「Javaアプリケーション脆弱性事例調査資料」について

- この資料は、Javaプログラマである皆様に、脆弱性を身近な問題として感じてもらい、セキュアコーディングの重要性を認識していただくことを目指して作成しています。

- 「Javaセキュアコーディングスタンダード CERT/Oracle版」と合わせて、セキュアコーディングに関する理解を深めるためにご利用ください。

JPCERTコーディネーションセンター
セキュアコーディングプロジェクト
secure-coding@jpcert.or.jp

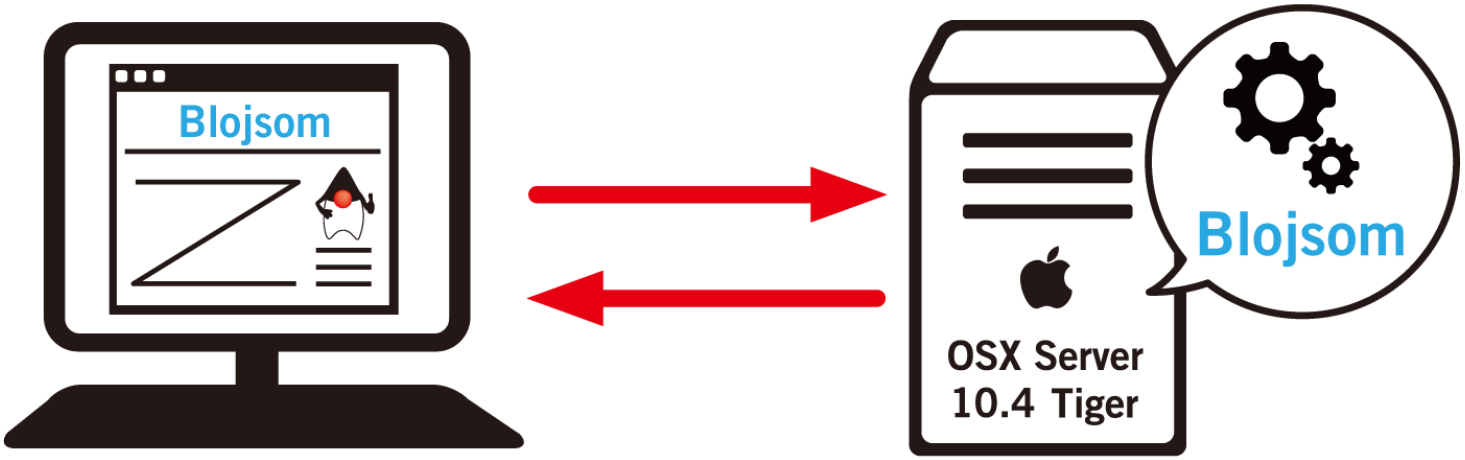
Blojsom におけるクロスサイトスクリプティングの脆弱性

CVE-2006-4829

JVNDB-2006-001260

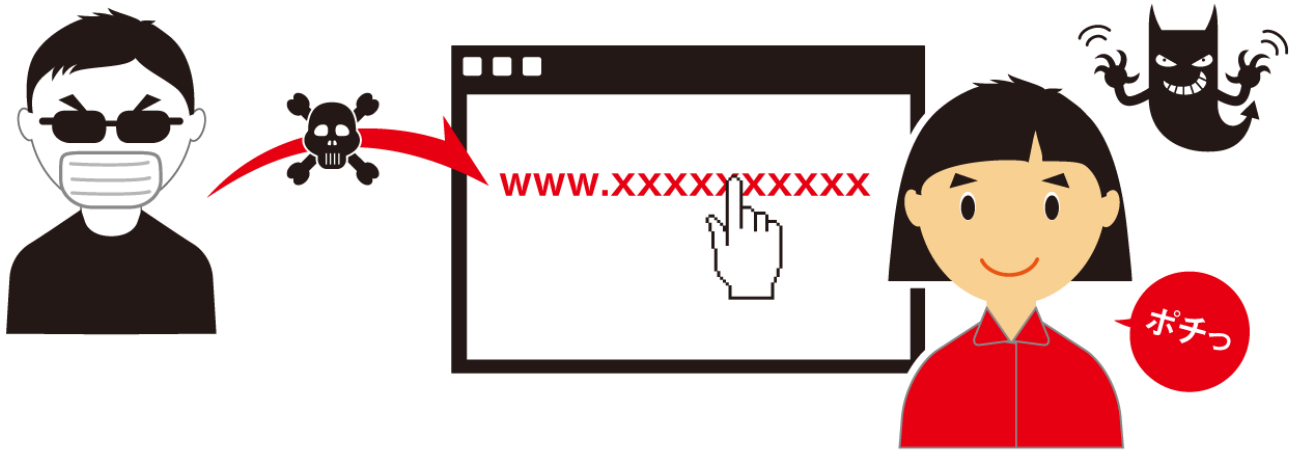
Blojsomとは

- BlojsomはJavaで書かれたブログシステム
- AppleのMac OS X Server 10.4 TigerのWeblog Serverは blojsomがベースとなっている



脆弱性の概要

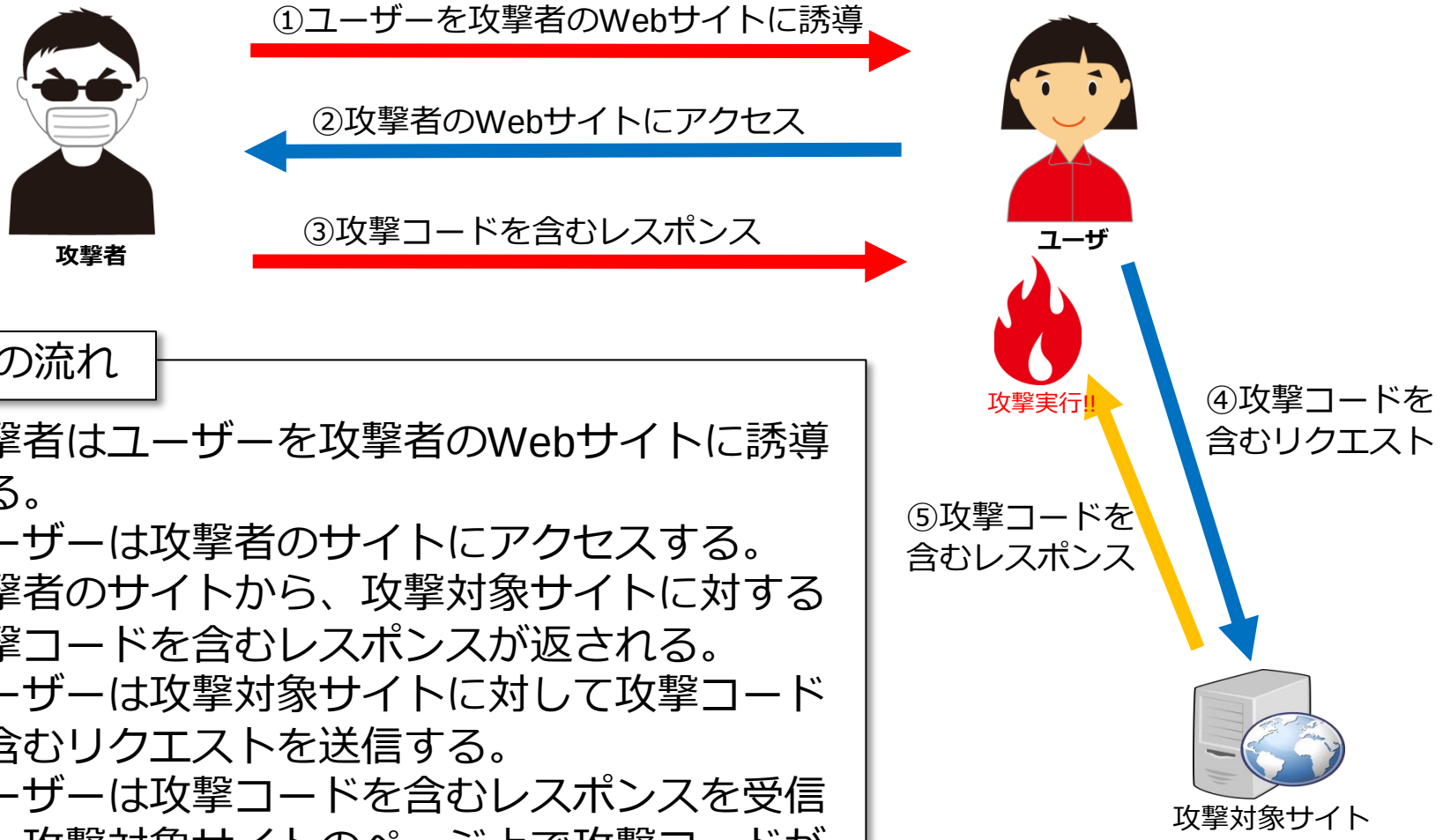
- Blojsom は入力された文字列の処理に不備があり、クロスサイトスクリプティング攻撃が可能
- 攻撃者の用意したリンクをクリックするだけでクロスサイトスクリプティング攻撃が成立する
- クロスサイトスクリプティングにより、アカウント乗っ取り等の被害が発生する



クロスサイトスクリプティングとは

- クロスサイトスクリプティングとは、ユーザの入力値を元にHTMLやURLなどを動的に生成しているWebサイトにおいて、攻撃者によってユーザのブラウザに表示されるコンテンツに任意のHTMLやJavaScriptを埋め込む攻撃。
- クロスサイトスクリプティングにより下記のような被害が発生する。
  - 不正なサイトへの誘導
  - Cookieの搾取によるセッションの乗っ取り
  - 不正プログラムの埋め込み・実行
  - 機密情報の漏えい（HTTPSページの情報搾取やHTMLフォームの偽装等）

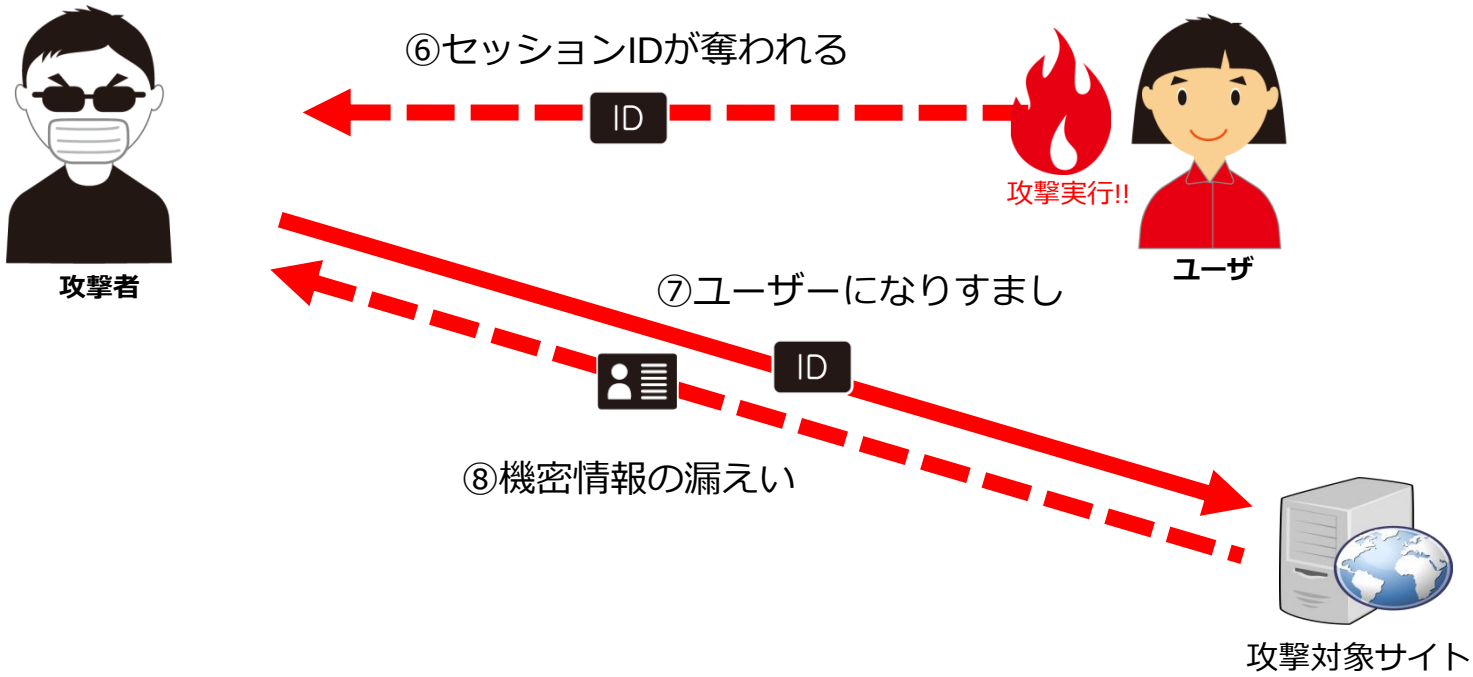
クロスサイトスクリプティング攻撃例



攻撃の流れ

① 攻撃者はユーザーを攻撃者のWebサイトに誘導する。
② ユーザーは攻撃者のサイトにアクセスする。
③ 攻撃者のサイトから、攻撃対象サイトに対する攻撃コードを含むレスポンスが返される。
④ ユーザーは攻撃対象サイトに対して攻撃コードを含むリクエストを送信する。
⑤ ユーザーは攻撃コードを含むレスポンスを受信し、攻撃対象サイトのページ上で攻撃コードが実行される。

クロスサイトスクリプティング攻撃例



攻撃の流れ

⑥ 攻撃コードが実行され、セッションIDが奪われる。
⑦ 攻撃者は奪ったセッションIDを利用して、ユーザーになりすまして攻撃対象サイトにアクセスする。
⑧ 攻撃対象サイト上のユーザー情報を奪うことが可能となる。

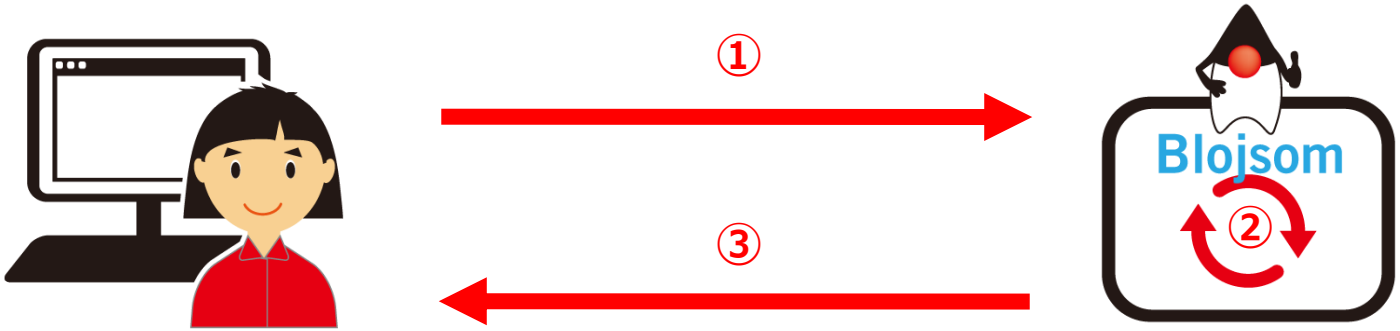
Blojsom の処理内容

ブログに新しい書き込みを実施した場合の処理フロー

① クライアントのブラウザからリクエストが送信され、アプリケーションが受信する。

② リクエストからパラメータを取り出し、処理（ブログの書き込み）を実施。

③ アプリケーションは書き込み結果とともに、書き込み内容をクライアントにレスポンスとして送信し、クライアントのブラウザに表示される



Blojsomの処理: ①クライアントからリクエストを受信する

クライアントから送信されるリクエストは下記のようになる。

※ここではパラメータ “rss-enclosure-url” に注目する。

HTTPリクエスト

```
POST /blojsom/blog/default/ HTTP/1.1
Host: 192.168.xxx.xxx
Content-Length: 370

action=add-blog-entry&flavor=….&rss-enclosure-url=test&…&submit=Add+blog+entry
```

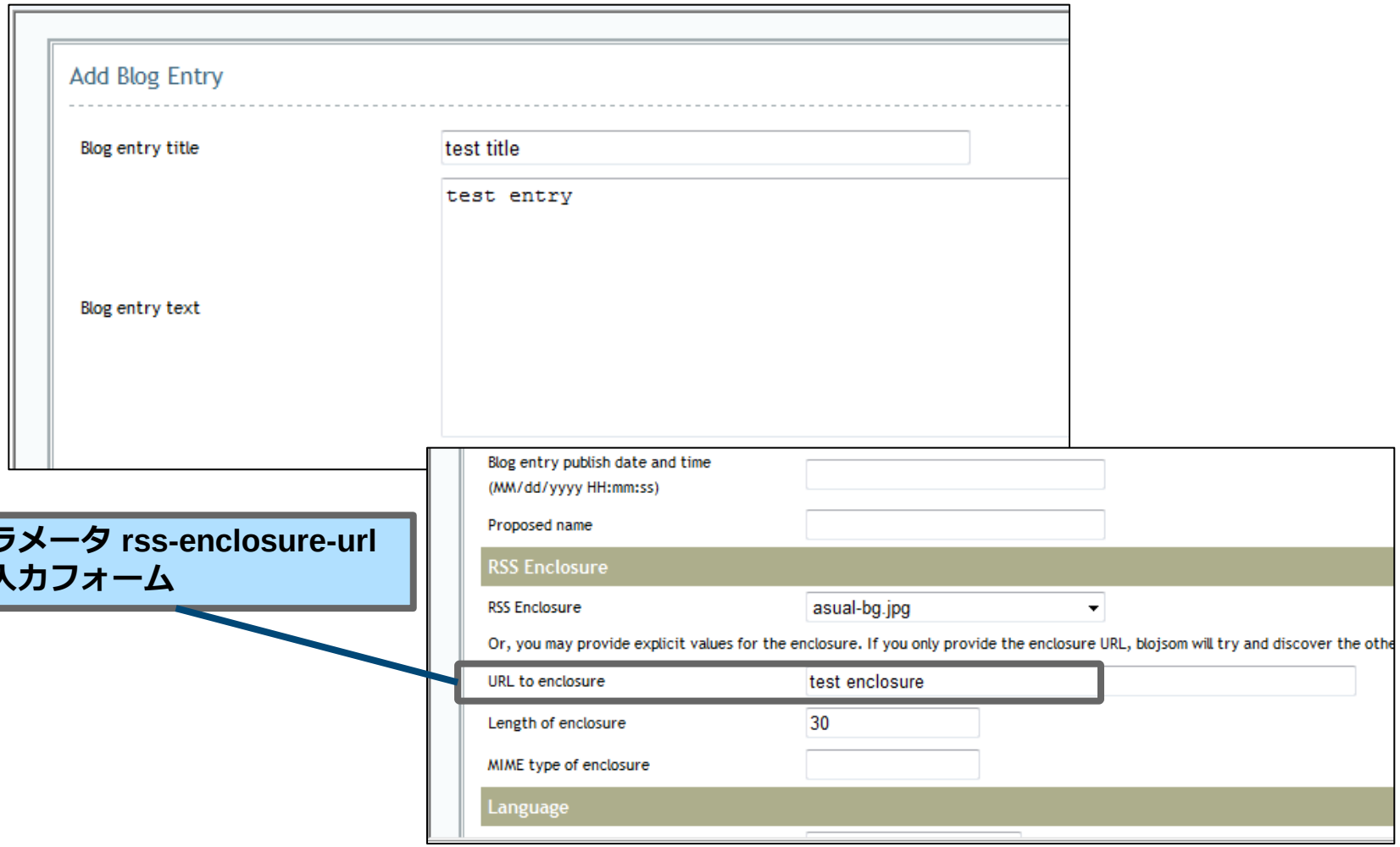
ここではパラメータ rss-enclosure-url の値は test

上記HTTPリクエストを送信するためのHTML

```
<form action=‘/blojsom/blog/default/’ method=‘POST’>
:
<input type=‘text’ name=‘rss-enclosure-url’ value=‘test’>
:
<input type=‘submit’ name=‘post’ value=‘Add blog entry’>
</form>
```

Blojsomの処理: ①クライアントからリクエストを受信する

ブログ書き込み画面



Blojsomの処理: ②リクエストの処理とブログへの書き込み処理

リクエストからパラメータ rss-enclosure-url の値を変数 rssEnclosureURL に格納し、ブログへの書き込み処理を実施する。

RSSEnclosurePlugin.java

```
public class RSSEnclosurePlugin implements BlojsomPlugin, BlojsomListener {
    :
    public static final String RSS_ENCLOSURE_URL = "rss-enclosure-url";
        :
        public void processEvent(BlojsomEvent event) {
                :
                ProcessBlogEntryEvent processBlogEntryEvent =
                (ProcessBlogEntryEvent) event;
                :
                //リクエストからパラメータrss-enclosure-urlの値を取り出す
                String rssEnclosureURL =
                BlojsomUtils.getRequestValue(RSS_ENCLOSURE_URL,
                processBlogEntryEvent.getHttpServletRequest());
                :
                （ブログへの書き込み処理）
                :
```

Blojsomの処理: ③処理結果をクライアントへ返す

書き込み処理を実施後、変数rssEnclosureURLの値をレスポンスとしてクライアントに返す。

RSSEnclosurePlugin.java

```
public class RSSEnclosurePlugin implements BlojsomPlugin, BlojsomListener {
    :
    public void processEvent(BlojsomEvent event) {
        :
        //クライアントに値を返す
        processBlogEntryEvent.getContext().put(RSS_ENCLOSURE_URL_ITEM,
        rssEnclosureURL);
```

Blojsomの処理: ③処理結果をクライアントへ返す

アプリケーションからクライアントへ送信されるHTTPレスポンスは下記のような内容になる。

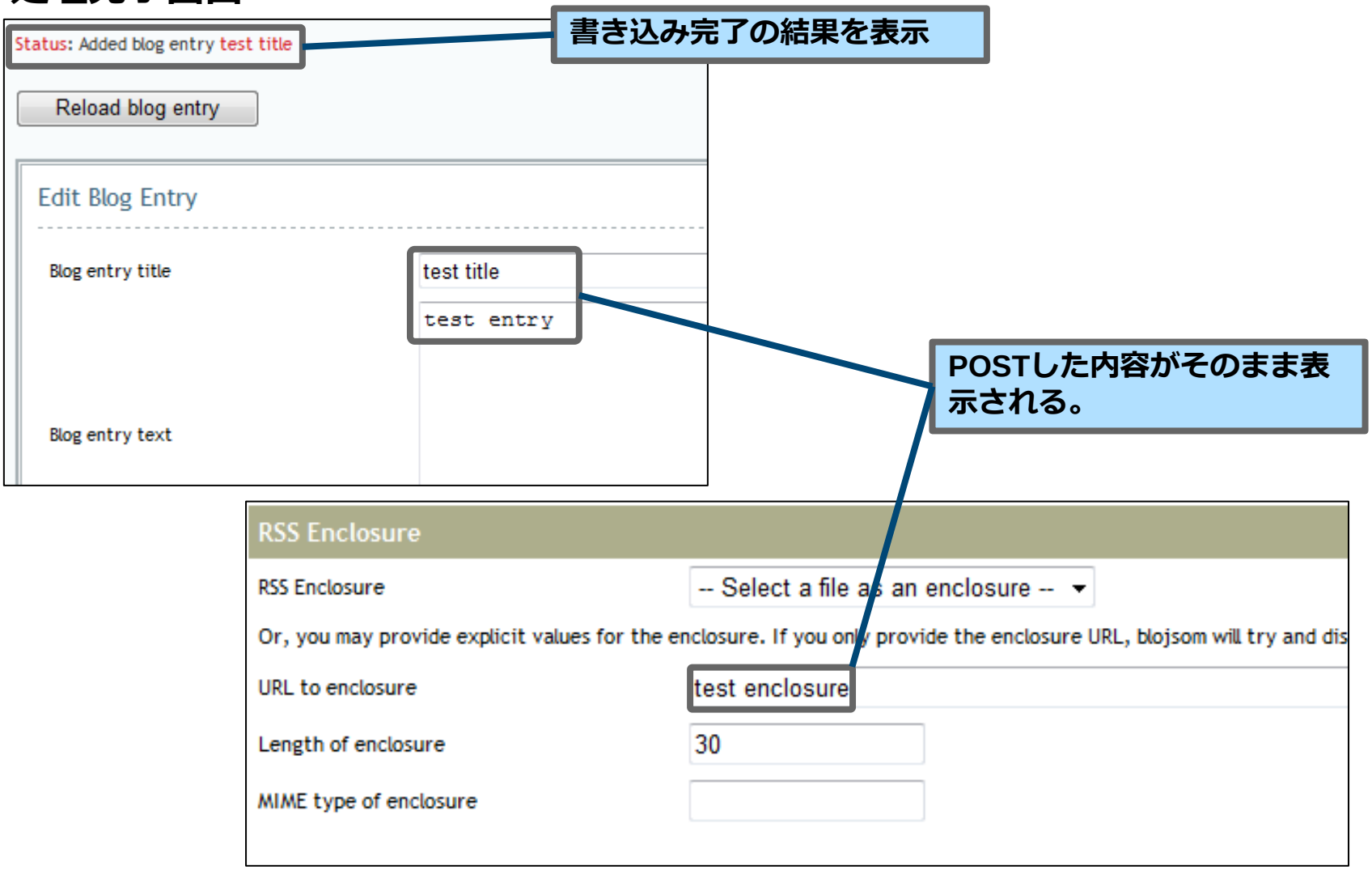
HTTPレスポンス

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Pragma: no-cache
Content-Type: text/html;charset=UTF-8
Date: Fri, 05 Oct 2012 07:29:17 GMT
Content-Length: 26881

<html>
<head>
 :
<tr>
<td>URL to enclosure</td>
<td><input type="text" name="rss-enclosure-url" value="test" size="60" /></td>
</tr>
 :
```

Blojsomの処理: ③処理結果をクライアントへ返す

処理完了画面



攻撃コード

攻撃コードのHTTPリクエスト

POST /blojsom/blog/default/ HTTP/1.1
Host: 192.168.xxx.xxx
Content-Length: 370

action=add-blog-entry&flavor=….&
rss-enclosure-url="><script>alert(123)</script>&…&submit=Add+blog+entry

■攻撃コードのポイント
• パラメータ rss-enclosure-url にスクリプトタグを含む値を送信する。

攻撃コード

攻撃コードのHTTPリクエストを送信するためのHTML

```
<form action='/blojsom/blog/default/' method='POST'>
:
<input type='hidden' name='rss-enclosure-url'
value='"><script>alert(123)</script>'>
:
<input type='submit' name='post' value='Add blog entry'>
</form>
```

被害者にこのフォームのリクエストを送信させることでクロスサイトスクリプティング攻撃が成立する。

攻撃コードが実行された際の処理

ブログに新しい書き込みを実施した場合の処理フロー

① クライアントのブラウザからリクエストが送信され、アプリケーションが受信する。

② リクエストからパラメータを取り出し、処理（ブログの書き込み）を実施。

③ アプリケーションは書き込み結果とともに、書き込み内容をクライアントにレスポンスとして送信し、クライアントのブラウザに表示される

攻撃が実施された際の処理フローは正常な処理とまったく変わらない。
③のアプリケーションから送信されるレスポンス内容に注目する。

攻撃コードが実行された際の処理

アプリケーションからクライアントに返されるHTTPレスポンス

HTTPレスポンス

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
:
<html>
<head>
 :
<tr>
<td>URL to enclosure</td>
 <td><input type="text" name="rss-enclosure-url" value="">
<script>alert(123)</script>"
size="60" /></td>
</tr>
```

```
<td>URL to enclosure</td>
 <td><input type="text" name="rss-enclosure-url" value="test" size="60" /></td>
</tr>
```

通常の処理の際のレスポンス

挿入された文字列により、input タグが閉じられ script タグが挿入された形になる。そのため、ブラウザは script タグ内の Javascript を実行してしまう。

問題点

- 今回のアプリケーションにおける具体的な問題点

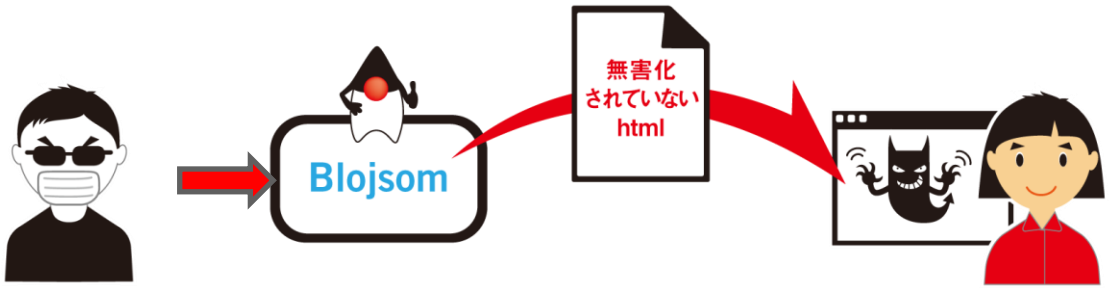
  HTMLへ出力する際に、Javascriptとして解釈される文字列を無害化していないために脆弱性が発生している。

  以下のコーティングガイドに違反している

  「 IDS00-J. 信頼境界を越えて渡される信頼できないデータは無害化する」

- 上記の問題点に対してどうすべきだったか

  Javascriptとして解釈される文字列を無害化してからHTMLに出力すべきであった。



修正版コード

本脆弱性はバージョン2.32で修正されている

ブログに新しい書き込みを実施した場合の処理フロー

① クライアントのブラウザからリクエストが送信され、アプリケーションが受信する。

② リクエストからパラメータを取り出し、処理（ブログの書き込み）を実施。

③ アプリケーションは書き込み結果とともに、書き込み内容をクライアントにレスポンスとして送信し、クライアントのブラウザに表示される

この処理のコードに修正が入っている

修正版コード

RSSEnclosurePlugin.java（修正前）

```
public class RSSEnclosurePlugin implements BlojsomPlugin, BlojsomListener {
    :
    public void processEvent(BlojsomEvent event) {
        :
        processBlogEntryEvent.getContext().put(RSS_ENCLOSURE_URL_ITEM,
        rssEnclosureURL);
```

RSSEnclosurePlugin.java（修正後）

```
public class RSSEnclosurePlugin implements BlojsomPlugin, BlojsomListener {
    :
    public void processEvent(BlojsomEvent event) {
        :
        processBlogEntryEvent.getContext().put(RSS_ENCLOSURE_URL_ITEM,
        BlojsomUtils.escapeBrackets(rssEnclosureURL));
```

クライアントに値を返す際BlojsomUtilsクラスのescapeBracketsメソッドを呼び出している。

修正版コード

BlojsomUtilsクラスのescapeBracketsメソッド

```
public static String escapeBrackets(String input) {
        if (input == null) {
                return null;
        }

        String unescaped = replace(input, "<", "&lt;");
        unescaped = replace(unescaped, ">", "&gt;");

        return unescaped;
}
```

無害化処理として、引数inputの値に含まれる値をエスケープ処理している。

エスケープ対象文字列	エスケープ処理後文字列
<	<
>	>

修正版コード

修正版コードに対して先ほどの攻撃コードを実行すると、下記のようなHTTPレスポンスになり、スクリプトが実行されなくなる。

HTTPレスポンス

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
:
<html>
<head>
 :
<tr>
<td>URL to enclosure</td>
<td><input type="text" name="rss-enclosure-url"
        value=""&gt;&lt;script&gt;alert(123)&lt;/script&gt; size="60" /></td>
</tr>
```

**エスケープ処理により
スクリプトが実行されなくなる。**

実はこの修正では不十分!!

- クロスサイトスクリプティングに対する無害化を行う際に、前述の処理だけでは完全に無害化できないケースが存在する。
- 具体的にはHTMLのタグ内への文字列の出力。
- パラメータrss-enclosure-urlに「“ onfocus=”javascript:alert(1)」という値を挿入することで、入力フォームにマウスカーソルを持っていくとJavascriptが動作する。

パラメータrss-enclosure-urlに「 " onfocus="javascript:alert(1) 」を挿入した際のHTTPレスポンス

```
<td>URL to enclosure</td>
<td>
<input type=“text” name=“rss-enclosure-url” value=“" onfocus="javascript:alert(1)”
size="60" /></td>
</tr>
```

BlojsomUtilsクラスのescapeBracketsメソッドによるエスケープ処理をバイパスできてしまう。

さらに修正

- 次の5種類の文字をHTMLエンコードする

エスケープ対象文字列	エスケープ処理後文字列
<	<
>	>
&	&
“	"
‘	'

- さらにユーザからの入力値をタグの属性値に埋め込む場合は、値全体を「“（ダブルクオート）」文字で括る。
- それにより、「a“ onmouseover=”alert(1);」という文字列を挿入させられた場合でも、次のようなHTMLが生成されるためJavaScriptは実行されない。

```
<input type="text" value="a&quot; onmouseover=&quot;alert(1);">
```

さらに修正

前述を踏まえてescapeBracketsメソッドを修正すると下記のようになる。

BlojsomUtilsクラスのescapeBracketsメソッド（追加修正版）

```
public static String escapeBrackets(String input) {
        if (input == null) {
                return null;
        }

        String unescaped = replace(input, "<", "&lt;");
        unescaped = replace(unescaped, ">", "&gt;");

        unescaped = replace(unescaped, " & ", "&amp;");
        unescaped = replace(unescaped, " ¥" ", "&quot;");
        unescaped = replace(unescaped, " ’ ", "&#39;");

        return unescaped;
}
```

追加されたコード。
３つの特殊記号に対する
エスケープ処理を追記する。

まとめ

- この脆弱性から学べるプログラミングの注意点
  - •アプリケーションの外部にデータを出力する場合、出力先でデータがどのように使用されるかを考慮すべきだった
- 上記への対策
  - • データの出力先がWebブラウザの場合、想定外の個所にHTML のメタ文字が含まれている場合に備えてHTMLエンコーディングを施す

【参考】
OWASP XSS Filter Evasion Cheat Sheet
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

XSS (Cross Site Scripting) Prevention Cheat Sheet
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

著作権・引用や二次利用について

■本資料の著作権はJPCERT/CCに帰属します。

■本資料あるいはその一部を引用・転載・再配布する際は、引用元名、資料名および URL の明示をお願いします。

記載例

引用元：一般社団法人JPCERTコーディネーションセンター
Java アプリケーション脆弱性事例解説資料
Blojsom におけるクロスサイトスクリプティングの脆弱性
https://www.jpcert.or.jp/securecoding/2012/No.03_Blojsom.pdf

■本資料を引用・転載・再配布をする際は、引用先文書、時期、内容等の情報を、JPCERT コーディネーションセンター広報(office@jpcert.or.jp)までメールにてお知らせください。なお、この連絡により取得した個人情報は、別途定めるJPCERT コーディネーションセンターの「プライバシーポリシー」に則って取り扱います。

本資料の利用方法等に関するお問い合わせ

JPCERTコーディネーションセンター
広報担当
E-mail：office@jpcert.or.jp

本資料の技術的な内容に関するお問い合わせ

JPCERTコーディネーションセンター
セキュアコーディング担当
E-mail：secure-coding@jpcert.or.jp