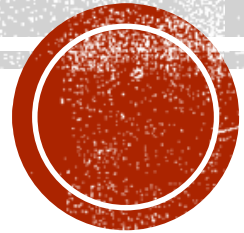
Yii2: Security

Oleh: Ahmad Syauqi Ahsan



Overview

- Pengelolaan keamanan yang baik sangat penting untuk kelangsungan hidup sebuah aplikasi.
- Untuk membuat aplikasi memiliki tingkat keamanan yang tinggi, Yii menyediakan beberapa fitur keamanan, yaitu:
  - Authentication
  - Authorization
  - Working with Password
  - Cryptography
  - Views security
  - Auth Client
  - Best Practices

Authentication

- Authentication adalah proses untuk memastikan bahwa user yang login adalah memang benar dirinya.
- Authentication biasanya dilakukan dengan menggunakan kombinasi username/email dan password.
- Pada Yii2 Advanced Template, secara default telah disertakan proses Authentication untuk user login.
- Sedangkat pada Yii2 Basic Template, proses Authentication hanya dilakukan menggunakan username dan password yang di "hardcode" pada model "User".

Proses Authentication

```
// in web.php or main.php
// ...
'components' => [
  'user' => [
    'identityClass' => 'common\models\User',
  ],
  // ...
],
```

- Pada Yii2 Advanced Template, proses Authentication melibatkan hal penting:
  - Mengatur komponen "User" pada Application Configuration.
  - Membuat (menggunakan) class "User" yang mengimplementasikan interface "**`yii\web\IdentityInterface`**".
- Proses login akan melakukan validasi password yang dimasukkan user dengan password dalam table "user" (dalam bentuk nilai hash).
- Untuk mendapatkan informasi terkait dengan user yang sedang login:
  - "id" : **`Yii::$app->user->id`**
  - "username" : **`Yii::$app->user->identity->username`**

Authentication Events

- Class **`yii\web\User`** akan memunculkan beberapa ***events*** ketika terjadi proses login dan logout.
  - **`EVENT_BEFORE_LOGIN`**: ***event*** ini akan muncul pada awal proses **`yii\web\User::login()`**. Jika metode ***event handler*** mengisi nilai property **`isValid`** dengan nilai **false**, maka proses login akan dibatalkan.
  - **`EVENT_AFTER_LOGIN`**: ***event*** ini akan muncul setelah proses login berhasil.
  - **`EVENT_BEFORE_LOGOUT`**: ***event*** ini akan muncul pada awal proses **`yii\web\User::logout()`**. Jika metode ***event handler*** mengisi nilai property **`isValid`** dengan nilai **false**, maka proses logout akan dibatalkan.
  - **`EVENT_AFTER_LOGOUT`**: event ini akan muncul setelah proses logout berhasil.
- Misal: anda ingin melakukan audit (dan/atau pengumpulan data statistic) untuk setiap user yang login.
  - Anda dapat membuat sebuah ***event handler*** untuk ***event*** **`EVENT_AFTER_LOGIN`**, dan meletakkan proses pencatatan waktu login, alamat ip, dll didalamnya.

Authorization

- Authorization merupakan proses untuk memverifikasi apakah seorang user berhak untuk melakukan sesuatu.
- Pada Yii2, authorization melibatkan komponen:
  - Access Control Filter
  - Role Based Access Control (RBAC)
- Kedua komponen diatas dapat berdiri sendiri, dan juga bisa digabungkan untuk mendapatkan kemampuan yang lebih lengkap.

Access Control Filter

- Access Control Filter (ACF) merupakan metode sederhana untuk proses authorization yang diimplementasikan sebagai **`yii\filters\AccessControl`**.
- ACF biasa digunakan pada controller untuk mengatur hak akses terhadap action-action (route) yang ada didalamnya.
- Perhatikan script disamping:
  - Property '**only**' berisi action apa saja yang akan diatur hak aksesnya.
  - Property '**rules**' berisi aturan yang akan diterapkan.
  - '**?**' merupakan role spesial yang berarti semua user yang belum login (guest).
  - '@' merupakan role spesial yang berarti semua user yang telah login.
  - Aturan-aturan tersebut akan dievaluasi dari atas kebawah.

```
use yii\web\Controller;
use yii\filters\AccessControl;

class SiteController extends Controller
{
  public function behaviors()
  {
    return [
      'access' => [
        'class' => AccessControl::className(),
        'only' => ['login', 'logout', 'signup'],
        'rules' => [
          [
            'allow' => true,
            'actions' => ['login', 'signup'],
            'roles' => ['?'],
          ],
          [
            'allow' => true,
            'actions' => ['logout'],
            'roles' => ['@'],
          ],
        ],
      ],
    ];
  }
}
```

Role Based Access Control

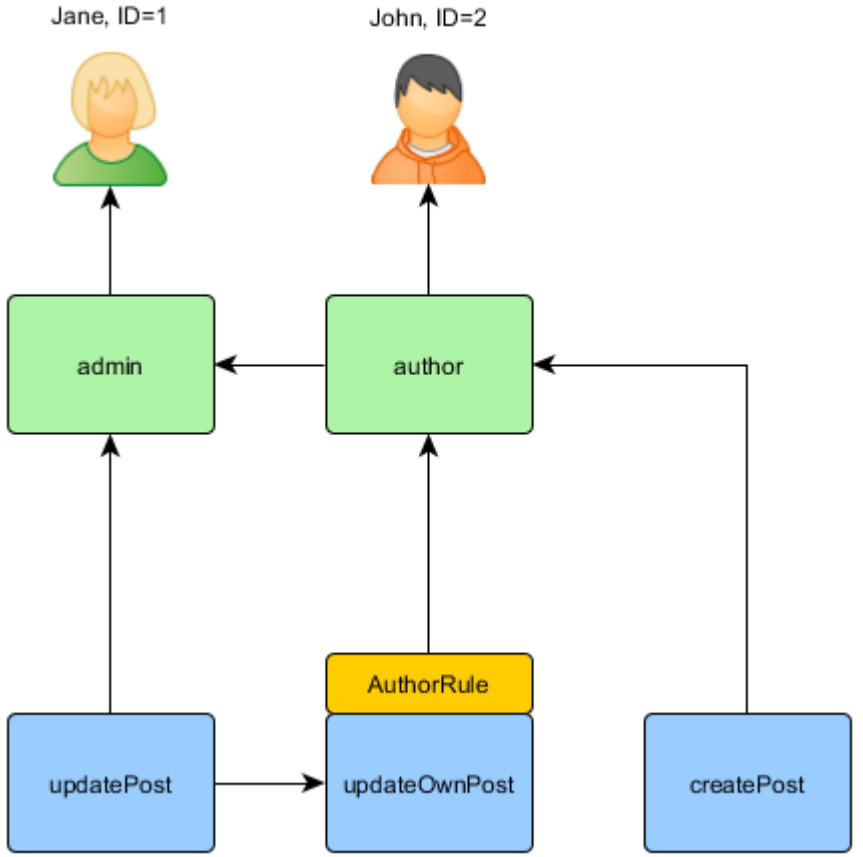
- Role Based Access Control (RBAC) dapat digunakan untuk pengelolaan hak akses secara lebih komprehensif.
- Yii2 mengimplementasikan RBAC melalui komponen dari applikasi yang bernama "**authManager**".
- Untuk menggunakan RBAC ada dua hal yang perlu dilakukan:
  - Membangun data authorization untuk RBAC.
  - Menggunakan RBAC untuk melakukan pengecekan hak akses (dapat dilakukan di Model, View, maupun Controller).

Konsep Dasar RBAC

- Komponen utama dari RBAC ada 3 yaitu:
  - Role: digunakan untuk menentukan "peran" dari seorang user (contoh: *admin*, *author*, dll). Role dapat memiliki beberapa permission, dan juga role yang lain. Seorang user dapat di-assign satu role atau lebih.
  - Permission: merupakan hak untuk melakukan sesuatu (contoh: *create post*, *update post*, dll).
  - Rule: merupakan script yang akan dijalankan untuk memastikan sebuah role atau permission dapat digunakan oleh user.
    Contoh: permission ***update post*** dapat memiliki rule yang akan mengecek apakah user tersebut adalah user pembuat ***post*** yang akan di-update.
- Jika menggunakan **DbManager**, terdapat 4 table yang digunakan untuk menyimpan data RBAC:
  - "***auth_item***": untuk menyimpan daftar item (role, permission, dan rule).
  - "***auth_item_child***": untuk menyimpan data hubungan antar item.
  - "***auth_assignment***": untuk menyimpan data hubungan antara user dengan role.
  - "***auth_rule***": untuk menyimpan rule.

Role - Permission - Rule

- Pada gambar disamping terdapat dua user (Jane dan John), dua role (admin dan author), tiga permission (createPost, updatePost, dan updateOwnPost), serta satu rule (AuthorRule).
- User Jane memiliki role "admin" sedangkan user John memiliki role "author"
- Role "admin" memiliki permission "updatePost" dan role "author"
  - → semua yang dimiliki oleh role "author" juga akan dimiliki oleh role "admin".
- Role "author" memiliki permission "createPost" dan "updateOwnPost".
- Permission "updateOwnPost" dikenakan rule "AuthorRule"
  - → permission "updateOwnPost hanya dapat digunakan jika rule "AuthorRule" bernilai true.



Bekerja dengan Password

```
$hash = Yii::$app->getSecurity()
  ->generatePasswordHash($password);

if (Yii::$app->getSecurity()
  ->validatePassword($password, $hash)) {
    // all good, logging user in
} else {
    // wrong password
}
```

- Semua developer yang baik pasti tahu bahwa password tidak boleh disimpan sebagai teks.
- Banyak developer masih percaya bahwa dengan menggunakan algoritma hashing "md5" atau "sha1" sudah cukup aman.
- Namun, dengan teknologi terbaru, password yang disimpan menggunakan algoritma tersebut **ada kemungkinan** untuk dapat diketahui dengan brute force attack. Serangan ini juga banyak didukung dengan adanya kamus data yang sangat besar yang menyimpan password "md5".
- Untuk meningkatkan keamanan, terdapat algoritma hashing yang lebih kuat terhadap serangan brute force attack yaitu "bcrypt".
- Fungsi "crypt" pada PHP dapat digunakan untuk mengenerate nilai hash menggunakan algoritma "bcrypt".
- Yii menyediakan helper `Yii::$app->getSecurity()` yang dapat digunakan untuk bekerja dengan password.

Cryptography

- Men-generate Data Random
  - Untuk proses me-reset password via email, biasanya dibutuhkan token yang disimpan dalam database dan juga dikirimkan ke user.
- Enkripsi dan Dekripsi
  - Jika anda ingin menyimpan data secara aman, sehingga orang yang dapat membuka database tetap tidak dapat membacanya → data tersebut perlu dienkripsi sebelum disimpan.
- Mengkonfirmasi integritas data.
  - Pada kondisi tertentu, ada kemungkinan anda ingin memverifikasi sebuah data telah diubah oleh pihak lain atau tidak.

```php
// GENERATE RANDOM DATA
$key = Yii::$app->getSecurity()
  ->generateRandomString();


// ENCRYPTION AND DECRYPTION
// $data and $secretKey are obtained from the form
$encryptedData = Yii::$app->getSecurity()
  ->encryptByPassword($data, $secretKey);
// store $encryptedData to database

// $secretKey is obtained from user input,
// $encryptedData is from the database
$data = Yii::$app->getSecurity()
  ->decryptByPassword($encryptedData, $secretKey);


// GENERATE RANDOM DATA
// $secretKey our application or user secret,
// $genuineData obtained from a reliable source
$data = Yii::$app->getSecurity()
  ->hashData($genuineData, $secretKey);

// $secretKey our application or user secret,
// $data obtained from an unreliable source
$data = Yii::$app->getSecurity()
->validateData($data, $secretKey);
```

Best Practices

- Basic principles (filter input & escape output)
- Avoiding SQL injections
- Avoiding XSS (Cross-Site Scripting)
- Avoiding CSRF (Cross-Site Request Forgery)
- Avoiding file exposure
- Avoiding debug info and tools at production
- Using secure connection over TLS
- Secure Server configuration

Latihan

- Pastikan aplikasi yang anda buat telah menerapkan konsep-konsep keamanan!

15

Terima Kasih