Cross-Site Request Forgery: Danger, Detection, and Defenses

Eric Sheridan
Aspect Security, Inc.
eric.sheridan@aspectsecurity.com

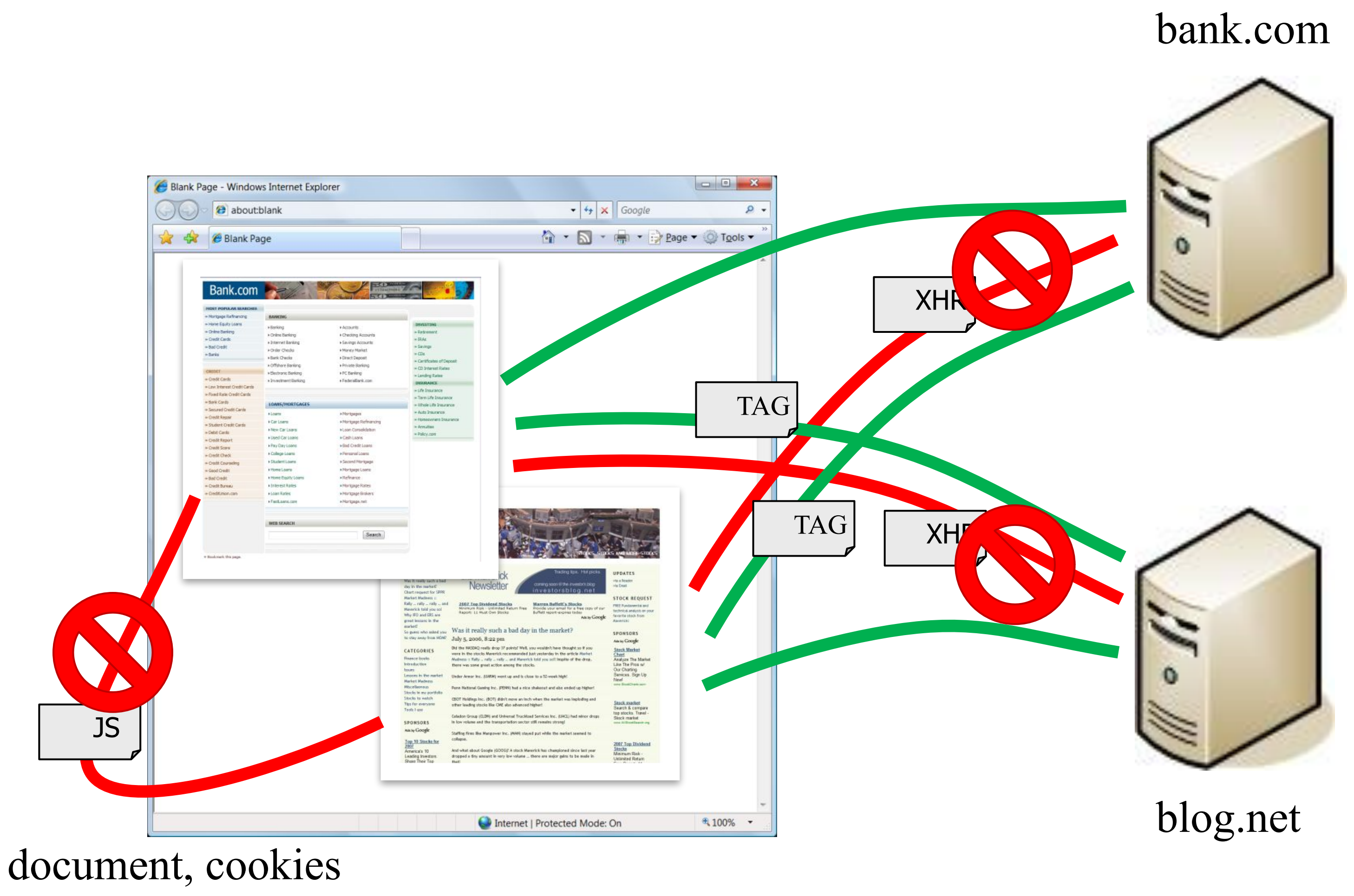
OWASP
11-14-2007

The OWASP Foundation
http://www.owasp.org

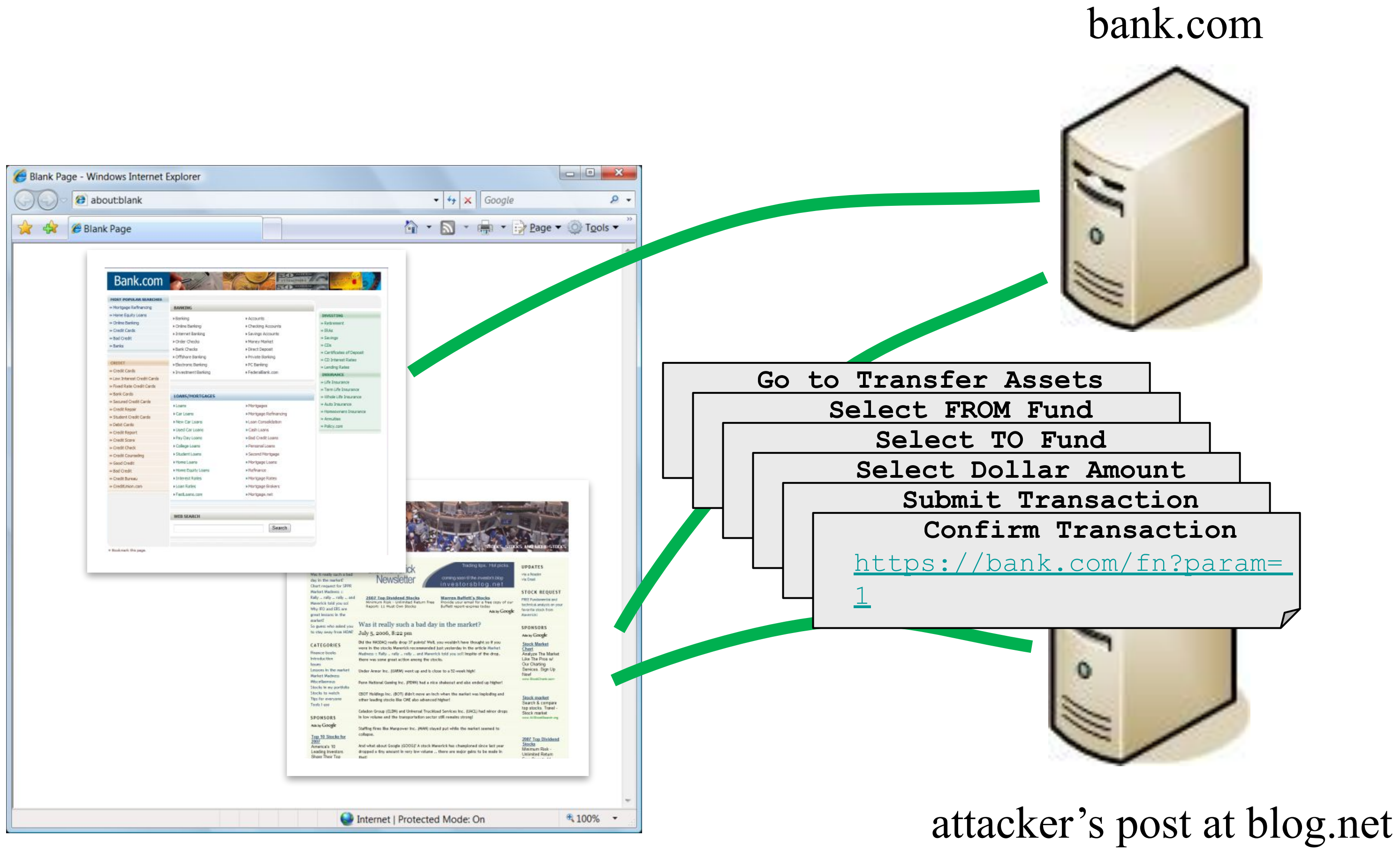
Overview

- Discussion of the “Same Origin Policy”
- Overview of the “Sleeping Giant”
- The Introduction of
  - 2 New OWASP Tools
  - A Series of New WebGoat Labs
- Enterprise CSRF Mitigation Strategy

The Browser "Same Origin" Policy


bank.com
blog.net
document, cookies
JS
TAG
TAG
XHR
XHR
Blank Page - Windows Internet Explorer
about:blank
Bank.com
Internet | Protected Mode: On

Cross-Site Request Forgery



How Does CSRF Work?

< Tags

```
<img src="https://bank.com/fn?param=1">
<iframe src="https://bank.com/fn?param=1">
<script src="https://bank.com/fn?param=1">
```

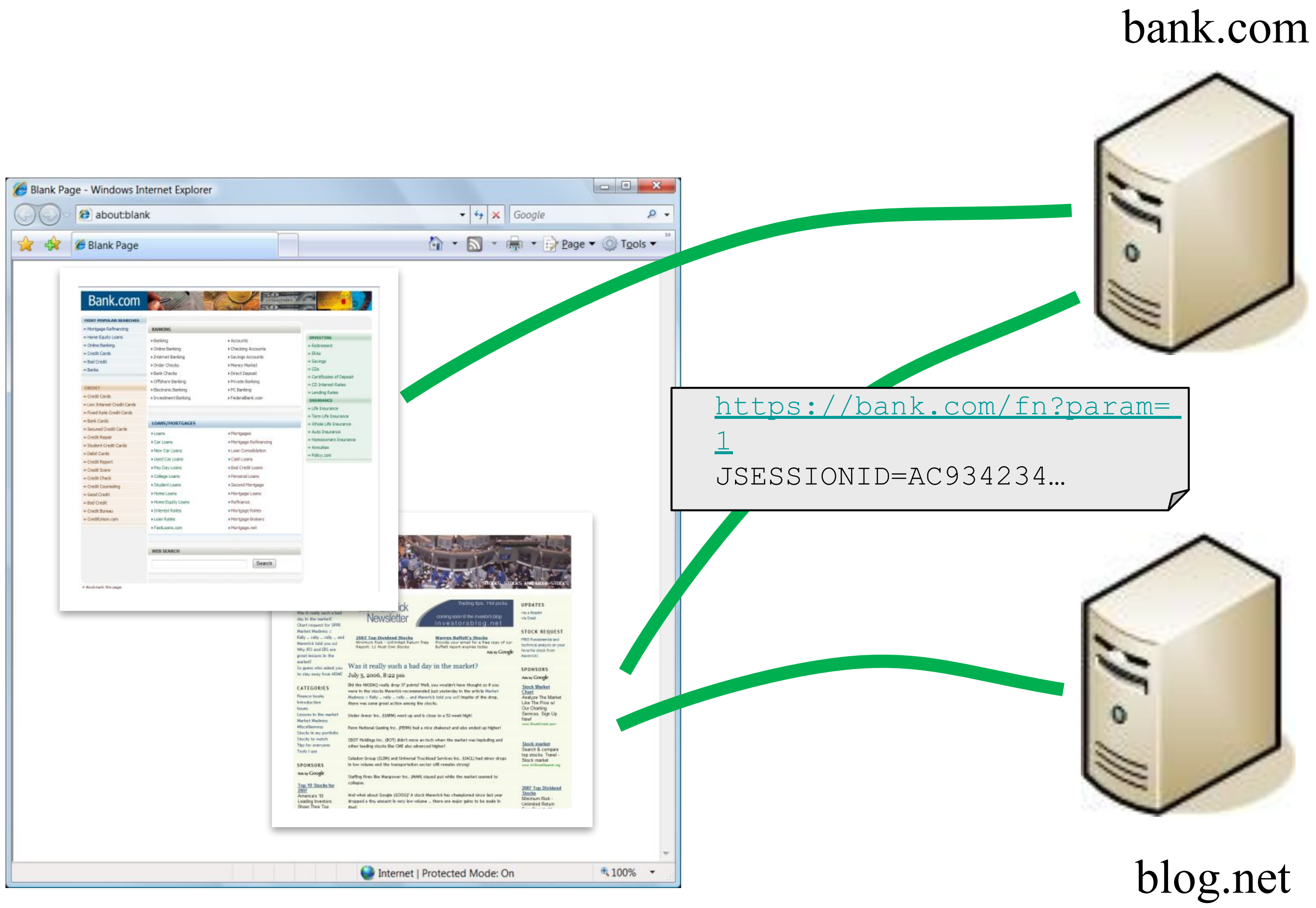
< Autoposting Forms

```
<body onload="document.forms[0].submit()">
<form method="POST" action="https://bank.com/fn">
   <input type="hidden" name="sp" value="8109"/>
</form>
```

< XmlHttpRequest

4 Subject to same origin policy

Credentials Included

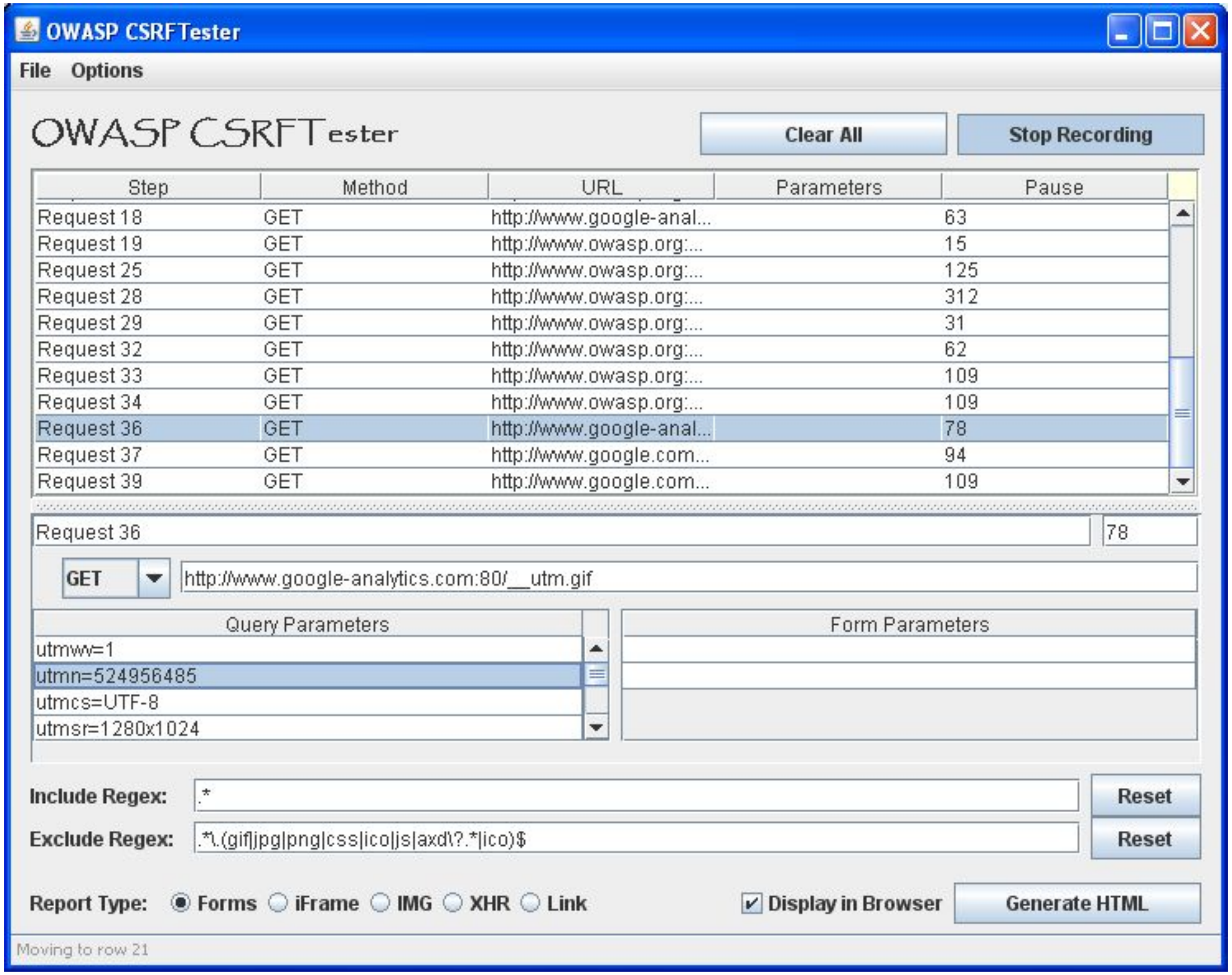

bank.com
Blank Page - Windows Internet Explorer
about:blank
Blank Page
Bank.com
Internet | Protected Mode: On
100%
https://bank.com/fn?param=1
JSESSIONID=AC934234…
blog.net

New Tool: OWASP CSRFTester

- Test your applications for CSRF
  - Record and replay transactions
  - Tune the recorded test case
  - Run test case with exported HTML document

- Test case alternatives
  - Auto-Posting Forms
  - Evil iFrame
  - IMG Tag
  - XMLHTTPRequest
  - Link

DEMO: OWASP CSRFTester



Step	Method	URL	Parameters	Pause
Request 18	GET	http://www.google-anal...		63
Request 19	GET	http://www.owasp.org:...		15
Request 25	GET	http://www.owasp.org:...		125
Request 28	GET	http://www.owasp.org:...		312
Request 29	GET	http://www.owasp.org:...		31
Request 32	GET	http://www.owasp.org:...		62
Request 33	GET	http://www.owasp.org:...		109
Request 34	GET	http://www.owasp.org:...		109
Request 36	GET	http://www.google-anal...		78
Request 37	GET	http://www.google.com...		94
Request 39	GET	http://www.google.com...		109

Request 36 — 78

GET — http://www.google-analytics.com:80/__utm.gif

Query Parameters	Form Parameters
utmwv=1	
utmn=524956485	
utmcs=UTF-8	
utmsr=1280x1024	

Include Regex: .* [Reset]

Exclude Regex: .*\.(gif|jpg|png|css|ico|js|axd\?.*|ico)$ [Reset]

Report Type: [● Forms] ○ iFrame ○ IMG ○ XHR ○ Link

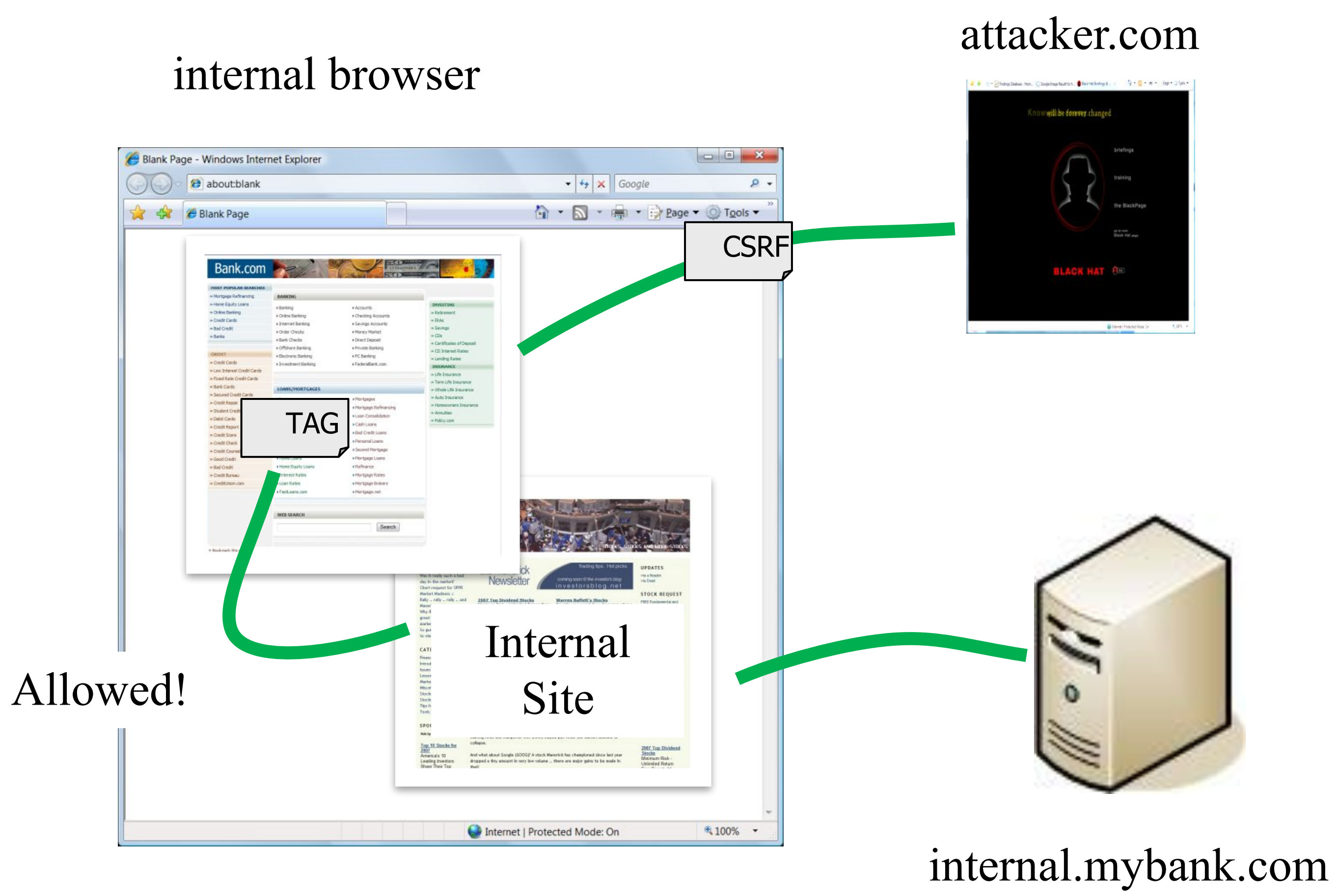
[✓] Display in Browser [Generate HTML]

Moving to row 21

What Can Attackers Do with CSRF?

- <u>Anything</u> an authenticated user can do
  - Click links
  - Fill out and submit forms
  - Follow all the steps of a wizard interface
- No restriction from same origin policy, except…
  - Attackers cannot read responses from other origins
  - Limited on what can be done with data
- Severe impact on accountability
  - Log entries reflect the actions a victim was tricked into executing

Using CSRF to Attack Internal Pages

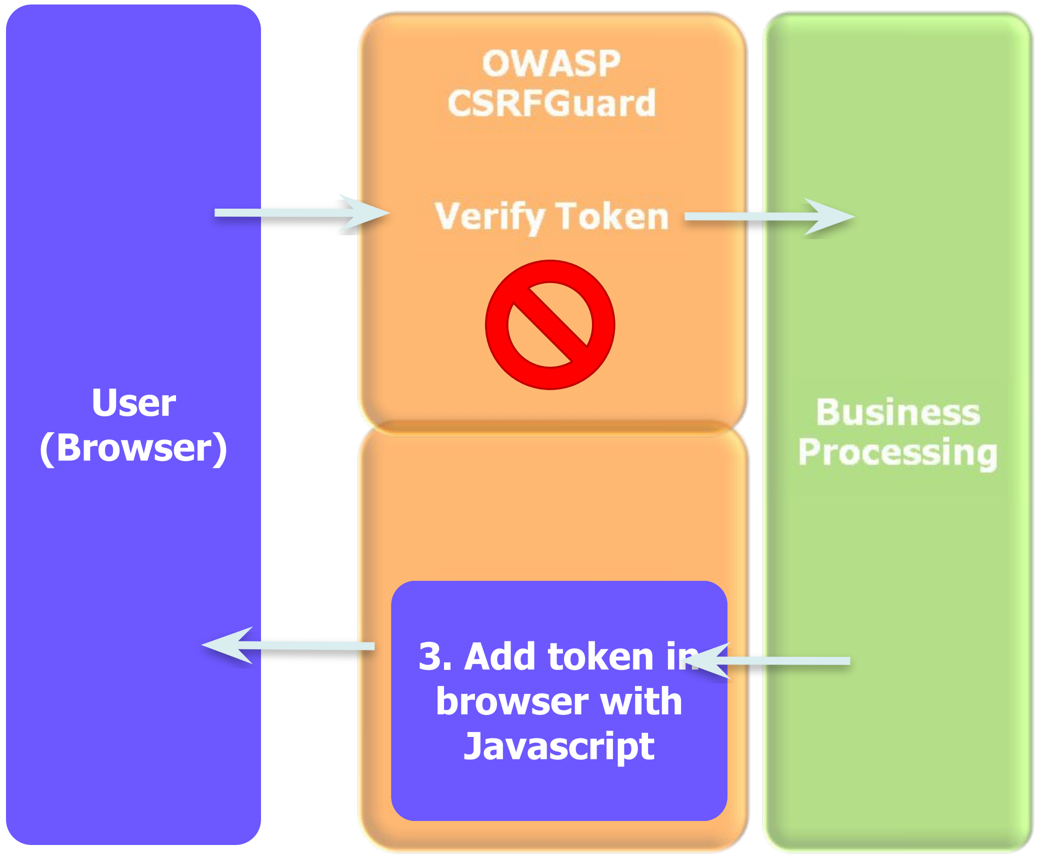


Misconceptions – Defenses That Don't Work

- Only accept POST
  - Stops simple link-based attacks (IMG, frames, etc.)
  - But hidden POST requests can be created with frames, scripts, etc…
- Referer checking
  - Some users prohibit referers, so you can't just require referer headers
  - Techniques to selectively create HTTP request without referers exist
- Requiring multi-step transactions
  - CSRF attack can perform each step in order
- URL Rewriting
  - General session id exposure in logs, cache, etc.

None of these approaches will sufficiently protect against CSRF!

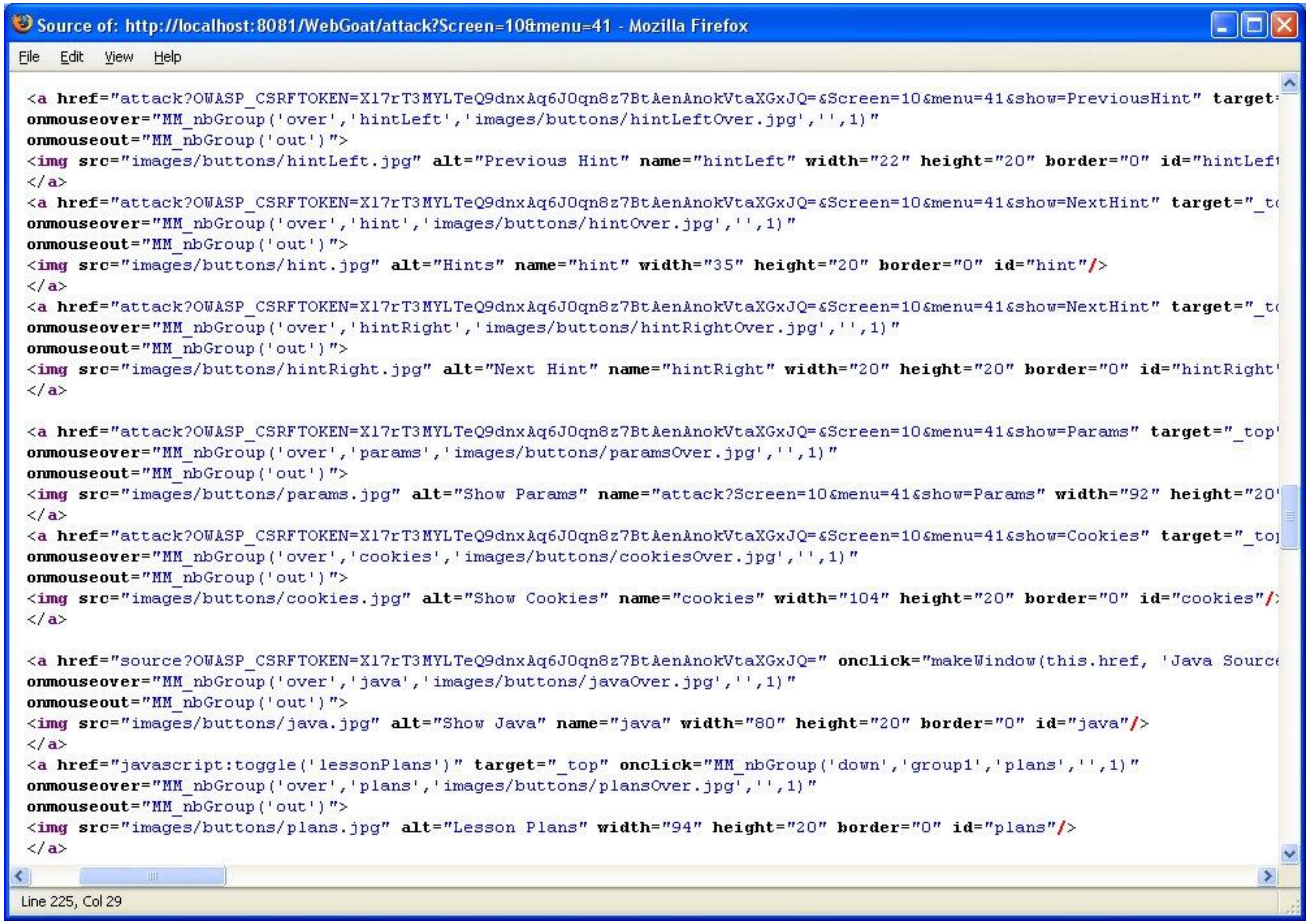
New Tool: OWASP CSRFGuard 2.0



- Adds token to:
  - href attribute
  - src attribute
  - hidden field in all forms
- Actions:
  - Log
  - Invalidate
  - Redirect

http://www.owasp.org/index.php/CSRFGuard

DEMO: OWASP CSRFGuard 2.0


Source of: http://localhost:8081/WebGoat/attack?Screen=10&menu=41 - Mozilla Firefox
File Edit View Help

```
<a href="attack?OWASP_CSRFTOKEN=X17rT3MYLTeQ9dnxAq6J0qn8z7BtAenAnokVtaXGxJQ=&Screen=10&menu=41&show=PreviousHint" target=
onmouseover="MM_nbGroup('over','hintLeft','images/buttons/hintLeftOver.jpg','',1)"
onmouseout="MM_nbGroup('out')">
<img src="images/buttons/hintLeft.jpg" alt="Previous Hint" name="hintLeft" width="22" height="20" border="0" id="hintLef
</a>
<a href="attack?OWASP_CSRFTOKEN=X17rT3MYLTeQ9dnxAq6J0qn8z7BtAenAnokVtaXGxJQ=&Screen=10&menu=41&show=NextHint" target="_t
onmouseover="MM_nbGroup('over','hint','images/buttons/hintOver.jpg','',1)"
onmouseout="MM_nbGroup('out')">
<img src="images/buttons/hint.jpg" alt="Hints" name="hint" width="35" height="20" border="0" id="hint"/>
</a>
<a href="attack?OWASP_CSRFTOKEN=X17rT3MYLTeQ9dnxAq6J0qn8z7BtAenAnokVtaXGxJQ=&Screen=10&menu=41&show=NextHint" target="_t
onmouseover="MM_nbGroup('over','hintRight','images/buttons/hintRightOver.jpg','',1)"
onmouseout="MM_nbGroup('out')">
<img src="images/buttons/hintRight.jpg" alt="Next Hint" name="hintRight" width="20" height="20" border="0" id="hintRight
</a>

<a href="attack?OWASP_CSRFTOKEN=X17rT3MYLTeQ9dnxAq6J0qn8z7BtAenAnokVtaXGxJQ=&Screen=10&menu=41&show=Params" target="_top
onmouseover="MM_nbGroup('over','params','images/buttons/paramsOver.jpg','',1)"
onmouseout="MM_nbGroup('out')">
<img src="images/buttons/params.jpg" alt="Show Params" name="attack?Screen=10&menu=41&show=Params" width="92" height="20
</a>
<a href="attack?OWASP_CSRFTOKEN=X17rT3MYLTeQ9dnxAq6J0qn8z7BtAenAnokVtaXGxJQ=&Screen=10&menu=41&show=Cookies" target="_to
onmouseover="MM_nbGroup('over','cookies','images/buttons/cookiesOver.jpg','',1)"
onmouseout="MM_nbGroup('out')">
<img src="images/buttons/cookies.jpg" alt="Show Cookies" name="cookies" width="104" height="20" border="0" id="cookies"/
</a>

<a href="source?OWASP_CSRFTOKEN=X17rT3MYLTeQ9dnxAq6J0qn8z7BtAenAnokVtaXGxJQ=" onclick="makeWindow(this.href, 'Java Sourc
onmouseover="MM_nbGroup('over','java','images/buttons/javaOver.jpg','',1)"
onmouseout="MM_nbGroup('out')">
<img src="images/buttons/java.jpg" alt="Show Java" name="java" width="80" height="20" border="0" id="java"/>
</a>
<a href="javascript:toggle('lessonPlans')" target="_top" onclick="MM_nbGroup('down','group1','plans','',1)"
onmouseover="MM_nbGroup('over','plans','images/buttons/plansOver.jpg','',1)"
onmouseout="MM_nbGroup('out')">
<img src="images/buttons/plans.jpg" alt="Lesson Plans" width="94" height="20" border="0" id="plans"/>
</a>
```

Line 225, Col 29

Similar Implementations

- PHP CSRFGuard
  - PHP Implementation of CSRFGuard
  - http://www.owasp.org/index.php/PHP_CSRF_Guard

- JSCK
  - PHP & JavaScript implementation
  - http://www.thespanner.co.uk/2007/10/19/jsck/

DEMO: Cross-Site Scripting vs. CSRFGuard

Enterprise CSRF Mitigation Strategy

< Balance Between Security, Usability, and Cost

MISSION CRITICAL FUNCTIONS	• Challenge Response • One-Time Token • CAPTCHA • Transaction Signing
EVERYDAY BUSINESS FUNCTIONS	• Unique Request Tokens • Unique URL Token
LUNCH MENU FUNCTIONS	• Worth the time and money?

http://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/articles/csrf-faq.shtml
http://www.darkreading.com/document.asp?doc_id=107651&WT.svl=news1_2

Extra: How Widespread Are CSRF Holes?

< Very likely in most web applications

  4 Including both intranet and external apps

  4 Including Web 1.0 and Web 2.0 applications

  4 Any function without specific CSRF defenses is vulnerable

< How do victims get attacked?

  4 Victim simply opens an infected webpage, HTML file, or email

  4 Single Sign On (SSO) extends “authenticated user”

< CSRF recently found in 8 security appliances

  4 Including CheckPoint

Extra: Real World CSRF Examples

tlhIngan Hol

```
<iframe style="display:none"
 src="http://www.google.com/setpre
 fs?hl=xx-klingon&amp;submit2=Save
 %20Preferences%20&amp;prev=http:/
 /www.google.com/&q=&submit=
 Save%20Preferences%20"></iframe>
```

```
<img
  src=http://www.netflix.com/AddTo
  Queue?
  movieid=70011204 width="1"
  height="1" border="0">
```

Extra: CSRF Defenses

- CAPTCHA
  - Attacker must know CAPTCHA answer
  - Assuming a secure implementation
- Re-Authentication
  - Password Based
    - Attacker must know victims password
    - If password is known, then game over already!
  - One-Time Token
    - Attacker must know current token
    - Very strong defense!
- Unique Request Tokens
  - Attacker must know unique request token for particular victim for particular session
  - Assumes token is cryptographically secure and not disclosed.
    - `/accounts?auth=687965fdfaew87agrde …`

Goat Hills Financial
Human Resources

Please Login
Larry Stooge (employee)
Password
Login