MODUL VI-KEAMANAN DATA

6.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami kegunaan session pada aplikasi web.
2. Mampu melakukan konfigurasi runtime untuk membuat session.
3. Memahami kegunaan cookies pada aplikasi web.
4. Membuat cookies pada aplikasi web.

6.2 Kemanan Data

PHP menawarkan metode dalam menyimpan variabel yang mempunyai sifat global. Dengan menggunakan variabel global ini, variabel dapat dikenali dan digunakan di semua halaman web tanpa harus dideklarasi ulang. Variabel global ini disebut session dan cookie.

Penggunaan session dan cookie biasanya berupa penyimpanan informasi yang bersifat private dan informasi ini digunakan secara berulang oleh user. Contoh penerapan session dan cookies adalah penyimpanan login account, menyimpan informasi barang yang sudah masuk keranjang belanja pada aplikasi e-commerce, atau aplikasi lainnya yang biasanya membutuhkan autentikasi.

Dalam membuat sebuah website, pembuat website juga harus mengantisipasi dari kemungkinan kegiatan-kegiatan yang berusaha membobol password melalui SQL injection untuk mengamankan data.

6.2.1 Session

Session merupakan hal yang cukup penting dalam aplikasi berbasis web. Dengan session memungkinkan programmer menyimpan informasi user secara semi-permanen, artinya selama masa tertentu informasi akan tersimpan. Penyimpanan isi variabel session berada di server, jadi tidak bisa diakses secara langsung oleh *client*.

Dalam aplikasi berbasis web, session banyak digunakan sebagai autentifikasi login. Dengan session memungkinkan programmer mengatur siapa saja yang bisa mengakses suatu halaman. Misalnya saja, untuk melihat halaman kotak surat pada email, kita harus login terlebih dahulu. Dalam proses login antara lain akan terjadi pembuatan suatu session yang akan dibawa oleh user di setiap halaman.

Di halaman kotak surat, session tersebut akan diperiksa. Jika session benar maka user dipersilahkan membuka halaman kotak surat, namun jika salah maka user tidak bisa membuka halaman kotak surat dan biasanya akan diminta untuk login terlebih dahulu. Itulah sebabnya, user tidak bisa mengakses halaman kotak surat secara langsung tanpa melakukan login.

Dalam penanganan session terdapat beberapa proses yang perlu diperhatikan :

1. Proses pembuatan session
2. Proses pemeriksaan session
3. Proses penghapusan session

Berikut adalah contoh penggunaan session:
Halaman ini merupakan halaman contoh penciptaan session. Perintah session_start() harus diletakkan di perintah pertama tanpa spasi di depannya. Perintah session_start() harus ada pada setiap halaman yang berhubungan dengan session.

Session01.php

```php
<?php
    session_start();
    if (isset ($_POST['login'])) {
        $user = $_POST['user'];
        $pass = $_POST['pass'];
        //periksa login
        if ($user == "gunawan" && $pass = "123") {
            //menciptakan session
            $_SESSION['login'] = $user;
            //menuju ke halaman pemeriksaan session echo "<h1>Anda berhasil LOGIN</h1>";
            echo "<h2>Klik <a href='session02.php'>di sini (session02.php)</a>
            untuk menuju ke halaman pemeriksaan session";
        }
    } else {
    ?>
    <html>
    <head>
    <title>Login here...</title>
    </head>
    <body>
        <form action="" method="post">
        <h2>Login Here...</h2>
        Username : <input type="text" name="user"><br>
        Password : <input type="password" name="pass"><br>
        <input type="submit" name="login" value="Log In">
        </form>
        </body>
        </html>
<?php } ?>
```

Session02.php

```php
<?php
    //pemeriksaan session
    session_start();
    if (isset($_SESSION['login'])) {
        //jika sudah login
        //menampilkan isi session
        echo "<h1>Selamat Datang ". $_SESSION['login'] ."</h1>";
        echo "<h2>Halaman ini hanya bisa diakses jika Anda sudah
        login</h2>";
        echo "<h2>Klik <a href='session03.php'>di sini (session03.php)</a> untuk LOGOUT</h2>";
    } else {
        //session belum ada artinya belum login
        die ("Anda belum login! Anda tidak berhak masuk ke halaman ini.Silahkan login
        <a href='session01.php'>di sini</a>");
    }
?>
```

Session03.php

```php
<?php
    session_start();
    if (isset($_SESSION['login'])) {
        unset ($_SESSION);
        session_destroy();

        echo "<h1>Anda sudah berhasil LOGOUT</h1>";
        echo "<h2>Klik <a href='session01.php'>di sini</a> untuk LOGIN kembali</h2>";
        echo "<h2>Anda sekarang tidak bisa masuk ke halaman
        <a href='session02.php'>session02.php</a> lagi</h2>";
    }
?>
```

6.2.2 Cookies

Seperti halnya session, cookies juga merupakan sebuah konsep penyimpanan informasi user. Hanya saja, jika session tempat penyimpanan berada di server, cookies berada di client. Oleh karena itu, konsep cookies sebaiknya jangan digunakan untuk menyimpan informasi login user seperti username, password dsb. Selain user bisa melihat informasi yang disimpan, user juga bisa men-*disable* cookies itu sendiri. Jika cookies di-*disable*, maka program yang memanfaatkan cookies tentunya tidak akan berjalan dengan baik.

Cookies sendiri biasanya dipakai dalam aplikasi *shooping cart*. Biasa digunakan untuk menyimpan sementara, produk-produk yang dipilih oleh pengunjung pada saatberbelanja.

Dalam penanganan cookies juga terdapat beberapa proses yang perlu diperhatikan :

1. Proses pembuatan cookies
2. Proses pemeriksaan cookies
3. Proses penghapusan cookies

Berikut adalah contoh program penggunaan cookies:

cookie01.php

```php
<?php
    $value = 'igun';
    $value2 = 'gunawan';
    setcookie("username", $value);
    setcookie("namalengkap", $value2, time()+3600); /* expire in 1 hour */
    echo "<h1>Ini halaman pengesetan cookie</h1>";
    echo "<h2>Klik <a href='cookie02.php'>di sini</a> untuk pemeriksaan cookies</h2>";
?>
```

cookie02.php

```php
<?php
    if(isset($_COOKIE['username'])) {
    echo "<h1>Cookie 'username' ada. Isinya : " .
    $_COOKIE['username'];
    } else {
    echo "<h1>Cookie 'username' TIDAK ada.</h1>";
    }
    if(isset($_COOKIE['namalengkap'])) {
    echo "<h1>Cookie 'namalengkap' ada. Isinya : " .
    $_COOKIE['namalengkap'];
    } else {
    echo "<h1>Cookie 'namalengkap' TIDAK ada.</h1>";
    }
    echo "<h2>Klik <a href='cookie01.php'>di sini</a> untuk penciptaan cookies</h2>";
    echo "<h2>Klik <a href='cookie03.php'>di sini</a> untuk penghapusan cookies</h2>";
?>
```

cookie03.php

```php
<?php
    // set the expiration date to one hour ago
    setcookie("username", "", time() - 3600);
    setcookie("namalengkap", "", time() - 3600);
    echo "<h1>Cookie Berhasil dihapus.</h1>";
    echo "<h2>Klik <a href='cookie01.php'>di sini</a> untuk penciptaan cookies</h2>";
    echo "<h2>Klik <a href='cookie02.php'>di sini</a> untuk pemeriksaan cookies</h2>";
?>
```

6.2.3 SQL Injection

SQL injection adalah kegiatan menyisipkan perintah SQL kepada suatu statement SQL yang ada pada aplikasi yang sedang berjalan. Dengan kata lain SQL injection ini merupakan suatu tehnik pengeksploitasi pada web apilikasi yang didalamna menggunakan database untuk penyimpanan datanya.

Terjadinya SQL injection tersebut dikarenakan security atau keamanan pada level aplikasi (dalam hal ini aplikasi web) masih kurang sempurna. Kurang sempurnanya adalah pada cara aplikasi meng-handle inputan yang boleh di proses ke dalam database. Misalnya pada suatu web yang terdapat fasilitas login, terdapat dua buah inputan pada umumnya, yaitu username dan password. Jika karakter yang masuk melalui dua buah inputan tersebut tidak difilter (disaring) dengan baik maka bisa menimbulkan efek SQL injection, ini dikarenakan biasanya inputan tersebut secara sistem akan menjadi bagian dari kriteria dari suatu perintah SQL di dalam aplikasi web-nya.

Secara garis besar terjadinya SQL injection tersebut adalah sebagai berikut:

1. Tidak adanya pemfilteran terhadap karakter - karakter tanda petik satu ’ dan juga karakter double minus -- yang menyebabkan suatu aplikasi dapat disisipi dengan perintah SQL.
2. Sehingga seorang Hacker dapat menyisipkan perintah SQL kedalam suatu parameter maupun pada text area suatu form.

PHP menyediakan fungsi untuk mengatasi SQL Injection sehingga system anda diharapkan akan aman dari serangan SQL Injection

```
<?php
$username= mysql_real_escape_string($_POST['username']);
$password= mysql_real_escape_string($_POST['password']);

mysql_query("INSERT into tb_login(username,password) VALUES
('$username','$password')");
?>
```

Keterangan :
Fungsi yang kita pakai yaitu **mysql_real_escape_string** berguna saat user menginputkan sebuah data yang mengandung karakter khusus seperti ” (tanda kutip tunggal), yang menyebabkan nilai data yang dikirimkan ke database, akan dimasukkan sesuai dengan nilai sebenarnya.