Web Services
CS 360 Internet Programming

Daniel Zappala
Brigham Young University
Computer Science Department
Cross-Site Scripting (XSS) attack

- attacker injects client-side script into a web page viewed by someone else
- relies on browser trusting the scripts given to it by the current web site

*if I visit Facebook, I should be safe to execute scripts the Facebook site gives me*
Example Vulnerability


2. `<p>Your search for 'flowers' returned the following results:</p>`

- if server does not check the input, then an attacker can inject a script
Example Vulnerability

1. `http://www.google.com/search?q=flowers+<script>alert(1)</script>`
2. `<p>Your search for 'flowers<script>alert(1)</script>'</p>`

- if server does not check the input, then an attacker can inject a script
- if you can execute a script, then you can
  - redirect to malware
  - deface a web site
  - steal cookies, passwords, clipboard
WhiteHat Web Site Security Statistics Report, 2010
- 64% of web sites vulnerable to XSS attack
- 105 days on average to fix it (banking is faster, retail is slower)

why aren’t they fixed?
- no one at organization understands them or is responsible for fixing them
- features prioritized ahead of security
- code owned by an unresponsive third party
- risk is accepted
XSS Types

- reflected
  - user input read from request parameters in URL and written directly to output
  - attack tries to get victim to visit the URL, executing the code when it is displayed

- persistent
  - script stored directly on a web site (e.g. a Facebook status or Flickr caption)
  - when victim visits the web page, viewing the page triggers the attack
XSS Vulnerability, Django

```python
1  c = Comment()
2  c.text = request.POST['text']
3  c.save()
```

- site accepts comments, stores input directly from user
- when comment is displayed, it can include anything, including script
Protection from XSS

- filter input
- escape output
- some web development frameworks do this for you automatically
Cross-Site Request Forgery (XSRF) attack
- attacker tricks victim into executing a script on a site where the victim has an account
- relies on server trusting the user’s identity

if the user logs in to my bank and sends me a request to withdraw funds that contains his login cookie, then I can trust that it is really her
Example Vulnerability

if you are currently logged into your bank, then the bank cannot tell that this request isn’t coming from you
XSRF Statistics

- WhiteHat Web Site Security Statistics Report, 2010
  - 24% of web sites vulnerable to XSRF attack
  - hard to capture because web site logs make it look like a legitimate user request, may be under-reported
- identified on ING Direct (banking), YouTube, MetaFilter, The NY Times in 2008
Protection from XSS

- tokens
  - require a GET request to get a form before accepting a POST request for the form
  - send a token in the GET request that must be echoed back in the POST
  - token should be random and unique to that form
  - expire the token after a short time

- require user authorization for significant transactions