Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: Засоби забезпечення безпеки серверних застосунків

Виконавець: студент IV курсу, групи КБ-41

_____ **Олександр МАЛИХІН**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Інна МИХАЛЬЧУК	

Нормоконтроль	Андрій БІГДАН	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-41** (група) **Малихіну Олександру Олександровичу** (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Засоби забезпечення безпеки серверних застосунків

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Концепція захисту серверних застосунків та види їх архітектур.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з побудовою серверних застосунків, їх видами архітектур, вразливостями з боку безпеки даних, обрати методи захисту вразливих місць та розробити рекомендації щодо захисту інформації при роботі з серверними застосунками.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблені рекомендації з вибору методів захисту даних, що використовуються в серверних застосунках.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видала _____ (підпис) Інна МИХАЛЬЧУК (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Олександр МАЛИХІН (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 03.11.2022	*виконано*
2	Аналіз літератури	04.11.2022 – 30.11.2022	*виконано*
3	Обґрунтування вибору рішення	01.12.2022 – 22.12.2022	*виконано*
4	Концепція серверного застосунку	23.12.2022 – 10.01.2023	*виконано*
5	Аналіз проблем інформаційної безпеки в серверних застосунках	11.01.2023 – 21.02.2023	*виконано*
6	Дослідження вразливостей та загроз	22.02.2023 – 30.03.2023	*виконано*
7	Вироблення рекомендацій щодо побудови коплексної системи захисту даних в серверних застосунках	31.03.2023 – 15.05.2023	*виконано*
8	Оформлення пояснювальної записки	16.05.2023 – 24.05.2023	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	25.05.2023 – 12.06.2023	*виконано*

Завдання видала _____ (підпис) Інна МИХАЛЬЧУК (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Олександр МАЛИХІН (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, має 64 сторінки основного тексту, 2 таблиці. Список використаних джерел містить 19 найменувань і займає 2 сторінки.

Методи дослідження кваліфікаційної роботи:

- аналіз літератури;
- аналіз документів;
- порівняння;

Об'єктом дослідження в кваліфікаційній роботі є процес захисту даних в серверних застосунках.

Предметом дослідження в даній роботі є методи, засоби і методики захисту серверних застосунків.

У роботі проаналізована існуюча література з теорії серверних застосунків, виконаний аналіз документів, порівняння, вивчення та узагальнення зарубіжної практики з теми серверних застосунків, розроблено рекомендації з створення системи захисту даних у серверних застосунках.

Розроблені рекомендації призначені для адміністраторів, що хочуть забезпечити безпеку персональних даних своїх користувачів в серверних застосунках.

Ключові слова: захист персональних даних, веб-додатки, безпека серверних застосунків, міжсайтовий скриптинг, ін’єкційна атака, дворівнева архітектура, трирівнева архітектура, контроль доступу, система аналізу.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

XSS	-	Cross-Site Scripting
SQLi	-	Structured Query Language Injection
IP	-	Internet Protocol
RBAC	-	Role-Based access-control
PBAC	-	Permission-Based access-control
ABAC	-	Attribute-Based access-control
WAF	-	Web Application Firewall
DPI	-	Deep Packet Inspection
IPS	-	Intrusion Prevention System
OSI	-	Open Systems Interconnection
СУБД	-	Система управління базами даних
ОС	-	Операційна система

ЗМІСТ

ВСТУП 8

РОЗДІЛ 1 ОСНОВНІ ВІДОМОСТІ ПРО СУЧАСНІ СЕРВЕРНІ ЗАСТОСУНКИ 9

1.1 Поняття сучасного серверного застосунку 9

1.2 Архітектура побудови серверних застосунків 11

1.2.1 Клієнт-серверна архітектура 11

1.2.2 Основні компоненти серверних застосунків 12

1.2.3 Дворівнева архітектура 14

1.2.4 Трирівнева архітектура 17

1.2.5 Порівняння дворівневої та трирівневої архітектури 20

Висновки за розділом 1 22

2.1 Вектори атак та вразливі місця серверних застосунків 23

2.2 Порушення контролю доступу 27

2.3 Захист від вразливості «Порушення контролю доступу» 29

2.3.1 Керування доступом на основі ролей 29

2.3.2 Керування доступом на основі атрибутів 31

2.3.3 Дискреційне керування доступом 33

2.3.4 Механізми для забезпечення керування доступом 35

2.4 Ін'єкційні атаки 37

2.4.1 Класична SQL ін'єкція 38

2.4.2 Сліпа SQL ін'єкція 38

2.4.3 SQL ін'єкція, заснована на експлуатації СУБД повідомлень про помилки (Error-based SQLi) 39

2.5 Захист від ін'єкційних атак 41

2.6 Міжсайтовий скриптинг 43

2.7 Захист від міжсайтового скриптингу 45

Висновки за розділом 2 47

РОЗДІЛ 3 ВИБІР ОПТИМАЛЬНИХ МЕТОДІВ ЗАХИСТУ СЕРВЕРНИХ ЗАСТОСУНКІВ ТА РОЗРОБКА РЕКОМЕНДАЦІЙ 49

3.1 Використання системи аналізу DPI 49

3.2 Використання систем WAF та IPS, як базової системи захисту 52

3.3 Розробка рекомендацій для захисту серверних застосунків 55

Висновки за розділом 3 60

ВИСНОВКИ 61

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63

ВСТУП

Актуальність роботи засобів забезпечення безпеки серверних застосунків в сучасному світі не може бути переоцінена. З огляду на постійний розвиток технологій та збільшення кількості загроз у кіберпросторі, захист серверних застосунків стає критично важливим завданням для організацій та компаній.

Серверні застосунки зберігають та обробляють великі обсяги конфіденційної інформації, такої як особисті дані користувачів, фінансові відомості. Недостатня безпека таких застосунків може призвести до витоку цієї інформації, порушення довіри користувачів та великих фінансових втрат.

Засоби забезпечення безпеки серверних застосунків є невід'ємною частиною процесу розробки застосунків. Вони допомагають зменшити ризики, пов'язані з втратою даних, і забезпечують захист користувачів у системи. Отже, актуальність роботи засобів забезпечення безпеки серверних застосунків надзвичайно важлива.

Тому **метою роботи** є розробка рекомендацій щодо вибору методу захисту даних при побудові серверного застосунку.

Для досягнення поставленої мети необхідно вирішити такі завдання**:**

- розглянути особливості побудови серверних застосунків;
- дослідити проблематику захисту інформації в серверних застосунках;
- розглянути існуючі методи захисту інформації при побудові серверного застосунку.

Об'єктом дослідження в даній роботі є процес захисту даних в серверних застосунках.

Предметом дослідження в даній роботі є методи, засоби і методики захисту серверного застосунку, а також дослідження видів відомих вразливостей.

Методи дослідження кваліфікаційної роботи:

- аналіз літератури;
- аналіз документів;
- порівняння;

РОЗДІЛ 1
ОСНОВНІ ВІДОМОСТІ ПРО СУЧАСНІ СЕРВЕРНІ ЗАСТОСУНКИ

1.1 Поняття сучасного серверного застосунку

Програмне забезпечення, яке працює на серверному комп'ютері і надає послуги клієнтам, називається сучасним серверним додатком. Серверні програми зберігають, обробляють і доставляють дані клієнтам. Серверні програми можуть використовуватися для запуску веб-сайтів, надання послуг електронної пошти, зберігання файлів тощо.

Більшість сучасних серверних застосунків створюються за допомогою об'єктно-орієнтованих мов програмування. Для того, щоб впоратися з великою кількістю користувачів і запитів, вони розробляються так, щоб бути масштабованими і надійними.

Сучасні серверні застосунків бувають різних категорій, але на мою думку слід зазначити найпопулярніші, до прикладу:

- Веб-сервери: клієнти отримують веб-сторінки з веб-серверів. Зазвичай такий тип серверів, використовується для роботи веб-сторінок і веб-додатків.
- Сервери додатків: бізнес-додатки працюють на серверах додатків. Вони пропонують ряд послуг, включаючи обробку транзакцій, доступ до баз даних і безпеку.
- Файлові сервери: файли зберігаються та використовуються на файлових серверах. На таких типах серверів, часто зберігаються документи, зображення та інші речі.
- Поштові сервери: електронна пошта надсилається та отримується за допомогою поштових серверів. Вони часто слугують провайдерами електронної пошти як для корпорацій, так і для приватних осіб.

Також слід зазначити основні характеристики враховуючи які створюють сучасні серверні застосунки, наприклад:

• Масштабованість: масштабовані серверні застосунки - це ті, які можуть обробляти велику кількість користувачів і запитів, не сильно втрачаючи свою продуктивність при цьому.

• Надійність: сучасні серверні застосунки створені для того, щоб бути надійними, що дозволяє їм продовжувати працювати, незважаючи на помилки.

• Безпека: серверні застосунки повинні враховувати питання безпеки, щоб захистити критично важливу інформацію та зупинити несанкціонований доступ. Це передбачає впровадження систем автентифікації та авторизації, шифрування даних, перевірки введення та захисту від поширених ризиків безпеки, таких як міжсайтовий скриптинг (XSS) та атаки на SQL-ін'єкції.

• Зручність використання: застосунки створюються для того, щоб бути зручними для користувачів, дозволяючи їм швидко отримувати доступ до необхідних ресурсів.

• Оптимізація продуктивності: покращення часу обробки та відгуку серверних застосунків вимагає оптимізації продуктивності. Продуктивність можна підвищити, застосовуючи стратегії, що включають кешування, оптимізацію запитів, асинхронну обробку та використання ефективних алгоритмів і структур даних.

• Моніторинг та ведення журналів: для відстеження стану системи, виявлення проблем та оцінки продуктивності серверні застосунки повинні мати надійні методи моніторингу та реєстрації. Інструменти моніторингу можуть виявити слабкі місця, використання ресурсів і потенційні збої, тоді як ведення журналів з помилками надає корисні дані для аудиту та налагодження їх виправлення.

Тому я вважаю, що серверні застосунки грають дуже велику роль у сучасному світі інформаційних технологій, так як вони виступають основним будівельним матеріалом для сучасних додатків.

1.2 Архітектура побудови серверних застосунків

1.2.1 Клієнт-серверна архітектура

Архітектура клієнт-сервер – це поширена модель, яка використовується в комп'ютерних мережах і розподілених обчислювальних системах, де завдання і обов'язки розподіляються між двома основними компонентами: клієнтом і сервером. Ця архітектурна модель широко використовується в найрізноманітніших додатках і сервісах в мережі Інтернет [1].

В архітектурі клієнт-сервер клієнтом, як правило, є користувацький додаток або пристрій, який запитує послуги або ресурси у сервера. Клієнтом може бути веб-браузер, мобільний додаток або будь-яке програмне забезпечення, яке взаємодіє з користувачем. Його основна функція - надсилати запити до сервера і відображати ці результати користувачеві.

З іншого боку, сервер – це потужний комп'ютер або система, яка надає послуги або ресурси багатьом клієнтам. Він відповідає за отримання запитів клієнтів, обробку та повернення запитуваних даних або виконання необхідних дій. Сервери призначені для одночасної обробки декількох клієнтських з'єднань, забезпечуючи масштабованість і швидкість реагування.

Зв'язок між клієнтом і сервером базується на моделі "запит-відповідь". Клієнт надсилає запит на сервер із зазначенням необхідної операції або даних [2]. Сервер отримує запит, обробляє його і формує відповідь із запитуваною інформацією або результатом операції. Потім відповідь надсилається назад клієнту, який може інтерпретувати та відображати дані або виконувати інші дії з відповіддю [3].

Структура такої системи полягає в тому, що клієнт надсилає запит до сервера, де він обробляється, а готовий результат повертається клієнту. Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно надійшло декілька запитів, вони ставляться в чергу й виконуються послідовно сервером. Запити можуть мати пріоритети, у такому випадку запити з вищими пріоритетами будуть виконуватися першими.

На сервері реалізуються такі функції:

- зберігання, доступ, захист та резервне копіювання даних;
- обробка запитів від клієнтів;
- надсилання результатів (відповідей) клієнтам.

На стороні клієнта реалізуються такі функції:

- надання користувальницького інтерфейсу;
- формулювання запиту до сервера й його надсилання;
- отримання результатів запиту й надсилання додаткових команд (запитів на додавання, оновлення або видалення даних).

Існують концепції побудови системи клієнт-сервер:

У концепції "слабкий клієнт – потужний сервер" вся обробка інформації здійснюється на сервері, а клієнт має обмежені права доступу. Сервер надсилає відповідь, яка не вимагає додаткової обробки [4]. Клієнт взаємодіє з користувачем, складає та надсилає запит, отримує результат та відображає інформацію на екрані.

У концепції "сильний клієнт - потужний сервер" частина обробки інформації здійснюється клієнтом. Сервер виступає сховищем даних, а вся робота з обробки та відображення інформації перекладається на комп'ютер клієнта.

Архітектура клієнт-сервер використовується в різних додатках, таких як веб-браузери, електронна пошта, файлообмінні сервіси та онлайн-ігри. Вона забезпечує балансування навантаження, ефективне управління ресурсами, просте обслуговування та оновлення, що робить її ключовою концепцією в сучасних мережах та обчислювальних системах.

1.2.2 Основні компоненти серверних застосунків

Веб-додатки зазвичай складаються з декількох ключових компонентів, які працюють один з одним , щоб забезпечити гарний функціональний досвід та презентабельний вигляд застосунку, щоб забезпечити гарний користувацький досвід.Далі я зазначив основні види компонентів на стороні клієнта та на стороні сервера:

Компоненти на стороні клієнта:

• Інтерфейс користувача (UI): це візуальна та інтерактивна частина веб-додатку, з якою взаємодіють користувачі. Складається з великої кількості елементів, серед яких: кнопки, форми, контекстні меню та інші графічні елементи, які у свою чергу дозволяють користувачам вводити дані і переміщатися по додатку.

• Вигляд презентації: цей пункт відповідає за представлення та рендеринг компонентів інтерфейсу. Для редагування вигляду компонентів інтерфейсу використовують такі технології, як мова розмітки гіпертексту (HTML), каскадні таблиці стилів (CSS) і JavaScript для структурування і стилізації веб-сторінок, управління взаємодією з користувачем і забезпечення динамічної поведінки.

Компоненти на стороні сервера:

• Логічна побудова додатку: логіка додатку відповідає за обробку запитів клієнтів, виконання необхідних обчислень та управління даними. Зазвичай це включає управління бізнес-правилами, виконання перевірок та взаємодію із зовнішніми базами даних або сервісами.

• Саме веб-сервер: він у свою чергу відповідає за отримання клієнтських запитів, перенаправлення їх вже наявним відповідним компонентам і повернення відповіді клієнту. API визначають набір правил і протоколів для зв'язку, що дозволяє різним додаткам взаємодіяти і обмінюватися даними.

• Формати даних: дані, якими обмінюються клієнт і сервер, можуть бути в різних форматах, таких як JSON (вид текстового формату, який використовується переважно для передачі структурованої інформації через мережу), XML (eXtensible Markup Language). Ці формати структурують дані і дозволяють легко аналізувати та інтерпретувати їх як на стороні клієнта, так і на стороні сервера.При цьому перевагою формату JSON перед XML є те, що він дозволяє мати складні структури у атрибутах, також він займає меньше місця.

• Аутентифікація та авторизація: основні складові безпеки веб-додатків, вони реалізують механізми автентифікації та авторизації користувачів для забезпечення безпечного доступу до ресурсів та конфіденційності інформації. У свою чергу може

включати у себе такі види автентифікації: автентифікація за допомогою імені користувача або пароля, керування сеансами та контроль доступу на основі ролей.

• Шифрування даних: конфіденційні дані, що передаються між клієнтами та серверами, можна зашифрувати за допомогою безлічі типів протоколів захисту інформації, наприклад: SSL/TLS (TLS і SSL використовують асиметричне шифрування для автентифікації, симетричне шифрування для конфіденційності та коди автентифікації повідомлень для збереження цілісності повідомлень.). Протоколи використовуються для того щоб запобігти несанкціонованому доступу та забезпечити цілісність інформації, яка передається між клієнтами.

•Перевірка даних: авжеж на стороні сервера повинна бути налаштована належна перевірка даних, які вводяться користувачем, це має величезне значення для запобігання вразливостям безпеки, більшість популярних вразливостей безпеки буде розглянуто мною далі, але для прикладу можна навести такі: міжсайтовий скриптинг (XSS) або SQL-ін'єкції.

•Компоненти моніторингу та ведення журналів дій: цей компонент необхідній для відстеження продуктивності застосунку, доступності та надважливий при виникненні помилок. Даний компонент збирає і зберігає дані про використання програми, помилки і споживання ресурсів. Він часто включає у себе: моніторинг у реальному часі, відстеження помилок та інструменти аналізу продуктивності.

Усі ці компоненти працюють разом, щоб забезпечити безперебійну роботу користувача, обробляти запити, керувати даними і надавати динамічний контент для серверних застосунків.

Слід зазначити, що компоненти і технології, які використовуються для побудови застосунку, можуть відрізнятися залежно від конкретних потреб, масштабу і технологічних завдань.

1.2.3 Дворівнева архітектура

Дворівнева архітектура, є однією з найпростіших і найстаріших форм архітектури клієнт-сервер. У цій архітектурі система поділяється на два основні

компоненти: клієнт і сервер. Далі я розібрав найважливіші, на мою думку, компоненти дворівневої архітектури. Вона поділяється на:

Клієнтський рівень:

- Клієнтський рівень відповідає за рендеринг користувацького інтерфейсу і усю взаємодію з користувачем.
- Клієнтська програма обробляє дані, що вводяться користувачем, відображає інформацію та надсилає запити на сервер для обробки.
- Клієнтський рівень часто включає компоненти локального зберігання даних (наприклад, базу даних або локальний кеш), якщо це необхідно для технічного завдання вашого застосунку.

Рівень серверів:

- Серверний рівень завжди відповідає за обробку клієнтських запитів, виконання логіки програми та управління ресурсами наявних даних.
- Серверний компонент отримує клієнтські запити, виконує необхідні операції та повертає результати або відповіді. Зазвичай обробляє бізнес-логіку, перевірку даних та операції з базами даних.Також включає додатки, які працюють на виділеному сервері, якщо такий потрібен.
- У дворівневій архітектурі сервер безпосередньо відповідає за обробку клієнтських запитів і взаємодію з рівнем зберігання даних.

Комунікація у дворівневій архітектурі:

- Зв'язок у дворівневій клієнт-серверній архітектурі зазвичай здійснюється через прямий зв'язок між клієнтом і сервером. Клієнт надсилає запити на сервер, а сервер відповідає безпосередньо клієнту.
- Комунікація може бути полегшена за допомогою різних протоколів і технологій, таких як TCP/IP для мережевих комунікацій і HTTP для інтернет-додатків.

Підключення до бази даних:

- У дворівневій архітектурі серверний компонент безпосередньо взаємодіє з базою даних для отримання або модифікації даних. Сервер відповідає за запити до бази даних і забезпечення цілісності даних.

• Серверний рівень зазвичай включає компоненти або модулі, які керують підключенням до бази даних. Також сервер взаємодіє з базою даних, використовуючи специфічні для бази даних протоколи та мови запитів, такі як мова структурованих запитів (SQL).

Розгортання дворівневої архітектури та її масштабованість:

• Дворівнева архітектура часто вимагає, щоб серверний компонент був розгорнутий на виділеній серверній машині.

• Масштабування системи у дворівневій архітектурі може бути складним, оскільки сервер обробляє як логіку додатків, так і операції з базами даних. Масштабування зазвичай вимагає модернізації серверного обладнання або додавання нових серверів для збалансування навантаження.

• Зі збільшенням кількості клієнтів та складності даних дворівнева архітектура може відчувати обмеження продуктивності через централізовану обробку та потенційні обмеження ресурсів.

Переваги дворівневої архітектури:

• Простота: дворівневу архітектуру відносно легко спроектувати, реалізувати та зрозуміти порівняно з більш складними архітектурами.

• Прямий контроль: у дворівневій архітектурі клієнт має прямий контроль над користувацьким інтерфейсом і може забезпечити більш гнучкий користувацький досвід.

• Зменшення складності: відсутність проміжного програмного забезпечення (наприклад, сервера додатків) авжеж зменшує загальну складність системи та усі накладні витрати.

Недоліки дворівневої архітектури:

• Масштабованість і продуктивність: дворівнева архітектура може дуже сильно втрачати свою ефективність при збільшенні кількості клієнтів, що ускладнює ефективне масштабування системи.

• Негнучкість: тісний зв'язок між клієнтськими та серверними компонентами обмежує гнучкість і ускладнює зміну або розширення функціональності системи.

• Міркування безпеки: дворівнева архітектура може мати вразливі місця в системі безпеки, оскільки клієнт взаємодіє безпосередньо з сервером і може отримати доступ до конфіденційної інформації користувачів або виконати несанкціоновані дії з наявною на сервері інформацією.

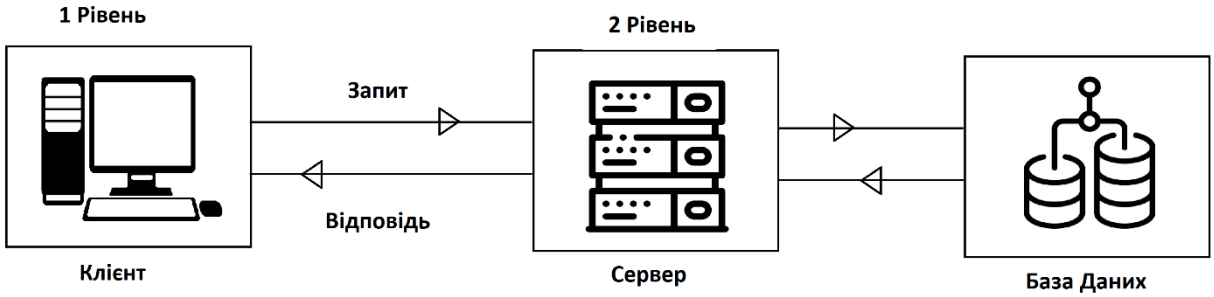

Рисунок 1.1 – Дворівнева архітектура.

Деякі ранні веб-додатки також використовували дворівневу архітектуру, де клієнт взаємодіє безпосередньо з веб-сервером, який обробляє запити і керує логікою роботи програми.

У цій архітектурі клієнтські та серверні компоненти тісно пов'язані між собою, а це означає, що робота і поведінка клієнта часто залежить від доступності та швидкості реагування сервера.

Однак для більших систем зазвичай використовуються більш масштабовані та гнучкі архітектури, такі як 3-рівнева або n-рівнева архітектура, для розподілу навантаження, підвищення продуктивності та спрощення обслуговування системи.

1.2.4 Трирівнева архітектура

Трирівнева архітектура клієнт-сервер, також відома як багаторівнева архітектура, є популярним архітектурним шаблоном для створення масштабованих і модульних серверних застосунків [5]. Вона розділяє систему на три основні рівні або яруси: рівень клієнта, рівень додатку та рівень даних. Далі я наведу розбір кожного рівня окремо:

Рівень клієнта:

• Рівень клієнта відповідає за управління інтерфейсом користувача і взаємодію з користувачем. Основна увага на даному рівні приділяється представленню інформації користувачеві та збору даних, що вводяться користувачем.

• Компоненти на стороні клієнта відповідають за рендеринг інтерфейсу, перевірку даних, введених користувачем, і функції взаємодії з користувачем.

• Технології, такі як HTML, CSS, JavaScript, зазвичай використовуються на рівні клієнта для створення гарних та інтерактивних користувацьких інтерфейсів.

Рівень додатків:

• Рівень додатків містить основну роль і обробляє клієнтські запити з клієнтського рівня.

• Він виконує такі завдання, як обробка даних, управління потоками додатків. Рівень додатків координує роботу між клієнтським рівнем п і рівнем даних для виконання клієнтських запитів.

• Залежно від складності та обсягу програми, рівень додатків може складатися з різних компонентів, таких як сервери додатків, веб-сервіси або мікросервіси.

Рівень даних або серверний рівень:

• Рівень даних відповідає за управління та зберігання даних. Він підтримує взаємодію з базами даних або іншими системами зберігання даних.

• Рівень даних надає методи та інтерфейси для доступу, маніпулювання та отримання даних, необхідних застосункам. Він забезпечує цілісність, безпеку та ефективну архівацію даних.

• На рівні даних часто використовуються реляційні бази даних (наприклад, MySQL або Oracle) також він може включати такі компоненти, як рівні доступу до даних або системи управління базами даних.

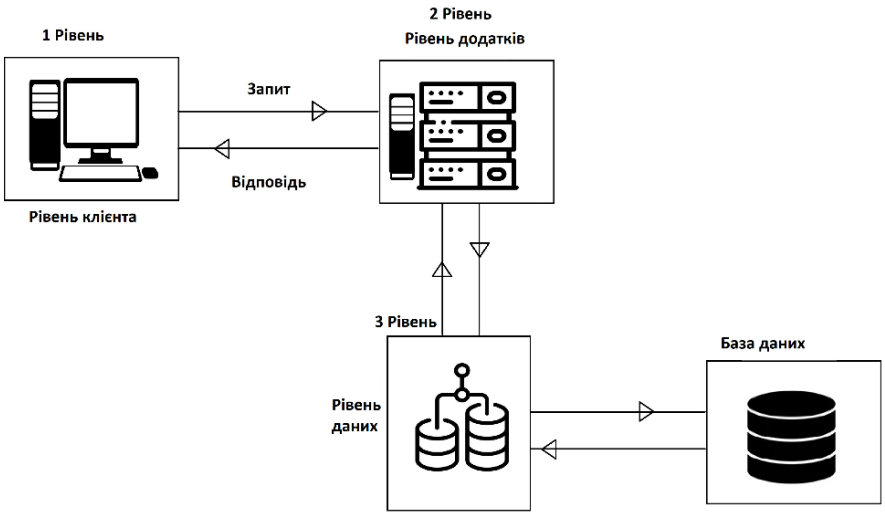

Рисунок 1.2 – Трирівнева архітектура.

Переваги трирівневої архітектури:

• Масштабованість: трирівнева архітектура пропонує кращу масштабованість і гнучкість, ніж дворівнева [6]. Кожен рівень можна масштабувати незалежно, що забезпечує розподілені обчислення та кращу продуктивність.

• Розподіл обов'язків: розділення рівнів дозволяє здійснювати модульну розробку, що полегшує обслуговування та оновлення окремих компонентів без впливу на всю систему.

• Безпека: чітке розділення рівнів дозволяє застосовувати заходи безпеки на всіх рівнях для захисту конфіденційних даних і забезпечення контролю доступу до інформації.

• Багаторазове використання: компоненти на кожному рівні можна повторно використовувати в різних додатках, що підвищує ефективність розробки.

Загалом, трирівнева клієнт-серверна архітектура забезпечує структурований і масштабований підхід до розробки. Вона сприяє розподілу обов'язків, модульній розробці та ефективному використанню ресурсів, що робить її загальноприйнятою архітектурною моделлю для сучасної розробки застосунків.

1.2.5 Порівняння дворівневої та трирівневої архітектури

На мою думку, було б доречно, провести порівняльний аналіз двох найпопулярніших архітектур побудови серверних застосунків. Тому далі ви можете ознайомитись з моєю аналітикою, спираючись на основні порівняльні метрики.

Порівняння структури:

Архітектура двох рівнів складається з клієнтського рівня та серверного рівня. У цій архітектурі сервер відповідає за обробку запитів і передачу відповідей клієнту, використовуючи власні ресурси, тоді як клієнтський рівень відповідає за надання користувальницького інтерфейсу [7]. Робота цієї архітектури полягає у тому, що сервер отримує запит, обробляє його і безпосередньо надсилає відповідь, не залучаючи зовнішніх ресурсів.

Трирівнева архітектура додатково включає рівень додатків між клієнтським і серверним рівнями. Рівень додатків відповідає за обробку клієнтських запитів, виконання бізнес-логіки та координацію пошуку та зберігання даних. У цій архітектурі кілька серверів співпрацюють у обробці запиту клієнта. Розподіл операцій допомагає зменшити навантаження на сервери і покращує ефективність системи.

Порівняння можливості масштабованості:

Дворівнева архітектура має обмеження щодо масштабованості. Зі збільшенням кількості клієнтів навантаження на сервер може стати вразливим місцем, що потенційно може спричинити проблеми з продуктивністю.

Трирівнева архітектура пропонує кращу масштабованість і гнучкість, ніж дворівнева. Розділення рівнів дозволяє розподілену обробку та незалежне масштабування кожного рівня. Рівень додатку можна масштабувати окремо від клієнтського рівня та рівня даних, що забезпечує кращу продуктивність та обробку більших клієнтських навантажень.

Порівняння гнучкості та модульності:

Дворівневій архітектурі не вистачає гнучкості та модульності [8]. Клієнтські та серверні компоненти тісно пов'язані між собою, що ускладнює зміну або розширення системи без впливу на весь застосунок.

Трирівнева архітектура заохочує модульність і розподіл обов'язків. Завдяки чіткому поділу рівнів ви можете розробляти та редагувати кожен компонент системи незалежно. Така модульність підвищує ремонтопридатність коду, а також можливість оновлення окремих компонентів без впливу на всю систему.

Порівняння складності:

Дворівнева архітектура відносно проста і легка в реалізації. Вона має просту структуру з прямою клієнт-серверною взаємодією.

Трирівнева архітектура вносить додаткову складність через додатковий прикладний рівень. Наявність проміжного рівня вимагає більш ретельного планування та координації між рівнями. Однак ця підвищена складність забезпечує кращу ефективність.

Порівняння продуктивності:

У дворівневій архітектурі можуть виникати обмеження продуктивності, оскільки сервер обробляє як логіку додатків, так і операції з даними. Це може призвести до уповільнення часу відгуку, особливо при великій кількості одночасних клієнтів.

Трирівнева архітектура може запропонувати кращу продуктивність. Розподіл завдань обробки між різними рівнями дозволяє оптимізувати обробку клієнтських запитів, покращити час відгуку та ефективно використовувати ресурси.

Порівняння безпеки:

У дворівневій архітектурі заходи безпеки необхідно впроваджувати як на рівні клієнта, так і на рівні сервера. Прямий зв'язок між клієнтом і сервером може підвищити потенційну вразливість безпеки, що вимагає додаткових запобіжних заходів.

Трирівнева архітектура надає більше можливостей для реалізації заходів безпеки. Кожен рівень може мати спеціальні засоби управління безпекою, які дозволяють краще захистити конфіденційні дані та забезпечити контроль доступу до інформації, яка потребує захисту.

Загалом, трирівнева архітектура клієнт-сервер пропонує кращу масштабованість, гнучкість, модульність і продуктивність, ніж дворівнева

архітектура. У той час як дворівнева архітектура може бути доречною для простих застосунків з невеликою кількістю клієнтів і простими вимогами. Звідси - трирівнева архітектура краще підходить для складних додатків, які вимагають масштабованості, модульності та поділу завдань.

Висновки за розділом 1

У даному розділі ми розглянули основні відомості про сучасні серверні застосунки та архітектури їх побудови. Серверні застосунки є невід'ємною частиною веб-розробки та надання послуг, вони забезпечують ефективну обробку запитів від клієнтів, зберігання та керування даними, забезпечують безпеку та інші функціональні можливості.

Ми ознайомилися з різними архітектурами побудови серверних застосунків, зокрема клієнт-серверною архітектурою, дворівневою архітектурою та трирівневою архітектурою.

Клієнт-серверна архітектура дозволяє клієнтам взаємодіяти з сервером, який забезпечує обробку запитів та надсилання відповідей. Вона проста у реалізації, але може бути обмеженою в масштабованості та розширюваності.

Дворівнева архітектура складається з клієнтської частини та серверної бази даних. Клієнтська частина взаємодіє з користувачем, а серверна база даних забезпечує зберігання та обробку даних. Вона проста у реалізації, але може бути обмеженою в масштабуванні та розподіленості.

Трирівнева архітектура розділяє серверний застосунок на презентаційний рівень, бізнес-логіку та рівень доступу до даних. Це дозволяє досягти вищого рівня розширюваності, масштабованості та модульності. Вона підходить для більших проектів зі складною бізнес-логікою та вимогами до масштабованості.

Знання про сучасні серверні застосунки та архітектуру їх побудови є важливим для розробників, що працюють у сфері веб-розробки. Вони допомагають зрозуміти різні підходи до побудови системи та обрати найбільш ліпший для конкретного проекту.

РОЗДІЛ 2
АНАЛІЗ ВРАЗЛИВИХ МІСЦЬ СЕРВЕРНИХ ЗАСТОСУНКІВ ТА ЇХ ЗАХИСТ

2.1 Вектори атак та вразливі місця серверних застосунків

Порушення конфіденційності, цілісності, доступності, функціональності або інших властивостей безпеки часто досягається зловмисниками шляхом використання вразливостей.У багатьох випадках вразливості в безпеці виникають, коли програма або система не в змозі безпечно обробити непередбачені або ненадійні дані. Уразливості можуть бути результатом помилок у кодуванні чи конфігурації, слабких місць у інтерфейсі користувача та небезпечних специфікацій протоколу та формату.

Незважаючи на значні зусилля щодо підвищення безпеки програмного забезпечення, сучасне програмне забезпечення та системи настільки складні, що неможливо побудувати їх без уразливостей [9].

Серед факторів, які підвищують ризик вразливості безпеки, є:

- Надмірна залежність від систем, які, мають вже заздалегідь відомі вразливі місця.
- Недостатні знання про вразливості.
- Не обізнаність про наявність вразливостей.

Класифікація загроз WASC — це всесвітньо визнана система класифікації вразливостей, яка стосується безпеки веб-додатків і керується міжнародною асоціацією.Це некомерційна організація, яка об'єднує експертів з інформаційної безпеки та фахівців з кібербезпеки з усього світу.

WASC класифікує вразливості веб-додатків на різні етапи життєвого циклу програмного забезпечення.

- Тип уразливості, який виникає на етапі проектування програми в результаті помилок, допущених у процесі проектування.

• Уразливості, які виникають через помилки під час реалізації певних компонентів програми, підпадають під категорію вразливостей на етапі впровадження.

• Вразливості, які викликаються неправильним налаштуванням застосунку на етапі розгортання.

Залежно від типу атаки уразливості класифікуються наступним чином:

1. Атаки на основі автентифікації, особливо ті, які використовують облікові дані адміністратора, є поширеним явищем в інцидентах кібербезпеки. Атаки грубої сили, недостатні протоколи автентифікації та ненадійні методи відновлення пароля – усе це приклади атак на рівні адміністратора.

2. Напад на авторизацію неминуче тягне за собою і удар по адміністратору. Цей напад може виникнути через вгадування ідентифікатора сеансу, невідповідну авторизацію або невстановлення часу очікування сеансу.

3. Існують різні форми атак, які можуть бути спрямовані на клієнтів, зокрема на їхні браузери. Ці атаки включають підміну вмісту, міжсайтовий скриптинг і фрагментацію HTTP-запитів.

4. Одним із способів атаки на сервери є виконання коду на самому сервері. Цей тип атаки може приймати різні форми, включаючи, атаки з переповненням буфера, атаки з ін’єкцією SQL, атаки на формат рядка та атаки з ін’єкцією LDAP. Крім того, зловмисники можуть спробувати виконати команди в операційній системі сервера.

5. Список потенційних загроз безпеці програмної системи великий і включає низку типів атак. До них належать прямі напади на розробника, а також більш абстрактні логічні напади на дизайнера. Інші загрози включають індексацію каталогів, ідентифікацію програм, витік інформації, перегляд каталогів і передбачуваний розподіл ресурсів.

Топ-10 OWASP — це загальновизнана компіляція вразливостей. OWASP – це некомерційна організація, яка займається підвищенням безпеки програмного забезпечення. Вони випускають OWASP Top 10, який є довідковим посібником для

веб-розробників, які зосереджені на безпеці. Цей посібник містить детальний огляд найбільш критичних уразливостей і загроз для веб-програм.

Список OWASP Top 10 за 2019 рік має наступні вразливості:

• Проблема порушення контролю доступу виникає, коли обмеження щодо певних авторизованих користувачів, які мають певні дозволи на внесення змін до системи, не накладаються ефективно. Через ці вразливості зловмисники можуть використовувати систему для отримання несанкціонованого доступу до конфіденційних даних і функцій. Серед іншого це може включати доступ до облікових записів інших користувачів, зміну важливих файлів, зміну даних інших користувачів і маніпулювання правами доступу.

• Можливість ін’єкції сторінки SQL виникає, коли в команду чи запит включено недійсний ввід. Ця ін’єкція зловмисних даних може виконувати небажані команди або надавати несанкціонований доступ до даних.

• Феномен міжсайтового скриптингу передбачає використання вразливостей безпеки в програмі. Ці вразливості часто спричинені неправильною перевіркою чи видаленням ненадійних даних, які додаються до нової веб-сторінки, або оновленням наявної веб-сторінки даними, створеними користувачами, через API браузера, здатний генерувати HTML або JavaScript. У результаті ця вразливість дозволяє хакерам виконувати сценарії в цільовому браузері користувача. Часто ці атаки використовуються для викрадення сеансів, видалення вмісту веб-сайту або перенаправлення трафіку на неприємні, а часом і шкідливі веб-сайти.

• Функції програмування, пов’язані з автентифікацією та керуванням сеансами, часто виконуються неправильно, що призводить до можливості отримання зловмисниками доступу до ключів сеансу, паролів, токенів або використання інших уразливостей у реалізації для доступу до облікових записів адміністратора та інших користувачів, тимчасово чи постійно.

• Ненадійність безпеки даних є важливою проблемою, яка мучить багато застосунків. Конфіденційна інформація користувача часто залишається вразливою через неналежні заходи захисту. Це дозволяє кіберзлочинцям легко викрадати або маніпулювати цими даними, що призводить до злочинних дій, таких як крадіжки

особистих даних, шахрайство з кредитними картками та інші подібні злочини. Ці конфіденційні дані часто зберігаються без будь-якого додаткового рівня захисту, як-от шифрування під час передачі чи передавання, і потребують спеціальних протоколів безпеки, коли вони передаються у вашому браузері.

- У ситуаціях, коли інтернет-ресурси застаріли або XML-процесори налаштовані ненадійно, сторонні URL-адреси в XML-документах можуть бути застосовані злочинниками. Це може відкрити двері для зловмисників, які можуть використовувати ці посилання для доступу до внутрішніх файлів за допомогою обробників URL-адрес, внутрішнього обміну файлами, внутрішнього сканування портів, віддаленого виконання коду або навіть атак на відмову в обслуговуванні.

- Однією з найпоширеніших проблем у безпеці є неправильні параметри безпеки. Ця вразливість часто виникає через небезпечні налаштування за умовчанням, неповні конфігурації, відкрите хмарне сховище, неправильно налаштовані заголовки HTTP та повідомлення про помилки, які містять конфіденційні дані. Вкрай важливо, щоб усі операційні системи, фреймворки, бібліотеки та додатки не лише мали безпечні налаштування, але й оновлювалися своєчасно.

- Неможливо недооцінити небезпеку десеріалізації, оскільки це може призвести до виконання віддаленого коду. Хоча помилки десеріалізації зазвичай не призводять до віддаленого виконання коду, їх можна використати для різноманітних інших атак, таких як впровадження, відтворення стороннього вмісту та зміна дозволів користувачів.

- Використовуючи такі компоненти, як бібліотеки, фреймворки та інші програмні модулі, важливо зазначити, що вони працюють із тим самим рівнем дозволів, що й сама програма. Якщо один із цих компонентів містить вразливість, зловмисник потенційно може спричинити хаос, викравши величезну кількість даних або навіть захопивши сервер. Ось чому дуже важливо бути обережним під час використання таких компонентів, оскільки їх присутність може значно збільшити ймовірність атак та інших форм використання, які можуть підірвати безпеку застосунку.

• Одним із найважливіших факторів, що сприяють порушенням кібербезпеки, є відсутність належного обліку та моніторингу даних. Коли обліку та моніторингу недостатньо, зловмисники можуть легко атакувати кілька систем, зберігаючи при цьому відмовостійкість і масштабуючи численні системи. Ця відсутність належної інтеграції в систему реагування на інциденти також дозволяє зловмисникам маніпулювати, використовувати або знищувати дані. Крім того, дослідження показують, що сканування зламів безпеки зазвичай залишається непоміченим протягом понад 200 днів, і їх частіше виявляють сторонні, а не внутрішні процеси чи моніторинг.

2.2 Порушення контролю доступу

Авторизація надає користувачеві права для виконання конкретних дій, а також перевіряє права доступу перед виконанням цих дій. Важливо розуміти, що авторизація не тотожна аутентифікації. Після успішної аутентифікації, авторизація визначає, до яких функцій і даних користувач може отримати доступ, забезпечуючи адекватне розподілення прав доступу. Саме тому веб-програмам потрібні механізми контролю доступу, що дозволяють користувачам використовувати програму з різними рівнями привілеїв.

Контроль доступу полягає в застосуванні обмежень щодо того, хто має право виконувати певні функції та отримувати доступ до запитаних ресурсів. Порушення контролю доступу є часто зустрічаються і критичні вразливості у системі безпеки.Існують кілька типів контролю доступу:

Вертикальний контроль доступу:

Вертикальний контроль доступу – це механізм безпеки, який регулює та обмежує доступ користувачів на основі ієрархічної структури або рівнів дозволів в організації. Він зазвичай використовується в середовищах, які вимагають різних рівнів привілеїв доступу, наприклад, в корпоративних середовищах, державних установах або військових організаціях.

Вертикальний контроль доступу розподіляє користувачів на різні рівні або групи відповідно до їхніх ролей, обов'язків або рівнів дозволів до інформації. Кожен рівень представляє певний рівень авторизації або прав доступу в організації. Чим вищий рівень, тим більші права доступу і тим більше доступу користувач має до конфіденційної інформації.

Декілька ключових аспектів, вкі включає в себе вертикальний контроль доступу:

- Рівні авторизації: Система контролю доступу визначає різні рівні авторизації, часто представлені ієрархічними позиціями або привілеями безпеки. Наприклад, адміністратор може змінювати або видаляти обліковий запис будь-якого користувача, у той час як звичайні користувачі не мають доступу до цих функцій.

- Дозволи на доступ: кожен рівень пов'язаний з набором дозволів на доступ, які визначають дії та ресурси, до яких користувач на цьому рівні може отримати доступ. Ці дозволи можуть включати читання, запис, зміну або видалення даних, а також виконання певних адміністративних завдань.

- Принцип найменших привілеїв: принцип найменших привілеїв є важливою концепцією вертикального контролю доступу. Він полягає в тому, що користувачам слід надавати лише мінімальні дозволи на доступ, необхідні для ефективного виконання їхньої роботи. Це допомагає мінімізувати ризик несанкціонованого доступу або випадкового нецільового використання конфіденційної інформації.

- Списки контролю доступу (ACL): ACL зазвичай використовуються для реалізації вертикального контролю доступу. Це списки, пов'язані з кожним ресурсом або об'єктом і визначають рівні або ролі, яким дозволено або заборонено доступ до цього ресурсу.

Вертикальний контроль доступу допомагає забезпечити розподіл обов'язків, обмежуючи доступ до конфіденційної інформації та гарантуючи, що працівники мають доступ лише до тих даних, які їм потрібні для роботи. Він також сприяє дотриманню нормативних вимог і допомагає захиститися від внутрішніх загроз або спроб несанкціонованого доступу до інформації.

Але впровадження вертикального контролю доступу вимагає ретельного планування, класифікації користувачів, визначення політик доступу та налаштування відповідних засобів контролю доступу в IT-інфраструктурі організації або підприємства.

Горизонтальний контроль доступу :

Механізми горизонтального контролю доступу призначені для обмеження доступу до виділених ресурсів виходячи з прав доступу кожного окремо взятого користувача. Використовуючи горизонтальний контроль доступу, різні користувачі можуть отримувати доступ до безлічі ресурсів одного і того ж типу. Наприклад, банківська веб-програма дозволяє користувачам переглядати транзакції та здійснювати платежі зі свого власного рахунку, але не з рахунків інших користувачів. Порушення горизонтального контролю доступу передбачає отримання доступу до прав іншого облікового запису з аналогічними привілеями.

2.3 Захист від вразливості «Порушення контролю доступу»

2.3.1 Керування доступом на основі ролей

Захиститися від такої атаки можливо багатьма способами, наприклад за допомогою методології RBAC (Role-Based access-control) – керування доступом на основі ролей.

Контроль доступу на основі ролей (Role-Based Access Control, RBAC) – це модель контролю доступу, яка керує правами доступу користувачів на основі попередньо визначених ролей в організації. За допомогою RBAC користувачам надається доступ на основі призначених їм ролей, а не на основі їхніх індивідуальних ідентифікаційних даних. Це широко використовуваний механізм контролю доступу, який пропонує гнучкий і масштабований підхід до управління правами доступу.

Це найбільш поширена та багатьом відома модель, яка добре накладається на предметні бізнес-області та корелює з посадовими функціями. Є певним розвитком

DAC, де привілеї групуються відповідні їм ролі. Кожен суб'єкт може мати переліком ролей, де роль своєю чергою може надавати доступом до якогось переліку об'єктів.

Далі буде досліджено декілька ключових складових, які включають в контроль доступу на основі ролей:

- Ролі визначаються відповідно до функцій, обов'язків або завдань в організації. Кожна роль являє собою набір дозволів, пов'язаних з відповідною робочою функцією. Прикладами ролей можуть бути Адміністратор, Менеджер, Співробітник або Гість".
- Дозволи пов'язані з кожною роллю і визначають дії або операції, які можуть виконувати користувачі, призначені для цієї ролі. Дозволи можуть включати читання, запис, модифікацію, видалення або виконання певних ресурсів чи функцій.
- Призначення ролей користувачів, тобто користувачам призначаються ролі на основі їхніх потреб або обов'язків. Зіставлення користувачів з ролями спрощує управління контролем доступу, дозволяючи надавати і відкликати дозволи на рівні ролей, а не на рівні користувачів.
- Політики контролю доступу визначають взаємозв'язок між ролями, дозволами та ресурсами. Ця політика визначає, які ролі мають доступ до яких ресурсів і які операції вони можуть виконувати над цими ресурсами.
- RBAC може містити ієрархію ролей, де певні ролі успадковують привілеї від ролей вищого рівня. Це спрощує процес адміністрування, оскільки загальні дозволи можуть бути призначені ролям вищого рівня, а ролі нижчого рівня успадковують ці дозволи за замовчуванням.
- Також RBAC підтримує принцип розподілу обов'язків і гарантує, що жоден користувач не має надмірних прав доступу. Призначаючи різні ролі різним користувачам, організація може розподілити обов'язки та впровадити систему стримувань і противаг.
- RBAC може дозволити динамічне призначення ролей на основі контекстуальних факторів, таких як час, місцезнаходження або певні умови для більш гнучкої політики контролю доступу.

RBAC пропонує кілька переваг, зокрема підвищену безпеку, спрощене управління та масштабованість. Вона допомагає організаціям забезпечити

дотримання принципу найменших привілеїв, спростити управління доступом користувачів і підтримувати узгодженість політик контролю доступу в масштабах всього підприємства. RBAC також покращує можливість аудиту та дотримання нормативних вимог, забезпечуючи прозорий зв'язок між ролями користувачів та відповідними дозволами на доступ.

2.3.2 Керування доступом на основі атрибутів

Контроль доступу на основі атрибутів (ABAC) - це модель контролю доступу, яка використовує атрибути для прийняття рішень щодо авторизації. На відміну від традиційних моделей контролю доступу, заснованих на ролях або ідентифікаторах користувачів, ABAC враховує різні атрибути, пов'язані з користувачем, ресурсом, середовищем та іншими контекстними факторами, щоб визначити дозволи на доступ.

Рисунок 2.1 – Керування доступом на основі атрибутів.

У цьому підході необхідне ведення спеціальних політик, які об'єднують атрибути суб'єктів та об'єктів, а рішення про допуск надається на основі аналізу та порівняльної оцінки цих атрибутів.

Це найбільш гнучкий з описаних підходів з величезною кількістю можливих комбінацій, який дозволяє приймати рішення на основі таких компонентів та принципів управління доступом:

• Атрибути – це характеристики або властивості, пов'язані з користувачами, ресурсами або середовищем. Прикладами атрибутів є ролі користувачів, посада, облікові дані безпеки, час доби, місцезнаходження, класифікація даних та будь-яка інша відповідна контекстна інформація.

• Оцінка політики, тобто правила ABAC визначають умови, за яких доступ надається або забороняється на основі атрибутів. Політики можуть бути виражені за допомогою логічних правил, виразів або мов політик. Після отримання запиту на доступ система оцінює відповідні атрибути і застосовує політики для прийняття рішення про надання доступу.

• ABAC дозволяє здійснювати детальний контроль доступу, враховуючи безліч атрибутів та їхніх значень. Це дозволяє вам приймати більш деталізовані та контекстні рішення щодо доступу, наприклад, надавати доступ до певного ресурсу лише в певний час або лише користувачам з певним рівнем доступу.

• ABAC підтримує динамічну авторизацію з урахуванням атрибутів у реальному часі та мінливих умов навколишнього середовища. Це дозволяє приймати рішення щодо контролю доступу на основі поточного статусу атрибутів, забезпечуючи більш високу кастомізацію.

• ABAC покладається на постачальників атрибутів для надання значень атрибутів. Ці постачальники можуть включати каталоги користувачів, системи управління ідентифікацією, авторизацію атрибутів або інші зовнішні системи, які можуть надавати інформацію про атрибути.

ABAC пропонує більш вільний і гнучкий підхід до контролю доступу, дозволяючи організаціям визначати політики на основі конкретних атрибутів і умов. ABAC може обробляти складні сценарії авторизації, підтримувати динамічні та контекстні рішення щодо доступу, а також полегшує впровадження політик у гетерогенних системах.

Контроль доступу становить дуже важливу частину веб-додатків, оскільки необхідно суворо дотримуватися розмежування доступу до ресурсів і даних залежно від привілеїв користувачів і особливо персональних даних, захист яких передбачений законодавчими аспектами.

2.3.3 Дискреційне керування доступом

Модель контролю доступу, відома як дискреційний контроль доступу (DAC), дозволяє власникам або адміністраторам встановлювати права доступу до ресурсів на свій розсуд. У системі DAC власник ресурсу має повноваження встановлювати права доступу та визначати, кому дозволяється отримати доступ до своїх ресурсів. Цей механізм контролю доступу широко використовується в різних операційних системах та файлових системах.

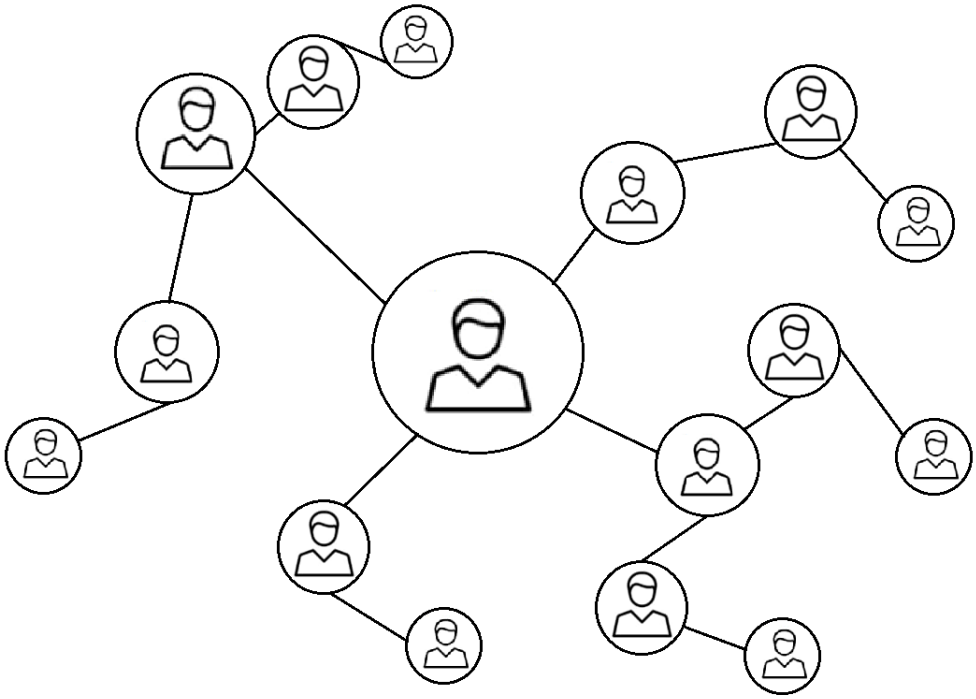

Рисунок 2.2 – Дискреційне керування доступом.

Нижче наведено ключові аспекти та принципи дискреційного контролю доступу:

- Контроль на основі власника: У DAC кожен ресурс асоціюється з власником, який має найвищі повноваження щодо дозволів доступу до цього ресурсу. Власник може вирішувати, які користувачі або групи мають доступ до ресурсу і який рівень доступу вони мають.

• Списки контролю доступу (ACL) – це список записів контролю доступу, пов'язаних з ресурсом, які визначають користувачів або групи та їхні дозволи на доступ до цього ресурсу. Кожен запис ACL зазвичай містить ідентифікатор користувача або групи та дозволи (наприклад, читання, запис, виконання), надані або заборонені.

• Успадкування прав доступу: DAC дозволяє успадкування прав доступу. Це означає, що коли ресурс створюється або змінюється, він може успадковувати дозволи доступу від батьківського каталогу. Це спрощує керування дозволами, застосовуючи узгоджений набір дозволів до всіх ресурсів у певному каталозі.

• Об'єктно-орієнтована авторизація: DAC визначає дозволи на доступ на основі об'єкта, який намагається отримати доступ до ресурсу, зазвичай це облікові записи користувачів або груп. Рішення про доступ приймається на основі дозволів, явно наданих або заборонених у списку ACL для цього об'єкта.

• Принцип найменших привілеїв: Хоча системи DAC є дискреційними, вони часто дотримуються принципу найменших привілеїв, який стверджує, що користувачам слід надавати лише мінімальні права доступу, необхідні для ефективного виконання їхніх завдань. Це мінімізує ризик несанкціонованого доступу або випадкового зловживання ресурсами.

• Обмеження: Одним з обмежень DAC є те, що він значною мірою покладається на рішення власника ресурсу щодо визначення прав доступу, що може призвести до невідповідностей або помилок. Крім того, DAC не забезпечує централізованого контролю або застосування політик, що ускладнює забезпечення послідовного контролю доступу у великій організації або в середовищі з високим рівнем регулювання.

Незважаючи на свої обмеження, DAC пропонує простоту і гнучкість, що робить його придатним для персональних або невеликих систем, де власники ресурсів несуть відповідальність за управління власними ресурсами. Він зазвичай використовується в комп'ютерних операційних системах, файлових серверах і додатках, де не потрібен суворий контроль над правами доступу.

Найбільш поширене застосування DAC відбувається у випадках, коли користувачі безпосередньо володіють деякими ресурсами і можуть самостійно вирішувати, кому дозволяти взаємодію з ними. Прикладом можуть бути операційні системи або соціальні мережі, де люди самостійно змінюють видимість їхнього контенту.

2.3.4 Механізми для забезпечення керування доступом

JSON Web Token – це широко застосовуваний механізм для реалізації контролю доступу в сучасних веб-додатках. Він представляє собою компактний та автономний формат токенів, який безпечно передає вимоги між сторонами. JWT використовується для передачі даних для аутентифікації у клієнт-серверних програмах. Сервер створює токени, які підписуються за допомогою секретного ключа, і передає їх клієнту, який потім використовує цей токен для підтвердження своєї ідентичності.

Токен JWT складається з трьох основних частин:

- заголовка (header);
- корисного навантаження (payload);
- підпису або зашифрованих даних.

Токени JWT для механізмів контролю доступу можна використовувати наступним чином:

- Для аутентифікації: коли користувач входить в систему або проходить аутентифікацію, сервер генерує JWT і вставляє відповідну інформацію про користувача, таку як ідентифікатор користувача або ім'я користувача, в запити на отримання токенів. Сервер підписує JWT секретним або приватним ключем, забезпечуючи цілісність та автентичність токена.
- Для авторизації: після того, як користувач автентифікований і має JWT, токен може бути використаний для авторизації. Сервер може включати вимоги до авторизації, такі як ролі користувачів або повноваження, в корисне навантаження JWT. Ці вимоги визначають дії або ресурси, до яких користувач має доступ.

• Для обміну токенами: токени JWT можуть обмінюватися між клієнтом і сервером для перевірки ідентичності користувача і дозволів. Клієнт додає JWT до наступних запитів, як правило, в заголовку авторизації, використовуючи схему резервного копіювання. Потім сервер перевіряє підпис JWT і витягує твердження для прийняття рішень щодо контролю доступу.

• Для перевірки токенів: коли сервер отримує JWT, він перевіряє цілісність токена шляхом перевірки підпису за допомогою спільного секретного або відкритого ключа. Він також перевіряє час закінчення терміну дії токена, який називається запитом на закінчення терміну дії, щоб переконатися, що термін дії токена не закінчився.

• Використовувати як один із кроків для створення системи контролю доступу на основі ролей (RBAC): JWT можна використовувати для реалізації RBAC шляхом включення ролей або дозволів користувачів у якості вимог до токена. Сервер може використовувати ці твердження, щоб визначити, чи має користувач дозвіл на виконання певних дій або доступ до певних ресурсів.

• Для детального контролю доступу: токен JWT може містити додаткові інструкції для забезпечення контролю доступу. Наприклад, користувацькі вимоги, такі як ідентифікатори ресурсів, діапазони або умови, можуть бути додані до корисного навантаження токена, щоб забезпечити більш детальне управління доступом.

• Для відкликання та закінчення терміну дії токенів: JWT може встановлювати час закінчення терміну дії, після якого токен вважається недійсним. Якщо права доступу користувача змінюються або токен потрібно відкликати, сервер може відкликати токен, скоротивши його термін дії або ведучи список відкликаних токенів.

Контроль доступу JWT дозволяють додаткам реалізовувати масштабовані механізми автентифікації та авторизації. Також забезпечує безпечний і портативний спосіб представлення вимог, ролей і дозволів користувачів, що дозволяє здійснювати детальний контроль доступу до ресурсів і дій.

2.4 Ін'єкційні атаки

Впровадження SQL-коду (SQL Injection) – це один з поширених методів злому веб-сайтів і програм, які працюють з базами даних. Цей метод полягає відправці в запит довільного SQL-коду, який потім виконується на сервері бази даних. Залежно від типу використовуваної системи управління базами даних (СУБД) та умов впровадження, атакуючий може отримати можливість виконати різноманітні операції з базою даних, такі як читання, видалення, зміна або додавання даних. Крім того, атака SQL Injection може дозволити зловмиснику отримати доступ до локальних файлів, виконати довільні команди на сервері, що піддається атакі, та виконати інші небажані дії.

Цей тип атаки можливий через некоректну обробку вхідних даних, які використовуються в SQL-запитах. Розробники веб-додатків, які працюють з базами даних, повинні бути свідомими про такі вразливості та приймати заходи для запобігання впровадженню SQL-коду.

Існує три основні класи атак, заснованих на впровадженні SQL-коду:

- Класична ін'єкція SQL (Classic SQLi).
- SQL ін'єкція, заснована на експлуатації СУБД повідомлень про помилки (Error-based SQLi).
- Сліпа SQL ін'єкція (Blind SQLi).

Уразливості в SQL-ін'єкції можуть мати серйозні наслідки, включаючи несанкціонований доступ до конфіденційних даних і потенційний витік даних. Впроваджуючи превентивні заходи та застосовуючи безпечні методи кодування, можна значно знизити ризик атак SQL-ін'єкцій. Регулярна оцінка безпеки та тестування на проникнення також може допомогти виявити та виправити уразливості SQL-ін'єкцій у вашому додатку.

2.4.1 Класична SQL ін'єкція

Класичні SQL-ін'єкції (SQLi) є найпоширенішим і найпростішим типом атак SQL-ін'єкцій. Це відбувається, коли зловмиснику вдається вставити шкідливі SQL-оператори в запит до бази даних програми, що призводить до несанкціонованого доступу або маніпуляцій з базою даних [10,11].

Далі я наведу приклад, як саме зазвичай працює класична SQL-ін'єкція:

- Зловмисник знаходить веб-додаток, вразливий до SQL-ін'єкцій. Ця вразливість виникає, коли додаток отримує дані від користувача і включає їх безпосередньо в SQL-запит без належної перевірки або параметризації.
- Зловмисник створює спеціальний запис, який може маніпулювати структурою або поведінкою SQL-запиту. Вхідні дані призначені для виходу з очікуваного контексту і введення SQL-коду, який програма не хоче виконувати.
- Вводячи зловмисні дані, зловмисник намагається модифікувати SQL-запит таким чином, щоб отримати, змінити або видалити дані, або навіть виконати довільні команди на сервері бази даних.

2.4.2 Сліпа SQL ін'єкція

Сліпі SQL-ін'єкції (Blind SQLi) - це тип SQL-атаки, який дозволяє зловмиснику отримати інформацію з відповіді програми, не бачачи безпосередньо результатів ін'єкційного SQL-запиту. Цей тип атаки зазвичай використовується, коли додаток не виявляє жодних помилок в базі даних або чутливих відповідей, які могли б розкрити результати ін'єкційних SQL-запитів.

Далі я наведу приклад, як саме зазвичай працює сліпа SQL-ін'єкція:

- Спочатку зловмисник визначає веб-додаток, який є потенційно вразливим до атаки SQL-ін'єкції. Це можна зробити за допомогою ручного або автоматизованого сканування.
- Далі зловмисник вставляє оператори SQL у поля введення або параметри користувача, так само, як і при традиційній SQL-ін'єкції. Однак у випадку Blind SQLi

зловмисник не отримує прямого зворотного зв'язку від програми про результати запиту.

- Аналіз відповіді програми, тобто замість того, щоб безпосередньо бачити результат запиту, зловмисник аналізує відповідь програми, щоб отримати з неї інформацію. Зазвичай це робиться шляхом надсилання спеціально створеного корисного навантаження та спостереження за поведінкою програми або часом відгуку.

- У деяких випадках зловмисник може використати Boolean Blind SQLi, вставляючи SQL-оператори, які обчислюють значення true або false. Аналізуючи варіації у відповіді програми, такі як затримка або певні повідомлення про помилки, зловмисник може отримати інформацію з бази даних або витягти дані поетапно.

- Інший варіант - часовий сліпий SQLi, де зловмисник вставляє SQL-оператори, які спричиняють затримки у відповіді програми. Спостерігаючи за часом відповіді, зловмисник може визначити, чи є певні умови істинними або хибними.

- Сліпа SQLi, заснована на помилках: методи, засновані на помилках, базуються на впровадженні операторів SQL, які навмисно генерують помилки у відповіді програми. Повідомлення про помилки можуть містити цінну інформацію про структуру бази даних або даних і можуть допомогти зловмиснику витягти інформацію.

- Останнім кроком буде вилучення інформації. За допомогою комбінації створення корисного навантаження, аналізу відповіді програми та висновків зловмисник поступово витягує конфіденційну інформацію з бази даних. Це можуть бути імена користувачів, паролі, будь-які записи або інші конфіденційні дані.

2.4.3 SQL-ін'єкція, заснована на експлуатації СУБД повідомлень про помилки (Error-based SQLi)

SQL-ін'єкції на основі помилок – це тип SQL-ін'єкцій, який використовує вразливості у веб-додатках, навмисно генеруючи помилки в базі даних. Вводячи шкідливі SQL-оператори в поля введення користувача, зловмисники намагаються

маніпулювати поведінкою програми та спровокувати повідомлення про помилки від сервера бази даних. Ці повідомлення про помилки можуть розкривати цінну інформацію про структуру бази даних, виконання запитів або основні дані, що дозволяє зловмисникам витягти конфіденційну інформацію або отримати несанкціонований доступ.

Далі я наведу приклад, як саме зазвичай працює SQL-ін'єкція заснована на повідомленнях про помилки у СУБД:

- Спочатку зловмисники визначають вразливі місця, які можуть бути використані для застосування SQL Injection. Це вони роблять можна за допомогою ручного аналізу або автоматизованих інструментів сканування.
- Далі зловмисники впроваджують ретельно розроблені шкідливі SQL-запроси в поля введення користувача, такі як форми для входу в систему, пошукові поля або розділи коментарів. Впроваджений SQL-код призначений для того, щоб викликати помилки в подальших запитах до бази даних.
- Зловмисники аналізують реакцію програми на впроваджений SQL-код, приділяючи особливу увагу повідомленням про помилки, що генеруються сервером бази даних. Ці повідомлення про помилки можуть відображатися безпосередньо користувачеві або записуватися в системні журнали.
- Виконують витяг інформації, зазвичай повідомлення про помилки часто містять цінну інформацію, таку як імена баз даних, імена таблиць, імена стовпців або конкретні коди помилок. Зловмисники витягують і аналізують цю інформацію, щоб отримати уявлення про структуру бази даних і дані.
- Зловмисники використовують видобуту інформацію для вдосконалення своєї стратегії атаки. Вони можуть проводити подальші атаки SQL Injection, виконувати несанкціоноване отримання або модифікацію даних, або навіть підвищувати свої привілеї, щоб отримати повний контроль над базою даних.

2.5 Захист від ін'єкційних атак

Як я вже зазначав раніше, SQL-ін'єкція – це форма вразливості безпеки, яка дозволяє зловмиснику налаштувати або впровадити шкідливий SQL-код у запит до бази даних веб-застосунку, надаючи доступ до системи без дозволу, маніпулюючи даними або займаючись іншими незаконними діями.

Далі я наведу перелік деяких заходів для забезпечення безпеки застосунку від вразливості SQLi:

Перевірка та параметризація вхідних даних:

- Перевірка вхідних даних гарантує, що вхідні дані користувача відповідають очікуваним шаблонам, вона запобігає прийняттю потенційно зловмисних даних. Вона включає перевірку типу, довжини, формату та діапазону вхідних значень.
- Санітарізація передбачає видалення або кодування спеціальних символів, які можуть бути використані для використання вразливостей SQL Injection.
- Параметризовані запити або підготовлені оператори дозволяють розробникам визначати заповнювачі для вхідних значень в SQL-запиті та окремо прив'язувати фактичні значення. Це гарантує, що користувацьке введення обробляється як дані, а не як виконуваний код, запобігаючи атакам SQL Injection.

Принцип найменших привілеїв:

- Принцип найменших привілеїв передбачає надання мінімально необхідних привілеїв обліковим записам користувачів бази даних. Це зменшує потенційну шкоду, яку може завдати зловмисник, навіть якщо йому вдасться успішно використати SQL Injection уразливість. Обмежте доступ лише до необхідних таблиць, збережених процедур.

Екранування та кодування:

- Екранування передбачає зміну спеціальних символів у користувацькому введенні, щоб запобігти їх інтерпретації як частини синтаксису SQL. Наприклад, екранування одинарних лапок (') двома одинарними лапками ('') запобігає передчасному закриттю рядкового літерала.

• Кодування передбачає перетворення спеціальних символів у їх еквівалентне кодоване представлення. Наприклад, перетворення одинарної лапки (') на її HTML-еквівалент (') гарантує, що вона сприйматиметься як буквальний символ, а не як синтаксис SQL.

Збережені процедури та параметризовані представлення:

• Використання збережених процедур або параметризованих представлень може допомогти зменшити ризики SQL-ін'єкцій. Ці об'єкти бази даних інкапсулюють логіку SQL і дозволяють розробникам безпечно передавати вхідні параметри. Вхідні значення обробляються як дані в процедурі або представленні, що зменшує ризик атак SQL Injection.

Білі списки та фільтрація вхідних даних:

• Білі списки передбачають визначення набору дозволених символів або шаблонів для введення користувачем. Будь-яке введення, яке не відповідає білому списку, відхиляється або дезінфікується. Такий підхід допомагає запобігти виконанню шкідливих SQL-запитів.

• Фільтрація вхідних даних передбачає аналіз введених користувачем даних і видалення або кодування будь-яких потенційно небезпечних символів або послідовностей. Це може включати фільтрацію ключових слів SQL, спеціальних символів або специфічного для SQL синтаксису.

Практики безпечної розробки:

• Навчання розробників безпечним методам кодування має вирішальне значення для запобігання уразливостей SQL Injection. Вони повинні знати про ризики, пов'язані з SQL-ін'єкціями, і розуміти важливість перевірки вхідних даних, рекомендацій щодо безпечного кодування та регулярного тестування безпеки.

• Дотримання практик безпечного кодування, таких як перегляд коду, використання безпечних бібліотек коду та навчання безпечному кодуванню, допомагає звести до мінімуму вразливості типу SQL Injection.

Регулярні оновлення безпеки:

• Важливо постійно оновлювати фреймворк додатків, веб-сервер та систему управління базами даних найновішими патчами безпеки. Постачальники

програмного забезпечення часто випускають оновлення безпеки, щоб усунути відомі вразливості та впровадити покращення безпеки. Своєчасне застосування цих оновлень допомагає захиститися від атак SQL Injection.

Брандмауери веб-додатків (WAF):

• Брандмауери веб-додатків – це рішення для забезпечення безпеки, які знаходяться між веб-додатком і користувачем. Вони можуть перевіряти вхідні запити і відповіді, виявляти шаблони, що вказують на атаки SQL Injection, і блокувати потенційно шкідливий трафік. Впровадження WAF може забезпечити додатковий рівень захисту від атак SQL Injection.

Принцип глибинного захисту:

• Принцип глибинного захисту підкреслює використання декількох рівнів контролю безпеки. Для запобігання SQL-ін'єкціям це може включати комбінацію перевірки вхідних даних, параметризації, безпечного кодування, безпечних мережевих конфігурацій та моніторингу. Кожен рівень додає додатковий бар'єр захисту, ускладнюючи зловмисникам використання вразливостей SQL Injection.

Регулярне тестування безпеки:

• Проведення регулярних оцінок безпеки, таких як тестування на проникнення та сканування вразливостей, має вирішальне значення. Тестування на проникнення передбачає моделювання реальних сценаріїв атак для виявлення та використання вразливостей SQLi. Сканування вразливостей.

Важливо відзначити, що запобігання SQL-ін'єкціям вимагає комплексного підходу, що поєднує в собі безпечні методи кодування, належну перевірку вхідних даних і глибоке розуміння базової системи баз даних. Регулярне оцінювання безпеки та постійне слідкування за новими методами захисту мають вирішальне значення для ефективного зниження ризику атак SQL Injection.

2.6 Міжсайтовий скриптинг

Міжсайтовий скриптинг (XSS) – це вразливість веб-додатків, яка дозволяє зловмисникам впроваджувати шкідливі скрипти на веб-сторінки, які переглядають

інші користувачі. Цей тип атаки виникає, коли додаток неналежним чином перевіряє дані, що вводяться користувачем, і дозволяє відображати ненадійні дані на веб-сторінках без належного значення або кодування [12].

XSS-атаки зазвичай націлені на поля введення користувача, такі як поля пошуку, розділи коментарів або форми, де програма не може перевірити або знезаразити дані, надані користувачем.

Далі я наведу більш детальне пояснення атаки типу Cross Site Scripting:

Існує два типи XSS:

• Збережений міжсайтовий скриптинг – це тип XSS-атаки, в якій шкідливий скрипт або корисне навантаження постійно зберігається на цільовому сервері, а потім стає доступним для інших користувачів, які отримують доступ до ураженої сторінки. Це відбувається, коли програма не перевіряє належним чином введення даних користувача і дозволяє зберігати та отримувати шкідливий код у базі даних сервера.

• Відображений міжсайтовий скриптинг – тип XSS-атаки, при якому шкідливий скрипт або корисне навантаження вбудовується в URL-адресу або інше поле введення і повертається користувачеві як частина відповіді програми. На відміну від збереженого XSS, шкідливий код не зберігається постійно на сервері, а динамічно впроваджується на веб-сайт.

Процес атаки XSS: зловмисник впроваджує скрипт або фрагмент коду під час введення користувачем даних. Коли інший користувач отримує доступ до ураженої сторінки, в його браузері виконується шкідливий скрипт.

Наслідки XSS-атак:

• Перехоплення сеансу: зловмисники можуть викрадати сесійні файли cookie або іншу конфіденційну інформацію користувачів, що дозволяє їм видавати себе за жертв і здійснювати несанкціоновані дії.

• Спотворення: зловмисники можуть змінювати вміст веб-сайту, замінюючи легітимний контент на власний шкідливий або виводячи його з ладу.

• Крадіжка даних: зловмисники можуть викрасти конфіденційну інформацію користувачів, таку як імена користувачів, паролі або дані кредитних карток, перехопивши дані та надіславши їх на віддалений сервер.

• Фішингові атаки: зловмисники можуть створювати фальшиві форми для входу або повідомлення, щоб обманом змусити користувачів ввести свої облікові дані або особисту інформацію.

Важливо вживати активні заходи для запобігання XSS-вразливостей, оскільки успішні атаки можуть призвести до витоку даних користувачів, шкоди репутації та потенційних юридичних наслідків. Впроваджуючи належну перевірку вхідних даних, кодування вихідних даних і тестування безпеки, організації можуть значно знизити ризик XSS-атак і підвищити загальний рівень безпеки своїх веб-додатків.

2.7 Захист від міжсайтового скриптингу

Як я вже зазначав раніше,міжсайтовий скриптинг – одна з вразливостей веб-додатків, яка дозволяє зловмисникам впроваджувати та виконувати шкідливі скрипти в браузерах нічого не підозрюючих користувачів. Це трапляється, коли програма не може належним чином перевірити або знезаразити дані, введені користувачем, що дозволяє впровадити шкідливий код [13].

Захист від міжсайтового скриптингу (XSS) полягає у впровадженні різних заходів безпеки для запобігання вставки та виконання шкідливих скриптів у веб-додатках. Ось детальний огляд методів захисту від XSS:

Далі я наведу перелік деяких методів для забезпечення безпеки застосунку від вразливості XSS:

Перевірка та очищення вхідних даних:

• Впровадьте сувору перевірку вхідних даних, щоб переконатися, що вхідні дані відповідають очікуваним форматам, типам даних і обмеженням по довжині.

• Використовуйте методи санітарної обробки даних для видалення або кодування спеціальних символів, які програма може інтерпретувати як код.

Вихідне кодування:

• Належним чином кодуйте або уникайте контенту, згенерованого користувачами перед його відображенням на веб-сторінках.

Політика безпеки вмісту (CSP):

• Впровадьте політику безпеки контенту, щоб встановити надійні джерела контенту та обмежити скрипти з неавторизованих джерел.

Тільки HTTP cookie:

• Використовуйте лише HTTP-файли cookie, щоб запобігти доступу клієнтських скриптів до конфіденційних сесійних файлів cookie.

• Встановивши прапорець "тільки HTTP" для файлів cookie, браузер обмежує їх доступ до JavaScript, зменшуючи ризик крадіжки файлів cookie.

Фільтрація вхідних і вихідних даних:

• Впроваджуйте вхідні та вихідні фільтри для виявлення та блокування потенційно шкідливого контенту.

• Використовуйте комбінацію методів фільтрації білих і чорних списків, щоб дозволити відомий безпечний вміст і заблокувати підозрілі або шкідливі шаблони.

Використання бібліотек і платформ безпеки:

• Використовуйте бібліотеки безпеки або платформи, які забезпечують вбудований захист від XSS-уразливостей.

• Фреймворки, такі як ASP.NET або Django, часто включають функції безпеки, такі як автоматичне маркування виводу або системи шаблонів, які забезпечують безпечний рендеринг.

Практики безпечної розробки:

• Навчіть розробників безпечним методам кодування, наголошуючи на важливості перевірки вхідних даних, шифрування вихідних даних та безпечної обробки користувацького вводу.

• Впроваджуйте настанови щодо безпечного кодування та проводьте аналіз коду, щоб виявити та виправити потенційні XSS-уразливості під час процесу розробки.

Брандмауер веб-додатків (WAF):

• Налаштуйте WAF, який може виявляти та блокувати XSS-атаки, аналізуючи вхідні запити та відповіді.

• WAF може допомогти захиститися від відомих і невідомих XSS-уразливостей, застосовуючи набір попередньо визначених правил безпеки.

Виконуйте регулярне тестування безпеки:

• Проводьте комплексні оцінки безпеки, включаючи сканування вразливостей, тести на проникнення та огляди коду для виявлення та усунення XSS-вразливостей.

• Автоматизовані інструменти та ручні тести можна використовувати для моделювання різних сценаріїв XSS-атак та перевірки ефективності впроваджених заходів безпеки.

Один із основних методів протидії вразливості XSS – це навчання користувачів:

• Поінформуйте користувачів про небезпеку переходу за незнайомими або підозрілими посиланнями та про важливість бути обережними при взаємодії з веб-додатками.

• Заохочуйте користувачів оновлювати свої браузери та додатки, щоб зменшити вплив можливих XSS-атак.

Впровадження багаторівневого підходу, що поєднує ці методи безпеки, може значно знизити ризик XSS-атак і сприяти підвищенню безпеки веб-додатків. Регулярний моніторинг, оновлення заходів безпеки та інформування про нові загрози мають важливе значення для надійного захисту від XSS-уразливостей.

Висновки за розділом 2

Розділ "Аналіз вразливих місць серверних застосунків та їх захист" дозволив нам детально дослідити різні аспекти безпеки серверних застосунків і виявити потенційні вразливості, які можуть бути використані зловмисниками для атак. Під час аналізу ми розглянули вектори атак, такі як порушення контролю доступу, ін'єкційні атаки та міжсайтовий скриптинг, і проаналізували їх вплив на безпеку серверних застосунків.

Для кожного вектора атаки ми розглянули різні методи захисту. Для порушення контролю доступу ми розглянули керування доступом на основі ролей, керування доступом на основі атрибутів, дискреційне керування доступом та механізми для забезпечення керування доступом. Для ін'єкційних атак ми розглянули класичну SQL ін'єкцію, сліпу SQL ін'єкцію та SQL ін'єкцію, засновану на експлуатації СУБД

повідомлень про помилки, і запропонували заходи захисту, такі як валідація та екранізація вхідних даних та використання параметризованих запитів. Для міжсайтового скриптингу ми розглянули захист через екранізацію та експортування вхідних даних, використання заголовків Content Security Policy (CSP) та використання фреймворків і бібліотек з автоматичним екрануванням.

Аналіз вразливих місць серверних застосунків та їх захист має велике значення для забезпечення безпеки систем. Застосування рекомендацій і методів захисту, запропонованих у цьому розділі, може значно знизити ризик вразливостей і захистити серверні застосунки від потенційних атак зловмисників.

Під час подальших досліджень рекомендується розширити аналіз і розглянути інші типи вразливостей та атак, а також провести практичні експерименти для перевірки ефективності запропонованих методів захисту. Вироблення та реалізація комплексної стратегії безпеки є важливим завданням, що допоможе забезпечити стійкість та надійність серверних застосунків у сучасному середовищі IT.

РОЗДІЛ 3
ВИБІР ОПТИМАЛЬНИХ МЕТОДІВ ЗАХИСТУ СЕРВЕРНИХ ЗАСТОСУНКІВ ТА РОЗРОБКА РЕКОМЕНДАЦІЙ

3.1 Використання системи аналізу DPI

Система DPI (Deep Packet Inspection), як ми можемо побачити видно з назви, виконує глибокий аналіз усіх пакетів, що проходять через неї. Крім вивчення пакетів за певними стандартними патернами, за якими можна однозначно визначити приналежність пакета до певного застосунку, скажімо, за форматом заголовків, номерами портів тощо, система DPI здійснює і так званий поведінковий аналіз трафіку, що дає змогу розпізнати застосунки, які не використовують для обміну даними заздалегідь відомі заголовки та структури даних [14].

Deep Packet Inspection або DPI – це технологія накопичення статистичних даних, перевірки та фільтрації мережевих пакетів за їхнім вмістом. У DPI застосовують кілька методів аналізу трафіку, серед них найбільш використовувані - евристичний і сигнатурний аналіз. Евристичний аналіз - здатність виявляти шкідливі програми в трафіку, які невідомі антивірусам. Є частиною сигнатурного аналізу. Зі свого боку сигнатурний аналіз являє собою дослідження трафіку за допомогою "підписів".

Традиційні методи перевірки пакетів зазвичай перевіряють лише заголовки пакетів, які містять інформацію про адресу джерела та призначення, порт і протокол. DPI, навпаки, виходить за межі заголовків і перевіряє фактичний вміст пакетів, що дозволяє проводити більш комплексний аналіз.

Зверніть увагу, що DPI включає відстеження вмісту веб-трафіку, що може викликати занепокоєння щодо конфіденційності. При впровадженні технології DPI організації та постачальники послуг повинні забезпечити належний рівень конфіденційності та дотримання правових норм.

DPI покращує видимість, безпеку та контроль мережі, дозволяючи організаціям краще розуміти свою мережеву інфраструктуру та керувати нею.

Щоб добитися справедливості, необхідно не лише відстежувати злочинні події у системах, а й притягувати до відповідальності тих, хто порушує закон. Типи даних, які залишаються під час роботи з веб-застосунками наведені у таблиці 3.1.

Таблиця 3.1

Дані, які можна відстежити

Ідентифікатор	**Зміст даних, які залишаються**
IP-адреса	Як мінімум, залишається інформація про місцезнаходження користувача, та базова інформація про провайдера.
MAC-адреса	Під час подключення користувача до загальнодоступної точки доступу WiFi, можна дізнатися MAC-адресу мережного інтерфейсу цього користувача.
DNS leaks	Якщо програмне забезпечення надсилає DNS-запити через DNS-сервер провайдера, існує ймовірність реєстрації активності клієнта та витоку інформації зі служби доменних імен.

Більшість браузерів містить такі функціональні компоненти:

• Файли cookie — це файли, що містять текстові дані, які використовуються програмами для різних цілей, наприклад для автентифікації. Ідентичність користувача розкривається, коли він спочатку отримує доступ до ресурсу через відкритий сеанс. У цей момент браузер зберігає файли cookie, а згодом клієнт встановлює з’єднання через анонімний хост. На завершальному етапі зв’язку з

сервером файли cookie відображаються та стають доступними. Цей процес призводить до деанонімізації користувача.

• Плагіни - це програмні технології, які засновані на технологіях Flash і Java, які завантажуються на стороні клієнта та можуть обходити проксі-сервери для функціонування, вони також зберігають свої власні файли cookie та налаштування.

• Браузер передає на сервер безліч категорій даних, що дозволяє створити унікальний цифровий відбиток браузера користувача. Цей відбиток дозволяє ідентифікувати браузер серед незліченних інших, навіть під час анонімних сеансів, зазвичай цифровий відбиток використовується для надання якісної цільової реклами.

• Скрипти JavaSctipt, вид коду, створений користувачами за допомогою сценаріїв JavaScript, має потенціал для накопичення особистої інформації для сервера, і навіть з потенційними вразливими місцями в ресурсах клієнта може створити можливості для успішних атак на інформаційний ресурс.

• Веб-сайт може визначити джерело трафіку за допомогою заголовка http-referrer. Цей заголовок дозволяє веб-сайту визначити точну особу чи організацію, яка створила трафік.

Налаштування параметрів безпеки браузера є рекомендованим рішенням для вирішення таких проблем, яке передбачає блокування всіх згаданих вище категорій.

Система, відома як Deep Packet Inspection, або DPI, включає систему збору статистичних даних, перевірки та фільтрації мережевих пакетів на основі їх вмісту. На відміну від традиційних брандмауерів, які аналізують виключно заголовки пакетів, DPI проводить ретельний аналіз повного вмісту трафіку. Це дозволяє DPI виявляти та запобігати поширенню вірусів, фільтрувати нерелевантні дані та проводити комплексний аналіз усіх пакетів, які проходять через нього.

Спеціальні служби мають можливість відстежувати активність конкретного користувача на веб-сайті та ретельно перевіряти трафік VPN і HTTPS, якщо це необхідно, використовуючи DPI. DPI може збирати різну інформацію без порушення особистих прав користувача. Крім того, DPI може виконувати такі функції:

• Виявлення спам-ботів стало можливим завдяки аналізу трафіку SMTP.

• Аномалії трафіку можна використовувати для виявлення атак DoS і DDoS.

• Виявлення вірусної інфекції ґрунтується на її унікальних сигнатурах. Коли адреса відправника надсилає аномально великий обсяг запитів SMTP, активується захист від спаму за допомогою механізму блокування.

3.2 Використання систем WAF та IPS, як базової системи захисту

Щоб підвищити ефективність систем безпеки веб-ресурсів, необхідно організувати системи, які можуть протидіяти атакам різниз рівнів. Одним із способів досягти цього є використання систем IPS та WAF. WAF - система безпеки, яка захищає веб-додатки від різних видів атак, включно з ін'єкціями, крос-сайтовим скриптингом, підробленням та іншими [15]. Вона контролює і фільтрує HTTP-трафік, виявляючи і блокуючи потенційно небезпечні запити, вихідні або вхідні, з метою запобігання вразливостей і забезпечення безпеки веб-додатків. IPS, з іншого боку, здійснює моніторинг атак низького рівня в реальному часі та оперативно реагує на будь-які зміни в потоці трафіку [16,17].

Система запобігання вторгненням – це механізм, який дозволяє ідентифікувати ознаки вторгнення у систему, а також виявляти та запобігати нападам. Під час аналізу система використовує кілька методів виявлення атак, таких як поведінковий, на основі сигнатур та виявлення аномалій протоколу.

За замовчуванням усі технології IPS здатні виконувати такі функції:

• IPS має здатність запобігати впливу атаки.

• Запобігає проникненню зловмисного фрагмента, дозволяючи неураженій частині входити в систему.

• У разі серйозних подій у системі важливо,щоб надавалося повідомлення адміністраторам безпеки щодо таких подій.

• У відповідь на події, проводять реагування,щоб завадити успіху атаки.

• Створювати звіти у потрібному адміністратору форматі.

Основні типи подій, які частіше за все виявляються системою запобігання вторгненням:

• Багато типів атак, включаючи розвідку та атаки на рівні додатків, такі як підбір пароля, переповнення буфера або передача зловмисного програмного забезпечення.

• Ретельно перевіряють протоколи додатків, щоб виявити та запобігти будь-якій шкідливій діяльності.

• Розвідка та атаки на мережевому рівні можуть приймати форму підміни IP-адреси та ненормальних значень IP-заголовків.

• Запуск програм , наприклад може призвести до виконання хостами дій, які не були схвалені.

• Недотримання політики призводить до використання заборонених протоколів.

WAF (Web Application Firewall) - міжмережевий екран для веб-додатків. Це інструмент для фільтрації трафіку, що працює на прикладному рівні та захищає веб-додатки методом аналізу трафіку.

Основні завдання WAF включають у себе:

• Швидко реагувати на будь-які різновиди атак на веб-додатки, які входять до OWASP Top 10.

• Захист забезпечується заданими активними правилами.

• Перевіряти трафік, що надходить на застосунок, та інші запити, адресовані веб-додаткам, після чого ухвалювати рішення на підставі заданих правил і політики (блокувати, дозволити, надіслати повідомлення).

• Запобігати витоку інформації, перевіряючи вихідний від веб-додатків трафік, і вживати заданих заходів на підставі заданих активних правил.

• Постійно вести журнал подій і записувати в нього всі виконані операції, аналітичну інформацію та інші події, що відбулися.

• Перевіряти всі вхідні дані, що застосовуються для надсилання/отримання інформації від веб-додатків.

• Захищати від атак, спрямованих конкретно на сам Web Application Firewall.

Головна відмінність міжмережевого екрана вебзастосунків від інших методів захисту веб-додатків – це саме глибокий аналіз трафіку протоколів на прикладному рівні моделі OSI.

Для посилення захисту застосунку на другому рівні моделі OSI ми будемо використовувати прозорий брандмауер, також відомий як брандмауер-мост, - це програма 2-го рівня, яка легко встановлюється в існуючу мережу без зміни адреси Інтернет-протоколу (IP).

Прозорий брандмауер не є маршрутизатором, а діє як міст, перевіряючи і переміщуючи мережеві кадри між інтерфейсами.

Прозорий брандмауер можна розглядати як "невидимий брандмауер", який підтримує зовнішні та внутрішні інтерфейси. За допомогою прозорого брандмауера обладнання безпеки підключається до однієї мережі через внутрішні та зовнішні порти, з окремою віртуальною локальною мережею (VLAN) для кожного інтерфейсу.

Активація прозорого режиму на брандмауері переводить його з режиму маршрутизації 3-го рівня в режим мостового пристрою 2-го рівня. Це допомагає організаціям вирішувати проблеми, пов'язані з прозорістю трафіку і захистом від загроз, без необхідності змінювати архітектуру мережі.

Ключовою перевагою прозорого міжмережевого екрана моста є те, що його можна швидко розгорнути без складних процесів налаштування. Це порівняно з режимом маршрутизації, який вимагає значної переадресації IP і мережі, що може бути складним і тривалим процесом.

Оскільки прозорі брандмауери не мають IP-адреси в мережі, вони більш непомітні та непомітні для зловмисників. Це дає велику перевагу в плані безпеки, оскільки означає, що мережа менш сприйнятлива до хакерських атак.

У таблиці 3.2 наведено повну структуру функціонування комплексної системи, у якій поєднуються Transparent Firewall, фрагменти докладного аналізу пакетів, системи запобігання вторгнень та міжмережевого екран вебзастосунків у рамках моделі OSI.

Таблиця 3.2

Захист по рівням моделі OSI

Рівень моделі OSI	**Transparent Firewall**	**Докладний аналіз пакетів**	**Система запобігання вторгнень**	**Міжмережевий екран вебзастосунків**
Канальний	+			
Мережевий		+	+	
Транспортний		+	+	
Сеансовий			+	
Представлення			+	
Прикладний				+

Отже, даний комплекс систем буде захищати наш WEB-застосунок на усіх рівнях моделі OSI (базової еталонної моделі взаємодії відкритих систем).

Використання WAF (брандмауера веб-додатків) у поєднанні із DPI або системою запобігання вторгнень забезпечує 30 відсотків підвищення продуктивності порівняно з використанням базового WAF.

Завдяки використанню сучасної високошвидкісної системи запобігання вторгнень і технології DPI доступ шкідливих файлів до мереж може бути заблокований, що забезпечує додатковий захист від цілеспрямованих атак на IT-ресурси. Використовуючи цей комплекс, можна укріпити будь-який онлайн-ресурс і зменшити навантаження на системних адміністраторів IT систем.

3.3 Розробка рекомендацій для захисту серверних застосунків

Система захисту веб-застосунка – це комплекс технологій, процедур та інструментів, призначених для захисту веб-застосунків від різних загроз та забезпечення їх безпеки. Вона включає в себе різноманітні заходи та захисні механізми, які сприяють виявленню, запобіганню і реагуванню на потенційні атаки та вразливості.

Система захисту веб-застосунків є важливою річчю з кількох причин:

• Захист від кібератак: Веб-застосунки постійно стикаються з різноманітними кібератаками, такими як DDoS-атаки, SQL-ін'єкції, кросс-сайтові скриптові атаки та багато інших. Система захисту веб-застосунків допомагає виявляти, блокувати та мінімізувати ризик таких атак, що забезпечує цілісність, конфіденційність та доступність даних.

• Захист конфіденційності та персональних даних: Веб-застосунки можуть містити чутливу інформацію, таку як особисті дані користувачів, фінансові дані або комерційну інформацію. Безпека веб-застосунків є важливою, щоб запобігти несанкціонованому доступу до таких даних та їх витоку.

• Забезпечення надійності та доступності: Кібератаки можуть призвести до відмови в обслуговуванні (DoS) або перебоїв в роботі веб-застосунків, що може призвести до втрати доходів, погіршення репутації та незадоволення користувачів. Система захисту допомагає запобігти таким ситуаціям шляхом виявлення та блокування атак, забезпечуючи надійну та доступну роботу веб-застосунків.

• Відповідність нормативним вимогам: Багато секторів, таких як фінанси, охорона здоров'я та роздрібна торгівля, мають специфічні нормативні вимоги щодо безпеки даних. Система захисту веб-застосунків допомагає виконувати такі вимоги та забезпечувати відповідність нормативним стандартам.

В цілому, система захисту веб-застосунків є важливою, оскільки вона допомагає захищати веб-застосунки від широкого спектру загроз, забезпечує безпеку даних, забезпечує надійність та доступність веб-системи та допомагає виконувати нормативні вимоги.

Авжеж, для того, щоб побудувати надійну комлексну систему захисту серверних застосунків, треба розглянути модель OSI.Так як модель OSI являє собою стандартний набір рівнів і протоколів, які дають змогу різним компонентам комп'ютерної мережі взаємодіяти один з одним.

Пропоную розглянути кожен рівень моделі:

• На першому рівні моделі OSI відбувається передача фізичних сигналів (струмів, світла, радіо) від джерела до одержувача [18,19]. На цьому рівні ми

оперуємо кабелями, контактами в роз'ємах, кодуванням одиниць і нулів, модуляцією тощо.Зазначимо, що як носій даних можуть виступати не тільки електричні струми. Радіочастоти, світлові або інфрачервоні хвилі використовуються також повсюдно в сучасних мережах.

• Далі ми отримуємо фізичний сигнал з першого рівня - фізичного. Це набір напруг різної амплітуди, хвиль або радіочастот. Під час отримання, на другому рівні перевіряються і виправляються помилки передачі. Тут з'являються перші ідентифікатори – MAC-адреси.

• Мережевий рівень вводить термін "маршрутизація" і, відповідно, IP-адресу. До речі, для перетворення IP-адрес на MAC-адреси і назад використовується протокол ARP.

• Транспортний рівень призначений для передавання надійної послідовностей даних довільної довжини через комунікаційну мережу від відправника до одержувача.

• Сеансовий рівень контролює структуру проведення сеансів зв'язку між користувачами. Він займається встановленням, підтриманням і перериванням сеансів, фіксує, яка зі сторін є активною на даний момент, здійснює синхронізацію обміну інформацією між користувачами.

• Рівень представлення займається представленням даних, що передаються прикладними процесами в потрібній формі. Дані, отримані від додатків з прикладного рівня, на рівні представлення перетворюються у формат, придатний для передачі їх мережею, а отримані мережею дані перетворюються у формат додатків.

• Прикладний рівень надає інтерфейс для взаємодії додатків із мережею. Тут працюють різні прикладні протоколи. Тут знаходяться мережеві служби, які дають змогу нам, як кінцевим користувачам, серфити простори інтернету.

На основі розглянутої моделі я обрав компоненти, які будуть входити у рекомендовану мною комплексну систему.Як я вважаю комплексна системи захисту для веб-застосунків на основі Transparent Firewall, IPS, DPI та WAF може забезпечити високий рівень безпеки системи.

Ось загальний опис компонентів системи захисту який я пропоную та їх функції:

Комплексна система захисту веб-застосунків, що включає Transparent Firewall, IPS, DPI та WAF, забезпечує захист системи на різних рівнях моделі OSI. Розглянемо, ці інструменти та методи захисту, які вони забезпечують на кожному рівні.

Фізичний рівень.Треба контролювати фізичний доступ до мережевого обладнання, такого як маршрутизатори та комутатори, запобігаючи несанкціонованому доступу до них. Також важливо забезпечувати захист від фізичних загроз, наприклад, шляхом обмеження фізичного доступу до серверних приміщень.

Канальний рівень. На канальному рівні Transparent Firewall буде використовуватись для контролю доступу до MAC адрес. Що захистить від атаки типу підміни MAC адреси (коли зловмисник замінює MAC адресу на свою, за рахунок чого намагається приймати пакети даних на свій пристрій). Transparent Firewall може обмежувати кількість підключь з однієї MAC адреси.

Мережевий та транспортний рівень.На цих рівнях буде працювати IPS разом з DPI, системи запобігання вторгнень досліджують трафік даних для виявлення нерегулярної активності. Хакери використовують наступний підхід, щоб обійти дані системи – це розділити пакети на менші сегменти, тоді при скануванні IPS не буде виявляти атаку, тому що жоден пакет не містить повної шкідливої сигнатури, тут нам і допоможе DPI, оскільки він буде збирати потоки пакетів з одного джерела, тому зможе виявити атаки навіть коли вони розділяються на декілька вхідних пакетів.

На сеансовому рівні найпоширенішими атаками будуть атаки перехоплення сеансу та атаки на аутентифікацію, для запобігання цьому ми будемо використовувати IPS, так як система запобігання вторгнень буде використовувати сигнатури атак, щоб запобігти вже відомим видам проведення атак, та буде виявляти незвичну активність та використання недійсних ідентифікаторів сеансу або спроби недійсної аутентифікації.

На рівні представлення частіше за все відбувається витік інформації або атаки на протокол SSL, у цьому нам також допоможе система запобігання вторгнень, вона буде фільтрувати вихідний трафік на вміст конфіденційних даних, таких як номери

кредитних карток або особисті дані.Також новітні системи IPS можуть аналізувати шифрований трафік SSL на вміст шкідливого коду, захищати від атак типу Man-in-the-Middle коли зловмисник намагається перехопити і розшифрувати шифрований трафік, а також фільтрувати конкретні типи файлів на шкідливий вміст, якщо це потрібно.

На прикладному рівні ми будемо використовувати міжмережевий екран вебзастосунків (WAF). На цьому рівні зловмисники дуже часто використовують атаки типу SQL-ін’єкцій або кроссайтові скриптинги, а також DDoS атаки. Функціями WAF буде фільтрація вхідних даних які надходять до застосунку, WAF буде перевіряти трафік на наявність шкідливого коду, також в міжмережевого екрану вебзастосунків є фукції розпізнавання та блокування ботів, функція обмеження швидкості трафіку, щоб захистити застосунок від DDoS атак.WAF зазвичай розміщується перед веб-додатком, слугуючи таким собі щитом між цим додатком та інтернетом. Тоді як проксі захищає дані клієнта, WAF можна назвати "зворотним проксі", оскільки він захищає сервер, пропускаючи клієнтів через себе, перш ніж надати доступ до ресурсу.

Авжеж окрім цих компонентів, рекомендується включити такі базові методи захисту інформації, як:

- Антивірусне програмне забезпечення: Встановлення антивірусного програмного забезпечення на сервери, що хостять веб-застосунки, допоможе виявити та блокувати шкідливі програми та віруси.

- Моніторинг та аналіз журналів: Регулярний моніторинг та аналіз журналів подій, логів серверів та інших компонентів дозволяють виявляти аномалії, підозрілу активність та атаки.

- Резервне копіювання та відновлення: Як можна створювати систему безпеки без регулярного резервного копіювання даних веб-застосунків та налаштувань, бо цей компонент допоможе відновити систему в разі випадкового видалення або пошкодження даних внаслідок атаки.

- Регулярні оновлення та патчі: Регулярно оновлюйте всі компоненти системи, включаючи веб-сервери, операційні системи, веб-фреймворки та інші програмні засоби, щоб усунути вразливості та підвищити безпеку.

Крім того, важливо провести аналіз ризиків і врахувати конкретні потреби та характеристики вашої веб-системи, щоб налаштувати систему захисту відповідно до ваших вимог.

Комбінація цих компонентів дозволяє системі захисту забезпечити захист на усіх рівнях. Вони взаємодоповнюють один одного і створюють потужний механізм захисту, що дозволяє виявляти і блокувати широкий спектр атак та загроз. Крім того, ці компоненти можуть бути налаштовані на регулярне оновлення, щоб враховувати нові загрози та вразливості, що забезпечує постійний рівень безпеки.

Висновки за розділом 3

Безпека веб-застосунків є надзвичайно важливим аспектом сучасного інтернет-середовища. Розробка комплексної системи захисту, яка поєднує в собі Firewall, IPS, DPI та WAF, є ефективним підходом до забезпечення безпеки веб-системи.

Використання Firewall дозволяє контролювати мережевий трафік та блокувати небажані підключення, що запобігає несанкціонованому доступу. IPS виявляє та блокує атаки, аномальну активність та небезпечний трафік, забезпечуючи раннє виявлення потенційних загроз. DPI глибоко аналізує пакети даних, що протікають через мережу, для виявлення шкідливого вмісту та вразливостей. WAF спеціалізується на захисті веб-додатків і блокує атаки, спрямовані на їх вразливості.

Комплексна система захисту, яка поєднує ці елементи, забезпечує надійний рівень безпеки веб-застосунків. Вона допомагає запобігти несанкціонованому доступу, атакам, вразливостям та зберігає конфіденційність, цілісність та доступність даних. Крім того, ця система захисту відповідає вимогам нормативів та стандартів безпеки.

При виборі оптимальних методів захисту та розробці рекомендацій слід враховувати конкретні потреби та характеристики веб-застосунку. Аналіз загроз, оцінка ризиків і вибір відповідних заходів захисту є важливими кроками для створення безпечної системи захисту веб-застосунків.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проаналізовано існуючі вразливості, методи та засоби захисту серверних застосунків. Для захисту від WEB-атак класичним пристроєм є Web Application Firewall, який застосовує набір правил захисту до протоколів високого (прикладного) рівня. Але цього не достатньо, тому в роботі пропонується комплексне рішення, яке включає в себе WAF,DPI, IPS та Transparent Firewall, тобто захист на всіх рівнях моделі OSI.

У ході вивчення теми про сучасні серверні застосунки, їх вразливості та методи захисту, було розглянуто декілька важливих аспектів. У першому розділі ми отримали загальне уявлення про серверні застосунки, їх роль і важливість у сучасному світі. Дізналися про основні складові та функції серверних застосунків, а також про різні види їх архітектур.

Захист серверних застосунків є важливим, оскільки вони використовуються для обробки та збереження цінної інформації. Вразливості або атаки на застосунки можуть призвести до незаконного доступу до даних, втрати конфіденційності. Ефективна система захисту веб-застосунків допомагає запобігати таким загрозам, виявляти та блокувати атаки, забезпечувати цілісність, конфіденційність і доступність даних, а також забезпечувати виконання вимог щодо безпеки.

У другому розділі ми детально розглянули вразливості, які можуть бути використані зловмисниками для атак на серверні застосунки. Проаналізували такі загрози, як SQL-ін'єкції, перехоплення сесій, вразливості протоколів та інші. Порушивши безпеку серверних застосунків, зловмисники можуть отримати несанкціонований доступ до даних, виконувати шкідливі дії.

У третьому розділі було розглянуто вибір оптимальних методів захисту для серверних застосунків. З'ясували, що комбінація захисних технологій, таких як Transparent Firewall, IPS, DPI та WAF, може забезпечити комплексний захист. Ці

методи дозволяють виявляти та блокувати шкідливий трафік, запобігати атакам, виявляти вразливості та контролювати доступ до серверних застосунків.

Загальною метою цієї роботи було розроблення рекомендацій щодо вибору та впровадження оптимальних методів захисту для серверних застосунків. Використання комплексної системи захисту на основі Transparent Firewall, IPS, DPI та WAF допоможе підвищити безпеку серверних застосунків на усіх рівнях моделі OSI та зменшити ризик вразливостей і атак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jason N. A Comparison of Thin-Client Computing Architectures [Електронний ресурс] / N. Jason, Y. S.Jae, N. Naomi. – 2000. – Режим доступу до ресурсу: https://academiccommons.columbia.edu/doi/10.7916/D8Z329VF.

2. Davis K. The Definitive Guide to Network Programming / K. Davis, J. Turner, Y. Nathan. – USA: Springer-Verlag New York, 2004.

3. Hura G. Client-server computing architecture: an efficient paradigm for project management [Електронний ресурс] / Gurdeep Hura. – 1995. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/523924.

4. Haroon S. O. Client-Server Model / Shakirat Oluwatosin Haroon. // IOSR Journal of Computer Engineering. – 2014. – №6. – С. 1–3.

5. Aarsten A. Patterns for three-tier client/server applications / A. Aarsten, D. Brugali, G. Menga. // Proceedings of Pattern Languages of Programs. – 1996. – №6.

6. What is three-tier architecture? [Електронний ресурс] // IBM – Режим доступу до ресурсу: https://www.ibm.com/topics/three-tier-architecture.

7. Gallaugher J. Choosing a Client/Server Architecture / J. Gallaugher, S. Ramanathan. // Information Systems Management. – 2007. – №2. – С. 7–13.

8. Evans G. Two-tier Vs Three-tier Architecture [Електронний ресурс] / Gacheru Evans. – 2019. – Режим доступу до ресурсу: https://medium.com/@gacheruevans0/2-tier-vs-3-tier-architecture-26db56fe7e9c.

9. Schläpfer M. Applied Information Security / M. Schläpfer, D. Basin, P. Schalle., 2011.

10. Roy S. A network based vulnerability scanner for detecting SQLI attacks in web applications [Електронний ресурс] / S. Roy, A. Kumar. – 2012. – Режим доступу до ресурсу: https://www.researchgate.net/publication/254031416_A_network_based_vulnerability_scanner_for_detecting_SQLI_attacks_in_web_applications.

11. Pramod A. SQLI detection system for a safer web application [Електронний ресурс] / A. Pramod, G. Agneev, A. Mohan. – 2015. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7154705

12. Gupta S. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art / S. Gupta, B. Gupta. // International Journal of System Assurance Engineering and Management. – 2017. – №8. – С. 512–516.

13. Sarmah U. A survey of detection methods for XSS attacks / U. Sarmah, D. Bhattacharyya, J. Kalita. // Journal of Network and Computer Applications. – 2018. – С. 113–130.

14. Deep Packet Inspection as a Service [Електронний ресурс] / A.Bremler-Barr, D. Hay, Y. Koral, Y. Harchol. – 2014. – Режим доступу до ресурсу: https://conferences2.sigcomm.org/co-next/2014/CoNEXT_papers/p271.pdf.

15. Clincy V. Web Application Firewall: Network Security Models and Configuration [Електронний ресурс] / V. Clincy, H. Shahriar. – 2018. – Режим доступу до ресурсу: https://journals.scholarsportal.info/details/07303157/v01inone/835_wafnsmac.xml.

16. Stiawan D. The trends of Intrusion Prevention System network [Електронний ресурс] / D. Stiawan, A. Hanan. – 2010. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/5529697.

17. INTRUSION DETECTION SYSTEM AND INTRUSION PREVENTION SYSTEM. // International Journal of Computing and Business Research. – 2013. – №4.

18. Mughal A. A. Cyber Attacks on OSI Layers: Understanding the Threat Landscape / Arif Ali Mughal. // Journal of Humanities and Applied Science Research. – 2020. – С. 33–40.

19. Kumar S. The OSI model: Overview on the seven layers of computer networks / S. Kumar, S. Dalal, V. Dixit. // International Journal of Computer Science and Information Technology Research. – 2014. – №3. – С. 461–466.