セッションの不備とHTTPヘッダインジェクション

セッションの不備

セッション・・・セッションIDを利用して、**現在の状態を記憶**させる仕組み。

セッションID・・・Webアプリケーションの**利用者を識別するためのID**。

セッションハイジャック

セッションIDを悪用して、他人に**なりすます**攻撃。

セッションIDを取得する手段

1. セッションIDの推測・・・セッションIDが第三者から**推測**される。<例>連番、日付+ユーザIDなど
2. セッションIDの盗み出し・・・**XSS**などを利用して盗み出す。
3. セッションIDの強制・・・セッションIDを利用しているブラウザに設定、**セッションIDの固定化**する攻撃

＊影響

重要情報の閲覧、権限で操作、IDによるメール、ブログ投稿、設定変更など。

セッションIDの取得について

1. 推測可能なセッションID

   セッションIDの**生成規則**に問題　→　**推測される可能性**がある。

   対策・・・ミドルウェアを利用した方がよい。

   <ダメな例>　ユーザID、メールアドレス、リモートIPアドレス、日時、乱数

   ▼ 攻撃手法

   1.対象アプリから**セッションIDを集める**

   2.セッションIDの**規則性の仮説を立てる**

   3.推測したセッションIDを対象のアプリで試す

＊脆弱性の原因

自作している(セッションID生成プログラムを作成)

＊対策

ツールのセッション管理機構を使用する（**セッションIDを生成する**ツールを使用）

2. URL埋め込みのセッションID

Refererヘッダを経由して、セッションIDが外部に漏えいする場合がある。

漏洩条件・・・URL埋め込みのセッションIDがある。外部サイトへのリンクがある。

＊脆弱性の原因

php.iniなどの不適切な設定、プログラミング（セッションIDの生成など）。

＊対策

クッキーにセッションIDを保存するように設定。

<例>php.ini

```
[Session]
session.use_cookie = 1         ・・・クッキーを使用する
session.use_only_cookie = 1      ・・・セッションIDをクッキーのみに保存
```

3. セッションIDの固定化

▼ 攻撃手段

セッションIDを**入手**→被害者に対してセッション**IDを強制**する→被害者は標的アプリにログインする

→攻撃者・・・**強制したID**によりアプリにアクセスする。

↓

なりすましによる情報漏えい、アプリの悪用、データの操作など。

＊対策

ログイン時に**セッションIDを変更**する。

セッションアダプション

未知のセッションIDを受け入れてしまう脆弱性のこと。

＊**クッキー**にセッションIDを保存した場合でも、**固定化される**ことがある。

＊原因

　URLに埋め込まない、クッキーモンスターバグあるブラウザは使用しない、クッキーモンスターバグが発生しやすいドメイン名を使用し

　ない、XSSの脆弱性をなくす、HTTPヘッダインジェクションの脆弱性をなくす、クッキーの書き換えの脆弱性をなくす。

　クッキーモンスターバグ・・・一部のブラウザのバグ。セッション情報がクッキーに送信されないなど。

＊対策

　認証後にセッションIDを変更する、トークンを使用する。

リダイレクト処理

リダイレクト処理とは・・・

指定したURLへユーザを**自動的**に誘導・転送する機能のこと。

リダイレクト処理の脆弱性

　オープンリダイレクト脆弱性、HTTPヘッダ・インジェクション脆弱性が該当する。

＊オープンリダイレクト

　任意のドメインにリダイレクトできる脆弱性。フィッシングやマルウェアのダウンロードなどに利用される。

＊原因

　リダイレクト先の**URLを外部から指定できる**。

　リダイレクト先の**ドメイン名のチェックをしない**。

＊対策

　リダイレクト先を固定する、ドメイン名をチェックする。

　ドメイン名のチェック方法・・・正規表現を利用してURLをチェックする。

HTTPヘッダ・インジェクション

リダイレクト処理、クッキー出力などの出力処理で発生する。　→ 脆弱性を利用

レスポンスヘッダを出力する時に**パラメータに改行を挿入**する。

↓

任意のレスポンスヘッダを追加、**レスポンスボディの偽装**

＊影響

任意のクッキーの生成（なりすまし）、任意のURLへのリダイレクト、表示内容の改変、JavaScriptによるXSSと同様な被害。

＊対策

HTTPヘッダの出力部分は手作りしない、ヘッダ出力用の**ライブラリやAPIを利用**する。

ヘッダの文字列に**改行コード**が含まれているか確認する。

URL中の**改行**はエラー、クッキー値の**改行はパーセントエンコード**する。

<例>CR(キャリッジリターン)→%0D　LFLF(ラインフィード) →%0A

- 外部からのパラメータを**HTTPヘッダとして出力しない**。

<例>リダイレクト、クッキー生成が該当

↓

リダイレクト先のURLでなく、固定または番号で行う。開発ツールが提供するセッション変数を使いURLを渡す。

＊PHP・・・対策している。

＊Apache・・・複数のLocationがある時は最後のものを返す。**%0D%0A**・・・改行

＊原因

HTTPレスポンスヘッダはテキスト形式で１行ずつに1つずつヘッダを定義する。

↓

ヘッダとヘッダは改行で区切られる。

クッキー出力

脆弱性

クッキーを**利用すべきでない目的**でクッキーを利用している。

クッキーの**出力方法に問題**がある

↓

受動型攻撃　・・・　HTTPヘッダ・インジェクション、クッキーのセキュア属性の不備。

＊クッキーに保存するもの

セッションID、***その他のデータは保存しない***。

セッション変数で代用ができる。　＊セッション変数・・・**タイムアウトすると表示されなくなる**

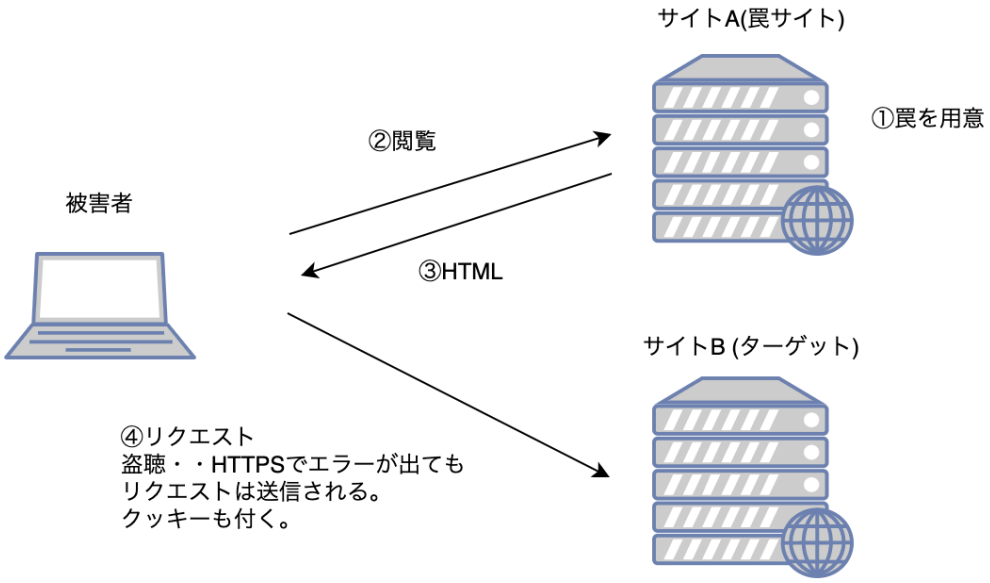
<例>ヤフーの認証・・・クッキーにはトークンのみを保存する。

クッキーのセキュア属性の不備

secure属性・・・HTTPS通信でも**secure属性**がついていないと、クッキーは**平文**で送られる。

↓　(盗聴される)

セッションIDに対してセキュア属性を設定するとアプリが動作しなくなることもある。



＊原因

開発者が**クッキーのセキュア属性**を知らない。

セキュア属性を付けると**アプリが動かなくなる**

↓

HTTP、HTTPSが混在する、**HTTPのページでセッションIDなどが受けれない**。

＊対策

1.セッションIDのクッキーにセキュア属性を付ける

方法・・・**php.ini**で設定、**session.cookie_secure = on**

2.トークンを用いる対策

トークンにセキュア属性を付ける。

▼ トークンを利用する理由

認証成功時に**一度だけサーバから出力**される、**HTTPSのページが生成**される(サーバからブラウザ)。

確実に**暗号化されてブラウザから送信**される(ブラウザからサーバ)。**HTTPS**のページを閲覧する場合には**トークンが必須**。