CSRF(Cross Site Request Forgery)

CSRF

CSRF攻撃・・・攻撃者の罠により、Webアプリケーションが持つ機能を利用して、ユーザが意図しない処理を行わせる攻撃。

XSS情報を取得する攻撃はしない。

＊CSRF脆弱性・・・ユーザが意図しない処理を実行されてしまう可能性(取消しできない重要な処理など)

攻撃手法

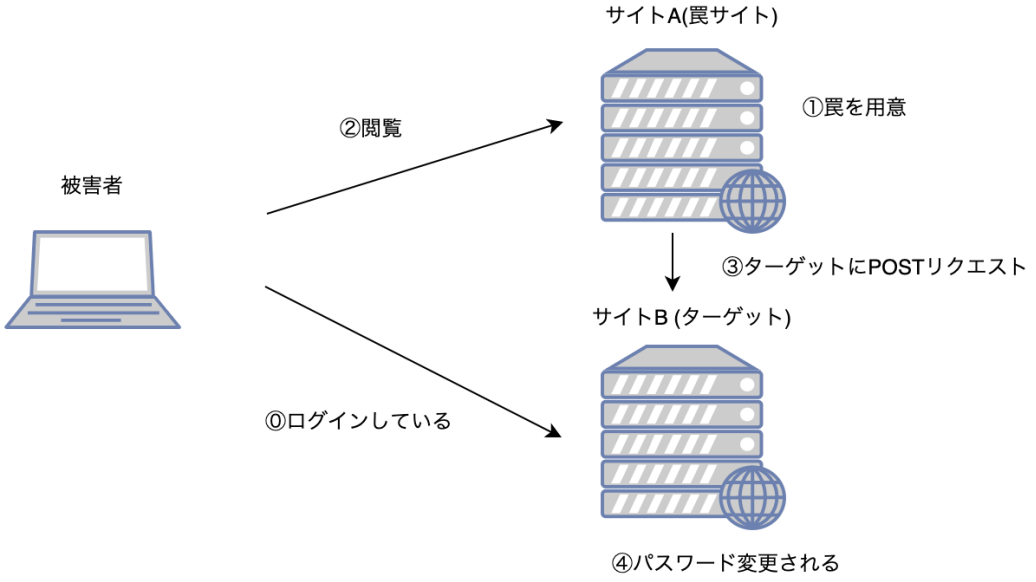
1. 「入力→実行」の攻撃例

   入力した値を使用してCSRF攻撃を実行する。

   <例>パスワード変更の処理

   パスワード変更条件

   ログインを**リクエストされており**、**ログイン状態**である。送信時のパラメータとして**パスワード**が指定されている。

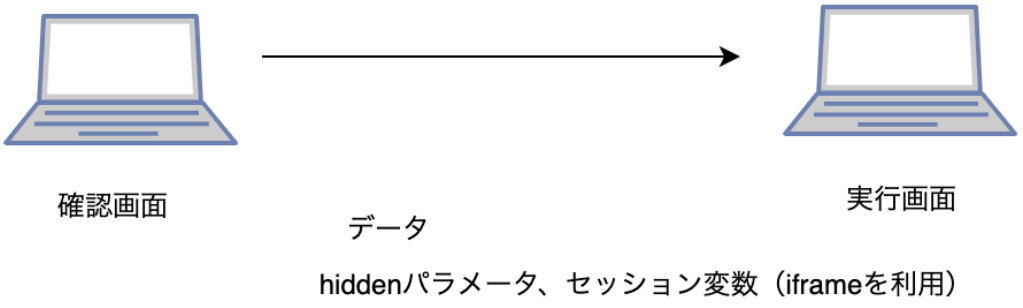


＊被害者のパスワードが変更され、不正ログインされて情報を盗み出されてしまう。

2. 「入力→確認画面→実行」の攻撃例

   実行前に確認画面を経由する例。**hiddenパラメータ**を利用して、変更に使用されるデータが送られる。

   <例>メールアドレスの変更



3. ファイルアップロードフォームでの攻撃例

   クロスオリジンに対応した**XML.HttpRequest(HTTP通信を行うAPI)**による攻撃ができる。

CSRF脆弱性の原因

1. **form要素**の**action属性**にどのメインのURLでも指定できる

→ 攻撃対象サイトにリクエスト送信できる(**同一オリジンポリシーの制限**に関わらないため)。

2. **Cookie**に保存された**セッションID**は、対象サイトに自動的に送信される。

→被害者ユーザの**Cookie**を利用されてしまうため。サーバの方で被害者ユーザと認識されてしまう。

対策

- CSRF対策の**必要なページ**を区別する

  必要なページのみ対策する。<例>購入ページ、変更ページなど。

- 正規利用者の意図したリ**クエストを確認**できるように実装する

  正規利用者が自ら「実行」ボタンを押した結果のリクエスト。

  判定方法・・・トークンの埋め込み、パスワードの再入力、Refererのチェック

  1. **トークンの埋め込み**

     トークン・・・第三者が知り得ない**秘密情報**のこと。乱数などを利用。トークンを利用して、罠サイトからのリクエストなどか検証する。

     トークンを生成、**セッション**ごとにサーバで保存する。

  2. **パスワードの再入力**

     ユーザ側にパスワードの再入力を求める。

  3. **Refererのチェック**

     「重要な処理」を実行するページで**Refererを確認**する。正規なリクエストが判断できる。

- その他にもSameSite Cookieを利用する方法などある。

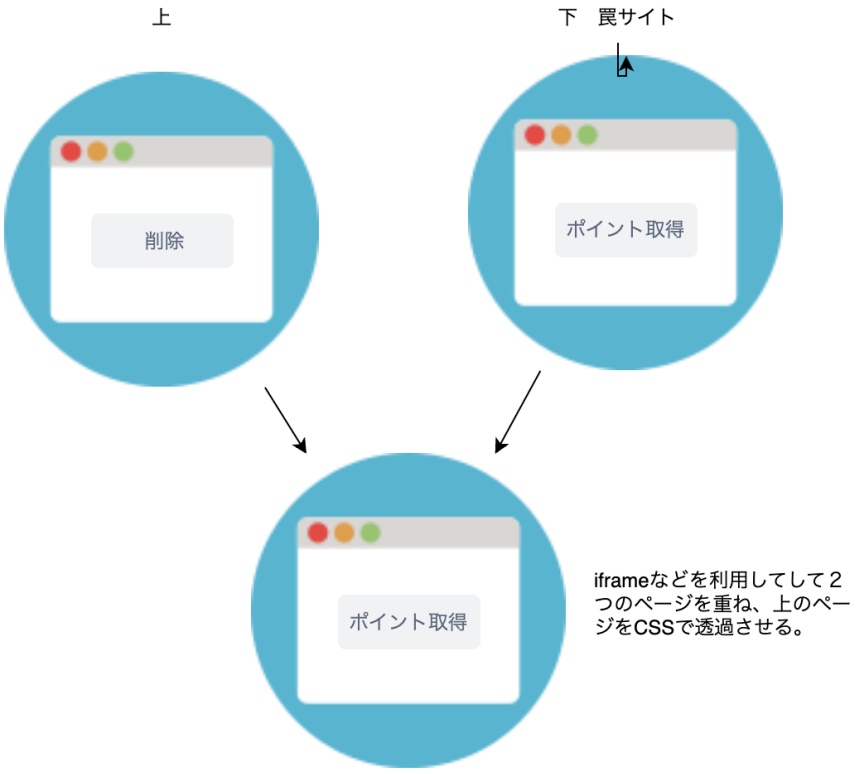
  SameSite Cookie・・・Cookieに設定できる属性の１つ。別のサイトへアクセスする場合にCookieを送るかどうか指定する。

＊保険的な対策・・・処理内容の通知メールを送信する。

クリックジャック

iframe,cssを利用して**透明にした攻撃対象ページと罠サイトを重ね合わせてクリックを誘導する**。ユーザが意図しない処理を実行させる。

1. 攻撃パターン



実際の例

Twitterのウェブインシデント機能の利用

2. 対策

iframeなどのフレーム内に**ページを埋め込むことを制限**する。**X-Frame-Optionsヘッダ**を指定する。

<例>header('X-Frame-Options:**SAMEORIGIN**');　・・・　クロスオリジンのフレームへの埋め込みを禁止する。

DENY (すべての埋め込みを禁止)、**ALLOW-FROM** (指定したオリジンのみ許可する)などもできる。