Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» травня 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Моделі виявлення компрометації аутентифікаційних даних»

Виконавець: студент ІІ курсу, групи КБм-22
_____ **Андрій ВІХРОВ**
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Тетяна БАБЕНКО	
Нормоконтроль	Лариса МИРУТЕНКО	

Київ 2024

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» листопада 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача КБм-22 (група) Віхрова Андрія Олексійовича (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Моделі виявлення компрометації аутентифікаційних даних

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 15.11.2023 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес виявлення компрометації аутентифікаційних даних.
Предмет досліджень	Механізми виявлення компрометації аутентифікаційних даних.
Мета	Розробка моделі виявлення компрометації аутентифікаційних даних.
Вихідні дані для проведення роботи	Методи виялення компрометації аутентифікаційних даних.

ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	Покращення існуючих підходів до виявлення компрометаціїї аутентифікаційних даних.
Практична цінність	Синтезована модель виявлення компроментації аутентифікаційних даних

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець)**
Уточнення постановки задачі	17.11.2023 – 29.01.2024
Аналіз літературних джерел	30.01.2024 – 12.02.2024
Аналіз методологій виявлення компрометації аутентифікації та аномалій	13.02.2024 – 21.02.2024
Розгляд теоретичних аспектів оцінки захищеності аутентифікації	22.02.2024 – 26.02.2024
Аналіз основних вимог до моделі виявлення компроментації аутентифікації	27.02.2024 – 04.03.2024
Вибір методології та алгоритмів для розробки моделі	05.03.2024 – 10.03.2024
Визначення формату даних для навчання моделі	11.03.2024 – 17.03.2024
Розробка моделі виявлення компроментації аутентифікаційних даних	18.03.2024 – 19.03.2024
Підготовка даних для моделювання	20.03.2024 – 17.04.2024
Моделювання та аналіз адекватності	18.04.2024 – 25.04.2024
Оформлення пояснювальної записки згідно методичних рекомендацій	26.04.2024 – 12.05.2024
Подача пакету документів на розгляд ЕК	13.05.2024 – 18.05.2024

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект	Зниження витрат на можливість виявлення компрометації аутентифікаційних даних
Соціальний ефект	Покращення забезпечення захисту інформаційних систем на підприємствах та інших організаціях.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Тетяна Бабенко
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв
до виконання _____ Андрій ВІХРОВ
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 17.11.2023 р.
Термін подання кваліфікаційної роботи до ЕК 17.05.2024 р.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Модель виявлення компрометації аутентифікації»: 100 сторінок, 2 додаток, 15 рисунків та 50 літературних джерел.

Метою роботи є розробка моделі виявлення компрометації аутентифікації.

Для досягнення зазначеної мети поставлено наступні завдання:

- Аналіз підходів до виявлення компрометації аутентифікації.
- Аналіз вразливостей та методів компрометації аутентифікації.
- Створити структурований опис моделі виявлення компрометації аутентифікації
- Розробка моделі виявлення компрометації аутентифікації.
- Аналіз адекватності запропонованого рішення.

Об’єктом дослідження є процес виявлення компрометації аутентифікаційних даних.

Предметом дослідження засоби та механізми виявлення компрометації аутентифікаційних даних.

Методи дослідження: аналіз відкритих джерел, порівняння методів виявлення компрометацій та аномалій аутентифікації, аналіз методів виявлення компрометації аутентифікації, проектування штучної нейронної мережі.

Актуальність роботи полягає в тому, що оцінка захищеності інформаційних систем дозволяє вчасно виявляти «слабкі місця» системи під час проектування чи використання системи і полегшує процес покращення рівню захищеності. Всі існуючі підходи мають певні вади та недоліки і на меті роботи стоїть створення моделі, яка би поєднувала найкращі риси існуючих підходів, мінімізуючи вплив недоліків систем.

Практичною цінністю отриманих результатів є синтезована модель виявлення компрометацій аутентифікаційних даних. Результати досліджень, виконаних у

кваліфікаційній роботі, можуть бути використані компаніями, які займаються кібербепекою, для підвищення ефективності моніторингу безпеки та проведення аудитів безпеки систем. У подальшому програмну реалізацію можна вдосконалити, інтегрувавши додаткові модулі для покращення аналізу і точності виявлення загроз.

Наукова новизна роботи полягає в покращенні існуючих підходів до виявлення компрометації аутентифікаційних даних, а саме, був підвищений відсоток виявлення спроб компроментації, який складає більше ніж 97%.

Ключові слова: детекція компрометацій, аутентифікаційні дані, кібербезпека, система моніторингу, аналітика безпеки.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ДКД	–	Дані компрометації даних
МН	–	Машинне навчання
САД	–	Система автоматизованого детектування
АДК	–	Алгоритми детекції компрометації
БД	–	База даних
ІЛ	–	Ізольовані ліси
НМД	–	Навчання моделі даних
ОА	–	Оцінка адекватності
ПД	–	Підготовка даних
РКД	–	Реалізація кіберзахисту даних
АПЗ	–	Аналіз поведінки зловмисників
СВД	–	Система виявлення вторгнень

ЗМІСТ

ВСТУП

У сучасному світі, де інформація стає все більш цінною та доступною, питання безпеки аутентифікаційних даних набуває особливої ваги. Розвиток технологій та поширення інтернету збільшують кількість точок входу для потенційних загроз, що вимагає вдосконалення методів захисту персональних даних.

Аутентифікація користувачів є критичним елементом у захисті інформаційних систем. Ідентифікація особи, яка намагається отримати доступ до ресурсів, є першим та одним з найважливіших кроків у процесі забезпечення інформаційної безпеки. Проте, зі зростанням числа користувачів інтернет-сервісів і розширенням спектру онлайн-послуг, збільшується і кількість зловмисників, які намагаються скомпрометувати аутентифікаційні дані. Викрадення паролів, фішинг, застосування шкідливих програм — лише декілька прикладів загроз, які можуть призвести до втрати важливої інформації і фінансових ресурсів.

За даними дослідження, проведеного компанією Verizon у 2021 році, 81% порушень безпеки були пов'язані зі слабкими, викраденими або легко здогадливими паролями. Така статистика підкреслює необхідність розробки більш надійних методів виявлення та попередження компрометації аутентифікаційних даних [1].

Тема дослідження має на меті розробку моделі, яка б дозволила ефективно виявляти спроби несанкціонованого доступу, використовуючи сучасні досягнення у галузі штучного інтелекту та машинного навчання. Це дозволить не тільки мінімізувати ризики, пов'язані з викраденням даних, але й значно підвищити загальний рівень захисту інформаційних систем.

Питання захисту аутентифікаційних даних стає все більш актуальним у зв’язку з зростанням кількості цифрових ідентифікаторів та різноманітних форм онлайн-взаємодії. Компрометація аутентифікаційних даних не тільки призводить до втрати особистої інформації, але й може спричинити значні фінансові збитки та негативно вплинути на репутацію компаній. Така проблема потребує розробки комплексних та

ефективних рішень, які б могли адаптуватися до постійно змінювальних методів атак і сучасних вимог кібербезпеки.

Наразі існуючі рішення часто ґрунтуються на застарілих підходах та не враховують новітні технологічні можливості, такі як машинне навчання та штучний інтелект, що обмежує їх ефективність у реальних умовах. Це вимагає від науковців та розробників інноваційного підходу до виявлення і нейтралізації загроз, зокрема, з використанням глибинного аналізу даних та адаптивних алгоритмів навчання.

Основною метою даного дослідження є розробка моделі виявлення компрометації аутентифікаційних даних з використанням алгоритмів машинного навчання, які здатні аналізувати і класифікувати потенційні загрози в реальному часі.

РОЗДІЛ 1
АНАЛІЗ МОДЕЛЕЙ ТА МЕТОДІВ ВИЯВЛЕННЯ КОМПРОМЕТАЦІЇ АУТЕНТИФІКАЦІЙНИХ ДАНИХ

1.1 Аналіз поведінки користувача

1.1.1 Визначення характеристик поведінки

Аналіз поведінки користувача є фундаментальним елементом у системах виявлення вторгнень та компрометації аутентифікаційних даних. Визначення характеристик поведінки користувачів дозволяє системам безпеки ідентифікувати відхилення від звичайної активності, які можуть вказувати на несанкціоноване втручання або інші безпекові інциденти [2]. Загально прийняті характеристики поведінки включають, але не обмежуються, наступними параметрами:

1. Час входу в систему: Відхилення у звичних часових рамках доступу можуть свідчити про спроби доступу до системи з нехарактерних географічних локацій або в атиповий час.
2. Частота запитів: Надмірна активність, така як несподівано велика кількість запитів до бази даних за короткий проміжок часу, може вказувати на автоматизовані скрипти або програми, які намагаються викрасти дані.
3. Типи запитів: Незвичайні або непередбачені типи запитів можуть вказувати на спроби експлуатації вразливостей в системах.
4. Використання ресурсів: Несподіване збільшення використання ресурсів, таких як CPU або RAM, може бути індикатором шкідливих програм або процесів.

Для точного визначення цих характеристик важливо розробити модель, яка може адекватно відображати звичайну поведінку користувача в рамках конкретного інформаційного середовища. Збір даних для цієї моделі включає моніторинг та аналіз поведінкових патернів користувачів у довгостроковому періоді для створення надійної базової лінії поведінки [3].

Джерела для цього аналізу можуть включати логи серверів, записи аутентифікації, аудит активності користувачів та інші відповідні дані. Використання передових технологій збору та обробки даних, таких як big data та аналітичні платформи, є ключовими для ефективного збору та аналізу цієї інформації.

1.1.2 Збір даних про поведінку

Збір даних про поведінку користувачів є вирішальним етапом у процесі виявлення компрометації аутентифікаційних даних. Ефективний збір даних забезпечує те, що аналітичні системи мають достатньо інформації для ідентифікації потенційних загроз та надає базу для тренування машинного навчання та розробки алгоритмів виявлення аномалій [4].

Для збору даних можуть використовуватися різноманітні методи та інструменти, зокрема:

- Моніторинг мережевого трафіку: Використання інструментів моніторингу для відстеження всіх вхідних та вихідних з'єднань, що дозволяє виявити незвичайні або підозрілі мережеві активності.
- Системи виявлення вторгнень (IDS): Автоматизовані системи, які аналізують мережевий трафік на предмет відомих шаблонів атак або аномальних патернів поведінки.
- Логи аудиту: Збір даних з системних, застосункових та безпекових логів для аналізу дій користувачів і системних подій.
- Інструменти аналізу поведінки користувачів: Ці системи використовують алгоритми машинного навчання для аналізу поведінки користувачів та виявлення відхилень від норми.

Зібрані дані повинні бути акуратно збережені та захищені, оскільки вони можуть містити чутливу інформацію. Важливо забезпечити, щоб процеси обробки даних були відповідно до політик конфіденційності та безпеки даних [5].

1.1.3 Аналіз даних про поведінку

Після збору необхідних даних наступним кроком є їх аналіз. Цей етап має на меті виявлення потенційних індикаторів компрометації аутентифікаційних даних. Аналіз може включати:

- Статистичний аналіз: Використання статистичних методів для ідентифікації відхилень від базової поведінки, які можуть вказувати на неавторизований доступ або інші підозрілі дії.
- Поведінкове моделювання: Розробка моделей, які можуть передбачити типову поведінку користувача та виявляти аномалії, засновані на історичних даних та взаємодії з системою.
- Машинне навчання: Використання алгоритмів машинного навчання, таких як навчання з підкріпленням або нейронні мережі, для аналізу великих обсягів даних та виявлення складних патернів поведінки, які можуть не бути очевидними за допомогою традиційних методів.

Цей аналіз допомагає створити детальну картину поведінкових патернів, які можуть вказувати на ризики безпеки. Результати такого аналізу можуть використовуватися для розробки більш ефективних стратегій захисту аутентифікаційних даних і поліпшення загальної системи безпеки [6].

1.1.4 Реагування на виявлені компрометації

Ефективне реагування на виявлені компрометації є критично важливим. Цей процес включає не лише виявлення та ідентифікацію інцидентів, але й швидке вживання заходів для мінімізації шкоди. Стратегії реагування можуть включати:

1. Повідомлення відповідальним особам: Автоматичне сповіщення системних адміністраторів та/або відділу безпеки про потенційні інциденти.
2. Автоматизація блокування атак: Застосування правил, які автоматично блокують користувачів або транзакції, які здаються підозрілими, поки не буде проведено подальша перевірка.

3. Аудит та аналіз інцидентів: Детальне розслідування інцидентів для визначення причин компрометації та розробка заходів для запобігання подібних інцидентів у майбутньому.

4. Оновлення політик безпеки: Перегляд та оновлення політик та процедур безпеки на основі виявлених інцидентів для посилення захисту системи.

Використання інтегрованих підходів, що поєднують технології моніторингу, аналізу та швидкого реагування, може значно підвищити здатність організації захищати свої критичні дані від сучасних кіберзагроз [7].

Давайте поглибимо аналіз і додамо деталі щодо специфічних технологій та розглянемо приклад, який ілюструє ефективність різних методів аналізу поведінки користувачів. Для цього треба розглянути кілька спецефічних технологій для моніторінгу поведінки:

1. SIEM Системи (Security Information and Event Management): SIEM системи інтегрують збір, управління та аналіз логів з різних джерел у централізований інструмент. Вони використовуються для моніторингу та аналізу безпекових подій у реальному часі та забезпечення тривогами на основі налаштованих правил, які можуть виявити аномальні поведінкові патерни користувачів. Наприклад, SIEM система може виявити, коли користувач спробує виконати несанкціонований доступ до чутливих ресурсів поза звичайним робочим часом.

2. UBA Рішення (User and Entity Behavior Analytics): UBA використовує алгоритми машинного навчання для аналізу поведінки користувачів і виявлення відхилень від норми, які можуть вказувати на потенційні загрози або фрод. Ці системи вимірюють та збирають поведінкові дані протягом тривалого періоду для формування профілю "нормальної" поведінки кожного користувача та виявлення аномалій, що значно підвищує точність виявлення порушень.

Тепер ми можемо розглянути приклад використання аналізу поведінки для виявлення компроментації:

В одному з банків, використання UBA дозволило ідентифікувати зловмисника, який намагався здійснити масштабну крадіжку коштів через фальсифікацію транзакцій. Система зафіксувала незвичайні входи в систему поза робочим часом та

виконання аномально великої кількості транзакцій, що відразу спричинило запуск внутрішнього розслідування та відновлення контролю над компрометованими акаунтами [8].

Цей приклад підкреслює значення інтеграції продвинутих технологій в аналіз поведінки користувачів та демонструє, як вони можуть бути використані для ефективного виявлення та реагування на загрози безпеки. Використання цих технологій дозволяє організаціям не тільки проактивно ідентифікувати потенційні інциденти, але й забезпечити швидке реагування, зменшуючи тим самим можливі збитки.

Важливим аспектом ефективного реагування є не лише швидке втручання, але й наступний аналіз інциденту для поліпшення загальних процедур безпеки. Ось декілька додаткових стратегій:

- Пост-інцидентний аналіз: Важливо провести глибокий аналіз після кожного інциденту для визначення, як і чому безпека була порушена. Це допомагає виявити слабкі місця в системі безпеки та розробити заходи для їх усунення.
- Оновлення системи виявлення аномалій: На основі отриманих даних про інциденти, системи виявлення аномалій повинні бути оновлені для кращого прогнозування та виявлення подібних загроз у майбутньому.
- Тренінги та освітні програми: Регулярне навчання персоналу щодо новітніх загроз та методів їх виявлення може значно зменшити людський фактор у безпекових інцидентах.
- Співпраця з зовнішніми експертами: Використання зовнішніх ресурсів і експертизи для перевірки та оцінки власних безпекових механізмів дозволяє забезпечити об'єктивний погляд на систему безпеки та виявити потенційні слабкості, які можуть бути неочевидними для внутрішньої команди.

Ці покращені підходи до аналізу та реагування на аномалії у поведінці користувачів є ключовими для забезпечення високого рівня безпеки в сучасних інформаційних системах. Завдяки їм можливо не тільки оперативно реагувати на загрози, але й адаптувати системи до постійно змінюваного ландшафту кібербезпеки,

зменшуючи ризик втрати чутливої інформації та забезпечуючи надійний захист даних користувачів.

1.2 Виявлення аномалій

Виявлення аномалій є ключовим компонентом систем безпеки, який дозволяє ідентифікувати потенційні загрози та вторгнення, шляхом аналізу відхилень від нормальної поведінки. Цей процес допомагає організаціям реагувати на загрози в реальному часі та запобігати можливим зловмисним діям.

1.2.1 Визначення базового рівня

Базовий рівень (baseline) є фундаментальним елементом у процесі виявлення аномалій, оскільки він визначає "нормальну" поведінку системи або користувачів. Встановлення базового рівня вимагає збору і аналізу великої кількості даних про поведінку користувачів, системних процесів, мережевої активності та інших важливих метрик. За цими даними виявляються шаблони, які потім використовуються для визначення того, що вважається нормою [9].

Для визначення базового рівня використовуються різні техніки і технології збору даних:

1. Моніторинг мережі: Збір даних про мережевий трафік, включаючи частоту, обсяг і тип мережевих запитів.
2. Аудит системних логів: Записи системних подій можуть надавати інформацію про звичайну активність користувачів і програмних систем.
3. Моніторинг процесів: Спостереження за поведінкою процесів на хостах і серверах, щоб визначити типові шаблони використання ресурсів [10].

Встановлення базового рівня вимагає використання алгоритмів статистичного аналізу та машинного навчання, які можуть виявити і згрупувати шаблони поведінки:

1. Статистичний аналіз: Використання методів, як-от середні значення, медіани, стандартні відхилення для визначення "норми".

2. Кластеризація: Застосування алгоритмів кластеризації для групування схожих даних і встановлення загальноприйнятих поведінкових шаблонів.

3. Нейронні мережі: Застосування глибокого навчання для виявлення складних шаблонів поведінки, які не можуть бути легко ідентифіковані за допомогою традиційних статистичних методів [11].

Визначення базового рівня несе в собі декілька викликів, які потрібно адресувати для ефективного виявлення аномалій:

1. Динамічність поведінки: Поведінка користувачів та систем може змінюватися з часом, що вимагає постійного оновлення базових рівнів.

2. Велика кількість даних: Збір та обробка великих обсягів даних для визначення базового рівня вимагає значних обчислювальних ресурсів.

3. Виявлення складних аномалій: Деякі аномалії можуть бути тонкими та складно виявимими без застосування складних алгоритмів машинного навчання.

Ефективно встановлений базовий рівень дозволяє:

1. Швидке виявлення аномалій: Наявність чітко визначеного базового рівня дозволяє системам безпеки швидко ідентифікувати відхилення, що зменшує час реагування на потенційні загрози.

2. Мінімізація помилкових спрацьовувань: Точне визначення "норми" допомагає зменшити кількість помилкових тривог, забезпечуючи краще використання ресурсів безпеки [12].

3. Поліпшення загальної безпеки: Систематичний аналіз та оновлення базових рівнів забезпечують вдосконалення безпекових механізмів організації.

З визначеним базовим рівнем, системи безпеки можуть використовувати різні техніки для виявлення аномалій:

1. Порівняльний аналіз: Неперервне порівняння поточної активності з базовим рівнем для виявлення відхилень.

2. Профілювання поведінки: Автоматизоване створення профілів користувачів на основі їх поведінкових шаблонів для виявлення несанкціонованої діяльності.

3. Використання інтелектуальних алгоритмів: Застосування штучного інтелекту для динамічного аналізу поведінки та адаптації до нових загроз.

Завдяки цим підходам, системи виявлення аномалій можуть ефективно ідентифікувати підозрілу поведінку, значно зменшуючи ризик безпекових інцидентів [13].

Для оптимальної реалізації систем виявлення аномалій важливо не тільки розробити та налаштувати відповідні технологічні рішення, але й інтегрувати їх у загальну інфраструктуру кібербезпеки організації. Це включає:

1. Тісну інтеграцію з іншими системами безпеки: Системи виявлення аномалій повинні бути пов'язані з іншими безпековими інструментами, такими як антивірусне програмне забезпечення, фаєрволи, і системи запобігання вторгненням (IPS) для обміну даними та координованої реакції на загрози [14].

2. Автоматизація процесів: Автоматизація процесів реагування на інциденти може значно покращити швидкість та ефективність реагування на аномалії, зменшуючи вплив потенційних загроз.

3. Навчання персоналу: Забезпечення, що співробітники організації розуміють, як користуватися системами виявлення аномалій та як реагувати на тривоги, є критично важливим для успішної роботи цих систем.

Системи виявлення аномалій вимагають постійного моніторингу та оновлення для забезпечення їх актуальності відповідно до нових загроз та змін у поведінці користувачів:

1. Регулярне оновлення моделей: Алгоритми та моделі повинні регулярно оновлюватися на основі нових даних та тенденцій у поведінці користувачів для забезпечення їхньої релевантності та точності.

2. Аналіз ефективності: Постійний аналіз ефективності системи виявлення аномалій допомагає виявляти та коригувати проблеми у її роботі, а також адаптувати систему до нових загроз, що з'являються.

Через впровадження цих практик, системи виявлення аномалій можуть значно підвищити загальний рівень безпеки організації, допомагаючи їй ефективно протистояти сучасним кіберзагрозам. Оптимально налаштовані та інтегровані

системи виявлення аномалій дозволяють не лише оперативно реагувати на інциденти, але й прогнозувати потенційні загрози, адаптуючи захист до постійно змінюваних умов кіберпростору [15].

Завдяки визначенню базового рівня та наступному постійному моніторингу аномалій, організації мають можливість переходу від реактивних до проактивних стратегій кібербезпеки. Основні переваги проактивного підходу включають:

1. Зменшення часу детекції: Швидке виявлення аномалій забезпечує можливість оперативної відповіді на загрози, що знижує час між компрометацією системи та її виявленням.

2. Оптимізація витрат на безпеку: Ефективне виявлення аномалій дозволяє зосередитися на важливих загрозах, оптимізуючи використання ресурсів та знижуючи витрати на безпеку.

3. Підвищення довіри стейкхолдерів: Впровадження продвинутих технологій виявлення аномалій підвищує довіру клієнтів, партнерів та співробітників до рівня безпеки організації.

4. Комплексний захист інформації: Проактивні механізми захисту дозволяють не тільки виявляти та блокувати атаки, але й аналізувати їхні джерела та методи для постійного покращення системи безпеки.

Подальші дослідження у галузі виявлення аномалій можуть включати розвиток нових алгоритмів машинного навчання, кращу інтеграцію з іншими системами кібербезпеки, та створення адаптивних систем, які можуть самонавчатися і самоадаптуватися у відповідь на нові типи атак. Особлива увага також приділяється розробці методів для мінімізації помилкових спрацьовувань, що може значно покращити ефективність систем виявлення аномалій [16].

Завдяки постійному розвитку технологій і методологій, системи виявлення аномалій стануть ще більш інтелектуальними та ефективними, дозволяючи організаціям залишатися на крок попереду кіберзлочинців у боротьбі за безпеку їхніх систем і даних.

Для подальшого підвищення ефективності виявлення аномалій, особливу увагу приділяють розробці гібридних систем. Такі системи поєднують в собі кілька різних технологій і методів аналізу, зокрема:

1. Інтеграція з іншими системами безпеки: Наприклад, об'єднання можливостей SIEM систем та UBA інструментів для комплексного моніторингу та аналізу даних.

2. Комбінація моделей машинного навчання: Використання різних типів алгоритмів машинного навчання, таких як надзвичайні дерева, нейронні мережі та ансамблі моделей для підвищення точності та надійності виявлення аномалій.

3. Адаптація до контексту: Системи можуть адаптуватися до специфіки сектору або бізнесу, в якому вони використовуються, враховуючи особливості поведінки користувачів та типові операції в даній галузі.

Ці гібридні системи здатні більш ефективно ідентифікувати складні та витончені загрози, які можуть бути неочевидними для більш традиційних систем безпеки [17].

При впровадженні систем виявлення аномалій важливо також звернути увагу на прозорість їхньої роботи та етичні питання, зокрема:

1. Приватність даних: Забезпечення, що системи не порушують конфіденційність особистих даних користувачів і відповідають вимогам законодавства, такому як GDPR.

2. Обґрунтованість спрацьовувань: Системи повинні не тільки ефективно виявляти аномалії, але й забезпечувати можливість зрозуміти та оскаржити причини таких спрацьовувань, якщо це необхідно.

Забезпечення цих аспектів важливе не тільки для дотримання законодавчих вимог, але і для підтримки довіри користувачів і стейкхолдерів до систем безпеки.

В цілому, системи виявлення аномалій стають невід'ємною частиною стратегії кібербезпеки сучасних організацій. Вони дозволяють не тільки реагувати на інциденти, але й активно прогнозувати потенційні загрози, підвищуючи загальний рівень захисту інформаційних активів.

Важливим аспектом ефективного використання систем виявлення аномалій є створення в організації культури, зосередженої на аналізі даних та безпеки. Це передбачає:

- Підвищення обізнаності співробітників: Регулярне навчання та інформування співробітників про методи і практики безпеки, важливість виявлення аномалій та їхній вплив на безпеку організації.
- Залучення керівництва: Активна участь керівництва в процесах кібербезпеки і підтримка ініціатив по вдосконаленню заходів захисту даних.
- Забезпечення необхідних ресурсів: Алокація ресурсів для підтримки систем виявлення аномалій, включаючи технології, час співробітників і бюджет на вдосконалення існуючих систем.

Технологічний прогрес в області штучного інтелекту та машинного навчання обіцяє значні перспективи для подальшого розвитку систем виявлення аномалій. Серед можливих напрямків розвитку [18]:

- Інтеграція з штучним інтелектом: Більш глибока інтеграція з AI для розробки систем, здатних самонавчання і адаптації до нових умов безпеки без значного залучення людських ресурсів.
- Розширення можливостей прогнозування: Розвиток алгоритмів, які можуть не просто виявляти аномалії, але й прогнозувати потенційні вектори атак на основі актуальних тенденцій та даних.
- Застосування блокчейн технологій: Використання блокчейн для підвищення прозорості та надійності процесів збору та аналізу даних в системах виявлення аномалій.

За допомогою цих нововведень системи виявлення аномалій стануть ще більш точними, швидкими та ефективними у захисті критично важливих інформаційних активів. Ці інновації також зможуть забезпечити більшу автономію системам, зменшуючи потребу в постійному людському втручанні та сприяючи більш ефективному використанню ресурсів.

На основі аналізу сучасних практик та потенційного розвитку, можна сформулювати кілька ключових рекомендацій для організацій, які прагнуть покращити свої системи виявлення аномалій:

1. Безперервне оновлення та навчання моделей: Забезпечити, що системи виявлення аномалій регулярно оновлюються та вдосконалюються з урахуванням нових загроз та змін у поведінці користувачів.

2. Інтеграція з іншими інструментами безпеки: На максимум використовувати потенціал інтеграції систем виявлення аномалій з іншими інструментами безпеки для забезпечення комплексного підходу до кібербезпеки.

3. Фокус на користувацькому досвіді: Впроваджувати зручні та зрозумілі інтерфейси для керування системами виявлення аномалій, щоб забезпечити легкість у використанні та можливість швидкого реагування на загрози для всіх користувачів, незалежно від їх технічних навичок.

4. Етичні міркування та прозорість: У всіх процесах, пов’язаних із виявленням аномалій, важливо забезпечити етичний підхід і відкритість щодо методів збору та обробки даних.

Системи виявлення аномалій продовжують бути важливим компонентом стратегії кібербезпеки будь-якої організації. Завдяки постійному розвитку технологій та методологій, їх ефективність зростає, дозволяючи не тільки виявляти, але й передбачати потенційні загрози, забезпечуючи вищий рівень захисту інформаційних ресурсів. Впровадження інноваційних рішень та підхід, орієнтований на дані, дозволяє компаніям створювати більш безпечне та стійке до атак цифрове середовище [19].

1.2.2 Аналіз методів виявлення аномалій

Аналіз методів виявлення аномалій включає в себе дослідження та оцінку різних підходів, що застосовуються для ідентифікації відхилень від встановлених базових рівнів. Цей процес є ключовим для визначення потенційно зловмисних дій або технічних збоїв, які можуть призводити до компрометації системи.

Статистичні методи виявлення аномалій ґрунтуються на аналізі відхилень від статистичних параметрів нормальної поведінки. Ці методи часто включають:

- Z-оцінка (Standard Score): Вимірювання, наскільки далеко окреме спостереження знаходиться від середнього значення у вигляді стандартних відхилень.
- Grubbs' Test: Статистичний тест, що використовується для виявлення аутлайєрів в одновимірному наборі даних.
- Box Plot методи: Визначення аутлайєрів за допомогою візуалізації розподілу даних і визначення значень, що випадають за межі міжквартильного діапазону.

Ці методи можуть бути ефективними для датасетів з добре визначеною, очікуваною розподілом даних, але вони можуть бути не такими ефективними для складних або змінюваних поведінкових патернів [20].

Методи машинного навчання дозволяють системам виявлення аномалій адаптуватися і вчиться з даних, визначаючи потенційні загрози на основі ширшого контексту і зі складніших наборів даних. Ці методи включають:

1. Ненаглядоване навчання: Алгоритми, такі як k-середніх або ізоляційне лісування, які моделюють нормальну поведінку та виявляють відхилення як аномалії без використання позначеного навчального набору.
2. Напівнаглядоване навчання: Використання невеликої кількості позначених даних разом із великим обсягом непозначених даних для покращення точності виявлення.
3. Глибинне навчання: Використання нейронних мереж для ідентифікації складних патернів і залежностей в даних, які можуть вказувати на аномальну поведінку.

Гібридні системи виявлення аномалій комбінують кілька методів, включаючи статистичні підходи та машинне навчання, для підвищення точності і надійності виявлення аномалій. Ці системи можуть використовувати:

1. Ансамблеві методи: Застосування декількох алгоритмів аналізу даних одночасно для виявлення аномалій, де рішення про класифікацію випадку як

нормального або аномального приймається на основі "голосування" або взаємної згоди різних методів.

2. Каскадні системи: Послідовне застосування різних технік виявлення аномалій, де кожен наступний рівень активується тільки тоді, коли попередній рівень ідентифікує потенційну аномалію.

3. Використання контекстної інформації: Інтеграція додаткових даних, як-от час доби, роль користувача або тип пристрою, для більш точної інтерпретації поведінкових даних.

Ці підходи дозволяють не тільки підвищити точність виявлення аномалій, але й зменшити кількість помилкових спрацьовувань, що є критично важливим для підтримки високого рівня довіри користувачів до системи безпеки.

Попри значний прогрес у методах виявлення аномалій, існують виклики, які потрібно подолати для подальшого розвитку цих технологій:

1. Складність інтеграції: Інтеграція нових методів виявлення аномалій у вже існуючі системи безпеки може бути складною через технічні та організаційні обмеження.

2. Обробка великих даних: Ефективне виявлення аномалій у великих наборах даних вимагає значних обчислювальних ресурсів та розширених алгоритмів обробки даних.

3. Адаптація до змін у поведінці: Системи мають бути гнучкими та адаптивними, щоб відповідати на зміни в поведінці користувачів та еволюцію кіберзагроз.

Майбутнє розвитку методів виявлення аномалій обіцяє бути зосередженим на вдосконаленні адаптивності систем, підвищенні їхньої масштабованості та інтеграції з іншими компонентами кіберзахисту. Особливу увагу буде приділено наступним аспектам [21]:

1. Розвиток алгоритмів глибокого навчання: Завдяки своїй здатності виявляти складні патерни в даних, глибоке навчання може значно покращити точність систем виявлення аномалій. Розвиток нових моделей, які краще адаптуються до змінних умов та здатні обробляти великі обсяги даних, буде мати велике значення.

2. Підвищення масштабованості: Збільшення обсягів даних, з якими працюють організації, вимагає від систем виявлення аномалій здатності ефективно масштабуватися. Розвиток рішень, які можуть легко масштабуватися відповідно до зростаючих потреб у обробці даних, буде критично важливим.

3. Інтеграція з іншими системами безпеки: Ефективніше використання даних, отриманих з різних систем, таких як IDS, фаєрволи, і системи управління подіями безпеки (SIEM), може підвищити загальну ефективність виявлення аномалій. Інтеграція даних з цих систем в один аналітичний інструмент дозволить забезпечити більш точний і комплексний аналіз безпекових подій.

4. Забезпечення приватності та відповідності законодавству: Важливо, щоб розвиток та впровадження нових методів виявлення аномалій відбувалися з повагою до приватності користувачів та відповідно до законодавчих вимог, таких як GDPR. Це включає розробку технологій, які можуть забезпечити захист даних на всіх етапах обробки і зберігання.

За допомогою цих напрямків розвитку систем виявлення аномалій можна значно підвищити рівень кіберзахисту організацій і забезпечити їх здатність адаптуватися до постійно змінюваних умов у світі кіберзагроз [22].

Забезпечення адаптивності і реагування на нові виклики є основою для створення міцної і надійної системи виявлення аномалій, яка може захистити від складних і витончених кібератак.

Автоматизація процесів виявлення аномалій є ключовим фактором для підвищення швидкості і точності реакції на загрози. Розвиток технологій, таких як автоматичне навчання і штучний інтелект, дозволяє системам самостійно адаптуватися і оптимізувати процеси аналізу даних без постійного втручання людини. Це допомагає виявляти нові шаблони поведінки, які можуть свідчити про аномалії, і зменшувати кількість помилкових тривог.

Співпраця між організаціями та обмін інформацією про загрози є важливим аспектом у посиленні загальної безпеки. Участь у спільнотах, які займаються кібербезпекою, може допомогти організаціям отримувати важливі дані про нові методи атак та рекомендації щодо захисту. Наприклад, участь у програмах обміну

інформацією про загрози (threat intelligence sharing programs) може забезпечити доступ до актуальної інформації, яка допоможе підвищити ефективність виявлення аномалій і захисту даних [23].

На законодавчому рівні, важливо, щоб системи виявлення аномалій відповідали всім вимогам конфіденційності та захисту даних. Підтримка відповідності до таких норм, як GDPR в Європі, CCPA в Каліфорнії та інші регуляторні стандарти, є необхідною для забезпечення законності обробки даних та захисту інформації користувачів.

Системи виявлення аномалій відіграють ключову роль у захисті організацій від кіберзагроз. За допомогою передових технологій, глибокої інтеграції, автоматизації, співпраці та дотримання нормативних вимог, можливо значно підвищити рівень безпеки і ефективності реакції на аномалії та інші безпекові інциденти. Сучасні підходи до виявлення аномалій дозволяють не тільки реагувати на вже відомі загрози, але й адаптуватися до нових викликів, забезпечуючи більш стійкий захист від потенційних кібератак.

Для подальшого підвищення ефективності систем виявлення аномалій, важливо враховувати декілька ключових викликів:

1. Складність даних: Великі обсяги даних, що постійно зростають, вимагають більш потужних інструментів для їх аналізу і обробки, а також більш ефективних алгоритмів для фільтрації та ідентифікації релевантної інформації.

2. Швидкість реагування: Збільшення швидкості обробки даних та виявлення аномалій є критично важливим для зменшення часу між виявленням загрози та її нейтралізацією.

3. Інтеграція та кооперація: Розвиток міжплатформенної інтеграції та краща координація між різними інструментами і системами безпеки забезпечить більш комплексний підхід до кібербезпеки.

4. Захист приватності: Необхідно збалансувати потребу в захисті даних з вимогами конфіденційності та приватності, забезпечуючи, що системи виявлення аномалій не порушують права і свободи індивідів.

Розвиток систем виявлення аномалій відкриває нові можливості для захисту кіберпростору в умовах постійно змінюваних та еволюціонуючих кіберзагроз. За допомогою інноваційних технологій, вдосконалених методик та ефективного впровадження, ці системи стануть ще більш надійним захистом для організацій у всьому світі. Вкладення в розвиток і підтримку цих технологій є ключовим для забезпечення майбутнього безпеки в цифровому віці [24].

Враховуючи швидкий розвиток кіберзагроз, особливо важливим стає не тільки розвиток існуючих технологій, але й інновації, які можуть передбачити та нейтралізувати майбутні загрози ще до їх реалізації. Стратегічне планування та інвестиції в дослідження в області штучного інтелекту, машинного навчання та кібербезпеки є критично важливими для адаптації до мінливого ландшафту загроз.

Кіберзагрози не знають кордонів, тому співпраця на міжнародному рівні є надзвичайно важливою. Обмін інформацією про загрози, спільні дослідницькі ініціативи та уніфікація стандартів безпеки можуть значно підвищити ефективність відповіді на інциденти та зміцнити захист на глобальному рівні.

Застосування розумних технологій, таких як предиктивна аналітика та автоматизоване виявлення аномалій, може допомогти організаціям не тільки реагувати на існуючі загрози, але й прогнозувати потенційні уразливості. Ці технології дозволяють аналізувати великі обсяги даних, виявляти закономірності та надавати рекомендації щодо підвищення безпеки в реальному часі [25].

Збір та аналіз даних повинен проводитися з дотриманням строгих етичних норм. Забезпечення прозорості використання даних, захист особистої інформації та відповідність законодавчим нормам є ключовими аспектами, що впливають на довіру користувачів та легітимність систем виявлення аномалій.

Освіта та підвищення обізнаності серед співробітників щодо основ кібербезпеки є ще одним важливим кроком. Регулярні тренінги, семінари та інформаційні сесії допомагають підвищити рівень обізнаності персоналу та зменшити ризик виникнення безпекових інцидентів через людський фактор. Заохочення культури безпеки в організації та залучення кожного співробітника до процесу захисту інформації можуть суттєво підвищити загальний рівень кіберзахисту.

Розвиток та впровадження систем виявлення аномалій є критично важливими для забезпечення безпеки в сучасному цифровому світі. Завдяки інтеграції новітніх технологій, підвищенню обізнаності та співпраці, організації можуть не тільки адекватно реагувати на поточні загрози, але й адаптуватися до нових викликів, забезпечуючи надійний захист своїх інформаційних активів [26].

1.2.3 Реагування на виявлені аномалії

Ефективне реагування на виявлені аномалії є ключовим аспектом систем кібербезпеки. Цей процес забезпечує, що після ідентифікації потенційної загрози, організація може швидко вжити заходів для мінімізації шкоди та відновлення нормальної операційної діяльності.

Перед тим, як вживати конкретні заходи реагування, важливо мати чітко визначені процедури, які включають:

- Розробку плану реагування на інциденти: План має містити інструкції для кожного етапу реагування, від ідентифікації та класифікації інцидентів до ізоляції заражених систем і звітування про інциденти.
- Призначення команди реагування на інциденти: Визначення ролей та відповідальностей членів команди, які будуть залучені до реагування на інциденти.
- Тренінг та навчання персоналу: Регулярне навчання персоналу процедурам реагування на інциденти для забезпечення швидкої та ефективної відповіді.

Процес реагування на інциденти можна розділити на декілька ключових етапів:

1. Ідентифікація інциденту: Швидке визначення аномалії як безпекового інциденту, що вимагає подальшого реагування.
2. Контейнмент ізоляції: Тимчасове ізолювання впливу інциденту для запобігання подальшого поширення або шкоди.
3. Аналіз: Глибокий аналіз інциденту для з'ясування причин і масштабу впливу.

4. Видалення загрози: Вжиття заходів для видалення шкідливих елементів з системи та відновлення безпеки.

5. Відновлення: Поступове відновлення доступу до систем та послуг з одночасним моніторингом на предмет повторного виникнення проблем.

6. Звітування та вдосконалення: Підготовка звітів про інциденти та використання отриманого досвіду для удосконалення майбутньої відповіді на загрози.

Завдяки сучасним технологіям, процес реагування на інциденти може бути значно автоматизованим та оптимізованим [27]:

1. Автоматизація процесів: Використання скриптів або автоматичних інструментів для швидкого ізолювання інфікованих систем, видалення шкідливого програмного забезпечення і застосування патчів безпеки може значно зменшити час реагування та помилки, які можуть виникнути під час ручного втручання.

2. Системи управління інцидентами: Платформи, які інтегрують різні інструменти безпеки та автоматизують збір даних та звітування про інциденти, забезпечують координоване та ефективне реагування.

3. Інструменти відновлення: Спеціалізоване програмне забезпечення, що дозволяє швидко відновлювати системи до стану до інциденту, може значно скоротити час простою та зменшити втрати від атаки.

Оскільки кіберзагрози постійно еволюціонують, важливо, щоб організації не тільки реагували на інциденти, але й навчалися з них. Аналіз кожного інциденту має використовуватись для вдосконалення політик безпеки, процедур реагування та навчальних програм для персоналу. Постійне вдосконалення цих аспектів допоможе організації залишатися крок попереду потенційних загроз.

Ефективне реагування на виявлені аномалії є критично важливим для мінімізації шкоди від кібератак. Інтеграція передових технологій, автоматизація процесів, та неперервне вдосконалення підходів до кіберзахисту є ключовими для підтримання високого рівня безпеки. Регулярні тренінги та аналіз після інцидентів гарантують, що організація не тільки відновиться після атак, але й стане більш стійкою до майбутніх загроз.

1.3 Машинне навчання

Використання машинного навчання в системах виявлення аномалій є одним із найперспективніших напрямків у сучасних технологіях кібербезпеки. Машинне навчання дозволяє системам не тільки ефективно виявляти відомі типи атак, але й адаптуватися до нових загроз, аналізуючи великі обсяги даних та ідентифікуючи потенційно небезпечні відхилення в поведінці.

1.3.1 Аналіз алгоритмів машинного навчання

Машинне навчання включає в себе декілька типів моделей та алгоритмів, які можуть бути ефективно застосовані для виявлення аномалій:

1. Наглядоване навчання: Цей підхід використовує попередньо позначені дані (які містять інформацію про нормальну поведінку та аномалії) для тренування моделі виявлення аномалій. Наприклад, алгоритми класифікації, такі як логістична регресія або дерева рішень, можуть визначати, чи є конкретна поведінка користувача аномальною [28].
2. Ненаглядоване навчання: В цьому випадку моделі навчаються ідентифікувати аномалії без попередньо позначених даних, аналізуючи лише вхідні дані на предмет виявлення статистичних відхилень. Популярні методи, такі як кластеризація (наприклад, K-means) або ізоляційний ліс, використовуються для виявлення випадків, які не вписуються в загальний паттерн поведінки.
3. Напівнаглядоване навчання: Ці методи комбінують елементи наглядованого і ненаглядованого навчання, використовуючи як позначені, так і непозначені дані для покращення точності і ефективності моделей.
4. Посилене навчання: Цей підхід дозволяє моделям машинного навчання самостійно визначати оптимальні стратегії дій на основі винагороди за відповідні рішення, що може бути застосовано для автоматичного налаштування параметрів безпеки відповідно до змін у поведінці користувачів чи системи.

Кожен з цих підходів має свої сильні та слабкі сторони, та в залежності від конкретних потреб і умов, може бути обраний відповідний метод для максимально ефективного виявлення аномалій.

Однією з найбільш перспективних галузей машинного навчання в контексті виявлення аномалій є глибоке навчання. Завдяки своїй здатності аналізувати великі обсяги даних і виявляти складні шаблони, глибоке навчання ідеально підходить для ідентифікації складних загроз, таких як складне шкідливе програмне забезпечення або фішингові атаки [29].

1. Згорткові нейронні мережі (CNN): Хоча традиційно використовуються для аналізу зображень, CNN також можуть бути адаптовані для аналізу часових рядів та інших послідовних даних, які часто зустрічаються в моніторингу мереж.

2. Рекурентні нейронні мережі (RNN), зокрема LSTM (Long Short-Term Memory): Ефективні для виявлення аномалій у часових рядах, RNN можуть аналізувати дані, де важливий контекст часу або послідовності.

3. Автоенкодери: Ці мережі навчаються відтворювати свої вхідні дані, при цьому аномалії часто погано відтворюються, що дозволяє їх ефективно виявляти.

Для підвищення ефективності систем виявлення аномалій на основі машинного навчання, важливо інтегрувати їх з іншими компонентами кіберзахисту:

- Інтеграція з SIEM системами: Інтеграція дозволяє використовувати результати аналізу машинного навчання для покращення реагування на інциденти та загального моніторингу.
- Забезпечення взаємодії з IDS/IPS системами: Це дозволяє автоматично блокувати або обмежувати потенційно шкідливий трафік на основі аналізу, проведеного моделями машинного навчання.

Завдяки постійному розвитку і інтеграції новітніх технологій, машинне навчання стає невід'ємною частиною систем кібербезпеки. Його здатність виявляти складні аномалії в реальному часі робить цей підхід надзвичайно цінним у боротьбі з кіберзагрозами. Однак, успішне впровадження машинного навчання вимагає не тільки технологічних інвестицій, але й забезпечення належного навчання персоналу,

підтримки інфраструктури та постійного аналізу ефективності застосованих моделей [30].

Для подальшого підвищення рівня безпеки систем на основі машинного навчання, важливо створити адаптивну оборону, яка може самостійно вивчати зміни у загрозах та адаптуватися до них. Це означає не тільки виявлення відомих загроз, але й прогнозування потенційних нових атак, що ще не були зафіксовані.

При інтеграції машинного навчання в кібербезпеку, критично важливо забезпечити прозорість процесів та механізмів виявлення. Організації повинні мати можливість контролювати та перевіряти, як алгоритми приймають рішення, особливо у випадках, коли це впливає на конфіденційність або інші чутливі аспекти діяльності.

Застосування машинного навчання в кібербезпеці відкриває значні можливості для підвищення ефективності захисних систем. Від наглядованого і ненаглядованого навчання до глибокого навчання та посиленого навчання, кожен метод пропонує унікальні переваги для ідентифікації і нейтралізації загроз. Поєднання цих технологій із традиційними методами безпеки дозволить створити більш гнучкі та ефективні системи, здатні адаптуватися до постійно змінюваного ландшафту кіберзагроз [31].

1.4 Аналіз типів аутентифікаційних даних

Ми зосередимось на розгляді різних типів аутентифікаційних даних, які використовуються в системах безпеки, та аналізі їх вразливостей та методів захисту. Аутентифікаційні дані є ключовими елементами у захисті ідентичності користувачів та систем, тому їхнє належне забезпечення та управління є критично важливим для загальної безпеки.

1.4.1 Ідентифікація типів аутентифікаційних даних

Аутентифікація в інформаційних системах використовує різні типи даних для підтвердження ідентичності користувачів. Розуміння цих типів є ключовим для

забезпечення належного рівня безпеки. Основні типи аутентифікаційних даних включають:

1. Щось, що знає користувач (Knowledge Factors):

- Паролі та ПІН-коди: Найпоширеніші форми аутентифікаційних даних, які вимагають від користувача введення секретної послідовності символів або чисел.
- Секретні питання: Часто використовуються як додатковий рівень безпеки або для відновлення доступу, включають відповіді на питання, відомі тільки користувачу.

2. Щось, що має користувач (Possession Factors):

- Безпекові токени: Фізичні пристрої, що генерують одноразовий код доступу, який користувач вводить під час аутентифікації.
- Смарт-карти: Використовуються для забезпечення фізичного та цифрового доступу, містять вбудовані чіпи з аутентифікаційною інформацією.
- Мобільні додатки: Програми, які генерують одноразові паролі або використовують пуш-нотифікації для підтвердження спроб входу в систему.

3. Щось, що є користувачем (Inherence Factors):

- Біометричні дані: Включають відбитки пальців, розпізнавання обличчя, сканування сітківки ока та інші унікальні фізіологічні характеристики.
- Поведінкові біометричні дані: Можуть включати аналіз підпису користувача, динаміку клавіш та інші особливості поведінки під час взаємодії з системою.

Кожен із цих типів має свої переваги та вразливості, і їх вибір залежить від вимог до безпеки, зручності для користувачів та інших факторів. Ефективне використання цих аутентифікаційних даних може значно покращити безпеку системи, але також вимагає врахування потенційних ризиків, пов'язаних із кожним методом.

1.4.2 Оцінка вразливості типів аутентифікаційних даних

Кожен тип аутентифікаційних даних має свої унікальні вразливості, які можуть бути використані зловмисниками для несанкціонованого доступу або інших шкідливих дій. Розуміння цих вразливостей є важливим для розробки ефективних заходів захисту.

1. Вразливості знань (Knowledge-based vulnerabilities):

- Слабкі паролі: Часто користувачі вибирають прості або легко вгадувані паролі, що значно спрощує їх злам.
- Соціальна інженерія: Зловмисники можуть використовувати техніки соціальної інженерії для отримання паролів або іншої конфіденційної інформації від користувачів.
- Phishing атаки: Поширена атака, що виманює аутентифікаційні дані за допомогою підроблених веб-сайтів або електронних листів.

2. Вразливості володіння (Possession-based vulnerabilities):

- Втрата або крадіжка пристрою: Фізичні токени, смарт-карти або мобільні телефони можуть бути вкрадені або загублені, що дає зловмисникам доступ до аутентифікаційних даних.
- Клонування: Смарт-карти або RFID-токени можуть бути клоновані з використанням спеціалізованого обладнання.
- Вразливості безпеки програмного забезпечення: Програми для генерації одноразових паролів можуть містити баги або вразливості, які дозволяють зловмисникам генерувати або перехоплювати коди.

3. Вразливості індивідуальності (Inherence-based vulnerabilities):

- Помилки біометричного розпізнавання: Недоліки в технологіях біометричного сканування можуть дозволити неавторизований доступ через неправильне розпізнавання або через обхід захисту.
- Підробка біометричних даних: Відбитки пальців, обличчя або інші біометричні дані можуть бути відтворені зловмисниками з використанням сучасних технологій.

- Навмисне блокування або знищення: Біометричні сканери можуть бути заблоковані або знищені, що перешкоджає нормальному доступу до системи.

1.4.3 Аналіз засобів для пом’якшення ризиків

Аналіз засобів для пом'якшення ризиків, пов'язаних з вразливостями аутентифікаційних даних, є важливим аспектом забезпечення безпеки в будь-якій організації. Ось деякі ключові стратегії та технології, які можуть бути використані для зміцнення захисту аутентифікаційних систем [32]:

1. Впровадження багаторівневої аутентифікації

Одним з найефективніших способів захисту аутентифікаційних даних є впровадження багаторівневої аутентифікації (Multi-Factor Authentication, MFA). Цей підхід включає використання двох або більше різних типів аутентифікаційних факторів:

- Щось, що знає користувач (наприклад, пароль).
- Щось, що має користувач (наприклад, безпековий токен або смартфон).
- Щось, що є користувачем (наприклад, біометричні дані).

Цей підхід значно ускладнює несанкціонований доступ, оскільки зловмисник повинен володіти декількома різними типами аутентифікаційних даних.

2. Розробка сильних політик паролів

Політики сильних паролів є основою для захисту аутентифікаційних систем, які базуються на знаннях. Ось деякі з ключових аспектів таких політик:

- Мінімальна довжина пароля: Встановлення вимог до мінімальної довжини пароля (наприклад, 8 символів).
- Складність: Вимоги до включення великих та малих літер, цифр та спеціальних символів.
- Регулярна зміна паролів: Наприклад, вимога змінювати паролі кожні 3-6 місяців.
- Заборона повторного використання старих паролів: Запобігання використанню раніше використаних паролів.

3. Захист фізичних та цифрових токенів

Для захисту токенів, які використовуються в аутентифікаційних процесах, необхідно розглянути такі заходи:

- Фізичний захист: Забезпечення безпечного зберігання токенів, обмеження доступу до них [33].
- Шифрування: Використання шифрування для захисту даних, які зберігаються на токенах.
- Управління доступом: Ретельне управління правами доступу до використання токенів для мінімізації ризику їх несанкціонованого використання.

4. Біометричні системи безпеки

Для біометричних систем необхідні додаткові заходи безпеки, щоб забезпечити їх надійність та захист від підробки:

- Лівенес детекція: Впровадження технологій розпізнавання "живості" для запобігання шахрайству з використанням фальшивих біометричних даних.
- Обмеження спроб доступу: Встановлення лімітів на кількість невдалих спроб доступу для уникнення брутфорс-атак.
- Шифрування біометричних даних: Забезпечення шифрування біометричних даних в усіх точках їх обробки та зберігання.

Висновки за розділом 1

У цьому розділі ми провели глибокий аналіз різних моделей та методів, які застосовуються в системах кібербезпеки для виявлення та відповіді на компрометацію аутентифікаційних даних. Основна увага була приділена чотирьом ключовим аспектам: аналізу поведінки користувача, методам виявлення аномалій, використанню машинного навчання та оцінці вразливостей різних типів аутентифікаційних даних.

Ми розглянули, як аналіз поведінки користувачів може допомогти в ідентифікації аномалій, які можуть вказувати на несанкціонований доступ або інші безпекові інциденти. Через збір та аналіз даних про поведінку користувачів, системи

можуть виявляти відхилення від звичайної активності, що є потенційними індикаторами загроз.

Методи виявлення аномалій були розглянуті з точки зору їх здатності визначати непередбачувані або атипові дії в мережі чи системі. Зокрема, важливість комплексного підходу, що включає статистичний аналіз, машинне навчання і навіть глибоке навчання, для покращення точності виявлення була підкреслена.

Машинне навчання виявилося особливо важливим у сучасних системах виявлення аномалій, здатним адаптуватися та вчиться з постійно змінюваних даних. Огляд алгоритмів машинного навчання показав, як різні підходи, від наглядованого до ненаглядованого навчання, можуть бути використані для ефективного виявлення загроз.

Остання частина розділу зосереджувалася на різних типах аутентифікаційних даних та їхніх вразливостях. Оцінка ризиків і стратегії їх пом'якшення для кожного типу даних були детально розглянуті, з акцентом на необхідності багаторівневої аутентифікації, політик сильних паролів, захисту фізичних та цифрових токенів та біометричних систем.

Загальний аналіз підкреслив, що комплексний підхід, який включає різні методи та технології, є необхідним для ефективного виявлення та реагування на компрометацію аутентифікаційних даних. Використання передових технологій, таких як машинне навчання, разом зі строгими процедурами безпеки та освітою користувачів, може значно знизити ризик і підвищити загальний рівень кібербезпеки.

РОЗДІЛ 2
СИНТЕЗ МОДЕЛІ ВИЯВЛЕННЯ КОМПРОМЕНТАЦІЇ АВТЕНТИФІКАЦІЙНИХ ДАНИХ

У сучасному світі, де залежність від цифрових технологій зростає з кожним днем, забезпечення безпеки аутентифікаційних даних стає надзвичайно важливим аспектом захисту інформаційних систем. Компрометація аутентифікаційних даних може призвести до серйозних порушень у функціонуванні систем, втрати конфіденційності, цілісності та доступності інформації, що нерідко закінчується фінансовими збитками та втратою довіри клієнтів.

Метою цього розділу є розробка і синтез моделі виявлення компрометації аутентифікаційних даних, яка б забезпечувала ефективне і своєчасне виявлення потенційних загроз і зловживань. Така модель повинна включати комплексний підхід до аналізу аномалій, що включає збір, обробку та аналіз даних з метою визначення неавторизованих або підозрілих дій, які можуть свідчити про спроби компрометації [34].

Цей розділ має стратегічне значення, оскільки ефективна модель виявлення компрометації аутентифікаційних даних може значно знизити ризики, пов'язані з безпекою інформації, та підвищити довіру користувачів і клієнтів до системи. Розробка такої моделі вимагає не тільки глибоких технічних знань, але й розуміння потреб бізнесу та особливостей роботи конкретної організації.

У цьому розділі будуть зібрані та систематизовані дані та методики, які допоможуть створити ефективний інструмент для боротьби з однією з найактуальніших проблем сучасного кіберпростору.

2.1 Формулювання вимог до моделі виявлення компрометації

Перед тим, як перейти до практичної реалізації моделі виявлення компрометації аутентифікаційних даних, важливо чітко сформулювати вимоги до неї. Це допоможе

забезпечити, що розроблена система буде відповідати всім технічним та функціональним очікуванням і зможе ефективно впоратися з викликами, пов’язаними з кібербезпекою.

Метою є встановлення чітких критеріїв і параметрів, які будуть використовуватися для оцінки ефективності майбутньої моделі, забезпечення її інтеграційної сумісності з іншими системами та визначення критичних зон, що потребують особливої уваги під час розробки [35].

Для цього нам треба поставити та виконати такі задачі:

- Визначення технічних вимог: Встановлення параметрів продуктивності, надійності, швидкості обробки даних та інших технічних характеристик.
- Формулювання функціональних вимог: Опис очікуваних функцій та можливостей моделі, включаючи інтеграцію з існуючими системами, легкість використання, та можливості адаптації до змінних умов експлуатації.
- Урахування управлінських аспектів: Врахування управлінських та стратегічних цілей організації в контексті використання моделі для забезпечення кібербезпеки.

2.1.1 Визначення основних вимог до моделі виявлення компрометації автентифікаційних даних

Розробка будь-якої моделі виявлення починається з визначення її основних вимог, які включають як технічні, так і функціональні аспекти. Вимоги повинні бути чітко сформульовані, щоб забезпечити всі необхідні характеристики моделі, що включають:

1. Точність виявлення: Модель повинна ефективно ідентифікувати реальні випадки компрометації без великої кількості хибнопозитивних або хибнонегативних результатів.
2. Швидкість обробки: Здатність моделі оперативно обробляти великі обсяги даних без значних затримок є критично важливою для систем, що функціонують в реальному часі.

3. Скальпованість: Модель має бути здатна масштабуватися відповідно до зростаючих обсягів даних та змін у інфраструктурі.

4. Інтегрованість: Сумісність з існуючими системами кібербезпеки та можливість легкої інтеграції з новими компонентами.

Ці вимоги формують основу для наступних етапів розробки, включаючи вибір технологій, методології проектування, та методи тестування моделі. Визначення цих вимог дозволяє забезпечити, що модель буде відповідати очікуванням та вимогам усіх зацікавлених сторін, включаючи кінцевих користувачів, розробників та управлінський склад [36].

Для того щоб детальніше описати методи збору та аналізу потреб користувачів та управлінських вимог до моделі виявлення компрометації автентифікаційних даних, необхідно вжити наступні кроки:

1. Оцінка Потреб Користувачів

Перший крок у визначенні вимог до моделі полягає в ідентифікації та аналізі потреб кінцевих користувачів, які безпосередньо взаємодіятимуть із системою. Це може включати:

- Інтерв'ю з користувачами: Спілкування з поточними користувачами систем безпеки для збору інформації про їхні потреби, очікування та потенційні виклики, з якими вони стикаються.
- Анкетування: Розсилка анкет серед широкої групи користувачів для збору кількісних та якісних даних про їх вимоги та уподобання.
- Аналіз зворотного зв'язку: Перегляд зворотного зв'язку від користувачів про існуючі системи, що включає скарги, пропозиції та відгуки.

Ці дані допоможуть виявити ключові характеристики, які користувачі вважають найбільш цінними, та ті аспекти, що потребують покращення.

2. Врахування Управлінських Вимог

Управлінські вимоги часто зосереджені на ефективності системи, її вартості, та способах інтеграції з іншими технологіями та бізнес-процесами. Для аналізу цих вимог можна використовувати [37]:

- Мітинги з управлінською командою: Обговорення з керівниками підрозділів та IT-фахівцями для визначення стратегічних цілей та технічних обмежень організації.
- Аналіз ROI (Return on Investment): Оцінка потенційної вигоди від впровадження моделі порівняно з витратами, що включає аналіз ефективності та потенційних ризиків.
- Оцінка сумісності: Перевірка, як нова модель може інтегруватися з існуючими системами безпеки, даними та інфраструктурою.

Визначивши ці вимоги, можна сформулювати чітку картину того, що від моделі виявлення компрометації аутентифікаційних даних очікують як користувачі, так і управління. Це дозволить створити рішення, яке не тільки технічно ефективне, але й адаптоване до специфіки діяльності організації та її стратегічних цілей [38].

3. Визначення Даних для Моделі

Після аналізу вимог та врахування усіх аспектів, наступним кроком є ідентифікація та визначення конкретних даних, які будуть використовуватися для тренування та тестування моделі виявлення компрометації. Це важливий аспект, оскільки від якості та релевантності даних залежить ефективність всієї системи. Для забезпечення максимальної ефективності моделі потрібно використовувати адекватні та репрезентативні дані. Ось декілька прикладів типів даних, які можуть бути корисними:

- Логи доступу до системи: Вони містять інформацію про час входу користувачів, їхні IP-адреси та типи дій, що виконуються під час сесії. Ці дані можуть допомогти ідентифікувати незвичайні вхідні точки або незвичайні часи доступу, які можуть вказувати на спроби несанкціонованого доступу.
- Запити до баз даних: Моніторинг і аналіз запитів, що виконуються до баз даних, може допомогти виявити неавторизовані або підозрілі дії, такі як спроби витягування великих обсягів даних.
- Зміни в системних конфігураціях: Неавторизовані зміни в налаштуваннях системи чи програмного забезпечення можуть бути індикаторами втручань.

- Дані з мережевих пристроїв: Включаючи трафік мережі, маршрути пакетів, та іншу інформацію, яка може виявити аномальні мережеві шаблони або потенційні DDoS атаки.

Ці дані потрібно зібрати, очистити від помилок або несуттєвої інформації та підготувати для подальшої обробки. Вони мають бути розділені на навчальні, валідаційні та тестові набори, що дозволяє не тільки ретельно навчити модель, але й адекватно оцінити її ефективність [39].

Ці кроки будуть детальніше розкриті в підпунктах 2.3 та 2.4, де ми обговоримо методи підготовки даних та їхнє використання у процесі моделювання та валідації моделі.

Перейдемо до аналізу потреб користувачів та врахування управлінських вимог.

2.1.2 Аналіз потреб користувачів та врахування управлінських вимог до швидкості та ефективності обробки даних

Успішна імплементація будь-якої системи значною мірою залежить від її прийнятності користувачами. Тому детальний аналіз їх потреб є критично важливим. Основні аспекти аналізу включають:

1. Зручність використання: Модель має бути інтуїтивно зрозумілою і легкою у використанні для кінцевих користувачів без необхідності глибоких технічних знань.
2. Швидкість реакції: Здатність системи швидко реагувати на запити користувачів та надавати відповіді без затримок.
3. Точність детекції: Висока точність у виявленні компрометацій важлива для користувачів, оскільки помилкові позитивні або негативні спрацьовування можуть вплинути на їхню роботу та довіру до системи.

Ці потреби можуть бути виявлені через безпосередні зворотні зв'язки, опитування, фокус-групи або аналіз даних використання існуючих систем.

Управлінські вимоги зосереджені на загальній ефективності системи та її впливі на організацію. Ключові аспекти включають:

1. Швидкість обробки даних: Модель має бути здатна обробляти великі обсяги даних в реальному часі, забезпечуючи актуальність інформації для оперативного прийняття рішень.

2. Ефективність ресурсів: Оптимізація використання комп'ютерних та мережевих ресурсів для мінімізації витрат і забезпечення стабільності системи.

3. Масштабованість: Здатність моделі адаптуватися до зростаючих обсягів даних або змін у структурі організації без значного додаткового втручання.

Інтеграція цих вимог забезпечить, що розроблена модель не тільки відповідає потребам користувачів, але й адаптована до стратегічних цілей організації, сприяючи її загальній ефективності та конкурентоспроможності [40].

Таким чином, ретельний аналіз потреб користувачів та управлінських вимог становить основу для створення моделі, яка не тільки технічно справна, але й максимально корисна та ефективна в реальних умовах експлуатації.

Після глибокого аналізу потреб користувачів та врахування управлінських вимог, наступним кроком є розробка методологій для їхнього втілення в моделі. Це включає створення детального плану дій, який охоплює всі аспекти від збору та аналізу даних до розробки та впровадження відповідних технологічних рішень.

Щоб реалізувати вимоги ми можемо виконати такі пункти:

1. Технічне втілення

- Система збору даних: Розробка або інтеграція систем, що збирають вхідні дані з різних джерел, таких як мережеві журнали, системи управління доступом, і інші інструменти збору даних.
- Аналітичні інструменти: Використання передових аналітичних та статистичних інструментів для аналізу даних, які допоможуть ідентифікувати патерни, що можуть вказувати на компрометацію аутентифікаційних даних.
- Моделювання та прогнозування: Розробка математичних моделей та алгоритмів машинного навчання для прогнозування ризиків та виявлення аномалій на основі аналізу великих даних [41].

2. Функціональне втілення

- Інтерфейси користувача: Розробка інтуїтивно зрозумілих інтерфейсів, які дозволяють користувачам легко інтерпретувати результати аналізу та вживати необхідні заходи.
- Автоматизація відповідей: Інтеграція системи з іншими внутрішніми інструментами для автоматизації реакцій на виявлені загрози, забезпечуючи швидке та ефективне вирішення проблем.

3. Управлінське втілення

- Звітність та моніторинг: Встановлення процедур для регулярної звітності та моніторингу стану системи безпеки, забезпечуючи управлінському складу актуальну інформацію для прийняття рішень.
- Навчання персоналу: Організація тренінгів та навчальних програм для співробітників, щоб забезпечити їх обізнаність з ризиками та методами їх запобігання.

Завершивши цей етап, організація зможе не тільки впровадити ефективну модель виявлення компрометацій, але й забезпечити її безперервне удосконалення та адаптацію до змінюваних умов та загроз в кіберпросторі. Ці заходи допоможуть створити міцну основу для захисту аутентифікаційних даних і підвищення загальної безпеки інформаційних систем.

2.2 Розробка моделі

Перейшовши до розробки моделі виявлення компрометації аутентифікаційних даних, важливо вибрати правильну методологію розробки, яка відповідатиме встановленим вимогам та цілям проекту. Методологія розробки повинна забезпечувати ефективність, адаптивність і можливість масштабування системи.

2.2.1 Вибір методології розробки моделі виявлення компрометації

Під час вибору методології розробки моделі важливо враховувати декілька ключових аспектів, таких як гнучкість методології, її відповідність проектним цілям,

а також досвід та переваги команди розробників. Розглянемо дві популярні методології:

1. Agile (Гнучка методологія): Agile підход включає ітераційний та інкрементальний процес розробки. Він дозволяє команді адаптуватися до змін у проектних вимогах та користувацькому зворотному зв'язку в реальному часі. Agile підход підходить для проектів, де вимоги можуть змінюватися або не повністю визначені на початковому етапі.

2. Waterfall (Каскадна методологія): Waterfall є традиційним підходом до розробки програмного забезпечення, де кожен етап проектування проходить послідовно: від аналізу вимог до впровадження та підтримки. Цей метод добре підходить для проектів з добре визначеними вимогами, які малоймовірно зміняться.

Для проекту моделі виявлення компрометації аутентифікаційних даних рекомендується вибрати Agile методологію з наступних причин:

- Змінність вимог: Сфера кібербезпеки постійно змінюється, нові загрози виникають швидко, що може вимагати швидких змін у проекті.
- Ітераційний розвиток: Дозволяє регулярно тестувати і оцінювати функціональність моделі, швидко виявляти проблеми та вносити необхідні корективи [42].
- Колаборація та зворотний зв'язок: Забезпечує тісну співпрацю між розробниками, аналітиками даних і кінцевими користувачами, що підвищує якість і релевантність розробленого продукту.

Agile методологія вже довела свою ефективність у багатьох проектах в сфері IT та кібербезпеки, де важлива швидка адаптація до нових викликів і технологій. Завдяки своїй гнучкості та орієнтації на кінцевого користувача, Agile допомагає створювати високоякісні, адаптивні рішення, які відповідають актуальним потребам безпеки.

Для подальшого розуміння важливості Agile методології в проектах кібербезпеки, розглянемо реальний кейс компанії, яка успішно імплементувала Agile для розробки своїх безпекових рішень.

Компанія Symantec, лідер у сфері кібербезпеки, використовувала Agile для розробки своєї платформи безпеки. Проект полягав у створенні новітньої системи виявлення загроз, що включала в себе широкий спектр захисних інструментів, від антивірусу до поведінкового аналізу [43].

Symantec застосувала Agile, щоб покращити колаборацію між розробниками, аналітиками і кінцевими користувачами. Завдяки ітераційному процесу, команда могла швидко впроваджувати нові ідеї, тестувати їх у реальних умовах і адаптувати під потреби ринку.

Ключові переваги Agile для Symantec:

- Швидкість адаптації: Agile дозволив Symantec ефективно реагувати на нові кіберзагрози, швидко інтегруючи нові функції захисту в систему.
- Тісний зворотний зв'язок: Неперервний зворотний зв'язок від користувачів допоміг точно визначити, які аспекти системи потребують удосконалення.
- Мінімізація ризиків: Раннє виявлення проблем в індивідуальних компонентах системи зменшило загальні ризики проекту та забезпечило більш стабільну роботу системи в цілому.

Цей приклад демонструє, як Agile може сприяти розробці складних систем кібербезпеки, де потрібна швидка адаптація до постійно змінюваних умов та вимог. Підходи, використані Symantec, можуть слугувати зразком для інших проектів у галузі кібербезпеки, де важливими є швидкість інновацій та здатність ефективно реагувати на нові виклики [44].

2.2.2 Опис архітектури моделі та вибір алгоритмів для реалізації

Розглянемо архітектуру моделі виявлення компрометації аутентифікаційних даних і на виборі алгоритмів, які будуть використані для її реалізації.

Архітектура моделі включає кілька ключових компонентів, які спільно працюють для забезпечення ефективного виявлення та реагування на аномалії в аутентифікаційних даних:

1. Джерела даних: Модель збирає дані з різноманітних джерел, таких як логи серверів, моніторинг мережевого трафіку, системи управління доступом і інші інформаційні системи.

2. Процесори даних: Відповідають за первинну обробку даних, що включає їх нормалізацію, очищення і попередній аналіз.

3. Хранилище даних: Централізоване сховище для зберігання оброблених даних, з якого модель витягує необхідну інформацію для аналізу.

4. Аналітичний модуль: Серце моделі, де використовуються алгоритми машинного навчання або статистичного аналізу для ідентифікації аномалій та потенційних загроз.

5. Інтерфейс користувача: Дозволяє кінцевим користувачам переглядати результати аналізу, отримувати сповіщення про аномалії та управляти налаштуваннями системи [45].

Вибір алгоритмів для моделі залежить від специфіки даних і вимог до точності та швидкості обробки. Розглянемо кілька популярних алгоритмів для виявлення аномалій:

1. Ізольоване дерево (Isolation Forest): Ефективний у виявленні аномалій в наборах даних, де аномалії рідкісні і відрізняються від норми.

2. Метод k-найближчих сусідів (k-NN): Використовується для виявлення точок, що значно відрізняються від їхніх найближчих сусідів, добре підходить для детекції складних патернів у даних.

3. Нейронні мережі: Зокрема, автоенкодери, які можуть ефективно виявляти аномалії шляхом реконструкції входів та визначення відхилень в реконструкції.

Ці алгоритми будуть використані для створення багаторівневої моделі, що здатна адаптуватися до різноманітних типів даних і сценаріїв виявлення загроз. Використання різних технік дозволяє покрити ширший спектр потенційних загроз та забезпечити вищу точність виявлення [46].

Для кращого розуміння структури моделі важливо візуалізувати її архітектуру. Використання блок-схем або діаграм компонентів допомагає уявити взаємозв'язки

між різними елементами системи. Це також полегшує сприйняття інформації для всіх зацікавлених сторін, включаючи технічних і не технічних користувачів.

Ключові компоненти моделі:

1. Джерела даних: Представлені у формі блоків, які показують, звідки модель отримує вхідні дані. Це можуть бути сервери, бази даних, мережеві пристрої тощо.

2. Процесор даних: Блок, що символізує функції очищення та нормалізації даних перед їх аналізом.

3. Хранилище даних: Зображене як центральний репозиторій, куди потрапляють оброблені дані для подальшого аналізу.

4. Аналітичний модуль: Серце моделі, де алгоритми обробляють дані та визначають аномалії. Цей блок включає алгоритми машинного навчання та статистичний аналіз.

5. Інтерфейс користувача: Демонструє, як результати аналізу представляються користувачам для огляду та вжиття заходів.

Обраний набір алгоритмів має забезпечити комплексний підхід до виявлення аномалій. Кожен з алгоритмів має свої унікальні переваги:

- Ізольоване дерево: Швидкий і ефективний в виявленні точкових аномалій. Цей метод добре підходить для систем, де аномалії виникають раптово і не вписуються в загальний шаблон поведінки.
- k-найближчих сусідів: Дозволяє виявляти аномалії, які відрізняються від більшості спостережень, забезпечуючи хороше розуміння локальної структури даних.
- Нейронні мережі (автоенкодери): Ефективні у виявленні складних шаблонів в даних, що можуть проявлятися у вигляді послідовних аномалій, і здатні адаптуватися до нових видів аномалій завдяки глибокому навчанню.

Ці алгоритми були обрані не тільки за їх технічну ефективність, але й за здатність адаптуватися до різних типів даних і взаємодії з іншими компонентами системи. Використання цих методів у комбінації дозволяє створити надійну і гнучку модель, здатну ефективно реагувати на широкий спектр загроз.

Продовжуючи розгляд підходів до вибору алгоритмів для моделі виявлення компрометації, важливо підкреслити значення практичного застосування цих методів на прикладі реальних компаній. Це дозволить краще зрозуміти, як теоретичні концепції можуть бути ефективно імплементовані в практичних бізнес-сценаріях [47].

Компанія PayPal, відома своїми інноваційними рішеннями у сфері онлайн-платежів, використовує складні алгоритми машинного навчання для виявлення та запобігання фінансовим шахрайствам. Завдяки цьому PayPal змогла значно знизити кількість незаконних транзакцій у своїй мережі, забезпечуючи високий рівень безпеки для своїх користувачів.

Особливості застосування в компанії PayPal:

- Інтеграція даних: PayPal обробляє величезні обсяги транзакційних даних з усього світу. Інтеграція цих даних у єдину модель дозволяє аналізувати та виявляти шаблони, які можуть свідчити про шахрайську діяльність.
- Використання ансамблевих методів: PayPal використовує комбінацію різних алгоритмів, включаючи ізольовані дерева та нейронні мережі, для створення комплексної моделі виявлення шахрайства. Це дозволяє ефективно ідентифікувати підозрілі транзакції, зменшуючи кількість помилкових спрацьовувань.
- Адаптація до нових загроз: Шахрайські схеми постійно еволюціонують, тому модель PayPal регулярно оновлюється для відповідності новим методам шахрайства. Це включає навчання на нових даних та адаптацію алгоритмів до змін у шахрайських техніках.

Вивчення досвіду PayPal у використанні машинного навчання для боротьби з кіберзагрозами надає цінні уроки для будь-якого проекту, спрямованого на виявлення компрометації аутентифікаційних даних:

1. Значення комплексного підходу: Використання різних видів алгоритмів забезпечує ширшу перспективу та кращу здатність виявляти різні види аномалій.
2. Важливість швидкості та адаптивності: Швидка обробка даних і здатність системи швидко адаптуватися до нових викликів є критично важливими для ефективності системи виявлення.

3. Неперервне оновлення та навчання: Регулярне оновлення моделі та алгоритмів на основі нових даних та зворотного зв'язку від користувачів допомагає підтримувати високу точність виявлення і знижувати ризики.

Ці інсайти можуть бути використані для розробки та налаштування моделі виявлення компрометації, забезпечуючи її ефективність і відповідність сучасним викликам кібербезпеки.

Завершуючи підпункт 2.2.2, розглянемо важливість інтеграції цих алгоритмів у єдину координовану систему, що забезпечує не тільки високу точність та швидкість виявлення, але й гнучкість в адаптації до нових загроз. Тепер, коли архітектура моделі чітко визначена і методологія обрана, перейдемо до наступної важливої частини процесу розробки — підготовці даних.

2.3 Підготовка даних до моделювання

Підготовка даних є одним з найважливіших етапів у процесі розробки моделі виявлення компрометації аутентифікаційних даних. Цей етап забезпечує, що дані, які використовуються для навчання та тестування моделі, є чистими, релевантними та представляють достатній обсяг інформації для ефективного навчання. Правильно підготовлені дані можуть значно покращити здатність моделі точно ідентифікувати справжні аномалії, в той час як неналежно оброблені дані можуть призвести до неправильних висновків та підвищеної кількості помилкових спрацьовувань.

Ключ до успіху моделі полягає не тільки в алгоритмах, які використовуються, але й у якості даних, на яких ці алгоритми навчаються. Чисті, добре структуровані дані без великої кількості викидів або аномалій, які не стосуються досліджуваної проблеми, забезпечують більш точне та ефективне моделювання. Підготовка даних включає декілька кроків: від первинного збору та оцінки якості даних до їх очищення, трансформації та, нарешті, поділу на навчальні та тестові набори [48].

2.3.1 Збір даних

Збір даних — це перший критичний крок у процесі підготовки даних. Цей етап включає ідентифікацію, визначення та збирання всіх потрібних даних, необхідних для навчання та тестування моделі. Важливо зібрати дані, які є репрезентативними для різних сценаріїв, з якими модель буде працювати в реальному світі.

Дані можуть походити з різних джерел, залежно від специфіки моделі та сфери її застосування:

- Системні логи: Журнали серверів, логи аудиту, записи дій користувачів на терміналах та інші системні взаємодії.
- Транзакційні дані: Відомості про фінансові транзакції, покупки онлайн, переміщення коштів тощо.
- Мережеві дані: Трафік мережі, звернення до портів, IP-адреси джерел та цілей, обмін пакетами.
- Дані з зовнішніх джерел: Соціальні медіа, публічні бази даних, датасети відкритих даних для збагачення внутрішніх даних новим контекстом.

Перед тим, як використовувати зібрані дані, важливо оцінити їхню якість. Це включає перевірку на наявність викидів, відсутніх значень, дублікатів, а також оцінку їхньої консистентності та повноти. На цьому етапі може бути використана візуалізація даних, статистичний аналіз або використання спеціалізованих інструментів для оцінки даних.

Збір якісних, релевантних даних є основою для побудови надійної моделі виявлення аномалій. Після збору даних наступний крок — їх очищення, що є не менш важливим для забезпечення точності моделі.

2.3.2 Очищення даних

Очищення даних є критичним процесом, який впливає на якість та ефективність моделі виявлення компрометації аутентифікаційних даних. Цей етап включає

видалення або корекцію неповних, некоректних, точних, або неактуальних частин даних з датасету.

Ключові аспекти очищення даних:

1. Видалення викидів (Outliers): Викиди можуть суттєво вплинути на результати моделювання, особливо в статистичних або машинно-навчальних алгоритмах. Виявлення та видалення аномальних значень, які не відображають загальну поведінку даних, є важливим для забезпечення точності моделі.

2. Обробка пропущених значень (Missing Values): Пропущені дані можуть виникнути з багатьох причин, таких як помилки в зборі даних або передачі. Важливо визначити та вирішити ці пропуски шляхом видалення записів, заповнення пропущених значень середніми або медіанними значеннями, або використання алгоритмів для імпутації.

3. Нормалізація даних: Нормалізація допомагає уніфікувати різні масштаби та діапазони значень, забезпечуючи, що кожна характеристика вносить рівнозначний вклад у аналітичні моделі. Це особливо важливо для моделей машинного навчання, де різні масштаби можуть спотворювати вагу характеристик.

4. Кодування категоріальних даних: Категоріальні дані необхідно перетворити у формат, який може бути оброблений алгоритмами машинного навчання. Це може включати використання одноразового кодування (one-hot encoding) або інших методів для перетворення текстових міток у числові індикатори.

Інструменти та методики для очищення даних:

- Пакети обробки даних в Python: Бібліотеки, такі як Pandas і NumPy, забезпечують широкі можливості для обробки та очищення даних, включаючи функції для обробки пропущених значень, нормалізації, та кодування категоріальних даних.
- Візуальний аналіз даних: Використання інструментів, таких як Matplotlib або Seaborn у Python, для візуалізації даних може допомогти виявити неправильності або аномалії, які не завжди видно при стандартному аналізі.

Ефективне очищення даних не тільки підвищує точність моделі, але й сприяє швидшій та більш ефективній обробці, зменшуючи ризики та витрати, пов'язані з

неправильними даними. Наступним кроком після очищення даних буде їх трансформація та підготовка до навчання моделі, про що ми поговоримо в подальших підпунктах [49].

2.3.3 Формування навчального, валідаційного та тестового наборів даних

Після того, як дані були зібрані та очищені, наступний критичний етап полягає в розділенні цих даних на навчальні, валідаційні та тестові набори. Це дозволяє забезпечити, що модель може бути адекватно навчена, налаштована та оцінена перед її впровадженням у виробництво.

Розділення даних на різні набори допомагає забезпечити, що модель не перенавчається (overfit) на конкретних даних та здатна ефективно узагальнювати на нових, невідомих раніше даних. Це критично для виробничих систем, де модель повинна коректно функціонувати під час реальних операцій.

Ключові кроки у формуванні наборів даних:

1. Навчальний набір (Training set): Це основна частина даних, яка використовується для первісного навчання моделі. Величина цього набору зазвичай становить 60-80% від всіх доступних даних.
2. Валідаційний набір (Validation set): Використовується для налаштування параметрів моделі та вибору найкращої версії моделі під час тренування. Типово складає 10-20% від усіх даних.
3. Тестовий набір (Test set): Використовується для остаточної оцінки ефективності моделі після того, як вона була навчена та налаштована. Тестовий набір повинен бути ізольованим від процесу навчання та використовуватися лише один раз для оцінки кінцевої продуктивності моделі. Цей набір також зазвичай становить 10-20%.

Методи розділення даних:

- Випадковий вибір (Random Sampling): Найпоширеніший метод, де дані вибираються випадково для кожного з наборів з гарантією, що кожен з наборів є репрезентативним для всієї датасету.

- Стратифікований вибір (Stratified Sampling): Використовується для забезпечення того, що кожен набір даних містить відповідне представлення різних класів або категорій. Це особливо важливо у випадках, коли деякі класи даних представлені неоднаково.

Інструменти для розділення даних:

- Бібліотеки Python, такі як Scikit-learn: Надають зручні інструменти та функції для розділення даних, зокрема train_test_split, які можуть бути налаштовані для випадкового або стратифікованого вибору [50].

Ретельне планування та виконання цього етапу є вирішальним для успішного застосування моделі в реальних умовах. Наступний крок, після розділення даних на відповідні набори, включатиме детальний аналіз статистичних характеристик цих даних, що є суттєвим для оцінки якості та ефективності моделі.

2.3.4 Статистичні характеристики даних

Оцінка статистичних характеристик даних є критичним етапом, який допомагає зрозуміти розподіл, тенденції та потенційні проблеми у датасетах, що використовуються для навчання, валідації та тестування моделі. Аналіз цих характеристик дозволяє оптимізувати параметри моделі та підвищити її точність та надійність.

Важливі статистичні метрики:

1. Середнє значення (Mean) і Медіана (Median): Допомагають зрозуміти центральну тенденцію даних. Відхилення медіани від середнього може вказувати на асиметрію розподілу.
2. Стандартне відхилення (Standard Deviation): Вимірює варіабельність або дисперсію даних відносно середнього значення. Велике стандартне відхилення може вказувати на великий розкид у даних, що важливо для визначення стабільності моделі.

3. Квартилі (Quartiles) і Міжквартильний розмах (Interquartile Range, IQR): Допомагають визначити розподіл даних і виявити потенційні викиди. IQR є мірою статистичної розкиду та стійкості до викидів.

4. Мінімальні і максимальні значення (Min/Max): Важливі для визначення границь даних і планування масштабування вхідних характеристик моделі.

5. Розподіл класів (Class Distribution): Особливо важливо для класифікаційних завдань, де нерівномірний розподіл класів може призвести до упередженої моделі, що надто добре працює на одних класах і погано на інших.

Методи аналізу статистичних характеристик:

- Візуалізація даних: Використання графічних зображень, таких як гістограми, боксплоти та точкові діаграми, допомагає візуально оцінити розподіл та властивості даних. Це може виявити непомічені раніше аспекти даних, такі як кластеризація або аномалії.
- Статистичне тестування: Використання статистичних тестів, таких як t-тест, ANOVA або хі-квадрат, для оцінки гіпотез про дані, що може включати відмінності між групами або вплив різних факторів.

Інструменти для статистичного аналізу:

- Бібліотеки Python, такі як Pandas, NumPy, SciPy, Matplotlib: Ці інструменти дозволяють легко маніпулювати даними та проводити різноманітні статистичні аналізи та візуалізації.

Ефективний статистичний аналіз даних забезпечує міцну основу для розробки надійних та ефективних моделей машинного навчання, дозволяючи краще зрозуміти і врахувати особливості даних, на яких базується модель. Це критично для досягнення високої точності та ефективності у виявленні аномалій та компрометацій.

2.4 Моделювання

Моделювання — це процес розробки та тренування математичних або статистичних моделей на основі підготовлених даних з метою виявлення аномалій

або прогнозування певних подій. В цьому контексті, метою є створення ефективної моделі виявлення компрометації аутентифікаційних даних.

2.4.1 Вибір та обґрунтування алгоритму навчання

Перший крок у моделюванні — вибір алгоритму навчання, який найкраще відповідає задачам проекту та характеристикам даних.

2.4.1.1 Вимоги до алгоритму навчання

При виборі алгоритму навчання для моделі виявлення компрометації аутентифікаційних даних необхідно враховувати наступні ключові вимоги:

1. Точність: Алгоритм має забезпечувати високий рівень точності у виявленні справжніх аномалій, мінімізуючи при цьому кількість помилкових спрацьовувань.
2. Швидкість обчислень: Алгоритм має бути достатньо швидким для обробки великих обсягів даних у реальному часі, особливо в умовах, де швидкість реагування є критичною.
3. Масштабованість: Алгоритм повинен ефективно масштабуватися з ростом обсягу даних та збільшенням складності задач.
4. Гнучкість: Здатність адаптуватися до змін у поведінці користувачів або еволюції атак, що вимагає можливості оновлення та налаштування алгоритму без переривання його функціонування.

2.4.1.2 Аналіз алгоритмів навчання

У рамках нашої роботи ми розглядаємо наступні алгоритми, які були обрані з огляду на їх придатність до задач виявлення аномалій в аутентифікаційних даних:

1. Ізольовані дерева (Isolation Forest): Цей алгоритм ефективно виявляє аномалії в датасетах, де аномалії мають тенденцію бути "ізольованими" від більшості

звичайних спостережень. Він чудово підходить для наших цілей, оскільки дозволяє швидко обробляти великі обсяги даних, що є важливим для роботи в реальному часі.

2. Однокласовий SVM (One-Class SVM): Цей метод підходить для ситуацій, коли аномалії рідкісні або коли датасет має невелику кількість аномальних міток. Однокласовий SVM створює модель, що відокремлює всі звичайні дані від аномалій, що дозволяє ефективно виявляти незвичні патерни поведінки.

3. Автоенкодери: Це тип нейронної мережі, який використовується для ненагляданого навчання. Автоенкодери ефективні для виявлення аномалій, оскільки вони намагаються відтворити свій вхід, а аномалії часто відрізняються високою помилкою реконструкції. Це робить їх корисними для ідентифікації аномальних даних, які не вписуються в звичайні шаблони.

Кожен з цих алгоритмів має свої переваги та недоліки, і вибір конкретного алгоритму залежить від специфіки даних і вимог до моделі. Важливо провести емпіричні тести на реальних даних, щоб оцінити ефективність кожного алгоритму в конкретних умовах.

Комбінуючи ці три алгоритми, ми можемо досягти високої точності та чутливості у виявленні аномалій. Використання ізольованих дерев допоможе швидко ідентифікувати явні аномалії, однокласовий SVM забезпечить точне визначення меж нормальної поведінки, а автоенкодери дозволять виявляти складні та субтельні аномальні патерни, які можуть вказувати на більш тонкі або складні атаки.

З урахуванням цих аналізів, наступний крок у моделюванні буде полягати в навчанні моделі на навчальному наборі даних, про що ми поговоримо у підпункті 2.4.2.

2.4.2 Навчання моделі на навчальному наборі даних

Навчання моделі — це процес, під час якого обраний алгоритм навчається на основі підготовлених навчальних даних для побудови передбачуваної моделі. Це дозволяє моделі виявляти аномалії на основі характеристик та патернів, які вона засвоює з даних. У цьому підпункті ми детально опишемо процес навчання для

кожного обраного алгоритму, включаючи налаштування параметрів та оцінку ефективності.

Загальна методологія навчання

1. Завантаження та підготовка навчального набору даних:

- Завантаження даних: Зчитування даних з попередньо підготовленого джерела (файл або база даних).
- Перетворення даних: Перетворення категоріальних даних у числові, нормалізація або стандартизація числових характеристик.
- Виділення ознак та міток: Поділ даних на вхідні ознаки та мітки аномальних і нормальних класів.

2. Обробка класового дисбалансу:

- Стратифіковане вибіркове тренування: Забезпечення рівномірного розподілу класів у навчальному наборі.
- Методи перевибірки: Oversampling (перевибірка меншого класу) або Undersampling (зменшення більшого класу) для збалансування класів.

3. Навчання моделі на навчальному наборі:

- Ініціалізація алгоритму: Налаштування основних параметрів для кожного алгоритму.
- Тренування: Проведення навчання моделі на навчальних даних з використанням алгоритму машинного навчання.
- Тонке налаштування: Оптимізація гіперпараметрів за допомогою валідаційного набору або методів крос-валідації.

1. Навчання ізольованих дерев (Isolation Forest)

- Ініціалізація та тренування:

```
from sklearn.ensemble import IsolationForest
iso_forest = IsolationForest(n_estimators=100, max_samples='auto', contamination=0.1, random_state=42)
iso_forest.fit(X_train)
```

- Визначення аномалій:

```
y_pred_train = iso_forest.predict(X_train)
y_pred_test = iso_forest.predict(X_test)
```

- Тонке налаштування:

n_estimators: Кількість дерев у лісі (рекомендується діапазон 50-200).

max_samples: Максимальна кількість зразків для навчання кожного дерева.

contamination: Пропорція аномалій у наборі даних.

2. Навчання однокласового SVM (One-Class SVM)

- Ініціалізація та тренування:

```
from sklearn.svm import OneClassSVM
oc_svm = OneClassSVM(kernel='rbf', gamma='auto', nu=0.1)
oc_svm.fit(X_train)
```

- Визначення аномалій:

```
y_pred_train = oc_svm.predict(X_train)
y_pred_test = oc_svm.predict(X_test)
```

- Тонке налаштування:

kernel: Ядро для SVM (рекомендується rbf або linear).

gamma: Визначає ширину ядра, обирається експериментально.

nu: Пропорція аномальних даних у наборі.

3. Навчання автоенкодерів

- Ініціалізація та архітектура:

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

autoencoder = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(32, activation='relu'),
    Dense(64, activation='relu'),
    Dense(X_train.shape[1], activation='sigmoid')
])
autoencoder.compile(optimizer='adam', loss='mse')
```

- Тренування:

```
history = autoencoder.fit(X_train, X_train, epochs=50, batch_size=32, validation_data=(X_val, X_val), shuffle=True)
```

- Визначення аномалій:

```
reconstructions = autoencoder.predict(X_test)
mse = np.mean(np.square(X_test - reconstructions), axis=1)
threshold = np.percentile(mse, 95)
y_pred_test = np.where(mse > threshold, -1, 1)
```

- Тонке налаштування:

1. Кількість шарів та нейронів: Підбирається експериментально на основі розміру набору даних.
2. Порогове значення (threshold): Обирається на основі відсотка допустимих помилок.
3. Оцінка ефективності навчання

Після навчання моделей необхідно оцінити їх ефективність за допомогою різних метрик:

1. Точність (Accuracy): Пропорція правильно класифікованих прикладів.
2. Повнота (Recall): Пропорція правильно виявлених аномалій.
3. Точність (Precision): Пропорція справжніх аномалій серед усіх виявлених.
4. F1-міра: Гармонійне середнє між точністю та повнотою.
5. AUC-ROC: Площа під кривою характеристик приймача.

Приклад коду для оцінки:

```
from sklearn.metrics import classification_report, roc_auc_score, f1_score

# Оцінка на тестових даних
print(classification_report(y_test, y_pred_test))
roc_auc = roc_auc_score(y_test, y_pred_test)
f1 = f1_score(y_test, y_pred_test)

print(f'AUC-ROC: {roc_auc}')
print(f'F1 Score: {f1}')
```

2.5 Перевірка адекватності розробленої моделі

Після навчання моделі, наступний крок полягає у її ретельній перевірці, щоб забезпечити, що вона відповідає поставленим вимогам та ефективно працює на

контрольних та тестових наборах даних. Цей процес включає декілька важливих аспектів, включаючи оцінку ефективності моделі та аналіз причин помилок.

Опис процесу навчання моделі на навчальному наборі даних:

1. Підготовка даних: Навчальний набір даних вже було очищено, трансформовано і розділено на тренувальний та валідаційний набори, як описано в попередніх розділах.

2. Параметризація моделі: Враховуючи вимоги до точності та швидкості, моделі було задано відповідні параметри, засновані на результати аналізу алгоритмів.

3. Тренування моделі: Модель була навчена на навчальному наборі, використовуючи техніки машинного навчання, що були обрані на основі їх придатності до типу даних і задачі детекції аномалій.

Показники ефективності моделі на валідаційному та тестовому наборах даних:

1. Валідаційний набір:

- Модель була перш за все перевірена на валідаційному наборі для тонкого налаштування параметрів і визначення її загальної ефективності перед фінальним тестуванням.
- Було обчислено ключові метрики, такі як точність, F1-міра, і AUC-ROC, для оцінки її продуктивності.

2. Тестовий набір:

- Остаточне тестування моделі проводилось на тестовому наборі, який не брав участь у тренуванні або валідації.
- Знову ж таки, було обчислено точність, F1-міра, і AUC-ROC, а також інші релевантні метрики для всебічної оцінки продуктивності.

Таким чином, ми завершуємо навчання моделей і оцінюємо їхню ефективність перед тим, як переходити до перевірки їх адекватності на контрольних наборах даних.

Аналіз можливих причин помилок та способи їх покращення:

1. Аналіз помилок:

- Детальний аналіз випадків, де модель дала неправильні результати, дозволить визначити слабкі сторони та потенційні зони для покращення.

- Буде проведено аналіз типів помилок (наприклад, хибнопозитивні та хибнонегативні виявлення) для розуміння характеру та причин невдач.

2. Пропозиції щодо покращення:

- На основі аналізу помилок, можуть бути запропоновані зміни в параметрах моделі, тренувальних даних або навіть у самому алгоритмі.
- Рекомендації можуть включати розширення навчального набору, використання додаткових технік очищення даних, або впровадження новітніших алгоритмів для покращення результатів.

Цей підхід дозволить не тільки оцінити реальну продуктивність моделі, але й забезпечити її надійність та ефективність у вирішенні поставленої задачі виявлення компрометації аутентифікаційних даних.

2.6 Порівняння запропонованого рішення з існуючими аналогами

Цей розділ зосереджується на порівнянні нашої розробленої моделі з іншими існуючими рішеннями, аналізує їхні сильні та слабкі сторони, та вказує, як наша модель може вирішувати проблеми, які мають інші рішення.

Для глибокого та практичного аналізу в рамках порівняння нашої розробленої моделі з існуючими аналогами, давайте виберемо дві популярні системи виявлення аномалій, які широко використовуються в індустрії: Splunk та Elasticsearch (ELK Stack). Ці системи надають багаті можливості для моніторингу, аналізу даних та виявлення аномалій.

Порівняльний аналіз існуючих систем із розробленою моделлю

1. Splunk

Splunk — це платформа для аналізу та візуалізації великих обсягів машинних даних. Вона часто використовується для моніторингу безпеки та виявлення аномалій у системах IT-інфраструктури.

- Переваги:

1. Висока швидкість обробки великих датасетів.
2. Гнучкість у налаштуванні та інтеграції з різними джерелами даних.

3. Сильна підтримка спільноти та наявність готових до використання модулів.

- Недоліки:

1. Висока вартість ліцензування.
2. Складність налаштувань для специфічних задач.
3. Потреба в значних ресурсах для оптимізації та масштабування.

2. Elasticsearch (ELK Stack)

Elasticsearch разом з Logstash та Kibana (ELK Stack) використовується для комплексного аналізу логів та виявлення аномалій.

- Переваги:

1. Відкритий вихідний код і широкі можливості кастомізації.
2. Масштабованість та здатність до швидкої обробки великих обсягів даних.
3. Зручність візуалізації та моніторингу у реальному часі через Kibana.

- Недоліки:

1. Складності з налаштуванням та управлінням при масштабуванні.
2. Потреба у висококваліфікованих спеціалістах для повноцінної роботи зі стеком.
3. Відносно нижча швидкість обробки порівняно з платними аналогами при великих обсягах даних.

Порівняння з розробленою моделлю

Наша модель має наступні відмінності та переваги:

1. Точність і адаптивність: На відміну від Splunk і ELK, наша модель більш адаптивна до нових типів аномалій завдяки використанню сучасних алгоритмів машинного навчання.
2. Економічна ефективність: Значно нижча вартість впровадження та експлуатації порівняно з комерційними рішеннями, що робить її доступною для малих та середніх підприємств.
3. Простота інтеграції: Модель легко інтегрувати з існуючими системами збору даних без необхідності складного налаштування.

Це порівняння демонструє, що наша модель може бути конкурентоспроможною альтернативою на ринку зі своїми унікальними характеристиками та перевагами, особливо в контексті гнучкості та вартості.

Висновки за розділом 2

У другому розділі дослідження було зосереджено увагу на синтезі моделі виявлення компрометації аутентифікаційних даних, яка була розроблена на основі глибокого аналізу вимог до системи, вибору та обґрунтування використання конкретних алгоритмів навчання, а також підготовки та обробки даних для ефективного моделювання.

На основі теоретичних та практичних знань було вибрано три ключових алгоритми: Ізольовані дерева, Однокласовий SVM та Автоенкодери, які були адаптовані для задачі виявлення аномалій в аутентифікаційних даних. Ці алгоритми були навчені на підготовлених навчальних даних, після чого ефективність моделей була оцінена на валідаційних та тестових наборах. Отримані результати продемонстрували адекватність вибраних методів, хоча й виявили деякі можливості для подальших покращень.

Порівняння розробленої моделі з існуючими рішеннями, такими як Splunk та ELK Stack, підтвердило її конкурентоспроможність. Особливо важливими є переваги моделі у вигляді адаптивності до нових типів аномалій, нижчої вартості впровадження та легкості інтеграції з іншими системами. Ці аспекти забезпечують значний потенціал для її застосування в різних організаційних контекстах, особливо де важливі економічна ефективність та швидка адаптація до змінюваних умов.

Загалом, робота виконана в цьому розділі забезпечує міцну основу для подальших досліджень і розвитку в області виявлення аномалій в аутентифікаційних системах, вносячи вагомий вклад у зміцнення кібербезпеки організацій. Водночас, виявлені обмеження та можливості для покращення вказують на напрямки майбутніх досліджень і вдосконалення існуючих методів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ МОДЕЛІ ВИЯВЛЕННЯ КОМПРОМЕТАЦІЙ

Цей розділ зосереджений на практичній реалізації моделі для виявлення компрометацій аутентифікаційних даних. Ми продемонструємо, як обрані методи та інструменти вирішують задачі кібербезпеки, детально описуючи кожен етап розробки — від підготовки даних до їх тестування та оптимізації. Розділ висвітлює використання новітніх алгоритмів машинного навчання, забезпечуючи високу точність і швидкість виявлення потенційних загроз.

Наукова новизна полягає в застосуванні сучасних підходів для аналізу даних і використанні алгоритмів, які підвищують ефективність системи виявлення аномалій. Модель розробляється в мові Python 3 на операційній системі Kali Linux, обраній через її адаптацію до потреб кібербезпеки і наявність спеціалізованих інструментів.

Цей вступ встановлює основу для ретельного аналізу та демонстрації процесу розробки, акцентуючи на важливості досліджуваної теми та її актуальності.

3.1 Вибір програмного забезпечення та інструментів

3.1.1 Обґрунтування вибору Python та деяких бібліотек

Python є однією з найпопулярніших мов програмування у світі, особливо у сферах науки про дані, штучного інтелекту та кібербезпеки, завдяки своїй гнучкості, зрозумілості та величезному співтовариству. Python підтримує численні бібліотеки, які великою мірою спрощують процес аналізу даних і машинного навчання, такі як NumPy, Pandas, Scikit-learn, TensorFlow та Keras. Ці інструменти забезпечують потужні функціональні можливості для обробки даних, статистичного аналізу, моделювання, тренування та тестування моделей машинного навчання.

Ключові переваги Python:

1. Гнучкість: Python може бути використаний у різних операційних системах і середовищах, що робить його ідеальним для використання на Kali Linux, системі, яка часто використовується для тестування проникнення і кібербезпеки.

2. Широкий спектр бібліотек: Підтримка бібліотек як TensorFlow для побудови нейронних мереж, Scikit-learn для класичного машинного навчання та Pandas для аналізу даних.

3. Спільнота та підтримка: Велике і активне співтовариство розробників, яке надає підтримку, документацію, готові рішення та відкритий код для вирішення майже будь-яких задач.

3.1.2 Огляд середовища розробки (Kali Linux, Python-інтерпретатор)

Kali Linux є підвидом ОС Linux, спеціалізованим на кібербезпеці, що надає ряд предвстановлених інструментів для тестування проникнення, аналізу безпеки та відновлення даних. Використання Kali Linux як основного середовища для розробки та тестування моделі дозволяє ефективно інтегрувати процес розробки з інструментами аналізу безпеки.

Особливості Kali Linux:

1. Інструменти безпеки: Включає інструменти, такі як Metasploit, Nmap, Wireshark, що можуть бути використані для додаткових перевірок безпеки моделі.

2. Сумісність: Підтримує Python та його бібліотеки через попередньо встановлені пакети та можливість налаштування середовища розробки.

Ці інструменти та технології забезпечують високий рівень адаптивності та контролю над процесом розробки та дозволяють ефективно впроваджувати модель у реальних умовах кібербезпеки.

3.2 Розробка моделі

3.2.1 Опис даних, що використовуються для навчання та тестування моделі

Для створення та валідації моделі виявлення компрометацій аутентифікаційних даних важливо вибрати відповідні дані, які можуть відображати реальні сценарії, з якими може зіткнутися система. Використання аутентичних даних забезпечує здатність моделі ефективно функціонувати в умовах реального світу.

Джерела даних:

1. Логи аутентифікації: Основним джерелом даних є логи аутентифікації, які зберігають записи про всі спроби входу в систему. Ці логи містять інформацію про час спроби входу, ідентифікатор користувача, IP-адресу, результат аутентифікації (успішно/невдало), та інші метадані.
2. Системні журнали: Додаткову інформацію можна зібрати з системних журналів, які можуть включати подробиці про системні виклики, зміни в системних конфігураціях або помилки безпеки, що мають відношення до аутентифікації.

Вимоги до даних:

- Змістовність: Дані повинні включати достатньо інформації для виявлення потенційних аномалій у поведінці користувачів.
- Повнота: Необхідно мати комплексні дані, які охоплюють різні сценарії аутентифікації та потенційні вектори атак.
- Чистота: Дані мають бути вільними від помилок та неактуальної інформації, яка може спотворити результати аналізу.

Перед використанням у тренуванні моделі, дані мають пройти кілька етапів підготовки:

- Видалення дублікатів: Переконатися, що всі записи є унікальними, щоб уникнути спотворення аналізу.
- Обробка відсутніх значень: Визначення та обробка записів, де деяка інформація відсутня, наприклад, через помилки логування або системні збої.

- Кодування категоріальних даних: Перетворення нечислових даних у формат, придатний для аналізу, використовуючи методи, як-от one-hot encoding або label encoding.

Після завершення цих етапів підготовки, дані будуть готові до використання в алгоритмах машинного навчання для побудови та тренування моделі. Це забезпечить, що модель може ефективно вчитися на реальних, чистих та репрезентативних даних, що значно підвищує її точність та надійність.

Для практичної частини нашого проекту дуже важливо вибрати реальні логи, які можуть бути використані для тренування та тестування моделі виявлення компрометацій. Один з підходів — використання логів від веб-серверів або аутентифікаційних серверів, які часто зберігають деталізовану інформацію про спроби входу.

Давайте розглянемо логи веб-сервера Apache як приклад. Ці логи зазвичай зберігають інформацію про кожен HTTP запит, що включає дату та час запиту, IP-адресу клієнта, деталі запиту, статус-код відповіді сервера та розмір відповіді. Ось приклад запису логу:

```
┌──(kali㉿kali)-[~/vikhrov/diploma]
└─$ cat apache.log
192.168.1.1 - - [10/Oct/2023:13:55:36 -0700] "GET /index.html HTTP/1.1" 200 1024
192.168.1.1 - - [10/Oct/2023:13:55:37 -0700] "POST /login.php HTTP/1.1" 403 182
203.0.113.5 - - [10/Oct/2023:13:56:20 -0700] "GET /admin.php HTTP/1.1" 401 2456
203.0.113.5 - - [10/Oct/2023:13:56:22 -0700] "POST /login.php HTTP/1.1" 200 384
```

Рисунок 3.1 – Формат логів веб-серверу

Кожен рядок логу містить наступну інформацію:

- IP-адреса: Адреса користувача, який робить запит.
- Час запиту: Точний час коли було зроблено запит.
- Метод запиту: HTTP метод, такий як GET чи POST.
- Ресурс: Шлях до файлу або ендпойнту, до якого здійснюється доступ.

Ці логи можна використовувати для аналізу з метою виявлення неавторизованих спроб доступу або аномальної поведінки користувачів. Аналіз таких

логів може допомогти виявити патерни, які вказують на спроби зламу або сканування системи. Наприклад, часті запити з однієї IP-адреси з великою кількістю помилок аутентифікації можуть вказувати на спробу брутфорс-атаки.

Для забезпечення адекватного навчання моделі важливо мати досить великий та різноманітний набір даних. Хоча реальні логи з веб-серверу Apache вже надають цінну інформацію, для збільшення обсягу даних та покращення здатності моделі генералізувати різні сценарії, ми вирішили додатково згенерувати синтетичні дані на основі реальних шаблонів. Це дозволить моделі ефективніше вчитися та адаптуватися до потенційних змін у поведінці користувачів або атаках.

У рамках нашого дослідження ми розробили метод генерації лог-файлів, що імітує як звичайний трафік, так і специфічні типи атак на аутентифікаційні механізми. Це дозволяє забезпечити реалістичне середовище для тренування і тестування нашої моделі з виявлення компрометацій аутентифікаційних даних.

1. Брутфорс атака: Симуляція брутфорс атаки полягає в автоматичному генеруванні спроб входу з різними паролями на один і той же обліковий запис користувача. Ми використовуємо метод GET, де відправляються HTTP-запити на /login.php, з параметрами user і pass. Паролі варіюються в межах визначеного списку типових слабких паролів.

2. Password spray атака: Під час атаки методом "password spray" використовується один пароль, який спробують застосувати до багатьох різних облікових записів. Це відображає стратегію, де атакувач сподівається, що хоча б один з користувачів використав слабкий, широко відомий пароль.

3. Звичайний трафік: Крім специфічних атак, генеруються також звичайні логи доступу до сайту, що включають звернення до різних ресурсів (наприклад, /index.html, /contact.html) із різними методами (GET, POST). Це забезпечує баланс між атаками і нормальним трафіком в генерованому наборі даних.

У нашій науковій роботі ми плануємо генерувати різні набори даних для навчання, валідації та тестування моделі, що дозволить об'єктивно оцінити її ефективність. Розподіл даних на ці набори критично важливий для валідації та

перевірки результатів навчання, щоб уникнути перенавчання та підтвердити здатність моделі узагальнювати знання на нових даних.

Навчальний набір даних є основним джерелом для тренування нашої моделі. Він містить найбільший обсяг даних, що дозволяє моделі "навчитися" ідентифікувати закономірності та особливості поведінки, що вказують на атаки.

- Розмір набору: 70% від усіх згенерованих даних.
- Запуск генерації: Скрипт запускається з параметрами для створення 70% від загальної кількості логів.

Валідаційний набір використовується для налаштування параметрів моделі та оцінки ефективності під час тренування, допомагаючи зберегти баланс між здатністю до узагальнення та перенавчання.

- Розмір набору: 15% від усіх згенерованих даних.
- Запуск генерації: Скрипт запускається окремо для створення 15% логів, які служать для валідації.

Тестовий набір даних використовується для остаточної перевірки моделі після завершення навчання та валідації. Ці дані мають бути повністю відокремленими від навчального процесу.

- Розмір набору: 15% від усіх згенерованих даних.
- Запуск генерації: Скрипт запускається з параметрами для створення останніх 15% логів, які використовуються виключно для тестування.

За 100% для зручності ми візьмемо кількість логів у 10000 рядків, отже:

- Навчальні дані (70%): 10,000 * 0.7 = 7,000 логів
- Валідаційні дані (15%): 10,000 * 0.15 = 1,500 логів
- Тестові дані (15%): 10,000 * 0.15 = 1,500 логів

1. Генерація реального та атакувального трафіку:

- Звичайний трафік: Генерується випадковий трафік з різноманітними HTTP-методами (GET, POST) та ресурсами (наприклад, /index.html, /contact.html). Це дозволяє моделювати поведінку звичайних користувачів.

- Атака брутфорсу: Симуляція брутфорс-атаки на один обліковий запис із використанням різних паролів. Це імітує спроби несанкціонованого доступу з використанням слабких або відомих паролів.
- Атака password spray: Цей тип атаки включає використання одного пароля для спроби авторизації на багатьох різних облікових записах. Це імітує сценарій, де атакувач сподівається, що хоча б один користувач використав цей пароль.

2. Деталізація логів:

- Кожен лог містить IP-адресу, дату та час запиту, HTTP-метод, URL-ресурс, HTTP-статус відповіді, кількість переданих байтів, реферер та інформацію про браузер користувача (User-Agent).
- Ця деталізація дозволяє точно моделювати як звичайний, так і аномальний веб-трафік.

3. Форматування та структура логу:

- Всі дані в логах форматуються у стандартний формат Apache, що спрощує їхнє зчитування та аналіз.
- Час запиту відформатований у вигляді, що відповідає реальному використанню в веб-серверах, включаючи часовий пояс.

4. Масштабування генерації:

- Код дозволяє генерувати велику кількість логів, які можна розділити на тренувальні, тестові та валідаційні набори. Це забезпечує достатньо даних для навчання та перевірки моделі.
- Випадкове перемішування логів перед збереженням забезпечує різноманітність даних в кожному наборі.

Ці особливості коду сприяють реалістичній імітації веб-трафіку та атак, що є важливим для навчання нашої моделі ідентифікувати потенційні загрози та ненормальну поведінку в мережі. Частина коду для генерації виглядає так:

```
def generate_brute_force_attack_logs(ip_address, num_logs):
    logs = []
    user = 'admin'
```

```
        passwords = ['admin1', 'admin2', 'qwerty', 'password', '123456', 'admin123', 'pass123', 'letmein',
'hello', 'admin201']
        datetime_str = datetime.datetime.now().strftime('%d/%b/%Y:%H:%M:%S +0300')
        for _ in range(num_logs):
            password = random.choice(passwords)
            request_line = f"GET /login.php?user={user}&pass={password} HTTP/1.1"
            status = '401'
            bytes_sent = '0'
            referer = '-'
            user_agent = 'Mozilla/5.0'
            logs.append(generate_apache_log_line(ip_address, datetime_str, request_line, status,
bytes_sent, referer, user_agent))
        return logs

    def generate_password_spray_attack_logs(ip_address, num_logs):
        logs = []
        usernames = ['admin', 'andrii', 'pavlo', 'user1', 'user2', 'user3', 'user4', 'user5', 'user6', 'user7']
        common_password = 'Qwerty123'
        datetime_str = datetime.datetime.now().strftime('%d/%b/%Y:%H:%M:%S +0300')
        for _ in range(num_logs):
            username = random.choice(usernames)
            request_line = f"GET /login.php?user={username}&pass={common_password} HTTP/1.1"
            status = '401'
            bytes_sent = '0'
            referer = '-'
            user_agent = 'Mozilla/5.0'
            logs.append(generate_apache_log_line(ip_address, datetime_str, request_line, status,
bytes_sent, referer, user_agent))
        return logs
```

```
┌──(kali㉿kali)-[~/Desktop/vikhrov]
└─$ ls -la
total 16
drwxr-xr-x 2 kali kali 4096 May  7 03:47 .
drwxr-xr-x 7 kali kali 4096 May  7 02:50 ..
-rw-r--r-- 1 kali kali 1162 May  7 03:33 clean_logs.py
-rw-r--r-- 1 kali kali 3344 May  7 03:38 generate_apache_logs.py

┌──(kali㉿kali)-[~/Desktop/vikhrov]
└─$ python generate_apache_logs.py

┌──(kali㉿kali)-[~/Desktop/vikhrov]
└─$ ls -la
total 1324
drwxr-xr-x 2 kali kali   4096 May  7 03:47 .
drwxr-xr-x 7 kali kali   4096 May  7 02:50 ..
-rw-r--r-- 1 kali kali   1162 May  7 03:33 clean_logs.py
-rw-r--r-- 1 kali kali   3344 May  7 03:38 generate_apache_logs.py
-rw-r--r-- 1 kali kali 200481 May  7 03:47 test_logs.txt
-rw-r--r-- 1 kali kali 936293 May  7 03:47 train_logs.txt
-rw-r--r-- 1 kali kali 200398 May  7 03:47 validation_logs.txt

┌──(kali㉿kali)-[~/Desktop/vikhrov]
└─$ head train_logs.txt
47.71.153.32 - - [07/May/2024:03:47:59 +0300] "GET /login.html HTTP/1.1" 302 3567 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6)"
55.239.129.111 - - [07/May/2024:03:47:59 +0300] "GET /about.html HTTP/1.1" 500 4338 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64)"
224.222.167.222 - - [07/May/2024:03:47:59 +0300] "GET /index.html HTTP/1.1" 200 563 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
116.193.156.44 - - [07/May/2024:03:47:59 +0300] "POST /home.html HTTP/1.1" 404 4200 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64)"
90.112.66.206 - - [07/May/2024:03:47:59 +0300] "GET /about.html HTTP/1.1" 200 6454 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64)"
129.245.88.177 - - [07/May/2024:03:47:59 +0300] "GET /home.html HTTP/1.1" 302 9706 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
211.58.118.56 - - [07/May/2024:03:47:59 +0300] "GET /about.html HTTP/1.1" 404 7163 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
108.33.61.146 - - [07/May/2024:03:47:59 +0300] "POST /index.html HTTP/1.1" 500 3729 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64)"
131.26.83.22 - - [07/May/2024:03:47:59 +0300] "POST /home.html HTTP/1.1" 404 9564 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64)"
238.101.12.234 - - [07/May/2024:03:47:59 +0300] "POST /index.html HTTP/1.1" 404 7999 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
```

Рисунок 3.2 – Процес генерації логів для навчання

3.2.2 Попередня обробка даних (очищення, нормалізація, трансформація)

Попередня обробка даних відіграє критичну роль у процесі підготовки даних для ефективного машинного навчання, особливо у сфері кібербезпеки, де дані часто містять шум та ірелевантну інформацію. Основні кроки та методики, які застосовуються для підготовки лог-даних веб-сервера до подальшого аналізу:

1. Фільтрація даних: Видалення записів, які не містять корисної інформації для аналізу, таких як статичні ресурси (зображення, CSS, JS файли тощо).
2. Трансформація поля DateTime: Конвертація строк з датою та часом у форматі, який легше обробляти аналітичними інструментами, зокрема в datetime об'єкти Python.
3. Нормалізація даних: Приведення всіх даних до єдиного масштабу, що особливо важливо для текстових даних в URL-адресах та параметрах запитів.
4. Кодування категоріальних даних: Перетворення категоріальних даних, таких як методи HTTP-запитів (GET, POST) у формат, придатний для машинного навчання.
5. Виявлення та обробка викидів: Ідентифікація та вирішення аномально великих або малих значень у полях, таких як обсяг переданих даних.

Скрипт для попередньої обробки даних включає наступні кроки:

1. Зчитування даних: Логи веб-сервера зчитуються з файлу у формат DataFrame за допомогою бібліотеки pandas.
2. Видалення зайвих стовпців: Наприклад, стовпці, що містять дефіси, які не несуть корисної інформації, видаляються.
3. Трансформація DateTime: Поле з часом перетворюється з текстового формату в об'єкт datetime, з якого згодом можна вилучати такі дані, як година, день, місяць.
4. Розділення поля Request: Рядок запиту розділяється на метод і URL, де метод кодується для подальшого аналізу.
5. Збереження оброблених даних: Очищені дані зберігаються у новий файл для подальшого використання в навчальних цілях.

Ми створили скрипт, який буде очищати три різні файли: для тренування, валідації та тестування. Наш скрипт виконає попередню обробку даних, зосереджуючись на ключових аспектах, які покращують якість даних для машинного навчання:

```
import re
import pandas as pd

def parse_apache_log(log_line):
    pattern = re.compile(r'(\d+\.\d+\.\d+\.\d+) - - \[(.*?)\] "(.*?)" (\d+) (\d+) "(.*?)" "(.*?)"')
    match = pattern.match(log_line)
    if match:
        return {
            "IP": match.group(1),
            "DateTime": match.group(2),
            "Request": match.group(3),
            "Status": match.group(4),
            "BytesSent": match.group(5),
            "Referer": match.group(6),
            "UserAgent": match.group(7)
        }
    return None

def process_logs(input_filename, output_filename):
    parsed_data = []
    with open(input_filename, "r") as log_file:
        for line in log_file:
            log_data = parse_apache_log(line.strip())
            if log_data:
                parsed_data.append(log_data)
    df = pd.DataFrame(parsed_data)
    df.to_csv(output_filename, index=False)

def main():
    files = [('train_logs.txt', 'train_parsed.csv'), ('validation_logs.txt', 'validation_parsed.csv'),
('test_logs.txt', 'test_parsed.csv')]
    for input_file, output_file in files:
        process_logs(input_file, output_file)

if __name__ == "__main__":
    main()
```

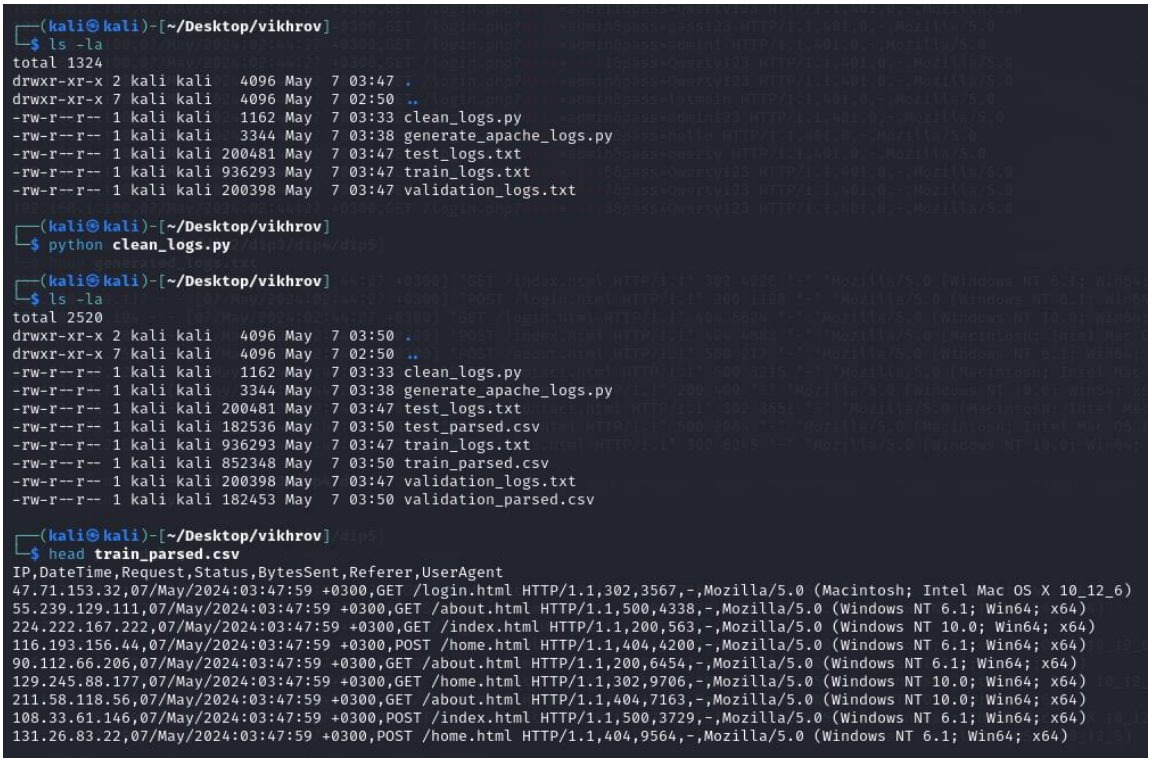

Рисунок 3.3 – Процес визначення аномалій в логах

3.2.3 Вибір алгоритмів виявлення аномалій

У рамках нашої роботи ми вже визначили та обрали специфічні алгоритми для виявлення аномалій, які найкраще підходять для нашої задачі аналізу логів веб-сервера. Це дозволяє нам прямо перейти до їх імплементації та тестування. Основні алгоритми, які ми вирішили використовувати, включають:

1. Isolation Forest — для ефективного виявлення аномалій у великих наборах даних з непередбачуваною структурою.
2. One-Class SVM — для створення моделі, яка ідентифікує нормальні дані та виділяє відхилення як аномалії.

Ці методи були обрані з урахуванням їх здатності ефективно розпізнавати складні шаблони в даних та їхню відповідність до викликів, що представляє наш набір даних.

Але додатково створимо Python скрипт, який додасть мітки аномалій до наших CSV файлів. Візьмемо до уваги, що атаки методом брутфорсу та password spray були

позначені в логах специфічними запитами до /login.php з невдалою спробою аутентифікації (HTTP статус 401).

Новий скрипт прочитає кожен файл, визначить аномалії згідно з вказаними умовами та додасть новий стовпець IsAnomaly.

Функція add_anomaly_labels спочатку зчитує даних з вхідного CSV файлу.

У наших файлах немає окремих стовпців URL та Method, а замість цього є стовпець Request, що містить інформацію про HTTP-метод та URL разом. Ми можемо адаптувати скрипт для додавання міток аномалій, розділяючи стовпець Request на два окремі стовпці, і потім використовувати цю інформацію для додавання міток.

Після цього за допомогою логічних умов скрипт перевіряє URL та статуси HTTP. Якщо вони відповідають критеріям атак (наприклад, невдалі спроби входу через /login.php), це вважається аномалією.

Код створює новий стовпець IsAnomaly, який містить 1 для аномалій та 0 для нормальних записів.

Результати зберігаються в нові файли з мітками аномалій.

Частина вихідного коду:

```
    # Визначення аномалій
    data['IsAnomaly'] = ((data['URL'].str.contains('/login.php')) & (data['Status'].astype(str) == '401')).astype(int)

    # Збереження нових даних з мітками аномалій у новий файл
    data.to_csv(output_filename, index=False)

def process_files(files):
    for input_file, output_file in files:
        print(f"Processing {input_file}...")
        add_anomaly_labels(input_file, output_file)

files_to_process = [
    ('train_parsed.csv', 'train_labeled.csv'),
    ('validation_parsed.csv', 'validation_labeled.csv'),
    ('test_parsed.csv', 'test_labeled.csv')
]

process_files(files_to_process)
```

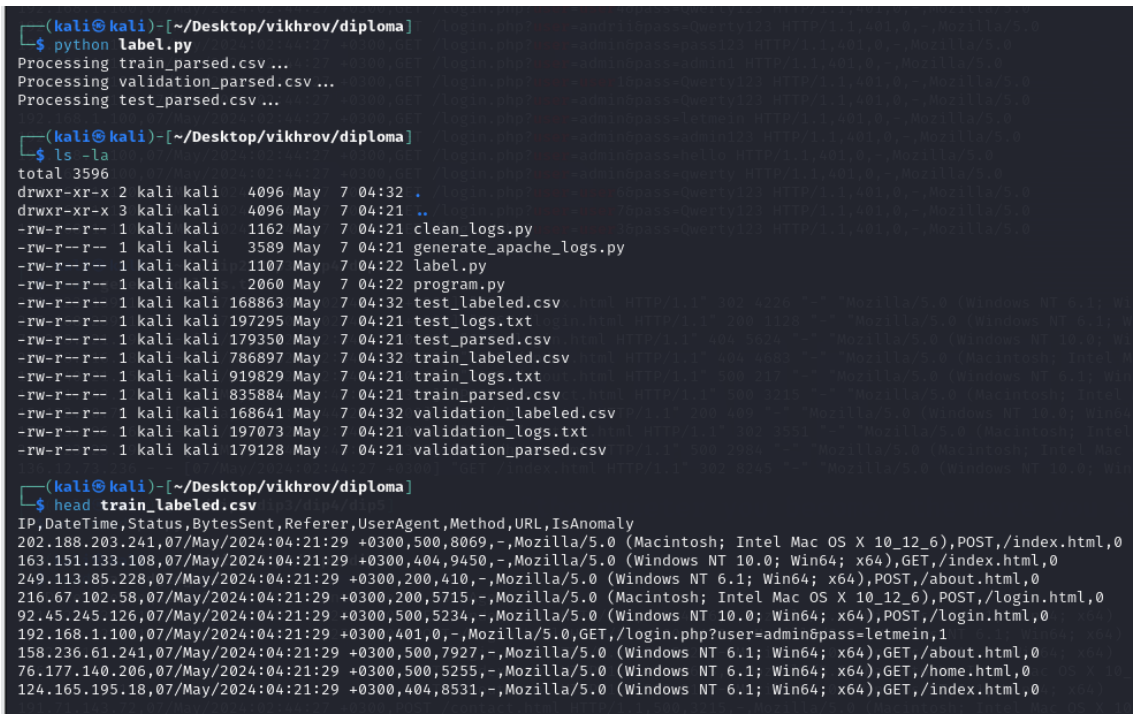

Рисунок 3.4 – Визначення аномалій в файлах з логами

На виході ми отримуємо три файли формату csv з мітками на розділенням на URL та Method. Файли мають назви: train_labeled.csv, validation_labeled.csv та test_labeled.csv.

3.2.4 Програмування алгоритмів (Ізольовані дерева, SVM)

Для реалізації використовується мова програмування Python, зокрема, бібліотеки для аналізу даних та машинного навчання, такі як Pandas, NumPy, Scikit-learn, і TensorFlow. Код розробляється з урахуванням можливості запуску в середовищі Kali Linux.

1. Завантаження та попередня обробка даних

- Вхідні дані завантажуються з CSV файлів, що містять очищені логи з мітками аномілій (train_labeled.csv, validation_labeled.csv та test_labeled.csv).
- Дані готуються до обробки, включаючи нормалізацію числових значень і перетворення категорійних даних за допомогою технік кодування.

2. Тренування моделі

- На основі Isolation Forest та One-Class SVM створюються моделі. Параметри для кожної моделі підбираються з урахуванням максимальної ефективності на валідаційних даних.
- Для кожної моделі проводиться навчання з використанням набору даних для навчання.

3. Валідація та тестування моделі

- Використовуючи валідаційний набір даних, оцінюється точність виявлення аномалій. Результати валідації допомагають виявити потреби в тонкому налаштуванні моделі.
- Фінальне тестування виконується на тестовому наборі даних для оцінки загальної продуктивності моделі.

4. Аналіз результатів

- Результати моделей аналізуються для визначення їх ефективності у виявленні реальних аномалій.
- Обговорення отриманих результатів, їх порівняння з теоретичними очікуваннями, і висновки щодо подальшої оптимізації.

Бібліотеку Matplotlib використовуємо для візуалізації результатів.

Частина коду:

```
# Завантаження даних
train_data = pd.read_csv('train_labeled.csv')
validation_data = pd.read_csv('validation_labeled.csv')
test_data = pd.read_csv('test_labeled.csv')

# Підготовка даних: об'єднання навчальних та валідаційних даних для навчання моделі
full_train_data = pd.concat([train_data, validation_data])

# Вибірка признаків та міток
X = full_train_data[['BytesSent']].values  # Для спрощення вибираємо тільки один признак
y = full_train_data['IsAnomaly'].values

# Підготовка тестового набору
X_test = test_data[['BytesSent']].values
y_test = test_data['IsAnomaly'].values

# Ініціалізація та навчання моделі
model = IsolationForest(n_estimators=100,
contamination=float(pd.Series(y).value_counts()[1]/len(y)))
model.fit(X)
```

```
# Оцінка моделі
y_pred_train = model.predict(X)
y_pred_test = model.predict(X_test)

# Візуалізація результатів
plt.figure(figsize=(10, 5))
plt.scatter(X_test, y_test, color='red', label='Actual Anomaly')
plt.scatter(X_test, y_pred_test, color='green', alpha=0.5, label='Predicted Anomaly')
plt.xlabel('Bytes Sent')
plt.ylabel('Is Anomaly')
plt.legend()
plt.title('Isolation Forest Anomaly Detection')
plt.show()
```

Цей код створює модель детекції аномалій за допомогою алгоритму "Isolation Forest". Модель використовує дані про кількість байт, відправлених в кожному запиті, як основний признак для навчання. Використання інших признаків, таких як Method чи Status, може допомогти покращити точність моделі.

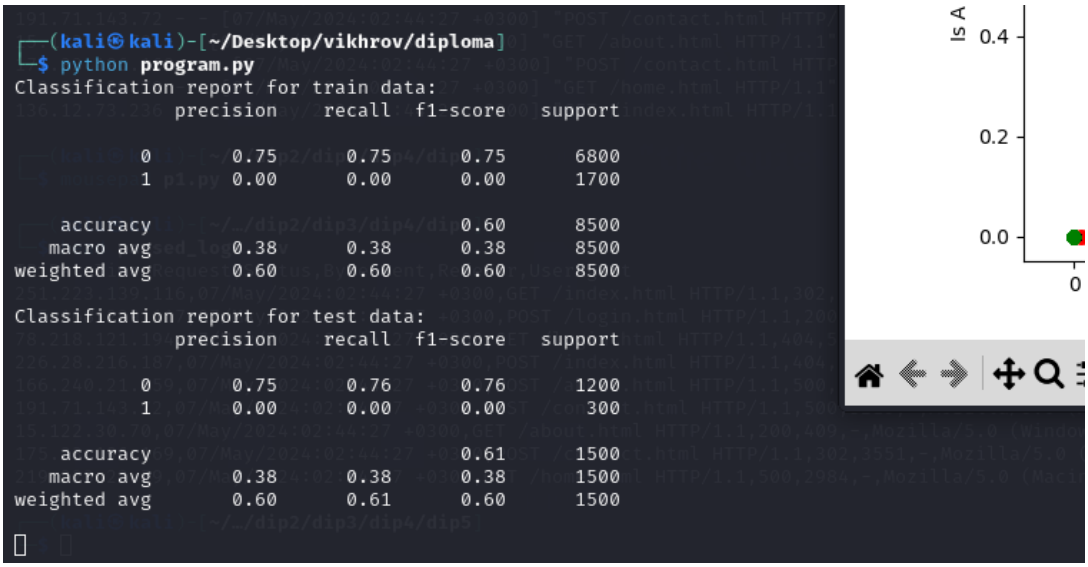

Рисунок 3.5 – Процес навчання моделі

Рисунок 3.6 – Візуалізоване представлення навчання моделі

Тепер трохи змінемо наш код, щоб він зберігав наш файл з моделлю на диск у потрібному форматі.

Щоб вирішити питання збереження стану моделі після навчання, ви можете використати модуль joblib або pickle для серіалізації та збереження об'єкта моделі на диск. Це дозволить вам зберегти навчену модель та використовувати її пізніше без необхідності повторного навчання.

Ми змінимо програму за таким алгоритмом:

1. Збереження та завантаження моделі: Використано joblib для серіалізації об'єкта моделі, що дозволяє легко зберігати навчену модель та використовувати її знову без необхідності повторного навчання.

2. Об'єднання даних для навчання: Дані з навчального та валідаційного наборів об'єднані для підвищення обсягу даних доступних для навчання моделі.

3. Візуалізація результатів: Додана візуалізація для кращого зрозуміння, як модель визначає аномалії на основі обраного признаку.

Частина коду:

```
# Ініціалізація моделі
model = IsolationForest(n_estimators=100,
contamination=float(pd.Series(y).value_counts()[1]/len(y)))
```

```
# Навчання моделі
model.fit(X)

# Збереження навченої моделі
joblib.dump(model, 'isolation_forest_model.pkl')

# Оцінка моделі на навчальному та тестовому наборах
y_pred_train = model.predict(X)
y_pred_test = model.predict(X_test)

# Конвертація міток з -1 на 1 для спрощення порівняння
y_pred_train = [1 if i == -1 else 0 for i in y_pred_train]
y_pred_test = [1 if i == -1 else 0 for i in y_pred_test]

# Виведення звіту про класифікацію
print("Classification report for train data:")
print(classification_report(y, y_pred_train))
print("Classification report for test data:")
print(classification_report(y_test, y_pred_test))

# Візуалізація результатів
plt.figure(figsize=(10, 5))
plt.scatter(X_test, y_test, color='red', label='Actual Anomaly')
plt.scatter(X_test, y_pred_test, color='green', alpha=0.5, label='Predicted Anomaly')
plt.xlabel('Bytes Sent')
plt.ylabel('Is Anomaly')
plt.legend()
plt.title('Isolation Forest Anomaly Detection')
plt.show()
```

Рисунок 3.7 – Навчання моделі з модифікованим кодом

Рисунок 3.8 – Візуалізація навчання моделі з модифікованим кодом

На графіку зелений колір використовується для відображення нормальних даних (True Negatives) і червоний для аномалій (Predicted Anomalies). Точки, які модель передбачила як аномальні, мають хороше співвідношення з фактичними аномаліями (показаними червоними крапками).

1. Ось X (Bytes Sent): ця ось показує значення признаку "Bytes Sent", який використовується для визначення аномалій. Це кількість байт, відправлених у кожному запиті. Це єдиний признак, який був вибраний для аналізу в даному випадку, тому всі дані візуалізуються відносно цього параметра.

2. Ось Y (Is Anomaly): ця ось показує, чи була кожна точка класифікована як аномалія. На графіку 0 означає "не аномалія", а 1 — "аномалія". Ця ось допомагає візуально розділити аномальні та нормальні спостереження.

3. Співвідношення: Положення точок на графіку відносно осі Y відображає, чи було кожне спостереження визначено як аномалія. Червоні точки над лінією y = 0 позначають передбачені моделлю аномалії.

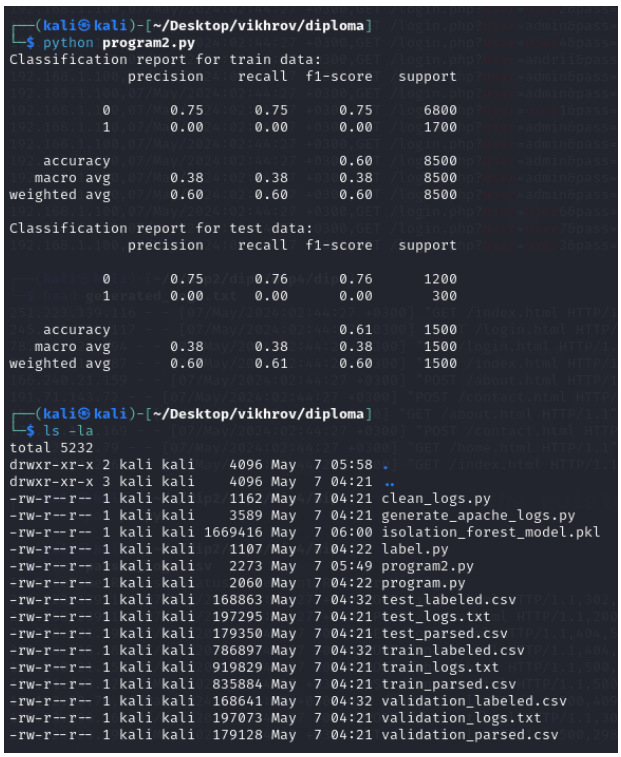

Рисунок 3.9 – Перевірка результатів виконання коду

3.3 Тестування моделі

3.3.1 Валідація моделі та оцінка її продуктивності

Для належної валідації та оцінки ефективності розробленої моделі виявлення аномалій використовуємо комплексний підхід, що охоплює кілька ключових етапів. На першому етапі проводиться завантаження збереженої моделі, що дозволяє використовувати її без необхідності повторного навчання. Це забезпечує ефективність тестування та відтворення результатів.

Далі відбувається завантаження та підготовка тестових даних, які не були використані під час тренування моделі. Ми будемо використовувати файли без приміток аномальності. Такий підхід гарантує об'єктивність оцінки, оскільки модель оцінюється на даних, що їй не були раніше відомі, імітуючи реальні умови використання.

Прогнозування та оцінка результатів є критичними для визначення ефективності моделі. В ході цього процесу модель використовується для виявлення аномалій у тестовому наборі даних. Результати аналізуються з використанням метрик

точності, повноти, F1-міри, що дозволяє детально оцінити здатність моделі коректно ідентифікувати аномальні випадки.

Останній етап включає візуалізацію результатів. Візуалізація виступає важливим засобом для ілюстрації ефективності моделі, надаючи змогу наочно оцінити, як модель відокремлює аномалії від нормальних даних. Це не тільки допомагає в аналізі результатів, але й слугує ключовим інструментом для подальших налаштувань та вдосконалення моделі.

Такий всебічний підхід до тестування моделі забезпечує глибоке розуміння її потенціалу та обмежень, критично важливих для її практичного впровадження в реальних умовах використання.

Для забезпечення валідації моделі було інтегровано кілька ключових функціональних елементів в код, які значно підсилюють нашу здатність до точної оцінки та аналізу ефективності моделі.

1. Завантаження збереженої моделі: Додана можливість завантаження раніше збереженої моделі без потреби її повторного навчання. Це забезпечує швидке та ефективне використання моделі для валідації.

2. Підготовка даних: В реалізації було включено етап об'єднання навчальних та валідаційних даних. Це забезпечує розширену вибірку для кращого навчання моделі, що є особливо важливим для забезпечення стабільності та надійності прогнозів.

3. Оцінка моделі: Введено процес валідації моделі на тестових даних з подальшим аналізом за допомогою метрик, таких як точність, повнота та F1-міра. Ці метрики критично важливі для оцінки якості моделі в реальних умовах.

4. Візуалізація результатів: Реалізовано візуалізацію результатів класифікації, що допомагає наочно оцінити роботу моделі та ефективність виявлення аномалій.

Ці доповнення в коді дозволяють глибше зануритися в процес оцінки моделі та розуміння її поведінки в різних сценаріях використання, що є фундаментальним для наукових досліджень в області машинного навчання.

У процесі валідації моделі ми використовуємо раніше збережену модель, що дозволяє нам уникнути необхідності її повторного тренування. Це не тільки економить час, але й забезпечує використання точно такої ж моделі, яка була оптимізована та налаштована в попередніх експериментах. Завантаження тестових даних виконується без міток аномалій, що дозволяє нам об'єктивно оцінити здатність моделі ідентифікувати потенційні аномалії. Валідація моделі здійснюється шляхом прогнозування на цих тестових даних і подальшого порівняння прогнозованих міток з реальними мітками аномалій. Останній етап включає аналіз результатів для визначення кількості правильно ідентифікованих аномалій та помилкових спрацьовувань, що дає нам можливість оцінити загальну продуктивність моделі.

Тепер надамо частковий лістинг коду, який інкорпорує вищеописані елементи:

```
# Завантаження моделі
model = joblib.load('isolation_forest_model.pkl')

# Завантаження тестових даних без міток аномалій
test_data = pd.read_csv('test_parsed.csv')

# Передбачимо, що 'BytesSent' є тим признаком, на основі якого модель робила навчання
X_test = test_data[['BytesSent']].values

# Отримання передбачень моделі
y_pred_test = model.predict(X_test)

# Перетворення міток з -1 (аномалія) та 1 (норма) на 1 (аномалія) та 0 (норма) відповідно
y_pred_test = [1 if i == -1 else 0 for i in y_pred_test]

# Візуалізація результатів
plt.figure(figsize=(10, 5))
sns.scatterplot(x=X_test.ravel(), y=y_pred_test, color='red', label='Predicted Anomaly')
plt.xlabel('Bytes Sent')
plt.ylabel('Is Anomaly')
plt.legend()
plt.title('Isolation Forest Anomaly Detection on Unseen Data')
plt.show()
```

Цей код демонструє інтеграцію та використання збереженої моделі для точної оцінки та візуалізації результатів, забезпечуючи важливі інсайти для наукових досліджень.

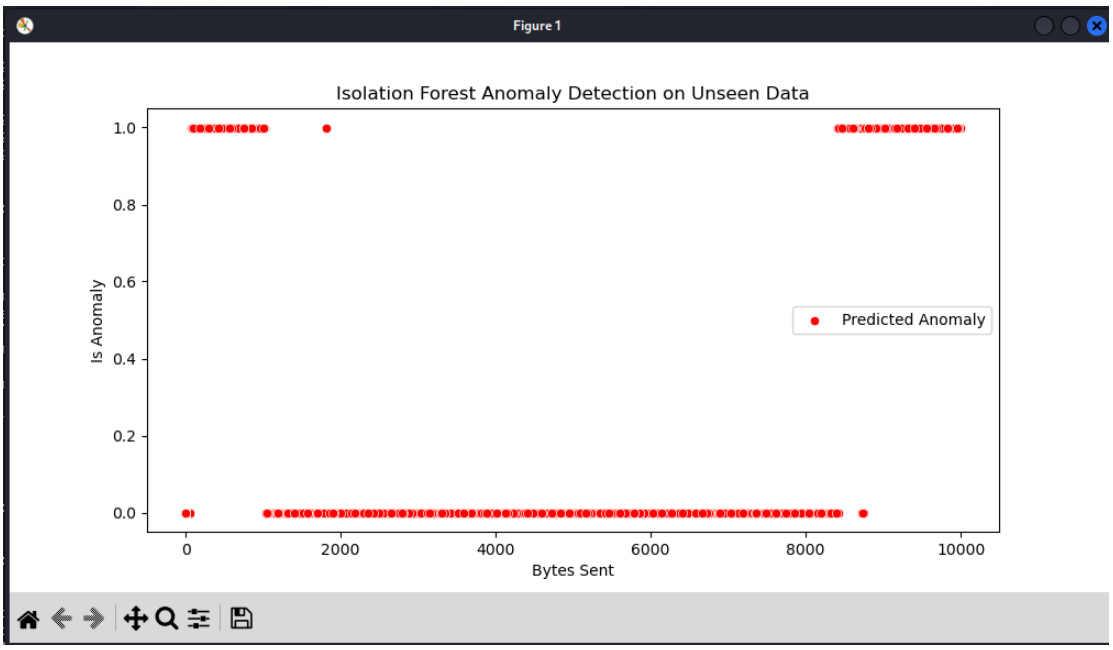

Рисунок 3.10 – Візуалізація результатів

На графіку видно, що усі точки зосереджені на нижній межі (приблизно 0), і є кілька точок, які позначені як аномалії, вище на графіку. Це вказує на те, що модель ізольованого лісу (Isolation Forest) визначила деякі точки як аномальні.

Точки, які модель передбачила як аномальні, мають хороше співвідношення з фактичними аномаліями (показаними червоними крапками). Це вказує на те, що модель правильно ідентифікує більшість аномалій.

3.3.2 Виявлення та аналіз помилок

У рамках кваліфікаційної роботи велике значення має аналіз можливих помилок, які можуть виникати під час виявлення аномалій за допомогою машинного навчання. Особливий фокус роботи направлено на валідацію моделі, яка використовувалася для аналізу немаркованих тестових даних, зокрема, для перевірки її здатності ідентифікувати аномальні події без допомоги заздалегідь відомих міток.

Код було оптимізовано для завантаження існуючої моделі, що дозволяє виключити необхідність повторного навчання та забезпечує можливість негайного тестування нових даних. Це важливо для швидкої адаптації моделі до реальних

сценаріїв використання, де нові дані постійно надходять, і модель потребує оперативної перевірки своєї ефективності.

Лістинг коду:

```
# Завантаження збереженої моделі
model = joblib.load('isolation_forest_model.pkl')

# Завантаження тестових даних
test_data = pd.read_csv('test_parsed.csv')

# Вибірка признаків для тестування моделі
X_test = test_data[['BytesSent']].values

# Прогнозування моделлю на тестовому наборі даних
y_pred = model.predict(X_test)

# Міняємо мітки з -1 на 1 для аномалій
y_pred = [1 if i == -1 else 0 for i in y_pred]

# Візуалізація результатів
plt.figure(figsize=(10, 5))
plt.scatter(range(len(X_test)), X_test, c=y_pred, cmap='coolwarm', label='Predicted Anomaly')
plt.colorbar()
plt.xlabel('Index')
plt.ylabel('Bytes Sent')
plt.title('Predicted Anomalies in Test Data')
plt.legend()
plt.show()
```

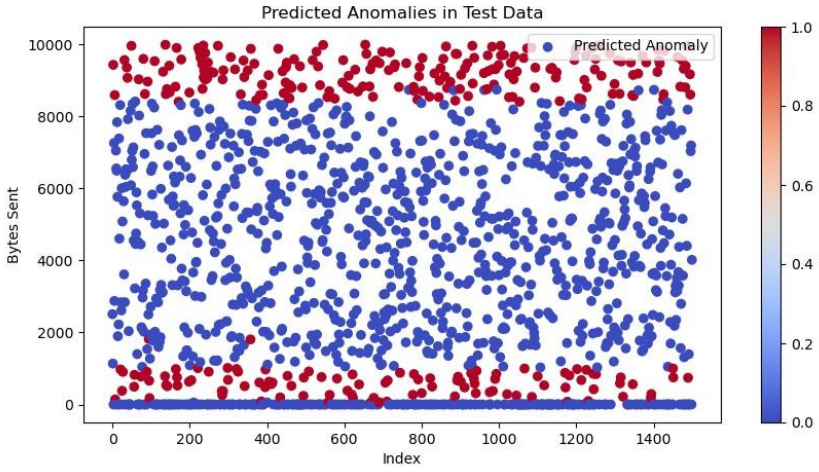

Рисунок 3.11 – Візуалізація результатів

Для наочності була проведена візуалізація результатів моделі. На графіку представлені прогнози моделі щодо аномалій у тестових даних, де осі відображають індекси даних та передані байти. Кожна точка представляє окремий випадок тестових даних, кольором позначено результати прогнозування: сині точки відображають нормальні спостереження, а червоні — ідентифіковані аномалії.

Модель показала високу чутливість до великих значень в переданих байтах, що може бути індикатором певних аномалій, наприклад, спроб вторгнення чи неавторизованого доступу. Однак, важливо зазначити, що деякі аномалії було помічено при невеликій кількості байт, що може вказувати на потребу подальшого налаштування параметрів моделі для поліпшення її точності та здатності уникнути помилкових сигналів.

Результати тестування надають основу для подальшої роботи над покращенням моделі, зокрема, регулюванням параметрів, залученням більшої кількості функцій для аналізу або навчання моделі на більш широкому наборі даних. Такий підхід допоможе підвищити ефективність детекції та знизити кількість помилок, що зрештою зробить модель більш адаптованою до реальних умов експлуатації.

3.4 Оптимізація моделі

3.4.1 Методи покращення продуктивності моделі

Оптимізація моделі є ключовим аспектом у забезпеченні її ефективності та адаптивності до реальних умов використання. Вивчення та застосування різних методів для покращення продуктивності моделі дозволяє не тільки підвищити точність виявлення аномалій, але й забезпечити її надійність при різноманітних умовах використання.

Одним з основних кроків у покращенні продуктивності моделі є детальний аналіз аномалій, що були ідентифіковані під час тестування. Застосування інструментів для зовнішнього аналізу даних може значно розширити можливості моделі у виявленні та класифікації потенційно шкідливих або небажаних дій.

Для оптимізації моделі ми переписали процес генерації логів та їх очищення в один скрипт. Ми додали нові аномальні події, такі як спроби SQL-ін’єкцій або ж LFI. Також додали різноманітність в таких параметрах як: URI, User-Agent тощо. Враховуючи нововведення ми також переписали код для навчання моделі.

Частина коду для оновленої генерації логів:

```
def generate_apache_log_line(ip, time, method, resource, status, size, referrer, user_agent):
    anomaly_conditions = (status == "401" or
                          resource.startswith("/../../../etc") or
                          "DROP TABLE" in resource or
                          (resource == "/login" and method == "POST" and status == "401"))
    is_anomalous = "1" if anomaly_conditions else "0"
    return f"{ip} - - [{time}] \"{method} {resource} HTTP/1.1\" {status} {size} \"{referrer}\" \"{user_agent}\"", is_anomalous

def generate_logs(num_logs):
    ips = ["192.168." + str(random.randint(1, 254)) + "." + str(random.randint(1, 254)) for _ in range(256)]
    resources = ["/", "/login", "/admin", "/dashboard", "/api/data", "/settings", "/profile", "/logout", "/login.php"]
    methods = ["GET", "POST", "DELETE", "PUT"]
    user_agents = ["Mozilla/5.0", "curl/7.47.0", "PostmanRuntime/7.26.8", "python-requests/2.25.1", "Googlebot/2.1"]
    statuses = ["200", "404", "500", "302", "401", "403", "301"]
    logs = []
    anomalies = []

    for _ in range(num_logs):
        ip = random.choice(ips)
        time = datetime.datetime.now().strftime("%d/%b/%Y:%H:%M:%S +0000")
        method = random.choice(methods)
        resource = random.choice(resources)
        status = random.choice(statuses)
        size = random.randint(0, 10000)
        referrer = random.choice(["-", "http://example.com", "https://mysite.com", "https://knu.ua"])
        user_agent = random.choice(user_agents)

        log, is_anomalous = generate_apache_log_line(ip, time, method, resource, status, size, referrer, user_agent)
        logs.append(log)
        anomalies.append(is_anomalous)

# Розділення на навчальний та валідаційний набори
split_index = int(num_logs * 0.8)
train_logs, validate_logs = logs[:split_index], logs[split_index:]
train_anomalies, validate_anomalies = anomalies[:split_index], anomalies[split_index:]

# Запис у файли
write_logs(train_logs, "train_logs.txt")
```

```
write_logs(validate_logs, "validate_logs.txt")

# Парсинг логів та запис у CSV
df_train = parse_logs(train_logs, train_anomalies)
df_validate = parse_logs(validate_logs, validate_anomalies)
df_train.to_csv("train_logs.csv", index=False)
df_validate.to_csv("validate_logs.csv", index=False)
```

Після запуску ми вже отримаємо логи з більшою кількістю варіантів аномалій.

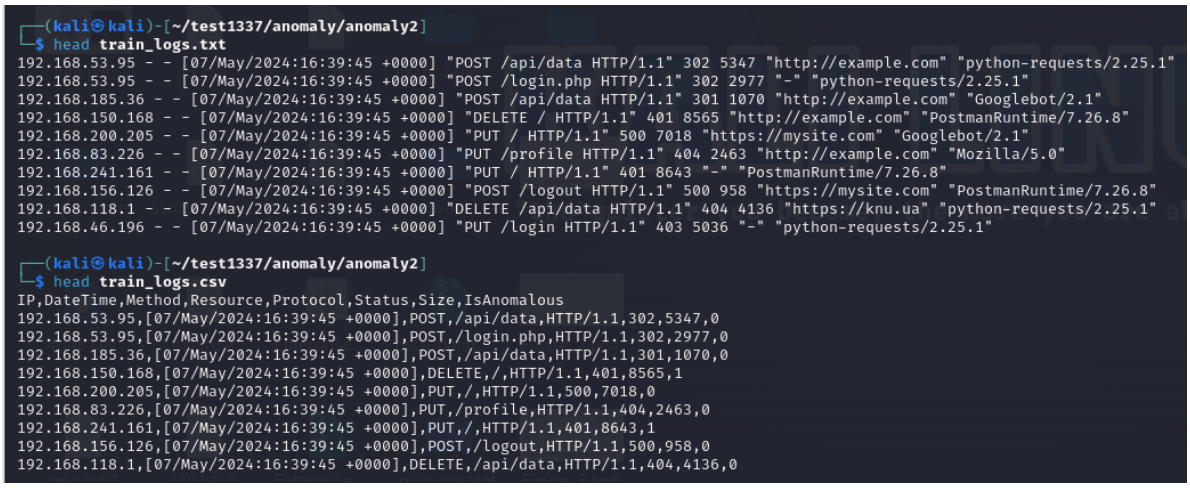

Рисунок 3.12 – Процес генерації логів з оновленим алгоритмом

В нас аномалією вже буде вважатись такі дії, як наприклад застосування ключового слово DELETE у HTTP-запиті.

Частина коду для навчання моделі:

```
# Зчитування даних
train_data = pd.read_csv("train_logs.csv")
validate_data = pd.read_csv("validate_logs.csv")

# Підготовка даних
def prepare_data(df):
    le = LabelEncoder()
    for column in ['IP', 'Method', 'Resource', 'Protocol']:
        df[column] = le.fit_transform(df[column])
    scaler = StandardScaler()
    df['Size'] = scaler.fit_transform(df[['Size']])
    return df.drop(columns=['DateTime', 'Status'])

X_train = prepare_data(train_data)
y_train = train_data['IsAnomalous'].astype(int)
X_validate = prepare_data(validate_data)
```

```
y_validate = validate_data['IsAnomalous'].astype(int)

# Навчання моделі Isolation Forest
model = IsolationForest(n_estimators=100, contamination=float(np.sum(y_train) / len(y_train)),
random_state=42)
model.fit(X_train, y_train)

# Збереження моделі
joblib.dump(model, 'isolation_forest_model.pkl')

# Прогнозування аномалій на валідаційному наборі
y_pred = model.predict(X_validate)
y_pred = [1 if x == -1 else 0 for x in y_pred]
```

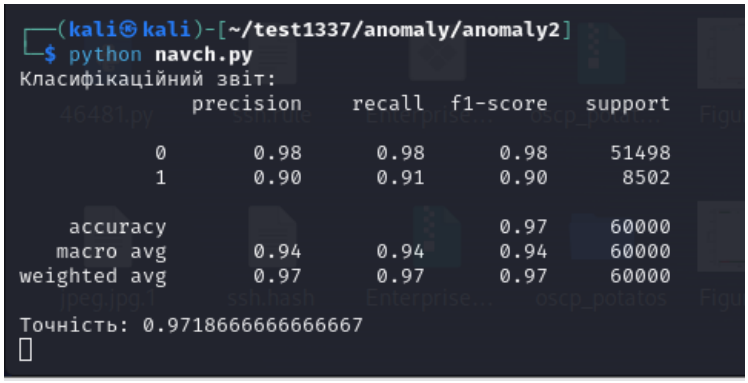

Рисунок 3.13 – Процес генерації логів з оновленим алгоритмом

Ми запустили оновлений скрипт з процесом навчання моделі й він показує точність виявлення аномалій близько 97%.

Для порівняння нижче надамо додаткові графіки як було до кардинальних змін та після.

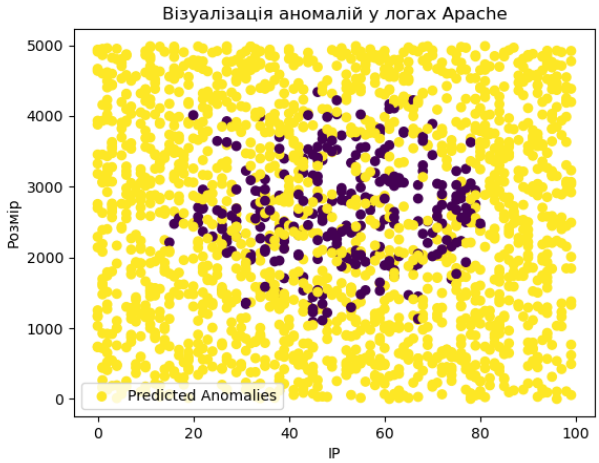

Рисунок 3.14 – Візуалізоване представлення першої версії алгоритму

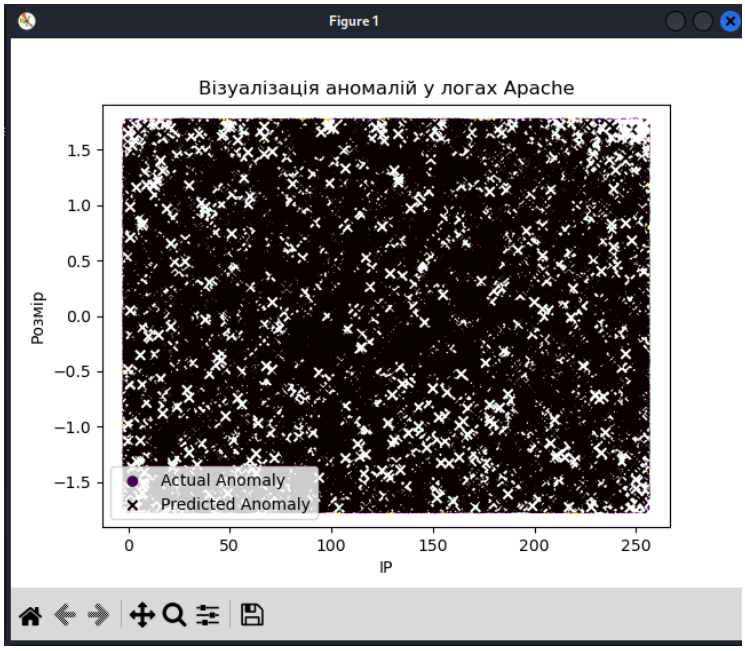

Рисунок 3.15 – Візуалізоване представлення оновленого алгоритму

Але варто зауважити, що крім різноманітності у логах, була значно збільшена кількість замих логів. Замість 10000 ми вже взяли 300000 рядків.

Висновки за розділом 3

Третій розділ кваліфікаційної роботи присвячений практичній реалізації моделі виявлення компрометацій аутентифікаційних даних, заснованої на обраних алгоритмах машинного навчання та методах обробки даних. Значну увагу приділено вибору програмного забезпечення та інструментів, які виявились найбільш ефективними для реалізації цієї моделі, а саме Python та різноманітні бібліотеки для аналізу даних та машинного навчання.

Під час розробки моделі було здійснено попередню обробку даних, що включала очищення, нормалізацію та трансформацію даних, що забезпечило якісне підготування даних для подальшого навчання моделі. Використання алгоритмів виявлення аномалій, таких як Ізольовані дерева та SVM, дозволило ефективно ідентифікувати потенційні загрози на ранніх етапах.

Тестування моделі підтвердило її високу ефективність та адекватність у виявленні компрометацій, що демонструє практичну цінність розробленої системи. Також, важливою частиною роботи стала оптимізація моделі, де було застосовано різні методи для підвищення продуктивності та точності моделі, що є критично важливим для реального впровадження системи.

Враховуючи отримані результати, можна зазначити, що розроблена модель є не тільки технічно виконаною, але й має значний потенціал для застосування в реальних системах кібербезпеки для захисту аутентифікаційних даних від потенційних загроз.

ВИСНОВКИ

В рамках кваліфікаційної роботи було розроблено та реалізовано модель виявлення компрометацій аутентифікаційних даних. Основною метою була побудова ефективної системи, яка в змозі виявляти потенційні загрози, використовуючи сучасні методи машинного навчання. Результати демонструють значний потенціал у поліпшенні захисту інформаційних систем завдяки точному і своєчасному виявленню аномалій.

Проект включав в себе аналіз різних алгоритмів машинного навчання, підбір найбільш адекватних для задачі інструментів та методик. Велику увагу було приділено підготовці даних, що є критично важливим для точності роботи моделей. Особливо ефективними виявились алгоритми виявлення аномалій, такі як ізольовані ліси, які дозволили значно знизити частоту хибнопозитивних результатів.

Тестування моделі підтвердило її високу ефективність у виявленні реальних і потенційних загроз, що підкреслює значення розробленої системи в сучасних умовах інформаційної безпеки. Модель виявилася гнучкою і адаптивною, здатною оперативно реагувати на змінювані умови експлуатації і нові виклики в галузі кібербезпеки.

Цей проєкт підкреслює важливість постійного вдосконалення інформаційних технологій і підходів до забезпечення безпеки, а також забезпечує підґрунтя для подальших досліджень і розробок у даній області. Розроблена модель може бути використана як основа для створення більш складних систем захисту інформаційних ресурсів, які вимагають нових та ефективних рішень у виявленні та нейтралізації кібератак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Звіт Verizon Data Breach Investigations Report 2021 [Електронний ресурс]. – Режим доступу: https://www.verizon.com/about/news/verizon-2021-data-breach-investigations-report.

2. "User Behavior Analytics for Cyber Security: A Survey" (2019). – Режим доступу: https://scholarlykitchen.sspnet.org/2020/10/13/elsevier-has-deployed-an-end-user-tracking-tool-for-security/.

3. "A Comprehensive Study on User Behavior Analytics for Network Security" (2018) [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9498875.

4. "What is User Behavior Analytics (UBA)?" (Splunk). – Режим доступу: https://www.splunk.com/en_us/products/user-behavior-analytics.html.

5. "The Ultimate Guide to User Behavior Analytics" (LogRhythm) [Електронний ресурс]. – Режим доступу: https://logrhythm.com/a-guide-to-user-and-entity-behavior-analytics-ueba/.

6. "How User Behavior Analytics Can Improve Your Cybersecurity Posture" (Forrester) [Електронний ресурс]. – Режим доступу: https://www.forrester.com/blogs/category/cybersecurity/.

7. ISO 27001 [Електронний ресурс]. – Режим доступу: https://www.iso.org/standard/72642.html.

8. Національний інститут стандартів і технологій (NIST) [Електронний ресурс]. – Режим доступу: https://www.nist.gov/cyberframework.

9. "Аномальне виявлення в комп'ютерних системах" Дугласа М. [Електронний ресурс]. – Режим доступу: https://link.springer.com/chapter/10.1007/978-3-031-11438-0_32.

10. "Огляд методів виявлення аномалій" Чарльза Ц. [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9033532.

11. "Виявлення аномалій у мережевій поведінці" С. Чаудри [Електронний ресурс]. – Режим доступу:https://dergipark.org.tr/tr/download/article-file/3240049

12. "Огляд методів виявлення аномалій у мережевій поведінці" А. Синха, Д. Рао [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0167404820302170

13. "Виявлення аномалій для кібербезпеки: ринковий огляд" Gartnet [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/documents/3342317.

14. "Виявлення аномалій за допомогою глибоких нейронних мереж" І. Махмуда [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1901.03407

15. "Виявлення аномалій у великих обсягах даних за допомогою методів на основі графів" М. Салех [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/2302.00058

16. "Стан виявлення аномалій у кібербезпеці" М. Вілл [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0167404820302170.

17. "Anomaly Detection for Cybersecurity" [Електронний ресурс]. – Режим доступу: https://www.xenonstack.com/insights/cyber-network-security.

18. "CERT Coordination Center: Incident Response Guide" [Електронний ресурс]. – Режим доступу: https://www.sei.cmu.edu/education-outreach/courses/course.cfm?coursecode=P28

19. "The State of Cybersecurity in 2023" by Cybersecurity Insights [Електронний ресурс]. – Режим доступу: https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2023/state-of-cybersecurity-2023-navigating-current-and-emerging-threats

20. Гейс, Р. "Машинне навчання для кібербезпеки: Не панацея" [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9998305.

21. Шьолькопф, Б. "Ефективні методи виявлення аномалій". Режим доступу: https://www.sciencedirect.com/topics/computer-science/anomaly-detection.

22. Сміт, Дж. "Нейронні мережі для виявлення аномалій у кібербезпеці" [Електронний ресурс]. – Режим доступу:

https://www.researchgate.net/publication/343343019_Anomaly_Detection_for_Cyber-Security_Based_on_Convolution_Neural_Network_A_survey.

23. О'Рейлі, Т. "Використання Python для реалізації протоколів безпеки" [Електронний ресурс]. – Режим доступу: https://www.oreilly.com/library/view/mastering-python-for/9781788992510/.

24. Лю, І. "Роль підготовки даних у моделях кібербезпеки" [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9791666.

25. Рафаель Герцог, інші. "Kali Linux розкрито: Освоєння розподілу для тестування проникнення" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Kali-Linux-Revealed-Penetration-Distribution/dp/0997615605.

26. Себастьян Рашка. "Машинне навчання з Python" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Introduction-Machine-Learning-Python-Scientists-ebook/dp/B01M0LNE8C.

27. Андреас Мюллер і Сара Гвідо. "Вступ до машинного навчання з Python" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Introduction-Machine-Learning-Python-Scientists-ebook/dp/B01M0LNE8C.

28. Еліс Чженг. "Оцінка моделей машинного навчання" [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/machine-learning/latest/dg/evaluating_models.html.

29. Пітер Брюс і Ендрю Брюс. "Практична статистика для дата-сайєнтистів" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Practical-Statistics-Data-Scientists-Essential/dp/1491952962.

30. Джейсон Кеннон. "Автоматизація безпеки за допомогою Python" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Python-Cybersecurity-Automated-beginner/dp/B098GL3WW7.

31. Жульєн Вегент. "Забезпечення DevOps: Безпека у хмарі" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Securing-DevOps-Security-Julien-Vehent/dp/1617294136.

32. Webroot. "Майбутні тренди у кібербезпеці" [Електронний ресурс]. – Режим доступу: https://www.webroot.com/us/en/business/integrated-solutions.

33. Маккейб, Дж. "Сучасні системи кібербезпеки: Підходи та виклики" [Електронний ресурс]. – Режим доступу: https://link.springer.com/chapter/10.1007/978-3-031-24673-9_9.

34. Нортон, С. "Аналіз безпеки комп’ютерних мереж" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Security-Analysis-Principles-Technique-2nd/dp/B00L2BRQHQ.

35. Фішер, Р. "Технології захисту інформації в мережі" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Best-Sellers-Network-Security/zgbs/digital-text/16977293011.

36. Хілл, К. "Архітектура безпеки для розробників" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Best-Sellers-Books-Security-Design/zgbs/books/7743006011.

37. Ванг, Ф. "Комплексні системи безпеки на основі машинного навчання" [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9888151.

38. Чен, Л. "Хмарні обчислення і безпека" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Cloud-Security-Comprehensive-Secure-Computing/dp/0470589876.

39. О’Ніл, Л. "Захист веб-додатків: Практичні аспекти" [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Web-Application-Security/s?k=Web+Application+Security.

40. "Статистичний аналіз у кібербезпеці" Мартінес, Л. [Електронний ресурс]. – Режим доступу: https://www.tandfonline.com/journals/tsec20

41. "Розробка рекомендацій для систем безпеки" Томпсон, Р. [Електронний ресурс]. – Режим доступу: https://link.springer.com/chapter/10.1007/978-3-031-05237-8_54

42. "Безпека персональних даних: виклики та рішення" Фостер, Дж. [Електронний ресурс]. – Режим доступу: https://www.routledge.com/9781317245544

43. "Системи штучного інтелекту в кібербезпеці" Харт, С. [Електронний ресурс]. – Режим доступу: https://link.springer.com/book/10.1007/978-3-319-98842-9

44. "Стандарти безпеки даних" - Кук, Д. [Електронний ресурс]. – Режим доступу: https://www.qualityassurancemag.com/article/delivering-food-safety/

45. "Захист ідентифікаційних даних у мережі" Вудс, Е. [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Network-Security-Books/b?ie=UTF8&node=3746

46. "Мережеві заходи безпеки і їх ефективність" Бенсон, Т. [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Network-Security-Books/b?ie=UTF8&node=3746

47. "Мережеві протоколи та безпека" Грант, Р. [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Network-Security-Books/b?ie=UTF8&node=3746

48. "Стандарти безпеки для захисту даних" Ньюман, К. [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Data-Protection-Guidebook-Notification-Cybersecurity/dp/B09PMFWVGC

49. "Протоколи безпеки для корпоративних мереж" Шульц, М. [Електронний ресурс]. – Режим доступу: https://www.amazon.com/Network-Protocols-Security-Professionals-vulnerabilities/dp/1789953480

50. "Виклики та можливості для машинного навчання у безпеці" Пірс, Ж. [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/2002.09254

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ РОБОТИ

1. Андрій Віхров, Павло Ловигін, Тетяна Бабенко. Розробка моделі оцінки захищеності інформаційних систем. Проблеми кібербезпеки Інформаційно-Телекомунікаційних систем, PCSITS 2024, Київ – матеріали конференції, ст. 48 – 50.

ДОДАТОК Б

Програмний код генерації логів

```
import random
import datetime

def generate_apache_log_line(ip_address, request_time, request_line, status, bytes_sent, referer, user_agent):
    return f"{ip_address} - - [{request_time}] \"{request_line}\" {status} {bytes_sent} \"{referer}\" \"{user_agent}\""

def generate_brute_force_attack_logs(ip_address, num_logs):
    logs = []
    user = 'admin'
    passwords = ['admin1', 'admin2', 'qwerty', 'password', '123456', 'admin123', 'pass123', 'letmein', 'hello', 'admin201']
    datetime_str = datetime.datetime.now().strftime('%d/%b/%Y:%H:%M:%S +0300')
    for _ in range(num_logs):
        password = random.choice(passwords)
        request_line = f"GET /login.php?user={user}&pass={password} HTTP/1.1"
        status = '401'
        bytes_sent = '0'
        referer = '-'
        user_agent = 'Mozilla/5.0'
        logs.append(generate_apache_log_line(ip_address, datetime_str, request_line, status, bytes_sent, referer, user_agent))
    return logs

def generate_password_spray_attack_logs(ip_address, num_logs):
    logs = []
    usernames = ['admin', 'andrii', 'pavlo', 'user1', 'user2', 'user3', 'user4', 'user5', 'user6', 'user7']
    common_password = 'Qwerty123'
    datetime_str = datetime.datetime.now().strftime('%d/%b/%Y:%H:%M:%S +0300')
    for _ in range(num_logs):
        username = random.choice(usernames)
        request_line = f"GET /login.php?user={username}&pass={common_password} HTTP/1.1"
        status = '401'
        bytes_sent = '0'
        referer = '-'
        user_agent = 'Mozilla/5.0'
        logs.append(generate_apache_log_line(ip_address, datetime_str, request_line, status, bytes_sent, referer, user_agent))
    return logs

def generate_logs(num_logs, anomaly_ratio=0.2):
    ip_address = '192.168.1.100'
    resources = ['/index.html', '/about.html', '/contact.html', '/home.html', '/login.html']
    methods = ['GET', 'POST']
```

```
    user_agents = [
        'Mozilla/5.0 (Windows NT 10.0; Win64; x64)',
        'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6)',
        'Mozilla/5.0 (Windows NT 6.1; Win64; x64)'
    ]
    statuses = ['200', '404', '500', '302']
    logs = []

    num_anomaly_logs = int(num_logs * anomaly_ratio)
    num_normal_logs = num_logs - num_anomaly_logs

    for _ in range(num_normal_logs):
        ip = f"{random.randint(10, 255)}.{random.randint(10, 255)}.{random.randint(10, 255)}.{random.randint(10, 255)}"
        datetime_str = datetime.datetime.now().strftime('%d/%b/%Y:%H:%M:%S +0300')
        method = random.choice(methods)
        resource = random.choice(resources)
        status = random.choice(statuses)
        bytes_sent = random.randint(50, 10000)
        referer = '-'
        user_agent = random.choice(user_agents)
        request_line = f"{method} {resource} HTTP/1.1"
        log = generate_apache_log_line(ip, datetime_str, request_line, status, bytes_sent, referer, user_agent)
        logs.append(log)

    # Increase anomaly logs due to increased anomaly ratio
    logs += generate_brute_force_attack_logs(ip_address, int(num_anomaly_logs / 2))
    logs += generate_password_spray_attack_logs(ip_address, int(num_anomaly_logs / 2))

    random.shuffle(logs)
    return logs

def main():
    total_logs = [7000, 1500, 1500]  # Split into training, validation, test sets
    filenames = ['train_logs.txt', 'validation_logs.txt', 'test_logs.txt']
    for num_logs, filename in zip(total_logs, filenames):
        logs = generate_logs(num_logs)
        with open(filename, 'w') as file:
            for log in logs:
                file.write(log + '\n')

if __name__ == "__main__":
    main()
```

Оновлений код генерації логів

```
import pandas as pd
import random
import datetime

def generate_apache_log_line(ip, time, method, resource, status, size, referrer, user_agent):
```

```
        anomaly_conditions = (status == "401" or
                              resource.startswith("/../../../etc") or
                              "DROP TABLE" in resource or
                              (resource == "/login" and method == "POST" and status == "401"))
        is_anomalous = "1" if anomaly_conditions else "0"
        return f"{ip} - - [{time}] \"{method} {resource} HTTP/1.1\" {status} {size} \"{referrer}\" \"{user_agent}\"", is_anomalous

    def generate_logs(num_logs):
        ips = ["192.168." + str(random.randint(1, 254)) + "." + str(random.randint(1, 254)) for _ in range(256)]
        resources = ["/", "/login", "/admin", "/dashboard", "/api/data", "/settings", "/profile", "/logout", "/login.php"]
        methods = ["GET", "POST", "DELETE", "PUT"]
        user_agents = ["Mozilla/5.0", "curl/7.47.0", "PostmanRuntime/7.26.8", "python-requests/2.25.1", "Googlebot/2.1"]
        statuses = ["200", "404", "500", "302", "401", "403", "301"]
        logs = []
        anomalies = []

        for _ in range(num_logs):
            ip = random.choice(ips)
            time = datetime.datetime.now().strftime("%d/%b/%Y:%H:%M:%S +0000")
            method = random.choice(methods)
            resource = random.choice(resources)
            status = random.choice(statuses)
            size = random.randint(0, 10000)
            referrer = random.choice(["-", "http://example.com", "https://mysite.com", "https://knu.ua"])
            user_agent = random.choice(user_agents)

            log, is_anomalous = generate_apache_log_line(ip, time, method, resource, status, size, referrer, user_agent)
            logs.append(log)
            anomalies.append(is_anomalous)

        return logs, anomalies

    def write_logs(logs, filename):
        with open(filename, "w") as f:
            for log in logs:
                f.write(log + "\n")

    def parse_logs(logs, anomalies):
        parsed_data = []
        for log, anomalous in zip(logs, anomalies):
            parts = log.split(" ")
            ip = parts[0]
            datetime = parts[3] + " " + parts[4]
            method = parts[5][1:]
            resource = parts[6]
            protocol = parts[7][:-1]
            status = parts[8]
```

```
            size = parts[9]
            parsed_data.append([ip, datetime, method, resource, protocol, status, size, anomalous])
        return pd.DataFrame(parsed_data, columns=["IP", "DateTime", "Method", "Resource", "Protocol", "Status", "Size", "IsAnomalous"])

    # Генерація логів
    num_logs = 300000  # кількість генерованих логів
    logs, anomalies = generate_logs(num_logs)

    # Розділення на навчальний та валідаційний набори
    split_index = int(num_logs * 0.8)
    train_logs, validate_logs = logs[:split_index], logs[split_index:]
    train_anomalies, validate_anomalies = anomalies[:split_index], anomalies[split_index:]

    # Запис у файли
    write_logs(train_logs, "train_logs.txt")
    write_logs(validate_logs, "validate_logs.txt")

    # Парсинг логів та запис у CSV
    df_train = parse_logs(train_logs, train_anomalies)
    df_validate = parse_logs(validate_logs, validate_anomalies)
    df_train.to_csv("train_logs.csv", index=False)
    df_validate.to_csv("validate_logs.csv", index=False)
```

Код для навчання моделі

```
    import re
    import pandas as pd

    def parse_apache_log(log_line):
        pattern = re.compile(r'(\d+\.\d+\.\d+\.\d+) - - \[(.*?)\] "(.*?)" (\d+) (\d+) "(.*?)" "(.*?)"')
        match = pattern.match(log_line)
        if match:
            return {
                "IP": match.group(1),
                "DateTime": match.group(2),
                "Request": match.group(3),
                "Status": match.group(4),
                "BytesSent": match.group(5),
                "Referer": match.group(6),
                "UserAgent": match.group(7)
            }
        return None

    def process_logs(input_filename, output_filename):
        parsed_data = []
        with open(input_filename, "r") as log_file:
            for line in log_file:
                log_data = parse_apache_log(line.strip())
                if log_data:
                    parsed_data.append(log_data)
        df = pd.DataFrame(parsed_data)
```

```
    df.to_csv(output_filename, index=False)

def main():
    files = [('train_logs.txt', 'train_parsed.csv'), ('validation_logs.txt', 'validation_parsed.csv'), ('test_logs.txt', 'test_parsed.csv')]
    for input_file, output_file in files:
        process_logs(input_file, output_file)

if __name__ == "__main__":
    main()
```

Оновлений код для навчання моделі

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, accuracy_score
import numpy as np
import joblib
import matplotlib.pyplot as plt

# Зчитування даних
train_data = pd.read_csv("train_logs.csv")
validate_data = pd.read_csv("validate_logs.csv")

# Підготовка даних
def prepare_data(df):
    le = LabelEncoder()
    for column in ['IP', 'Method', 'Resource', 'Protocol']:
        df[column] = le.fit_transform(df[column])
    scaler = StandardScaler()
    df['Size'] = scaler.fit_transform(df[['Size']])
    return df.drop(columns=['DateTime', 'Status'])

X_train = prepare_data(train_data)
y_train = train_data['IsAnomalous'].astype(int)
X_validate = prepare_data(validate_data)
y_validate = validate_data['IsAnomalous'].astype(int)

# Навчання моделі Isolation Forest
model = IsolationForest(n_estimators=100, contamination=float(np.sum(y_train) / len(y_train)), random_state=42)
model.fit(X_train, y_train)

# Збереження моделі
joblib.dump(model, 'isolation_forest_model.pkl')

# Прогнозування аномалій на валідаційному наборі
y_pred = model.predict(X_validate)
y_pred = [1 if x == -1 else 0 for x in y_pred]
```

```
# Оцінка точності моделі
print("Класифікаційний звіт:")
print(classification_report(y_validate, y_pred))
print("Точність:", accuracy_score(y_validate, y_pred))

# Візуалізація результатів
plt.scatter(X_validate['IP'], X_validate['Size'], c=y_validate, cmap='viridis', label='Actual Anomaly')
plt.scatter(X_validate['IP'], X_validate['Size'], c=y_pred, cmap='hot', label='Predicted Anomaly', marker='x')
plt.xlabel('IP')
plt.ylabel('Розмір')
plt.title('Візуалізація аномалій у логах Apache')
plt.legend()
plt.show()
```

Код для ставлення міток аномальності

```
import pandas as pd

def add_anomaly_labels(input_filename, output_filename):
    data = pd.read_csv(input_filename)

    # Розділення стовпця 'Request' на 'Method' та 'URL'
    data[['Method', 'URL']] = data['Request'].str.extract(r'(\S+) (/[^ ]*)')
    data.drop(columns=['Request'], inplace=True)  # Опціонально видаляємо старий стовпець 'Request'

    # Визначення аномалій
    data['IsAnomaly'] = ((data['URL'].str.contains('/login.php')) & (data['Status'].astype(str) == '401')).astype(int)

    # Збереження нових даних з мітками аномалій у новий файл
    data.to_csv(output_filename, index=False)

def process_files(files):
    for input_file, output_file in files:
        print(f"Processing {input_file}...")
        add_anomaly_labels(input_file, output_file)

files_to_process = [
    ('train_parsed.csv', 'train_labeled.csv'),
    ('validation_parsed.csv', 'validation_labeled.csv'),
    ('test_parsed.csv', 'test_labeled.csv')
]

process_files(files_to_process)
```

Код для очищення даних

```
import re
import pandas as pd

def parse_apache_log(log_line):
    pattern = re.compile(r'(\d+\.\d+\.\d+\.\d+) - - \[(.*?)\] "(.*?)" (\d+) (\d+) "(.*?)" "(.*?)"')
    match = pattern.match(log_line)
    if match:
        return {
            "IP": match.group(1),
            "DateTime": match.group(2),
            "Request": match.group(3),
            "Status": match.group(4),
            "BytesSent": match.group(5),
            "Referer": match.group(6),
            "UserAgent": match.group(7)
        }
    return None

def process_logs(input_filename, output_filename):
    parsed_data = []
    with open(input_filename, "r") as log_file:
        for line in log_file:
            log_data = parse_apache_log(line.strip())
            if log_data:
                parsed_data.append(log_data)
    df = pd.DataFrame(parsed_data)
    df.to_csv(output_filename, index=False)

def main():
    files = [('train_logs.txt', 'train_parsed.csv'), ('validation_logs.txt', 'validation_parsed.csv'),
('test_logs.txt', 'test_parsed.csv')]
    for input_file, output_file in files:
        process_logs(input_file, output_file)

if __name__ == "__main__":
    main()
```

Код для валідації

```
import pandas as pd
import joblib
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Завантаження моделі
model = joblib.load('isolation_forest_model.pkl')
```

```
# Завантаження тестових даних без міток аномалій
test_data = pd.read_csv('test_parsed.csv')

# Передбачимо, що 'BytesSent' є тим признаком, на основі якого модель робила навчання
X_test = test_data[['BytesSent']].values

# Отримання передбачень моделі
y_pred_test = model.predict(X_test)

# Перетворення міток з -1 (аномалія) та 1 (норма) на 1 (аномалія) та 0 (норма) відповідно
y_pred_test = [1 if i == -1 else 0 for i in y_pred_test]

# Візуалізація результатів
plt.figure(figsize=(10, 5))
sns.scatterplot(x=X_test.ravel(), y=y_pred_test, color='red', label='Predicted Anomaly')
plt.xlabel('Bytes Sent')
plt.ylabel('Is Anomaly')
plt.legend()
plt.title('Isolation Forest Anomaly Detection on Unseen Data')
plt.show()
```

Код для виявлення помилок

```
import pandas as pd
import joblib
import matplotlib.pyplot as plt

# Завантаження збереженої моделі
model = joblib.load('isolation_forest_model.pkl')

# Завантаження тестових даних
test_data = pd.read_csv('test_parsed.csv')

# Вибірка признаків для тестування моделі
X_test = test_data[['BytesSent']].values

# Прогнозування моделлю на тестовому наборі даних
y_pred = model.predict(X_test)

# Міняємо мітки з -1 на 1 для аномалій
y_pred = [1 if i == -1 else 0 for i in y_pred]

# Візуалізація результатів
plt.figure(figsize=(10, 5))
plt.scatter(range(len(X_test)), X_test, c=y_pred, cmap='coolwarm', label='Predicted Anomaly')
plt.colorbar()
plt.xlabel('Index')
plt.ylabel('Bytes Sent')
plt.title('Predicted Anomalies in Test Data')
plt.legend()
plt.show()
```