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

ДИПЛОМНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Самоорганізована розподілена система виявлення аномалій в комп'ютерних системах на основі методу головних компонент»

ДРКІСПр 15080.15.14.01 ПЗ

Виконав: студент 2 курсу, група КІ2м-19-1 Савенко Б.О.
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент Нічепорук А.О.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІСП, д.т.н., проф.
Т. О. Говорущенко
05 05 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-наукова програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Т.О.Говорущенко

"01" 09 2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Савенко Богдан Олегович
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод оцінювання інтерфейсу користувача програмних систем на відповідність гештальт-принципам
Керівник проекту (роботи) Нічепорук А.О., к.т.н., доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
Затверджена наказом ректора університету від 15.01.2021 р. № 7
2. Строк подання студентом проекту (роботи) на кафедру 06.05.2021 р.
3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
аналіз відомих методів виявлення аномалій в комп'ютерних системах;
проєктування архітектури розподіленої системи та методу підтримки її цілісності;
метод виявлення аномалій в комп'ютерних системах;
дослідження ефективності запропонованих рішень, реалізація розподіленої системи.
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 «_01_»_____09_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	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	05.10.2020	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	05.11.2020	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	05.12.2020	виконано
5	Робота над науковою статтею та тезами	05.01.2021	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2021	виконано
7	Робота над розділом 4 – проектування та розробка розподіленої системи для вирішення поставленої задачі, експериментальна частина	05.04.2021	виконано
8	Оформлення пояснювальної записки згідно вимог	15.04.2021	виконано
9	Попередній захист ДРМ	25.04.2021	виконано
10	Захист ДРМ на засіданні ЕК	До 31.05.2021	

Студент _____ Богдан САВЕНКО
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Андрій НІЧЕПОРУК
Підпис ініціали, прізвище

РЕФЕРАТ

Тема дипломної роботи: «Самоорганізована розподілена система виявлення аномалій в комп’ютерних системах на основі методу головних компонент»

Автор роботи: Савенко Богдан Олегович

Керівник роботи: Нічепорук Андрій Олександрович

Пояснювальна записка: 107 ст., 12 рис., 3 табл., 1 дод., 68 джерел.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ: аномалії, метод головних компонент, розподілена система, комп’ютерна система, комп’ютерна мережа.

Об’єкт дослідження є процес функціонування самоорганізованих розподілених систем виявлення аномалій в комп’ютерних системах.

Предмет дослідження є методи і засоби створення самоорганізованих розподілених систем виявлення аномалій в комп’ютерних системах.

Метою роботи є покращення ефективності виявлення аномалій в комп’ютерних системах при використанні самоорганізованих розподілених систем.

Для розв’язання поставлених задач використовувалися методи теорії розподілених систем; теорій множин, графів та штучного інтелекту; головних компонент для виявлення аномалій; теорії комп’ютерних мереж для організації функціонування розподіленої системи.

Наукова новизна одержаних результатів полягає в наступному:

1) удосконалено архітектуру розподіленої системи виявлення аномалій в комп’ютерних системах, в якій синтезовано вимоги самоорганізованості, централізованості, розподіленості, багаторівневості, і на відміну від відомих рішень, дало змогу удосконалити її внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії;

2) розроблено новий метод підтримки цілісності архітектури

самоорганізованої розподіленої системи в локальних комп'ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп'ютерних атак;

3) удосконалено метод виявлення аномалії згідно методу головних компонент в комп'ютерних системах в мережі, який надав змогу застосовувати його не до однієї комп'ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп'ютерних системах в мережі, що на відміну від відомих рішень, при застосуванні надав змогу скоротити обсяг даних і відповідно прискорити їх обмін між компонентами системи.

Практичне значення одержаних результатів полягає у розробленій архітектурі і компонентах розподіленої системи виявлення аномалій в комп'ютерних системах, в якій синтезовано вимоги самоорганізованості, централізованості, розподіленості, багаторівневості та на її основі створено самоорганізовану розподілену систему. Здійснена розробка методики оцінки ефективності запропонованих рішень для розробленої самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах, яка була застосована до неї для підтвердження можливості реалізації запропонованих рішень. Проведені експериментальні дослідження з розробленою реалізацією самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах підтвердили ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп'ютерній мережі.

Теоретичні та практичні результати роботи впроваджено при виконанні науково-дослідних робіт, які виконувались в Хмельницькому національному університеті.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АПЗ - антивірусне програмне забезпечення

БД - база даних

ЦПР - центр прийняття рішень

СРС – самоорганізована розподілена система

ОС - операційна система

ПЗ - програмне забезпечення

СВВ - система виявлення вторгнень

DDoS - Distributed Denial of Service (розподілена відмова в обслуговуванні)

IDS - система виявлення вторгнень

КС - комп’ютерні системи

ВСТУП

Актуальність роботи. Актуальним напрямом, який потребує напрацювання методів і засобів протидії зловмисним діям, є напрям пов’язаний з функціонуванням комп’ютерних мереж, бо вони використовуються практично в усіх підприємствах, організаціях та установах. Проблеми в їх функціонуванні, які викликані впливом зловмисного програмного забезпечення та комп’ютерних атак, а ще гірше, що полягає в приховуванні присутності зловмисника, можуть призвести до фінансових втрат підприємствами, організаціями та установами.

Дослідження зловмисних проявів в корпоративних та локальних мережах можуть бути проведені з використанням апарату математичної статистики. В корпоративних та локальних мережах підприємств, організацій та установ може перебувати велика кількість комп’ютерів і для дослідження процесів, які протікають в них, в тому числі і зловмисних, потрібні ефективні методи та відповідні засоби опрацювання отриманих даних про події. Ефективність протидії зловмисним проявам досягається за рахунок комплексного підходу орієнтованого на інтеграцію методів виявлення та систем, в яких вони реалізовані. Для зловмисників такі підходи суттєво ускладнюють досягнення результативності.

Взагалі поява в комп’ютерах чи комп’ютерних системах і мережах зловмисного програмного забезпечення або комп’ютерних атак, можливо зловмисних дій користувачів, крім безпосереднього виникнення технічних несправностей апаратних пристроїв, вичерпує множину об’єктів, які своєю нестандартною поведінкою можуть привернути до себе увагу. Для обробки подій потрібна розподілена система, яка б збирала та опрацьовувала данні і видавала б результат з дослідження виявлення. Враховуючи потребу опрацьовування даних оперативно і без втручання людини, то така розподілена система повинна бути самоорганізованою.

Розвитком елементів теорії розподілених систем різного призначення активно займаються Бернс Б., Луцький Г. М., Мельник А. О., Марковський Г., Мухін В. Є. Фахівцями, які займаються застосуванням методів виявлення аномалії

є Корченко А. О. [6], Wawryn, K. [4], Mohiuddin Ahmed [10], Xiang Yu [12], Liu, L. [15], Xiaodan Xu [16], Mukrimah Nawir [21].

В роботі пропонується використання самоорганізованих розподілених систем, розроблених згідно принципів централізації та самоорганізації, для виявлення аномалій у комп'ютерних системах.

Зв'язок роботи з науковими програмами, планами, темами. Дослідження, представлені у кваліфікаційній роботі, 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 роботи є покращення ефективності виявлення аномалій в комп'ютерних системах при використанні самоорганізованих розподілених систем.

Задачі дослідження формулюються в роботі наступним чином:

1) дослідити особливості прояву аномалії в комп'ютерних системах за умов функціонування зловмисного програмного забезпечення і здійснення комп'ютерних атак в локальних комп'ютерних мережах та проаналізувати сучасні методи виявлення аномалії, їх особливості та методи створення і архітектури розподілених систем;

2) удосконалити модель архітектури розподіленої системи виявлення аномалії в комп'ютерних системах, в якій синтезувати вимоги розподіленості, централізованості та самоорганізованості, для створення на її основі розподілених систем та їх компонентів, що функціонуватимуть під керівництвом одного центру розподіленого між різними компонентами і самостійно прийматимуть рішення про наявність аномалії;

3) розробити метод підтримки цілісності самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах для підтримки її цілісності, на основі якого система змогла б самостійно змінювати свою архітектуру без

втручання користувача, а також визначати стратегію своєї подальшої роботи;

4) удосконалити метод централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент для зменшення розмірності з моменту отримання та надсилання даних в центр прийняття рішень системи;

5) розробити програмне забезпечення самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах для підтвердження можливості практичного створення таких систем згідно запропонованих результатів роботи та використання в експериментальних дослідженнях для порівняння з відомими системами виявлення.

Об'єкт дослідження – процес функціонування самоорганізованих розподілених систем виявлення аномалій в комп'ютерних системах.

Предмет дослідження – методи і засоби створення самоорганізованих розподілених систем виявлення аномалій в комп'ютерних системах.

Методи дослідження. Для досягнення поставлених задач використано основні положення:

1) теорії розподілених систем, на основі якої можуть бути розроблені програмні та апаратно-програмні засоби;

2) теорій множин, графів та штучного інтелекту;

4) методи головних компонент для виявлення аномалій;

5) теорії комп'ютерних мереж для організації функціонування розподіленої системи.

Наукова новизна одержаних результатів полягає в наступному:

1) удосконалено архітектуру розподіленої системи виявлення аномалій в комп'ютерних системах, в якій синтезовано вимоги самоорганізованості, централізованості, розподіленості, багаторівневості, і на відміну від відомих рішень, дало змогу удосконалити її внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії;

2) розроблено новий метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп’ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп’ютерних атак;

3) удосконалено метод виявлення аномалії згідно методу головних компонент в комп’ютерних системах в мережі, який надав змогу застосовувати його не до однієї комп’ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп’ютерних системах в мережі, що на відміну від відомих рішень, при застосуванні надав змогу скоротити обсяг даних і відповідно прискорити їх обмін між компонентами системи.

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій. Наукові положення, висновки і рекомендації дипломної роботи магістра обґрунтовані коректним та доцільним використанням математичного апарату, алгоритмами здійснення виявлення аномалій, успішною реалізацією розробленої розподіленої системи виявлення аномілій в комп’ютерних системах, ефективним практичним впровадженням результатів, яке продемонструвало відповідність теоретичних розробок з реальними результатами застосування.

Практичне значення одержаних результатів. У результаті виконаного дослідження розроблено архітектуру і компоненти розподіленої системи виявлення аномалій в комп’ютерних системах, в якій синтезовано вимоги самоорганізованості, централізованості, розподіленості та на її основі створено самоорганізовану розподілену систему. Здійснена розробка методики оцінки ефективності запропонованих рішень для розробленої самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах, яка була застосована до неї для підтвердження можливості реалізації запропонованих рішень. Проведені експериментальні дослідження з розробленою реалізацією

самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах підтвердили ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп'ютерній мережі.

Теоретичні та практичні результати роботи впроваджено при виконанні науково-дослідних робіт, які виконувались в Хмельницькому національному університеті.

Особистий внесок здобувача. Всі основні результати дослідження, які представлені до захисту дипломної роботи, одержані автором особисто. В роботах, опублікованих у співавторстві, автору належать основні ідеї, теоретична та практична розробка положень, відображених у характеристиці наукової новизни отриманих результатів, а саме: [1, 3] – запропоновано використання розподіленої системи, в яу імплементовано метод виявлення незадокументованих закладок в програмному забезпеченні згідно аналізу виявлення аномалій; [2, 68] – розроблено метод підтримки цілісності компонентів розподіленої системи в комп'ютерній мережі.

Апробація результатів дисертації. Основні положення та результати проведених у дипломній роботі досліджень доповідалися та обговорювалися на двох міжнародних наукових конференціях: the 1nd International Workshop on Intelligent Information Technologies & Systems of Information Security (Khmelnytskyi, Ukraine, June 10-12, 2020), the 2nd International Workshop on Intelligent Information Technologies & Systems of Information Security (Khmelnytskyi, Ukraine, March 24–26, 2021).

Публікації. За результатами проведених досліджень основні наукові результати опубліковано у 1 науковій статті [1] у фаховому науковому виданні України. Апробація засвідчена публікаціями 3 праць в матеріалах міжнародної конференції [2, 3, 68], дві з яких індексовані у наукометричній базі Scopus.

Структура кваліфікаційної роботи. Дипломна робота складається з анотації, вступу, чотирьох розділів, висновків, списку використаних джерел з 68 найменувань на 7 сторінках та одного додатку на 88 сторінках. Загальний обсяг роботи становить 93 сторінки, з них 106 сторінок основного тексту, 12 рисунків, 3 таблиці.

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ І ЗАСОБІВ ВИЯВЛЕННЯ АНОМАЛІЙ В КОМП'ЮТЕРНИХ СИСТЕМАХ

1.1 Огляд та поняття виявлення аномалій в комп'ютерних системах

Комп'ютерні системи (КС) продовжують активно використовуватись в усіх сферах діяльності людини. Вони дозволяють суттєво підвищити продуктивність праці та автоматизувати багато складних процесів і це покращує перспективи їх використання в майбутньому.

Програмне забезпечення, яке використовується в КС, виконує дуже важливу роль та забезпечує ефективне вирішення задач. Але із зростанням актуальності задач, при розв'язанні яких використовують КС, зловмисники спрямовують свої зусилля на взяття під контроль КС шляхом впливу на їх програмне забезпечення. Для цього вони застосовують різноманітні атаки на КС [1-4]. Частина з них стає успішною і цим створює проблеми користувачам КС. Для важливих критичних сфер [2] діяльності людини втрата контролю над КС стає серйозною проблемою з відповідними катастрофічними наслідками. Тому, проблемі побудови ефективних систем протидії зловмисним діям спрямованим на КС користувачів все більше приділяють уваги дослідники та розробники відповідних методів і засобів [2-7].

Комп'ютери користувачів, комп'ютерні системи та мережі стали широким полем для проявів зловмисників і залишатимуться ще довго. Для протидії їм важливим є продовження вже напрацьованих досліджень з урахуванням перспектив можливого розвитку зловмисного програмного забезпечення та комп'ютерних атак.

Провідними організаціями, які досліджують і публікують актуальну статистику поширення і створення нового зловмисного програмного забезпечення є Virus Bulleten [8] та AV-TEST [9]. Оприлюднені ними результати досліджень підтверджують постійне зростання кількості зловмисного програмного забезпечення. На рис. 1.1 представлено динаміку кількості зареєстрованого зловмисного програмного забезпечення на основі даних AV-TEST [9] за 2020 рік.

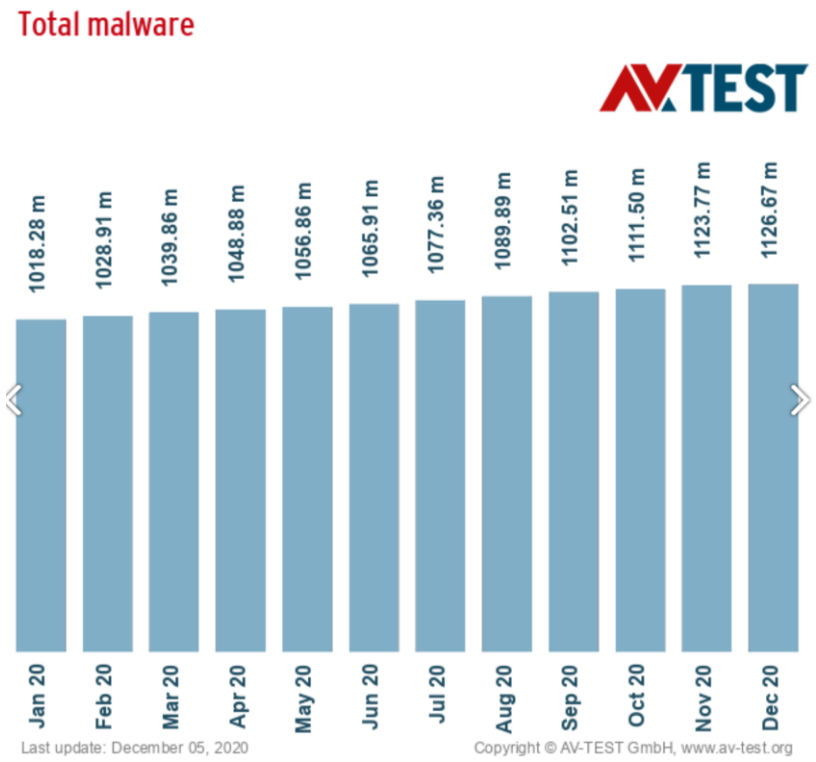


Рисунок 1.1 – Кількість зареєстрованого зловмисного програмного забезпечення за даними AV-TEST [9]

Одним з найбільш актуальних на сьогодні напрямом, який потребує активного напрацювання методів і засобів протидії зловмисним діям, є напрям пов’язаний з функціонуванням корпоративних та локальних мереж [1, 2], оскільки вони використовуються практично в усіх підприємствах, організаціях та установах. Виведення їх з ладу, а ще гірше, що полягає в приховуванні присутності зловмисника в мережі, може призвести до зупинки роботи підприємства, організації та установи або нанести їм значної фінансової, матеріальної, моральної та організаційної шкоди.

Для дослідження зловмисних проявів в корпоративних та локальних мережах може бути використаний апарат математичної статистики, в якому прояви зловмисної активності будуть відноситись до аномальної поведінки користувачів та досліджуватимуться відповідними статистичними методами. В корпоративних та локальних мережах підприємств, організацій та установ може перебувати велика кількість комп’ютерів і для дослідження процесів, які протікають в них, в тому

числі і зловмисних, потрібні ефективні методи та відповідні засоби опрацювання отриманих даних про події. Ефективність протидії зловмисним проявам досягається за рахунок комплексного підходу орієнтованого на інтеграцію методів виявлення та систем, в яких вони реалізовані. Для зловмисників такі підходи суттєво ускладнюють досягнення результативності.

Якщо розглядати саме комп'ютери в мережах, а не персональні домашні комп'ютери, то відповідно підбір стратегій і методів виявлення, а також і систем виявлення може різнитись. Це пов'язано з можливістю обробки типової інформації, що отримана не з одного комп'ютера, а з декількох. Її інформативність в цьому випадку буде суттєво більшою, ніж у випадку інформації з одного комп'ютера. У випадку одного персонального комп'ютера необхідним є збір статистичних даних певного визначеного набору специфічної інформації протягом тривалого часу, обробка цієї інформації для представлення образу «нормальної» поведінки або її порівняння з підозрілою активністю для формування висновку. У випадку методів і засобів для одного комп'ютера технології виявлення зловмисних дій можуть базуватись на моделі зловмисної поведінки або на порівнянні моделі з потоком подій, які протікатимуть в комп'ютері. Знання про моделі зловмисної поведінки заносяться в базу моделей і можуть бути або статичними сигнатурами або поведінковими сигнатурами. У другому випадку здійснюють формування моделі поведінки та її порівняння з потоком подій, які відбуваються в комп'ютері. Ці ж підходи та стратегії можуть бути використані окремо і для комп'ютерів, які під'єднані в мережу. Але вони можуть, також, комбінуватись з підходами базованими на технологіях виявлення аномальної поведінки. Для комп'ютерів, які під'єднані в мережу, виявлення зловмисної активності на основі дослідження аномальної поведінки може бути ефективніше, ніж підходи базовані тільки на основі моделей зловмисної поведінки. Оскільки, з появою нових моделей зловмисного програмного забезпечення, відомості про які не містяться в засобах виявлення, результат виявлення буде негативним. За період від появи нових моделей зловмисних і до оновлення засобів виявлення зловмисних дій може пройти небагато часу, але таке нове зловмисне програмне забезпечення може проникнути

в комп’ютер або в комп’ютери під’єднані в мережу. Тому, розглянемо технології виявлення зловмисного програмного забезпечення та комп’ютерних атак, які базуються на виявленні аномальної поведінки. Ця технологія використовує знання про нормальну поведінку, яких можна досягти побудувавши профіль користувачів комп’ютерів, під’єднаних до мережі, або ідентифікує аномальні входження в потік подій, які протікають в мережі.

Розглянемо детальніше поняття аномальної поведінки, її особливості з орієнтацією на предметну область та реалізацію в системах певного спеціалізованого типу.

Взагалі поява в комп’ютерах чи комп’ютерних системах і мережах зловмисного програмного забезпечення або комп’ютерних атак, можливо зловмисних дій користувачів, крім безпосередньо виникнення технічних несправностей апаратних пристроїв, вичерпує множину об’єктів, які своєю нестандартною поведінкою можуть привернути до себе увагу. Ці прояви віднесемо до аномальних проявів, виявлення яких досліджуватимемо. Аномалія в перекладі з грецької мови (ανωμαλια) означає ненормальність, неправильність, відхилення від норми, від загальної закономірності. Для формування повної групи подій розглядатимемо також нормальність або нормальну поведінку. Таким чином, процеси в комп’ютерах чи комп’ютерних системах і мережах, які викликані розв’язуванням певних задач класифікуватимемо за нормальною або аномальною поведінкою [3]. Задачі, які розв’язуються, можуть бути задані як користувачами з певною корисною метою автоматизації робіт, так і зловмисниками.

В роботі [4] авторами представлена система виявлення вторгнень, що базується на ідеях з імунної системи людини. Спеціальна штучна імунна система контролює локальну область, що містить критичні файли в операційній системі. Запропонований метод включає сканування файлів та перевірку наявності можливих змін, спричинених зловмисним програмним забезпеченням. Система складається з двох модулів: блок генерації рецепторів, який генерує рецептори, використовуючи оригінальний метод, базований на шаблонах, і блок виявлення аномалій. Аномалії, виявлені у файлах за допомогою раніше створених рецепторів,

повідомляються користувачеві. Впроваджена система та проведено експерименти для порівняння ефективності алгоритмів із ефективністю інших методів генерації рецепторів, який називають методом генерації випадкових рецепторів. У контрольованому середовищі тестування аномалії у вигляді зміненого байта програмного коду вводили в моніторингові програми. Реальні тести цієї системи проводились щодо її продуктивності та масштабованості. Крім цієї роботи, відомо багато ефективних спроб використати штучні імунні системи для виявлення аномалій в різних складових програмної частини КС.

В роботі [5] представлено штучну імунну систему (AIS), яка базується на біологічній імунній системі (BIS). AIS використовується для виявлення зловмисних програм. Найвідомішою моделлю AIS є алгоритм негативного відбору (NSA), і він може використовувати лише звичайні вибірки для тренування. Традиційні алгоритми негативного відбору (АНВ) генерують детектори на етапі навчання, а потім виявляють елементи аномалії на етапі тестування. У традиційних АНВ є певні недоліки. Зокрема, авторами роботи встановлено, що реальні програми часто змінюються, звичайні можуть змінюватися аномальними, і навпаки. Традиційні АНВ легко виробляють багато помилкових тривог та помилкових негативних наслідків у реальних застосунках. Традиційним АНВ бракує здатності до постійного навчання на етапі тестування, і дорого генерувати достатню кількість детекторів, щоб покрити загальний несамостійний простір у навчанні. Щоб подолати недоліки традиційних АНВ, до таких алгоритмів авторами роботи введено нову схему з адаптивним онлайн-навчанням, яка включає в себе побудову відповідного профілю системи, генерування нових детекторів, що закривають отвори несамостійного простору, видаляючи ті детектори, які лежать у самопросторі, зменшують кількість помилкових тривог, а коригування цих детекторів, які частково охоплюють власний простір, зменшує помилкові тривоги та збільшує швидкість виявлення. Таким чином, подальший розвиток застосування відомих методів виявлення аномалій є перспективним в напрямі інтеграції та комбінування сучасних методів виявлення аномалій.

В авторефератах [6, 7] дисертаційних робіт представлено розроблені методи ідентифікації аномальних станів для систем виявлення вторгнень та стратегії забезпечення функціональної стійкості розподілених інформаційних систем до кібернетичних загроз. Наукові результати цих робіт підтверджують доцільність дослідження аномальних станів для виявлення зловмисного програмного забезпечення та комп'ютерних атак в КС [6], а також застосування засобів, які базуються на використанні розподілених систем для забезпечення функціональної стійкості інформаційних систем [7].

Таким чином, виявлення зловмисного програмного забезпечення та комп'ютерних атак в КС на основі виявлення та встановлення аномальних проявів є перспективним напрямом досліджень. Крім того, застосування різноманітних методів виявлення аномалій до програмного забезпечення та особливо системних програм певного застосування в КС не завжди дозволяє досягти бажаного результату з виявлення зловмисного програмного забезпечення та комп'ютерних атак, тому проведення подальших досліджень з розробки нових підходів до виявлення на основі виявлення аномалій залишається актуальною науковою задачею. Для визначення стратегії її розв'язання першочергово необхідним є проведення аналізу відомих методів виявлення аномалій в КС з метою визначення їх переваг та недоліків, а також виділення нерозв'язаних підзадач, які впливають на досягнення ефективного результату з виявлення. Важливою частиною такого дослідження, яка суттєво впливає на ефективність і особливо достовірність виявлення зловмисного програмного забезпечення та комп'ютерних атак в КС, є засоби, які підтримуватимуть реальне застосування розроблених методів. Засоби, в яких реалізовані методи виявлення суттєво впливають на результат виявлення, тому необхідним є їх розробка інтегровано з імплементованими в них розробленими методами виявлення.

1.2. Відомі методи та засоби виявлення аномалій в комп’ютерних системах

1.2.1. Аналіз відомих методів та алгоритмів виявлення аномалій в комп’ютерних системах

Дослідженню аномалій в комп’ютерних системах для виявлення зловмисного програмного забезпечення і комп’ютерних атак приділяють увагу багато дослідників. Вони розробили багато різних методів виявлення [10-18]. Для розробки нових чи покращення відомих рішень з виявлення аномалій в КС потрібно встановити переваги та недоліків відомих розроблених методів. Виділення нерозв’язаних підзадач, що впливають на досягнення ефективного результату з виявлення, та стратегії з усунення частини недоліків відомих методів дозволять підвищити достовірність та покращити ефективність виявлення. Розглянемо відомі методи виявлення зловмисного програмного забезпечення і комп’ютерних атак в КС, які базуються на встановленні аномальних станів. До розгляду візьмемо ті з них, які запропоновані відомими дослідниками в цій галузі.

У роботі [10] представлений поглиблений аналіз чотирьох основних категорій методів виявлення аномалій, які включають класифікацію, статистику, теорію інформації та кластеризацію. Основна увага зосереджена на проблеми дослідження з наборами даних, що використовуються для виявлення вторгнень у мережу. Результати дослідження дають змогу пов’язати класифікацію, статистичну обробку даних та кластеризацію із поставленою науковою задачею в частині розробки застосовуваних методів при обробці вхідних даних з метою виявлення мережних аномалій.

Проблемі узгодження низьковимірних багатовимірних об’єктів до високомірних даних розглянуто в роботі [11] як з теоретичної, так і з обчислювальної точки зору. Оскільки набори даних стають більш неоднорідними та ускладненими, простори, які використовуються для їх апроксимації, повинні стати більш неоднорідними. В цій роботі відображеног результати роботи з переходами до змінених базисів, що є актуальним в розрізі поставленої задачі з

виявлення аномалій і зменшення розмірності даних. Також, проаналізована обчислювальна складність таких перетворень для оцінки необхідних обчислювальних ресурсів.

З розповсюдженням Інтернету речей через бездротові сенсорні мережі генерується величезна кількість даних датчиків з безпрецедентною швидкістю, що призводить до дуже великої кількості явної або неявної інформації [12]. При аналізі таких даних датчиків особливо важливо точно та ефективно виявляти не тільки окремі аномальні поведінки, але й аномальні події (тобто моделі поведінки). Однак більшість попередніх робіт були зосереджені лише на виявленні аномалій, водночас ігноруючи співвідношення між ними. Навіть у підходах, що враховують кореляцію між аномаліями, більшість ігнорує той факт, що аномалія стану даних датчиків змінюється з часом. У цій статті [12] запропоновано безконтрольний метод виявлення аномалій контексту в Інтернеті речей за допомогою бездротових сенсорних мереж, який враховує як статус динамічної аномалії, так і кореляцію між аномаліями, заснованими в контексті на їх просторових та часових сусідах. А, також, досліджено в роботі ефективність запропонованого методу в моделі виявлення аномалії. Внесення в роботу відомостей про аномалії і аномальні прояви є важливим з точки зору врахування динаміки отримання даних.

В роботі [13] процес виявлення несподіваних елементів або подій у наборах даних, які відрізняються від норми, розглядається як пошук аномалій. На відміну від стандартних задач класифікації, виявлення аномалій часто застосовується до немаркованих даних, беручи до уваги лише внутрішню структуру набору даних. Ця проблема відома як неконтрольоване виявлення аномалій і вирішується у багатьох практичних додатках, наприклад, у виявленні вторгнень у мережу, виявленні шахрайства, а також у галузі біології та медицини. У цій статті представлено результати здійсненої оцінки 19 різних алгоритмів виявлення аномалій на 10 різних наборах даних із декількох доменів додатків. Робота є важливою для досліджень безконтрольного виявлення аномалій.

Виявляти та обробляти аномалії для великих даних у режимі реального часу є складним завданням. Обсяг і швидкість даних у багатьох системах ускладнює

типовим алгоритмам масштабування та збереження своїх характеристик у реальному часі [14]. Поширеність даних у поєднанні з проблемою, що багато існуючих алгоритмів враховують лише зміст джерела даних, а не контент. Запропоновані в роботі рішення визначають контекст виявлення аномалій. Він складається з двох різних кроків: виявлення вмісту та виявлення контексту. Детектор вмісту використовується для визначення аномалій у режимі реального часу. Детектор контексту використовується для обрізання результатів детектора вмісту, виявляючи ті аномалії, які вважаються як змістовими, так і контекстно аномальними. Детектор контексту використовує концепцію профілів, які є групами аналогічно згрупованих точок даних, що генеруються багатовимірним алгоритмом кластеризації. Дослідження було оцінено на основі проведених експериментів для двох реальних наборів даних датчиків. Резуьтати цієї роботи [14] важливі в контексті важливості обробки контенту датчиків.

В роботі [15] пропонується поступовий метод безконтрольного виявлення аномалії, який дозволяє швидко аналізувати та обробляти великі дані в режимі реального часу. Оцінка набору даних під час експерименту показує, що метод зближується зі своїм автономним аналогом для нескінченно зростаючих потоків даних.

В роботі [16] проаналізовано відомі рішення з виявлення аномалій, особливо для даних із великими розмірами та змішаними типами, де виявлення аномальних моделей чи поведінки є нетривіальною роботою. В результаті важливість даної роботи в проведених дослідженнях відомих підходів і їх порівнянні, що дозволить врахувати ці результати при виборі перспективних рішень.

В роботі [17] зосереджено увагу на ранньому виявленні несподіваних спостережень у фізичній інфраструктурі, що має велике значення для запобігання поломки системи та подальших втрат. Однак сучасна техніка для виявлення аномалій в існуючій платформі моніторингу інфраструктури головним чином залежить від методу фіксованого порогу. Очевидним недоліком методу є те, що він, як правило, призводить до високого рівня помилкового виявлення. У цьому дослідженні підхід до виявлення статистичних аномалій запроваджено до

моніторингу фізичної інфраструктури. В роботі розглядаються три важливі типи аномалій, які зустрічаються на платформі моніторингу інфраструктури, а саме наївні точкові аномалії, контекстні аномалії точок та зміщення рівня. В роботі пропонується до застосування розроблений метод, заснований на моделі Гаусса, для виявлення зазначених трьох аномалій. Оскільки запропонований метод може ефективно виявляти лише наївні точкові аномалії; запропоновано вдосконалений підхід, що поєднує результати статистичних випробувань на вихідних та першовідмінних даних моніторингу. Оцінюються результати запропонованих методів на реальному наборі даних. Результати показують, що оптимізований підхід до виявлення аномалій має хорошу точність і може значно знизити швидкість неправильного виявлення. Отримані результати дають розуміння обробки трьох типів аномалій.

Однією із сучасних проблем виявлення аномалій є здатність виявляти і розрізняти як точкові, так і колективні аномалії в межах послідовності даних або часових рядів [18]. В роботі [18] розроблено метод та засоби, щоб надати користувачам вибір методів виявлення аномалій, і, зокрема, забезпечує реалізацію нещодавно запропонованого сімейства алгоритмів виявлення аномалій. У статті [18] описуються реалізовані методи, а також висвітлюється їх застосування до модельованих даних, а також реальні приклади даних, що містяться в пакеті. Поділ на точкові і колективні аномалії, а також, методи їх виявлення є важливим в розвитку методології з виявлення аномалій.

Метою роботи [19] є швидке та точне виявлення ненормальних даних складного та складного промислового обладнання із датчиками. Завдяки стрімкому розвитку Інтернету речей, все більше обладнання обладнується датчиками, особливо більш складне та складне промислове обладнання встановлюється з великою кількістю датчиків. Для моніторингу роботи обладнання швидко збирається велика кількість даних моніторингу. Обробка таких даних, причому у великій кількості, представлена в роботі.

В роботі [20] представлено застосування такого методу, який називається однокласною машиною векторної підтримки, для пошуку аномальних шаблонів

серед джерел, попередньо вибраних із середньо-інфрачервоного каталогу. Для створення моделі очікуваних даних в роботі описано результат тренувань алгоритму на наборі об'єктів зі спектроскопічними ідентифікаціями. Виявлення аномалій додає гнучкості автоматизованим процедурам поділу джерел та допомагає перевірити надійність та репрезентативність навчальних зразків. Таким чином, це слід розглядати як важливий крок у контрольованих схемах класифікації для забезпечення повноти та чистоти створених каталогів.

У роботі [21] описано загальний механізм аналітики, який забезпечує надійні попередження про зміни та аномалії в сенсорному потоці даних. У той час як більшість існуючих аналітичних реалізацій IoT вимагають припущень, що стосуються конкретних областей, в роботі надано значну інформацію через методи машинного навчання та вдосконалені статистичні тести без попередніх знань. Система виявлення аномалій мережі дозволяє контролювати комп’ютерну мережу, яка поводиться інакше, ніж мережевий протокол, і її багато застосовується в різних доменах. Проте, проблема виникає там, де різні домени застосунків мають різні визначальні аномалії у своєму середовищі. Вони ускладнюють вибір найкращих алгоритмів, які відповідають вимогам певних доменів. Крім того, проблема централізації, яка спричиняє руйнування мережевої системи, коли в систему вливається потужний зловмисний код. Тому в цій роботі показано результати проведеного експерименту із використанням контрольованого машинного навчання для системи виявлення аномалій мережі, яка мінімізує вартість зв'язку та пропускну здатність мережі, мінімізовану за допомогою набору даних для порівняння їх продуктивності в термінах їх точності та часу обробки для класифікатора для побудови моделі. В результаті, розподілений алгоритм вирішує проблему централізації з точністю та часом обробки, як і раніше, значним у порівнянні з централізованим алгоритмом, хоча є певна втрата точності та часу.

1.2.2. Аналіз методів та стратегій створення розподілених систем

Побудова розподілених систем в локальних комп’ютерних мережах поряд з розробкою нових методів чи удосконаленням відомих з виявлення аномалій є актуальною, бо від ефективності їх функціонування залежить оперативність при виявленні аномалії та реагування на неї. Крім того, ефективна інтеграція при імплементації методу виявлення аномалії в розподілену систему може покращити загальну ефективність з виявлення аномалії та реагування на неї. Розглянемо відомі методи, що стосуються проєктування розподілених систем та оптимізаційні стратегії для покращення їх ефективності функціонування.

Для розподілених систем вартість зв'язку є найбільш часто використовуваною метрикою з метою оцінки ефективності операцій у розподілених алгоритмах для середовищ передачі повідомлень [22]. При цьому постійне припущення полягає в тому, що вартість обчислень в компонентах незначна порівняно з вартістю зв'язку. Однак у багатьох випадках реалізації операцій покладаються на складні обчислення, які не слід ігнорувати. Тому більш точна оцінка ефективності роботи повинна враховувати як обчислювальні витрати, так і витрати на зв’язок. У роботі [22] основна увага приділяється ефективності операцій читання та запису в емуляціях атомної спільної пам’яті читання / запис в асинхронному середовищі, що передає повідомлення, схильному до збоїв. В роботі розроблено і запропоновано новий обчислюваний предикат та алгоритм його обчислення за лінійний час. Результати опубліковані в роботі [22] надають нового значення терміну швидкості, оцінюючи як зв'язок, так і ефективність обчислень кожної операції і є важливими для побудови розподілених систем в частині організації зв’язку між компонентами та обчисленнями в них.

В роботі [23] проаналізована ефективність слабо узгоджених, але швидко реагуючих розподілених сховищ даних. Актуальність цього напряму обгрунтована проблемами, які пов’язані з узгодженістю між розробкою складних застосунків та отриманням лише слабких гарантій узгодженості в сховищах даних. Компроміс узгодженості спрямований на досягнення як сильної узгодженості, так і низької

затримки у загальному випадку. У розподілених системах зберігання досліджене в [23] загальне поняття майже сильної узгодженості з точки зору розробки алгоритмів швидкого зчитування, одночасно гарантуючи ймовірнісну атомність з чітко обмеженою кількістю записів одночасно. Загальний випадок цієї проблеми полягає в тому, коли кілька клієнтів можуть писати дані в розподілених сховищах даних. Важливим показником в цьому випадку є межа застарілості даних та ймовірність порушення атомарності, розкладаючи непослідовні зчитування на інверсію читання та шаблони інверсії запису. Результат цієї роботи важливий для організації розподілених сховищ даних і узгодження інформації в них за критерієм актуальності.

В роботі [24] представлена розподілена система для управління дуже великими обсягами структурованих даних, розподілених на багатьох товарних серверах, забезпечуючи при цьому високодоступні послуги без жодної точки відмови. Ця система може працювати на інфраструктурі з сотнями вузлів розповсюджено за різними центрами обробки даних. Вона не підтримує повну реляційну модель даних, а надає клієнтам модель простих даних, яка підтримує динамічний контроль над розміщенням даних та форматуванням.

В роботі [25] представлено створені оптимальні комунікаційні протоколи у трьох сценаріях, що є важливим для підтримки цілісності розподіленої системи. Проблема складності при організації комунікації пошуку наближеного максимального узгодження в багатосторонній моделі передачі повідомлень є актуальною [26]. Задача максимального узгодження є однією з найбільш фундаментальних комбінаторних задач графа, що має різноманітні програми, і вирішенню оцінки її складності присвячена робота авторів в [26]. Проблемам виявлення мережевих структур та їх топологій присвячена робота [27], яка відіграє центральну роль у розподілених обчисленнях. В роботі [28] досліджено різні налаштування розподілених обчислень та відповідні для них методи. У роботі [29] досліджено обчислювальну потужність популяційних протоколів за деяких ненадійних або слабших моделей взаємодії. Це необхідно для організації підтримки цілісності розподілених систем.

Протягом багатьох років елегантна ієрархія консенсусу, що базується на обчислюваності, була найкращим поясненням відносної сили різних об’єктів [30]. Оскільки справжні мультипроцесори дозволяють застосовувати різні інструкції, які вони підтримують, до будь-якого місця пам'яті, то можливість поєднання інструкцій, що підтримуються різними об’єктами, а не розгляд колекції різних об’єктів є актуальним. У цій роботі запропоновано класифікацію відносної потужності наборів багатопроцесорних команд синхронізації на основі складності, що охоплюється мінімальною кількістю місць пам'яті необмеженого розміру, необхідних для вирішення безперешкодного консенсусу при використанні різних наборів інструкції [30]. Вивченню складності повідомлення неявних виборів лідера в синхронних розподілених мережах діаметром два присвячені дослідження в роботі [31]. Результати характеризують складність повідомлення виборів лідера щодо діаметра графіка [31]. В роботі [32] розглянуто проблеми планування в моделі потоку даних розподіленої транзакційної пам'яті. Об'єкти, спільно використовувані транзакціями, переміщуються з одного мережного вузла на інший, слідуючи мережним шляхам. Авторами досліджено, як передача об’єктів у мережі впливає на час завершення всіх транзакцій та загальну вартість зв'язку. Розроблені авторами алгоритми планування, які для зв’язку працюють майже оптимально і ефективно виконуються за часом. У роботі [ЗЗ] розглядається модель зв'язку, в якій в кожному раунді кожен агент витягує інформацію з кількох випадково обраних агентів. Таким чином, метою є визначення найменшлшл обсяг інформації, виявленої в кожній взаємодії (розмір повідомлення), що, тим не менше, дозволяє ефективно та надійно обчислювати основні завдання розповсюдження інформації. Розроблений протокол використовує лише 3 біти на взаємодію.

У роботі [34] пропонується нова техніка розподіленого за функціями зловмисного програмного забезпечення, яке динамічно розподіляє свої функції серед багатьох програмних компонентів, щоб обійти різні механізми безпеки, такі як додавання до білого списку програм та виявлення поведінки антивіруса. Для оцінки такого підходу авторами впроваджено інструмент, який автоматично

генерує такі екземпляри зловмисного програмного забезпечення, та проведено серію експериментів, що показують ризики.

Розпізнати цільове зловмисне програмне забезпечення за допомогою антивірусів, IDS, IPS та спеціальних засобів виявлення досить складно [35]. Автори порівнюють різні методи машинного навчання, що використовуються для аналізу шкідливих програм, зосереджуючись на статичному аналізі.

В роботі [36] авторами розробляємо офіційну систему пасивного тестування програмних систем, де сторони спілкуються асинхронно.

Авторами в роботі [37] розроблено протокол, який з великою ймовірністю підраховує розмір розподіленої системи, компоненти якої з самого початку є добре змішаними і в процесі функціонування кооперуються. В роботах [38-41] авторами, також, досліджуються протоколи для розподілених систем.

Робота [42] авторів стосується локальних проблем розподілених обчислень та досліджує розрив між рандомізованими та детермінованими рішеннями за обмеження пропускної здатності. Їх головний внесок полягає у наданні інструментів для дерандомізації рішень локальних проблем. В роботах [43-45] досліджується ця ж проблема і запропоновано її рішення.

В роботі [46] авторами досліджено процеси, які обмінюються даними через спільну пам’ять в розподілених системах, з метою встановлення можливості випадковим чином отримувати їх з основного планувальника. Ними представлено загальний метод обчислення цих величин шляхом класифікації розподілених алгоритмів за їхньою схемою доступу до спільної пам'яті. В роботах [47-49] досліджується цю ж проблема, яка пов’язана з генерацією випадкових чисел.

В роботі [50] авторами представлено підхід для керування ресурсами обчислювальної мережі за умови встановлення довіри до компонентів системи.

Таким чином, розробка розподіленої системи виявлення аномалії в комп’ютерних системах базується на двох складових: розробка розподіленої системи; використання та удосконалення методів виявлення аномалії. Аналіз наукових результатів з цих напрямків є основою для створення розподіленої системи виявлення аномалії в комп’ютерних системах.

1.2.3. Аналіз існуючих систем з виявлення аномалій та вторгнень в комп’ютерні системи

Найближчими програмними рішеннями до розроблюваної системи виявлення аномалій в локальних комп’ютерних мережах згідно поставленої задачі є мережні системи виявлення вторгнень, мережні антивірусні програми та некомерційні розробки відповідного призначення.

Мережна система виявлення зловмисного програмного забезпечення або комп’ютерних атак переважно має модуль централізованого управління [51-54]. Це дає змогу адміністратору системи керувати оновленнями і налаштуваннями всіх параметрів в мережі з єдиної консолі. Мережні антивірусні засоби, як правило, використовують сумісно із засобами антивірусного захисту вузлів мережі в якості другого рівня захисту [55]. За такої архітектури мережних систем рекомендованим є використання мережних і хостових частин від різних виробників. Наприклад, в якості технології захисту у мережних системах використовується система захисту хостів у корпоративних мережах [52, 56]. Її елементи дозволяють здійснювати контроль над застосунками, веб-трафіком і пристроями, які під’єднуються. Керування функціями системи здійснюється з єдиної консолі. Мережною системою від компанії «Dr.Web» [57, 58] є застосунок «Dr.Web CureNet!». Він побудований згідно централізованої архітектури. Symantec Endpoint Protection є мережною антивірусною системою. Вона забезпечує адміністратора мережі необхідними інструментами з розгортання антивірусних засобів в мережі [53]. Апаратно-програмна система для виявлення зловмисного програмного забезпечення та комп’ютерних атак розроблена компанією Palo Alto Networks [59, 60]. Мережна система «Malwarebytes Endpoint Security» [61] надає розширений набір локальних засобів для виявлення і видалення загроз в комп'ютерах у мережі. Технологія «Cisco® Network Admission Control (NAC)» була розроблена для забезпечення захисту усіх хостів в мережі [57, 58]. Мережна система Kaspersky Administration Kit реалізує самостійну роботу без втручання на етапі дослідження чи виявлення

адміністратора [62].

Засоби для хостових систем, які призначені для виявлення зловмисного програмного забезпечення або комп’ютерних атак: Avast! [63], AVG Antivirus [64], AntiVir (Avira) [65], BitDefender [66], Clam AntiVirus [60] тощо.

Таким чином, результати аналізу, зокрема в [8, 9], показують, що хостові засоби та мережні системи виявлення зловмисного програмного забезпечення або комп’ютерних атак не забезпечують його повного виявлення. З такими результатами погоджуються і виробники засобів і користувачі цих засобів. Перспективним для покращення достовірності виявлення є розробка і використання мережних системи виявлення, які б базувались на виявленні аномалій у вузлах в мережі.

1.3. Постановка задачі дослідження

Для розв’язання наукової задачі з покращення ефективності виявлення аномалії в комп’ютерних системах, прояви якої зумовлені впливами зловмисного програмного забезпечення та комп’ютерних атак необхідно здійснити розроблення методів та розподілених систем з виявлення аномалій на основі синтезу в них принципів самоорганізації та централізації і розв’язати такі завдання:

1) дослідити особливості прояву аномалії в комп’ютерних системах за умов функціонування зловмисного програмного забезпечення і здійснення комп’ютерних атак в локальних комп’ютерних мережах та проаналізувати сучасні методи виявлення аномалії, їх особливості та методи створення і архітектури розподілених систем;

2) удосконалити модель архітектури розподіленої системи виявлення аномалії в комп’ютерних системах, в якій синтезувати вимоги розподіленості, централізованості та самоорганізованості, для створення на її основі розподілених систем та їх компонентів, що функціонуватимуть під керівництвом одного центру розподіленого між різними компонентами і самостійно прийматимуть рішення про

наявність аномалії;

3) розробити метод підтримки цілісності самоорганізованої розподіленої системи виявлення аномалії в комп’ютерних системах для підтримки її цілісності, на основі якого система змогла б самостійно змінювати свою архітектуру без втручання користувача, а також визначати стратегію своєї подальшої роботи;

4) удосконалити метод централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент для зменшення розмірності з моменту отримання та надсилання даних в центр прийняття рішень системи;

5) розробити програмне забезпечення самоорганізованої розподіленої системи виявлення аномалії в комп’ютерних системах для підтвердження можливості практичного створення таких систем згідно запропонованих результатів роботи та використання в експериментальних дослідженнях для порівняння з відомими системами виявлення.

1.4. Висновки до першого розділу

Виявлення зловмисного програмного забезпечення та комп’ютерних атак у локальних комп’ютерних мережах згідно досліджених методів та засобів виявлення може бути реалізовано методами виявлення аномалії в комп’ютерних системах. Аналіз даного напряму та мережних систем виявлення аномалії, прояви яких зумовлені впливами зловмисного програмного забезпечення або комп’ютерних атак показало наступні результати: застосування відомих зловмисникам методів, які базуються на сигнатурному аналізі, контрольних сум тощо, не розв’язує задачу з ефективної протидії зловмисному програмному забезпеченню або комп’ютерних атакам; використання методів виявлення аномалії в комп’ютерних системах порівняно з традиційними методами є перспективним напрямом досліджень; напрям з використання мережних систем виявлення аномалії для покращення ефективності виявлення за рахунок залучення більших обчислювальних ресурсів в локальних мережах та імплементації в них удосконалених методів виявлення активно розвивається дослідниками;

особливості архітектури та організації процесів в них, а також, методів підтримки цілісності їх та розподілення центрів прийняття рішень і сховищ даних активно розвиваються дослідниками і є перспективними для застосування при виявленні аномалії в комп’ютерних системах.

Отже, з метою покращення ефективності виявлення аномалії в комп’ютерних системах перспективним і актуальним є удосконалення та розроблення методів виявлення аномалії та створення розподілених систем виявлення аномалії.

2 АРХІТЕКТУРА САМООРГАНІЗОВАНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ ВИЯВЛЕННЯ АНОМАЛІЙ В КОМП'ЮТЕРНИХ СИСТЕМАХ НА ОСНОВІ МЕТОДУ ГОЛОВНИХ КОМПОНЕНТ

2.1 Архітектура самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах

Виявлення аномалій в комп'ютерних системах, які викликані проявами зловмисного програмного забезпечення або в результаті здійснення комп'ютерних атак потребують не тільки ефективних методів, які з високим ступенем достовірності встановлюватимуть наявність зловмисних аномальних проявів, але не менш важливим є система, в яку імплементовано розроблені методи.

Вимоги до системи, в яку будуть імплементовані розроблені, потрібно задавати такі, щоб в подальшій своїй роботі така система могла підтримувати свою працездатність в умовах проявів зловмисного програмного забезпечення або в результаті здійснення комп'ютерних атак. Якщо система в умовах зловмисних проявів не зможе підтримувати свою працездатність, тоді і методи, які закладені в неї з виявлення аномалій, не будуть застосовними. Тому, вимоги до системи такого призначення повинні бути сформовані з врахуванням не тільки особливостей застосування, але і враховуючи середовище функціонування, в якому, наприклад, здійснюватимуться зловмисні прояви.

Система виявлення аномалій в комп'ютерних системах для реалізації можливості залучення інформації з різних комп'ютерних станцій, під'єднаних в локальну мережу, повинна мати розподілену архітектуру, оскільки в такому випадку вона зможе скористатись перевагами залучення більшої обчислювальної потужності за рахунок об'єднання обчислювальних ресурсів всіх комп'ютерних станцій, в яких встановлені її компоненти, порівняно з зловмисними проявами, які можуть надходити чи здійснюватись в одному або декількох вузлах в мережі. Розподіленість в її архітектурі в мережі в комп'ютерних станціях надає переваги над зловмисним програмним забезпеченням або при здійсненні комп'ютерних атак

на комп’ютерні системи в мережі завдяки можливості першочергово забезпечити функціонування компонентів у вузлах мережі, які не піддаються атаці чи впливу зловмисного програмного забезпечення, і тому можуть бути залучені до процесу виявлення, навіть за умови втрати частини системи через виведення її компонентів з безпечного стану через втрату контролю над вузлами в мережі через їх ураження зловмисним програмним забезпеченням або внаслідок успішно проведеної комп’ютерної атаки. Але розподілення компонентів системи між різними комп’ютерними станціями в архітектурі системи має недолік, який пов’язаний першочергово з витратами часу на пересилання зібраних для обробки даних до центру чи центрів прийняття рішень, а потім так само поверненням результатів обробки в формі прийнятого рішення щодо подальших дій компонентів системи. Побудова відповідної архітектури системи, яка б враховувала баланс переваг і недоліків такої архітектури та вплив на її компоненти проявів зловмисного програмного забезпечення чи комп’ютерних атак, є актуальною науковою задачею.

Для швидкої взаємодії компонентів системи важливим є метод організації взаємодії компонентів системи, а також оптимізація зв’язків між частинами системи в процесі оперативного реагування на аномальні прояви. Якщо б події відбувались в межах однієї комп’ютерної станції, тоді б і рішення про наявність аномальних проявів приймалось безпосередньо в ній згідно певного реалізованого методу в системі, яка розміщена саме в комп’ютерній станції і з іншими вузлами в мережі не має комунікації в питаннях, які стосуються виявлення аномалій. Але в такому хостовому випадку не має гарантії того, що час обробки подій, встановлення факту наявності аномальних проявів буде суттєво меншим або взагалі менше, ніж час витрачений на комунікацію і пересилання повідомлень між компонентами розподіленої системи за умови, що ефективність методів, які імплементовані в неї, та взаємодії між її компонентами можуть бути швидшими при обробці подій, пов’язаних з аномальними проявами. Таким чином, виявлення аномалій окремими хостовими системами в окремих комп’ютерних станціях порівняно з виявленням розподіленими системами в багатьох вузлах в мережі за витраченим часом може бути різним, зокрема, як більшим, так і меншим. Крім того,

хостова система в комп'ютерній станції може не впоратись із зловмисними проявами самостійно і її робота з виявлення зловмисних проявів може суттєво затягнутись в часі. В зв'язку з такими проблемами, які виникають при вирішення питання про архітектуру систем виявлення аномалій та її місце розміщення та місце використання, сформулюємо наступні гіпотези. До першої групи гіпотез віднесемо врахування застосування і розміщення для хостових або мережних систем. Тоді, нехай гіпотеза H_0 – розміщення та застосування хостових систем виявлення аномальних проявів ефективніше порівняно із застосуванням мережних систем. Гіпотеза H_1 – розміщення та застосування хостових систем виявлення аномальних проявів неефективне порівняно з мережними системами. Класифікуємо результати за цими двома гіпотезами за критерієм отримання кращого результату і представимо в таблиці спряження на основі результатів класифікації і фактичної належності класам.

Таблиця 2.1 – Таблиця спряження

Гіпотеза	Результати правильно класифікованих об'єктів для хостових систем, %	Результати неправильно класифікованих об'єктів для хостових систем, %
H_0	K_{TP}	K_{FP}
H_1	K_{FN}	K_{TN}

Позначення в таблиці 2.1 означають такі величини:

1) $K_{TP}\%$ - відсоток правильно класифікованих об'єктів для хостових систем;

2) $K_{FP}\%$ - відсоток неправильно класифікованих об'єктів для хостових систем як правильно класифікованих об'єктів (відсоток помилок другого роду; відсоток хибного виявлення);

3) $K_{FN}\%$ - відсоток дійсних об'єктів для хостових систем класифікованих неправильно (помилка першого роду; відсоток хибно пропущених об'єктів);

4) $K_{TN}\%$ - відсоток правильно класифікованих неправильних об'єктів об'єктів для хостових систем.

За результатами тестування, яке проводиться на постійній основі регулярно, антивірусних засобів та систем виявлення вторгнень провідними загальновизнаними лабораторіями [1, 2] стосовно достовірності виявлення ними зловмисного програмного забезпечення та комп'ютерних атак було здійснено вибірку результатів тестування хостовими та мережними системами і розподілено їх два класи. Данні, які отримані в результаті їх розподілу, оброблені за гіпотезами та представлені відповідно між чотирма класами. В результаті такого дослідження було встановлено, що достовірність виявлення зловмисного програмного забезпечення та комп'ютерних атак антивірусними засобами та системами виявлення вторгнень усереднена за двома типами засобів для виявлення в хостових засобах хостові засоби мають менший відсоток виявлення в комп'ютерній станції порівняно з виявленням мережними засобами в окремій комп'ютерній станції. Але в такому випадку мережні засоби використовуються не тільки для здійснення виявлення в окремій комп'ютерній станції, але мають ще спрямованість на інші вузли в комп'ютерній мережі. Така спрямованість мережних систем виявлення призводить до витрат часу на задачі з виявлення, як в окремих комп'ютерних станціях, так і в цілому в мережі. Все це вимагає відповідних витрат часу, які можуть перевищити витрати часу на виконання завдань з виявлення окремими хостовими системами в окремих комп'ютерних станціях, причому цей процес здійснюється паралельно. Тому, фактично наявними є такі варіанти архітектури системи виявлення аномальних проявів: хостова система; мережна система з основними задачами виявлення в мережі та її вузлах; мережна система винятково для виявлення аномальних проявів в мережі; мережна система для виявлення аномальних проявів в мережі та в кожній комп'ютерній станції під'єднаній в мережу. Системи виявлення вторгнень як окремі системи і комбіновані з системами виявлення аномалій не розглядатимемо. Таким чином, виберемо тип мережної системи з розглянутих та проаналізований такий, що включатиме в себе необхідність вирішення задач виявлення аномалій в мережі та в кожній комп'ютерній станції під'єднаній в мережу. Вибір такого типу системи виявлення аномалій на основі дослідження проведеного з джерел [1, 2] надасть можливість

здійснювати виявлення аномалій і засобами хостової частини системи з повноцінним функціоналом та залученням мережної частини системи, яка надаватиме додаткові обчислювальні потужності та імплементовані в неї методи. Такий вибір спрямування системи виявлення аномалій на вузли в мережі та саму мережу визначатиме відповідну її архітектуру, особливістю якої буде розподілення в мережі. Місцем розміщення такої системи виберемо локальну мережу. Масштабування розподіленої системи виявлення аномалій в корпоративній мережі за потреби можна буде здійснити в межах окремих її сегментів локальних мереж не тільки незалежно одна від однієї, а й інтегровано. Необхідність локалізації місця застосування розподіленої системи обгрунтовано ще тим, що відомості про вузли мережі відомі адміністратору і можуть бути враховані в архітектурі розподіленої системи при налаштуванні її під час встановлення. Локалізація місця встановлення розподіленої системи дає змогу отримати перевагу над зловмисними проявами в окремих комп'ютерних станціях не тільки за рахунок залучення більшої потужності обчислювальних ресурсів, але й за рахунок прийняття рішення про наявність аномалії не безпосередньо у атакованій комп'ютерній станції, а в окремому центрі, що суттєво підвищує довіру до отриманого результату.

Прийняття рішень системою виявлення аномалій повинно здійснюватись або в одному центрі або в розподілених центрах на різних рівнях ієрархії. Якщо прийняття рішень буде відбуватись тільки винятково в одному центрі, тоді вся інформація повинна пересилатись в нього, очікувати на обробку, прийняття рішення і надсилання його іншим компонентам системи для виконання подальших дій. Все це може суттєво сповільнити роботу системи в цілому, якщо в центрі накопичиться багато завдань з різних компонентів системи, і може призвести до втрати актуальності прийнятого рішення, бо процеси в мережі та окремих її вузлах виконуються швидко і тому потребують динамічного реагування на події. Визначення місця прийняття рішень з питань, які пов'язані з функціонуванням системи або з результатами обробки подій у мережі та її вузлах з використанням імплементованих в систему методів, потрібно розподілити в залежності від їх важливості та віднесення до частини системи чи до всієї системи. Найкращим

рішенням в такому випадку було б рішення, коли центр прийняття рішень був би найближче до тієї частини системи, яка його потребує. В такому випадку центр прийняття рішень повинен розподілитись між рівнями в компонентах системи. Для реалізації цього необхідно в архітектурі системи вибудувати ієрархічні рівні. Хоча компоненти різних рівнів можуть комунікувати між собою, але на кожному рівні ієрархії будуть центри прийняття рішень, які матимуть можливість приймати рішення тільки з певних чітко визначених питань згідно отриманих з компонентів системи зібраних початкових даних. Але не завжди чітко визначені функції, наприклад, реагування на встановлені аномальні прояви, можуть бути віднесені тільки до центру прийняття рішень, що знаходиться в компоненті нижнього рівня ієрархії. Такі реагування на аномальні прояви повинні бути узгоджені або з самого початку або після первинної реакції основним центром прийняття рішень всієї системи. Також, інші центри прийняття рішень, також, повинні оперативно інформуватись про встановлення аномальних проявів у певному вузлі мережі. Таким чином, правильний розподіл в системі центрів прийняття рішень, визначатиме її ефективність і можливість оперативного реагування на виявлені аномальні прояви.

Самоорганізованість як характерна особливість проєктованої системи виявлення аномалій є необхідною, оскільки події в комп’ютерних системах відбуваються дуже швидко і реагування на них повинно бути таким, щоб результат обробки та прийняття рішення був актуальним, а не із запізненням. Хоча він може надходити із запізненням і бути врахованим, але переважно, враховуючи наслідки впливу зловмисного програмного забезпечення та комп’ютерних атак, є необхідним швидке реагування на події. Якщо б пропонований результат обробки подій з метою виявлення аномалій для остаточного прийняття рішення в кожній ситуації яка виникає у вузлах мережі, в які встановлено компоненти системи, покладався винятково на системного адміністратора мережі або фахівця з кібербезпеки, тоді більшість подій оброблялись би із суттєвим запізненням. Таке залучення до прийняття рішень системного адміністратора мережі або фахівця з кібербезпеки може бути потрібним на етапі аналізу журналу реєстрації виявлених

аномальних подій, зареєстрованим системою і прийнятих нею рішень. Але оперативність в прийнятті рішень найкраще покласти саме на систему, тому в її основі має бути можливість до самоорганізованості для визначення своїх подальших кроків. В загальному така характеристична властивість системи як самоорганізованість може містити механізми не тільки визначення подальших наступних кроків, але і механізми з динамічної перебудови своєї архітектури в залежності від впливів зловмисного програмного забезпечення, комп'ютерних атак, а також результатів оброблених подій з виявлення аномалій. Самоорганізованість системи на рівні закладених в систему механізмів і функцій може бути реалізована як частина основного центру прийняття рішень, тобто тієї частини центру, яка знаходиться на верхньому рівні ієрархії.

Враховуючи такі характерні властивості та особливості проєктованої системи як самоорганізованість та розподіленість, потребує вирішення питання динамічного формування системи з наявних активних компонентів на певний час. Це повинно бути відповідним чином спроєктовано, як на випадок для початкового формування, так і у випадку тривалого використання системи та зміни її архітектури в залежності від зміни активних її компонентів та реагування на аномальні прояви.

Специфіка застосування проєктованої системи виявлення аномалій в комп'ютерних системах пов'язана з руйнуючими впливами зловмисного програмного забезпечення та комп'ютерних атак, причому ці руйнуючі впливи можуть стосуватись і функціонування проєктованої системи з метою виведення її з ладу або окремих її компонентів чи спотворення передаваних даних між компонентами системи. Складність виявлення зі сторони зловмисника систем захисту комп'ютерних систем полягає, зокрема і у відсутності відомостей про засоби захисту атакованих систем. Це дозволяє здійснювати ефективний захист та виявлення аномальних проявів такими системами без втрати відповідної функційності з виявлення аномалій чи частини функційності. За умови проєктування системи, як розподіленої у вузлах мережі, досить важливою стає організація правильної взаємодії компонентів проєктованої системи на основі

нового мережного протоколу передачі даних або удосконаленням існуючого, але який би враховував специфіку задач і ускладнених зловмисним програмним забезпеченням чи комп'ютерними атаками умов, в яких здійснюватиметься обмін інформацією між компонентами системи. Протокол, який регламентуватиме обмін повідомленнями між компонентами проєктованої системи, повинен мати додаткові елементи для підтвердження отримання повідомлення, враховувати динамічне формування системи з її компонентів в різний час, уникнення блокування компоненту системи у випадку затримки повідомлення тощо. Тобто вимоги до протоколу обміну інформацією між компонентами системи повинні бути інші, ніж при відомих, наприклад, IRC. Така вимога до протоколу потрібна, також, для того, щоб підтримувати цілісність проєктованої системи.

Архітектура проєктованої системи в процесі її функціонування повинна динамічно сформуватись з обов'язкової компоненти, в якій частина центру верхнього рівня ієрархії, та частини компонентів системи, необов'язково всіх компонентів. Не обов'язково, щоб усі компоненти проєктованої системи в процесі її функціонування, були наявними та активними. Частина з них може бути у вимкнених комп'ютерних станціях або в процесі функціонування системи частину комп'ютерних станцій з її компонентами користувачі вимкнуть. В такому випадку система повинна продовжувати виконання своїх функцій.

Функціонування хостових компонентів системи окремо за відсутності центру системи і виконання ними повноцінних дій з виявлення аномальних проявів засобами імплементованих функцій повинно підтримуватись, оскільки центр тимчасово може бути недоступним і, тоді, вся система не зможе протистояти зловмисному програмному забезпеченню і комп'ютерним атакам. Важливою є реалізація за таких випадків горизонтального інтерфейсу між компонентами системи нижнього ієрархічного рівня. Це надало б змогу ефективніше протистояти зловмисним проявам за відсутності центру. Для організації такої взаємодії компонентів необхідним є відповідний протокол і обробка таких подій частинами центру, що знаходиться в компонентах проєктованої системи.

При таких вимогах до архітектури проєктованої розподіленої системи, в якій

хостові частини системи розміщені в комп'ютерних станціях повинні приймати рішення про наявність аномалії та передавати результат виявлення в центр розподіленої системи і мережна частина повинна виконувати завдання з виявлення аномалій в локальній мережі, в ній необхідно синтезувати наступні характерні властивості, методи та функційні можливості:

1) централізованість (єдиний центр прийняття рішень) у прийнятті рішень в системі;

2) наявність рівнів ієрархії в питанні прийняття рішень в розподілених частинах центру в усіх компонентах системи у вузлах мережі;

3) розподілення компонентів системи у різних вузлах в мережі;

4) самоорганізованість системи при прийнятті рішень про подальші кроки в роботі системи та її компонентів;

5) динамічне формування системи в процесі її функціонування, як під час початкового формування, так і в процесі тривалого використання;

6) мережний протокол для взаємодії компонентів проєктованої системи;

7) динамічне формування архітектури системи в процесі її функціонування з обов'язкової компоненти, в якій частина центру верхнього рівня ієрархії, та частини компонентів системи, необов'язково всіх компонентів;

8) функціонування хостових компонентів системи окремо за відсутності центру системи і виконання ними повноцінних дій з виявлення аномальних проявів засобами імплементованих функцій;

9) імплементація методів виявлення аномалій в комп'ютерних системах в проєктовану систему.

Синтезуючи виділені характерні властивості, методи та функційні можливості, отримуємо самоорганізовану розподілену систему, яка здатна функціонувати в локальній комп'ютерній мережі і вирішувати завдання з виявлення аномалій в комп'ютерних системах при наповненні її відповідним функціоналом.

Представимо самоорганізовану розподілену систему як множину її компонентів. Нехай позначимо множиною M_{SDS} множину компонентів

самоорганізованої розподіленої системи (SDS, self-organized distributed system). Компоненти системи позначимо $M_{SDS,i}$, де i – це номер компонента системи. Центр самоорганізованої розподіленої системи позначимо через $M_{SDS,0}$, тобто це елемент множини компонентів системи при $i = 0$. Тоді, множину компонентів самоорганізованої розподіленої системи задамо так:

$$M_{SDS} = \{M_{SDS,0}, M_{SDS,1}, M_{SDS,2}, \dots, M_{SDS,N}\}, \quad (2.1)$$

де N – кількість компонентів самоорганізованої розподіленої системи без врахування компоненти, яка містить центр.

Таким чином, загальна кількість компонентів самоорганізованої розподіленої системи становить $N + 1$. Мінімальна кількість компонентів становить один. В цьому випадку самоорганізована розподілена система масштабується зменшенням до одного компоненту, який забезпечує виявлення аномалій в одній комп'ютерній станції і не містить функціоналу з встановлення наступних кроків всієї системи, навіть якщо цей єдиний компонент системи є такий, який містить центр системи. Якщо ж система містить більше одного компонентів і серед них немає того, який містить центр, тоді всі вони функціонують теж для забезпечення виявлення аномалій в своїх комп'ютерних станціях, не містять функціоналу з встановлення наступних кроків всієї системи, але здійснюють обмін відомостями про об'єкти, в яких ними були виявленні джерела аномальних проявів.

Представимо синтезовану архітектуру самоорганізованої розподіленої системи з врахуванням її подання через множину компонентів та характерних властивостей, методів виявлення аномалій, функційних можливостей структурною схемою, яку зображено на рис. 2.1. Архітектура самоорганізованої розподіленої системи з відображенням центру системи в якості компоненти зображена на рис. 2.2.

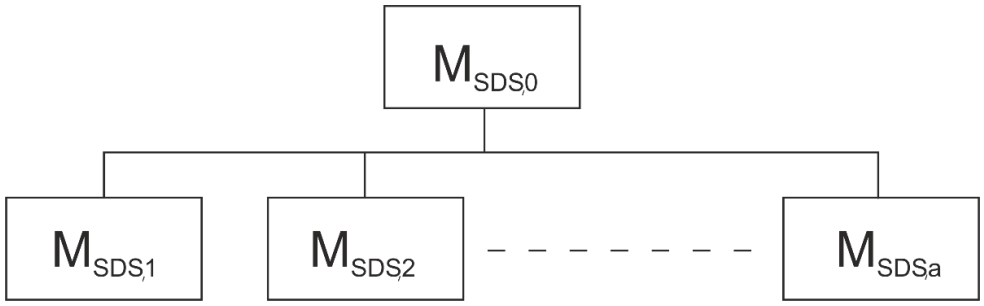


Рисунок 2.1 - Архітектура самоорганізованої розподіленої системи

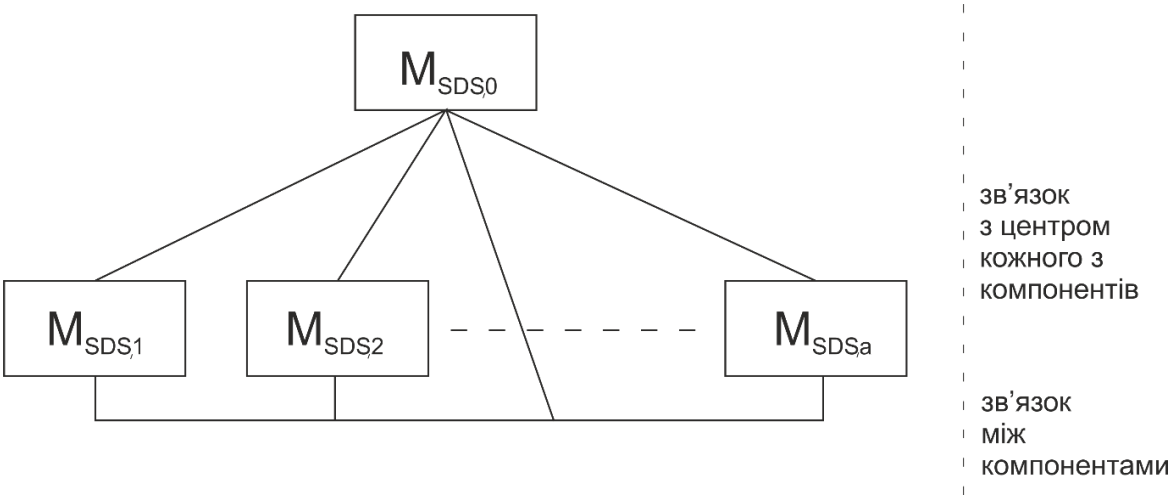


Рисунок 2.2 - Архітектура самоорганізованої розподіленої системи з відображенням центру системи в якості компоненти

В представленій архітектурі самоорганізованої розподіленої системи, на відміну від відомих рішень, удосконалено внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час. Це дало змогу розподілити частину задач з центру на нижчий рівень ієрархії для прийняття рішення в залежності від застосовуваних методів з виявлення аномалій в конкретній комп'ютерній станції. Крім того, поділ задач центру на частини в залежності від їх призначення і за умови відсутності компоненти з вищим рівнем ієрархії, в якій знаходиться частину центру всієї самоорганізованої розподіленої системи, дає змогу здійснювати обмін повідомленнями про виявленні джерела об'єктів, які провокують аномальні прояви. Це є важливим в контексті специфіки вирішуваних задач системою та умов її функціонування, зокрема при руйнуючих впливах зловмисного програмного

забезпечення. Зображення місць розміщення центру прийняття рішень в розроблюваній архітектурі самоорганізованої розподіленої системи представлено на рис. 2.3.

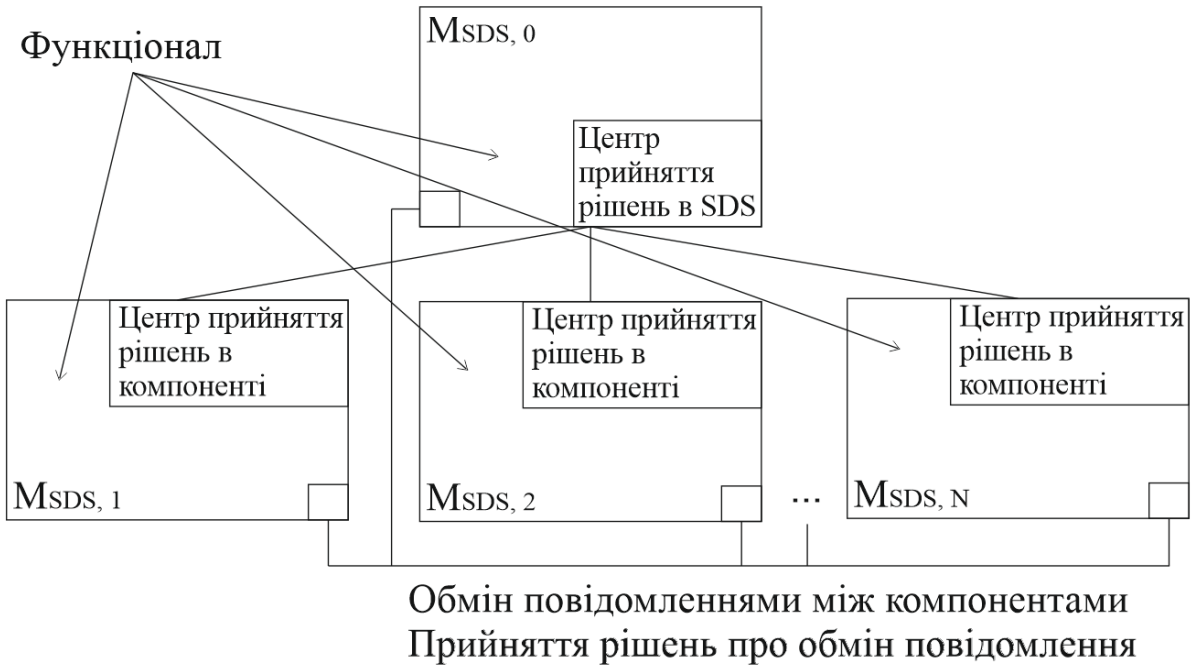


Рисунок 2.3 - Місця центру прийняття рішень в архітектурі самоорганізованої розподіленої системи

В результаті спроєктована архітектура самоорганізованої розподіленої системи дає змогу нарощувати її можливості за рахунок наповнення імплементованими методами з виявлення аномалій в комп'ютерних системах. В архітектурі системи розподілені функції її центру між компонентами, що пришвидшує обробку подій за рахунок здійснення обробки безпосередньо у вузлі мережі, в якому відбувся аналіз аномального прояву. Крім того, система спроєктована таким чином, що її компоненти можуть обмінюватись результатами обробки аномальних проявів та виявленими їх джерелами.

2.2. Метод підтримки цілісності самоорганізованої розподіленої системи

Розподілення компонентів самоорганізованої розподіленої системи

актуалізує проблему забезпечення цілісності системи та ефективної взаємодії компонентів, оскільки всі компоненти розміщені у різних вузлах в мережі. Крім того, підтримка цілісності самоорганізованої розподіленої системи за рахунок організації ефективної комунікації між ними є важливим завданням і не тільки за нормальних умов. Особливої ваги ефективність в організації такої комунікації в підтримці цілісності системи набуває за умов руйнуючих впливів зловмисного програмного забезпечення та комп'ютерних атак. Тому, метод підтримки цілісності самоорганізованої розподіленої системи за рахунок ефективної комунікації між компонентами повинен базуватись на невідомому для зловмисників мережному протоколі і мати набір варіантів подальших кроків всієї системи за настання відповідних умов. Інтеграція цих двох складових в методі підтримки цілісності необхідна для покращення безпеки безпосередньо для самої системи порівняно з іншими функціонуючими системами функціонуючими за відомими мережними протоколами.

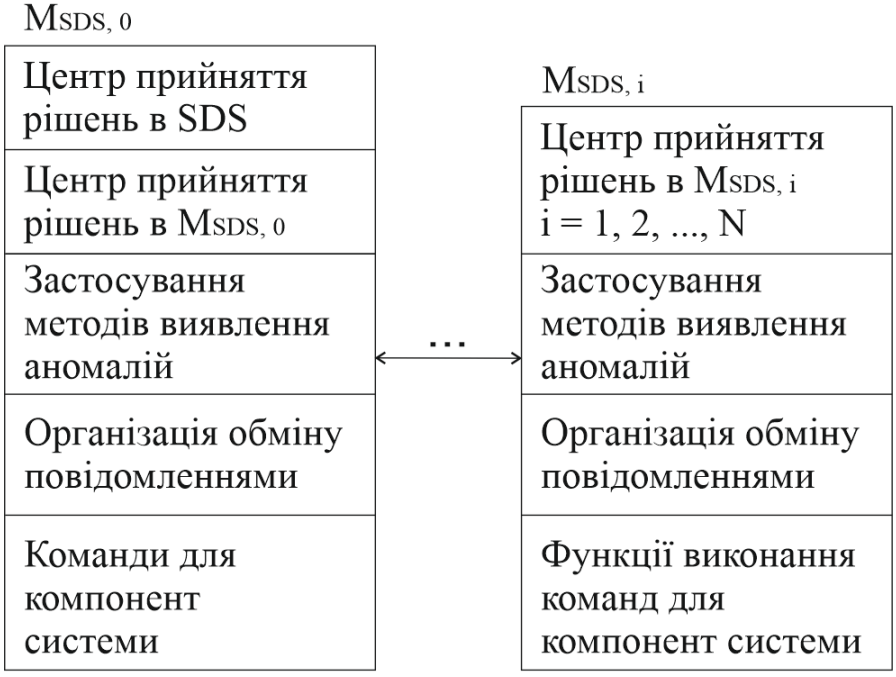


Рис. 2.4 – Архітектура компонент системи та зв'язків між ними

Розглянемо спочатку формування мережного протоколу для організації взаємодії компонентів. Врахуємо архітектурні особливості системи в тій частині,

що відноситься до розподілення центрів прийняття рішень. Фактично саме з центрів прийняття рішень будуть надходити вказівки про пересилання даних. Тому, враховуючи три типи зв'язків між компонентами системи, які залежать від рівнів ієрархії на яких вони розміщені, отримуємо різні пари елементів по три компоненти в кожному так:

1) $(c_{SDS,0}, c_{SDS,i}, 1)$ – відображає передачу команди з центру верхнього рівня, який приймаємо таким позначенням як $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$, до центрів системи в компонентах, які знаходяться на нижньому рівні, тобто до центрів з позначенням $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$, де $i = 1, 2, \dots, N$;

2) $(c_{SDS,i}, c_{SDS,0}, 2)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру верхнього рівня $c_{SDS,0}$ для здійснення їх обробки;

3) $(c_{SDS,i}, c_{SDS,0}, 3)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті до центру верхнього рівня $c_{SDS,0}$;

4) $(c_{SDS,i}, c_{SDS,j}, 4)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, N$;

5) $(c_{SDS,i}, c_{SDS,j}, 5)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, N$;

6) $(c_{SDS,i}, c_{SDS,j}, 6)$ – відображає передачу повідомлення (при наявності відомостей про відсутність компоненти в системі з центром верхнього рівня, тобто

фактичного центру прийняття рішень в системі) з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, N$;

7) $(c_{SDS,i}, c_{SDS,j}, 7)$ – відображає передачу повідомлення (при наявності відомостей про відсутність компоненти в системі з центром верхнього рівня, тобто фактичного центру прийняття рішень в системі) з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, N$;

8) $(c_{SDS,0}, c_{SDS,j}, 8)$ – відображає передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру нижнього рівня $c_{SDS,j}$ для здійснення їх обробки при цьому $j = 1, 2, \dots, N$;

9) $(c_{SDS,0}, c_{SDS,j}, 9)$ – відображає передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру нижнього рівня $c_{SDS,j}$ для здійснення їх обробки при цьому $j = 1, 2, \dots, N$;

10) $(c_{SDS,i}, c_{SDS,0}, 10)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$ і де $i = 1, 2, \dots, N$ до центру верхнього рівня $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи;

11) $(c_{SDS,0}, c_{SDS,i}, 11)$ – відображає передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$ до центру нижнього рівня $c_{SDS,i}$ і де $i = 1, 2, \dots, N$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі

і успішний запуск програмного забезпечення компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи, в якій функціонує центр системи, тобто оновлення даних про наявну в даний момент архітектуру;

12) $(c_{SDS,i}, c_{SDS,0}, 12)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$ і де $i = 1, 2, \dots, N$ до центру верхнього рівня $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ з метою повідомлення про коректне вимкнення комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі;

13) $(c_{SDS,0}, c_{SDS,i}, 13)$ – відображає передачу повідомлення з центру верхнього рівня $c_{SDS,0}$ до центру нижнього рівня $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$, $c_{SDS,0} \in M_{SDS,0}$ для $i = 1, 2, \dots j-1, j+1, \dots, N,\ j \neq i$ з метою повідомлення про вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j -тої компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі, тобто повідомлення всім решті активним компонентам системи про наявну архітектуру системи;

14) $(c_{SDS,j}, c_{SDS,i}, 14)$ – відображає передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,j}$ до решти активних центрів нижнього рівня $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ для $i = 1, 2, \dots j-1, j+1, \dots, N,\ j \neq i$ з метою повідомлення про коректне вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j-тої компоненти системи із збереженнях характерних ознак профілю комп'ютерної станції в мережі;

15) $(c_{SDS,0}, c_{SDS,i}, 15)$ – відображає передачу повідомлення з центру верхнього рівня $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ до центру нижнього рівня $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ для $i = 1, 2, \dots j-1, j+1, \dots, N,\ j \neq i$ з метою повідомлення про проблеми в j - тій комп'ютерній станції в мережі і надсилання їй команди з вимогою блокування роботи програмного забезпечення в ній і примусового завершення роботи програмного забезпечення j -тої компоненти системи із збереженням характерних

ознак профілю комп'ютерної станції в мережі, тобто повідомлення всім решті активним компонентам системи про зміну наявної архітектури системи пов'язаної з вилученням j -тої компоненти системи;

16) $(c_{SDS,j}, c_{SDS,i}, 16)$ – відображає передачу повідомлення від центрів всіх рівнів до решти центрів всіх рівнів $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ для $i = 0, 1, 2, \dots j-1, j+1, \dots, N$, $j \neq i$ з метою повідомлення про архітектуру сформованої розподіленої системи, в яку увійшли всі початково задані компоненти і їх готовність до виконання заданих функцій чи продовження роботи.

Всі повідомлення між компонентами системи обов'язково обробляються центрами прийняття рішень в компонентах, незалежно від того до якого рівня ієрархії вони відносяться, і тільки після обробки повідомлень чи схвалення отриманих команд ними вони піддаються обробці чи виконанню іншими частинами компонент системи.

Зі старту комп'ютерних станцій в мережі може виявитись ситуація, коли визначена комп'ютерна станція, в якій знаходиться центр прийняття рішень системи, першою ввімкнеться або буде ввімкненою раніше інших комп'ютерних станцій, в яких решта компонент системи, тоді виконання події, яка описана трійкою елементів $(c_{SDS,i}, c_{SDS,0}, 10)$, $c_{SDS,i} \in M_{SDS,i}$, $i = 1, 2, \dots, N$, $c_{SDS,0} \in M_{SDS,0}$, відбудеться планово і подальші кроки з передачі повідомлень будуть стандартними.

Якщо ж виявиться, що комп'ютерна станція, в якій знаходить компонента з центром прийняття рішень системи, увімкнеться пізніше тих комп'ютерних станцій, в яких знаходяться решта компонентів системи, тоді виконання події, яка описана трійкою елементів $(c_{SDS,0}, c_{SDS,i}, 11)$, $i = 1, 2, \dots, N$ відбудеться в нестандартний спосіб. Це зумовлено тим, що пізнє увімкнення станції або взагалі збій в ній можуть бути викликані руйнуючими впливами зловмисного програмного забезпечення або комп'ютерних атак. Компоненти системи, в яких центр відноситься до нижнього рівня, встановлять факт відсутності компоненти з центром верхнього рівня системи і будуть здійснювати обмін повідомленнями між

собою, поки центр системи знову не повідомить їх про свою активність та готовність до роботи. Але підтвердження таких дій буде здійснюватись за певним алгоритмом, щоб унеможливити зовнішнє втручання підміною компоненти з центром верхнього рівня системи. Якщо певні комп'ютерні станції вимикатимуться через певний період, то компоненти присутні в них будуть надсилати повідомлення і

Команди, які надходять з центру верхнього рівня системи до центру системи нижнього рівня стосуються наступних подій чи станів: призупинити роботу компоненти; обробити дані характерних ознак для дослідження на аномалії; відновити роботу компоненти; надіслати компоненті з центром нижнього рівня повідомлення або вказану команду, тобто здійснити непряме звернення до іншої компоненти системи; надати данні про поточний стан комп'ютерної станції, тобто про профіль її характерних ознак та досліджуваних ознак на аномалії. Крім команд, з центру верхнього рівня системи можуть надходити повідомлення з інформацією, яку треба опрацювати, переслати в іншу компоненту чи зберігати.

Граф з дугами переходів, в якому враховуються три типи зв'язків між компонентами системи, що залежать від рівнів ієрархії на яких вони розміщені, зображено на рис. 2.5.

Відображення зв'язків між компонентами розподіленої системи та подіями, які можуть бути опрацьовані системою і задані елементами (1-16), на графі з дугами переходів є завершеним і не містить висячих вершин першого ступеня, тому описані події заданими елементами є достатніми для забезпечення функціонування розподіленої системи та можуть бути імплементовані в кроках методу підтримки цілісності самоорганізованої розподіленої системи. Додавання нових події, які можуть відбуватись в системі чи оброблятись в ній, є можливим, бо зв'язки в графі відображають замкнутість всіх подій, а їх збільшення фактично дозволятиме нарощувати функціональні можливості самої системи, оскільки кількість компонентів при цьому не збільшуватиметься.

Перелік подальших кроків системи визначатиметься станами, в які може входити самоорганізована розподілена система або її компоненти. Також, ці стани

залежатимуть від кількості активних компонентів системи, стану в комп'ютерних станціях в мережі. І, при цьому, ці стани будуть в часовому вимірі проміжними, оскільки система динамічно змінюватиме свою архітектуру та переходитиме зі стану в стан. Стани залежать від станів компонентів системи, причому як активних, так і вимкнених або вилучених системою.

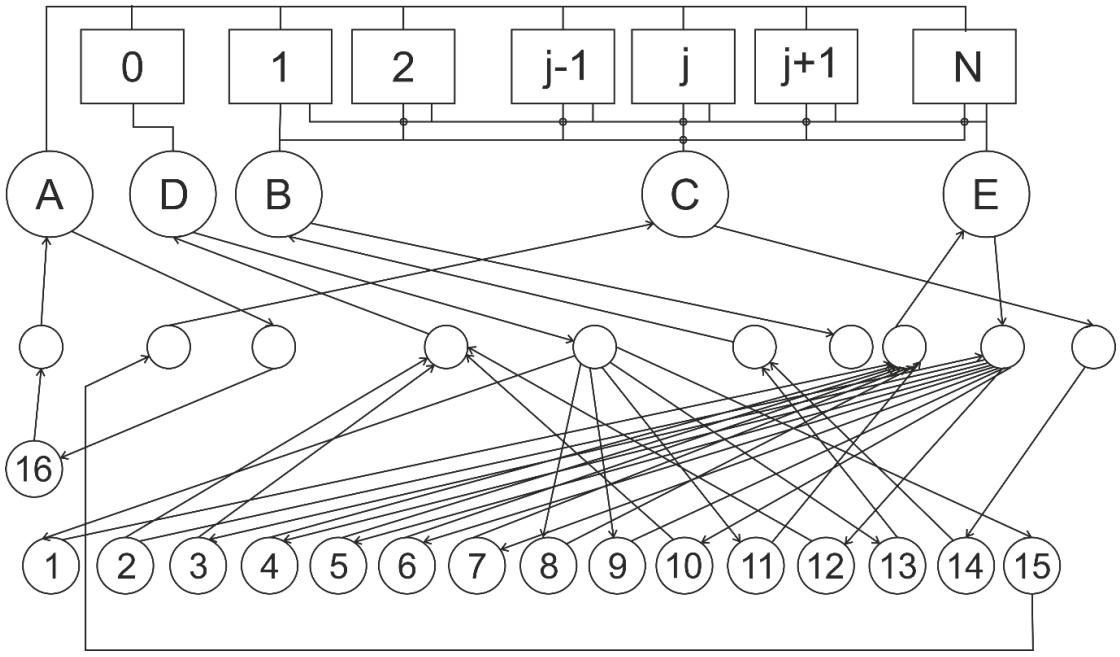


Рисунок 2.5 - Граф з дугами переходів

Метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах включає такі ітераційні кроки:

- крок 1: виконання $(c_{SDS,i}, c_{SDS,0}, 10)$ передачі повідомлення з центру нижнього рівня до центру верхнього рівня, $c_{SDS,0} \in M_{SDS,0}$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи;

- крок 2: виконання $(c_{SDS,0}, c_{SDS,i}, 1)$ для передачі команди з центру верхнього рівня до центрів системи в компонентах, які знаходяться на нижньому рівні, і отримання підтвердження про отримання команди;

- крок 3: виконання команди в компоненті з центром нижчого рівня і

надсилання звіту компоненті системи з центром вищого рівня;

- крок 4: виконання $(c_{SDS,i}, c_{SDS,0}, 2)$ та виконання $(c_{SDS,i}, c_{SDS,j}, 4)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів верхнього та нижнього рівнів для здійснення їх обробки;

- крок 5: виконання $(c_{SDS,i}, c_{SDS,0}, 3)$ та виконання $(c_{SDS,i}, c_{SDS,j}, 5)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про результати обробки аномалії до центрів верхнього та нижнього рівнів для здійснення їх обробки;

- крок 6: виконання $(c_{SDS,i}, c_{SDS,j}, 6)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів нижнього рівнів для здійснення їх обробки за умови відсутності центру верхнього рівня;

- крок 7: виконання $(c_{SDS,i}, c_{SDS,j}, 7)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про результати обробки аномалії до центрів нижнього рівнів для здійснення їх обробки за умови відсутності центру верхнього рівня;

- крок 8: виконання $(c_{SDS,0}, c_{SDS,j}, 8)$ для надсилання повідомлення з центру верхнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів нижнього рівнів для здійснення їх обробки;

- крок 9: виконання $(c_{SDS,0}, c_{SDS,j}, 9)$ для надсилання повідомлення з центру верхнього рівня, в якому міститься інформація про результати обробки аномалії до центрів нижнього рівнів для здійснення їх обробки;

- крок 10: виконання $(c_{SDS,0}, c_{SDS,j}, 11)$ для надсилання повідомлення з центру верхнього рівня всім активним компонентам з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення j-тої компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи, в якій функціонує центр системи;

- крок 11: виконання $(c_{SDS,i}, c_{SDS,0}, 12)$ для передачі повідомлення з центру нижнього рівня до центру верхнього рівня з метою повідомлення про коректне вимкнення комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі;

- крок 12: виконання $(c_{SDS,0}, c_{SDS,i}, 13)$ для передачі повідомлення з центру верхнього рівня до центру нижнього рівня з метою повідомлення про вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j -тої компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі, тобто повідомлення всім решті активним компонентам системи про наявну архітектуру системи;

- крок 13: виконання $(c_{SDS,j}, c_{SDS,i}, 14)$ для передачі повідомлення з центру нижнього рівня до решти активних центрів нижнього рівня з метою повідомлення про коректне вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j-тої компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі;

- крок 14: виконання $(c_{SDS,0}, c_{SDS,i}, 15)$ для передачі повідомлення з центру верхнього рівня до центру нижнього рівня з метою повідомлення про проблеми в j - тій комп'ютерній станції в мережі і надсилання їй команди з вимогою блокування роботи програмного забезпечення в ній і примусового завершення роботи програмного забезпечення j-тої компоненти системи із збереженнях характерних ознак профілю комп'ютерної станції в мережі, тобто повідомлення всім решті активним компонентам системи про зміну наявної архітектури системи пов'язаної з вилученням j-тої компоненти системи;

- крок 15: виконання $(c_{SDS,j}, c_{SDS,i}, 16)$ для передачі повідомлення від центрів всіх рівнів до решти центрів всіх рівнів з метою повідомлення про архітектуру сформованої розподіленої системи, в яку увійшли всі початково задані компоненти і їх готовність до виконання заданих функцій чи продовження роботи.

Схема методу підтримки цілісності архітектури самоорганізованої

розподіленої системи в локальних комп’ютерних мережах полягає в тому, що в його кроках враховано стани компонентів системи, переходи між компонентами і завдяки закладеним в нього крокам вся розподілена система може визначати свої подальші кроки. Основні кроки (1-15) методу виконуються не послідовно, а відповідають ітераційній схемі з одночасним паралельним виконанням певних кроків в різних компонентах одночасно. Тобто певна кількість кроків може виконуватись паралельно. Частина кроків може в певний момент часу не виконуватись. Перехід системи до визначених подальших станів відбувається на основі виконання певних кроків методу і залежить від сукупності можливих заданих станів, але фактично система і її компоненти переходять до виконання визначених кроків методу, тобто переходи до наступних станів системи і компонентів відповідають визначеним центром системи для виконання наступних кроків методу. Це надає змогу реалізувати за рахунок такого підходу таку характеристику системи як самоорганізованість, що на відміну від інших методів, які враховують для переходів дискретні стани системи або інтервальні проміжки для визначення станів системи чи її компонентів, враховує для переходів в якості мети для наступних дій виконання кроків методу. Завершенням виконання кроків методу за умов коректного вимкнення комп’ютерних станцій та програмного забезпечення компонентів системи буде завершення функціонування самоорганізованої розподіленої системи.

Таким чином, розроблено метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп’ютерних мережах, що враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи. Це дозволило будувати системи, які є централізованими та самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп’ютерних атак.

2.3 Висновки до другого розділу

В удосконаленій архітектурі самоорганізованої розподіленої системи, на відміну від відомих рішень, удосконалено внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії. Це дало змогу розподілити частину задач з центру на нижчий рівень ієрархії для прийняття рішення в залежності від застосовуваних методів з виявлення аномалій в конкретній комп'ютерній станції. Розподіл задач центру в залежності від їх призначення і за умови відсутності компоненти з центром вищого рівня, в якій знаходиться частина центру всієї самоорганізованої розподіленої системи, дає змогу здійснювати обмін повідомленнями про виявленні джерела об'єктів, які провокують аномальні прояви.

Результатом так спроєктованої архітектури самоорганізованої розподіленої системи є можливість нарощувати її фугкціонал за рахунок наповнення імплементованими методами з виявлення аномалій в комп'ютерних системах. Система спроєктована таким чином, що її компоненти можуть обмінюватись результатами обробки аномальних проявів та виявленими їх джерелами.

Розроблено новий метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах, що враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи. Це надає змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп'ютерних атак.

3 ВИЯВЛЕННЯ АНОМАЛІЙ В КОМП'ЮТЕРНИХ СИСТЕМАХ НА ОСНОВІ МЕТОДУ ГОЛОВНИХ КОМПОНЕНТ

3.1 Метод головних компонент для виявлення аномалій в комп'ютерних системах

Розроблена самоорганізована розподілена система для виявлення аномалій в комп'ютерних системах потребує наповнення її відповідними методами. Оскільки вона реалізована в локальній комп'ютерній мережі і представляє собою розподілену систему, то предметною областю для дослідження виступає мережне виявлення аномалій. Крім того, процеси, які досліджуватимуться, для збереження актуальності за результатами їх протікання, потребуватимуть оперативного рішення за часом, а також з врахуванням чітко обмеженої кількості вузлів в мережі, якою є локальна комп'ютерна мережа. Оскільки в часі відомості про зібрані данні для прийняття рішень стрімко змінюватимуться, то потрібне врахування коригування по ним. І врахування цих особливостей (час, коригування зібраних даних, обмежена кількість вузлів в мережі, використання розподіленої системи для збору даних і прийняття рішень) може забезпечити метод головних компонент. Особливістю цього методу є те, що при виявленні аномалій в мережі безперервно відслідковується проекція даних на залишковий підпростір. Використання розподіленої системи, яка накопичуватиме данні для аналізу зі всіх вузлів мережі, в яких розміщено компоненти системи, потребуватиме врахування інформації з детекторів в конкретних вузлах мережі. Накопичення даних з великої кількості вузлів в мережі, а також їх періодичне доповнення, впливатиме на швидкість обробки та необхідність їх оптимізації. Крім того, отримувані данні з вузлів в мережі матимуть багато різнотипних представлень і не завжди всі з них будуть мати однакову вагу і значимість. Частина зібраних даних потребуватиме оптимізації і зменшення їх розмірності з мінімальною втратою кількості інформації. Ці вимоги враховані у методі головних компонент (розроблено К. Пірсоном, 1901 р.).

Метод головних компонент може мати певні оптимізації та удосконалення

в залежності від області застосування та можливості до комбінованого застосування з іншими методами чи в межах імплементації в архітектуру певних систем. Розглянемо класичну постановку, суть та кроки методу головних компонент. Початкові данні зібрані з вузлів в комп'ютерній мережі збираються в центрі розподіленої системи, де представимо їх матрицею. Нехай k – це кількість вузлів в мережі, в яких встановлено компоненти розподіленої системи і з яких здійснюється збір даних для аналізу. Представимо скінчену множину різних даних з вузла в мережі впорядкованою послідовністю вектором:

$$V_f = (v_1, v_2, \dots, v_f), \tag{3.1}$$

де v_i – значення даних з досліджуваної компоненти вузла в мережі.

Частина значень даних v_i з досліджуваної компоненти вузла в мережі відносяться до типових компонент і тому вони можуть бути згруповані в класи. Наприклад, такими типовими досліджуваними компонентами можуть бути файли операційної системи, окремі множини файлів з каталогів або взагалі виконувані програми. Також, в окремі класи можуть бути введеними виконувані процеси в оперативному запам'ятовуючому пристрої і данні з усіх портів.

Таким чином, в якості даних з вузлів в мережі розглядатимемо файли операційної системи, виконувані файли в каталогах, процеси у внутрішній пам'яті, данні з портів комп'ютера, характеристики профілю користувача, мережна активність з комп'ютера. Виконувані програми можуть мати декілька характеристик. Зокрема, наприклад час створення та розмір. Певні класи можуть бути відсутніми або кількість їх елементів може братись не вся. Всі дані будуть представлені в числовій формі. Кількість показників з вузла в мережі є великою. Не всі показники в класах матимуть значущі значення, які впливатимуть на результат при прийнятті рішення про наявність аномалії. Тому, необхідним є залучення відповідного математичного апарату, зокрема методу головних компонент, для зменшення розмірності даних.

Отримана з вузла в мережі матриця початкових даних має розмірність $k \times f$,

де k – кількість вузлів в мережі, в яких здійснюється спостереження, f – кількість елементарних показників. Для застосування методу головних компонент потрібно здійснити кроки: отримання матриці початкових даних Q_1; перехід від матриці початкових даних до матриці центрованих і нормованих значень ознак Q_2; перехід від матриці центрованих і нормованих значень ознак до матриці парних кореляцій Q_3; перехід від матриці парних кореляцій до діагональної матриці власних (характеристичних) чисел; перехід від діагональної матриці власних (характеристичних) чисел V_z до матриці факторного відображення Q_4; перехід від матриці факторного відображення до матриці значень головних компонент меншої розмірності Q_5, ніж матриця початкових даних. Схему кроків застосування методу головних компонент зображено на рис. 3.1, де V_v – матриця нормованих власних векторів.

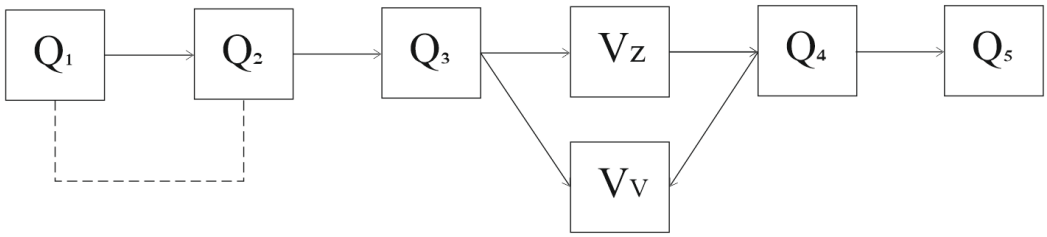


Рисунок 3.1 – Схема зв’язку кроків в методі головних компонент

Елементи матриці факторного відображення Q_4 є ваговими коефіцієнтами. Спочатку матриця Q_4 має розмірність $k \times f$ – за кількістю елементарних ознак. В процесі перетворень залишаються тільки найвагоміші ознаки і розмірність стає меншою.

Метод головних компонент належить до статистичних методів факторного аналізу, в якому аналізується вплив окремих факторів на результуючий показник. Головні компоненти обчислюються як власні вектори і власні значення коваріаційної матриці початкових даних. Завданням аналізу головних компонент є апроксимування даних лінійними комбінаціями меншої розмірності або знаходженням підпростору меншої розмірності.

В розробленій самоорганізованій розподіленій системі наявні компоненти,

які розміщені у вузлах в мережі. Кожна з компонент має детектори різного виду, які збирають визначену інформацію для подальшої обробки в центрі. Ця інформація отримується шляхом надходження потоків даних певного часового ряду. В кожному вузлі в мережі у визначені часові інтервали відбувається збір даних одночасно.

Зібраною інформацією може бути така: всі виконувані файли в комп'ютері з даними про час їх створення чи останньої зміни, а також, про їх розмір; кількість TCP-запитів на встановлення з'єднання за секунду; кількість транзакцій за хвилину; обсяг трафіку в портах за секунду; кількість запущених процесів; обсяг вільної внутрішньої пам'яті; обсяг жорсткого диску і вільний простір в ньому. Важливим є, також, врахування можливості порівняння отриманих результатів з попередніми результатами згідно часових зрізів. Самоорганізована розподілена система зберігає зібрані і оброблені данні. Потім при здійсненні аналізу враховує останні отримані значення ознак і попередні, тобто обробляє їх з певним часовим вікном. Центр прийняття рішень в самоорганізованій розподіленій системі здійснює контроль загальної множини значень ознак тимчасового ряду та приймає рішення щодо питань безпеки окремих вузлів та усієї комп'ютерної мережі. Для цього системі потрібно визначати об'ємні аномалії. Аномальні прояви в мережі відносяться до незвичних рівнів навантаження, які викликані worm-вірусами, розподіленими атаками, відмовою в обслуговуванні, відмовою пристроїв, неправильними конфігураціями, поширенням зловмисного програмного забезпечення у вузлах і в мережі в цілому тощо. В кожному вузлі здійснюється збір інформації за заданими ознаками і на певному часовому кроці ці результати збору надсилаються в центр обробки розподіленої системи. Центр прийняття рішень розподіленої системи відстежує в ковзному вікні часу розміру r для кожного часового ряду з кожного вузла в мережі. Число r вказує на кількість найостанніших даних серед всіх даних, отриманих з вузла в мережі і які зберігаються в центрі самоорганізованої розподіленої системи. Ці данні представлені у відповідному часовому ряді. Позначимо ці данні з врахуванням формули (1) і того, що такі вектори будуть отримуватись в різні часові проміжки, а також, з врахуванням того, що кількість

різних вузлів в мережі становить k. Нехай $V_{f,j}$ – вектор ознак з j – того вузла в мережі. Тоді, скінчену множину різних даних з j – того вузла в мережі представимо впорядкованою послідовністю:

$$\left(v_{1,j}, v_{2,j}, \dots, v_{f_j,j}\right), \tag{3.2}$$

де $v_{i,j}$ – значення даних з досліджуваної i - тої компоненти з j – того вузла в мережі; $i = 1, 2, \dots f_j$; f_j – кількість значень даних ознак з j – того вузла в мережі; $j = 1, 2, \dots, k$; k – кількість вузлів в мережі.

Отримані вектори з окремих вузлів в мережі представимо в матрицях W_q, де q – номер вузла в мережі з 1 до k. Матриці W_q мають розмірність $t_i \times f_j$, де t_i – час, в який фіксувались результати збору даних ознак з усіх вузлів в мереді одночасно, i – кількість зрізів за часом протягом всього часу спостереження; f_j – кількість значень даних ознак з j – того вузла в мережі; $j = 1, 2, \dots, k$; k – кількість вузлів в мережі. Не всі комп'ютери під'єднані до мережі можуть бути увімкнені одночасно. Так само і не всі з них можуть вимикатись одночасно. Тому, відлік береться з основного комп'ютера, де функціонує центр системи. Фактично центр системи здійснює поділ на часові проміжки і формування часового ряду. Якщо певні комп'ютери вимкнені, тоді данні в системі за всіма їх показниками у векторах ознак будуть дорівнювати -1. Нулі, як числові значення, не можуть бути використані, бо вони можуть бути значущими результатами ознак. Важливим, також, є відображення в матрицях W_q зв'язків, які можуть між компонентами векторів. Наприклад, час та розмір файлу є різними компонентами вектора, але відносяться до одного об'єкту. Тому, для відображення такої інформації потрібно підтримувати актуальні данні, наприклад, у векторі початкових (ініціалізованих) даних ознак, яким може мати ту ж розмірність, але значення компонент визначатись може за формулою при t_0:

$$V_{f_j} = \left(v_{1,j}, v_{2,j}, \dots, v_{f_j,j}\right); \tag{3.3}$$

$v_{i,j} = v_{i+1,j}, v_{i,j} = i$, якщо сусідні ознаки стосуються одного об′єкту,
якщо кожна наступна ознака теж стосується цього ж об′єту,
то її значення встановлюється теж i;
$v_{i,j} = 0$, якщо сусідні (правий і лівий) ознаки стосуються різних об′єктів;

де $v_{i,j}$ – значення даних з досліджуваної i - тої компоненти з j – того вузла в мережі; $i = 1, 2, \dots f_j$; f_j – кількість значень даних ознак з j – того вузла в мережі; $j = 1, 2, \dots, k$; k – кількість вузлів в мережі.

Задамо матрицю значень показників ознак W_j з j – того вузла в мережі так:

$$W_j = \begin{pmatrix} v_{1,j,t_0} & v_{2,j,t_0} & \cdots & v_{f_j,j,t_0} \\ v_{1,j,t_1} & v_{2,j,t_1} & \cdots & v_{f_j,j,t_1} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1,j,t_g} & v_{2,j,t_g} & \cdots & v_{f_j,j,t_g} \end{pmatrix}, \tag{3.4}$$

де v_{i,j,t_s} – значення даних з досліджуваної i - тої компоненти з j – того вузла в мережі в момент часу t_s; $i = 1, 2, \dots f_j$; f_j – кількість значень даних ознак з j – того вузла в мережі; $j = 1, 2, \dots, k$; k – кількість вузлів в мережі; s – номер, який відповідає ітерації отримання значень ознак; $s = 0, 1, 2, \dots, g$; g – остання ітерація отримання значень ознак з j – того вузла в мережі.

Матриці W_j для певних j – тих вузлів в мережі в системі надають можливість представляти зібрані значення ознак. На рис. 3.2 зображено їх як частину самоорганізованої розподіленої системи.

Матриці W_j для певних j – тих вузлів в мережі можуть мати різну кількість стовпців, тобто числа f_j у всіх матрицях будуть переважно різними, бо це пов’язано з кількістю об’єктів, які включено для моніторингу у вузлі в мережі. Можна вибрати при налаштуванні самоорганізованої розподіленої системи, в яку буде імплементовано метод головних компонент, щоб кількість ознак взятих з різних

вузлів в мережі була однаковою, тобто вибрати спільні для дослідження. Але такий підхід ускладнить покращення виявлення аномалії, бо при ньому може втратитись суттєва частина об’єктів, в яких будуть прояви аномальних станів. Тому, в різних вузлах в мережі кількість ознак для моніторингу буде вибиратись різною. Таким чином, всі матриці будуть мати різну кількість стовпців, причому кількість стовпцв буде скінченою, але може бути дуже великою, бо міститиме інформацію про багато ознак і їх значень.

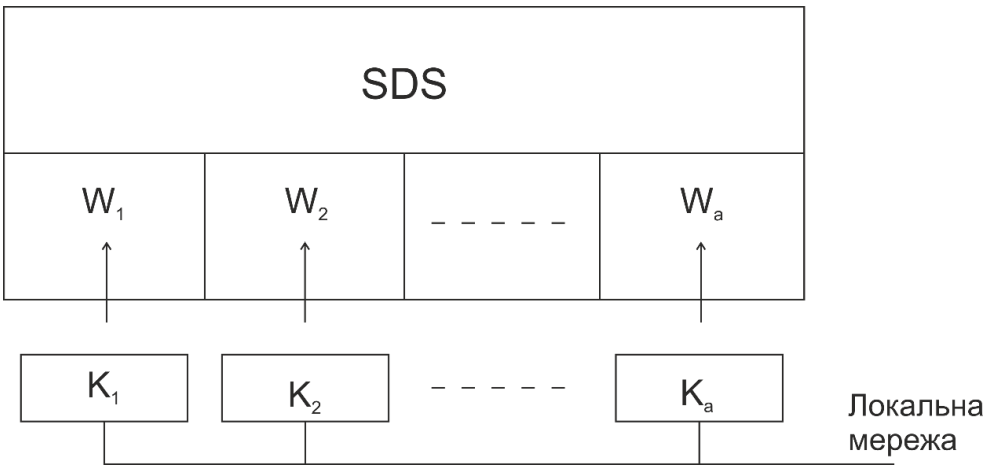


Рисунок 3.2 - Матриці W_j в архітектурі самоорганізованої розподіленої системи

Кількість рядків матриці W_j для різних j – тих вузлів в мережі буде однакова, бо значення з них будуть отримуватись за командою в конкретні моменти часу для всіх однаково. Іншою проблемою, яка виникатиме в цьому процесі буде невпинне накопичення інформації. При цьому матриці W_j для різних j – тих вузлів в мережі будуть збільшуватись динамічно за кількістю рядків, що потребуватиме відповідного врахування при реалізації.

Таким чином, накопичена інформація в матрицях W_j для різних j – тих вузлів в мережі є необхідною для застосування методу головних компонент з метою виявлення аномальних проявів. В матрицях W_j для кожного j – тих вузла в мережі зберігаються часові ряди. Ковзаючим вікном буде інтервал $]t_i;\ t_l[$, де i та l – номери часових показників для отримання значень ознак з різних j – тих вузлів в мережі. В подальшому кожна з матриць W_j повинна переходити в матрицю Q_1 з рис.3.1 для

початку застосування методу головних компонент.

3.2 Метод головних компонент

Інформація в матрицях W_j для кожного j – тих вузла в мережі є різнорідною і не дозволяє без відповідної обробки здійснити висновок про наявність чи відсутність аномальних проявів. Самоорганізована розподілена система для виявлення аномалії в мережі здійснює вимірювання сумарного обсягу трафіку (в байтах) для кожного мережного з'єднання і періодично збирає дані в центрі. Потім, застосовується в автоматичному режимі, імплементований в неї, метод головних компонент з врахуванням даних зібраних в матрицю (3.4). Аналогічно опрацьовуються системою дані ознак про файли, процеси та завантаженість ресурсів комп’ютерів. Для оперативного опрацювання даних потрібно, щоб періоди їх поновлення були відносно малими. Хоча малі періоди призводять до збільшення додаткових витрат на залучення ресурсів та здійснення комунікаційних робіт.

Розглянемо суть методу головних компонент і його особливості при застосуванні до виявлення аномалій. Геометрична інтерпретація цього методу така; не площині розміщені точки; проведено пряму; визначено відстані від точок до прямої, тобто з точок на пряму проведено проєкції; визначено суму квадратів проєкцій точок на пряму; здійснення пошуку нової прямої, щоб сума квадратів проєкцій точок на пряму була мінімальною. Аналогічно, постановка цієї задачі масштабується на трьохмірний простір та n-мірний. При вдалій спробі знаходження такої прямої, кількість точок, які впливатимуть на величину суми квадратів проєкцій зменшиться, бо частина з них буде знаходиться на цій новій прямій. Ті ж точки, які будуть віддалені від прямої, впливатимуть на результат суми квадратів проєкцій та будуть значимими. В задачі виявлення аномалій значення враховуваних ознак може бути задано теж точками на площині і пошук прямої для розв’язання задачі найкращої апроксимації скінченої множини точок дозволяє

визначити ті точки, тобто ознаки, які суттєво впливають на результат суми, а отже є такими, що можуть бути аномальними проявами.

Здійснимо формальну постановку задачі апроксимації і перейдемо від неї до методу головних компонент. Наприклад, нехай дано скінчену кількість векторів $v_{1,j}, v_{2,j}, \dots, v_{f_j,j} \in R^u$, де вектори $v_{1,j}, v_{2,j}, \dots, v_{f_j,j}$ відповідають значенням ознак з j – того вузла, f_j – кількість ознак, тобто векторів. Для кожного $p = 1, 2, \dots, u-1$ потрібно знайти $M_p \subset R^u$ з p - мірних лінійних комбінацій в R^u і за умови, щоб сума квадратів відхилень $v_{i,j}$ від M_p була мінімальною. Зокрема, формальний запис такої задачі задамо формулою:

$$\sum_{i=1}^{f_j} d^2\left(v_{i,j}, M_p\right) \to min, \tag{3.5}$$

де $d\left(v_{i,j}, M_p\right)$ – відстань від точки $v_{i,j}$ до лінійного комбінації M_p.

Відстань від точки до лінійної комбінації може визначатись різними метриками, зокрема і евклідовою. Задамо ортонормований набір векторів $\alpha_1, \alpha_2, \dots, \alpha_{f_j} \in R^u$, тоді лінійні комбінації M_p для всіх $p = 1, 2, \dots, u-1$ задамо за формулою:

$$M_p = \alpha_0 + \sum_{i=1}^{p} \beta_i * \alpha_i, \tag{3.6}$$

де $\beta_i \in R$ і є коефіцієнтами в лінійному розкладі M_p.

Задача апроксимації для кожного $p = 1, 2, \dots, u-1$ розв'язується шляхом знаходження лінійних комбінацій $M_1 \subset M_2 \subset M_3 \subset \cdots \subset M_{u-1}$, де M_p представляється за формулою (3.6). За формулою (3.5) необхідно мати значення додатків, що відповідають за відстані. Представимо визначення доданків з формули (3.5) з врахуванням заданих представлень лінійних комбінацій за формулою:

$$d^2\left(v_{i,j}, M_p\right) = \left|v_{i,j} - \alpha_0 - \sum_{s=1}^{p} \alpha_s \cdot \left(\alpha_s, v_{i,j} - \alpha_0\right)\right|^2, \tag{3.7}$$

де квадрат відстані $d^2(v_{i,j}, M_p)$ визначається за евклідовою нормою; вираз $(\alpha_s, v_{i,j} - \alpha_0)$ визначається як скалярний добуток векторів α_s та $v_{i,j} - \alpha_0$.

Всі лінійні комбінації M_p визначаються ортонормованим набором векторів $\{\alpha_1, \alpha_2, \dots, \alpha_{f_j-1}\}$, які є векторами головних компонент, і вектором α_0. Знаходження вектора α_0 здійснюємо за формулою:

$$\alpha_0 = \arg \min_{\alpha_0 \in R^u} \left(\sum_{s=1}^{f_j} d^2(v_{s,j}, M_0) \right). \quad (3.8)$$

Згідно формули (3.8) отримуємо, що α_0 обчислюється за формулою:

$$\alpha_0 = \frac{1}{f_j} \sum_{i=1}^{f_j} v_{i,j} = \overline{v_j}. \quad (3.9)$$

Таким чином, α_0 мінімізує суму квадратів відстаней до точок даних, тобто значень ознак і згідно формули (3.9) є середнім значенням.

Для знаходження векторів головних компонент потрібно виконати таку послідовність кроків задачі оптимізації:

1) всі $v_{i,j}$ зменшуємо на величину $\overline{v_j}$, тоді сума всіх отриманих $v_{i,j}$ дорівнюватиме нулеві;

2) першу головну компоненту обчислюємо за формулою:

$$\alpha_1 = \arg \min_{|\alpha_1|=1} \left(\sum_{s=1}^{f_j} \left| v_{s,j} - \alpha_1 \cdot (\alpha_1, v_{s,j}) \right|^2 \right).$$

При отриманні декількох розв'язків для подальших обчислень вибираємо один з них;

3) з даних віднімаємо проекцію на першу головну компоненту за формулою:

$$v_{i,j} = v_{i,j} - \alpha_1 \cdot (\alpha_1, v_{i,j});$$

4) другу головну компоненту знаходимо за формулою:

$$\alpha_2 = \arg \min_{|\alpha_2|=1} \left(\sum_{s=1}^{f_j} \left| v_{s,j} - \alpha_2 \cdot (\alpha_2, v_{s,j}) \right|^2 \right).$$

При отриманні декількох розв’язків для подальших обчислень вибираємо один з них;

5) аналогічно до кроку 3 з даних віднімаємо проекцію на p-1 головну компоненту за формулою:

$$v_{i,j} = v_{i,j} - \alpha_{p-1} \cdot (\alpha_{p-1}, v_{i,j});$$

6) p-ту головну компоненту знаходимо за формулою:

$$\alpha_p = \arg \min_{|\alpha_p|=1} \left(\sum_{s=1}^{f_j} \left| v_{s,j} - \alpha_p \cdot (\alpha_p, v_{s,j}) \right|^2 \right).$$

При отриманні декількох розв’язків для подальших обчислень вибираємо один з них.

Для кожного ітераційного етапу застосування алгоритму здійснюється віднімання проєкції на попередню головну компоненту. Отримані таким чином вектори $\{\alpha_1, \alpha_2, \dots, \alpha_{u-1}\}$ будуть ортонормованими. Це досягається в результаті розв’язання задачі оптимізації. З метою уникнення помилок обчислень через вплив заокруглень використовують включення в умови задачі оптимізації такої умови:

$$\alpha_p \perp \{\alpha_1, \alpha_2, \dots, \alpha_{p-1}\}. \quad (3.10)$$

Обчислення α_i можуть здійснювати і іншими способами. Зокрема, перша головна компонента максимізує вибіркову дисперсію проекції даних. Тому фактично, як і для задачі апроксимації, розв’язування якої подано в представленому алгоритмі, на кожному кроці ітерації потрібно обчислювати першу головну компоненту для даних, з яких вилучені проекції на всі раніше знайдені головні компоненти. Задачі про обчислення головних компонент зводяться до задачі діагоналізації коваріаційної матриці. Базис з власних векторів представлено у коваріаційній матриці, тому матриця діагональна. В такому випадку коефіцієнт коваріації між різними координатами дорівнює нулю. Математичним змістом методу головних компонент є спектральне розкладання коваріаційної матриці, але при певних перетвореннях ця задача перетворюється в задачу про сингулярне розкладання матриці даних. Хоча формально завдання сингулярного розкладання

матриці даних і спектрального розкладання коваріаційної матриці збігаються, алгоритми обчислення сингулярного розкладання безпосередньо, без обчислення коваріаційної матриці і її спектра, більш ефективні і стійкі. Це необхідно при використанні методу головних компонент для розв'язування прикладних задач певної предметної області з інформаційних технологій, де при великих наборах вхідних даних можуть виникати помилки обчислень або збільшуватись тривалість обчислень, тоді потрібно вибирати більш стійкіші алгоритми та з високою швидкодією.

Для перетворення даних до головних компонент будуємо матрицю з векторів головних компонент таким чином, щоб ортонормовані вектори-стовпці головних компонент були розташовані в порядку спадання власних значень. Це дає змогу після перетворення зосередити велику частину варіації даних в перших координатах. В результаті можна відкинути ті, що залишилися, і отримати простір зменшеної розмірності.

Таким чином, метод головних компонент надає можливість зменшити розмірність даних, що є важливим при виявленні зловмисного програмного забезпечення та комп'ютених атак, бо необхідно обробити багато початкової різнотипної інформації, яка динамічно накопичується.

3.3 Удосконалення методу централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент

Використання самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах надає можливість проводити пошук безпосередньо в одній комп'ютерній станції або одночасно в декількох. При цьому в обох випадках можна використати метод головних компонент в якості покрокового ітераційного алгоритму для отримання числових значень показників характерних ознак безпосередньо отриманих в одній комп'ютерній станції та декількох напротязі певного часу в деякому ковзному вікні. Також, отриману з вузлів в мережі

інформацію про процеси, що протікають в них, самоорганізована розподілена система виявлення аномалій може досліджувати на предмет проявів аномалій, які відповідатимуть або зловмисному програмному забезпеченню або комп'ютерним атакам. Враховуючи складність виявлення зловмисного програмного забезпечення чи комп'ютерних атак через обмеженість кількості ознак за якими можна встановити аномальні прояви, а також, наявність надмірних обсягів різнотипної та різнорідної інформації зібраної з вузлів в мережі, необхідним є удосконалення методу централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент, який дозволив би зменшити розмірність інформації зібраної у вузлах в мережі без втрати її цінності та швидко обробити в єдиному центрі для забезпечення актуальності результату виявлення, що в результаті б покращило ефективність виявлення.

Для забезпечення виявлення розподілених аномалій з використанням методу централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент, розробимо метод виявлення аномалій в одній з комп'ютерних станцій в мережі з врахуванням інтеграції цього методу в розроблену самоорганізовану розподілену систему з єдиним центром.

Збільшення активності в мережі до її вузлів виступає ознакою того, що це можуть бути зловмисні прояви і їх необхідно досліджувати. В реальному часі підвищена активність швидко змінюється на помірну, тому потрібні ефективні засоби і реалізовані в них методи, які б швидко реагували на такі події. Інакше, актуальність отриманої системою інформації про активність в мережі спрямовано до її вузла, а також, реакція на неї втрачатимуть необхідність. Основною ознакою, яку необхідно досліджувати першочергово в мережі, що фактично відповідає за підвищену активність, є обсяг трафіку. Відомо багато методів обробки трафіку мережі, причому з врахуванням різних топологій мереж та каналів входження в корпоративні чи локальні мережі. Зокрема, враховують також особливості трафіку при здійсненні розподілених атак і пошук самоподібності в його частинах.

Поняття мережного трафіку в постановці задачі дослідження аномалій включає дослідження кількості даних, що переміщуються в мережі протягом

певного часу. Для правильного функціонування комп’ютерних мереж потрібно здійснювати в них контроль, аналіз, моделювання та управління відповідними спеціалізованими засобами. Особливо важливими в процесі виявлення аномалій є здійснення аналізу та вимірювання мережного трафіку, що включають моніторинг трафіку, зміни в ньому, тенденції, вимірювання кількості та виду трафіку. Отримання звітів різними спеціалізованими засобами про мережний трафік дає інформацію щодо запобігання зловмисним проявам та дозволяє забезпечити безпеку в мережі. Фрагменти обсягу трафіку даних протягом певного часу (три різних часових інтервали) в мережі Хмельницького національного університету зображено на рис. 3.3. Як видно із графіків в часових інтервалах обсяг трафіку змінюється і може суттєво відхилятись від середнього значення, що можна використати для встановлення аномальних проявів.

Пересилання даних в комп'ютерних мережах здійснюється переважно в мережних пакетах. Ці пакети забезпечують навантаження в мережі. Варіантів передачі пакетів може бути багато і здійснюється за протоколами мережі. По прибуттю за місцем призначення в залежності від правил і протоколів пакети потребують здійснення перевірки наявності всіх, контролю цілісності і джерела надходження. Якщо розглядати трафік в магістральних лініях, то аномалії його обсягу можуть залишатися непоміченими через укрупнене представлення. Результати вимірювань можуть мати велику розмірність, в залежності від кількості ліній, але нормальні моделі трафіку знаходяться в підпросторі меншої розмірності. Виділення цього підпростору мережного трафіку, використовуючи метод головних компонент в трафіку, дає змогу ідентифікувати аномалії обсягу в підпросторі.

Для аналізу аномальних проявів в мережному трафіку спочатку використаємо такі характеристичні параметри: коефіцієнт завантаження трафіку; типовий розмір пакету; середнє число фрагментованих пакетів. Для дослідження коефіцієнту завантаженості трафіку мережі розглянемо такі варіанти: мережний трафік протягом певного часу розкладається в часовий ряд; мережні трафіки порівнюється за певні часові періоди.

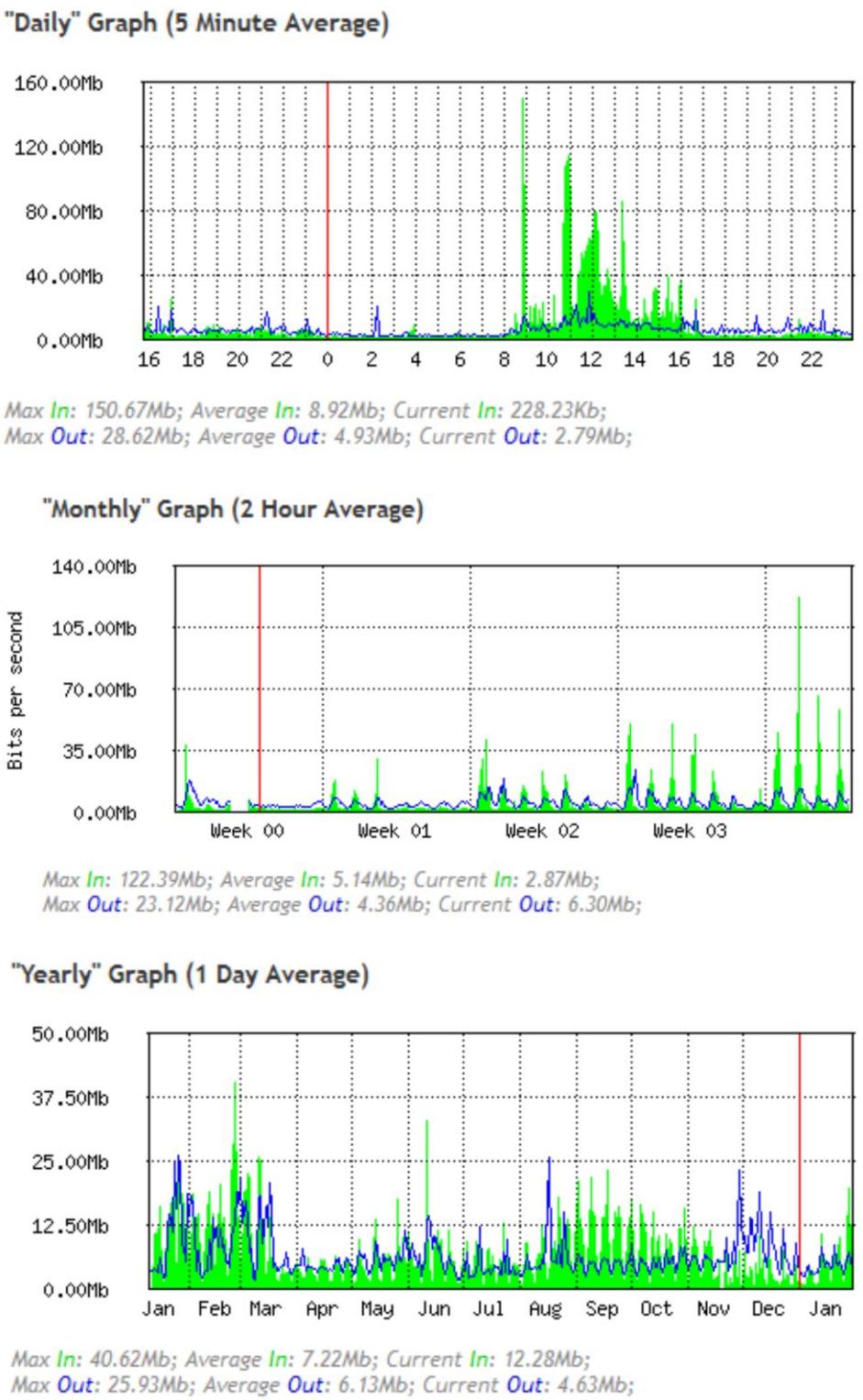


Рисунок 3.3 - Зображення обсягу трафіку даних протягом певного часу (три різних часових інтервали) в мережі Хмельницького національного університету

Якщо мережний трафік отримується динамічно протягом певного часу, то розкладемо його в часовий ряд. Наприклад, нехай дано скінчену кількість векторів

для його представлення $v_{1,j}, v_{2,j}, \dots, v_{f_j,j} \epsilon R^u$, де вектори $v_{1,j}, v_{2,j}, \dots, v_{f_j,j}$ відповідають значенням ознак трафіку з j – того вузла в мережі, f_j – кількість ознак, тобто векторів. Для випадку представлення трафіку, приклад якого зображено графіками на рис. 3.3, через час його надходження та обсяг в конкретний момент часу, отримуємо, що значення f_j= 2. Тоді, пара векторів $v_{1,j}, v_{2,j}$ представлятиме трафік мережі протягом часу, який задано вектором $v_{1,j}$. Сумарний обсяг трафіку в конкретний момент часу представлятиметься вектором $v_{2,j}$ і вимірюватимемо в байтах для всіх з'єднань. Таким чином, кожна точка графіку, що представляє обсяг мережного трафіку, задається парою значень. За певний інтервал часу самоорганізованою розподіленою системою виявлення аномалій з певними сталими періодами часу здійснюється збір цих пар точок. Нумерація точок розпочинається з першої отриманої пари і продовжується до тієї точки, яка є останньою з очікуваних точок. Після того, як зібрано задану кількість пар, система здійснює їх централізацію. Представлені таким чином данні є двомірними.

Здійснимо представлення пари векторів для визначеної кількості q точок спостереження так:

$$\begin{pmatrix} v_{1,j.1} & v_{1,j.2} & \dots & v_{1,j.q} \\ v_{2,j.1} & v_{2,j.2} & \dots & v_{2,j.q} \end{pmatrix} \quad (3.11)$$

Після отримання q пар системою і обробкою, синхронно обсяг трафіку мережі продовжує відображатись системою в подальших парах. Самоорганізована розподілена система після обробки певного набору даних, які задано формулою (3.11), отримує частину даних фактично оновлених і залишає з оброблених даних пари, які надійшли останніми. Перші пари точок, після обробки, видаляються з подальших обчислень. Кількість таких видалених пар залежить від часу обробки системою та часу, який витрачений на пересилання даних. Якщо витрачений час більше, ніж час витрачений на збір q нових пар системою, тоді ці зібрані нові пари втрачаються, бо розпочнеться новий набір наступних пар. Для вирішення цієї

проблеми необхідно збільшувати інтервал збору сусідніх пар векторів. Розрахунок інтервалу часу між сусідніми парами здійснимо так:

$$t_{int,1} = \frac{t_{obr,1}+t_{dos,1}+t_{dod,1}}{q}, \quad (3.12)$$

де $t_{int,1}$ – інтервал часу між отриманням даних обсягу трафіку, тобто між сусідніми значеннями $(v_{1,j,i-1}, v_{2,j,i-1})$ та $(v_{1,j,i}, v_{2,j,i})$; $t_{obr,1}$ – час, який витрачено на обробку даних з матриці (3.11); $t_{dos,1}$ – час, який витрачено на переміщення даних до центру самоорганізованої розподіленої системи; $t_{dod,1}$ – додаткові часові втрати, які пов'язані з затримками в обробці даних через вищу пріоритетність інших задач.

Додаткові часові втрати оцінюються експериментально і встановлюються значенням, яке є максимальним зі всіх досліджуваних. Але ці додаткові часові втрати не можуть перевищувати час обробки даних $t_{obr,1}$, тобто $t_{obr,1} \geq t_{dod,1}$. Тому, з самого початку цей час може бути заданий, як такий що дорівнює часу обробки, тобто $t_{dod,1} = t_{obr,1}$. На кожному етапі обробки даних з матриці (3.11) центр системи фіксуватиме значення додаткових часових втрат $t_{dod,1}$ та усереднюватиме його з попередніми значеннями, якщо з самого початку перше таке значення дорівнювало $t_{obr,1}$.

Аналогічно проводимо оцінку для часу, який витрачено на переміщення даних до центру самоорганізованої розподіленої системи $t_{dos,1}$. Приймемо з самого початку його таким що дорівнює часу обробки, тобто $t_{dos,1} = t_{obr,1}$. В подальшому усереднюватимемо його зі всіма попередніми значеннями і отримаємо оцінку цього часу.

Результатом такого підходу буде визначення початкового інтервалу часу між отриманням даних обсягу трафіку, тобто між сусідніми значеннями $(v_{1,j,i-1}, v_{2,j,i-1})$ та $(v_{1,j,i}, v_{2,j,i})$, який визначатиметься в залежності від часу обробки так:

$$t_{int,1} = \frac{3*t_{obr,1}}{q}. \quad (3.13)$$

Наслідком з формули (3.13) є те, що при значенні інтервалу часу між отриманням даних обсягу трафіку суттєво меншому за значення $\frac{3*t_{obr,1}}{q}$ за певний час надходження мережного трафіку частина значень не буде врахована при обчисленні та очікувані результати обчислень втратять актуальність, бо процес надходження мережного трафіку є динамічним. Ці обчислення стосуються дослідження мережного трафіку за обсягами, що надходять.

Якщо значення інтервалу часу відповідає вимогам, тоді переходимо до обчислення, зокрема першочергово до централізації отриманих даних. Централізація потрібна, щоб в подальшому оперувати з середніми значеннями близькими до нуля або взагалі нульовими. Для здійснення централізації даних знаходимо середнє значення серед всіх значень кожної з характеристичних ознак і від кожного значення із отриманого набору (формула (3.11)) віднімає середня значення, що відноситься до його характеристичної ознаки. Геометричний зміст централізації означає переміщенню центра координат в нову точку, таким чином, що вся вибірка буде розміщена в межах цього нового центру. Це перепредставлення даних з матриці (3.11) дає можливість не тільки зменшити розрядність в числах, але і відображає розсіювання вибірки. Тому, знаходимо середні значення вибірки з даних представлених в формулі (3.11) так:

$$v_{1,j,s} = \frac{1}{q}\sum_{i=1}^{q} v_{1,j,i},\ v_{2,j,s} = \frac{1}{q}\sum_{i=1}^{q} v_{2,j,i}, \tag{3.14}$$

де $v_{1,j,s}$ – середньоарифметичне значення аргументів першого вектора $v_{1,j}$ з j – ої комп'ютерної станції; $v_{2,j,s}$ – середньоарифметичне значення аргументів першого вектора $v_{2,j}$ з j – ої комп'ютерної станції; q – кількість значень.

Згідно значень формул (3.11) та (3.14) здійснимо централізацію значень векторів так:

$$v_{1,j,i,c} = v_{1,j,i} - v_{1,j,s},\ v_{2,j,i,c} = v_{2,j,i} - v_{2,j,s}, \tag{3.15}$$

де $v_{1,j,i,c}$ – централізоване значення першого вектора $v_{1,j}$ з j – ої комп'ютерної станції; $v_{2,j,i,c}$– централізоване значення першого вектора $v_{2,j}$ з j – ої комп'ютерної станції; $i = 1, 2, \dots, q$; q – кількість значень.

Для отриманих централізованих даних будуємо коваріаційну матрицю, яка описує сумісне чередування декількох змінних, зокрема для розглядуваного випадку – двох змінних. Головна діагональ матриці коваріацій містить дисперсії ознак, а інші елементи містять коваріації один з одним. Дисперсію ознак визначаємо так:

$$s^2_{v_{1,j}} = \frac{1}{q-1} \sum_{i=1}^{q} (v_{1,j,i,c})^2, s^2_{v_{2,j}} = \frac{1}{q-1} \sum_{i=1}^{q} (v_{2,j,i,c})^2, \quad (3.16)$$

де $s^2_{v_{1,j}}$ – квадрат дисперсії значень вектора $v_{1,j}$ з j – ої комп'ютерної станції; $s^2_{v_{2,j}}$ – квадрат дисперсії значень вектора $v_{2,j}$ з j – ої комп'ютерної станції; $i = 1, 2, \dots, q$; q – кількість значень.

Коефіцієнт коваріації ознак векторів $v_{1,j}$ та $v_{2,j}$ між собою визначимо за формулою так:

$$cov(v_{1,j}, v_{2,j}) = \frac{1}{1-q} \sum_{i=1}^{q} (v_{1,j,i,c} \cdot v_{2,j,i,c}), \quad (3.17)$$

де $cov(v_{1,j}, v_{2,j})$ – коваріація ознак векторів $v_{1,j}$ та $v_{2,j}$ між собою.

Коефіцієнт парної кореляції ознак векторів $v_{1,j}$ та $v_{2,j}$ визначаємо за формулою так:

$$r_{v_{1,j},v_{2,j}} = \frac{cov(v_{1,j},v_{2,j})}{s_{v_{1,j}} \cdot s_{v_{2,j}}}, \quad (3.18)$$

де $r_{v_{1,j},v_{2,j}}$ – коефіцієнт кореляції.

Побудуємо коваріаційну матрицю $cov(K)$ так:

$$cov(K) = \begin{pmatrix} s^2_{v_{1,j}} & s_{v_{1,j},v_{2,j}} \\ s_{v_{2,j},v_{1,j}} & s^2_{v_{2,j}} \end{pmatrix}, \quad (3.19)$$

де $s_{v_{2,j},v_{1,j}}$ та $s_{v_{1,j},v_{2,j}}$ - коваріації між значеннями компонентів векторів; $s_{v_{1,j},v_{1,j}} = s^2_{v_{1,j}}$; $s_{v_{2,j},v_{2,j}} = s^2_{v_{2,j}}$; $s_{v_{2,j},v_{1,j}} = s_{v_{1,j},v_{2,j}}$.

Для оцінки внеску головних компонент в загальну мінливість знайдемо власні значення. Дисперсія вздовж власних векторів є пропорційною їх власним значенням. Власних векторів буде стільки, скільки початкових змінних, тобто для розглядуваного прикладу їх буде два. Власні вектори будуть перпендикулярними між собою і вони задають напрями осей головних компонент. Вздовж першого вектора буде відкладена максимальна дисперсія даних, в вздовж наступного власного вектора – максимальна дисперсія з тих значень що залишились.

За допомогою власних значень і власних векторів знаходимо в просторі ознак нові осі, вздовж яких буде максимальне розсіювання точок. В результаті знаходимо нові координати точок в новому координатному просторі. Власні значення та власні вектори знаходимо з характеристичного рівняння так:

$$det(cov(K) - \lambda \cdot E) = 0, \quad (3.20)$$

де $det(cov(K) - \lambda \cdot E)$ – це детермінант матричного виразу $(cov(K) - \lambda \cdot E)$; λ – набір власних значень.

В розглядуваному випадку кількість значень λ буде два. Якщо врахувати третю ознаку в мережному трафіку, яка вказує на кількість мережних пакетів, то кількість значень λ буде три і, відповідно, власних векторів буде три. Аналогічно, в залежності від кількості ознак факторів, кількість значень λ може бути більшою.

Знаходження власних значень і власних векторів за формулою (3.20) дає змогу зменшити розмірність та визначити найбільш суттєві ознаки факторів.

За даними збору мережного трафіку, які представлено в матриці, що відображають данні часового ряду. Здійснимо центрування даних матриці для отримання нульового середнього значення. Кількість рядків матриці відображатиме кількість об'єктів, з яких отримуються данні. Причому кожен рядок матриці відповідатиме вектору вимірювань для всіх ліній за один крок вимірювань за певним часом. Збір даних здійснюється самоорганізованою розподіленою системою в центр, в якому відбувається обробка. Нехай скінчена кількість векторів $v_{1,j}, v_{2,j}, \dots, v_{f_j,j} \in R^u$, де вектори $v_{1,j}, v_{2,j}, \dots, v_{f_j,j}$ відповідають значенням мережного трафіку з магістральних ліній, данні про які зібрано в j – ту комп'ютерну станцію, а f_j – кількість магістральних ліній, данні з який отримуються в j – ту комп'ютерну станцію. Тоді, вектор $v_{i,j}$ позначає вектор вимірювань так:

$$v_{i,j} = v_{i,j}(t), \tag{3.21}$$

де t – час, коли відбулось вимірювання.

Самоорганізована розподілена система здійснює в центрі усереднення отриманих значень виконує метод головних компонент для отриманих даних та здійснює обчислення за методом головних компонент в кожній своїй компоненті, що розміщена в різних комп'ютерних станціях. Отримана інформація з комп'ютерних станціій про результати обробки надсилається в центр системи і там приймається рішення про наявність аномальних проявів. На відміну від класичного підходу з одним центром прийняття рішення, в такому варіанті з'являються варіації даних, що пов'язано з можливою неповнотою даних, що отримуються в усі комп'ютерні станції. Це надає змогу здійснити обчислення результатів в різних компонентах самоорганізованої розподіленої системи і обробити отримані результати з більшою точністю достовірності. Схема основних кроків удосконаленого методу представлена на рисунку 3.4.

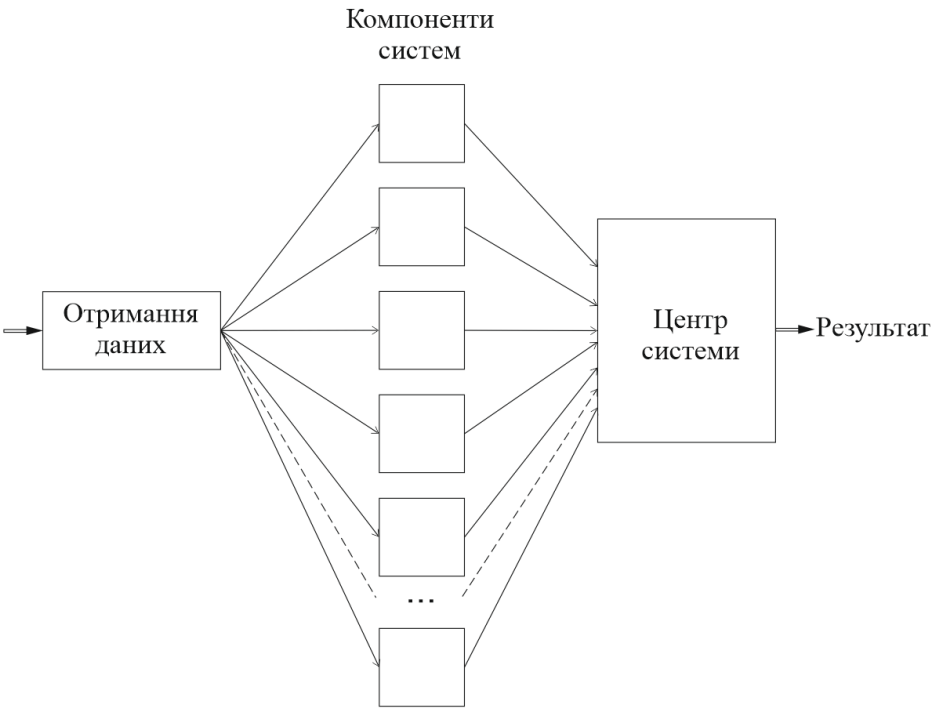


Рисунок 3.4 - Схема основних кроків удосконаленого методу

В результаті обробки даних ознак зібраних у місці призначення мережного трафіку, встановлено низьку внутрішню розмірність магістралей. Базові нормальні потоки трафіку ефективно знаходяться в низькому l-мірному підпросторі, який називається нормальним підпростором трафіку. Ті (n - l) головні компоненти, які залишаються, складають аварійний підпростір мережного трафіку. Виявлення аномалій обсягу мережного трафіку покладається на розкладання потоку трафіку $v_j = v_j(t)$, який отримується і обробляється в j – комп'ютерній станції, в будь-який час на компоненти, які поділимо на нормальні та аварійні. Представимо їх так:

$$v_j = v_{j,n} + v_{j,a}, \tag{3.22}$$

де $v_{j,n}$ - відповідає змодельованому нормальному трафіку, тобто отримана проєкція v_j на нормальний підпростір; $v_{j,a}$ - відповідає залишковому трафіку, тобто отримана проєкція v_j на аварійний підпростір.

Обчислення значень $v_{j,n}$ та $v_{j,a}$ здійснюємо з використанням методу головних

компонент, вибираючи при цьому перші k основних компонентх ті, які отримують домінуючу відмінність в даних. Аномалія обсягу переважно призводить до суттєвої зміни $v_{j,a}$. Тобто за результатами такого розрахунку отримується сигнал про аномалію обсягу. При цьому враховується порогова статистична величина, яка розраховується за певного визначеного довірчого рівня.

Для реалізації запропонованого удосконаленого методу виявлення аномалії за методом головних компонент в комп'ютерних системах в мережі використаємо розподілене відстеження мережного трафіку. Застосування методу головних компонент не до однієї комп'ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп'ютерних системах в мережі. Враховуючи великий обсяг даних, що надходить і потребує оперативного аналізу, необхідно у вузлових компонентах самоорганізованої розподіленої системи застосовувати метод головних компонент, щоб скоротити обсяг даних, які кожна компонента системи обробляє окремо і які надсилає в центр системи. При цьому необхідно, щоб в центрі системи та у її компонентах дійсно здійснювалось виявлення аномалій. Точність відстеження мережного трафіку не є обов'язковим, а може бути приблизним після застосування методу головних компонент. Головне завдання імплементованого в систему методу полягає в виокремленні стану в момент аномального прояву, тобто відстежувати стан дуже точно не потрібно за наявності нормальних умов. Це надає змогу скоротити обсяг даних і відповідно прискорює їх обмін між компонентами системи. Таким чином, удосконалено метод централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент, який на відміну від відомих імплементований в самоорганізовану розподілену систему з центром прийняття рішень та надає змогу визначати аномальні прояви на основі обробки даних в центрі та компонентах системи одночасно з подальшим їх усередненням. В результаті такого застосування методу в частини даних буде зменшено розмірність з моменту отримання, а в другої частини даних після надсилання в центр. Але їх обробка стане уточненням оброблених в центрі даних отриманих після першої обробки в компонентах системи.

3.4 Висновки до третього розділу

Згідно представленого методу головних компонент було удосконалено метод централізованого виявлення розподілених аномалій в комп’ютерних системах за алгоритмом пошуку головних компонент, який на відміну від відомих імплементований в самоорганізовану розподілену систему з центром прийняття рішень та надає змогу визначати аномальні прояви на основі обробки даних в центрі та компонентах системи одночасно з подальшим їх усередненням. В результаті такого застосування методу в частини даних буде зменшено розмірність з моменту отримання, а в другої частини даних після надсилання в центр. Їх обробка стане уточненням оброблених в центрі даних отриманих після першої обробки в компонентах системи.

4 ЕФЕКТИВНІСТЬ ЗАСТОСУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ З ВИКОРИСТАННЯ САМООРГАНІЗОВАНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ ВИЯВЛЕННЯ АНОМАЛІЇ В КОМП'ЮТЕРНИХ СИСТЕМАХ

4.1 Ефективність застосування самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах

Дослідження ефективності застосування самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах є важливим обов'язковим етапом, який проводиться з метою підтвердження коректності, доцільності та можливості реалізації розроблених рішень, зокрема щодо удосконаленої архітектури, методу підтримки цілісності системи, методу виявлення аномалії та реалізованої системи.

Для проведення дослідження з ефективності застосування самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах необхідно визначитись з критеріями для оцінювання. Визначення особливостей застосування системи впливає, також, і на вибір критеріїв для оцінювання ефективності.

Оскільки, розглядувана система є самоорганізованою, тоді вона функціонуватиме фактично при виконанні всіх функцій без втручання користувача і, тому, вимогою критерію, який задовольняє вимоги користувача, є максимізація часу в роботі системи, коли вона не залучає користувача для прийняття рішення про подальші кроки в її роботі або її компонент, а також про зміну її архітектури. В якості змінюваної величини, тобто аргументом, в функції, що описуватиме та задаватиме вимоги в цьому критерії буде час, оскільки його можна вимірювати в процесі функціонування системи. Час можна визначати для визначення всього періоду функціонування системи з початку запуску системи, протягом робочого дня, а також, час витрачений на обробку користувачем певних станів системи. Витрачений час на обробку певних станів системи користувачем (адміністратором системи або мережі) може поділятись на час, який визначається коли система зупинена повністю і коли її частина працює. Такий поділ дозволяє врахувати

особливості процесів, які відбуватимуться протягом періоду функціонування системи, і надає можливість здійснювати співвіднесення різних часових проміжків.

Нехай t_1^1 – час, протягом якого функціонувала самоорганізована розподілена система. Час визначатимемо в годинах. Це увесь час витрачений на роботу всієї системи, який включає функціонування і її окремих модулів в автономному режимі в період відсутності центру і функціонування системи у вимкненому стані, тобто час в пасивному стані між періодами функціонування системи. Таким чином, це час повного циклу експлуатації системи. Введемо $t_{2,i}^1$ в якості часу функціонування системи або окремих її компонентів в активному стані протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи, тоді справедливе співвідношення $t_1^1 =< 24 * d$. Оскільки, протягом доби система може перебувати в активному стані або бути неактивною, коли вимкнені всі комп'ютерні станції, в яких вона встановлена, то доцільно розподілити час між цими випадками, і також, можна розглядати періодичність для неї, якщо в якості часового інтервалу брати добу. В якості часу перебування системи або окремих її компонент у неактивному стані, коли вимкнено комп'ютерні станції, в які вони встановлені, введемо $t_{3,i}^1$ протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи, тоді будуть справедливі такі співвідношення:

$$t_{2,i}^1 + t_{3,i}^1 = 24, (t_{2,i}^1 + t_{3,i}^1) * d \geq t_1^1, \tag{4.1}$$

Розглянемо час t_1^2 - час, протягом якого функціонувала самоорганізована розподілена система без втручання користувача чи системного адміністратора. Тоді, час t_1^3 - час, протягом якого функціонувала самоорганізована розподілена система з втручанням користувача чи системного адміністратора на вимогу системи. Час, коли функціонувала система і було втручання користувача чи системного адміністратора не розглядатимемо і не враховуватимемо в цьому випадку, оскільки він відноситься до часу, протягом якого досліджуватимуться аномалії, що пов'язано зі специфікою системи. Взагалі досліджувані часові

проміжки, позначені як t_1^1, t_1^2, t_1^3, $t_{2,i}^1$, $t_{3,i}^1$ можна віднести до зовнішніх характеристик системи, а наприклад час, коли було втручання користувача чи системного адміністратора без запиту на таку дію системи внутрішньою характеристикою системи. Задані часові величини, пов'язані співвідношенням:

$$t_1^2 + t_1^3 = t_1^1. \tag{4.2}$$

Якщо розглядати часову величину t_1^3 як таку, що враховує час протягом якого система або її компонента обслуговувалась користувачем або системним адміністратором і при цьому була повністю неактивною, позначимо який величиною $t_1^{3,1}$, та час протягом якого система продовжувала працювати, а обслуговувались користувачем або системним адміністратором на її запит компоненти, який позначимо через величину $t_1^{3,2}$. У випадку з визначенням величини $t_1^{3,2}$ наявність працюючого компоненту з центром прийняття рішень вищого рівня ієрархії в системі є обов'язковою. Розділимо величину $t_1^{3,2}$ на різні часові величини, що характеризуватимуть часові проміжки в різних випадках: $t_1^{3,2,1}$ – час роботи системи або компоненти з центром прийняття рішень вищого рівня ієрархії в системі, коли користувач або системний адміністратор обслуговують за запитом системи частину її компонент; $t_1^{3,2,2}$ – час обслуговування компоненти, яка не містить центру прийняття рішень вищого рівня ієрархії в системі, користувачем або системним адміністратором.

Використовуючи введені числові характеристики системи, визначимо величину r_1^1, яка характеризуватиме частку часу, що витрачається користувачем або системним адміністратором на обслуговування компоненти, в якій не міститься центр прийняття рішень вищого рівня ієрархії в системі, за формулою:

$$r_1^1 = \frac{t_1^{3,2,2}}{t_1^{3,2,1} - t_1^{3,2,2}}. \tag{4.3}$$

Величина r_1^1 характеризує випадок, коли частина системи з центром прийняття рішень вищого рівня ієрархії працює і при цьому частина системи обслуговується системним адміністратором чи користувачем, але вважається що витрачений час на обслуговування впливає на систему і, тому, розглядається час повного самостійного функціонування системи. Якщо ж його не враховувати в час самостійного функціонування системи, тоді введемо відповідну величину r_1^2 і обчислимо її за формулою:

$$r_1^2 = \frac{t_1^{3,2,2}}{t_1^{3,2,1}}. \tag{4.4}$$

Введемо критерій для оцінки ефективності роботи згідно коефіцієнту K_1, виходячи з її можливості самостійно приймати рішення без залучення користувача чи системного адміністратора. Величини r_1^1 та r_1^2 будуть збіжними до нуля, якщо мінімізуватиметься час $t_1^{3,2,2}$, але швидкість збіжності дуже залежати від величини $t_1^{3,2,1}$, тому враховуючи, що вони є близькими за значеннями між ними врахуємо їх в остаточних представленнях для визначення коефіцієнту K_1 критерію для оцінки ефективності роботи так:

$$K_1 = \frac{r_1^1 + r_1^2}{2}. \tag{4.5}$$

Значення коефіцієнту K_1, обчислене за формулою (4.5) буде усередненим і точніше описуватиме задану часову величину як характеристику в системі. Критерій для оцінки ефективності роботи системи задамо так:

$$K_1 = f\left(t_1^{3,2,1}, t_1^{3,2,2}\right) = \frac{\frac{t_1^{3,2,2}}{t_1^{3,2,1} - t_1^{3,2,2}} + \frac{t_1^{3,2,2}}{t_1^{3,2,1}}}{2} = 2 \cdot \frac{t_1^{3,2,1} \cdot t_1^{3,2,2} - t_1^{3,2,2} \cdot t_1^{3,2,2}}{t_1^{3,2,1} \cdot (t_1^{3,2,1} - t_1^{3,2,2})},$$

$$K_1 \to 0, \text{при} \min\left(t_1^{3,2,2}\right), \max\left(t_1^{3,2,1}\right). \tag{4.6}$$

Отже, визначення коефіцієнту K_1 за формулою (4.6) надає змогу оцінити ефективність функціонування системи. Наприклад, якщо $t_1^{3,2,1} \leq t_1^{3,2,2}$, тоді очевидно що $K_1 > 1$ і функціонування системи протягом досліджуваного було неефективним.

Отримані результати для критерію ефективності згідно коефіцієнту K_1 стосуються випадку для загального часу функціонування системи і можуть бути уточнені з врахуванням кожної компоненти системи. Тобто, для кожної компоненти системи здійснимо розрахунок часу її функціонування в різних випадках та представимо в критерії оцінки ефективності функціонування системи згідно характеристик в кожній з компонент. Компонента самоорганізованої розподіленої системи в комп'ютерній станції в мережі при здійснення аналізу її функціонування за часовим показником може бути віднесена до одного з випадків: функціонуюча одночасно з компонентою, в якій знаходиться центр прийняття рішень вищого рівня ієрархії; функціонуюча сумісно з іншими компонентами системи без компоненти, в якій знаходиться центр прийняття рішень вищого рівня ієрархії; нефункціонуюча і перебуває в комп'ютерній системі, яка вимкнена; нефункціонуюча, бо перебуває на обробці в користувача чи системного адміністратора. Враховуючи, що компоненти системи створюють паралельно виконуючі процеси в різних вузлах в мережі і проміжки часу, в яких вони активні, є різними, то дослідження функціонування системи за часовим показником необхідно проводити протягом всього часу функціонування системи в цілому або за усередненими даними протягом доби, оскільки активне функціонування системи є періодичним.

Введемо для кожної j – тої компоненти системи такі величини, що характеризуватимуть її функціонування за часовими показниками, причому $j = 1, 2, \dots, N$, N – кількість компонент системи. Для випадку $j = 0$, тобто компоненти, в якій міститься центр прийняття рішень вищого рівня ієрархії, визначення часу за її різними станами здійснимо окремо. Нехай $t_{1,j}^1$ – час, протягом якого

функціонувала j – тої компонента самоорганізованої розподіленої системи. Тобто, це увесь час витрачений на роботу j – тої компоненти системи, який включає функціонування в таких випадках: в автономному режимі за присутності компоненти з центром прийняття рішень вищого рівня ієрархії; в період відсутності центру; функціонування компоненти системи у вимкненому стані, тобто час в пасивному стані між періодами функціонування системи; часовий проміжок, коли компонента не функціонувала, а обслуговувалась користувачем чи системним адміністратором. Таким чином, вважатимемо ці чотири часові проміжки такими, що описують час повного циклу експлуатації j – тої компоненти системи. Тоді, введемо $t_{2,i,j}^{1}$ в якості часу функціонування j – тої компоненти системи в активному стані за присутності компоненти з центром прийняття рішень вищого рівня ієрархії протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи. Для характеристики періоду відсутності центру, але функціонування компоненти системи введемо величину $t_{3,i,j}^{1}$ в якості часу функціонування j – тої компоненти системи в активному стані за відсутності компоненти з центром прийняття рішень вищого рівня ієрархії протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи. Функціонування j – тої компоненти системи у вимкненому стані, тобто час в пасивному стані між періодами функціонування системи для означення характеристики задамо величиною $t_{4,i,j}^{1}$. Часовий проміжок, коли j – та компонента не функціонувала, а обслуговувалась користувачем чи системним адміністратором задамо величиною $t_{5,i,j}^{1}$, в якій ця характеристика визначена протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи.

Аналогічно введемо часові характеристики для компоненти з центром прийняття рішень вищого рівня ієрархії, тобто коли $j = 0$. Введемо величину $t_{2,i,0}^{1}$ в якості часу функціонування компоненти з центром прийняття рішень вищого рівня ієрархії протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи. Для характеристики періоду відсутності компоненти з центром прийняття рішень вищого рівня ієрархії через її недоступність решті

компонент системи з причини визначення виконання функцій для визначення подальших дій чи кроків системи введемо величину $t_{3,i,0}^{1}$ протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи. Функціонування компоненти системи з центром прийняття рішень вищого рівня ієрархії у вимкненому стані, тобто час в пасивному стані між періодами функціонування системи для означення характеристики задамо величиною $t_{4,i,0}^{1}$. Часовий проміжок, коли компонента системи з центром прийняття рішень вищого рівня ієрархії не функціонувала, а обслуговувалась користувачем чи системним адміністратором задамо величиною $t_{5,i,0}^{1}$, в якій ця характеристика визначена протягом i – того дня, де $i = 1, 2, \dots, d$, d – кількість днів функціонування системи.

Встановимо таке співвідношення між часовими характеристиками компонентів системи:

$$\frac{\sum_{i=1}^{d}\sum_{j=0}^{N}\sum_{k=2}^{5} t_{k,i,j}^{1}}{N} = t_{1,j}^{1}. \tag{4.7}$$

Всім компонентам системи задано часові характеристики таким чином, що вони є частинами системи на увесь час функціонування системи і це задано формулою (4.7). Але в процесі тривалого функціонування системи деякі з її компонентів можуть бути вилучені з неї і для цього випадку справедливі такі співвідношення:

$$\sum_{j=0}^{N} t_{1,j}^{1} = \sum_{i=1}^{d}\sum_{j=0}^{N}\sum_{k=2}^{5} t_{k,i,j}^{1}, \tag{4.8}$$

$$t_{1}^{1,\mathrm{c}} = \frac{\sum_{i=1}^{d}\sum_{j=0}^{N}\sum_{k=2}^{5} t_{k,i,j}^{1}}{N} \geq t_{1,j}^{1}, \tag{4.9}$$

де $t_{1}^{1,\mathrm{c}}$ – середньоарифметичне значення часу компоненти в системи протягом всього періоду її функціонування.

Визначимо коефіцієнт K_2 для критерію оцінки ефективності функціонування системи, виходячи із введених характеристик та отриманих співвідношень, заданих

формулами (4.7)-(4.9), для врахування особливостей характеристик, які залежать від компонент системи.

Використовуючи введені числові характеристики компонентів системи, визначимо величину $r_{1,j}^{1}$, яка характеризуватиме частку часу, що витрачається користувачем або системним адміністратором на обслуговування j-тої компоненти за формулою:

$$r_{1,j}^{1} = \frac{t_{1,j}^{3,2,2}}{t_{1,j}^{3,2,1} - t_{1,j}^{3,2,2}}. \tag{4.10}$$

де $t_{1,j}^{3,2,1}$ – час роботи j-тої компоненти з центром прийняття рішень вищого рівня ієрархії в системі, коли користувач або системний адміністратор обслуговують за запитом системи частину її компонент, включаючи j-ту компоненту; $t_{1,j}^{3,2,2}$ – час обслуговування j-тої компоненти, яка не містить центру прийняття рішень вищого рівня ієрархії в системі, користувачем або системним адміністратором.

Величина $r_{1,j}^{1}$ характеризує випадок, коли частина системи з центром прийняття рішень вищого рівня ієрархії працює і при цьому частина системи обслуговується системним адміністратором чи користувачем, але вважається що витрачений час на обслуговування впливає на систему і, тому, розглядається час повного самостійного функціонування системи. Якщо ж його не враховувати в час самостійного функціонування системи, тоді введемо відповідну величину $r_{1,j}^{2}$ для j-тої компоненти і обчислимо її за формулою:

$$r_{1,j}^{2} = \frac{t_{1,j}^{3,2,2}}{t_{1,j}^{3,2,1}}. \tag{4.11}$$

Введемо критерій для оцінки ефективності роботи для j-тої компоненти згідно коефіцієнту $K_{2,j}$, виходячи з можливості частини системи самостійно

приймати рішення без залучення користувача чи системного адміністратора. Величини $r_{1,j}^{1}$ та $r_{1,j}^{2}$ будуть збіжними до нуля, якщо мінімізуватиметься час $t_{1,j}^{3,2,2}$, але швидкість збіжності дуже залежати від величини $t_{1,j}^{3,2,1}$, тому враховуючи, що вони є близькими за значеннями між ними врахуємо їх в остаточних представленнях для визначення коефіцієнту $K_{2,j}$ критерію для оцінки ефективності роботи j-тої компоненти так:

$$K_{2,j} = \frac{r_{1,j}^{1} + r_{1,j}^{2}}{2}. \tag{4.12}$$

Значення коефіцієнту $K_{2,j}$, обчислене за формулою (4.12) буде усередненим і точніше описуватиме задану часову величину як характеристику j-тої компоненти в системі. Критерій для оцінки ефективності роботи j-тої компоненти системи задамо так:

$$K_{2,j} \to 0, \text{при} \min\left(t_{1,j}^{3,2,2}\right), \max\left(t_{1,j}^{3,2,1}\right). \tag{4.13}$$

Отже, визначення коефіцієнту $K_{2,j}$ за формулою (4.13) надає змогу оцінити ефективність функціонування j-тої компоненти системи. Загальний коефіцієнт ефективності системи з врахування часових характеристик компонентів системи і значень коефіцієнтів, які отримані із формули (4.13), знаходимо за формулою:

$$K_2 = \max\left(K_{2,0}, K_{2,1}, \dots, K_{2,N}\right). \tag{4.14}$$

Критерій ефективності роботи системи, який задаватиметься з використанням значення коефіцієнта K_2, аналогічно до випадку з K_1, буде відображатиме найкращий результат при мінімізації його значення. Таким чином, значення коефіцієнтів K_1 та K_2 будуть збігатись до нуля, за умови відсутності тривалих збоїв в роботі системи і її компонентів, а також за умови відсутності

руйнуючих впливів зловмисного програмного забезпечення та комп'ютерних атак на вузли в мережі, в яких розміщені компоненти системи, і нетривалих обслуговуючих робіт зі сторони користувача чи системного адміністратора.

Крім того, отримані значення коефіцієнтів, які розраховані за певний проміжок часу враховуватимемо при визначенні подальших дій системи.

4.2 Самоорганізована розподілена система виявлення аномалії в комп'ютерних системах

4.2.1 Реалізація самоорганізованої розподіленої системи

Самоорганізовану розподілену систему виявлення аномалії в комп'ютерних системах згідно розробленої її архітектури, методу підтримки цілісності та методу виявлення аномалій реалізуємо проміжним програмним забезпечення, що об'єнуватиме в одне ціле комп'ютерні станції в мережі, з двома типами інтерфейсу: для компоненти, в якій міститься центр прийняття рішень верхнього рівня ієрархії: для компонент, в яких міститься центр прийняття рішень нижнього рівня ієрархії.

Опис основних функцій розробленого програмного забезпечення подано в зведеній таблиці 4.1.

Таблиця 4.1 – Опис основних функцій розробленого ПЗ

№ з/п	Назва функції	Опис функції
1	2	3
1	SendCommandToComponents(int commandId)	Функція здійснює передачу команди з центру верхнього рівня, який приймаємо таким позначенням як $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$, до центрів системи в компонентах, які знаходяться на нижньому рівні, тобто до центрів з позначенням $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$, де $i = 1, 2, \dots, N$.

Продовження таблиці 4.1 – Опис основних функцій розробленого ПЗ

1	2	3
2	SendDetectedPotentialAnomaliesToMainCenter()	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру верхнього рівня $c_{SDS,0}$ для здійснення їх обробки.
3	SendAnomalyProcessingResultToMainCenter(int anomalyId)	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті до центру верхнього рівня $c_{SDS,0}$.
4	SendDetectedPotentialAnomaliesToCurrentCenter()	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, N$.
5	SendAnomalyProcessingResultToCurrentCenter(int anomalyId)	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, N$.

Продовження таблиці 4.1 – Опис основних функцій розробленого ПЗ

1	2	3
6	SendDetectedPotentialAnomaliesToCurrentCenterIfMainCenterAbsent()	Функція здійснює передачу повідомлення (при наявності відомостей про відсутність компоненти в системі з центром верхнього рівня, тобто фактичного центру прийняття рішень в системі) з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i$, $i = 1, 2, \dots, N, j = 1, 2, \dots, N$.
7	SendAnomalyProcessingResultToCurrentCenterIfMainCenterAbsent(int anomalyId)	Функція здійснює передачу повідомлення (при наявності відомостей про відсутність компоненти в системі з центром верхнього рівня, тобто фактичного центру прийняття рішень в системі) з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру цього ж рівня $c_{SDS,j}$ для здійснення їх обробки за умови $j \neq i, i = 1, 2, \dots, N, j = 1, 2, \dots, N$.
8	SendDetectedPotentialAnomaliesToComponent(int componentId, int anomalyId)	Функція здійснює передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру нижнього рівня $c_{SDS,j}$ для здійснення їх обробки при цьому $j = 1, 2, \dots, N$.

Продовження таблиці 4.1 – Опис основних функцій розробленого ПЗ

1	2	3
9	SendAnomalyProcessingResultToComponent(int componentId, int anomalyId)	Функція здійснює передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті, до центру нижнього рівня $c_{SDS,j}$ для здійснення їх обробки при цьому $j = 1, 2, \dots, N$.
10	SendWorkReadinessMessageToMainCenter()	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$ і де $i = 1, 2, \dots, N$ до центру верхнього рівня $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення компоненти системи.
11	SendUpdatedArchitectureMessageToComponent(int componentId)	Функція здійснює передачу повідомлення з центру верхнього рівня, який позначено $c_{SDS,0}$ до центру нижнього рівня $c_{SDS,i}$ і де $i = 1, 2, \dots, N$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи, в якій функціонує центр системи.

Продовження таблиці 4.1 – Опис основних функцій розробленого ПЗ

1	2	3
12	SendShutDownMessageToMainCenter()	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$ і де $i = 1, 2, \dots, N$ до центру верхнього рівня $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ з метою повідомлення про коректне вимкнення комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі.
13	SendUpdatedArchitectureMessageToActiveComponents()	Функція здійснює передачу повідомлення з центру верхнього рівня $c_{SDS,0}$ до центру нижнього рівня $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$, $c_{SDS,0} \in M_{SDS,0}$ для $i = 1, 2, \dots j-1, j+1, \dots, N$, $j \neq i$ з метою повідомлення про вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j -тої компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі.
14	SendShutDownComponentMessageToActiveComponents(int componentId)	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,j}$ до решти активних центрів нижнього рівня $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ для $i = 1, 2, \dots j-1, j+1, \dots, N$, $j \neq i$ з метою повідомлення про коректне вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j-тої компоненти системи із збереженнях характерних ознак профілю комп'ютерної станції в мережі.

Кінець таблиці 4.1 – Опис основних функцій розробленого ПЗ

1	2	3
15	SendBlockCommandToComponent(int componentId)	Функція здійснює передачу повідомлення з центру верхнього рівня $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$ до центру нижнього рівня $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ для $i = 1, 2, \dots j-1, j+1, \dots, N$, $j \neq i$ з метою повідомлення про проблеми в j - тій комп'ютерній станції в мережі і надсилання їй команди з вимогою блокування роботи програмного забезпечення в ній і примусового завершення роботи програмного забезпечення j -тої компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі, тобто повідомлення всім решті активним компонентам системи про зміну наявної архітектури системи пов'язаної з вилученням j -тої компоненти системи.
16	SeparateCreatedArchitecture()	Функція здійснює передачу повідомлення від центрів всіх рівнів до решти центрів всіх рівнів $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ для $i = 0, 1, 2, \dots j-1, j+1, \dots, N$, $j \neq i$.

Зображення віконної форми інтерфейсу компоненти розподіленої системи, в якій міститься центр верхнього рівня ієрархії, представлено на рис. 4.1.

Таким чином, розроблене програмне забезпечення проміжного рівня підтверджує можливість реалізації запропонованих рішень.

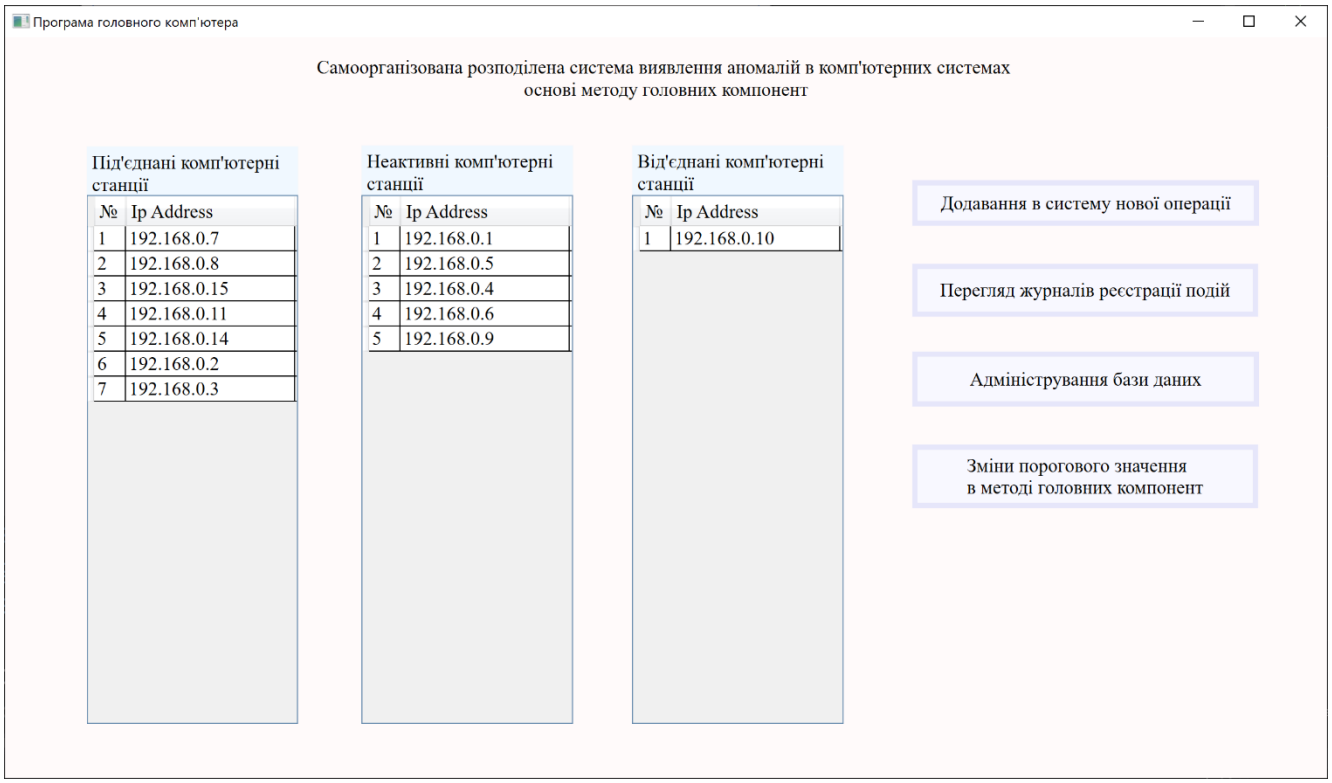


Рисунок 4.1 – Віконна форма інтерфейсу розробленої розподіленої системи

4.2.2 Експериментальні дослідження з використання самоорганізованої розподіленої системи

Метою експериментальних досліджень є дослідження ефективності функціонування самоорганізованої розподіленої системи та достовірність виявлення аномалії в комп’ютерних системах. Дослідження ефективності функціонування самоорганізованої розподіленої системи є необхідним для встановлення виконання нею функцій з підтримки цілісності та координації роботи компонентів системи. Достовірність виявлення аномалії в комп’ютерних системах потребує дослідження для встановлення можливості її використання в реальних умовах. Частина розподіленої системи, яка відповідає за виявлення аномалії, імплементована, як відповідний метод в неї, і її тестування дозволить оцінити достовірність з виявлення аномалії в комп’ютерних системах.

Тому, розглянемо тестування розподіленої системи спочатку з відімкненим модулем, що відповідає за виявлення аномалії. Показники, які досліджуватимемо

згрупуємо за такими характеристиками: час комунікації між окремими компонентами; час комунікації між компонентами системи і компонентою, в якій розміщено центр прийняття рішень вищого рівня ієрархії; час комунікації між компонентами в залежності від кількості активних компонентів, які формують систему; час, який витрачено на роботу, коли в системі здійснено розподілення центру, тобто тестування з розподіленим центром, і час, який витрачений, коли центр не розподіляється між рівнями ієрархії, а знаходиться повністю в одній компоненті.

Експериментальні дослідження з розробленою самоорганізованою розподіленою системою проведено в локальній комп'ютерній мережі, створеній за технологією Ethernet з швидкістю передачі даних 1 Гб/с між вузлами в мережі. Оскільки повідомлення для передачі між компонентами системи містять дуже невелику кількість інформації, то вони формуються в короткі пакети і вважатимемо, що кожне з них передавалось одним пакетом обсягом 64 байти.

Досліджувана реалізована самоорганізована розподілена система містила вісім компонентів, в одній з яких знаходився центр прийняття рішень вищого рівня ієрархії. Експериментальні дослідження проводились протягом 50 діб окремо для випадку, коли центр прийняття рішень був розподілений між всіма компонентами з врахуванням двох рівнів ієрархії і так само 50 діб для випадку, коли центр прийняття рішень був розміщений тільки в одній компоненті. На протязі всього часу експерименту було здійснено фіксування часу відправки повідомлень, їх номеру та фіксування часу отримання. Це здійснювалось для того, щоб провести дослідження витрат часу на комунікацію в середині самої системи. Результати проведених експериментальних досліджень представлені в табл. 4.1.

Результати експериментальних досліджень, представлені в табл. 4.1 підтверджують збільшення кількості переданих повідомлень для випадку, коли час витрачено на роботу системи з розподіленням центру, тобто тестування з розподіленим центром, порівняно для випадку, коли час витрачено на роботу системи без розподілення центру між компонентами. Крім того, час обробки переданих пакетів з повідомленнями зростає, бо для випадку, коли час витрачено

Таблиця 4.1 – Результати експериментальних досліджень щодо ефективності функціонування самоорганізованої розподіленої системи

№ / з. п.	Характеристика часової величини	Інтервал часу для випадку, коли час витрачено на роботу системи з розподіленням центру, тобто тестування з розподіленим центром, с	Інтервал часу для випадку, коли час витрачено на роботу системи без розподілення центру між компонентами, с
1	Час комунікації між окремими компонентами	1,42 – 2,61	1,41 – 2,56
2	Час комунікації між компонентами системи і компонентою, в якій розміщено центр прийняття рішень вищого рівня ієрархії	1,81 – 2,87	1,67 – 2,23
3	Час комунікації між компонентами в залежності від кількості активних компонентів, які формують систему. Випадки 3.1. Кількість компонент 2-4. 3.2. Кількість компонент 5-8.	 1,41 – 2,51 1,83 – 2,81	 1,27 – 2,39 1,72 – 2,43
4	Кількість переміщених пакетів для досліджуваної події 1	7546	5788
5	Кількість переміщених пакетів для досліджуваної події 2	12458	8386

на роботу системи з розподіленням центру, повідомлення від компонент системи паралельно в один час надходять до компоненти, в якій розміщено центр прийняття рішень і розсилаються всім компонентам, на відміну від випадку, коли центр прийняття рішень, який представлений тільки в одній компоненті самостійно приймає рішення про комунікацію з рештою компонент, що зменшує суттєво кількість повідомлень між компонентами. Але при цьому часові витрати на передачу повідомлень є суттєво невеликими в обох випадках, тому використання архітектури з розподіленням центру між рівнями ієрархії в двох різних типах компонент системи є єфективним, що підтверджено результатами експериментів.

Наступним етапом в обробці експериментальних досліджень є визначення співвіднесення часу обробки і функціонування самоорганізованої розподіленої системи згідно введених коефіцієнтів її ефективності роботи K_1 і K_2. Оскільки часові характеристики процесів, які відбувались в досліджуваній самоорганізованій розподіленій системі, зафіксовані, крім того, що вони використовувались самою системою для визначення подальших дій, то використаємо їх для побудови графіків функцій згідно коефіцієнтів її ефективності роботи K_1 і K_2. Для проведення експерименту використовувався варіант системи з розподіленням центру між двома рівнями ієрархії. Спочатку аналізувався випадок, коли втручань користувача чи системного адміністратора не здійснювалось, тоді час витрачений на обробку дорівнює нулеві і, дійсно, в системі не було витрат, пов’язаних з обслуговуванням частини компонентів. Другий випадок в проведеному експерименті передбачав і був реалізований, коли час обслуговування компонентів системи користувачем або системним адміністратором був суттєво меншим часу роботи всієї системи і коли був більшим. Результати таких експериментів представлені на рис. 4.2. Вони стосуються двох параметрів, які характеризують систему. Цими параметрами є час роботи компоненти системи у вузлі і час, який витрачено на обробку події в комп’ютерній станції користувачем або системним адміністратором. Визначення таких часових параметрів дозволить здійснити обчислення коефіцієнтів K_1 і K_2. На графіках часові інтервали зображено часовими діаграмами. З восьми компонент тільки три (4,5,7) потребували втручань

користувача чи системного адміністратора. Решта не потребували. Останній дев’ятий графік відображає зведену часову діаграми роботи всієї системи.

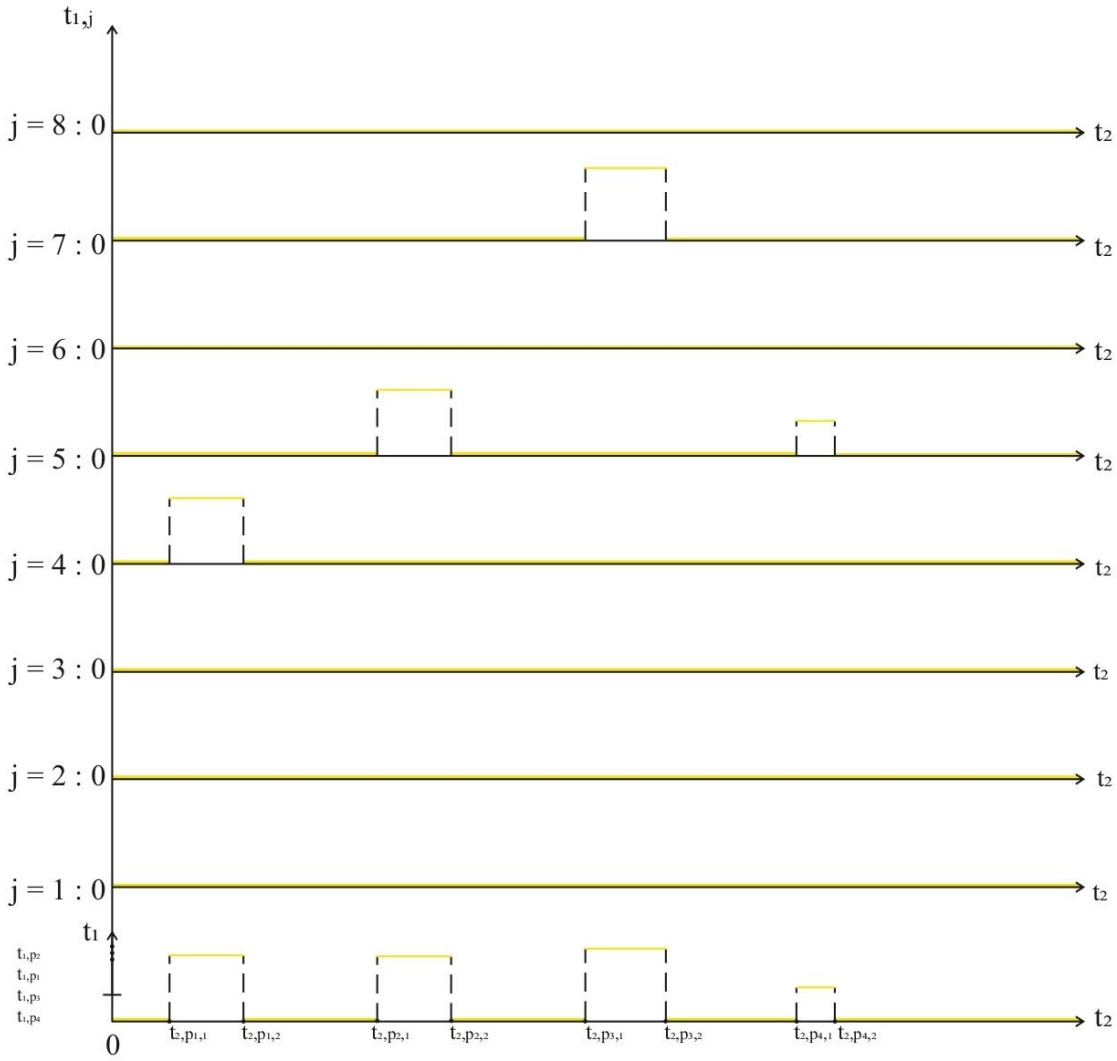


Рисунок 4.2 – Часові діаграми за результатами проведених експериментальних досліджень

Значення коефіцієнтів K_1 і K_2 після проведених експериментів представлено в табл. 4.2.

Отримані коефіцієнти підтверджують ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп’ютерній мережі. У другому випадку не великі значення коефіцієнтів обгрунтовуються тим, що тривалий час обслуговувались лише окремі компоненти системи, а більшість компонент систем функціонувала. Це доводить ефективність використання

розподілених систем такого типу.

Таблиця 4.2 – Значення коефіцієнтів K_1 і K_2

№ з./п.	Час обслуговування компонентів системи користувачем або системним адміністратором був суттєво меншим часу роботи всієї системи	Час обслуговування компонентів системи користувачем або системним адміністратором був суттєво більшим часу роботи всієї системи
	Випадок 1	Випадок 2
K_1	0,00987567	0,04873418
K_2	0,00986972	0,04873326

Також, за результатами проведеного експерименту було встановлено, що коли центр прийняття рішень розподілений між рівнями ієрархії, то ефективність за часом краща, бо обробка на нижньому рівні ієрархії скорочує час обробки події порівняно з використанням одного центру. Достовірність при обробці аномалії в комп'ютерних системах, які вводились штучно, становить 0,8356, що є задовільним результатом і підтверджує достатню ефективність запропонованих рішень.

4.3 Висновки до четвертого розділу

Для оцінки ефективності запропонованих рішень розроблено методику розрахунку ефективності використання самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах. В ній використано два введені критерії, результати застосування яких визначаються через відповідні коефіцієнти. Крім того, отримані значення коефіцієнтів, які розраховані за певний проміжок часу враховуються при визначенні подальших дій системи.

Розроблене програмне забезпечення для забезпечення функціонування

самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах підтверджує можливість реалізації запропонованих рішень.

Проведені експериментальні дослідження з розробленою реалізацією самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах згідно отриманих коефіцієнтів підтверджують ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп’ютерній мережі.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено самоорганізовану розподілену систему виявлення аномалій в комп’ютерних системах згідно методу головних компонент для покращення ефективності виявлення зловмисного програмного забезпечення та комп’ютерних атак.

При цьому отримано такі основні результати:

1. Встановлено, що виявлення зловмисного програмного забезпечення та комп’ютерних атак у локальних комп’ютерних мережах згідно досліджених методів та засобів виявлення може бути реалізовано методами виявлення аномалії в комп’ютерних системах та створенням розподілених систем виявлення аномалії.

2. Удосконалено архітектуру самоорганізованої розподіленої системи, в якій на відміну від відомих рішень, удосконалено внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії. Результатом так спроєктованої архітектури самоорганізованої розподіленої системи є можливість нарощувати її функціонал за рахунок наповнення імплементованими методами з виявлення аномалій в комп’ютерних системах. Система спроєктована таким чином, що її компоненти можуть обмінюватись результатами обробки аномалій та виявленими їх джерелами.

3. Розроблений метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп’ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп’ютерних атак.

4. Удосконалення методу виявлення аномалії згідно методу головних компонент в комп’ютерних системах в мережі надало змогу застосовувати його не до однієї комп’ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп’ютерних системах в мережі. Його застосування надало змогу скоротити обсяг даних і відповідно прискорити їх обмін між компонентами системи.

5. Здійснена розробка методики оцінки ефективності запропонованих рішень для розробленої самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах. Розроблено програмне забезпечення для забезпечення функціонування самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах для підтвердження можливості реалізації запропонованих рішень. Проведені експериментальні дослідження з розробленою реалізацією самоорганізованої розподіленої системи виявлення аномалій в комп’ютерних системах згідно отриманих коефіцієнтів підтвердили ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп’ютерній мережі.

За темою дипломної роботи магістра опублікована одна стаття у фаховому науковому виданні, що входить до рекомендованого переліку МОН України, в якому можуть публікуватись результати наукових досліджень на здобуття наукових ступенів кандидата та доктора наук (категорія Б), а також в двох публікаціях матеріалів конференції, які індексуються в наукометричній базі Scopus.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Савенко О. С., Паюк В. П., Савенко Б. О., Каштальян А. С. Моделі незадокументованих закладок програмного забезпечення в локальних комп'ютерних мережах. Вимірювальна та обчислювальна техніка в технологічних процесах. 2019. №2, С. 84-90.

2. Stetsyuk M., Bedratyuk L., Savenko B., Stetsyuk V., Savenko O. Providing the Resilience and Survivability of Specialized Information Technology Across Corporate Computer Networks. CEUR-WS. 2020. Vol. 2623. P. 219-238.

3. Nicheporuk A., Paiuk V., Savenko B., Savenko O., Geidarova O. Detecting Software Malicious Implant Based on Anomalies Research on Local Area Networks. CEUR-WS. 2020. Vol. 2623. P. 194-207.

4. Wawryn, K., Widuliński P. Detection of anomalies in compiled computer program files inspired by immune mechanisms using a template method. Journal of Computer Virology and Hacking Techniques. 2020. https://doi.org/10.1007/s11416-020-00364-w

5. Zeng J., Tang W. (2015) Negative Selection Algorithm Based Unknown Malware Detection Model. In: Gong M., Linqiang P., Tao S., Tang K., Zhang X. (eds) Bio-Inspired Computing - Theories and Applications. BIC-TA 2015. Communications in Computer and Information Science, vol. 562. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-49014-3_53

6. Корченко А. О. Методи ідентифікації аномальних станів для систем виявлення вторгнень: автореф. дис. ... д-ра техн. наук: 05.13.21, Київ, 2019, 40 с.

7. Лукова-Чуйко Н. В. Методологічні основи забезпечення функціональної стійкості розподілених інформаційних систем до кібернетичних загроз: автореф. дис. ... д-ра техн. наук: 05.13.06, Київ, 2018, 40 с.

8. Virusbulletin [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.virusbulletin.com/testing/ (Viewed on April 21, 2020). – Title from the screen.

9. Av-test [Electronic resource]: [Web-site]. – Electronic data. – Mode of access:

https://www.av-test.org/en/statistics/malware/ (Viewed on April 21, 2020). – Title from the screen.

10. Mohiuddin Ahmed, Abdun Naser Mahmood, Jiankun Hu. A survey of network anomaly detection techniques. Journal of Network and Computer Applications Vol. 60, January 2016. P. 19-31.

11. Bernadette J. Stolz, Jared Tanner, Heather A. Harrington, Vidit Nanda Geometric anomaly detection in data. Proceedings of the National Academy of Sciences Aug 2020, 117 (33) 19664-19669; DOI: 10.1073/pnas.2001741117

12. Xiang Yu, Hui Lu, Xianfei Yang, Ying Chen, Haifeng Song, Jianhua Li, Wei Shi An adaptive method based on contextual anomaly detection in Internet of Things through wireless sensor networks. International Journal of Distributed Sensor Networks 2020. Vol. 16(5) DOI: 10.1177/1550147720920478

13. Goldstein M., Uchida S. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. 2016. PLOS ONE 11(4): e0152173. https://doi.org/10.1371/journal.pone.0152173

14. Hayes, M.A., Capretz, M.A. Contextual anomaly detection framework for big sensor data. Journal of Big Data 2**,** 2. 2015. https://doi.org/10.1186/s40537-014-0011-y

15. Liu, L., Hu, M.; Kang, C., Li, X. Unsupervised Anomaly Detection for Network *Data Streams in Industrial Control Systems.* Information *2020*, 11, *105.* https://doi.org/10.3390/info11020105

16. Xiaodan Xu, Huawen Liu, Minghai Yao. Recent Progress of Anomaly Detection. Complexity, vol. 2019, Article ID 2686378, 11 pages, 2019. https://doi.org/10.1155/2019/2686378

17. Jianwen Huang, Zhen Chai and Hailong Zhu. Detecting anomalies in data center physical infrastructures using statistical approaches. Journal of Physics: Conference Series, Volume 1176, Issue 2. Jianwen Huang *et al* 2019 *J. Phys.: Conf. Ser.* 1176 022056

18. Fisch, A., Grose, D., Eckley, I.A., Fearnhead, P., & Bardwell, L. (2020). anomaly: Detection of Anomalous Structure in Time Series Data. arXiv: Applications.arXiv:2010.09353

19. Lu, X., Wang, S., Kang, F., Liu, S., Li, H., Xu, X. and Cui, L. (2019). An

anomaly detection method to improve the intelligent level of smart articles based on multiple group correlation probability models. International Journal of Crowd Science, Vol. 3, № 3. P. 333-347. https://doi.org/10.1108/IJCS-09-2019-0024

20. Solarz A., Bilicki M., Gromadzki M., Pollo A. , Durkalec A., Wypych M. Automated novelty detection in the WISE survey with one-class support vector machines. Published online: 05 October 2017. DOI: 10.1051/0004-6361/201730968

21. Mukrimah Nawir, Amiza Amir, Naimah Yaakob, Ong Bi Lynn. Effective and efficient network anomaly detection system using machine learning algorithm. Bulletin of Electrical Engineering and InformaticsVol.8, No.1, March 2019, pp. 46~51ISSN: 2302-9285, DOI: 10.11591/eei.v8i1.1387

22. Anta A., Hadjistasi T., Nicolaou N. et al. Tractable low-delay atomic memory. Distrib. Comput. 34, 33–58 (2021). https://doi.org/10.1007/s00446-020-00379-y

23. Ouyang L., Huang Y., Wei H., Lu J. Achieving Probabilistic Atomicity With Well-Bounded Staleness and Low Read Latency in Distributed Datastores. in IEEE Transactions on Parallel and Distributed Systems, vol. 32, № 4, pp. 815-829, 1 April 2021, doi: 10.1109/TPDS.2020.3034328.

24. Lakshman A. and Malik P. Cassandra: A decentralized structured storage system, *SIGOPS Operating Syst. Rev.*, vol. 44, no. 2, pp. 35-40, Apr. 2010.

25. Ganesh, C., Patra, A. Optimal extension protocols for byzantine broadcast and agreement. Distrib. Comput. 34, 59–77 (2021). https://doi.org/10.1007/s00446-020-00384-1

26. Huang, Z., Radunovic, B., Vojnovic, M. et al. Communication complexity of approximate maximum matching in the message-passing model. Distrib. Comput. 33, 515–531 (2020). https://doi.org/10.1007/s00446-020-00371-6

27. Czumaj, A., Konrad, C. Detecting cliques in CONGEST networks. Distrib. Comput. 33, 533–543 (2020). https://doi.org/10.1007/s00446-019-00368-w

28. Abboud, A., Censor-Hillel, K., Khoury, S. et al. Fooling views: a new lower bound technique for distributed computations under congestion. Distrib. Comput. 33, 545–559 (2020). https://doi.org/10.1007/s00446-020-00373-4

29. Di Luna, G.A., Flocchini, P., Izumi, T. et al. Fault-tolerant simulation of

population protocols. Distrib. Comput. 33, 561–578 (2020). https://doi.org/10.1007/s00446-020-00377-0

30. Ellen, F., Gelashvili, R., Shavit, N. et al. A complexity-based classification for multiprocessor synchronization. Distrib. Comput. 33, 125–144 (2020). https://doi.org/10.1007/s00446-019-00361-3

31. Chatterjee, S., Pandurangan, G. & Robinson, P. The complexity of leader election in diameter-two networks. *Distrib. Comput.* **33,** 189–205 (2020). https://doi.org/10.1007/s00446-019-00354-2

32. Busch, C., Herlihy, M., Popovic, M. *et al*. Time-communication impossibility results for distributed transactional memory. *Distrib. Comput.* **31,** 471–487 (2018). https://doi.org/10.1007/s00446-017-0318-y

33. Boczkowski, L., Korman, A. & Natale, E. Minimizing message size in stochastic communication patterns: fast self-stabilizing protocols with 3 bits. *Distrib. Comput.* 32**,** 173–191 (2019). https://doi.org/10.1007/s00446-018-0330-x

34. Min B., Varadharajan V. (2014) Feature-Distributed Malware Attack: Risk and Defence. In: Kutyłowski M., Vaidya J. (eds) Computer Security - ESORICS 2014. ESORICS 2014. Lecture Notes in Computer Science, vol 8713. Springer, Cham. https://doi.org/10.1007/978-3-319-11212-1_26

35. Nath H. V., Mehtre B.M. (2014) Static Malware Analysis Using Machine Learning Methods. In: Martínez Pérez G., Thampi S.M., Ko R., Shu L. (eds) Recent Trends in Computer Networks and Distributed Systems Security. SNDS 2014. Communications in Computer and Information Science, vol 420. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-54525-2_39

36. Merayo, M. G., Hierons, R. M., Núñez, M. Passive testing with asynchronous communications and timestamps. *Distrib. Comput.* 31, 327–342 (2018). https://doi.org/10.1007/s00446-017-0308-0

37. Michail, O. Terminating distributed construction of shapes and patterns in a fair solution of automata. *Distrib. Comput.* **31,** 343–365 (2018). https://doi.org/10.1007/s00446-017-0309-z

38. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols

with a leader. Distrib. Comput. **21**, 183–199 (2008)

39. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) Middleware for Network Eccentric and Mobile Applications, pp. 97–120. Springer, Berlin (2009)

40. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations, and Advanced Topics, vol. 19. Wiley, New York (2004)

41. Chen, H.-L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. Nat. Comput. **13**, 517–534 (2014)

42. Censor-Hillel, K., Parter, M. & Schwartzman, G. Derandomizing local distributed algorithms under bandwidth restrictions. *Distrib. Comput.* **33,** 349–366 (2020). https://doi.org/10.1007/s00446-020-00376-1

43. Barenboim, L.: Deterministic (Δ+1)-coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In: PODC, pp. 345–354 (2015)

44. Barenboim, L., Elkin, M., Gavoille, C.: A fast network-decomposition algorithm and its applications to constant-time distributed computation. In: SIROCCO, pp. 209–223 (2015)

45. Benjamini, I., Gurel-Gurevich, O., Peled, R.: On k-wise independent distributions and Boolean functions. arXiv preprint arXiv:1201.3261 (2012)

46. Antoniadis, K., Blanchard, P., Guerraoui, R. *et al.* The entropy of a distributed computation random number generation from memory interleaving. *Distrib. Comput.* **31,** 389–417 (2018). https://doi.org/10.1007/s00446-017-0311-5

47. Alistarh, D., Sauerwald, T., Vojnovic, M.: Lock-free algorithms under stochastic schedulers. In: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21–23, 2015, pp. 251–260 (2015). doi:10.1145/2767386.2767430

48. Barker, E., Kelsley, J.: Recommendation for random bit generator (rbg) constructions. SP 800-90C (2012)

49. Zhou, H., Bruck, J.: Generalizing the Blum-Elias method for generating random bits from markov chains. In: Proceedings of IEEE International Symposium on Information Theory (ISIT) (2010)

50. Чженбин Ху, Мухин В. Е., Корнага, О. Ю., Герасименко Я. И. Управление ресурсами распределенной компьютерной системы с учетом уровня доверия к вычислительным компонентам. Кибернетика и системный анализ. № 2, т. 53. 2017. С. 168–180.

51. Corporate Endpoint Protection Products Group Test: Socially-Engineered Malware Q2 [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.nsslabs.com/ research/ endpoint-security/ anti-malware/q2-2010-endpoint-protection-product-group-test.html (Viewed on April 3, 2021). – Title from the screen.

52. ESET Endpoint Security [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.eset.com/ (Viewed on April 3, 2021). – Title from the screen.

53. Symantec Endpoint Protection [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.anti-malware.ru/reviews/ Symantec_Endpoint_Protection (Viewed on April 3, 2021). – Title from the screen.

54. Branitskiy A., Kotenko I. Hybridization of computational intelligence methods for attack detection in computer networks. Journal of Computational Science. 2017. №23. P. 145–156.

55. Savenko O. Interoperability of distributed multiple system for malware detection based on components levels of safety. Проблеми інформаційних технологій. 2018. № 24. С. 78-92.

56. Securelist. FAQ: Disabling the new Hlux / Kelihos Botnet [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://securelist.com/blog/research/32634/faq-disabling-the-new-hluxkelihos-botnet-13/ (Viewed on April 3, 2021). – Title from the screen.

57. COMSS1 [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.comss.ru/page.php?id=2758 (Viewed on April 3, 2021). – Title from the screen.

58. Enterprise End Point Protection Comparative Analysis - Socially Engineered Malware: Report Overview [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: www.nsslabs.com/reports (Viewed on April 3, 2021). – Title from the screen.

59. Bakotech [Electronic resource]: [Web-site]. – Electronic data. – Mode of access:

https://bakotech.ua/ (Viewed on April 3, 2021). – Title from the screen.

60. ClamAV [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.clamav.net/ (Viewed on April 3, 2021). – Title from the screen.

61. Malwarebytes Endpoint Security [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://ru.malware bytes. com/business/endpoint security (Viewed on April 3, 2021). – Title from the screen.

62. Kaspersky Lab [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.kaspersky.ru (Viewed on April 2, 2019). – Title from the screen.

63. Avast! [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.avast.com/index (Viewed on April 21, 2020). – Title from the screen.

64. AVG [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.avg.com (Viewed on April 21, 2020). – Title from the screen.

65. Avira [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.avira.com (Viewed on April 21, 2020). – Title from the screen.

66. Bitdefender [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.bitdefender.com/(Viewed on April 21, 2020). – Title from the screen.

67. Wireshark.org [Електронний ресурс]. – Режим доступу: http://www.wireshark.org/docs/dfref.

68. Stetsyuk M., Stetsyuk V., Savenko B., Savenko O., Dobrowolski M. Implementation of Control by Parameters of Client Automated Workplaces of Specialized Information Systems for Neutralization malware. CEUR-WS. 2021. Vol. 2853. P. 340-352. URL: http://ceur-ws.org/Vol-2853/paper40.pdf.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ САМООРГАНІЗОВАНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ

```
using DistributedSystem.LowerCenter.Database;
using DistributedSystem.LowerCenter.Database.Entities;
using DistributedSystem.LowerCenter.Database.Repository;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.Anomaly
{
   public class AnomalyService : IAnomalyService
   {
      private readonly IRepository<AnomalyEntity> _repository;

      public AnomalyService()
      {
         _repository = new Repository<AnomalyEntity>();
      }

      public async Task<List<long>> GetDetectedAnomaliesIdList()
      {
         return await _repository.Entities.Where(e => e.Status == AnomalyStatus.DETECTED)
                                  .Select(e => e.Id)
                                  .ToListAsync();
      }

      public async Task<List<long>> GetDetectedPotentialAnomaliesIdList()
      {
         return await _repository.Entities.Where(e => e.Status == AnomalyStatus.POTENTIAL)
                                  .Select(e => e.Id)
                                  .ToListAsync();
      }
   }
}
using System.Collections.Generic;
using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.Anomaly
{
   public interface IAnomalyService
   {
      Task<List<long>> GetDetectedAnomaliesIdList();
      Task<List<long>> GetDetectedPotentialAnomaliesIdList();
   }
```

```
}

using DistributedSystem.LowerCenter.Database.Entities;
using DistributedSystem.LowerCenter.Database.Repository;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.Command
{
   public class CommandService : ICommandService
   {
      private readonly IRepository<CommandEntity> _repository;

      public CommandService()
      {
         _repository = new Repository<CommandEntity>();
      }

      public async Task<int> GetCommandId(string commandName)
      {
         return await _repository.Entities.Where(e => e.Name == commandName).Select(e =>
e.Id).FirstOrDefaultAsync();
      }
   }
}

using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.Command
{
   public interface ICommandService
   {
      Task<int> GetCommandId(string commandName);
   }
}

using System;

namespace DistributedSystem.LowerCenter.Database.Entities
{
   public class AnomalyEntity
   {
      public long Id { get; set; }
      public string Name { get; set; }
      public string Description { get; set; }
      public AnomalyStatus Status { get; set; }
      public DateTime DetectedOn { get; set; }
   }
}

namespace DistributedSystem.LowerCenter.Database.Entities
```

```
{
   public class CommandEntity
   {
      public int Id { get; set; }
      public string Name { get; set; }
      public string Description { get; set; }
   }
}

using System;

namespace DistributedSystem.LowerCenter.Database.Entities
{
   public class InvestigationResultEntity
   {
      public int Id { get; set; }
      public string Conclusion { get; set; }
      public DateTime CreatedOn { get; set; }

      public long AnomalyFK { get; set; }
      public AnomalyEntity Anomaly { get; set; }

      public int CommandFK { get; set; }
      public CommandEntity Command { get; set; }
   }
}

using System;

namespace DistributedSystem.LowerCenter.Database.Entities
{
   public class LogActivityEntity
   {
      public long Id { get; set; }
      public string Name { get; set; }
      public string Description { get; set; }
      public string MachineIP { get; set; }
      public DateTime CreatedOn { get; set; }
   }
}

namespace DistributedSystem.LowerCenter.Database
{
   public enum AnomalyStatus
   {
      DETECTED = 1,
      POTENTIAL = 2,
      INVESTIGATED = 3,
      RESOLVED = 4
   }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.Database.Repository
{
   public interface IRepository<T> where T : class
   {
      IQueryable<T> Entities { get; }
      T GetById(int id);
      IEnumerable<T> GetAll();
      IEnumerable<T> Find(Expression<Func<T, bool>> expression);
      Task Add(T entity);
      Task AddRange(IEnumerable<T> entities);
      Task Update(T entity);
      Task Remove(T entity);
      Task RemoveRange(IEnumerable<T> entities);
   }
}

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.Database.Repository
{
   public class Repository<T> : IRepository<T> where T : class
   {
      protected readonly AppDbContext _dbContext;

      public Repository()
      {
         _dbContext = new AppDbContext();
      }

      public IQueryable<T> Entities => _dbContext.Set<T>();

      public virtual async Task Add(T entity)
      {
         await _dbContext.AddAsync(entity);
         await _dbContext.SaveChangesAsync();
      }

      public virtual async Task AddRange(IEnumerable<T> entities)
      {
         await _dbContext.AddRangeAsync(entities);
         await _dbContext.SaveChangesAsync();
```

```
        }

        public virtual async Task Update(T entity)
        {
            _dbContext.Update(entity);
            await _dbContext.SaveChangesAsync();
        }

        public virtual async Task<IEnumerable<T>> Find(Expression<Func<T, bool>>
expression)
        {
            return await _dbContext.Set<T>().Where(expression).ToListAsync();
        }

        public virtual async Task<IEnumerable<T>> GetAll()
        {
            return await _dbContext.Set<T>().ToListAsync();
        }

        public virtual async Task<T> GetById(int id)
        {
            return await _dbContext.Set<T>().FindAsync(id);
        }

        public virtual async Task Remove(T entity)
        {
            _dbContext.Remove(entity);
            await _dbContext.SaveChangesAsync();
        }

        public virtual async Task RemoveRange(IEnumerable<T> entities)
        {
            _dbContext.RemoveRange(entities);
            await _dbContext.SaveChangesAsync();
        }
    }
}

using DistributedSystem.LowerCenter.Database.Entities;
using Microsoft.EntityFrameworkCore;

namespace DistributedSystem.LowerCenter.Database
{
    public class AppDbContext : DbContext
    {
        public DbSet<AnomalyEntity> Anomalies { get; set; }
        public DbSet<CommandEntity> Commands { get; set; }
        public DbSet<InvestigationResultEntity> InvestigationResults { get; set; }
        public DbSet<LogActivityEntity> ActivityLogs { get; set; }

        public AppDbContext(DbContextOptions<AppDbContext> options)
            : base(options)
```

```
            {
                Database.Migrate();
            }

            public AppDbContext()
            {

            }
        }
    }

    using DistributedSystem.LowerCenter.Database.Entities;
    using System.Threading.Tasks;

    namespace DistributedSystem.LowerCenter.InvestigationResult
    {
        public interface IInvestigationResultService
        {
            Task<InvestigationResultEntity> GetInvestigationResultByAnomalyId(long anomalyId);
        }
    }

    using DistributedSystem.LowerCenter.Database.Entities;
    using DistributedSystem.LowerCenter.Database.Repository;
    using Microsoft.EntityFrameworkCore;
    using System.Linq;
    using System.Threading.Tasks;

    namespace DistributedSystem.LowerCenter.InvestigationResult
    {
        public class InvestigationResultService : IInvestigationResultService
        {
            private readonly IRepository<InvestigationResultEntity> _repository;

            public InvestigationResultService()
            {
                _repository = new Repository<InvestigationResultEntity>();
            }

            public async Task<InvestigationResultEntity> GetInvestigationResultByAnomalyId(long
anomalyId)
            {
                return await _repository.Entities.Where(e => e.Id ==
anomalyId).FirstOrDefaultAsync();
            }
        }
    }

    using DistributedSystem.LowerCenter.Database.Entities;
    using System.Threading.Tasks;

    namespace DistributedSystem.LowerCenter.LogActivity
```

```
{
    public interface ILogActivityService
    {
        Task Log(LogActivityEntity newRecord);
    }
}

using DistributedSystem.LowerCenter.Database.Entities;
using DistributedSystem.LowerCenter.Database.Repository;
using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.LogActivity
{
    public class LogActivityService : ILogActivityService
    {
        private readonly IRepository<LogActivityEntity> _repository;

        public LogActivityService()
        {
            _repository = new Repository<LogActivityEntity>();
        }

        public async Task Log(LogActivityEntity newRecord)
        {
            await _repository.Add(newRecord);
        }
    }
}

using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.TcpClientSender
{
    public interface ITcpClientSender
    {
        Task SendWorkReadinessMessageToMainCenter();
        Task SendDetectedPotentialAnomaliesToMainCenter();
        Task SendAnomalyProcessingResultToMainCenter(int anomalyId);
        Task SendShutDownMessageToMainCenter();
        Task SendAnomalyProcessingResultToCurrentCenterIfMainCenterAbsent(int
anomalyId);
        Task SendDetectedPotentialAnomaliesToCurrentCenter();
    }
}

using DistributedSystem.LowerCenter.Anomaly;
using DistributedSystem.LowerCenter.Command;
using DistributedSystem.LowerCenter.Database.Entities;
using DistributedSystem.LowerCenter.InvestigationResult;
using DistributedSystem.LowerCenter.LogActivity;
using System;
using System.Net;
```

```
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace DistributedSystem.LowerCenter.TcpClientSender
{
   public class TcpClientSender : ITcpClientSender
   {
      private readonly TcpClient _client;
      private readonly ILogActivityService _logActivityService;
      private readonly ICommandService _commandService;
      private readonly IAnomalyService _anomalyService;
      private readonly IInvestigationResultService _investigationResultService;

      private TcpClientSender()
      {
         _client = new TcpClient(Constants.MAIN_CENTER_IP_ADDRESS,
Constants.MAIN_CENTER_PORT);
         _logActivityService = new LogActivityService();
         _commandService = new CommandService();
         _anomalyService = new AnomalyService();
         _investigationResultService = new InvestigationResultService();
      }

      public async Task SendWorkReadinessMessageToMainCenter()
      {
         try
         {
            NetworkStream stream = _client.GetStream();

            while (true)
            {
               var commandId = await _commandService.GetCommandId("Readiness");

               byte[] data = Encoding.Unicode.GetBytes(commandId.ToString());
               stream.Write(data, 0, data.Length);
            }
         }
         catch (Exception ex)
         {
            string myHost = Dns.GetHostName();

            var exception = new LogActivityEntity()
            {
               Name = "Exception",
               MachineIP = (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
               Description = ex.Message,
               CreatedOn = DateTime.Now,
            };

            await _logActivityService.Log(exception);
```

```
            }
            finally
            {
                _client.Close();
            }
        }

        public async Task SendDetectedPotentialAnomaliesToMainCenter()
        {
            try
            {
                NetworkStream stream = _client.GetStream();

                while (true)
                {
                    var detectedAnomaliesList = await
_anomalyService.GetDetectedAnomaliesIdList();

                    byte[] data = Encoding.Unicode.GetBytes(string.Join(",",
detectedAnomaliesList));
                    stream.Write(data, 0, data.Length);
                }
            }
            catch (Exception ex)
            {
                string myHost = Dns.GetHostName();

                var exception = new LogActivityEntity()
                {
                    Name = "Exception",
                    MachineIP = (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                    Description = ex.Message,
                    CreatedOn = DateTime.Now,
                };

                await _logActivityService.Log(exception);
            }
            finally
            {
                _client.Close();
            }
        }

        public async Task SendAnomalyProcessingResultToMainCenter(int anomalyId)
        {
            try
            {
                NetworkStream stream = _client.GetStream();

                var investigationResult =
await_investigationResultService.GetInvestigationResultByAnomalyId(anomalyId);
```

```
                byte[] data = Encoding.Unicode.GetBytes(string.Join(",", investigationResult));
            while (true)
            {
                stream.Write(data, 0, data.Length);
            }
        }
        catch (Exception ex)
        {
            string myHost = Dns.GetHostName();

            var exception = new LogActivityEntity()
            {
                Name = "Exception",
                MachineIP = (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                Description = ex.Message,
                CreatedOn = DateTime.Now,
            };

            await _logActivityService.Log(exception);
        }
        finally
        {
            _client.Close();
        }
    }

    public async Task SendShutDownMessageToMainCenter()
    {
        try
        {
            NetworkStream stream = _client.GetStream();

            while (true)
            {
                var commandId = await
_commandService.GetCommandId("ShutDownLowerCenter");

                byte[] data = Encoding.Unicode.GetBytes(commandId.ToString());
                stream.Write(data, 0, data.Length);
            }
        }
        catch (Exception ex)
        {
            string myHost = Dns.GetHostName();

            var exception = new LogActivityEntity()
            {
                Name = "Exception",
                MachineIP = (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
```

```
                Description = ex.Message,
                CreatedOn = DateTime.Now,
            };

            await _logActivityService.Log(exception);
        }
        finally
        {
            _client.Close();
        }
    }

    public async Task SendAnomalyProcessingResultToCurrentCenterIfMainCenterAbsent(int anomalyId)
    {
        try
        {
            NetworkStream stream = _client.GetStream();

            while (true)
            {
                var investigationResult = await _investigationResultService.GetInvestigationResultByAnomalyId(anomalyId);

                byte[] data = Encoding.Unicode.GetBytes(string.Join(",", investigationResult));
                stream.Write(data, 0, data.Length);
            }
        }
        catch (Exception ex)
        {
            string myHost = Dns.GetHostName();

            var exception = new LogActivityEntity()
            {
                Name = "Exception",
                MachineIP = (await Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                Description = ex.Message,
                CreatedOn = DateTime.Now,
            };

            await _logActivityService.Log(exception);
        }
        finally
        {
            _client.Close();
        }
    }

    public async Task SendDetectedPotentialAnomaliesToCurrentCenter()
    {
        try
```

```
                {
                    NetworkStream stream = _client.GetStream();

                    while (true)
                    {
                        var detectedPotentialAnomaliesList = await
_anomalyService.GetDetectedPotentialAnomaliesIdList();

                        byte[] data = Encoding.Unicode.GetBytes(string.Join(",",
detectedPotentialAnomaliesList));
                        stream.Write(data, 0, data.Length);
                    }
                }
                catch (Exception ex)
                {
                    string myHost = Dns.GetHostName();

                    var exception = new LogActivityEntity()
                    {
                        Name = "Exception",
                        MachineIP = (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                        Description = ex.Message,
                        CreatedOn = DateTime.Now,
                    };

                    await _logActivityService.Log(exception);
                }
                finally
                {
                    _client.Close();
                }
            }
        }
    }

    // Main Center has code that extends code above:

    using System.Collections.Generic;
    using System.Threading.Tasks;

    namespace DistributedSystem.MainCenter.Host
    {
        public interface IHostService
        {
            Task<List<string>> GetLocalNetworkIPAddresses();
        }
    }

    using System.Collections.Generic;
    using System.Net;
    using System.Threading.Tasks;
```

```
namespace DistributedSystem.MainCenter.Host
{
   public class HostService : IHostService
   {
      public async Task<List<string>> GetLocalNetworkIPAddresses()
      {
         string myHost = Dns.GetHostName();
         IPHostEntry ipAddressesList = await Dns.GetHostEntryAsync(myHost);

         List<string> result = new();

         foreach (var ipAddress in ipAddressesList.AddressList)
         {
            result.Add(ipAddress.ToString());
         }

         return result;
      }
   }
}

using System.Threading.Tasks;

namespace DistributedSystem.MainCenter.TcpClientSender
{
   public interface ITcpClientSender
   {
      Task SendShutDownComponentMessageToActiveComponents();
      Task SeparateCreatedArchitecture();
      Task SendCommandToComponents(int commandId);
      Task SendDetectedPotentialAnomaliesToComponent(string componentIp, int
anomalyId);
      Task SendAnomalyProcessingResultToComponent(string componentIp, int anomalyId);
      Task SendBlockCommandToComponent(string componentIp);
   }
}

using DistributedSystem.LowerCenter.Anomaly;
using DistributedSystem.LowerCenter.Command;
using DistributedSystem.LowerCenter.Database.Entities;
using DistributedSystem.LowerCenter.InvestigationResult;
using DistributedSystem.LowerCenter.LogActivity;
using DistributedSystem.MainCenter.Host;
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace DistributedSystem.MainCenter.TcpClientSender
{
```

```
public class TcpClientSender : ITcpClientSender
{
   private readonly ILogActivityService _logActivityService;
   private readonly ICommandService _commandService;
   private readonly IAnomalyService _anomalyService;
   private readonly IInvestigationResultService _investigationResultService;
   private readonly IHostService _hostService;

   private TcpClientSender()
   {
      _logActivityService = new LogActivityService();
      _commandService = new CommandService();
      _anomalyService = new AnomalyService();
      _investigationResultService = new InvestigationResultService();
      _hostService = new HostService();
   }

   public async Task SendShutDownComponentMessageToActiveComponents()
   {
      var activeLowerCentersList = await _hostService.GetLocalNetworkIPAddresses();
      var commandId = await
_commandService.GetCommandId("ShutDownLowerCenter");

      foreach (var activeLowerCenter in activeLowerCentersList)
      {
         TcpClient client = new TcpClient(activeLowerCenter, 465);

         try
         {
            NetworkStream stream = client.GetStream();
      byte[] data = Encoding.Unicode.GetBytes(commandId.ToString());

            while (true)
            {
               stream.Write(data, 0, data.Length);
            }
         }
         catch (Exception ex)
         {
            string myHost = Dns.GetHostName();

            var exception = new LogActivityEntity()
            {
               Name = "Exception",
               MachineIP = (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
               Description = ex.Message,
               CreatedOn = DateTime.Now,
            };

            await _logActivityService.Log(exception);
         }
```

```
            finally
            {
                client.Close();
            }
        }
    }

    public async Task SendCommandToComponents(int commandId)
    {
        var activeLowerCentersList = await _hostService.GetLocalNetworkIPAddresses();

        foreach (var activeLowerCenter in activeLowerCentersList)
        {
            TcpClient client = new TcpClient(activeLowerCenter, 465);

            try
            {
                NetworkStream stream = client.GetStream();
    byte[] data = Encoding.Unicode.GetBytes(commandId.ToString());

                while (true)
                {
                    stream.Write(data, 0, data.Length);
                }
            }
            catch (Exception ex)
            {
                string myHost = Dns.GetHostName();

                var exception = new LogActivityEntity()
                {
                    Name = "Exception",
                    MachineIP                                =                                (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                    Description = ex.Message,
                    CreatedOn = DateTime.Now,
                };

                await _logActivityService.Log(exception);
            }
            finally
            {
                client.Close();
            }
        }
    }

    public async Task SendBlockCommandToComponent(string componentIp)
    {
        var commandId = await _commandService.GetCommandId("BlockLowerCenter");

        TcpClient client = new TcpClient(componentIp, 465);
```

```
                try
                {
                    NetworkStream stream = client.GetStream();
            byte[] data = Encoding.Unicode.GetBytes(commandId.ToString());

                    while (true)
                    {
                        stream.Write(data, 0, data.Length);
                    }
                }
                catch (Exception ex)
                {
                    string myHost = Dns.GetHostName();

                    var exception = new LogActivityEntity()
                    {
                        Name = "Exception",
                        MachineIP = (await
Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                        Description = ex.Message,
                        CreatedOn = DateTime.Now,
                    };

                    await _logActivityService.Log(exception);
                }
                finally
                {
                    client.Close();
                }
            }

            public async Task SendAnomalyProcessingResultToComponent(string componentIp, int
anomalyId)
            {
                TcpClient client = new TcpClient(componentIp, 465);

                try
                {
                    NetworkStream stream = client.GetStream();

                    var investigationResult = await
_investigationResultService.GetInvestigationResultByAnomalyId(anomalyId);

                    byte[] data = Encoding.Unicode.GetBytes(string.Join(",", investigationResult));

                    while (true)
                    {
                        stream.Write(data, 0, data.Length);
                    }
                }
                catch (Exception ex)
```

```
            {
                string myHost = Dns.GetHostName();

                var exception = new LogActivityEntity()
                {
                    Name = "Exception",
                    MachineIP = (await Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                    Description = ex.Message,
                    CreatedOn = DateTime.Now,
                };

                await _logActivityService.Log(exception);
            }
            finally
            {
                client.Close();
            }
        }

        public async Task SendDetectedPotentialAnomaliesToComponent(string componentIp, int anomalyId)
        {
            TcpClient client = new TcpClient(componentIp, 465);

            try
            {
                NetworkStream stream = client.GetStream();

                var detectedPotentialAnomaliesList = await _anomalyService.GetDetectedPotentialAnomaliesIdList();

                byte[] data = Encoding.Unicode.GetBytes(string.Join(",", detectedPotentialAnomaliesList));

                while (true)
                {
                    stream.Write(data, 0, data.Length);
                }
            }
            catch (Exception ex)
            {
                string myHost = Dns.GetHostName();

                var exception = new LogActivityEntity()
                {
                    Name = "Exception",
                    MachineIP = (await Dns.GetHostEntryAsync(myHost)).AddressList[0].ToString(),
                    Description = ex.Message,
                    CreatedOn = DateTime.Now,
                };
```

```
                await _logActivityService.Log(exception);
            }
            finally
            {
                client.Close();
            }
        }
    }
}
```

ДОДАТОК Б
СТАТТІ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

1. Савенко О. С., Паюк В. П., Савенко Б. О., Каштальян А. С. Моделі незадокументованих закладок програмного забезпечення в локальних комп’ютерних мережах. Вимірювальна та обчислювальна техніка в технологічних процесах. 2019. №2, С. 84-90.

Міжнародний науково-технічний журнал
«ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ»
ISSN 2219-9365

УДК 004.49
DOI: 10.31891/2219-9365-2019-64-14

САВЕНКО О. С., ПАЮК В. П., САВЕНКО Б. О., КАШТАЛЬЯН А. С.
Хмельницький національний університет

МОДЕЛІ НЕЗАДОКУМЕНТОВАНИХ ЗАКЛАДОК ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

В роботі здійснено постановку актуальної наукової задачі з виявлення в програмному забезпеченні незадокументованих закладок, які можуть бути самостійними об'єктами або частиною певного зловмисного програмного забезпечення. Місцем дослідження вибрано локальні комп'ютерні мережі.

Для незадокументованих закладок програмного забезпечення було проаналізовано види загроз, які можуть бути здійснені ними в локальній мережі, та здійснено їх формалізацію та деталізацію. Така формалізація представлена в моделі зловмисного програмного забезпечення в локальній комп'ютерній мережі частковим випадком. Це дозволило застосувати результати до незадокументованих закладок програмного забезпечення, які є частинами певного зловмисного програмного забезпечення, отриманої для цього цілком використання розподіленої багаторівневої системи виявлення. Для такого застосування були розроблені моделі незадокументованих закладок програмного забезпечення, які використовуються в локальних комп'ютерних мережах.

Моделі дозволили після відповідної формалізації включити їх в засоби виявлення. Застосування розроблених моделей незадокументованих закладок програмного забезпечення в розподіленій багаторівневій системі виявлення дало можливість покращити ефективність виявлення бот-мереж, складовими яких вони були. Підтвердженням результатів покращення виявлення був проведений протягом тривалого часу експеримент з виявлення бот-мереж, в складі кожної з яких були незадокументовані закладки програмного забезпечення.

Ключові слова: незадокументовані закладки, програмне забезпечення, модель, зловмисне програмне забезпечення, локальна комп'ютерна мережа

SAVENKO O., PAIUK V., SAVENKO B., KASHTALIAN A.
Khmelnytskyi National University

MODELS OF SECRET CODES OF SOFTWARE IN LOCAL COMPUTER NETWORKS

The paper deals with the actual scientific task of detecting secret bookmarks in the software, which may be separate objects or part of certain malicious software. Local computer networks are selected as the study site.

For secret software bookmarks, the types of threats that can be committed to the LAN were analyzed and formalized and detailed. This formalization is a partial case of malware on LANs. This allowed the results to be applied to secret software bookmarks that are part of certain malicious software obtained for it by using a distributed multilevel detection system. Models of secret bookmarking software for use on local area networks have been developed for this purpose.

The models were allowed, after appropriate formalization, to include them in detection tools. The use of developed models of secret software bookmarks in a distributed multilevel detection system made it possible to improve the detection efficiency of the botnets they were part of. Confirmation of the results of the improvements was carried out for a long time an experiment to identify botnets, each of which were secret software bookmarks.

Keywords: secret software bookmarks, software, model, malicious software, local computer network

Вступ. Постановка задачі. Розвиток інформаційних технологій в різних сферах продовжує супроводжуватись невпинним бажанням зловмисників отримати вигоду за рахунок недоліків в їх захисті. Найбільш актуальними для отримання вигоди з погляду зловмисників є організації та підприємства, в яких функціонують інформаційні технології. Відомо багато способів проникнення в локальні комп'ютерні мережі підприємств (організацій) з метою несанкціонованого доступу до інформації в них. Одним із способів доступу зловмисників до інформаційних ресурсів підприємств (організацій) є використання незадокументованих можливостей в програмному та апаратному забезпеченні персональних комп'ютерів і периферійному обладнанні, які дозволяють здійснювати прихований несанкціонований доступ до ресурсів системи, як правило, за допомогою локальної мережі. Основне призначення незадокументованих програмних закладок - забезпечити несанкціонований доступ до конфіденційної інформації.

Програмна закладка - потай впроваджена програма, яка створює загрозу для інформації, що міститься у комп'ютері [1]. Програмна закладка може бути реалізована у вигляді зловмисної програми чи зловмисного програмного забезпечення або незадокументованого програмного коду в програмному забезпеченні.

В якості об'єкту дослідження розглядатимемо незадокументовані закладки програмного забезпечення, які використовуються в локальних комп'ютерних мережах підприємств (організацій).

Складність виявлення такого таємно вписаного в програмне забезпечення функціонального об'єкту, який за певних умов здатний забезпечити несанкціонований програмний вплив, пов'язана з можливістю відсутності його прояву протягом тривалого часу. Такий об'єкт може бути частиною програмного комплексу, який виконує поставлені завдання, заміновати повністю певні частини програмного комплексу,

Міжнародний науково-технічний журнал
«ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ»
ISSN 2219-9365

займають певну потрібну програму. Як правило, такі несадокументовані закладки програмного забезпечення дозволяють зберігати заявлені виробником функції програмного забезпечення і реалізуються частиною функцій, які входять до програмного комплексу.

Підприємство може використовувати готове програмне забезпечення, в якому вже присутні несадокументовані закладки, або проблеми під замовлення, в якому була здійснена невдало його верифікація при прийнятті в експлуатацію.

Програмне забезпечення, яке експлуатується в локальних мережах підприємств, як правило, є розподіленим і тоді несадокументовані закладки програмного забезпечення є активними в усіх комп'ютерах мережі. Це підвищує загрози для підприємств та організацій.

Несадокументовані закладки програмного забезпечення можуть приймати участь в створені бот-мереж, впливу троянських програм, тощо. Тому, актуальною продовжує залишатись проблема виявлення зловмисного програмного забезпечення, зокрема, і несадокументованих закладок програмного забезпечення.

Одним із завдань, які потребують вирішення, є розробка моделей несадокументованих закладок програмного забезпечення в локальних комп'ютерних мережах.

Попередні роботи. Несадокументована закладка програмного забезпечення може бути частиною захищеної системи. Тоді вона здатна маскувати свою присутність в комп'ютерній системі. При цьому в системі створюється прихований канал інформаційного обміну. Він, як правило, залишається непоміченим для адміністраторів системи протягом тривалого часу. Виявити несадокументовану закладку програмного забезпечення стандартними засобами адміністрування складно. Вона може функціонувати необмежено тривалий час. Таким чином, протягом цього часу зловмисник отримує необмежений доступ до системних ресурсів.

Оскільки всі події відбуваються в локальних комп'ютерних мережах, то потрібна розподілена система, яка здійснюватиме тривале спостереження та аналіз подій на предмет виявлення несадокументованих закладок програмного забезпечення.

Розроблені в [2]-[5] розподілені системи дозволяють здійснювати виявлення зловмисного програмного забезпечення (ЗПЗ), але орієнтовані переважно на прояви більшої інтенсивності, ніж прояви, які отримуються від несадокументованих закладок програмного забезпечення (НЗПЗ). Крім того, НЗПЗ мають певну специфіку, яка фактично є лише частиною складового ЗПЗ. Також, засобами виявлення таких НЗПЗ можуть бути спеціального виду приманки [6], які розміщуються в мережі для зловмисника і виступають помилковими об'єктами атаки. Вони дозволяють зловмиснику проявити свої дії на них. І це відбувається швидше, ніж для реальних об'єктів атаки. Але всі розглянуті методи та засоби необхідно відповідним чином скомбінувати, узгодити та налаштувати. Крім того, методи та засоби виявлення мають враховувати наявні моделі НЗПЗ та ті, які з'являтимуться в майбутньому. Тому, виявлення НЗПЗ потребує доповнення розподілених систем виявлення новими методами, розробленими на основі актуальних моделей НЗПЗ.

Метою роботи є розробка нових моделей НЗПЗ та їх використання в РБС [5] виявлення ЗПЗ в ЛКМ для покращення ефективності виявлення.

Основна частина. Розглянемо види загроз від НЗПЗ, які можуть бути здійснені в локальній мережі. Їх аналіз пов'язаний з вимогами, які висуваються до безпеки комп'ютерних систем в мережі: конфіденційність, цілісність, доступність та аутентичність. Для порушення цих вимог розробники НЗПЗ закладають в його механізми здійснення загроз у вигляді таких атак: перехоплення, перехоплення, зміна, підробка. В локальних мережах здійснення таких атак або їх комбінацій відбувається по відношенню до апаратного забезпечення, програмного забезпечення, ліній зв'язку та даних. На рис. 1 зображено об'єкти в комп'ютерних системах локальних мереж, які можуть бути піддані атакам та повним типами загроз.

Рис. 1 Об'єкти комп'ютерних систем, які можуть бути піддані атакам

Позначення:

$i = \overline{1, n}$;

Міжнародний науково-технічний журнал
«ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ»
ISSN 2219-9365

$P_{i,1}$ – множина файлів i – ої комп'ютерної системи;

$P_{i,2}$ – множина користувацьких процесів i – ої комп'ютерної системи;

$P_{i,3}$ – множина запитів користувачів i – ої комп'ютерної системи;

$P_{i,4}$ – множина мережних пакетів i – ої комп'ютерної системи.

Функціонування комп'ютерних систем в локальних мережах пов'язане з обробкою, зберіганням та поширенням інформації. Саме при виконанні цих дій можливим є здійснення атак, узагальнені види яких виділимо наступним чином:

1) $Z_{i,1}$ – множина несанкціонованих змін;

2) $Z_{i,2}$ – множина підроблених об'єктів, розміщених в систему в результаті атаки;

3) $Z_{i,3}$ – множина перехоплень зі сторони зловмисника засобами програм або комп'ютерів;

4) $Z_{i,4}$ – множина переривань, яка здійснена для виведення з ладу компонентів системи.

Розглянемо детальніше можливі події при проведенні атак. Зокрема, елемент множини $P_{i,1}$ може бути скопійований чи перенесений в інше місце пам'яті. Тоді, можливі два випадки: ця подія відбулась успішно, ця подія не відбулась (або через помилки, пов'язані з роботою операційних систем чи компонентів комп'ютерної системи; або в результаті проведеної атаки). Задамо можливі події за формулою (1):

$$P_{f,1} \xrightarrow{\substack{f \\ z}} P_{f,1} \quad (1)$$
$$P_{f,1} \xrightarrow{} P_{f,1,z}$$

Задамо матрицями категорії атак [7]. Введемо такі вершини матриці: джерело інформації, отримувач інформації, подія переривання, подія перехоплення, подія зміни, підробка. Зв'язки між цими вершинами зобразимо напрямленими дугами. Зображення графу, що відповідає таким подіям, на рис. 2.

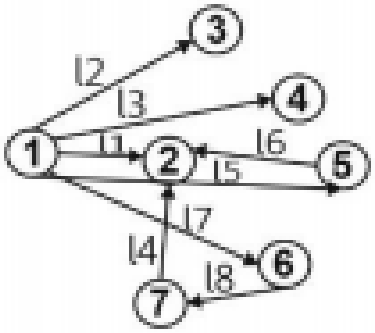


Рис. 2 Граф видів загроз

Позначення:
1-джерело інформації;
2-отримувач інформації;
3-подія переривання;
4-подія зміни інформації;
5-вершина, в якій відбувається перехоплення інформації;
6-отримання перехопленої інформації;
7-подія підробки інформації;
l_1 – передавання інформації здійснено вірно;

l_2 – відбулось переривання передавання інформації і вона не дійшла до отримувача;

l_3 – передавання інформації перервано і здійснюється її зміна;

l_4 – змінена інформація надсилається отримувачу;

l_5 – передавання інформації перехоплено, при цьому вона далі передається отримувачу без спотворень;

Міжнародний науково-технічний журнал
«ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ»
ISSN 2219-9365

l_6 – передавання перехопленої інформації далі отримувачу;

l_7 – перехоплена інформація обробляється зловмисником;

l_8 - інформація від джерела не надсилалась, але зловмисник надсилає псевдо підроблену інформацію до отримувача.

Матриця інцидентності, що відповідає графу з рис. 2 зображена табл. 1.

Таблиця 1

Матриця інцидентності видів загроз

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
1	1	1	1	0	1	0	0	0
2	-1	0	0	-1	0	-1	0	-1
3	0	-1	0	0	0	0	0	0
4	0	0	-1	1	0	0	0	0
5	0	0	0	0	-1	1	1	0
6	0	0	0	0	0	0	-1	0
7	0	0	0	0	0	0	0	1

Види загроз залежать від особливостей комп'ютерних систем та їх компонентів. Виділимо компоненти, для яких встановимо їх взаємозв'язок з видами загроз: апаратне забезпечення, програмне забезпечення, дані, засоби організації зв'язку. Апаратне забезпечення через його доступність після збоїв викликаних втручанням може відновляти в обслуговуванні. Програмне забезпечення може мати такі види загроз: відмова користувачам в доступі, несанкціоноване копіювання, зміна функціоналу програми. Дані, які зберігаються, можуть бути видалені через видалення файлів, відмовити в доступі до них користувачам, несанкціоновано прочитані, змінено їх вміст. Засоби організації зв'язку можуть мати такі загрози, при здійсненні яких дозволять читання повідомлень, спостереження за трафіком, зміну вмісту, зміну часу доставки, порядку доставки повідомлень або їх дублювання, підробка повідомлень, видалення повідомлень. Таким чином, здійснено виділення типових загроз у локальних мережах та запропоновано їх формалізоване представлення, використання якого є важливим при створенні розподілених систем виявлення зловмисного програмного забезпечення.

Позначимо множину всього ЗПЗ через $\tilde{V}$, яке перебуває в комп'ютерах локальних мережах. Тобто розглядатимемо ті ЗПЗ, яке за певних обставин та на протязі певного часу експлуатації локальних комп'ютерних мереж, проникло в комп'ютерні системи, змогло пройти певні системи захисту і функціонує там. Серед такого ЗПЗ є, також, несадокументовані закладки програмного забезпечення окремими об'єктами або частинами інших об'єктів. Представимо ЗПЗ в локальних комп'ютерних мережах алгебраїчною системою типу $\tau = (\alpha, \beta)$:

$$\tilde{V}_T = \langle V, \Omega_p, \Omega_p \rangle, \quad (2)$$

де $\Omega_p = \{F_0, F_1, F_2, \ldots, F_{\alpha_1}, \ldots\}$ – множина операцій заданих на множині V для кожного $\alpha_1 = 0, 1, 2, \ldots$; $\Omega_p = \{P_0, P_1, P_2, \ldots, P_{\beta_1}, \ldots\}$ – множина предикатів заданих на множині V для кожного $\beta_1 = 0, 1, 2, \ldots$; $\alpha = 1$, $\beta = 1$ –

файлами, оперативною пам'яттю та командами роботи в мережі: створення, відкриття, закриття, видалення, читання, записування, додавання, знаходження, отримання атрибутів і встановлення атрибутів, команди доступу до ОП, команди для роботи в мережі. Реалізація характеристичних властивостей ЗПЗ пов'язана з системними викликами та командами для роботи в мережі визначатиме виконання функцій з множини Ω_p і залежатиме від них, що дозволить ідентифікувати такі дії.

На основі видів загроз і моделі ЗПЗ задано для НЗПЗ моделі, які потрібні для систем їх виявлення.

Впровадження несадокументованих програмних закладок на різних етапах життєвого циклу програмного забезпечення може відбуватись так:

1) робота зловмисників в складі розробників програмних засобів;

2) створення команд, які динамічно формуються, або паралельних обчислювальних процесів;

3) здійснення переадресації команд та запис зловмисної інформації в використовувані інформаційною системою або іншими програмами ділянки пам'яті;

4) внесення в програмний код НЗПЗ;

5) створення замаскованого пускового механізму НЗПЗ;

6) внесення НЗПЗ в окремі підпрограми і в керуючу програму;

7) підготовка тестових даних для виявлення НЗПЗ;

8) приховування НЗПЗ внесенням в програмний засіб помилок;

9) розміщення НЗПЗ в гілках програмного засобу, які не перевіряються при контролі;

Міжнародний науково-технічний журнал
«ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ»
ISSN 2219-9365

10) участь зловмисників при здійсненні верифікації;

11) розробка НЗПЗ при доопрацюванні програмного засобу;

12) розробка оновлення та доповнення для НЗПЗ.

Подальше використання НЗПЗ може здійснюватись зловмисниками, які працюють безпосередньо на підприємстві особисто або через третіх осіб, або віддалено з використанням відповідних технічних засобів.

Моделі НЗПЗ в комп'ютерах локальних мереж:

1) модель «перехоплення», в якій НЗПЗ розміщується в програмне забезпечення, зберігає всі або вибрані фрагменти ПЗ, вводиться або виводиться в приховані області локальної або віддаленої зовнішньої пам'яті прямого доступу; об'єктом зберігання може бути клавіатурне введення, документ, що виводиться на принтер, або створюються файли-документи; для цієї моделі потрібна наявність у зовнішній пам'яті місця зберігання інформації, яке має бути організоване таким чином, щоб забезпечити її збереження протягом заданого проміжку часу і можливість подальшого зняття та приховування від інших користувачів чи процесів;

2) модель «шпигуна», в якій НЗПЗ вбудовується в мережеве або телекомунікаційне програмне забезпечення; дане програмне забезпечення, як правило, завжди активне, тому НЗПЗ здійснює контроль за процесами обробки інформації в комп'ютері, установку і видалення закладок, а також зняття накопиченої інформації; НЗПЗ може ініціювати події для раннього впровадження закладок;

3) модель «компрометації», в якій НЗПЗ або передає задану зловмисником інформацію (наприклад, клавіатурний ввід) в канал зв'язку, або зберігає її, не покладаючись на гарантовану можливість подальшого прийому або зняття; НЗПЗ може, також, ініціювати постійне звернення до інформації, що приводить до зростання відношення сигнал / шум при перехопленні побічних випромінювань;

4) модель «спотворення або ініціатор помилок», в якій НЗПЗ спотворює потоки даних, що виникають при роботі прикладних програм (вихідні потоки), або спотворює вхідні потоки інформації, або ініціює (або провокує) виникають при роботі прикладних програм помилки;

5) модель «прибирання сміття», в якій НЗПЗ при здійсненні прямого впливу на програмний засіб може і не створити руйнівного результату; основною метою такого впливу є забезпечення максимізації утворених «сміттєвих» інформації для подальшого вивчення; зловмисник отримує або дані фрагменти, використовуючи закладки попередніх моделей, або безпосередній доступ до комп'ютера під виглядом ремонту або профілактики.

НЗПЗ в процесі свого функціонування створюватиме свої процеси в комп'ютерних системах та впливатиме на інформаційні потоки. Крім того, результати її функціонування можуть впливати на інші комп'ютери в локальній мережі, що може і повинно досліджуватися з метою виявлення НЗПЗ. Для цього потребуватимуть моніторингу інформаційні процеси, які відносяться до звернень до мережних та файлових ресурсів, а також, до паролів користувача.

Дослідження НЗПЗ професійними фахівцями з кібербезпеки дозволяє встановити їх наявність за такими ознаками: наявність модулів ПЗ, які не відповідають призначенню процесу; наявність об'єктів операційних систем, які відкриті процесом, що не відповідають призначенню процесу; висока інтенсивність операцій введення-виведення зі сторони певного процесу; великий відсоток завантаження процесора або внутрішньої пам'яті зі сторони певного процесу; подібність імені файла до імені файла, що відноситься до операційної системи; виконуваний файл процесу операційної системи розміщено не в загальноприйнятому каталозі; процес, який відноситься до операційної системи, виконується від імені локального користувача; системи захисту від виконання коду в області даних, які вимкнена для всіх процесів, для розглядуваного процесу відімкнена для процесу, що відносить до операційної системи, задано інших каталог, відмінний від того, що повинен бути для такого процесу; відсутній цифровий підпис у виконуваних файлах програмного засобу; велика мережна активність процесу, який повинен працювати локально; тощо. Але для покращення ефективності виявлення НЗПЗ потрібні засоби, які дозволять здійснювати встановлення факту наявності НЗПЗ без втручання адміністратора мережі, який може не опрацювати певні з ознак з різних причин. НЗПЗ можуть використовувати засоби маскування в системі, що ускладнює їх виявлення.

З технічної сторони при створенні ПЗ з наявними НЗПЗ використовують методи програмування, які не є поширеними при створенні типового програмного забезпечення, тому ці особливості можуть теж бути додатковими ознаками для їх виявлення. Зокрема, для виконуваного файлу операційного середовища Windows такими ознаками можуть бути такі: додаткова секція в кінці файла; точка входу вказує на перехід в середину секції, яка не є секцією коду; точка входу вказує на команду переходу, яка задає перехід за секцію коду; наявність ознак секції коду, яка не є секцією коду. Аналогічно для інших середовищ в комп'ютерній системі, інформація з яких може бути пов'язана з розміщенням в оперативній пам'яті.

Моделі НЗПЗ є основою для їх подальшої формалізації та використання в розподільних системах виявлення.

Експериментальні дослідження. Для проведення експериментів було використано розподілену багаторівневу систему виявлення ЗПЗ [5]. НЗПЗ було розроблено як складову кожної з типових бот-мереж. Тоді, метою експерименту була перевірка застосування методу виявлення бот-мереж, роботи класифікатору в структурі розподільної системи та визначення залежності відсотку виявлених вузлів бот-мережі від їх представлення векторами, в складі яких були НЗПЗ. Для проведення експериментів було здійснено

Міжнародний науково-технічний журнал
«ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ»
ISSN 2219-9365

конструювання 28 штучних бот-мереж та отриманих коїв відомих виявлених бот-мереж, згрупувано їх за класами, виділено в них 25 структурних елементів в трьох стадіях функціонування і 81 функцію, причому не всі так отримані бот-мережі містили повністю всі структурні елементи та функції. Експеримент проводився для класифікатора без додавання екземплярів створених бот-мереж та з ними, тобто здійснювалась перевірка без навчання класифікатору на створених зразках і з попереднім відношенням зразків по класах. Другий варіант є необхідним для перевірки точності відношення до класів тих же зразків є в них введені, бо при здійсненні моніторингу API функцій можуть бути похибки. Тривалість моніторингу КС локальної мережі становила 350 годин для кожного екземпляру бот-мережі кожного з двох класифікаторів. Атаки з вузлів бот-мережі не здійснювалась. Вузли бот-мережі працювали тільки в режимі контролю КС та підтримки структури бот-мережі через відправлені повідомлення. Таким чином, для компонент РВС об'єктами дослідження були ситуації в КС процеси і відповідно побудова векторів по них. Для проведення експерименту було обрані бот-мережі, які використовують стратегію отримання повного контролю в КС за рахунок активації їх складових НЗПВ. Для здійснення експерименту засобами API моніторингу в КС було отримано вектори, які почергово оброблено класифікатором компоненти. Результати обробки представлено в табл. 2.

Експерименти передбачали виявлення наступних показників ефективності виявлення вузлів бот-мереж для класів і підкласів класифікатору:

1) $P_{1,1}$ – відсоток векторів зловмисних дій та атак для вузлів бот-мереж, що належать даному класу відносно всіх тестових зразків, які система відносила до цього класу з використанням попереднього навчання;

2) $P_{1,2}$ – аналогічно до п. 1, однак без використання попереднього навчання;

3) $P_{2,1}$ – відсоток векторів зловмисних дій та атак для вузлів бот-мереж, що належать даному підкласу класу відносно всіх тестових векторів, які система відносила до цього підкласу класу в тестовій вибірці (ті, які були правильно віднесені до підкласів) з використанням попереднього навчання;

4) $P_{2,2}$ – аналогічно до п. 3, однак без використання попереднього навчання;

5) $P_{3,1}$ – відсоток правильно виявлених вузлів бот-мереж з використанням попереднього навчання;

6) $P_{3,2}$ – аналогічно до п. 5, однак без використання попереднього навчання;

7) $P_{4,1}$ – відсоток неправильно класифікованих вузлів бот-мереж як корисних додатків (помилка 1-го роду) з використанням попереднього навчання;

8) $P_{4,2}$ – аналогічно до п. 7, однак без використання попереднього навчання;

9) $P_{5,1}$ – відсоток неправильно класифікованих вузлів бот-мереж як таких, що є вузлами бот-мереж, але віднесені не до того класу (помилка 3-го роду), з використанням попереднього навчання;

10) $P_{5,2}$ – аналогічно до п. 9, однак без використання попереднього навчання.

Результати оцінки ефективності виявлення програмного забезпечення вузлів бот-мереж на основі роботи двох класифікаторів для введених класів та підкласів у класифікаторі наведено у табл. 2.

Таблиця 2

Результати експерименту

Показники експерименту	Отримані значення для різних класів							Середні значення
	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	
$P_{1,1}$, %	90,74	84,29	73,66	86,30	94,04	94,18	96,60	88,44
$P_{1,2}$, %	75,93	63,57	68,22	70,32	68,77	67,60	69,36	67,71
$P_{2,1}$, %	85,88	88,57	72,58	85,39	98,88	93,92	96,60	88,42
$P_{2,2}$, %	74,69	63,57	59,14	70,32	67,37	66,58	67,66	66,80
$P_{3,1}$, %	92,11	84,21	71,83	89,47	90,53	88,42	93,68	87,72
$P_{3,2}$, %	76,32	57,89	63,16	64,91	71,58	54,74	75,79	65,89
$P_{4,1}$, %	7,89	14,47	26,87	10,53	7,37	11,58	6,32	11,70
$P_{4,2}$, %	21,05	40,79	36,84	31,58	24,21	44,21	22,11	31,97
$P_{5,1}$, %	0	1,32	0	0	2,11	0	0	0,01
$P_{5,2}$, %	2,63	1,32	0	3,51	4,21	1,05	2,11	2,14

В результаті проведення експерименту отримано відношення до потрібного підкласу та класу

Міжнародний науково-технічний журнал
«ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ»
ISSN 2219-9365

отриманих на основі моніторингу векторів з точністю до 66% для класифікатору без введених векторів 28 штучно згенерованих бот-мереж та 88% для класифікатору, в який попередньо було додано вектори шляхом здійснення його навчання, зберігаючи в ньому шаблони попередніх напрямків. Відсоток спиж, які були використані РБС для виявлення бот-мереж і пов'язані з проявами НЗПЗ, становить приблизно 27% від загальної кількості виявлених. Інтенсивність проявів від НЗПЗ суттєво нижче від типових проявів бот-мереж. Таким чином, НЗПЗ в складі бот-мереж можуть бути виявлені розподіленими багаторівневими системами [8] і напрям таких досліджень є перспективним.

Висновки. Несадокументовані закладки програмного забезпечення, яке використовується в локальних комп'ютерних мережах, можуть завдавати значної шкоди користувачам персональних комп'ютерів, а особливо підприємствам, які експлуатують комп'ютерні мережі та використовують спеціалізоване програмне забезпечення.

Моделі несадокументованих закладок програмного забезпечення дозволяють після відповідної формалізації включати їх в засоби виявлення. Застосування розроблених моделей НЗПЗ в РБС [8] дало можливість покращити ефективність виявлення бот-мереж, складовими яких вони були.

Напрямком подальших досліджень є конкретизація та визначення множини функцій, які формуватимуть елементи НЗПЗ, з метою представлення їх поведінковими сигнатурами для покращення ефективності виявлення.

Література

1. ДЕРЖАВНИЙ СТАНДАРТ УКРАЇНИ Захист інформації. Технічний захист інформації. Терміни та визначення. ДСТУ 3396.2-97

2. Kumar, N.J., Singh, P., Bali, R.S., Misra, S., Ullah, S. (2015). An intelligent clustering scheme for distributed intrusion detection in vehicular cloud computing, Cluster Computing, 18(3), 1263–1683. DOI: 10.1007/s10586-015-0463-7

3. Ibrokhimof, D., Kumar, O., Kahlouf, L. (2016). Network Security: Distributed Intrusion Detection System using Mobile Agent Technology, International Journal of Communication Networks and Distributed Systems, 16(4). DOI: 10.1504/IJCNDS.2016.10001612

4. Ibrokhimof, D., Kumar, O. (2012). Hybrid Approach based Mobile Agent for Distributed Intrusion Detection System, Journal of Information Security Research, 3(1), 30–40. DOI: 10.1109/ICEEI.2012.6360647

5. Markowsky, G., Savenko, O., Sachenko, A. (2019). Distributed Malware Detection System Based on Decentralized Architecture in Local Area Networks. Advances in Intelligent Systems and Computing III, 871, 582–598. DOI: 10.1007/978-3-030-01069-0_42

6. Sochor T. Attractiveness Study of Honeypots and Honeynets in Internet Threat Detection / T. Sochor, M. Zuzcak // Proceedings of the 22-nd International Conference Computer Networks. – Brunów (Poland), June 16-19, 2015, Vol. 522. – Pp. 69-81.

7. Саенко О. С. Генерація моделей комп'ютерних вірусних програм в системі оцінки достовірності результатів роботи антивірусних засобів / О. С. Саенко, С. В. Мостовий // Вісник Хмельницького національного університету. Технічні науки. – 2005. – № 4, т. 1. – С. 198-200.

8. Саенко О. С. Архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Саенко // Вчені записки Таврійського національного університету. Технічні науки. – 2018. – Т. 29 (68), № 2. – С. 172–181.

Reference

1. DERZHAVNI STANDART UKRAYINI Zahist informaciyi. Tehnichnij zahist informaciyi. Tarmini ta viznachennya. DSTU 3396.2-97

2. Kumar, N.J., Singh, P., Bali, R.S., Misra, S., Ullah, S. (2015). An intelligent clustering scheme for distributed intrusion detection in vehicular cloud computing, Cluster Computing, 18(3), 1263–1683. DOI: 10.1007/s10586-015-0463-7

3. Ibrokhimof, D., Kumar, O., Kahlouf, L. (2016). Network Security: Distributed Intrusion Detection System using Mobile Agent Technology, International Journal of Communication Networks and Distributed Systems, 16(4). DOI: 10.1504/IJCNDS.2016.10001612

4. Ibrokhimof, D., Kumar, O. (2012). Hybrid Approach based Mobile Agent for Distributed Intrusion Detection System, Journal of Information Security Research, 3(1), 30–40. DOI: 10.1109/ICEEI.2012.6360647

5. Markowsky, G., Savenko, O., Sachenko, A. (2019). Distributed Malware Detection System Based on Decentralized Architecture in Local Area Networks. Advances in Intelligent Systems and Computing III, 871, 582–598. DOI: 10.1007/978-3-030-01069-0_42

6. Sochor T. Attractiveness Study of Honeypots and Honeynets in Internet Threat Detection / T. Sochor, M. Zuzcak // Proceedings of the 22-nd International Conference Computer Networks. – Brunów (Poland), June 16-19, 2015, Vol. 522. – Pp. 69-81.

7. Savenko O. S. Generaciya modelej kompʼyuternih virusnih program v sistemi ocinki dostovirnosti rezultativ roboti antivirusnih zasobiv / O. S. Savenko, S. V. Mostovij // Visnik Hmelnickogo nacionalnogo universitetu. Tehnichni nauki. – 2005. – № 4, t. 1. – S. 198-200.

8. Savenko O. S. Arhitektura rozpodilenoyi bagatorivnevoyi sistemi viyavlennya shkidlivogo programnogo zabezpechennya v lokalnih kompʼyuternih merezhah / O. S. Savenko // Vcheni zapiski Tavrijskogo nacionalnogo universitetu. Tehnichni nauki. – 2018. – T. 29 (68), № 2. – S. 172–181.

Рецензія/Peer review : 26.11.2019 Надрукована/Printed : 09.01.2020

2. Stetsyuk M., Bedratyuk L., Savenko B., Stetsyuk V., Savenko O. Providing the Resilience and Survivability of Specialized Information Technology Across Corporate Computer Networks. CEUR-WS. 2020. Vol. 2623. P. 219-238.

Providing the Resilience and Survivability of Specialized Information Technology Across Corporate Computer Networks

Mykola Stetsyuk[0000-0003-3875-0416], Leonid Bedratyuk[0000-0002-6076-5772], Bohdan Savenko [0000-0001-5647-9979], Vasyl Stetsyuk[0000-0001-9880-2666] and Oleg Savenko[0000-0002-4104-745X],

Khmelnytsky National University, Khmelnytsky, Ukraine
mikstt777@gmail.com, leonid.uk@gmail.com,
savenko_bohdan@ukr.net, swmuau@gmail.com,
savenko_oleg_st@ukr.net

Abstract. The approach to determining the effectiveness of IT based on quantitative values that characterize resilience and survivability is developed and can be expanded to include other characteristic values. To ensure the resilience and survivability of IT, a system of measures has been developed which results in the acquisition of highly specialized IT for various applications, where the accompanying processes are irrational or unrealistic times with sufficiently high parameters of resilience, survivability and overall resilience. at the same time, acceptable to the financial cost of its operation. The fault tolerance of the client part is ensured by performing a set of measures, including hardware and functional redundancy: power of client PCs from a separate line with security devices; use of lightning protection devices on computer network lines organization of automatic updating of client PC system software; development of algorithms of procedures that implement the critical functions of the client part of the IT, with the inclusion of non-trivial (intellectual) block of error processing in them, which is performed in parallel with the procedure itself; the use of non-trivial data editors that include an interactive pro-procedure in their algorithm that eliminates uncontrolled manipulation of the database data by the operator; implementation of critical resources, calculation procedures with the ability to promptly select the location of their execution, which does not allow overloading of hardware. The survivability of IT is ensured by: redundancy of the server part of the IT with the territorial separation of the main and backup server; redundancy of the client part software, the feature of the backup is the fact that the reserve is not specially dedicated computers, but the performance reserve of individual client computers, on which, according to the backup plan, the client part software is installed, which redundant, which at a critical moment will be used as a regular, preventing the loss of IT functionality.

Keywords: **Keywords:** Software, Fault tolerance, Resilience, Survivability, Computer.

1 Introduction

For information technologies (IT) that provide the livelihoods of an institution or enterprise, that is, specialized in, for example, areas such as financial and economic activities, the issues of survivability and resilience are more important, especially as their quantitative parameters increase in their function. zoning (increase of users, servers, volumes of information in databases of them) and level of complexity.

Under fault tolerance, we will consider the property of the system to maintain full or partial performance in cases of failure of individual elements that are not related to external unregulated activities. The survivability of an information system means its ability to remain operable with a permissible decrease in productivity in the face of negative external influences (unregulated actions). These concepts correspond to the State Standard of Ukraine [1]. They set one goal - to ensure the availability of IT, which is achieved in different ways. The effectiveness of all IT is directly dependent on providing these parameters. One of its parameters is the time of unavailability, that is, the time when the system is unable to perform its functions within certain requirements. For different systems this time is different and may range from zero to a certain, still acceptable value. So for automated control systems of complex technological processes, the time of inaccessibility is zero. For such critical systems, the probability of non-accessibility should be zero. For specialized IT operating in corporate computer networks and serving as information support in such highly specialized subject area as financial and economic activity in various fields of application, this parameter is well above zero, but the requirements for such IT are also high enough that it is impossible perform at low parameters of fault tolerance and survivability, especially with a steady increase in the number of users, increasing the complexity of information flows and volumes of data processed.

Ensuring high efficiency of specialized IT is carried out on the basis of the implementation of the principles of resilience and survivability in them, which is an urgent scientific problem, which begins to be solved in the process of developing specialized IT.

2 Related works

Known methods and techniques for providing resilience and survivability of specialized IT are focused on their different types, applications, application features, computer deployment and implementation features in various IT components. Also, an urgent area that needs research is the impact of external factors (distributed attacks, malware) on the functioning of IT and the resilience and survivability.

In [2], information technology was proposed to evaluate the structural reliability of technical objects, the structure of which corresponds to one of the known types of neural networks. The structure of information technology contains a morphological model, which allows to shape and change the structure of the object model studied by the rules, based on the determination of the probability of failure-free operation in the

theory of reality. The model developed allows you to modify the object model, which takes into account the resilience of IT.

In [3, 4] the reliability of information systems is considered. An approach based on risk assessment and risk mitigation is used to enhance the reliability of information systems. This approach allows for an early assessment of the risk of the software development process and identifies the most effective mitigation strategies.

In [5] it is shown how customizable software systems consist of many different, critical, non-critical and interdependent configurations. The reliability and efficiency of a configured system depends on the successful termination of communication or interaction between its configurations. Most of the time, users of configured systems are more likely to use critical configurations than non-critical configurations. The paper shows how failure of critical configurations will affect the reliability and performance of the system. To solve the problem, critical configurations that play a vital role are investigated, and a suitable candidate for failures is provided for each critical configuration. The article proposes an algorithm that identifies the optimal candidate failure for each critical configuration of the software system. Two schemes for classification of configurations are offered - critical and non-critical configurations based on: 1) frequency of configuration interactions, 2) characteristics and frequency of interactions. These schemes have played a very important role in achieving the reliability and resiliency of the software system in a cost-effective manner. Schema performance was tested using a file structure system.

In [6], clouds are considered to be an important platform for scientific work programs. However, as many nodes deploy in the clouds, managing resource reliability becomes a critical issue, especially for performing real-time, real-time workflows where deadlines are met. Therefore, cloud resilience is extremely important. PB (Primary Backup) scheduling is a popular technique for fault tolerance and is effectively used in cluster and network computing. However, applying this technique to real-time workflows in a virtualized cloud is much more complex and rarely explored. This work develops a real-time workflow failure model that extends the traditional PB model to include cloud performance. This model builds on task allocation and messaging approaches to make sure mistakes can be made while running a workflow. Also, a dynamic fault-tolerant scheduling algorithm is proposed, rather, for real-time workflows in a virtualized cloud. It has three key features: 1) it uses a backward displacement method to take full advantage of idle resources and uses overlapping tasks and VM migration for high resource utilization; 2) applies vertical / horizontal scaling technology to quickly secure resources 3) uses a vertical reduction scheme to avoid unnecessary and inefficient changes to resources due to fluctuations in workflow requests. The evaluation of the algorithm is based on synthetic workflows and workflows compiled from real scientific and business applications, and compared with six basic algorithms.

Articles [7, 8] investigate cloud applications that are considered to be components of several cloud services components that communicate with each other through web services interfaces, where each component performs certain functional capabilities. The lack of an effective failure resiliency scheme is one of the main obstacles to increasing the availability and efficiency of complex cloud systems for deployment. The paper proposes a comprehensive recovery scheme based on software rejuvenation

for cloud applications, which has three essential parts: adaptive fault detection, aging assessment, and component-based rejuvenation checkpoint. Preliminary and qualitative evaluations show that the new resiliency scheme brings improvements in the availability of cloud programs.

In [9] presented an approach for avoiding functional failures during execution in component application systems. The approach uses the internal redundancy of components to find workarounds as alternate sequences of operations to avoid crashes. The first Java prototype is presented and an evaluation plan developed as a preliminary result.

In [10] a proactive recovery scheme based on migration of services is proposed, tolerant systems are described. Active recovery is an important method of ensuring this. The main advantage of the developed proactive recovery scheme is the reduction of vulnerability in normal operation. This is achieved in two ways.

In [11], refusal tolerance is a major issue in guaranteeing the availability and reliability of critical services, as well as the implementation of applications. In order to minimize the impact of failures on the system and the execution of applications, deviations should be anticipated and acted upon. Failure tolerance methods are used to predict these failures and take appropriate action before the failures actually occur. This document discusses existing cloud computing resilience methods based on their policies, tools used, and research issues. A cloud-based virtualization-based system architecture is proposed. The proposed system implemented an autonomous rejection. The experimental results show that the proposed system can solve various software malfunctions for server-side applications in a cloud-based virtualized environment.

In [12], cloud computing offers new power and flexibility for high-performance computing applications with the provision of a large number of virtual machines for computing intensive applications. Fault Tolerance allows systems in a cloud with multiple nodes to complete computationally intensive applications at the moment of failure. The most common fault tolerance methods for such systems are checkpoint / restart. However, a checkpoint / restart increases program execution time, which increases the cost of running it. This paper introduces the framework of resiliency for high performance cloud computing. This framework proposes to use process level redundancy methods to reduce the runtime of computationally intensive applications.

In [13-16], cyber-resilience and cyber-viability are presented as closely related concepts that share similar technologies and practices. For historical reasons, these concepts have been built into different frameworks that define different constructs to describe problems and areas of solution. Cyber-resilience constructs allow you to define system requirements, identify metrics and security, and identify and analyze solutions. Identifying the relationships between cyber-resilience constructs and cyber-survivorship attributes is shown to use cyber-resilience to enhance cyber-resilience and vice versa.

In [17-22] show the impact on the resilience and survivability of IT of various types of malware and computer attacks.

Known methods and methods for providing resiliency and survivability of specialized IT are not sufficiently systematized and may not always be implemented due to the specific use and structure of specialized IT. Therefore, it is necessary to further research and develop new methods and techniques that can improve the resiliency and survivability of specialized IT, including cyber-attacks and malware.

3 Criteria for the effectiveness and sustainability of specialized information technologies in corporate computer networks

Let's introduce specialized IT in corporate computer networks with many components:

$$S_{IT} = \{S_1, S_2, \dots, S_n\}, \quad (1)$$

where S_i - i component of specialized IT in corporate computer networks, $i = 1, 2, \dots, n$, n – number of components.

For each component S_i we will apply a feature that will include all performance criteria for enterprise computer networks that need to be used for IT development in the future. In particular, such criteria will include the criteria for fault tolerance and survivability. Specify the performance criteria for a specialized IT vector whose components will be performance features that will meet specific criteria:

$$K_e = (f_1, f_2, \dots, f_m), \quad (2)$$

where f_j - j a function that sets one of the performance criteria, $j = 1, 2, \dots, m$, m - number of functions.

Given that, overall, the task of maximizing performance depends on specific criteria that may be related to each other and affect each other accordingly, while improving one's performance may impair the other. In addition, because specialized IT is made up of components that are subject to the same criteria from a given vector, the task is complicated by the fact that some IT components are different and, accordingly, the achievement of efficiency by the same set of criteria will be different. Therefore, choosing the best solution is a difficult multicriteria task. The general statement of the task of finding the best performance for specialized IT in corporate computer networks is formulated as follows:

$$\begin{cases} K_e(S_{IT}) \to max; \\ f_j(S_i) \to max, i = 1, 2, \dots, n, j = 1, 2, \dots, m \end{cases}, \quad (3)$$

In addition, some of the IT components may be functionally repetitive, depending on the tasks and deployment on corporate computer networks. This will affect the overall effectiveness of specialized IT. However, achieving performance by certain criteria in the same components of specialized IT does not have to be the same, because these components will solve different tasks or the same tasks, but at different times they will go through different stages. Keeping these features in mind is important, so we detail the task of finding the best performance for specialized IT on enterprise computer networks as follows:

$$\begin{cases} K_e(S_{IT}) \to max; \\ f_{j,q}(S_{i,p}) \to max, i = 1, 2, \dots, n, j = 1, 2, \dots, m, \\ q = 0, 1, \dots, n_q, j = 0, 1, \dots, n_p \end{cases} \quad (4)$$

where q - the number of specialized IT components in a particular corporate computer network node; j - an index for the criterion of performance of a component of specialized IT in a specific node of a corporate computer network; $q = 0, 1, \dots, n_q, j = 0, 1, \dots, n_p$; n_q - the number of identical components of specialized IT in a corporate computer network; n_p - the criterion number for the same components of specialized IT in the corporate computer network.

Let us introduce a function that will determine the maximum value of the criterion of effect:

$$F: K_e(S_{IT}) \to max; \quad (5)$$

The value of the efficiency criterion is given by the expression taking into account the weighting factors:

$$K_e(S_{IT}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{p=0}^{n_p} \sum_{q=0}^{n_q} \left(\alpha_{i,j,p,q} \cdot f_{j,q}(S_{i,p}) \right), \quad (4)$$

where $\alpha_{i,j,p,q}$ - weighting factors.

Consider maximizing the criteria for failover and survivability in information technology configurations that are built on a client-server architecture with their provision across all systems from users (client side) to mission-critical server time. The choice to consider the client-server architecture depends on its features, which are manifested in the following: basic client-to-client functions are shared between the client and the server; the client computer's automated workplace software handles data through requests to the server software; full support for multi-user work; data integrity is guaranteed. This distinguishes it from other architectures and allows for the provision of fault tolerance and survivability to each of the units of the system separately.

The main directions for increasing the survivability and resiliency of IT are to make redundancy in the configuration of hardware and software, supporting infrastructure, redundancy of information resources (programs and da-them). In doing so, IT must meet the following basic requirements: the system must be built so that it does not have a component (resource), the failure of which will lead to a complete failure of the entire system. For real-time systems, in addition, time constraints are imposed on the result.

Consider specialized IT related to the systems of unrealistic cha-su. Therefore, time constraints are much less rigid for her than real-time systems. In view of the server-side and client-side IT implementation architecture chosen for consideration, we will accordingly consider the issue of resiliency and survivability in relation to the functions assigned to them. Despite the fact that these two parts, being components of a single, logically indissoluble IT, perform within their specific functions, ensuring resiliency for each component of IP is achieved in different ways.

There are two approaches to building fault-tolerant IT. The first approach is based on the use of fault tolerant components. Such IT provides its functions in the event of failure of subcomponents of some components. This is the simplest method, but also the most expensive, because of the use of the most expensive components - the fault

tolerant components of IT. The second way is to build fault-tolerant IT using non-fault-tolerant components. Failure in such systems is achieved through the introduction of redundancy in them through redundancy of critical links of hardware, software, intercomponent communications and special algorithms for the functioning of IT, which provide for its reconfiguration when some components fail.

The main feature of fault tolerance is the transparency of failures of its individual components for the end user. This means that the fault-tolerant system automatically changes its configuration in the event of failure. Its runtime software is looking for workarounds, trying in failure conditions to bring the executable function to a successful completion. We define the function $f_1(S_i), i = 1,2,\ldots,n$ quantification of fault tolerance in computer systems as follows:

$$f_1(S_i) = \frac{T_{f_1(S_i),1}}{T_{f_1(S_i),1} - (T_{f_1(S_i),2} + T_{f_1(S_i),3})}, \quad (5)$$

where i - number of components of specialized IT, $i = 1, 2, \ldots, n$, $T_{f_1(S_i),1}$ – time between adjacent failures; $T_{f_1(S_i),2}$ – the time it takes to detect a failure and find a way around it; $T_{f_1(S_i),3}$ – the time required to recover IT after a failure.

As can be seen from formula (7), for IT with an automatic fault tolerance system, it will approach the maximum, due to the response speed. There are no theoretical obstacles to the construction of such systems, but in practice, when implementing them, a number of important factors must be taken into account: financial costs of implementing an automatic system to ensure survivability and fault tolerance; system complexity. For IT designed for information support in a narrow specialized subject area, such as financial and economic activities of a higher education institution, it will be appropriate to abandon the automatic fault management system in favor of the automated one. With this approach, some of the costly functions of managing redundancies present in IT will be entrusted to the individual, unless it threatens possible significant losses. Then, according to formula (7), the fault tolerance will be lower than in the first case. But the solution to the problem of building IT (similarly to other design problems) is not to ensure the maximum possible fault tolerance of the system, but to find an acceptable balance of system parameters, within a certain technological basis. And also, including taking into account the requirements of the criterion "fault tolerance / cost". Let us explore the solutions to ensure IT resilience when using such a strategy. Let us analyze the factors that negatively affect the client's IT resilience. This is necessary in order to assess and develop adequate countermeasures. The scheme of influence of negative factors on the failure of the client side of specialized IT is shown in Fig. 1.

As can be seen from the proposed model, the negative factors that affect the resiliency of the client side of the IT are divided into external and internal. Among the external factors, the greatest threats are power outages and natural phenomena that can lead to failures of computer components and computer networks. In order to avoid such cases, the power of the client computers of the IT must be performed from a separate line

equipped with security devices, such as arresters. You also need to use security devices to protect against lightning storms over long lines on computer networks.

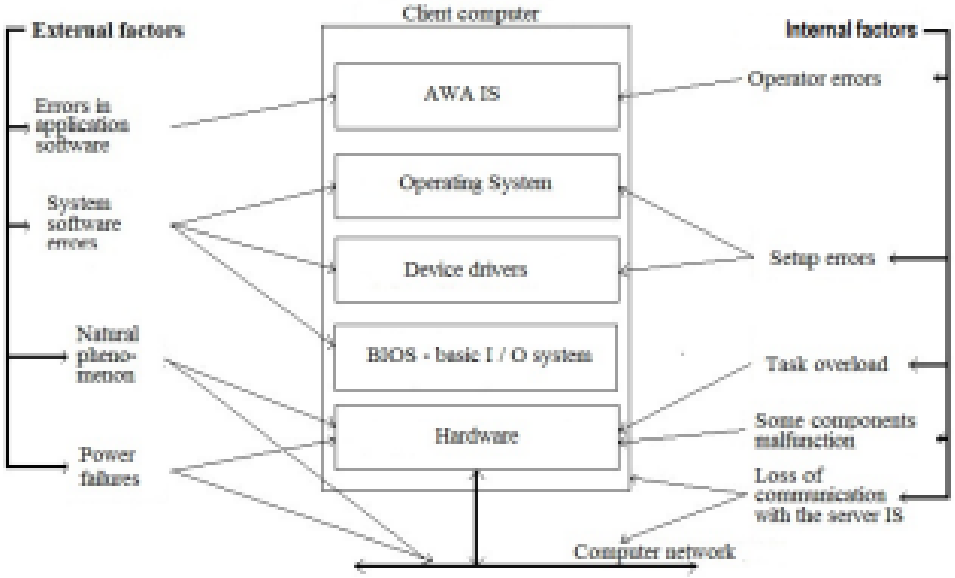


Fig. 1. The scheme of action of negative influencing factors on the fault tolerance of the client side of specialized IT

Another important factor is errors in the system software code. Previously, this factor was force majeure. Today the situation has changed and licensed operating systems can be configured with the automatic software update function enabled. This eliminates the human factor and reduces the burden on IT staff, although this does not fully solve the problem, but only reduces the likelihood of destructive behavior. The reason is that the system software is a complex system and each fix, a previously found error, does not guarantee the absence of a new one. This aspect of system software is another factor that can reduce fault tolerance. Due to its complexity, configuration errors may occur during software configuration. You can reduce its manifestations by using software with automatic adjustment, which is not always acceptable, and the involvement of more qualified personnel.

For application software, which includes client parts of specialized IT, the critical errors that occurred during the operation of jobs, are recorded together with their parameters in the system registry automatically and then used for analysis with me- the elimination of the causes that caused them. This is achieved through an approach that is based on introducing some redundancy into the software of the client part of IT. To this end, all the calculation procedures that may be critical to the functioning of the error, designed to comply with a certain type of template construction algorithm for its implementation. The essence of the algorithm is shown in Fig. 2.

In this structure, the algorithm for performing any non-trivial procedure is divided into two interacting blocks. The first block implements the function of the IT procedure, and in the second, the error handler. In the process of performing some procedure that implements one of the functions of the IT component, the two units interact with each other, transferring control of the computational process to each other until

the performed function is completed. Its essence is that the algorithm that implements the function of IT is divided by markers (label 1, ..., label n in Fig. 2) into fragments according to the principle of functional completeness.

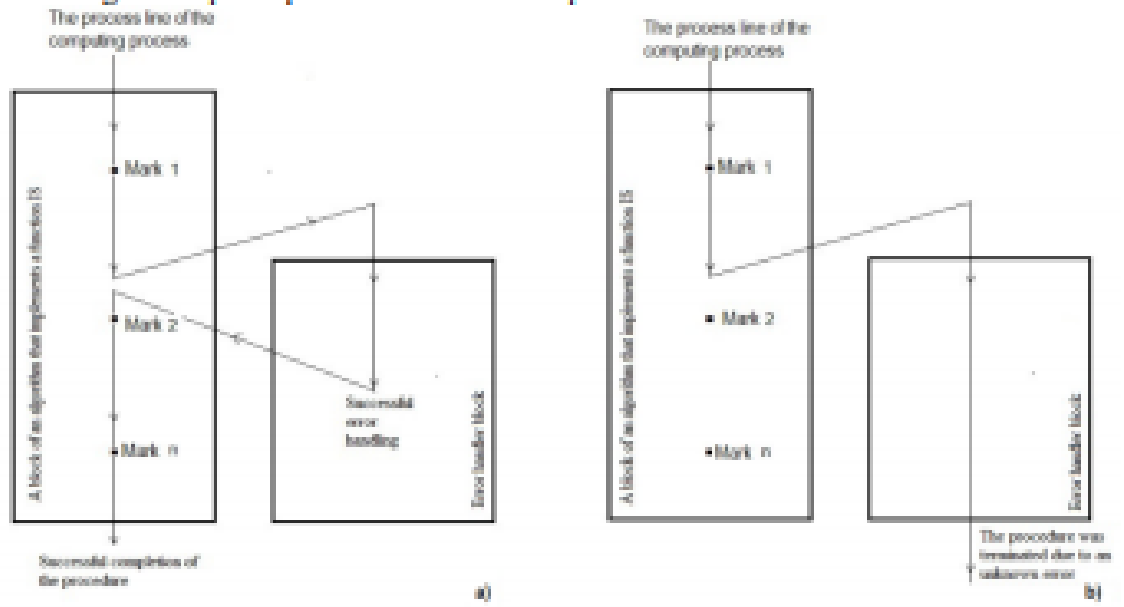


Fig. 2. Model of the algorithm procedure: a) in case of successful completion of the procedure; b) for the case where the error is unknown to the error handler

Before executing the current fragment of the algorithm, information about a hypothetically possible error (the client's workplace instance code, function code, tag number, time, etc.) is entered in the fatal error register. The following are possible developments in the future:

1. The fragment of the function algorithm was successfully executed. In this case, the information in the registry about the failed error is deleted and the computing process proceeds to the next fragment.

2. An error occurred while executing the fragment, but it was successfully localized by the error handler (Fig. 2a). In this case, the error information can also be deleted from the registry.

3. An error occurred during the execution of the fragment, which was not localized by the error handler (Fig. 2b). In this case, information about a possible error will remain in the registry.

Collected in this way, information about fatal errors that occurred in the process of the functioning of IT, allows them to classify and, in the process of further analysis, identify weaknesses in IT to address them by improving the software of the client side of IT.

The software of client IT parts during the IT life cycle, for various reasons, including due to the detection of errors in it, can change, going through their own update cycles (Fig. 3).

Consider the internal factors that affect the resilience of the client part of IT (Fig. 1) and the methods that were used to reduce it. The first of these, by frequency of occurrence, occurs because of operator errors. This problem is solved by using a standard data editor, in which all the procedures for making changes to the database are implemented using a template, the structure of which includes redundancies in the

form of blocks of algorithm, which allow to check the actions of the operator. Hereinafter, we will consider a basic element as a basic element, which is taken as a basis for the development of the entire set of editors used in IT.

Fig. 3. Client software update cycles in the IT lifecycle

Another important internal factor that negatively affects the security is the overload of the client computer hardware platform with tasks, which can dramatically worsen the time parameters of the tasks performed by the client part of IT, or even make it impossible to work, due to depletion of technical resources . To neutralize the effect of this factor in IT, functional redundancy (Fig. 4) was applied in the development of the software, namely the part that is responsible for the implementation of "business logic".

The presence of a functional reserve of "heavy" calculation functions allows to maneuver the computing power of the IT hardware platform, in case of overloading of some of its links, thus increasing the failure-stability. Because the procedure that is functionally redundant (for example, Funk1 in Fig. 6) is developed in two variants by one algorithm, but in different software environments, to be performed in different technical means. This fact can be used to neutralize such a negative factor as the presence of an error in the application software of the client part, in the event that an error occurs in one of the variants of the procedure. This shows a positive multiplicity of the effect of functional redundancy, which increases the overall fault tolerance of information technology.

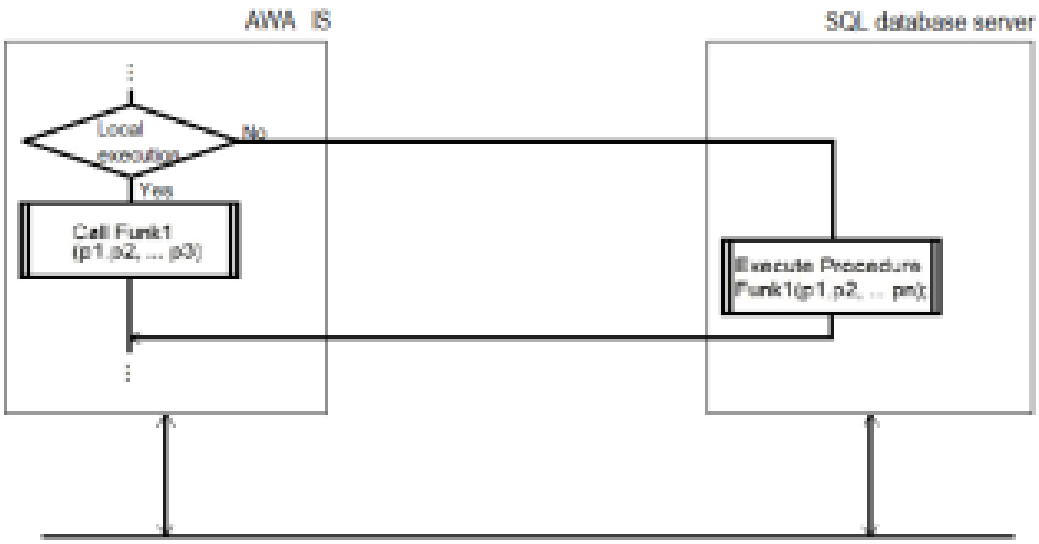


Fig. 4. Model of application of functional reservation of calculated functions of IS in environment "client - server" architecture

Thus, the fault tolerance of the client part is ensured by performing a set of measures, including hardware and functional redundancies: powering client PCs from a separate line with protection devices; use of lightning protection devices in computer network lines; organization of automatic updating of client PC system software; development of algorithms of procedures that implement critical functions of the client part of IT, with the inclusion of a non-trivial (intelligent) error handling unit, which is performed in parallel with the procedure itself; use of non-trivial data editors, which include in their algorithm an interactive procedure that eliminates uncontrolled manipulation of database data by the operator; implementation of critical resources, calculation procedures with the ability to promptly choose the location of their implementation, which does not allow overloading of hardware.

As a result of the analysis of the factors that negatively affect the resilience of the server part of the IT, the model of the action of the negative factors, shown in Fig. 5.

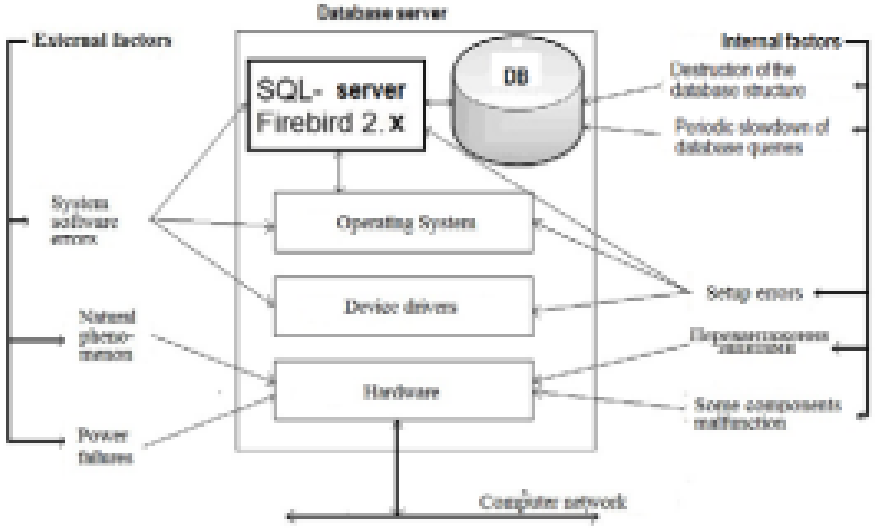


Fig. 5. Model of action of negative factors affecting the fault tolerance of the server of IS

In the presented model, all negative factors are divided into external, caused by causes that are outside the system, and internal. External factors that reduce server resiliency are neutralized in the same way as in the client side of IT. But due to its importance, this is not enough. Due to the fact that the IT server is the location of the database, where all the information processed in the system is concentrated, for him such a factor as failures in the power system is especially dangerous. This is due to the fact that the database, being the most complex system, is sensitive to violations of the technology of its handling. Sudden loss of power or failure of the server hardware due to voltage fluctuations can, with a high probability, damage the database with all subsequent negative consequences for the information. In order to prevent this from happening, a non-interruptible power supply unit with dual voltage conversion and sufficient time for server autonomous operation was introduced into the circuit of the power system. In addition, the uninterruptible power supply must have a status controller that includes an output with a serial RS-232 interface. It is necessary to send the server a signal of decay in the event that due to prolonged loss of external power, the internal power supply of the uninterruptible power supply device will be exhausted to an unacceptable limit. Upon this signal, the server will correctly terminate all applications that have been started without preventing the database from being destroyed.

No less threatening for the server part, which reduce its fault tolerance, are internal factors. Among them, the most severe in terms of consequences is the failure of hardware, namely drives. IT server drives are the most responsible part of it, the failure of which can lead not only to prolonged unavailability of information resources, but also irreversible data loss. This is due to the fact that the IT database is stored on storage devices, which in most cases are mechanical devices and, accordingly, have a smaller resource of operation than electronic circuits. Since the loss of the database is unacceptable, measures are needed to neutralize the threat of sudden loss of storage. This problem can be solved by backing up the drives. Instead of a separate drive for storing the database and other critical data, a RAID array of type 1 drives can be used. In addition, it is necessary to organize periodic diagnostics of drives, which will allow, in most cases, to identify drive problems in advance. To do this, you can use the smartmontools package, which is included in the official repositories of most distributions of the Linux operating system. It has convenient and quite flexible settings for the diagnosis schedule. This allows long-term procedures for diagnosing drives to take out of the working hours of the institution. Each diagnostic report sends to the specified email address or log file.

Another way to prevent database loss is to back up. In spite of the described measures of ensuring the resiliency of the server part of the IT, they still cannot claim to be absolute. Because the database and the entire sequence of software that ensure its operation is a complex system, the presence of errors in its operation remains quite high. To reduce the effects of destructions, such as unknown errors that can lead to database destruction, you can include a database backup subsystem in the server software loop. It operates automatically according to a schedule that records the backup frequency during office and non-working hours of the institution. The database copy repository is another computer located in a location remote from the main server.

Because the copy of the database is a fairly large array of information, so as not to depend on network traffic, the main server and the computer with the database copy repository have their own communication channel (Fig. 8). To prevent uncontrolled changes to the data in the database that could compromise the integrity of the IT data, all changes are performed under transaction management. This approach is guaranteed to ensure the transition of the database from one agreed state to another, when manipulating data.

Therefore, the fault tolerance process is continuous throughout the IT life cycle. It begins with the planning of measures to ensure the resilience of IT, which is designed and lasts until the end of its operation in general. In general, the task of ensuring the fault tolerance of the server part of IT is solved, as well as for the client part, as a set of measures to counteract the negative factors (Fig. 7). It includes: inclusion in the circuit of the subsystem of the power supply of an intelligent uninterruptible power supply unit, which interacts with the operating system of the server, providing automatic correct closing of all server applications, preventing a database crash, sudden power loss and elimination of power supply fluctuations; the use of a high-probability type 1 RAID storage array eliminates the loss of the IT database due to drive failure; automatic diagnostics of the condition of drives according to the established schedule, which allows to quickly identify the causes of future failures; organizing the work of the database backup subsystem in automatic mode, according to the schedule, with the territorial diversity of the main database and its copies, by its own network channel; using the client-side transaction subsystem software, which ensures that any manipulation of data in the database is performed with consistency of data at all times.

Vitality indicators in a complex system: multifunctionality of individual components; the existence of a single (main) purpose of the whole system; not only the possibility of information exchange between individual components, but also information interaction with users; availability of means of protection, control, diagnostics and self-organization. The task of structural survivability analysis requires the definition of: the system architecture required to fulfill the purpose of IT functioning at some point or time when undesirable effects on the system occur; requirements for particular types of system resources and their interconnection; requirements for functionality of system resources; the nature of the nature of the undesirable effects or their consequences. We define a function $f_2(S_i)$, in which $i = 1,2,\dots,n$ the definition of surviv bility in quantitative units in computer networks is expressed as follows:

$$f_2(S_i) = \frac{T_{f_2(S_i),1} + T_{f_2(S_i),2}}{T_{f_2(S_i),1}}, \tag{8}$$

where $T_{f_2(S_i),1}$ - time of operation of the IT process in standard mode, $T_{f_2(S_i),2}$ - time spent on survival processes, $i = 1,2,\dots,n$.

This definition of the survivability function makes it possible to display the standard mode of operation with a unit value, and if there is a need to ensure survivability and in the case of a much longer time than the standard mode of operation, the function value will display a quantitative ordinal value.

From Fig. 6 shows how the problem of increasing the survivability of IT was realized within the framework of the developed system by structural redundancy of its main components, namely its server part. In the event of a failure of the main IT server, its functions can assume a backup, which has exactly the same settings as the main one. The main and backup servers are geographically spaced and fed from different lines. Since the failure of two servers at once is an unlikely event, it ensures high survivability of the server part of IT. Reconfiguration of a real system takes no more than 10 minutes. The copy of the database is kept up to date by the replication service, so the replacement of the main database with the database - the copy is performed without loss of information. But a slight loss of information in such a scheme is still possible. This can happen if some sensitive components of the server hardware platform fail. Typically, these are recent transactions that will be terminated due to hardware failure. And if it is a transaction to change the information in the database, then in this case the information will be lost. But since such an event in the life cycle of the information system itself is rare, such a possible amount of information loss can be neglected. After the server part is restored, the operators whose transactions were lost need to perform the last operations again to recover the lost information. In the regular diagnosis of critical hardware of the server, in most cases it is possible to detect a ripening failure and to replace the corresponding component in a timely manner. Thus, the organization of work can reduce the likelihood of failure of the server-side part of IT and thereby negate the loss of information. As can be seen from Fig. 8, in addition to performing the backup function of the primary server, the backup server serves as a data source for the WEB server through which the IT publishes information to its remote users. In addition, another function is assigned to the backup server - it serves as a repository of database copies maintained by the backup service. Server backup guarantees sufficient IT survivability in general, but does not guarantee that it will lose some of its functions related to the failure of the client computer's hardware components, which critically affect the functioning of the client part as a whole. The solution to the problem, then, is to create some reserve. Analysis of the software of the client part of IT showed that some of them have a reserve of time. Therefore, it is natural to decide to use this reserve at critical moments in the work of the client part of IT. This approach does not allow you to keep a standalone computer as a standby, but also to have stockpiles of components that reduce operating costs without losing the overall viability of the system.

Typically, software modules, as configured, are stored in the IT software repository and on those client computers where they are scheduled to be used at critical times according to the backup plan. If critical computer hardware fails to make it impossible for the client software to perform its functions, it is transferred to a suitable other computer. The time spent on reconfiguring the client side is calculated in minutes, which is an acceptable value to ensure the viability of IT that provides information support in such a subject area, such as financial and economic activities of higher education institutions.

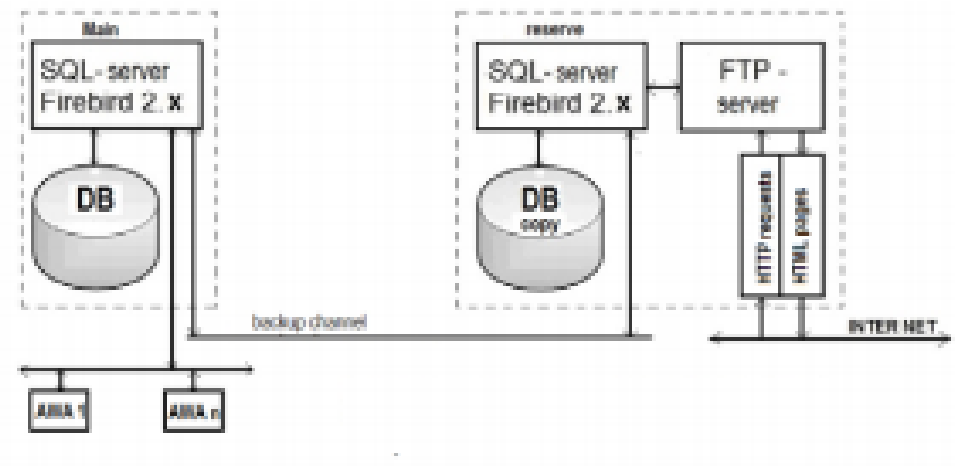


Fig. 6. Server backup scheme IS

This reconfiguration of the client part is made possible by the fact that absolutely no data is stored on the client computers running the software. And the software module itself, for convenience, is grouped into one file and does not require an installation procedure. All you have to do is copy it to another computer and it will be ready to go. This approach allows you to maintain the full functionality of IT even after the failure of several computers, which in itself has a low probability. There is only one limitation - each copy of the client-side software must be pre-registered with IT. Otherwise, an attempt to run such a program will be considered an attempt to gain unauthorized access to the system, even with the correct user credentials. IT's control over all instances of its client parts allows it to block attempts by malicious intruders who have managed to master user account data to gain access to the system. At the same time, the program mastered by the attacker does not gain access to the IT data, and the fact that such program attempts to connect to the systems is recorded in the registry of fatal errors with the corresponding data, which allows to use them to take organizational measures against the attacker.

Thus, IT survivability is ensured by: redundancy of the server-side IT with territorial separation of the main and backup server, the peculiarity of the redundancy is that the server function, at a critical moment, takes over the mirrored SQL server, which in regular mode provides the work FTP servers; redundancy of client part software, a feature of redundancy is that the reserve is not specially dedicated computers, but the performance reserve of individual client computers, which, according to the backup plan, installed the client client software, which is in the critical moment will be used as a regular, preventing loss of IT functionality.

Based on formulas (6) - (8) we obtain the value of efficiency for IT, taking into account the indicators of fault tolerance and survivability:

$$K_e(S_{IT}) = \sum_{j=1}^{m} \sum_{p=0}^{n_p} \sum_{q=0}^{n_q} \left(\alpha_{1,j,p,q} \cdot \frac{T_{f_1(s_i),1}}{T_{f_1(s_i),1} = (T_{f_1(s_i),2} * T_{f_1(s_i),3})} + \alpha_{2,j,p,q} \cdot \frac{T_{f_2(s_i),1} * T_{f_2(s_i),2}}{T_{f_2(s_i),1}} \right), \quad (9)$$

where $\alpha_{1,j,p,q}$ - coefficient for the value that determines the fault tolerance in quantitative units; $\alpha_{2,j,p,q}$ - coefficient for the value that determines the survivability in quantitative units; $\alpha_{1,j,p,q} + \alpha_{2,j,p,q} = 1$.

Similarly, the terms in formula (6) and its specification by formula (9) for the two quantities may be other indicators that characterize the effectiveness of IT.

As a result of the use of these measures was obtained IT narrowly specialized use for various applications, where the accompanying processes are unrealistic or unrealistic time with high parameters of fault tolerance, survivability and overall resilience and, at the same time, acceptable equal financial costs for its operation.

4 Experiments and evaluation

To determine how effective the proposed solutions to ensure fault tolerance and survivability, we will compare the criterion of efficiency for IT without ensuring fault tolerance and survivability and including these characteristics on the basis of formula (9).

The value of the value of the criterion of IT efficiency, which does not meet the requirements of fault tolerance and survivability, we obtain from formula (9) as follows: which constantly monitors the functioning of IT; problem situations are solved only when they are detected. In the first case, the calculation according to formula (9) can be similar and the value of the obtained value is orders of magnitude higher than the value of the criterion for IT, which provides fault tolerance and survivability. How to consider the second option, then $K_e(S_{IT}) = 1$. In this case, the relationship between the values is determined by formula (10) and allows to establish the effectiveness of the proposed solutions to ensure fault tolerance and survivability, as well as to improve efficiency by adjusting the coefficients:

$$\mu = \frac{1}{\sum_{j=1}^{m}\sum_{p=0}^{n_p}\sum_{q=0}^{n_q}\left(\alpha_{1,j,p,q}\cdot\frac{T_{f_1(S_i),1}}{T_{f_1(S_i),1}-(T_{f_1(S_i),2}+T_{f_1(S_i),3})}+\alpha_{2,j,p,q}\cdot\frac{T_{f_2(S_i),1}+T_{f_2(S_i),2}}{T_{f_2(S_i),1}}\right)}, \quad (10)$$

where the absence of specialized IT or external interruptions will mean that the time spent processing them will be zero and the ratio will be equal to one.

If a failure or external interference occurs, the value will be greater than one. The effective value is the value minimally deviated from one.

The results of ensuring the fault tolerance and survivability of specialized IT are shown in the implemented IT in Fig. 7.

For convenience, all lines of the log file fragment have been numbered and critical positions have been highlighted.

Position 19 revealed a fatal error in the operation of the network adapter "eth0" at the time of access of the user's computer with IP 192.168.168.2.

Position 35.36 closes the current session of the SWM user.

At position 37, the system notifies that reconfiguration of network devices is required.

At position 38 it is stated that the device "eth0" is switched off.

Fig. 7. Fragment of the log file of the subsystem of control of work of network devices

At positions 40,41 it is reported that the backup network adapter "eth1" is activated

At position 52 it is stated that a SWM user session has been opened which was terminated due to a failure of the eth0 network adapter.

This snippet (Fig. 8) reflects the operation of the database transaction subsystem during its backup. The backup procedure for the GBAK utility on February 4, 2019, performed a data error that led to a rollback of the transaction. Critical positions are highlighted.

Position 31. Creating a temporary database file.

Position 32-34. GBAK error message when trying to write to the [FK] field of the ORGILCSPISVSR table the default value of NULL.

Item 35. Rollback of the current transaction due to an error.

Item 40. Notification that the backup procedure was completed incorrectly.

Fig. 8. A fragment of the log file of the database backup subsystem

The graphs (Fig. 9) obtained according to the calculations according to the formula (10) for the results of fault tolerance (a), survivability (b) and for the case of combined manifestations and resilience and survivability (c).

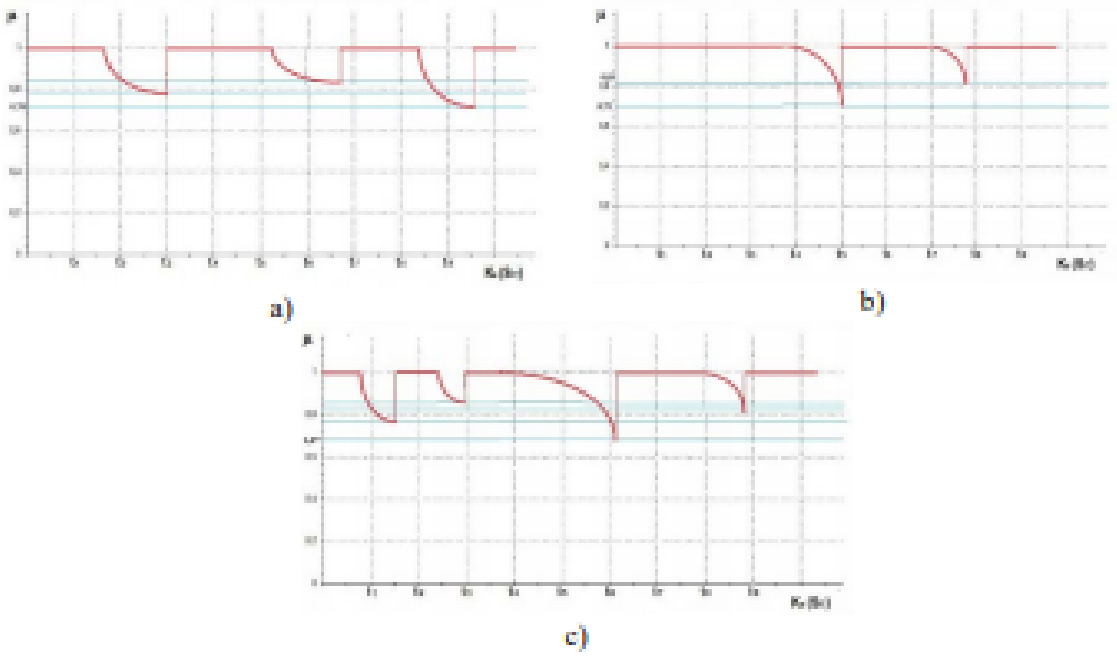


Fig. 9. A schedule for fault tolerance (a); schedule for manifestations of survivability (b); graph of reflection of simultaneous manifestations and fault tolerance and survivability (c)

The results of the study confirm the high level of resiliency and survivability in corporate computer networks, which is more than 75%.

5 Discussion and Future work

An important area of further research to improve the efficiency of IT is to develop a method to ensure effective protection of information directly in the structure of IT and computational processes that take place in computational processes. Their inclusion in the general criterion for determining the effectiveness of IT will balance such values as survivability, resilience and protection of information, expressed in quantitative form, and will become the basis for the development of specialized IT with improved performance.

6 Conclusion

Thus, an approach has been developed to determine the effectiveness of IT based on quantitative values that characterize resilience and survivability, and can be expanded to include other characteristic values. To ensure the resilience and survivability of IT, a system of measures has been developed which resulted in highly specialized IT applications for various applications, where the accompanying processes are unrealistic or unrealistic time with fairly high parameters of resilience, survivability and overall resilience and, at the same time time, acceptable equal to the financial cost of its operation.

Fault tolerance is continuous throughout the IT life cycle and includes: inclusion in the power subsystem of the intelligent uninterruptible power supply that interacts with the server operating system, ensuring automatic correct closure of all server relationships, preventing database crashes, sudden power failure and exclusion of fluctuations in supply voltage; the use of RAID array of type 1 drives, which with a high probability, eliminates the loss of the IT database due to the failure of the drive; automatic diagnostics of the state of drives according to the established schedule that allows to reveal the reasons of future failures quickly; organization of the database reservation subsystem in automatic mode, according to the schedule, with territorial distribution of the main database and its copies, on its own network channel; using the transaction subsystem of the client software, which ensures that any manipulation of the data in the database is performed with data consistency at all times.

The survivability of IT is ensured by the following methods: redundancy of the server part of IT with the territorial separation of the main and backup server, the peculiarity of redundancy is that the server function, at a critical moment, takes over the mirror SQL server, which provides FTP -servers; redundancy of client software, a feature of redundancy is that the reserve is not a dedicated computer, and the performance reserve of individual client computers, which, according to the redundancy plan, is installed software of the client client, which is the critical moment will be used as a regular, preventing the loss of IT functionality.

References

1. DSTU 3396.2-97 Protection of information. Technical protection of information. Terms and definitions. State Committee of Ukraine, Kyiv (1997) [in Ukrainian]
2. Savelyeva, O. S., Krasnozhon, O. M., Lebedeva, O. U. (2014). Using the structural fault-tolerance index in project designing. Odes'kyi Politechnichnyi Universytet. Pratsi, 2, 130–135. doi: 10.15276/opu.2.44.2014.24.
3. S. Boranbayev, S. Altayev, A. Boranbayev. Applying the method of diverse redundancy in cloud based systems for increasing reliability, in Proceedings of the 12th International Conference on Information Technology: New Generations (ITNG 2015) (Las Vegas, Nevada, 2015), pp. 796–799.
4. Boranbayev A., Boranbayev S., Yersakhanov K., Nurusheva A., Taberkhan R. (2018) Methods of Ensuring the Reliability and Fault Tolerance of Information Systems. In: Latifi S. (eds) Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 738. Springer, Cham.
5. Chinnaiah, M., Niranjan, N. Fault tolerant software systems using software configurations for cloud computing. *J Cloud Comp* 7, 3 (2018). https://doi.org/10.1186/s13677-018-0104-9.
6. Zhu X, Wang J, Guo H, Zhu D, Yang LT, Liu L (2016) Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds. IEEE Trans Parallel Distrib Syst 27(12):3501–3517. https://doi.org/10.1109/TPDS.2016.2543731.
7. Liu J, Zhou J, Buyya R (2015) Software rejuvenation based fault tolerance scheme for cloud applications In: 2015 IEEE 8th International Conference on Cloud Computing, 1115–1118, New York. https://doi.org/10.1109/CLOUD.2015.164.

8. Liu J, Wang S, Zhou A, Kumar SAP, Yang F, Buyya R (2016) Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability. IEEE Trans Cloud Comput PP(99):1–1. http://dx.doi.org/10.1109/TCC.2016.2567392.
9. Nicolo P (2013) A frame work for self-healing software systems In: IEEE 35th International Conference on Software Engineering (ICSE), 1397–1400. https://doi.org/10.1109/ICSE.2013.6606726.
10. Zhao W, Wenbing Z, Melliar-Smith PM, Moser LE (2010) Fault Tolerance Middleware for Cloud Computing In: 2010 IEEE 3rd International Conference on Cloud Computing, 67–74, Miami. https://doi.org/10.1109/CLOUD.2010.26.
11. Bala A, Chana I (2012) Fault tolerance- challenges, techniques and implementation in cloud computing, ISSN (Online): 16940814. IJCSI Int J Comput Sci 9(1). www.IJCSI.org.
12. Egwutuoha IP, Chen S, Levy D, Selic B (2012) A fault tolerance framework for high performance computing in cloud, Cluster, Cloud and Grid Computing (CCGrid) In: Proceedings of the 12th IEEE/ACM international symposium. 13-16 May, 709–710. https://doi.org/10.1109/CCGrid.2012.80.
13. S. Pitcher, "New DoD Approaches on the Cyber Survivability of Weapon Systems (25 March 2019)," 25 March 2019. [Online]. Available: https://www.itea.org/wp-content/uploads/2019/03/Pitcher-Steve.pdf.
14. D. J. Bodeau, R. D. Graubart, R. M. McQuaid and J. Woodill, "Cyber Resiliency Metrics and Scoring in Practice: Use Case Methodology and Examples (MTR 180449)," The MITRE Corporation, Bedford, MA, 2018.
15. D. Fitzpatrick, D. Bodeau, R. Graubart, R. McQuaid, C. Olin and J. Woodill, "(DRAFT) Cyber Resiliency Evaluation Framework for Weapon Systems: Foundational Principles and Their Potential Effects on Adversaries," The MITRE Corporation, Bedford, MA, 2019.
16. NIST, "Initial Public Draft of NIST SSP 800-160 Volume 2, Systems Security Engineering: Cyber Resiliency Considerations for the Engineering of Trustworthy Secure Systems," 21 March 2018. [Online]. Available: https://csrc.nist.gov/CSRC/media/Publications/sp/800-160/vol-2/draft/documents/sp800-160-vol2-draft.pdf.
17. Lysenko, S., Savenko, O., Kryshchuk, A., Kljots, Y. Botnet detection technique for corporate area network. In: Proc. of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems, pp. 363-368 (2013).
18. Savenko, O., Lysenko, S., Nicheporuk, A., Savenko, B.: Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search. CEUR Workshop, Vol. 1844, pp. 555–569 (2017).
19. Savenko, O., Lysenko, S., Nicheporuk, A., Savenko, B.: Approach for the Unknown Metamorphic Virus Detection. In: 9-th IEEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems. Technology and Applications, pp. 453–458 (2017).
20. Pomorova, O., Savenko, O., Lysenko, S., Kryshchuk, A. and Nicheporuk, A.: A technique for detection of bots which are using polymorphic code. In: 21st International Conference, CN, Springer, Brunów, Poland, pp. 265-276 (2014)
21. Kondratenko, Y., Kondratenko, N.: Soft Computing Analytic Models for Increasing Efficiency of Fuzzy Information Processing in Decision Support Systems. Chapter in book: Decision Making: Processes, Behavioral Influences and Role in Business Management, R. Hudson (Ed.), Nova Science Publishers, New York, 41-78 (2015)
22. Balyk, A., Karpinski, M., Naglik, A., Shangytbayeva, G., Romanets, I.: Using graphic network simulator 3 for DDoS attacks simulation. International Journal of Computing. Vol. 16, Issue 4, pp. 219-225 (2017).

3. Nicheporuk A., Paiuk V., Savenko B., Savenko O., Geidarova O. Detecting Software Malicious Implant Based on Anomalies Research on Local Area Networks. CEUR-WS. 2020. Vol. 2623. P. 194-207.

Detecting Software Malicious Implant Based on Anomalies Research on Local Area Networks

Andrii Nicheporuk[0000-0002-7230-9475], Vadym Paiuk[0000-0002-7253-893X], Bohdan Savenko[0000-0002-9969-8239], Oleg Savenko[0000-0002-4104-745X] and Olena Geidarova[0000-0002-7253-893X]

Khmelnytsky National University, Khmelnytsky, Ukraine
andrey.nicheporuk@gmail.com, vadympaiuk@gmail.com, savenko_bohdan@ukr.net, savenko_oleg_st@ukr.net, geydarova@ukr.net

Abstract. The paper analyzes malicious software implants that use undocumented software features on local area networks. They can cause significant harm both users of personal computers and enterprises that utilize computer networks and use specialized software. In order to detect this type of malware, its possible models and behavioral scenarios, features, stages of research in local area networks have been proposed. Based on this data, a method for detecting computer anomalies has been developed, which is part of a general process for detecting malicious software implants that use undocumented software features. The result of the method is a division of computers on a local network into classes in purpose for further investigation of behavioral patterns. Thus, groups of computer are highlighted in which similar profiles have been formed, that in the overall scheme allows to improve the accuracy of detection. The adopting of the developed models of software implants that use undocumented software features, as well as a method for detecting computer anomalies, have allowed to carry out experimental researches with the use of distributed detection system. The results of the experiments have shown the correctness of the proposed detection enhancement solutions.

Keywords: Software Malicious Implant, Local Network, Computer, Malware, Czekanowski Diagram.

1 Introduction

According to research [1] of IDC firm and the University of Singapore, malware-related security breaches cause users worldwide damage of at least on $ 500 billion annually. Moreover, the number of malicious software is growing every year [2]. The most relevant for the benefit of attackers are organizations and enterprises that operate information technology on local computer networks. There are many ways to gain entry into the local computer networks of businesses (organizations) for the purpose of unauthorized access to information in them. One way for attackers to access enterprises (organizations) information resources is to use undocumented capabilities in

the software and hardware of computers and peripherals that allows unauthorized access to system resources, typically, via a local network. This is achieved through the using of software implants, which primary purpose is to provide unauthorized access to sensitive information.

Software implant is a secretly implemented program which poses a threat to the information contained on the computer [3]. The software implant can be implemented as a separate malware, or as undocumented software code in the software.

We will consider the software implants that use undocumented software features on the local computer networks of enterprises (organizations) as an object of research. The difficulty of detecting such a secretly functioning software object, which under certain conditions is capable of providing unauthorized access, is due to the absence of its activity during a long time. As a rule, such software implants allow to keep software features and are implemented by some else of the features included in the software package.

An enterprise may use ready-made software that already has software implants, or software made to order, which has been poorly verified upon commissioning. Software that runs inside enterprise LANs is typically a distributed, which makes software implants are active on all computers on the network. This increases the threat to businesses and organizations. Software implants can be used to create botnets [4], implement trojans, or produce metamorphic or polymorphic components [5], and so on. Therefore, the scientific problem of detecting software implants on the local area networks is relevant.

One of the primary tasks that need to be addressed is to develop appropriate methods for creating effective components of a comprehensive system for detecting software implants on local area networks.

2 Related works

Software implants detection studies are presented in many ways that depend on the considering of specific malware types. The main problem that accompanies the process of software implants detection is the discrepancy between what the user sees and what actually happens [1]. To achieve these results, the attackers have developed quite a few tools and approaches.

The most common research in this area is Backdoor malware detection studies [6-12]. In [6] authors proposed a new approach to detecting and removing Backdoor malware using neural networks. The experimental results obtained are based on the classification made. The work focuses on such type of attack that allows attackers to insert a hidden function or hidden program code into a malicious action model. Detection of such types of malware is difficult task, because unexpected behavior occurs only when there is a launch to execute a hidden function or program code known only to the attacker. The adequacy of the proposed solution requires more convincing evidence since a small set of validated and reliable datasets was used [6].

In [8] presents a model used by intruders to hide the invasion path. One of these techniques is done by using multiple hosts on the network, which can be detected

using the approach suggested by the authors. Authors explored the opportunity the use of this approach to detect others type of malware, including Backdoor. The study shows that the proposed approach can produce a very low rate of false negative and false positive and allow to reduce the detection time of the scanning process.

In [9] it is analyzed and substantiated that, due to the complexity of the studied topic, there are few effective solutions. One way analyzed by the authors is to encode code fragments using specially designed interrupts that, when triggered, manipulate the run time state and, under certain conditions, can perform arbitrary computations without fail.

In [10] proposes to solve the problem of formalizing a terminal machine by modeling a program or system using a so-called machine with a designated end state that allows one to treat the software fragment as an emulator. Also authors have developed the concept of a multi-step game where an attacker and a defender get to take turns interacting that allows thinking about it as the system with states and transitions between them.

In works [11, 12] analyze the complexity of the problem with using many tools. This could jeopardize the platform by other means. For example, this is may be a hardware component, a custom program or a piece of malware. And this is a prerequisite for development methods for detecting them. A significant obstacle to the effectiveness of this approach is the lack of test samples.

Another equally malicious tool is the implementation of the software and the launch of the secret exchange feature. The features of such a hidden function are presented in [13]. To address this in [14] developed five steps to identify a hidden function in software.

In [15], models of hidden schemes of function exchange are considered and analyzed. They take into account the availability of the secure protocol of distributed information transmission. The reliability of such a protocol under the conditions of existence of hidden functions is investigated.

Another type of malicious software which can introduce software implants is a botnets [16, 17]. In [16] a technique for botnet detection based on a DNS-traffic is presented. Botnets detection based on the property of bots group activity in the DNS-traffic, which appears in a small period of time in the group DNS-queries of hosts during trying to access the C&C-servers, migrations, running commands or downloading the updates of the malware. The method takes into account abnormal behaviors of the hosts' group, which are similar to botnets: hosts' group does not honor DNS TTL, carry out the DNS-queries to non-local DNS-servers. Method monitors large number of empty DNS-responses with NXDOMAIN error code.

In [17] a DNS-based anti-evasion technique for botnets detection in the corporate area networks is proposed. Authors have combine of the passive DNS monitoring and active DNS probing and have construct BotGRABBER detection system for botnets in corporate area network, which uses such evasion techniques as cycling of IP mapping, "domain flux", "fast flux", DNS-tunneling. BotGRABBER system is based on a cluster analysis of the features obtained from the payload of DNS-messages and uses active probing analysis.

In [18-26] it is shown how the use of metamorphic transformations makes it possible to hide program codes of functions. Along with polymorphic technologies [27], these methods are quite effective and widely used by attackers.

The works in [28-30] analyzed the use of known mathematical methods for processing events related to software operation. The discussed methods can be used to detect software implants that use undocumented software features, but only after the process of processing big data obtained during monitoring has been completed.

Thus, software implants that use undocumented software features [31-36] pose a problem for computer users, especially when they can be distributed on local computer networks. Known detection methods target to specific subclasses or typical classes of malware and are not sufficiently represented in related works. The various mathematical methods used for detection process, means require the initial stages of preparation of the data for processing, which aims to design a comprehensive approach of detection.

3 A method for detecting anomalies in the computer systems based on the search for deviations from the mean values of the behavior profiles

In order to detect software implants that use undocumented software features, let's build profiles of computer systems (CS) based on behavior of software in which it is executed. Because the client side of the software is the same, they should have similar behavioral profiles. System profiles can be clustered in the local area networks (LANs) into CS groups that use typical parts of specialized software. Such profiles can range from one to 5-10, because specialized software has a narrow focus and, therefore, cannot be sprayed for various purposes. We define these profiles as formalized models of software behavior in CS. After some time of functioning of the specialized software and the available software in the CS the statistics is collected, that is use for the refinement of profiles. Developing a formal representation of behavior profiles is based on a numerical expression of the features. After receiving profiles, the system functions and, on a daily start, analyzes the similarity of profiles and results of the functioning of the software in the CS. The presence of clustering profiles allows to more accurately identifying possible anomalies in the CS that belong to certain groups.

To detect software implants that use undocumented software features in LANs, it is important to create a set of their behavioral signatures. In order to form a database of behavioral signatures, appropriate models of software implants were developed based on scenarios of their functioning. The implementation of software implants that use undocumented software features at different stages of the software life cycle can occur by the following scenarios:

1. Work of attackers inside software development team.
2. Creation of dynamically formed commands or parallel computing processes.

3. Implementation of command forwarding and recording malicious information in the memory of information system.
4. Inserting software implants that use undocumented software features into the code.
5. Creating a masked trigger mechanism for software implants that use undocumented software features.
6. Inserting software implants that use undocumented software features into separate routines and into the management program.
7. Preparation of test data for the detection of software implants that use undocumented software features.
8. Hiding software implants that use undocumented software features by making bugs in the software;
9. Placing software implants that use undocumented software features in the branches of the software that are not verified under testing.
10. The participation of the attackers in the software verification;
11. Development of software implants that use undocumented software features during software revision.
12. Development of updates and additions for software.

Scenarios of introducing of software implants that use undocumented software features are directly dependent and affect on their structure, so they may be separate entities or parts of another entity. Let's present all scenarios as graphs and formalize them with respect to the structure of the software implants that use undocumented software features, as well as all possible combinations of them.

As an example, on the fig. 1 has shown a graph of an irregular Markov process.

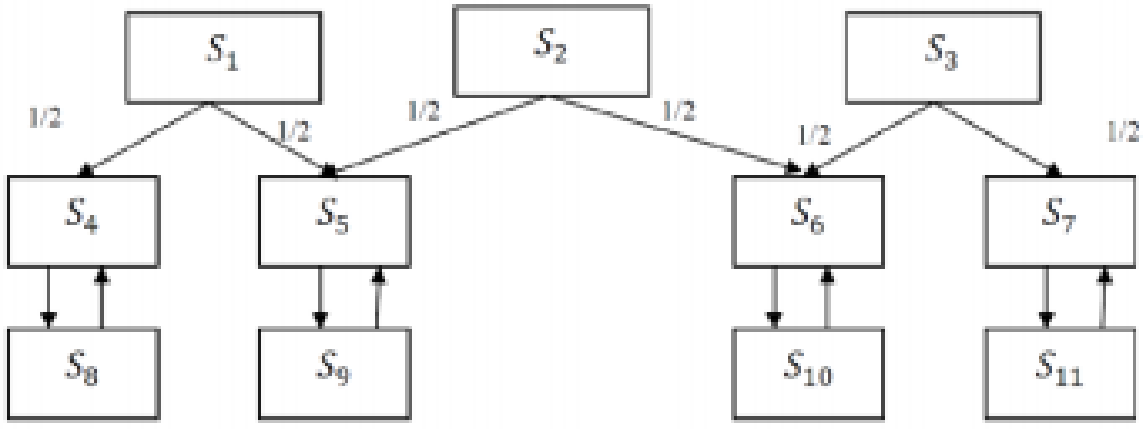


Fig. 1. Graph of irregular Markov process

Here, S_1 - software implants that use undocumented software features in a separate file, S_2 - software implants that use undocumented software features in a working file, S_3 - software implants that use undocumented software features hidden partially in two or more files. Suppose with equal probability $p_i = \frac{1}{2}$ information on S_1, S_2, S_3

arrives in pairs on states S_4, S_5, S_6, S_7. The states S_8, S_9, S_{10}, S_{11} will be counted. For example, if more information is obtained from states S_{10}, S_{11}, then it will indicate that software implants that use undocumented software features are placed in several working files of the program.

We detect software implants that use undocumented software features by manifestations which are based on file analysis and network activity. These manifestations depend on the models of functioning and structure incorporated in them. These features are: presence of software modules that do not meet the purpose of the process; presence of operating system objects that are open by the process but do not conform to the purpose of the process; high intensity of input/output operations for a certain process; a high CPU or internal memory usage from a certain process; the similarity of the file name to the system file name; the operating system process executable file is not in the common directory; the system process is run on behalf of the local user; code enforcement in the data area, which is enabled for all processes, is disabled for the certain process; the system process has a directory other than what it must be for that process; the absence of digital signature in the executable; high network activity of the process, which must run locally, etc. However, to improve detection efficiency, it is need tools that allow to establish the fact of the threat without the intervention of a network administrator who may not be able to process certain attributes for various reasons. Software implants that use undocumented software features may use masking tools on the system, making it difficult to detect them. On the technical side software implants use programming methods that are not common when creating standard software, so these features can also be additional attribute to identify them. In particular, for a Windows executable, these features can be: an additional section at the end of the file; an entry point indicates a transition to the middle of a section that is not a code section; the entry point correspond to a jump command that specifies a jump for the code section; the presence of features that indicates a code section, which is not a code section. Similarly for other environments on the CS, information from which may be related with RAM.

Let's describe the models of software implants that use undocumented software features on LANs:

1. software implants are injected in the software, stores all or selected pieces of software, entered or displayed in the hidden area of local or remote external direct access memory; the object of storage may be keyboard inputs, documents that will be printed; this model requires external storage, which must be organized in such a way as to ensure that it is stored for a specified period of time and can be further removed and hidden from other users or processes;
2. software implants are embedded in network or telecommunication software; As a rule, this software is always active, so software implants control the processing of information on the computer, performing installation and deletion other implants, as well as removing the accumulated information; software implants that use undocumented software features can trigger events for previously implemented implants;

3. software implants that use undocumented software features transmit information stored by an attacker (such as keyboard input) into a communication channel, or store it without relying on the guaranteed possibility of further reception or removal; software implants may also initiate continuous access to information, leading to an increase in signal-to-noise ratio when intercepting side radiation;
4. software implants that use undocumented software features distort data streams that occur during applications running (source streams), or distort input streams of information, or initiate (or suppress) errors that occur when running applications;
5. software implants which do not produce direct affect on software. The main purpose is to maximize the resulting "residual" information for further study; the attacker either obtains these fragments using the implants of the previous models, or directly accesses the computer in the guise of repair or diagnosis.

To detect software implants that use undocumented software features it is need to detect anomalies in a particular CS based on searching of deviations from the mean values of the behavior profiles. Thus, it is need to develop a detection method based on a combination of anomaly detection technologies and behavioral signature matching.

The implementation of the detection of software implants that use undocumented software features is based on the further development of information technology, which will include profile models, models of software implants and method of anomaly detection and theirs applying on LANs to investigate specialized software.

To determine the software profile in the CS, we use the system call monitor [42]. First, let's form API call sequences for each of the processes over a long time and build a software profile in the CS. After profile creation their clusterization is done. And the last step, if it is possible to divide profiles by more than one class, analysis of the obtained classes of CS is conducted.

For the study we use the anomaly search scheme. In order to reduce the number of input, grouping of similar values was done. The resulting profile scheme is a partial case of multidimensional analysis, where multiple objects are considered on many grounds. When processing statistic data in multivariate analysis [4, 5, 6], taxonomic methods have been used that do not require expert evaluation but are based only on observation results. The input for the study is the observation matrix:

$$X = \begin{bmatrix} x_{11} & x_{21} & \dots & x_{1k} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2k} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{i1} & x_{i2} & \dots & x_{ik} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{s1} & x_{s1} & \dots & x_{sk} & \dots & x_{sn} \end{bmatrix} \quad (1)$$

where n is the number of features observed on the objects; s is number of objects; x_{ik} – the number of manifestations of k-th feature in i-th object during the observation period. The features normalization is carried out according to:

$$V_{ik} = \frac{x_{ik}}{\sum_{i=1}^{s} x_{ik}} \tag{2}$$

After applied formula 2 the matrix V was created:

$$V = \begin{bmatrix} V_{11} & V_{21} & \dots & V_{1k} & \dots & V_{1n} \\ V_{21} & V_{22} & \dots & V_{2k} & \dots & V_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ V_{i1} & V_{i2} & \dots & V_{ik} & \dots & V_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ V_{s1} & V_{s1} & \dots & V_{sk} & \dots & V_{sn} \end{bmatrix} \tag{3}$$

Then the isotonic coefficient that shows the position of the S-th object on the set of all features was used:

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \dots \\ W_i \\ \dots \\ W_S \end{bmatrix} \tag{4}$$

where W_i defined as:

$$W_i = \sum_{i=1}^{n} V_{ik} \tag{5}$$

According to matrix (4), it is possible to arrange the objects by isotonic metric, that is, by the rank that characterizes the object by the set of features. Next stage involves structural (isomorphic) ordering of objects. To do this, with the matrix X, the matrix Z is formed by using formulas (4) and (5):

$$Z_{ik} = \frac{x_{ik}}{\sum_{i=1}^{s} x_{ik}} / \sum_{i=1}^{n} \frac{x_{ik}}{\sum_{i=1}^{s} x_{ik}} \tag{6}$$

$$Z = \begin{bmatrix} Z_{11} & Z_{21} & \dots & Z_{1k} & \dots & Z_{1n} \\ Z_{21} & Z_{22} & \dots & Z_{2k} & \dots & Z_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ Z_{i1} & Z_{i2} & \dots & Z_{ik} & \dots & Z_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ Z_{s1} & Z_{s1} & \dots & Z_{sk} & \dots & Z_{sn} \end{bmatrix} \tag{7}$$

The distance matrix in D is then constructed using formulas (4) and (5):

$$d_{im} = \left(\frac{1}{N} \sum_{k=1}^{n} \left(Z_{ik} - Z_{mk} \right)^2 \right)^{\frac{1}{2}} \tag{8}$$

Sometimes the average absolute difference of features values is used, which is determined by the formula:

$$d_{im} = \frac{1}{N} \sum_{k=1}^{n} \left| Z_{ik} - Z_{mk} \right| \tag{9}$$

$$D = \begin{bmatrix} 0 & d_{12} & \dots & d_{1k} & \dots & d_{1s} \\ d_{21} & 0 & \dots & d_{2k} & \dots & d_{2s} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ d_{i1} & d_{i2} & \dots & 0 & \dots & d_{is} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ d_{s1} & d_{s2} & \dots & d_{sk} & \dots & 0_{sn} \end{bmatrix} \tag{10}$$

Thus, the square matrix $S \times S$ is obtained, where the number of rows and the number of columns is equal to the number of objects. In this matrix it is possible to distinguish subsets for homogeneity. The easiest way to do this is to apply the Czekanowski method [5, 6, 7, 8].

The numerical values in the distance matrix are replaced by labels, for example:

× – distance $B < 0.4$

○ – distance $0.4 < B < 0.8$

• – distance $B > 0.8$

As a result a disorderly diagram of Czekanowski will be obtained. To get an ordered diagram, the rows and columns are rearranged so that they are as close to the main diagonal as possible. For example, the ordered Czekanowski diagram is given in table. 1.

Table 1. Ordered Czekanowski diagram

	1	2	3	4	5	6	7	8
1	×	×	×	×	○	○	○	•
2	×	×	×	×	○	○	•	•
3	×	×	×	×	○	○	•	•
4	×	×	×	×	•	•	○	•
5	○	○	○	•	•	•	○	○
6	○	○	○	•	•	×	•	•
7	•	•	•	○	○	•	×	•
8	•	•	•	•	○	•	•	×

The diagram shows that the objects 1, 2, 6, 7 are grouped together and have the smallest deviation (with all group) from the main diagonal. In our case, these may be computers that have specialized software installed. Objects 5 and 3 are form own two subsets that are far from the main diagonal. At the same time the outermost objects belong to numbers 4 and 8.

Thus, it is possible to divide software profiles in the CS into classes and, further, conduct analyzes the deviations in the classes with purpose of anomalies searching.

4 Experiments and evaluation

In order to conduct series of experiments a distributed system [43] for detecting malware was used. Software implants that use undocumented software features were developed as part of each of the typical botnets. The purpose of the experiments was verification of the botnets detection method, the effieiency of the classifier in the structure of the distributed system and determination of the dependence of the percentage of detected nodes of the botnets, which had contained software implants. To perform the experiments, 28 botnets and codes of known detected botnets were constructed [44]. All generated botnets were grouped by classes. From generated botnets 25 structural elements in three stages of operation and 81 functions were allocated.

The experiment was conducted for the classifier without adding instances of the created botnets and with them, that is, the check was performed without training the classifier on the created samples and with the preliminary classification of the samples by classes. The second variant is necessary to check the accuracy of classifying the same samples from which this class is built. This is important because during monitoring API functions may occur errors. The monitoring time of the CS in LAN was 350 hours for each instance of the botnet of each of the two classifiers. It is should be noted that the functionality of the botnet nodes was simplified and did not include the attack option. The botnet nodes only worked in control and support modes of own functioning. Thus, for a distributed system the objects of research were running processes on CS. In order to conduct the experiment botnets that use the strategy of obtaining full control by activating their components were selected. That is software implants that use undocumented software features were presents on each CS. In order to obtain software profile in the CS we perform API monitoring call. Based on the obtained profiles the features vectors are formed. After feature vector creation their clusterization is done. The results of calculation different metrics are presented in table. 2.

The experiments involved determining the following metrics for the detection of bot nodes:

$P_{1,1}$ – the percentage of vectors of malicious actions belonging to certain class relative to all test samples that the system assigned to this class with previous training;

$P_{1,2}$ – similar to metric $P_{1,1}$, but without previous training;

$P_{2,1}$ – the percentage of malicious action vectors belonging to a given subclass of a class relative to all test vectors that the system assigned to that subclass of the class in

the test sample (those that were correctly assigned to the subclasses) with previous training;

$P_{2,2}$ – similar to metric $P_{2,1}$, but without previous training;

$P_{3,1}$ – the percentage of correctly detected botnet nodes with previous training

$P_{3,2}$ – similar to metric $P_{3,1}$, but without previous training;

$P_{4,1}$ – the percentage of incorrectly classified botnet nodes as benign applications (type I error) with previous training;

$P_{4,2}$ – similar to metric $P_{4,1}$, but without previous training;

$P_{5,1}$ – the percentage of incorrectly assigned bot nodes to one of the botnet classes (type III error), with previous training;

$P_{5,2}$ – similar to metric $P_{5,1}$, but without previous training.

The results of evaluating the efficiency of detection the software of botnets' nodes based on the work of two classifiers for the entered classes and subclasses in the classifier are shown in table 2.

Table 2. Results of experiments

Metrics	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Mean
$P_{1,1}$, %	90,74	84,29	73,66	86,30	94,04	94,18	96,60	89,44
$P_{1,2}$, %	75,93	63,57	60,22	70,32	68,77	67,60	69,36	67,71
$P_{2,1}$, %	85,80	83,57	72,58	85,39	98,88	93,92	96,60	88,42
$P_{2,2}$, %	74,69	63,57	59,14	70,32	67,37	66,58	67,66	66,80
$P_{3,1}$, %	92,11	84,21	71,93	89,47	90,53	88,42	93,68	87,72
$P_{3,2}$, %	76,32	57,89	63,16	64,91	71,58	54,74	75,79	65,89
$P_{4,1}$,%	7,89	14,47	28,07	10,53	7,37	11,58	6,32	11,70
$P_{4,2}$,%	21,05	40,79	36,84	31,58	24,21	44,21	22,11	31,97
$P_{5,1}$,%	0	1,32	0	0	2,11	0	0	0,49
$P_{5,2}$,%	2,63	1,32	0	3,51	4,21	1,05	2,11	2,14

As a result of the experiment correctly clasterized 66% of test samples for the classifier without the entered vectors of artificially generated botnets and 88% for the classifier to which the vectors were previously added by performing its training. The percentage of features that the distributed system used to detect botnets and have related to manifestations of software implants, is approximately 27% of the overall detected. The intensity of manifestations of software implants is significantly lower than typical manifestations of botnets. Thus, software implants that use undocumented software features within botnets can be detected by distributed systems [43] and the direction of such research is promising.

5 Discussion and Future work

Software implants that use undocumented software features used on local area networks can be developed and used in various malicious models. Important task is further developing formal profiles and its behavioral signatures. The combination of these components will allow you to get metrics to investigate the presence of this type of malware.

6 Conclusion

Software implants that use undocumented software features used on local area networks can cause significant harm to users of personal computers, especially to businesses that use computer networks and use specialized software.

The obtained class divisions according to the developed solution will allow to perform further analysis of deviations for anomalies search in the classes. The use of developed models of Software implants that use undocumented software features in distributed detection systems [43] has made it possible to improve the detection efficiency of the botnets they were part of.

The direction of further research is the specification and definition of the many functions that will form elements of Software implants that use undocumented software features, with the aim of representation of theirs behavioral signatures to improve detection efficiency.

References

1. Sanjam, G., Gentr, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Hiding Secrets in Software: A Cryptographic Approach to Program Obfuscation. Communications of the ACM, Vol. 59, No. 5, pp. 113-120 (2016).
2. McAfee Mobile Threat Report Q1, 2019. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf
3. DSTU 3396.2-97 Protection of information. Technical protection of information. Terms and definitions. State Committee of Ukraine, Kyiv (1997) [in Ukrainian]
4. Lysenko, S., Savenko, O., Kryshchuk, A., Kljots, Y. Botnet detection technique for corporate area network. In: Proc. of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems, pp. 363-368 (2013).
5. Savenko, O., Lysenko, S., Nicheporuk, A., Savenko, B.: Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search. CEUR Workshop, Vol. 1844, pp. 555–569 (2017).
6. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B. et al: Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. CEUR Workshop, Vol. 2301, (2019)
7. Adups Backdoor. Available: https://www.kryptowire.com/adups_security_analysis.html.

8. Alminshid K., Omar, M. N.: Detecting backdoor using stepping stone detection approach. In: Proc. of 2013 Second International Conference on Informatics & Applications (ICIA), Lodz, Poland, pp. 87-92 (2013)
9. Zaddach, J., Kurmus, A., Balzarotti, D., Blass, E.-O., Francillon, A., et al.: Implementation and Implications of a Stealth Hard-drive Backdoor. In. Proc. of 29th Annual Computer Security Applications Conference, New Orleans, Louisiana, US (2013).
10. Dullien, T. F.: Weird machines, exploitability, and provable unexploitability. IEEE Transactions on Emerging Topics in Computing, No. 99, pp. 1-15 (2017)
11. Thomas, S. L., Chothia, T., Garcia, F. D.: HumIDIFy: A Tool for Hidden Functionality Detection in Firmware. In: Proc. of 14th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Bonn, Germany, pp. 279-300 (2017).
12. Thomas, S. L., Chothia, T., Garcia, F. D.: Measuring the Importance of Static Data Comparisons to Detect Backdoors and Undocumented Functionality. In: Proc. of 22nd European Symposium on Research in Computer Security. Oslo, Norway, pp. 513-531 (2017).
13. Schönegge A.: The Hidden Function Question Revisited. In: Proc. of Algebraic Methodology and Software Technology: 6th International Conference, AMAST '97. Sydney, Australia, pp. 451-464 (1997).
14. The Secret Code of Software Validation….In 5 Easy Steps. Available: https://www.cebos.com/blog/the-secret-code-of-software-validation-in-5-easy-steps/
15. Kawamoto, Y., Yamamoto, H.: Secret function sharing schemes and their applications to the oblivious transfer. In: IEEE International Symposium on Information Theory, pp. 281-295, Yokohama, Japan (2003).
16. Pomorova, O., Savenko, O., Lysenko, S., Kryshchuk, A., Bobrovnikova, K.: A Technique for the Botnet Detection Based on DNS-Traffic Analysis. Communications in Computer and Information Science, Vol. 522, pp.127-138 (2015).
17. Pomorova, O., Savenko, O., Lysenko, S., Kryshchuk, A., Bobrovnikova, K.: Anti-evasion Technique for the Botnets Detection Based on the Passive DNS Monitoring and Active DNS Probing. Communications in Computer and Information Science, Vol. 608, pp.83-95 (2016).
18. Savenko, O., Lysenko, S., Nicheporuk, A., Savenko, B.: Approach for the Unknown Metamorphic Virus Detection. In: 9-th IEEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems. Technology and Applications, pp. 453–458 (2017).
19. Desai, P., Stamp, M.: A highly metamorphic virus generator. Int. J. Multimedia Intelligence and Security, Vol. 1(4), pp. 402-427 (2010)
20. Podlovchenko, R.I., Kuzyurin, N.N., Shcherbina V.S., Zakharov V.A.: Using algebraic models of programs for detecting metamorphic malwares. Journal of Mathematical Sciences, Vol. 172 (5), pp. 740-750 (2011)
21. Anderson, B., Quist, D., Neil, J., Storlie C., Lane, T.: Graph-based malware detection using dynamic analysis. Journal in Computer Virology, 7, pp. 247-258 (2011)
22. Runwal, N., Low, R.M., Stamp, M.: Opcode Graph Similarity and Metamorphic Detection. Journal in Computer Virology, 8, pp. 37-52 (2012)
23. Nagaraju, A.: Metamorphic malware detection using base malware identification approach. Journal Security and Communication Networks, 7, pp. 1719-1733 (2014)
24. Patel, M.: Similarity tests for metamorphic virus detection. Master's thesis, San Jose State University (2011)
25. Wong, W.: Analysis and Detection of Metamorphic Computer Viruses. Master's thesis, San Jose State University (2006)

26. Kuriakose, J., Vinod, P.: Unknown Metamorphic Malware Detection: Modelling with Fewer Relevant Features and Robust Feature Selection Techniques, IAENG International Journal of Computer Science, Vol. 42(2), p139-151 (2015)
27. Pomorova, O., Savenko, O., Lysenko, S., Kryshchuk, A. and Nicheporuk, A.: A technique for detection of bots which are using polymorphic code. In: 21st International Conference, CN, Springer, Brunów, Poland, pp. 265-276 (2014)
28. Tarhio, J., Ukkonen, E.: Approximate BoyerMoore String Matching. SIAM Journal on Computing, Vol. 22, No. 2, pp. 243-260 (1993)
29. Drozd, J., Drozd, A., Antoshchuk, S., Kharchenko, V.: Natural Development of the Resources in Design and Testing of the Computer Systems and their Components. In: 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 233-237. Berlin, Germany (2013)
30. Kondratenko, Y., Kondratenko, N.: Soft Computing Analytic Models for Increasing Efficiency of Fuzzy Information Processing in Decision Support Systems. Chapter in book: Decision Making: Processes, Behavioral Influences and Role in Business Management, R. Hudson (Ed.), Nova Science Publishers, New York, 41-78 (2015)
31. Proskurin, V.: Software malicious implant in secure systems. Available: http://www.crime-research.ru/library/progwir98.htm [in Ukrainian].
32. Kaarin, O. V.: Program protection theory and practice. MGUL, pp. 450 (2004) [in Russian].
33. Kaarin, O. V. Computer system software security. MGUL, pp. 212 (2003) [in Russian].
34. Shanugin, V. F. Protection of computer information. Effective methods and tools: a textbook. DMK Press, pp.544 (2008) [in Russian].
35. Shanugin, V. F. Protection of information in computer systems and networks. DMK Press, pp. 592 (2012) [in Russian].
36. Balakrishnan, G., Reps, T.: WYSINWYX: What You See Is Not What You eXecute. In: Proc of ACM Transactions on Programming Languages and Systems, Vol. 32, Issue 6, (2010).
37. Igumnov, B. N., Zavgorondnyaya, T. P. Cybernetic basis of construction of economic systems of enterprises. Khmelnitsky, TUP, pp. 344 (2000) [in Russian].
38. Palyata, V. B. Comparative multidimensional analysis in economic research. Statistica, pp. 151 (1980) [in Russian].
39. Shatalkin, A.I. Taxonomy: Grounds, principles and rules. Tovarischestvo nauchnyih izdaniy KMK, pp. 600 (2012) [in Russian].
40. Zhambyu, M. Hierarchical cluster analysis and matching. Finance and statistics, pp. 345 (1988) [in Russian].
41. Ward, J.H.: Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association, Vol. 58, No. 301, pp. 236-244 (1963).
42. Savenko, O., Nicheporuk, A., Hurman I., Lysenko, S.: Dynamic Signature-based Malware Detection Technique Based on API Call Tracing. CEUR Workshop, Vol. 2393, pp. 633–643 (2019).
43. Savenko, O., Lysenko, S., Kryschuk, A.: Multi-agent based approach of botnet detection in computer systems. Communications in Computer and Information Science, Vol. 291, pp.171-180 (2012).
44. Balyk, A., Karpinski, M., Naglik, A., Shangytbayeva, G., Romanets, I.: Using graphic network simulator 3 for DDoS attacks simulation. International Journal of Computing. Vol. 16, Issue 4, pp. 219-225 (2017).

68. Stetsyuk M., Stetsyuk V., Savenko B., Savenko O., Dobrowolski M. Implementation of Control by Parameters of Client Automated Workplaces of Specialized Information Systems for Neutralization malware. CEUR-WS. 2021. Vol. 2853. P. 340-352. URL: http://ceur-ws.org/Vol-2853/paper40.pdf.

Implementation of Control by Parameters of Client Automated Workplaces of Specialized Information Systems for Neutralization malware

Mykola Stetsyuk[a], Vasyl Stetsyuk[a], Bohdan Savenko[a], Oleg Savenko[a], Maciej Dobrowolski[b]

[a] *Khmelnytskyi National University, Instytutska str., 11, Khmelnytskyi, 29016, Ukraine*
[b] *Kazimierz Pulaski Technology and Humanitarian University, Malczewskiego St 29, Radom, 26-600, Poland*

Abstract

The paper presents a topical scientific problem for the development of information technology, which automatically allows you to neutralize the manifestations of malicious software on specialized information systems.

The risks of malicious software attacks depending on the executable file formats are analyzed. The analysis of methods to ensure fault tolerance and survivability of specialized IP showed that the current methods and technologies do not fully ensure their fault tolerance and survivability in terms of counteracting the impact of malware. Despite the invariance of the methods used, the counteraction procedure is reduced to the organization of a single-level scheme at the system-wide level.

This is enough to ensure the functionality of an ordinary computer system that provides general information needs, but not enough to ensure access to the functionality of specialized IP at any time.

A method of parametric control of client automated workstations (AWP) of specialized information systems to neutralize the effects of malicious software has been developed. The proposed technology to ensure fault tolerance and survivability of automated workstations of specialized IT and developed a method of parametric control of the relevance of client workstation software provide a high level of IP stability in general against the effects of malware. In fact, it realizes the second line of counteraction to malicious software, in comparison with the system-wide one, where it is not always possible to neutralize the destruction by malicious software. At the same time, being combined with the software support service, it does not require additional costs to support its operation. Experimental studies were conducted with the developed information system, which confirmed the improvement of its efficiency, reliability and proposed solutions.

Keywords

malfunctioning software, information system, information technology, performance, software comparator, vitality

1. Introduction

Today, it is difficult to identify areas where the total use of information technology has not found its recognition. Information technology has penetrated into almost all spheres of modern society, including such specialized as financial activities, medical, military.

IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24-26, 2021, Khmelnytskyi, Ukraine
EMAIL: mikstt777@gmail.com (M. Stetsyuk); svvasan@gmail.com (V. Stetsyuk); Savenko_bohdan@ukr.net (B. Savenko); savenko_oleg_st@ukr.net (O. Savenko); m.dobrowolski@uthrad.pl (M. Dobrowolski)
ORCID: 0000-0003-3675-0416 (M. Stetsyuk); 0000-0001-9880-2666 (V. Stetsyuk); 0000-0001-5647-9979 (B. Savenko); 0000-0002-4184-745X (O. Savenko); ORCID 0000-0003-0296-9651 (M. Dobrowolski)

But along with the positive aspects of their use, we have to accept the idea that new information technologies are very sensitive to various kinds of destruction, one of which is the various ways of malicious software, which in the absence of properly organized counteraction paralyzes the information system which entails a lot of negative consequences [1-7].

Therefore, the task of organizing the work of specialized information systems in the face of malicious software, which in turn is part of a more global task of ensuring fault tolerance and survivability of the information system.

This task is considered to have a continuous solution and, at the same time, being complex, includes a number of sub-tasks, such as legal, organizational and, of course, software and hardware, which are responsible for developing mechanisms to counter the effects of malware. It is the latter that have become the subject of this article.

2. Analysis of known solutions

The analysis of methods to ensure fault tolerance and survivability of specialized information systems showed that the current methods and technologies do not fully ensure their fault tolerance and survivability in terms of counteracting the impact of malicious software. Despite the invariance of the methods used, the counteraction procedure is reduced to the organization of a single-level scheme at the system-wide level.

This is sufficient to ensure the operability of an ordinary computer system that provides general information needs, but not enough to guarantee access to the functionality of a specialized information system at any time.

In [1] presents approaches to responding to accidents in computer systems under the influence of malicious software. This is important, including the operation of information systems in computer networks. [2] explains the features of hardware security. In [3] such type of malicious software as botnets is analyzed. Using them causes significant harm to users of computers connected to the Internet. In [4] the security features of IP networks are analyzed. [5] presents forecasts on trends in the development of threats from malicious software. In [6] the influence on the possibility of detecting this type of viruses as metamorphic was analyzed. Their masking complicates the processes of their effective detection. In [7] the possibilities of protection of the hardware and software of the user from external influences are presented. Considered various aspects of the problem area to ensure security in computer systems, indicate the existence of an unresolved problem to ensure the security of processes in them due to the impact of malicious software.

If the object of the attack is a specific information system, and the goal is to block its work, then one level of resistance, as the events of 2016 have shown, may not be enough. This is confirmed by successful malware attacks recorded on December 6, 2016 [8]. Their targets were the internal telecommunications networks of the Ministry of Finance, the State Treasury, the Pension Fund and, as a result, blocking access to critical databases, which led to delays in budget payments. On December 15, an attack was made on Ukrzaliznytsia's information system, as a result of which its work was completely blocked during the day.

Another aspect that was considered in the analysis of the construction of anti-malware systems is that the construction of such systems is by typification and standardization [9-11]. This is a natural way of developing the defense mechanisms of computer systems, which has many positive, no doubt, moments, but at the same time, its undeniable drawback is that the typing process itself facilitates the creation of mechanisms to overcome the means of protection. And here there is a collision, when on the one hand, we can not give up the benefits of typification and standardization in creating mechanisms to counter the effects of malicious software, and on the other hand, we can not accept the fact that such an approach effectively, in turn, simplifies the creation of mechanisms to overcome the protection of the information system. Thus, standardization in the development of IP makes it easier for attackers to develop malicious software focused on such IP.

An important area of ensuring the stability of IP under the influence of malware is the choice of an appropriate effective mathematical apparatus as a basis for the search for abnormal or malicious manifestations [12-15]. Malicious software controlled by an attacker, which is a botnet [16-17], is aimed precisely at taking control of reptiles by user computer systems and gaining access to

information systems [18-25]. The authors of the article [24, 25] consider cloud programs, which are considered to be components of several components of cloud services that interact with each other, where each component performs certain functionalities. A comprehensive recovery scheme based on software rejuvenation for cloud applications is proposed, which consists of three important parts: adaptive fault detection, aging assessment, and component-based rejuvenation checkpoint. In the article [26] the use of the clonal selection algorithm as a mathematical apparatus is considered. Therefore, the choice of mathematical software as the basis of methods for detecting abnormal or malicious manifestations, when creating IP that must meet the requirements of fault tolerance and survivability in the face of malware, is an important task.

We will also consider other strategic approaches to solving the problem of ensuring the stability and survivability of IP in the face of malware. In [27] the approach for avoiding functional failures during execution in component application systems is presented. The approach uses the internal redundancy of components to find workarounds as alternative sequences of operations to avoid failures.

In articles [28, 29] methods of ensuring reliability and functional security of software packages in real time are offered. In [30], tolerance to failure is a major problem in ensuring the availability and reliability of critical services, as well as program implementation. In order to minimize the impact of failures on the system and the implementation of applications, it is necessary to anticipate deviations and take measures for them. Failure tolerance methods are used to predict these failures and take appropriate action before the failures actually occur. In [31] the use of application software interface calls in malware detection problems is shown. This is required for inclusion in detection systems or as part of certain IPs.

In [32, 33], cyber resilience and viability are presented as closely related concepts with similar technologies and practices. For historical reasons, these concepts have been embedded in different frameworks that define different constructs to describe problems and areas of solution.

In [34-40] shows the impact on the resilience and survival of IT of various types of malware and computer attacks. Paper [39] presents and discusses a method for classifying Android applications to detect malware. Based on the use of an artificial immune system and artificial neural networks, an antivirus system has been proposed, especially for the Android system, which can detect and block unwanted and malicious programs. This system can be characterized by self-adaptation and self-evolution and can detect even unknown and previously unseen malware. That is, the proposed approaches allow the system to respond dynamically to events.

Problems to ensure fault tolerance and survivability of specialized information technologies in the face of malicious software and computer attacks are issues of research, including the hardware infrastructure where they operate. Work [40] shows a study of such a class of devices as a router. This document examines the spread of DDoS attacks on the router subsystems of the Smart Office system. This paper analyzes and solves the problem of optimizing the search for the minimum path of attack on the router subsystems. The result of this work is to determine the most vulnerable subsystems of the router to the consequences of DDoS-attacks.

Paper [41-44] considers the problems of hidden faults that are inherited in security systems aimed at ensuring the functional safety of high-risk facilities to combat accidents, which is also important and should be taken into account when ensuring resilience and survivability of specialized information technologies.

The problem of hidden faults is considered in terms of resource-oriented approach as a problem of growth from the lowest level of replication to the next level of diversification in the development of models, methods and tools [45-46].

To consider malicious manifestations and methods of counteraction to them allow to use these results at creation of information technologies with the increased level of maintenance of fault tolerance and survivability in the conditions of influences of malicious software.

3. Formulation of the problem

The practice of using information technology has shown that viable methods to ensure fault tolerance and survivability of the information system are those that are characterized not only by

potentially high efficiency parameters, but at the same time, remain simple and cheap to use. This fully applies to measures to neutralize the effects of malicious software.

The events that took place in Ukraine in the period from 2013 to 2018 showed the vulnerability of modern information technologies, which in turn makes it urgent to develop methods to improve the efficiency of fault tolerance and survivability of the information system and, even more, specialized information systems in which critical data is usually processed.

It is proposed to supplement the existing methods of ensuring the resilience and survivability of the information system in terms of neutralizing the impact of malicious software technology, which is based on the idea of ease of implementation and ensuring high efficiency of specialized information technology in the effects of malicious software. In this case, despite the simplicity of implementation, a feature of the new technology of information system protection is its operation in automatic mode.

An important point of its operation is the inclusion in its tasks to document the identified manifestations of malware, which allows for constant analysis of information about events in the information system in order to improve methods of counteraction.

4. Main part

One of the ways to solve this problem is to use two levels of counteraction to the effects of malicious software, the first of which, system-wide, is built using conventional counteraction mechanisms, and the second, local, is implemented within the most specialized information system, using its nuances of operation and architecture (Figure 1).

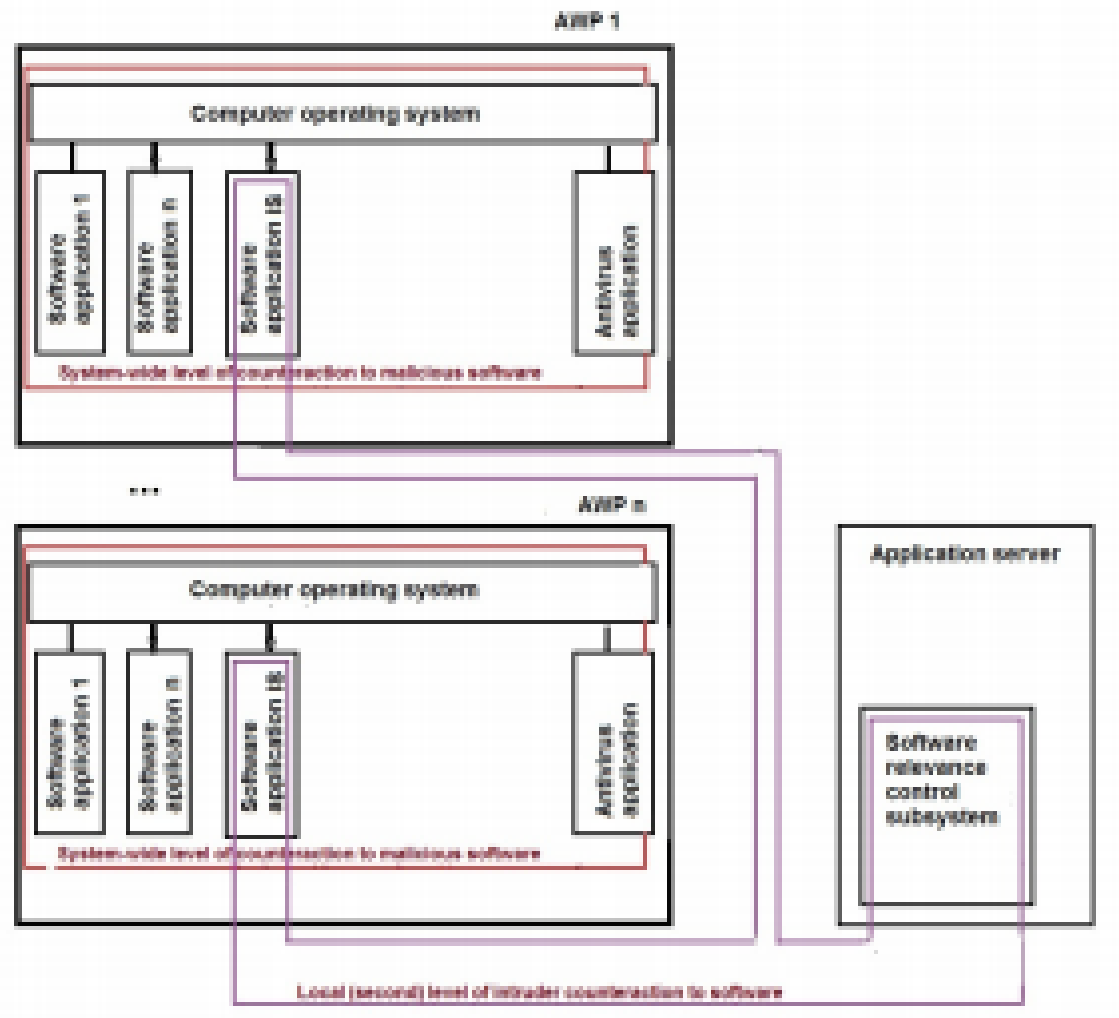


Figure 1: A two-tier anti-malware scheme for client automated workstations of a specialized information system.

This approach will increase the likelihood of neutralizing the target attack of malicious software on the objects of a specialized information system, make the work of malicious software as difficult as possible in order to increase the availability of the information system at any time. It is proposed to use as a mechanism to counter attacks, already existing in most developed specialized information systems, the service of maintaining the relevance of client software, giving it new qualities, described in the following method.

5. A method of ensuring the fault tolerance and survivability of the information system in the face of malicious software using parametric control of the relevance of software modules of client automated workstations and their masking

To solve the problem of restoring the functionality of the information system in order to prevent the effects of malicious software, increase the degree of warranty of the information system, a method of ensuring fault tolerance and survivability is proposed, the model of which is shown in Figure 2.

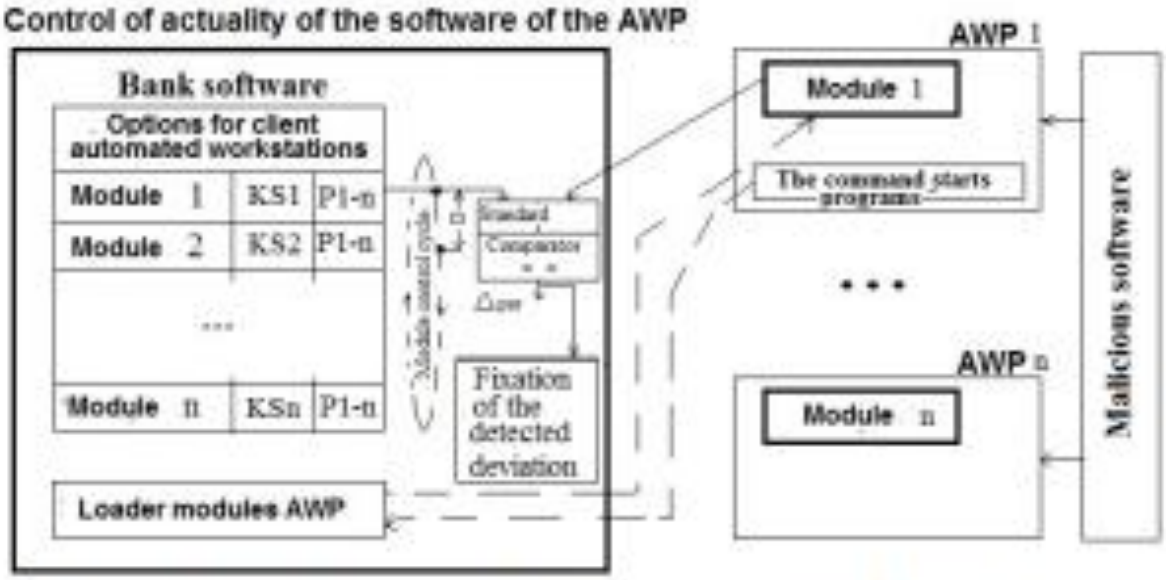


Figure 2: Model of the method of ensuring fault tolerance and survivability of the information system under the influence of malicious software using parametric control of the relevance of software modules of client automated workstations and their masking

The essence of the method is to carry out constant cyclic control of the parameters of the modules of client automated workstations with a given discreteness D.

Discreteness is a parameter whose value is chosen based on the level of system-wide performance of the computer network and client computers on which automated workstations are based. This allows you to adapt this technology to the hardware platforms of the information system of different performance.

To ensure the robustness of the method, its model includes a software bank that contains software modules of all client automated workstations of the information system and their reference parameters, such as checksums KS1 - KSn code pages, calculated according to a given rule. The set of control parameters can be changed according to the structure of the controlled software modules. In the process of monitoring the relevance of the state of the software module of the client automated workplace, its availability is checked in a given way, its parameters are calculated and compared with the reference.

The task of parametric control of relevance of modules within the limits of this method is assigned to the software implemented comparator.

In the absence of a controlled module in a given place, or a difference between the actual and reference parameters Δpar at the output of the comparator, the software of the client automated workstation is restored using the reference software stored in the bank. The very fact of detecting discrepancies Δpar is automatically documented while maintaining the necessary parameters for further analysis in the database.

If no discrepancies are found between the standard and the module that has passed the relevance check, it can, in addition, be masked by renaming. This will reduce the likelihood of the module being attacked by malicious software, which is known to primarily affect executable files. This method allows you to control the relevance of the modules of client automated workstations in automatic mode, which in turn allows you to ensure fault tolerance and survivability of the information system under the influence of malicious software.

In this case, the nature of the attack of malicious software on client software does not play a special role. Because no information system data is stored on client computers, malicious software can only damage program files. This fact is manifested in the process of monitoring the relevance of software modules for automated workstations and they are replaced by reference.

6. Technology to ensure fault tolerance and survivability of the information system under the influence of malicious software using the method of parametric control of software relevance of client automated workstations

It is known that all information systems are characterized by a long life cycle, during which their software, under the influence of many external and internal factors is subject to change. And these changes are all the more significant the larger the subject area covered by the information system.

The task of maintaining the relevance of the software of client workstations of a specialized information system is quite extensive, so, as a rule, the natural course of development of the information system leads to the transition to an automated subsystem to restore the relevance of software. In the future, we will call it the software support service for client automated workstations. Its task is to detect software updates for client automated workstations of the information system and perform its replication to all workstations with settings for a specific automated workstation.

The main component of the software relevance support service is the reference software bank. In the process of improving the information system, changes are made to the software, which in turn leads to the replacement of the standard software in the bank. The task of the support service is to identify the fact of changing the standard of the software and, in response to this, to start the procedure of updating the software of all client workstations where it is used.

If we compare the above method of ensuring the fault tolerance and survivability of the information system under the influence of malicious software with the work of the information system software support service, we will see that their implementation will be based on similar algorithms. The only difference is that the algorithms of the software relevance service respond to the change of the software standard, and the algorithms of the method of ensuring the fault tolerance and survivability of the information system to the loss of compliance with the software instance standard n-th client workplace. The reaction in both cases will be the same - the restoration of the software of the client workplace of its standard in the bank.

Therefore, the basis of the proposed technology to ensure fault tolerance and survivability of a specialized information system under the influence of malicious software is to put the idea of using the functionality of the existing service to maintain the relevance of software for client automated workstations. To do this, its algorithms have been improved by including in its composition functions that ensure fault tolerance and survivability of the client part of the information system in the conditions of malicious software.

The method of counteracting attacks of malicious software, the model of which is shown in Figure 2, implemented in the form of information technology, which includes several interacting at the

software level processes. This technology is in addition to the already known ways to protect the information system by counteracting malicious software attacks. It includes a background process, the algorithm of which is shown in a simplified form in Fig. 3, during which, in fact, the relevance of software modules of client automated workstations and a special procedure are checked launching software modules for automated workstations for execution (Figure 4).

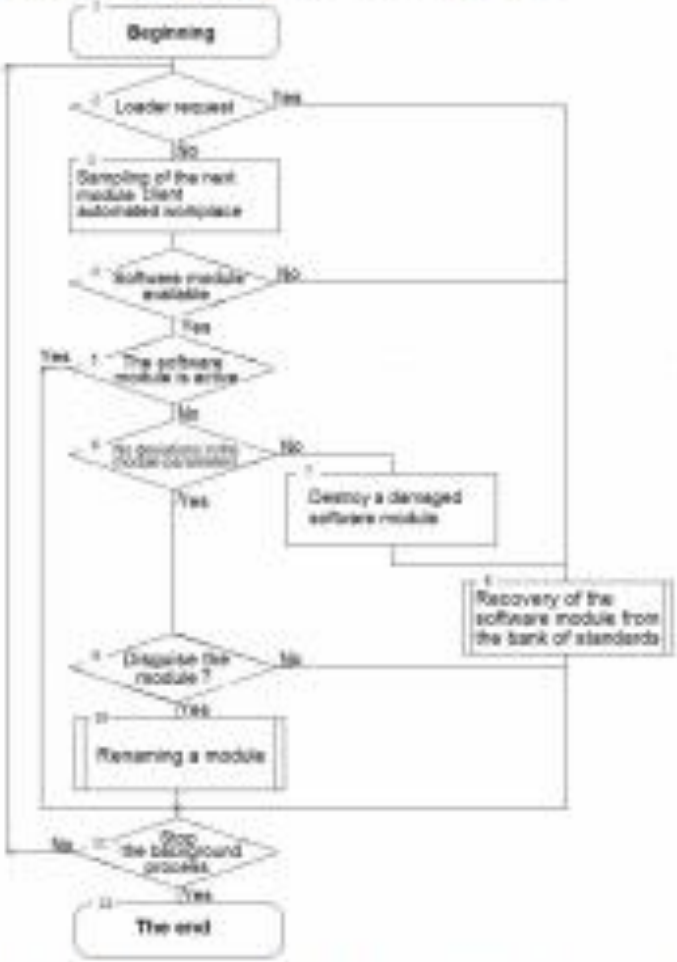


Figure 3: Algorithm of the background process of the software relevance support service in terms of ensuring fault tolerance and survivability of the client part of the specialized information system under the influence of malicious software.

1. The background process checks the availability of a service request from the loader of the client automated workstation modules. This is necessary in order that the reaction to the fact of failure to start some software module of the automated workplace was the fastest (Figure 3, operator 2).

The background process of the software relevance service, running at a specified frequency, monitors the software modules of each registered in the information system client automated workstation. In each iteration, the process performs a given sequence of operations, which implements the algorithm for monitoring the relevance of the software of client automated workstations:

This situation in the information system can occur when the operator of an automated workstation tries to run its client program for execution, and it is for some reason unavailable. The program downloader, at the time of attempting to run it, detects this fact and submits an application to the background process for the primary recovery of the software specified in the application automated workstation.

Upon receipt of the application, the background process reads from it the number of the software module, which requires priority maintenance and goes to step 6.

2. If there is no request for emergency maintenance, the background process proceeds to check the next module in the queue (Figure 3, operator 3).

3. In the next step, it is checked whether the module of the client automated workstation under analysis is located in a certain place in the file directory of the client computer. In case of its absence for any reason the transition to point 6 (Figure 3, operator 4) is carried out.

4. If the module is available and in a given place, then check its activity (Figure 3, operator 5). At this stage, it is determined whether it is loaded into the memory of the PC and performs the function assigned to it within the information system. If at the time of testing the module is active, then go to step 7.

5. Control of parameters of the next software module of the n-th automated workplace on conformity to the reference stored in bank of service of actuality of the software (figure 3, operator 6). If no deviations from the reference parameters are detected, the file is masked by renaming (Figure 3, operators 9,10). This allows you to remove it from a possible attack by malware, knowing that it attacks executable files, focusing on their extension. Then go to step 7, otherwise to the next.

6. Replace the damaged or restore the missing software module with a reference from the software bank (Figure 3, operator 8).

7. The supply of the command to stop the background process is checked (Figure 3, operator 11). If not, the current iteration is completed, followed by step 1.

8. Completion of the background process.

Since this technology is intended for specialized information systems, which in themselves can be the object of targeted attack, the algorithm (Figure 3) may include another function, the task of which is to form the location of the executable modules of client automated working places special files-traps that should serve as false objects of attack for malware, while the real modules are disguised.

Trap files are no different from software modules in automated information system workstations except that they can never be downloaded for execution. Before starting the real module, they are destroyed and then re-created after the module completes its work.

Along with the function of masking the software modules of automated workstations, the function of creating false objects of attack allows you to direct the destructive actions of malicious software in a direction that does not threaten the functioning of the information system.

The process of launching software modules of client automated workstations for execution also has its own feature - it is performed in two stages. First, from the client PC, a short bootloader program is launched, which is permanently stored in a secure directory of the software service of the specialized information system. The bootloader starts, finds the file of the corresponding module in a certain place, restores its name and transfers control to it.

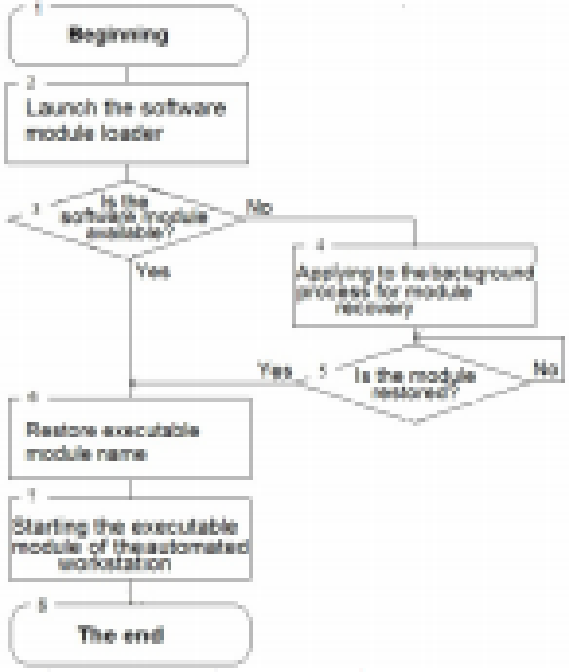


Figure 4: Algorithm of work of the loader of modules of the automated workplaces of service of support of actuality of the client software.

If the file for some reason is not found, the module loader will issue to the background process a request for an extraordinary software update of the specified automated workstation and will go into standby mode (Figure 4. operators 4,5). After the background process executes the loader's request, it will in turn repeat the procedure of starting the corresponding module (Figure 4 operators 6,7).

The ability of this technology to restore damaged and destroyed software of client automated workstations allows you to eliminate the effects of malicious software attacks that penetrate into the module of automated workstations information system, and those whose destructive actions are manifested in data encryption.

7. Influence of executable file format on the frequency of malware attacks

In order to find ways to improve the efficiency of the proposed method of ensuring the resilience and survivability of specialized information systems, an analysis of the frequency of malware attacks depending on the type of executable files within the operating system MS Windows. This operating system allows you to work with a fairly wide range of executable files. These are primarily COM and EXE program files. Next are the system drivers. They have the extension SYS or BIN. Executable files also include batch files. These are called BAT files.

Executable files also include overlay files and dynamically loaded libraries that are used by programs as needed.

The analysis (Table 1) showed that malicious software most often uses files in COM and EXE formats as objects of its attack. They are followed by CMD and BAT files and SYS and BIN driver files.

But the use of other file types (INF, INS, MSC, MSI, PIF, REG, VBS, MDB, MDE) for destructive purposes by malicious software is rare.

Table 1

Dependence of the frequency of malware attacks on the format of the executable file

File type	Risk of being attacked by malware
EXE	Highest
COM	High
CMD & BAT	Average
SYS & BIN	Low
MDB, MDE	The lowest
Other types	Not analyzed

Of all the above files, we will be interested only in the file with the MDE format of the MS Access package. Executed with functionality that allows you to implement a software system of any complexity, it has the lowest risk of being attacked by malicious software. This means that it is ignored by the developers of the malicious code and no cases of infection have been found.

To date, only attempts to destroy the contents of the MDE file by the destructive actions of the Blackmal virus by entering the line "DATA Error" have been detected. But the destruction of the contents of the file is not an infection of the file and, accordingly, such destruction by malicious software does not threaten serious consequences for the data, but only requires the replacement of the distorted file with a new one.

Therefore, this fact (the presence of its own format of the executable MDE-file, little prone to infection with malware), among others, significantly influenced the recommendation to choose MS Access as a tool for software development of client automated workstations of specialized information systems.

The MDE file is a special format of the MS Access database, and in turn is derived from the MDB-type database of MS Access. Its feature is that part of the database components, which may include executable modules - forms, reports, modules, macros - is stored in the middle of the MDE-file in compiled form, which does not allow any changes to their source text, as well as their review, but it remains possible to make changes to the table and queries. It is positioned as a DBMS file with

advanced data manipulation capabilities. The database data can be in the same file, or in another MDE file, or MDB-file. It is also possible to work with data contained in any non-MS Access database that supports ODBC data access technology. An MDE file is an executable file in MS Windows and MAC. It can be started by MS Microsoft Access or RUN Time Access.

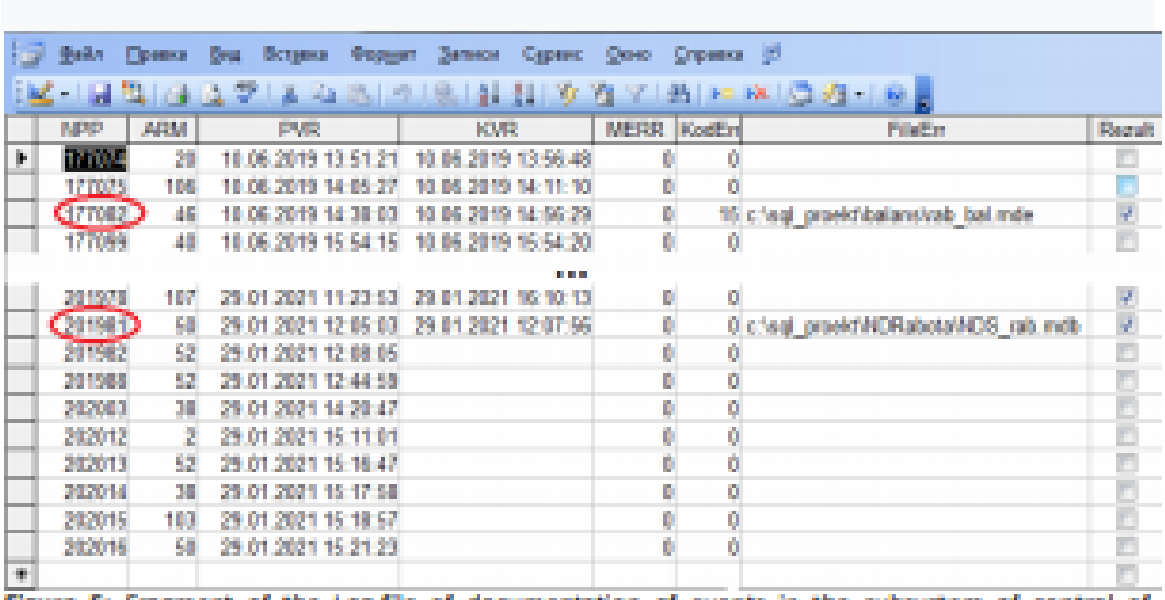
Such properties of the MDE-file can significantly increase the security of the information system as a whole, because its users do not have access to the source code of the software modules of the user's automated workstation components, and therefore their potentially destructive actions against databases.

8. Experimental studies

The subsystem of control of integrity of the client software is realized in the specialized information system "Management of financial resources of KhNU" in its automated workplace "ADMINISTRATOR". The software of this information system was developed in MS Access, which was caused by a number of points, one of which was the desire to reduce the likelihood of its modules to be attacked by malicious software. Unfortunately, this approach only works if this development tool is not widely used.

An experiment was performed with this information system, in which the situation of damage to one of the files of the automated workplace №46 by malicious software was simulated - changes were made to one of its code pages using a HEX editor. As a result, its structure began to differ from the reference.

As can be seen from the fragment of the log file shown in Figure 5, when trying to run at 14:38 10.6.19 the program of the automated workstation "BALANCE" for execution, the software comparator of comparison of files of the subsystem of control of actuality of the client software, deviations from reference parameters were revealed.



NPP	ARM	PVR	KVR	MERR	KodErr	FileErr	Result
[illegible]	28	10.06.2019 13:51:21	10.06.2019 13:56:48	0	0		
177071	106	10.06.2019 14:05:27	10.06.2019 14:11:10	0	0		
177082	46	10.06.2019 14:38:03	10.06.2019 14:56:29	0	15	c:\sql_proekt\balans\rab_bal.mde	
177099	48	10.06.2019 15:54:15	10.06.2019 15:54:20	0	0		
...							
201978	107	29.01.2021 11:23:53	29.01.2021 16:10:13	0	0		
201981	58	29.01.2021 12:05:03	29.01.2021 12:07:56	0	0	c:\sql_proekt\NDRabota\NDS_rab.mdb	
201982	52	29.01.2021 12:08:05		0	0		
201988	52	29.01.2021 12:44:58		0	0		
202003	38	29.01.2021 14:29:47		0	0		
202012	2	29.01.2021 15:11:01		0	0		
202013	52	29.01.2021 15:16:47		0	0		
202014	38	29.01.2021 15:17:58		0	0		
202015	103	29.01.2021 15:18:57		0	0		
202016	58	29.01.2021 15:21:23		0	0		

Figure 5: Fragment of the Log-file of documentation of events in the subsystem of control of actuality of the client software of the information system of management of financial resources of KhNU

The situation was recorded in the log-file of this subsystem in the form of a record number 177082 with error code "15".

This code indicates the mismatch of the checksum of the code module of the file rab_bal.mde workplace №46 to the parameters of the file stored in the database of standards.

As a result of further operation of the client software relevance control subsystem, the file with the damaged part of the code was deleted and replaced with a new one from the database of standards.

Another operation of the comparator, recorded in line 201981 of the listing, documented the event of a discrepancy between the parameters of the standard and the file NDS_rab.mdb in the workplace №50. Error code "0" indicates that the cause of the operation was an update of the software version of this automated workstation.

9. Conclusions

The proposed technology to ensure fault tolerance and survivability of automated workplaces of specialized information systems based on the method of parametric control of software relevance of client automated workplaces provides a high level of stability of the information system as a whole against malware.

In fact, it implements the second line of counteraction to malicious software, compared to the system-wide, which is not always possible to neutralize the destruction of malicious software. At the same time, being combined with the software relevance support service, it does not require additional costs to support its operation.

The direction of further research is to find ways to increase the efficiency of the proposed technology in ensuring fault tolerance and survivability of specialized information systems.

10.References

[1] A. Steve. Applied Incident Response. John Wiley & Sons, Inc., 2020.

[2] S. Bhunia, M. Tehranipoor Hardware Security: A Hands-on Learning Approach. Morgan Kaufmann, 2019

[3] O. Savenko, S. Lysenko, A. Kryschuk Multi-agent based approach of botnet detection in computer systems, Communications in Computer and Information Science 291 (2012) 171-180

[4] B. Swarup, R. Sandip, S.-K. Susmita. Fundamentals of IP and SoC Security: Design, Verification, and Debug. Springer, 2017

[5] R. S. Grinyov, O. V. Severinov, Analysis of trends in viral threats in Ukraine, in: Proceedings of Modern directions of development of information and communication technologies and management tools, Kharkiv, 2019 [in Ukrainian]

[6] O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko, Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search, CEUR-WS, 1844 (2017) 555–569.

[7] Y. Y. Gromov, O. G. Ivanova, K. V. Starodubov, A. A. Kadykov, Software and hardware means of protection of information systems, TSTU, 2017 [in Russian]

[8] Electronic magazine "Nowoe Vremya". The largest cyber attacks against Ukraine since 2014. Infographics, URL: https://nv.ua/ukraine/events/krupnejshie-kiberataki-protiv-ukrainy-s-2014-goda-infografika-1438924.html [in Russian]

[9] O. S. Savelyeva, O. M. Krasnozhon, O. U. Lebedeva, Using the structural fault-tolerance index in project designing. Odes'kyi Politechnichnyi Universytet. Pratsi, 2 (2014) 130–135. doi: 10.15276/opu.2.44.2014.24.

[10] S. Boranbayev, S. Altayev, A. Boranbayev. Applying the method of diverse redundancy in cloud based systems for increasing reliability, in: Proceedings of the 12th International Conference on Information Technology: New Generations (ITNG 2015), Las Vegas, Nevada, 2015, pp. 796-799.

[11] A. Boranbayev, S. Boranbayev, K. Yersakhanov, A. Nurusheva, R. Taberkhan R, Methods of Ensuring the Reliability and Fault Tolerance of Information Systems, Advances in Intelligent Systems and Computing, 738 (2018)

[12] Y. Kondratenko, N. Kondratenko, Soft Computing Analytic Models for Increasing Efficiency of Fuzzy Information Processing in Decision Support Systems. Chapter in book: Decision Making: Processes, Behavioral Influences and Role in Business Management, R. Hudson (Ed.), Nova Science Publishers, New York, 2015, 41-78

[13] L. Bedratyuk O. Savenko, The Star Sequence and the General First Zagreb Index, MATCH Communications in Mathematical and in Computer Chemistry, 79 2 (2018) 407-414.

[14] M. Chinnaiah, N. Niranjan, Fault tolerant software systems using software configurations for cloud computing, J Cloud Comp 7 3 (2018). doi: https://doi.org/10.1186/s13677-018-0104-9.

[15] D. Fitzpatrick, D. Bodeau, R. Graubart, R. McQuaid, C. Olin and J. Woodill, (DRAFT) Cyber Resiliency Evaluation Framework for Weapon Systems: Foundational Principles and Their Potential Effects on Adversaries, The MITRE Corporation, Bedford, MA, 2019.

[16] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, Multi-Agent Based Approach for Botnet Detection in a Corporate Area Network Using Fuzzy Logic, Communications in Computer and Information Science 370 (2013) 243-254

[17] S. Lysenko, O. Savenko, A. Kryshchuk, Y. Kljots, Botnet detection technique for corporate area network, in: Proceedings of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems, 2013, pp. 363-368

[18] NIST, "Initial Public Draft of NIST SSP 800-160 Volume 2, Systems Security Engineering: Cyber Resiliency Considerations for the Engineering of Trustworthy Secure Systems, 2018 URL: https://csrc.nist.gov/CSRC/media/Publications/sp/800-160/vol-2/draft/documents/sp800-160-vol2-draft.pdf.

[19] X. Zhu, J. Wang, H. Guo, D. Zhu, L .T. Yang, L. Liu Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds, IEEE Transactions on Parallel and Distributed Systems, 27 12 (2016) 3501–3517. doi: https://doi.org/10.1109/TPDS.2016.2543731.

[20] A. Bala, I. Chana, Fault tolerance-challenges, techniques and implementation in cloud computing, International Journal of Computer Science Issues 9(1) (2012)

[21] W. Zhao, Z. Wenbing, P. M. Melliar-Smith, L. E. Moser, Fault Tolerance Middleware for Cloud Computing, in: Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing, Miami, USA, 2010, pp. 67–74. doi: https://doi.org/10.1109/CLOUD.2010.26.

[22] I. P. Egwutuoha, S. Chen, D. Levy, B. Selic A fault tolerance framework for high performance computing in cloud, Cluster, Cloud and Grid Computing (CCGrid), in: Proceedings of the 12th IEEE/ACM international symposium. 2012, 709–710. doi: https://doi.org/10.1109/CCGrid.2012.80.

[23] S. Lysenko, K. Bobrovnikova, S. Matiukh, I. Hurman, O. Savenko, Detection of the botnets' low-rate DDoS attacks based on self-similarity, International Journal of Electrical and Computer Engineering (Q2) 10 4 (2020) 3651-3659

[24] J. Liu, J. Zhou J., R. Buyya, Software rejuvenation based fault tolerance scheme for cloud applications, in: Proceedings of 2015 IEEE 8th International Conference on Cloud Computing, New York, USA, 2015, pp. 1115–1118. https://doi.org/10.1109/CLOUD.2015.164.

[25] J. Liu, S. Wang, A. Zhou, S.A.P. Kumar, F. Yang, R. Buyya, Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability, in: Proceedings of IEEE Trans Cloud Computing PP(99), 2016. doi: http://dx.doi.org/10.1109/TCC.2016.2567392.

[26] S. Lysenko, K. Bobrovnikova, O. Savenko, A Botnet Detection Approach Based on The Clonal Selection Algorithm, in: Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DeSSerT-2018, Kyiv, Ukraine, 2018, pp. 424-428.

[27] P. Nicolo, A frame work for self-healing software systems, in: Proceedings of the IEEE 35th International Conference on Software Engineering, 2013, pp. 1397–1400. doi: https://doi.org/10.1109/ICSE.2013.6606726.

[28] A. S. Markov V. L. Tsirlov, A. V. Barabanov, Methods for assessing the inconsistency of information security measures, Radio and communication, Echelon-Espadon, 2012 [in Russian]

[29] V. V. Lipaev Reliability and functional security of real-time software packages, Institute of System Programming, Russian Academy of Sciences, 2013 [in Russian]

[30] A. Balyk, M. Karpinski, A. Naglik, G. Shangytbayeva, I. Romanets, Using graphic network simulator 3 for DDoS attacks simulation, International Journal of Computing 16 4 (2017) 219-225

[31] O. Savenko, A. Nicheporuk, I. Hurman, S. Lysenko, Dynamic signature-based malware detection technique based on API call tracing, CEUR-WS 2393 (2019) 633-643

[32] S. Pitcher, "New DoD Approaches on the Cyber Survivability of Weapon Systems, 2019. URL: https://www.itea.org/wp-content/uploads/2019/03/Pitcher-Steve.pdf.

[33] D. J. Bodeau, R. D. Graubart, R. M. McQuaid and J. Woodill, Cyber Resiliency Metrics and Scoring in Practice: Use Case Methodology and Examples (MTR 180449), the MITRE Corporation, Bedford, MA, 2018.

[34] K. Alminshid M. N. Omar, Detecting backdoor using stepping stone detection approach, in: Proceedings of the 2013 Second International Conference on Informatics & Applications, Lodz, Poland, 2013, pp. 87-92

[35] J. Zaddach, A. Kurmus, D. Balzarotti, E.-O. Blass, A. Francillon, et al., Implementation and Implications of a Stealth Hard-drive Backdoor, in: Proceedings of the 29th Annual Computer Security Applications Conference, New Orleans, Louisiana, US, 2013.

[36] T. F. Dullien, Weird machines, exploitability, and provable unexploitability, IEEE Transactions on Emerging Topics in Computing 99 (2017) 1-15

[37] S. L. Thomas, T. Chothia, F. D. Garcia, HumIDIFy: A Tool for Hidden Functionality Detection in Firmware, in: Proceedings of the 14th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Bonn, Germany, 2017, pp. 279-300

[38] S. L. Thomas, T. Chothia, F. D. Garcia, Measuring the Importance of Static Data Comparisons to Detect Backdoors and Undocumented Functionality, in: Proceedings of the 22nd European Symposium on Research in Computer Security. Oslo, Norway, 2017, pp. 513-531

[39] S. Bezobrazov, A. Sachenko, M. Komar, V. Rubanau, The method of artificial intelligence for malicious applications detection in android OS, International Journal of Computing, 15(3) (2016) 184-190

[40] M. Kolisnyk, V. Kharchenko, I. Piskachova, Research of the attacks spread model on the smart office's router, International Journal of Computing, 19(4) (2020) 629-637.

[41] V. Golovko, Y. Savitsky, T. Laopoulos, A. Sachenko, L. Grandinetti, Technique of learning rate estimation for efficient training of MLP, in: Proceedings of the International Joint Conference on Neural Networks 1, 2000, pp. 323-328

[42] R. Kochan, K. Lee, V. Kochan, A. Sachenko, Development of a dynamically reprogrammable NCAP, in: Proceedings of the Conference Record - IEEE Instrumentation and Measurement Technology Conference 2, 2004, pp. 1188-1192

[43] A. Melnyk, V. Melnyk, Specialized Processors Automatic Design Tools-the Basis of Self-Configurable Computer and Cyber-Physical Systems, in: Proceedings of the 2019 IEEE International Conference on Advanced Trends in Information Theory, ATIT 2019, pp. 326-335. doi:10.1109/ATIT49449.2019.9030481

[44] J. Drozd, A. Drozd, M. Al-dhabi, A resource approach to on-line testing of computing circuits, in: Proceedings of the IEEE East-West Design & Test Symposium, Batumi, Georgia, 2015, pp. 276-281. doi: 10.1109/EWDTS.2015.7493122.

[45] M. Drozd, A. Drozd, "Safety-Related Instrumentation and Control Systems and a Problem of the Hidden Faults," The 10th International Conference on Digital Technologies 2014, Zhilina, Slovak Republic, 2014, pp. 137–140. DOI: 10.1109/DT.2014.6868692

[46] J. Drozd, A. Drozd, S. Antoshchuk, A. Kushnerov, V. Nikul, "Effectiveness of Matrix and Pipeline FPGA-Based Arithmetic Components of Safety-Related Systems," The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw, Poland, 2015, pp. 785–789. DOI: 10.1109/IDAACS.2015.7341410

ДОДАТОК В
ПРЕЗЕНТАЦІЯ ДОПОВІДІ

Самоорганізована розподілена система виявлення аномалій в комп'ютерних системах на основі методу головних компонент

Савенко Б. О.

Науковий керівник
к.т.н., доцент Нічепорук А.О.

Хмельницький
2021

Актуальність роботи.

Дослідження зловмисних проявів в корпоративних та локальних мережах можуть бути проведені з використанням апарату математичної статистики. В корпоративних та локальних мережах підприємств, організацій та установ може перебувати велика кількість комп'ютерів і для дослідження процесів, які протікають в них, в тому числі і зловмисних, потрібні ефективні методи та відповідні засоби опрацювання отриманих даних про події. Ефективність протидії зловмисним проявам досягається за рахунок комплексного підходу орієнтованого на інтеграцію методів виявлення та систем, в яких вони реалізовані. Для зловмисників такі підходи суттєво ускладнюють досягнення результативності.

В роботі пропонується використання самоорганізованих розподілених систем, розроблених згідно принципів централізації та самоорганізації, для виявлення аномалій у комп'ютерних системах.

Метою роботи є покращення ефективності виявлення аномалій в комп'ютерних системах при використанні самоорганізованих розподілених систем.

Об'єкт дослідження – процес функціонування самоорганізованих розподілених систем виявлення аномалій в комп'ютерних системах.

Предмет дослідження – методи і засоби створення самоорганізованих розподілених систем виявлення аномалій в комп'ютерних системах.

Методи дослідження. Для досягнення поставлених задач використано основні положення:

1) теорії розподілених систем, на основі якої можуть бути розроблені програмні та апаратно-програмні засоби;

2) теорій множин, графів та штучного інтелекту;

4) методи головних компонент для виявлення аномалій;

5) теорії комп'ютерних мереж для організації функціонування розподіленої системи.

Наукова новизна одержаних результатів полягає в наступному:

1) удосконалено архітектуру розподіленої системи виявлення аномалій в комп'ютерних системах, в якій синтезовано вимоги самоорганізованості, централізованості, розподіленості, багаторівневості, і на відміну від відомих рішень, дало змогу удосконалити її внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії;

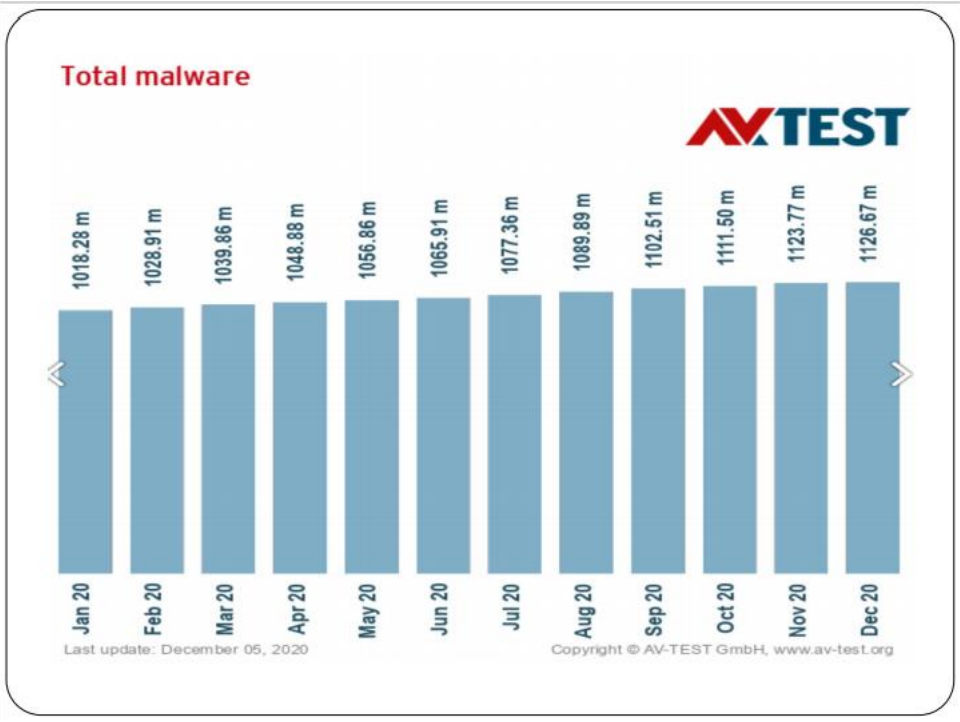
2) розроблено новий метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп'ютерних атак;

3) удосконалено метод виявлення аномалії згідно методу головних компонент в комп'ютерних системах в мережі, який надав змогу застосовувати його не до однієї комп'ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп'ютерних системах в мережі, що на відміну від відомих рішень, при застосуванні надав змогу скоротити обсяг даних і відповідно прискорити їх обмін між компонентами системи.

5

Практичне значення одержаних результатів. У результаті виконаного дослідження розроблено архітектуру і компоненти розподіленої системи виявлення аномалій в комп'ютерних системах, в якій синтезовано вимоги самоорганізованості, централізованості, розподіленості, багаторівневості та на її основі створено самоорганізовану розподілену систему. Здійснена розробка методики оцінки ефективності запропонованих рішень для розробленої самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах, яка була застосована до неї для підтвердження можливості реалізації запропонованих рішень. Проведені експериментальні дослідження з розробленою реалізацією самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах підтвердили ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп'ютерній мережі.

Теоретичні та практичні результати роботи впроваджено при виконанні науково-дослідних робіт, які виконувались в Хмельницькому національному університеті.



Постановка задачі дослідження

1) дослідити особливості прояву аномалії в комп'ютерних системах за умов функціонування зловмисного програмного забезпечення і здійснення комп'ютерних атак в локальних комп'ютерних мережах та проаналізувати сучасні методи виявлення аномалії, їх особливості та методи створення і архітектури розподілених систем;

2) удосконалити модель архітектури розподіленої системи виявлення аномалії в комп'ютерних системах, в якій синтезувати вимоги розподіленості, централізованості та самоорганізованості, для створення на її основі розподілених систем та їх компонентів, що функціонуватимуть під керівництвом одного центру розподіленого між різними компонентами і самостійно прийматимуть рішення про наявність аномалії;

3) розробити метод підтримки цілісності самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах для підтримки її цілісності, на основі якого система змогла б самостійно змінювати свою архітектуру без втручання користувача, а також визначати стратегію своєї подальшої роботи;

4) удосконалити метод централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент для зменшення розмірності з моменту отримання та надсилання даних в центр прийняття рішень системи;

5) розробити програмне забезпечення самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах для підтвердження можливості практичного створення таких систем згідно запропонованих результатів роботи та використання в експериментальних дослідженнях для порівняння з відомими системами виявлення.

Архітектура самоорганізованої розподіленої системи

Синтез наступних характерних властивостей, методів та функційних можливостей:

1) централізованість (єдиний центр прийняття рішень) у прийнятті рішень в системі;

2) наявність рівнів ієрархії в питанні прийняття рішень в розподілених частинах центру в усіх компонентах системи у вузлах мережі;

3) розподілення компонентів системи у різних вузлах в мережі;

4) самоорганізованість системи при прийнятті рішень про подальші кроки в роботі системи та її компонентів;

5) динамічне формування системи в процесі її функціонування, як під час початкового формування, так і в процесі тривалого використання;

6) мережний протокол для взаємодії компонентів проєктованої системи;

7) динамічне формування архітектури системи в процесі її функціонування з обов'язкової компоненти, в якій частина центру верхнього рівня ієрархії, та частини компонентів системи, необов'язково всіх компонентів;

8) функціонування хостових компонентів системи окремо за відсутності центру системи і виконання ними повноцінних дій з виявлення аномальних проявів засобами імплементованих функцій;

9) імплементація методів виявлення аномалій в комп'ютерних системах в проєктовану систему.

Множина компонентів самоорганізованої розподіленої системи

$$M_{SDS} = \{M_{SDS,0}, M_{SDS,1}, M_{SDS,2}, \dots, M_{SDS,N}\}, \quad (2.1)$$

де N – кількість компонентів самоорганізованої розподіленої системи без врахування компоненти, яка містить центр.

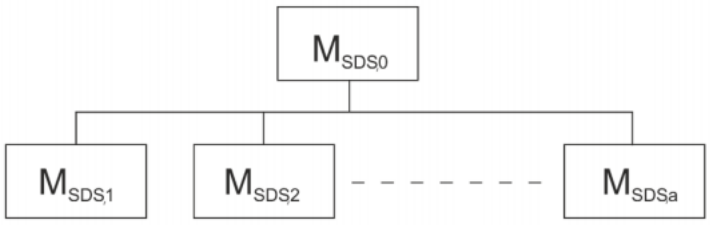
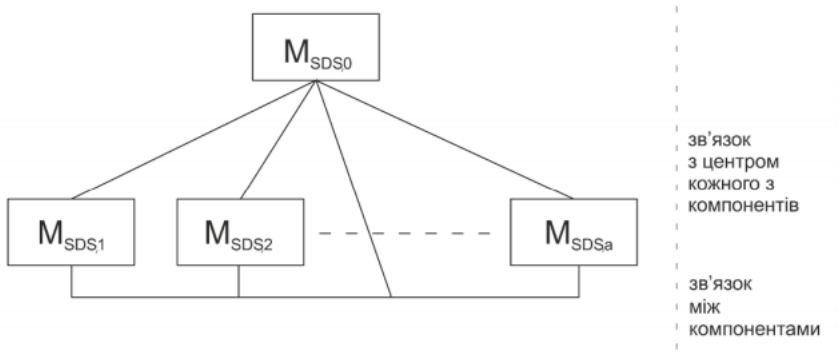
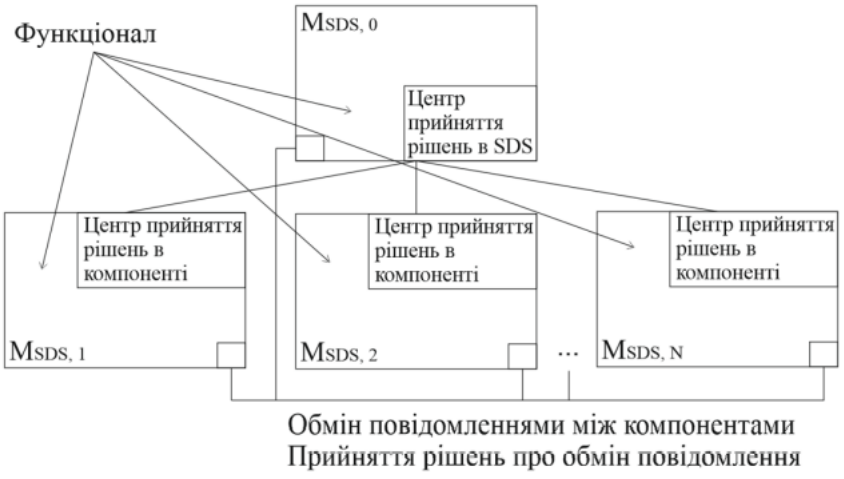


Рисунок 2.1 – Архітектура самоорганізованої розподіленої системи

Архітектура самоорганізованої розподіленої системи з відображенням центру системи в якості компоненти



Місця центру прийняття рішень в архітектурі самоорганізованої розподіленої системи



Метод підтримки цілісності самоорганізованої розподіленої системи

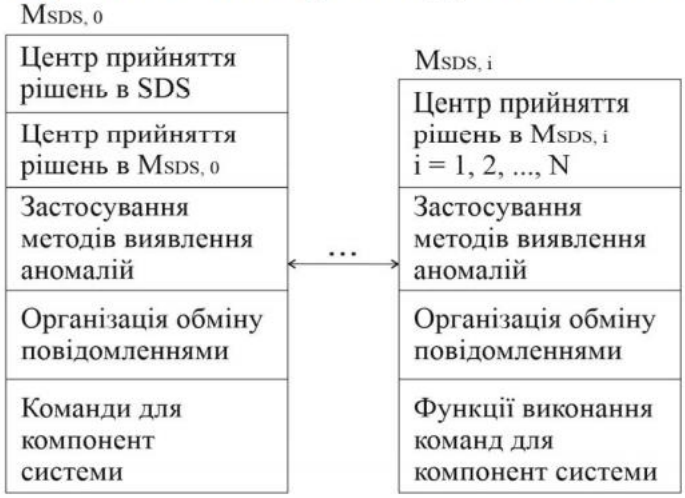


Рис. 2.4 – Архітектура компонент системи та зв'язків між ними

Метод підтримки цілісності самоорганізованої розподіленої системи

- крок 1: виконання $(c_{SDS,i}, c_{SDS,0}, 10)$ передачі повідомлення з центру нижнього рівня до центру верхнього рівня, $c_{SDS,0} \in M_{SDS,0}$ для $i = 1, 2, \dots, N$ з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи;

- крок 2: виконання $(c_{SDS,0}, c_{SDS,i}, 1)$ для передачі команди з центру верхнього рівня до центрів системи в компонентах, які знаходяться на нижньому рівні, і отримання підтвердження про отримання команди;

- крок 3: виконання команди в компоненті з центром нижчого рівня і надсилання звіту компоненті системи з центром вищого рівня;

- крок 4: виконання $(c_{SDS,i}, c_{SDS,0}, 2)$ та виконання $(c_{SDS,i}, c_{SDS,j}, 4)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів верхнього та нижнього рівнів для здійснення їх обробки;

- крок 5: виконання $(c_{SDS,i}, c_{SDS,0}, 3)$ та виконання $(c_{SDS,i}, c_{SDS,j}, 5)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про результати обробки аномалії до центрів верхнього та нижнього рівнів для здійснення їх обробки;

Метод підтримки цілісності самоорганізованої розподіленої системи

- крок 6: виконання $(c_{SDS,i}, c_{SDS,j}, 6)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів нижнього рівнів для здійснення їх обробки за умови відсутності центру верхнього рівня;

- крок 7: виконання $(c_{SDS,i}, c_{SDS,j}, 7)$ для надсилання повідомлення з центру нижнього рівня, в якому міститься інформація про результати обробки аномалії до центрів нижнього рівнів для здійснення їх обробки за умови відсутності центру верхнього рівня;

- крок 8: виконання $(c_{SDS,0}, c_{SDS,j}, 8)$ для надсилання повідомлення з центру верхнього рівня, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центрів нижнього рівнів для здійснення їх обробки;

- крок 9: виконання $(c_{SDS,0}, c_{SDS,j}, 9)$ для надсилання повідомлення з центру верхнього рівня, в якому міститься інформація про результати обробки аномалії до центрів нижнього рівнів для здійснення їх обробки;

- крок 10: виконання $(c_{SDS,0}, c_{SDS,j}, 11)$ для надсилання повідомлення з центру верхнього рівня всім активним компонентам з метою повідомлення про ввімкнення комп'ютерної станції в мережі і успішний запуск програмного забезпечення j-тої компоненти системи, тобто повідомлення про готовність до початку роботи як частини системи, в якій функціонує центр системи;

- крок 11: виконання $(c_{SDS,i}, c_{SDS,0}, 12)$ для передачі повідомлення з центру нижнього рівня до центру верхнього рівня з метою повідомлення про коректне вимкнення комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі;

Метод підтримки цілісності самоорганізованої розподіленої системи

- крок 12: виконання $(c_{SDS,0}, c_{SDS,i}, 13)$ для передачі повідомлення з центру верхнього рівня до центру нижнього рівня з метою повідомлення про вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j -тої компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі, тобто повідомлення всім решті активним компонентам системи про наявну архітектуру системи;

- крок 13: виконання $(c_{SDS,j}, c_{SDS,i}, 14)$ для передачі повідомлення з центру нижнього рівня до решти активних центрів нижнього рівня з метою повідомлення про коректне вимкнення j -тої комп'ютерної станції в мережі і успішне завершення роботи програмного забезпечення j-тої компоненти системи із збереженням характерних ознак профілю комп'ютерної станції в мережі;

- крок 14: виконання $(c_{SDS,0}, c_{SDS,i}, 15)$ для передачі повідомлення з центру верхнього рівня до центру нижнього рівня з метою повідомлення про проблеми в j - тій комп'ютерній станції в мережі і надсилання їй команди з вимогою блокування роботи програмного забезпечення в ній і примусового завершення роботи програмного забезпечення j-тої компоненти системи із збереженнях характерних ознак профілю комп'ютерної станції в мережі, тобто повідомлення всім решті активним компонентам системи про зміну наявної архітектури системи пов'язаної з вилученням j-тої компоненти системи;

- крок 15: виконання $(c_{SDS,j}, c_{SDS,i}, 16)$ для передачі повідомлення від центрів всіх рівнів до решти центрів всіх рівнів з метою повідомлення про архітектуру сформованої розподіленої системи, в яку увійшли всі початково задані компоненти і їх готовність до виконання заданих функцій чи продовження роботи.

Метод підтримки цілісності самоорганізованої розподіленої системи

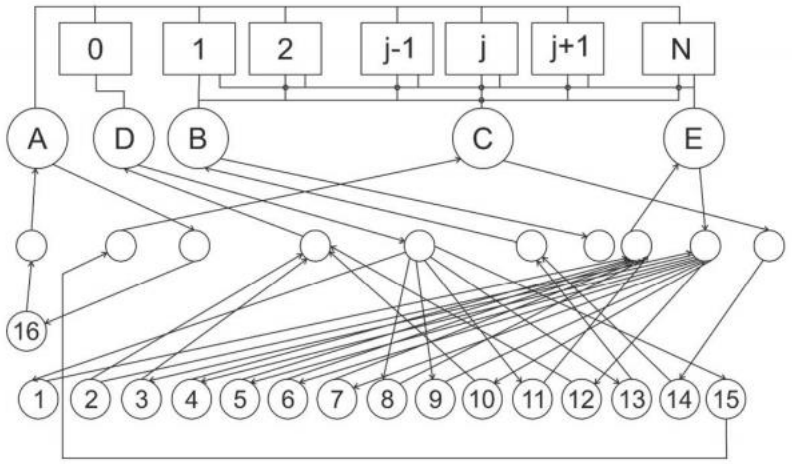


Рисунок 2.5 - Граф з дугами переходів

Метод головних компонент для виявлення аномалій в комп'ютерних системах

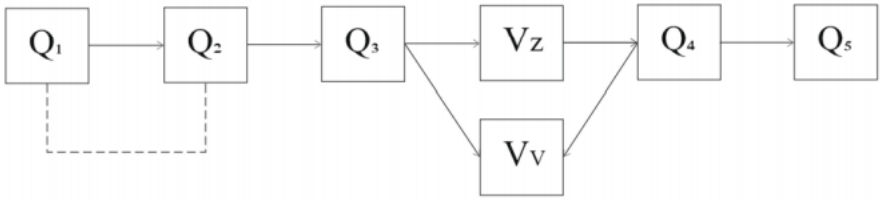


Рисунок 3.1 – Схема зв'язку кроків в методі головних компонент

Метод головних компонент для виявлення аномалій в комп'ютерних системах

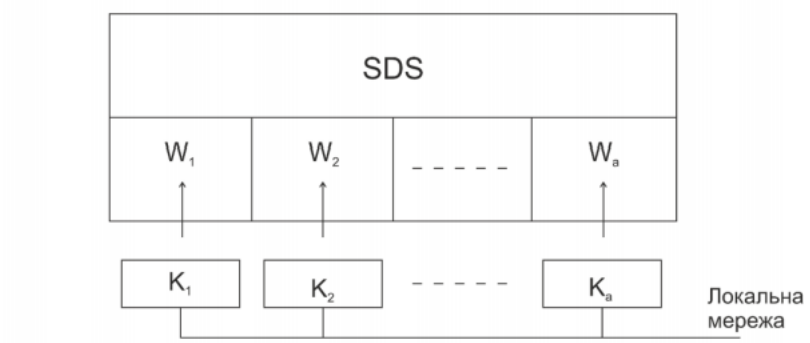
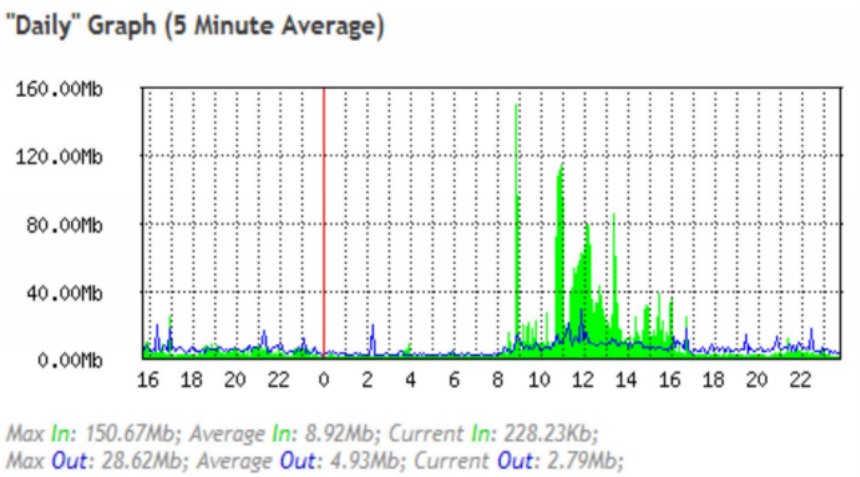
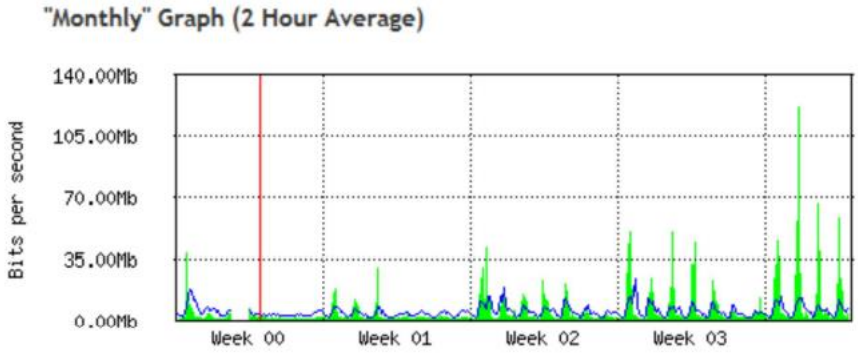


Рисунок 3.2 - Матриці W_j в архітектурі самоорганізованої розподіленої системи

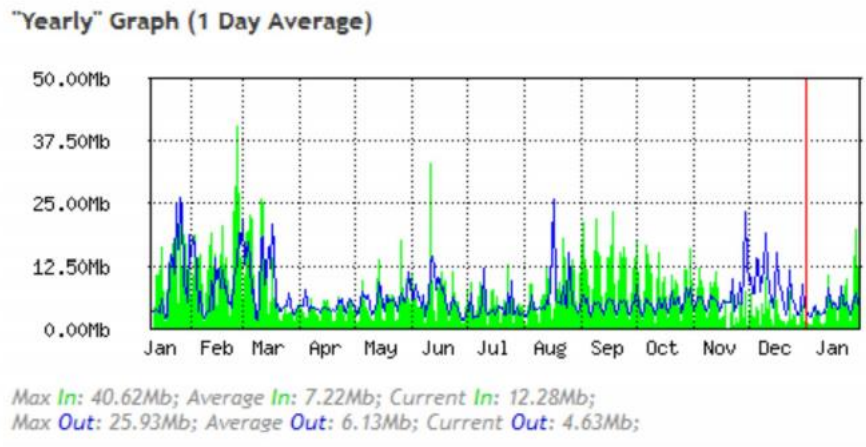
Удосконалення методу централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент



Удосконалення методу централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент



Удосконалення методу централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент



Удосконалення методу централізованого виявлення розподілених аномалій за алгоритмом пошуку головних компонент

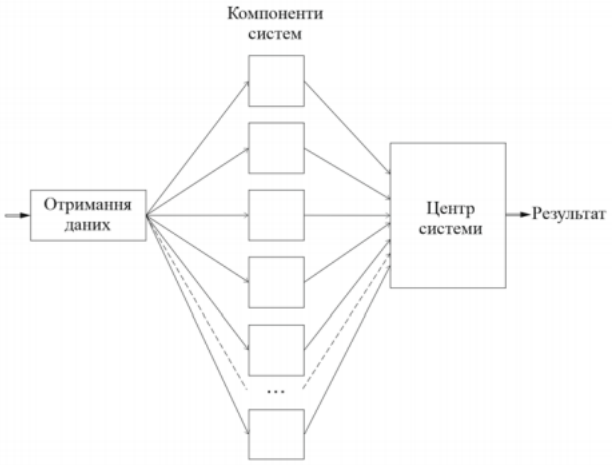


Рисунок 3.4 - Схема основних кроків удосконаленого методу

Ефективність застосування самоорганізованої розподіленої системи виявлення аномалії в комп'ютерних системах

$$K_1 = f\left(t_1^{3,2,1}, t_1^{3,2,2}\right) = \frac{\frac{t_1^{3,2,2}}{t_1^{3,2,1} - t_1^{3,2,2}} + \frac{t_1^{3,2,2}}{t_1^{3,2,1}}}{2} = 2 \cdot \frac{t_1^{3,2,1} \cdot t_1^{3,2,2} - t_1^{3,2,2} \cdot t_1^{3,2,2}}{t_1^{3,2,1} \cdot (t_1^{3,2,1} - t_1^{3,2,2})},$$

$$K_1 \to 0, \text{при } \min\left(t_1^{3,2,2}\right), \max\left(t_1^{3,2,1}\right).$$

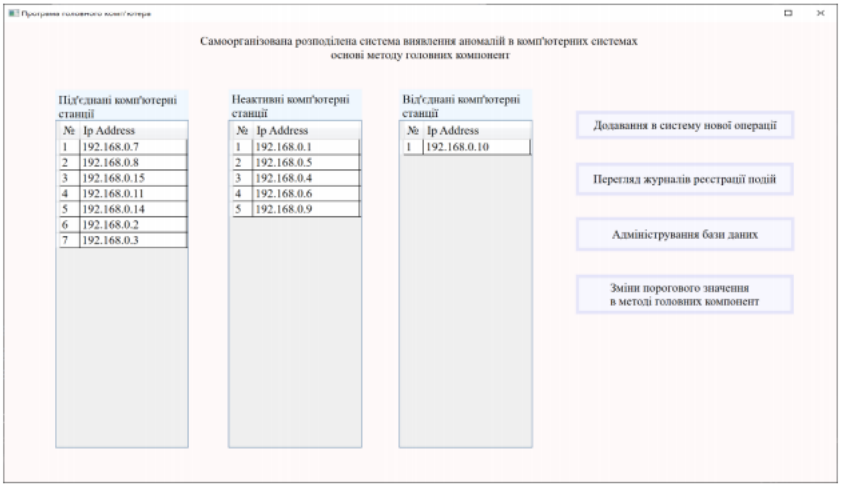
$$K_{2,j} = \frac{r_{1,j}^1 + r_{1,j}^2}{2}.$$

$$K_{2,j} \to 0, \text{при } \min\left(t_{1,j}^{3,2,2}\right), \max\left(t_{1,j}^{3,2,1}\right).$$

Опис основних функцій розробленого програмного забезпечення

№ з/п	Назва функції	Опис функції
1	2	3
1	SendCommandToComponents(int commandId)	Функція здійснює передачу команди з центру верхнього рівня, який приймаємо таким позначенням як $c_{SDS,0}$, $c_{SDS,0} \in M_{SDS,0}$, до центрів системи в компонентах, які знаходяться на нижньому рівні, тобто до центрів з позначенням $c_{SDS\,i}$, $c_{SDS\,i} \in M_{SDS\,i}$, де $i = 1, 2, \dots, N$.
2	SendDetectedPotentialAnomaliesToMainCenter()	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про можливі аномалії тобто зібрані характерні ознаки, які потребують обробки, до центру верхнього рівня $c_{SDS,0}$ для здійснення їх обробки.
3	SendAnomalyProcessingResultToMainCenter(int anomalyId)	Функція здійснює передачу повідомлення з центру нижнього рівня, який позначено $c_{SDS,i}$, $c_{SDS,i} \in M_{SDS,i}$ і де $i = 1, 2, \dots, N$, в якому міститься інформація про результати обробки аномалії в цій компоненті до центру верхнього рівня $c_{SDS,0}$.

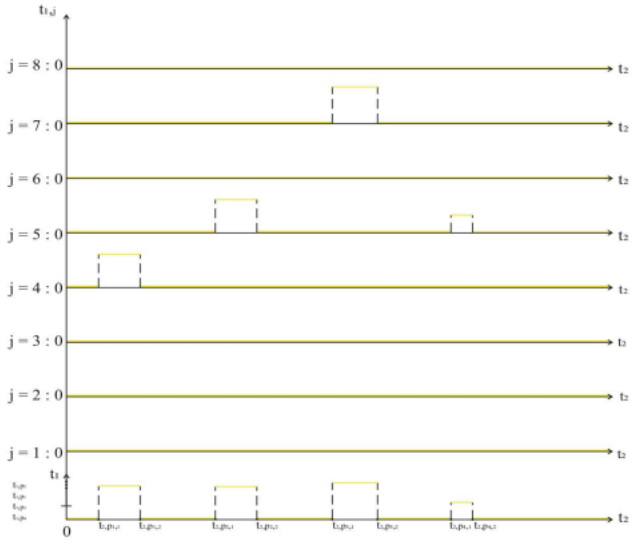
Віконна форма інтерфейсу розробленої розподіленої системи



Експериментальні дослідження з використання самоорганізованої розподіленої системи

1	Час комунікації між окремими компонентами	1,42 – 2,61	1,41 – 2,56
2	Час комунікації між компонентами системи і компонентою, в якій розміщено центр прийняття рішень вищого рівня ієрархії	1,81 – 2,87	1,67 – 2,23
3	Час комунікації між компонентами в залежності від кількості активних компонентів, які формують систему. Випадки 3.1. Кількість компонент 2-4. 3.2. Кількість компонент 5-8.	 1,41 – 2,51 1,83 – 2,81	 1,27 – 2,39 1,72 – 2,43
4	Кількість переміщених пакетів для досліджуваної події 1	7546	5788
5	Кількість переміщених пакетів для досліджуваної події 2	12458	8386

Експериментальні дослідження з використання самоорганізованої розподіленої системи



Експериментальні дослідження з використання самоорганізованої розподіленої системи

№ з./п.	Час обслуговування компонентів системи користувачем або системним адміністратором був суттєво меншим часу роботи всієї системи	Час обслуговування компонентів системи користувачем або системним адміністратором був суттєво більшим часу роботи всієї системи
	Випадок 1	Випадок 2
K_1	0,00987567	0,04873418
K_2	0,00986972	0,04873326

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено самоорганізовану розподілену систему виявлення аномалій в комп'ютерних системах згідно методу головних компонент для покращення ефективності виявлення зловмисного програмного забезпечення та комп'ютерних атак.

При цьому отримано такі основні результати:

1. Встановлено, що виявлення зловмисного програмного забезпечення та комп'ютерних атак у локальних комп'ютерних мережах згідно досліджених методів та засобів виявлення може бути реалізовано методами виявлення аномалії в комп'ютерних системах та створенням розподілених систем виявлення аномалії.

2. Удосконалено архітектуру самоорганізованої розподіленої системи, в якій на відміну від відомих рішень, удосконалено внутрішню організацію взаємодії частин центру системи між різними рівнями ієрархії та в залежності від активності компонент системи в певний час, основою для якої став розподіл центру прийняття рішень в системі між її компонентами з поділом центру між верхнім та нижніми рівнями ієрархії. Результатом так спроєктованої архітектури самоорганізованої розподіленої системи є можливість нарощувати її функціонал за рахунок наповнення імплементованими методами з виявлення аномалій в комп'ютерних системах. Система спроєктована таким чином, що її компоненти можуть обмінюватись результатами обробки аномалій та виявленими їх джерелами.

ВИСНОВКИ

3. Розроблений метод підтримки цілісності архітектури самоорганізованої розподіленої системи в локальних комп'ютерних мережах, який враховує стани компонентів системи, переходи між компонентами і визначає подальші кроки системи, що надало змогу будувати розподілені системи з єдиним центром прийняття рішень, які стануть самоорганізованими і можуть приймати рішення про свої подальші кроки в залежності від впливів зловмисного програмного забезпечення і комп'ютерних атак.

4. Удосконалення методу виявлення аномалії згідно методу головних компонент в комп'ютерних системах в мережі надало змогу застосовувати його не до однієї комп'ютерної станції, а до групи станцій, в яких встановлена самоорганізована розподілена система виявлення аномалій в комп'ютерних системах в мережі. Його застосування надало змогу скоротити обсяг даних і відповідно прискорити їх обмін між компонентами системи.

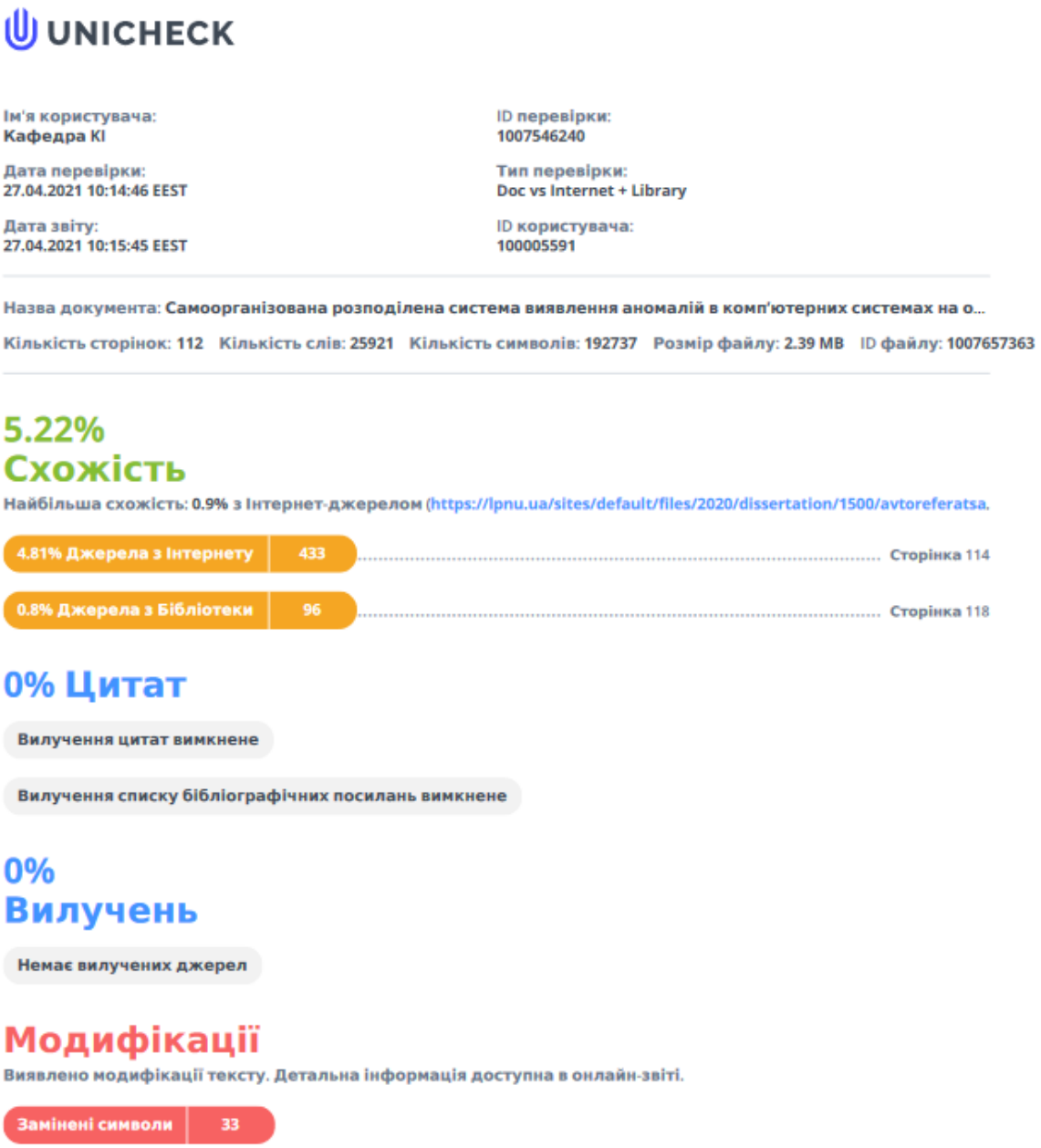
5. Здійснена розробка методики оцінки ефективності запропонованих рішень для розробленої самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах. Розроблено програмне забезпечення для забезпечення функціонування самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах для підтвердження можливості реалізації запропонованих рішень. Проведені експериментальні дослідження з розробленою реалізацією самоорганізованої розподіленої системи виявлення аномалій в комп'ютерних системах згідно отриманих коефіцієнтів підтвердили ефективність запропонованих рішень і розробленої розподіленої системи щодо її функціонування в комп'ютерній мережі.

За темою дипломної роботи магістра опублікована одна стаття у фаховому науковому виданні, що входить до рекомендованого переліку МОН України, в якому можуть публікуватись результати наукових досліджень на здобуття наукових ступенів кандидата та доктора наук (категорія Б), а також в трьох публікаціях матеріалів конференції, які індексуються в наукометричній базі Scopus.

31

Доповідь закінчено
Дякую за увагу!

Результати перевірки на плагіат:



UNICHECK

Ім'я користувача: Кафедра КІ
ID перевірки: 1007546240
Дата перевірки: 27.04.2021 10:14:46 EEST
Тип перевірки: Doc vs Internet + Library
Дата звіту: 27.04.2021 10:15:45 EEST
ID користувача: 100005591

Назва документа: Самоорганізована розподілена система виявлення аномалій в комп'ютерних системах на о...

Кількість сторінок: 112 Кількість слів: 25921 Кількість символів: 192737 Розмір файлу: 2.39 MB ID файлу: 1007657363

5.22%
Схожість

Найбільша схожість: 0.9% з Інтернет-джерелом (https://lpnu.ua/sites/default/files/2020/dissertation/1500/avtoreferatsa.

4.81% Джерела з Інтернету 433 ..... Сторінка 114

0.8% Джерела з Бібліотеки 96 ..... Сторінка 118

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 33

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 1.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 9%**

ID: 89439 Название: Самоорганізована розподілена система виявлення аномалій в комп'ютерних системах на основі методу головних компонент Добавлено в БД: 2021-04-26 Авторы: Савенко Б.О. Руководители: Нічепорук А.О. Консультанты: Оппоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	174085	1052	3196 (2%)	41 (4%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник студент групи КІ2м-19-1 Савенко Б. О.

Тема Самоорганізована розподілена система виявлення аномалій в комп'ютерних системах на основі методу головних компонент

Спеціальність 123 – Комп'ютерна інженерія

Обсяг дипломної роботи:

Кількість листів креслень 0; кількість сторінок записки 106

1.Короткий зміст ДР та прийнятих рішень Представлена робота присвячена актуальній темі в області виявлення аномалій в комп'ютерних системах з використанням розподілених систем і складається з наступних розділів: вступ, аналіз предметної області та постановка задачі, архітектура розподіленої системи, метод головних компонент, ефективність запропонованих рішень, реалізація системи та експерименти, висновки, додатки.

2. Висновок про відповідність ДР поставленому завданню Кваліфікаційна робота виконана у відповідності з виданим завданням із дотриманням всіх встановлених вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі здійснено детальний аналіз предметної області, було розглянуто розподілені системи, їх переваги та недоліки, особливості і вимоги при розробці, використання методів виявлення аномалій в наукових задачах з предметної області з комп'ютерної інженерії, згідно проведеного аналізу було сформулюовано актуальність роботи і визначено вимоги для створюваної системи. В другому розділі згідно досліджених джерел було розроблено архітектуру розподіленої системи, її апаратну та програмну складову, детально розглянуто всі компоненти системи і їх взаємодія, а також крім удосконалення її архітектури розроблено метод підтримки цілісності її архітектури. В третьому розділі було удосконалено метод виявлення аномалій в комп'ютерних системах згідно методу головних компонент та його використання у розподілених системах для виявленні аномалій, описано інтеграцію в існуючу систему. У четвертому розділі представлена методика оцінювання ефективності застосування розподілених систем згідно запропонованих рішень щодо архітектури та методу підтримки цілісності її архітектури, описана розроблена розподілена система, наведено результати експериментальних досліджень з розробленою системою та проаналізовано результати.

4. Позитивні сторони роботи До позитивних сторін роботи слід віднести актуальність напрямку дослідження, отримані наукові і практичні результати з предметної області з комп'ютерної інженерії, розроблену методику розрахунку ефективності запропонованих рішень, реалізацію запропонованих рішень та експериментальні дослідження з розробленою системою.

5. Негативні сторони роботи недостатньо деталізовано представлення розподіленої системи

6. Оцінка графічного оформлення та пояснювальної записки роботи Матеріали кваліфікаційної роботи є структурованими у чіткій та логічній формі та відображають послідовність виконання поставлених завдань.

7. Відгук про роботу в цілому Зміст представленої роботи в повній мірі розкриває обрану тему. Дослідження, проведені в матеріалах є достатньо аргументованими.

8. Інші зауваження

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

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

"30" 04 2021 р.

(підпис)

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

Савенка Богдана Олеговича
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.

5.05.2021
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ 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: «Самоорганізована розподілена система виявлення аномалій в комп'ютерних системах на основі методу головних компонент»

Автор: Савенко Богдан Олегович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Нічепорук Андрій Олександрович, к.т.н, доцент

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) в тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень в бланках (титулка, бланк завдання, в структурі підрозділів ВСТУПУ) та в назвах публікацій джерел посилання;
2) найбільшу схожість встановлено з одним документом і становить вона 0,9 відсотка в частині загальноприйнятої термінології;
3) збігів та ідентичності в тексті кваліфікаційної роботи немає, наявна лише схожість.

Сумарний обсяг всіх запозичень, визначений системою виявлення схожості, складає 5.22% і адресується до 433 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ А. О. Нічепорук

Професор кафедри КІСП _____ С. М. Лисенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко