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

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。
ローカルの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. セットアップ

■git リポジトリ

```
git clone https://github.com/halkichi0308/seminar-20180825.git
```

■Docker コマンド

(Dockerfile があるディレクトリ上で)

```
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';
```

---

4. 環境

・SQL の環境

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

・XSS が試せる環境

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

■Burp 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 followling 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 followling 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
```

■Firefox
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 は独立した Proxy 設定を持つことができるため、ブラウザはこちらを推奨します。

---

5. ハンズオン

■ハンズオン 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
更新処理で使わないようにしてください。
診断の際は DB の既存データへの影響が少ない文字列を使用します。

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

■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。

■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。

■ハンズオン XSS 例
(入力文字列)

```
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)</sc
ript>
```

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

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

third:
URL エンコードを行う

fourth:

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

■ハンズオン DOM-base

```
secound:
<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

---

6. 最後に

知らない脆弱性は対策ができない

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

present by kouda

主催:

ENGINEER STYLE TOKYO

https://tec.connpass.com/