МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
Кафедра комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Жуков І.А
"_____"_____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"

ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Захист комп'ютерної мережі з використанням технології UEBA»

Виконавець: студент, КС-101Мз, Шелудько Роман Павлович

Керівник: Малярчук Василь Олександрович

Нормоконтролер: _____ Андрєєв Володимир Ілліч

Засвідчую, що у магістерській роботі немає
запозичень праць інших авторів
без відповідних посилань

Студент _____ Р.П.Шелудько

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж
Освітнього ступеня магістр
Напрям 123 "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри

_____ Жуков І.А.

"___" _____ 2020 р

ЗАВДАННЯ
на виконання дипломної роботи студента
Шелудька Романа Павловича

1. Тема роботи: «Захист комп'ютерної мережі з використанням технології UEBA» затверджена наказом ректора від « 25 » вересня 2020 р. № 1793/ст.

2. Термін виконання роботи: з 05.10.2020 р. до 10.12.2020 р.

3. Вихідні дані до роботи: Розробка програмного забезпечення та технологій для хмарних платформ. Використання сучасних підходів проектування та створення веб-сервісів та хмарних додатків. Вимоги до змісту та оформлення дипломних проектів студентів НАУ.

4. Зміст пояснювальної записки: дослідження технології UEBA як об'єкту застосування в інформаційних системах. Порівняльний аналіз існуючих технологій аналізу трафіку. Огляд та порівняння апаратного забезпечення для аналізу трафіку. Вибір інструментів та технологій розробки. Проектування та розробка прототипу програмного забезпечення для аналізу трафіку з використанням технології UEBA

5. Перелік обов'язкових слайдів презентації:

1) Тема, виконавець, керівник.
2) Існуючи методики, аналіз недоліків, постановка завдання.
3) Вимоги до програмного засобу.
4) Структура засобу, діаграма класів
5) Інтерфейс програмного засобу
6) Демонстрація роботи прототипу засобу

6. Календарний план-графік

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Отримання завдання на дипломну роботу. Дослідження технологій по аналізу трафіку	25.09.20	
2	Підбір та вивчення науково-технічної літератури за темою дипломної роботи	26.09.20 – 15.10.20	
3	Опрацювання матеріалу та порівняння технологій аналізу трафіку в мережах	16.10.20 – 31.10.20	
4	Аналіз впроваджених технологій	01.11.20 – 12.11.20	
5	Розробка практичної частини	13.11.20 – 01.12.20	
6	Оформлення пояснювальної записки	02.11.20 – 07.12.20	
7	Оформлення графічних матеріалів роботи та представлення роботи на кафедру	08.12.20 – 10.12.20	

7. Дата видачі завдання «25» вересня 2020 р.

Керівник дипломної роботи _____ Малярчук В.О.
(підпис)

Завдання прийняв до виконання _____ Шелудько Р.П.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Захист комп’ютерної мережі з використанням технології UEBA» містить: 91 с., 64 рис., 2 табл., 22 літературних джерела.

ТЕХНОЛОГІЯ МОНІТОРИНГУ МЕРЕЖІ UEBA, АНАЛІЗ МЕРЕЖЕВОГО ТРАФІКУ, МОДЕЛЮВАННЯ, СИСТЕМНИЙ ДИЗАЙН, ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ, ХМАРНІ ОБЧИСЛЕННЯ.

Об'єкт дослідження – система захисту комп’ютерної мережі.

Предмет дослідження – інформаційні процеси які виникають під час захисту мережі з використанням технології UEBA

Мета дипломної роботи – проектування та розробка системи моніторингу трафіку з використанням технології UEBA.

Методи дослідження – методи аналізу функціональних та нефункціональних вимог до архітектури програмного забезпечення, методи системного аналізу, методи оптимізації архітектури, методи математичної статистики та моделювання.

Технічні та програмні засоби – середовище проектування та програмування Visual Studio 2020 для написання коду, GitLab контролю версій та збірки веб-застосунку.

Результати роботи рекомендується використовувати для покращення реакції на інциденти в мережі та підвищення швидкості роботи системи, точності виявлення аномалій та інцидентів, кількість одночасних подій, та ін.

Прогнозні припущення щодо розвитку об'єктів розроблення – розроблювана система з використанням технології UEBA може бути використана у майбутньому шляхом застосування принципів описаних в цій роботі для покращення показників швидкості обробки аномалій та інцидентів в мережах.

Актуальність теми полягає у розробці системи захисту комп’ютерних мереж з використанням технології UEBA та відкритті нових можливостей виявлення аномалій в мережах.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ГІІ – глобальна інформаційна інфраструктура

*AJAX – Asynch JavaScript And XML (*асинхронний *JavaScript і XML)*

API – Applications programming interfaces (інтерфейси створення розподілених додатків)

CSS – Cascading Style Sheet (таблиці стилів)

CDN – *Content Delivery Network* – розподілена мережа доставки та розповсюдження контенту

DNS – *Domain Name System* ієрархічна розподілена система перетворення імені хоста в IP-адресу

DOM – *Document Object Model* (модель об'єкта документа)

Internet Exchange Point (IX, IXP) — Точка обміну інтернет-трафіком

MAC – Media Access Control (унікальний ідентифікатор для активного обладнання)

Npm – *Node Package Manager* (менеджер пакетів)

PaaS - *Platform as a Service* – платформа, як сервіс

Saas - *Software as a Service* – платформа забезпечення, як сервіс

IaaS – *Infrastructure as a Service* – платформа забезпечення, як сервіс

TE – *Traffic Engineering* – керування поступаючим трафіком

QoE – *Quality of Expirience* - оцінка якості сприйняття сервісу користувачем

QoS – *Quality of Service* - оцінка якості сервісу що поставляється

UEBA – *User and Entity Behavior Analytics* систему аналітики поведінки користувачів і сутностей

.NET Framework – програмна технологія, запропонована фірмою *Microsoft* як платформа для створення як звичайних програм, так і веб-застосунків.

ВСТУП

Згідно з недавніми звітами компанії *Gratner*, в період між 2018 роком і до 2023 року світ побачить трикратне збільшення підключених пристроїв до мережі Інтернет, що також вплине на зростання кількості атак на як мережеві пристрої так і на корпоративні мережі компаній в цілому. Велику кількість атак на мережі, стає все складніше покрити стандартними засобами безпеки.

Відсутність видимої загрози дозволяє компаніям ігнорувати загрози з боку зловмисників в мережі Інтернет, що робить сервіси компанії вразливими для просунутих зловмисників та нових видів загроз, що охоплюють як дані користувачів, так і фінансові активи компаній.

Технологія *UEBA* покращує систему виявлення та розслідування передових загроз на основі даних за допомогою аналізу поведінки користувачів, використання машинне навчання та поведінкове моделювання для визначити аномальної діяльності що мають високий ризик для інфраструктури компанії.

Процес створення продуктів з подальшим виведенням нових послуг на ринок для будь-яких компаній тісно зв’язаний з поширеними ризиками які виникають при створенні нового продукту та впровадженням нових технологій. Цей процес вимагає побудови сервісу з високим рівнем якості послуг, що надаються користувачам. Збій в роботі сервісів може привести до втрати багатьох клієнтів та значних фінансових та репутаційних втрат.

Таким чином актуальність теми полягає у тому, що діяльність компанії стає все більш вимогливою до високої ефективності та значущою для розвитку бізнесу, а будь-який збій в роботі інфраструктури (активному мережевому обладнанні чи каналах зв'язку.) може стати як фінансовими так і репутаційними збитками та незадоволеними клієнтами.

Тому виникає проблема, яка полягає у тому що постійно з’являються нові тип загроз для мереж. Для того щоб захистити мережу, необхідно інструмент який може автоматично підлаштовуватись під нові типи загроз та автоматично

виявляти аномалії в мережі. Для цього пропонується розробка нової системи аналізу трафіку та інтеграція технології *UEBA* що є одною із найпотужніших технологій забезпечення безпеки роботи бізнесу та забезпечить точний аналіз дій користувачів, а також виокремлення трафіку користувачів від роботів-зловмисників, що значно підвищить ефективність роботи системи захисту.

В даній роботі об'єктом дослідження була система захисту комп'ютерної мережі. Предмет дослідження в дані роботі це інформаційні процеси які виникають під час захисту мережі з використанням технології *UEBA*. Метою даної роботи є проектування та розробка системи моніторингу трафіку з використанням технології *UEBA*.

В якості методів дослідження обрано системний аналіз відомих підходів та методів, систем та алгоритмів з автоматизації сервісів, порівняння отриманих даних, аналізу функціональних та нефункціональних вимог до побудови архітектури мережі, методи покращення оптимізації системи, методи математичної статистики.

Наукова новизна одержаних в роботі результатів і практичне їх значення полягає у тому що розроблена система аналізу трафіку має високу швидкість обробки вхідних даних, проста у використані, а інтерфейс системи дає можливість користувачам не проходити додаткове навчання та задовольняє оснівні потреби користувача..

Базові дії при роботі з інтерфейсом додатку, такі як, вхід до системи, обробка аномалій, інцидентів та моніторинг подій - інтуїтивно зрозумілі. Для подій з критичним пріоритетом передбачена системі інформування адміністратора по *SMS*.

Реалізована система з використанням технології *UEBA* має порівняно низьку вартість. Система потребує мінімальної модифікації апаратного та програмного забезпечення. Результатом є те, що розроблена система та її архітектура може використовуватись для проектів з відносно низьким бюджетом.

РОЗДІЛ 1
АНАЛІЗ СИСТЕМИ ПОВЕДІНКОВОЇ АНАЛІТИКИ UEBA

1.1. Аналіз інформаційних систем та хмарних технологій

Інформаційна система – це система яка є взаємозв'язаною сукупністю засобів та методик які використовуються для видачі, зберігання та обробки різного виду інформації. Таку інформацію використовуються для досягнення різного типу мети як в корпоративних мережах так і в мережі інтернет.

Сучасне розуміння інформаційної системи допускає що технології *UBA*, *UEBA* будуть використані в якості основного технічного засобу обробки та інформації отримані від користувача. Крім того, технічна реалізація ІС з використанням технології *UEBA* собі майже нічого не означати, якщо не врахована концепція система-користувач, для якого призначена вироблена інформація і без якого неможливе її одержання і представлення.

Найважливішими функціями системи на базі інтелектуального аналізу поведінки користувача в системам *UEBA* є прогнозування витоків інформації, облік та моніторинг аномалій, аналіз інцидентів, контроль і регулювання дій користувачів в мережі.

Раніше інформаційні системи та технології *SIEM* використовувались у мережах корпоративного типу чи у локальних мережах, компаніях середніх та великих розмірів. Зараз же такі технології стають більш доступнішими, тому знаходять використання навіть у невеликих компаніях та стартапах на початкових етапах розвитку.

Впровадження і активне застосування *SIEM* систем і *UBA* технологій дозволяє значно економити час та ресурси співробітників.

Інформаційна система призначена для обміну даними між модулями в мережі, а також з системами у навколишньому середовищі(глобальні мережі Інтернет). В процесі роботи користувачі та співробітники одержують інформацію та документацію про стан системи та окремих її модулів, дані про інциденти та аномалії які можуть негативно впливати на систему чи навіть повністю паралізувати роботу як користувачів так і бізнесу.

Інформаційна система – це поєднання внутрішніх і зовнішніх даних і чітко визначених модулів, об'єктів, засобів, користувачів, що беруть участь у генерації інформації та продукуванні управлінських рішень в рамках корпоративної мережі.

В наш час значна кількість інформаційних систем організуються на основі автоматизованих засобів обчислювальної техніки які часто поєднані в хмари чи ферми.

У відповідності з джерелом [4], в якому була розглянута класифікація ІС можна зробити розподілення інформаційних систем по 6 базовим ознакам, таких як:

– масштабність;
– рівнем розподіленості зберігання даних;
– типом апаратних і програмних засобів;
– областю застосування;
– способом організації архітектури;
– типом або сферою діяльності.

За розмірами інформаційні системи поділяються наступні групи: одиничні та групові, корпоративні та глобальні.

Одиничні ІС (рис.1.1) використовуються, часто на звичайному ПК без підключення до комп’ютерної мережі чи з’єднання з мережею інтернет. Така система може складатися лише з декількох простих додатків із спільною інформацією на декілька користувачів. Подібні мережі створюються часто за

допомогою таких мінімального набору програмного забезпечення, такого як *Nginx* веб-сервер, чи *MongoDB*.

Рис.1.1. Класифікація ІС за масштабністю

В свою чергу групові інформаційні системи орієнтовані на колективне користування інформацією і зазвичай будуються на базі локальної корпоративної мережі. Для розробки додатків такого типу в основному використовуються сервери БД *SQL* та віртуальні сервери. Серед найбільш розповсюджених серверів є Amazon, *Mysql*, *MariaDB*, тощо.

Корпоративні інформаційні системи використовуються у великих мережах і як наслідок у великих компаніях та можуть підтримувати віддалені вузли в різних центрах обробки інформації чи різних точках планети. Як правило корпоративні мережі мають ієрархічну клієнт-серверну арзітектуру де переважають сервери зі значними обчислювальними ресурсами.

Для корпоративних систем поширеними рішеннями є сервери з використанням такого ПЗ як *Microsoft SQL Server, Oracle* та інколи *DB2*.

Глобальні інформаційні системи охоплюють територію держави, континенту чи, частіше, усієї планети. Прикладом такої глобальної ІС є глобальна мережа Інтернет.

За ступенем централізації розділяють такі інформаційні системи: централізовані інформаційні системи, децентралізовані інформаційні системи та інформаційні системи колективного використання.

1.2. Апаратне забезпечення інформаційної системи

Апаратне забезпечення - електронні та механічні частини обчислювального пристрою, які є частиною системи або мережі (програмне забезпечення та дані, що обробляються системою, не є апаратними).

Апаратне забезпечення включає: електронні схеми (арифметичні, логічні, цифрові та аналогові), реалізовані у вигляді різних електронних приладів та приладів, пристрої вводу-виводу, схеми та компоненти живлення (акумулятори, перетворювачі напруги та струму), діагностичне та випробувальне обладнання, пасивне компоненти (шасі, корпуси, стійки, комп'ютерні роз'єми тощо). На рис.1.2 представлено схему комп'ютерної мережі IT-компанії.

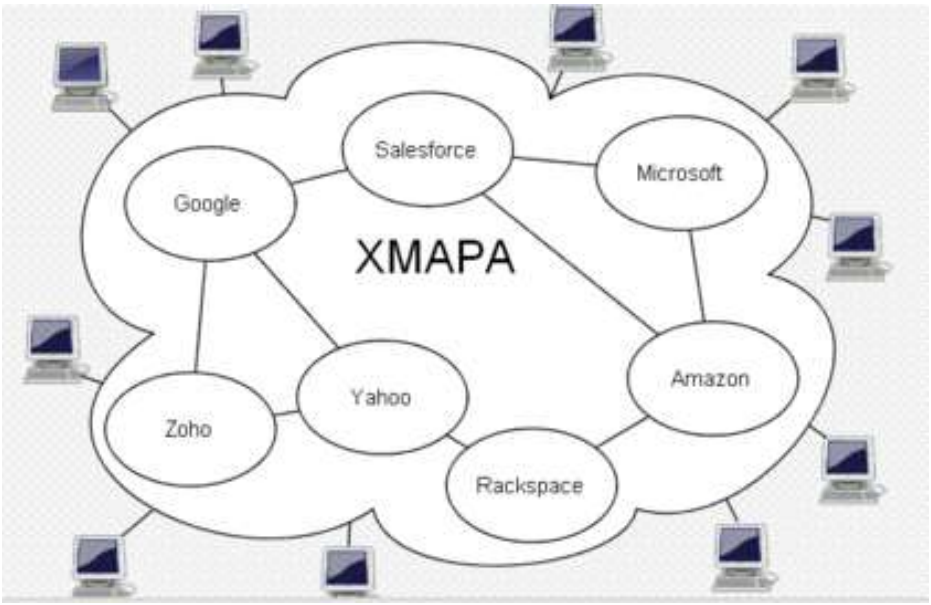


Рис.1.2. Схема організації комп'ютерної мережі IT-компанії

Дана мережа гарантувала порівняно низький рівень помилок та високу інфраструктуру доступності. Однак навіть на момент проектування був ряд недоліків. Зокрема, використання *IP*-адреси, розробленої для 254 пристроїв, - це замало для великої компанії. Відсутність поділу на *VLAN*, перетин IP-трафіку з тим трафіком який надходив на інших вузли у задані мережі, використання одного провайдера, - все це вплинуло на доступність системи в цілому.

Проектування - один з найважливіших етапів у процесі побудови обчислювальної техніки, оскільки такі фактори, як ефективність мережі, надійність, безпека та, звичайно, вартість, залежать від правильної архітектури мережі.

1.3. Технології аналізу поведінки користувачів

Системи аналізу поведінки користувачів (*User Behavior Analitycs - UBA*) - це комплекс рішень, які призначені для збору та аналізу всіх дій користувачів, включаючи дані, які обробляються користувачем, з метою визначення внутрішніх загроз і атак, а також фінансового шахрайства.

Одним із завдань, які вирішують системи поведінкового аналізу являється виявлення компрометації облікових записів. Технології машинного навчання *UBA* і аналітики дозволяють створювати профіль кожного користувача на основі його дій в системі. У разі компрометації облікового запису, поведінка зловмисника буде різко відрізнятися від поведінки власника облікового запису. В даному випадку системи аналізу поведінки будуть сигналізувати про аномальну поведінку.

Так, наприклад, для роботи додатків часто використовуються системні облікові записи, які володіють розширеними правами на роботу в системі. У разі, коли така обліковий запис потрапляє під контроль зловмисника, він буде діяти, відхиляючись від створеного системою *UBA* профілю.

Основним завданням технології *UBA* є слідкування за зловживанням правами привілейованих облікових записів. Зазначена можливість систем *UBA* перетинається з рішеннями для контролю привілейованих облікових записів (*PAM*). Різниця полягає в тому, що системи *PAM* ведуть моніторинг сесій і здійснюють збір даних, в той час як системи аналізу поведінки дозволяють під час виявити і попередити про відхилення поведінки таких користувачів.

Запобігання витоку конфіденційної інформації в системах з технологією *UBA* використовуються для вирішення завдань пов’язаних з інсайдерською діяльністю. В організаціях використовують, як правило, *DLP*-рішення, які

відстежують канали витоку інформації. Але в разі інтеграції *UBA* з системами *DLP* рішення самостійно проаналізує поведінку користувача і попередить про можливий витік (наприклад, в разі раптової відправки великих файлів через електронну пошту або копіювання даних на *USB*-носії).

Виявлення підозрілого часу підключення одна з найдієвіших функцій в системах з технологією *UBA*, так як більшість співробітників компаній працюють по конкретному графіку і приходять в один і той же час, та йдуть точно в один і той же час. Коли співробітник є інсайдером і планує скопіювати важливу інформацію зі свого комп'ютера для передачі третім особам, він може залишитися на роботі допізна або прийти раніше початку робочого дня, для того щоб інші співробітники не змогли звернути увагу на те, чим він займається. Системи поведінкового аналізу зможуть відстежити аномальну активність і передати дані співробітникові, відповідальному за безпеку.

Виявлення спроб спільного використання облікових записів це коли одні співробітники діляться своїми паролями з іншими, наприклад, в разі виходу у відпустку. Передача облікових записів несе в собі певні ризики. У разі компрометації облікового запису неможливо буде відстежити, хто відповідальний за витоку конфіденційної інформації. В даному випадку при отриманні даних з *SIEM*-систем і систем контролю доступу, рішення *UBA* дозволять виявити факт порушення правил використання облікових записів.

При детальному налаштуванні систем поведінкового аналізу така система як виявлення помилок налаштування прав доступу не дозволить користувачам, які не повинні мати права на певні дії, виконувати їх. Розвитком систем аналізу поведінки користувачів є системи аналізи поведінки користувачів і об'єктів (*UEBA - User and Entity Behavior Analytics*). Такі системи виконують моніторинг і аналіз не тільки поведінки користувачів, а й об'єктів інфраструктури.

User and Entity Behavior Analytics (*UEBA*), або поведінкова аналітика користувачів і сутностей в мережі - новий тренд, активно набирає обертів на ринку інформаційної безпеки. *UEBA* забезпечує профілювання і виявлення аномалій в поведінці користувачів та інших сутностей.

Протягом довгого часу виробники програмних продуктів створювали рішення, відштовхуючись від технологій які вже добре зарекомендували себе на ринку, ті які дозволяли вирішувати повний спектр завдань. Але з часом вендори часто намагалися видавати свої рішення за щось більше, ніж вони були в реальності. Можливо, прикладом такого рішення і є *User and Entity Behavior Analytics* (*UEBA*), або поведінкова аналітика користувачів і сутностей - новий тренд, активно набирає обертів на ринку інформаційної безпеки.

Gartner пояснює його суть наступним чином: *UEBA* забезпечує профілювання і виявлення аномалій в поведінці користувачів і сутностей (*entities*). До останніх відносяться робочі станції, додатки, мережевий трафік, сховища даних та інше. Для виявлення аномалій використовуються різні аналітичні методи (статистика, відповідність паттернам, машинне навчання). Підхід передбачає, що незвичайна активність, яка не відповідає стандартному профілю, може більш точно вказувати на потенційні інциденти інформаційної безпеки. Теоретично, в рамках *UEBA* технології можна виділити таки види подій:

– *Incident Management* – модуль, який відповідає за комплексний підхід до вирішення інцидентів. В системі де використовується технологія *UEBA* реєструються інциденти і час їх виправлення;

– *Change Management* – це осмислені зміни у технології *UEBA* і узгодження їх реалізації як з бізнесом так і з усіма користувачами бізнес-сервісів;

– *Release Management* – це процес, в основу якого покладено завдання не порушувати роботу компанії при виконанні будь-яких оновлень системи чи виконання змін в процесах. Система керування оновлень спостерігає і виконує оновлення ПЗ і активного обладнання через додатки в службі *UEBA*.

– *Problem Management* – це процес, задача якого полягає в забезпеченні низької кількості інцидентів, що реєструються системою. Для цього виявляються причини виникнення проблем та аномалій та усуваються їх причини;

– *Quality Service Management* – модуль в системі, який визначає кількість співробітників та їх активність, що задіяні в роботі компанії, а також регулюють на контролюють якість роботи системи *UEBA*. З його допомогою відбувається

моніторинг рівня якості роботи всіх модулів і проводяться роботи по зниженню вірогідності того, що можуть не точно визначатися аномалії та події.

Management of Finance – процес, який регулює розпорядження та контроль фінансових потоків для забезпечення діяльності відділку продаж, бухгалтерія та інші комерційні підрозділи.

Availability Management – система яка виконує контроль доступності IT підрозділу в цілому. Тут максимально стараються виділити всі процеси в окремий модуль чи кластер, щоб їх була можливість контролювати і генерувати звіти окремо від всіє системи. Рівні доступності на надійності визначаються такими властивостями як постійність, ремонтопригоднійсть та надійність.

Information Security Management – процеси що забезпечують безперервність забезпечення безпеки сервісу та надійність збереженні інформації.

1.4. Системи аналізу поведінки користувачів

В даний час на ринку систем з використанням технології *UBA* чи *UEBA* існує невелика кількість продуктів. *OpenSource* продукти є, але вони значно поступаються якістю так кількістю функцій порівняно з їх комерційними конкурентами. Загалом існуючі продукти поширюються за технологією *SaaS* та розраховані на великі компанії, але без проблем можна інтегрувати і в компанії будь-яких розмірів. Вартість продуктів порівняно висока, складність інтеграції висока та займає досить багато часу. При цьому якість та точність роботи таких систем не відрізняються стабільністю.

Система *Exabeam* – потужне програмне забезпечення, але воно має бути тільки одним з компонентів добре підібраного набору інструментів мережевої безпеки поряд з іншими інструментами, які надає компанія в цілому (рис.1.2).

Майже всі найбільші витоки даних, зачіпають державні і приватні підприємства, та відбуваються коли хтось краде облікові дані авторизованого користувача і потім використовує ці облікові дані для крадіжки даних.

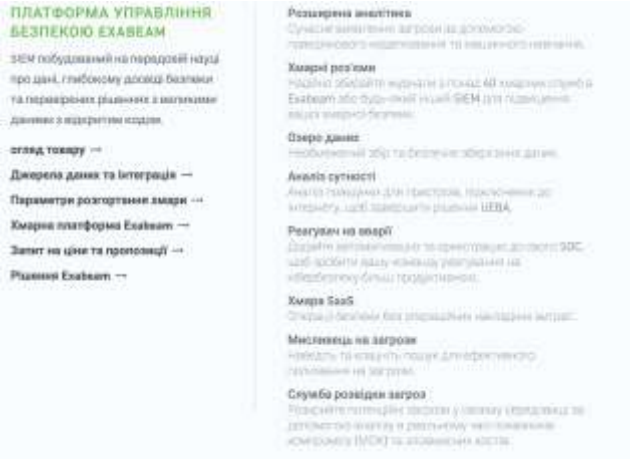


Рис.1.3. Набір доступних продуктів в системі *Exabeem*

В ході недавніх порушень, на такі компанії як *Target* чи *Sony* співробітники бачили, що атака сталася, але, на жаль, ніхто не помітив сили атаки чи її глибини.

SIEM у *Exabeem* побудований на передовій науці про дані, глибокому досвіді безпеки та перевірених рішеннях з великими даними з відкритим кодом.

Платформа *Exabeam User Behavior Intelligence Platform* (вартість якої починається від 25 000 доларів США) призначена для збору інформації з різних джерел, включаючи *Active Directory (AD)* і програмне забезпечення і пристрої управління інформацією про безпечність та подіями (*SIEM*), а також для повідомлення про підозрілу поведінку в своєчасно.

Exabeam, який може поставлятися як на фізичному носію або як віртуальний додаток, – працює шляхом вивчення історії подій вашої організації, щоб визначити, що є нормальним, а потім вивчення подій, які відхиляються від нормальних.

Exabeam також має вартість підписки, яка варіюється в залежності від кількості що відслідковуються користувачів і пристроїв. Якщо функціональність, за якою спостерігає *Exabeem* спеціалізована, то вартість може зростати.

Функції *Exabeam*:

– потужна аналітика;

– можливість збирати дані та аналітику з понад 40 хмарних сервісів;

– система для збору та зберігання необмеженого об’єму даних;

– потужний, простий і сучасний користувальницький досвід;

– аналіз поведінки для пристроїв, підключених до Інтернету;

– портал для клієнтів з документацією для навчання;

– автоматизований контроль аварій;

– хмарне рішення *SaaS*;

– пошук можливих загроз в мережі Інтернет;

– потужні можливості для фільтрації та пошуку інформації в систему;

– понад тисячу надбудов;

– база знань;

– автоматизована служба розвідки в реальному часі.

Exabeam Advanced Analytics – це найбільш розгорнуте рішення безпеки *UEBA* у світі. Сучасне виявлення загроз за допомогою поведінкового моделювання та машинного навчання допомагає проводити складну ідентифікацію загроз за допомогою поведінкового аналізу.

Так як кібератаки стають все більш складними і їх важче знайти. Часто правила кореляції не можуть знайти атаки, оскільки їм бракує контексту або пропускаються випадки, яких система аналізу ніколи не бачила, що в свою чергу породжує помилкові спрацювання та негативні наслідки.

Правила кореляції також вимагають значного обслуговування. *Advanced Analytics*, – рішення безпеки Exebeam *UEBA*, автоматично виявляє поведінку, яка свідчить про загрозу. Система повністю інтегрується з розвідувальними службами *Exabeam Threat Intelligence Services* (*TIS*), щоб забезпечити діючу інформацію в реальному часі щодо потенційних загроз у середовищі компанії шляхом виявлення показників скомпрометованих даних (*IOC* – індикатор компрометації) та зловмисних хостів.

Заздалегідь розроблені заходи безпеки автоматично оброблюють інциденти та автоматично відновлюють систему.

Рис.1.4.Поведінкова аналітика користувача в системі *Exabeem*

Аналітики не повинні витрачати дні чи тижні на збір доказів та побудову історії інциденту шляхом запитів та обробки через їх *SIEM*. За допомогою *Advanced Analytics* попередньо вбудована шкала ризику подій позначає аномалії та відображає деталі інциденту для повного обсягу події та її контексту.

Таким чином аналітики можуть припинити витрачати час на вивчення необроблених журналів для розслідування інцидентів. Те, що може займати тижні для вивчення, тепер це можна зробити за лічені секунди за допомогою рішення безпеки *UEBA*.

За допомогою системи *Exabeem* також можна розширити поведінкову аналітику для об’єктів хмарного сховища. Організації зберігають свої дані в хмарі, щоб використовувати масштабованість, безпеку та продуктивність служби зберігання об’єктів. Однак хмарне зберігання даних було причиною багатьох порушень. Це тому, що помилки конфігурації можуть легко викрити конфіденційні дані та не виявитись організацією.

Exabeam Advanced Analytics реєструє активність із хмарних об’єктів зберігання у багатохмарних середовищах, а саме *Amazon S3*, *Azure Blobs* та *Google Cloud Platform Cloud Storage*, та створює поведінкові моделі, щоб впевнено ідентифікувати шкідливу активність користувача за звичайною поведінкою користувачів.

Exabeam оброблює можливі ризики, що властиві хмарним сховищам та базам даних що можуть бути ненавмисно доступні з глобального Інтернету.

Система аналітики безпеки та інструментів безпеки часто ускладнює співпрацю між системою та користувачем під час виявлення та розслідування загроз. Структура *Exabeem* вирішує цю проблему, надаючи загальну структуру, яку аналітики можуть використовувати для опису тактики та техніки поведінки зловмисника.

Advanced Analytics поєднує методи виявлення *Exabeam* та мітки подій з різних модулів системи, що дозволяє аналітикам безпеки переглядати та фільтрувати методи в рамках розумних шкал часу *Exabeam*.

Наприклад, аналітики можуть навести мишу на ярлики, щоб з'явився спливаючий опис техніки злому, або натиснути на ярлики, щоб відкрити веб-сторінку із детальним описом цієї техніки. Управління операціями дорогим, так як включає організацію ресурсів та визначення пріоритетів інцидентів, крім розслідування та пом'якшення наслідків, що впливають на бізнес.

Рис.1.5. Щоденна аналітика інцидентів та загроз *Exabeem*

Іншим больовим моментом є відсутність кваліфікованих аналітиків для аналізу та визначення пріоритетів інцидентів. Час, необхідний для швидкого вирішення інцидентів, впливає на результат. За допомогою рішення безпеки *UEBA* від *Exabeam* можна автоматизувати ці завдання, щоб зменшити середній час до вирішення (*MTTR*), дозволяючи вашому вже розтягнутому персоналу охорони зробити більше за менший час.

Exabeam Case Manager повністю інтегрований в *Advanced Analytics*, що дозволяє оптимізувати робочий процес аналітиків та гарантувати, що жодні загрози не проскакують через тріщини. Динамічне групування користувачів це

система в основі якої закладені моделі поведінки користувачів, які часто відрізняються залежно від безлічі атрибутів, зокрема: команда, в якій вони працюють, в яких проектах вони беруть участь, де вони знаходяться тощо. Таким чином, поведінкові базові показники не повинні бути статичними.

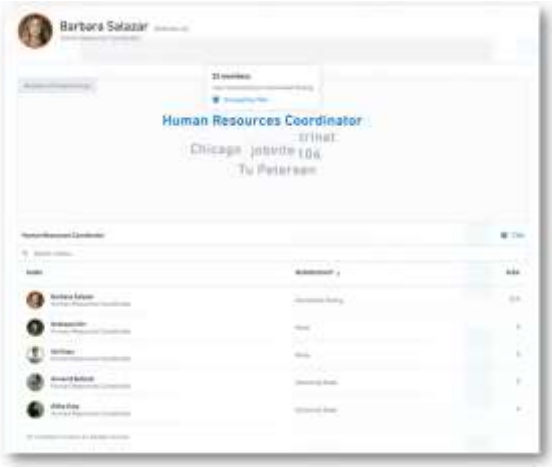


Рис.1.6. Моніторинг активності користувача на основі груп

Виявлення побічної активності користувачів (рис. 1.6) в системі це набір методів, які використовуються для перевірки переміщення зловмисників по мережі за допомогою IP-адрес, облікових даних та машин у пошуку ключових алгоритмів.

Відстеження подібних активностей складне, оскільки інформація про пересування зловмисника розповідає лише частину історії. Дані потрібно аналізувати звідусіль, пов'язуючи атаку з джерелом. Технологія *Advanced Analytics* відстежує підозрювані дії, навіть якщо в пристрої, *IP*-адресах або облікових даних є зміни.

Рис.1.7. Виявлення побічної активності користувачів в системі *Exabeem*

Одним із модулів системи *Exabeem* є *Exabeam Entity Analytics* – *п*оведінкова аналітика для *IT*, *IoT* та *OT* пристроїв та для об'єктів що зберігаються в хмарах.

Так як зловмисники пересуваються через мережу, між користувачами, машинами та активами в пошуках цінних даних, то такі підключені активи, як медичне обладнання, техніка та інфраструктура електромереж, разом із хмарними активами можуть стати легкою мішенню. Технічні активи вимагають того ж моніторингу, що і люди, оскільки вони часто контролюють виробництво та діяльність і можуть зберігати конфіденційні корпоративні дані, включаючи особисту інформацію, дані клієнтів або дані *API*.

Entity Analytics встановлює базову поведінку, використовуючи шаблони зв'язку, порти та протоколи та операційну діяльність - автоматично визначаючи нерегулярні дії сутності, що свідчать про інцидент чи аномалії.

Окремим модулем забезпечення безпеки є попередньо складені набори подій. Ідея полягає в тому, що для всіх аномалій, виявлених *Entity Analytics*, – *Exabeam* що вбудований апаратне забезпечення збирає метрики інцидентів порівнюючи як нормальну, так і ненормальну поведінку машин, активів, пристроїв *IoT* та об’єктів хмарного сховища.

Так, на відміну від конкурентних рішень *Exabeem UEBA* детально описує те, що сталося під час інциденту, та визначає поведінковий контекст, щоб визначити, чи була діяльність нормальною - зменшуючи ручні зусилля під час збору доказів.

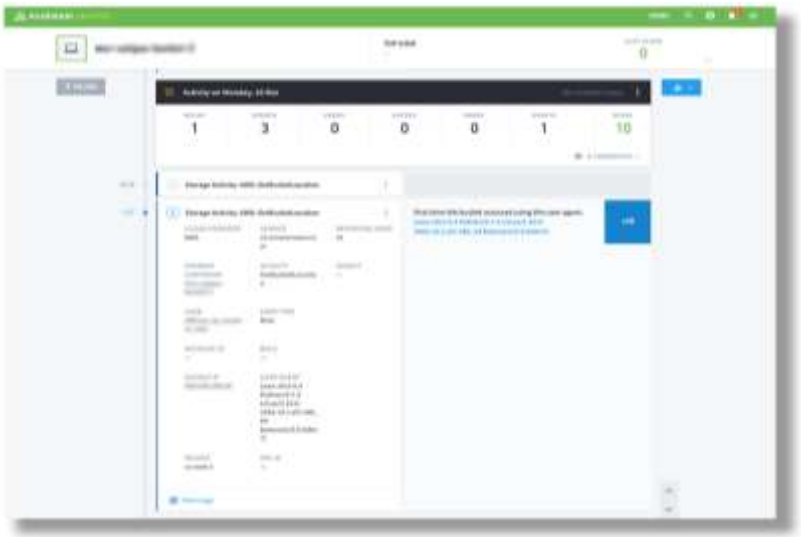


Рис.1.8. Аналітика роботи хмарного сховища *S3* в *Exabeem*

Наскрізна видимість мережі. Незалежно від того, здійснюється моніторинг локальної мережі або активів всієї інфраструктури, дані, які розглядаються окремо від всієї системи, можуть здаватися доброякісними, хоча насправді не бути такими.

Рис.1.9. Наскрізний моніторинг роботи мережі в *Exabeem*

Exabeam поєднує та аналізує журнали з різних джерел, включаючи *VPN*, хмарні програми, служби електронної пошти, брандмауери, *NetFlow* та інші специфічні датчики *IoT*. Потім машинне навчання та моделювання поведінки, що лежать в основі рішення *UEBA*, використовуються для виявлення складних загроз, які в іншому випадку не виявляться.

Система *Securonix* – це платформа яка пропонує систему аналітики та управління операціями наступного покоління для сучасної ери великих даних та передових кіберзагроз. Набір доступних функцій налічує більше 20 модулів (рис.1.10).

Рис.1.10. Продукти та рішення які пропонує компанія *Securonix*

Сьогодні багато атак спеціально побудовані, щоб уникнути традиційних

методів захисту на основі підписів, таких як збіг хеш-файлів та шкідливі списки доменів. Вони використовують низькі та повільні тактики, що виконуються лише з часом. Це дозволяє непомітно виконувати зловмисний код невидимо для адміністратора мережі.

Securonix UEBA використовує машинне навчання та аналітику поведінки для аналізу та кореляції взаємодії між користувачами, системами, програмами, *IP*-адресами та даними. Система швидка, легка, не потребує багато ресурсів і швидко розгортається. *Securonix UEBA* виявляє розширені інсайдерські загрози, кіберзагрози, шахрайство, компрометацію хмарних даних та недійсність користувачів.

Сьогодні напади на великі компанії розроблені таким чином, що їх важко виявити. Такі атаки вимагають розуміння базової поведінки користувачів, щоб можна було виявити аномальну активність всередині та за межами організації. *Securonix UEBA* виявляє невідомі, загрози нульового дня та вдосконалені постійні загрози, використовуючи заздалегідь створений вміст загроз, орієнтований на інсайдерські загрози, шахрайство та інші ключові випадки використання.

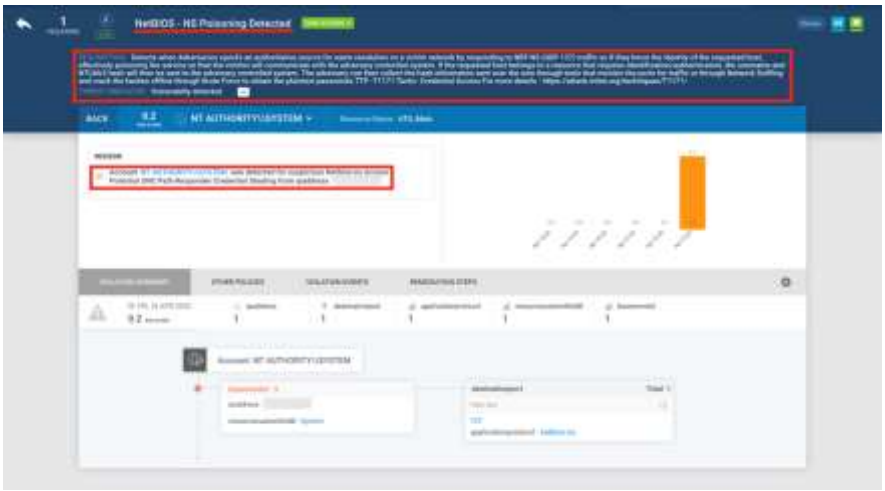


Рис.1.11. Вікно аналітики аномалії в *Securonix*

Сутність такого створює вичерпні профілі ідентичності та ризику для кожного користувача та організації у вашому середовищі.

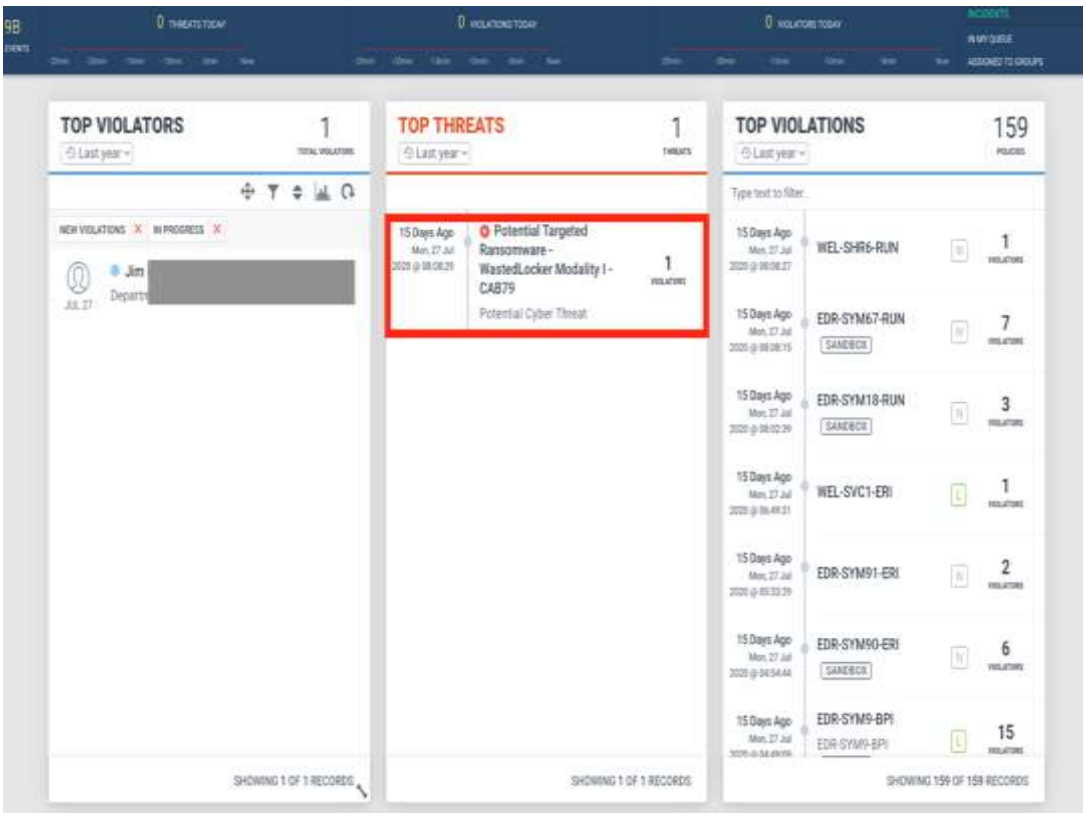


Рис.1.12. Вікно моніторинга проникнень та загроз в *Securonix*

Сьогодні більшість підприємств підприємств працюють з *SAP*, і протягом багатьох років *SAP* перетворилась на складну, всеохоплюючу систему, яка керує безліччю бізнес-додатків і є базою для великих обсягів критичних та конфіденційних даних.

Система *Netskope* – це постачальник рішень для забезпечення безпеки в корпоративній мережі, який пришвидшує цифрову трансформацію компанії завдяки перевіреній платформі безпеки, орієнтованій на дані, інтелектуальну обробку даних у хмарі та

Advanced Analytics - це рішення від *Netskope* для аналізу бізнес-аналітики та аналізу великих даних, що надає організаціям максимальний контроль своєї інфраструктури, користувачів та даних. *Netskope* дає доступ до інформації про використання додатків в мережі, її ризики та ефективність. За допомогою *Advanced Analytics* можна визначити тенденції, не враховуючи проблемні сфери, і не заглибитися в деталі.

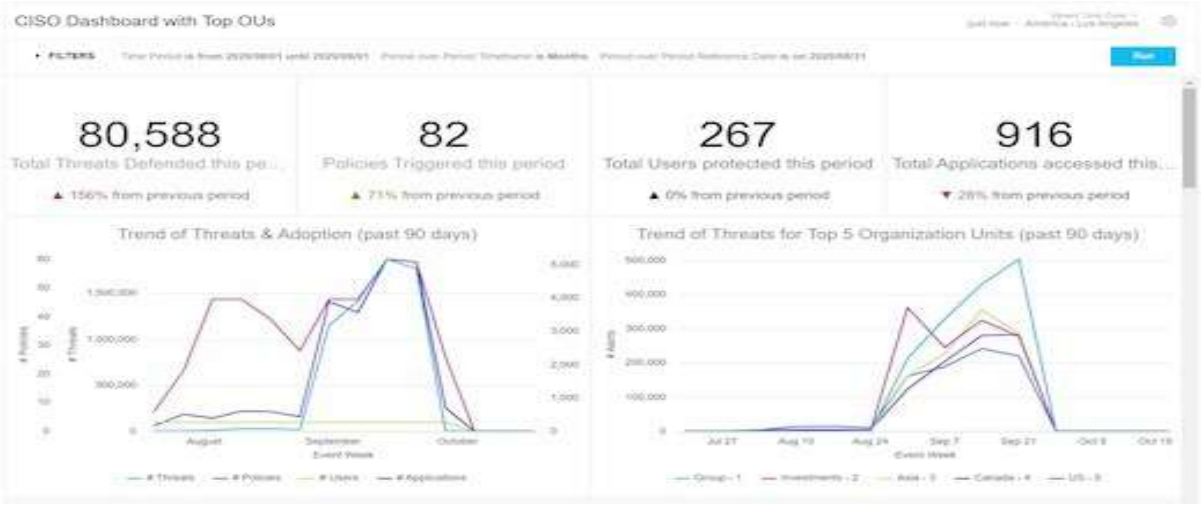


Рис.1.13. Головна сторінка для моніторингу інфраструктури у *Netskope*

В системі *Netskope* реалізовано модуль, де є попередньо визначений набір інструментів, який призначено для управління аномаліями та інцидентами. Таку панель інструментів можна доналаштувати відповідно до потреб компанії, куди *Netskope* було інтегровано. Модуль допомагає командам безпеки та бізнесу контролювати мережу та надавати оцінку захищеності інфраструктури усім зацікавленим сторонам.

Рис.1.14. Головна сторінка для моніторингу інфраструктури у *Netskope*

Додатки, до яких здійснюють доступ користувачі організації, можуть змінюватися постійно, тому потрібно постійно оцінювати те, що відбувається в мережі, відстежувати інциденти та ризики, приймати та оброблювати заявки,

визначати потенціал ризику та впроваджувати відповідні політики безпеки для безпечного прийняття рішень до ваших бізнес-потреб. Для таких задач в системі *Netskope* реалізовано модуль для контролю та моніторингу загроз, які можуть бути спрямовані на хмарні ресурси компанії (рис. 1.14).

Компанія *Netskope* пропонує досить багато *UBA* можливостей для забезпечення безпеки компанії, які розміщують свої ресурси та бази даних в хмарах, але все ще працює над функціоналом *UEBA* та пропонує лише базові модулі для поведінкового аналізу користувачів в мережі.

Система *Forcepoin* – це провідна компанія з забезпечення кібербезпеки для користувачів та захисту даних, яка орієнтована на організації різного рівня. Рішення *Forcepoint* в режимі реального часу адаптуються до того, як люди взаємодіють з даними, забезпечуючи безпечний доступ, дозволяючи працівникам компанії мінімально турбуватись про безпеку в мережі (рис1.15).

Рис.1.15. Список продуктів та можливостей системи *Forcepoin*

Forcepoint – забезпечує порівняно великий набір функціоналу та високу швидкість роботи з вхідними даними. Система, під час аналізу, застосовує сучасні підходи до аналізу загрози та аналізу поведінку для вдосконалення рішень для кіберзахисту.

Технологія динамічного захисту користувача (*DUP*) в рамках *Forcepoint* – це це хмарне рішення для моніторингу активності користувачів, яке використовує підхід Індикатори поведінки (*Indicators of Behavior*) - багатофункціональний підхід до моніторингу, який аналізує дії користувачів - для зменшення та виявлення ризику в ранніх точках (рис.1.16.).

Рис.1.16. Список продуктів та можливостей системи *Forcepoin*

Динамічний захист даних – це модуль в системі *Forcepoint,* який призначений для підвищення ефективність *DLP(Data Leak Prevention)* за допомогою поведінкової аналітики користувачів (рис.1.17.).

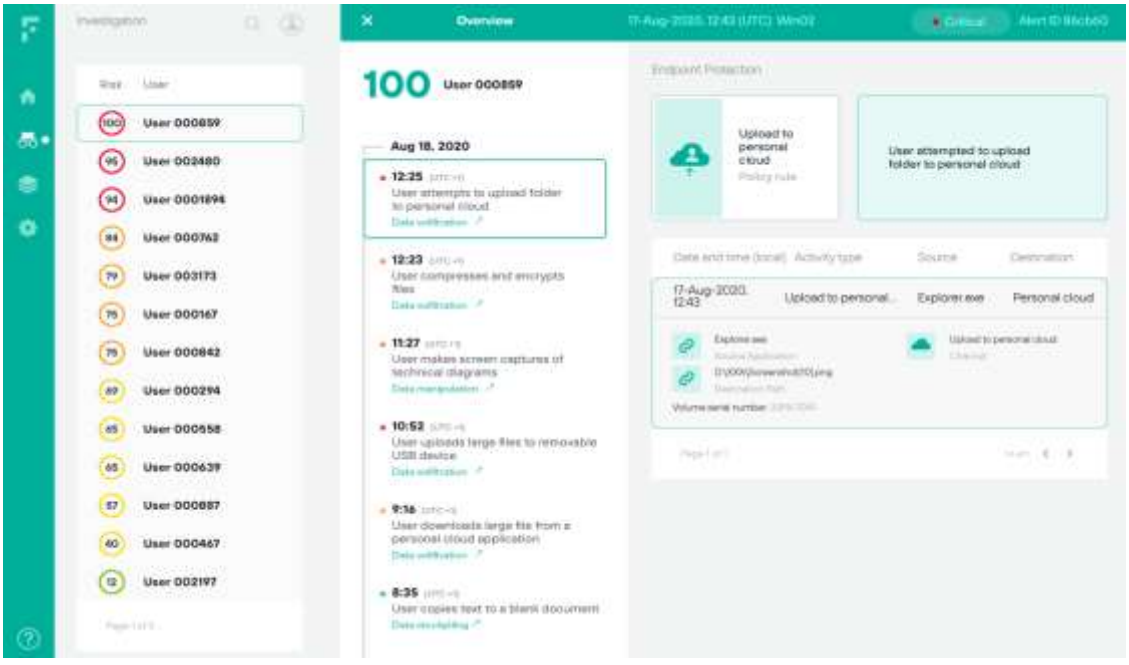


Рис.1.17. Графічний інтерфейс модулю динамічного захист даних в *Forcepoint*

В цілому, модуль дозволяє змінити широку, всеохоплюючу безпеку на індивідуальну адаптивну політику по контролю кожного користувача, яка не сповільнює роботу співробітника.

Використання комплексної поведінкової аналітики для створення унікальної оцінки ризику для кожного користувача дозволяє застосовувати детальну політику індивідуально, залежно від автоматичної оцінки ризику окремого користувача.

1.5. Аналіз проблем в системах поведінкової аналітики

Всі розглянуті рішення мають певні відмінності в архітектурі, політиці ліцензування, функціональних можливостях та аналітичних методах. Тим не менше, кожен з них відповідає основним вимогам: запобігання витокам чутливої інформації в організації.

Тож тому для аналізу проблем були запропоновані такі параметри порівняння, за допомогою яких можна зробити висновок про зміни та використання тих чи інших рішень. Безумовно, ідеальні засоби не існують, і кожна конкретна організація обирає для себе ряд найбільш важливих для неї критеріїв, за результатами аналізу яких і буде укладено висновок про введення того чи іншого продукту.

Порівняння систем які використовують технології *UBA* та *UEBA* здійснено за 39-критеріями і вказано в табл.1.1. Проаналізувавши можливості зазначених вище систем, було прийнято рішення про розробку своєї системи з використанням технології *UEBA* для моніторингу мережі та аналітики аномалій. Це ґрунтується на тому, що розглянуті системи є перевантаженими як с точки зору функціоналу так і інтерфейсу, а їх використання є економічно не доцільним в малих та середніх за розмірами компаніях.

Таблиця 1.1

Порівняльний аналіз систем з технологією *UEBA*

з/п	Критерій	Назва системи			
Дані		*Exabeam*	*Securonix*	*Netskope*	*Forcepoint*
	Вартість	Ціна публічно не розголошується та формується відповідно до потреб клієнта. Орієнтовна вартість від 25 000$-75 000$.	Ціна публічно не розголошується та формується відповідно до потреб клієнта.	Від 8$ за користувача за місяць в рамках знайомства з системою. Ціна формується відповідно до потреб клієнта.	Від 50$ за користувача в рамках в місяць. Ціна формується відповідно до потреб клієнта.
	Інтеграція з іншими сервісами	70+ сервісів, з них *Amazon, Microsoft, Google, Cisco, IBM.*	*Octa, Demisto, Amazon, Microsoft, Google.*	*Microsoft, Amazon, Dropbox, Google, Cisco, IBM.*	*26* сервісів, з них *Amazon, Microsoft, Google, Cisco, IBM, Okta.*
3	Платформи	Web, Android, iOS.	Web, Android,	Web	Web,
4	Розгортання	Сервер Хмара	Хмара	Хмара	Сервер Хмара
5	Доступні мови	Англійська. Німецька, Японська	Англійська, Німецька, Японська, Китайська	Англійська	Англійська

Продовження таблиці 1.1

з/п	Критерій	Назва системи			
	Функціонал	*Exabeam*	*Securonix*	*Netskope*	*Forcepoint*
6	Автоматизовані звіти	+	+	+	+
7	Контроль доступу	+	+	+	+
8	Звіти адміністратору	+	+	+	+
9	Звіти для користувачів	+	+	+	+
	Технічна підтримка				
10	Автоматичний пріоритет для заявок	+	+	-	+
11	База знань для користувчів	+	+	+	+
12	Статуси аномалій	+	+	+	+
13	Коментарі	-	+	+	+
14	Портал самообслуговування	+	+	-	+
15	Вкладення файлів	-	+	+	+
16	Геолокаційні дані	+	+	+	-
17	Аналітика ефективності	+	+	+	+
18	Автоматизація звітів	+	+	+	+
19	Шаблони розсилок	-	+	+	+
20	Інтеграція з *E-mail*	-	+	+	+
21	Керування інцидентами	+	-	-	+
22	Керування аномаліями	+	-	-	+
23	Фільтри	+	+	+	+
24	Рівні доступом	+	+	+	+
25	Користувацькі фільтри	-	-	+	+
26	Коментарі до збоїв	-	-	+	+
27	Автоматизовані черги	+	+	-	+
28	Інтелектуальне сортування	-	-	-	-
29	Графічна статистика	+	-	+	+
30	Прогнозування аномалій	+	-	-	+
	Аналітика				
31	Локальне програмне забезпечення	+	-	-	-
32	Просунута аналітика	+	+	+	+
33	Автореагуваня на інциденти	-	-	+	+
34	Машинне навчання	-	-	+	+
35	Глибоке навчання	+	-	-	+
36	Звіти про загрози	+	+	+	+
37	Сповіщеня в реальному часі	+	+	+	+
38	Інструментарій кримінал експертизи	+	+	-	-
39	Налаштовуємі повідомлення	+	-	+	+

Система аналітики і моніторингу *UEBA* повинна забезпечувати такий набір функціоналу необхідного для надання послуг:

– список повідомлень про загрози;

– налаштування користувачів та груп користувачів;

– Доступ до розділів в інфраструктурі повинен бути розділений на користувацький (в залежності від груп користувача) і адміністративний: коли адміністратор бачить всі інциденти та аномалії. Інциденти в основному створюються автоматично системою. Система може автоматично їх вирішити чи передати адміністратору

– Можливість прикріпити чи згенераувати автоматично додаткові дані, та довідкову інформацію щодо внциденту;

– Врахувати можливість роботи додатку не тільки за допомгою *web*-інтерфейсу комп'ютеру , але і через локальні програми.

Висновки

На даний момент, при реалізації *SIEM* систем різного типу в основному використовуються застарілі технології. З часом, зростання ролі захисних механізмів в компанії будь-якого розміру зростає, а необхідність в забезпеченні високого рівня захисту даних, при забезпеченні високої (*SLA*) доступності послуг стає одною із найголовніших вимог бізнесу.

Останнім часом багато дискусій щодо аналітики поведінки користувачів (*UBA*), і фактична корисність *UBA* часто завищена. Аналітика поведінки має численні переваги, які можна використовувати, одразу як тільки система встановлена, так і коли вона вже працює в парі з іншими заходами безпеки, щоб допомогти запобігти потенційним атакам.

Технологія *UEBA* забезпечує менший ризик для компанії та має менше шансів на помилкові спрацьовування. *UEBA* може допомогти запобігти шкоді, заподіяній як сторонніми, так і внутрішніми атаками. Незалежно від того, чи це хакер, який намагається отримати доступ до мережі, або працівник, який став внутрішньою

загрозою, технологія *UEBA* здатна виявляти зловмисну поведінку та попереджати відповідальних за контроль за безпекою компанії.

Компрометовані облікові записи користувачів - це один із найпоширеніших випадків використання *UEBA* в тому випадку, якщо користувач має скомпрометований обліковий запис. У цьому випадку *UEBA* може виявити, що обліковий запис виконує ненормальні дії, такі як доступ до конфіденційної інформації, до якої користувач зазвичай не намагається отримати доступ, а потім попередити аналітика безпеки для розслідування.

Крадіжку даних *UEBA* також добре запобігає або, принаймні, мінімізує. Він здатний виявляти, коли користувач завантажує інформацію, якої зазвичай їм не дозволяється навіть переглядати. У цьому випадку сповіщення, як правило, не надсилається, поки зловмисник вже не отримав доступ і не завантажив дані, але після спрацьовування адміністратор зможе закрити доступ до облікового запису та запобігти подальшим крадіжкам.

Таким чином, в результаті аналізу наявних проблем було виявлено що основними проблемами є швидкість спрацювання на аномалію, швидкість обробки даних що поступають для аналізу, а також високу вартість систем для моніторингу та аналітики активності користувачів в мережі.

РОЗДІЛ 2
ЗАСОБИ РОЗРОБКИ СИСТЕМИ АНАЛІТИКИ

Розробка програм - складний процес, основною метою якого є створення, супровід програмного коду, що забезпечує необхідний рівень надійності і якості. Для досягнення основної мети розробки програм використовуються спеціалізовані засоби розробки програмного забезпечення.

При виборі засобів розробки системи враховуються наступні компоненти такі як ріст кількості користувачів буде зростати навантаження сервери для обробки даних та на *API*, також на обробку одного запиту по *API* необхідно витрачати мінімум процесорного часу.

Залежно від предметної області і завдань, поставлених перед розробником, розробка програми може являти собою досить складний, поетапний процес, в якому задіяна велика кількість учасників і різноманітних засобів.

Для того, щоб визначити, коли і в яких випадках які кошти застосовуються, виділимо основні етапи розробки програмного забезпечення. Найбільший інтерес для проблематики розглянутого питання представляють наступні етапи розробки:

– проектування додатка;
– реалізація програмного коду програми;
– тестування програми.

Технології, інструменти та інші рішення для розробки, які дозволяють чином задовольнити усі вимоги замовника щодо створення системи з використанням технології *UEBA*. В рамках економії часу та ресурсів для первого етапу розробки варто розглядати готові рішення на базі відкритого програмного забезпечення - програмне забезпечення з відкритим (O*pensource*) вихідним кодом.

Таким чином, процес розробки програм є складним процесом і те, які кошти необхідно застосовувати багато в чому залежить від завдань, поставлених перед розробниками. Незалежно від задачі на розробку кошти не можна обмежувати лише набором якихось інструментальних засобів, також необхідно включати методи, способи, підходи і все-тое що застосовується для створення програми, що відповідає заданим замовникомвимогам.

2.1. Кросплатформенне середовище *Python*

Python - це відкрите кросплатформенне об'єктно-орієнтоване середовище програмування високого рівня.

– вільний так як всі вихідні тексти інтерпретатора і бібліотек доступні для будь-якого, в тому числі комерційного, використання;

– об'єктно-орієнтована - класична модель, включаючи всі переваги ООП, такі як поліморфізм, наслудвання та інкапсуляція;

– розширюваний, так як має *API* для створення модулів, типів і класів на C або C ++;

– вбудований, так як має *API* для вбудовування інтерпретатора в інші програми;

Python - це універсальна мова, вона широко використовується в усьому світі для найбільш різних цілей - бази даних і обробка текстів, інтерпретатори та ігри, програмування *GUI* і швидке створення прототипів (*RAD*).

Звичайно, *Python* використовується і для програмування *Internet* і *Web* додатків - серверних (*CGI*), клієнтських (роботи чи краулери), *Web*-серверів і серверів додатків. *Python* має багату стандартну бібліотекою, і ще більше модулів, написаних третіми особами. Python підтримує структурне, узагальнене, об'єктно-орієнтоване, функціональне і аспектно-орієнтоване програмування. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних.

Еталонної реалізацією *Python* є інтерпретатор *CPython*, що підтримує більшість активно використовуваних платформ. Він поширюється під вільною ліцензією *Python Software Foundation License*, що дозволяє використовувати його без обмежень в будь-яких додатках, включаючи пропрієтарні. Є реалізація інтерпретатора для *JVM* з можливістю компіляції, *CLR, LLVM*, інші незалежні реалізації. Проект *PyPy* використовує *JIT*-компіляцію, яка значно збільшує швидкість виконання *Python*-програм.

Так як *Python* додаок з відкритим (O*pensource*) вихідним кодом, інтерпретатор *Python* використовується по всьому світу і поставляється в складі операційних систем на базі *Linux*, а також в комп'ютерах від фірми *Apple*. *Python* популярний серед індивідуальних розробників, але також використовується великими компаніями в досить серйозних продуктах, орієнтованих на отримання прибутку

Деякі проекти реалізують базову частину на більш продуктивних мовах програмування, а для спрощення роботи надають повноцінний інтерфейс прикладного програмування на *Python*. Так, движок вільного редактора відео *OpenShot* реалізований у вигляді бібліотеки *libopenshot*, написаної на *C++* з використанням бібліотек на Сі, а всі можливості повністю покриті прикладним інтерфейсом програмування *Python*.

Python з пакетами *NumPy*, *SciPy* і *MatPlotLib* активно використовується як універсальне середовище для наукових розрахунків в якості заміни поширеним спеціалізованим комерційним пакетам, таким як *Matlab*, надаючи аналогічну функціональність і більш низький поріг входження.

2.2. Архітектура *RESTful API*

RESTful API - це архітектурний стиль інтерфейсу прикладних програм (*API*), який використовує *HTTP*-запити для доступу та використання даних. Ці дані можуть бути використані для отримання, встановлення, відправки та видалення типів даних, що стосується зчитування, оновлення, створення та видалення операцій, що стосуються ресурсів. *API* для веб-сайту чи червісу - це код, який

дозволяє двом програмам взаємодіяти між собою. *API* визначає правильний спосіб для розробника писати програму, що вимагає послуги від операційної системи або іншого додатка.

RESTful API або REST API - заснований на репрезентативній передачі стану (REST), що є архітектурним стилем та підходом до комунікацій, часто використовуваних при розробці веб-служб. Зазвичай технологія *REST* є кращою перед іншими подібними технологіями. Це має місце, оскільки *REST* використовує меншу пропускну здатність, що робить його більш придатним для ефективного використання Інтернету.

Логіка роботи *API RESTful* полягає в тому, що технологія розбиває транзакцію, створюючи серію невеликих модулів. Кожен модуль звертається до основної частини транзакції. Ця модульність надає розробникам велику гнучкість, але розробникам може бути складно розробити свій *REST API* з нуля. В даний час кілька компаній пропонують моделі для використання розробниками; моделі, що надаються *Amazon S3 , Cloud Data Management Interface* (*CDMI*) та *OpenStack Swift*, є найбільш популярними.

Стан ресурсу в будь-яку частину часу називається поданням ресурсу. *RESTful* використовує існуючі методології *HTTP*, визначені протоколом *RFC 2616*, такі як *GET* отримати ресурс, *PUT* для зміни стану або оновлення ресурсу, який може бути об'єктом, файлом або блоком, *POST* для створення цього ресурсу, *DELETE*, щоб видалити його.

Формати даних, які підтримує *API REST*, включають: a*pplication / json application / xml application / x-wbe + xml application / x-www-form-urlencoded multipart / form-data.*

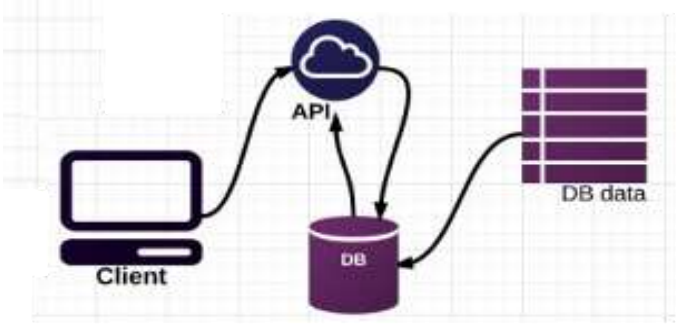


Рис.2.1.Структура архітектури *REST API*

З точки зору звернень до ресурсів із клієнтського додатку, що надають *URI*, визначають, наскільки інтуїтивним буде веб-сервіс *REST* і він буде використовуватися так, як передбачається розробником. Третя характеристика *Web*-сервісу *RESTful* повністю посвячена *URI*.

URI-адреса веб-сервісу *REST* має бути інтуїтивно зрозумілою. *URI* це як самодокументуючий інтерфейс, і йому майже не потрібні пояснення або обробки в розробнику для його розуміння та для отримання відповідних ресурсів. Тому структура *URI* повинна бути простою, передбачуваною та зрозумілою.

Один із способів досягнення такого рівня зручності використання - побудова *URI* за аналогією з структурою каталогів. Такий роду *URI* є ієрархічним, що знаходяться з одного кореневого шляху, вказівки якого відображаються основні функції сервісу.

Відповідно до цього визначення, *URI* - це дерево з інформацією про, з'єднаними в вузлах. Наприклад, у сервісі обговорення різних тем (від *Java* до *Python*) можна визначити структурований набір *URI* будь якого зручного для розробника вигляду.

2.3. Формат обміну даними JSON

JSON (*JavaScript Object Notation*) - простий формат обміну даними, зручний для читання і написання як людиною, так і комп'ютером. Він заснований на підмножині мови програмування *JavaScript* , визначеного в стандарті *ECMA-262 3rd Edition - December +1999*.

JSON - текстовий формат, повністю незалежний від мови реалізації, але він використовує угоди, знайомі програмістам *C*-подібних мов, таких як *C, C++, C#, Java, JavaScript, Perl, Python* і багатьох інших. Ці властивості роблять *JSON* ідеальним мовою обміну даними. *JSON* заснований на таких структурах як: пара ключ і значення. Незважаючи на те, що *JSON* є похідним від *JavaScript*, він підтримується як на рідній мові, так і через бібліотеки більшості основних мов програмування. *JSON* зазвичай, але не виключно, використовується для обміну інформацією між веб-клієнтами і веб-серверами.

За останні 15 років JSON став повсюдно використовуватися в Інтернеті. Сьогодні це формат, який вибирають майже для всіх загальнодоступних веб-сервісів, і він часто використовується і для приватних веб-сервісів.

Популярність JSON також привела до того, що багато баз даних підтримують *JSON*. Реляційні бази даних, такі як *PostgreSQL* і *MySQL*, тепер поставляються з рідною підтримкою зберігання і запиту даних *JSON*. Бази даних *NoSQL*, такі як *MongoDB* і *Neo4j*, також підтримують *JSON*, хоча *MongoDB* використовує злегка змінену двійкову версію *JSON* за лаштунками.

JSON - це універсальний формат даних з мінімальною кількістю типів значень: рядки, числа, булеан, списки, об'єкти і нуль. Хоча нотація є підмножиною JavaScript, ці типи представлені у всіх поширених мовах програмування, що робить JSON хорошим кандидатом для передачі даних через мовні прогалини..

Дані в форматі *JSON* виглядають так:

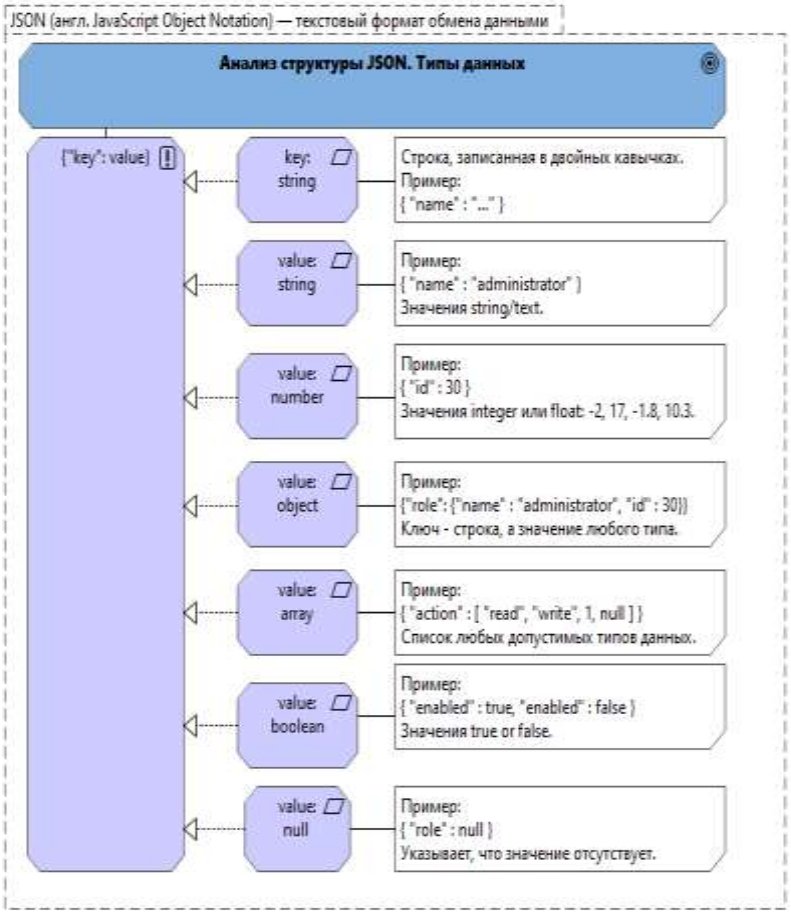


Рис.2.2. Аналіз структури даних *JSON*. Типи даних

JSON це легкий формат, який дозволяє вам легко ділитися, зберігати і працювати з даними. Як формат, *JSON* переживає зростаючу підтримку *API*, включаючи і *Twitter API*.

Так як ви швидше за все не будете створювати ваші власні *.json* файли, але будете отримувати їх з інших джерел, то стає дуже важливим менше думати про саму *JSON* структурі і більше про те, як краще його застосовувати в ваших програмах.

2.4. Технологія *AJAX*

AJAX, – це набір методів веб-розробки з використанням багатьох веб-технологій на стороні клієнта для створення асинхронних веб-додатків.

З *Ajax*, веб-додатки можуть надсилати і отримувати дані від сервера асинхронно (в фоновому режимі) без втручання в відображення у поведінку існуючої сторінки. Розв'язуючи рівень обміну даними з шаром уявлення, *Ajax* дозволяє веб-сторінкам і, як наслідок, веб-додатків динамічно змінювати вміст без необхідності перезавантаження всієї сторінки. На практиці в сучасних реалізаціях зазвичай використовується *JSON* замість *XML*.

Ajax - це не єдина технологія, а швидше група технологій. *HTML* і *CSS* можуть використовуватися в комбінації для розмітки інформації про стилях. Потім веб-сторінка може бути модифікована за допомогою *JavaScript* для динамічного відображення і дозволяє користувачеві взаємодіяти з новою інформацією.

Зазвичай Ajax використовується для виконання коду на веб-сторінках, дозволяючи веб-сайтам завантажувати вміст на екран без оновлення сторінки.

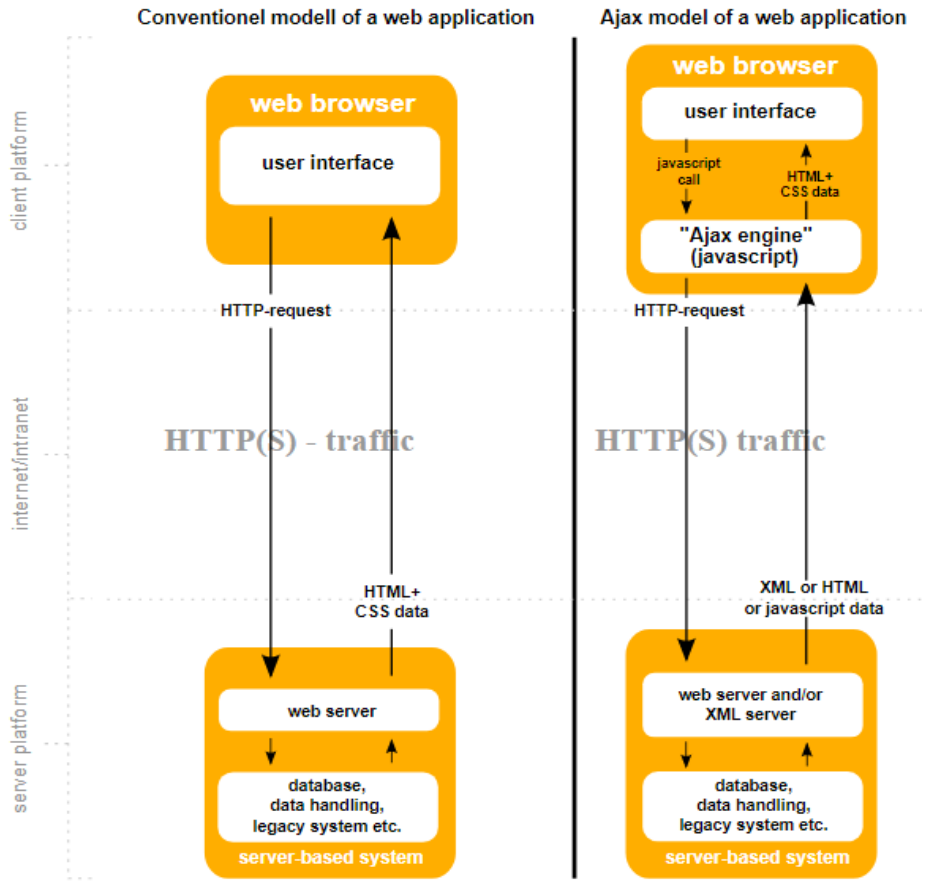


Рис.2.3. Порівняння класичного додатку і додатку із застосуванням *AJAX*

Для обміну даними на сторінці створюється об'єкт *XMLHttpRequest*, він буде виконувати функцію посередника між браузером і сервером. Запити можуть відправлятися в одному двох типів - *GET* і *POST*. У першому випадку звернення проводиться до документа на сервері, в ролі аргументу йому передається URL сайту. Для запобігання переривання запиту можна скористатися функцією

2.5. Мова опису стилів CSS

CSS (*Cascading Style Sheets*) - це код, який використовується для стилізації вашої веб-сторінки. CSS це мова опису стилів про те, як зробити текст чорним або червоним? Як зробити так, щоб контент з'являвся в певному місці на екрані? Як прикрасити веб-сторінку за допомогою фонових зображень і кольорів?

CSS використовується розробниками веб-сторінок для задання кольорів, шрифтів, стилів, розташування окремих блоків і інших аспектів представлення зовнішнього вигляду цих веб-сторінок. Це дозволяє максимально гнучко

налаштувати точність відображення кожного елементу на сторінці. Також *CSS* використовується для оптимізації відображення сторінки в мобільних пристроях, таких як планшет чи смартфон.

Основною метою розробки *CSS* було відділення опису логічної структури веб-сторінки (яке проводиться за допомогою *HTML* або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер проводиться за допомогою формального мови *CSS*).

Такий поділ може збільшити доступність документа, надати велику гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, *CSS* дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друковане подання, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при виведенні пристроями, що використовують шрифт Брайля

Об'єктно-орієнтована *CSS* (англ. *Object-Oriented CSS*, *OOCSS*) - один з підходів до організації коду *CSS*.

BEM (*Block-Element-Modifier*) - компонентний підхід до веб-розробці, який може використовуватись паралельно з *LESS*. В його основі лежить принцип поділу інтерфейсу на незалежні блоки. Він дозволяє легко і швидко розробляти інтерфейси будь-якої складності, називаючи класи *CSS* з використанням нижніх ліній: *.card__img - large*

Блоки є повторно використовуваними компонентами. Під час іменування блокі, необхідно зосередитись на описі їх цілі (тобто, що робить клас), а не стан класу.

Мікси дозволяють в *BEM* підході поєднувати поведінку і стилі декількох БЕМ-сутностей без дублювання коду і створювати семантично нові компоненти інтерфейсу на основі наявних *BEM*-сутностей. Якщо, в проекті посилання реалізовані блоком link. Необхідно зробити посиланнями пункти меню.

Скористатися міксом універсального блоку *link* і елемента *link* блоку *menu*. Мікс двох *BEM*-сутностей дозволить застосувати базову функціональність посилань з блоку *link* і додаткові *CSS*-правила з блоку *menu* без копіювання коду.

Існують і інші моделі організації коду: *SMACSS*, *Atomic CSS* і т.д. У кожного з цих підходів є свої переваги і недоліки. Який з них вибрати - вирішувати вам як розробнику, спираючись на власний досвід і переваги. Крім того, ніщо не заважає вам виробити свій спосіб організації *CSS*.

2.6 Документо-орієнтована база даних *MongoDB*

БД MongoDB - це система управління базами даних (СУБД) з відкритим кодом, орієнтована на документи, яка не вимагає опису схеми таблиці. *MongoDB* - швидка та масштабована система, яка працює з даними ключ / значення та реляційними СУБД, які є функціональними та простими для запитів (рис. 2.4).

База даних *MongoDB* це швидка БД, що масштабується, високодоступних і повністю керована документная база даних, яка підтримує робочі навантаження *MongoDB*. Оскільки *MongoDB* є документную базу даних, він спрощує зберігання та індексування даних *JSON*, а також виконання запитів до них.

MongoDB - це нереляційних база даних, яка розроблялася з метою забезпечити користувачам необхідну продуктивність, масштабованість і доступність при обробці критично важливих робочих навантажень MongoDB в будь-якому масштабі. В *MongoDB* сховище і обчислювальні ресурси розділені, що дозволяє масштабувати їх окремо. Можна підвищити продуктивність операцій читання до мільйонів запитів в секунду, додавши до 15 реплік читання з низькою затримкою. Репліка створюється за лічені хвилини незалежно від обсягу даних.

Розрахункова доступність *MongoDB* може становить 99,99%. Сервіс шестикратно може реплікувати дані в трьох зонах доступності (наприклад *AWS*). За допомогою сервісу *AWS Database Migration e* (*DMS*) можна абсолютно безкоштовно і з мінімальними простоями перенести в *MongoDB* різні базі.

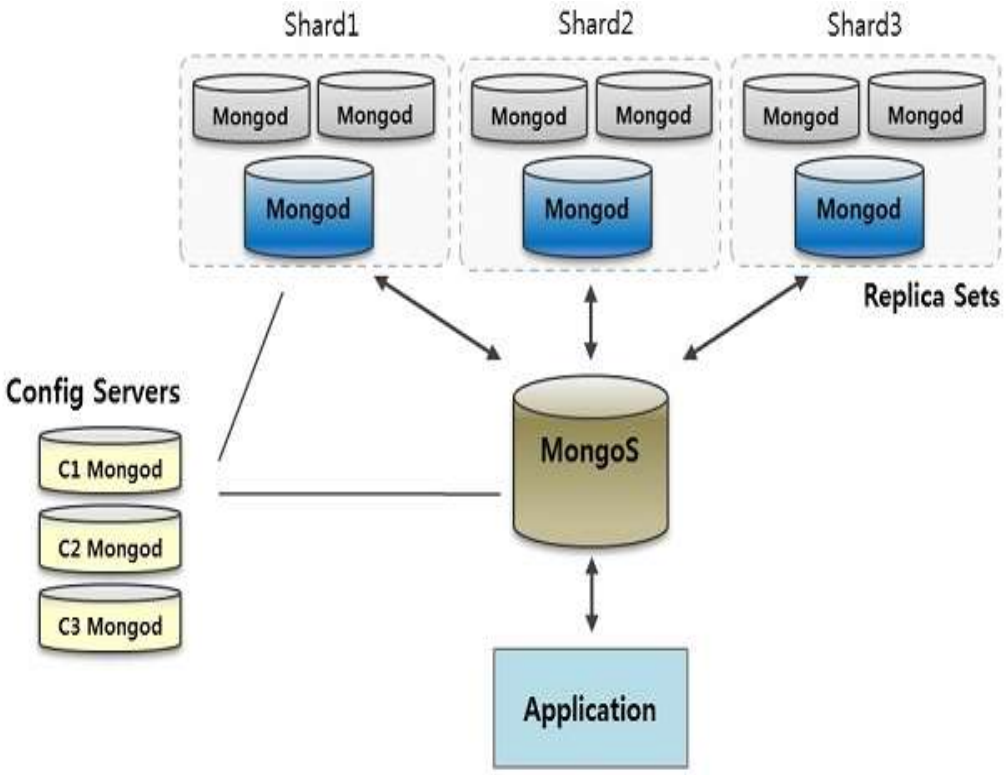


Рис.2.4. Схема даних *MongoDB*

MongoDB підтримує протоколювання операцій зі зміни і додаванню даних в базу даних, може працювати за карткою / редукування парадигми, підтримує реплікацію і побудова стійких до відмов конфігурацій.

MongoDB має вбудовані засоби для забезпечення шардингу (розподілу усіх даних по різним серверам на базі заданого ключа шардування), поєднання яких з реплікацією даних дозволяє побудувати горизонтально масштабувати кластер зберігання, в якому немає єдиної точки відмови (відмова будь-якого вузла не впливає на роботу всієї бази даних), підтримується повністю автоматичне чи напівавтоматичне відновлення після відмови і перенесення навантаження з вийшов з ладу вузла.

Якщо розмістити базу даних в хмарному сервісі то в цілому за допомогою *MongoDB* кластери самовідновлення складаються з географічно розподілених екземплярів баз даних, щоб не було жодної точки відмови. Можна гарантувати високу стійкість до відмов у декількох регіонах, увімкнувши міжрегіональну реплікацію. *MongoDB* також включає потужні функції для підвищення надійності

ваших критично важливих виробничих баз даних, такі як безперервне резервне копіювання та відновлення за часом.

MongoDB спрощує контроль доступу до бази даних. Екземпляри бази даних можуть розгортатися в унікальній віртуальній приватній хмарі (*VPC*), щоб забезпечити ізоляцію мережі.

Інші функції безпеки включають білий список *IP*-адрес або *VPC Peering*, постійно ввімкнену автентифікацію, шифрування в стані спокою та шифрування в процесі передачі, складне управління рольовим доступом тощо.

MongoDB автоматизує надання, налаштування та розгортання інфраструктури, щоб можна було отримати необхідні ресурси бази даних, коли вони потрібні. Виправлення та незначні оновлення версій застосовуються автоматично. І коли потрібно змінити свій кластер - будь то для масштабування чи оновлення - *MongoDB* дозволяє це зробити за кілька кліків, – без простою.

Висновки

Ідентифікація та виявлення цілеспрямованих атак, які не можуть бути виявлені стандартними засобами, існуючими на ринку інформаційної безпеки в даний час – це важлива вимога до сучасного програмного забезпечення, що використовуються для захисту комп’ютерних мереж.

Враховуючи розглянуті раніше вимоги до створення системи з використанням технології *UEBA* було обрано архітектуру на базі сервісу *Digitalocean*. У другому розділі розглянуто інструменти та підходи, нові види архітектур та технологій для вирішення наявних задач при реалізації системи. В якості серверної мови програмування обрано платформу *Python*.

Асинхронний механізм *AJAX* дозволяють *Python* серверу одночасно обробляти багато підключень та різних типів запитів по *API*, при цьому розробник не витрачає час на обробку і управління потоками даних

Форматом обміну даними між клієнтом та сервером, було використано *JSON* тк як він дозволяє обмінюватись даними між браузером і сервером (*AJAX*), так і між серверами (*HTTP*).

Для опису стилів на *HTML* сторінках обрано мову програмування стилів *CSS*, а також препроцесор *LESS*.

Таким чином, на підставі аналізу вимог до системи, які полягають у можливості горизонтальної масштабованості та реплікації, – було обрано підхід до реалізації який масштабує сховища (БД) інформації з гнучкою моделлю даних, що відрізняється від класичних реляційних СУБД. Таку можливість пропонує *NoSQL*. В рамках зазначеного вище, оптимальним рішенням було обрано СУБД *MongoDB*.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМИ ПОВЕДІНКОВОЇ АНАЛІТИКИ UEBA

Програмне забезпечення для системи поведінкової аналітики *UEBA* базується на ролях користувачів. Це дозволяє керувати доступом, налаштовувати права та здійснювати моніторинг процесів як в окремих ланках мережі так і контролювати всю інфраструктуру в цілому. В системі для поведінкового аналізу користувачів за замовчуванням передбачені наступні ролі доступу: адміністратор, директор, фінансовий директор, технічний директор, комерційний директор, менеджер з продажів, системний адміністратор, користувач, гість.

3.1. Специфікація програмного забезпечення

Система моніторингу поведінки користувачів з використанням технології *UEBA* потребує значних зусиль при інтеграції, тому перед початком роботи системний інтегратор виконує початкове налаштування системи. Після виконання інтеграції до базових компонентів інфраструктури та впровадження початкових налаштувань, — тонкі налаштування здійснює адміністратор компанії в яку система була інтегрована, а системний інтегратор більше не має доступу до компонентів мережі які були інтегровані.

Системний адміністратор (рис.3.1) авторизується в системі (базовий обліковий запис створюється системним інтегратором напряму в базі даних) та починає створювати ролів користувачів а потім і самих користувачів.

З питань безпеки та економії часу інтегратор рішення на технології *UEBA* та системний адміністратор компанії займуються різними задачами під час інтеграціх системи. Спочатку основну частину роботи виконує інтегратор, далі системний адміністратор виконує базові налаштування ситеми.

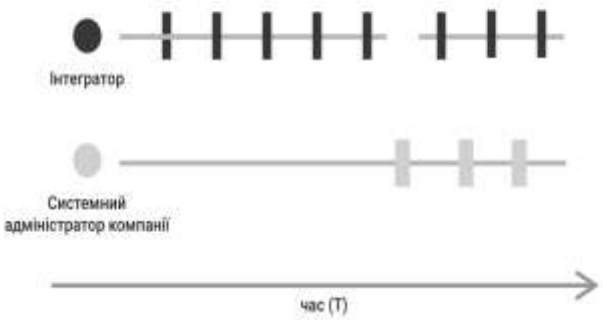


Рис.3.1. Алгоритм інтеграції UEBA в інфраструктуру компанії

Після цього інтегратор продовжує поширювати своє рішення на інші модулі мережі в інфраструктурі компанії, а системний адміністратор виконує лише доналаштування та інші операції які слід виконати після появи в системі нового модулю.

Системний адміністратор компанії, куди інтегрується система на базі технологіх *UEBA* (рис.3.2) контролює загальний стан системи та її основні показники, переглядає інциденти та аномалії що виникають в ході роботи, створює та керує користувачами в рамках інфраструктури компанії, налаштовує рівні доступу для груп користувачів.

Рис.3.2. Задачі які виконує адміністратор компанії в системі *UEBA*

Адміністратор (рис.3.3) може змінювати статуси та пріоритети інцидентів та аномалій, додавати коментарі. Отримавши інцидент чи аномалію, адміністратор в залежності від автоматично визначеного системою пріоритету приймає рішення

про наступні кроки: закрити користувачу доступ, відправити звіт службі безпеки компанії на перевірку чи поставити користувача в список на посилений моніторинг. Виконавши заявку, адміністратор змінює її статус на "оброблено" чи "в роботі", на випадок якщо інцидент чи аномалію швидко вирішити неможливо.

Якщо в компанії більше одного адміністратора то передбачено можливість делегувати перевірку та виконання роботи по заявці іншому адміністратор.

Рис.3.3. Основні типи роботи з заявками які виконує адміністоратор

Наступні кроки системного адміністратору змінюються в залежності від специфікації кожного інциденту, його пріоритету чи додаткових параметрів.

Інциденти інформаційної безпеки в мережі можуть бути навмисними або випадковими (наприклад, бути наслідком будь-якої людської помилки або природних явищ) чи викликані як технічними, так і нетехническими засобами.

Їх наслідками можуть бути такі події, як несанкціоновані розкриття або зміна інформації, її знищення або інші події, які роблять її недоступною, а також нанесення шкоди активів організації або їх розкрадання. Інциденти ІБ, про які не було повідомлено, але які були визначені як інциденти, розслідувати неможливо, а застосувати захисних заходів для запобігання повторного появи таких інцидентів не можна.

Таким чином, знаючи всі доступні можливості системи, розглянемо детальніше найбільш часті варіанти використання системи. Процес виникнення інциденту (рис.3.4).

Рис.3.4. Процес виникнення інцидентув системі

Відповідно до зазначеного вище рисунку, процес виникнення інциденту (інциденту чи аномалії) виконується як за допомогою автоматизованої системи по визначеним нею правилам, так і повністю в ручному режимі, що усуває можливі неточності, несправності і дозволяє адміністратору найбільш повно оцінити проблему та прийняти зважене рішення. Якщо ж автоматизована система виявила значне порушення в роботі мережі то вона може ініціювати дії по усуненню чи локалізації збою в автоматичному режимі без втручання адміністратору.

3.2. Алгоритм роботи системи поведінкової аналітики

Традиційні методи аналітики зводяться до визначення фахівцями з безпеки наборів правил, на підставі яких система визначають, чи є та чи інша подія порушенням правил безпеки чи ні. Даний підхід часто не є підготовленим до нових типів загроз або зміни поведінки користувачів системи.

Відмінною рисою рішень *UBA* є розширена аналітика, призначена головним чином для запобігання витоку конфіденційної інформації. Ознакою хорошого аналітичного апарату *UBA* вважається не тільки здатність виявити аномалії в поведінці, а й уміння розрахувати ймовірність того, наскільки аномалія свідчить про дійсну загрозу безпеці. В даний час в основу розширеної аналітики *UBA*-

рішень закладаються технології машинного навчання і статистичні моделі. До таких моделей можна віднести наступні алгоритми роботи технології *UEBA.*

Unsupervised algorithms – це алгоритми, що не вимагають наявності заздалегідь підготовлених прикладів для подальшої класифікації користувачів і детектування аномалій поведінки. Це дозволяє використовувати рішення *UBA* без попередньої адаптації та налаштування, що безсумнівно є великим плюсом. Але також має і негативний ефект, - це безліч малозначущих з точки зору безпеки результатів класифікації користувачів і велику частку помилкових спрацьовувань.

Supervised algorithms - це алгоритми, що мають більш високу якість оцінки та аналіху поведінки та класифікації користувачів системи. На практиці така висока якості потребує регулярного перенавчання та донавчання системи, а також значних апаратних ресурсів. Це кожен раз вимагає залучення фахівців з аналізу даних і тісних комунікацій між ними і співробітниками відділу безпеки компанії де інтегровано рішення *UEBA.*

Змішані моделі поєднують в собі як «автономні» алгоритми, так і *supervised*-алгоритми.

У деяких алгоритмах можна виділити моделі з «жадібними» підходами до аналізу. Такі алгоритми постійно автоматично підлаштовують модель, приймаючу рішення, під кожну нову активність користувача. Це дозволяє миттєво уточнювати модель і коригувати її якість. Однак ці алгоритми більш вимогливі до ресурсів і не позбавлені своїх проблем.

Згідно з дослідженнями *Gartner*, в найближчому майбутньому рішення *UBA* будуть поповнюватися інструментарієм глибокого навчання і нейронних мереж.

Поведінкові фактори (ПФ) - дані про поведінку користувачів, які система з використанням технології *UEBA* враховує при ранжируванні дій користувачів. Визначає наскільки вагомими факторами ранжирування є та чи інша поведінка користувачів. Єдиного і повного списку поведінкових факторів немає, оскільки такий список постійно змінюється: розробники змінюють існуючі фактори та додають нові. Умовно можна розділити поведінкові фактори на ті, які заміряються поза зоною корпоративної мережу (зовнішні), і поведінка користувачів в мережі (внутрішні).

До зовнішніх чинників відносять такі дії як:

– інформація з зовнішніх відкритих баз даних;

– інформація з зовнішніх закритих баз даних;

– пристрої з яких користувач здійснює доступ до внутрішніх сервісів;

– інші дані про користувачів які працюють з системою;

До внутрішніх чинників відносять такі дії як:

– інформація з внутрішніх закритих баз даних;

– інформація що отримана внаслідок автоматичного навчання системи;

– фактори поведінки користувача у внутрішні мережі;

– алгоритм роботи користувача з сервісами в мережі;

– інші дані про користувача який працює з системою;

– комунікації співробітників в компанії;

– аналіз кола спілкування співробітників;

– моніторинг ресурсів, робочих станцій, серверів, мережевих каналів;

– інші непрямі дані про користувачів які працюють з системою;

Компанія активно збирає всі можливі дані про користувача своєї мережі, про корпоративну мережу в цілому, а також аналізує мережу інтернет на предмет виникнення нових загроз, витоків даних та інше.

3.3. Підключення до бази даних

Для початку роботи з базою даних необхідно додати модуль ініціалізації *BaseModule* в корінь *App*, щоб почати з’єднання з БД, додатками та іншими ланками системи. Задано спільний модуль для надання моделей *SQLAlchemy ORM* для запитів *ConfigDB*.

Модуль використовує службу підключення *MongoDB — wc_configdb (ConfigDB)*. Налаштування файлу служби з'єднання pg_service.conf:

Для цього створюється та редагується файл *config.conf* в який записується такий код (рис. 3.5).

```python
from pymysqlpool import ConnectionPool

config = {
    'pool_name': 'test',
    'host': 'localhost',
    'port': 3306,
    'user': 'root',
    'password': 'root',
    'database': 'test'
}

def connection_pool():
    # Return a connection pool instance
    pool = ConnectionPool(**config)
    return pool

# 直接访问并获取一个 cursor 对象, 自动 commit 模式会在这种方式下启用
with connection_pool().cursor() as cursor:
    print('Truncate table user')
    cursor.execute('TRUNCATE user')

    print('Insert one record')
    result = cursor.execute('INSERT INTO user (name, age) VALUES (%s, %s)', ('Jerry', 20))
    print(result, cursor.lastrowid)

    print('Insert multiple records')
    users = [(name, age) for name in ['Jacky', 'Mary', 'Micheal'] for age in range(10, 15)]
    result = cursor.executemany('INSERT INTO user (name, age) VALUES (%s, %s)', users)
    print(result)

    print('View items in table user')
    cursor.execute('SELECT * FROM user')
    for user in cursor:
        print(user)

    print('Update the name of one user in the table')
    cursor.execute('UPDATE user SET name="Chris", age=29 WHERE id = 16')
    cursor.execute('SELECT * FROM user ORDER BY id DESC LIMIT 1')
    print(cursor.fetchone())

    print('Delete the last record')
    cursor.execute('DELETE FROM user WHERE id = 16')
```

Рис. 3.5. Код програми для з’єднання з БД *MongoDB*

Наступним кроком необхідно створити початкову схему БД, задати типи даних зберігання. Також необхідно базово нормалізувати структуру даних в базі даних, незважаючи на те, що *Nosql* бази не потребують звичної нормалізації.

Створюємо файл *fileteste.py* та додаємо код, який забезпечить створення необхідних таблиць, колонок та рядків в БД (рис. 3.6). Після цього вважається, що базові налаштування з базою даних виконано. Можна провести тестове підключення до бази даних аби перевірити якість виконання задачі.

```
setup(
    name='Pyrseas',
    version='0.9.1',
    packages=['pyrseas', 'pyrseas.dbobject', 'pyrseas.lib', 'pyrseas.augment',
              ],
    package_data={'pyrseas': ['config.yaml']},
    entry_points={
        'console_scripts': [
            'dbtoyaml = pyrseas.dbtoyaml:main',
            'yamltodb = pyrseas.yamltodb:main',
            'dbaugment = pyrseas.dbaugment:main']},

    install_requires=[
        'psycopg2 >= 2.5',
        'pgdbconn >= 0.8.0',
        'PyYAML >= 3.10'],

    tests_require=['pytest'],
    cmdclass={'test': PyTest},

    author='Joe Abbate',
    author_email='jma@freedomcircle.com',
    description='Utilities to assist in database schema versioning',
    long_description=open('README.rst').read(),
    url='https://perseas.github.io/',
    classifiers=[
        'Development Status :: 4 - Beta',
        'Environment :: Console',
        'Intended Audience :: Developers',
        'Intended Audience :: Information Technology',
        'Intended Audience :: System Administrators',
        'License :: OSI Approved :: BSD License',
        'Natural Language :: English',
        'Operating System :: OS Independent',
        'Programming Language :: Python',
        'Programming Language :: Python :: 2',
        'Programming Language :: Python :: 2.7',
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.4',
        'Programming Language :: Python :: 3.5',
        'Programming Language :: Python :: 3.6',
        'Programming Language :: Python :: 3.7',
        'Programming Language :: SQL',
        'Topic :: Database :: Front-Ends',
        'Topic :: Software Development :: Code Generators',
        'Topic :: Software Development :: Version Control'],
    platforms='OS-independent',
    license='BSD')
```

Рис. 3.6. Розробка схеми та налаштування бази даних

3.4. Створення *API*

API - це інтерфейс взаємодії між сайтом і сторонніми програмами і серверами. Програміст може скористатися API для отримання доступу до функціоналу сторонньої програми. API робить можливим роботу ресурсів, які використовують потенціал і потужність іншого сайту або програми.

Розподілення логіки системи від контролерів у системі виконує сервіс *API.*

REST API (*Representational State Transfer*) - це *API*, яке використовує *HTTP*-запити для обміну даними.

REST API повинні відповідати певним критеріям:

– Архітектура клієнт-сервер: клієнт взаємодіє з призначеним для користувача інтерфейсом, а сервер - з бекенд і сховищем даних. Клієнт і сервер незалежні, будь-який з них може бути замінений окремо від іншого.

– *Stateless* - ніякі клієнтські дані не зберігаються на сервері. Стан сеансу зберігається на стороні клієнта.

– Кешування - клієнти можуть зберігати відповіді сервера для поліпшення загальної продуктивності системи.

Дії, які можна виконувати за допомогою *API* описані в концепції *CRUD.*

CRUD - концепція програмування, яка описує чотири базових дії (*create*, *read*, *update* і *delete*). У *REST API* типи запитів і методи запиту відповідають за такі дії, як *post*, *get*, *put*, *delete*.

Для розробки API необхідні такі додатки до мови програмування *Python* як *Flask* - простий і легкий у використанні фреймворк для створення веб-додатків, а також *Flask-RESTful* - розширення для *Flask*, яке дозволяє розробити *REST API* швидко і з мінімальною налаштуванням.

Встановлення *Flask* виконується наступною командою:

pip install flask-restful

Приклад коду інсталяції показанийна рис. 3.7.

Аби була можливість виконувати операції зв’язаних з базою даних: створення автоматичного запиту, отримання відповіді від серверу, або закриття заявки використовується метод *UseModel*, метод впровадження в класі *apis.*

Як і в більшості клієнт-серверної інфраструктури для з’єднання з Інтернетом створено декілька різних екземплярів класів, аби повертати точні відповіді на запити клієнтів.

```
from flask import Flask
from flask_restful import Api, Resource, reqparse
import random
app = Flask(__name__)
api = Api(app)
```

Рис. 3.7. Імпорт необхідних модулів і налаштування *Flask*

У цьому сніпеті *Flask*, *Api* і *Resource* - це класи, які потрібні для базової роботи додатку. *Reqparse* - це інтерфейс парсинга запитів *Flask-RESTful*. Також знадобиться модуль *random* для тестового відображення випадкових даних.

Ресурсний клас *Quote* буде визначати операції ендпоінтов нашого *API*. Усередині класу потрібно заявити 4 методи: *get*, *post*, *put*, *delete*.

Після цього відкриваємо необхідний контролер в коді та розробляємо функції, які мають необхідні методи (табл. 3.1). Фрагмент коду додано на рис.3.8.

```
def accounts(self):
    return self.request('accounts')

def activity(self):
    return self.request('activity')

def balances(self):
    return self.request('balances')

def push_notifications(self):
    return self.request('pushNotifications')

def open_orders(self):
    """
        FIXME: untested
    """
    return self.request('orders', json=True)

def alerts(self):
    all_alerts = self.request('alerts', json=True)['data']
    open_alerts = pd.DataFrame(all_alerts['open_alerts'])
    alert_history = pd.DataFrame(all_alerts['alert_history'])
    return alerts(open_alerts=open_alerts, alert_history=alert_history)

def exchanges(self):
    return self.request('exchanges')

def markets(self, exchange):
    return self.request('markets',exchange_code=exchange)

def history(self, exchange, market):
    return self.data(exchange=exchange, market=market, data_type='history')['history']

def asks(self, exchange, market):
    return self.data(exchange=exchange, market=market, data_type='asks')['asks']

def bids(self, exchange, market):
    return self.data(exchange=exchange, market=market, data_type='bids')['bids']

def orders(self, exchange, market):
    return self.data(exchange=exchange, market=market, data_type='orders')
```

Рис. 3.8. Створення методів *api_rest.py* для роботи з *API*

Таблиця 3.1

API методи в рамках *HTTP*

HTTP	*URL*	Властивість
POST	*/open*	Запит на відкриття нової заявки
GET	*/list*	Запит на отримання списку усіх інцидентів
GET	*/data/code /:taskinfo*	Запит на отримання додаткової інформації про інцидент
POST	*/close*	Запит на закриття інциденту
POST	*/add*	Запит на створення нового інциденту в ручному режимі

3.5. Створення інтерфейсу додатку

Файли інтерфейсу що реалізують візуальну частину системи розташовані в папцв в папці \pub\home\view\. Система для моніторингу та роботи з користувачами складається з декількох модулів, тому необхідно інтерфейс розкласти на на різні файли – *main.html, dashboard.html, incedents.html, anomaly.html, users.html, acces-sers.html, login.html, restore.html*. В даних файлах розміщено *HTML*-код з використанням *Python* шаблонізатору *Jinja*, який дозволяє швидко реалізовувати функціонал інтерфейсу для систем з використанням мови програмування *Python*.

На рис.3.10 продемонстровано фрагмент коду *main.html*, який містить *HTML* код для головної сторінки додатку.

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Flask Template Example</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="http://netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css" rel="stylesheet" media="screen">
    <style type="text/css">
      .container {
        max-width: 500px;
        padding-top: 100px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <p>My string: {{my_string}}</p>
      <p>Value from the list: {{my_list[3]}}</p>
      <p>Loop through the list:</p>
      <ul>
        {% for n in my_list %}
        <li>{{n}}</li>
        {% endfor %}
      </ul>
    </div>
    <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
    <script src="http://netdna.bootstrapcdn.com/bootstrap/3.0.0/js/bootstrap.min.js"></script>
  </body>
</html>
```

Рис.3.9 Фрагмент коду файлу *main.html*

Сторінка роботи з користувачами описує файл login*.html* (рис.3.10)

```python
def validate_signup_form():
    username = request.form["username"]
    password = request.form["password"]
    verify_password = request.form["verify_password"]
    email = request.form["email"]
    username_error = ""
    password_error = ""
    verify_password_error = ""
    email_error = ""
    empty_input_msg = "Please submit a valid input."
    invalid_name_pass_msg = "Must be between 3 and 20 characters, no spaces."
    invalid_verify_pass_msg = " Must match the previous password."
    invalid_email_msg = "Please submit a valid email."
    if is_empty(username):
        username_error = empty_input_msg
    else:
        if is_invalid_char(username) or is_invalid_length(username):
            username_error = invalid_name_pass_msg
    if is_empty(password):
        password_error = empty_input_msg
    else:
        if is_invalid_char(password) or is_invalid_length(password):
            password_error = invalid_name_pass_msg
    if is_empty(verify_password):
        verify_password_error = empty_input_msg
    else:
        if is_invalid_verify_pass(password, verify_password):
            verify_password_error = invalid_verify_pass_msg
    if is_invalid_email(email):
        email_error = invalid_email_msg

    if not username_error and not password_error and not verify_password_error and not email_error:
        return redirect("/welcome?username={0}".format(username))
    else:
        return render_template("signup_form.html",username=username, password='', verify='', email=email,
                               username_error=username_error, password_error=password_error,
                               verify_password_error=verify_password_error, email_error=email_error)
@app.route("/welcome")
def welcome():
    username = request.args.get("username")
    return render_template("signup_welcome.html",name=username)
app.run()
```

Рис.3.10. Фрагмент коду файлу *login.html*

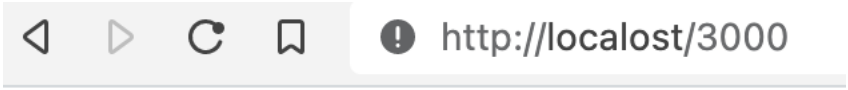
Для опису стилів використано *CSS* з використанням препроцесору LESS. Перевага *LESS* в тому що технологія надає можливість використання змінних, що недоступні в *CSS* без препроцесору. На рис.3.11 показано початкову коду з використанням *LESS* що задає стилі для *HTML* сторінки *users.*

```
body {
    background-image: url("localhost/imgs/123.jpg");
    background-repeat: no-repeat;
    background-size: cover;
    background-position: center;
    background-attachment: fixed;
}

.fs-it-btn {
    border: 1px solid darkslategrey;
    border-radius: 0;
    color: #fff;
    font-weight: bold;
}

.fs-it-btn-vertical-line {
    text-align: center;
    padding: 4px 0 5px 10px;
    margin-left: 10px;
    border-left: 1px solid darkslategrey;
}

.bg-blue {
    background-color: #2c9deb;
}

.text-uppercase {
    text-transform: uppercase;
}

.btn {
    padding: .675rem .95rem;
}

.content-container {
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
```

Рис.3.11. Фрагмент коду *CSS* для сторінки *users.css*

3.6. Робота користувача з додатком

Для тестового запуску додатку в веб-браузері необхідно для початку запустити веб-сервер: а потім відкрити шлях L*ocalHost.* Або ввести зазначену в коді *IP*-адресу та порт. Порт можна вказати 3000. Приклад сторінки показано на рис. 3.12.



Test

Рис. 3.12. Запит сторінки додатку у браузері

Після запуску додатку у браузері, якщо код програми створений правильно то відкриється основна тестова сторінка, як показано на рис. 3.13.

Рис. 3.13. Головна сторінка входу в веб-додаток

Програмна реалізація системи з використанням *CSS* дає можливість працювати з нею з мобільних пристроїв (планшетів, смартфонів). На рис.3.14 показано роботу з системи з мобільного пристрою який працює на базі ОС *Android.*

Рис. 3.14. Сторінка авторизації в системі з мобільного пристрою

Для того, щоб користувач міг відновити доступ до системи самостійно передбачено модуль, який реалізує процес відновлення паролю з використанням *SMS*-підтвердження. Код відправляється з використанням стороннього *API* після натиснення кнопки «Надіслати *SMS*» (рис. 3.15).

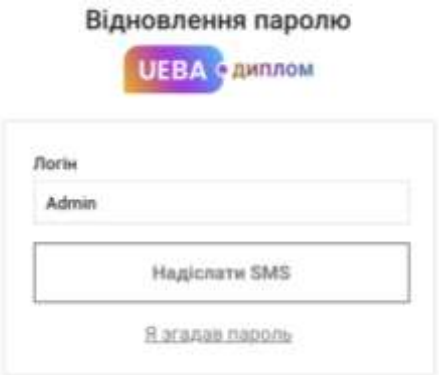


Рис. 3.15. Форми подачі заявки на відновлення паролю

Якщо дані для відновлення доступу до акаунту успішно відправлені користувачу то передбачено екран з повідомленням що "Деталі для відновлення паролю надіслані в *SMS* повідомленні на ваш номер телефону" (рис. 3.16).

Рис. 3.16. Процес відновлення паролю розпочато

Для того щоб почати взаємодію з системою, всім користувачам необхідно пройти авторизацію: ввести необхідний дані(логін та пароль) і натиснути кнопку "Вхід" (рис. 3.17).

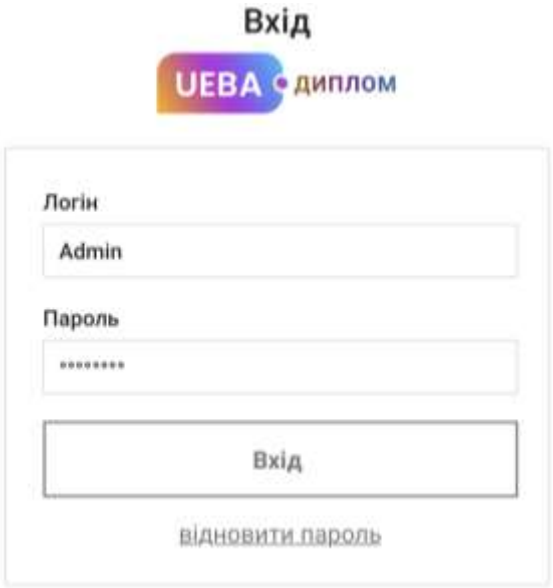


Рис. 3.17. Форми авторизації за допомогою ПК

Після успішної авторизації в систему адміністратор має можливість здійснювати моніторинг основні параметрів інфраструктури в реальному часі (рис. 3.18).

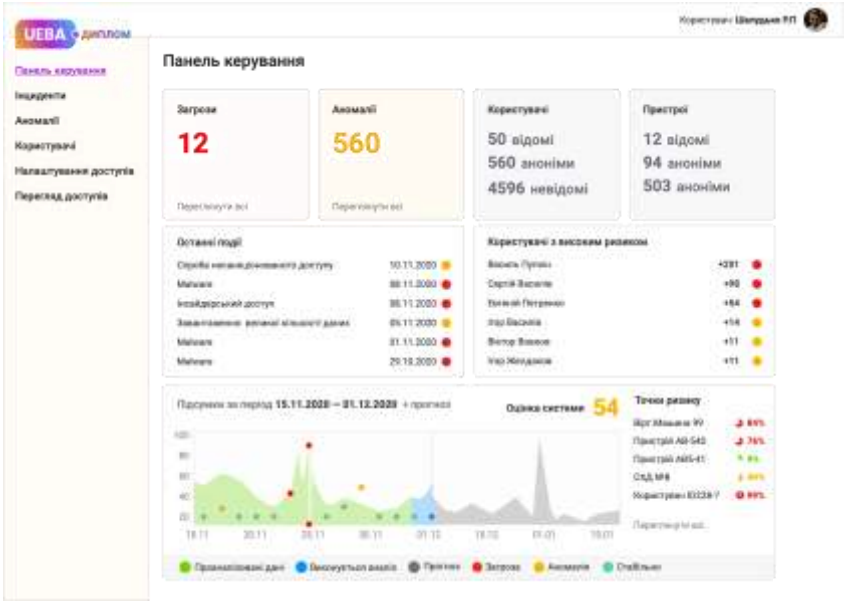


Рис. 3.18. Головна сторінка системного адміністратору в системі

Адміністратор системи з використанням технології *UEBA* на головні сторінці панелі керування кабінеті переглядає всі інциденти та аномалії (рис. 3.19). Тут адміністратор бачить основні дані про систему та додаткові параметри без можливості керування, – тільки моніторинг.

Рис. 3.19. Кількість загроз та аномалій в мережі компанії

Адміністратор мережі також має можливість бачити останні події в мережі, які вже були оброблені автоматично чи іншим адміністратором. Система автоматично збирає дані про користувачів в мережі та відповідно до свої критеріїв безпеки відображає їх в таблиці "Користувачі з високим рівнем ризику" (рис. 3.20)

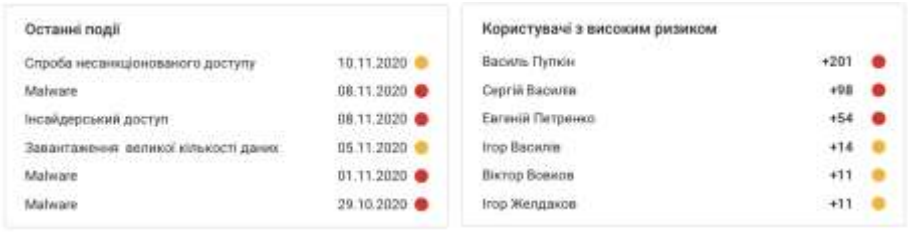


Рис. 3.20. Аномалії, події та користувачі в зоні ризику

Основна можливість системи з використанням технології *UEBA* це функціонал по аналізу поведінки користувачів в реальному часі а також прогнозування можливих загроз для мережі. Приклад роботи алгоритму, який на графіку відображає проаналізовані дані, дані які аналізуються зараз, прогноз, а також загальний рівень безпеки системи і точки з високим ризиком в мережі (рис. 3.21).

Рис. 3.21. Підсумки роботи системи та прогноз можливих загроз

Модуль "Інциденти" (рис. 3.22) в системі відображає підбірку подій які вже відбулись в мережі компанії, пройшли верифікацію та знаходяться в процесі обробки відповідним підрозділом.

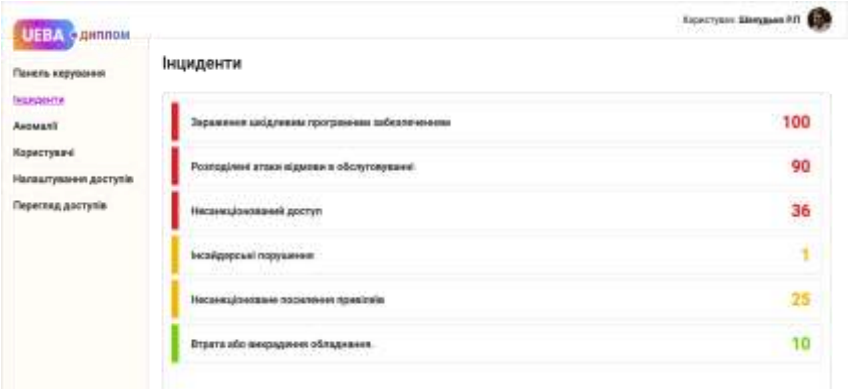


Рис. 3.22. Розділ системи "Інциденти" зі списком подій

Кожен інцидент має пріоритет, назву, опис, оцінку та додаткові дані, які вдалось зібрати автоматично чи адміністратору.

Модуль "Аномалії" – це розділ в системі, в якому знаходиться список виявлених аномалій в мережі компанії. Аномалії (рис. 3.23) створюються системою автоматично відповідно до її налаштувань та бази аномалій.

Кожна аномалія має назву, пріоритет, оцінку критичності, а також кількість подобних аномалій. В даному розділі аномалію по замовчуванням відсортовані по даті, що дозволяє адміністратору легко виявляти події, які виникли останніми.

Рис. 3.23. Розділ системи "Інциденти" зі списком подій

Передбачено також можливість виконати сортування по кількості подібних аномалій, пріоритету чи по оцінці критичності. В системі також реалізовано інтелектуальний режим сортування, який враховує також складність проблеми та вірогідність поширення аномалії на іншій ланки в мережі (рис. 3.24).

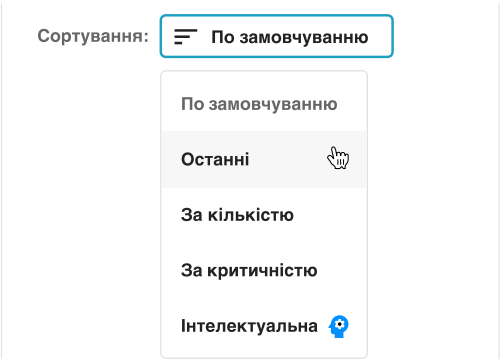


Рис. 3.24. Способи сортування подій в розділі "Аномалії"

Кожна аномалія має автоматично згенерований набір інформації, яка може використовуватись адміністратором.

Деталі аномалії (рис. 3.24) містять ідентифікаційний номер аномалії, назву, кількість та оцінку. Якщо система змогла "зв’язати" декілька подій в одну аномалію, то такі події будуть об’єднані в одну аномалію. Це також призведе до підвищення пріоритету такої аномалії.

Подія містить такі властивості як дані про користувача який виконав дію, дату події, та інші додаткові дані в залежності від типу аномалії.

Наприклад, якщо система "побачила" що Користувач вперше переглянув файли с папки до якої він раніше ніколи не звертався то це буде аномалією з низькою оцінкою критичності. Якщо користувач переглянув декілька файлів з цієї папки під час однієї сесії – це додатковий фактор, що підвищить рівень критичності аномалії.

Система з використанням технології *UEBA* збирає інформацію про те, скільки часу в середньому користувачі переглядають файл та з якою частотою. Якщо користувач відкрив 8 файлів за декілька секунд, – це також буде вважатись додатком фактором для підвищення критичності події. Так як система збирає велику кількість користувацьких даних, то вона може порівнювати середні значення та надавати адміністратору відповідну статистику (рис. 3.25).

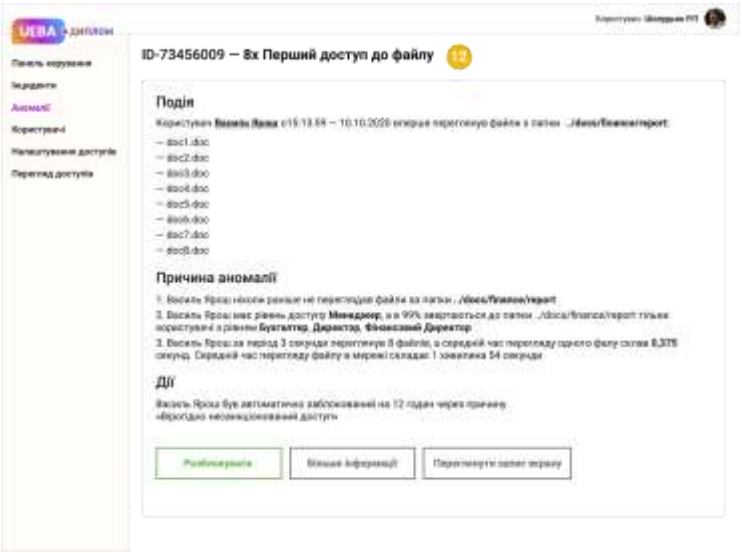


Рис. 3.25. Деталі аномалії що доступні для аналізу

Аномалія "Перший доступ до файлу" виникла тому що система виявила такі події:

– Користувач ніколи раніше не переглядав файли за папки *../docs/finance/report;*

– Користувач має рівень доступу Менеджер, а в 99% звертаються до папки *../docs/finance/report* тільки користувачі з рівнем Бухгалтер, Директор, Фінансовий Директор;

– Користувач за період 3 секунди переглянув 8 файлів, а середній час перегляду одного фалу склав 0,375 секунд. Середній час перегляду файлу в корпоративні мережі складає 1 хвилина 54 секунди.

Після проведення аналізу система автоматично може прийняти рішення про блокування користувача, про продовження спостереження чи про те, що необхідна дія адміністратору. Крім цього система пропонує адміністратору що можна зробити з цією аномалією: "Користувач був автоматично заблокований на 12 годин через причину "Вірогідно несанкціонований доступ".

В системі реалізована можливість переглянути інформацію про користувача, який зв’язаний (рис. 3.26) з зазначеною аномалією.

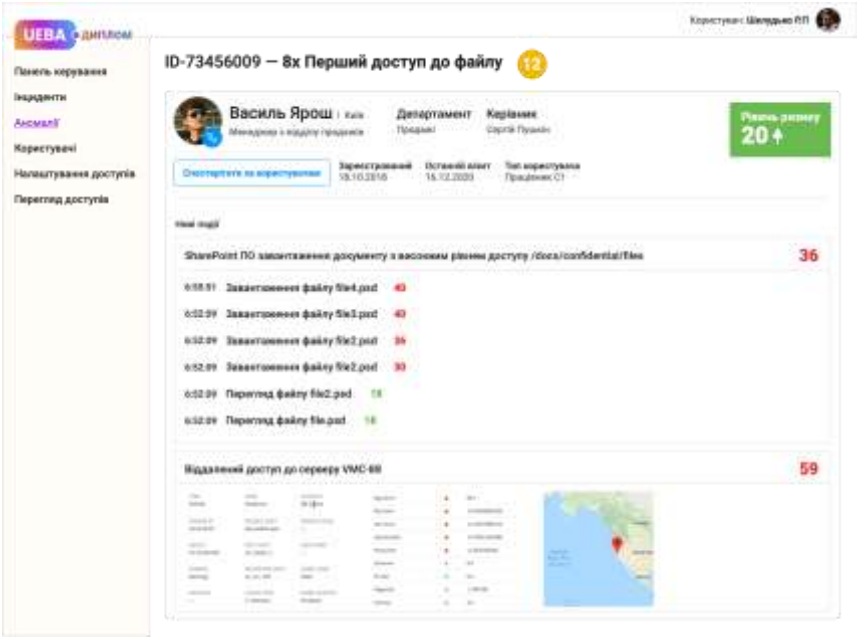


Рис. 3.26. Інформація про користувача який є учасником аномалії

На сторінці інформації користувача, який задіяний в аномалії вказана така загальна інформація про користувача як дата реєстрації, дата останнього візиту, рівень доступу користувача, департамент в якому працює користувач та його керівник. Кожен користувач також має оцінку – рівень ризику від 1 до 100. Рівень ризику визначає система автоматично в залежності від кількості подій, в яких задіяний користувач, від його поведінки в системі та різних додаткових факторів.

Адміністратор має можливість переглянути як користувач взаємодіє з системою, які файли переглядає чи завантажує, з якого пристрою, *IP*-адресу та якого місцезнаходження відвідує корпоративні ресурси. Корпоративна мережа може використовуватись десятками, сотнями чи навіть тисячами користувачами, тому для роботи з користувачами всередині системи моніторингу користувачів існує модуль "Користувачі", який відображає список користувачів, які зареєстровані в системі. Кожному користувачу призначено внутрішній "рейтинг" в залежності від поведінки користувача в мережі.

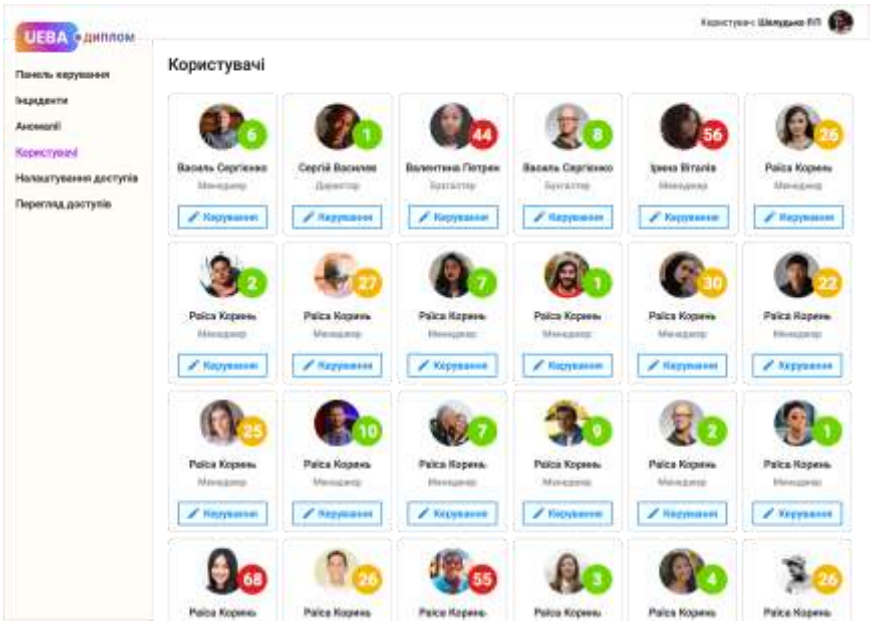


Рис. 3.27. Сторінка зі списком усіх користувачів в системі

В панелі керування користувачами адміністратор має можливість керувати кожним користувачем окремо, а також переглядати додаткові дані та властивості обраного користувача. Частина даних про користувача прихована від

адміністратора і знаходиться лише в базі даних та доступна тільки для машинного аналізу.

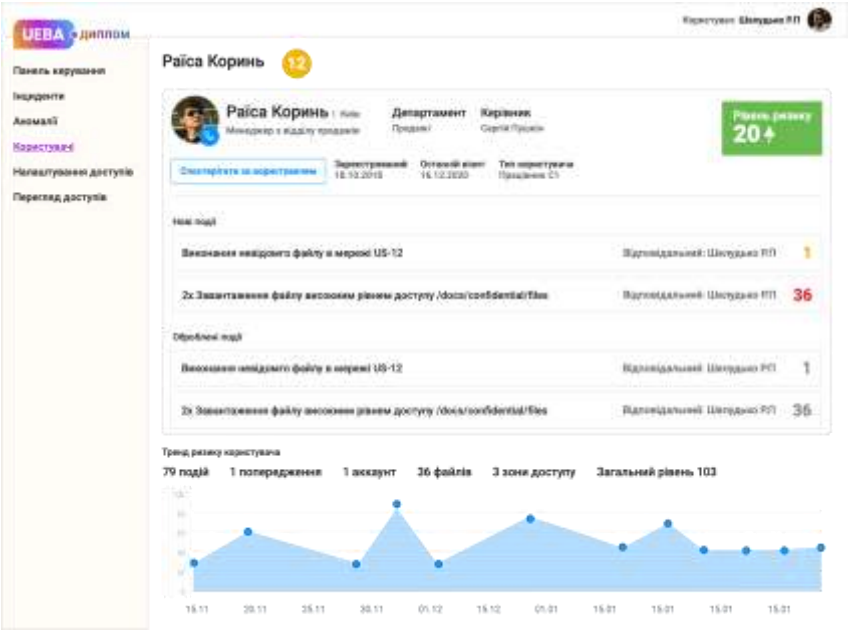


Рис. 3.28. Сторінка користувача з статистикою подій

Серед особливостей технології *UEBA* є те, що система може прогнозувати тренд ризику користувача, що дозволяє не тільки переглядати історію рівня ризику так і створювати прогноз для кожного користувача (рис. 3.29). Це дозволяє адміністратору системи розуміти рівень ризику більш точно.

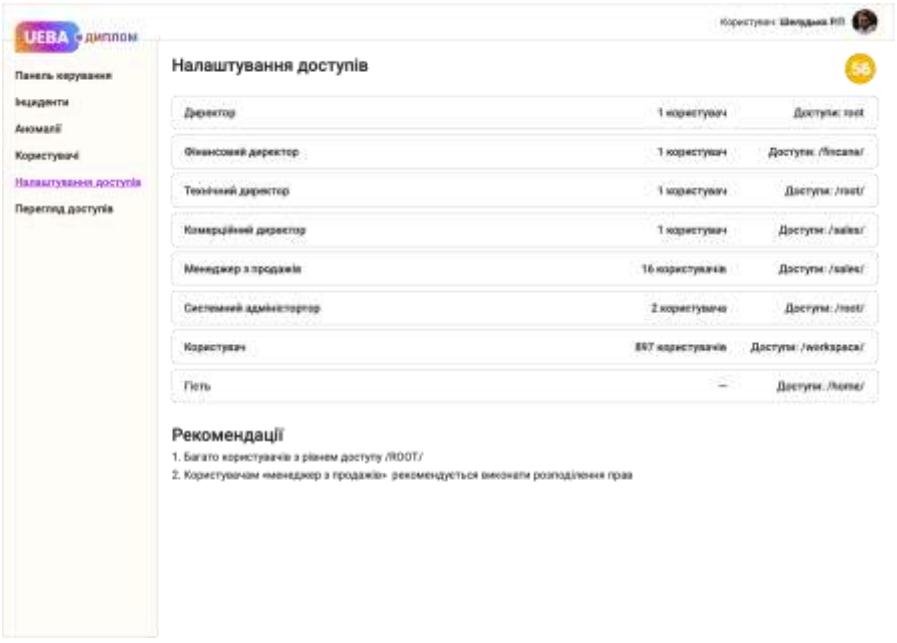


Рис. 3.29. Сторінка налаштування доступів для користувачів

Якщо рівень ризику постійно зростає, то можна робити висновки що користувач вірогідно займається збором інсайдерської інформації. Якщо рівень ризику "середній по системі", то рівень ризику низький. Нульовий рівень ризику сигналізує що *UEBA*-налаштування задані некоректно, а система не виконує свої задачі.

Користувачі мають ключове значення для контролю доступу до функцій і того, як все здійснюється кожним користувачем, як фізичною особою в компанії.

Користувачі, зареєстровані в системі і асоційовані з конкретними правами (рис.), можуть мати різні рівні доступу для інфраструктури, а адміністратор мережі може виконувати такі налаштування:

– Надання / заборони доступу до додатку;

– Отримання інформації з бази даних;

– Управління доступом до певних функцій програми;

– Історія роботи користувача;

– Призначення задач окремим співробітникам.

Кожен користувач може брати участь в декількох групах. Це дозволяє призначати права доступу (які призначені групам) користувачам за допомогою комбінування груп.

Висновки

Програмне забезпечення системи з використанням технології *UEBA,* - це сукупність методів та правил, описів та інструкцій, математичних алгоритмів і моделей, що призначені для обробки великих масивів інформації та даних, для прийняття відповідних рішень автоматично, а також технічна документація, що розроблена для системи моніторингу користувачів, яка дозволяє використовувати комп’ютерні системи для більш точного виконання задач, а також для модифікації уже створеного рішення.

Системи аналізу поведінки користувачів і сутностей (*User and Entity Behavior Analytics - UEBA*) - рішення, спрямовані на пошук і виявлення аномалій в поведінці користувачів і різних систем.

Клас рішень поведінкового аналізу з'явився так як для забезпечення інформаційної безпеки в компаніях впроваджують безліч різних систем збору даних. При цьому співробітники не завжди здатні переглядати всі отримані відомості і вчасно реагувати на потенційні інциденти. Складаючи профілі, системи *UEBA* підвищують ефективність і забезпечують своєчасне реагування на можливі витоки даних.

Системи *UEBA* дуже близькі до рішень поведінкового аналізу користувачів (*UBA*) і, по суті, є їх продовженням. Відмінністю цих систем один від одного служить наявність в *UEBA* профілювання та аналізу сутностей, під якими розуміються додатки, системи зберігання даних, мережевого трафіку, пристрої, сервери і дані. Крім цього, системи *UEBA* дозволяють вирішувати проблеми не тільки внутрішніх витоків конфіденційних даних, але і зовнішніх цілеспрямованих на систему атак.

В четвертому розділі виконано роботу по проектуванню та розробці програмного забезпечення системи моніторингу користувачів з використанням технологіях *UEBA.* Проєктування програмного забезпечення розпочато з розробки прототипу, моделювання сутностей та діаграм.

Діаграми роботи системи представляють специфікацію різних варіантів щодо використання системи. Проаналізовані варіанти використання програмного забезпечення, ролі користувачів і моделі та алгоритми роботи у системі дозволяють повно представити адміністратору стан мережі, рівень загроз та загальну оцінку "здоров’я" системи.

Також було описано та розроблено структуру програми та алгоритм роботи системи. На хмарному серверному обладнанні було створено сервер БД та платформу з використанням мови програмування *Python,* на якій і буде розроблено додаток. Розроблено схему бази даних та ініціалізовано з’єднання до БД, розроблено, налаштовано та мінімізовано схему даних.

Інтерфейс веб-додатку було з використанням популярних мов програмування *JavaScript, HTML, CSS* та *AJAX*. Детально описано та показано фунціонал системи та приклад роботи користувача з ним.

РОЗДІЛ 4
АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ ПОВЕДІНКОВОЇ АНАЛІТИКИ UEBA

Одною з найважливіших характеристик обчислювальної системи (ОС) є її продуктивність. Продуктивність ОС визначається кількість обчислювальної роботи за одиницю часу. Важливими факторами, що впливають на продуктивність ОС є першою чергою такі фактори, як: тактова частота роботи процесора системи, число команд програм (задач, алгоритмів) та число тактів для виконання однієї команди (середній час виконання однієї команди).

4.1 Підготовка тестового середовища

Для тестування роботи додатку використані інструмент тестування навантаження *k6* та хмарний сервіс *Digitalocean.*

GitLab CI використовує *YAML* файл *.gitlab-ci.yml* для визначення конфігурацій проекту, що включають в себе визначення всіх етапів, які будуть виконуватися після того, як конвеєр *CI/CD* запускається у відповідь на *git push / merge*. Потрібно провести *unit*-тест над простим проектом, щоб переконатися, що в коді немає помилок.

У наведеному файлі конфігурації *YAML* (рис. 4.1) процес збірки розбито на 3 етапи. Кожен з етапів це просто *gulp.task*, заданий в *gulpfile*. Так як у нас встановлений *Python*, користувач може окремо запускати будь-який з етапів.

Але в *GitLab C*I потрібно вказати, який з образів *Docker* потрібен.

Це вузол: 6.11.2. Крім того, даний *Docker*-атрибут можна задати всередині певного етапу, тому ви зможете використовувати різні інструменти для будь-якого з етапів.

```
stages:
  - lint-css
  - lint-js
  - unit-test

image: node:6.11.2


lint css:
  stage: lint-css
  before_script:
    - npm install
  cache:
    untracked: true
  only:
    - master

  script:
    - ./node_modules/gulp/bin/gulp.js lint-css

lint js:
  stage: lint-js
  cache:
    untracked: true
    policy: pull
  only:
    - master
  script:
    - ./node_modules/gulp/bin/gulp.js lint-js

run unit test:
  stage: unit-test
  cache:
    untracked: true
    policy: pull
  only:
    - master
  script:
    - ./node_modules/gulp/bin/gulp.js test
```

Рис. 4.1. Файл конфігурації збірки проекту на *Gitlab*

Далі за допомогою веб-інтерфейсу *Gitlab* необхідно запустити збірку проекту. Збірка відбувається автоматично і в середньому займає 10 хвилин.

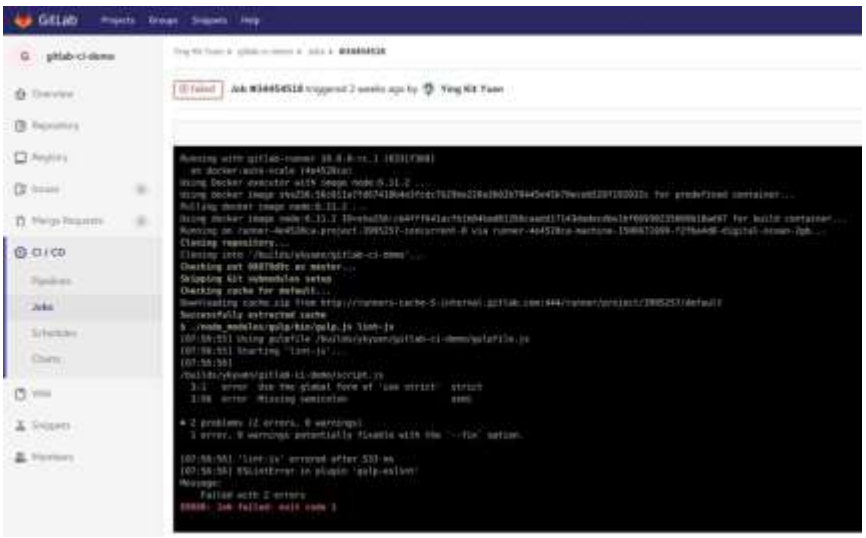


Рис. 4.2. Логи збірки *CI/CD* проекту за допомогою *GitLab*

Після виконання збірки буде доступна інформація про результати виконання. Успішно пройдено чи збірку провалено. Якщо відбулась помилка то дані доступні для перегляду в повному описі пайплайну.

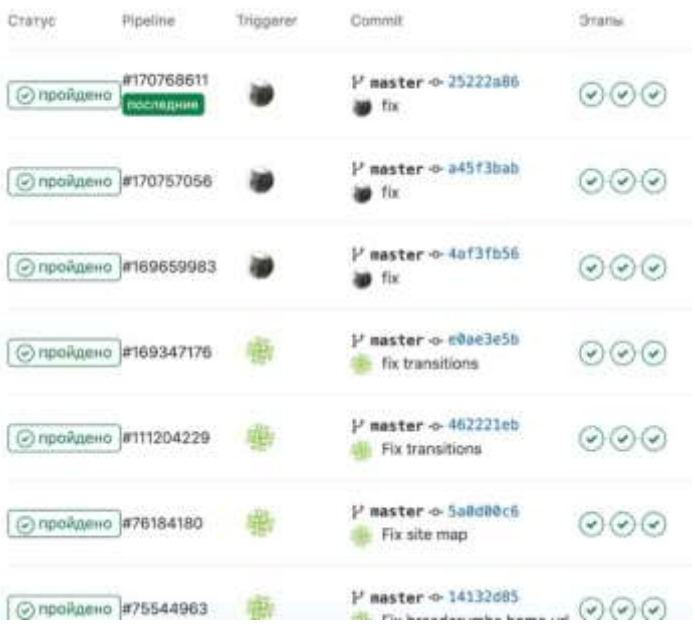


Рис. 4.3. Пайплайни збірки та розгортання проекту на веб-сервері

Таким чином, в результаті збірки проекту, додаток буде автоматично розгорнуто на веб-сервері та він стане доступним для тестування.

4.2. Аналіз ефективності роботи системи

Після запуску відповідних тестів для перевірки веб-сервісу було отримано дані телеметрії від *k6* сервісу для проведення тестів та візуалізації результатів. Перевірці підлягали ключові показники роботи веб-сервісу, такі як кількість помилок, час відповіді, кількість даних що обробляються.

Також виконувався моніторинг навантаження на веб-сервер таких параметрів як навантаження на процесор, використання оперативної пам’яті, використання мережі та навантаження на дискову підсистему.

Для запуску тесту необхідно виконати програму, в якій задано такі параметри тестування навантаження як кількість запитів, час проведення, ціль

тестування а також зони, з який буде відбуватись тестування. В рамках даного проекту булу обрано зону *Frankfurt.*

```
import { sleep } from "k6";
import http from "k6/http";

export const options = {
  stages: [
    { duration: "2m24s", target: 25 },
    { duration: "7m12s", target: 50 },
    { duration: "2m24s", target: 0 },
  ],
  ext: {
    loadimpact: {
      distribution: {
        "amazon:de:frankfurt": {
          loadZone: "amazon:de:frankfurt",
          percent: 100,
        },
      },
    },
  },
};

export default function main() {
  let response;

  response = http.get("my-ueba-service");

exit();
```

Рис. 4.4. Код програми для проведення тестування навантаження

До проведення оптимізації архітектури системи показником кількості одночасних в секунду запитів було значення на рівні 10-20 *RPS,* з показником до 20-25% запитів які оброблювались не успішно під час максимального навантаженя.

Високий показник відмов спричинений як тим, що на систему діє порівняно високе навантаження (250 користувацьких з’єднань), так і слабким процесором і малою кількістю оперативної памя’ті що виділено для сервісу в хмарі.

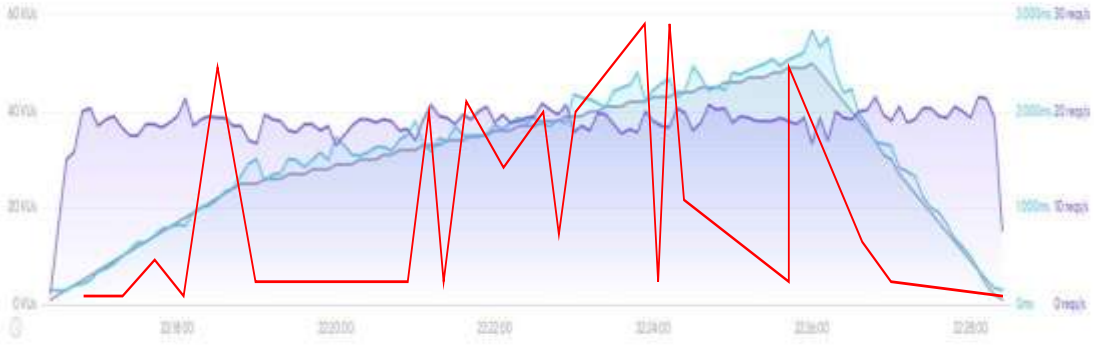


Рис. 4.5. Значення *RPS* для сервісу без оптимізації

На рис. 4.5 приведено значення *RPS* для сервісу який не було оптимізовано, а також лініями показано різкий ріст запитів які не могли бути оброблені під час високого чи максимального навантаження.

4.3. Аналіз ефективності взаємодії користувача з системою

Для проведення аналізу взаємодії користувача з веб-інтерфейсом було використано програму *xlabsgaze*, яка здійснює моніторинг руху очей користувача за допомогою веб камери, та будує теплову карту того, де користувач концентрує свою увагу, а які зони екрану проглядає лише частково.

В проведенні аналізу була задіяна фокус група із трьох користувачів з різним рівнем володінням ПК, від звичайного користувача до системного адміністратору.

Рис. 4.6. Теплова карта використання сторінки "панель керування"

На рис. 4.6 видно що користувач орієнтується по кольорам на сторінці та акцентує увагу на тих елементах, що виділені зеленим чи червоним кольором. Таким чином в інтерфейсі кольором необхідно показувати лише важливі елементи, а звичайний текст та додаткова інформація може бути показана звичайним чорнім кольором.

Також слід зазначити що користувачі акцентують увагу на секціях в інтерфейсі, тому важливо розділяти різні види інформації на сторінці за допомогою окремих блоків з інформацією.

При проведенні аналізу взаємодії користувача зі сторінкою "Інциденти" виявлено (рис. 4.7) що користувач спочатку переглядає елементи що вказані вверху сторінки, а потім звертає увагу на текст та додаткову інформацію по праву сторону екрану.

Рис. 4.7. Теплова карта використання сторінки "інциденти"

Сторінка інциденту містить велику кількість інформації та може бути складною для непідготовленого користувача. Про це свідчить те, що зони концентрації уваги на теплові карті поширені по усій сторінці як по горизонталі так і по вертикалі.

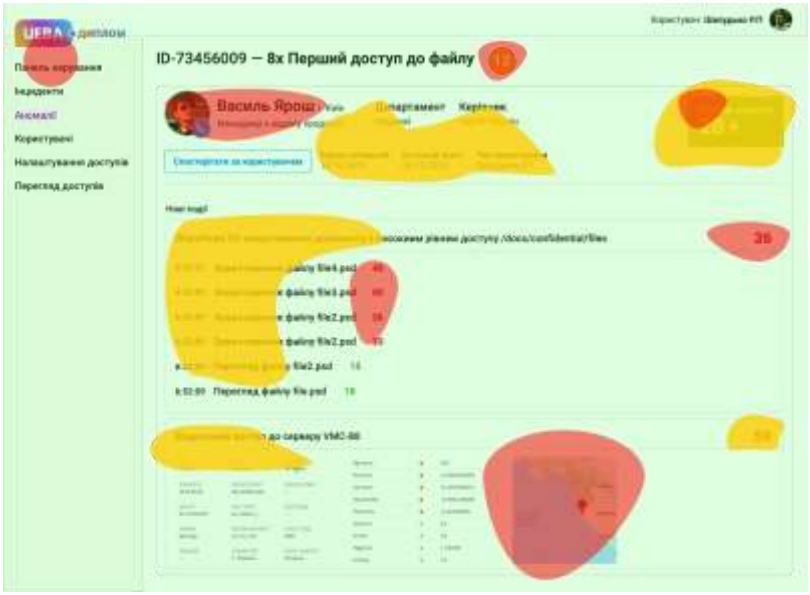


Рис. 4.8. Теплова карта використання сторінки інциденту

Під час навігації по сторінці користувача, учасники фокус групи часто звертали увагу на графічні елементи, а також на елементи що виділені кольором. Також користувачі приділяли увагу графіку подій, а особливо точкам з різким ростом.

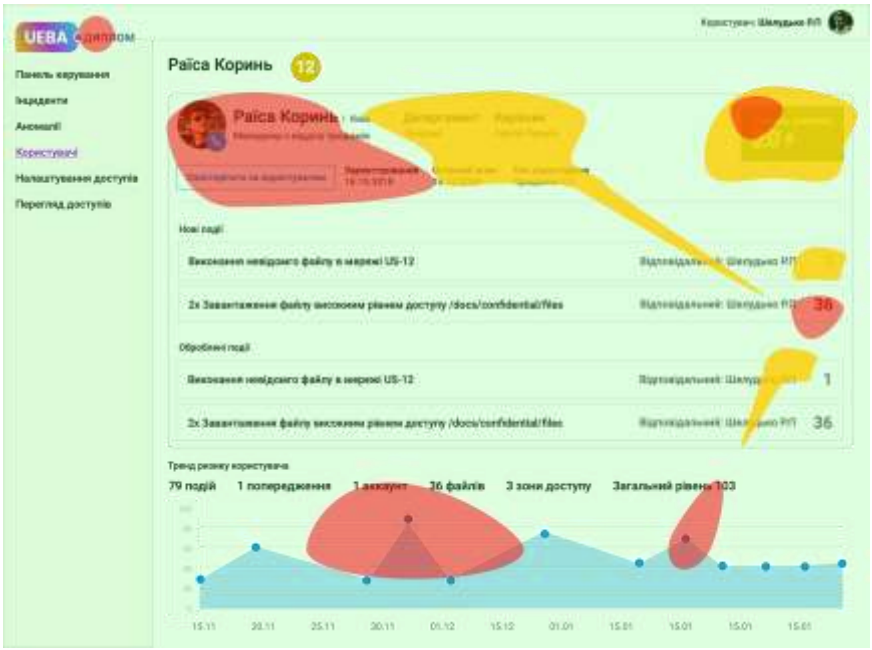


Рис. 4.9. Теплова карта використання сторінки обраного користувача

4.4. Аналіз результатів роботи

Після проведення доробки коду модулів та оптимізації архітектури показники кількості запитів (*RPS*) за секунду склала 40-50 *RPS*, з показником відхилених запитів на піку навантаження в 0%. Відсутність помилок а також покращення ефективності роботи веб-сервісу спричинене тим, що використовуються нові модулі для роботи з чергами на обробку даних. На рис. 4.10 приведено графік з показниками кількості оброблюваних запитів за секунду часу для вже оптимізованого сервісу в хмарі.

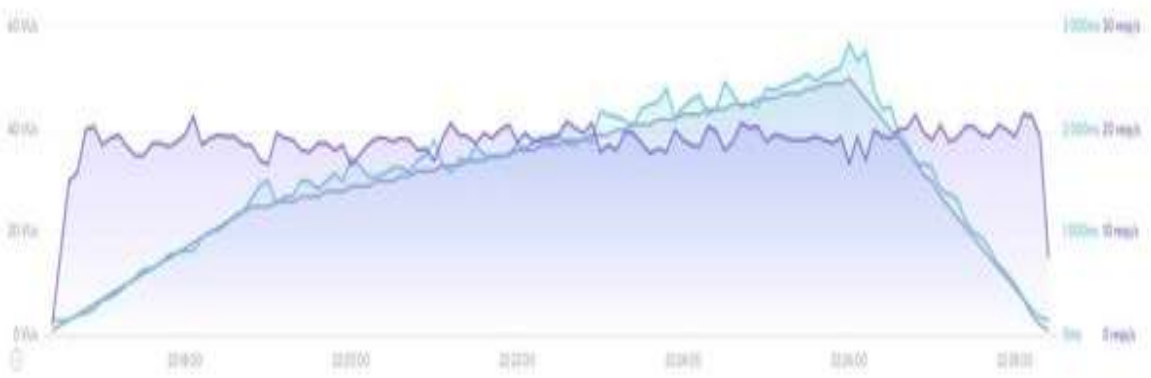


Рис. 4.10. Дані про кількість запитів для оптимізованого сервісу

Також, в результаті проведення тестування навантаження, отримані наступні дані роботи системи:

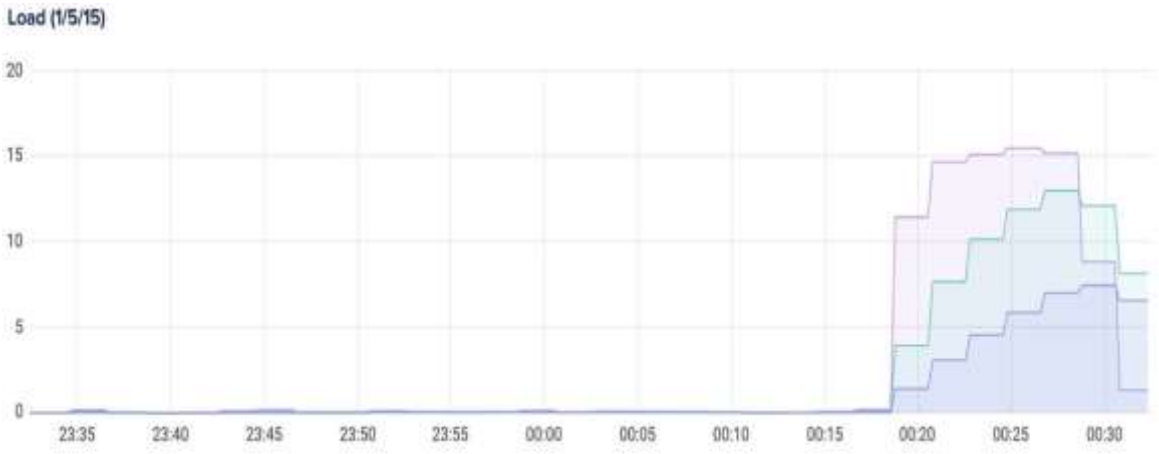


Рис. 4.11. Порівняння коефіцієнту навантаження до та після оптимізації

Після оптимізації навантаження на процесор знизилось на 50-60%.

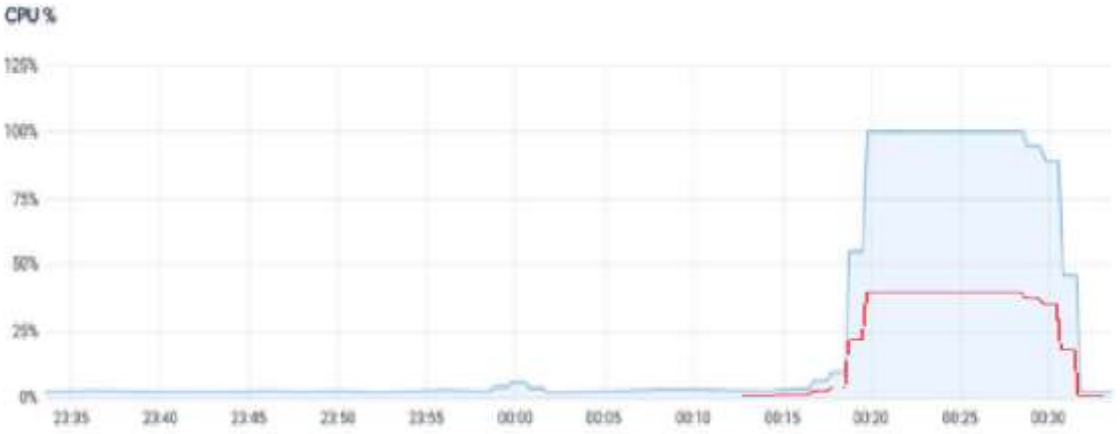


Рис. 4.12. Порівняння навантаження на процесор до та після оптимізації

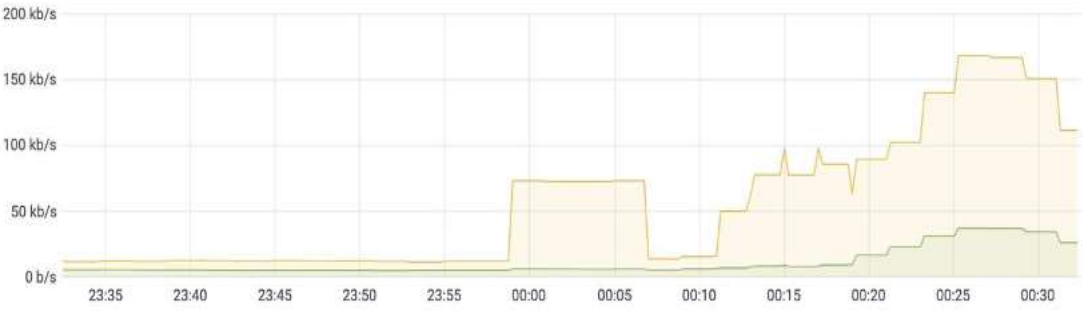


Рис. 4.13. Порівняння навантаження на дискову підсистему до та після оптимізації

Результатом оптимізації можна вважати покращення показників кількості одночасних оброблюваних запитів та 50-60%. Кількість невданих(відхилених) запитів впала до відмітки0, так як запити які не було оброблено чекають в окремі черзі на обробку, а не віддають відповідь про помилку.

При однаковому навантаженні навантаження на процесор зменшилось на 60%. Дискова підсистема також працює стабільніше так як дані зберігаються в оперативні пам’яті. В рамках оптимізації користувацького досвіду та на базі роботи з фокус групою проведено дослідження інтерфейсу користувача

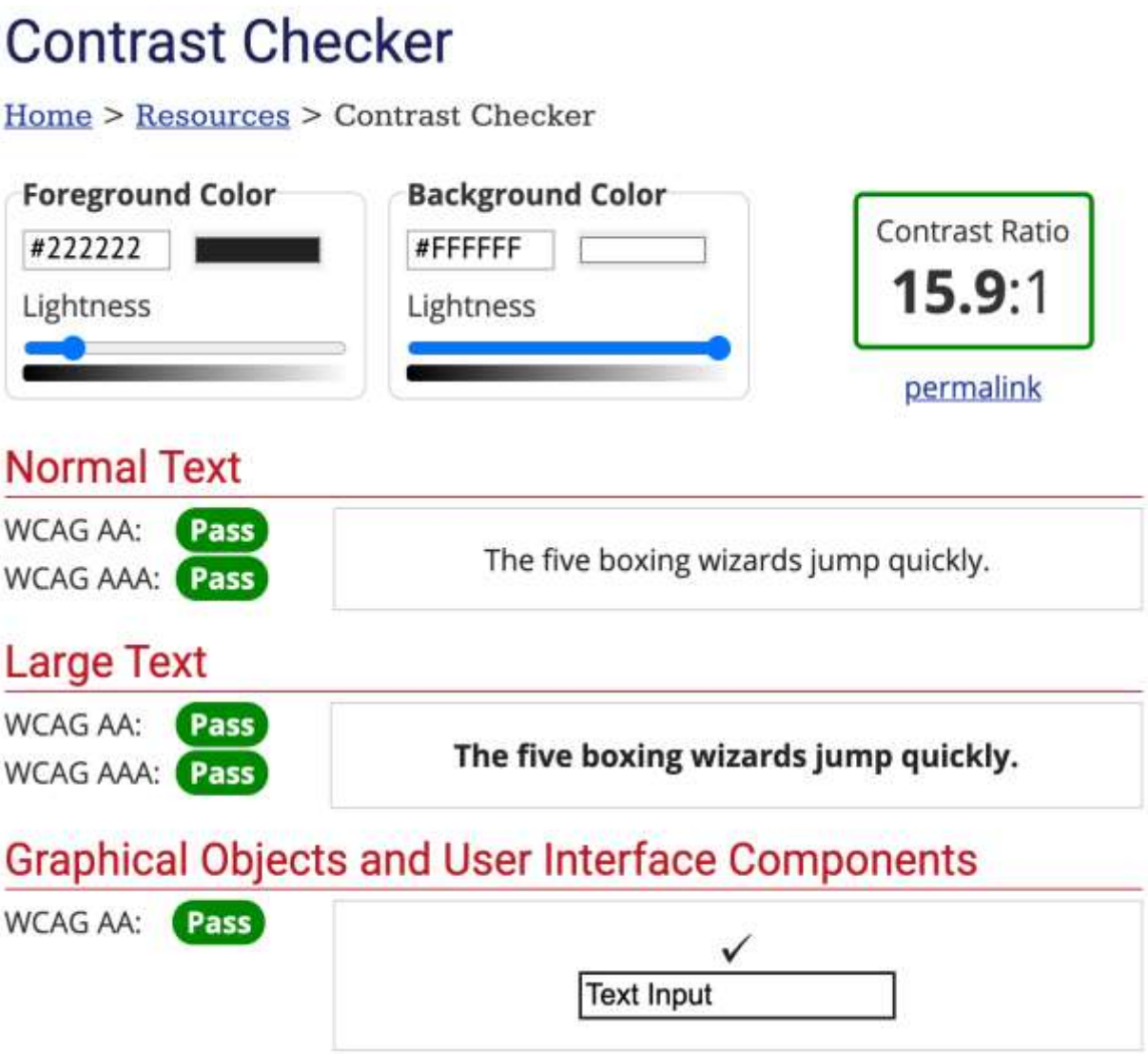


Рис. 4.14. Результат оптимізації контрастності інтерфейсу

Після роботи з фокус групою, результатом роботи є оптимізація користувацького інтерфейсу що дозволить знизити навантаження на очі у користувача, а також знизити часові втрати на навчання по роботі з системою.

Висновки

Для виконання перевірки ефективності роботи сервісу та його оптимізації, було розгорнуто веб-сервер на базі *Digitalocean*, та проведено генерацію не легітимного трафіку та емуляцію навантаження до 100 одночасних запитів за допомогою сервісу *k6*.

Результати замірів на графіках свідчать про покращення показників кількості оброблюваних запитів. Кількість успішно оброблених запитів збільшилась на 30-60%.

Таких результатів вдалось досягнуто завдяки використання модулю по роботі з чергами, зберігання даних в оперативній пам’яті а також використанні сервісу *digitalocaen* який гарантує на рівні *SLA* високий рівень доступності веб-сервісу що в свою чергу зменшує кількість запитів які не були оброблені.

Важливим елементом оптимізації також є використання сервісу *Gitlab* та підходу до розгортання *CI/CD*.

ВИСНОВКИ

Сучасні IT-технології стрімко розвиваються, тому важливо забезпечити своєчасне рішення проблем і завдань, що виникають в результаті стрімкого зростання числа людей які користуються глобальною мережею, і загроз в Інтернеті. Це створює додаткові виклики для розробників онлайн-сервісів в області як розподілених так і в хмарних обчисленнях.

Локальні, корпоративні та глобальні мережі постійно мають справу з хакерськими та інсайдерськими загрозами. Такі виклики можуть спричинити такі загрози як різкий ріст навантаженням, яке спричиняє різкий ріст пікового навантаження на мережу що спричиняє неочікувану відмову сервісу(*DoS).* Для того щоб забезпечити захист мережі, компанії використовують такі технології як *SIEM, UBA* чи *UEBA* в залежності вимог компанії до безпеки та її можливостей.

Вимоги щодо захищеності мережі, що стосуються мереж, стосуються як функціональних, так і нефункціональних вимог до характеристик сервісів. Кожна мережа має відповідати базовим параметрам захисту. Вони включають надійність, час обслуговування, ціну на інтеграцію нових рішень, цілісність, репутація, точність, стабільність та інші *QoS* що пов'язані з безпекою в мережі. Базові вимоги до розробників систем захисту мереж полягають у підвищенні якості обслуговування, обладнання та якості інших послуг що надаються.

Таким чином, для вирішення зазначених проблеми було проаналізовано існуючі підходи о проектування мереж, наявні технології та структури архітектур сервісів по забезпеченню захисту мережі.

Технологія *UEBA* – це найновіший підхід до захисту мережі будь-якого типу чи розміру. Технологія на 10-20% покращує ефективність роботи модулю з виявлення та розслідування передових загроз порівняно з іншими *SIEM* чи *UBA* рішеннями. *UEBA* працює на основі даних, які отримано за допомогою аналізу поведінки користувачів, використовуючи машинне навчання та поведінкове моделювання для визначення аномальної діяльності що мають високий ризик для інфраструктури компанії.

Інтеграція нових продуктів і модулів в мережу для будь-яких компаній завжди пов'язані з впровадженням нових технологій, що вимагають висококваліфікованої підтримки високої якості сервісів та послуг, що надаються в мережі користувачам.

В умовах постійного та динамічного розвитку IT-архітектури та сервісів які надаються клієнтам – необхідно підтримувати службу моніторингу зовнішніх і внутрішніх інцидентів. Збій в роботі сервісів може привести до втрати клієнтів та користувачів та значних фінансових та репутаційних втрат.

На відміну від стандартних систем захисту мереж, *UBA* та *UEBA* програмні технології підвищують на рівень складності при розробці, але вони вирішують багато наявних та можливих проблем і краще пристосовані до майбутніх типів загроз, які постійно змінюються та стають більш витонченими.

Для розробки системи була обрана технологія *UEBA* як найновіша технологія захисту мережі, оскільки вона дозволяє забезпечити найбільш потужний захист мережі, та має ряд ключових переваг для покращення показників виявлення загроз в мережі. Незважаючи на досить високі вимоги до апаратного забезпечення, технологія *UEBA* спроможна обробляти велику кількість запитів за секунду. Важливим елементом, який дозволяє забезпечити низький рівень відмов є черга, яка обробляє запити за принципом *FIFO*, – першим прийшов, першим пішов.

Застосування цього компоненту, дозволяє покращити характеристики сервісу який їх застосовує навіть на порівняно низьких обчислювальних ресурсах.

Для оптимізації системи з використанням технології *UEBA* було обрано хмару *DigitalOcean*. Для збірки проекту – *Gitlab.* В хмарі було розгорнуто застосунок на базі *Python.* Він являє собою програму, яка містить базу даних з інформацією про дії користувачів, модуль який забезпечує аналіз та додавання нових дій користувачів в базу, а також модуль який обробляє дані що поступають в режимі реального часу. Процес обробки даних асинхронний, дані записуються у персистентне сховище що забезпечує низький рівень ризику втрати даних. В програмі також закладена можливість вертикального та горизонтального

розширення потужності системи в разі різкого росту користувачів та інших причин пікового навантаження.

Для виміру якості та ефективності оптимізації, яку було проведено, було в хмарі *Digitalocean* розгорнуто окремо сервіс з оптимізацією та без оптимізації, так створено емуляцію навантаження за допомогою сервісу *k6*, що дозволяє створити навантаження схоже на реальне навантаження яке генерують легітимні користувачі.

Таким чином, після проведення робіт по розгортанню хмари, налаштування збірки проекту, – заміри ефективності роботи сервісу демонструють покращення ключових показників, а саме ріст кількості оброблених запитів за секунда та зменшення запитів які не було оброблено.

Запропонований підхід до обробка даних дозволив обробити більшу кількість даних за одиницю часу, без збільшення апаратних ресурсів, що в свою чергу дозволяє забезпечити більш якісний захист системи та не витрачати ресурси на горизонтальне чи вертикальне масштабування.

Роботи по оптимізації, що запропоновано в даній роботі рекомендується застосовувати при розробці подібних за завданням та технологіями сервісів, потреба яких полягає у забезпеченні високого рівня захисту мережі від зовнішніх зловмисників та інсайдерських загроз.

Завдяки використанню запропонованих підходів до побудови архітектури сервісу та інтерфейсу очікується що відбудеться покращення роботи базового функціоналу роботи сервісу що впливає на рівень задоволення як користувачів сервісу так і представників бізнесу. Також, запропоновані підходи можуть значно знизити собівартість інфраструктури, що прямо впливає на прибутковість бізнесу сервісу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ І ЛІТЕРАТУРИ

1. С.В.Бойченко Положення про дипломні роботи (проекти) випускників національного авіаційного університету / Бойченко С.В., Іванченко О.В. – Київ: «НАУ-друк», 2017. – 63 с.
2. ДСТУ 3008-95 Структура і правила оформлення. Документація. Звіти у сфері науки і техніки.
3. Е. Баранова, А. Бабаш "Інформаційна безпека та захист інформації 3". / Е. Баранова, А. Бабаш: «Видавництво «ИЦ Риор, НИЦ ИНФРА-М», 2016. – 322 с.
4. Ю. А. Родичев Інформаційні технології та принципи моделювання бізнес-процесів. Навч. посіб. / Ю. А. Родичев: «Видавництво «Пітер», 2017. – 256 с.
5. Іванов І.В Теорія інформаційних процесів і систем / Іванов І.В: «Видавництво Юрайт», 2015 – 229с.
6. *UEBA* – Поведінковий аналіз користувачів і сутностей – 2019. – Режим доступу до ресурсу *https:// encyclopedia.kaspersky.ru /glossary/ueba/* (дата звернення до ресурсу: 26.11.2020).
7. Оліфер В. Г. "Комп'ютерні мережі ". 4 изд./ В. Г. Оліфер, Н.А.Оліфер. – Санкт Петербург: Видавництво "Пітер", 2003. - 215 с.
8. Системи поведінкового аналізу - *User and Entity Behavior Analytics (UBA / UEBA*) – 2019. – Режим доступу до ресурсу: *https://www.anti-malware.ru/ analytics/Market_Analysis/user-and-entity-behavioral-analytics-ubaueba* (дата звернення до ресурсу: 8.11.2020).
9. Виявлення та попередження інсайдерських погроз за допомогою аналізу поведінки користувачів і організацій (*UEBA*) – 2018. – Режим доступу до ресурсу *https://www.fortinet.com/ru/products/ueba* (дата звернення до ресурсу: 16.11.2020).
10. Аналіз поведінки користувачів в мережі та організаціях – 2020. – *https://digitalguardian.com/blog/what-user-and-entity-behavior-analytics-definition-ueba-benefits-how-it-works-and-more* (дата звернення до ресурсу: 16.11.2020).
11. Стандарти телекомунікаційної інфраструктури – 2012. – Режим доступу до ресурсу: *https://www.ixbt.com/comm/sks-1.shtml* (дата звернення до ресурсу: 16.12.2020).

12. Документація - *Python* – 2015. – Режим доступу до ресурсу: *https://www.python.org/doc/ /* (дата звернення до ресурсу: 18.11.2020).

13. The Exabeam Security Management Platform – 2020. – Режим доступу до ресурсу: https://www.exabeam.com/product/ (дата звернення до ресурсу: 23.11.2020).

14. Securonix datasheets UBA – 2020. – Режим доступу до ресурсу: https://www.securonix.com/resource-type/datasheets/ (дата звернення до ресурсу: 23.11.2020).

15. Talking Cloud Security with Amazon Web Services – 2020. – Режим доступу до ресурсу: https://www.forcepoint.com/blog/insights/talking-cloud-security-with-aws (дата звернення до ресурсу: 23.11.2020).

16. A New Vision for Secure Web Gateways – 2020. – Режим доступу до ресурсу: https://www.netskope.com/blog/a-new-vision-for-secure-web-gateways (дата звернення до ресурсу: 23.11.2020).

17. Архітектура REST – Habrahabr – 2020. – Режим доступу до ресурсу: https://habr.com/ru/post/483202/ (дата звернення до ресурсу: 21.12.2020).

18. Best User and Entity Behavior Analytics (UEBA) Tools – 2020. – Режим доступу до ресурсу: https://www.esecurityplanet.com/products/best-user-and-entity-behavior-analytics-ueba-tools/ (дата звернення до ресурсу: 22.11.2020).

19. Крістіан Венц. Програмування в *ASP.NET AJAX* / К. Венц, Д. Крейн, Э. Паскарелло, Кісільов Д. . – М.: Символ Плюс, 2016. – 640 с.

20. Deep Learning in Security – An Empirical Example in User and Entity Behavior Analytics – 2019. – Режим доступу до ресурсу: https://databricks.com/session/deep-learning-in-security-an-empirical-example-in-user-and-entity-behavior-analytics-ueba (дата звернення до ресурсу: 24.11.2020).

21. User and Entity Behavior Analytics (UEBA) defined and explained – 2019. – Режим доступу до ресурсу: https://www.rapid7.com/fundamentals/user-behavior-analytics/ (дата звернення до ресурсу: 24.11.2020).

22. The MongoDB 4.4 Manual – 2020. – Режим доступу до ресурсу: *https://docs.mongodb.com/manual/reference/operator/aggregation/first/* (дата звернення до ресурсу: 02.11.2020).

ДОДАТОК А.
КОД ПРОГРАМИ ДЛЯ АНАЛІЗУ ТРАФІКУ

Файл з *opensource* бібіотеками що використовуються.

```
import collections
import gzip
import os

import numpy
from six.moves import urllib
from six.moves import xrange  # pylint: disable=redefined-builtin

from tensorflow.python.framework import dtypes
from tensorflow.python.framework import random_seed
from tensorflow.python.platform import gfile
from tensorflow.python.util.deprecation import deprecated
```

Opensource метод Collection для отримання значення read32 з об’єктів uint32.

```
_Datasets = collections.namedtuple("_Datasets", ["train", "validation", "test"])

# CVDF mirror of http://yann.lecun.com/exdb/mnist/
DEFAULT_SOURCE_URL = "https://storage.googleapis.com/cvdf-datasets/mnist/"


def _read32(bytestream):
  dt = numpy.dtype(numpy.uint32).newbyteorder(">")
  return numpy.frombuffer(bytestream.read(4), dtype=dt)[0]
```

Opensource функція gzip.gzipFile для роботи с мінімізацією об’єктів та виводом інформації в файл логів.

```
            print("Extracting", f.name)
  with gzip.GzipFile(fileobj=f) as bytestream:
    magic = _read32(bytestream)
    if magic != 2051:
      raise ValueError(
          "Invalid magic number %d in MNIST image file: %s" % (magic, f.name)
      )
    num_images = _read32(bytestream)
    rows = _read32(bytestream)
    cols = _read32(bytestream)
    buf = bytestream.read(rows * cols * num_images)
    data = numpy.frombuffer(buf, dtype=numpy.uint8)
    data = data.reshape(num_images, rows, cols, 1)
    return data
@deprecated(None, "Please use tf.one_hot on tensors.")
def _dense_to_one_hot(labels_dense, num_classes):
```

```
    """Convert class labels from scalars to one-hot vectors."""
    num_labels = labels_dense.shape[0]
    index_offset = numpy.arange(num_labels) * num_classes
    labels_one_hot = numpy.zeros((num_labels, num_classes))
    labels_one_hot.flat[index_offset + labels_dense.ravel()] = 1
    return labels_one_hot
```

Функція def dense to one для початкової обробки вхідних даних.

```
@deprecated(None, "Please use tf.one_hot on tensors.")
def _dense_to_one_hot(labels_dense, num_classes):
    """Convert class labels from scalars to one-hot vectors."""
    num_labels = labels_dense.shape[0]
    index_offset = numpy.arange(num_labels) * num_classes
    labels_one_hot = numpy.zeros((num_labels, num_classes))
    labels_one_hot.flat[index_offset + labels_dense.ravel()] = 1
    return labels_one_hot
```

Функція init для модулю роботи даними які поміщені в чергу.

```
    @deprecated(
        None,
        "Please use alternatives such as official/mnist/_DataSet.py"
        " from tensorflow/models.",
    )
    def __init__(
        self,
        images,
        labels,
        fake_data=False,
        one_hot=False,
        dtype=dtypes.float32,
        reshape=True,
        seed=None,
    ):
```

Opensource методи seed1 і seed2 які пріоріетизує потоки обробки файлів в пакетному режимі.

```
seed1, seed2 = random_seed.get_seed(seed)
        # If op level seed is not set, use whatever graph level seed is returned
        numpy.random.seed(seed1 if seed is None else seed2)
        dtype = dtypes.as_dtype(dtype).base_dtype
        if dtype not in (dtypes.uint8, dtypes.float32):
            raise TypeError("Invalid image dtype %r, expected uint8 or float32" % dtype)
        if fake_data:
            self._num_examples = 10000
            self.one_hot = one_hot
        else:
            assert (
```

```
            images.shape[0] == labels.shape[0]
        ), f"images.shape: {images.shape} labels.shape: {labels.shape}"
        self._num_examples = images.shape[0]
```

Opensource метод який отримує та читає отримані дані з функції gzip.gzipFile

```
def read_data_sets(
    train_dir,
    fake_data=False,
    one_hot=False,
    dtype=dtypes.float32,
    reshape=True,
    validation_size=5000,
    seed=None,
    source_url=DEFAULT_SOURCE_URL,
):
    if fake_data:

        def fake():
            return _DataSet(
                [], [], fake_data=True, one_hot=one_hot, dtype=dtype, seed=seed
            )

        train = fake()
        validation = fake()
        test = fake()
        return _Datasets(train=train, validation=validation, test=test)

    if not source_url:  # empty string check
        source_url = DEFAULT_SOURCE_URL

    train_images_file = 'train-images-idx3-ubyte.gz'
    train_labels_file = 'train-labels-idx1-ubyte.gz'
    test_images_file = 't10k-images-idx3-ubyte.gz'
    test_labels_file = 't10k-labels-idx1-ubyte.gz'
```