



Web Security Project

Patrick Mutchler
cs155



New to Web Programming?

There are four different languages you need to know for this project

HTML, PHP, JS, CSS

Fortunately, there are tons of resources for learning about these technologies



HTML

Hyper**T**ext **M**arkup **L**anguage

Used to tell the browser the structure of a web page

HTML - Tags

Surrounded by angle brackets `<>`

Header information: `<head>`

Links to other documents: `<a>`

Images: ``

Page regions: `<div> content here </div>`

Input forms:

`<form>`

`<input type="text"/>`

`<input type="submit" value="Submit"/>`

`</form>`



HTML - Properties

Key=value pairs inside of a tag

```

```

Can represent URLs in different ways

Relative: images/cats.jpg

Server-relative: /images/cats.jpg

Absolute: http://pictures.com/images/cats.jpg

Can delimit value with ' or " or nothing



HTML - iFrames

Embed HTML documents in other documents

```
<iframe name="myframe" src="http://www.google.com"/>
```

Very useful for this project



HTML - Forms

```
<form action = "/login.php">  
  <input type="text" name="username">  
  <input type="password" name="password">  
  <input type="submit" value="Log in">  
</form>
```

Use method='GET' for requests without side effects

Use method='POST' for requests with side effects

Use target='myframe' to avoid navigating away



CSS

Style information for html tags

Can set style as an html property

```
<span style="color: red"> Red text </span>
```

Can also set styles globally

```
#mynodeid {color:red;}
```




Javascript

Browser language for manipulating page content

Full of incredibly bad design choices (unrelated to this project)

Spec is poorly written and even more poorly followed



Invoking Javascript

```
<script> alert('hello world') </script>
```

```
<a href="javascript:alert('hello world')">
```

```
<button onclick="alert('hello world')">
```



Manipulating the DOM

```
document.getElementById(id)
document.getElementsByTagName(tag)
document.write(htmltext)
document.createElement(tagname)
document.body.appendChild(node)
document.forms[index].fieldname.value = ...
document.formname.fieldname.value = ...
frame.contentDocument.getElementById(id)
someHTMLElement.innerHTML
```



Other Useful Functions

Navigation

```
document.location
```

```
document.formname.submit()
```

```
document.forms[0].submitfield.click()
```

Events

```
<script>
```

```
    var foo = document.getElementById('foo');
```

```
    foo.addEventListener('click', function(){
```

```
        alert('hello world');
```

```
    }, false);
```

```
</script>
```



Useful CSS

```
var node = document.getElementById('id');  
  
node.style.display = 'none';  
node.style.visibility = 'hidden';  
node.style.position = 'absolute';  
  
// all tags with id "mynodeid" are hidden  
document.write("<style> \  
    #mynodeid {visibility:hidden;} </style>")
```



PHP

Scripting language to produce HTML and code

```
<input value=<?php echo $myvalue; ?>>
```

Form data in global arrays `$_GET`, `$_POST`



Super Useful

Download Firebug for Firefox

Turn on the console for script and error logging

Allows you to manipulate the DOM manually from the client

Extremely useful for web development (and web attacks)



Attacks



Cross-Site Scripting (XSS)

Attacks 1, 4, and 5 are all variants

The most common web vulnerability

Browser executes JS controlled by the attacker
in a context the attacker doesn't control



Cross-Site Scripting (XSS)

Famous example:

- Myspace Worm (2005)
- One million affected users in 20 hours

Recent example (Published last Wednesday):

- Stored XSS in Facebook chat
- Clicking a link in FB chat executes malicious Javascript



Cross-Site Scripting (XSS)

Three major types

- Reflected
- Stored
- DOM-Based

You will need to create Reflected and Stored attacks for Project 2.



Reflected XSS - How it works

```
<body> <?php
    $userid = $_GET['id'];
    echo("Your id is " . $userid);
?> </body>
```

www.vulnerable-site.com/?id=<script> ... </script>

```
<body>
    Your id is <script>...</script>
</body>
```



Stored XSS - How it works

```
<?php
    $profile = $_POST['profile'];
    DBStoreProfile('id', $profile)
    ...
    $id = $_GET['id'];
    $profile = DBReadProfile('id');
    echo("Requested profile is " . $profile);
?>
```

Requested profile is `<script>...</script>`



XSS - Javascript vectors

Lots more than just `<script>`

- `onError`, `onLoad`, etc in html tags
- `javascript:`
- `eval` (please don't ever use this)
- ...

Almost all blacklist filters can be defeated on at least some browsers



Cross-Site Request Forgery (CSRF)

Attack 2 is a CSRF attack

Executes commands on another website

Exploits the fact that the website doesn't check who is making the request

```

```

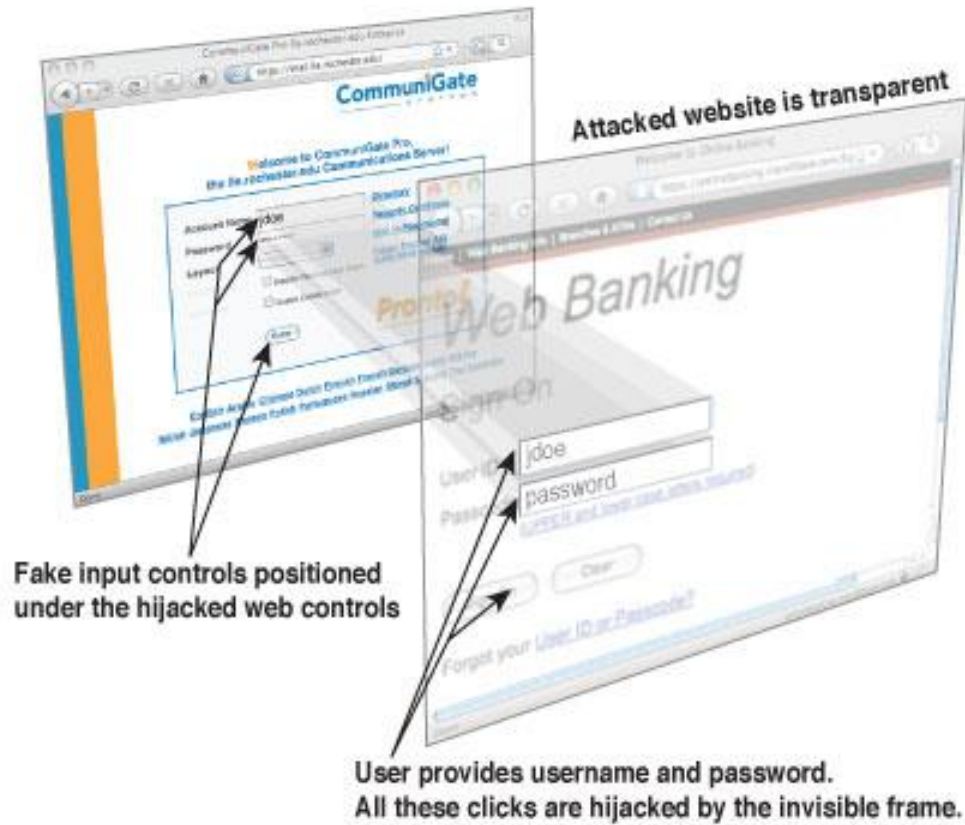


Clickjacking

Attack 3 is a Clickjacking attack

Uses malicious styling and framing to trick users into interacting with another website

Clickjacking





Clickjacking

For attack 3 you will need to get a form to be filled out with malicious info before submitting

Several options

- Use GET and POST params
- Trick user into typing form data
- Lots of other fancy tricks
- media.blackhat.com/bh-eu-10/presentations/Stone/BlackHat-EU-2010-Stone-Next-Generation-Clickjacking-slides.pdf



Clickjacking

You will also need to defeat some basic
framebusting

See link for ideas

<http://seclab.stanford.edu/websec/framebusting/framebust.pdf>



Tips

You have the source code! This is extremely useful for evading filters.

Firebug is your friend.



Defenses



Defenses - XSS

Sanitize all user inputs using strong filters

PHP: htmlspecialchars(str)

& => &	" => "
< => <	> => >

```
<a href='test'>Test</a> =>  
&lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
```

htmlspecialchars is not invincible! (attack 5)



Defenses - XSS

Do not let any user input into an eval call

Really you shouldn't use eval at all

htmlspecialchars is not invincible! (attack 5)

Think about quotes and event listeners



Defenses - CSRF

Need to verify that the request actually came from within the site

Secret token unique to each user

- For this project a hash of the session token is enough
- includes/auth.php has some helpful code



Defenses - Clickjacking

Framebusting code

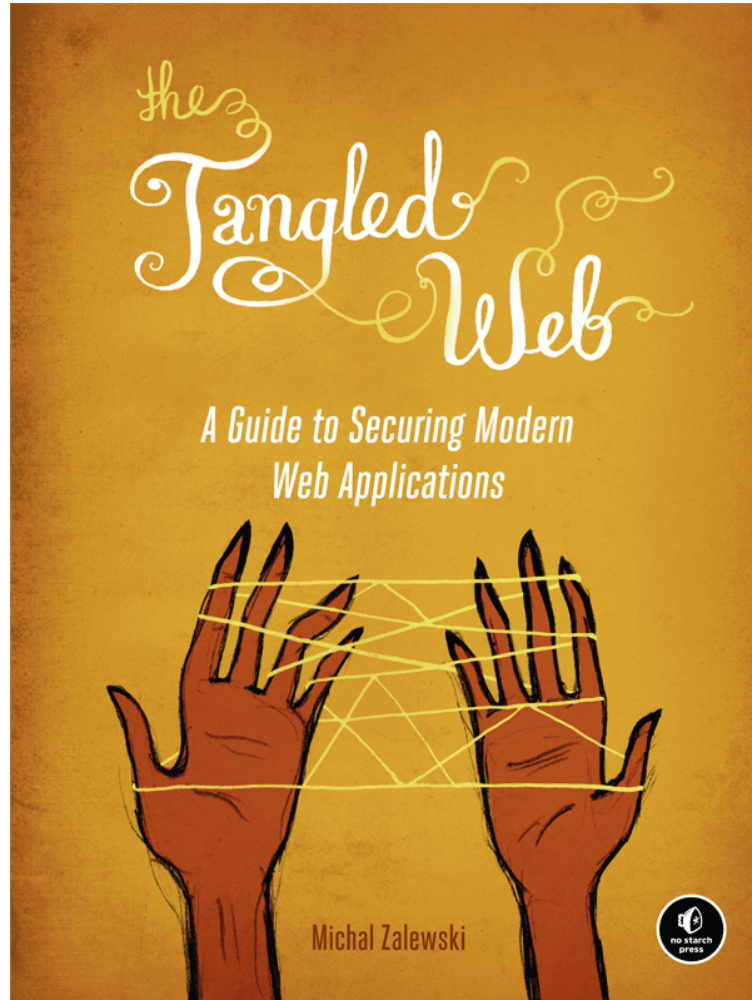
- Prevents a page from being loaded in an iframe

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

Not good enough!

- <http://seclab.stanford.edu/websec/framebusting/framebust.pdf>

Want to learn more?





Questions?
