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。
このハンズオンを通じて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．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

- ブラウザから送信される通信。POST,GET などの種類がある。

・HTTP レスポンス

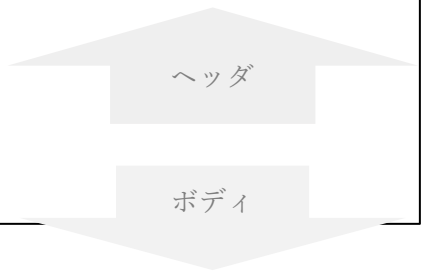
- サーバから送信される通信。

・Cookie

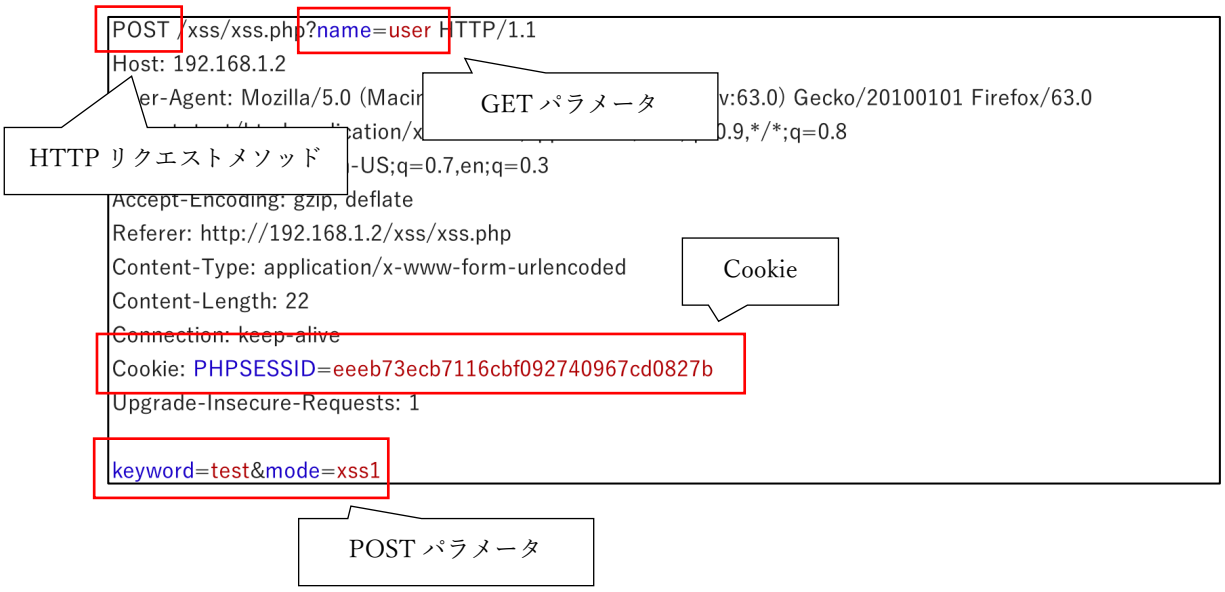
- Web アプリケーションから HTTP を使用して発行される値。
- ブラウザに保存されユーザの識別に使われる。同一ドメインでのみ送信される

通常、**HTTP 通信のリクエスト**は以下のような形式です。

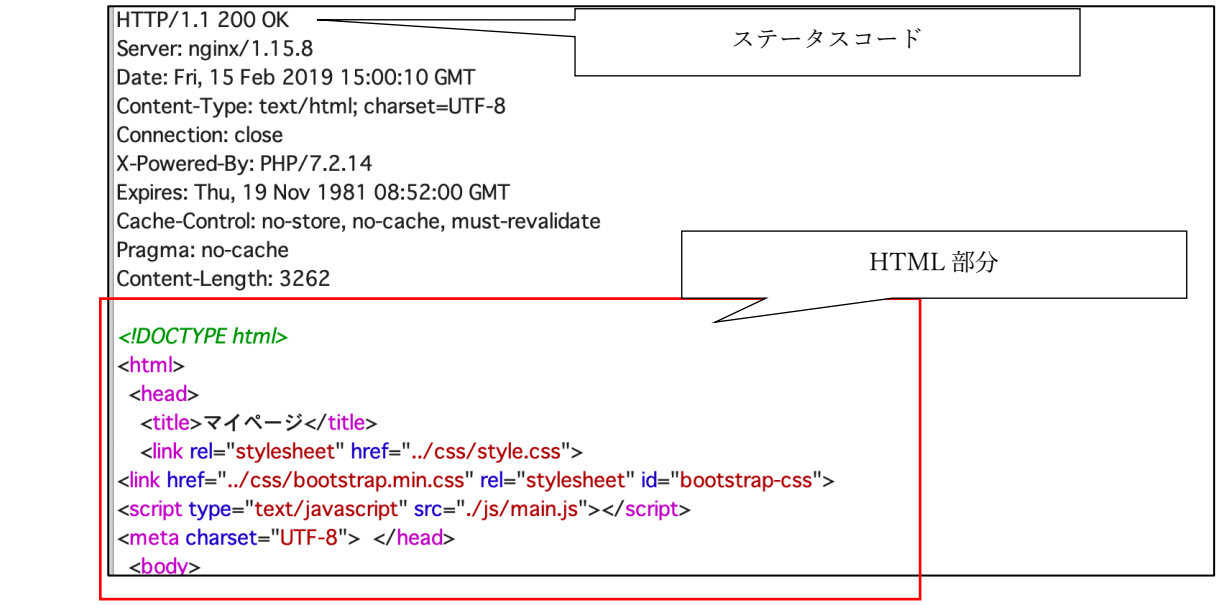
```
POST /xss/xss.php?name=user HTTP/1.1
Host: 192.168.1.2
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:63.0) Gecko/20100101 Firefox/63.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.2/xss/xss.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 22
Connection: keep-alive
Cookie: PHPSESSID=eeeb73ecb7116cbf092740967cd0827b
Upgrade-Insecure-Requests: 1

keyword=test&mode=xss1
```



以下は、前ページで列挙した HTTP リクエストの詳細です。



対して、以下は **HTTP レスポンス**です。



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_20190216.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
```

---

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

・クロスサイトスクリプティングの検証環境

http://<localIP>/xss

---

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 -> burp_config.json を選択

➔ Start Burp を選択

burp_config.json の設定を引き継ぐため、上記の 2 点で準備は完了です。

各項目の説明

Target Scorp: 通信を許可する対象の設定。

Intercept Client Requests: リクエストを送信する際のルールの指定。

Intercept Server Responses: レスポンスを受け取る際のルールの指定。

Drop all out-of -scorp-requests: 通信を許可しない対象を 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

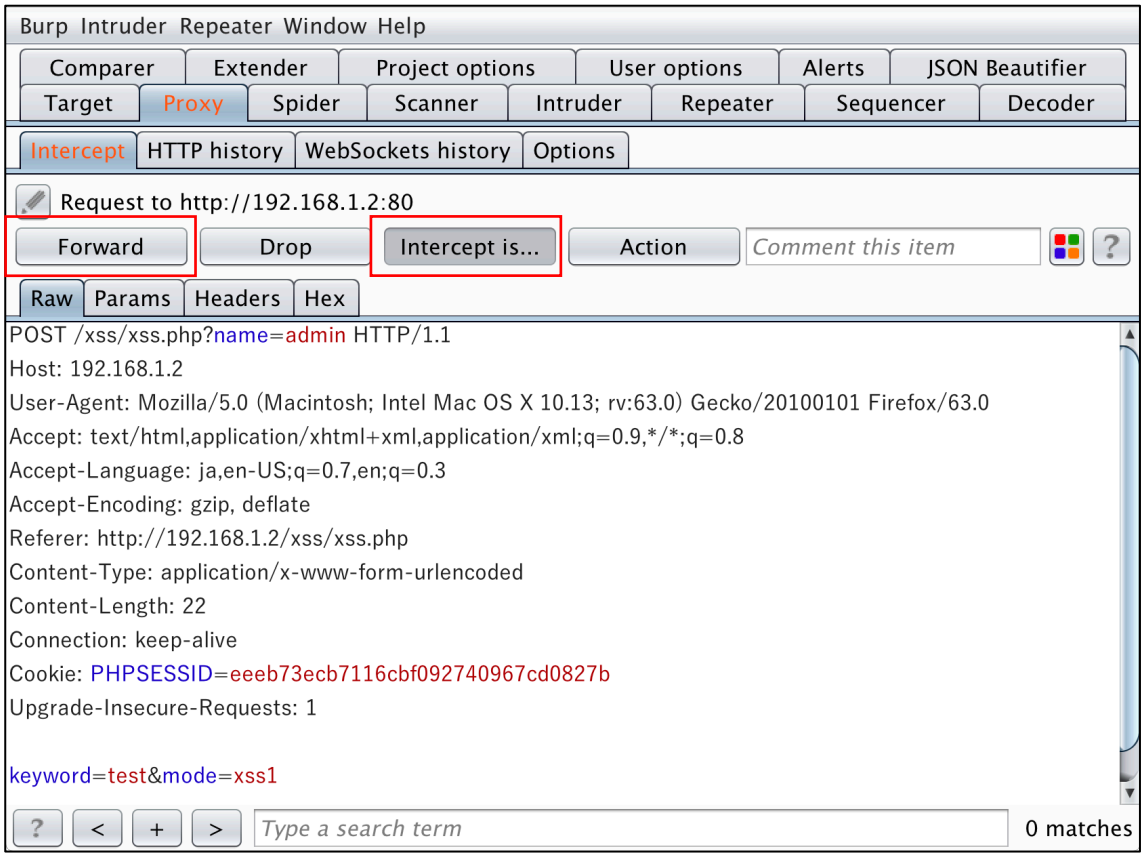
Firefox は PC のシステム設定とは独立した Proxy 設定を持つことができるため、使用するブラウザはこちらを推奨します。

7．ハンズオン手順

Burp が HTTP リクエストを受け取ると Proxy -> Intercept のタブに HTTP リクエストが表示されます。

[Forward]を押下すると、表示されているリクエスト/レスポンスを送信します。

[Intercept is on/off]を切り替えると、Burp が通信を途中で止めなくなります。遷移を確認したい時に便利です。



■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:
/xss?mode=xss2&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:
htmlspecialchars()などでメタ文字を実体参照へ変換
 <script>  ->  &lt;script$gt;

secound:
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

.キーワードとして

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

幸田 将司

主催:

ENGINEER STYLE TOKYO

https://tec.connpass.com/