількість символів: 106682 Розмір файлу: 574.07 KB ID файлу: 100762158

0.27%
Схожість

Найбільша схожість: 0.16% з Інтернет-джерелом (https://megapredmet.ru/1-81668.html)

0.16% Джерела з Інтернету	2	Сторінка 74
0.11% Джерела з Бібліотеки	2	Сторінка 74

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 7%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 89440 Название: Розподілена система виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах на основі баєсовської мережі Добавлено в БД: 2021-04-26 Авторы: Шевцов О.О. Руководители: Савенко О.С. Консультанты: Оппоненты:	Символы	Лексемы	Символы	Лексемы
	89089	1419	445 (0%)	9 (1%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник студент групи КІ2м-19-1 Шевцов О. О.

Тема Розподілена система виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах

Спеціальність 123 – Комп'ютерна інженерія

Обсяг дипломної роботи:

Кількість листів креслень 0; кількість сторінок записки 118

1.Короткий зміст ДР та прийнятих рішень Представлена робота присвячена актуальній темі в області виявлення зловмисного програмного забезпечення та антивірусних розподілених систем і складається з наступних розділів: вступ, аналіз предметної області та постановка задачі, архітектура розподіленої системи, Баєсовська мережа у розподіленій системі, реалізація системи та експерименти, висновки, додатки.

2. Висновок про відповідність ДР поставленому завданню Магістерська кваліфікаційна робота виконана у відповідності з завданням із дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі студент провів детальний аналіз предметної області, було розглянуто розподілені системи, їх переваги та недоліки, особливості і вимоги при розробці, на основі цього довів актуальність роботи і визначив вимоги для створюваної системи. В другому розділі на основі досліджених джерел було приведено архітектуру розподіленої системи, її апаратну та програмну складову, детально розглянуто всі компоненти системи і їх взаємодія. Відповідно в третьому була розглянута Баєсівська мережа, особливості її побудови, можливості використання у розподілених системах для виявленні кіберзагроз, описано інтеграцію в існуючу систему та приведено алгоритм роботи. У четвертому розділі автором описана розподілена система та користувацькі інтерфейси, наведено експерименти, тестові кейси для тестування роботи системи, проаналізовано результати.

4. Позитивні сторони роботи До позитивних сторін роботи слід віднести актуальність даного направлення дослідження, деталізацію аналізу усіх розглянутих стратегій вирішення проблеми та поглиблене опрацювання всіх аспектів реалізації з практичним використанням запропонованого рішення.

5. Негативні сторони роботи До негативних сторін роботи слід віднести недоліки по оформленню представленого матеріалу, що були виправлені.

6. Оцінка графічного оформлення та пояснювальної записки роботи Дані матеріали роботи є структурованими у чіткій та логічній формі та відображають послідовність виконання поставлених завдань. І хоча й в них було знайдено декілька стилістичних та орфографічних помилок, вони були пізніше усунені. Тому дане виконання пояснювальної записки та графічного оформлення заслуговує оцінки «добре».

7. Відгук про роботу в цілому Загалом, зміст представленої роботи в повній мірі розкриває обрану тему. Дослідження, проведені в матеріалах є достатньо аргументованими. Прослідковуються високі теоретичні та практичні рівні у даному виконанні. Результатом проведення досліджень стали відповідні висновки і конкретні пропозиції щодо вдосконалення процесу виявлення зловмисного програмного забезпечення у локальних комп'ютерних мережах з використанням Баєсовської мережі.

8. Інші зауваження

9. Оцінка дипломної роботи Робота заслуговує оцінки «добре», а її автор – присвоєння кваліфікації «магістра» з комп'ютерної інженерії.

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Кльоц Юрій Павлович, доктор технічних наук, доцент кафедри кібербезпеки та комп'ютерних систем і мереж ХНУ

"05" 05 2021 р. _____
(підпис)

Завідувачу кафедри КІСП
д-р.техн.наук, проф. Говорущенко Т. О.

Шевцов Олександр Олександрович
ПІБ здобувача вищої освіти

ФПКТС, 2 курсу, групи КІ2м-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

23.04.2021
дата


підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Розподілена система виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах на основі баєсівської мережі

Автор: Шевцов Олександр Олександрович

Спеціальність: 123 – Компютерна інженерія та програмування

Освітня програма: освітньо-наукова

Науковий керівник: О. С. Савенко, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу алгоритмів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 2 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності символів, які є елементами формули і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 0.27% і адресується до 80 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. С. Савенко

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко