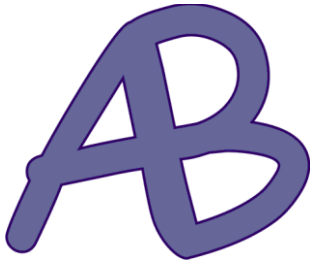


PHP

- SESJE

dr Artur Bartoszewski
UTH Radom

Sesje

Mechanizm sesji umożliwia przekazywanie parametrów między stronami.

- Zmienne są przechowywane po stronie serwera.
- Po stronie klienta przechowywane jest tylko ID sesji.
- ID jest zwykle zapisane w pliku cookie.

ID może być także przekazane przez URL - PHP jest w stanie sam rozpoznać czy na komputerze klienta włączony jest mechanizm cookies i w razie potrzeby dodać identyfikator sesji do każdego URLu i formularza. Wymaga to jednak posiadania PHP skompilowanego z opcją --enable-trans-sid.

Po otrzymaniu żądania klienta PHP sprawdza, czy przypisano już ID sesji. Jeśli tak, to PHP odczytuje zmienne zarejestrowane w tej sesji. Jeśli nie, generowany jest nowy, unikalny identyfikator sesji.

```php
<?php
  session_start();
?>
```

Mechanizm sesji może być uruchomiony automatycznie (jeśli w konfiguracji PHP włączona została opcja auto_start) lub „ręcznie” przez programistę (za pomocą funkcji session_start())

Sesje

Jedna z tablic super globalnych: $_SESSION, przechowuje zmienne zarejestrowane w sesji.
Kluczem jest nazwa zarejestrowanej zmiennej.

Aby PHP zaczęło śledzić wartość zmiennej w sesji, najpierw trzeba ją zarejestrować.
W tym celu wpisujemy zmienną bezpośrednio do tablicy $_SESSION;

```php
<?php
  $_SESSION['nazwa_zmiennej'] = "Wartość zmiennej";
?>
```

Sesje – usuwanie sesji

Aby usunąć wszystkie globalne zmienne sesji i zniszczyć sesję, używamy metody session_destroy()

```php
<?php
  session_destroy();
?>
```

Sesje

Przykład:
Skrypt, który informuje, ile razy strona była odwiedzana w czasie sesji

```php
<?php
session_start();
if (!isset($_SESSION['licznik']))
    $_SESSION['licznik'] = 1;
else
    $_SESSION['licznik']++;
?>
<!doctype html>
<html>
<head>
    <meta charset="UTF-8" />
    <title></title>
</head>

<body>
    <?php
    echo "<p>Odwiedziłeś stronę: " .
$_SESSION['licznik'] . " razy";
    ?>
</body>

</html>
```

Sesje

Przykład: Mechanizm logowania (na razie bez sprawdzania hasła- wystarczy wpisać cokolwiek)

```php
<?php
session_start();
if (
    isset($_POST['login']) &&
    isset($_POST['pass']) &&
    !empty($_POST['login']) &&
    !empty($_POST['pass'])
) {
    $_SESSION['user'] = $_POST['login'];
}
?>
<!doctype html>
<html>

<head>
    <meta charset="UTF-8" />
    <title></title>
</head>
```

```php
<body>
    <form method="POST">
        <label>Login:</label>
        <input type="text" name="login" />
        <label>Pass:</label>
        <input type="password" name="pass" />
        <input type="submit" name="Submit"
value="Zaloguj" />
    </form>
    <br>
    <a href="podstrona.php">Link do podstrony</a>
    <br>
    <?php
    if (isset($_SESSION['user'])) {
        echo "Witaj " . $_SESSION['user'];
    } else
        echo "Niezalogowano";
    ?>
</body>
</html>
```

Sesje – mechanizm logowania

Mechanizm logowania

- W momencie pierwszego trafienia na stronę interpreter tworzy specjalny, losowy oraz unikalny identyfikator przesyłany między żądaniami za pomocą ciastek lub parametru PHPSESSID doklejanego automatycznie do adresów URL.
- Na jego podstawie odczytywany jest później odpowiedni plik z danymi sesji zapisany gdzieś na serwerze.
- Pod koniec przetwarzania żądania wszystkie wprowadzone przez skrypt zmiany są z powrotem zapisywane do wspomnianego pliku tak, aby były widoczne przy wejściu na kolejną podstronę.

Bezpieczeństwo sesji

Podpięcie się do sesji przez wygenerowanie ciasteczka z losowym ID sesji nie jest oczywiście możliwe, jednak przy braku zabezpieczeń ryzykujemy przejęcie sesji.

Dwa popularne mechanizmy ataku na niezabezpieczoną sesję to:

- Session Fixation
- Session Hijacking.

Bezpieczeństwo sesji

Session Fixation – polega na „podsunięciu” serwerowi PHP własnego ID. Do użytkownika wysyłana jest iadomość z prośbą o zalogowanie się przez link.
Link zawiera URL zawierający ciąg ?PHPSESSID=######, gdzie „#####" jest podsuniętym identyfikatorem sesji. Użytkownik loguje się do systemu, a atakujący generuje dla siebie ciasteczko sesyjne z podsuniętym ID.

Zabezpieczeniem przed tego rodzaju atakiem jest zresetowanie ID (wygenerowanie nowego ID sesji) już po zalogowaniu.

```php
if (!isset($_SESSION[,zainicjowana']))
{
    session_regenerate_id();
    $_SESSION[,zainicjowana'] = true;
}
```

Bezpieczeństwo sesji

Session Hijacking – jest możliwy, gdy przechwycony zostanie link do jakiegoś zasobu wysłany przez użytkownika który ma wyłączone ciastka. W linku znajdzie się wtedy jego identyfikator sesji (patrz slajd 3).

Najprostszym zabezpieczeniem jest sprawdzenie, czy użytkownik loguje się z tego samego IP na jakiem wygenerowany został identyfikator sesji.

```php
session_start();
if(!isset($_SESSION['ip']))
{
    $_SESSION['ip'] = $_SERVER['REMOTE_ADDR'];
}
```

Pierwszym co robimy po otwarciu nowej sesji jest sprawdzenie zmiennej „ip”, jeżeli nie istnieje zapamiętujemy w nim IP komputera

Przed wykonaniem akcji sprawdzamy, czy aktualny adres jest zgodny z zapamiętanym

```php
if ($_SESSION['ip'] != $_SERVER['REMOTE_ADDR’])
{
    die('Proba przejecia sesji udaremniona!');
}
```

Przykład - logowanie

Przykład pochodzi ze strony:
https://pl.wikibooks.org/wiki/PHP/Sesje

```php
<?php
session_start();

$uzytkownicy = array(
    1 => array('login' => 'user1', 'haslo' => crc32('ppp')),
    2 => array('login' => 'user2', 'haslo' => crc32('ddd')),
    3 => array('login' => 'user3', 'haslo' => crc32('fff'))
);
```

- Na początku użyte jest polecenie session_start(), aby rozpocząć lub wznowić sesję na serwerze. Sesja jest używana do przechowywania informacji o zalogowanym użytkowniku między różnymi żądaniami HTTP.

- Zdefiniowana jest tablica asocjacyjna $uzytkownicy, która przechowuje dane o użytkownikach, takie jak login i hasło (zakodowane za pomocą funkcji crc32).

Przykład - logowanie

Przykład pochodzi ze strony:
https://pl.wikibooks.org/wiki/PHP/Sesje

```php
function czyIstnieje($login, $haslo)
{
    global $uzytkownicy;

    $haslo = crc32($haslo);

    foreach ($uzytkownicy as $id => $dane) {
        if ($dane['login'] == $login && $dane['haslo'] == $haslo) {
            // O, jest ktos taki - zwroc jego ID
            return $id;
        }
    }
    // Jeżeli doszedłeś aż tutaj, to takiego użytkownika nie ma
    return false;
} // end czyIstnieje();
```

- Funkcja sprawdza, czy podane dane logowania istnieją w bazie użytkowników.
- Porównuje podane dane login i hasło (po zakodowaniu crc32) z danymi w tablicy $uzytkownicy.
- Jeśli użytkownik istnieje, zwraca jego ID w tablicy $uzytkownicy, w przeciwnym razie zwraca false.

Przykład - logowanie

```php
if (!isset($_SESSION['uzytkownik'])) {
    // Sesja się zaczyna, wiec inicjujemy użytkownika anonimowego
    $_SESSION['uzytkownik'] = 0;
}
```

- Sprawdzane jest, czy w sesji istnieje zmienna $_SESSION['uzytkownik’].
- Jeśli nie istnieje, oznacza to, że sesja się zaczyna, więc inicjuje się użytkownika anonimowego, ustawiając $_SESSION['uzytkownik'] na 0.
- Sprawdzenie stanu zalogowania:

Przykład - logowanie

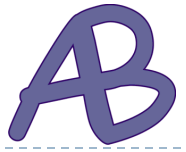
```php
if ($_SESSION['uzytkownik'] > 0) {
    // Ktos jest zalogowany
    echo 'Witaj, ' . $uzytkownicy[$_SESSION['uzytkownik']]['login'] . ' na naszej stronie!';
} else {

    // Niezalogowany
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        if (($id = czyIstnieje($_POST['login'], $_POST['haslo'])) !== false) {
            // Logujemy uzytkownika, wpisal poprawne dane
            $_SESSION['uzytkownik'] = $id;
            echo 'Dziekujemy, zostales zalogowany! <a href="index.php">Dalej</a>';
        } else {
            echo 'Podales nieprawidlowe dane, zegnaj! <a href="index.php">Dalej</a>';
        }
    } else {
        echo '<form method="post" action="index.php">
                Zaloguj sie: <input type="text" name="login"/>
                             <input type="password" name="haslo"/>
                             <input type="submit" value="OK"/></form>';
    }
}
```

Przykład - logowanie

Sprawdzenie stanu zalogowania:

- Jeśli $_SESSION['uzytkownik'] jest większe niż 0, to oznacza, że ktoś jest zalogowany, i program wyświetla odpowiednie powitanie dla zalogowanego użytkownika.
- W przeciwnym razie, jeśli żądanie jest metodą POST, sprawdzane są podane dane logowania. Jeśli są poprawne, użytkownik zostaje zalogowany, a ID użytkownika przypisane zostaje do $_SESSION['uzytkownik']. W przeciwnym razie, użytkownik otrzymuje komunikat o błędzie.
- Jeśli żądanie nie jest metodą POST, program wyświetla formularz logowania.



Literatura

W prezentacji użyto przykładów z książki:

Żygłowicz Jerzy - PHP - Kompendium wiedzy, Helion

https://www.php.net

https://pl.wikibooks.org/wiki/PHP/Sesje