Cross Site Request Forgery (CSRF)
Recall: Session Cookies

Sent on every page request... ...intentional or not
Authenticates with bank.com

/transfer?amount=500&dest=grandson

Cookie checks out!
Sending $500 to grandson
/transfer?amount=10000&dest=evilcorp

Cookie checks out!
Sending $10000 to EvilCorp
Cross Site Request Forgery (CSRF)

A **CSRF attack** causes the end user browser to execute unwanted actions on a web application in which it is currently authenticated.
Another Example: Home Router

1. configure router
2. visits malicious site
3. malicious page
4. config access

Browser

Home router

Attacker

Attacker can enable remote admin, reset password, etc.

50% of home routers have default or no pw*

CSRF Defenses

- Secret Validation Token
  
- Referer Validation
  
- Origin Validation

* Referrer is misspelled as “referer” in HTTP header field
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
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)

Slice Name

Add Slice or cancel

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

HTTP Origin header

✓  Origin: http://www.facebook.com/
☐  Origin:

Lenient: Accept when not present (insecure)
Strict: Don’t accept when not present (secure)
From HW2: The CRIME Attack

Malicious Script that sends forced requests to good.com

Forced request to good.com containing session token + some attacker controlled input

Compressed, then Encrypted

Eavesdrop on packet size

CSRF Defenses do not prevent this!
Web Frameworks
Web Frameworks

• Automatic CSRF Tokens

```html
<input type=hidden value=23a3af01b>
```

• Don’t need to actually write SQL queries

```ruby
Post.find(params[:id]) =>
 "select * from posts where id='"
 + safe(params[:id]) + "'
```

• Automatic XSS Sanitization
Web Frameworks – XSS Sanitization

Rails HTML Templating:

```html
<html>
<body>
    Welcome to the site <%= user.username %>!
</body>
</html>
```

```html
<html>
<body>
    Welcome to the site &lt;b&gt;jburket&lt;/b&gt;!
</body>
</html>
```

```
user.username = "<b>jburket</b>"
```
Web Frameworks

Increased automation in web frameworks can introduce new vulnerabilities
Remote File Inclusion

colors.php:

```php
<?php
    if (isset( $_GET['COLOR'] ) ){
        include( $_GET['COLOR'] . '.php' );
    }
?>
```

```
“/colors.php?COLOR=red” will include contents of red.php
“/colors.php?COLOR=blue” will include contents of blue.php
“/colors.php?COLOR=/hidden/dangerous” will include /hidden/dangerous.php
```

Perfect for executing an XSS attack

Example from wikipedia.org/File_inclusion_vulnerability
Mass Assignment Vulnerabilities

```ruby
users_new.rb:
...
form_data = params[:post]
User.new(form_data)
...
```

Mass Assignment Vulnerabilities

```
users_new.rb:
...
form_data = params[:post]
User.new(form_data)
...  
```

Malicious Servers and Browser Security
CSS History Probing

Client has visited Google, Facebook and the Facebook Group 12345

Client has NOT visited Twitter or Facebook Group 98765

Attacker uses JavaScript + CSS to check which links are visited

evil.com:
http://www.google.com
http://www.facebook.com
http://www.twitter.com
http://www.facebook.com/group?id=12345
http://www.facebook.com/group?id=98765

Image from http://matthewjamestaylor.com/blog/experimenting-with-visited-links
How does the “Like” button work?

Like button knows about your Facebook session!

Appears in “Mashup” with content from other domains
How does the “Like” button work?

Like Button Requirements:
• Needs to access cookie for domain facebook.com
• Can be deployed on domains other than facebook.com
• Other scripts on the page should not be able to click Like button

We need to *isolate* the Like button from the rest of the page
IFrames

Here's an IFrame:

I'm in an IFrame!

Parent page

Embedded page

Here's an IFrame:

ESPN FC HOMEPAGE: GLOBAL USA EN ESPAÑOL

Any page can be embedded
The same-origin policy states that the DOM from one domain should not be able to access the DOM from a different domain.
How does the “Like” button work?

The same-origin policy prevents the host from clicking the button and from checking if it’s clicked.
The same-origin policy prevents malicious sites from clicking their own “Like” button.

What if the site can trick you into clicking it yourself?
Clickjacking occurs when a malicious site tricks the user into clicking on some element on the page unintentionally.

Slides modeled after presentation by Lin-Shung Huang at USENIX 2012.
Clickjacking

Real Cursor

Fake Cursor

Click for a FREE iPad!
Clickjacking

This is the button that gets clicked!

Real Cursor Hidden

Fake Cursor

Click for a FREE iPad!
Advanced Clickjacking

Malicious site now has access to your webcam!

Also work done at CMU!

Clickjacking - Mitigation

Adding a delay between a button appearing and being usable helps prevent Clickjacking.
If pages with sensitive buttons can be put in an IFrame, then it may be possible to perform a Clickjacking attack.
Framebusting is a technique where a page stops functioning when included in a frame.

```html
<script type="text/javascript">
  if(top != self) top.location.replace(self.location);
</script>
```

If the page with this script is embedded in a frame, then it will escape out of the frame and replace the embedding page.
Don’t roll your own crypto

Don’t write your own sanitization

Don’t write your own framebusting solution
Framebusting is Complicated

```javascript
if(top.location!=self.location) {
  parent.location=self.location;
}
```

Fails if page is embedded two Iframes deep

```html
<script type="text/javascript">
  if(top != self) top.location.replace(self.location);
</script>
```

If the embedding page sets the onBeforeUnload event, the script can be blocked

If the embedding page makes lots of requests that return “204 – No Content” responses, we don’t even need the dialog

Framebusting is Complicated

```html
<style>
  body { display: none; }
</style>

<script>
  if (self == top) {
    document.getElementsByTagName("body")[0].style.display = 'block';
  } else {
    top.location = self.location;
  }
</script>

Javascript-based Framebusting is a just a hack.
Is there a better way?

X-Frame-Options Header

**DENY:**
The page cannot be embedded in a frame

**SAMEORIGIN:**
The page can only be framed on a page with the same domain

**ALLOW-FROM origin:**
The page can only be framed on a page with a specific other domain

---

Can limit flexibility and might not work on older browsers

*This content cannot be displayed in a frame.*

To protect your security, the publisher of this content does not allow it to be displayed in a frame.

[Click here to open this content in a new window]
Multi-Party Web Applications
Same-origin policy won’t stop parties from communicating directly to share information

This can be *good*: Single Sign-On Multiparty E-Commerce
Disclaimer: The exact details of the following protocols may not be 100% correct (i.e. Facebook might use a slightly different implementation than presented here). Our goal is to get a feel for how these systems work.

This section won’t be on the test. Something similar may come up in the homework, however.
Multi-Party E-Commerce Applications

Multi-Party E-Commerce Applications

Order 123 is completed

I’d like the $40 Vest

Paypal.com/pay?id=123&total=40

Paypal.com/pay?id=123&total=1

give me $1

Here’s my $1

Cool

Client

JIMMY’S ARMY SURPLUS

Multi-Party E-Commerce Applications

Redirect to paypal.com/pay:
- id=123
- total=40
- Signed by PayPal
- callback = jimmy.com
- Signed by Jimmy

Signature checks out.
Sending you your vest.

$40

Multi-Party E-Commerce Applications

I’d like to sign in with Facebook

Who has token “X”? My secret is Y

Redirect to Facebook (include callback URL) and identifier Z

It’s Alice. She has 5 friends.

Here’s the token “X” for user Z

Yeah

OK. Here’s a special token “X”. Redirect to callback with identifier Z

Alice

Z is authenticated as Alice

Z linked to Alice’s session

Facebook secret: Y

Knows Udacity’s secret is Y

Who has token “X”? My secret is Y

Redirect to Facebook (include callback URL) and identifier Z

Here’s the token “X” for user Z

Give your permission to Udacity?

Yeah

OK. Here’s a special token “X”. Redirect to callback with identifier Z

Alice

OAuth Security Advisory: 2009.1
I'd like to sign in with Facebook (include callback URL) and identifier Z

Who has token “X”? My secret is Y

It’s Alice. She has 5 friends.

Eve

Hey Alice! Check out this URL!

Alice

Eve is authenticated as Alice

Who has token “X”? My secret is Y

It’s Alice. She has 5 friends.

Eve

Hey Alice! Check out this URL!

Type of *Session Fixation* Attack – Fixed in OAuth 2.0

Knows Udacity’s secret is Y

OK. Here’s a special token “X”. Redirect to callback with identifier Z

OAuth Security Advisory: 2009.1
Questions?