安全な Web アプリケーションの構築

4D デベロッパカンファレンス 2010 配付資料

目次

はじめに

このセッションでは、4D を使用して Web アプリケーションを作成される方向けに、安全なアプリケーション構築のための情報を提供します。

参考資料

安全なウェブサイトの作り方　改訂第 4 版
情報処理推進機構
http://www.ipa.go.jp/security/vuln/websecurity.html

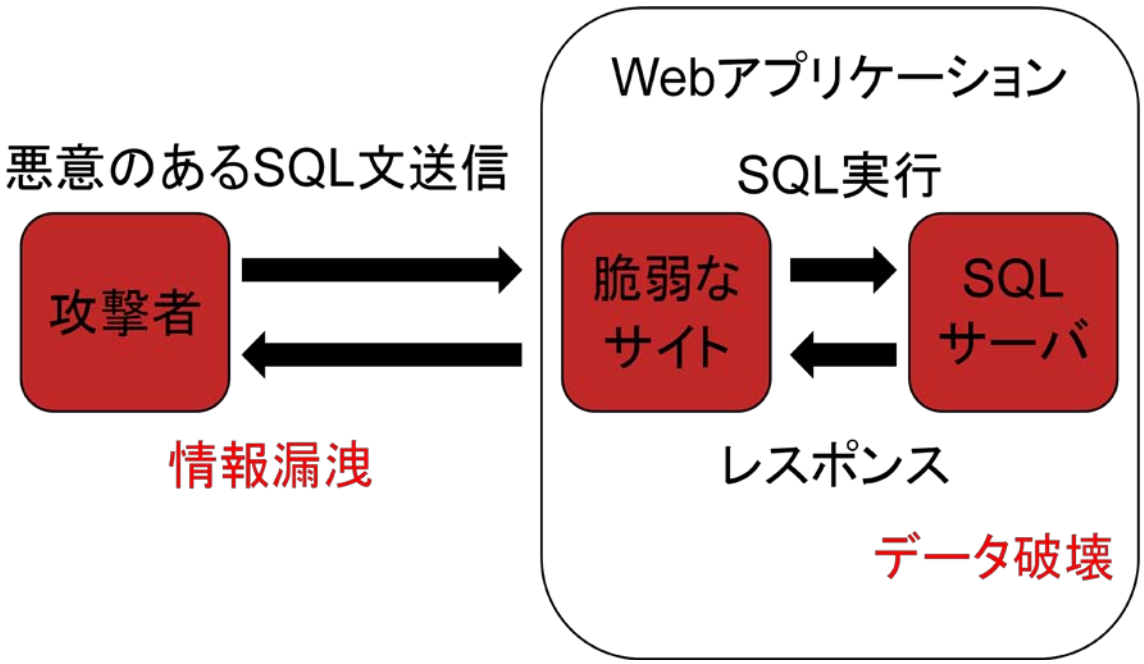
留意点

- 試しに他人のサイトを攻撃するなんてことのないようにお願いします。
- Web アプリケーションを開発する際の対策を、攻撃手法ごとに紹介します
    - 紹介する方法を強制するものではありません
    - 紹介する方法で完全であることを保証するものでもありません
        - すべてを紹介しているわけではありません
        - 日々新しい脅威が生まれています
    - 開発者様による日々の情報収集が最も大切

扱う脅威

- SQL インジェクション
- コマンドインジェクション
- ディレクトリトラバーサル
- セッション管理の不備
- クロスサイトスクリプティング (XSS)
- クロスサイトリクエストフォージェリー (CSRF)
- HTTP ヘッダインジェクション

SQL インジェクション



原理

コードが以下のとき
$t:=″login='″+$login+″' AND pass='″+$pass+″';″
QUERY BY SQL([users];$t)

$login: any
$pass: ' OR 'A' = 'A
を与えると、結果の SQL は
login='any' AND pass='' OR 'A'='A'
となり、常に True となります。

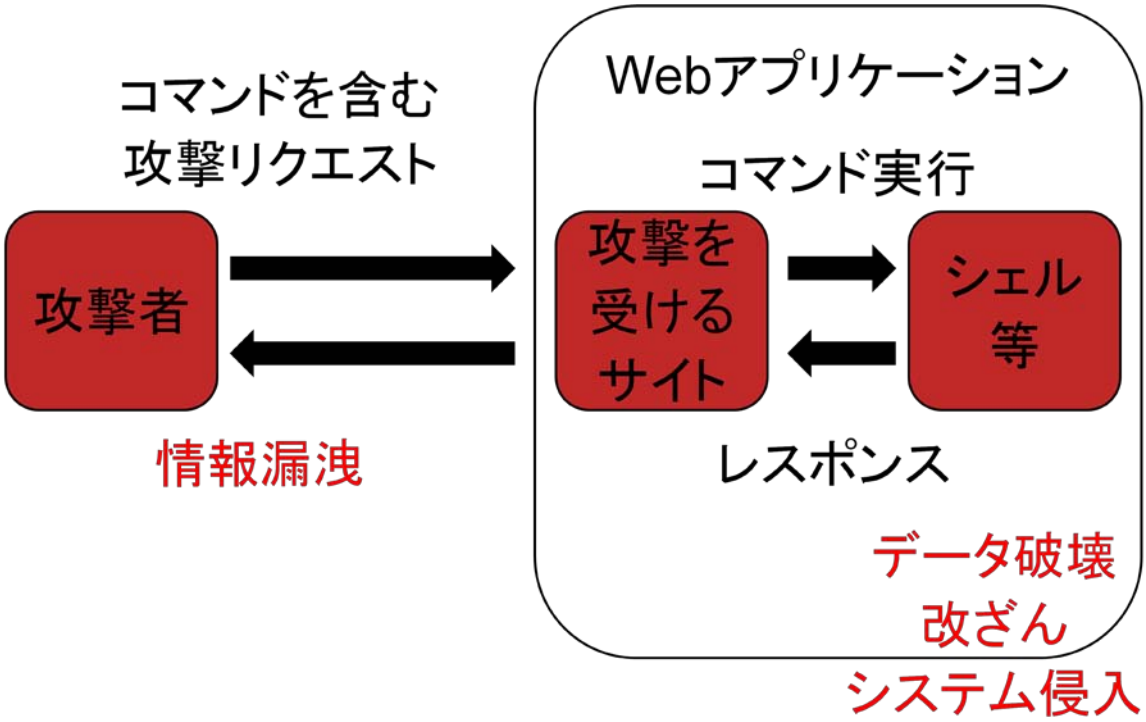
なぜ発生するのか

入力値に不正に SQL 特別文字を挿入され、期待した SQL 文とは異なるものにされてしまう。

対策

- SQL を使用しない (4D のクエリを使用)
- SQL 文の組み立てにバインド機構を使用する
  $t:=″login=:$login AND pass=:$pass;″
  QUERY BY SQL([users];$t)

コマンドインジェクション



どのような時に影響を受けるか

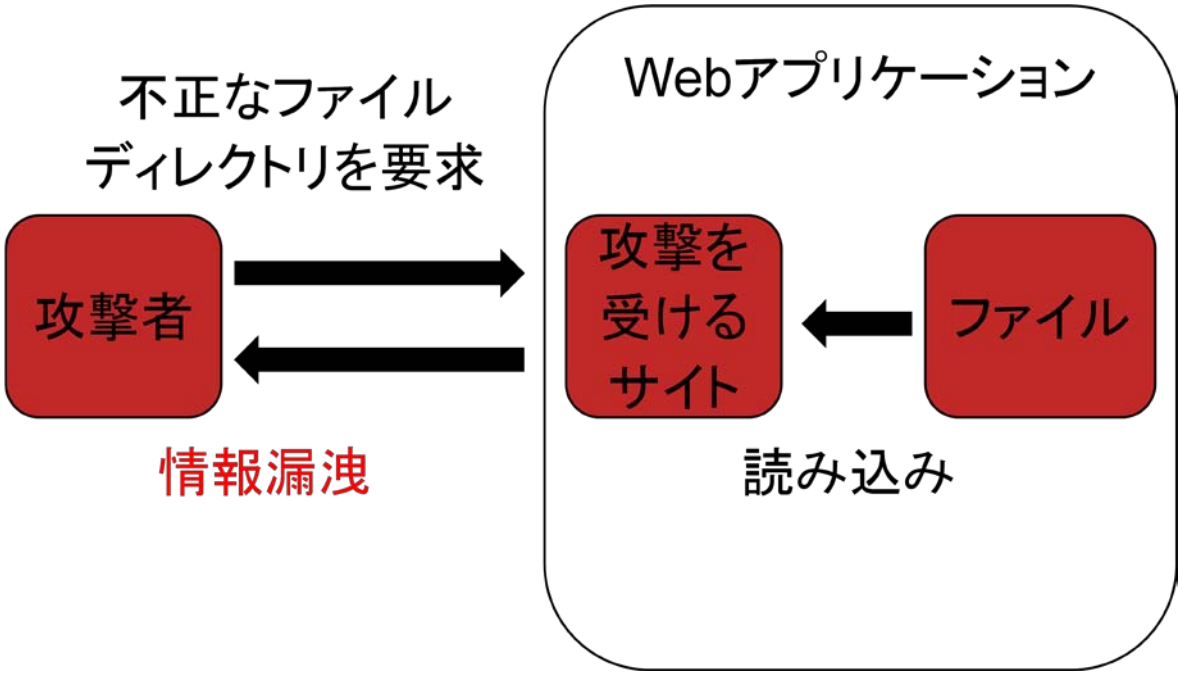
プログラム呼び出しコマンドを使用している

- LAUNCH EXTERNAL PROCESS
- EXECUTE FORMULA
- EXECUTE METHOD

対策

- 可能ならこれらのコマンドを使用しない
- 使用する場合、外部入力値は含めない
    - ユーザ入力値
    - データベースのデータ
    - 等すべて
- EXECUTE XXX は外部データが悪意あるフォーミュラやメソッド名にならないよう手当てする

ディレクトリトラバーサル



対策

- パラメタにファイル名を指定させない
    - エイリアスからたどれるように

- 環境設定の HTML ルートフォルダを必ず設定
- HTML ルート以下の階層に秘密ファイルを入れない
    - TML ルートフォルダの中身は守れない
- システムドキュメントコマンドでファイルをロードするときに注意

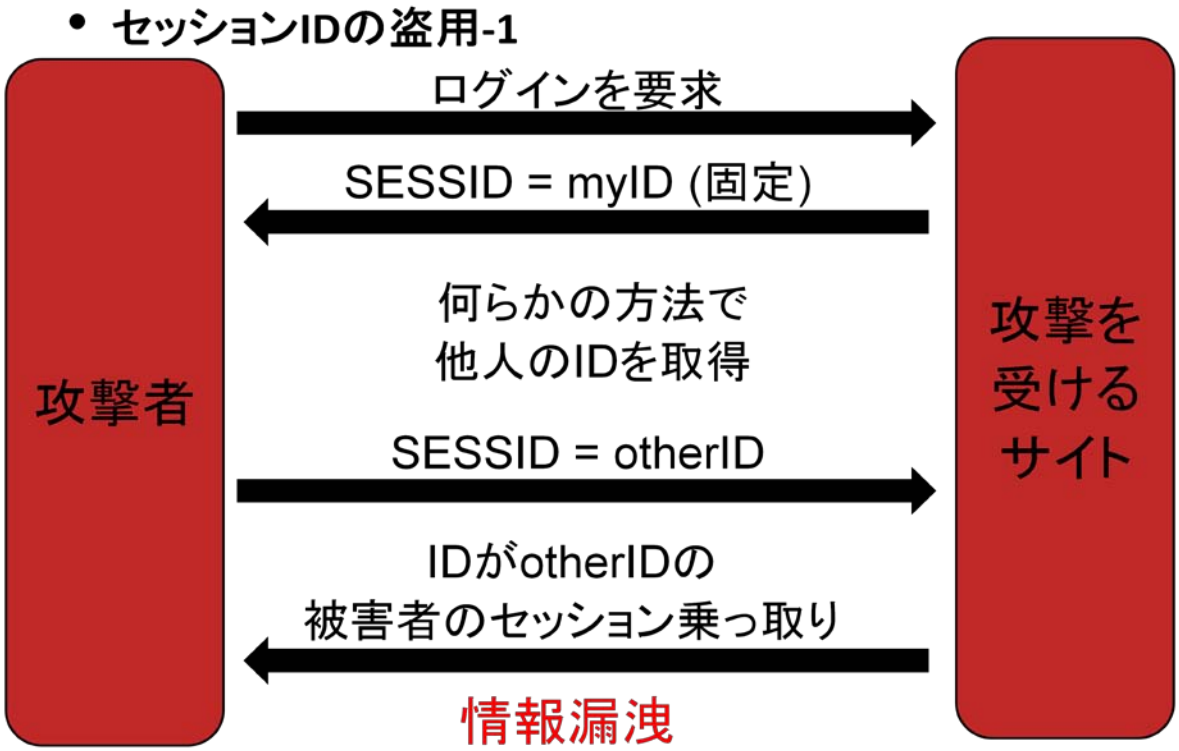
セッション管理の不備

セッション ID の推測

対策

- セッション ID には疑似乱数関数を使用して予測不能な文字列を与える
  例 (あくまで一例です): OpenSSL の乱数およびハッシュ関数を使用
  LAUNCH EXTERNAL PROCESS (″openssl rand 32″;$input;$output;$error)
  LAUNCH EXTERNAL PROCESS (″openssl dgst -sha1″ ;$output;$output;$error)
  $sessid:=Convert to text($output;″UTF-8″)
  Windows には別途バイナリのインストールが必要
  正しく動作しないときがあった

セッション ID の盗用-1



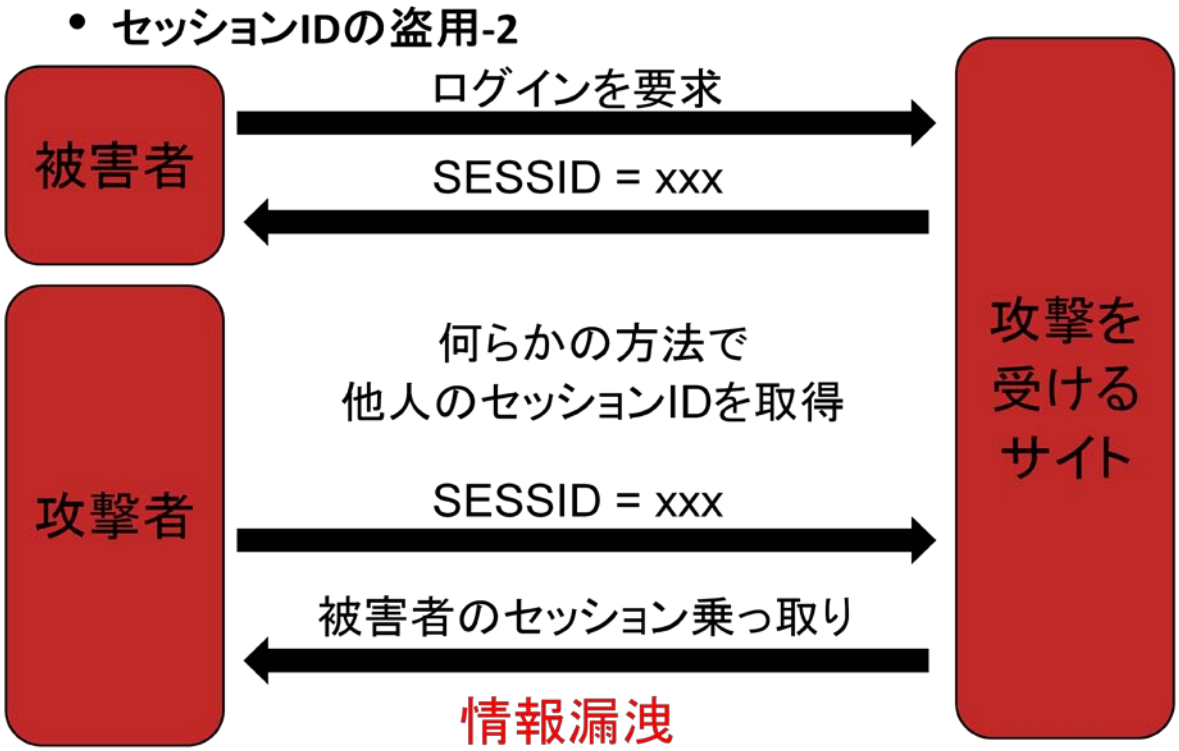
どうやって他人の ID を手に入れるのか

- SQL インジェクション
- 携帯の簡単ログインを使用していて、十分な実装を行っていないサイト
    - 携帯電話の IP をチェックしていない
- ユーザーが DNS Rebinding 攻撃を受ける

対策

- 固定 ID をセッション管理に使用しない
- 携帯の簡単ログインは使用しない
- 脆弱性を排除する

セッション ID の盗用-2



どうやって他人のセッション ID を手に入れるの

- URL にセッション ID が格納されているケース (PHP)
    - 外部リンククリック時にセッション ID が Referer として流出
- 盗聴

盗聴は実際に起こるのか

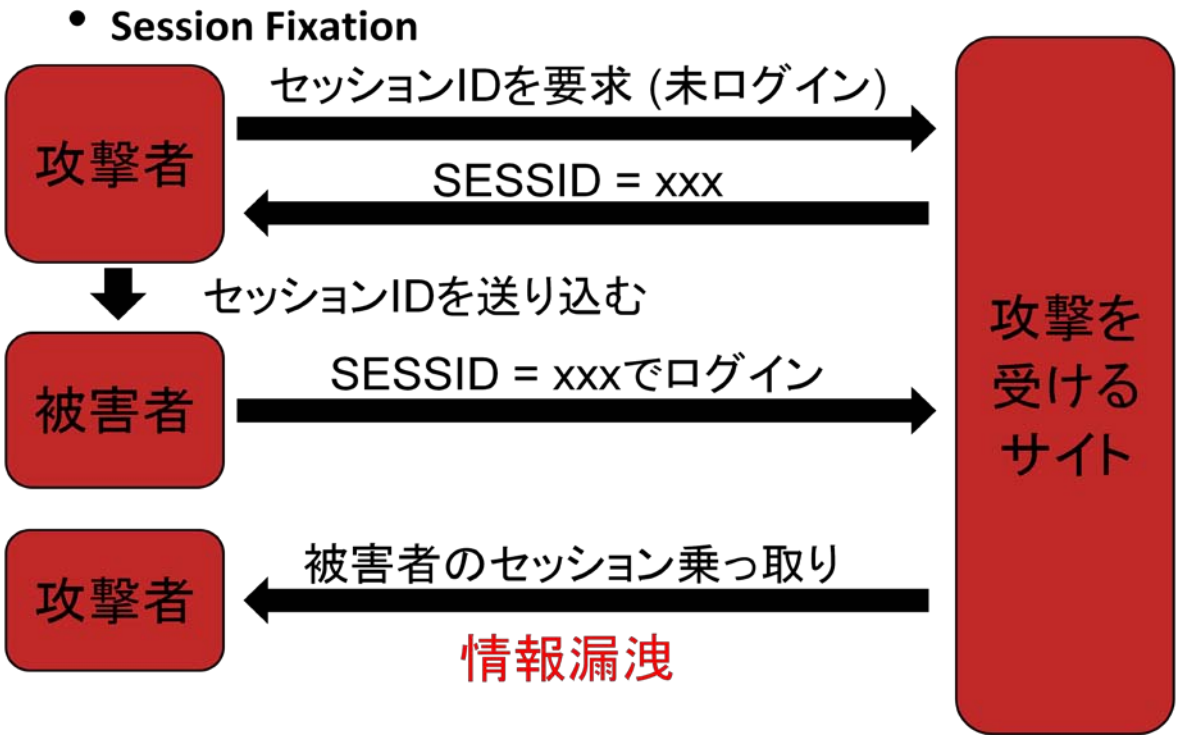
ほんの数年前に実例あり
Web サーバのホスト先で攻撃者のマシンが LAN に収容されていた

Man In The Middle 攻撃
通信 Web サーバーとユーザーの経路途中に攻撃者のマシンが Proxy として入る

対策

- 脆弱性を排除する (XSS 等)
- ログインを伴う (秘密情報のやり取りがある) セッションでは SSL を使用する
    - ログイン画面から SSL であることが重要
- HTTPS で通信する COOKIE に secure 属性をつける
- URL にセッション ID を含めない、または他のサイトへのリンクをページに含めない

Session Fixation



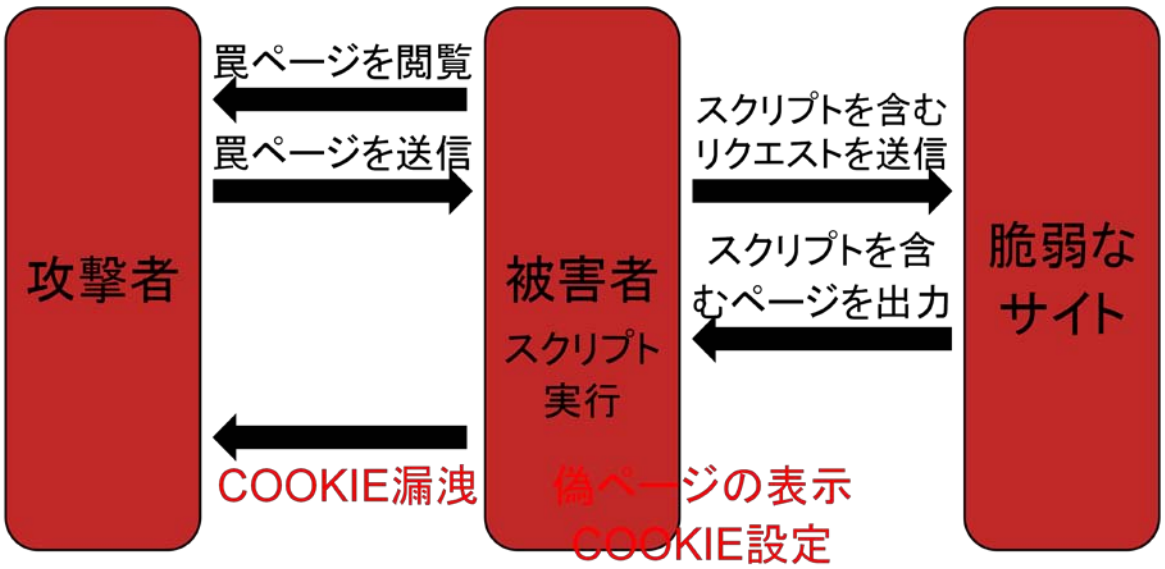
どうやって自分のセッション ID を他人に使わせるのか

- リクエストのデータ部にセッション ID を入れている場合
  - 攻撃を受けるサイトへの罠のリンクをクリックさせたり、フォームをポストさせたりする
- ユーザのブラウザに Cookie Monster の問題がある場合
  - SET-COOKIE: SESSID=xxx; domain=.co.jp;
- Web アプリケーションに Session Adaption の問題がある場合
  - Web アプリケーションに送られてきたセッション ID が存在しないとき、アプリケーションがその ID を使用してしまう
- Web アプリケーションに XSS や HTTP ヘッダインジェクションなどの問題がある場合
  - スクリプトを使用して COOKIE を操作される

対策

- ユーザーがログインしたら新しいセッション ID を発行、以前の ID は無効にする
- XSS 等の脆弱性を排除する

クロスサイトスクリプティング



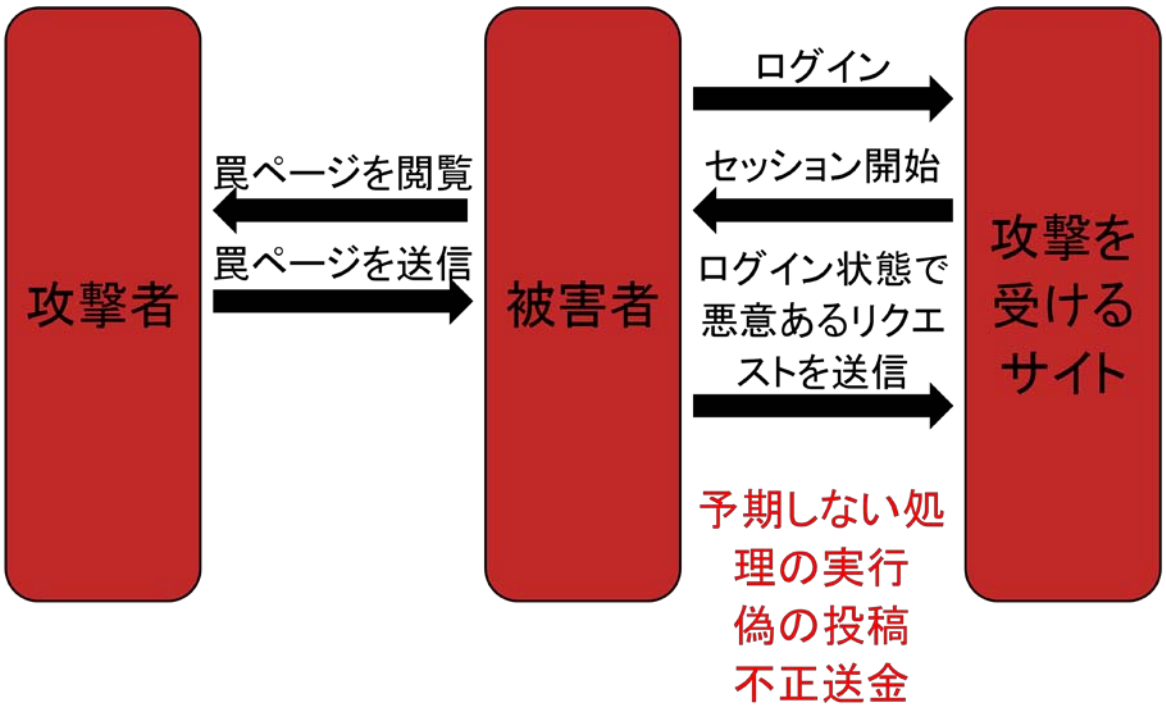
なぜ発生するのか

- データを出力するページがあるとき
  データ出力時に HTML 特別文字をエスケープしない
  - < (<)
  - > (>)
  - & (&)
  - “ (")
  - ‘ (')
- スクリプトを含むデータを被害者が脆弱サイトに送信するよう仕向ける

対策

- データ出力時に HTML 特別文字をエスケープ
  - ユーザ入力値だけでなく、データベースからのデータ、外部ファイルから読み込んだテキストなどすべての出力
- HTML タグを許可する場合は、安全性の確認が必要
- 4DHTMLVAR を使用せずに、4DVAR や 4DSCRIPT を使用して、
  データの先頭から Char(1)を取り除く
  - 4DVAR や 4DSCRIPT はデータが Char(1)から始まっていると、HTML 特別文字をエンコードしない

CSRF



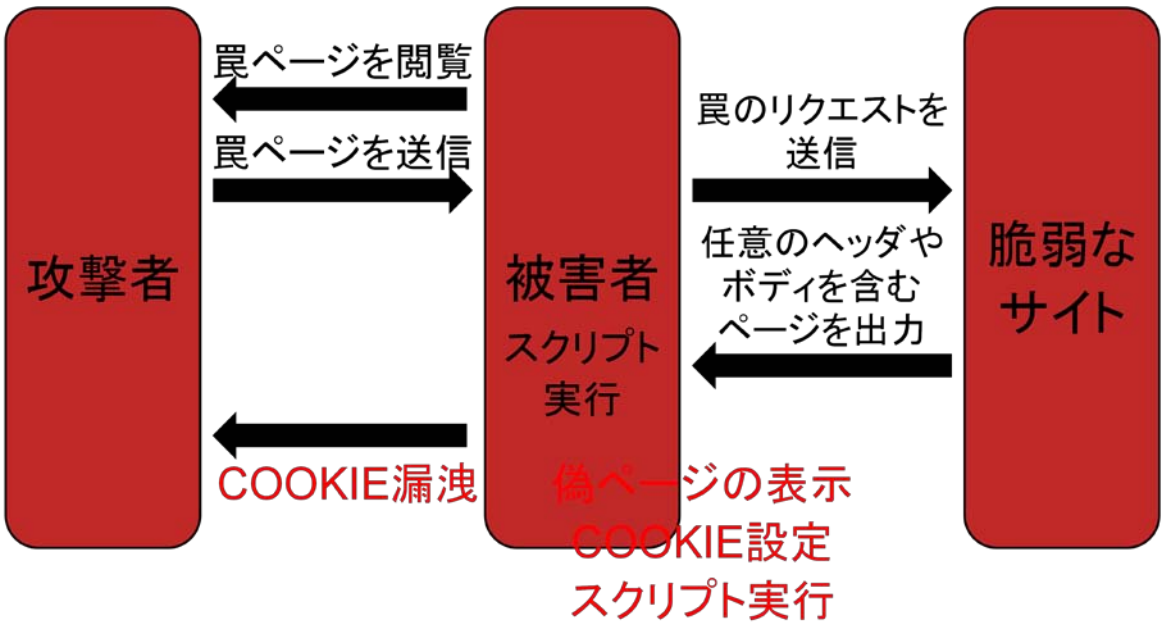
問題は

Web アプリケーションにとって、正当なユーザからの正当なリクエストに見える
ぼくはまちゃん事件など

対策

- 秘密情報を使用した画面遷移の確認
  一連の処理開始画面の hidden パラメタにセッション ID などの秘密情報を入れる
  次のリクエストで正しい秘密情報が送信されているかを確認
- 処理の最終段階でパスワードの入力を求める

HTTP ヘッダインジェクション



なぜ発生するのか

- データから動的にヘッダを生成
- データに CRLF と悪意あるヘッダを注入される
  例
  $header_t:="Location: "+$redirectUri_t
  SET HTTP HEADER($header_t)

  $redirectUri_t はデータが使用される、このとき$header_t に
  http://mysite.com/[CR][LF]Set-Cookie: my_cookie=bad_cookie
  が与えられると、COOKIE が設定されてしまう

対策

- ヘッダをデータから生成しない
- ヘッダに使用するデータに CR や LF が含まれていたら処理を中止

まとめ

まだまだ紹介しきれていないことがたくさんあります。
是非参考資料に目を通すなど、情報収集に努めてください。
設計段階から脆弱性対策を念頭に置いてください。
後から修正するのはとても大変です。