KONTROLA DOSTĘPU

UWIERZYTELNIANIE I AUTORYZACJA
REJESTRACJA UŻYTKOWNIKÓW
PRZECHOWYWANIE HASEŁ

Waldemar Korłub

Wytwarzanie Aplikacji Internetowych
KASK ETI Politechnika Gdańska

Kontrola dostępu

- W aplikacjach internetowych typowo występuje potrzeba rozróżnienia dostępnych funkcji i zasobów w zależności od użytkownika, np.:
  - Administrator sklepu może edytować produkty
  - Zalogowany użytkownik może składać zamówienia
  - Niezalogowany użytkownik może tylko przeglądać ofertę i dodawać produkty do koszyka
- Na kontrolę dostępu składają się:
  - Uwierzytelnianie – weryfikacja tożsamości użytkownika
  - Autoryzacja – nadanie uprawnień

Uwierzytelnianie

- Weryfikacja tożsamości użytkownika
- Polega na sprawdzeniu informacji, którą z dużym prawdopodobieństwem zna tylko dany użytkownik:
  - Hasło
  - Token dostępowy wysłany w wiadomości SMS
  - Treść wygenerowana z użyciem klucza prywatnego (kryptografia asymetryczna)
- Uwierzytelnianie dwuetapowe/wieloetapowe
  - Wykorzystanie więcej niż jednej informacji
  - np. potwierdzanie przelewów w banku internetowym

Autoryzacja

- Nadanie uprawnień do wykonywania operacji i dostępu do zasobów
- Najczęściej w oparciu o ustaloną wcześniej tożsamość użytkownika
- Wykorzystuje często metadane przypisane użytkownikowi, np.:
  - Przynależność do grup
  - Przypisane role
  - Nadane uprawnienia

Użytkownicy

- Typowo kontrola dostępu opiera się o bazę użytkowników aplikacji
- Baza musi zawierać informacje umożliwiające uwierzytelnienie użytkownika, np.:
  - Hasło lub jego skrót (hash)
  - Numer telefonu (token dostępowy z wiadomości SMS)
  - Klucz publiczny (kryptografia asymetryczna)
- Najczęściej spotykany sposób uwierzytelniania użytkowników wykorzystuje parę: login + hasło

Hasła użytkowników

- W idealnym świecie użytkownicy aplikacji…
  - …wybierają długie hasła
  - …nie stosują wyrażeń słownikowych
  - …używają dużych i małych liter, cyfr i innych znaków
  - …nigdy nie zapisują haseł na kartkach
    - np. pod klawiaturą, w portfelu, przyklejone do monitora
  - …nigdy nie używają tego samego hasła w różnych serwisach
  - …regularnie zmieniają swoje hasła

Hasła użytkowników

- W rzeczywistym świecie nie każdy użytkownik trzyma się powyższych zasad…
- …a wymuszanie zbyt restrykcyjnej polityki haseł może przynieść skutki odwrotne do oczekiwanych

- Autorzy aplikacji i administratorzy powinni kompensować błędy swoich użytkowników
  - Albo przynajmniej się starać

Przechowywanie haseł użytkowników

- Nie należy przechowywać haseł w postaci jawnej (ang. *plaintext*)
  - Włamanie do bazy → natychmiastowy dostęp do kont użytkowników
  - Szczególnie groźne dla użytkowników wykorzystujących to samo hasło w wielu serwisach
- Należy wykorzystywać funkcje hashujące (mieszające)
  - Skrót hasła nie powinien umożliwiać łatwego odtworzenia oryginalnego hasła, np. dla hasła p@ssw0rd:
  - $2y$10$8Bc9/72wX4TRsMoZcCyF/.7 fOs5gteQf4ioarhkzGcikb2iFT03/C

Przechowywanie haseł użytkowników

- Wiele funkcji, niegdyś uznawanych za bezpieczne, obecnie jest łatwa do złamania
  - Rosnąca moc obliczeniowa komputerów ułatwia ataki siłowe (ang. *brute-force*) na funkcje skrótu
  - Odkrywane są nowe słabości istniejących funkcji
- **Nie należy stosować funkcji MD5**
  - Niezalecana od 1999 roku (rekomendacja NIST)
  - Podatna na kolizje
- Podobnie nie zaleca się algorytmu SHA-1

Przechowywanie haseł użytkowników

- Od PHP 5.5 dostępne są funkcje pomocnicze:
  - **password_hash()** – hashowanie
    - Algorytm bcrypt bazujący na szyfrze Blowfish

```php
$hash = password_hash('p@ssw0rd', PASSWORD_DEFAULT);
```

  - **password_verify()** – weryfikacja hasła:

```php
if (password_verify('p@ssw0rd', $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
}
```

Rejestracja użytkowników

1. Formularz rejestracji:

```
<input type="text" name="login" .../>
<input type="password" name="pass" .../>
<input type="password" name="repeat_pass" .../>
<!-- ... -->
```

2. Pobranie danych z zapytania POST:

```
$password = $_POST['pass'];
$repeat_password = $_POST['repeat_pass'];
//...
```

3. Weryfikacja:

```
if ($password === $repeat_password){ //...
```

Rejestracja użytkowników

4. Zahashowanie hasła:

```
$hash = password_hash($password, PASSWORD_DEFAULT);
```

5. Zapis do bazy danych:

```
$db->users->insert([
    'login' => $login,
    'password' => $hash
    //...
]);
```

6. Redirect:

```
header('Location: success.php');
exit;
```

Logowanie

1. Formularz logowania:

```
<input type="text" name="login" .../>
<input type="password" name="pass" .../>
```

2. Pobranie danych z zapytania POST:

```
$password = $_POST['pass'];
$login = $_POST['login'];
```

3. Pobranie użytkownika z bazy danych

```
$user = $db->users->findOne(['login' => $login]);
```

Logowanie

4. Weryfikacja hasła:

```
if($user !== null &&
  password_verify($password, $user['password'])){
   //hasło poprawne
}
```

Hasło z formularza

Hash hasła z bazy danych

5. Zmiana id sesji i zapisanie informacji o użytkowniku:

```
session_regenerate_id();
$_SESSION['user_id']  = $user['_id'];
```

6. Przekierowanie:

```
header('Location:  profile.php');
exit;
```

Wylogowanie

- Wyczyszczenie danych bieżącej sesji:
  - Programistyczne:
    session_destroy(); //usuwa stan sesji

    //następnie usunięcie cookies!
  - W wyniku wygaśnięcia sesji po zadanym czasie nieaktywności
  - *Uwaga:* $_SESSION[] wciąż zawiera stan sesji wczytany przy session_start() na początku skryptu; możliwe jest dalsze operowanie na tych wartościach chyba że je usuniesz
    - `session_unset()`
    - `$_SESSION = []`

Pytania?