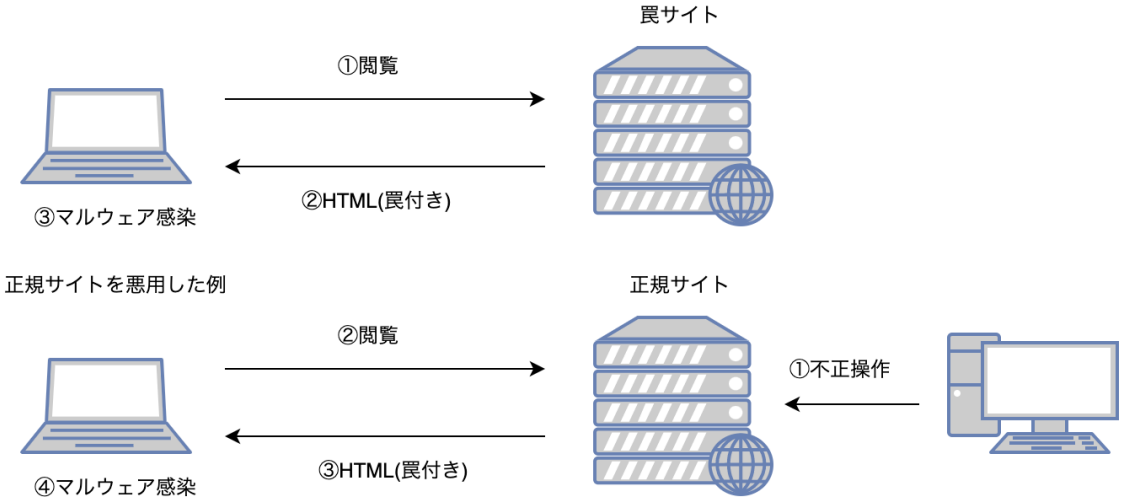
受動的攻撃・クロスドメイン受動的攻撃

能動的攻撃・・・攻撃者が直接攻撃する。<例>SQLインジェクションなど

受動的攻撃・・・罠にかかったユーザを通して、アプリケーションを攻撃する。



＊不正操作

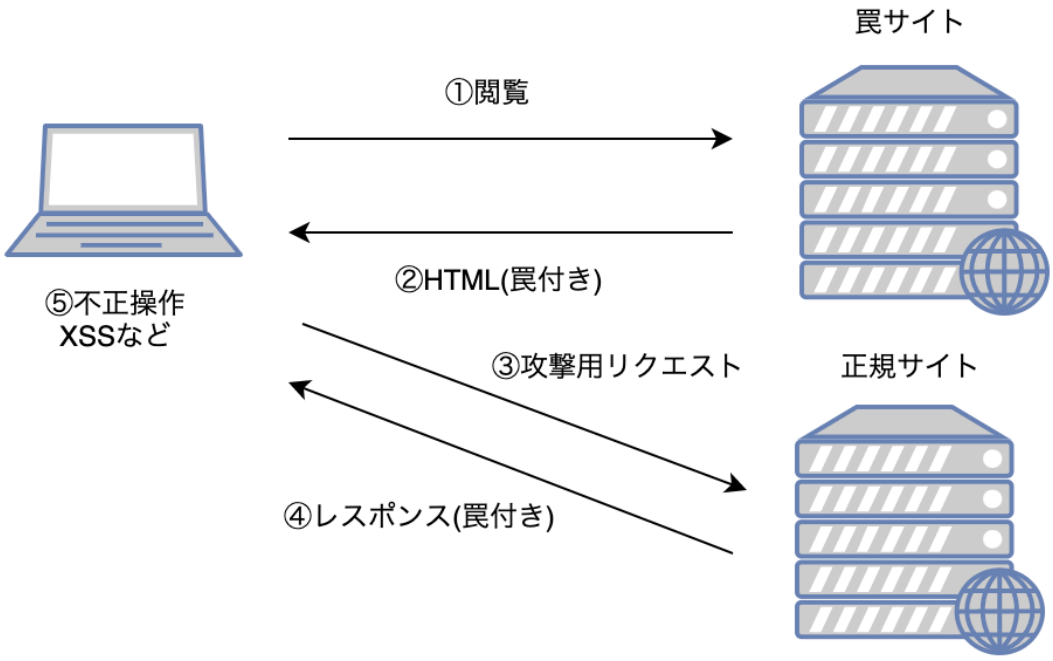
FTPなどのパスワードを入手して**コンテンツを書き換える**。　←　FTPのパスワードはデフォルト**暗号化**されていない。

Webサーバの脆弱性をついた攻撃によりコンテンツを書き換える。

SQLインジェクション攻撃によりコンテンツを書き換える。　←　**Update**や**Insert**を利用して書き換える。

XSS脆弱性を悪用。　←　**JavaScript**が実行され、**クッキーID**が取得される。

サイトをまたがった受動的攻撃



サンドボックス

砂場、プログラムの制約がある場所。

<例>JavaScript？

ローカルファイルへのアクセス禁止、ネットワークアクセスの制限など。

同一オリジンポリシー(Same Origin Policy)

クライアントスクリプトから**サイトをまたがったアクセスを禁止**するアクセス制限のこと。

<例>iframeの悪用

iframeの外側からiframeの内側のHTMLの内容をJavaScriptにより参照することができる。

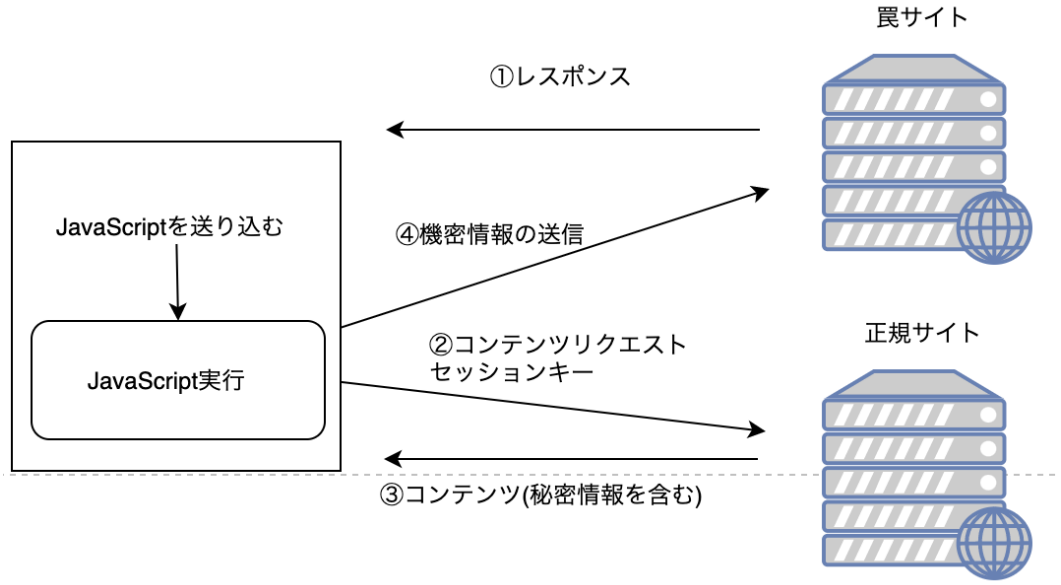
＊**同一オリジンの条件**

URLのホスト(FQDN)が一致、プロトコルが一致、ポート番号が一致。

Cookieにはプロトコルとポート番号は関係がないため、JavaScriptの制限が最も厳しい。

アプリケーションの脆弱性

ブラウザは同一オリジンポリシーにより受動的攻撃を防止する。ただし、**アプリケーションに脆弱性がある**とXSSなどの**受動的攻撃**を受けることがある。



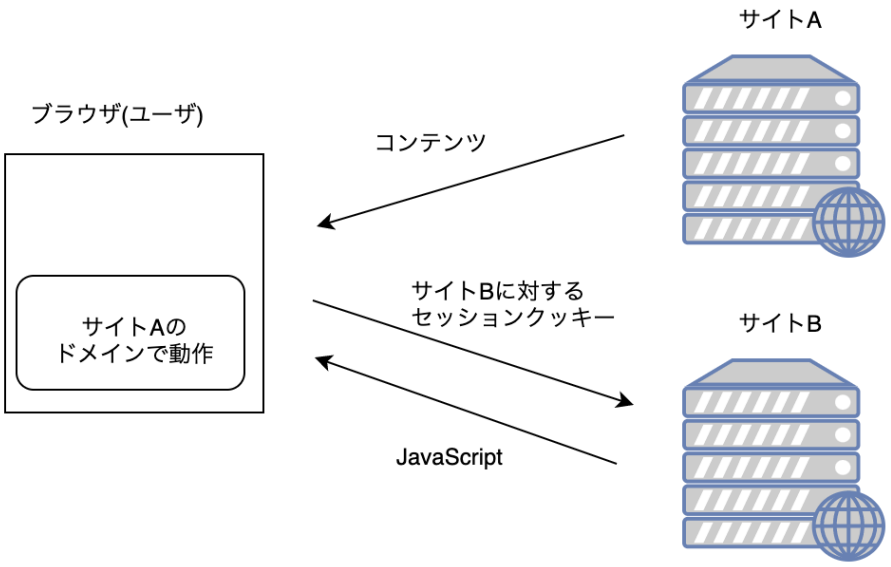
クロスドメインがブラウザ機能で**許可**されているもの

img要素・・・src属性はクロスドメインの指定が可能。

script要素・・・src属性で他のサイトからのJavaScriptを埋め込みは可能。

＊**JSONP(JSON with Padding)**

Ajaxアプリケーションから同一オリジンでないサーバ上のデータにアクセスするための手法。



<例>**サイトA**のドキュメントを**サイトB**上のJavaScriptを読み込んだケース

サイトBにある**JavaScriptは**読み込み元の**サイトAのドメインで実行されるため**、**document.cookie**を実行するとサイトAのクッキーを取得できる。

1. frame、iframe属性

   クロスドメインのアクセスは可能、ただし、JavaScriptによりクロスドメインのドキュメントへのアクセスは禁止されている。

2. CSS

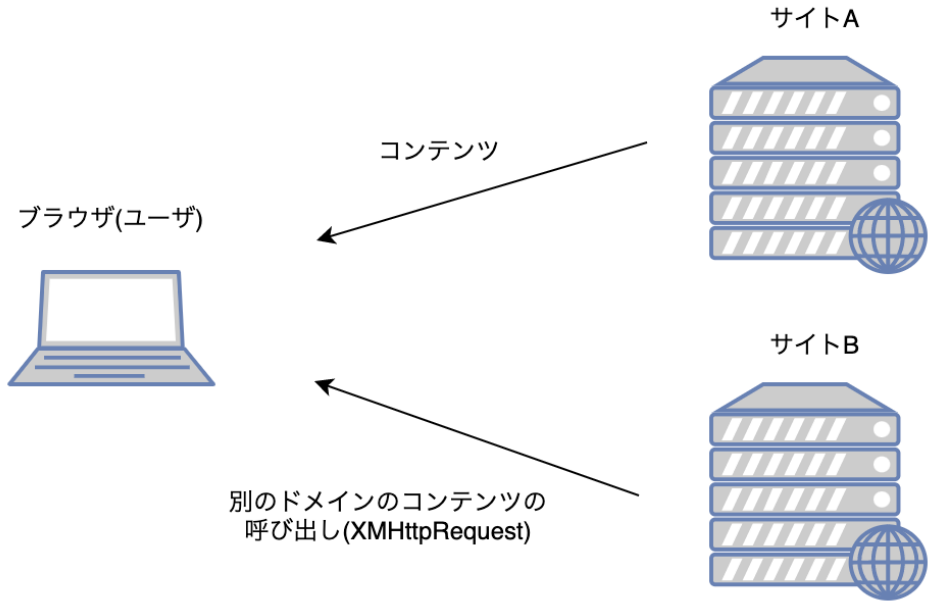
   クロスドメインの読み込み可能。

3. formのaction

   クロスドメインの指定が可能、CSRFに利用される。

CORS (Cron-Origin Resource Sharing)

サイトを越えて異なるオリジンのデータをやり取りを可能にする仕様。
XMLHttpRequestなどで使用される。



▼ **Access-Control-Allow-Origin**

クロスオリジンからの呼び出しを許可する仕組み。情報の提供元がHTTPレスポンスヘッダを出力する。XMLHttpRequestなどのアクセ

スを許可する場合に使用する。**指定することで異なるオリジンからのJavaScriptを実行できるようになる**。

<例>Access-Control-Allow-Origin: http://～

プリフライトリクエスト

クロスオリジンアクセスにおいてブラウザは**ある条件を満たさない場合**、**プリフライトリクエスト(pre-flight request:HTTPリクエスト)**を返す。

プリフライトリクエストの不備によりエラーを出力してしまう。

▼ ある条件

メソッドはGET、HEAD、POSTのいずれか。

Content-Typeヘッダはapplication/x-www-form-urlencoded、multipart/form-data、text/plainのいずれか。

XMLHttpRequestのsetRequestHeaderで指定するリクエストヘッダがAccept、Accept-Language、Content-Language、Content-Type

認証情報を含むリクエスト

クロスオリジンに対するリクエストにHTTP認証、クッキーなどの認証に使用されるリクエストヘッダは自動的に送信されない。これらを用いる場合は、**XMLHttpRequestのwithCredential**を**true**に設定する必要がある。

<例>Access-Control-Allow-Credentials: true