### ЛЕКЦІЯ 9. БЕЗПЕКА КЛІЄНТ-СЕРВЕРНОЇ ВЗАЄМОДІЇ. ОСНОВНІ ЗАГРОЗИ ТА ВРАЗЛИВОСТІ. ШИФРУВАННЯ ДАНИХ (TLS/SSL). АУТЕНТИФІКАЦІЯ КОРИСТУВАЧІВ (OAUTH, JWT)

Безпека клієнт-серверної взаємодії — це важливий аспект розробки будь-яких веб-додатків та інформаційних систем. Вразливості в системах можуть призвести до витоку конфіденційних даних, шахрайства та порушення конфіденційності користувачів. У цій лекції ми розглянемо основні загрози та вразливості в клієнт-серверній взаємодії, зосередимо увагу на шифруванні даних за допомогою TLS/SSL та обговоримо методи аутентифікації користувачів за допомогою OAuth та JWT.

**1. Основні загрози та вразливості в клієнт-серверній взаємодії**

У процесі передачі даних між клієнтом і сервером існує багато потенційних загроз. Безпека даних, які передаються через мережу, стає критичним елементом для забезпечення цілісності та конфіденційності інформації. Розглянемо найпоширеніші загрози та вразливості.

**1.1 Перехоплення даних (Man-in-the-Middle Attack)**

Однією з найбільш поширених загроз є атака «людина посередині» (Man-in-the-Middle, MITM). Під час цієї атаки зловмисник перехоплює трафік між клієнтом та сервером і може змінювати або зчитувати дані, що передаються. Без шифрування дані можуть бути легко доступні для аналізу та маніпуляції.

**1.2 Атаки на сеанс (Session Hijacking)**

Атака на сеанс — це тип атаки, коли зловмисник викрадає ідентифікатор сесії користувача і отримує доступ до ресурсу без введення логіну та пароля. Це може статися, якщо ідентифікатор сесії передається незашифрованим або зловмисник отримав доступ до cookie.

**1.3 XSS (Cross-Site Scripting)**

XSS є однією з найпоширеніших вразливостей у веб-додатках. Вона виникає, коли зловмисник вводить шкідливий код на сторінку, яка відображається іншому користувачеві. Це дозволяє виконувати небезпечні дії від імені жертви або викрадати її дані.

**1.4 SQL Injection**

SQL-ін'єкція є вразливістю, коли зловмисник вводить шкідливі SQL-команди у форму вводу або запит на сервер, що дозволяє маніпулювати базою даних або отримувати несанкціонований доступ до конфіденційної інформації.

**1.5 DoS/DDoS атаки**

Атаки типу Denial of Service (DoS) та Distributed Denial of Service (DDoS) спрямовані на перевантаження серверів або сервісів великою кількістю запитів. Це може призвести до зниження продуктивності або навіть повної відмови в обслуговуванні користувачів.

**1.6 Фішинг**

Фішинг є соціальною інженерією, яка використовується для крадіжки облікових даних користувача. Зловмисники створюють фальшиві веб-сторінки або електронні листи, що імітують справжні сервіси, щоб виманити у користувачів конфіденційні дані.

**2. Шифрування даних: TLS/SSL**

Щоб захистити дані від перехоплення та модифікації, використовуються протоколи шифрування. Основними стандартами для захисту даних при передачі через мережу є SSL (Secure Sockets Layer) і TLS (Transport Layer Security).

**2.1 Історія та еволюція TLS/SSL**

SSL був розроблений компанією Netscape в середині 1990-х років для забезпечення захищених комунікацій у мережі Інтернет. Після виявлення вразливостей у SSL був розроблений протокол TLS, який є його покращеною версією. Останньою версією є TLS 1.3, яка була випущена у 2018 році.

**2.2 Як працює TLS/SSL?**

Процес шифрування даних за допомогою TLS/SSL відбувається через кілька ключових етапів:

1. **Рукостискання (Handshake)**: Клієнт і сервер встановлюють захищений зв'язок, обмінюючись сертифікатами та ключами. Сертифікат сервера зазвичай видається авторитетним центром сертифікації (CA), який підтверджує автентичність серверу.
2. **Шифрування даних**: Після встановлення захищеного зв'язку клієнт і сервер обмінюються даними, які зашифровані симетричними ключами. Це забезпечує, що будь-яка передача між ними не може бути легко прочитана зловмисниками.
3. **Цілісність даних**: TLS/SSL використовують механізми для перевірки цілісності даних, щоб гарантувати, що інформація не була змінена під час передачі.

**2.3 HTTP vs HTTPS**

HTTPS (HyperText Transfer Protocol Secure) — це розширення HTTP, яке використовує шифрування через TLS/SSL. Основна різниця між HTTP і HTTPS полягає в тому, що HTTPS захищає передані дані від перехоплення та модифікації, що є критично важливим для передачі конфіденційної інформації, такої як паролі або фінансові дані.

**2.4 Приклад використання TLS/SSL**

Для веб-розробників одним з кроків для захисту трафіку є встановлення TLS/SSL-сертифіката на сервер. Це можна зробити за допомогою сервісу, такого як Let’s Encrypt, який надає безкоштовні сертифікати та дозволяє автоматизувати їх оновлення.

**3. Аутентифікація користувачів**

Аутентифікація є ключовим процесом у забезпеченні того, що тільки авторизовані користувачі можуть отримати доступ до захищених ресурсів. Сучасні методи аутентифікації використовують різні механізми для забезпечення безпеки. Одними з найпопулярніших є OAuth та JWT.

**3.1 OAuth**

OAuth (Open Authorization) — це відкрита технологія для надання обмеженого доступу до ресурсів без необхідності передачі паролів між сервісами. Вона дозволяє користувачам надати третім сторонам доступ до своїх ресурсів на одному сервісі (наприклад, Google або Facebook), не надаючи свої паролі.

**3.1.1 Як працює OAuth?**

1. **Авторизаційний запит**: Користувач відправляє запит на авторизацію до стороннього сервісу (наприклад, Google), використовуючи свій обліковий запис.
2. **Код авторизації**: Сервіс генерує код авторизації та повертає його сторонній програмі.
3. **Отримання токену доступу**: За допомогою коду авторизації стороння програма запитує токен доступу.
4. **Доступ до ресурсів**: Використовуючи токен доступу, стороння програма отримує доступ до захищених ресурсів.

**3.2 JWT (JSON Web Token)**

JWT — це стандарт для безпечної передачі інформації між сторонами у вигляді JSON-об'єктів. JWT широко використовується для аутентифікації, особливо в сучасних веб-додатках, що працюють за принципом REST API.

**3.2.1 Структура JWT**

JWT складається з трьох частин:

1. **Header (заголовок)**: містить інформацію про тип токена та алгоритм шифрування.
2. **Payload (дані)**: містить основну інформацію (наприклад, ідентифікатор користувача або час дії токена).
3. **Signature (підпис)**: це хешована комбінація заголовка, даних та секретного ключа. Підпис використовується для перевірки цілісності токена.

**3.2.2 Як працює JWT?**

JWT використовується для передачі даних між клієнтом і сервером у вигляді зашифрованих токенів. Коли користувач проходить аутентифікацію, сервер створює JWT та відправляє його клієнту. Після цього клієнт включає токен у заголовок запиту, щоб отримати доступ до захищених ресурсів.

**3.3 Приклад використання OAuth та JWT**

У багатьох сучасних веб-додатках використовуються обидві технології для забезпечення захисту користувачів. Наприклад, авторизація через Google (OAuth) може видавати JWT для подальшого використання в додатку. Це дозволяє зберігати авторизаційні дані на клієнтській стороні та легко передавати їх на сервер без ризику витоку паролів.

**Висновок**

Безпека клієнт-серверної взаємодії є багатогранним процесом, що включає в себе шифрування даних, запобігання атакам, а також забезпечення належної аутентифікації користувачів. Протоколи шифрування, такі як TLS/SSL, забезпечують захищену передачу даних, тоді як сучасні методи аутентифікації, такі як OAuth та JWT, дозволяють безпечно взаємодіяти з сервісами та ресурсами в Інтернеті.