Cross-site request forgery

Collin Jackson
Outline

- Classic CSRF
- Server-side Defenses
- Advanced Attacks
- Proposals for client-side changes
Data export

Many ways to send information to other origins

```html
<form action="http://www.b.com/">
  <input name="data" type="hidden" value="hello">
</form>

<img src="http://www.b.com/?data=hello"/>

No user involvement required

Cannot read back response
Classic CSRF attack

- User visits victim site site
  - Logs in
- User loads attacker's site
  - Or encounters attacker's iframe on another site
- Attacker sends HTTP requests to victim
  - Victim site assumes requests originate from itself
Classic CSRF Attack

User credentials

GET /blog HTTP/1.1

POST /transfer HTTP/1.1
Referer: http://www.attacker.com/blog
recipient=attacker&amount=5100
Cookie: SessionID=523FA4cd2E

HTTP/1.1 200 OK
Transfer complete!
DEFENSES
CSRF Defenses

- Secret Validation Token
  - `<input type=hidden value=23a3af01b>`

- Referer Validation

- Custom HTTP Header
  - X-Requested-By: XMLHttpRequest
Secret Token Validation

- Requests include a hard-to-guess secret
  - Unguessability substitutes for unforgeability

- Variations
  - Session identifier
  - Session-independent token
  - Session-dependent token
  - HMAC of session identifier

See "Robust Defenses for Cross-Site Request Forgery" for a comparison of these options.
Secret Token Validation

Add a Slice

Slice Size
- 256 slice: $20.00/month - 10GB HD, 100GB BW
- 512 slice: $38.00/month - 20GB HD, 200GB BW
- 1GB slice: $70.00/month - 40GB HD, 400GB BW
- 2GB slice: $130.00/month - 80GB HD, 800GB BW
- 4GB slice: $250.00/month - 160GB HD, 1600GB BW
- 8GB slice: $450.00/month - 320GB HD, 2000GB BW
- 15.5GB slice: $800.00/month - 620GB HD, 2000GB BW

System Image
- Ubuntu 8.04.1 LTS (hardy)

NOTE: You will be charged a prorated amount based upon the number of days remaining in your billing cycle.

```html
<input name="authenticity_token" type="hidden" value="0114d5b35744b522af8643921bd5a3d899e7fbd2" />
```

```
<input name="image" type="hidden" value="/images/logo.jpg" width='110'/>
```
Referer Validation

Facebook Login

For your security, never enter your Facebook password on sites not located on Facebook.com.

Email:

Password:

- Remember me

Login or Sign up for Facebook

Forgot your password?
Referer Validation Defense

HTTP Referer header
- Referer: http://www.facebook.com/ ✓
- Referer: http://www.attacker.com/evil.html ✗
- Referer: ?

Lenient Referer validation
- Doesn't work if Referer is missing

Strict Referer validation
- Secure, but Referer is sometimes absent...
Referer Privacy Problems

- Referer may leak privacy-sensitive information


- Common sources of blocking:
  - Network stripping by the organization
  - Network stripping by local machine
  - Stripped by browser for HTTPS -> HTTP transitions
  - User preference in browser
  - Buggy user agents

- Site cannot afford to block these users
Suppression Measurement

283,945 impressions
Suppression over HTTPS is low
Lenient Validation Vulnerability

My site uses HTTPS, am I safe?

Problem: Browsers do not append Referer if the source of the request is not an HTTP page

ftp://attacker.com/attack.html
data:text/html,<html>…</html>
javascript:'<html>…</html>''
Strict Validation Problems

- Some sites allow users to post forms
  - XSS sanitization doesn't include `<form>`
  - These sites need another defense

- Many sites allow users to post hyperlinks
  - Solution: Respect HTTP verb semantics
  - GET requests have no side effects
  - POST requests can change state
Custom Header Defense

- XMLHttpRequest is for same-origin requests
  - Can use setRequestHeader within origin

- Limitations on data export format
  - No setRequestHeader equivalent
  - XHR2 has a whitelist for cross-site requests

- Issue POST requests via AJAX:
  
  ```
  X-Requested-By: XMLHttpRequest
  ```

- Doesn't work across domains
ANNOUNCEMENTS

Project 2, Mac OSX Tiger
ADVANCED ATTACKS
Broader view of CSRF

● Abuse of cross-site data export feature
  ■ From user’s browser to honest server
  ■ Disrupts integrity of user’s session

● Why mount a CSRF attack?
  ■ Network connectivity
  ■ Read browser state
  ■ Write browser state

● Not just “session riding”
Login CSRF

[Diagram showing the process of a Login CSRF attack]

1. www.attacker.com sends a request to GET /blog HTTP/1.1.
2. The victim browser sends a POST request with credentials to www.google.com.
3. The server responds with HTTP/1.1 200 OK and sets a cookie: SessionID=ZA1Fa34.
4. The victim browser is redirected to GET/search?q=llamas HTTP/1.1.
5. The browser sends a request with the cookie: SessionID=ZA1Fa34.

Web History for attacker
Apr 7, 2008
9:20pm  Searched for llamas
Payments Login CSRF

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:

![PayPal Donate](image)

**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..."
Payments Login CSRF
Rails vs. Login CSRF
Login CSRF Fails

SliceManager Login

Your web session has expired.

Email
Password
Login
Cookies must be turned on

SliceManager Mobile :: Reset password? :: Login using OpenID
CLIENT-SIDE DEFENSES
Can browsers help with CSRF?

- Does not break existing sites
- Easy to use
- Hard to misuse
- Allows legitimate cross-site requests
- Reveals minimum amount of information
- Can be standardized
Proposed Approaches

HTTP Headers
- Identify the source of requests
- Change Referer header or add a new Origin header
- Send more information for POST than GET
- Experiment: Cross-domain POSTs out of firewall accounted for ~0.0001% of traffic
- Problem: Unsafe GET requests
- Problem: Third-party content within an origin
- Problem: How to handle redirects

Same-origin-only cookies
- Doesn't help multi-domain sites: amazon.com and amazon.co.uk
- These sites could use other defenses
Conclusion

- **Server-side defenses are required**
  - Secret token validation – use frameworks like Rails
  - Referer validation – works over HTTPS
  - Custom headers – for AJAX

- **No easy solution**
  - User does not need to have an existing session for attacks to work
  - Hard to retrofit existing applications with defenses