Autenticació i seguretat PHP

Introducció

Context i importància

Per què és important la seguretat en aplicacions web?

- Les aplicacions web són vulnerables a múltiples tipus d'atacs: robatori de dades, manipulació d'informació i accés no autoritzat.

- La seguretat és essencial per protegir:
  - Les dades dels usuaris: com contrasenyes, dades personals, etc.
  - La reputació de l'organització: un atac pot danyar la confiança dels usuaris.
  - El funcionament del sistema: un accés no autoritzat pot interrompre o destruir els serveis.

Introducció

Objectius de la sessió

- Comprendre els conceptes bàsics:
  - Què són l'autenticació i l'autorització, i com és diferencien.
  - Com gestionar usuaris, perfils i rols per organitzar l'accés a recursos.

- Aprendre tècniques bàsiques de seguretat:
  - Implementació d'un sistema d'autenticació senzill amb PHP.
  - Hashing de contrasenyes per emmagatzemar credencials de forma segura.

- Reconèixer bones pràctiques:
  - Mètodes per protegir aplicacions contra atacs comuns.

Introducció

Conceptes clau

- Autenticació:
  - Primer pas per accedir a un sistema, on és confirma que l'usuari és qui diu ser.
  - Procés de verificació de la identitat d'un usuari.
    - ➢ **Exemple:** Introduir nom d'usuari i contrasenya en un formulari d'inici de sessió, verificar un codi enviat al mòbil, un certificat digital…
    - ➢ **Conseqüència:** Si l'autenticació falla, l'usuari no pot accedir.

- Autorització:
  - Procés que determina si un usuari té permisos per accedir a un recurs específic per realitzar una acció concreta.
    - ➢ **Exemple:** Només un admin pot modificar configuracions, la resta no poden accedir al seu perfil, però no pot editar la configuració global del sistema…
    - ➢ **Conseqüència:** Si l'autorització falla l'usuari veu un missatge d'error del tipus: "No tens permisos per accedir."

Introducció

Conceptes clau

- Rols:
  - Un conjunt de permisos que es defineixen per agrupar els usuaris amb funcions similars dins d'un sistema.

  - **Exemples:**
    - ➢ Admin: Gestió d'usuaris, dades sensibles, modificar configuracions globals…
    - ➢ Editor: Creació, edició de contingut, NO gestiona usuaris ni configuracions.
    - ➢ Usuari: Només accés al seu perfil personal i utilitzar funcionalitats bàsiques.

  - **Propòsit:** Simplificar la gestió de permisos assignant-los a grups d'usuaris en lloc d'assignar permisos individualment.

Introducció

Conceptes clau

- Perfils:
  - Informació personalitzada associada a un usuari específic, que defineix les seves preferències, configuracions, i historial dins del sistema.

  - **Exemple:**
    - ➢ Configurar l'idioma preferit o el tema de color en una aplicació.
    - ➢ Veure l'historial de comandes en una botiga en línia.
    - ➢ Desar configuracions personalitzades, com ara notificacions activades o desactivada.

  - **Propòsit:**
    - ➢ Fer que el sistema sigui més personal i adaptat a les necessitats específiques de cada usuari.

Autenticació d’usuaris

Sistema d’autenticació

1. Flux de l'aplicació

- Formulari d'inici de sessió:
  - L'usuari introdueix el seu nom d'usuari i contrasenya en un formulari.

- Comprovació de les credencials a la base de dades:
  - És valida que el nom d'usuari existeix.
  - Es comprova que la contrasenya introduïda coincideix amb la base de dades (utilitzant password_verify per a contrasenyes amb hashing).

- Establiment de la sessió :
  - Si les credencials són correctes, s'inicia una sessió per mantenir l'estat de l'usuari autenticat.
  - Les dades de l'usuari es guarden a la sessió per utilitzar-les en altres pàgines.

Sistema d’autenticació

2. Formulari d’inici de sessió (index.html)

3. Comprovació de les credencials (login.php)

- Connexió a la base de dades:
  - Connecta't a la base de dades per obtenir les credencials de l'usuari.

- Validació:
  - Comprova que l'usuari existeix i que la contrasenya és correcta.

- Establiment de sessió:
  - Guarda informació de l'usuari autenticat en una sessió PHP.

Sistema d’assignació de rols

Taula SQL per gestionar usuaris

```
CREATE DATABASE sistema_usuaris;
USE sistema_usuaris;

CREATE TABLE usuaris (

    id INT AUTO_INCREMENT PRIMARY KEY,
    nom_usuari VARCHAR(50) UNIQUE NOT NULL,
    contrasenya VARCHAR(255) NOT NULL,
    rol ENUM('Administrador', 'Editor', 'Visitant') NOT NULL
);

INSERT INTO usuaris (nom_usuari, contrasenya, rol) VALUES
('admin', 'admin123', 'Administrador'),
('editor', 'editor123', 'Editor'),
('visitant', 'visitant123', 'Visitant');
```

Sistema d’assignació de rols

Codi per gestionar rols i mostrar contingut diferent segons el nivell d'accés:

```
session_start();

// Suposem que el rol es guarda a la sessió

$rol = $_SESSION['rol'];
if ($rol === 'administrador') {
    echo "Benvingut, Administrador. Pots gestionar usuaris.";
} elseif ($rol === 'editor') {
    echo "Benvingut, Editor. Pots editar continguts.";
} else {
    echo "Benvingut, Visitant. Pots veure continguts públics.";
}
```

Sistema d’assignació de rols

Bones pràctiques de gestió d'usuaris, rols i perfils:

1. Xifrar contrasenyes
    - La seguretat de les contrasenyes és essencial per protegir el sistema davant de possibles intrusions. Mai ha d'emmagatzemar contrasenyes en text pla.
2. Assignació de rols per defecte
    - És important limitar els permisos d'un usuari nou fins que s'hagin validat i assignat els permisos necessaris.

Seguretat web

Amenaces comunes en seguretat web

1. Injecció SQL

- És un atac que explota la manca de validació d'entrades d'usuari per injectar codi SQL maliciós en consultes de bases de dades.

- Objectiu: Accedir, manipular o esborrar dades no autoritzades.

- Exemple teòric (consulta SQL insegura):

```
$usuari = $_POST['usuari'];
$query = "SELECT * FROM usuaris WHERE nom_usuari = '$usuari'";
```

Si un atacant introdueix admin' OR '1'='1, la consulta es converteix en:

```
SELECT * FROM usuaris WHERE nom_usuari = 'admin' OR '1'='1';
        //retorna tots els usuaris, comprometent la base de dades.
```

Amenaces comunes en seguretat web

2. XSS (Cross-Site Scripting)

- Consisteix a injectar codi maliciós que s'executa al navegador d'un altre usuari.
- Objectiu: Robar cookies, redirigir a pàgines malicioses o executar accions en nom de l'usuari.
- Exemple teòric (Permetre entrades sense netejar):

```
echo "Comentari: " . $_POST['comentari'];
```

Un atacant pot introduir:

```
<script>alert('Atac XSS!')</script>
```

- Exemple pràctic (solució):

```
$comentari = htmlspecialchars($_POST['comentari']); echo "Comentari: " . $comentari;
```

Amenaces comunes en seguretat web

3. Man-in-the-Middle (MITM)

- Un atacant intercepta la comunicació entre un client i un servidor, permet llegir o modificar dades sensibles.
- Objectiu: Capturar credencials, dades bancàries o altra informació confidencial.
- Exemple de pràctica: Sense HTTPS, la informació viatja en text pla i pot ser capturada.
- Solució: Utilitzar HTTPS per xifrar la comunicació.

  Exemple de configuració d'Apache per forçar HTTPS:

```
<VirtualHost *:80>
    ServerName exemple.com
    Redirect / https://exemple.com/
</VirtualHost>
```

Amenaces comunes en seguretat web

4. Atacs per força bruta

- Consisteix a provar múltiples combinacions de noms d'usuari i contrasenyes fins a trobar-ne una vàlida.
- Objectiu: Accedir sense autorització a un compte.
- Exemple d’atac: Accedir a comptes amb contrasenyes febles.
- Prevenció: Limitar intents de connexió i implementar contrasenyes segures.

Amenaces comunes en seguretat web

4. Atacs per força bruta

- Exemple pràctic (prevenció): Implementar un sistema de bloqueig temporal.

```
$usuari = $_POST['usuari'];
$errors = $_SESSION['errors'][$usuari] ?? 0;
if ($errors >= 3) {
    echo "Compte bloquejat temporalment.";
} else {
    // Comprovar credencials
if ($contrasenya_correcta) {
        $_SESSION['errors'][$usuari] = 0; // Reiniciar errors
   } else {
        $_SESSION['errors'][$usuari] = $errors + 1;
echo "Contrasenya incorrecta.";}
}
```

Amenaces comunes en seguretat web

5. Bones pràctiques

- Validar i netejar totes les entrades de l'usuari:
  - Ús de filtres PHP: Exemple amb filter_var() per validar correus electrònics:

```
$email = $_POST['email'];
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email no vàlid.";
}
```

Hashing de contrasenyes

1. Per què és important el hash de contrasenyes?

- Risc emmagatzemar contrasenyes en text pla:
  - Emmagatzemar contrasenyes en text pla posa en risc la seguretat dels usuaris si la base de dades és compromesa.
  - Les contrasenyes són fàcilment llegibles i reutilitzables en altres sistemes si no estan protegides.

- Seguretat amb hashing:
  - El hashing converteix la contrasenya en una cadena única i irreversible, fent-la molt més segura en cas de filtració.
  - Un bon hashing utilitza tècniques com salting (afegir dades úniques a cada contrasenya) i funcions que són resistents a atacs.

Hashing de contrasenyes

2. Mètodes segurs amb PHP

PHP ofereix funcions integrades per gestionar el hashing de contrasenyes fàcilment:

- password_hash():
  - Genera un hash segur per una contrasenya.
  - Utilitza salts automàtics per prevenir atacs de força bruta.
  - Els salts fan que cada contrasenya sigui única, fins i tot si diverses persones utilitzen la mateixa contrasenya.

- password_verify():
  - Comprova si una contrasenya introduïda coincideix amb el hash emmagatzemat.

Hashing de contrasenyes

3. Exemple pràctic sense hashing (mala pràctica)

```
// Emmagatzemant contrasenya en text pla (insegur!)
$contrasenya = 'contrasenya123';

// Simulem l'emmagatzematge en una base de dades:
$db_contrasenya = $contrasenya;


// Verifiquem la contrasenya:
if ($contrasenya_input === $db_contrasenya) {
    echo "Accés concedit";
} else {
    echo "Contrasenya incorrecta";
}
```

Problema: Si un atacant accedeix a la base de dades, pot veure directament totes les contrasenyes.

Hashing de contrasenyes

4. Exemple pràctic amb hashing (bona pràctica)

- Emmagatzemar una contrasenya amb ***password_hash():***

```
// Creem un hash segur per a la contrasenya
$contrasenya = 'contrasenya123';
$hashed_password = password_hash($contrasenya, PASSWORD_DEFAULT);

// Emmagatzemem el hash (no el text pla) a la base de dades:
echo "Hash generat: " . $hashed_password;
```

- Exemple de hash generat:

    $2y$10$Ehc5yLtIwO7R9U7fL8... (cadena llarga i segura)

Hashing de contrasenyes

4. Exemple pràctic amb hashing (bona pràctica)

- Comprovar una contrasenya amb ***password_verify():***

```
// Entrada de l'usuari
$contrasenya_input = 'contrasenya123';

// Verifiquem la contrasenya
if (password_verify($contrasenya_input, $hashed_password)) {
    echo "Accés concedit";
} else {
    echo "Contrasenya incorrecta";
}
```

- **Avantatge:** Encara que un atacant accedeixi al hash emmagatzemat, no pot recuperar la contrasenya original (el hashing és irreversible).

Hashing de contrasenyes

5. Bones pràctiques de gestió de contrasenyes

- Actualitza contrasenyes si utilitzes mètodes antics
  - Algoritmes antics com MD5 o SHA-1 són vulnerables a atacs avançats.
  - Si la teva base de dades conté contrasenyes hashades amb aquests mètodes, haurien d'actualitzar-se immediatament.

- Estratègia d'actualització:
  - Quan un usuari introdueix la seva contrasenya, pots actualitzar automàticament el hashing si detectes que és un mètode antic.

- Aquest procés és transparent per a l'usuari i reforça la seguretat de la base de dades.

Hashing de contrasenyes

5. Bones pràctiques de gestió de contrasenyes

- Exemple:

```
// Verificar contrasenya amb un mètode antic
if (hash('md5', $contrasenya_input) === $hashed_password_emmagatzemat) {
    echo "Contrasenya correcta. Actualitzant el hash.";

// Actualitzar amb un hash més segur
    $nou_hash = password_hash($contrasenya_input, PASSWORD_DEFAULT);

// Guardar el nou hash a la base de dades
    $stmt = $conn->prepare("UPDATE usuaris SET contrasenya = ? WHERE nom_usuari
= ?");
    $stmt->bind_param("ss", $nou_hash, $nom_usuari);
    $stmt->execute();
}
```