НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри

_____ Дмитро ЛАНДЕ

(підпис)

«_____»_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Розробка підсистеми аналізу журналу транзакцій MS SQL для визначення інсайдерських атак

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-04
(шифр групи)

Омелянович Олександр Віталійович
(прізвище, ім'я, по батькові) (підпис)

Керівник Коломицев Михайло Володимирович, кандидат наук, доцент кафедри ІБ.

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2024 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ **Дмитро ЛАНДЕ**
(підпис)

«___»_____2024 р.

ЗАВДАННЯ на дипломну роботу здобувачу вищої освіти

Омеляновичу Олександру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка підсистеми аналізу журналу транзакцій MS SQL для визначення інсайдерських атак

керівник роботи Коломицев Михайло Володимирович, кандидат наук, доцент кафедри інформаційної безпеки

затверджені наказом по університету від _____ _____ 2024 р. No

2. Термін подання здобувачем вищої освіти роботи ___ _____ 2024 р.

3. Вихідні дані до роботи: наукові матеріали, стаття про фреймворк для виявлення аномалій

4. Зміст роботи: аналіз та вибір методів фіксації та аналізу аномалій, реалізація системи

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): Презентація до захисту дипломної роботи

6. Дата видачі завдання: 10 лютого 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання та узгодження теми роботи	10.02.2024	Виконано
2	Пошук інформаційних джерел	10.02.2024 - 15.03.2024	Виконано
3	Огляд інструментів для фіксації дій	15.03.2024 – 25.04.2024	Виконано
4	Огляд алгоритмів виявлення аномалій	26.04.2024 – 10.05.2024	Виконано
5	Створення підсистеми аналізу	11.05.2024 – 27.05.2024	Виконано
6	Тестування та оцінення моделі	28.05.2024 – 09.06.2024	Виконано
7	Передзахист дипломної роботи	12.06.2024	Виконано
8	Захист дипломної роботи	19.06.2024	Виконано

Здобувач вищої освіти _____ Омелянович Олександр
(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

Керівник роботи _____ Михайло Коломицев
(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

РЕФЕРАТ

Обсяг роботи 50 сторінок, 11 ілюстрацій, 2 додатки, 8 джерел літератури.

Об’єктом дослідження є системи управління базами даних, як ключовий компонент інформаційної інфраструктури організацій

Предметом дослідження є методи та засоби фіксації і вивлення аномалій в журналу транзакцій

Метою дослідження є розробка системи для аналізу журналів транзакцій MS SQL, яка дозволяє ідентифікувати потенційні інсайдерські атаки. Система повинна забезпечувати ефективне виявлення зловмисних дій або ненавмисних помилок з боку користувачів, які мають доступ до корпоративних баз даних.

ABSTRSACT

The volume of the paper is 50 pages, 11 illustrations, 2 appendices, and 8 references.

The object of research is database management systems as a key component of the information infrastructure of organizations

The subject of research is methods and means of fixing and detecting anomalies in the transaction log

The purpose of the study is to develop a system for analyzing MS SQL transaction logs, which allows identifying potential insider attacks. The system should provide effective detection of malicious actions or unintentional errors by users who have access to corporate databases.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СУБД - Система управління базами дааних

XEvent - Extended Events

MS sql - Microsoft SQL Server

ODBC - Open Database Connectivity

IF - Isolation Forest

Recall - частка реальних позитивних випадків модель вдало ідентифікувала

Precision - відсоток ідентифікованих моделлю позитивних випадків є коректними.

F-score — це комбінація recall і precision, виражена у формі гармонійного середнього.

ВСТУП

На поточний момент, серйозною проблемою для багатьох організацій є забезпечення безпеки їхніх інформаційних систем, особливо у контексті захисту від інсайдерських загроз. Інсайдерські атаки можуть бути особливо руйнівними, оскільки зловмисники мають легітимний доступ до системи і глибоке розуміння її архітектури. Ця дипломна робота присвячена розробці системи, яка дозволяє виявляти та аналізувати такі загрози за допомогою детального моніторингу транзакцій в MS SQL Server, використовуючи сучасні інструменти і методики аналізу даних.

Актуальність роботи обумовлена необхідністю захисту організацій від інсайдерських атак, які становлять серйозну загрозу для безпеки корпоративних даних. Зростаюча залежність від цифрових систем підвищує ризики, пов'язані з діями осіб, що мають легітимний доступ до ресурсів, роблячи своєчасне виявлення та реагування на такі загрози критично важливими для забезпечення інформаційної безпеки.

Об’єктом дослідження є системи управління базами даних, як ключовий компонент інформаційної інфраструктури організацій

Предметом дослідження є методи та засоби фіксації і вивлення аномалій в журналу транзакцій

Метою дослідження є розробка системи для аналізу журналів транзакцій MS SQL, яка дозволяє ідентифікувати потенційні інсайдерські атаки. Система повинна забезпечувати ефективне виявлення зловмисних дій або ненавмисних помилок з боку користувачів, які мають доступ до корпоративних баз даних.

Методи дослідження. Аналіз теоретичних джерел для вивчення сучасних підходів і технологій. Вибір необхідних інструментів і параметрів для реалізації підсистеми.

Практичне значення одержаних результатів визначається можливістю їх застосування для підвищення безпеки інформаційних систем шляхом своєчасного виявлення інсайдерських атак. Розроблена система сприяє забезпеченню захисту конфіденційної інформації.

1 Теоретичний огляд

1.1 Огляд інсайдерських атак

Інсайдерські атаки є одними з найбільш серйозних і складних загроз для інформаційної безпеки організацій. Ці атаки здійснюються особами, які мають легальний доступ до внутрішніх ресурсів, таких як співробітники, підрядники, партнери або колишні працівники. Оскільки інсайдери вже мають доступ до критично важливих даних та систем, виявлення та запобігання їх діям є складним завданням.

Визначення інсайдерських атак. Інсайдерські атаки включають будь-які зловмисні дії, що здійснюються особами з внутрішнім доступом до систем організації. Вони можуть варіюватися від простих помилок або недбалості до цілеспрямованих злочинних дій з метою отримання вигоди або нанесення шкоди організації.

Основні причини інсайдерських атак

Фінансові вигоди: Інсайдери можуть здійснювати атаки з метою особистого збагачення, продаючи конфіденційну інформацію конкурентам або використовуючи її для власних потреб.

Інтереси третіх сторін: Інсайдери можуть бути під впливом зовнішніх осіб або організацій, які використовують їх для отримання доступу до цінних даних або систем.

Недбалість: Часто атаки здійснюються через недбалість або незнання співробітників, які не усвідомлюють наслідків своїх дій.

Методи інсайдерських атак

Інсайдерські атаки можуть здійснюватися різними способами, включаючи:

Несанкціоноване зчитування даних: Інсайдери можуть використовувати свої права доступу для перегляду конфіденційної інформації, що може бути використано для зловмисних цілей.

Модифікація даних: Внесення змін до даних може призвести до серйозних порушень в роботі систем та бізнес-процесів.

Видалення даних: Інсайдери можуть видаляти важливі файли або записи, що може спричинити значні втрати для організації.

1.2 Види інсайдерських атак

Інсайдерські загрози можна класифікувати на кілька типів залежно від мотивів і методів зловмисників: [2]

Зловмисні інсайдери: цей тип загрози стосується осіб, які навмисно завдають шкоди організації, наприклад, викрадають конфіденційну інформацію, інтелектуальну власність або саботаж систем.

Випадкові інсайдери: до цього типу загрози залучаються особи, які ненавмисно завдають шкоди організації, наприклад через втечу даних або збій системи.

Скомпрометовані інсайдери: ця загроза стосується осіб, чиї облікові записи або пристрої було зламано, що дозволяє зловмисникам отримати несанкціонований доступ до такої інформації.

Сторонні інсайдери: ця загроза стосується осіб, пов’язаних із постачальником, підрядником або діловим партнером, які мають авторизований доступ до конфіденційної інформації та систем.

Співпраця внутрішніх і зовнішніх осіб: ця загроза передбачає співпрацю внутрішніх осіб із зовнішніми зловмисниками для вчинення зловмисних дій.

1.3 Огляд інсайдерських атак на СУБД

Системи управління базами даних (СУБД) є ключовим компонентом інформаційної інфраструктури організацій, оскільки вони зберігають і обробляють великі обсяги конфіденційних даних. Інсайдерські атаки на СУБД можуть бути надзвичайно руйнівними через доступ зловмисника до важливих даних.

Несанкціоноване читання даних.

Інсайдери можуть використовувати свої права доступу для перегляду конфіденційної інформації, яка може включати:

Фінансові дані: Інформація про фінансові транзакції, баланси, звіти тощо.

Персональні дані клієнтів: Імена, адреси, номери телефонів, ідентифікаційні номери, медичні записи тощо.

Несанкціоноване читання даних може здійснюватися для подальшого продажу цієї інформації, використання її для шантажу або для особистої вигоди.

Модифікація або видалення даних

Інсайдери можуть змінювати або видаляти дані, що може призвести до серйозних наслідків:

Фальсифікація даних: Зміна даних для шахрайства, приховування незаконних дій або створення неправдивої інформації.

Видалення даних: Знищення критично важливої інформації, що може порушити бізнес-процеси, спричинити фінансові втрати або пошкодити репутацію організації.

1.4 Важливість фіксування інсайдерських атак

Фіксування інсайдерських атак є дуже важливим елементом забезпечення інформаційної безпеки організацій. Це не лише допомагає виявляти та запобігати зловмисним діям, але й забезпечує доказову базу для проведення розслідувань і дотримання нормативних вимог. Розглянемо детально, чому фіксування таких атак має велике значення.

Виявлення і запобігання атакам

Виявлення і запобігання атакам є критично важливими для забезпечення інформаційної безпеки організації. Фіксація дій користувачів, особливо тих, що мають доступ до критичних систем, дозволяє своєчасно виявляти підозрілу активність. Це включає виявлення аномалій у поведінці, таких як часті спроби доступу до конфіденційних даних поза робочими годинами, нехарактерна активність або масове копіювання файлів. Також важливо фіксувати підозрілі зміни у системі, такі як створення нових облікових записів, зміна прав доступу або несанкціоноване видалення даних. Аналіз дій користувачів допомагає виявляти спроби несанкціонованого доступу до даних.

Збереження доказів

Збереження доказів є критично важливим аспектом забезпечення інформаційної безпеки. Фіксація інсайдерських атак забезпечує наявність

доказів, необхідних для проведення розслідувань інцидентів. Збережені журнали та записи дій користувачів є важливими для внутрішніх розслідувань і визначення винних осіб. Крім того, ці докази можуть бути використані в судових процесах для притягнення зловмисників до відповідальності. Детальні звіти про інциденти також дозволяють керівництву організації приймати обґрунтовані рішення щодо покращення заходів безпеки.

Підвищення безпеки

Аналіз зафіксованих інцидентів дозволяє організаціям виявляти слабкі місця в системах безпеки та розробляти ефективніші заходи захисту.

Аналіз вразливостей: Вивчення патернів атак допомагає виявляти типові слабкі місця і вразливості, які використовують інсайдери.

Поліпшення заходів безпеки: На основі аналізу інцидентів організації можуть розробляти та впроваджувати нові заходи захисту, такі як додаткові політики доступу або підвищення рівня навчання співробітників.

Підвищення обізнаності співробітників: Регулярні тренінги та інформування співробітників про типові загрози та правила безпеки допомагають знизити ризик інсайдерських атак.

Висновки до першого розділу

У цьому розділі було розглянуто інсайдерські атаки як значну загрозу інформаційній безпеці організацій. Аналізувалися основні причини таких атак, включаючи фінансові вигоди, вплив третіх осіб та недбалість співробітників. Освітлено різні методи здійснення

інсайдерських атак, від несанкціонованого доступу до даних до їх знищення чи модифікації.

Була підкреслена важливість фіксування інсайдерських атак через їх роль у виявленні та запобіганні інцидентам, а також у забезпеченні доказової бази для розслідувань.

2 Методи фіксації підключень, дій та їх аналіз

2.1 Методи фіксації транзакцій в MS SQL

Фіксація транзакцій у Microsoft SQL Server є важливою складовою для забезпечення аудиту та безпеки даних. Основними методами фіксації транзакцій є використання тригерів, функції fn_dblog() та використання аудиту.

Тригери в MS SQL

Тригери в Microsoft SQL Server є спеціальними об’єктами бази даних, які автоматично виконують визначені дії у відповідь на певні дії, такі як вставка, оновлення або видалення рядків. Вони допомагають забезпечити цілісність даних, автоматизувати процеси та здійснювати аудит змін.

Типи тригерів

AFTER тригери: Виконуються після завершення операції вставки, оновлення або видалення даних. Вони можуть перевіряти стан даних та виконувати додаткові дії після того, як основна транзакція успішно завершилася.

INSTEAD OF тригери: Замінюють стандартну операцію. Вони дозволяють визначити альтернативні дії, які виконуються замість стандартної операції.

Переваги

Автоматизація: Тригери дозволяють автоматично виконувати певні дії при зміні даних, що зменшує потребу у ручному втручанні.

Забезпечення цілісності даних: Тригери можуть гарантувати дотримання правил цілісності даних, таких як обмеження, залежності або бізнес-правила.

Аудит і безпека: Тригери можуть фіксувати зміни в даних, створюючи журнали аудиту для відстеження дій користувачів.

Недоліки

Продуктивність: Тригери можуть збільшувати навантаження на систему, особливо у випадках великих обсягів транзакцій, оскільки виконуються кожного разу при виникненні відповідної події.

Складність: Тригери можуть ускладнювати структуру бази даних і процеси її адміністрування, що може призвести до труднощів у відлагодженні та підтримці.

Непередбачувані ефекти: Неправильно налаштовані тригери можуть призвести до непередбачуваних результатів або циклічних викликів, що може пошкодити дані або спричинити збої в системі.

Аудит в MS SQL Server

Аудит в Microsoft SQL Server є важливим компонентом для забезпечення безпеки баз даних. Він надає можливість моніторингу і запису дій користувачів та змін у базах даних, що є критично важливим для виявлення потенційних загроз, забезпечення відповідності нормативним вимогам, та проведення внутрішніх розслідувань.

SQL Server Audit дозволяє створювати детальні журнали дій користувачів, включаючи, виконання запитів, зміну даних, а також зміни в конфігурації бази даних. Ці журнали можуть зберігатися у файлах, журналах подій Windows або таблицях бази даних.

Налаштування аудиту включає створення специфікації аудиту, в якій визначаються події та дії, що підлягають моніторингу. Важливо правильно

налаштувати специфікації аудиту, щоб уникнути пропуску важливих подій та надмірного накопичення даних.

Переваги:

Безпека: Аудит дозволяє відстежувати всі дії користувачів, що допомагає виявляти та запобігати несанкціонованим діям.

Відповідність нормативним вимогам: Багато стандартів та регуляторних актів вимагають ведення журналів аудиту для забезпечення прозорості та відповідальності.

Аналіз та звітування: Детальні журнали аудиту забезпечують наявність доказів для внутрішніх розслідувань та юридичних процесів.

Недоліки:

Вплив на продуктивність: Ведення журналів аудиту може створювати додаткове навантаження на систему, особливо при великому обсязі транзакцій.

Складність налаштування: Неправильна конфігурація аудиту може призвести до пропуску важливих подій або накопичення надмірної кількості даних.

Функція fn_dblog()

Функція fn_dblog() в Microsoft SQL Server є потужним інструментом для доступу до журналу транзакцій бази даних. Вона дозволяє отримувати детальну інформацію про всі транзакції, що відбувалися в базі даних, і може бути корисною для аудиту та моніторингу змін.

Ця функція забезпечує доступ до активного журналу транзакцій бази даних, дозволяючи користувачам отримувати інформацію про всі операції,

що були зафіксовані в цьому журналі. Журнал транзакцій зберігає інформацію про всі зміни, внесені до бази даних, включаючи вставку, оновлення та видалення даних.

Функцію fn_dblog() можна викликати для перегляду записів у журналі транзакцій. Приклад використання цієї функції:

SELECT * FROM fn_dblog(NULL, NULL)

Цей запит повертає всі записи з активного журналу транзакцій. Перший параметр функції визначає початкову транзакцію, а другий – кінцеву. Якщо обидва параметри встановлені на NULL, функція повертає всі доступні записи.

Переваги:

Низьке навантаження на систему: На відміну від тригерів, fn_dblog() не створює додаткового навантаження на систему, оскільки просто читає існуючі записи в журналі транзакцій.

Детальна інформація: Функція дозволяє отримати вичерпну інформацію про кожну транзакцію, включаючи її тип, час, користувача, що її виконав, і багато інших деталей.

Реальний час: fn_dblog() дозволяє отримувати дані в реальному часі, що є важливим для оперативного моніторингу та аудиту.

Недоліки:

Складність інтерпретації: Дані, що повертаються функцією, можуть бути складними для інтерпретації без відповідних знань про структуру журналу транзакцій.

2.2 Методи фіксації підключень до бази даних в MS SQL Server Extended Events

XEvent у MS SQL – це потужний і гнучкий механізм для збору діагностичних даних про різні події, що відбуваються в SQL Server. Він дозволяє налаштовувати відстеження та запис подій, що важливо для діагностики, моніторингу продуктивності та безпеки.

XEvent розроблен з урахуванням мінімального впливу на продуктивність системи. Extended Events дозволяють створювати користувацькі сесії для збору конкретної інформації про події, що відбуваються в базі даних.

Extended Events складаються з трьох основних компонентів:

Events: Описують конкретні події, що відбуваються в SQL Server, які необхідно відстежувати.

Targets: Визначають, де і як зберігати зібрані дані (наприклад, у файлах, таблицях або буферах пам'яті).

Actions: Дозволяють збирати додаткову інформацію про події (наприклад, SQL-запити, що викликали подію).

Використання SQL Server Audit

Після налаштування аудиту всі події, що відповідаютьналаштуванням, будуть записуватися у визначену ціль Ці журнали можуть бути проаналізовані для виявлення аномалій, розслідування інцидентів або дотримання нормативних вимог.

Переваги:

Безпека. Відстеження всіх спроб підключення та відключення допомагає виявляти несанкціоновані дії.

Відповідність нормативним вимогам. Забезпечує наявність необхідних журналів для аудиту та розслідувань.

Детальна інформація: Записи аудиту містять вичерпну інформацію про події, включаючи час, користувачів та інші деталі.

Недоліки:

Вплив на продуктивність. Ведення аудиту може створювати додаткове навантаження на систему.

Складність налаштування. Неправильна конфігурація аудиту може призвести до пропуску важливих подій або накопичення надмірної кількості даних.

2.3 Методи аналізу логів.

Статистичні методи

Метод Z-рахунку є стандартизованим способом визначення, наскільки далеко певне спостереження знаходиться від середнього значення, виміряного у стандартних відхиленнях. Цей метод широко використовується для виявлення аномалій, оскільки значення, які значно виходять за рамки стандартного розподілу, можуть вказувати на потенційні відхилення або аномалії [4]

Тест Граббса спеціалізується на виявленні екстремальних викидів у наборі даних. Цей тест використовується для того, щоб визначити, чи є найбільше або найменше значення у наборі даних статистично значимим викидом, порівняно з іншими значеннями.

Метод відстані Махаланобіса вимірює відстань між точкою даних та розподілом даних, враховуючи кореляції між змінними. Відстань

Махаланобіса є ефективним способом ідентифікації аномалій, оскільки вона дозволяє враховувати нормальні коливання в межах многовимірного датасету.

Глибоке навчання, підгалузь машинного навчання, базується на архітектурах глибоких нейронних мереж, які здатні вивчати складні шаблони та залежності у великих обсягах даних. Для аналізу логів глибоке навчання може використовуватися для виявлення аномалій, класифікації подій та прогнозування потенційних інсайдерських атак з високою точністю. [5]

Автоенкодери

Це тип нейронних мереж, які навчаються стиснути вхідні дані до більш компактної форми, а потім розшифровувати їх назад до початкового формату. Вони відмінно підходять для виявлення аномалій, оскільки аномальні дані часто погано реконструюються, що дозволяє виявити відхилення від норми.

Рекурентні нейронні мережі

RNN і, зокрема, їх варіанти, такі як LSTM (Long Short-Term Memory), дуже ефективні в аналізі послідовних даних, наприклад, часових рядів або лог-файлів. Вони можуть визначати та прогнозувати поведінку користувачів, виявляючи нехарактерні послідовності дій, які можуть вказувати на інсайдерські загрози. [5]

Згорткові нейронні мережі (CNN)

Хоча CNN зазвичай асоціюють із обробкою зображень, вони також використовуються для аналізу логів у вигляді матриць або багатовимірних даних. Здатність CNN ефективно визначати просторові ієрархії

характеристик може бути використана для виявлення складних шаблонів в лог-даних.

Спектральні методи

Спектральні методи також можна використовувати для виявлення аномалій у часових рядах. Ці методи використовуються, зокрема, для виявлення циклічних коливань та аномальних частотних характеристик у великих наборах даних.

Spectral Anomaly Detection один з популярних підходів у спектральних методах — це виявлення спектральних аномалій, яке базується на частотних складових у часових рядях. Цей метод дозволяє виявляти незвичайні частотні компоненти, які можуть вказувати на аномалії.

Машинне навчання

Машинне навчання включає в себе методи, які дозволяють системам вчитися з даних і виконувати завдання без явного програмування. В аналізі даних часових рядів це важливий інструмент для ідентифікації аномалій, класифікації поведінки користувачів та прогнозування загроз. Машинне навчання поділяється на дві основні категорії: контрольоване та неконтрольоване навчання.

Контрольоване машинне навчання

У контрольованому навчанні моделі навчаються на попередньо позначених даних, що означає наявність як вхідних даних, так і відповідних виходів. Моделі контрольованого навчання зазвичай надають вищу точність порівняно з методами неконтрольованого навчання. Проте, для їхнього застосування необхідна ретельна підготовка, оскільки вони потребують ручного анотування даних. [1] Цей підход підходить для

задач, де відомі конкретні категорії поведінки або коли вже існує чітке розуміння того, що вважається аномалією чи загрозою.

Неконтрольоване машинне навчання

Неконтрольоване навчання використовується, коли мітки для даних відсутні. Моделі намагаються знайти структуру або взаємозв'язки в даних без попереднього знання про результати. Це особливо корисно в ситуаціях, коли аномалії рідкісні або неоднорідні і важко визначити вручну.

Серед методів неконтрольованого навчання, Isolation Forest вирізняється своєю ефективністю у виявленні аномалій. Цей алгоритм працює на принципі ізоляції точок даних. Він рекурсивно ділить простір даних за допомогою випадкових розділів до тих пір, поки кожен об'єкт не буде "ізольований". Аномальні дані зазвичай ізолюються швидше, оскільки вони відрізняються від нормальних, що дозволяє IF ефективно їх ідентифікувати.

Висновки до другого розділу

В цьому розділі описано методи фіксації підключень та дій у базах даних MS SQL, які є критично важливими для забезпечення аудиту та безпеки. Основними інструментами, які використовуються для цих цілей, є Extended Events, функція fn_dblog(), а також системи аудиту. Кожен з цих інструментів має свої переваги та недоліки, що дозволяє вибрати найбільш оптимальний метод в залежності від специфічних потреб та умов експлуатації системи.

Цей розділ також охоплює різні методи аналізу логів, що є важливим аспектом розробки системи захисту від інсайдерських атак. Увага приділяється як статистичним методам, так і методам машинного навчання, зокрема неконтрольованому навчанню з використанням Isolation Forest, яке демонструє високу ефективність у виявленні аномалій. Глибоке навчання і спектральні методи також використовуються для детального аналізу та ідентифікації складних шаблонів поведінки, які можуть вказувати на потенційні загрози.

Цей розділ надає загальне розуміння того, як можна ефективно фіксувати та аналізувати дії у базах даних, що є важливим для створення надійних систем безпеки, спроможних ідентифікувати можливі інсайдерські атаки.

3 Практична реалізація

Основна мета цього розділу це сворення системи фіксації даних та аналізу аномалій з використанням Isolation Forest.

3.1 Структура системи і вибір інструментів

Робота системи реалізується згідно статті [1] наступним чином: спочатку відбувається збір даних, які можуть включати журнали аудиту баз даних, що містять інформацію про всі взаємодії користувачів із системою. Після збору, ці дані проходять етап підготовки, під час якого відбувається їх структурування для подальшого моніторингу. Нормалізовані дані трансформуються в формат, придатний для аналізу, і аналізуються за допомогую IF. Детектор аномалій аналізує моніторингові дані на предмет виявлення відхилень від норми.

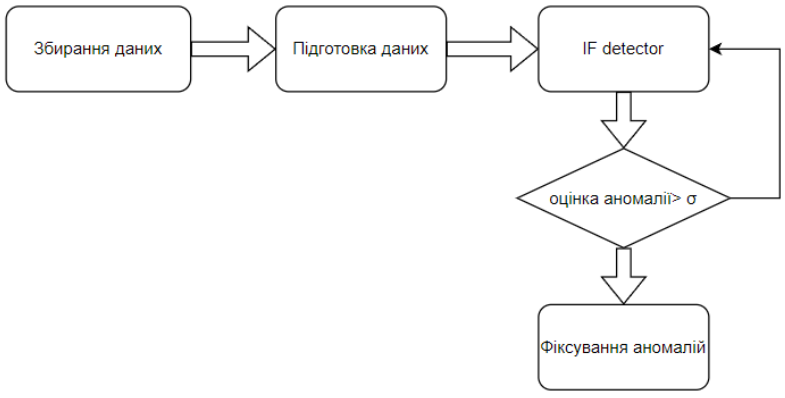

Рисунок 3.1 – Схема роботи системи

Для фіксації транзакцій у дослідженні було обрано використання функції fn_dblog() через її здатність надавати детальний огляд змін у базі даних. Окрім того, fn_dblog() характеризується низьким навантаженням на

систему порівняно з іншими методам. Ці характеристики забезпечують відносно мале втручання в продуктивність бази даних, дозволяючи зберегти швидкість обробки даних та ефективно реагувати на потенційні внутрішні загрози, що є критично важливим для безпеки системи.

Для фіксації підключень користувачів було використано Extended Events, оскільки він забезпечуючи дуже мале навантаження на систему, що є важливим для тривалого моніторингу. Extended Events дозволяє детально налаштовувати збір даних, точно вказуючи які події та параметри моніторити, і використовує асинхронний збір даних для зниження впливу на продуктивність системи. Це робить його ідеально підходящим для ефективного виявлення специфічних дій користувачів і забезпечує можливість легкої інтеграції зібраних даних з іншими інструментами аналізу. Такий підхід дозволяє забезпечити високий рівень безпеки баз даних, оперативно реагуючи на неавторизовані або підозрілі дії.

Для дослідження отриманих даних було використано методи неконтрольованого навчання з наступних причин:

Аномальні дані зустрічаються рідко порівняно з великою кількістю нормальних даних, що ускладнює використання методів контрольованого навчання.

Різноманітність типів аномалій може суттєво ускладнити процес навчання моделей, особливо коли обсяги аномальних даних є незначними. Це особливо важливо у випадках інсайдерських атак, де навчальні вибірки можуть взагалі відсутні, а ідентифіковані загрози зазвичай швидко усуваються.

Крім того, неконтрольоване навчання має меншу обчислювальну складність порівняно з керованим навчанням, що дозволяє ефективно обробляти дані в реальному часі. [1]

3.2 Реалізація системи

3.2.1 Фіксування входу та виходу користувачів

Спочатку я налаштовую Extended Events на сервері MS SQL для моніторингу подій входу та виходу користувачів із системи, за винятком дій системного адміністратора (sa). Це дозволяє мені сфокусуватися на активності звичайних користувачів та ігнорувати несуттєві системні події. Для кожної події я реєструю різні параметри, такі як додаток клієнта, ім'я хоста, ім'я користувача та ідентифікатор бази даних. Усі зібрані дані зберігаю у файл із розширенням .xel, що у директорії D:\Audit. Після налаштування сесії аудиту я запускаю її, щоб розпочати збір даних.

```
cursor = conn.cursor()

connection_string = """
    DRIVER={ODBC Driver 17 for SQL Server};
    SERVER=PERRY-NITRO;
    DATABASE=master;
    Trusted_Connection=yes;
"""

query = '''
    SELECT
        event_data.value('(event/@name)[1]', 'VARCHAR(50)') AS EventType,
        event_data.value('(event/@timestamp)[1]', 'DATETIME') AS EventDate,
        COALESCE(
            event_data.value('(event/action[@name="username"]/value)[1]', 'VARCHAR(128)'),
            event_data.value('(event/action[@name="server_principal_name"]/value)[1]', 'VARCHAR(128)'),
            event_data.value('(event/action[@name="nt_username"]/value)[1]', 'VARCHAR(128)')
        ) AS UserName,
        event_data.value('(event/action[@name="client_hostname"]/value)[1]', 'VARCHAR(128)') AS ClientHostName
    FROM (
        SELECT CAST(event_data AS XML) AS event_data
        FROM sys.fn_xe_file_target_read_file('D:\Audit\LoginLogoutAudit*.xel', NULL, NULL, NULL)
    ) AS Events
    WHERE
        event_data.value('(event/@name)[1]', 'VARCHAR(50)') IN ('login', 'logout');'''

cursor.execute(query)
```

Рисунок 3.2 – Збирання даних з аудиту

Використовуючи Python і підключення через ODBC до SQL Server, я витягую записані дані аудиту. Моє завдання — виконати запит SQL, який

парсить XML-дані з файлу аудиту та витягує інформацію про події входу та виходу, включаючи тип події, час, ім'я користувача та ім'я хоста. Це дозволяє мені аналізувати, коли і з яких комп'ютерів користувачі входили в систему і виходили з неї, що є вкрай корисним для забезпечення безпеки та моніторингу активності в базі даних.

3.2.2 Фіксація дій користувачів

```
query = """
SELECT
    [Transaction ID],
    Operation,
    AllocUnitName
FROM
    fn_dblog(NULL, NULL)
WHERE
    Operation IN ('LOP_INSERT_ROWS', 'LOP_MODIFY_ROW', 'LOP_DELETE_ROWS', 'LOP_FORMAT_PAGE')
    AND AllocUnitName LIKE 'dbo.%';
"""

cursor.execute(query)
ins_mod_del = cursor.fetchall()
```

Рисунок 3.3 – Отримання дій користувачів

У першій частині я виконую запит до функції fn_dblog, яка надає доступ до журналу транзакцій SQL Server. Цей журнал містить детальну історію всіх операцій, виконаних на рівні бази даних, включаючи кожну вставку, зміну та видалення даних.

Запит фільтрує записи за такими критеріями:

Операції: Обмежуюсь операціями LOP_INSERT_ROWS, LOP_MODIFY_ROW, LOP_DELETE_ROWS та LOP_FORMAT_PAGE. Ці операції вказують на вставку нових рядків, зміну існуючих рядків, видалення рядків та форматування сторінок бази даних відповідно. Вони є активними змінами в структурі даних і вмістом.

AllocUnitName: Фільтрую операції так, щоб вони відносилися тільки до об'єктів у схемі dbo, яка часто використовується для основних об'єктів бази даних у програмах користувача та систем.

Вибір таких специфічних операцій допомагає мені виявляти лише значні зміни в базі даних, за винятком численних інших типів операцій, які можуть бути менш релевантними для поточного аналізу.

```
transaction_ids = [row[0] for row in ins_mod_del]
ids_string = ', '.join(f"'{id}'" for id in transaction_ids)

query = f"""
SELECT
    Operation,
    [Transaction ID],
    [Begin Time],
    [Transaction Name],
    [Transaction SID],
    SUSER_SNAME([Transaction SID])
FROM
    fn_dblog(NULL, NULL)
WHERE
    [Transaction ID] IN ({ids_string})
AND
    [Operation] = 'LOP_BEGIN_XACT'
"""


cursor.execute(query)
time_sid = cursor.fetchall()
```

Рисунок 3.4 – Отримання додаткової інформації

Після отримання ID транзакції з першого запиту, я зібрав ці ID в рядок, щоб використовувати їх для фільтрації в наступному SQL-запиті. Цей запит має на меті отримання контекстної інформації про початок транзакцій, використовуючи операцію LOP_BEGIN_XACT з журналу транзакцій SQL Server. Операція LOP_BEGIN_XACT фіксує початок транзакції, що дозволяє мені виявити точний час запуску транзакції та ідентифікатор безпеки користувача (SID), який її ініціював.

Завдяки SID я додатково витягнув інформацію про користувача, який розпочав транзакцію, використовуючи функцію SUSER_SNAME(), яка перетворює SID на ім'я користувача. Це дало мені можливість зрозуміти,

хто саме відповідав за початок кожної транзакції, що важливо для аналізу активності.

```
query = '''
    SELECT
    [Transaction Id],
    [Transaction SID],
    [Transaction Name],
    [Begin Time],
    SUSER_SNAME([Transaction SID])

    FROM fn_dblog (NULL, NULL)
    WHERE [Transaction Name] IN ('DROPOBJ', 'ALTER TABLE')
'''
cursor.execute(query)
time_sid = cursor.fetchall()
```

Рисунок 3.5 – Отримання додаткових дій користувачів

Аналогічно, я зафіксував зміни у структурі бази даних, такі як видалення об'єктів та зміна таблиць, використовуючи операції DROPOBJ та ALTER TABLE у журналі транзакцій SQL Server. Я отримав інформацію про ідентифікатори транзакцій, ідентифікатори користувачів (SID), час початку транзакцій та імена транзакцій. Це допомогло мені встановити, хто і коли проводив зміни у базі даних. Використовуючи функцію SUSER_SNAME(), я також перетворив SID на імена користувачів, щоб краще зрозуміти, хто відповідав за ці дії.

3.2.3 Аналіз логів за допомогою IF

Після етапу збору та підготовки даних, на якому відбулися перетворення часових даних у потрібний формат та перетворення категоріальних змінних у формат, придатний для обробки. Також проводиться нормалізація числових значень для усунення масштабних відмінностей.

Аналіз під’єднань

Спочатку відбувається отримання та підготовка зібраних даних

```
import pandas as pd
from datetime import datetime
from sklearn.preprocessing import LabelEncoder

data = pd.read_csv("user_session.csv")


data['EventDate'] = pd.to_datetime(data['EventDate'])
data['Date'] = data['EventDate'].dt.date
data['Time'] = data['EventDate'].dt.time
data['Time_ordinal'] = data['EventDate'].map(lambda x: x.hour / 24.0 + x.minute / 1440.0 + x.second / 86400.0)

le_user = LabelEncoder()
le_host = LabelEncoder()
le_event_type = LabelEncoder()
le_date = LabelEncoder()
data['Date_encoded'] = le_date.fit_transform(data['Date'])
data['EventType_encoded'] = le_event_type.fit_transform(data['EventType'])
data['ClientHostName_encoded'] = le_host.fit_transform(data['ClientHostName'])
```

Рисунок 3.6 – Отримання та підготовка даних

Після підготовки даних відбувається виявлення аномалій за допомогою IF. Алгоритм ефективно ідентифікує аномальні точки, моделюючи їх ізоляцію за допомогою структури випадкових дерев.

```
from sklearn.ensemble import IsolationForest

scaler = MinMaxScaler()
X = scaler.fit_transform(data[['Date_encoded', 'EventType_encoded', 'Time_ordinal']])

model = IsolationForest(n_estimators=200, max_samples=256, max_features=3, contamination=0.02, random_state=42)
data['anomaly'] = model.fit_predict(X)
```

Рисунок 3.7 – Нормалізація і аналіз даних

Виходячи з результатів, ми можемо побачити, що система працює коректно з досить високою точністю.

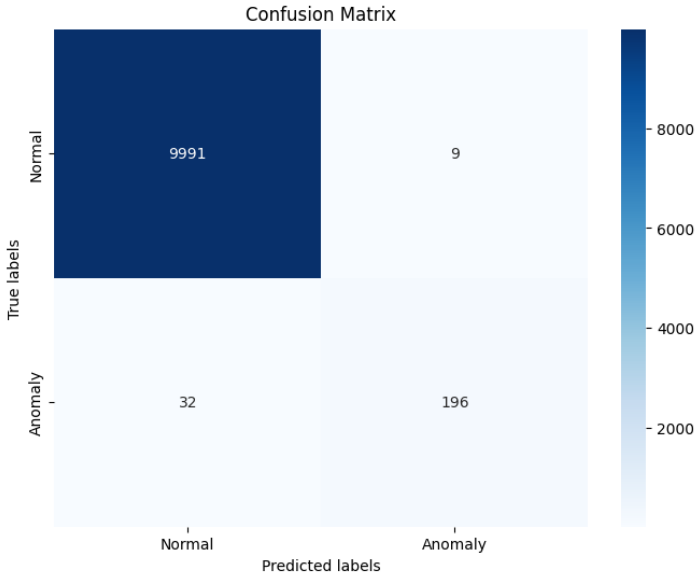

Рисунок 3.8 Результати аналізу даних

Recall: 0.8596

Precision: 0.9561

F-score: 0.9053

Аналіз дій

Для аналізу дій користувачів також спочатку отримуються та обробляються дані і виконується пошук аномалій.

```
data = pd.read_csv("transaction_logs.csv")


data['EventDate'] = pd.to_datetime(data['Begin Time'], format='%Y/%m/%d %H:%M:%S:%f')
data['Date'] = data['EventDate'].dt.date
data['Time'] = data['EventDate'].dt.time


data['Time_ordinal'] = data['EventDate'].map(lambda x: x.hour / 24.0 + x.minute / 1440.0 + x.second / 86400.0)
le_transaction = LabelEncoder()
le_user = LabelEncoder()
data['Transaction_encoded'] = le_transaction.fit_transform(data['Transaction Name'])
data['UserName_encoded'] = le_user.fit_transform(data['UserName'])
```

Рисунок 3.9 - Отримання та підготовка даних

```
scaler = MinMaxScaler()
X = scaler.fit_transform(data[['Transaction_encoded', 'UserName_encoded', 'Time_ordinal']])

model = IsolationForest(n_estimators=200, max_samples=256, max_features=3, contamination=0.02, random_state=42)
data['anomaly'] = model.fit_predict(X)
```

Рисунок 3.10 - Нормалізація і аналіз даних

Виходячи з отриманих результатів, ми можемо побачити, що система успішно ідентифікує більшість аномалій:

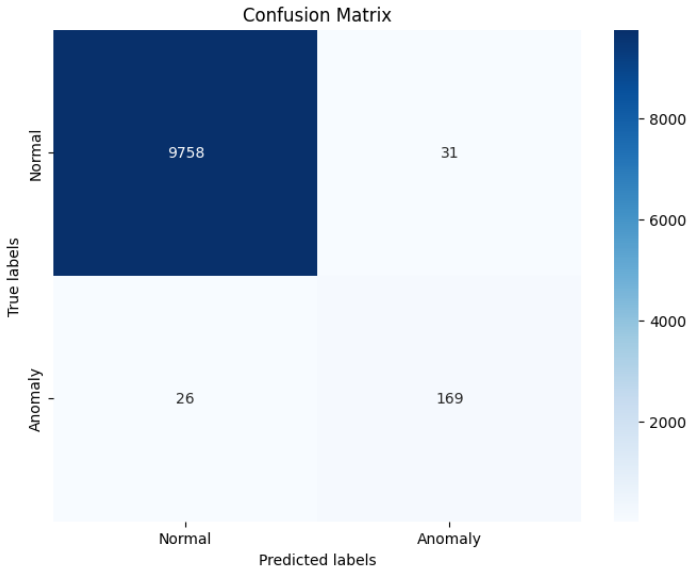

Рисунок 3.11 - Результати аналізу даних

Recall: 0.8667

Precision: 0.8450

F-score: 0.8557

Висновки до третього розділу

У цьому розділі була розроблена та реалізована система фіксації даних та аналізу аномалій з використанням методу Isolation Forest. Було впроваджено інструменти для ефективного моніторингу дій у базах даних,

зокрема fn_dblog() для фіксації транзакцій та Extended Events для слідкування за підключеннями користувачів. Ці інструменти дозволили з мінімальним навантаженням на продуктивність системи збирати дані, критично важливі для забезпечення безпеки.

Було досягнуто високої точності у виявленні аномалій, що підтверджено показниками Recall, Precision та F-score. Ці результати свідчать про ефективність використання Isolation Forest для аналізу часових рядів у контексті виявлення інсайдерських загроз.

Завдяки реалізованій системі було забезпечено засоби як для аналізу дій користувачів, так і автоматичного пошуку аномалій, що дозволяє оперативно виявляти потенційні загрози та вживати заходів до їх нейтралізації.

ВИСНОВКИ

У даній дипломній роботі було розроблено систему для аналізу дій користувачів та автоматичного пошуку аномалій, що спрямована на підвищення рівня безпеки в СУБД. Основною метою системи є виявлення потенційних внутрішніх загроз, що здатні спричинити шкоду інформаційній інфраструктурі організацій.

Фіксація дій користувачів в системах управління базами даних відіграє важливу роль у забезпеченні безпеки, що було реалізовано у даній дипломній роботі. За допомогою вибраних інструментів система точно реєструє всі взаємодії користувачів з базою даних, включаючи зміни в даних та доступи до системи. Це дозволяє виявляти аномалії та реагувати на потенційні загрози, що суттєво підвищує рівень інформаційної безпеки організації.

Система пошуку аномалій базується на використанні методу Isolation Forest для ідентифікації аномалій у поведінці користувачів, що дозволяє виявляти нестандартні дії, що можуть свідчити про інсайдерські загрози. Впровадження такої системи забезпечує можливість детального моніторингу та аналізу дій всередині СУБД, знижуючи ризики від небажаних або зловмисних дій.

Впровадження системи фіксації дій і пошуку аномалій є гарним інструментом для зміцнення інформаційної безпеки організацій. Система дозволяє не лише виявляти потенційні загрози, але й сприяє розробці стратегій для управління ризиками, що виникають в результаті діяльності користувачів у СУБД. Такий підхід дозволяє компаніям бути крок попереду потенційних внутрішніх загроз, підвищуючи загальну надійність та стабільність інформаційних систем.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

[1] Mykhailo V. Kolomytsev, Svitlana O.Nosok, Oksana V. Tsygankova Framework for detecting outlier and unknown database intrusions, Інформаційні технології та безпека. Матеріали XXIII Міжнародної науково-практичної конференції ІТБ-2023. – Київ: Інжиніринг. – 202 с. ISBN: 978-966-2344-96-7

[2] Kumar Rahul Insider Threat 01.2024 URL: https://www.wallstreetmojo.com/insider-threat/#insider-risk-vs-insider-threat

[3] Zhi-Hua Zhou National Key Laboratory for Novel Software Technology https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf

[4] J Behboodian Akbar Asgharzadeh, On the distribution of Z-scores https://www.researchgate.net/publication/228646283_On_the_distribution_of_Z-scores

[5] Deep Learning as a Frontier of Machine Learning: A Review https://www.researchgate.net/publication/326429676_Deep_Learning_as_a_Frontier_of_Machine_Learning_A_Review

[6] Guerrino Mazzarolo, Anca D Jurcut. Insider threats in Cyber Security: The enemy within the gates https://www.researchgate.net/publication/337438838_Insider_threats_in_Cyber_Security_The_enemy_within_the_gates

[7] Chuang Peng, Weilin Hu and Lunwen Wang, Spectrum Anomaly Detection Based on Spatio-Temporal Network Prediction https://www.mdpi.com/2079-9292/11/11/1770

[8] Cory Maklin, Isolation Forest https://medium.com/@corymaklin/isolation-forest-799fceacdda4

ДОДАТОК А

```
import csv
from datetime import datetime
import pyodbc
import binascii


connection_string = """
    DRIVER={ODBC Driver 17 for SQL Server};
    SERVER=PERRY-NITRO;
    DATABASE=NEWDB;
    Trusted_Connection=yes;
"""


conn = pyodbc.connect(connection_string)
cursor = conn.cursor()


query = """
SELECT
    [Transaction ID],
    Operation,
    AllocUnitName
FROM
```

```
    fn_dblog(NULL, NULL)
WHERE
    Operation IN ('LOP_INSERT_ROWS', 'LOP_MODIFY_ROW',
'LOP_DELETE_ROWS', 'LOP_FORMAT_PAGE')
    AND AllocUnitName LIKE 'dbo.%';
"""

cursor.execute(query)
ins_mod_del = cursor.fetchall()

transaction_ids = [row[0] for row in ins_mod_del]
ids_string = ', '.join(f"'{id}'" for id in transaction_ids)

query = f"""
SELECT
    Operation,
    [Transaction ID],
    [Begin Time],
    [Transaction Name],
    [Transaction SID],
    SUSER_SNAME([Transaction SID])
FROM
```

```
    fn_dblog(NULL, NULL)
WHERE
    [Transaction ID] IN ({ids_string})
AND
    [Operation] = 'LOP_BEGIN_XACT'
"""

cursor.execute(query)
time_sid = cursor.fetchall()



final_data = []
for id1, op1, alloc1 in ins_mod_del:
    found = False
    for op2, id2, time, table, sid, username in time_sid:
        if id1 == id2:
            final_data.append((id1, alloc1, time, table, username))
            found = True
            break

    if not found:
        final_data.append((id1, op1, alloc1, None, None, None))
```

```
query = '''
    SELECT
    [Transaction Id],
    [Transaction SID],
    [Transaction Name],
    [Begin Time],
    SUSER_SNAME([Transaction SID])

    FROM fn_dblog (NULL, NULL)
    WHERE [Transaction Name] IN ('DROPOBJ', 'ALTER TABLE')
'''
cursor.execute(query)
time_sid = cursor.fetchall()

for id, sid, tr, time, username in time_sid:
    final_data.append((id, "None", time, tr, username))



def parse_datetime(dt_str):
    return datetime.strptime(dt_str, "%Y/%m/%d %H:%M:%S:%f")
```

```
sorted_final_data = sorted(final_data, key=lambda x: parse_datetime(x[2]))



# print(f"{'\n\nTransaction ID':<25} {'AllocUnitName':<60} {'Begin Time':<30} {'Transaction Name':<20} {'UserName':<20}")

# print('-' * 150)

# for row in sorted_final_data:

#     print(f"{row[0]:<25} {row[1]:<60} {row[2]:<30} {row[3]:<20} {row[4]:<20}")


csv_file_path = 'TransactionData.csv'

with open(csv_file_path, mode='w', newline='') as file:

    writer = csv.writer(file)

    writer.writerow(['Transaction ID', 'AllocUnitName', 'Begin Time', 'Transaction Name', 'UserName'])

    writer.writerows(sorted_final_data)


print(sorted_final_data)

cursor.close()

conn.close()
```

```
conn = pyodbc.connect(connection_string)
cursor = conn.cursor()

connection_string = """
    DRIVER={ODBC Driver 17 for SQL Server};
    SERVER=PERRY-NITRO;
    DATABASE=master;
    Trusted_Connection=yes;
"""

query = '''
    SELECT
        event_data.value('(event/@name)[1]', 'VARCHAR(50)') AS EventType,
        event_data.value('(event/@timestamp)[1]', 'DATETIME') AS EventDate,
        COALESCE(
            event_data.value('(event/action[@name="username"]/value)[1]',
'VARCHAR(128)'),
```

```
event_data.value('(event/action[@name="server_principal_name"]/value)[1]',
'VARCHAR(128)'),

        event_data.value('(event/action[@name="nt_username"]/value)[1]',
'VARCHAR(128)')

    ) AS UserName,

    event_data.value('(event/action[@name="client_hostname"]/value)[1]',
'VARCHAR(128)') AS ClientHostName

  FROM (

    SELECT CAST(event_data AS XML) AS event_data

    FROM sys.fn_xe_file_target_read_file('D:\Audit\LoginLogoutAudit*.xel',
NULL, NULL, NULL)

  ) AS Events

  WHERE

    event_data.value('(event/@name)[1]', 'VARCHAR(50)') IN ('login',
'logout');'"



cursor.execute(query)



loginlogout = cursor.fetchall()



csv_file_path = 'LoginLogoutData.csv'
```

```
formatted_loginlogout = [(event, dt.strftime('%Y-%m-%d %H:%M:%S'), user,
server) for event, dt, user, server in loginlogout]


with open(csv_file_path, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Event', 'Datetime', 'User', 'Server'])
    writer.writerows(formatted_loginlogout)


cursor.close()
conn.close()
```

ДОДАТОК Б

```
import pandas as pd

from datetime import datetime

from sklearn.preprocessing import LabelEncoder

data = pd.read_csv("user_session.csv")


data['EventDate'] = pd.to_datetime(data['EventDate'])

data['Date'] = data['EventDate'].dt.date

data['Time'] = data['EventDate'].dt.time

data['Time_ordinal'] = data['EventDate'].map(lambda x: x.hour / 24.0 +
x.minute / 1440.0 + x.second / 86400.0)


le_user = LabelEncoder()

le_host = LabelEncoder()

le_event_type = LabelEncoder()

le_date = LabelEncoder()

data['Date_encoded'] = le_date.fit_transform(data['Date'])

data['EventType_encoded'] = le_event_type.fit_transform(data['EventType'])

data['ClientHostName_encoded'] =
le_host.fit_transform(data['ClientHostName'])

from sklearn.preprocessing import MinMaxScaler

from sklearn.ensemble import IsolationForest
```

```
scaler = MinMaxScaler()
X = scaler.fit_transform(data[['Date_encoded', 'EventType_encoded',
'Time_ordinal']])

model = IsolationForest(n_estimators=200, max_samples=256,
max_features=3, contamination=0.02, random_state=42)
data['anomaly'] = model.fit_predict(X)

import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns

data['anomaly_encoded'] = data['anomaly'].apply(lambda x: -1 if x == -1 else 1)
cm = confusion_matrix(data['Normal'], data['anomaly_encoded'], labels=[1, -1])

fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', ax=ax)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
```

```
ax.xaxis.set_ticklabels(['Normal', 'Anomaly'])

ax.yaxis.set_ticklabels(['Normal', 'Anomaly'])


plt.show()


tn, fp, fn, tp = cm.ravel()

print(f"True Positives (TP): {tp}")

print(f"True Negatives (TN): {tn}")

print(f"False Positives (FP): {fp}")

print(f"False Negatives (FN): {fn}")


recall = tp / (tp + fn)

print(f"Recall: {recall:.4f}")


precision = tp / (tp + fp)

print(f"Precision: {precision:.4f}")


f_score = 2 * (recall * precision) / (recall + precision)

print(f"F-score: {f_score:.4f}")

data.to_csv("user_session _detected.csv", index=False)

data = pd.read_csv("transaction_logs.csv")
```

```
data['EventDate'] = pd.to_datetime(data['Begin Time'], format='%Y/%m/%d %H:%M:%S:%f')

data['Date'] = data['EventDate'].dt.date

data['Time'] = data['EventDate'].dt.time


data['Time_ordinal'] = data['EventDate'].map(lambda x: x.hour / 24.0 + x.minute / 1440.0 + x.second / 86400.0)

le_transaction = LabelEncoder()

le_user = LabelEncoder()

data['Transaction_encoded'] = le_transaction.fit_transform(data['Transaction Name'])

data['UserName_encoded'] = le_user.fit_transform(data['UserName'])


scaler = MinMaxScaler()

X = scaler.fit_transform(data[['Transaction_encoded', 'UserName_encoded', 'Time_ordinal']])


model = IsolationForest(n_estimators=200, max_samples=256, max_features=3, contamination=0.02, random_state=42)

data['anomaly'] = model.fit_predict(X)


data['anomaly'] = data['anomaly'].apply(lambda x: -1 if x == -1 else 1)

cm = confusion_matrix(data['is_anomaly'], data['anomaly'], labels=[1, -1])
```

```
fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', ax=ax)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['Normal', 'Anomaly'])
ax.yaxis.set_ticklabels(['Normal', 'Anomaly'])
plt.show()

tn, fp, fn, tp = cm.ravel()
print(f"True Positives (TP): {tp}")
print(f"True Negatives (TN): {tn}")
print(f"False Positives (FP): {fp}")
print(f"False Negatives (FN): {fn}")

recall = tp / (tp + fn)
print(f"Recall: {recall:.4f}")
precision = tp / (tp + fp)
print(f"Precision: {precision:.4f}")
f_score = 2 * (recall * precision) / (recall + precision)
print(f"F-score: {f_score:.4f}")
```