

# Web アプリケーション セキュリティ入門ハンズオン

## 目次

- 1.はじめに
- 2.注意事項
- 3.知識として
- 4.検証環境のセットアップ
- 5.ハンズオンをする際の環境.
- 6.使用するツールのセットアップ
- 7.ハンズオン手順
- 8.最後に

---

## 1. はじめに

今回は Web アプリケーションのセキュリティ診断入門ということで、Web アプリケーションのメジャーな脆弱性の概要と対策、検出の仕方をハンズオン形式でご紹介します。  
このハンズオンでセキュリティ分野にも関心を向けていただければ幸いです。

---

## 2. 注意事項

ご紹介した手法は絶対に自分の管理外のアプリケーションに試さないこと。  
ローカル環境やプライベート IP のみの環境で検証してください。  
AWS や Azure 上の環境はプライベートな環境とは言いません。  
自身の管理下であってもクラウド上でグローバル IP が振られている場合は注意が必要です。

---

### 3. 知識として

- HTTP (Hyper Text Transfer Protocol)

- ブラウザ ⇄ サーバ間で HTML をやり取りする際の通信プロトコル。
- 原則として送信したリクエストに対する応答レスポンスは必ず 1 対 1 になる。

- HTTP リクエストメソッド

- HTTP リクエスト送信時のリクエストの種類を指定する。POST, GET などの種類がある。

- Cookie

- Web アプリケーションから HTTP を使用して発行される値。
- ブラウザに保存されユーザの識別に使われる。同一ドメインでのみ送信される

(HTTP リクエスト)

HTTP リクエストメソッド

```
POST /sql/login.php HTTP/1.1
Host: www.hacker.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www.hacker.com/sql/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 67
Cookie: PHPSESSID=2237590ec1d3e299e26a81bc04485b50
userid=test&passwd=12345&login=%E3%83%AD%E3%82%B0%E3%82%A4%E3%83%B3
```

Cookie

## 4. 検証環境のセットアップ

※セミナー当日は検証環境サーバを立ち上げているため、読み飛ばしてもらっても構いません

自宅などで再検証する際に参照してください。

### ■git リポジトリ

```
$git clone https://github.com/halkichi0308/seminar_20181027.git
```

### ■Docker コマンド

(Dockerfile があるディレクトリ内で実行。これを叩くだけで環境が立ち上がります。)

```
$docker-compose up -d --build
```

### ■docker-compose しない場合

3つのコマンドを実行

```
$docker run --name mysql-server -e MYSQL_ROOT_PASSWORD=pass -d -p 3306:3306 mysql:5.7
```

```
$docker build -t apache_php:1.0 .
```

```
$docker run --link mysql-server -v $(pwd)/html:/var/www/html -p 80:80 -d apache_php:1.0
```

\$(pwd)は powershell でも使えますが、フルパスでないと認識しないかもしれません。

※新規登録でエラーが出る時。(mysql/mysql-server の場合)

```
[mac]$docker exec -it <mysql-server のコンテナ ID> /bin/bash
[docker-bash]$mysql -u root -p
password->pass
[sql]GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'pass';
```

---

## 5. ハンズオンをする際の環境

※<localIP>は検証環境の IP に読み替えてください。セミナー当日は講師が提示します。

・SQL インジェクションの検証環境

`http://<localIP>/sql/login.php`

・クロスサイトスクリプティングの検証環境

`http://<localIP>/xss/xss.php`

---

## 6. 使用するツールのセットアップ

### ■①Burp の設定

画面が立ち上がったら

- ① `Temporary project` -> `Next` (デフォルトのまま)
- ② `Use Burp Defaults` -> `Start Burp` (デフォルトのまま)
- ③ `Target` -> `Scorp` -> `Target Scorp`  
[add]をクリック -> prefix に検査の対象 URL(<ip>)を入力
- ④ `Proxy` -> `Options` -> `Intercept Client Requests`  
Intercept requests based on the following rules: にチェック  
一番下の ☐ [And] [URL] [Is in target scorp] にチェック
- ⑤ `Proxy` -> `Options` -> `Intercept Server Responses`  
Intercept responses based on the following rules: にチェック  
真ん中あたり ☐ [Or] [Request] [Was Intercepted] にチェック
- ⑥ `Project Options` -> `Out-Of-Scorp-Requests`  
[Drop all out-of -scorp-requests] にチェック



### 各項目の説明

- ③ - `Target Scorp`: 通信を許可する対象の設定。
- ④ - `Intercept Client Requests`: リクエストを送信する際のルールの指定。
- ⑤ - `Intercept Server Responses`: レスponseを受け取る際のルールの指定。
- ⑦ - `Drop all out-of -scorp-requests`: ③で許可しない対象を Drop する。

## ■②Firefox

### プロキシの設定

画面右上のハンバーガーアイコン→設定



設定 → 一般 → ネットワークプロキシ[接続設定]

インターネットで使用するプロキシを設定する

→手動でプロキシを設定するにチェック

→HTTP プロキシ 127.0.0.1 ポート 8080(Burp と合わせて下さい)

→全てのプロトコルでこのプロキシを設定する

→Burp のポートの確認方法

Proxy → Options → Proxy Listeners

Firefox は PC のシステム設定とは独立した Proxy 設定を持つことができるため、使用するブラウザはこちらを推奨します。

## 7. ハンズオン手順

### ■SQL インジェクション例

(入力文字列)

```
test 'OR' a '=' a
test ' || '
```

※SQL の WHERE 句で OR(真)を使うと全件マッチになるため、全てのデータに更新処理が実行されます。更新と伴う処理で使わないようにしてください。実際の診断の際は DB の既存データへの影響が少ない以下のような文字列を使用します。

```
test 'AND' a '=' a
test 'AND' a '=' b
```

### ■SQL インジェクション対策

言語で用意されたプレースホルダ or フレームワークで推奨されたメソッドを使う

PDO→prepare 等

## ■認証処理の注意点

- ・パスワードはハッシュ化する。
  - ハッシュ関数は自分で実装せず、言語で推奨されている関数を使うこと。
- ・ユーザの推測ができるようなメッセージを表示しない。
  - 「メールアドレスが登録されていません」などのメッセージもこれに該当する。

## ■アプリケーション全体の注意点

DB や言語が推測できるようなエラーメッセージを表示しない。

mysql が動いていることが推測できる => 攻撃者の手間を格段に減らすことができる。

## ■クロスサイトスクリプティング（入力文字の列）

```
first:
keyword ><script>alert(document.cookie)</script>

secound:
keyword ' ;alert(document.cookie); '

third:
userid ');alert(document.cookie);//

fourth:
/xss/xss.php?mode=xss4&keyword=<script>alert(document.cookie)</script>
```

## ■クロスサイトスクリプティング対策

全体に共通する内容として、ユーザの入力値は必ず無害化を行う、必ず言語のエスケープ関数かフレームワークで推奨されたメソッドを使う。

ハンズオン XSS での対策例

```
first: secound:
htmlspecialchars()などでメタ文字を実体参照へ変換
```

```
<script> -> &lt;script>
```

```
third:
URL エンコードを行う
```

```
<script> -> %3Cscript%3E
```

fourth:

レスポンスを Content-Type: application/json にする

#### ■ハンズオン DOM-base

first:

```
<a href="javascript:alert(document.cookie)">click</a>
```

#### ■DOM-Base 対策

レンダリングする関数に innerHTML を使用しない

innerText などを使う or フレームワークで推奨された関数

---

## 8. 最後に

知らない脆弱性是对策ができない

設計段階でセキュリティを要件に組み込む

脆弱性を作り込まない意識

サービスが完成してから発見される脆弱性の方が改修が難しい

登壇者

幸田 将司

主催:

ENGINEER STYLE TOKYO

<https://tec.connpass.com/>