КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (26), 2024
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2024.26.666
УДК 004.056

Тишик Іван Ярославович
к.т.н., доцент кафедри захисту інформації
Національний університет «Львівська політехніка», Львів, Україна
ORCID ID: 0000-0003-1465-5342
ivan.y.tyshyk@lpnu.ua

РЕАЛІЗАЦІЯ ЗАХИСТУ БАЗИ ДАНИХ НА ОСНОВІ «ORACLE AUDIT VALUT AND DATABASE FIREWALL»

Анотація. В роботі надані рекомендації щодо захисту баз даних від несанкціонованих вторгнень, організовано збір статистики щодо актуальності та потенційної кількості атак на бази даних, які на цей час відбуваються в світі, проведено загальний огляд атак на бази даних та засоби протидії їм. Для моніторингу активності бази даних на предмет несанкціонованих дій використано рішення у вигляді Oracle Audit Vault and Database Firewall, яке поєднує вбудовані дані аудиту з мережевим захопленням трафіку SQL. Подане рішення запобігає несанкціонованим підключенням до бази даних. Є очевидним те, що деякі бази даних можуть бути зламані кіберзлочинцями просто за щасливим випадком або використавши необачність співробітника. Багато компаній можуть знехтувати заміною облікових даних для входу до інформаційної системи зі значень за замовчуванням. Слабкі паролі співробітників, які легко вгадати, або паролі, які не часто змінюються, також можуть бути розкриті кіберзлочинцями. Часто хакери з метою одержання несанкціонованого доступу до інформаційної системи використовують автоматизоване програмне забезпечення для підбору паролю, яке перебирає слова з відповідного словника. Багато користувачів вибирають для паролю прості слова, які потенційно можуть міститись в словнику, що робить вразливою інформаційну систему. Також доступ до бази даних можна легко досягти, використовуючи зловмисне програмне забезпечення, призначене для використання незахищених чи неоновлених систем. Це також може стосуватися додаткових функцій баз даних, які не використовуються або мають власні вразливості. Атакуючі можуть використовувати вразливості програмного забезпечення декількома способами, наприклад, таким як написання коду для цільової вразливості, яка може бути прихована всередині шкідливого програмного забезпечення. Оновлення здатні покращити продуктивність програмного забезпечення, усунути наявні помилки та видалити застарілі функції, що займають простір на жорсткому диску, а також покращити загальний стан безпеки системи. Оновлення доступні як для операційної системи, так і для систем керування базами даних, і є достатньо важливими для забезпечення їх безпеки та стабільності роботи.

Ключові слова: система управління базами даних; модель авторизації; управління доступом на основі ролей; SQL ін’єкція; мова запитів програми; моніторинг активності бази даних; брандмауер баз даних.

ВСТУП

Дані є основним компонентом організації, яка займається опрацюванням інформації. Нині ця залежність настільки вагома, що успіх практично будь-якої організації залежить від якості та захищеності даних. Тому, природно, що підприємства та організації зацікавленні у збереженні життєво важливих для них даних як в процесі їх опрацювання, так і під час передачі. Основна частина даних зберігається в репозитарії, який називається базою даних. Дані, що зберігаються в базі, структуровані і зазвичай зберігаються у формі реляційних таблиць, і хоча більшість організацій

використовує реляційні бази даних, нереляційні бази також отримали широку популярність, за рахунок відсутності потреби структурувати дані та записувати їх в нетабличному вигляді. Коли використовується реляційна модель даних, дані, що зберігаються в різних реляційних таблицях, пов'язані один з одним.

Відомо, що система управління базами даних (СУБД) є набором програм, які керують наявними в базі даними. Це допомагає організувати дані для кращої продуктивності та пришвидшити пошук завдяки підтримці індексів. Також ведуть журнали запитів, які допомагають відновлювати дані та здійснювати їх моніторинг. СУБД контролює паралельність опрацювання запитів, а також виконує операції щодо відновлення даних.

Оскільки дані, які зберігаються в базах даних, можуть бути критичними для організації, то важливо організувати належний їх захист від атак. Базу даних можна атакувати багатьма способами. Існує можливість атаки виключно на дані бази з метою їх несанкціонованого копіювання чи пошкодження, проте, оскільки бази даних взаємодіють з багатьма додатками чи пов'язані з ними, то можна атакувати й сам додаток, і, як наслідок, отримати доступ до бази [1].

Ситуація стає критичною, коли користувачі бази даних просочують інформацію в зовнішній світ. Комп'ютерна безпека грунтується на трьох важливих аспектах, а саме: конфіденційності, цілісності та доступності. Конфіденційність забезпечує доступ до активів, пов'язаних із комп'ютером, лише авторизованим користувачам. Цілісність означає, що комп'ютерні ресурси можуть бути змінені лише автентифікованими користувачами авторизованими способами. Доступність гарантує, що активи будуть доступні авторизованим користувачам у відповідний час. База даних є комп'ютерним активом, тому перед застосуванням будь-якої політики безпеки до систем баз даних слід враховувати ці аспекти [1], [2].

Існують певні вимоги безпеки до бази даних, як-от фізична, логічна та безпека у вигляді контролю цілісності елементів, а також можливість перевірки прав доступу користувачів до даних певної категорії. Фізична цілісність бази даних стосується фізичних проблем, які пов'язані з охороною обладнання опрацювання даних, обмеженням доступу до нього неуповноважених осіб, забезпечення безперебійної роботи серверів і т.д. До забезпечення логічної цілісності бази даних відносяться підтримка та збереження структури зв'язків у базі даних (в реляційних БД), та цілісність колекцій в нереляційних базах. Цілісність елементів забезпечує також і точність даних. Моніторинг забезпечує можливість відстеження внесених змін у базі даних разом із користувачами, які їх внесли. Контроль доступу забезпечує користувачам доступ лише до даних, на які вони авторизовані [1], [2].

Постановка проблеми. Враховуючи існуючі численні загрози щодо несанкціонованого втручання до баз даних, постає проблема вибору ефективних методів та засобів захисту як баз даних безпосередньо, так і систем управління ними.

Аналіз останніх досліджень і публікацій. Механізм контролю доступу забезпечує керування правами доступу і є основним засобом захисту об'єктів у базах даних, який підтримується більшістю СУБД. Методами, які використовуються для захисту баз даних впровадженням механізмів контролю доступу, є шифрування та маскування даних [1] –[3].

Механізм контролю доступу перевіряє права користувача на відповідність набору авторизації на певні об'єкти. Зазвичай вони визначаються адміністратором безпеки або співробітником служби безпеки. Авторизації надаються відповідно до політики безпеки організації. Поряд з механізмом контролю доступу, для підтвердження того, що користувач системи є тим, за кого себе видає, потрібен ефективний механізм

автентифікації. Після процедури автентифікації, контроль доступу системи визначає відповідні права користувача до певних об’єктів бази даних.

Існує кілька моделей контролю доступу в системах реляційних баз даних. Ці моделі надають різні підходи до реалізації контролю доступу до баз даних.

Дискреційні моделі контролю доступу. У цьому підході, залежно від ідентичності користувача та правил авторизації, доступ надається до об’єктів даних відповідно до деяких дискреційних політик. Основна перевага тут полягає в тому, що користувачі можуть надавати авторизацію на об’єкти даних іншим користувачам. Завдяки такій гнучкості ця методика широко використовується в багатьох системах [4].

Адміністрування авторизації є функцією надання та анулювання дозволів. Таким чином, адміністрування авторизації вводить нові механізми авторизації або видаляє старі. Існує два основних типи таких адмініструвань: централізоване адміністрування, при якому деякі привілейовані суб’єкти можуть надавати або відкликати авторизації та адміністрування власності, при якому сам власник об’єкта надає або відкликає доступ до об’єкта. Власник може надати іншим користувачам право надавати або анулювати деякий або весь доступ до об’єкта [3] – [4].

Модель авторизації System R. Об’єктами даних, які можна захистити за цією моделлю, є таблиці та їх перегляд. Режими доступу, пов’язані з цією моделлю, можуть бути вибором, вставкою, оновленням та видаленням. Адміністрування авторизації в цій моделі засноване на авторизації власності разом із делегуванням авторизації. Творець таблиці стає її власником, і він може авторизувати інших користувачів за допомогою параметрів надання доступу. Для цієї моделі існують деякі розширення. Модель авторизації System R заснована на політиці закритого світу, тому, коли користувач намагається отримати доступ до таблиці, а в системному каталозі немає авторизації, доступ користувачу не надається. Але недоліком цього підходу є те, що відсутність авторизації в будь-який момент часу не може гарантувати жодної авторизації в майбутньому. Щоб прибрати цей недолік, було введено поняття негативної авторизації. Таким чином, негативна авторизація означає відмову в доступі до конкретного об’єкта даних у заданому режимі. У моделі System R, при скасуванні доступу користувачеві до певної таблиці, рекурсивно відкликається доступ до цієї таблиці всім користувачам, яким він надав цей доступ. Це є накладними витратами на механізм доступу, оскільки користувач може дуже часто змінювати роль або функції, таке рекурсивне відкликання відбувається дуже часто. Для нівелювання цього недоліку, використовується інший підхід, який називається некаскадним відкликанням. У цьому підході, після відкликання прав доступу у конкретного користувача, надані ним права доступу не відкликаються. Таким чином, завдяки використанню каскадних, а також некаскадних варіантів авторизації, системи контролю доступу стають більш гнучкими і можуть підтримувати велику кількість додатків [4].

Існує ще третій механізм, який називається керування доступом на основі контексту, де час розглядається як контекст. Зазвичай доступ надається користувачеві з моменту його присвоєння до моменту його скасування. Часовий доступ також може мати періодичний характер.

Чутливий контроль доступу. Цей механізм підтримує контроль доступу на рівні кортежу. Тому й називається чутливим. Він забезпечує детальний рівень контролю доступу. Для реалізації такої схеми потрібна спеціалізація поглядів. Віртуальна приватна база даних Oracle та модель Трумена є прикладами реалізації такого механізму доступу [5].

RBAC (управління доступом на основі ролей) означає контроль доступу на основі ролей і відповідно заснований на концепції ролей. У цій моделі всі дозволи, необхідні

для виконання певної діяльності або роботи, надаються відповідно до ролей, пов'язаних з цією діяльністю чи роботою. Управління авторизацією в цій моделі спрощене. Коли користувач змінює свої функціональні обов'язки в організації, потрібно відкликати лише дозвіл виконувати йому роль, пов'язану з попередньою функцією [6].

Деякі RBAC містять ієрархію ролей, яка дозволяє визначати відносини типу роль-підроль. Вони також дозволяють розділяти обмеження обов'язків, щоб користувач не міг отримати забагато прав доступу. Існує два типи обмежень поділу обов'язків. Статичні обмеження накладають обмеження на перетини ролей. Динамічні обмеження засновані на історії використання ролей користувачами.

Модель обов'язкового контролю доступу надає право контролю доступу лише власникам компанії та авторизованим менеджерам компанії. Це означає, що кінцевий користувач не має контролю над будь-якими налаштуваннями в рамках інформаційної системи.

Відомо, що SQL ін'єкція (SQLi) є однією з найнебезпечніших атак на бази даних. Підходи виявлення для SQLi можна широко розділити на попередньо згенеровані та постгенеровані. Постгенеровані підходи, як правило, корисні під час аналізу динамічного SQL, який генерується веб-додатком. Попередньо згенеровані підходи зазвичай використовуються на етапі тестування веб-додатка. Оцінка з урахуванням синтаксису є постзгенерованим підходом, у якому вхідні рядки спочатку надаються системі для виявлення SQLi, далі вона класифікує вхідні рядки та знаходить шкідливі рядки на льоту. Оцінка з урахуванням синтаксису виконується для поширюваних рядків, щоб визначити які з них шкідливі. Оцінка синтаксису виконується в точці взаємодії з базою даних. Поданий метод має певні недоліки. Ініціалізація довірених рядків залежить від розробника, і зберігання таких записів може призвести до іншої атаки [7].

Контекстно-залежна оцінка рядка: тут будь-які дані, надані користувачем, вважаються тими, які не заслуговують довіри, а дані програми вважаються надійними. Недовірені метадані використовуються для аналізу синтаксису. Цей синтаксичний аналіз розрізняє рядкові та числові константи. Потім усі небезпечні символи видаляються з буквено-цифрових ідентифікаторів. Цей підхід має теж недоліки: ініціалізація небезпечних символів залежить від розробника, а видалення небезпечних символів обмежує функціональність програми.

Оцінка дерева синтаксичного аналізу на основі синтаксису: у цьому підході SQL-запити, згенеровані на основі введення користувача, аналізуються за допомогою попередньо визначених синтаксичних правил. Для позначення початку та кінця рядка використовуються спеціальні літерали. Недоліком тут є те, що зловмисник може маніпулювати введеним рядком за допомогою спеціальних символів.

Інструмент статичного аналізу Ріху використовується для виявлення вразливостей у веб-додатку аналізуючи потік даних для формування статистичної інформації для кожної точки програми. Розробляються дерева аналізу та використовується інструмент аналізу потенційно вразливих точок щоб визначити, де шкідливі дані можуть потрапити до програми. Недоліком цього інструменту є його відкритий вихідний код, тому зловмисник може атакувати, досліджуючи функції та знаходячи вразливості в самому інструменті.

Мова запитів програми використовується веб-розробниками для пошуку запитів, пов'язаних із атакою. Це мова з попередньо визначеним синтаксисом. Переклад виконується з PQL (Program Query Language) в журнали даних. Ці журнали допомагають програмам надавати підтримку для виявлення шкідливих запитів. Знову ж

таки виявлення шкідливих запитів залежить від введених даних розробником, які також використовуються в журналах даних.

Основною технікою, яка використовується для захисту будь-яких даних, є шифрування. Ця техніка широка застосовується для захисту баз даних.

Дані бази шифруються за допомогою ключів та алгоритмів шифрування. Зашифровані дані зберігаються у базі і розшифровуються, коли їх потрібно використати для обробки. Під час виконання шифрування бази даних необхідно прийняти рішення про виконання шифрування всередині бази даних або поза нею. З використанням поданої техніки необхідно організувати належний захист ключів системи від зловмисника, вирішити питання надання повноважень на маніпулювання даними за допомогою цих ключів та забезпечити обмежений доступ до них.

Важливо забезпечити належні механізми аутентифікації, оскільки без них легко отримати доступ до ключів за допомогою методів соціальної інженерії.

Рекомендований підхід до зберігання ключів полягає в тому, щоб розділити ключі та дані, які знаходяться в базі даних. Як правило, ключі зберігаються в апаратному забезпеченні, як-от файли з обмеженим доступом або апаратні модулі зберігання. Процес шифрування може виконуватися як всередині бази даних, так і поза базою даних. Якщо шифрування виконується в базі даних, це має менший вплив на середовище програми. Але існують компроміси з продуктивністю та безпекою, які необхідно враховувати під час впровадження цієї політики. Розуміння алгоритму шифрування, що підтримується СУБД, також відіграє ключову роль при розробці стратегії реалізації цієї техніки [8].

Інший спосіб реалізувати шифрування в базі даних полягає у виконанні його на окремих серверах шифрування. Шифрування та дешифрування обчислень виконуватимуться сервером. Тому тут накладні витрати на шифрування видаляються з СУБД і переходять на окремі сервери шифрування для підтримки продуктивності СУБД. Ключі шифрування та дані також можна розділити. Такого підходу зазвичай дотримуються найчастіше під час шифрування бази даних. Алгоритмами, які зазвичай використовуються для шифрування бази даних і часто підтримуються СУБД, є DES, Triple DES, RC2, RC4, DESX та AES [8].

Схема шифрування бази даних може бути реалізована з використанням різних підходів. Розглядаючи питання шифрування бази даних, слід враховувати дві основні речі. Перш за все, це деталізація даних, які підлягають шифруванню або розшифровці. Деталізація може бути на рівні поля, рядка або сторінки. Деталізація на рівні рядка або сторінки може призвести до шифрування великої кількості даних, які можуть бути накладними для системи. Тому зазвичай виконується шифрування на рівні стовпців лише конфіденційних даних. По-друге, вибір алгоритму шифрування базується на найбільш підходящому для певної бази даних.

Існує багато схем для шифрування на рівні СУБД. Одна з схем заснована на теоремі китайського залишку, в якій кожен рядок шифрується за допомогою різних підключів для різних клітинок. Таким чином, за цією схемою можливе шифрування на рівні рядка та дешифрування на рівні клітинки або поля. Існують деякі схеми, засновані на інтерполяційних поліномах Ньютона, які використовуються для шифрування бази даних. Також існує схема SPDE, яка шифрує кожну комірку в базі даних з її координатами, як-от ім’я таблиці, ім’я стовпця та ідентифікатор рядка тощо. Таким чином, у цій схемі унеможливлюються статичні атаки витоку та атаки з’єднання [9].

Актуальним на сьогодення є захист баз даних від атак на основі аналізу даних (Inference Attack). Для уникнення подібних атак у базах даних використовують метод

поліінстанціювання. У цьому методі допускається кілька екземплярів одних і тих самих даних, але класифікація таких даних має відрізнятися. Метод впливає на зв'язки, кортежі та елементи даних. Зв'язки з поліекземпляром є зв'язками з різними класами доступу. Існує два типи поліекземплярів, а саме видимі та невидимі поліекземпляри [10].

Аудит є ще одним прийомом, який можна використовувати для боротьби з атакою такого типу. При цьому підході зберігається історія всіх запитів, створених користувачами. Аналіз таких запитів може допомогти у пошуку атаки, і її можна уникнути на пізнішому етапі. Але цей підхід виявляє дуже обмежені екземпляри подібних атак і його важко реалізувати на практиці.

Існує багато можливостей для вдосконалення методів, що використовуються для забезпечення безпеки баз даних. Однією з таких областей удосконалення є підхід SPIDER (Self Protecting DatabasE Research), запропонований дослідницьким центром IBM, який фокусується на безпеці автономної бази даних. Архітектура цієї системи містить монітор запитів, систему виявлення вторгнень, журнал попереджень і журнал запитів [11].

Для спостереження та звітування за функціонуванням бази даних широко використовуються моніторинг активності бази даних (МАБД). Інструменти МАБД використовують технологію забезпечення безпеки в реальному часі для незалежного моніторингу та аналізу налаштованих правил, не покладаючись на аудит або журнали СУБД. Ці інструменти також допомагають виявляти аномальну чи несанкціоновану внутрішню або зовнішню діяльність, одночасно оцінюючи ефективність інструментів та політик безпеки. Таким чином, системні адміністратори можуть запобігти несанкціонованому доступу та покращити захист конфіденційних даних від зловмисників.

Інструменти МАБД реалізуються як окремі конфігурації або як програмні модулі, завантажені на сервери баз даних. У будь-якому випадку, вони забезпечують моніторинг бази даних у режимі реального часу збираючи та зберігаючи журнали аналізу, повідомляючи про порушення політики безпеки, при цьому не впливають на продуктивність системи.

Моніторинг активності бази даних здійснюється шляхом поєднання кількох методів, таких як перевірка мережі, очищення пам'яті та читання системних таблиць і журналів аудиту бази даних. Незалежно від використовуваних методів, інструменти МАБД забезпечують кореляцію отримуваної інформації для забезпечення точного уявлення про всі дії в базі даних [12], [13].

Ці інструменти також дозволяють виявляти, ідентифікувати та вживати коригуючих заходів проти загроз і атак, а також надавати криміналістичні докази у разі порушення цілісності даних. Залежно від конфігурації інструментів МАБД адміністратор або аудитор може відновити втрачені дані до початкового стану.

Мета статті. Для забезпечення належного захисту баз даних від несанкціонованого втручання, у статті ставиться завдання надати рекомендації щодо вибору відповідних засобів для якісного проведення аудиту баз даних на предмет оцінювання їх вразливостей до несанкціонованого доступу та організації належного захисту баз даних з подальшим оцінюванням ефективності вибраних засобів.

ISSN 2663 - 4023

РЕАЛІЗАЦІЯ ЗАХИСТУ БАЗИ ДАНИХ НА ОСНОВІ “ORACLE AUDIT VALUT AND DATABASE FIREWALL”

Фаєрволи баз даних включають набір визначених політик аудиту безпеки та здатні ідентифікувати атаки на базу даних на основі евристичного аналізу і/або сигнатур. Таким чином, вхідні оператори/запити SQL порівнюються з цими сигнатурами для виявлення відомих атак на базу даних [12], [13].

Oracle Audit Vault and Database Firewall (AVDF) надає ефективне рішення для моніторингу активності бази даних (МАБД), яке поєднує вбудовані дані аудиту з мережевим захопленням трафіку SQL. Поданий фаєрвол Oracle Database Firewall виступає як перша лінія захисту для баз даних, запобігаючи несанкціонованому доступу до них (рис. 1). Технологія на основі синтаксису SQL відстежує та блокує несанкціонований трафік SQL у мережі до того, як він досягне бази даних.

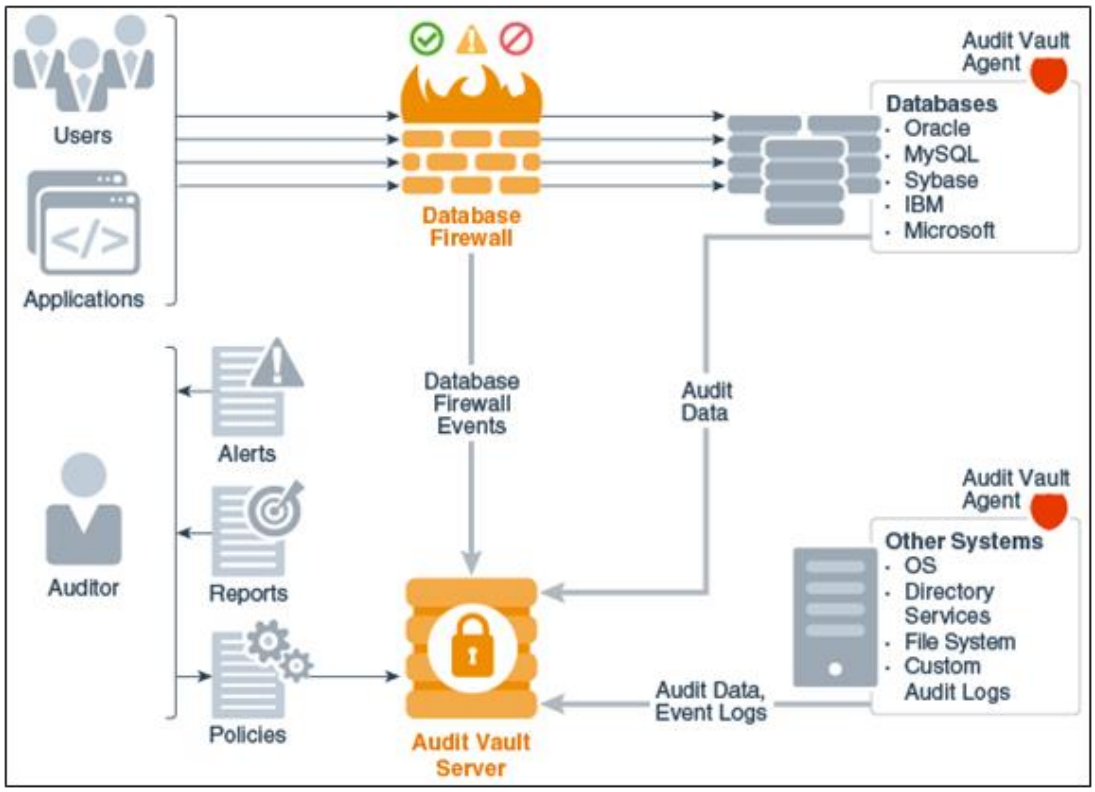

Рис. 1. Схема розташування Oracle AVDF у мережі

Для збору даних аудиту з бази даних встановлено Audit Vault Agent на сервері бази даних. Бінарний файл агента доступний з веб-консолі управління Oracle AVDF.

Після реєстрації хостів на сервері аудиторського сховища (Audit Vault) необхідно виконати дії для запису даних:

- Завантажити програмне забезпечення Audit Vault Agent з консолі Audit Vault Server.
- Розгорнути агента аудиту сховища.
- Активувати Audit Vault Agent.
- Зареєструвати одну або кілька цілей, з яких необхідно збирати дані аудиту.
- Запустити журнали аудиту за допомогою консолі Audit Vault Server.

ISSN 2663 - 4023

Завантаження агента аудиту на сервер бази даних на основі попередньо встановленої програми java 1.8, подано на рис. 2.

```
chown oracle:oinstall agent.jar
export JAVA_HOME=$ORACLE_HOME/jdk
export PATH=$JAVA_HOME/bin:$PATH
[oracle@crs01 dbhome_1]$ java -jar /home/Stage/agent.jar -d
/u01/app/oracle/avdf_agent
Agent installed successfully.
If deploying hostmonitor please refer to product documentation for additional
installation steps.
[oracle@crs01 dbhome_1]$
oracle@crs01 dbhome_1]$ java -jar /home/Stage/agent.jar -d
/u01/app/oracle/avdf_agent
Agent installed successfully.
```

Рис. 2. Встановлення програми агента аудиту на сервер БД

Налаштування аудиту бази даних включає такі кроки, як створення нового аудиту табличного простору та виконання процедури для встановлення відповідних параметрів (рис. 3).

```
SQL> BEGIN
DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_LOCATION(
audit_trail_type => dbms_audit_mgmt.audit_trail_unified,
audit_trail_location_value => 'AVDF_AUD_DATA');
END;
/  2    3    4    5    6
PL/SQL procedure successfully completed.

SQL> BEGIN
DBMS_AUDIT_MGMT.INIT_CLEANUP(
AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL,
DEFAULT_CLEANUP_INTERVAL => 1 );
END;
/  2    3    4    5    6
PL/SQL procedure successfully completed.

SQL> SQL> SQL> BEGIN
DBMS_AUDIT_MGMT.CREATE_PURGE_JOB (
AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL,
AUDIT_TRAIL_PURGE_INTERVAL => 1,
AUDIT_TRAIL_PURGE_NAME => 'CLEANUP_OS_DB_AUDIT_RECORDS',
USE_LAST_ARCH_TIMESTAMP => TRUE );
END;
/  2    3    4    5    6    7    8
PL/SQL procedure successfully completed.
```

Рис. 3. Створення табличного простору для запису параметрів аудиту

Для збору інформації аудиту від вибраних цілей, необхідно розгорнути агент Audit Vault на окремому хост-комп'ютері, який зазвичай є тим самим комп'ютером, де знаходиться ціль. Агент аудиту сховища містить плагіни для кожного цільового типу.
Для збору аудиторських слідів слід виконати наступні процедури: зареєструвати цільовий хост та розгорнути агента аудиту сховища.

Після завершення конфігурації на стороні сервера бази даних, необхідно налаштувати журнал аудиту на веб-консолі хоста. Для цього необхідно перейти на консоль та вибрати збір даних аудиту. Тоді додати параметри бази даних, такі як тип сигнатури, хост агента (рис. 4).

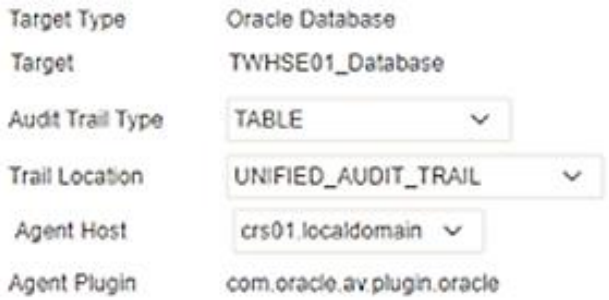

Рис. 4. Додавання налаштувань бази з використанням веб-консолі хоста

Процес збору даних для агента Oracle Audit Vault відбувається згідно з діаграмою, наведеною на рис.5.

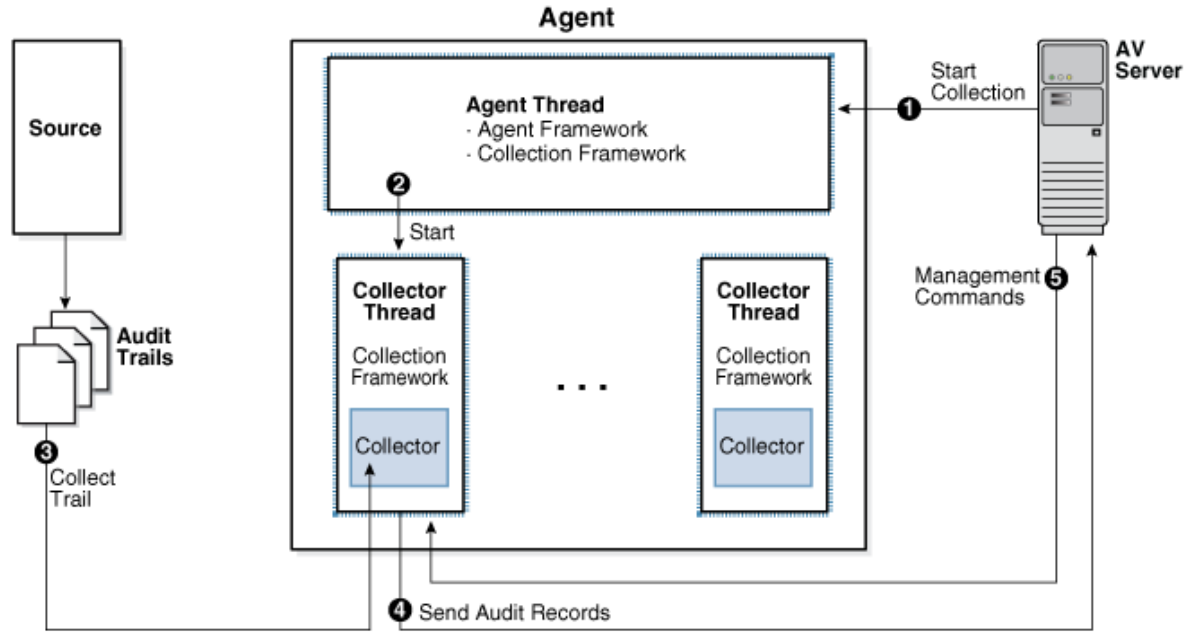

Рис. 5. Потік збору даних для агента Oracle Audit Vault

Плагін, який виконує потік даних під час збору журналів аудиту, отримує доступ до журналу аудиту, витягує запис аудиту та його пов'язані поля з журналу. Далі він відображає запис аудиту на подію Oracle Audit Vault, після чого плагін збору передає подію та поля Oracle Audit Vault агенту Audit Vault, який відправляє цю інформацію на сервер Oracle Audit Vault.

Процес збору, згідно рис.5, виглядає наступним чином:

1. Сервер Oracle Audit Vault дає команду фреймворку агента створити потік для збору певного журналу аудиту.
2. Новий потік, створений агентом, збирає конкретний журнал аудиту і передає контроль над потоком фреймворку збору.
3. Фреймворк збору у потоці підключається до сервера Oracle Audit Vault і запитує конфігураційну інформацію для журналу аудиту, яка збирається. Крім того, він запитує інформацію про останню точку відновлення (checkpoint), встановлену для цього журналу. Точка відновлення зберігається на сервері Oracle Audit Vault у відповідній таблиці.
4. Фреймворк збору, маючи необхідну інформацію, звертається до файлу плагіна, щоб знайти потрібний Java-клас у відповідному плагіні збору. Він передає цій сутності конфігураційну інформацію та ініціює запит на її ініціалізацію.
5. Після ініціалізації колектора фреймворк збору повторює наступні дії в циклі:
   – Запитує у колектора додаткові записи аудиту з журналу.
   – Колектор перетворює (відображає) ці нові записи аудиту у формат, вказаний у файлі відображення (mapper file), і передає їх фреймворку збору через API збору.
6. Агент надсилає інформацію про точку відновлення та інші метрики, отримані від плагіна збору, на сервер Audit Vault. Сервер зберігає цю інформацію в таблиці точок відновлення.
7. Якщо сервер Audit Vault надсилає фреймворку збору команди, наприклад команду завершення роботи, фреймворк передає ці команди колектору для виконання. Якщо фреймворк збору отримує команду STOP від сервера Oracle Audit Vault, він повідомляє колектор про припинення надсилання записів, виходить із потоку збору та вимикається.

Агент запускає потоки збору. У межах кожного потоку Audit Vault Collection Framework виконує код, наданий плагіном збору.

Фаза збору є етапом, на якому Audit Vault збирає записи журналу аудиту. Під час фази збору плагін отримує доступ до журналу аудиту для вилучення нових записів. Точний механізм доступу до журналів залежить від конкретного журналу аудиту. Після отримання запису з журналу, плагін перетворює (відображає) його у формат, який можна надіслати на сервер Audit Vault.

Плагін збору також має отримати інформацію про кодування символів цільових записів, типи кодування та питання, пов’язані з часовими мітками, щоб забезпечити відповідність вимогам сервера Audit Vault.

AVDF є інструментом, за допомогою якого можна отримати проактивні вимірювання безпеки для баз даних. Для цього необхідно налаштувати аудит (рис.6), який буде відслідковувати будь-які дії по відношенню до записів в базі даних:

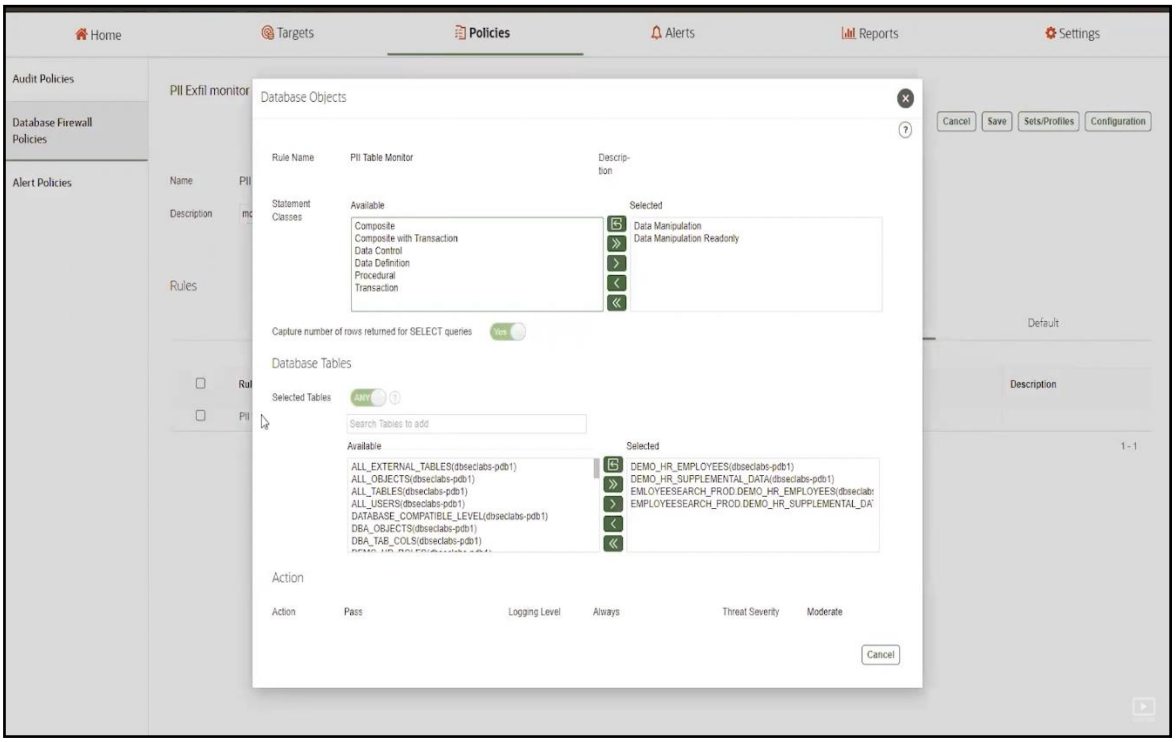

Рис. 6. Налаштування нової політики аудиту

Після ввімкнення процесу захоплення дій над записами, можна отримати результат у вигляді наступної інформаційної панелі, поданої на рис. 7.

У розглянутій моделі, Database Firewall можна налаштувати для моніторингу та блокування або тільки для моніторингу запитів до бази даних. Для моніторингу та блокування необхідно налаштувати брандмауер у режимі проксі, щоб весь трафік баз даних проходив через Database Firewall (рис.8). Для мережного моніторингу SQL-трафіку можна налаштувати порт SPAN мережних комутаторів так, щоб вони відправляли трафік на Database Firewall, або налаштувати монітор хоста на комп'ютерах з базами даних так, щоб він перенаправляв SQL-трафік на Database Firewall.

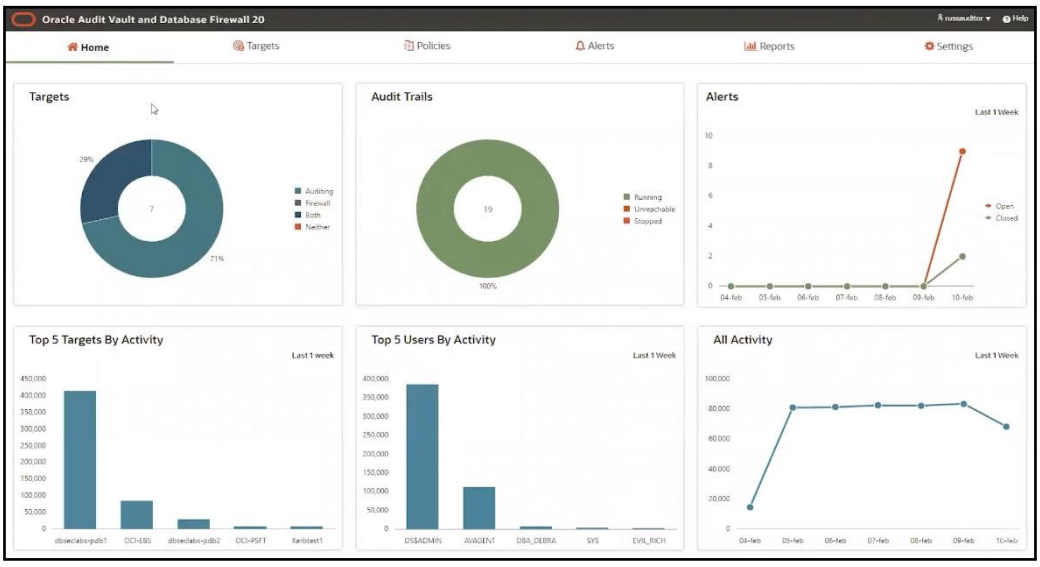

Рис. 7. Відображення отриманих даних аудиту у вигляді графіків

Цей інтерфейс надає можливість вибрати стовпчасті графіки відображення даних, які збираються для подальшого аналізу.

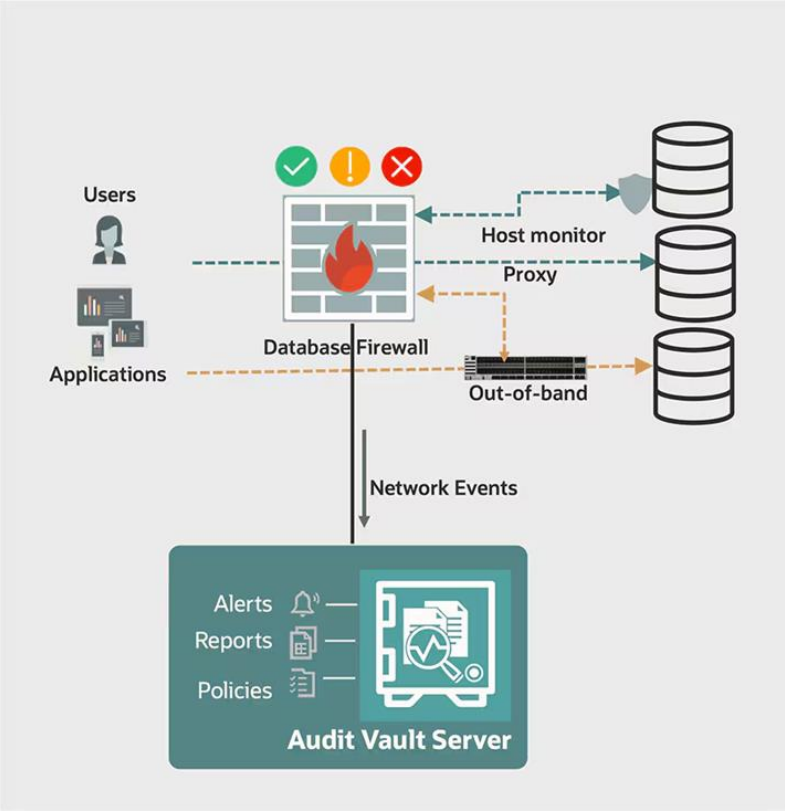

Рис. 8. Налаштування Database Firewall для моніторингу та блокування запитів до бази даних

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Встановлено, що база даних є одним з найкритичніших аспектів в світі інформаційних технологій, компрометація якого призводить до інформаційних, матеріальних та репутаційних втрат організації. Небезпечними для інформаційної системи організації в цілому є атаки на бази даних, які можуть призвести до так званої Supply Chain Attack, реалізація якої, наприклад, через несанкціонований доступ до якихось збережених в базі даних записів клієнта, дозволить отримати доступ до аккаунту цього ж клієнта на інших сервісах.

На основі проведених моделювань встановлено, що програмний продукт Oracle Audit Vault and Database Firewall здатний ефективно захищати бази даних різних систем за допомогою моніторингу мережевого трафіку та аналізу аудиторських даних. Цей продукт є важливим інструментом для забезпечення превентивного та виявляючого контролю з метою захисту від несанкціонованого доступу до баз даних. Oracle Audit Vault and Database Firewall охоплює майже всі способи компрометації даних і типи кібератак, розширює можливості захисту інформації за межами баз даних Oracle та інших постачальників завдяки підтримці аудиту операційних систем, каталогів та користувацьких джерел даних аудиту.

Перспективами подальших досліджень можуть стати впровадження інструментів керування доступом Zero Trust, які дозволяють відносно легко налаштовувати рівні доступу для користувачів, пристроїв або робочих середовищ. Організації можуть встановлювати суворі стандарти щодо підключення сторонніх користувачів до інформаційної системи, наприклад, вимога надання мінімального доступу та застосувати метод мікросегментації мережі, який вимагає пройти повторну автентифікацію для отримання доступ до різних зон у мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пасічник, В. В., & Резніченко В. А. (2006). *Організація баз даних та знань.* К.: Видавнича група BHV.
2. Basta, A., & Z. Melissa. (2011). *Database Security.* Cengage Learning.
3. Gertz, M., Jajodia, S. (2008). Handbook of Database Security: Applications and Trends. Springer. https://doi.org/10.1007/978-0-387-48533-1
4. Chen, P. P. (2002). Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned. *Software Pioneers*, 296–310. https://doi.org/10.1007/978-3-642-59412-0_17
5. Matvieiev, D., & Fedorenko, D. (2019). The Problem of Protection of Personal Data on the Internet. *ΛΟΓΟΣ.ONLINE: International scientific e-journal, 4.*
6. *Managing Claims and Authorization with the Identity Model.* (n. d.). https://learn.microsoft.com/en-us/dotnet/framework/wcf/feature-details/managing-claims-and-authorization-with-the-identity-model
7. *Тестування безпеки: SQL-ін'єкції.* (2020). https://training.qatestlab.com/blog/technical-articles/security-testing-sql-injection
8. Попадюк, В. В. (2020). Шифрування в базах даних SQL SERVER. *Кібербезпека в сучасному світі: матеріали ІІ Всеукраїнської науково-практичної конференції.*
9. *Basic Security Practices for SQLite: Safeguarding Your Data.* (n. d.). https://dev.to/stephenc222/basic-security-practices-for-sqlite-safeguarding-your-data-23lh
10. Omotunde, H., & Maryam A. (2023). A Comprehensive Review of Security Measures in Database Systems: Assessing Authentication, Access Control, and Beyond. *Mesopotamian Journal of CyberSecurity, 2023,* 115–133. https://doi.org/10.58496/MJCSC/2023/016
11. *Server Performance and Activity Monitoring.* (n. d.). https://learn.microsoft.com/en-us/sql/relational-databases/performance/server-performance-and-activity-monitoring?view=sql-server-ver16
12. *Oracle® Audit Vault and Database Firewall.* (n. d.). https://docs.oracle.com/en/database/oracle/audit-vault-database-firewall/20/sigdv/what-is-oracle-audit-vault-and-database-firewall
13. Гулак, Г. М., Жильцов, О. Б., Киричок, Р. В., Коршун, Н. В., & Складанний, П. М. (2024). *Інформаційна та кібернетична безпека підприємства.* Підручник. Львів : Видавець Марченко Т. В.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (26), 2024
ISSN 2663 - 4023

Ivan Tyshyk
PhD, Associate Professor at the Department of Information Security
National University "Lviv Polytechnic", Lviv, Ukraine
ORCID ID: 0000-0003-1465-5342
ivan.y.tyshyk@lpnu.ua

IMPLEMENTATION OF DATABASE SECURITY BASED ON "ORACLE AUDIT VAULT AND DATABASE FIREWALL"

Abstract. The paper provides recommendations for protecting databases from unauthorized intrusions, organizes the collection of statistics on the relevance and potential number of database attacks currently occurring worldwide, and offers a general overview of database attacks and countermeasures. To monitor database activity for unauthorized actions, the solution used is Oracle Audit Vault and Database Firewall, which combines built-in audit data with network SQL traffic capture. This solution prevents unauthorized database connections. It is evident that some databases may be compromised by cybercriminals through sheer luck or by exploiting employee negligence. Many companies may neglect to change default login credentials for information systems. Weak employee passwords that are easy to guess, or passwords that are not changed regularly, can also be exposed by cybercriminals. Hackers often use automated software to gain unauthorized access to an information system by brute-forcing passwords, cycling through words from a dictionary. Many users choose simple words for passwords that could potentially be included in a dictionary, making the information system vulnerable. Additionally, database access can easily be achieved using malicious software designed to exploit unprotected or outdated systems. This can also apply to unused database features, which may have their own vulnerabilities. Attackers can exploit software vulnerabilities in various ways, such as writing code for a target vulnerability hidden inside malware. Updates can improve software performance, fix existing bugs, remove outdated features that take up disk space, and enhance the overall security of the system. Updates are available for both operating systems and database management systems, and are crucial for ensuring their security and stability.

Keywords: Database Management System; Authorization Model; Role-Based Access Control; SQL Injection; Program Query Language; Database Activity Monitoring; Database Firewall.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Pasichnyk, V. V., & Reznychenko V. A. (2006). *Organization of Databases and Knowledge*. Kyiv: BHV Publishing Group.
2. Basta, A., & Z. Melissa. (2011). *Database Security.* Cengage Learning.
3. Gertz, M., Jajodia, S. (2008). Handbook of Database Security: Applications and Trends. Springer. https://doi.org/10.1007/978-0-387-48533-1
4. Chen, P. P. (2002). Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned. *Software Pioneers*, 296–310. https://doi.org/10.1007/978-3-642-59412-0_17
5. Matvieiev, D., & Fedorenko, D. (2019). The Problem of Protection of Personal Data on the Internet. *ΛΟΓΟΣ.ONLINE: International scientific e-journal, 4.*
6. *Managing Claims and Authorization with the Identity Model.* (n. d.). https://learn.microsoft.com/en-us/dotnet/framework/wcf/feature-details/managing-claims-and-authorization-with-the-identity-model
7. *Security Testing: SQL Injections.* (2020). https://training.qatestlab.com/blog/technical-articles/security-testing-sql-injection
8. Popadyuk, V. V. (2020). Encryption in SQL SERVER Databases. *Cybersecurity in the Modern World: Materials of the II All-Ukrainian Scientific and Practical Conference*.
9. *Basic Security Practices for SQLite: Safeguarding Your Data.* (n. d.). https://dev.to/stephenc222/basic-security-practices-for-sqlite-safeguarding-your-data-23lh
10. Omotunde, H., & Maryam A. (2023). A Comprehensive Review of Security Measures in Database Systems: Assessing Authentication, Access Control, and Beyond. *Mesopotamian Journal of CyberSecurity, 2023,* 115–133. https://doi.org/10.58496/MJCSC/2023/016

11. *Server Performance and Activity Monitoring.* (n. d.). https://learn.microsoft.com/en-us/sql/relational-databases/performance/server-performance-and-activity-monitoring?view=sql-server-ver16
12. *Oracle® Audit Vault and Database Firewall.* (n. d.). https://docs.oracle.com/en/database/oracle/audit-vault-database-firewall/20/sigdv/what-is-oracle-audit-vault-and-database-firewall
13. Hulak, H. M., Zhiltsov, O. B., Kyrychok, R. V., Korshun, N. V., & Skladannyi, P. M. (2024). *Information and cyber security of the enterprise.* Textbook. Lviv: Publisher Marchenko T. V.