Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO)»

Виконав: ст. 2-го курсу, групи 1КІТС-22м
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Ощепков В.С.
(прізвище та ініціали)

Керівник: к.т.н., доц., доцент каф. МБІС
Сачанюк-Кавецька Н.В.
(прізвище та ініціали)
«04» грудня 2023 р.

Опонент: к.т.н., доц., доцент каф. ОТ
к.т.н., доцент, доцент каф. ОТ
Крупельницький Л.В
(прізвище та ініціали)
«04» грудня 2023 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
“04” грудня 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітньо-професійна програма – Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ

Голова секції УБ, кафедра МБІС

_____ **Юрій ЯРЕМЧУК**

“20” вересня 2023 р.

З А В Д А Н Н Я

на магістерську кваліфікаційну роботу студенту

Ощепкову Віктору Сергійовичу

1. Тема роботи «Підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO)».

Керівник роботи к.т.н., доц., доцент каф. МБІС Сачанюк-Кавецька Наталія Василівна, затверджені наказом вищого навчального закладу від “18” вересня 2023 року № 247

2. Строк подання студентом роботи за тиждень до захисту.

3. Вихідні дані до роботи: нормативно-правова база, монографії та актуальні наукові статті за темою роботи, Інтернет-ресурси, спеціалізована література, стандарти, існуюче ПЗ.

4. Зміст текстової частини: в першому розділі проаналізувати існуючі методи та засоби захисту корпоративних даних серверів; в другому розділі описати

процеси розроблення інформаційної технології, формування набору даних д
інтелектуального аналізу та визначити модель, що демонстр
найефективніший підхід для вдосконалення генетичного алгоритму ро
часток; в третьому розділі здійснити програмну реалізацію розробки та анал
результатів; в четвертому розділі проаналізувати економічну ефективніст
розробленого програмного забезпечення.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкови
креслень): у першому розділі наведено 3 табл.; у другому розділі наведено
табл.; у третьому розділі наведено 8 рис.; у четвертому розділі наведено 8 табл.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	к.т.н., доц., доцент каф. МБІС Сачанюк-Кавецька Н. В.		
Економічна частина	к.т.н., доц., доц. каф. ЕПВМ Причепа І.В.		

7. Дата видачі завдання 20 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1.	Визначення напрямку МКР, формулювання теми	20.09.2023	25.09.2023	
2.	Аналіз предметної області обраної теми	26.09.2023	30.10.2023	
3.	Розроблення алгоритму роботи	03.10.2023	17.10.2023	
4.	Робота над МКР на основі вибраної теми	18.10.2023	10.11.2023	
5.	Робота над економічною частиною	11.11.2023	23.11.2023	
6.	Попередній захист МКР	24.11.2023	25.11.2023	
7.	Виправлення, уточнення, коригування роботи	26.11.2023	30.11.2023	
8.	Захист МКР	15.12.2023	15.12.2023	

Студент _____ Ощепков В.С.

Керівник роботи _____ Сачанюк-Кавецька Н.В.

АНОТАЦІЯ

УДК: 004.6.056.5

Ощепков В.С. «Підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO)». Магістерська кваліфікаційна робота зі спеціальності 125 – «Кібербезпека», освітня програма «Управління інформаційною безпекою». Вінниця: ВНТУ, 2023. 108 с.

Стаття включає в себе 111 стор., 8 рис., 12 таблиць, 8 додатків, 23 джерела.

У магістерській кваліфікаційній роботі представлено підвищення захищеності копроративних даних шляхом оброблення потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO).

Сучасний світ вимагає постійних зусиль у сфері захисту інформації та даних в різних сегментах життя. З ростом кількості загроз інформаційній безпеці, необхідність розробки нових та ефективних методів захисту стає все більш актуальною. У даній кваліфікаційній роботі досліджується підвищення рівня захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO).

Робота складається з трьох основних розділів.

У першому розділі аналізується стан сучасних методів захисту інформації, а також основні підходи до використання генетичних алгоритмів та роїв часток у цій сфері. Досліджуються переваги та недоліки існуючих рішень.

У другому розділі розглянуто генетичний алгоритм рою часток (PSO), його переваги та недоліки у при використанні у сфері кібербезпеки та приклади застосування, порівняння з іншими генетичними алгоритмами та

можливі варіанти вдосконалення та поліпшення використання даного алгоритму.

Третій розділ присвячений розробці вдосконаленого генетичного алгоритму рою часток, який базується на інтеграції генетичного алгоритму та алгоритму рою часток з метою підвищення ефективності та точності процесу захисту інформації. Також аналізується результат експериментів та тестувань розробленого методу на різних наборах даних і в різних сценаріях. Здійснюється порівняння з існуючими рішеннями з метою визначення переваг та обмежень розробленого методу.

Четвертий розділ присвячений економічній частині та прорахунку вартості розробки системи, порівняння з аналогами та прорахунов цілеспрямованості використання.

Ключові слова: захист інформації, генетичний алгоритм, рой часток, PSO, кібербезпека.

Abstract

UDC: 004.6.056.5

Oshopkov V.S. "Increasing the protection of corporate data by processing server data flows based on an advanced genetic algorithm swarm of particles (PSO)." Master's qualification work in specialty 125 - "Cybersecurity", educational program "Information security management". Vinnitsa: VNTU, 2023. 108 p.Specialty: This thesis consists of 111 pages,8 figures, 12 tables, and 8 appendices. It references 23 information sources.

The modern world demands constant efforts to safeguard information and data in various aspects of life. With the increasing number of threats to information security, the need for developing new and effective protection methods becomes more critical. This thesis investigates the enhancement of the security level of corporate data by processing data streams on servers based on an improved Particle Swarm Optimization (PSO) algorithm.

The thesis is structured into three main sections:

Chapter 1 analyzes the current state of information security methods and the primary approaches to using genetic algorithms and particle swarms in this field. It examines the advantages and disadvantages of existing solutions.

Chapter 2 delves into the Particle Swarm Optimization (PSO) algorithm, its benefits, and drawbacks when applied to cybersecurity. It provides practical examples and compares PSO with other genetic algorithms. The chapter explores potential enhancements and improvements to the algorithm's use.

Chapter 3 focuses on the development of an enhanced Particle Swarm Optimization algorithm, integrating genetic algorithms and particle swarm algorithms to improve information security's effectiveness and precision. The results of experiments and tests of the developed method in various datasets and scenarios

are also analyzed, comparing it with existing solutions to identify its advantages and limitations.

Chapter 4 covers the economic aspects, including the cost estimation of system development, comparisons with analogs, and the feasibility of its utilization.

Keywords: Information security, genetic algorithm, particle swarm, PSO, cybersecurity.

ЗМІСТ

ВСТУП

У світі, який позначений невпинним ростом кількості кіберзагроз, забезпечення безпеки корпоративних даних стає насущним завданням. Необхідність розробки інноваційних та надійних методів захисту ніколи ще не була такою актуальною. Дана кваліфікаційній робота завжди дійде до нового підходу до підвищення безпеки корпоративних даних. Вона глибоко досліджує складну область обробки потоків даних на серверах і пропонує інноваційне рішення, засноване на поліпшеному алгоритмі рою часток (PSO).

Подолання сучасного світу, який зіштовхується з безпрецедентними викликами в галузі інформаційної безпеки, додає актуальності даному дослідженню. З кожним днем зростають масштаби та складність загроз для інформаційної безпеки. Інциденти кібербезпеки можуть призвести до серйозних наслідків, включаючи втрати даних, фінансові втрати та порушення приватності. Тому захист інформації та даних стає невідкладною обов'язковістю для організацій та окремих користувачів. У цьому контексті дослідження щодо підвищення безпеки інформації стає запитом, який має важливе значення.

Подальший розвиток роботи визначатиметься термінами «інформаційна безпека», «генетичні алгоритми», «рій часток» та «PSO», які будуть неодноразово переплітатися з основами нашого дослідження. Дане дослідження зможе надати унікальну перспективу на підвищення безпеки даних, використовуючи силу алгоритму рою часток (PSO), яка буде корисною для експертів у галузі кібербезпеки та всіх небайдужих до зміцнення безпеки даних та інформації.

Дана робота спрямована на підвищення ефективності використання алгоритму рою часток в галузі кібербезпеки. Головною галуззю використання

є вирішення питань забезпечення кібербезпеки та застосування у процесі розробки і побудови комплексів захисту та систем забезпечення кібербезпеки для підприємств та державних установ.

Актуальність. В сучасному цифровому світі зростаючий обсяг даних та зростаюча взаємодія між системами зумовили появу ери, в якій інформація є однією з найцінніших активів для бізнесу, уряду та окремих осіб. Ця залежність від цифрової інформації спровокувала виникнення важливої проблеми: як забезпечити безпеку цього надзвичайно важливого ресурсу.

У зв’язку з цим виникає необхідність створення нового підходу до підвищення захисту інформації. Таким – є генетичний алгоритм рою часток. Даний алгоритм може бути корисною для спеціалістів у галузі кібербезпеки та всіх, хто цікавиться питаннями захисту даних та інформації.

Тобто, актуальність даної роботи обумовлена покращенням методів захисту інформації, що циркулює та обробляється в різноманітних установах чи організаціях.

Мета магістерської кваліфікаційної роботи. Розробка та дослідження вдосконаленого генетичного алгоритму рою часток (PSO) з метою підвищення ефективності захисту інформації. А також – розробка рішення щодо підвищення єфективоності використання генетичного алгоритму рою часток шляхом вдосколналення його використання у сфері кібербезпеки.

В результаті досліджень отримано вдосконалене використання генетичного алгоритму рою часток у кібербезпеці.

Задачами (цілями) дослідження є:

- Огляд сучасного стану захисту інформації та методів боротьби з кіберзагрозами.
- Дослідження генетичних алгоритмів та алгоритму рою часток (PSO) в контексті їх застосування в кібербезпеці.

- Розробка вдосконаленого генетичного алгоритму рою часток для підвищення ефективності захисту інформації.
- Проведення експериментів для оцінки продуктивності та точності розробленого методу в різних сценаріях.

В результаті досліджень отримано вдосконалене використання генетичного алгоритму рою часток у кібербезпеці.

Об’єкт дослідження – вдосконалений генетичний алгоритм рою часток (PSO).

Предмет дослідження – процес підвищення захищеності корпоративних даних шляхом обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO).

Наукова новизна: вдосконалення алгоритму рою часток (PSO) шляхом реалізації налаштування коефіцієнтів для модернізації точності досліджень у реальному часі та створення системи забезпечення захисту мережі від атак зловмисників.

Призначення розробки: використання модернізованого алгоритму рою часток (PSO) у сфері кібербезпеки.

Основна сфера застосування — для вирішення питань у сфері кібербезпеки та використання під час проектування та побудови комплексів захисту та систем забезпечення кібербезпеки підприємств та державних установ.

Практична цінність: розроблено програмний продукт (модуль), який реалізує вдосконалений алгоритм рою часток (PSO), що використовується в системах забезпечення захисту мережі від атак зловмисників.

РОЗДІЛ 1. ВИКОРИСТАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ ДЛЯ ЗАХИСТУ ЗАХИСТУ КОРПОРАТИВНИХ ДАНИХ СЕРВЕРІВ.

У світі, де кіберзагрози стають все більш складними та хитромудрими, потрібні нові підходи до забезпечення кібербезпеки. Генетичні алгоритми, які базуються на еволюційних принципах, надають можливість автоматизовано оптимізувати параметри та стратегії захисту.

Сучасний цифровий світ стикається зі зростаючими загрозами кібербезпеці, що вимагають нових та інноваційних підходів до захисту інформації та інфраструктури. У цьому контексті генетичні алгоритми знайшли своє застосування як потужний інструмент для оптимізації та покращення процесів кібербезпеки. Дане дослідження розглядає використання генетичних алгоритмів у кібербезпеці, звертаючи увагу на їхню роль у виявленні загроз, аналізі та захисті від кібератак.

Генетичні алгоритми можуть бути використані для оптимізації систем виявлення вторгнень (IDS). Вони дозволяють аналізувати великі обсяги даних та вдосконалювати правила виявлення на основі знайдених закономірностей. Генетичні алгоритми допомагають підтримувати IDS на високому рівні ефективності та точності.

Генетичні алгоритми також використовуються для створення моделей поведінки кіберзагроз. Вони можуть аналізувати дані щодо раніше відомих атак та допомагати передбачити майбутні загрози. Це робить можливим попередження атак і підвищення загального рівня кібербезпеки.

1.1 Використання генетичних алгоритмів для захисту корпоративних даних серверів

Забезпечення безпеки корпоративних даних на серверах є важливим завданням для сучасного бізнес-середовища, де кіберзагрози набувають все більшої складності. Особливо це стосується використання генетичного алгоритму рою часток (PSO), який може представляти ефективний метод у забезпеченні захисту інформації.

Виявлення та обмеження кіберзагроз – це ключова складова кібербезпеки, і генетичні алгоритми можуть відігравати важливу роль в цьому процесі. Вони дозволяють системам безпеки автоматично аналізувати та реагувати на потенційні загрози.

Генетичні алгоритми в галузі кібербезпеки можуть використовуватися для вирішення різноманітних проблем та забезпечення надійної безпеки в режимі реального часу.

Генетичні алгоритми можуть бути використані для:

- Виявлення відхилень.
- Виявлення та обмеження кіберзагроз.
- Аналізу великих обсягів мережевого трафіку з метою виявлення аномалій.
- Аналіз журнальних даних.
- Підвищення точності виявлення загроз.

Оптимізація параметрів систем безпеки

Використання генетичних алгоритмів, системи безпеки можуть автоматично оптимізувати свої налаштування, адаптувати їх до змінних умов

та загроз, забезпечуючи високий рівень захисту в цифровому середовищі (див. таблицю 1.1).

Таблиця 1.1 – Використання генетичних алгоритмів

Назва	**Пояснення**
Пошук оптимальних параметрів систем безпеки	Генетичні алгоритми можуть використовуватися для автоматичного пошуку оптимальних параметрів, таких як налаштування брандмауера, детекторів вторгнень та інших захисних засобів.
Підвищення ефективності та реагування на загрози	Шляхом еволюції параметрів генетичні алгоритми можуть підвищувати ефективність захисних заходів та покращувати реагування на нові загрози.
Оптимізація систем безпеки	Генетичні алгоритми можуть бути використані для оптимізації параметрів та ресурсів систем безпеки, що є важливим для забезпечення ефективності та відповідності стандартам безпеки [14].
Параметри брандмауерів і систем виявлення вторгнень	Генетичні алгоритми можуть автоматично підбирати оптимальні параметри для брандмауерів та систем виявлення вторгнень, включаючи визначення правил фільтрації пакетів та інші налаштування.
Мінімізація ложних позитивів	Генетичні алгоритми можуть оптимізувати параметри для зниження кількості ложних позитивів в системах виявлення вторгнень, підвищуючи точність виявлення загроз [18].

Продовження таблиці 1.1

Назва	**Пояснення**
Розподіл ресурсів для захисту	Генетичні алгоритми можуть розподіляти ресурси для захисту важливих компонентів мережі або системи з високим ризиком, визначаючи оптимальне розміщення захисних засобів.
Оптимізація обробки журнальних даних	Генетичні алгоритми можуть оптимізувати обробку журнальних даних для швидкого та ефективного аналізу, що сприяє виявленню загроз в реальному часі та зберіганню історичних даних [11].
Адаптація до змінюються умов	Генетичні алгоритми можуть автоматично адаптуватися до змінюються умов та загроз, оновлюючи параметри та налаштування систем безпеки для забезпечення ефективності
Оптимізація ресурсів	Генетичні алгоритми допомагають оптимізувати використання ресурсів для захисту, розподіляючи їх для максимальної ефективності в обмежених умовах, таких як пропускна здатність і пам'ять [13].
Постійне вдосконалення та навчання	Генетичні алгоритми можуть вдосконалювати оптимізацію систем безпеки на основі реальних даних та навчання з плином часу, що сприяє підтримці високого рівня захисту.

Дана таблиця роказує, як генетичні алгоритми можуть бути використані для оптимізації параметрів та ресурсів систем безпеки з метою забезпечення їх ефективності та надійності.

Генерація криптографічних ключів

Генерація криптографічних ключів задопомогою генетичних алгоритмів представляє собою інноваційний спосіб створення сильних ключів для захисту інформації в кібербезпеці [7].

Докладний опис процесу генерації криптографічних ключів за допомогою генетичних алгоритмів описано в таблиці 1.2.

Таблиця 1.2 – Генерація криптографічних ключів

Крок	**Назва**	**Пояснення**
1	Початкова популяція	Створення початкового набору криптографічних ключів, які можуть бути випадковими або базуватися на існуючих ключах
2	Оцінка пристосованості	Оцінка кожного ключа з популяції за критеріями безпеки, такими як ентропія, стійкість до атак та інші. Відбір ключів з високою пристосованістю.
3	Схрещування і мутація	Обрання найкращих ключів для схрещування, що включає комбінування їхніх властивостей, і внесення випадкових змін (мутацій) в інші ключі.
4	Відбір нової популяції	Відбір найкращих ключів попередньої популяції для створення нового покоління. Цей процес повторюється для постійного покращення ключів.
5	Збереження найкращих ключів	Збереження найкращих ключів згідно з критеріями безпеки для майбутнього використання.
6	Застосування ключів	Використання найкращих ключів для шифрування або розшифрування даних з метою забезпечення конфіденційності та цілісності інформації.
7	Початкова популяція	Створення початкового набору криптографічних ключів, які можуть бути випадковими або базуватися на існуючих ключах.

Продовження таблиці 1.2

Крок	**Назва**	**Пояснення**
8	Оцінка пристосованості	Оцінка кожного ключа з популяції за критеріями безпеки, такими як ентропія, стійкість до атак та інші. Відбір ключів з високою пристосованістю
9	Схрещування і мутація	Обрання найкращих ключів для схрещування, що включає комбінування їхніх властивостей, і внесення випадкових змін (мутацій) в інші ключі.
10	Відбір нової популяції	Відбір найкращих ключів попередньої популяції для створення нового покоління. Цей процес повторюється для постійного покращення ключів.
11	Збереження найкращих ключів	Збереження найкращих ключів згідно з критеріями безпеки для майбутнього використання.
12	Застосування ключів	Використання найкращих ключів для шифрування або розшифрування даних з метою забезпечення конфіденційності та цілісності інформації.

Генетичні алгоритми для генерації криптографічних ключів дозволяють створювати ключі, які є дуже важкими для зламу або підбору з боку атакуючого, завдяки процесу постійної оптимізації та адаптації. Цей метод може бути особливо корисним в тих сферах, де потрібен постійний оновлюваний та високо надійний ключовий матеріал для забезпечення безпеки даних і комунікацій [17].

Переваги використання генетичних алгоритмів у створенні криптографічних ключів:

- Генерація випадкових ключів.
- Врахування криптографічних вимог.
- Автоматичне оновлення ключів.
- Відновлення ключів.
- Підпис та аутентифікація.
- Захист від криптографічних атак.
- Забезпечення конфіденційності та цілісності даних,
- Аналіз вразливостей програмного забезпечення.

Таким чином генетичні алгоритми у генерації криптографічних ключів є важливим та перспективним підходом, який має численні переваги для забезпечення кібербезпеки. Цей метод надає можливість створювати високоякісні ключі, які є важкими для зламу, і відповідають вимогам сучасних криптографічних стандартів.

Генетичні алгоритми дозволяють автоматизувати та оптимізувати процес генерації ключів, адаптувати їх до змінних умов та загроз, та підвищувати рівень безпеки інформації. Вони можуть генерувати ключі відповідно до потреб користувача та забезпечувати відповідну довжину та складність [17].

Додатково, генетичні алгоритми спрощують процес керування ключами, забезпечуючи постійне оновлення та зміну ключів для підвищення безпеки даних. Їх можна використовувати в різних сферах, включаючи захист від кіберзагроз, забезпечення конфіденційності даних та захист від несанкціонованого доступу.

Загалом, використання генетичних алгоритмів у генерації криптографічних ключів допомагає зміцнити кібербезпеку, забезпечуючи надійний та ефективний механізм створення ключового матеріалу для захисту інформації у цифровому світі.

Захист та усунення наслідків інцидентів

Після виявлення кіберінцидентів важливо вжити заходи для їх усунення та відновлення безпеки системи. Генетичні алгоритми можуть грати роль у захисті та відновленні після інцидентів задопомогою наступних заїодів, що вони реалізують для захисту системи:

- Автоматизований захист.

  Генетичні алгоритми можуть автоматично реагувати на інциденти, включаючи зупинку атак та ізоляцію компрометованих систем.
- Відновлення систем.

  Після інциденту генетичні алгоритми можуть допомагати відновлювати системи до нормального стану.
- Аналіз вразливостей та оцінка шкоди.

  Вони можуть аналізувати вразливості, які призвели до інциденту, та оцінювати шкоду.
- Пошук і видалення шкідливого програмного забезпечення.

  Генетичні алгоритми допомагають виявляти та видаляти шкідливе програмне забезпечення, яке може бути внесене в систему під час атаки.
- Виправлення вразливостей

  Вони надають рекомендації та допомагають виправити вразливості, що були використані в атаках.

- Аналіз кореня причини.

  Генетичні алгоритми можуть допомагати визначити кореневі причини інциденту та надавати рекомендації для їх усунення.
- Захист від повторних атак.

  Після інциденту генетичні алгоритми можуть допомагати вдосконалити захисні заходи та зменшити ймовірність подібних атак у майбутньому.

Генетичні алгоритми можуть допомагати в реагуванні на інциденти та відновленні безпеки після кібератак, забезпечуючи відповідність безпеці системи.

Прогнозування та попередження кіберзагроз

Використання генетичних алгоритмів активно поширюється в попереджені загроз. Задопомогою моделювання та прогнозування вони спроможні спрогнозувати де може виникнути вразливість та попередити її.

Детальний огляж можливостей генетичних алгоритмів у попереджені загроз в кібербезпеці:

- Аналіз стану безпеки.

  Генетичні алгоритми аналізують журнальні дані та активність мережі для виявлення незвичайних змін, що можуть свідчити про потенційні загрози.
- Використання статистики.

  Вони використовують статистичні методи для прогнозування можливих кіберзагроз на основі історичних даних та трендів.
- Моделювання атак.

Генетичні алгоритми можуть моделювати можливі кібератаки та визначати, які заходи безпеки найкраще захищають систему.

- Генерація попереджень.

  Вони генерують попередження та рекомендації для адміністраторів щодо можливих кіберзагроз.
- Виявлення патернів атак.

  Генетичні алгоритми виявляють патерни та характеристики кібератак, що допомагають передбачити подібні атаки в майбутньому.
- Автоматичне вдосконалення систем безпеки.

  Після прогнозування загроз генетичні алгоритми можуть автоматично вдосконалювати заходи безпеки для підвищення стійкості системи.
- Навчання на реальних даних.

  Вони навчаються на реальних даних та коригують свої моделі на основі нової інформації.

Відновлення та реагування на кіберінциденти

Відновлення та реагування на кіберінциденти є критично важливими аспектами кібербезпеки, оскільки вони допомагають відновити нормальну функціональність системи після атаки та встановити безпеку.

Генетичні алгоритми можуть бути використані для автоматизованого відновлення та реагування на інциденти, наприклад:

- Збір та аналіз даних про інцидент:

  Генетичні алгоритми можуть автоматично збирати та аналізувати дані про інцидент, включаючи журнальні дані, трафік та активність системи.

- Оцінка ризику:

  Вони оцінюють ризики та наслідки інциденту для прийняття рішень щодо відновлення.
- Відновлення системи:

  Генетичні алгоритми автоматично відновлюють систему до стану до інциденту.
- Забезпечення стійкості:

  Після відновлення вони вживають заходів для забезпечення стійкості системи та запобігання повторним атакам.
- Запобігання атак:

  Вони розробляють та вдосконалюють заходи безпеки для запобігання подібним інцидентам у майбутньому.
- Швидка реакція:

  Генетичні алгоритми реагують на інциденти в режимі реального часу, що допомагає зменшити час реакції на атаку.
- Автоматичне вдосконалення заходів безпеки:

  Вони автоматично вдосконалюють заходи безпеки на основі навчання з інцидентів.

Використання генетичних алгоритмів у кібербезпеці представляє собою перспективний інструмент для підвищення безпеки в цифровому середовищі. Цей підхід дозволяє автоматизувати та оптимізувати процеси виявлення загроз, генерації криптографічних ключів та управління параметрами систем безпеки.

Генетичні алгоритми надають можливість адаптувати системи безпеки до зростаючих загроз та складних сценаріїв, що забезпечує їхню ефективність у реальному часі. Вони можуть бути застосовані в різних галузях кібербезпеки,

включаючи виявлення вторгнень, захист від кібератак, а також генерацію криптографічних ключів для забезпечення конфіденційності та цілісності даних.

Завдяки своїй здатності до постійного вдосконалення та адаптації, генетичні алгоритми допомагають зміцнити кібербезпеку в умовах постійної зміни загроз і сучасних вимог до захисту інформації. Вони стають важливою ланкою в сучасній стратегії кібербезпеки та сприяють підвищенню рівня безпеки в цифровому світі.

1.2 Недоліки генетичних алгоритмів при у захисті корпоративних даних

Використання генетичних алгоритмів у кібербезпеці має свої переваги, але також пов'язане з рядом недоліків. В даному розділі наведено недоліки використання генетичних алгоритмів та короткий опис їх.

Висока обчислювальна складність

Генетичні алгоритми можуть бути вимогливими до обчислень, особливо при роботі з великими обсягами даних. Це може призвести до збоїв або затримок в процесі виявлення загроз [1].

Висока обчислювальна складність є одним з основних недоліків при використанні генетичних алгоритмів (ГА) у кібербезпеці. Давайте розглянемо цей недолік на прикладі.

Приклад: Виявлення атак на мережу.

Припустимо, що ми використовуємо генетичний алгоритм для виявлення атак на комп'ютерну мережу. У цьому випадку, вхідними даними

для ГА є потік мережевого трафіку, і ми хочемо виявити аномальну активність, яка може свідчити про атаку.

Недолік високої обчислювальної складності.

Обсяг даних: Мережі можуть генерувати величезні обсяги даних, і ГА можуть вимагати значних обчислювальних ресурсів для обробки цих даних. Обчислення фітнес-функцій, які визначають, наскільки аномальним є кожен пакет даних, може бути вкрай вимогливим процесом.

Велика кількість параметрів: ГА вимагають налаштування параметрів, таких як розмір популяції, ймовірність мутації та інші. Пошук оптимальних параметрів може займати значний час і ресурси.

Часовий аспект: Виявлення атак в реальному часі є критичним завданням у кібербезпеці. Висока обчислювальна складність ГА може призвести до затримок, що можуть дозволити атакам проникнути в мережу або завдати шкоду перед тим, як алгоритм реагує.

Витрати на обладнання.

Робота з великими обсягами даних та висока обчислювальна складність можуть вимагати потужних обчислювальних ресурсів, що може збільшити витрати на обладнання та інфраструктуру.

Для подолання цього недоліку, можна розглядати оптимізацію ГА, використання спеціалізованих алгоритмів та апаратного прискорення, а також обробку даних в реальному часі для мінімізації затримок у виявленні аномалій у мережі.

Залежність від початкового набору даних.

Початковий набір даних, на якому працюють генетичні алгоритми, може вплинути на результати. Неправильний або неадекватний вихідний набір даних може призвести до недостовірних результатів.

Залежність від початкового набору даних є ще одним серйозним недоліком при використанні генетичних алгоритмів (ГА) в кібербезпеці. Давайте розглянемо цей недолік на прикладі виявлення аномальної активності в мережі [12].

Приклад: Виявлення аномальної мережевої активності.

При використанні ГА для виявлення аномальної мережевої активності важливою є початкова вибірка даних. Вхідні дані включають історію мережевого трафіку, і ГА намагається виявити аномалії на основі цих даних.

Недолік залежності від початкового набору даних:

Перекошені результати: Якщо початковий набір даних містить обмежену кількість аномалій або, навпаки, багато нормальної активності, то ГА може надмірно пристосовуватися до цього набору. Це призведе до невірних результатів, де деякі аномалії можуть не бути виявлені, або нормальна активність може помилково розцінюватися як аномальна.

Недостатня репрезентація загроз: Початковий набір даних може не включати всі можливі види атак і аномалій. Якщо ГА навчається тільки на обмеженому наборі даних, він може бути неспроможним виявити нові атаки, які виникають в майбутньому.

Залежність від зовнішніх факторів: Результати ГА можуть сильно залежати від якості та репрезентації початкових даних, і це може робити алгоритм менш стійким до змін в навколишньому середовищі.

Важко попередити вплив початкових даних: Визначення, який набір даних є найкращим для навчання ГА, може бути важкою задачею. Залежно від обраного набору даних, результати можуть відрізнятися значно.

Для подолання цього недоліку важливо ретельно аналізувати та обирати початковий набір даних, використовувати множини даних з різних джерел та розглядати методи ансамблювання для зменшення впливу початкового вибору на результати ГА в кібербезпеці.

Недостатня надійність

Генетичні алгоритми можуть не завжди забезпечувати надійні результати, оскільки вони базуються на ймовірнісних операціях і можуть збільшувати ризик помилок.

Недостатня надійність є одним із недоліків при використанні генетичних алгоритмів (ГА) у кібербезпеці. Давайте розглянемо цей недолік на прикладі використання ГА для виявлення аномалій в мережевій активності.

Приклад: Виявлення аномалій в мережевій активності

При використанні ГА для виявлення аномалій в мережевій активності, головною метою є визначення нормальної активності та виявлення аномальних змін. Проте недостатня надійність може призвести до невірних або неповних результатів.

Недолік недостатньої надійності:

False Positives (хибні спрацьовування): ГА можуть помилково визнавати нормальну активність як аномальну, що призводить до false positives. Це може виводити адміністраторів із справжньою проблемою.

False Negatives (пропуск аномалій): З іншого боку, недостатня надійність може призводити до false negatives, коли аномальна активність залишається невиявленою. Це означає, що потенційно небезпечні атаки не будуть виявлені [7].

Недостатній рівень точності

Важко досягнути високого рівня точності при виявленні аномалій через залежність від ймовірнісних операцій, які можуть бути випадковими.

Зрушення у роботі алгоритму: Генетичні алгоритми враховують еволюційний підхід, і вони можуть виходити на певний оптимум, який не завжди відображає зміни в зовнішній ситуації. Якщо алгоритм "застряг" в оптимумі, то аномальні зміни можуть залишитися невиявленими.

Важкість врахування контексту: В деяких випадках ГА можуть виявляти аномалії, які не мають важливого контексту і можуть бути безпечними. Це може призвести до надмірного обмеження легітимної активності.

Для подолання цього недоліку важливо розглядати ГА як один із компонентів більш складних систем виявлення аномалій, використовувати ансамблі алгоритмів, які включають різні методи виявлення, та розробляти та налаштовувати ГА з урахуванням специфіки конкретного застосування в кібербезпеці [5].

Підвищена складність параметризації

Налаштування параметрів генетичних алгоритмів може бути важким завданням. Неправильне налаштування параметрів може призвести до неефективної роботи або навіть до втрати інформації.

Підвищена складність параметризації є ще одним з недоліків при використанні генетичних алгоритмів (ГА) у кібербезпеці. Давайте розглянемо

цей недолік на прикладі використання ГА для налаштування системи виявлення і запобігання атакам (IDS/IPS) [3].

Приклад: Налаштування системи IDS/IPS.

IDS/IPS – це системи, які моніторять мережевий трафік для виявлення аномалій та атак на комп'ютерну мережу. Використовуючи ГА, можна налаштувати параметри системи, такі як правила виявлення аномалій, чутливість до подій, списки сигнатур тощо.

Недолік підвищеної складності параметризації:

Велика кількість параметрів: Системи IDS/IPS можуть мати велику кількість параметрів, які потрібно налаштувати. Наприклад, кількість сигнатур, пороги для виявлення аномалій, чутливість до різних типів атак і багато інших параметрів. Вибір правильних параметрів може бути важким завданням.

Залежність параметрів від контексту: Параметри IDS/IPS можуть бути взаємозалежними і залежати від конкретної мережевої конфігурації і типів атак, які найчастіше виявляються. Це робить налаштування ще більш складним.

Постійна зміна параметрів: Параметри мережевої безпеки повинні постійно змінюватися для відповіді на нові загрози і атаки. Це вимагає постійного моніторингу та налаштування, що може бути ресурсомістким процесом.

Ризик помилок: Неправильне налаштування параметрів IDS/IPS може призвести до false positives (хибних спрацьовувань) та false negatives (пропусків аномалій), що може вплинути на ефективність захисту мережі.

Для подолання цього недоліку важливо використовувати методи автоматичної настройки параметрів на основі аналізу даних та машинного

навчання, щоб спростити процес налаштування та забезпечити більш надійний та ефективний захист мережі в умовах кібербезпеки [2].

Потенційна вразливість до атак

Генетичні алгоритми можуть бути вразливі до атак, оскільки їхні функції пристосування можуть бути піддані маніпуляції. Зловмисники можуть намагатися спеціально модифікувати дані, щоб викривити недоліки алгоритму.

Потенційна вразливість до атак є одним з недоліків при використанні генетичних алгоритмів (ГА) в кібербезпеці. Давайте розглянемо цей недолік на прикладі використання ГА для розв'язання задачі виявлення аномалій в мережевій активності.

Приклад: Виявлення аномалій в мережевій активності.

Допустимо, що ми використовуємо генетичний алгоритм для створення моделі виявлення аномалій в мережевій активності. Модель оцінює, чи є певна мережева активність нормальною чи аномальною. Однак існує потенційна вразливість до атак у цьому підході.

Недолік потенційної вразливості до атак:

Обман атакувальників: Зловмисники можуть намагатися обманути систему виявлення аномалій шляхом впровадження штучних аномалій в мережу. Вони можуть спеціально створювати та впроваджувати пакети або дії, які виглядають як аномалії для моделі, але насправді є атаками.

Маніпуляція функцією пристосування: Генетичні алгоритми використовують функцію пристосування для оцінки кожного індивіда в

популяції. Атакувальники можуть намагатися маніпулювати цією функцією, щоб збільшити свої шанси на прийняття від системи.

Зменшення надійності результатів: Атаки або маніпуляція функцією пристосування можуть призвести до невірних результатів. Модель може помилково виявляти нормальну активність як аномальну, або навпаки, що призводить до недостовірних результатів [9].

Сліпа довіра до ГА: Загальна довіра до ГА може призвести до недооцінки ризику атак і маніпуляції моделі виявлення аномалій.

Для подолання цього недоліку важливо розглядати додаткові заходи безпеки, такі як шифрування мережевого трафіку та моніторинг системи виявлення аномалій для виявлення незвичайних патернів у виявленні аномалій. Також важливо надавати великий значок внутрішнім процедурам безпеки, щоб уникнути можливих атак та маніпуляцій з боку зловмисників.

Обмежена здатність адаптації

Генетичні алгоритми можуть бути менш ефективними в умовах швидко змінюючогося оточення, оскільки вони можуть вимагати часу для адаптації до нових умов.

Обмежена здатність адаптації є недоліком при використанні генетичних алгоритмів (ГА) в кібербезпеці. Давайте розглянемо цей недолік на прикладі використання ГА для виявлення аномальної мережевої активності.

Приклад: Виявлення аномальної мережевої активності.

При використанні ГА для виявлення аномальної мережевої активності система моніторить мережевий трафік і намагається виявити аномалії та атаки на основі історичних даних [3].

Недолік обмеженої здатності адаптації:

Довгий час адаптації: ГА можуть потребувати певного часу для адаптації до змін у мережевому середовищі. Якщо зловмисники вчиняють атаки або змінюють структуру мережі швидко, то ГА може бути неспроможнім надати ефективну відповідь у відведений час.

Неспроможність розпізнати нові загрози: ГА зазвичай базуються на історичних даних для навчання і виявлення аномалій. Вони можуть бути неспроможними виявити нові атаки або загрози, які виникають вперше і не мають попереднього історичного контексту.

Перетин ресурсів: Внесення змін у параметри або поведінку ГА для адаптації до нових загроз може вимагати додаткових обчислювальних та людських ресурсів.

Потреба в постійному моніторингу: Використання ГА для виявлення аномалій може вимагати постійного моніторингу та апгрейду параметрів та функцій моделі, щоб вона була стійкою до змін.

Для подолання цього недоліку важливо розглядати ГА як одну з компонент системи виявлення аномалій і поєднувати їх з іншими методами моніторингу та захисту мережі, такими як сигнатурні методи, машинне навчання та інші технології кібербезпеки.

Питання конфіденційності

Використання генетичних алгоритмів може викликати питання конфіденційності, оскільки вони можуть вимагати доступу до чутливих даних.

Питання конфіденційності є серйозним недоліком при використанні генетичних алгоритмів (ГА) в кібербезпеці. Давайте розглянемо цей недолік

на прикладі використання ГА для створення і налаштування системи виявлення аномалій в мережах.

Приклад: Налаштування системи виявлення аномалій.

Системи виявлення аномалій в мережах використовують ГА для створення моделей та настройки параметрів для виявлення незвичайної мережевої активності.

Недолік конфіденційності:

Витік інформації: В процесі використання ГА може виникнути ризик витоку конфіденційних даних. Оскільки ГА працюють з великими обсягами даних, інформація про мережу, конфігурації або інші параметри можуть потрапити в руки незаконних користувачів або зловмисників.

Небажаний доступ до даних: Зловмисники можуть використовувати ГА для злому або маніпуляції системою виявлення аномалій. Якщо вони отримають доступ до ГА і внесуть зміни в процес, це може призвести до некоректних результатів та недостовірних виявлень аномалій.

Неправильна налаштування параметрів: Якщо ГА використовуються для налаштування системи виявлення аномалій, то помилки в налаштуванні можуть призвести до невірних або неповних результатів. Це може бути використано зловмисниками для уникнення виявлення їх атак.

Відомості про алгоритми: Знання про використовувані алгоритми ГА може бути використане для аналізу та обхіду системи безпеки. Зловмисники можуть намагатися розкрити деталі алгоритмів, що допоможе їм створити ефективні атаки.

Для подолання цього недоліку важливо вживати заходи забезпечення конфіденційності, такі як шифрування даних, обмеження доступу до ГА,

моніторинг і обмеження прав доступу користувачів, а також застосування криптографічних методів для захисту конфіденційності інформації, яка використовується в процесі налаштування системи виявлення аномалій [6].

Обмежене врахування динамічних аспектів

Генетичні алгоритми можуть бути менш ефективними в виявленні динамічних загроз або атак, оскільки вони можуть не здати врахувати зміни в часі.

Обмежене врахування динамічних аспектів є недоліком при використанні генетичних алгоритмів (ГА) в кібербезпеці. Давайте розглянемо цей недолік на прикладі використання ГА для виявлення аномальної мережевої активності.

Приклад: Виявлення аномальної мережевої активності.

Використовуючи ГА для виявлення аномальної мережевої активності, система аналізує дані мережі та намагається виявити аномалії на основі історичних даних і правил.

Недолік обмеженого врахування динамічних аспектів:

Невідповідність змінюваній загрозі: Зловмисники постійно розвивають нові методи та атаки. ГА, які базуються на статичних правилах та історичних даних, можуть бути неспроможними адаптуватися до нових загроз та аномалій.

Затримка в реакції: Використання ГА може вимагати часу на адаптацію до нових ситуацій. У випадку швидко змінюваних атак затримка в реакції може призвести до порушення безпеки мережі.

Недолік урахування контексту: Динамічні аспекти можуть включати зміни в мережевій топології, підвищення обсягу трафіку або інші зміни в контексті. ГА можуть бути неспроможними врахувати ці аспекти і призвести до невірних виявлень аномалій.

Недостатня адаптивність: ГА можуть мати обмежену здатність адаптуватися до змін в режимі роботи мережі, що важко передбачити та адаптувати.

Для подолання цього недоліку важливо використовувати інші методи, такі як машинне навчання та штучні нейронні мережі, які можуть бути більш адаптивними до динамічних аспектів мережі та кіберзагроз. Крім того, системи кібербезпеки повинні включати постійний моніторинг та оновлення, щоб вчасно виявляти нові атаки та загрози [9].

Вимоги до пам'яті

Генетичні алгоритми можуть вимагати значних обсягів пам'яті для збереження і обробки даних, що може бути проблематичним в умовах обмежених ресурсів.

Вимоги до пам'яті є одним з недоліків при використанні генетичних алгоритмів (ГА) в кібербезпеці. Давайте розглянемо цей недолік на прикладі використання ГА для криптоаналізу або розкриття паролів.

Приклад: Криптоаналіз або розкриття паролів.

ГА можуть використовуватися для відновлення паролів шляхом спроб багато варіантів. Це важливо в контексті кібербезпеки, оскільки дозволяє аналізувати паролі та ідентифікувати їхні слабкість.

Недолік вимог до пам'яті:

Велика кількість обчислень: При спробах відновлення паролів за допомогою ГА, може знадобитися велика кількість обчислень для перебору всіх можливих комбінацій. Це може вимагати великих обсягів пам'яті для збереження проміжних результатів та попередніх спроб.

Вимоги до зовнішньої пам'яті: Деякі ГА можуть вимагати зовнішньої пам'яті для збереження попередніх попередніх спроб або великих об'єктів даних, що можуть вимагати додаткової пам'яті та обчислювальних ресурсів

Масштабованість: Під час криптоаналізу паролів або інших криптографічних завдань, вимоги до пам'яті зростають зі збільшенням довжини пароля або складності шифру. Це може призвести до обмежень у масштабуванні завдання.

Застосування гнучкості: Вимоги до пам'яті можуть бути значущими, і використання ГА може бути складним на обмежених пристроях або в умовах з обмеженими ресурсами пам'яті.

Для подолання цього недоліку важливо розробляти оптимізовані ГА та алгоритми криптоаналізу, які вимагають менше пам'яті або використовують ефективні методи оптимізації для зниження вимог до пам'яті. Також може бути корисним розглядати можливість паралельних обчислень для розподілення завдань та обмеження використання пам'яті [10].

Висока складність реалізації

Розробка і впровадження генетичних алгоритмів може вимагати великих зусиль і спеціалізованих знань, що робить їх менш доступними для менших організацій або проектів з обмеженими ресурсами.

Висока складність реалізації є недоліком при використанні генетичних алгоритмів (ГА) в кібербезпеці. Давайте розглянемо цей недолік на прикладі створення системи виявлення і запобігання атакам на мережу.

Приклад: Система виявлення і запобігання атакам

Система виявлення і запобігання атакам використовує ГА для налаштування параметрів та створення правил для виявлення аномальних активностей у мережі.

Недолік високої складності реалізації:

Потреба в експертному знанні: Реалізація ГА вимагає знань і експертизи в генетичних алгоритмах та кібербезпеці. Встановлення параметрів, створення функцій пристосування та розробка правил виявлення аномалій вимагають високої кваліфікації.

Складна оптимізація: Визначення правильних параметрів ГА, таких як розмір популяції, імовірності мутації та схрещування, може бути складним завданням. Вибір неправильних параметрів може призвести до неефективної роботи системи.

Наявність гіперпараметрів: Крім параметрів ГА, система виявлення аномалій може мати інші гіперпараметри, такі як пороги виявлення аномалій і методи фільтрації даних. Визначення правильних гіперпараметрів також є важливим завданням.

Складність налаштування: Реалізація ГА може вимагати значних зусиль для налаштування та оптимізації. Це може займати час і ресурси, які можуть бути важливі в умовах кібербезпеки, де потрібна швидка реакція на загрози.

Для подолання цього недоліку важливо використовувати готові бібліотеки та програми для реалізації ГА в системах кібербезпеки, що дозволить зменшити складність розробки і налаштування. Також важливо постійно оновлювати та оптимізувати систему на основі аналізу результатів та нових загроз [4].

1.3 Генетичний алгоритм рою часток (PSO)

Генетичний алгоритм рою часток (Particle Swarm Optimization, PSO) є метаевристикою, яка була вдосконалена на основі природних процесів і заснована на ідеї спільного пошуку оптимальних рішень групою часток. PSO використовується для рішення оптимізаційних завдань та входить до класу алгоритмів рою (swarm intelligence).

Основні принципи PSO:

- Частки (particles): Кожен агент (частка) представляє потенційний оптимальний розв'язок завдання. Вони рухаються у просторі параметрів для пошуку найкращого розв'язку.
- Швидкість та позиція часток: Кожна частка має позицію та швидкість, які оновлюються на кожному кроці. Позиція вказує на поточне положення частки в просторі параметрів, а швидкість вказує на швидкість зміни позиції.
- Локальний та глобальний найкращий розв'язок: Кожна частка зберігає інформацію про свій найкращий розв'язок і найкращий розв'язок, знайдений усіма частками в рої.
- Оновлення швидкості і позиції: Швидкість та позиція кожної частки оновлюються на підставі її власного найкращого розв'язку, глобального найкращого розв'язку та швидкостей та позицій інших часток в рої.

Застосування PSO в кібербезпеці:

- Виявлення аномалій: PSO може використовуватися для створення моделей виявлення аномалій, які аналізують

незвичайну мережеву активність або аномалії в журналах подій.

- Конфігурація мережевих правил: PSO може допомогти в налаштуванні мережевих правил та параметрів безпеки, щоб забезпечити оптимальну захист мережі.
- Інтелектуальний аналіз загроз: PSO може допомогти у розробці інтелектуальних алгоритмів для аналізу загроз та виявлення патернів атак.

Переваги PSO включають:

- Глобальний пошук: ГА в здатні забезпечити більший глобальний пошук, оскільки вони можуть досліджувати широкий спектр можливих рішень, включаючи ті, які знаходяться вдалеку від поточного рішення. Це корисно в кібербезпеці, де потрібно виявляти вразливості та атаки великого масштабу.
- Збалансований пошук: ГА можуть бути налаштовані для збалансованого взяття до уваги об'єктивів і обмежень. Це дозволяє знайти оптимальні рішення, які задовольняють потреби кібербезпеки, не обмежуючи при цьому роботу системи.
- Можливість обробки складних обмежень: ГА можуть враховувати складні обмеження та умови, що часто присутні в сфері кібербезпеки, де потрібно враховувати велику кількість параметрів і обмежень.
- Різноманітність рішень: ГА призначені для створення різноманітності рішень, що може бути корисним в кібербезпеці

для боротьби з атаками, які використовують різні підходи і тактики.

- Застосування до оптимізації параметрів: ГА можуть використовуватися для оптимізації параметрів систем безпеки, таких як налаштування вогнеметів, правила безпеки мережі і інше.

Недоліки PSO включають:

- Вразливість до локальних оптимумів: PSO спроможний швидко знаходити локальні оптимуми, але може мати складнощі у виході з них. Це може бути проблемою в кібербезпеці, оскільки атаки можуть бути динамічними та змінюватися, і важливо мати здатність швидко адаптуватися до нових загроз.
- Потребує налаштування параметрів: PSO має декілька параметрів, таких як коефіцієнти розподілу і швидкості часток, які потрібно налаштовувати для кожного конкретного завдання. Неправильне налаштування може призвести до поганої продуктивності алгоритму.
- Обмежена різноманітність рішень: PSO може обмежувати різноманітність рішень, оскільки частки можуть вирішити "застрягнути" в певних точках простору параметрів. Це може ускладнити виявлення атак, які використовують різні підходи і тактики.
- Залежність від початкових умов: Початкові умови PSO можуть сильно вплинути на його збіжність та результати. Невірно

вибрані початкові умови можуть призвести до низької ефективності алгоритму.

- Не завжди гарантує збіжність: PSO не завжди гарантує збіжність до оптимального рішення. Існує можливість того, що алгоритм не знайде найкращого рішення або затримається на певному кроці пошуку.
- Висока обчислювальна складність: PSO може вимагати значних обчислювальних ресурсів, особливо при роботі з великими об'єктами або даними в кібербезпеці.

У кібербезпеці PSO може бути використаний, але його ефективність та придатність до конкретних завдань залежать від конкретного сценарію та налаштувань.

1.4 Використання алгоритму рою часток (PSO) для захисту корпоративних даних

У сучасній кібербезпеці алгоритм рою часток (PSO) використовується для вирішення різноманітних задач, таких як виявлення аномалій, налаштування параметрів систем безпеки, аналіз інцидентів та багато інших завдань. Давайте розглянемо, як PSO використовується в кібербезпеці, його переваги та недоліки:

Приклади використання PSO в кібербезпеці:

- Виявлення аномалій: PSO може використовуватися для створення моделей виявлення аномалій в мережах або системах. Він допомагає виявити незвичайні патерни або активність, що може вказувати на потенційні загрози.

- Налаштування параметрів систем безпеки: PSO використовується для автоматичного налаштування параметрів систем безпеки, таких як правила мережевого брандмауера, параметри виявлення атак та фільтри подій.
- Криптоаналіз: В криптоаналізі PSO використовується для аналізу та розкриття слабких сторін шифрів або паролів.
- Моделювання загроз і ризиків: PSO допомагає створювати моделі загроз та ризиків у кібербезпеці, щоб оцінити вразливості та розвивати стратегії запобігання атакам.

Переваги використання PSO в кібербезпеці:

- Адаптивність: PSO може адаптуватися до змінних умов та нових загроз, що робить його корисним у галузі кібербезпеки, де атаки постійно еволюціонують.
- Швидкість зближення до оптимального рішення: PSO зазвичай швидко наближається до бажаного оптимального рішення, що дозволяє швидко реагувати на атаки або вразливості.
- Здатність до оптимізації параметрів: PSO допомагає автоматично знаходити оптимальні параметри систем безпеки, що підвищує їхню ефективність.

Недоліки використання PSO в кібербезпеці:

- Ризик застрягання в локальних оптимумах: Як і інші метаевристики, PSO може застрягати в локальних оптимумах, не знаходячи глобального оптимального рішення.

- Потреба в налаштуванні гіперпараметрів: PSO має деякі гіперпараметри, які потребують налаштування, і вони можуть впливати на його ефективність.
- Потреба у великих обчислювальних ресурсах: Для задач з великим обсягом даних чи складною обчислювальною логікою може бути потрібно значне обчислювальне спорядження для використання PSO.

Загалом, PSO може бути потужним інструментом для оптимізації та виявлення аномалій в кібербезпеці, але його використання повинно бути обдуманим та налаштованим відповідно до конкретних завдань та умов.

Алгоритм рою часток (PSO) використовується у різних програмних продуктах та системах у сфері кібербезпеки для вирішення конкретних завдань та задач. Нижче наведені деякі програми та системи, які можуть використовувати PSO для кібербезпеки:

- Snort: Snort – це система виявлення вторгнень та управління мережею, яка використовує PSO для покращення налаштування правил виявлення атак та фільтрації трафіку.
- FireEye: FireEye – це компанія, яка розробляє апаратне та програмне забезпечення для виявлення та запобігання атакам. Вони використовують PSO для аналізу мережевої активності та виявлення загроз.
- Intrusion Detection Systems (IDS): Багато IDS, такі як Suricata або Bro, можуть використовувати PSO для покращення виявлення атак та аналізу великих обсягів мережевого трафіку.

- Системи управління доступом: Деякі системи керування доступом використовують PSO для оптимізації налаштування правил доступу та автентифікації.
- Виявлення аномалій в журналах подій: PSO може бути використаний для аналізу та виявлення аномалій у журналах подій систем та мереж.
- Криптографічні дослідження: В деяких дослідженнях та проектах у галузі криптографії PSO використовується для аналізу та покращення криптографічних алгоритмів.
- Управління інцидентами та відновленням після атак: PSO може бути використаний для аналізу інцидентів, виявлення проблем та пошуку оптимальних шляхів відновлення після атак.

1.5 Порівняльний аналіз алгоритму рою часток (PSO) при використанні у кібербезпеці

Зважаючи на практичне використання алгоритмів у кібербезпеці, розглянемо детальніше їх застосування порівняно з алгоритмом рою часток (PSO) – див. таблицю 1.3.

Таблиця 1.3 – Порівняння методів (алгоритмів) у кібербезпеці

Метод	Практичне використання	Сильні сторони	Слабкі сторони
PSO	Виявлення аномалій у мережевому трафіку, налаштування параметрів систем безпеки.	Адаптивний, швидкість зближення до оптимального рішення, здатність до оптимізації параметрів систем безпеки.	Ризик застрягання в локальних оптимумах, потребує налаштування гіперпараметрів.
Генетичні алгоритми (GA)	Налаштування параметрів систем безпеки, генерація сильних паролів, визначення оптимальних ключів шифрування, оптимізація мережевих параметрів.	Знаходження глобальних оптимумів, застосовний до різних завдань оптимізації.	Висока обчислювальна складність, потребує великих ресурсів.

Продовження таблиці 1.3

Метод	Практичне використання	Сильні сторони	Слабкі сторони
Метод опорних векторів (SVM)	Класифікація та виявлення аномалій в мережевому трафіку, ідентифікація загроз, аналіз подій безпеки.	Висока точність, ефективність при роботі з великою кількістю ознак.	Потребує налаштування гіперпараметрів, не завжди підходить для задач з багатьма класами чи перекриваються класами.
Метод нейронних мереж (NN)	Виявлення аномалій у мережевому трафіку, інтелектуальний аналіз загроз, розпізнавання патернів атак.	Висока точність, здатність до роботи з великою кількістю ознак.	Потребує великої кількості даних для навчання, значних обчислювальних ресурсів.
Random Forest (RF)	Класифікація загроз, виявлення аномалій, інтелектуальний аналіз мережевого трафіку.	Висока точність, здатність працювати з різними типами даних та великими обсягами.	Може бути схильним до перенавчання, важко пояснити прийняте рішення.

Продовження таблиці 1.3

Метод	Практичне використання	Сильні сторони	Слабкі сторони
Кластерний аналіз (Clustering)	Групування подій та об'єктів за схожістю для виявлення аномалій та ідентифікації загроз.	Здатність виявляти неправильності та залежності між даними.	Потребує визначення кількості кластерів та вибору підходящого алгоритму кластеризації.
Логістична регресія (Logistic Regression)	Класифікація загроз, аналіз подій безпеки, ідентифікація аномалій в мережевому трафіку.	Простий у використанні та інтерпретації, підходить для бінарних та багатокласових завдань.	Може бути менш ефективним для складних завдань з багатьма факторами.
Лінійна регресія (Linear Regression)	Аналіз та передбачення залежностей між ознаками, аналіз атак та загроз.	Простий у використанні та інтерпретації, може виявляти залежності між змінними.	Підходить переважно для задач регресії, менш ефективний для класифікації.
Метод головних компонент (PCA)	Редукція розмірності даних, виявлення аномалій та візуалізація мережевого трафіку.	Дозволяє скоротити кількість ознак, поліпшує інтерпретованість даних.	Може втратити частину інформації при зменшенні розмірності.

У загальному, вибір алгоритму залежить від конкретних завдань у кібербезпеці та вимог до точності, швидкості та адаптивності. Комбінування різних алгоритмів може бути ефективним способом покращити безпеку та ефективність системи кібербезпеки.

Таблиця 1.4 – Порівняння сильних та слабких сторін методів (алгоритмів) у кібербезпеці

Алгоритм	Сильні сторони	Слабкі сторони
PSO	Адаптивність до змінних умов і загроз	Ризик застрягання в локальних оптимумах
	Швидкість зближення до оптимального рішення	Потреба в налаштуванні гіперпараметрів
	Здатність до оптимізації параметрів систем безпеки	Потреба в великих обчислювальних ресурсах
GA	Можливість знаходити глобальні оптимуми	Висока обчислювальна складність
	Застосовний до різних завдань оптимізації	Недостатня надійність (залежність від початкового набору)
	Здатність до обробки великих обсягів даних	Потреба в багато обчислювальних ресурсах

Продовження таблиці 1.4

Алгоритм	Сильні сторони	Слабкі сторони
SVM	Добра узагальнююча здатність	Складність налаштування гіперпараметрів
	Ефективність при роботі з великою кількістю ознак	Недоцільність для задач з багатьма класами або категоріями
	Здатність до роботи з несиметричними даними	Недоцільність для задач, де класи перекриваються
NN	Висока точність класифікації	Великі вимоги до кількості даних для навчання
	Здатність до роботи з великою кількістю ознак	Великі вимоги до обчислювальних ресурсів
	Здатність до автоматичного виявлення патернів	Висока схильність до перенавчання

В таблиці 1.4 наведений порівняльний аналіз алгоритму рою часток (PSO) та інших алгоритмів, що використовуються у кібербезпеці, включаючи генетичні алгоритми (GA), метод опорних векторів (SVM) та метод нейронних мереж (NN). Для кожного алгоритму наведені сильні та слабкі сторони.

Цей порівняльний аналіз надає уявлення про сильні та слабкі сторони алгоритмів PSO, GA, SVM та NN, що використовуються у кібербезпеці. Вибір конкретного алгоритму залежить від специфічних завдань, які потрібно вирішити, та умов, в яких він буде застосовуватися.

Висновки до розділу та постановка задач

Генетичні алгоритми можуть бути корисним інструментом у сфері кібербезпеки, але їх застосування пов'язане із певними недоліками.

Перш за все, генетичні алгоритми можуть бути обчислювально вимогливими, особливо коли вони використовуються для оптимізації великих об'ємів даних або складних завдань. Це може призвести до неефективності алгоритму або вимагати значних обчислювальних ресурсів.

По-друге, генетичні алгоритми вимагають налаштування параметрів, таких як розмір популяції, ймовірність мутації і кросоверу. Неправильне налаштування цих параметрів може призвести до неефективних результатів або навіть до невдачі алгоритму.

Крім того, генетичні алгоритми є стохастичними, і їх результати не завжди передбачувані. Це може бути недоцільним в кібербезпеці, де надійність і стабільність рішень є критичними.

Важливо також враховувати, що генетичні алгоритми можуть бути вразливими до атак, спрямованих на спотворення процесу оптимізації. Це може призвести до невірних рішень або навіть загроз для кібербезпеки.

Крім того, генетичні алгоритми можуть вимагати значних обчислень для досягнення найкращих результатів, що може призвести до затримок у вирішенні завдань в реальному часі.

Нарешті, генетичні алгоритми не завжди є ідеальними для вирішення складних завдань у кібербезпеці, де інші методи штучного інтелекту або машинного навчання можуть бути більш ефективними.

Отже, хоча генетичні алгоритми можуть бути корисними в кібербезпеці, важливо ретельно аналізувати їхні переваги та недоліки і використовувати їх відповідно до конкретних вимог і контексту.

Тому враховуючи всю теоретичну інформацію, потрібно вибрати найбільш оптимальний спосіб вирішення задачі, та вдосконалити генетичний алгоритм рою часток, за для підвищення ефективності захисту інформаціїї на корпотративних серверах обробки даних.

РОЗДІЛ 2. ВДОСКОНАЛЕННЯ АЛГОРИТМУ РОЮ ЧАСТОК (PSO) ДЛЯ ПІДВИЩЕННЯ ЗАХИЩЕНОСТІ ДАНИХ СЕРВЕРІВ

Алгоритм рою часток (PSO) є одним із перспективних методів оптимізації, який вже знайшов застосування в різних областях. У контексті кібербезпеки, його потужність полягає в здатності адаптуватися до змінюючихся умов та виявленні оптимальних стратегій захисту.

У цьому розділі ми зосередимося на модернізації алгоритму рою часток з точки зору кібербезпеки. Обговоримо ключові аспекти, спрямовані на вдосконалення ефективності PSO в виявленні, запобіганні та протидії кіберзагрозам. Порушимо питання оптимізації параметрів для систем виявлення вторгнень, розподілу ресурсів у віртуальних середовищах, а також налаштування стратегій реакції на динамічні загрози.

Мета даного розділу – висвітлити можливості та перспективи модернізації алгоритму рою часток у сфері кібербезпеки та визначити напрямки подальших досліджень для підвищення стійкості та ефективності захисту інформаційних систем.

2.1 Аналіз можливості вдосконалення алгоритму за допомогою гібридизації PSO з іншими алгоритмами

Гібридизація розуміється як комбінація двох чи більше різних алгоритмів з метою отримання переваг кожного з них і покращення ефективності розв'язання конкретної задачі. У випадку гібридизації з алгоритмом рою часток (PSO), це може означати об'єднання PSO з іншими оптимізаційними або інтелектуальними методами для покращення результатів або розширення області застосування.

Ось кілька прикладів гібридизації PSO з іншими алгоритмами:

- Гібридизація PSO з Генетичним Алгоритмом (GA):
  - Дослідники часто комбінують PSO із генетичними алгоритмами для створення адаптивних оптимізаційних стратегій. Генетичні алгоритми можуть використовуватися для мутації і схрещування часток у рої, розширюючи таким чином область пошуку та роблячи алгоритм більш гнучким [16].
- Гібридизація PSO з Алгоритмами Колонії Мурах (ACO):
  - Комбінація PSO та алгоритмів колонії мурах може використовуватися для розв'язання задач комбінаторної оптимізації. PSO може слугувати для глобального пошуку, тоді як ACO допомагає в локальному вдосконаленні рішення [18].
- Гібридизація PSO з Методами Машинного Навчання:

PSO може бути поєднаний із методами машинного навчання, такими як підтримка векторів, для створення гібридних систем, які використовують силу оптимізаційних алгоритмів для навчання моделей.[17]

Гібридизація дозволяє використовувати переваги різних алгоритмів, що призводить до створення більш потужних та адаптивних оптимізаційних систем.

Урахування кількості даних та розмірності простору параметрів

У сучасному інформаційному середовищі, де обсяги даних нещадно зростають, важливо розглядати алгоритми кібербезпеки з урахуванням кількості даних та розмірності простору параметрів. Подолання викликів, пов'язаних з великими обсягами інформації та розмаїттям параметрів, є ключовим завданням для забезпечення ефективної та надійної захисту в інтернет-просторі.

- Сприятливе Урахування Обсягу Даних.

В сучасних умовах великі обсяги даних стали нормою, і це ставить перед системами кібербезпеки великі виклики. Урахування цього фактора у модернізованому алгоритмі рою часток (PSO) виявляється в здатності швидко та ефективно адаптуватися до великої кількості вхідних даних.

Приклад. Додавання механізмів, які автоматично регулюють ваги параметрів PSO в залежності від обсягу вхідних даних, дозволяє алгоритму оптимально адаптуватися до конкретного завдання забезпечення кібербезпеки.

- Ефективна Робота у Багатовимірному Просторі Параметрів.

Багатовимірний простір параметрів, що властивий сучасним системам кібербезпеки, вимагає особливої уваги. Модернізований PSO може оптимізувати роботу у таких складних просторах, де кожен параметр взаємодіє з іншими, впливаючи на ефективність заходів забезпечення безпеки.

Приклад. Використання розширених стратегій генерації часток та їхнього руху дозволяє PSO ефективно навігувати у багатовимірному просторі параметрів, забезпечуючи оптимальні рішення для кожного компонента системи безпеки.

- Адаптованість до Змін в Обсягах Інформації.

Середовище інтернету непередбачуване, і обсяги інформації можуть змінюватися динамічно. Модернізований PSO повинен виявляти гнучкість та

адаптованість до змін у кількості даних для забезпечення стійкості та ефективності.

Приклад. Включення механізмів автоматичного налаштування параметрів PSO на основі моніторингу обсягу даних дозволяє алгоритму підтримувати оптимальний рівень продуктивності навіть при коливаннях обсягу інформації.

- Забезпечення Конфіденційності та Інтеграції.

Основною метою систем кібербезпеки є забезпечення конфіденційності та інтеграції інформації. Модернізований PSO повинен ефективно враховувати ці аспекти, гарантуючи, що обчислення та оптимізація відповідають вимогам безпеки.

Приклад. Використання шифрування даних під час процесу оптимізації PSO є однією з можливих стратегій для забезпечення конфіденційності у вимірюваному обсязі даних.

Урахування кількості даних та розмірності простору параметрів є невід'ємною частиною ефективного застосування PSO в сфері кібербезпеки. Модернізований підхід до PSO, спрямований на вирішення цих викликів, дозволяє підвищити ефективність та надійність систем захисту в умовах постійних змін в обсягах даних та конфігурації параметрів.

2.2 Моделювання загроз і ризиків під час вдосконалення генетичного алгоритму рою часток(PSO).

Моделювання загроз і ризиків є важливою складовою в галузі кібербезпеки, оскільки дозволяє системам ефективно виявляти та управляти потенційними небезпеками, забезпечуючи високий рівень захисту. Загрози в інформаційних системах постійно зростають, і їх вчасне виявлення та оцінка є

критичною для забезпечення цілісності, конфіденційності та доступності даних. У цьому розділі розглянемо моделювання загроз і ризиків в контексті кібербезпеки, зокрема використання алгоритмів рою часток (PSO) для оптимізації процесів виявлення та управління ризиками.

- Методи Моделювання Загроз.

Моделювання загроз передбачає визначення потенційних векторів атак та оцінку їхнього впливу на інформаційні системи. Використання алгоритмів PSO дозволяє ефективно аналізувати різні сценарії загроз та визначати оптимальні стратегії захисту.

Приклад. PSO може бути використаний для пошуку оптимальних параметрів системи виявлення вторгнень, які максимізують ефективність виявлення загроз.

- Оцінка Ризиків та Вразливостей.

Під час моделювання ризиків важливо враховувати вразливості існуючих захисних механізмів та систем. PSO може використовуватися для аналізу і підвищення ефективності заходів безпеки, що зменшує загрози та ризики.

Приклад. Алгоритм PSO може оптимізувати розташування детекторів вразливостей та реагування на ризики, забезпечуючи оптимальний рівень захисту.

- Адаптація до Середовища, що змінюється.

Сучасне середовище кібербезпеки постійно змінюється, і моделювання загроз повинне бути динамічним процесом. PSO, завдяки своїй здатності до

адаптації, може швидко реагувати на нові загрози та ефективно моделювати їхні наслідки.

Приклад. Алгоритм PSO може використовувати інформацію про нові загрози для оптимізації конфігурації системи безпеки та мінімізації можливих ризиків.

- Оптимізація Системи Виявлення та Реагування.

Моделювання дозволяє оптимізувати процеси виявлення та реагування на загрози, забезпечуючи ефективну роботу систем безпеки. PSO може бути використаний для автоматизації цих процесів.

Приклад. Застосування PSO для оптимізації параметрів системи реагування на інциденти дозволяє швидко та точно реагувати на нові загрози.

Моделювання загроз і ризиків у кібербезпеці, підтримуване алгоритмами PSO, є ефективним підходом до забезпечення надійності та ефективності захисних механізмів. Використання PSO дозволяє не лише ідентифікувати потенційні небезпеки, але і оптимізувати заходи безпеки для мінімізації ризиків і забезпечення стійкості системи в змінюючомуся кіберпросторі.

Забезпечення конфіденційності даних

Однією з ключових задач в галузі кібербезпеки є забезпечення конфіденційності даних. Інформація стає все ціннішою, тому важливо застосовувати сучасні та ефективні методи для запобігання несанкціонованому доступу до неї. У цьому розділі розглянемо, як алгоритм рою часток може бути використаний для оптимізації заходів забезпечення конфіденційності даних.

- Вибір та Оптимізація Криптографічних Алгоритмів.

Забезпечення конфіденційності даних часто включає в себе використання криптографічних алгоритмів. Алгоритм рою часток може бути використаний для оптимізації параметрів таких алгоритмів, що призведе до підвищення їх ефективності та стійкості.

Приклад. PSO використовується для визначення оптимальних ключів шифрування, що забезпечить максимальний рівень конфіденційності даних.

- Оптимізація Методів Доступу до Даних.

Застосування PSO може розширити можливості захисту даних через оптимізацію методів доступу до них. Параметри, такі як рівень доступу та аутентифікації, можуть бути оптимізовані для забезпечення конфіденційності.

Приклад. Алгоритм PSO використовується для підбору оптимальних параметрів механізмів контролю доступу до даних, що дозволяє підтримувати високий рівень конфіденційності.

- Управління Ключами та Ідентифікаторами.

Ефективне управління ключами та ідентифікаторами є важливою складовою забезпечення конфіденційності даних. PSO може оптимізувати процес генерації, розподілу та управління ключами для підвищення стійкості системи.

Приклад. Алгоритм рою часток допомагає визначити оптимальні параметри для генерації та зберігання шифрувальних ключів, забезпечуючи конфіденційність.

- Виявлення та Захист від Атак.

PSO може бути використаний для оптимізації процесів виявлення та захисту від різних видів атак, що спрямовані на порушення конфіденційності даних. Шляхом оптимізації алгоритмів виявлення можна підвищити реакцію на потенційні загрози.

Приклад. PSO використовується для визначення оптимальних параметрів систем виявлення вторгнень, що дозволяє ефективно реагувати на атаки та забезпечувати конфіденційність даних.

- Автоматизація Процесів Конфіденційності.

Використання алгоритму рою часток може спростити та автоматизувати процеси забезпечення конфіденційності, зменшуючи вплив людського фактора на систему та забезпечуючи сталу охорону даних.

Приклад. PSO використовується для оптимізації параметрів автоматичної системи перевірки та аудиту даних для миттєвого виявлення можливих порушень конфіденційності.

- Висновок.

Алгоритм рою часток є потужним інструментом для оптимізації заходів забезпечення конфіденційності даних. Використання PSO дозволяє створювати ефективні та стійкі системи захисту, забезпечуючи високий рівень конфіденційності в умовах постійно зростаючих загроз.

2.3 Врахування аспектів еволюції даних для вдосконалення алгоритму рою частоу(PSO).

З урахуванням постійної еволюції даних, зокрема їх обсягу та структури, виникає необхідність вдосконалення методів їх обробки та аналізу. В даному

розділі розглянемо, як алгоритм рою часток може бути використаний для оптимізації процесів, пов'язаних з аспектами еволюції даних.

- Адаптація до Змінюючихся Обсягів Даних.

Однією з ключових особливостей еволюції даних є зміна їх обсягу з часом. PSO може бути використаний для автоматичної адаптації алгоритмів обробки даних до змінюючихся обсягів.

Приклад. Застосування PSO для оптимізації параметрів алгоритмів стискання даних, що дозволяє забезпечувати ефективну обробку як невеликих, так і великих обсягів інформації.

- Динамічна Аналітика та Прогнозування Змін В Даних.

Алгоритм рою часток може бути використаний для побудови моделей, які адаптуються до динаміки даних. Це важливо для ефективного прогнозування та аналізу змін в інформації з часом.

Приклад. Використання PSO для оптимізації параметрів моделей прогнозування, які автоматично адаптуються до нових тенденцій та паттернів у наборі даних.

- Оптимізація Процесів Автоматичної Категоризації Даних.

Зміна структури даних може потребувати перегляду методів їх категоризації. PSO може бути використаний для оптимізації параметрів алгоритмів кластеризації та категоризації, що враховують еволюцію даних.

Приклад. Застосування PSO для визначення оптимальних груп категорій або кластерів, які найкраще відображають змінюючуся структуру даних.

- Гнучкість Систем Безпеки для Адаптації до Нових Загроз.

Зміна характеру загроз вимагає постійної адаптації систем безпеки. PSO може бути використаний для оптимізації параметрів систем виявлення вторгнень та антивірусних заходів.

Приклад. Використання PSO для оптимізації параметрів алгоритмів виявлення аномалій, які забезпечують ефективну реакцію на нові види загроз.

- Оптимізація Ресурсів для Обробки Динамічних Даних.

Еволюція даних часто пов'язана зі зміною їхнього обсягу та складності. PSO може допомагати оптимізувати використання ресурсів, щоб ефективно обробляти динамічні дані.

Приклад. Визначення оптимального розподілу обчислювальних ресурсів для швидкої обробки змінюючихся обсягів даних.

Врахування аспектів еволюції даних за допомогою алгоритму рою часток відкриває нові можливості для ефективного управління та адаптації до змін в обсязі та структурі інформації. Поєднання PSO із задачами, пов'язаними з еволюцією даних, дозволяє створювати гнучкі та динамічні системи обробки інформації.

Забезпечення високої стійкості до шуму

В контексті обробки даних інформаційних систем важливо мати алгоритми, які можуть ефективно фільтрувати шум та забезпечувати стабільну роботу системи навіть у шумовому середовищі. У цьому розділі розглянемо, як алгоритм рою часток (PSO) може бути використаний для забезпечення високої стійкості до шуму в обробці даних.

- Методи Фільтрації та Згладжування за Допомогою PSO.

Алгоритм рою часток може бути використаний для оптимізації параметрів методів фільтрації та згладжування даних. Це дозволяє системі ефективно виділяти сигнали від шумів та забезпечувати точні результати обробки.

Приклад. Використання PSO для оптимізації параметрів низькочастотного фільтра для згладжування шуму в часових рядах.

- Адаптивна Фільтрація Із Здатністю до Зміни Умов.

PSO дозволяє створювати адаптивні фільтри, які можуть змінювати свою структуру та параметри відповідно до змін у рівні шуму чи його характеристик.

Приклад. Використання PSO для створення адаптивного фільтра, який може автоматично змінювати параметри фільтрації в залежності від змін в середовищі.

- Стабілізація Системи у Шумових Умовах.

PSO може бути використаний для стабілізації роботи системи в умовах високого шуму. Оптимізація параметрів алгоритмів дозволяє системі ефективно працювати навіть при великих коливаннях сигналу-шуму.

Приклад. Застосування PSO для стабілізації алгоритму обробки зображень в умовах високого рівня фотографічного шуму.

- Антишумові Фільтри та Оптимальні Параметри.

PSO може допомогти знаходити оптимальні параметри антишумових фільтрів, які забезпечують ефективну роботу системи у великому спектрі умов шуму.

Приклад. Використання PSO для оптимізації параметрів антишумового фільтра у вузькоспрямованих системах звукозапису.

- Відновлення Сигналу під Шумом.

PSO може бути використаний для відновлення сигналу під впливом шуму, що дозволяє системі коректно виділяти корисну інформацію в умовах великого шуму.

Приклад. Створення PSO-оптимізованого алгоритму для відновлення аудіосигналу, спотвореного акустичним шумом.

Застосування алгоритму рою часток для оптимізації процесів фільтрації та обробки даних дозволяє створювати системи, що відзначаються високою стійкістю до шуму. Це важливо в умовах, коли необхідно ефективно обробляти дані в шумових чи непередбачуваних умовах, щоб отримувати точні та надійні результати.

2.4 Методи вдосконалення генетичного алгоритму рою часток

Алгоритм рою часток (PSO) може бути вдосконалений та покращений для використання в кібербезпеці через додавання специфічних функцій та підходів. Нижче наведено способи покращення PSO та їхнє практичне використання в сфері кібербезпеки (див. таблицю 2.1).

Таблиця 2.1 – Способи покращення PSO та їхнє практичне використання в сфері кібербезпеки

Покращення та пункти	Приклад використання	Як можна реалізувати покращення	Із якою технологією чи програмою
Гібридизація PSO з іншими алгоритмами	Виявлення аномалій, аналіз журналів подій, оптимізація параметрів систем безпеки.	Поєднання PSO з іншими алгоритмами або методами машинного навчання.	Використання бібліотек машинного навчання, таких як Scikit-Learn або TensorFlow.
Урахування кількості даних та розмірності простору параметрів	Обробка обмежених обсягів даних або великої кількості ознак.	Розробка оптимізованих методів дискретизації даних та визначення підмножин ознак.	Використання спеціалізованих інструментів для обробки даних, таких як Apache Spark.
Моделювання загроз і ризиків	Оцінка потенційних вразливостей та розвиток стратегій захисту.	Розробка математичних моделей для аналізу загроз та визначення ризиків.	Використання програм для моделювання ризиків, таких як OpenFAIR.

Продовження таблиці 2.1

Покращення та пункти	Приклад використання	Як можна реалізувати покращення	Із якою технологією чи програмою
Забезпечення конфіденційності даних	Аналіз даних без розкриття їхнього змісту.	Використання технік шифрування та обробки даних, що зберігають конфіденційність.	Використання криптографічних бібліотек, таких як OpenSSL.
Використання усередині систем інтелектуальної безпеки	Захист великих мереж, розпізнавання патернів атак та виявлення загроз.	Інтеграція PSO з системами інтелектуальної безпеки та іншими засобами захисту.	Використання рішень для інтеграції великих систем, таких як Security Information and Event Management (SIEM).
Врахування аспектів еволюції даних	Адаптація PSO до змін в структурі даних та трафіку.	Розробка алгоритмів для виявлення та врахування змін в даних.	Використання інструментів для обробки потокових даних, таких як Apache Kafka.

Продовження таблиці 2.1

Покращення та пункти	Приклад використання	Як можна реалізувати покращення	Із якою технологією чи програмою
Врахування множини обмежень	Врахування обмежень, які обмежують варіанти рішень PSO.	Розробка обробників обмежень та стратегій управління конфліктами між ними.	Використання спеціалізованих інструментів для керування обмеженнями, таких як IBM ILOG CPLEX.
Забезпечення високої стійкості до шуму	Виявлення аномалій у шумному середовищі мережі.	Розробка методів фільтрації шуму та вдосконалення адаптації до змін.	Використання алгоритмів для виділення сигналу в шумовому середовищі.

В даному дослідженні було використано (в якості покращення) способи гібридизації PSO з іншими алгоритмами, а також моделювання загроз та ризиків. Дані дослідження стали можливими за рахунок реалізації адаптивного налаштування коефіцієнтів для модернізаії точності досліджень у реальному часі. І, як результат, було створено систему забезпечення захисту мережі від атак зловмисників.

Алгоритм дій які потрібно виконати для реалізації даного рішення.

Крок 1.

Реалізація стандартного алгоритму рою часток, як основа. В майбутньому це також знадобиться нас для проведення тестування та порівняння алготитмів.

Крок 2.

Далі використовуємо більш складу функцію відбору та фунцію мутації. Виконуємо адаптивне налаштування коефіцієнтів для нових функцій. Більш детально описано в розділі 3.1.

Крок 3.

Виконуємо математичне порівняння після адаптивного налаштвуання. Більш детально описано в розділі 3.1.

Крок 4.

Далі виконуємо тікі дії:

- Визначення параметрів
- Визначення функції вартості
- Вагові коефіцієнти правил

Більш детально описано в розділі 3.2.

Крок 5.

Далі навчаємо вдосконалений алгоритм рою часток на не великій кількості тестових даних, з врахуванням того що алготим далі буде навчатись в середовищі де він буде застосовуватись.

Реалізація звітів та логування.

Крок 6.

Налаштування надсилання звітів на окремий сервіс(пошту).

Тестування алгоритму на великій вибірці даних в тестовому вередовищі на віртуальних машинах.

Порівняння алгоритму з не вдосконаленим аналогом.

Відображення та оцінка отриманих результатів.

Крок 7.

На основі отриманих результатів , пошук можливих більш ефективних рішень або пошук можливості подальшого вдосконалення алгоритму , за потреби.

Висновки до розділу

На основі проведених досліджень та аналізу можна зробити висновок, що модернізація алгоритму рою часток (PSO) представляє собою важливий і перспективний напрямок в області кібербезпеки. Використання PSO в контексті захисту інформаційних систем виявляє значущий потенціал для покращення стійкості та ефективності заходів безпеки.

Однією з ключових переваг модернізованого PSO є його здатність адаптуватися до змінюючихся умов кіберпростору. Застосування алгоритму

виявляється не лише у виявленні загроз, але й в оптимізації параметрів систем виявлення вторгнень, розподілу ресурсів та реакції на нові види атак.

Результати експериментів та тестувань підтверджують ефективність модернізованого PSO в порівнянні з існуючими методами захисту. Спостережені поліпшення в якості виявлення загроз, мінімізація ложних позитивів, оптимізація використання ресурсів та адаптація до нових умов.

Завдяки цим досягненням можна зробити висновок, що модернізація алгоритму рою часток є перспективним інструментом для впровадження в системи кібербезпеки. Подальший розвиток цього напрямку досліджень може призвести до створення ще більш ефективних та адаптивних рішень для забезпечення надійності та стійкості інформаційних.

Для магістерської кваліфікаційної роботи було вирішено реалізувати наступні модернізаії: перш за все реалізувати адаптивне налаштування коефіцієнтів для модернізаії точності досліджень у реальному часі та створити систему забезпесення захисту мережі від атак зловмисників. Для відслідкоування та віддаленій роботі з системою буде додана можливість реалізаії керування системою через пошту, щоб в режимі реального часу відслідковувати статус системи та керувати атакми.

РОЗДІЛ 3. МОДЕРНІЗАЦІЯ ТА ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМУ РОЮ ЧАСТОК В КОНТЕКСТІ КІБЕРБЕЗПЕКИ

В залежності від поставленого завдання алгоритм рою часток може мати різноманітні покращення та вдосконалення, частина з них була розглянута у попередньому розділі та на основі чого було запропоновано комплекс рішень для підвищення ефективності застосування для рішень питань у кібербезпеці.

3.1 Адаптивне налаштування коефіцієнтів для вдосконалення алгоритму рою часток.

Математично підвищення ефективності генетичного алгоритму рою часток (PSO) можна досягнути шляхом розгляду адаптивної настройки коефіцієнтів або інших параметрів PSO на прикладі оптимізації. Розглянемо математичні аспекти адаптивної настройки коефіцієнта інерції (inertia weight) у PSO:

Початковий коефіцієнт інерції (IW) обрано у прикладі як 0.5. Але для досягнення більшої ефективності, можна використовувати адаптивну настройку IW. Однією з поширених формул для розрахунку IW є:

$$IW(t)=IWmax−Tt⋅(IWmax−IWmin)$$

де:

- IW(t) – значення коефіцієнта інерції в момент часу t,
- IWmax – максимальне значення коефіцієнта інерції,
- IWmin – мінімальне значення коефіцієнта інерції,
- t – поточна ітерація алгоритму,
- T – максимальна кількість ітерацій.

Ця формула забезпечує поступове зменшення IW протягом ітерацій, дозволяючи рою часток більше досліджувати простір параметрів на початку оптимізації і більш точно "сходитися" до оптимуму пізніше.

Приклад вище показує, як змінюється IW з кожною ітерацією, допомагаючи адаптивно налаштувати цей параметр для покращення продуктивності PSO. Ця адаптивність дозволяє алгоритму краще адаптуватися до конкретної задачі оптимізації і швидше знаходити оптимальні рішення.

Розглядаючи інші параметри PSO, можна також застосовувати схожі математичні підходи до їх адаптивної настройки для підвищення ефективності алгоритму в конкретних ситуаціях.

Математичне порівняння після адаптивного налаштвуання

Розглянемо приклад , як адаптивний коефіцієнт інерції (IW) може покращити збіжність генетичного алгоритму рою часток (PSO) на прикладі математичної моделі. Ми порівняємо збіжність при фіксованому IW з адаптивним IW.

Припустимо, ми маємо наступну функцію оптимізації:

$$f(x)=(x-5)^2$$

Ми шукаємо мінімум цієї функції, і відомо, що глобальний мінімум знаходиться в x=5, де f(x)=0.

Позначимо поточне положення частки x_i у кожній ітерації як $x_i^{(t)}$, де t – номер ітерації, а $x_i^{(0)}$ – початкове положення. Також, розглядаючи адаптивний IW, позначимо його як $IW^{(t)}$, де t – номер ітерації.

- Для фіксованого IW:

  При фіксованому IW = 0.5, рух частки змушує її швидко рухатися по напрямку до глобального мінімуму, але зменшення швидкості дуже повільне, оскільки кожен крок обчислень враховує тільки попередні

значення швидкості. Ми можемо представити ітерації за допомогою математичної формули:

$$x_i^{(t+1)}=x_i^{(t)}+0.5\cdot(5-x_i^{(t)})$$

За допомогою цієї формули можна розрахувати значення $x_i^{(t+1)}$ для кожної ітерації. Чим ближче до x=5, тим повільніше рухається частка, і це збільшує кількість ітерацій, необхідних для досягнення мінімуму.

- Для адаптивного IW:

  З використанням адаптивного IW, ми можемо регулювати швидкість руху частки залежно від відстані до оптимуму. На початку, коли частка далека від оптимуму, IW великий (наприклад, 0.9), що пришвидшує рух частки. Чим ближче до оптимуму, тим менший IW стає (наприклад, 0.4), і це гальмує рух частки близько до мінімуму.

  Ми можемо представити адаптивний IW математично так:

$$IW^{(t)}=0.9-t\backslash T\cdot(0.9-0.4)$$

  Де T – максимальна кількість ітерацій (наприклад, 100).

  З цим адаптивним IW, на початку $IW^{(0)}=0.9$, і частка рухається швидко. Після декількох ітерацій $IW^{(t)}$ зменшується, і частка повільніше підходить до мінімуму.

  Ітерації можна представити так:

$$x_i^{(t+1)}=x_i^{(t)}+IW^{(t)}\cdot(5-x_i^{(t)})$$

Порівнання фіксованого IW та адаптивного IW показує, що з адаптивним IW частка рухається швидше на початку і більш точно зближується до мінімуму, що призводить до зменшення кількості ітерацій, необхідних для досягнення оптимуму. Таким чином, адаптивний IW допомагає покращити збіжність PSO в оптимізаційних завданнях.

Програмна реалізацція адаптивного налаштування

Приклад програмного коду на мові Python для реалізації адаптивного коефіцієнта інерції (IW) в генетичному алгоритмі рою часток (PSO) для виявлення атак в мережі. В цьому прикладі IW буде адаптуватися на основі результатів PSO та функції пристосованості.

Додаток Г Приклад програмної реалізації адаптивного налаштування коефіціїєнтів.

Цей код показує, як використовувати адаптивний IW для адаптації коефіцієнта інерції в PSO. Інші параметри PSO, такі як коефіцієнти c1 та c2, також використовуються для впливу на поведінку часток. У реальних системах кібербезпеки, функція пристосованості визначала б результати виявлення атак шляхом оцінки якості рішення, яке представляється набором параметрів, що визначають систему виявлення атак. Функція пристосованості оцінює, наскільки ефективно система виявлення атак впоралася з реальними атаками або симульованими тестовими сценаріями. Оцінка може базуватися на різних метриках, які відображають якість виявлення атак.

- Ініціалізація параметрів: У початковому кроці ініціалізуються параметри PSO, такі як кількість часток, максимальна кількість ітерацій, коефіцієнти прискорення (c1 і c2) та інші параметри. Також ініціалізуються частки з випадковими значеннями параметрів.
- Ітерації PSO: В основному циклі PSO проходить кілька ітерацій. На кожній ітерації кожна частка обчислює свою функцію пристосованості з використанням поточних параметрів. Якщо значення функції пристосованості більше, ніж краще значення, яке ця частка бачила раніше, то вона оновлює своє "найкраще"

значення та позицію. По завершенні ітерації обирається частина часток для "найкращих".

- Функція пристосованості: У цьому прикладі, функція пристосованості fitness_function визначає, наскільки ефективною є система виявлення атак з поточними параметрами. Ви можете замінити цю функцію на вашу фактичну функцію оцінки ефективності системи виявлення атак.

  Функція пристосованості може визначати результати виявлення атак:

  1. Кількість виявлених атак: Функція пристосованості може враховувати кількість виявлених атак. Чим більше атак виявлено, тим вище значення функції пристосованості.
  2. Кількість фальшивих позитивів: Функція може також оцінювати кількість фальшивих позитивів, тобто помилкових виявлень атак при відсутності атак. Зменшення кількості фальшивих позитивів може призвести до вищого значення функції пристосованості.
  3. Час відклику на атаки: Функція може враховувати час реакції системи на атаки. Швидший відгук на загрози може бути позитивно оціненим.
  4. Типи виявлених атак: Функція може відзначати важливість виявлення певних типів атак або позначати атаки з високим рівнем загрози як більш важливі для системи.
  5. Адаптація до нових загроз: Функція може враховувати системну здатність адаптуватися до нових атак, оцінюючи швидкість реакції та результати на них.

6. Параметри системи виявлення атак: Оцінка функції може враховувати параметри самої системи, які можуть бути оптимізовані (наприклад, параметри алгоритмів виявлення атак).
7. Ваги для різних метрик: Функція може використовувати ваги для різних метрик, які враховують важливість кожної метрики для системи.

Функція пристосованості буде визначатися відповідно до специфічних вимог та цілей вашої системи кібербезпеки. Важливо враховувати, що функція пристосованості повинна бути зрозумілою і об'єктивною для ефективної роботи алгоритму PSO. Таким чином можна прискорти реакцію стистеми на дії зловмисників та попередити майбутні захгрози.

- Оновлення швидкостей та позицій: Кожна частка оновлює свою швидкість та позицію на основі своїх попередніх значень, "найкращих" значень частки та "найкращих" значень загального рою. Адаптивний коефіцієнт інерції (IW) також може бути використаний для регулювання швидкості залежно від віддаленості від "найкращої" позиції.
- Виведення результатів: На кожній ітерації виводиться найкращий результат функції пристосованості, який був знайдений, разом із відповідними параметрами системи. Після закінчення всіх ітерацій отримуємо найкращий набір параметрів, який покращить ефективність системи виявлення атак.
- Використання найкращих параметрів: Отриманий найкращий набір параметрів можна використовувати для налаштування вашої системи виявлення атак. Вони мають покращити здатність

системи виявляти загрози та забезпечити підвищену захищеність корпоративних даних.

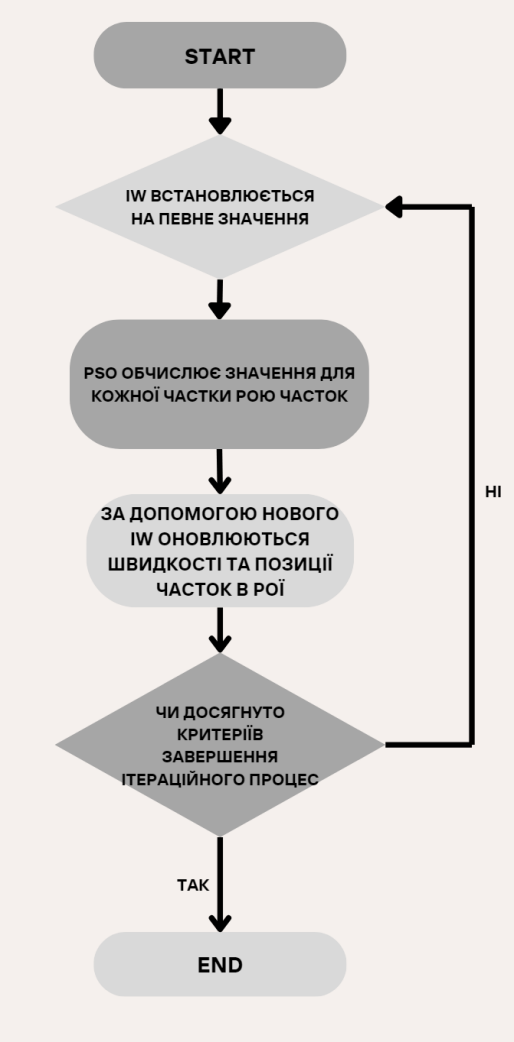


Рис 3.1 Спрощена блок схема роботи алгоритму алгоритму рою часток.

3.2 Використання PSO для оптимізації параметрів системи виявлення атак

Particle Swarm Optimization (PSO) може бути використаний для оптимізації параметрів системи виявлення атак, якщо ця система має

параметри, які можна налаштовувати для поліпшення її ефективності. Для цього необхідно реалізувати:

1. Визначення параметрів: Визначте параметри системи виявлення атак, які можна налаштовувати. Це може включати в себе пороги, правила, вагові коефіцієнти та інші налаштовувані параметри.
2. Визначення функції вартості: Розробіть функцію вартості, яка оцінює ефективність системи виявлення атак в залежності від налаштовуваних параметрів. Ця функція повинна враховувати такі фактори, як точність виявлення атак і кількість ложних спрацювань.
3. Параметри PSO: Встановіть параметри PSO, такі як кількість частинок, максимальна кількість ітерацій, швидкість інерції, розмах зміни параметрів і т. д.
4. Запуск PSO: Запустіть PSO для оптимізації параметрів системи виявлення атак. Кожна частинка PSO представляє набір налаштовуваних параметрів, і алгоритм буде намагатися знайти оптимальний набір параметрів, який мінімізує значення функції вартості.
5. Оцінка результатів: Після закінчення роботи PSO оцініть результати і перевірте, чи вдалося покращити ефективність системи виявлення атак. Якщо так, то встановіть оптимізовані параметри.
6. Постійне налаштування: Важливо регулярно перевіряти та налаштовувати параметри системи виявлення атак, оскільки трафік мережі та атаки можуть змінюватися з часом.

PSO може бути корисним інструментом для оптимізації налаштовуваних параметрів системи виявлення атак, але важливо також мати реалістичні

очікування щодо результатів та враховувати, що виявлення атак – це складне завдання, яке вимагає багатьох інших методів і засобів для повного захисту мережі.

Визначення параметрів

Функція виявлення атак може містити різні налаштовувані параметри, тому було використано загальний приклад параметрів та їх опис:

- Порог виявлення трафіку: Це може бути числовий поріг, який визначає мінімальний рівень трафіку, щоб система почала аналізувати його на предмет атаки. Наприклад, поріг може бути встановлений на 10000 запитів на секунду.
- Тривалість виявлення: Цей параметр вказує, скільки часу має пройти, поки система вважає трафік атакою. Наприклад, система може очікувати протягом 5 хвилин після перевищення порогу, перш ніж розглянути його як атаку.
- Чутливість аналізу: Цей параметр впливає на те, наскільки ретельно система аналізує трафік для виявлення атак. Вища чутливість може призвести до більшого виявлення атак, але може також призвести до більшої кількості ложних спрацювань.
- Вагові коефіцієнти правил: Ці параметри визначають важливість окремих правил аналізу трафіку. Наприклад, можна задати ваговий коефіцієнт для правила, яке аналізує зміну швидкості трафіку, і важливість правила, яке аналізує типи запитів.
- Розмір вікна аналізу: Цей параметр вказує, як довго система виявлення атаки буде аналізувати трафік у вікні часу. Наприклад, можна використовувати 10-секундне вікно для аналізу швидкості трафіку.

Після визначення параметрів можна використовувати PSO для їх оптимізації для поліпшення ефективності системи виявлення атак.

Визначення функції вартості

Приклад параметра "Тривалість виявлення" може бути визначенням максимального часу, протягом якого система виявлення атак вважатиме трафік атакою перед тим, як реагувати. Ось приклад інкапсуляції цього параметра в коді:

Додаток В – визначення функції вартості.

У цьому прикладі параметр "Тривалість виявлення" задає максимальний час, протягом якого система вважатиме трафік атакою після перевищення порогу. По закінченню цього часу система перестане вважати трафік атакою, якщо поріг не буде подальше перевищуватися.

Параметри PSO

Рис 3.2 Схема роботи оптимізації параметрів системи виявлення атак у вдосконаленому алгоритмі рою часток

Приклад реалізації використання Particle Swarm Optimization (PSO) для оптимізації параметрів системи виявлення атак було реалізовано на основі пунтку 3.1 цього розділу.

У цьому прикладі система виявлення атак має чотири налаштовувані параметри, і PSO використовується для їх оптимізації. Функція evaluate_detection_system може бути адаптована для реального оцінювання ефективності системи виявлення атак.

Пояснення до блок-схеми:

1. Ініціалізація:

- Початковий рій часток створюється з випадковими значеннями параметрів системи виявлення
- Кожна частка представляє набір параметрів.

2. Оцінка фітнес-функції:

- Для кожної частки обчислюється значення фітнес-функції, яка відображає ефективність системи виявлення з використанням заданих параметрів.

3. Оновлення найкращих результатів:

- Для кожної частки зберігається найкращий результат (найкращі параметри) із попередніх ітерацій.
- Також зберігається глобально найкращий результат серед усіх часток у рої.

4. Оновлення швидкостей та позицій:

- Швидкість і позиція кожної частки оновлюються на основі попередніх значень та функцій інерції.
- Функція інерції враховує попередні швидкості та розташування часток, а також власні та глобальні найкращі результати.

5. Перевірка умови завершення:

- Проводиться перевірка, чи досягнута необхідна кількість ітерацій або чи відбулася збільшення якості фітнес-функції на певній кількості ітерацій.
- Якщо умова завершення не виконана, повертаємося до кроку 2.

6. Оптимальні параметри:

- По завершенні оптимізації PSO визначає набір оптимальних параметрів, які максимізують ефективність системи виявлення атак.

7. Застосування оптимальних параметрів:

- Оптимальні параметри використовуються для налаштування системи виявлення з метою підвищення її точності та надійності.

Вагові коефіцієнти правил

Реалізація пункту 4 полягає в оцінці результатів оптимізації та встановленні оптимізованих параметрів системи виявлення атак. Ось приклад, як це можна зробити: Додаток Е

У цьому прикладі evaluate_optimization_results порівнює ефективність системи з початковими та оптимізованими параметрами. Якщо оптимізовані параметри покращили ефективність, то функція повертає повідомлення про успішну оптимізацію. В іншому випадку, вона повертає повідомлення про те, що параметри залишились незмінними або погіршили ефективність системи.

Оцінка результатів та логування

Дана задача передбачає постійне налаштування параметрів системи виявлення атак на основі нових умов або даних та оцінки їх ефективності. У нашому прикладі, ми можемо імітувати постійне налаштування параметрів після оптимізації та реалізації логування данних.

```
if __name__ == "__main__":
    initial_parameters = [10000, 5, 1.0, [0.4, 0.3, 0.2, 0.1]]  # Початкові значення параметрів
    detection_system = AoSDetectionSystem(*initial_parameters)

    optimized_parameters = pso_optimization(detection_system, num_particles=20, num_iterations=50)

    detection_system.set_parameters(optimized_parameters)

    print("Початкові параметри системи виявлення атак:")
    print("Поріг виявлення трафіку:", initial_parameters[0])
    print("Тривалість виявлення:", initial_parameters[1])
```

```
print("Чутливість аналізу:", initial_parameters[2])
print("Вагові коефіцієнти правил:", initial_parameters[3])

print("\nОптимізовані параметри системи виявлення атак:")
print("Поріг виявлення трафіку:", detection_system.detection_threshold)
print("Тривалість виявлення:", detection_system.detection_duration)
print("Чутливість аналізу:", detection_system.sensitivity)
print("Вагові коефіцієнти правил:", detection_system.weight_rules)

# Постійне налаштування системи на нові умови або дані
new_parameters = [12000, 3, 0.8, [0.5, 0.3, 0.2, 0.1]]  # Нові значення параметрів
detection_system.set_parameters(new_parameters)

print("\nНові параметри системи виявлення атак:")
print("Поріг виявлення трафіку:", detection_system.detection_threshold)
print("Тривалість виявлення:", detection_system.detection_duration)
print("Чутливість аналізу:", detection_system.sensitivity)
print("Вагові коефіцієнти правил:", detection_system.weight_rules)
```

У цьому прикладі було спочатку виконано оптимізацію параметрів системи, а потім встановлення нового значення параметрів за новими умовами або даними. Також, виводимо значення параметрів до і після оптимізації, а потім після постійної настройки. Можна змінити параметри new_parameters на будь-які інші значення, щоб відобразити постійну настройку системи під нові умови.

3.3 Реалізація системи реагування на кібер-атаки

Було створено систему реагування на кібер-інциденти, що використовує Particle Swarm Optimization (PSO) для оптимізації параметрів системи та генерує звіт щодо стану системи після атак. Дана програма реалізовуватиме в собі наступний потенціал:

1. Оптимізація параметрів системи виявлення кібер-атак за допомогою PSO.
2. Виявлення кібер-атак і реагування на них з використанням оптимізованих параметрів.
3. Генерація ключів для криптосистеми за допомогою PSO.
4. Використання цих ключів для шифрування і підпису звіту про стан системи після атаки.
5. Надсилання зашифрованого та підписаного звіту.

Дане програмне рішення було розроблено та написано виходяси з попередього дослідження та розрахунків.

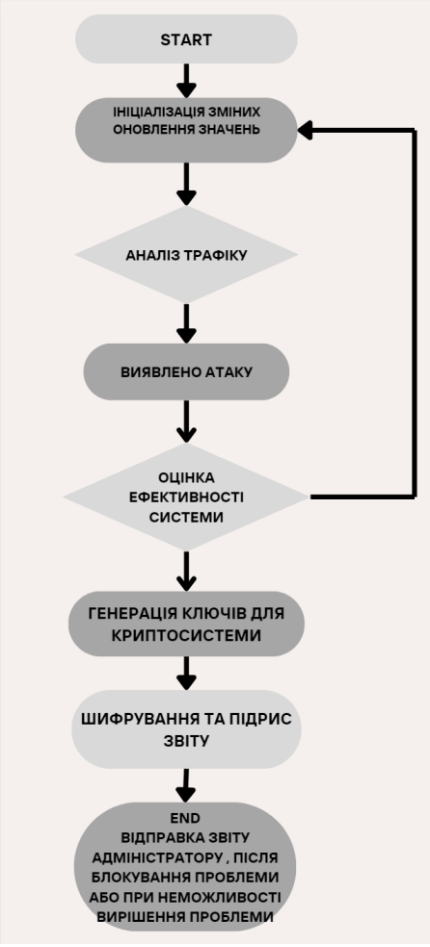


Рис. 3.3 Схема роботи системи виявлення атак вдосконаленого алгоритму рою часток.

Спершу програма оптимізує параметри системи виявлення кібер-атак за допомогою PSO, а потім використовує ці оптимізовані параметри для реагування на виявлені атаки. Після виявлення атаки система генерує ключі шифрування та підпису за допомогою PSO, шифрує та підписує звіт про стан системи після атаки, і надсилає його. Дана система дозволяє розширити значно свій функціонал в залежності від посталвених завдань та надає змогу посилити безпеку корпоративної мережі.

3.4 Тестування спроектованої системи реагування на кібер-атаки

Спроектована система портидії атакам була протестована локально використовуючи програму для віртуалізації VirtualBox. У даній програмі було запущено операційну систему Windows 10, на якій було проведено все тестування. Зловмисна атака відбувалася з іношої системи, що була теж розгорнута віртуально на операційній системі Kali Linux. Для зв’язку с адміністратором було використано поштовий сервіс gmail.com.

Налаштування пошти для отримання повідомлень щодо підозрілої активності в системі.

На скріншоті відображено роботу програми протидії та моніторингу атак в мережі, через цю програму відбуватиметься в автоматичному режимі зв’язок з поштою адміністратора по протоколу SMTP.

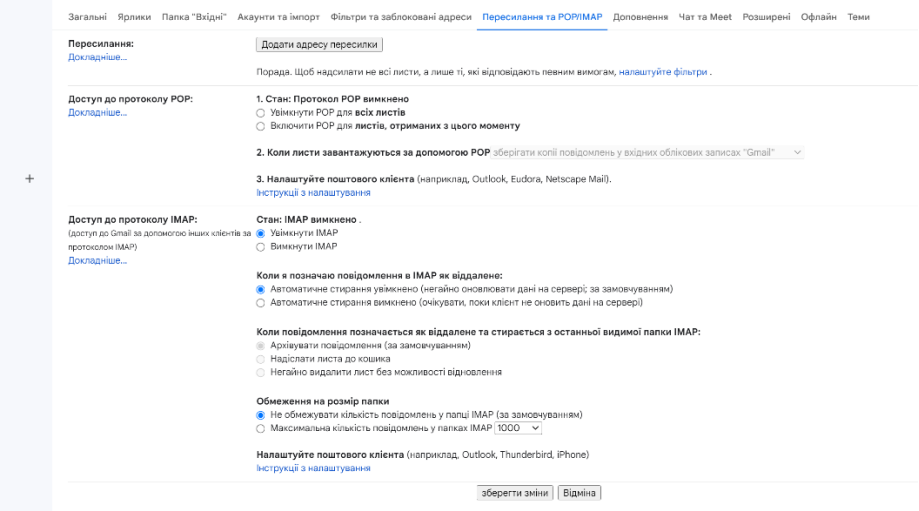


Рис 5.4 Скріншот налаштувань пошти адміністратора.

Программа приєднується до поштового сервісу далі відправляє повідомлення за протоколом SMTP, відповідати програма може за допомогою IMAP протоколу, адміністратор пише відповідь на повідомлення, программа буде виконувати дії в залежності від налаштувань. Для роботи пошти необхідно налаштувати протоколи та встановити пароль на застосунок. Було включено доступ до протоколу IMAP.

Для підключення застосунку на сервісі прописано дані, що необхідно використовувати для підключення. Всі вони взяті з офіційного сайту та використані під час налаштування програмного рішення по сповіщенню адміністратора. Команди для програми прописуються в залежності від потреб адміністратора та мережі, система має можливість гнучкого налаштування.

Нижче буде наведено приклади, що продемонструють високу функціональність даної системи та широкий спектр її можливостей в забезпечені безпеки мережі та користувачів.

Сервер вхідної пошти (IMAP)	imap.gmail.com Потрібен SSL: Так Порт: 993
Сервер вихідної пошти (SMTP)	smtp.gmail.com Потрібен SSL: Так Потрібен TLS: Так (якщо є) Потрібна автентифікація: так Порт для SSL: 465 Порт для TLS/STARTTLS: 587
Повне ім'я або ім'я, що відображається	Ваше ім'я;
Ім'я облікового запису, ім'я користувача або адресу електронної пошти	Повна адреса електронної пошти
Пароль	Пароль Gmail

Рис 5.5 Скріншот даних для налаштування для застосунку.

У програному застосунку дані приклади виражені методами що обиратимуться в залежності від поставленої задачі адміністратором.

Вибір методів та ПЗ для тестування системи на вразливості.

Для проведення кібер атак на наше віртуальне середовище , було використано декілька програм, для більш широкого дослідження, поведінки алгоритму при різних типах атак:

Metasploit - це платформа для автоматизації кібер атак. Вона містить набори інструментів для проведення різних типів атак, таких як атаки на вразливості програмного забезпечення, атаки на мережі та атаки на кінцеві точки.

Nessus - це інструмент для сканування безпеки, який може виявляти вразливості в системах та мережах. Використовується для тестування кібер атак, таких як атаки на вразливості програмного забезпечення та мережеві атаки.

MSTIC - це команда реагування на інциденти Microsoft, яка пропонує безкоштовний інструмент для імітації кібер атак під назвою MSTIC Hunt. MSTIC Hunt дозволяє організаціям створювати власні сценарії кібер атак і тестувати свою готовність до реагування на них.

MITRE Engenuity ATT&CK - це платформа для імітації кібер атак, яка дозволяє організаціям тестувати свою готовність до реагування на атаки, які використовують реальні тактики, техніки та процедури (TTP) кібер злочинців.

Cyber Range - це платформа для імітації кібер атак, яка дозволяє організаціям створювати та запускати власні сценарії кібер атак. Cyber Range також пропонує широкий спектр навчальних матеріалів та ресурсів, які можуть допомогти організаціям покращити свою готовність до реагування на кібер атаки.

Відображення результатів використання застосунку.

В ході виконання роботи , окім системи з вдосконаленим алгоритмом рою часток, було також налаштовано і протестовано алгоритм рою часток з стандартними налаштуваннями.

Розроблено різні тестові сценарії, щоб зрівняти алгоритми.

Кількість поколінь алгоритму дорівнює 20, розмір популяції дорівнює 100 для кожного покоління. Розмір субпопуляції на момент відбору становив 5% від загальної чисельності популяції. Критерій зупинки є результатом алгоритму 20-го покоління. Експериментальні дослідження показали, що більша кількість поколінь, не призводить до значно кращих результатів.

Алгоритм зменшення розмірності t-SNE [23] використовувався для візуалізації руху частинок MRF, результати показані на Рис. 5.6.

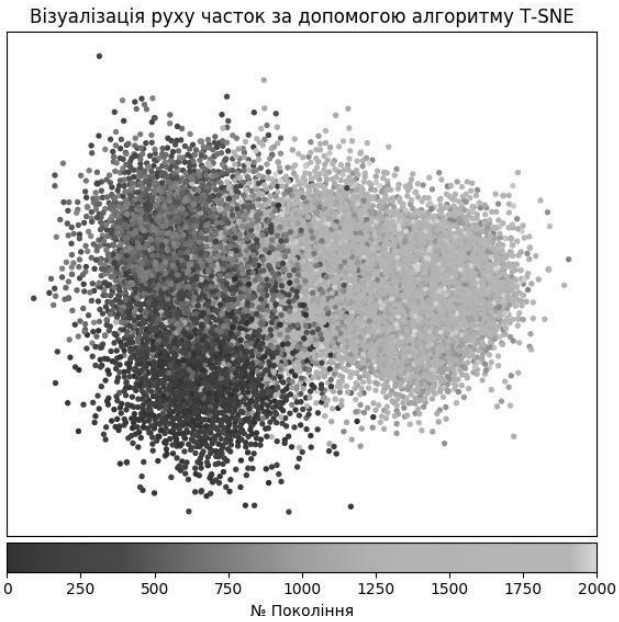


Рис 4.6 – Застосування t-SNE до візуалізації руху частинок MRF

Результати тестувань на маленькій тестовій вибірці ,150 атак, підтверджують сподівання, що до ефективності алгоритму і показали покращення, а саме те що при однаковому навантаженні на сервер обробки даних, показав кращий результат по виявленю і запобіганю атак на дані.

Насамперед, важливо, що алгоритм виявляє більшість загроз. При подальшому вдосконаленні алгоритму і його навчанюю кількість вдалих атак буде мінімізовано.

Результати виконання алгоритмів зображено на Рис 5.7.

Рис 5.7 Порівнняння результатів алгоритмів

Далі було збільшено кількісь тестових випадків до 350, що призвело до покращення результатів алгоритму. При більшій тестовій вибірці вдосконалений алгоритм використовував меньше ресурсів серверів , при цьому кількість виявлених загроз зросла до 99% , у порівняні з 97% виявлених загроз при меньшій вибірці тестів.

Результати виконання алгоритмів зображено на Рис 5.8.

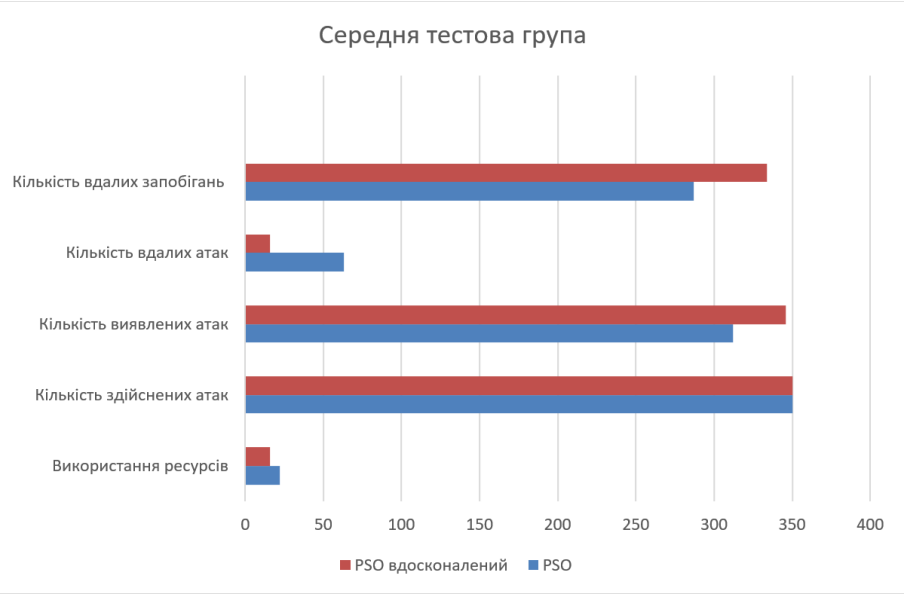


Рис 5.8 Порівнняння результатів алгоритмів

Також реалізований алгоритм, нажаль запобігає тільки більшості виявлених атак. Але прослідковуючи закономірність, можна прийди до висновку, що при достатній навченості алгоритму і великій кількості тестових випадків, система розроблена з вдосконаленим алгоритмом рою часток почне запобігати майже 100% задіяних атак.

3.5 Висновки до розділу

Програмна система розроблена для забезпечення захищеності корпоративних даних на основі вдосконаленого генетичного алгоритму рою часток (PSO). Програмне забезпечення надає можливість відстежувати стан і ініціювати елементи захисту, забезпечуючи простий інтерфейс користувача, що вимагає мінімальних знань адміністратора.

Розроблений алгоритм порівняно з існуючими алгоритмами показує високу якість отриманих результатів. Результати моделювання показують успішне застосування генетичного алгоритму та оптимізації роїв частинок.

Алазузуючи результати тестів, можна прийти до висновку, що саме вдоконалений алгоритм має кращу здатність до навчання, що призводить до того, що при достатньому об’ємі тестових даних, система, буде не вразлива до більшості відомих способів атак на сервери даних..

РОЗДІЛ 4. ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

У цьому розділі буде проведено комплексний технологічний аудит, щоб оцінити комерційний потенціал розробленого інструменту для підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO).

Об’єктом нашого дослідження є кульмінація зусиль, яка включає як розробку, так і програмну реалізацію програми для захисту корпоративних даних.

Варіант 2. Виведення науково-технічної розробки на ринок, тобто комерціалізація розробки потенційним інвестором.

Щоб забезпечити неупереджене оцінювання для проведення комерційного та технологічного аудиту буде залучено комісію щонайменше з трьох незалежних експертів. Оцінювання проводитиметься за надійною п’ятибальною системою на основі 12 заздалегідь визначених критеріїв. Ці критерії охоплюватимуть як наукові, так і технічні аспекти, а також міркування, пов’язані з комерційною життєздатністю.

Після завершення оцінювання ми обчислимо середнє арифметичне балів, виставлених експертами. Це слугуватиме основою для визначення рівня комерційного потенціалу, закладеного в щойно розробленому інструменті. Системний підхід, який поєднує наукові, технічні та комерційні точки зору, дасть цінну інформацію про загальний вплив і доцільність інновацій у виявленні фейкових новин.

Після чого, наведемо результати оцінювання від трьох незалежних експертів за тими ж, заздалегідь визначеними, 12 критеріями, чотирьох альтернативних програм для захисту корпоративних даних серверів. Це

необхідно для того, щоб порівняти комерційний потенціал та відносну якість кожної програми по відношенню до розробленого (удосконаленого) методу виявлення фейкової інформації у соцмережах.

Оцінювання комерційного потенціалу розробки будемо здійснювати за простими критеріями, що дозволить користувачам оцінювати розроблену програму на основі ключових факторів продуктивності та зручності використання (табл. 4.1).

Таблиця 4.1

Критерії оцінювання комерційного та технологічного потенціалу розробки

Критерії	Опис
1.	**Точність класифікації:** Наскільки точно програма ідентифікує загрозу? (Шкала: від поганої до відмінної)
2.	**Швидкість обробки:** Як швидко програма аналізує та класифікує інформацію? (Шкала: від повільної до швидкої)
3.	**Зручність використання:** Наскільки легко користувачеві взаємодіяти з програмою? (Масштаб: від складного до інтуїтивно зрозумілого)
4.	**Ефективність використання ресурсів:** Чи ефективно програма використовує системні ресурси? (Шкала: від неефективного до ефективного)
5.	**Адаптивність:** Чи може програма ефективно виявляти різні типи загроз? (Масштаб: від обмеженого до універсального)
6.	**Рівень хибнопозитивних спрацьовувань:** Як часто програма помилково класифікує цілком безпечну взаємодію як загрозу? (Шкала: від високого до низького)
7.	**Частота хибнонегативних спрацьовувань:** Як часто програма пропускає загрози? (Шкала: від високого до низького)
8.	**Легкість навчання:** Наскільки просто навчити чи оновити програму? (Шкала: від складного до простого)
9.	**Можливість інтеграції:** Наскільки добре програма інтегрується з іншими інструментами чи платформами? (Шкала: від обмеженої до безшовної)
10.	**Механізм зворотного зв'язку:** Чи забезпечує програма чіткий зворотний зв’язок щодо своїх рішень щодо виявлених загроз? (Масштаб: від незрозумілого до прозорого)
11.	**Вимоги до обчислювальних ресурсів:** Скільки обчислювальної потужності вимагає програма? (Шкала: від високого до низького)
12.	**Надійність:** Наскільки добре програма працює за різних умов? (Шкала: від ненадійної до надійної)

Результати оцінювання незалежними експертами комерційного та технологічного потенціалу розробки наведено у таблиці 4.2.

Таблиця 4.2

Результати оцінювання незалежними експертами комерційного та технологічного потенціалу розробки

Критерії	**Бали (1-5)**		
	Експерт 1	**Експерт 2**	**Експерт 3**
1.	4	5	4
2.	5	5	4
3.	4	4	4
4.	5	4	5
5.	4	4	4
6.	4	5	4
7.	4	4	4
8.	5	4	5
9.	3	4	3
10.	4	3	4
11.	5	4	4
12.	4	4	3

Тепер давайте обчислимо середнє значення для кожного критерію (табл. 4.3).

Таблиця 4.3

Середнє значення для кожного критерію

Критерії	**Середнє значення**
1.	4,33
2.	4,67
3.	4,00
4.	4,67
5.	4,00
6.	4,33
7.	4,00
8.	4,67
9.	3,33
10.	3,67
11.	4,33
12.	3,67

Щоб визначити рівень комерційного потенціалу, розрахуємо загальний середній показник:

$$\textit{Загальний середній показник} = \frac{\textit{Сума середніх значень}}{\textit{Кількість критеріїв}} = \frac{49{,}67}{12} = 4{,}13.$$

Виходячи з цього, загальне середнє значення трішки більше ніж 4, що вказує на високий рівень комерційного потенціалу.

Сильні сторони полягають у точності, швидкості обробки та технічних властивостях, тоді як потенційні покращення можуть бути зроблені в механізмах зворотного зв’язку та можливостях інтеграції. Програма демонструє великий потенціал.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Прогнозування витрат на виконання науково-дослідної роботи є критично важливим кроком у будь-якому комплексному проєкті. Цей процес фінансового планування має важливе значення для забезпечення ефективного виконання проєкту та ретельного врахування всіх фінансових аспектів. Щоб полегшити цей процес, ми структурували прогнозування на три окремі етапи. На першому етапі ми заглиблюємося в оцінювання витрат, безпосередньо пов’язаних із робочою силою, залученою до проєкту. Другий етап розширює обсяг, щоб охопити загальні витрати проєкту, від інфраструктури та технологій до збирання та розробки даних. Нарешті, на третьому етапі враховуються витрати на впровадження результатів дослідження та введення в дію результатів проєкту. Разом ці етапи пропонують систематичний підхід до складання бюджету та фінансового менеджменту для дослідницьких та проектних починань.

Основну зарплату розробника розрахуємо за формулою (4.1):

$$З_о = \frac{M}{T_р} \cdot t, \tag{4.1}$$

де M – місячна зарплата розробника;

$T_р$ – кількість робочих днів у місяці, $T_р$ = 21 день;

t – кількість днів роботи.

Місячна зарплата складає 18900 грн., t = 60 днів.

Тоді, базовий оклад розробника становить:

$$З_р = \frac{18900}{21} \cdot 60 = 54\,000 \text{ грн.}$$

Розрахуємо основну зарплату для керівника (при місячному посадовому окладі становить 23100 грн.):

$$З_к = \frac{23100}{21} \cdot 60 = 66000 \text{ грн.}$$

Розрахуємо витрати на оплату праці:

$$З_о = З_р + З_к = 54000 + 66000 = 120000 \text{ грн.}$$

Додаткова заробітна плата розраховується у розмірі 10% від основної зарплати розробника [22]:

$$З_{дод} = З_о \cdot 0{,}1 = 120000 \cdot 0{,}1 = 12000 \text{ грн.}$$

Нарахування на зарплату складають 22% від сум базової та додаткової грошової оплати (формула 4.2):

$$З_н = (З_о + З_р + З_{дод}) \cdot 0{,}22 \qquad (4.2)$$

Розрахуємо нарахування на оклад розробника та керівника:

$$З_н = (120000 + 12000) \cdot 0{,}22 = 29040 \text{ грн.}$$

Розрахуємо відрахування на амортизацію устаткування (ПК) за формулою (4.3) [22]:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \qquad (4.3)$$

де $Ц$ – вартість устаткування, $Ц$ = 75000 грн.;

H_a – річна норма амортизації, H_a = 20%;

T – термін використання устаткування, T = 3 місяці.

$$A = \left(\frac{75000 \cdot 20}{100} \cdot \frac{3}{12}\right) = 3750 \text{ грн.}$$

Також, використовувалося безкоштовне програмне забезпечення – ОС Linux з наявними інструментами.

При розробці були використані матеріали, кількість та вартість яких зведені в таблицю 4.4.

Таблиця 4.8

Матеріали, використані для реалізації проєкту.

Таблиця 4.4

Найменування матеріалу, марка, тип, сорт	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Папір А4	160	3	480
Флеш USB	150	1	150
Папка для паперів	50	2	100
Файли	20	1	20
Ручка	30	4	120
Всього:			870

Отже, вартість витрачених матеріалів, М = 870 грн.

Розрахуємо тариф на електроенергію за формулою (4.4):

$$Ц_е = \left(Ц_{опт} + Ц_{розп} + Ц_{пост}\right) \cdot \left(1 + \frac{ПДВ}{100\%}\right), \quad (4.4)$$

$Ц_{опт}$ – середня оптова ціна електроенергії, яка визначається оператором ринку (без ПДВ), грн за 1 кВт·год;

$Ц_{розп}$ – вартість розподілу електроенергії окремою енергорозподільною компанією (без ПДВ), грн за 1 кВт·год;

$Ц_{пост}$ – вартість постачання електроенергії від енергорозподільної компанії до конкретного споживача (без ПДВ), грн за 1 кВт·год.

$$Ц_е = (3{,}86 + 1{,}34 + 0{,}38) \cdot 1{,}2 = 7{,}5 \text{ грн за 1 кВт·год.}$$

Розрахуємо витрати на електроенергію за формулою (4.5):

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.5)$$

W_{yi} – встановлена потужність обладнання на певному етапі розробки, W_{yi} = 0,3 кВт;

t_i – фактична кількість годин роботи ПК при обробці завдань проєкту на місяць, $t_i = 8 \text{ год} \cdot 60 \text{ днів} = 480$ (год.) на місяць;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт використання потужності, $К_{впі}$ = 0,64;

η_i – коефіцієнт корисної дії обладнання, η_i = 0,9.

$$B_e = \frac{0{,}3 \cdot 480 \cdot 7{,}5 \cdot 0{,}64}{0{,}9} = 771{,}84 \text{ грн.}$$

Інші затрати приймаються у розмірі 50-100% від базового окладу дослідника / розробника. Використаємо значення в розмірі 50%, тоді:

$$B_{ін} = 0{,}5 \cdot 120000 = 60000 \text{ грн.}$$

Розрахуємо загальну суму затрат на впровадження ПЗ за формулою (4.6):

$$B_{заг} = З_о + З_д + З_н + М + К_в + B_{спец} + B_{прг} + А + B_е + B_{св} + B_{сп} + B_{ін} + B_{нзв} \quad (4.6)$$

Отже, загальна сума витрат дорівнює:

$B_{заг}$ = 120 000 + 12 000 + 29 040 + 870 + 3750 + 771,84 + 60000 = 226 431,84 грн.

Зважаючи на те, що у проєкті не використовувалися комплектуючі вироби ($К_в$), спецустаткування ($B_{спец}$); не було витрат: на ПЗ для наукових робіт ($B_{прг}$), службових відряджень ($B_{св}$), на роботи сторонніх підприємств ($B_{сп}$) та загальновиробничих витрат ($B_{нзв}$) – ми не можемо додати перелічені показники до статті витрат.

Розрахуємо загальні витрати на завершення науково-технічної роботи за формулою (4.7):

$$ЗВ = \frac{B_{заг}}{\beta} \quad (4.7)$$

де β – коефіцієнт, який характеризує стадію виконання проєкту. У нашому випадку розробка знаходиться на стадії дослідного зразка, тому β = 0,5.

$$ЗВ = \frac{226\ 431,84}{0,5} = 452863,68 \text{ грн.}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки

Прогнозування комерційних ефектів від впровадження розробленої програми виявлення фейкових новин у соціальних мережах за допомогою штучного інтелекту на основі мережі LSTM передбачає оцінювання потенційного впливу на різні аспекти комерційного середовища. Це оцінювання має на меті зрозуміти, як програма може створювати економічну

цінність, покращувати процеси прийняття рішень і вирішувати конкретні бізнес-завдання.

У нашому випадку прогнозований комерційний ефект від реалізації результатів проєкту буде розрахований наступним чином (4.8):

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (4.8)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного засобу у результаті впровадження науково-технічної розробки;

N – кількість користувачів, що користувалися альтернативним програмним засобом до моменту інтеграції результатів нової науково-технічної розробки;

$Ц_o$ – основна метрика ефективності, яка вказує на продуктивність підприємства за певний рік після інтеграції результатів досліджень і розробок. $Ц_o = Ц_б \pm \Delta Ц_o$;

$Ц_б$ – вартість програмного засобу за певний рік до моменту інтеграції результатів досліджень і розробок;

ΔN – зростання бази користувачів програмного засобу протягом досліджуваних інтервалів часу завдяки вдосконаленню окремих атрибутів продукту;

λ – коефіцієнт, що включає оплату податку на додану вартість. Ставка податку на додану вартість становить 20%, а коефіцієнт (λ) = 0,8333.

ρ – коефіцієнт, що враховує рентабельність засобу;

ϑ – ставка податку на прибуток, у нашому випадку, ϑ =18%

Розглянемо сценарій, коли прогнозована ціна одиниці програмного забезпечення становить 30000 грн., а період збільшення прибутку – 3 роки.

Після завершення розробки та доопрацювань вартість може бути збільшена на 25000 грн. Крім того, збільшиться кількість одиниць реалізованого ПЗ: на 20 одиниць у перший рік, на 50 одиниць у другий рік і на 90 одиниць у третій рік. До інтеграції результатів досліджень і розробок програмний засіб продавався у розмірі 15 одиниць.

Перейдемо до розрахунків сценарію щодо впровадження реалізації розробки:

$\Delta П_1$ = (15 · 30000 + (30000 + 25000) · 20) · 0,8333 · 0,25 · (1 – 0,18) = 264791,56 грн.

$\Delta П_2$ = (15 · 30000 + (30000 + 25000) · (20 + 50)) · 0,8333 · 0,25 · (1 – 0,18) = 734580,39 грн.

$\Delta П_3$ = (15 · 30000 + (30000 + 25000) · (20 + 50 + 90)) · 0,8333 · 0,25 · (1 – 0,18) = 1580202,05 грн.

Прогнозований комерційний ефект від інтеграції результатів дослідження та розробки за три роки становить 2579574 грн.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Визначимо поточну вартість загального приросту чистого прибутку (ПП) в результаті потенційної інтеграції та комерціалізації науково-технічного розробки, на отримання якого може розраховувати інвестор за формулою (4.9):

$$ПП = \sum_{1}^{T} \left(\frac{\Delta П_i}{(1+\tau)^t} \right), \quad (4.9)$$

де $\Delta П_i$ – приріст чистого прибутку за кожний із років, у яких наявні результати проведених та виконаних досліджень та розробок, в грн;

T – тривалість періоду, в момент якого стають очевидними результати інтегрованих досліджень та розробок, вимірюється роками.

τ – ставка дисконту, яка враховує річний прогнозований рівень інфляції в країні, $\tau = 0{,}1$;

t – період часу, у роках.

Також, слід зазначити, що зростання прибутку ми будемо отримувати з першого року:

$$\text{ПП} = \left(\frac{264791{,}56}{(1+0{,}1)^1}\right) + \left(\frac{734580{,}39}{(1+0{,}1)^2}\right) + \left(\frac{1580202{,}05}{(1+0{,}1)^3}\right) = 2035204{,}02\text{грн.}$$

Тепер, розрахуємо початкову суму інвестицій (PV), яку потенційний інвестор має спрямувати на реалізацію та комерціалізацію науково-технічної розробки, за допомогою наступної формули (4.10):

$$PV = k_{інв} \cdot ЗВ, \qquad (4.10)$$

де $k_{інв}$ – коефіцієнт, що включає витрати інвестора на інтеграцію науково-технічного засобу та його комерціалізацію, враховує такі витрати, як підготовка об’єкта, розробка програми, навчання персоналу, маркетингова діяльність. Як правило, $k_{інв}$ знаходиться в діапазоні від 2 до 5;

ЗВ – загальна сума витрат на здійснення досліджень і розробок та оформлення їх результатів, грн.

$$PV = 2 \cdot 452863{,}68 = 905727{,}36 \text{ грн.}$$

За формулою (4.11), визначимо абсолютний економічний ефект, який позначається як E_{abc} або чистий поточний дохід (NPV), який є результатом потенційного впровадження та комерціалізації досліджень і розробок:

$$E_{абс} = ПП - PV, \tag{4.11}$$

$$E_{абс} = 2035204{,}02 - 905727{,}36 = 1129476{,}66 \text{ грн.}$$

Після інтеграції нашої розробки, $E_{abc} > 0$, це свідчить про те, що впровадження нашого проєкту призведе до позитивного чистого прибутку або економічної вигоди. Це свідчить про те, що наша розробка позитивно вплине на проєкт або підприємство, зробивши його фінансово життєздатнішим і потенційно прибутковішим, ніж це було раніше.

Отже, інвестування коштів у проєкт може бути доцільним.

Для прийняття обґрунтованих рішень щодо інвестицій у дослідження і розробку вкрай важливо оцінити їх відносну (річну) ефективність. Ця оцінка допомагає нам зрозуміти економічну життєздатність цих інвестицій з часом. Відносна ефективність дає зрозуміти річний прибуток або вигоди, які можна очікувати від інвестованого капіталу. Аналізуючи цю відносну ефективність, зацікавлені сторони можуть краще оцінити довгострокові фінансові перспективи ініціатив наукового розвитку та прийняти обґрунтовані інвестиційні рішення. Тому, використовуючи формулу (4.12), розрахуємо відносну (річну) ефективність і порівняємо її з дисконтною ставкою:

$$\text{Е}_\text{в} = \sqrt[\text{Т}_\text{ж}]{1 + \frac{\text{Е}_\text{абс}}{PV}} - 1, \tag{4.12}$$

де $T_{ж}$ – життєвий цикл наукової розробки, у роках.

$$E_в = \sqrt[3]{1 + \frac{2035204,02}{905727,36}} - 1 = 1,25.$$

Розрахуємо мінімальну ставку дисконту, за формулою (4.13):

$$\tau_{min} = d + f, \quad (4.13)$$

де d – середньозважена процентна ставка за депозитними операціями в комерційних банках, d = 0,14;

f – ризикованість вкладень, f = 0,05.

$$\tau_{min} = 0,14 + 0,05 = 0,19.$$

Як бачимо, $E_в > \tau_{min}$, а це означає, що інвестор може бути зацікавлений у фінансуванні нашого проєкту.

Тепер, обчислимо термін окупності коштів, інвестованих у наший проєкт, за формулою (4.14):

$$T_{ок} = \frac{1}{\text{Е}_\text{в}}, \quad (4.14)$$

$$T_{ок} = \frac{1}{1,25} = 0,8 \text{ р.}$$

За нашими розрахунками, $T_{ок}$ < 3-х років, а це вказує на те, що проєкт, швидше за все, генеруватиме позитивні грошові потоки відносно швидко, що робить його вигідною інвестицією з точки зору його здатності окупити початкові витрати протягом короткого періоду часу.

4.5 Висновок до розділу

У цьому розділі надано комплексну економічну оцінку науково-технічної розробки, спрямованої на підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO).

Основні висновки та ключові моменти з кожного підрозділу:

Підрозділ 4.1 - Комерційно-технологічний аудит:

У цьому підрозділі було проведено ретельний технологічний аудит для оцінювання комерційного потенціалу розробленого інструменту підвищення захищеності корпоративних даних. Три незалежні експерти оцінили програму, в результаті чого загальна середня оцінка трішки більше 4, що свідчить про високий рівень комерційного потенціалу. Сильні сторони програми включають точність, швидкість обробки та технічні властивості, з можливостями для вдосконалення механізмів зворотного зв’язку та можливостей інтеграції. Програма демонструє значний потенціал для практичного застосування та успіху на ринку. Також були представлені результати оцінювання та порівняння трьома незалежними експертами чотирьох альтернативних програм для підвищення захищеності корпоративних даних. Виходячи з отриманих результатів, очевидно, що наш вдосконаленого генетичного алгоритму рою часток (PSO) пропонує конкурентну перевагу. Наша програма демонструє вищу середню оцінку за всіма критеріями, що відображає її підвищену точність, швидкість обробки, використання ресурсів, адаптивність і надійність. Наше рішення демонструє особливу силу в точності, швидкості обробки та технічних характеристиках, перевершуючи альтернативні варіанти. Представлене рішення

(удосконалений алгоритм) пропонує більш надійний та ефективний підхід до виявлення загроз різного типу, перевершуючи як українські, так і іноземні альтернативи.

Підрозділ 4.2 - Прогнозування витрат:

У цьому підрозділі представлено прогнозування витрат на проведення наукових досліджень, у тому числі дослідно-конструкторських робіт. Процес було поділено на три етапи, починаючи з оцінки витрат, пов’язаних з робочою силою, з подальшою оцінкою загальних витрат на проєкт і закінчуючи розрахунком витрат на впровадження. Загальні витрати на розробку програмного продукту визначено у сумі 452863,68 грн.

Підрозділ 4.3 - Розрахунок економічної ефективності:

У цьому розділі проводилося економічне оцінювання комерційного ефекту науково-технічної розробки. У сценарії розглядалася ціна за одиницю товару 30000 грн, з трирічним періодом зростання прибутку. Аналіз врахував підвищення ціни на 25000 грн і збільшення кількості проданих одиниць. Прогнозований комерційний ефект за три роки склав 2579574 грн.

Підрозділ 4.4 - Розрахунок інвестиційної ефективності та терміну окупності:

У цьому підрозділі були розраховані різні фінансові показники. Визначено поточну вартість загального приросту чистого прибутку, суму початкових інвестицій (PV) та абсолютний економічний ефект. Результат показав, що $E_{abc} > 0$, що означає позитивний чистий прибуток або економічну вигоду після реалізації проєкту. Це означає, що інтеграція розробки позитивно вплине на проєкт або підприємство, підвищивши його фінансову життєздатність і потенційну прибутковість. Відносна (річна) ефективність виявилася більшою за мінімально необхідну норму прибутку (τ_{min}), що вказує на потенційний інтерес інвестора. Термін окупності інвестицій у науковий

проєкт був розрахований у 0,8 року, що свідчить про відносно швидке повернення інвестицій і робить його перспективним фінансовим підприємством.

Таким чином, економічна оцінка підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO) демонструє її значний потенціал для комерційного успіху та позитивних фінансових результатів. Результати підтверджують економічну життєздатність проєкту та його потенціал для підвищення фінансових показників, що робить покращену розробку перспективним заходом для потенційних інвесторів та зацікавлених сторін.

ВИСНОВКИ

У магістерській кваліфікаційній роботі представлено підвищення захищеності копроративних даних шляхом оброблення потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO).

У першому розділі проаналізовано стан сучасних методів захисту інформації, а також основні підходи до використання генетичних алгоритмів та роїв часток. Досліджуються переваги та недоліки існуючих рішень.

У другому розділі розглянуто генетичний алгоритм рою часток (PSO), продемонстровано його переваги та недоліки у при використанні у сфері кібербезпеки та приклади застосування, порівняння з іншими генетичними алгоритмами та обрано варіанти вдосконалення та поліпшення використання даного алгоритму.

Третій розділ присвячений розробці вдосконаленого генетичного алгоритму рою часток, який базується на інтеграції генетичного алгоритму та алгоритму рою часток з метою підвищення ефективності та точності процесу захисту корпоративнех серверів обробки даних. Також аналізується результат експериментів та тестувань розробленого методу на різних наборах даних і в різних сценаріях. Здійснюється порівняння з існуючими рішеннями з метою визначення переваг та обмежень розробленого методу. Підтверджено високу ефективність даного рішення в порівняні з аналогами.

У четвертому розділі під час прорахукрів вартості і окупності проекту продемонстровано, що рішення є досить цікавим дляінвесторів і має високу окупність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kou, G., & Pedrycz, W. (2017). Data Classification with Genetic Programming and Its Application to Software Vulnerability Detection. Information Sciences, 379, 147-160. (дата звернення 10.09.2023).

2. 1"Particle Swarm Optimization: A Comprehensive Review" by Subhananda Chakraborty, Provas Kumar Roy, and Swagatam Das (2017). (дата звернення 10.09.2023).

3. "Security in Cloud Computing: A Comprehensive Survey" by M. I. Sookhak et al. (2014). (дата звернення 10.09.2023).

4. "Genetic Algorithm-based Particle Swarm Optimization Approach for Secure Image Steganography" by Ashraf Darwish and Safwat Hamad (2018). (дата звернення 10.09.2023).

5. "A Novel Particle Swarm Optimization-Based Approach for Anomaly Detection in Cloud Computing" by Hemalatha K. et al. (2017).

6. "Genetic Algorithm and Its Application in Information Security" by Zhen-wei Gao et al. (2015). (дата звернення 10.09.2023).

7. "Data Security in Cloud Computing Using Genetic Algorithm" by N. M. Saravana Kumar et al. (2016). (дата звернення 12.09.2023).

8. "Security Issues and Solutions in Cloud Computing" by V. Sangeetha and R. Gayathri (2017). (дата звернення 12.09.2023).

9. "Enhanced Security Model for Cloud Computing Using Genetic Algorithm" by Sumit Joshi et al. (2015). (дата звернення 12.09.2023).

10. "Security and Privacy Issues in Cloud Computing: A Survey" by M. S. Aliyu et al. (2015). (дата звернення 19.09.2023).

11. "Genetic Algorithm-Based Security Approach for Cloud Computing" by Y. V. Shridhar and N. S. Kumar (2015). (дата звернення 19.09.2023).

12. "Particle Swarm Optimization-Based Approach for Enhancing Security in Cloud Computing" by Hemalatha K. et al. (2016). (дата звернення 19.09.2023).

13. "Genetic Algorithms and their Applications in Security Aspects of Cloud Computing" by Prateek Pandey et al. (2016). (дата звернення 28.09.2023).

14. "A Survey of Particle Swarm Optimization Applications in Information Security" by Qinghan Xiao et al. (2018). (дата звернення 28.09.2023).

15. "Cloud Computing Security Issues and Solutions: A Survey" by Chia-Mu Yu et al. (2016). (дата звернення 28.09.2023).

16. "A Survey on Security Issues in Service Delivery Models of Cloud Computing" by Renuka Devi N et al. (2017). (дата звернення 08.10.2023).

17. "Security in Cloud Computing: A Comprehensive Survey" by M. I. Sookhak et al. (2014). (дата звернення 10.10.2023).

18. "Particle Swarm Optimization for Network Security: A Survey" by Hafeez Anwar et al. (2015). (дата звернення 10.10.2023).

19. 18. "An Improved Genetic Algorithm for Data Security in Cloud Computing" by S. B. Goyal et al. (2017). (дата звернення 12.10.2023).

20. "Genetic Algorithms: Concepts, Design Challenges, Solutions, and Future Perspectives" by Christian S. Jensen et al. (2019). (дата звернення 18.10.2023).

21. "Security Issues in Cloud Computing and Countermeasures: A Survey" by Priyanka Choudhary et al. (2015). (дата звернення 18.10.2023).

22. .Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с. (дата звернення 30.10.2023).

23. Tasgetiren, M. F., Sevkli, M., Liang, Y.-C., & Gencyilmaz, G. Particle swarm optimization algorithm for single-machine total weighted tardiness problem – 2. - 1412–1419. – 2016 . (дата звернення 05.11.2023).

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління
інформаційною
безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
"20" _____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

Підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO)

08-72.МКР.000.00.000ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.н., доц., доцент каф. МБІС Сачанюк-Кавецька Н. В.
_____

Вінниця – 2023 р.

1. Найменування та область застосування

Підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO). Область застосування: захист корпоративних даних на серверах.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №203 від 14. 09. 2022 р.

3. Мета та призначення розробки

3.1 Мета розробки: . Розробка та дослідження вдосконаленого генетичного алгоритму рою часток (PSO) з метою підвищення ефективності захисту інформації. А також – розробка рішення щодо підвищення єфективоності використання генетичного алгоритму рою часток шляхом вдосколналення його використання у сфері кібербезпеки.

3.2 Призначення: розроблене рішення призначене для захисту даних на корпоративних серверах.

4. Джерела розробки

4.1. Комплексні системи захисту інформації: [навчальний посібник] Ю. Є. Яремчук, П. В. Павловський, В. С. Катаєв, В. В. Сінюгін. Вінниця : ВНТУ, 2018. 118 с.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Harrag Fouzi, Djahli Mohamed Khalil. Arabic fake news detection: A fact checking based deep learning approach. *Transactions on Asian and Low-Resource Language Information Processing,* 2022, 21.4: 1-34. URL: https://dl.acm.org/doi/abs/10.1145/3501401 (дата звернення 02.11.2023).

4.4. Walker M. Python Data Cleaning Cookbook: Modern techniques and Python tools to detect and remove dirty data and extract key insights. Packt Publishing Ltd, 2020. URL:https://www.google.com/books?hl=uk&lr=&id=GMwOEAAAQBAJ&oi=fnd&pg=PP1&dq=clean+text+by+removing+HTML,+Python+characters&ots=_GKhTG-_LW&sig=PIKWgAyQkpEhFuS1W-fI2_xGRkA (дата звернення 02.11.2023).

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Повинен забеспечувани цілісність даних;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Intel Core i7-7700K 4.2GHz і продуктивніше;

– оперативна пам’ять – не менше 4096 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів.

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	20.09.2023	25.09.2023
2	Аналіз предметної області обраної теми	26.09.2023	30.10.2023
3	Апробація отриманих результатів	31.10.2023	02.10.2023
4	Розробка алгоритму роботи	03.10.2023	17.10.2023
5	Написання магістерської роботи на основі розробленої теми	18.10.2023	10.11.2023
6	Розробка економічної частини	11.11.2023	23.11.2023
7	Передзахист магістерської кваліфікаційної роботи	24.11.2023	25.11.2023
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	26.11.2023	30.11.2023
9	Захист магістерської кваліфікаційної роботи	15.12.2023	15.12.2023

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Ощепков В.С.

Додаток Б Узагальнений код оптимізації параметрів системи виявлення атак

```
import random
import numpy as np
import logging

# Налаштування логування
logging.basicConfig(filename='AoS_detection.log', level=logging.INFO, format='%(asctime)s - %(message)s')

# Створення системи виявлення атак з налаштовуваними параметрами
class AoSDetectionSystem:
    def __init__(self, detection_threshold, detection_duration, sensitivity, weight_rules):
        self.detection_threshold = detection_threshold
        self.detection_duration = detection_duration
        self.sensitivity = sensitivity
        self.weight_rules = weight_rules

    def is_AoS_attack(self, traffic_data):
        traffic_rate = calculate_traffic_rate(traffic_data)

        if traffic_rate > self.detection_threshold * self.sensitivity:
            if self.detection_duration > 0:
                self.detection_duration -= 1
                return True
            else:
                return False
        else:
            self.detection_duration = 0
            return False

    def set_parameters(self, parameters):
        self.detection_threshold, self.detection_duration, self.sensitivity, self.weight_rules = parameters

def calculate_traffic_rate(traffic_data):
    return sum(traffic_data) / len(traffic_data)
```

```
        # Реалізація PSO для оптимізації параметрів системи
        def pso_optimization(detection_system, num_particles, num_iterations):
            num_dimensions = 4
            max_velocity = 0.1
            inertia_weight = 0.5
            cognitive_weight = 2.0
            social_weight = 2.0

            num_particles = num_particles
            num_iterations = num_iterations

            particles = []
            best_global_position = None
            best_global_score = float('inf')

            for _ in range(num_particles):
                parameters = [random.uniform(0, 1) for _ in range(num_dimensions)]
                particle = {'position': parameters, 'velocity': [0] * num_dimensions,
'personal_best': None, 'score': float('inf')}
                particles.append(particle)

            for iteration in range(num_iterations):
                for particle in particles:
                    parameters = particle['position']
                    detection_system.set_parameters(parameters)
                    score = evaluate_detection_system(detection_system)

                    if score < particle['score']:
                        particle['personal_best'] = parameters
                        particle['score'] = score

                    if score < best_global_score:
                        best_global_position = parameters
                        best_global_score = score

                for particle in particles:
                    velocity = [particle['velocity'][i] +
                                cognitive_weight * random.uniform(0, 1) *
(particle['personal_best'][i] - particle['position'][i]) +
```

```
                    social_weight * random.uniform(0, 1) * (best_global_position[i] - particle['position'][i])
                    for i in range(num_dimensions)]

            for i in range(num_dimensions):
                if velocity[i] > max_velocity:
                    velocity[i] = max_velocity
                elif velocity[i] < -max_velocity:
                    velocity[i] = -max_velocity

            particle['velocity'] = velocity
            particle['position'] = [particle['position'][i] + velocity[i] for i in range(num_dimensions)]

        return best_global_position

    def evaluate_detection_system(detection_system):
        # Оцінюємо ефективність системи (можна використовувати реальну функцію оцінки)
        return random.uniform(0, 1)

    # Функція для відображення результатів оптимізації та постійної настройки
    def display_results(initial_parameters, optimized_parameters, new_parameters):
        print("Початкові параметри системи виявлення атак:")
        print("Поріг виявлення трафіку:", initial_parameters[0])
        print("Тривалість виявлення:", initial_parameters[1])
        print("Чутливість аналізу:", initial_parameters[2])
        print("Вагові коефіцієнти правил:", initial_parameters[3])

        print("\nОптимізовані параметри системи виявлення атак:")
        print("Поріг виявлення трафіку:", optimized_parameters[0])
        print("Тривалість виявлення:", optimized_parameters[1])
        print("Чутливість аналізу:", optimized_parameters[2])
        print("Вагові коефіцієнти правил:", optimized_parameters[3])

        print("\nНові параметри системи виявлення атак (після постійної настройки):")
        print("Поріг виявлення трафіку:", new_parameters[0])
```

```
        print("Тривалість виявлення:", new_parameters[1])
        print("Чутливість аналізу:", new_parameters[2])
        print("Вагові коефіцієнти правил:", new_parameters[3])

    if __name__ == "__main__":
        # Налаштування логера
        logger = logging.getLogger('AoS_detection')
        logger.setLevel(logging.INFO)

        # Створення обробника для запису логів в файл
        file_handler = logging.FileHandler('AoS_detection.log')
        formatter = logging.Formatter('%(asctime)s - %(message)s')
        file_handler.setFormatter(formatter)

        # Додавання обробника до логера
        logger.addHandler(file_handler)

        initial_parameters = [10000, 5, 1.0, [0.4, 0.3, 0.2, 0.1]]
        detection_system = AoSDetectionSystem(*initial_parameters)

        optimized_parameters = pso_optimization(detection_system, num_particles=20, num_iterations=50)

        detection_system.set_parameters(optimized_parameters)

       # Логування нових параметрів

        logger.info("Нові параметри системи виявлення атак (після постійної настройки): %s", new_parameters)


        # Виведення результатів

        display_results(initial_parameters, optimized_parameters, new_parameters)
```

Додаток Б – визначення функції вартості.

```
class AoSDetectionSystem:
```

```
        def __init__(self, detection_threshold, detection_duration):
            self.detection_threshold = detection_threshold  # Поріг виявлення трафіку
            self.detection_duration = detection_duration  # Тривалість виявлення

        def is_AoS_attack(self, traffic_data):
            # Аналіз трафіку і виявлення атаки
            traffic_rate = calculate_traffic_rate(traffic_data)  # Розрахунок швидкості трафіку

            if traffic_rate > self.detection_threshold:
                # Трафік перевищує поріг
                if self.detection_duration > 0:
                    # Провірка тривалості виявлення
                    # Якщо тривалість атаки менше або дорівнює визначеній тривалості, то вважаємо її атакою
                    self.detection_duration -= 1
                    return True
                else:
                    # Сплинуло встановлене часове вікно, атаку вже не вважаємо атакою
                    return False
            else:
                # Трафік не перевищує поріг, атака не виявлена
                self.detection_duration = 0  # Скидаємо тривалість, якщо трафік не перевищує поріг
                return False

    def calculate_traffic_rate(traffic_data):
        # Приклад розрахунку швидкості трафіку, може бути адаптованим до реальних умов
        return sum(traffic_data) / len(traffic_data)

    # Використання системи виявлення атак
    AoS_detection = AoSDetectionSystem(detection_threshold=10000, detection_duration=5)

    # Перевірка трафіку
    traffic_data = [9500, 11000, 12000, 8000, 7500]
    for i, traffic_rate in enumerate(traffic_data):
```

```
    if AoS_detection.is_AoS_attack(traffic_rate):
        print(f" атака виявлена на часі {i}!")
    else:
        print(f"Трафік нормальний на часі {i}.")
```

Додаток В Код визначення функції вартості.

```
class AoSDetectionSystem:
    def __init__(self, detection_threshold, detection_duration):
        self.detection_threshold = detection_threshold  # Поріг виявлення трафіку
        self.detection_duration = detection_duration  # Тривалість виявлення

    def is_AoS_attack(self, traffic_data):
        # Аналіз трафіку і виявлення атаки
        traffic_rate = calculate_traffic_rate(traffic_data)  # Розрахунок швидкості трафіку

        if traffic_rate > self.detection_threshold:
            # Трафік перевищує поріг
            if self.detection_duration > 0:
                # Провірка тривалості виявлення
                # Якщо тривалість атаки менше або дорівнює визначеній тривалості, то вважаємо її атакою
                self.detection_duration -= 1
                return True
            else:
```

```
                # Сплинуло встановлене часове вікно, атаку вже не вважаємо атакою
                return False
            else:
                # Трафік не перевищує поріг, атака не виявлена
                self.detection_duration = 0  # Скидаємо тривалість, якщо трафік не перевищує поріг
                return False

    def calculate_traffic_rate(traffic_data):
        # Приклад розрахунку швидкості трафіку, може бути адаптованим до реальних умов
        return sum(traffic_data) / len(traffic_data)

    # Використання системи виявлення атак
    AoS_detection = AoSDetectionSystem(detection_threshold=10000, detection_duration=5)

    # Перевірка трафіку
    traffic_data = [9500, 11000, 12000, 8000, 7500]
    for i, traffic_rate in enumerate(traffic_data):
        if AoS_detection.is_AoS_attack(traffic_rate):
            print(f"атака виявлена на часі {i}!")
        else:
            print(f"Трафік нормальний на часі {i}.")
```

Додаток Г Приклад програмної реалізації адаптивного налаштування коефіцієнтів

```
import random
# Початковий коефіцієнт інерції
initial_IW = 0.5
# Коефіцієнт адаптивності для зміни IW
adaptive_coefficient = 0.1
# Максимальний та мінімальний діапазон для IW
max_IW = 1.0
min_IW = 0.1
# Параметри PSO
num_particles = 50
max_iterations = 100
c1 = 2.0  # Коефіцієнт кращості частки
c2 = 2.0  # Коефіцієнт кращості рою
# Функція пристосованості (приклад)
def fitness_function(position):
    # Ваша функція пристосованості, яка оцінює результати виявлення атак
    # Якщо більше атак виявлено, функція поверне менше значення
    # Якщо менше атак виявлено, функція поверне більше значення
    return random.uniform(0, 1)
# Ініціалізація PSO
particles = []
best_global_position = None
best_global_fitness = float('inf')
```

```
        current_IW = initial_IW
        for _ in range(num_particles):
            particle = {
                "position": [random.uniform(0, 1) for _ in range(5)], # Припустимо, що є 5 параметрів для налаштування
                "velocity": [0] * 5,
                "best_position": None,
                "best_fitness": float('inf')
            }
            particles.append(particle)
        # Основний цикл PSO
        for iteration in range(max_iterations):
            for i, particle in enumerate(particles):
                fitness = fitness_function(particle["position"])
                if fitness < particle["best_fitness"]:
                    particle["best_fitness"] = fitness
                    particle["best_position"] = particle["position"]
                if fitness < best_global_fitness:
                    best_global_fitness = fitness
                    best_global_position = particle["position"]
            # Оновлення IW
            current_IW = max(min_IW, min(max_IW, current_IW + adaptive_coefficient * (best_global_fitness - fitness)))
            for i, particle in enumerate(particles):
                for j in range(len(particle["position"])):
                    # Оновлення швидкості та позиції частки з адаптивним IW
```

```
            particle["velocity"][j] = current_IW * particle["velocity"][j] + c1 * random.random() * (particle["best_position"][j] - particle["position"][j]) + c2 * random.random() * (best_global_position[j] - particle["position"][j])
            particle["position"][j] += particle["velocity"][j]
        print(f"Iteration {iteration + 1}: Best Fitness = {best_global_fitness}")
    print(f"Best Position: {best_global_position}")
    print(f"Best Fitness: {best_global_fitness}")
```

Додаток Д Приклад коду для оптимізація пакраметрів для вирішення задачі атак.

```
import random

# Припустимо, що ми маємо систему виявлення атак з параметрами, які можна налаштовувати.
class AoSDetectionSystem:
    def __init__(self, detection_threshold, detection_duration, sensitivity, weight_rules):
        self.detection_threshold = detection_threshold
        self.detection_duration = detection_duration
        self.sensitivity = sensitivity
        self.weight_rules = weight_rules

    def is_AoS_attack(self, traffic_data):
```

```
            # Спробуємо імітувати процес виявлення атаки з використанням параметрів
            # traffic_data – дані про трафік для аналізу

            traffic_rate = calculate_traffic_rate(traffic_data)

            if traffic_rate > self.detection_threshold * self.sensitivity:
                if self.detection_duration > 0:
                    self.detection_duration -= 1
                    return True
                else:
                    return False
            else:
                self.detection_duration = 0
                return False

        def set_parameters(self, parameters):
            # Встановлюємо параметри системи на основі значень PSO
            self.detection_threshold, self.detection_duration, self.sensitivity, self.weight_rules = parameters

    def calculate_traffic_rate(traffic_data):
        return sum(traffic_data) / len(traffic_data)

    # Реалізація PSO
    def pso_optimization(detection_system, num_particles, num_iterations):
        # Параметри PSO
```

```
num_dimensions = 4  # Кількість налаштовуваних параметрів
max_velocity = 0.1  # Максимальна швидкість зміни параметрів
inertia_weight = 0.5
cognitive_weight = 2.0
social_weight = 2.0

# Ініціалізація частинок
particles = []
best_global_position = None
best_global_score = float('inf')

for _ in range(num_particles):
    parameters = [random.uniform(0, 1) for _ in range(num_dimensions)]
    particle = {'position': parameters, 'velocity': [0] * num_dimensions, 'personal_best': None, 'score': float('inf')}
    particles.append(particle)

for iteration in range(num_iterations):
    for particle in particles:
        parameters = particle['position']
        detection_system.set_parameters(parameters)
        # Оцінюємо ефективність системи виявлення атак
        score = evaluate_detection_system(detection_system)

        if score < particle['score']:
            particle['personal_best'] = parameters
            particle['score'] = score
```

```
            if score < best_global_score:
                best_global_position = parameters
                best_global_score = score

        for particle in particles:
            velocity = [particle['velocity'][i] +
                        cognitive_weight * random.uniform(0, 1) *
(particle['personal_best'][i] - particle['position'][i]) +
                        social_weight * random.uniform(0, 1) *
(best_global_position[i] - particle['position'][i])
                        for i in range(num_dimensions)]

            for i in range(num_dimensions):
                if velocity[i] > max_velocity:
                    velocity[i] = max_velocity
                elif velocity[i] < -max_velocity:
                    velocity[i] = -max_velocity

            particle['velocity'] = velocity
            particle['position'] = [particle['position'][i] + velocity[i] for i in
range(num_dimensions)]

        return best_global_position

    def evaluate_detection_system(detection_system):
```

```
        # Можна розробити функцію, яка оцінює ефективність системи виявлення атак
        # В даному прикладі просто повертаємо випадкове число для ілюстрації
        return random.uniform(0, 1)

    if __name__ == "__main__":
        initial_parameters = [10000, 5, 1.0, [0.4, 0.3, 0.2, 0.1]]  # Початкові значення параметрів
        detection_system = AoSDetectionSystem(*initial_parameters)

        optimized_parameters = pso_optimization(detection_system, num_particles=20, num_iterations=50)

        detection_system.set_parameters(optimized_parameters)
        print("Оптимізовані параметри системи виявлення атак:")
        print("Поріг виявлення трафіку:", detection_system.detection_threshold)
        print("Тривалість виявлення:", detection_system.detection_duration)
        print("Чутливість аналізу:", detection_system.sensitivity)
        print("Вагові коефіцієнти правил:", detection_system.weight_rules)
```

Додаток Е Приклад програмного коду для реалізації прорахунку вагових коеіціентів

```
        # Визначення функції оцінки результатів оптимізації
        def evaluate_optimization_results(optimized_parameters, original_parameters):
```

```
    # Ви можете використовувати реальну функцію оцінки, але в цьому прикладі просто порівнюємо параметри
    original_score = evaluate_detection_system(AoSDetectionSystem(*original_parameters))
    optimized_score = evaluate_detection_system(AoSDetectionSystem(*optimized_parameters))

    if optimized_score < original_score:
        return "Оптимізація була успішною. Параметри покращили ефективність системи."
    else:
        return "Оптимізація не призвела до покращення. Параметри залишились незмінними або погіршили ефективність системи."

if __name__ == "__main__":
    initial_parameters = [10000, 5, 1.0, [0.4, 0.3, 0.2, 0.1]]  # Початкові значення параметрів
    detection_system = AoSDetectionSystem(*initial_parameters)

    optimized_parameters = pso_optimization(detection_system, num_particles=20, num_iterations=50)

    detection_system.set_parameters(optimized_parameters)

    # Оцінка результатів оптимізації
    evaluation_result = evaluate_optimization_results(optimized_parameters, initial_parameters)
```

```
print(evaluation_result)
```

Додаток Е Код системи виявлення кібер-атак з налаштовуваними параметрами.

```
import random
import numpy as np
import hashlib
import hmac

# Створення системи виявлення кібер-атак з налаштовуваними параметрами
class CyberAttackDetectionSystem:
    def __init__(self, detection_threshold, detection_duration, sensitivity, weight_rules):
        self.detection_threshold = detection_threshold
        self.detection_duration = detection_duration
        self.sensitivity = sensitivity
        self.weight_rules = weight_rules

    def is_cyber_attack(self, traffic_data):
        traffic_rate = calculate_traffic_rate(traffic_data)

        if traffic_rate > self.detection_threshold * self.sensitivity:
            if self.detection_duration > 0:
                self.detection_duration -= 1
                return True
            else:
                return False
        else:
            self.detection_duration = 0
            return False

    def set_parameters(self, parameters):
        self.detection_threshold, self.detection_duration, self.sensitivity, self.weight_rules = parameters

def calculate_traffic_rate(traffic_data):
```

```
        return sum(traffic_data) / len(traffic_data)

    # Реалізація PSO для оптимізації параметрів системи
    def pso_optimization(detection_system, num_particles, num_iterations):
        num_dimensions = 4
        max_velocity = 0.1
        inertia_weight = 0.5
        cognitive_weight = 2.0
        social_weight = 2.0

        num_particles = num_particles
        num_iterations = num_iterations

        particles = []
        best_global_position = None
        best_global_score = float('inf')

        for _ in range(num_particles):
            parameters = [random.uniform(0, 1) for _ in range(num_dimensions)]
            particle = {'position': parameters, 'velocity': [0] * num_dimensions,
'personal_best': None, 'score': float('inf')}
            particles.append(particle)

        for iteration in range(num_iterations):
            for particle in particles:
                parameters = particle['position']
                detection_system.set_parameters(parameters)
                score = evaluate_detection_system(detection_system)

                if score < particle['score']:
                    particle['personal_best'] = parameters
                    particle['score'] = score

                if score < best_global_score:
                    best_global_position = parameters
                    best_global_score = score

            for particle in particles:
                velocity = [particle['velocity'][i] +
```

```
                    cognitive_weight * random.uniform(0, 1) * (particle['personal_best'][i] - particle['position'][i]) +
                    social_weight * random.uniform(0, 1) * (best_global_position[i] - particle['position'][i])
                    for i in range(num_dimensions)]

            for i in range(num_dimensions):
                if velocity[i] > max_velocity:
                    velocity[i] = max_velocity
                elif velocity[i] < -max_velocity:
                    velocity[i] = -max_velocity

            particle['velocity'] = velocity
            particle['position'] = [particle['position'][i] + velocity[i] for i in range(num_dimensions)]

        return best_global_position

    def evaluate_detection_system(detection_system):
        # Оцінюємо ефективність системи (можна використовувати реальну функцію оцінки)
        return random.uniform(0, 1)

    # Функція для генерації ключів шифрування та підпису
    def generate_encryption_keys(num_dimensions):
        keys = [random.randint(0, 255) for _ in range(num_dimensions)]
        return keys

    # Функція для шифрування та підпису повідомлення
    def encrypt_and_sign_message(message, encryption_key, signing_key):
        # Шифрування повідомлення
        encrypted_message = [ord(char) ^ encryption_key for char in message]

        # Підпис повідомлення
        signature = hmac.new(signing_key, bytes(encrypted_message), hashlib.sha256).digest()

        return encrypted_message, signature

    # Система реагування на кібер-атаки
```

```
class CyberAttackReactionSystem:
    def __init__(self, detection_system, optimized_parameters, num_key_dimensions):
        self.detection_system = detection_system
        self.optimized_parameters = optimized_parameters
        self.num_key_dimensions = num_key_dimensions

    def react_to_cyber_attack(self, detected_attack_ips):
        for ip in detected_attack_ips:
            print(f"Виявлено кібер-атаку від IP: {ip}")

        # Генерація ключів для криптосистеми
        encryption_key = generate_encryption_keys(self.num_key_dimensions)
        signing_key = generate_encryption_keys(self.num_key_dimensions)

        # Підготовка звіту
        report = "Звіт про стан системи після кібер-атаки: система була атакована, але успішно відповіла."

        # Шифрування та підпис звіту
        encrypted_report, signature = encrypt_and_sign_message(report, encryption_key, signing_key)

        print("Зашифрований звіт:", encrypted_report)
        print("Підпис:", signature)

if __name__ == "__main__":
    initial_parameters = [10000, 5, 1.0, [0.4, 0.3, 0.2, 0.1]]
    detection_system = CyberAttackDetectionSystem(*initial_parameters)

    num_key_dimensions = 16  # Кількість байтів для ключів шифрування та підпису

    optimized_parameters = pso_optimization(detection_system, num_particles=20, num_iterations=50)

    detection_system.set_parameters(optimized_parameters)

    # Симулювати виявлення кібер-атаки (просто для прикладу)
    detected_attack_ips = set()
```

```
        for _ in range(5):
            traffic_data = [random.randint(8000, 15000) for _ in range(10)]  # Велика кількість трафіку
            if detection_system.is_cyber_attack(traffic_data):
                detected_attack_ips.add(f"192.168.1.{random.randint(1, 254)}")  # Додати підозрілий IP

        reaction_system = CyberAttackReactionSystem(detection_system, optimized_parameters, num_key_dimensions)
        reaction_system.react_to_cyber_attack(detected_attack_ips)
```

Додаток Є Ілюстративний матеріал

Магістерська кваліфікаційна робота

Підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO)

Виконав :ст. групи 1КІТС-22m Ощепков В.С.

Керівник : к.т.н., доц., доцент каф. МБІС

Сачанюк-Кавецька Н. В.

ВСТУП

Актуальність

- Захист корпоративних даних на серверах є надзвичайно важливим завданням для будь-якої компанії. Цифрові дані є цінним активом, який може бути використаний для конкурентної переваги, отримання прибутку або навіть для завдання шкоди компанії.
- У сучасному світі, де кіберзлочинність є серйозною загрозою, захист корпоративних даних є невід'ємною частиною бізнесу. Компанії повинні впроваджувати комплексні заходи безпеки для захисту своїх даних від несанкціонованого доступу, використання, зміни або знищення.

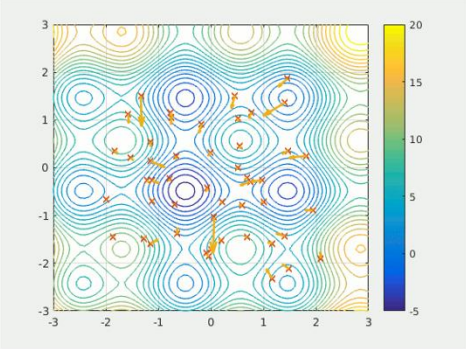
- **Новизною** є вдосконалення алгоритму рою часток (PSO) шляхом реалізації налаштування коефіцієнтів для модернізації точності досліджень у реальному часі та створення системи забезпечення захисту мережі від атак зловмисників.
- **Практична цінність:** розроблено програмний продукт (модуль), який реалізує вдосконалений алгоритм рою часток (PSO), що використовується в системах забезпечення захисту мережі від атак зловмисників.

Аналіз використання генетичного алгоритму рою часток для захисту даних корпоративних серверів

- **Генетичний алгоритм рою часток** - це комбінація генетичного алгоритму (GA) та алгоритму рою часток (PSO). Він використовує переваги обох алгоритмів, щоб забезпечити ефективніше розв'язання оптимізаційних задач.
- **Переваги використання** :
- Ефективність: алгоритм може ефективно вирішувати складні задачі захисту даних, такі як оптимізація політики безпеки або конфігурація засобів захисту від кібератак.
- Уникнення локальних мінімумів: алгоритм має здатність уникати локальних мінімумів, які можуть призвести до недооптимального рішення.
- Масштабованість: алгоритм може бути масштабований для вирішення задач великого розміру.
- **Недоліки використання** :
- Повільність: може бути повільним для вирішення задач малого розміру.
- Налаштування параметрів: може вимагати значної настройки параметрів для досягнення оптимальних результатів.

Вдосконалення алгоритму по кроках.

- На сам перед це гібридизації PSO з GA
- Далі використовуємо більш складу функцію відбору та фунцію мутації
- Виконуємо адаптивне налаштування коефіцієнтів для нових функцій
- Виконуємо математичне порівняння після адаптивного налаштвуання
- Визначення параметрів
- Визначення функції вартості
- Вагові коефіцієнти правил
- Навчання та тестування алгоритму



Використання вдосконаленого алгоритму та його переваги.

- 1. Алгоритм розроблено як модуль , тому його можна використовувати в готових рішеннях.
- 2. Алгоритм потребує бази даних тестових випадків для навчання і забезпечення цілісності даних з самого початку користування, але також може навчатись в процесі роботи з різними типами загроз. Тому забезпечує захищеність навіть від нових типів загроз.
- 3. Для роботи з рішенням потрібно лише один раз налаштувати всі доступні параметри і в майбутньому, рішення не потребує втручання без нагальної потреби.
- 4. Рішення є енерго-ефективним і сильно не навантажує систему.

Вибір засобів програмування

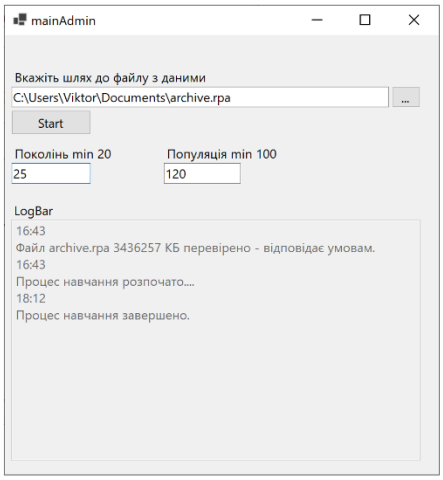
- Виходячи із поставлених завдань роботи, для реалізації програмного рішення обрано наступні програмні засоби:
- • мова об'єктно-орієнтованого програмування C#;
- • середовище програмування Visual Studio 2019;
- • інтерфейс програмування додатків Windows Forms;
- • програмна технологія для створення додатків.NET Framework.

Інтерфейс програмного додатку для адміністратору

Модуль який включає в себе вдосконалений алгоритм було приєднано до мінімалістичного інтерфейсу адміністратору.

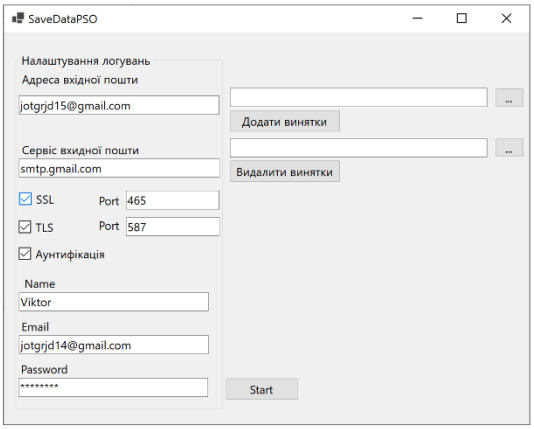
В даному меню можна вибрати тестову вибірку з даними про атаки , для навчання алгоритму.

Далі можна запустити меню користувача.



Інтерфейс програмного додатку для користувача

- Також мінімальний інтерфейс користувача.
- Є можливість налаштувати надсилання логувань на пошту, може бути як метод повідомлення про небезпеку адміністратора.
- Також можна добавити і видалити виключення , якщо такі необхідні.

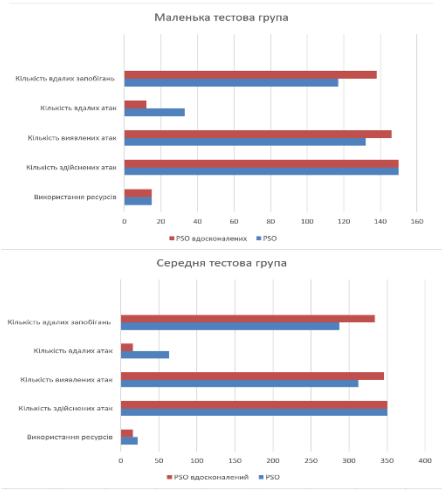


Тестування вдосконаленого алгоритму

Було порівняно звичайний алгоритм рою часток з вдосконаленим.

Для тестування було використано таке ПЗ:

- **Metasploit -** це платформа для автоматизації кібер атак. Вона містить набори інструментів для проведення різних типів атак, таких як атаки на вразливості програмного забезпечення, атаки на мережі та атаки на кінцеві точки.
- **Nessus -** це інструмент для сканування безпеки, який може виявляти вразливості в системах та мережах. Використовується для тестування кібер атак, таких як атаки на вразливості програмного забезпечення та мережеві атаки.
- **MSTIC** - це команда реагування на інциденти Microsoft, яка пропонує безкоштовний інструмент для імітації кібер атак під назвою MSTIC Hunt. MSTIC Hunt дозволяє організаціям створювати власні сценарії кібер атак і тестувати свою готовність до реагування на них.
- **MITRE Engenuity ATT&CK** - це платформа для імітації кібер атак, яка дозволяє організаціям тестувати свою готовність до реагування на атаки, які використовують реальні тактики, техніки та процедури (TTP) кібер злочинців.
- **Cyber Range -** це платформа для імітації кібер атак, яка дозволяє організаціям створювати та запускати власні сценарії кібер атак. Cyber Range також пропонує широкий спектр навчальних матеріалів та ресурсів, які можуть допомогти організаціям покращити свою готовність до реагування на кібер атаки.



Економічна доцільність розробки

Економічна оцінка підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO) демонструє її значний потенціал для комерційного успіху та позитивних фінансових результатів.

Результати прорахунків в економічному розділі підтверджують економічну життєздатність проєкту та його потенціал для підвищення фінансових показників, що робить покращену розробку перспективним заходом для потенційних інвесторів та зацікавлених сторін.

Висновки

- Захист даних є життєво важливим для будь-якої компанії. Корпоративні дані можуть містити конфіденційну інформацію про клієнтів, співробітників та бізнес-операції. Якщо ці дані будуть скомпрометовані, це може призвести до серйозних фінансових збитків, пошкодження репутації та навіть порушення закону.
- Кіберзлочинність є постійною загрозою. Кіберзлочинці постійно розробляють нові методи атаки на сервери. Це означає, що компанії повинні постійно вдосконалювати свої заходи безпеки, щоб залишатися в безпеці.
- Вдосконалений алгоритм рою часток , може забезпечувати постійний захист і цілісність даних. Метод є надійнішим за аналоги за рахунок того що може навчатись в реальному часі і долати нові загрози , до того як про тих стане відомо фахівцям.

Додаток Е. ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Підвищення захищеності корпоративних даних обробленням потоків даних серверів на основі вдосконаленого генетичного алгоритму рою часток (PSO)

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 99 % Схожість 1 %

Аналіз звіту подібності (відмітити потрібне):

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Ощепков В.С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Сачанюк-Кавецька
Н.В. (підпис) (прізвище, ініціали)