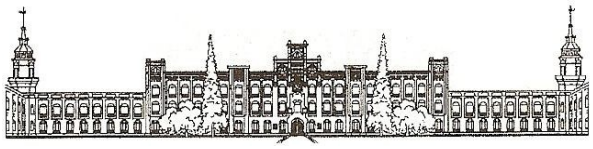

Міністерство освіти і науки України Національний
технічний університет України

«Київський політехнічний інститут»

Розрахунково-графічна робота

з дисципліни «Захист інформації в комп’ютерних системах»

Виконав студент групи: КВ-22

ПІБ: Крутогуз Максим Ігорович

Перевірив:

_____

Київ 2024

Тема розрахунково-графічної роботи

Розробити додаткові компоненти захисту СУБД.

Реферат

У наші дні через велике використання даних та зберігання даних, майже кожні організації та установі потребують використання баз даних, тому потреба в додаткових компонентах захисту СУБД стає дедалі більш актуальною. Ми можемо побачити статистику щодо зростання кіберзлочинів на рисунку 1.

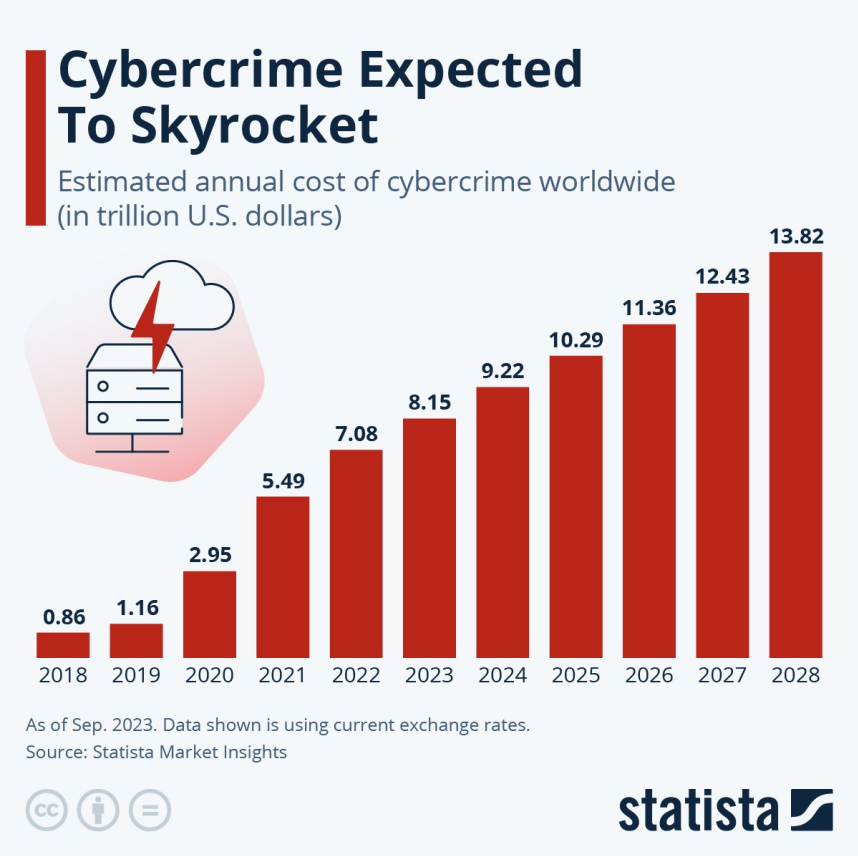

Рисунок 1 — Оцінка збільшення кількості кіберзлочинів

Через потребу в посиленні захисту СУБД, зокрема в розробці систем додаткового захисту; в цій роботі буде розглянуто методи захисту, в яких за основу взято використання алгоритмів штучного інтелекту та машинного навчання. Також буде спроба реалізації додаткового захисту.

Спочатку задамо собі питання навіщо нам захищати СУБД та навіщо це може бути потрібно. Через те, що за допомогою Інтернету можна підключитися до бази даних або сервісів практично будь-якого місця; то зловмисник може спробувати нанести якусь шкоду базам даних, які можуть

знаходитися в тисячах кілометрів від його місцезнаходження. Розглянемо приклади можливих атак на бази даних. Перший тип атак — це атаки із вторгненням. Вони можуть впроваджуватися, наприклад, за допомогою злому паролю або вставки SQL-ін'єкцій. Другий тип атак — це спотворення даних. Тертій тип атак — це атаки на відмову(DoS). Цей тип атак націлений не на викрадення даних чи їх спотворення, а на надмірне навантаження на системи з метою їх відмови для законних користувачів. Четвертий тип атак — це викрадення даних, які в подальшому можуть використатися в несанкціонованих діях.

Тепер задаймо собі питання як використання технологій штучного інтелекту та машинного навчання може допомогти в захисту баз даних чи взагалі варто використовувати ці технології для вирішення такого класу задач. Для початку розберімося із цими поняттями. Штучний інтелект — це інтелект, який використовується машинами, зокрема комп'ютерними системами. Основою для розвитку такого інтелекту служать методи, що дозволяються машинам сприймати середовище чи матеріал, з яким вони працюють чи взаємодіють; вчитися за допомогою них з подальшою метою максимізації шансів максимально бажаний результат. А що таке машинне навчання? Машине навчання — це галузь дослідження штучного інтелекту, яке використовує статистичні алгоритми для навчання ШІ, що дозволяє отримати нові дані, після аналізу старих даних. Якщо використовувати технології ШІ та машинного навчання, то аналізуючи дані підключень користувачів та їхньої взаємодією із базою даних, можна виявляти аномальну поведінку з більш високою точністю з порівнянням до традиційних методів та швидко вжити заходи захисту для заборони цим користувачам взаємодіяти із системою на постійні основі або на тимчасовий час. Але це буде працювати лише при умові якщо штучний інтелект пройде навчання і для цього потрібні помічені дані для його навчання.

Розглянемо загрози більш детальніше, аби зрозуміти яким чином краще їх виявляти, а далі і захищатися. Почнемо спочатку із SQL-ін'єкцій, але аби зрозуміти SQL-ін'єкції краще, перед цим потрібно розглянути, що представляє у собі SQL запит. SQL запит — це рядок-команда, яка може виконуватися системами управління баз даних, наприклад MySQL, PostreSQL. Тобто за допомогою SQL запитів виконується взаємодія та керування із даними баз даних. Через це, в деяких випадках зловмиснику може успішно отримати несанкціонований непрямий контроль до системи управління базою даних, якщо існує вразливість у системі, щодо можливості виконання чужого SQL запиту. Інша відома загроза — це так звана атака нульового дня. Ця загроза використовує вразливості програмного забезпечення для нанесення шкоди. Тому на відмінну від інших загроз для загроз нульового дня складно підготуватися, оскільки їй важко передбачити. Вони з'являється, коли під час

розробки програмного забезпечення не було ретельно випробовувано програмний продукт на вразливості, з подальшою метою їх усунення. При умові, якщо вразливості є у системі, зловмисник може знайти ці вразливості, для того щоб в подальному нанести шкоду системі. Крім цих, існує низка інших загроз за допомогою зловмисник може отримати неавторизований доступ до баз даних. Причини цього явища можуть бути такими як: недосконала система авторизації користувачів, фішингові атаки, атаки з викраденням паролю, соціальна інженерія тощо. Можемо зробити висновок, що базових заходів для системи захисту баз даних може бути недостатньо, тому впровадження додаткових заходів захисту є актуальною проблемою.

Повернімося знову до штучного інтелекту та машинного навчання і з'ясуймо чому використання цих технологій є ефективними та використовуються на практиці. По-перше, цей метод не потребує використання складних алгоритмів, тобто завдяки використання ШІ швидкість виявлення загроз зростає. Тому ці методи можуть використовуватися в реальному часі для виявлення загроз системі. По-друге, після гарного навчання моделі, ШІ може виявляти загрози досить точно. Також, при виявлені загроз іншого типу модель штучного інтелекту може швидко адаптуватися і до нових загроз. Для розуміння ефективності цього метода переглянемо дослідження, яка було опубліковане 30 листопада 2024 року в *„International Journal of Scientific Research in Computer Science, Engineering and Information Technology“*, дослідження доступне за посиланням. В цьому дослідженні приводять приклад щодо статистичних даних про провадження кіберзахисту в баз даних з використанням штучного інтелекту. По-перше, компанії, які покращили системи захисту з використанням штучного інтелекту продемонстрували точність виявлення загроз 96.8% пункт 3.2 дослідження. Якщо ідентифікація потенційних загроз компаніями, які використовували традиційні методи всього складала 67% точності.

Перейдемо до більш детального способу реалізації AI/ML(Artificial intelligence/Machine learning) для захисту баз даних. По-перше, потрібно стежити за діями, які робить користувач, аби в подальшому можна було автоматично виявляти незвичні патерни доступу до системи, і в подалі вжити заходів для запобіганню нанесенню шкоди системи. Тобто спочатку вручну при виявлені аномальної поведінки можна позначити в даних для подальшого навчання моделі. Найбільш важливими елементами для простежування являється: логіни користувачів та локація, на якій була зроблена спроба для входу у систему; надзвичайна кількість запитів за короткий час: не звична поведінка для нормального користування системою; типи виконуваних операцій, тощо. З використанням AL/ML на основі аналізу минулих атак, можна складати прогнози щодо наступних атак. Робота штучного інтелекту часто базується на знаходження функції, яка інтерполюється завдяки вже

відомих даних, тому ми можемо робити прогноз з якоюсь точністю про майбутні атаки. Після виявлення аномальної поведінкою засобами AI/ML відбувається автоматичне блокування підозрілих запитів. Для цього може застосовуватися: блокування користувача або його IP адреси, надсилання сповіщення адміністратору системи, тимчасове блокування до доступу бази даних.

Тепер спробуємо розробити систему додаткового захисту вже на практиці. Для того щоб спрости проектування такої системи, ми не будемо взаємодіяти із справжніми базами даними, а просто використаємо дата-сет, із платформи *Kaggle*. Оберемо файл із даними, який призначений для тренування моделі рисунок 2. В кожному рядочку таблиці є різноманітні дані щодо здійсненної операції, але нам найбільш цікаві є два стовпеця: перший із назвою sus(suspicious), в якому зберігається 1 або 0, і за допомогою цього значення можна визначити, що операція була підозрілою чи ні; другий стовпець *evil* цікавий оскільки за допомогою якого можна сказати чи операція завдала шкоди системи чи ні.

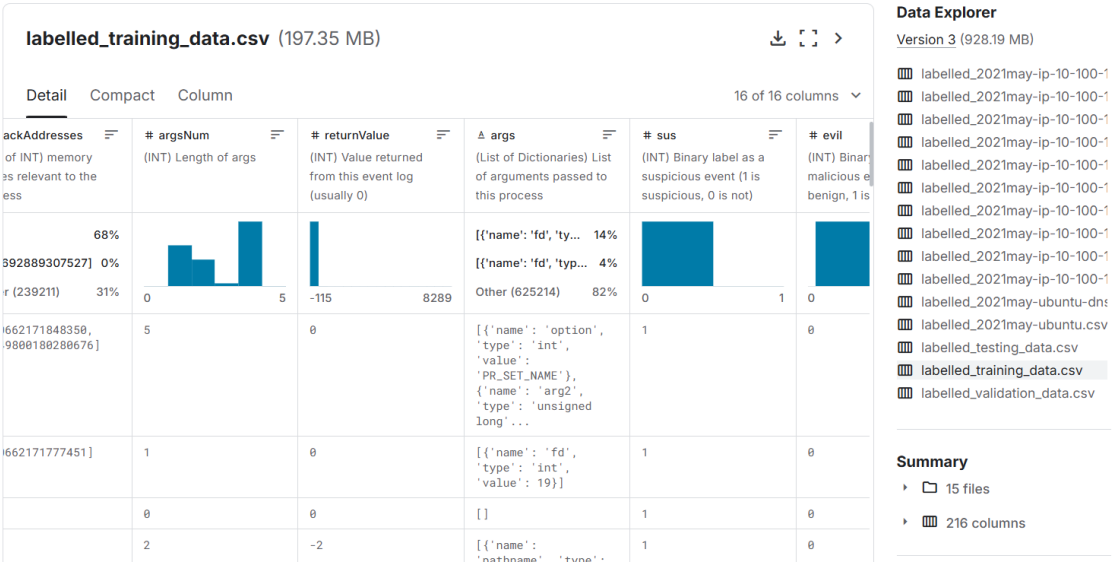

Рисунок 2 — Дані для тренування моделі

Тепер перейдімо до спроби візуалізації даних в Python з використанням бібліотеки *matplotlib* та *pandas*.

Після того як дані були завантаженні в програму, вони були спочатку відображенні в консолі рисунок 3 та рисунок 4. Також потрібно звернути увагу про тип подій, які зображені на рисунку 4. Можна побачити, що на систему не було здійснено жодного успішного нападу протягом 763144 операцій, але підозрілі операції досягнули відмітки 1269 раз.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 763144 entries, 0 to 763143
Data columns (total 18 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   timestamp        763144 non-null  datetime64[ns]
 1   processId        763144 non-null  int64
 2   threadId         763144 non-null  int64
 3   parentProcessId  763144 non-null  int64
 4   userId           763144 non-null  int64
 5   mountNamespace   763144 non-null  int64
 6   processName      763144 non-null  object
 7   hostName         763144 non-null  object
 8   eventId          763144 non-null  int64
 9   eventName        763144 non-null  object
 10  stackAddresses   763144 non-null  object
 11  argsNum          763144 non-null  int64
 12  returnValue      763144 non-null  int64
 13  args             763144 non-null  object
 14  sus              763144 non-null  int64
 15  evil             763144 non-null  int64
 16  is_suspicious    763144 non-null  bool
 17  is_evil          763144 non-null  bool
dtypes: bool(2), datetime64[ns](1), int64(10), object(5)
memory usage: 94.6+ MB
None
```

Рисунок 3 — Відображення інформації щодо даних: назви колонок

	timestamp	processId	threadId	parentProcessId	userId	...	args	sus	evil	is_suspicious	is_evil
0	2024-01-01 00:30:09.495787	381	7337	1	100	...	[{'name': 'option', 'type': 'int', 'value': 'P...	1	0	True	False
1	2024-01-01 00:30:09.495832	381	7337	1	100	...	[{'name': 'fd', 'type': 'int', 'value': 19}]	1	0	True	False
2	2024-01-01 00:30:09.495921	381	7337	1	100	...	[]	1	0	True	False
3	2024-01-01 00:31:34.139651	7347	7347	7341	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
4	2024-01-01 00:31:34.142127	7347	7347	7341	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
5	2024-01-01 00:31:34.142589	7347	7347	7341	0	...	[{'name': 'dirfd', 'type': 'int', 'value': -10...	1	0	True	False
6	2024-01-01 00:31:34.142753	7347	7347	7341	0	...	[{'name': 'fd', 'type': 'int', 'value': 3}, {'...	1	0	True	False
7	2024-01-01 00:31:34.143329	7347	7347	7341	0	...	[{'name': 'fd', 'type': 'int', 'value': 3}]	1	0	True	False
8	2024-01-01 00:31:34.143403	7347	7347	7341	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
9	2024-01-01 00:31:34.143855	7347	7347	7341	0	...	[{'name': 'dirfd', 'type': 'int', 'value': -10...	1	0	True	False
10	2024-01-01 00:31:34.144259	7347	7347	7341	0	...	[{'name': 'fd', 'type': 'int', 'value': 3}, {'...	1	0	True	False
11	2024-01-01 00:31:34.145427	7347	7347	7341	0	...	[{'name': 'fd', 'type': 'int', 'value': 3}]	1	0	True	False
12	2024-01-01 00:31:34.147710	7347	7347	7341	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
13	2024-01-01 00:31:34.149590	7347	7347	7341	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
14	2024-01-01 00:31:34.149924	7347	7347	7341	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
15	2024-01-01 00:31:34.150057	7347	7347	7341	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
16	2024-01-01 00:31:34.150950	7347	7347	7341	0	...	[{'name': 'flags', 'type': 'int', 'value': 'CL...	1	0	True	False
17	2024-01-01 00:31:34.151534	7348	7348	7347	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
18	2024-01-01 00:31:34.153310	7348	7348	7347	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False
19	2024-01-01 00:31:34.154757	7348	7348	7347	0	...	[{'name': 'pathname', 'type': 'const char*', '...	1	0	True	False

```
[20 rows x 18 columns]
Evil event count: 0
Suspicious event count: 1269
Total event count: 763144
```

Рисунок 4 — Перші 20 рядків даних та інформація про тип події

Тепер, аби зрозуміти тенденцію змін підозрілих та шкідливих операцій у час, зробимо візуалізацію цих даних. Візуалізацію цих даних можна побачити на рисунку 5. Суть візуалізації полягає в такому, що дані були по груповані 15 хвилинним інтервалом та на кожному інтервалі було підраховано кількість шкідливих та підозрілих подій. Код реалізації логіки, за допомогою якого було зроблені рисунки 3, 4 та 5 можна побачити в таблиці із кодом 1. Як висновок до візуалізованих даних можемо зробити такий: кількість підозрілих дій зменшувалася на протязі часу. Це тенденція може бути спричинена блокуванням користувачів, які роблять підозрілі дії; або підозрілі дії вже занеслися до нормальної поведінки користувачів. Навіщо блокувати

користувачів, які роблять підозрілі дії? Відповідь на це питання полягає в тому, що вони є потенційними зловмисниками і якщо їх заблокувати перед тим як вони спробують завдати якусь шкоду, то шанс виникнення шкідливих подій різко може знизитися, схожу тенденцію можемо спостерігати на графіку рисунку 5.

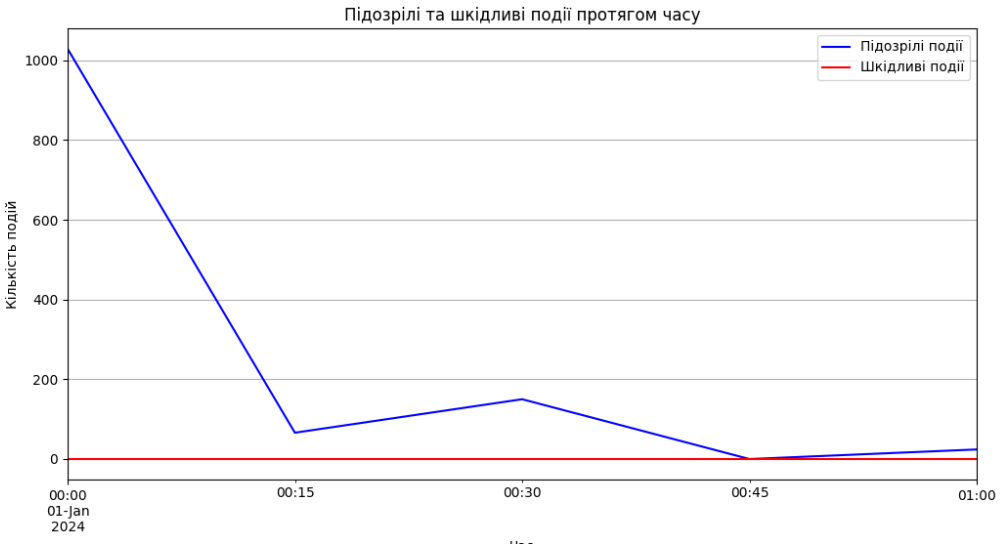

Рисунок 5 — Тенденція змін шкідливих та підозрілих операцій протягом час

Код 1 — Код виведення інформації щодо даних та їх візуалізація

```
import kagglehub # type: ignore
import pandas as pd
import matplotlib.pyplot as plt # type: ignore
from datetime import datetime, timedelta

# Читання csv файлу, який був завантажений перед цим
file_path = "C:/Users/pc/.cache/kagglehub/datasets/katehighnam/beth-
dataset/versions/3/labelled_training_data.csv"
data = pd.read_csv(file_path)


# Встановлюємо стартову дату початку роботи сервера
start_time = datetime(2024, 1, 1, 0, 0, 0)
data['timestamp'] = data['timestamp'].apply(lambda x: start_time +
timedelta(seconds=x))

# Додаємо до кожного рядка у вигляді флагу значення, яка відображає чи
була підозріла чи шкідлива операція
data['is_suspicious'] = data['sus'] == 1
data['is_evil'] = data['evil'] == 1


# Робимо стовпець timestamp головним, розбиваємо дані на блоки по 15
хвилин та обраховуємо суму елементів, якщо вони істині
time_group = data.set_index('timestamp').resample('15min').agg({
    'is_suspicious': 'sum',
```

```
    'is_evil': 'sum'
})

# Візуалізація даних
plt.figure(figsize=(12, 6))
time_group['is_suspicious'].plot(label='Підозрілі події', color='blue')
time_group['is_evil'].plot(label='Шкідливі події', color='red')
plt.title('Підозрілі та шкідливі події протягом часу')
plt.xlabel('Час')
plt.ylabel('Кількість подій')
plt.legend()
plt.grid(True)
plt.show()

print(data.info())
print(data.head(20))
print(sum(1 for v in data['is_evil'] if v == True))
print(sum(1 for v in data['is_suspicious'] if v == True))
```

Оскільки розробка повноцінної моделі ШІ може зайняти багато часу, спочатку спробуємо проаналізувати дані, які вважаються підозрілими і спробуємо з’ясувати чому ці спрацювання не є властивими для користувачів, які користуються системою як потрібно. Після того коли розробляти модель ШІ, розуміння таких тенденцій може допомогти в розробці.

Спочатку перегляньмо, які типи подій класифікуються частіше за підозрілі; дивляться записи, які вже прокласифіковані підозрілими. Список доступних подій можна подивитися на рисунку 6.

```
['prctl' 'close' 'sched_process_exit' 'access' 'security_file_open'
 'openat' 'fstat' 'stat' 'clone' 'execve' 'security_bprm_check'
 'getdents64' 'lstat' 'security_inode_unlink' 'unlink']
```

Рисунок 6 — Назви подій які можуть використовуватися легітимним користувачем

Після візуалізації подивімося на сам розподіл кількості підозрілих операцій в залежності від їхньої назви на рисунку 7. Також є код(код 2), за допомогою була побудована ця гістограма.

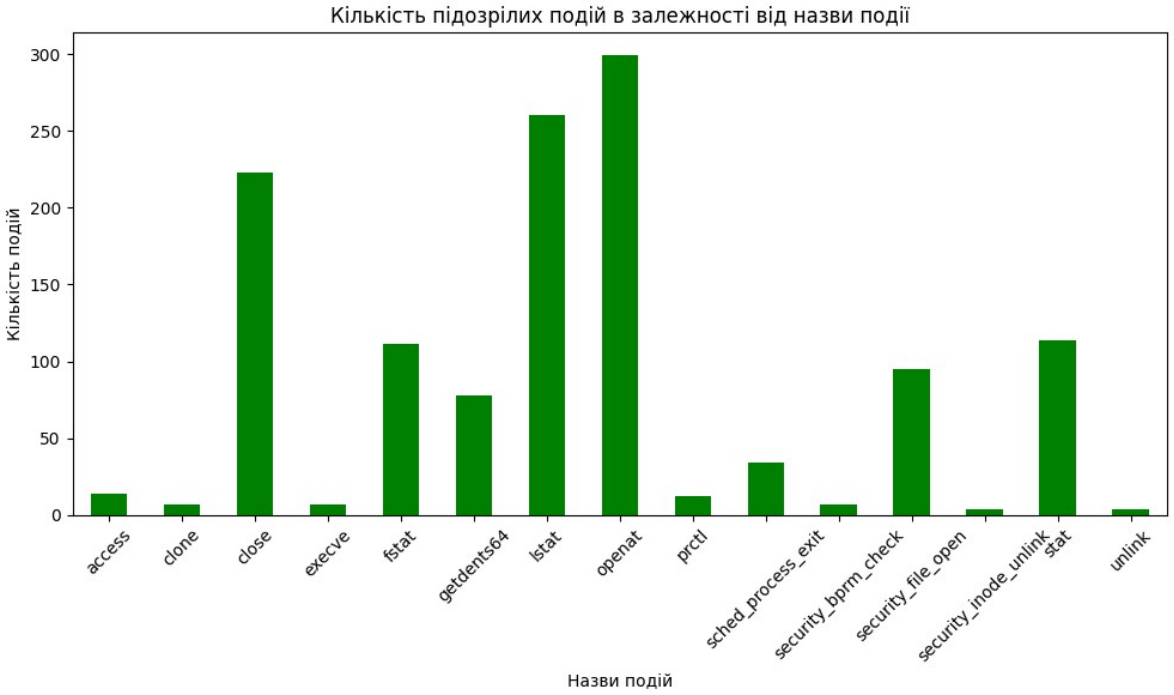

Рисунок 7 — Кількість підозрілих подій в залежності від назви події

Код 2 — Код візуалізації, який визначає кількість підозрілих подій в залежності від назви події

```
import pandas as pd
import matplotlib.pyplot as plt # type: ignore
from datetime import datetime, timedelta
import numpy as np

# Читання csv файлу, який був завантажений перед цим
file_path = "C:/Users/pc/.cache/kagglehub/datasets/katehighnam/beth-
dataset/versions/3/labelled_training_data.csv"
data = pd.read_csv(file_path)

data['is_suspicious'] = data['sus'] == 1
data['is_evil'] = data['evil'] == 1

data_suspicious = data[data['is_suspicious']]

unique_fields = data_suspicious['eventName'].unique()

# створення словника для відображення даних на гістрограмі
eventname_count = {}
eventname = {}
eventname['name'] = []
eventname['value'] = []

for field in unique_fields:
    count = data_suspicious[data_suspicious['eventName'] ==
field].shape[0]
    eventname_count[field] = count
    eventname['name'].append(field)
```

```
        eventname['value'].append(count)

    print(unique_fields)

    df = pd.DataFrame(eventname)

    # Будування гістограми
    plt.figure(figsize=(10, 6))
    df.groupby('name')['value'].sum().plot(kind='bar', color='green')
    plt.title('Кількість підозрілих подій в залежності від назви події')
    plt.xlabel('Назви подій')
    plt.ylabel('Кількість подій')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

На наступній гістограмі рисунок 8 показано, з яким ідентифікатором користувача і з якою кількістю було проведена підозріла операція. На рисунку 9 показано в загалом, з яким ідентифікатором користувача були здійснені операції. Рисунок 10, те саме що на рисунку 9, але було виключено користувача з ідентифікатором 0. Рисунок 9 та рисунок 10 був згенерований для того, щоб можна порівняти кількість всіх операцій та підозрілих операцій, здійснені з певним конкретним користувачем.

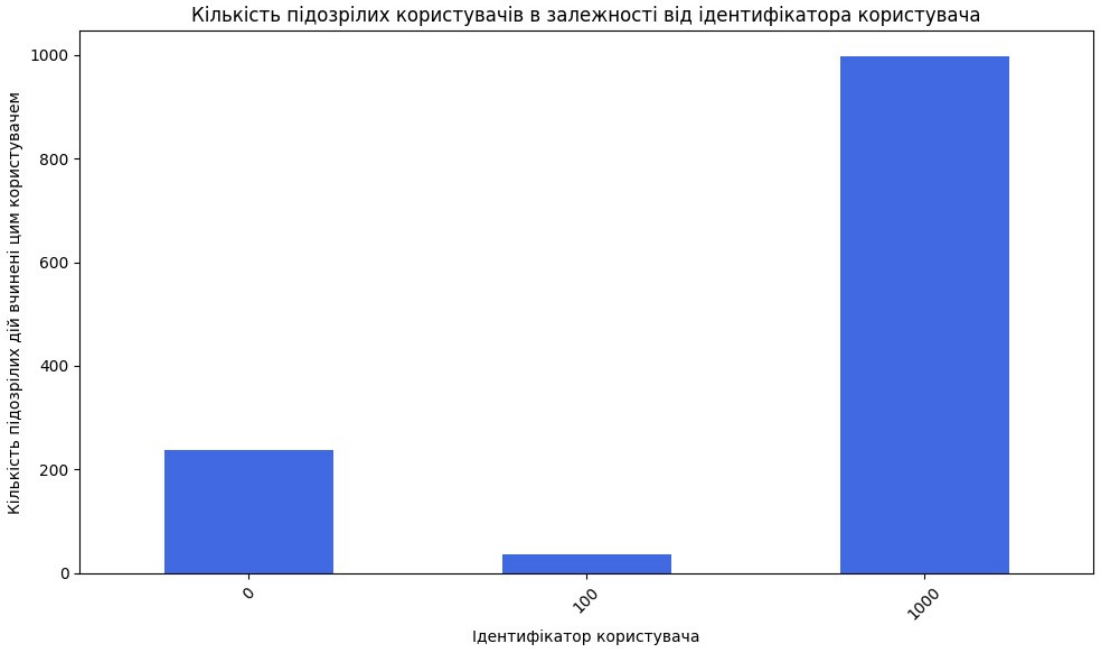

Рисунок 8 — Кількість підозрілих користувачі з залежності від їхнього ідентифікатори, які класифікується в підозрілі події

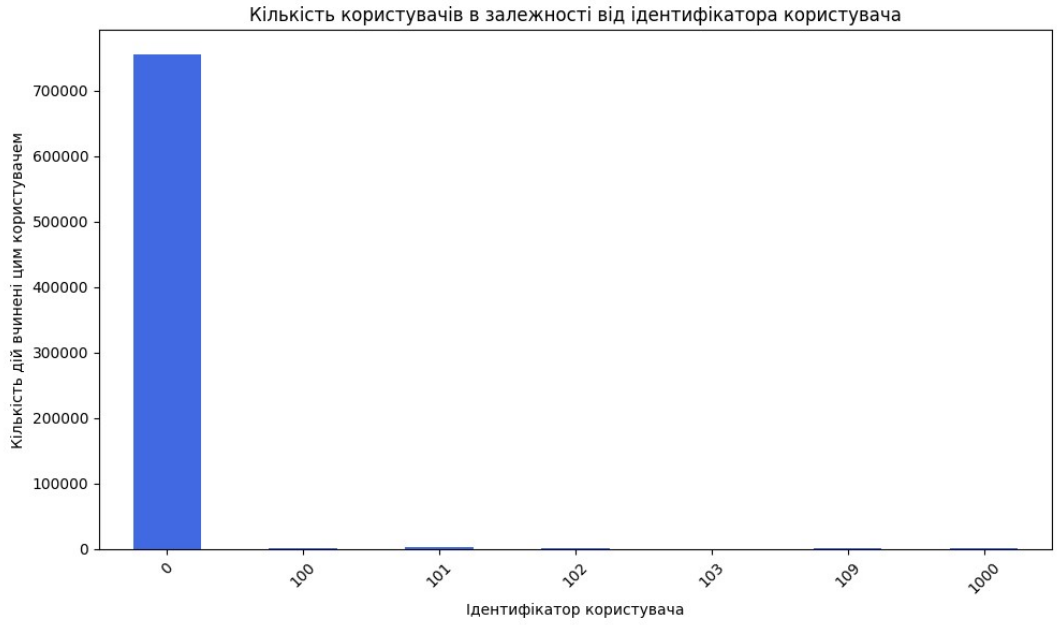

Рисунок 9 — Кількість усіх користувачів, які здійснювали якісь дії у системі

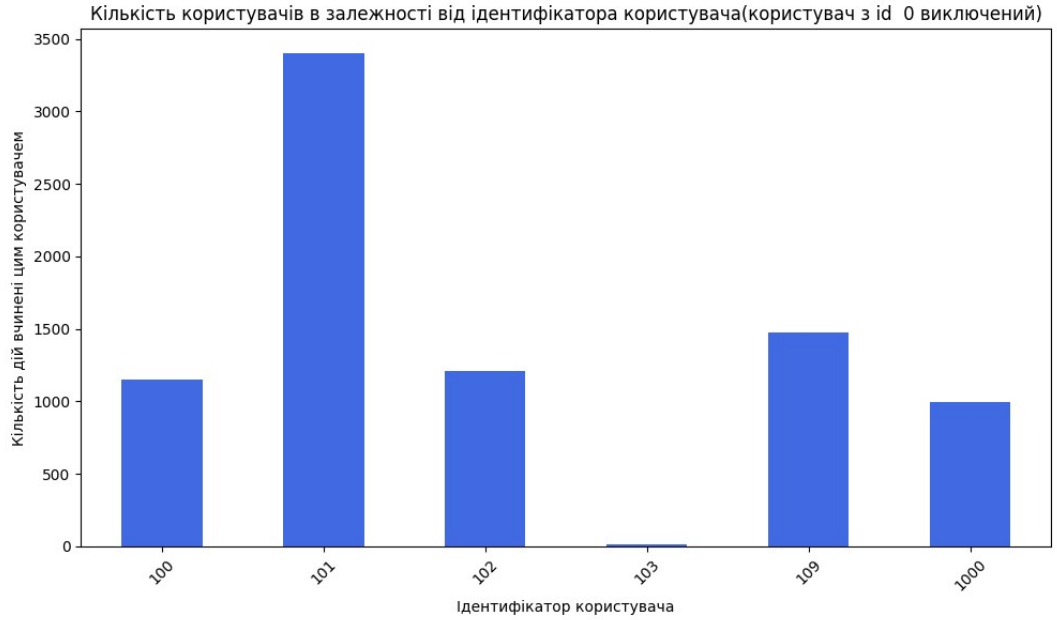

Рисунок 10 — Кількість усіх користувачів, за виключення користувача з ідентифікатором 0(ті самі дані які на рисунку 9 без користувача із ідентифікатором 0)

Тепер спробуємо підсумувати дані, які були отримані на діаграмах. На рисунку 7, ми можемо бачити домінування *openat*, *lstat*, *close*. *openat* можливо була спроба доступу до критичних файлів системи. *lstat* можливо була спроба дізнатися про інформацію щодо специфічних файлів. close можливо було спроба закрити файл/зупинити процес в незвичайний момент. Найменш

підозрілими операціями були *unlink, security_file_open, clone*. *unlink* можливо приховування слідів(можливо рідко це відбувається). security_file_open можливо аналіз системи захисту, аби швидше і точніше наносити шкоду системи(можливо рідко, оскільки доступ можна перевірити: є чи нема). *clone* можливо для розпаралелення чисельних псевдо-задач, аби заплутати системних адміністраторів від основного процесу нанесення шкоди системи.

Щодо рисунків 8, 9, 10. Ми можемо побачити що існує один користувач, який здійснював лише підозрілі операції, ідентифікатор 1000. Та є користувач 0, який здійснив переважну кількість операцій у системі, але кількість підозрілих операцій не дуже висока. Як висновок, ці операції були позначені підозрілими від специфіки цих операцій.

Висновок

Можемо вже підбити підсумки. По-перше, використання штучного інтелекту може підвищити точність та ефективність розпізнавання атак, або унеможливити їх виникнення. Для цього потрібно навчити модель штучного інтелекту, для цього потрібні позначені дані щодо інформації про здійснення атак або вживання шкідливих атак. Для запобігання атак краще блокувати користувачів, які здійснюють аномальну поведінку заздалегідь, замість того щоб чекати коли атака буде здійснена. Щодо позначання підозрілих операцій в даних для тренування моделі ШІ, потрібно виходити від специфіки системи: як вона працює та якими операціями не властивими звичайним користувача, система може попасти під загрозу.

Список використаної літератури

Bakharev, N. (2024, January 10). *Unauthorized Access: Risks, Examples, and 6 Defensive Measures*. Bright Security. https://brightsec.com/blog/unauthorized-access-risks-examples-and-6-defensive-measures/

Kumar, S. (2024). Best Practices for Database Security in the Age of AI. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, *10*(6), 1127–1136. https://doi.org/10.32628/cseit241061152

MadPenguin. (2024, December 14). *Why Database security is important? - Mad Penguin*. Mad Penguin. https://www.madpenguin.org/why-database-security-is-important/

Wikipedia. (2019, February 18). *Artificial Intelligence*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Artificial_intelligence

Wikipedia Contributors. (2019a, April 29). *Machine learning*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Machine_learning

Wikipedia Contributors. (2019b, October 2). *SQL injection*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/SQL_injection

Wikipedia Contributors. (2024, April 10). *Zero-day vulnerability*. Wikipedia. https://en.wikipedia.org/wiki/Zero-day_vulnerability