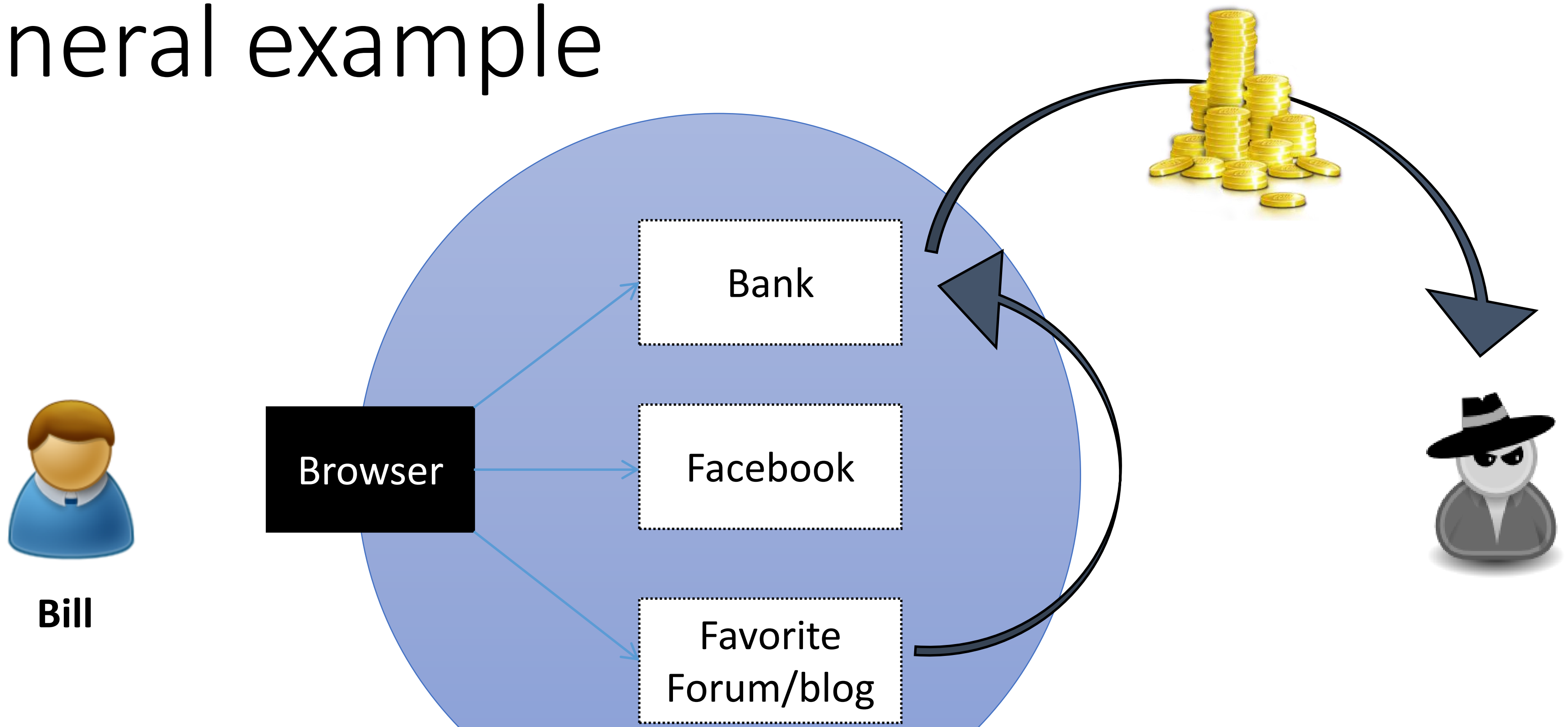
COSC 458-647
Application Software Security

Cross-site Request Forgery (CSRF)

Introduction (OWASP)

- CSRF is an attack which ***forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated***.
- With a little help of social engineering (like sending a link via email/chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing.
- A successful CSRF exploit can compromise end user data and operation in case of normal user. If the targeted end user is the administrator account, this can compromise the entire web application.

A general example



http://mybank.com/showaccount?id=bill

http://mybank.com/withdraw?from=bill&amount=10000&for=someguy

<img src=http://mybank.com/withdraw?from=bill&amount=10000&for=someguy />

Example

- There are numerous ways in which an end-user can be tricked into loading information from or submitting information to a web application.
- In order to execute an attack, we must first understand how to generate a malicious request for our victim to execute.
- Example: Alice wishes to transfer $100 to Bob using bank.com

The legal request generated by Alice will look similar to the following

```
POST http://bank.com/transfer.do HTTP/1.1
...
...
...
Content-Length: 19;
acct=BOB&amount=100
```

Example

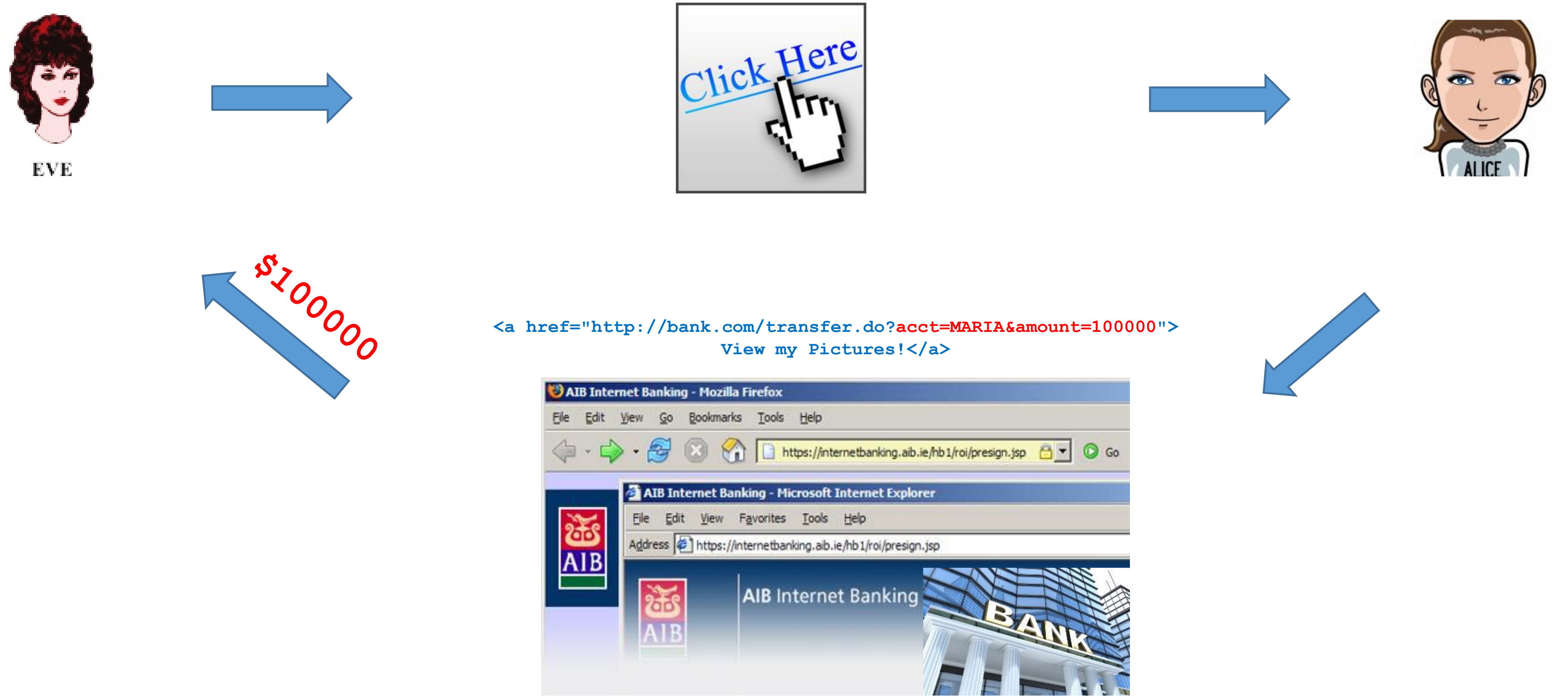
- However, Eve notices that the same web application will execute the same transfer using URL parameters as follows

```
GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1
```

- Eve now decides to exploit this web application vulnerability using Alice as her victim.
    - Eve first constructs the following URL which will ***transfer $100,000 from Alice's account to her account***

Example

```
<a href="http://bank.com/transfer.do?acct=MARIA&amount=100000">
View my Pictures!</a>
```


EVE
Click Here
ALICE
$100000
AIB Internet Banking - Mozilla Firefox
https://internetbanking.aib.ie/hb1/roi/presign.jsp
AIB Internet Banking - Microsoft Internet Explorer
AIB Internet Banking
BANK

```
<a href="http://bank.com/transfer.do?acct=MARIA&amount=100000">
View my Pictures!</a>
```

Example

- Assuming Alice is authenticated with the application when she clicks the link, the transfer of $100,000 to Maria's account will occur.
- However, Maria realizes that if Alice clicks the link, then Alice will notice that a transfer has occurred.

```
HTTP/1.1 200 OK.
Transfer amount=100000 complete!
```

- Therefore, Maria decides to hide the attack in a zero-byte image

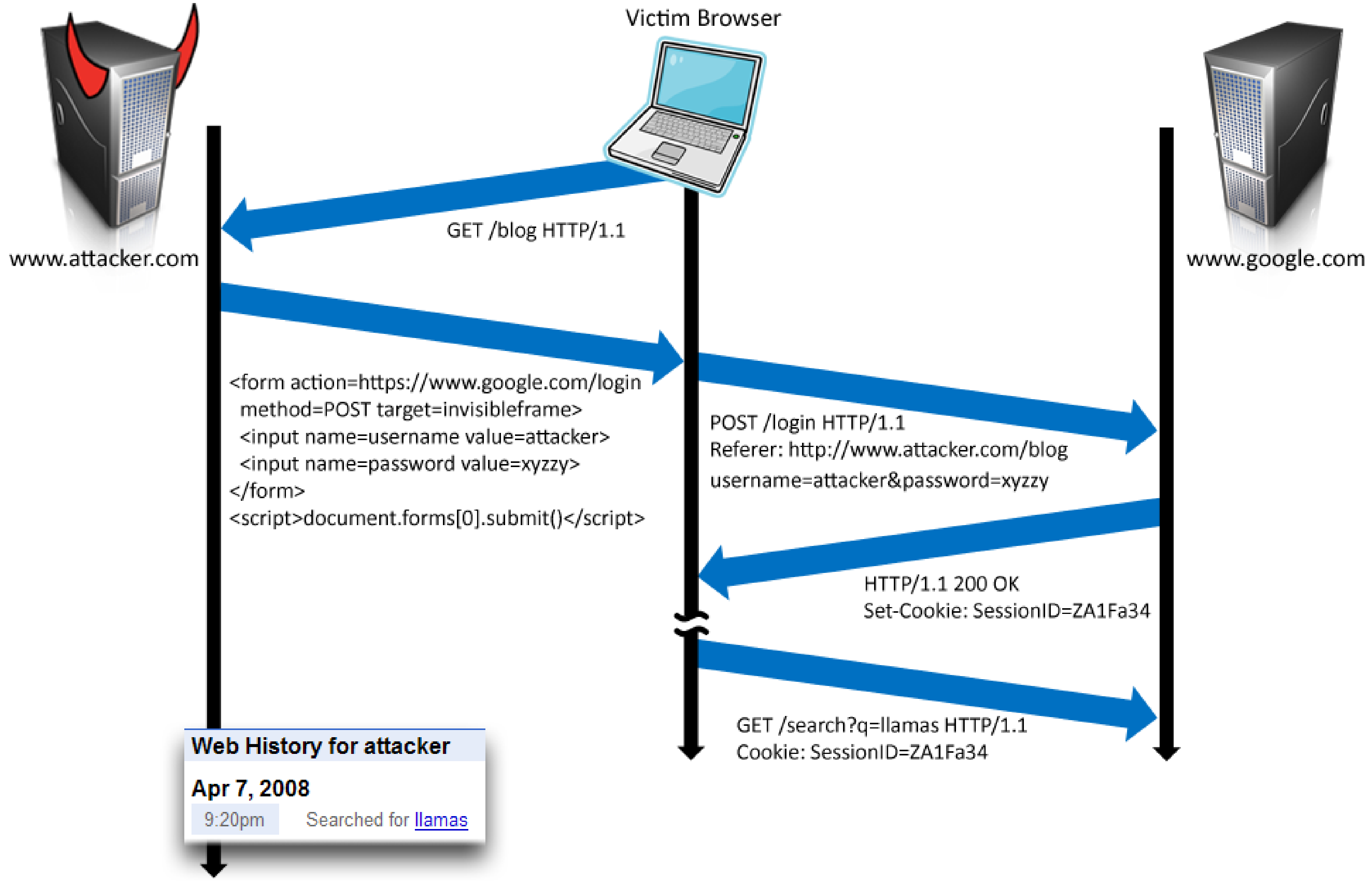
```
<img src="http://bank.com/transfer.do?acct=MARIA&amount=100000"
width="1" height="1" border="0">
```

- If this image tag were included in the email, Alice would only see a little box indicating that the browser could not render the image. However, the browser *will still* submit the request to bank.com without any visual indication that the transfer has taken place.

CRSF (OWASP - Top 10 2013)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	**Exploitability AVERAGE**	**Prevalence COMMON**	**Detectability EASY**	**Impact MODERATE**	**Application / Business Specific**
Consider anyone who can load content into your users' browsers, and thus force them to submit a request to your website. Any website or other HTML feed that your users access could do this.	Attacker creates forged HTTP requests and tricks a victim into submitting them via image tags, XSS, or numerous other techniques. If the user is authenticated, the attack succeeds.	CSRF takes advantage the fact that most web apps allow attackers to predict all the details of a particular action. Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones. Detection of CSRF flaws is fairly easy via penetration testing or code analysis.		Attackers can trick victims into performing any state changing operation the victim is authorized to perform, e.g., updating account details, making purchases, logout and even login.	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.

Login CSRF



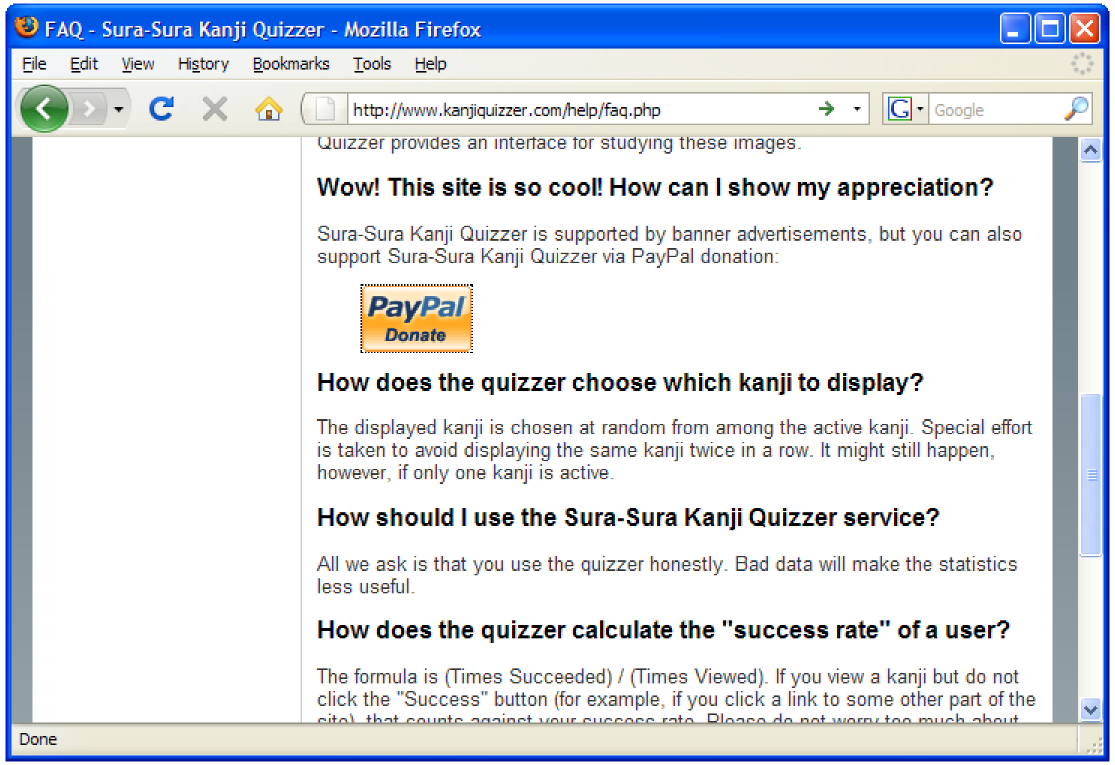
```
<form action=https://www.google.com/login
  method=POST target=invisibleframe>
  <input name=username value=attacker>
  <input name=password value=xyzzy>
</form>
<script>document.forms[0].submit()</script>
```

```
POST /login HTTP/1.1
Referer: http://www.attacker.com/blog
username=attacker&password=xyzzy
```

```
HTTP/1.1 200 OK
Set-Cookie: SessionID=ZA1Fa34
```

```
GET /search?q=llamas HTTP/1.1
Cookie: SessionID=ZA1Fa34
```

Payments Login CSRF


FAQ - Sura-Sura Kanji Quizzer - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.kanjiquizzer.com/help/faq.php
Google
PayPal Donate
Done

Quizzer provides an interface for studying these images.

Wow! This site is so cool! How can I show my appreciation?

Sura-Sura Kanji Quizzer is supported by banner advertisements, but you can also support Sura-Sura Kanji Quizzer via PayPal donation:

How does the quizzer choose which kanji to display?

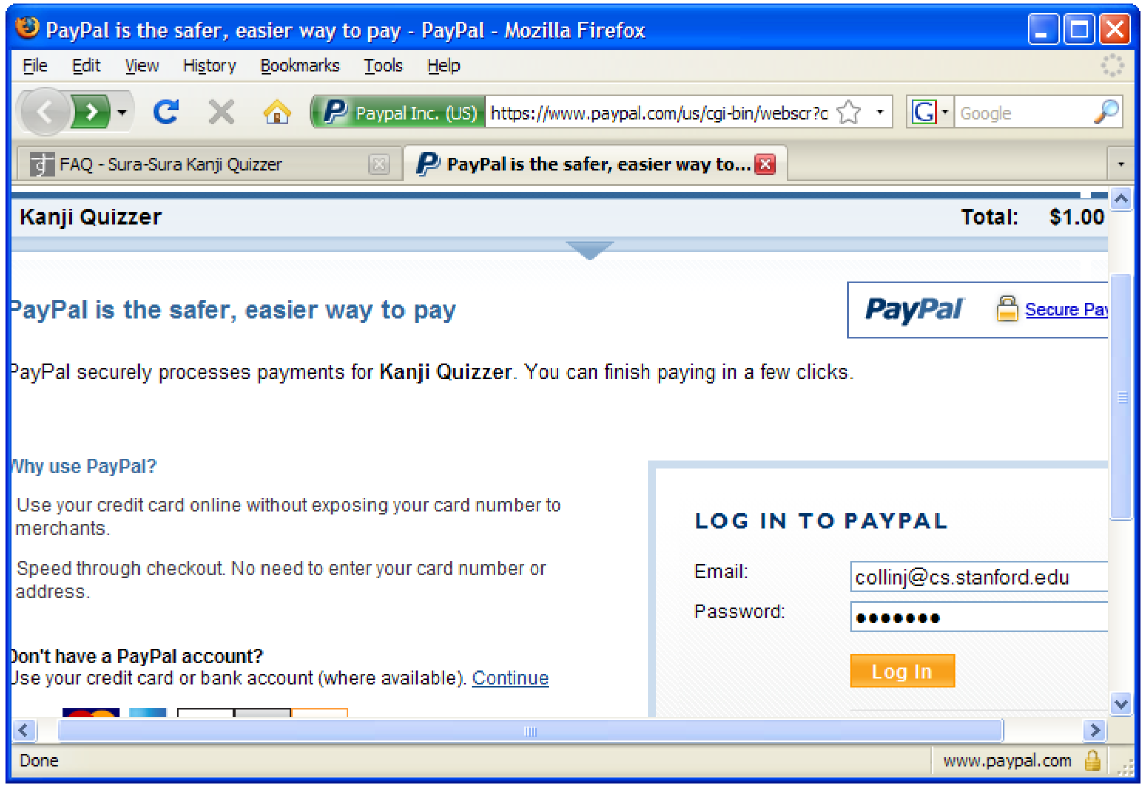
The displayed kanji is chosen at random from among the active kanji. Special effort is taken to avoid displaying the same kanji twice in a row. It might still happen, however, if only one kanji is active.

How should I use the Sura-Sura Kanji Quizzer service?

All we ask is that you use the quizzer honestly. Bad data will make the statistics less useful.

How does the quizzer calculate the "success rate" of a user?

The formula is (Times Succeeded) / (Times Viewed). If you view a kanji but do not click the "Success" button (for example, if you click a link to some other part of the site), that counts against your success rate. Please do not worry too much about


PayPal is the safer, easier way to pay - PayPal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Paypal Inc. (US)
https://www.paypal.com/us/cgi-bin/webscr?c
Google
FAQ - Sura-Sura Kanji Quizzer
PayPal is the safer, easier way to...
PayPal
Secure Pay
Log In
Done
www.paypal.com

Kanji Quizzer **Total: $1.00**

PayPal is the safer, easier way to pay

PayPal securely processes payments for **Kanji Quizzer**. You can finish paying in a few clicks.

Why use PayPal?

Use your credit card online without exposing your card number to merchants.

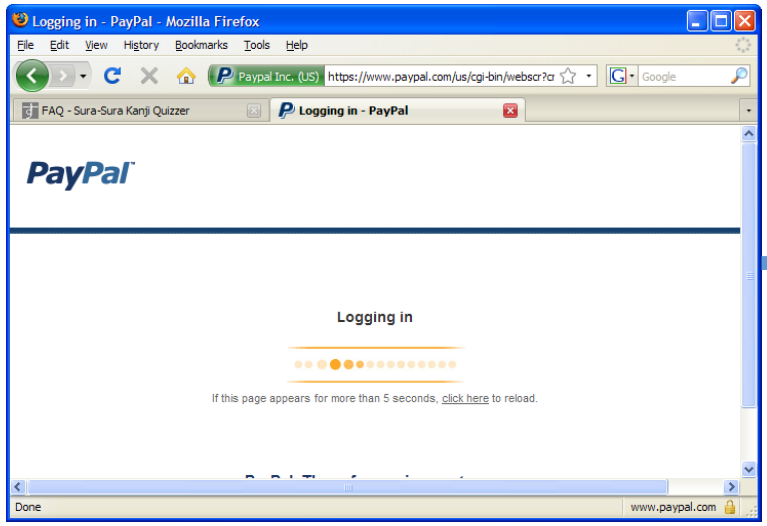
Speed through checkout. No need to enter your card number or address.

Don't have a PayPal account?
Use your credit card or bank account (where available). Continue

LOG IN TO PAYPAL

Email:	collinj@cs.stanford.edu
Password:	●●●●●●●

Payments Login CSRF

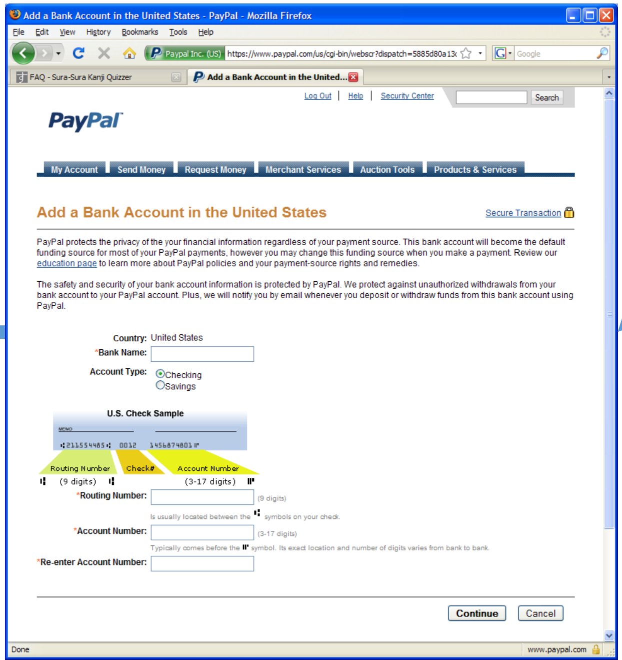

Logging in - PayPal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Paypal Inc. (US) https://www.paypal.com/us/cgi-bin/webscr?c
FAQ - Sura-Sura Kanji Quizzer
Logging in - PayPal
Done
www.paypal.com

Logging in

If this page appears for more than 5 seconds, click here to reload.

Username: attacker

Password: attacker


Add a Bank Account in the United States - PayPal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Paypal Inc. (US) https://www.paypal.com/us/cgi-bin/webscr?dispatch=5885d80a13
FAQ - Sura-Sura Kanji Quizzer
Add a Bank Account in the United...
Log Out | Help | Security Center
Search
My Account
Send Money
Request Money
Merchant Services
Auction Tools
Products & Services
Secure Transaction
U.S. Check Sample
Routing Number
Check#
Account Number
(9 digits)
(3-17 digits)
Continue
Cancel
Done
www.paypal.com

Add a Bank Account in the United States

PayPal protects the privacy of the your financial information regardless of your payment source. This bank account will become the default funding source for most of your PayPal payments, however you may change this funding source when you make a payment. Review our education page to learn more about PayPal policies and your payment-source rights and remedies.

The safety and security of your bank account information is protected by PayPal. We protect against unauthorized withdrawals from your bank account to your PayPal account. Plus, we will notify you by email whenever you deposit or withdraw funds from this bank account using PayPal.

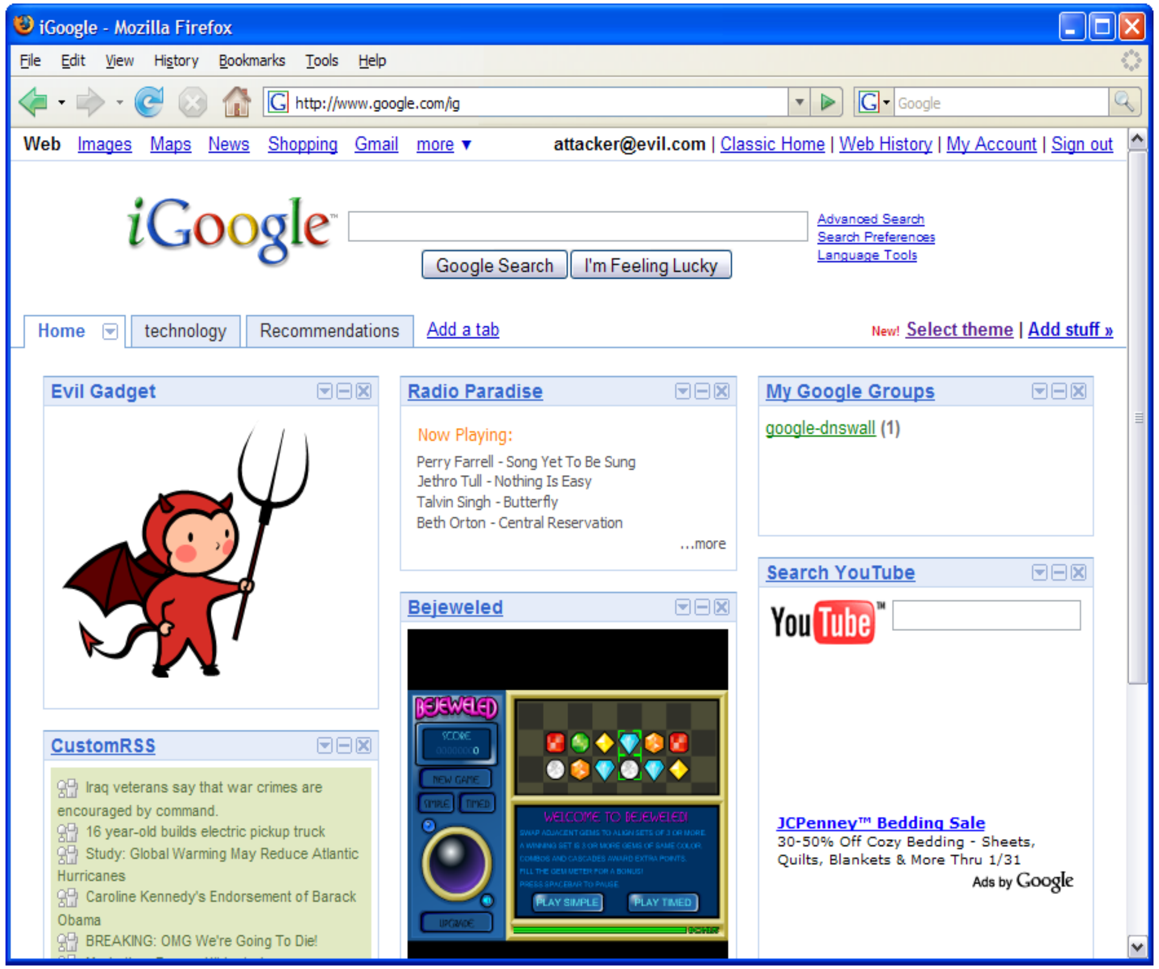
Country: United States
*Bank Name:
Account Type: [✓ Checking] Savings
*Routing Number: (9 digits)
Is usually located between the ⑆ symbols on your check.
*Account Number: (3-17 digits)
Typically comes before the ⑈ symbol. Its exact location and number of digits varies from bank to bank.
*Re-enter Account Number:

Credit card

Attacker

Information

CSRF in Google



iGoogle - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.google.com/ig
Google

Web Images Maps News Shopping Gmail more ▾ attacker@evil.com | Classic Home | Web History | My Account | Sign out

Now Playing:
Perry Farrell - Song Yet To Be Sung
Jethro Tull - Nothing Is Easy
Talvin Singh - Butterfly
Beth Orton - Central Reservation
...more

google-dnswall (1)

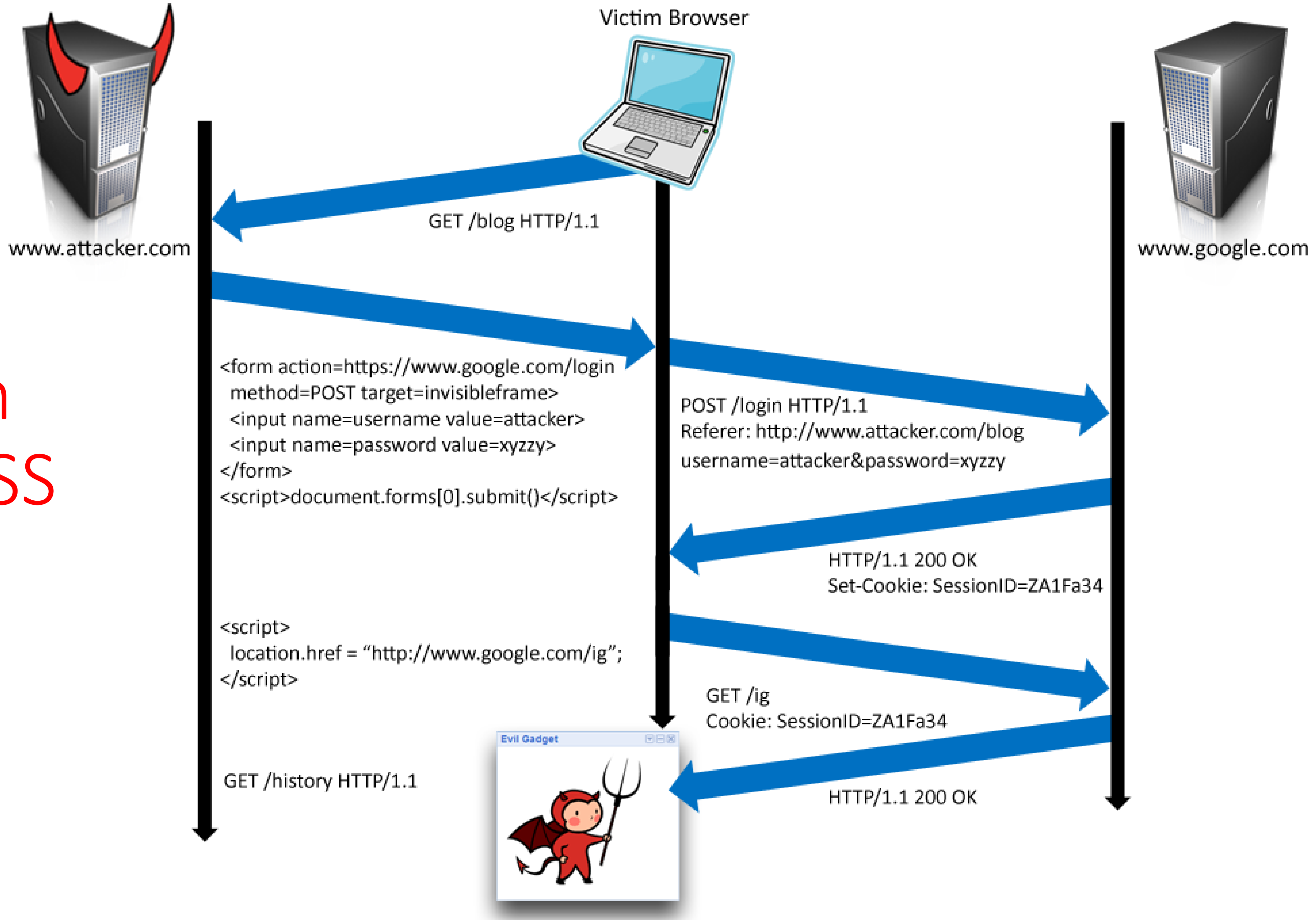
Iraq veterans say that war crimes are encouraged by command.
16 year-old builds electric pickup truck
Study: Global Warming May Reduce Atlantic Hurricanes
Caroline Kennedy's Endorsement of Barack Obama
BREAKING: OMG We're Going To Die!

JCPenney™ Bedding Sale
30-50% Off Cozy Bedding - Sheets, Quilts, Blankets & More Thru 1/31
Ads by Google

Google™ tlhIngan Hol

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

Using login CSRF for XSS



```
<form action=https://www.google.com/login
  method=POST target=invisibleframe>
  <input name=username value=attacker>
  <input name=password value=xyzzy>
</form>
<script>document.forms[0].submit()</script>
```

```
<script>
  location.href = "http://www.google.com/ig";
</script>
```

How serious is the problem?

- #12 in CWE/SANS Top 25 Most Dangerous Software Errors list
- The US Department Of Homeland Security considers CSRFs attacks more dangerous than most buffer overflows
- Often overlooked, does not receive the same level of attention of other vulnerabilities
- 8th on Top 10 Web App Vulnerabilities

What?

- Confused deputy attack against a Web browser
- Affects sites that rely on a user's identity
  - Exploits the site's trust in that identity
- Trick the user's browser into sending HTTP requests to a Target site
- Involve HTTP requests that have side effects

How?

- A malicious website instructs a victims browser to send a particular HTTP request to an honest site
- The request appears like a legitimate action performed by the user
- Leverages the victim's network connectivity and the browser's state, such as cookies
- Disrupts the integrity of the victim's session with the honest site

Differences from Cross-site scripting (XSS)?

- XSS is all about code injection

- CSRF is about making the user's browser do/request nasty things

- They exploit two different kinds of trust relationships:
  - XSS exploits the user $\rightarrow$ site/server trust
    - i.e., user’s browser does whatever the site/server instructs it to do
  - CSRF exploits the site/server $\rightarrow$ user's browser trust
    - i.e., site/server does whatever the authenticated browser requests

Attack methods in practice

- Forging GET requests via image tags
  - Image tag with SRC attribute that points to a state changing URL
  - The URL might be obfuscated by a http redirect

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

- POST/PUT/DELETE requests via JavaScript form submissions
  - Attacker creates an IFRAME (or a pop-up window)

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

  - The frame is filled with a HTML form
  - This form is submitted via JavaScript

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

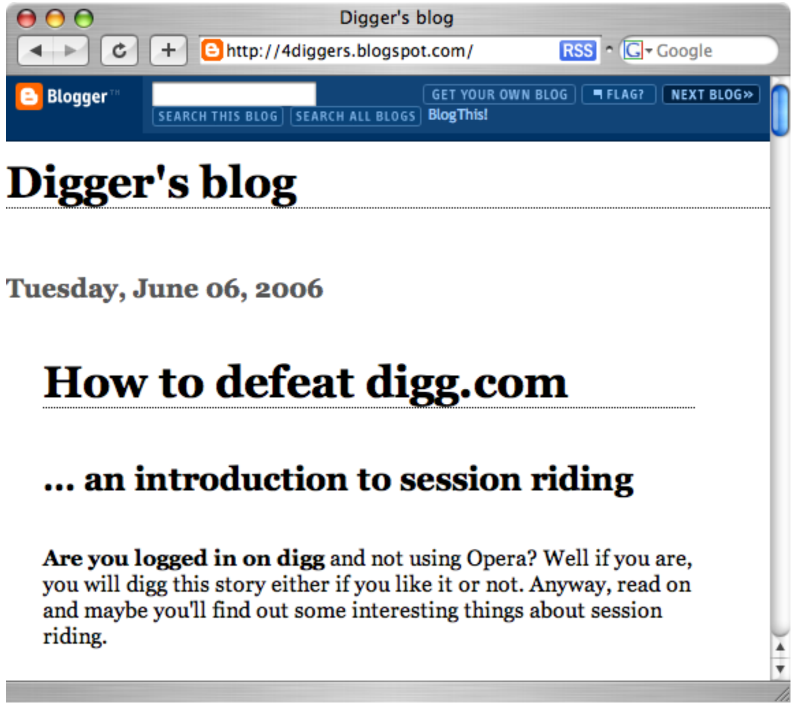
Attack methods in practice

- Can be combined with XSS to broaden the impact

- Benefits
  - Reach machines behind a firewall
  - Confuse services relying on IP address authentication
  - Exploit websites that rely only on browser state, such as
  - Cookies or HTTP basic access authentication
  - Set cookies

Example: Breaking Applications

- Vulnerable: digg.com
  - Digg.com's frontpage is determined by the number of "diggs" a certain story gets
  - Using XSRF a webpage was able to cause the victim's browser to "digg" an arbitrary URL
  - The demo page "digged" itself



Digger's blog

Tuesday, June 06, 2006

How to defeat digg.com

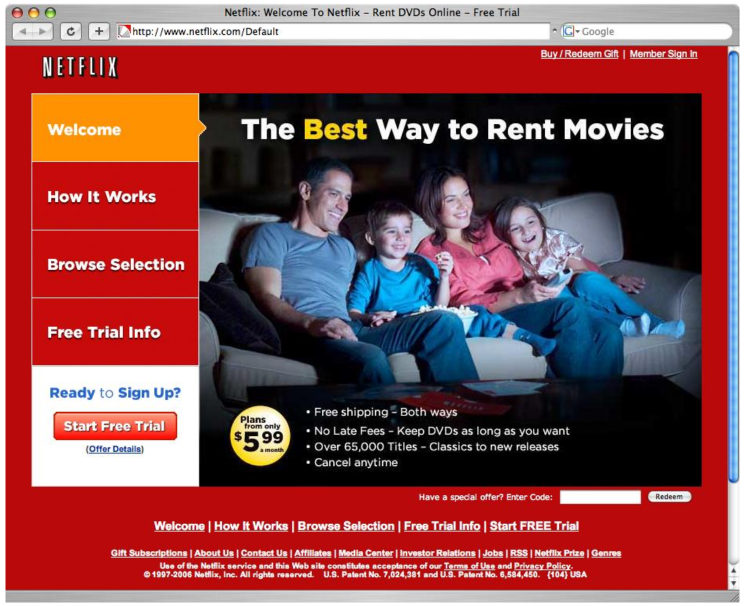
... an introduction to session riding

Are you logged in on digg and not using Opera? Well if you are, you will digg this story either if you like it or not. Anyway, read on and maybe you'll find out some interesting things about session riding.

Example: Causing Financial Loss

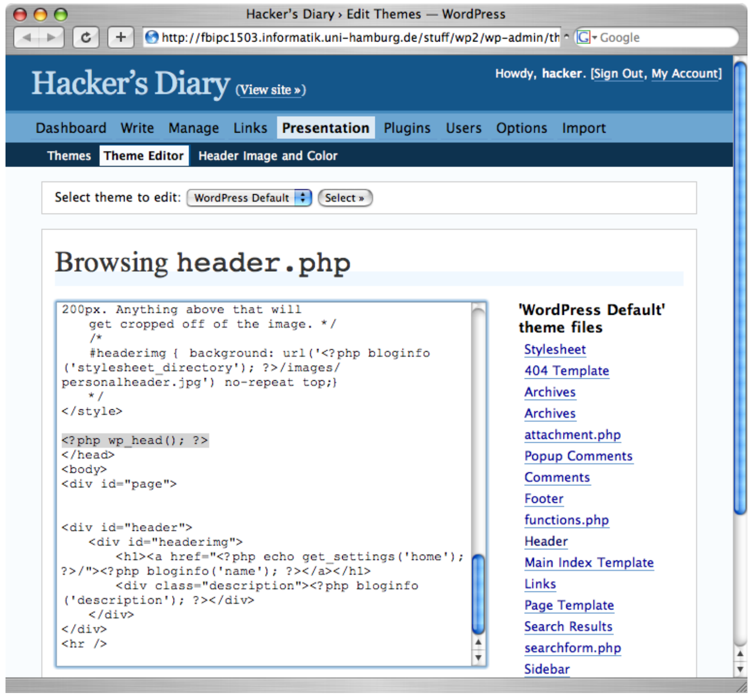
- Vulnerable: Netflix.com
  - Add movies to your rental queue
  - Add a movie to the top of your rental queue
  - Change the name and address on your account
  - Change the email address and password on your account (i.e., takeover your account)
  - Cancel your account (Unconfirmed/Conjectured)

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


Netflix: Welcome To Netflix – Rent DVDs Online – Free Trial
http://www.netflix.com/Default
NETFLIX
Buy / Redeem Gift | Member Sign In
Welcome
How It Works
Browse Selection
Free Trial Info
Ready to Sign Up?
Start Free Trial
(Offer Details)
The Best Way to Rent Movies
Plans from only $5.99 a month
• Free shipping – Both ways
• No Late Fees – Keep DVDs as long as you want
• Over 65,000 Titles – Classics to new releases
• Cancel anytime
Have a special offer? Enter Code:
Redeem
Welcome | How It Works | Browse Selection | Free Trial Info | Start FREE Trial

Example: Owning the Server

- Vulnerable: Wordpress 2.02
  - Wordpress' theme editor was susceptible to XSRF
  - Wordpress theme-files can be php-files
  - Via XSRF an attacker could modify those files to contain arbitrary php-code



Hacker's Diary › Edit Themes — WordPress
http://fbipc1503.informatik.uni-hamburg.de/stuff/wp2/wp-admin/th
Google
Hacker's Diary (View site »)
Howdy, hacker. [Sign Out, My Account]
Dashboard Write Manage Links Presentation Plugins Users Options Import
Themes Theme Editor Header Image and Color
Select theme to edit: WordPress Default Select »
'WordPress Default' theme files
Stylesheet
404 Template
Archives
Archives
attachment.php
Popup Comments
Comments
Footer
functions.php
Header
Main Index Template
Links
Page Template
Search Results
searchform.php
Sidebar

Browsing header.php

```
200px. Anything above that will
    get cropped off of the image. */
    /*
    #headerimg { background: url('<?php bloginfo
('stylesheet_directory'); ?>/images/
personalheader.jpg') no-repeat top;}
    */
</style>

<?php wp_head(); ?>
</head>
<body>
<div id="page">


<div id="header">
    <div id="headerimg">
        <h1><a href="<?php echo get_settings('home');
?>/"><?php bloginfo('name'); ?></a></h1>
        <div class="description"><?php bloginfo
('description'); ?></div>
    </div>
</div>
<hr />
```

CSRF

- General problem:
  - CSRF vulnerabilities are NOT caused by programming mistakes
  - Completely correct code can be vulnerable
  - The reason for Session Riding lies within http:
    - No dedicated authentication credential
    - State-changing GET requests
    - JavaScript
- "Preventing Session Riding" is actually "fixing the protocol"

Mitigations: isn't the Same Origin Policy enough?

- The **Same Origin Policy** limits DOM access to scripts from the same "origin"
  - The same-origin policy permits scripts running on pages originating from the same site to access each other's Document Object Models (DOM) with no specific restrictions, but prevents access to DOM on different sites.
  - Same-origin policy also applies to XMLHttpRequest
- Origin = (domain name, protocol, tcp port)
- evil.com cannot read cookies from homebanking.com
- **Making requests** to other origins is allowed!

Method 1: Switch to explicit authentication

- URL rewriting: Session token is included in every URL
  - Attention: Token leakage via proxy-logs/referrers
  - Does not protect against local attacks
    - All URLs produced by the application contain the token
- Form based session tokens
  - Session token is communicated via “hidden” form fields
  - Attention: May break the “Back” button
- Combination of explicit and implicit mechanisms
  - e.g. Cookies AND URL-rewriting
  - Has to be supported by the used framework
  - SID-leakage may still be a problem

Method 2: Manual Protection (I)

- Reflected attacks:
  - Allow only POST requests to commit state changing requests
    - (as it was intended by the inventors of http)
  - Use one-time form tokens (nonces)
    - This assures that the POST request's origin was an HTML form that was provided by the web application in the first place
- Example:

```
<form action="submit.cgi" method="POST">
  <input type="text" name="foo">
  <input type="hidden" name="nonce"
         value="xulkjsf22enbsc">
</form>
```

Method 2: Manual Protection (II)

- Local attacks:
  - Mirror all foreign content
  - Don’t allow arbitrary URLs
  - Only serve images from your own servers
  - Careful: Do not allow attackers to store arbitrary data on your computers

Method 3: Automatic protection

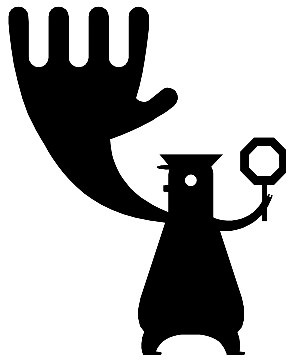
- NoForge [1]
  - Reverse Proxy
  - Positioned between web application and internet
  - Parses http responses and adds tokens to all internal URLs
  - Drops requests, that do not contain a token
  - Only protects applications that use cookies for session tracking

- [1] Nenad Jovanovic, Engin Kirda and Christopher Kruegel, Preventing Cross Site Request Forgery Attacks, IEEE International Conference on Security and Privacy in Communication Networks (SecureComm), Baltimore, MD, August 2006, http://www.seclab.tuwien.ac.at/papers/noforge.pdf

Misconceptions – Defenses That Don’t Work

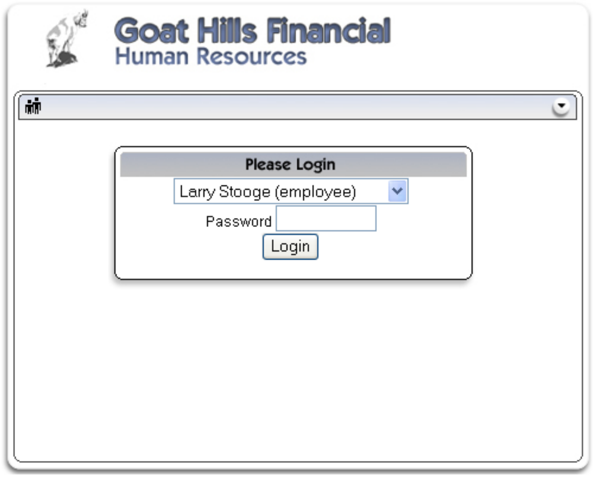
- Only accept POST
  - Stops simple link-based attacks (IMG, frames, etc.)
  - But hidden POST requests can be created with frames, scripts, etc…
- Referrer checking
  - Some users prohibit referrers, so you can’t just require referrer headers
  - Techniques to selectively create HTTP request without referrers exist
  - Furthermore, referrers can be spoofed with Flash
- 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!

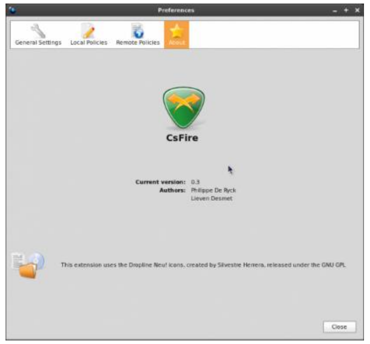


Extra: CSRF Defenses

- 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` …



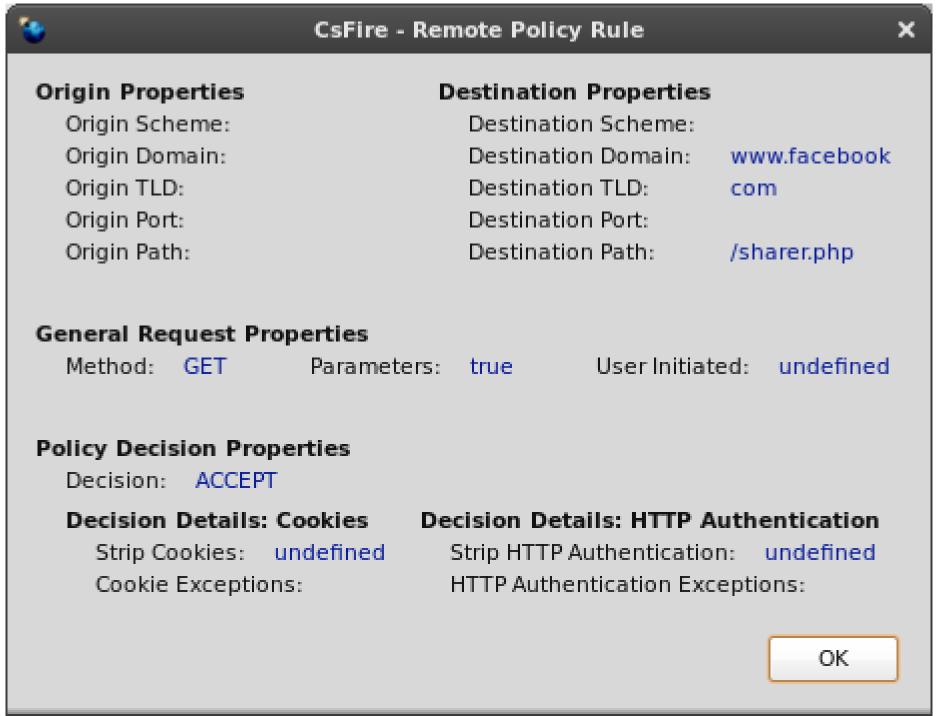
Protection example: CeaseFire (CsFire)



Current version: 0.3
Authors: Philippe De Ryck
Lieven Desmet

This extension uses the Dropline Neu! icons, created by Silvestre Herrera, released under the GNU GPL.

- Blocking all cross-domain traffc is the most secure policy, but would degrade user experience
- CsFire is a browser plugin for Firefox that inspects the outgoing HTTP requests and decides which action to perform on them:
  - block
  - allow
  - strip authentication information



Origin Properties
Origin Scheme:
Origin Domain:
Origin TLD:
Origin Port:
Origin Path:

Destination Properties
Destination Scheme:
Destination Domain: www.facebook
Destination TLD: com
Destination Port:
Destination Path: /sharer.php

General Request Properties
Method: GET Parameters: true User Initiated: undefined

Policy Decision Properties
Decision: ACCEPT

Decision Details: Cookies
Strip Cookies: undefined
Cookie Exceptions:

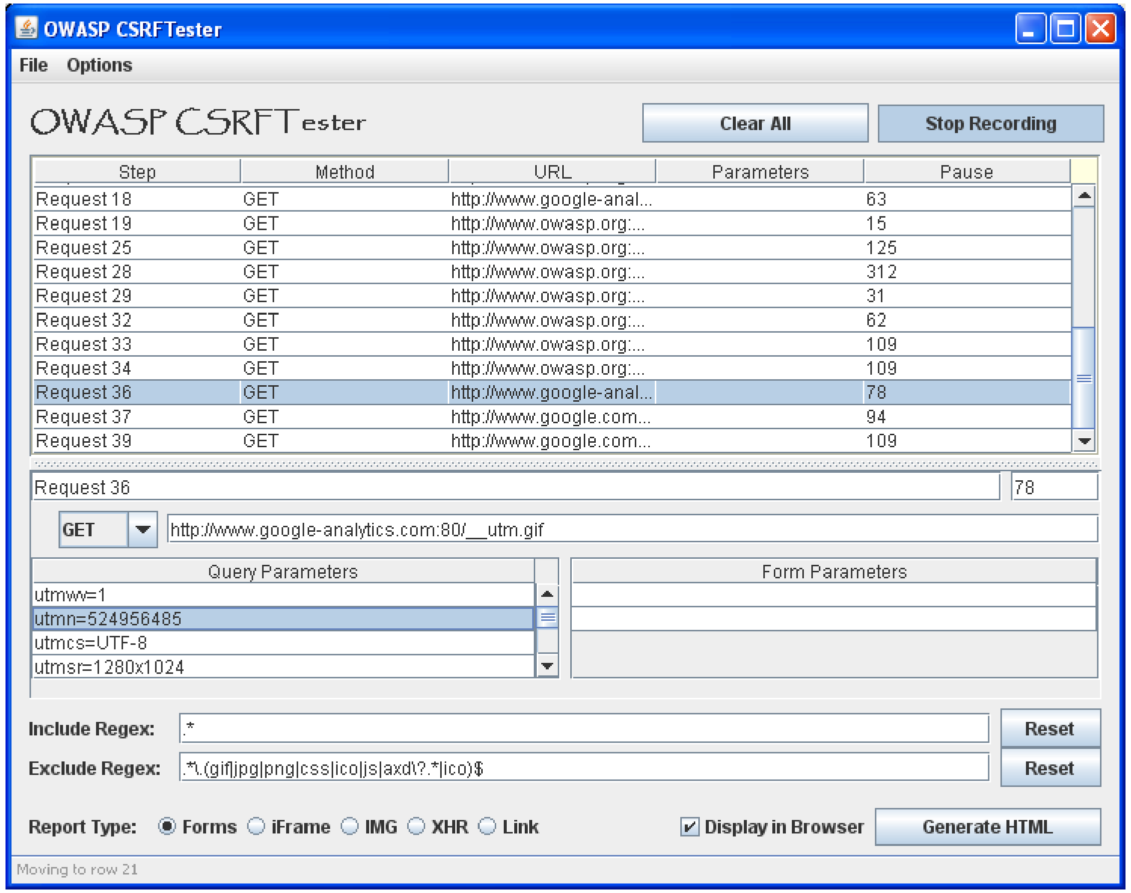
Decision Details: HTTP Authentication
Strip HTTP Authentication: undefined
HTTP Authentication Exceptions:

CsFire: default policy and remarks

- POST: always strip
- GET: accept if it is user initiated and with no parameters, strip otherwise
- Does not alter the user experience when it comes to AJAX or Single Sign On
- Can be inadequate on mashup sites relying on implicit authentication to construct their content

CSRF testing 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



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	

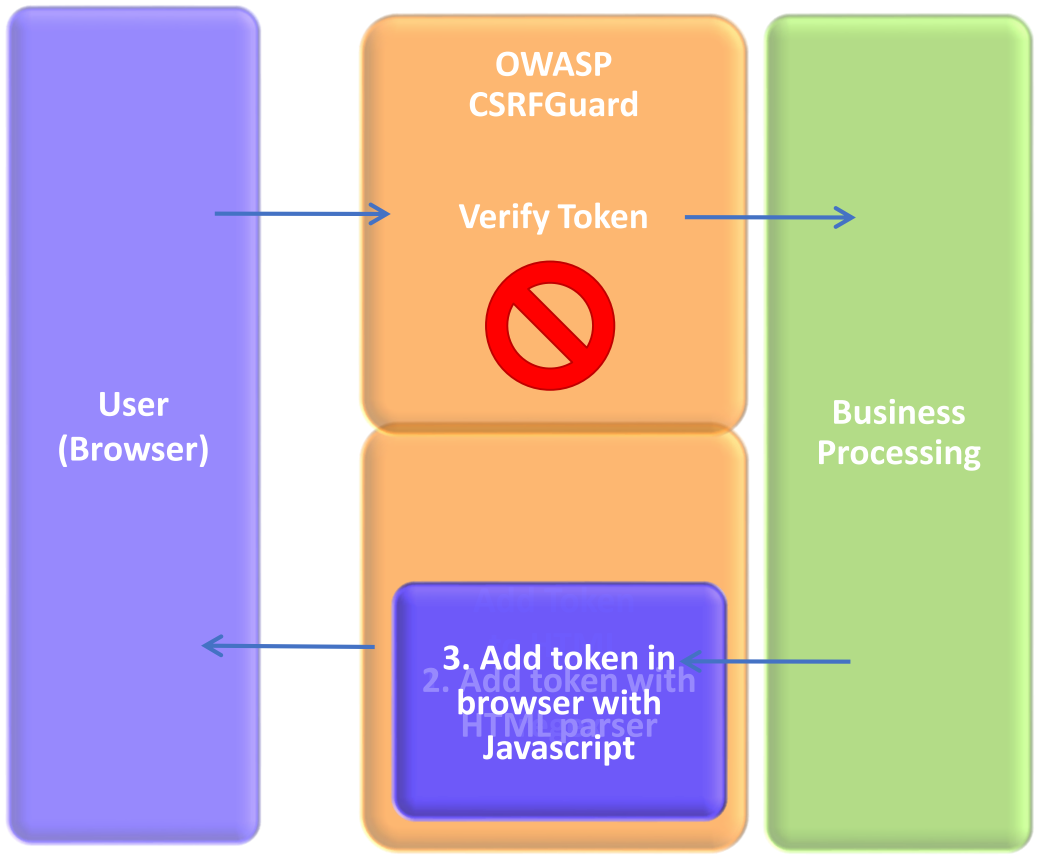
Include Regex: .* — Reset

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

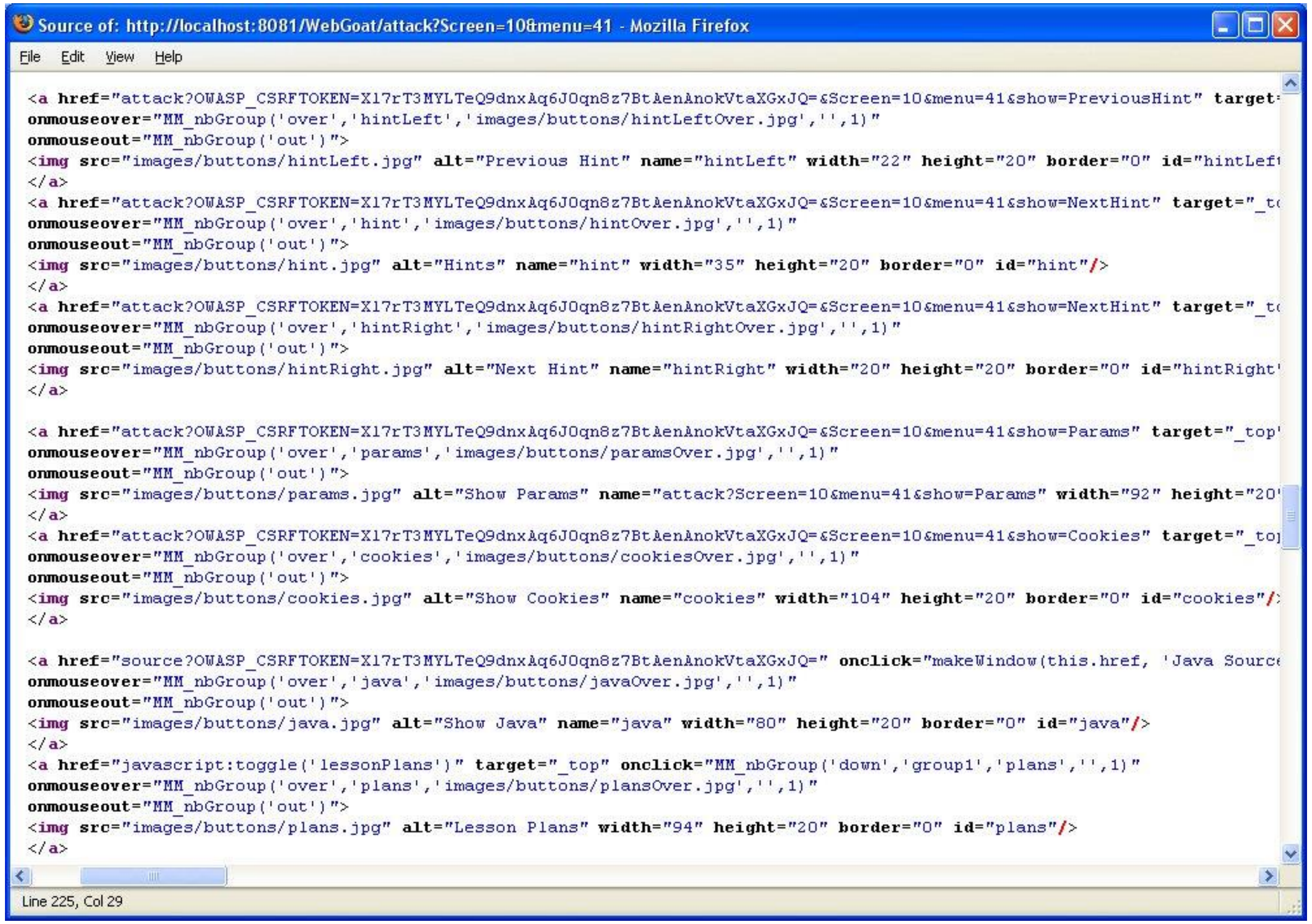
New Tool: OWASP CSRFGuard 2.0


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

- 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

```
<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/