INTER-Mediatorが備える
セキュリティ機能

2019/08/24
INTER-Mediator《大》勉強会 2019

松尾篤（株式会社エミック）

株式会社エミック

Agenda

- Webアプリで見つかりやすい脆弱性
- INTER-Mediatorのセキュリティ機能
- INTER-Mediator Training Course

Webアプリで

見つかりやすい脆弱性

Webアプリの脆弱性を知る

- 安全なウェブサイトの作り方を参照

  https://www.ipa.go.jp/security/vuln/websecurity.html

  （IPA 独立行政法人 情報処理推進機構）

見つかりやすい脆弱性

- SQLインジェクション
- OSコマンド・インジェクション
- ディレクトリ・トラバーサル
- セッション管理の不備

見つかりやすい脆弱性

- クロスサイト・スクリプティング（XSS）
- クロスサイト・リクエスト・フォージェリ（CSRF）
- HTTPヘッダ・インジェクション

見つかりやすい脆弱性

- メールヘッダ・インジェクション
- クリックジャッキング
- バッファオーバーフロー
- アクセス制御や認可制御の欠落

INTER-Mediatorの
セキュリティ機能

XSS対策

- INTER-MediatorはHTML出力時にデフォルトでエスケープ処理を考慮

```
<td colspan="3" class="grayback" data-im="messageauth@message">
```

innerHTMLプロパティ

- 仕様上エスケープ処理をしない場合は innerHTMLプロパティに代入

<td colspan="3" class="grayback" data-im="messageauth@message@innerHTML">

CSRF対策

- params.phpで$webServerNameを設定
  - デフォルトでは未設定
  - Webアプリケーションが稼働しているホストのドメイン名もしくはFQDN（完全修飾ドメイン名）を配列で指定

CSRF対策

- params.phpでの$webServerName設定例

```
$webServerName = array('inter-
mediator.com', 'inter-mediator.info');
```

CSRF対策

- リクエストヘッダーにX-Fromおよび
Originを利用する手法を利用

http://hasegawa.hatenablog.com/entry/20130302/p1

クリックジャッキング対策

- params.phpで$xFrameOptionsを設定
  - 現在のところデフォルトでは未設定
- 設定例

```
$xFrameOptions = 'SAMEORIGIN';
```

INTER-Mediatorの認証機能

- ネイティブ認証
  - データベースエンジンに組み込まれたユーザーを利用する方法
- ユーザー認証
  - データベースに含まれるテーブルあるいはビューを利用する方法

INTER-Mediatorの認証機能

- INTER-Mediatorでの認証やアクセス権設定では**ユーザー**や**グループ**を使用
- LDAPやOAuth2による認証にも対応

INTER-Mediatorの認証機能

- authuser、authgroup、authcorのそれぞれのテーブルに記録しておくのが基本（ネイティブ認証以外の手法では）
- 認証をチャレンジ-レスポンスによって行うためのissuedhashテーブルも必要

認証は定義ファイルで設定

```
IM_Entry(
    array(array(
            'name' => 'chat',
            'key' => 'id',
            'authentication' => array('all' => array('target' => 'field-user', 'field' => 'user',),),
            'protect-writing' => array( 'user' ),
    ),),
    array(
        'authentication' => array( // オプション設定
            'user' => array('user1'), // ログイン可能なユーザー
            'group' => array('group2'), // ログイン可能なグループ
        ),
    ),
    array('db-class' => 'PDO'),
    false
);
```

特定ユーザーのみログイン

- オプション設定のauthenticationキーにuserキーの配列を指定

特定グループのみログイン

- オプション設定のauthenticationキーにgroupキーの配列を指定

レコード単位のアクセス権

- コンテキスト定義のauthenticationキーの配列の中で、操作名をキーにした配列で、 targetキーとfieldキーを指定

レコード単位のアクセス権

- targetキーの値が「field-user」ならfieldキーで指定したフィールドにある名前のユーザーに対して権限を付与
- targetキーの値が「field-group」ならfieldキーで指定したフィールドにある名前のグループに対して権限を付与

その他の設定項目

- params.phpで記述するセキュリティ関連の設定項目
  - $contentSecurityPolicy
  - $generatedPrivateKey
  - $passwordPolicy

詳細については

- INTER-Mediator Training Courseを参照
  - Chapter 7「セキュリティと認証・アクセス権」
  - Chapter 8「サーバーサイドでのプログラミング」

その他知っておきたいこと

- 暗号化通信のためのSSL/TLS
  - HTTPでは通信は暗号化されない
  - SSL/TLSを有効化したHTTPSを用いる

常時SSL

- 用途・目的に応じてHTTPを使用ではなく常にHTTPSの利用が推奨される状況
  - SSL/TLSを有効にするには認証局からSSLサーバー証明書を要購入
  - 無料の証明書（Let's Encrypt）も存在

INTER-Mediator Training Course

トレーニングコース

- INTER-Mediatorの開発手法を演習形式で自習する有償のトレーニングコース
  - ePub形式の電子出版物
  - INTER-Mediator-Server VMを利用しながら演習を進められる

サーバーサイドで出力調整（定義ファイルでの設定）

- コンテキスト定義にextending-classキーで記述
- クラス名に「.php」をつけたファイル名のファイルを定義ファイルと同一階層に配置

サーバーサイドで出力調整（定義ファイルでの設定例）

```
IM_Entry(
    array(
        array(
    "name" => "salesitems",
    "view" => "items",
    "query" => array(
                    array("field" => "year", "operator" => "=", "value" => "2016"),
                ),
    "extending-class" => "AdditionalProccess",
),
```

サーバーサイドで出力調整（PHPによる拡張例）

```
<?php
class AdditionalProccess implements Extending_Interface_BeforeRead, Extending_Interface_AfterRead
{
    public function doBeforeReadFromDB()  {
    }

    public function doAfterReadFromDB($result)  {
        /* ここに独自の処理を記述 */
        return $result;
    }
}
```

サーバーサイドで出力調整

- 詳細はINTER-Mediator Training Courseの Chapter 8「サーバーサイドでのプログラミング」を参照

まとめ

まとめ

- Webアプリケーションの脆弱性をなくす一般的な解決策を知る
- フレームワークが提供するセキュリティ機能と前提条件を把握する
- データベースソフトウェアが備えるセキュリティ機能を理解する