

Web sikkerhed

Anders Skovsgaard
anders@trustskills.com



About Me

Founded first company in 1997.

- Helped: banks, media- and gambling companies, payment industry, CMS vendors...
- Google, Twitter, Typo3

Developing security scanner Hackavoid 2008-2014.

MSc @ AAU 2009, PhD in Computer Science @ AU 2014.

Now working at TrustSkills.

Outline

Cross-Site Scripting

SQL Injection / XPath Injection

Cross-Site Request Forgery

Clickjacking

DNS Rebinding

Mixed Recommendations

Cross-Site Scripting

What is Cross-Site Scripting

Also known as “XSS” or “CSS”.

Widespread in web applications.

- Test of 1.133 Danish web shops.
- ~50% vulnerable to XSS.
- Performed Feb 2011.



Problemstillingen bliver ikke mindre ved, at mange danske netbutikker har alvorlige sikkerhedsfejl og således potentielt kan indgå i fødekæden for stjalne kreditkortoplysninger. En undersøgelse foretaget af firmaet Hackavoid viste, at 49 procent af de undersøgte danske netbutikker havde kritiske sårbarheder¹⁹. Otte procent af de undersøgte butikker havde SQL-injection-sårbarheder, som blandt andet kan misbruges til at læse informationer i databasen eller placere skadelig kode på den besøgenes computer.

DK-CERT, 2010; "Trendrapport 2009".

Finansrådet, 2011; "Historisk få netbankindbrud".

Version2, 2011; "Hver anden danske webshop har alvorlige sikkerhedsfejl".

versionN



IT-NYHEDER

BLOGS

IT-JOB

IT-FIRMAER

WHITEPAPERS

EMNER *Nethandel, Sikkerhedshuller*

 [Se kommentarer \(11\)](#)

Hver anden danske webshop har alvorlige sikkerhedsfejl

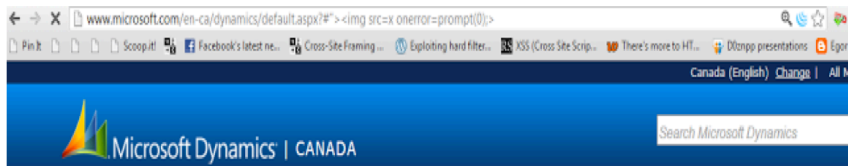
What is Cross-Site Scripting

Still highly relevant.

DOM Based XSS In Microsoft

Lately, i have been researching on DOM based XSS a bit, In my previous post i talked about the DOM based XSS i found inside AVG, DOM based XSS is caused due to lack of input filtering inside client side javascripts, since most of the code is moving towards client side, therefore DOM based xss have been very common now a days, It is predicted by the experts that the DOM based xss mostly occurs in the websites that heavily rely upon javascripts.

I have reported several DOM based XSS inside Microsoft, most of them were due to the lack of input filtering/sanitization inside of the several tracking scripts such as sitecatalyst and riotracking scripts as they often introduce some vulnerable sources and sinks. With that being said, let's take a look at the POC of the attack:



**hoo Mail users still seeing accounts hacked
XSS exploit amid reports Yahoo failed to fix
old flaw (Update: Fixed)**



Kim Dot Com's new Mega site has XSS Security Holes

By Dan Morrill on January 22, 2013

Any new site, not just Mega is going to have security holes, and reports have surfaced in Twitter, Reddit, and over on ZDNet that Mega has a couple of persistent XSS security holes that are going to make users days a little bit harder. Beyond the crypto issues that you can read on ZDNet, persistent XSS could allow a hacker to steal your Public/Private keys generated by the system at the worst case, in the best case, make it so that you cannot get to your data at all.

How I got a \$3,500 USD Facebook Bug Bounty

I recently found a Stored XSS on Facebook, which resulted in a Bug Bounty I know how an XSS could be exploited, you can read my colleague Mathias' blog. Anyway, here's how it went down.

I was actually working on finding flaws on Dropbox to begin with. I noticed the web interface there were some restrictions on what filenames that were allowed to rename a file to for example:

```
'"><img src=x onerror=alert(document.domain)>.txt
```

it was not possible. You got this error:

What is Cross-Site Scripting

May occur when:

- A Web Application accepts user inputs.



First Name

Last Name

E-mail

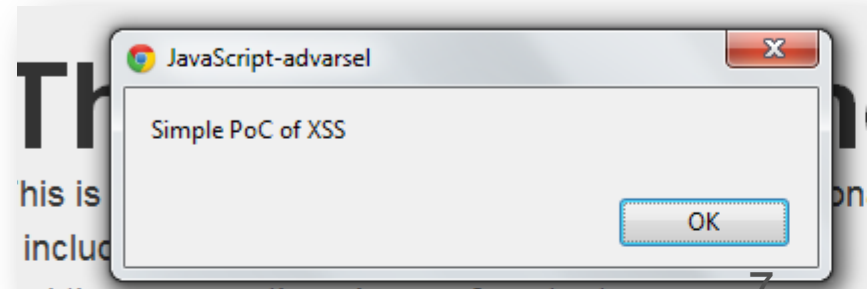
Service

Message

- Dynamic content is created from the input.

Thank you Name!

- Insufficiently validation or encoding of input.



What is Cross-Site Scripting

An attack involves:



A web server.

- Containing the web application (shop, community, CRM)



A client.

- A user accessing the web server.



An attacker.

- Internet user.


Attacker can inject script into a trusted *web application*.

Scripts are executed by the *client*.

What is Cross-Site Scripting

Three types of XSS:

- Reflective or Non-persistent
 - HTTP Request parameters are used server-side to generate the response.

 `www.trustedsite.com/?id=2&title=News"><script>alert('XSS')</script>`

- Persistent or Stored
 - Values are saved server-side and permanently displayed to users.

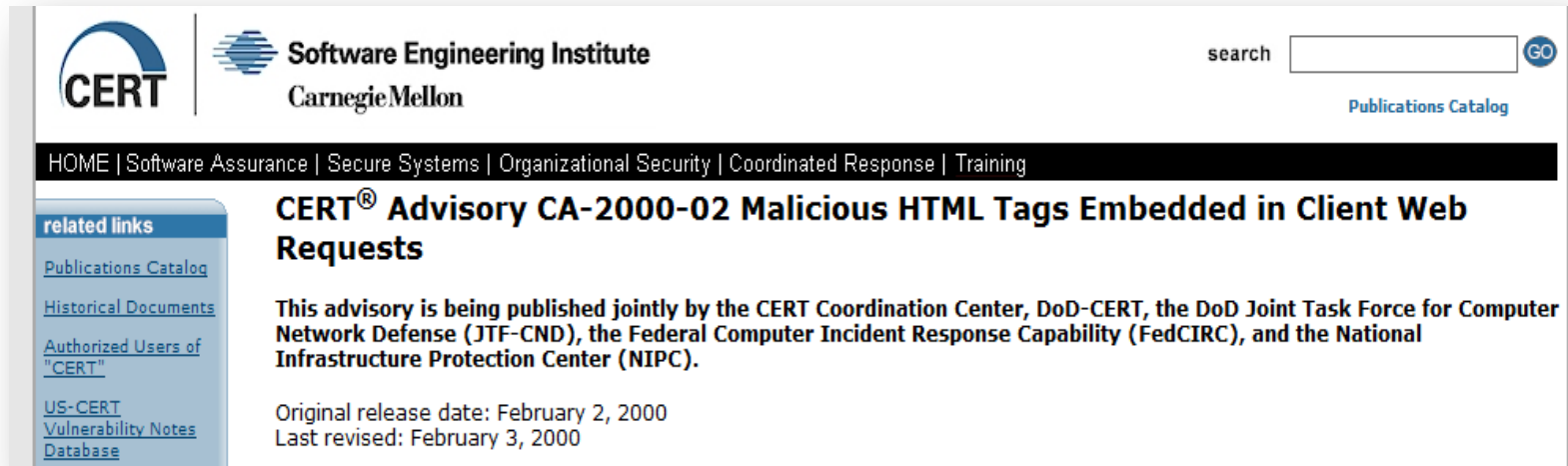


- DOM-based
 - Client side scripts handles input values without proper escaping.

 `www.trustedsite.com/?id=2#lang=<script>alert('XSS')</script>`

What is Cross-Site Scripting

Is XSS a new threat?



Original release date: **February 2, 2000**

*“A web site may inadvertently include **malicious HTML tags or script** in a **dynamically generated page** based on **unvalidated input** from untrustworthy sources. This can be a problem when a web server does not adequately ensure that generated pages are **properly encoded** to prevent unintended execution of scripts, and when input is **not validated** to prevent malicious HTML from being presented to the user.”*

Cross-Site Scripting Attacks

Should a web application with everything public (no login or CMS) be worried?

Real-world example:

- News article about XSS on Computerworld
December, 2010



- CTO at a bank replies:
- “It’s important to underline that our **costumers safely can use our web site** as usual. [...] Both parts (XSS) are **not dangerous**. It’s something where you can mix web pages and for example put in a picture and make a joke with us, or something.”

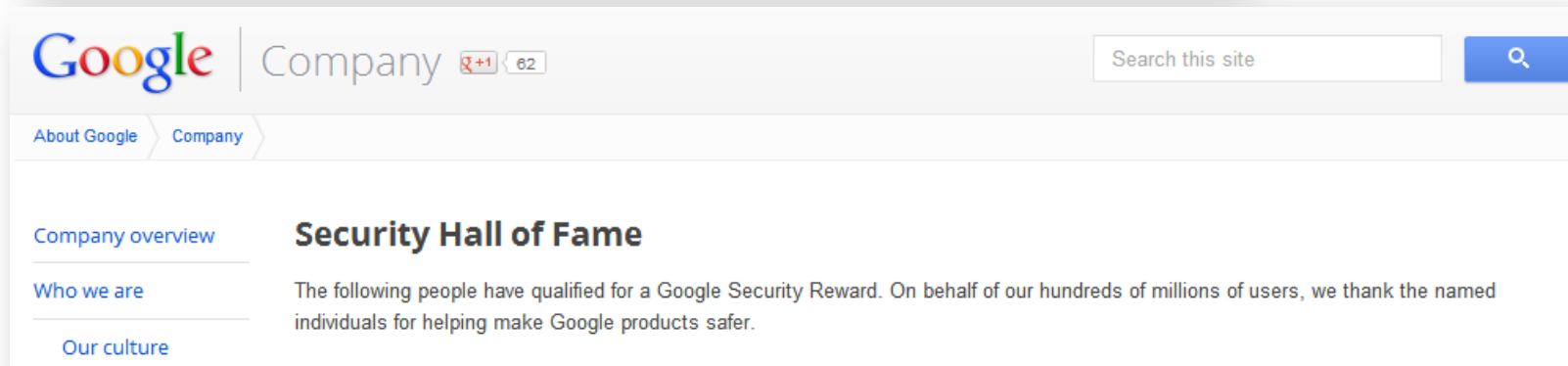


Demo

Cross-Site Scripting Attacks

Cross-Site Scripting Attacks

Add users and XSS becomes more dangerous.





Demo

Cross-Site Scripting Attacks

Cross-Site Scripting Attacks

The browsers are getting better.

- Safari, Chrome, (Firefox), Internet Explorer.

But they can not filter everything.

- E.g., `http://www.../error?message=<h1>Critical error...`

DOM-based Attack

- Today much work is done client-side (e.g., jQuery, AngularJS)
- Parameter values may not be sent to the server.
 - No logging.
 - WAF's of no use.

Cross-Site Scripting Attacks

DOM-based Attack Examples

```
<script>
document.write('<img src='+
decodeURIComponent(document.location
    .href.substring(document.location.href.index
Of("lang=")  + 5)) + '.png>');
</script>
```

```
/clientlanguage.aspx#lang=a onerror=alert('XSS') tag=
```

```
<img src=a onerror=alert('XSS') tag=.png>
```

```
$(function(){
    var hashValue = $(location.hash);
})
```

```
#<img src=/ onerror=alert(...)>
```

Preventing Cross-Site Scripting

Content-Security-Policy HTTP Header

- New browser feature
- Whitelists safe script hosts

Example:

Content-Security-Policy: script-src 'self'
ajax.googleapis.com

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/latest/  
jquery.min.js"></script>  
<script src="/js/app.js"></script>  
<script src="//attacksite.ru/pwn.js"></script>  
<script>alert(document.cookie);</script>
```

Does not load //attacksite.ru/pwn.js and
alert(document.cookie); because of the directive.

Preventing Cross-Site Scripting

What is an input?

Request URL: `http://gotocon.com/cph-2012/`

Request Method: GET

Status Code:  200 OK

▼ Request Headers [view source](#)

Accept: `text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`

Accept-Charset: `ISO-8859-1,utf-8;q=0.7,*;q=0.3`

Accept-Encoding: `gzip,deflate,sdch`

Accept-Language: `da-DK,da;q=0.8,en-US;q=0.6,en;q=0.4`

Cache-Control: `max-age=0`

Connection: `keep-alive`

Cookie: `JSESSIONID=TRIFORK84267487841`

Host: `gotocon.com`

Referer: `http://gotocon.com/`

User-Agent: `Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.19 (KHTML, like Gecko)`

Do not trust any input.

Preventing Cross-Site Scripting

Do not trust client-side validation.

Input Validation

- Check if input is as expected.
- Do not make black lists – white lists are better.
- E.g., an age field should only consists of Integers.

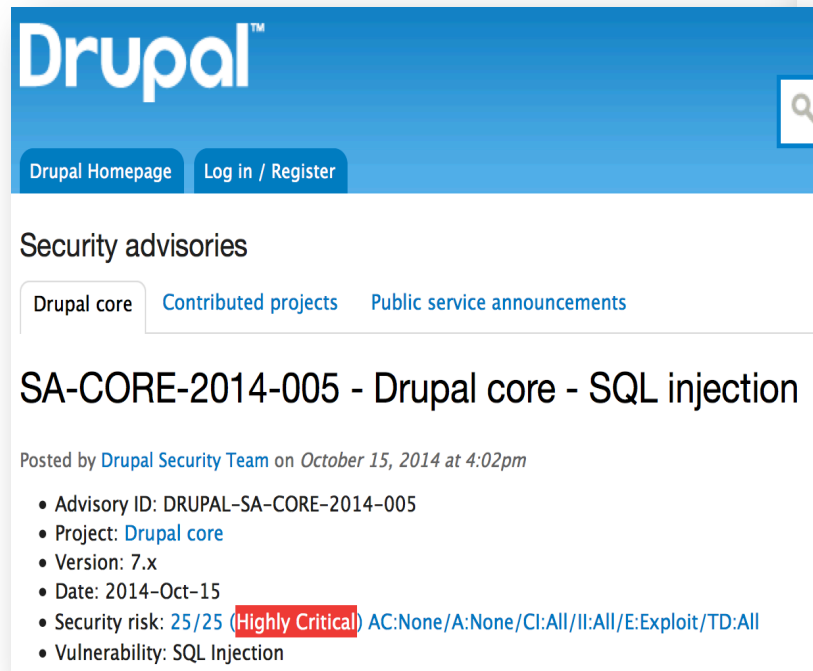
HTML Encode Input

- Not only HTML tags.
 - Used in ASP.NET Request Validation.
 - JavaScript events may be used.

Mark Cookies “httpOnly”.

SQL Injection / XPath Injection

SQL Injection – still relevant?



Drupal

Drupal Homepage Log in / Register

Security advisories

Drupal core Contributed projects Public service announcements

SA-CORE-2014-005 - Drupal core - SQL injection

Posted by [Drupal Security Team](#) on *October 15, 2014 at 4:02pm*

- Advisory ID: DRUPAL-SA-CORE-2014-005
- Project: [Drupal core](#)
- Version: 7.x
- Date: 2014-Oct-15
- Security risk: 25/25 (Highly Critical) AC:None/A:None/CI:All/II:All/E:Exploit/TD:All
- Vulnerability: SQL Injection

SECURITY

SQL injection flaw in Wall Street Journal database led to breach



Jeremy Kirk

Jul 23, 2014 3:45 AM



Vulnerable to SQL Injection and Remote Code Execution

April 13, 2014 Wang Wei

Here's how Bell was hacked – SQL injection blow-by-blow

Tuesday, 4 February 2014

What is SQL Injection?

The flaw is in the web application or stored procedure, not the DBMS itself.

Involves only two players: the web-server and the attacker.



Example stored procedure using HTTP input:

```
ALTER PROCEDURE dbo.Search @Term VARCHAR(50)
```

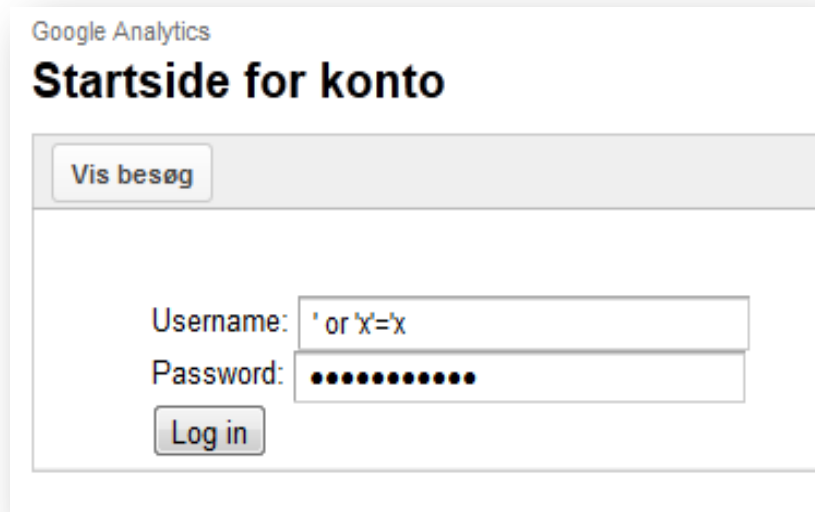
```
...
```

```
    SET @query = 'SELECT Article FROM dbo.News WHERE Article LIKE  
' + '%' + @Term + '%'  
    EXEC(@query)
```

SQL Injection Attacks

Attacking a login form.

- var sql = "SELECT * FROM Users
WHERE username=''" + Request.QueryString('username') + "'
AND password=''" + Request.QueryString('password') + "'"



Google Analytics

Startside for konto

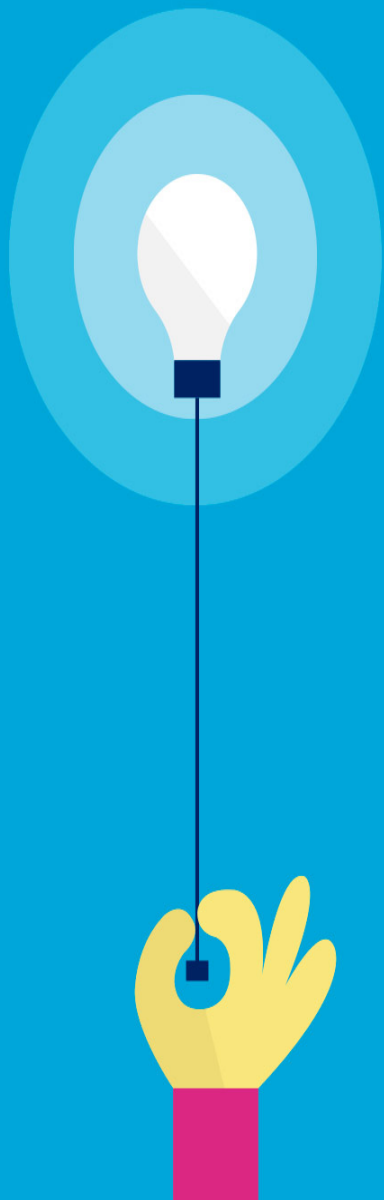
Vis besøg

Username: ' or 'x'='x'

Password:

Log in

- var sql = "SELECT * FROM Users
WHERE username=''" + Request.QueryString('username') + "' or 'x'='x'
AND password=''" + Request.QueryString('password') + "' or 'x'='x'"



Demo

Extracting sensitive information
from tables.

SQL Injection Attacks

Blind SQL Injection

- No output or error messages.

Real-world example:

```
string pass = String.Empty;
foreach (char c in PASSWORD_CHARS) {
    string url = "http://ANON.com/portfolio_ANONCompanies.php?port_id={0}";
    string hack = "-1 UNION SELECT BENCHMARK(3000000,MD5(user_email)),2
                  FROM wp_users WHERE user_email LIKE '" + pass + c
+ "%'";
    if (GetTimeToRequestPage(String.Format(url, hack)) > 3)
    {
        pass += c;
    }
}
```

Preventing SQL Injection

Often recommendations of escaping strings.

- Escaping ' with \.

What if you write:

```
SELECT * FROM News
```

```
WHERE
```

```
Id="+REPLACE(Request.QueryString('username'),"'","\\'")
```

An attacker could do:

```
SELECT * FROM News
```

```
WHERE Id=-1 UNION SELECT username, password FROM Users
```

Preventing SQL Injection

Instead, use Prepared Statements

- Used by ORM internally.

Explicit:

First define all SQL code, then later pass each parameter.

```
var sql= "SELECT * FROM News WHERE Id = ?";  
[...] command.Parameters.Add(new OleDbParameter("customerId",  
    PARAMETER_VALUE));
```

No attacker can change the intent of the query.

However, stored procedures may concatenate query strings.

- *Use prepared statements.*

Cross-Site Request Forgery

What is Cross-Site Request Forgery?

Also known as XSRF, CSRF or Session Riding.

Involves three players:



A web server.

Not protected against XSRF.



A client.

Who creates the actual request.



An attacker.

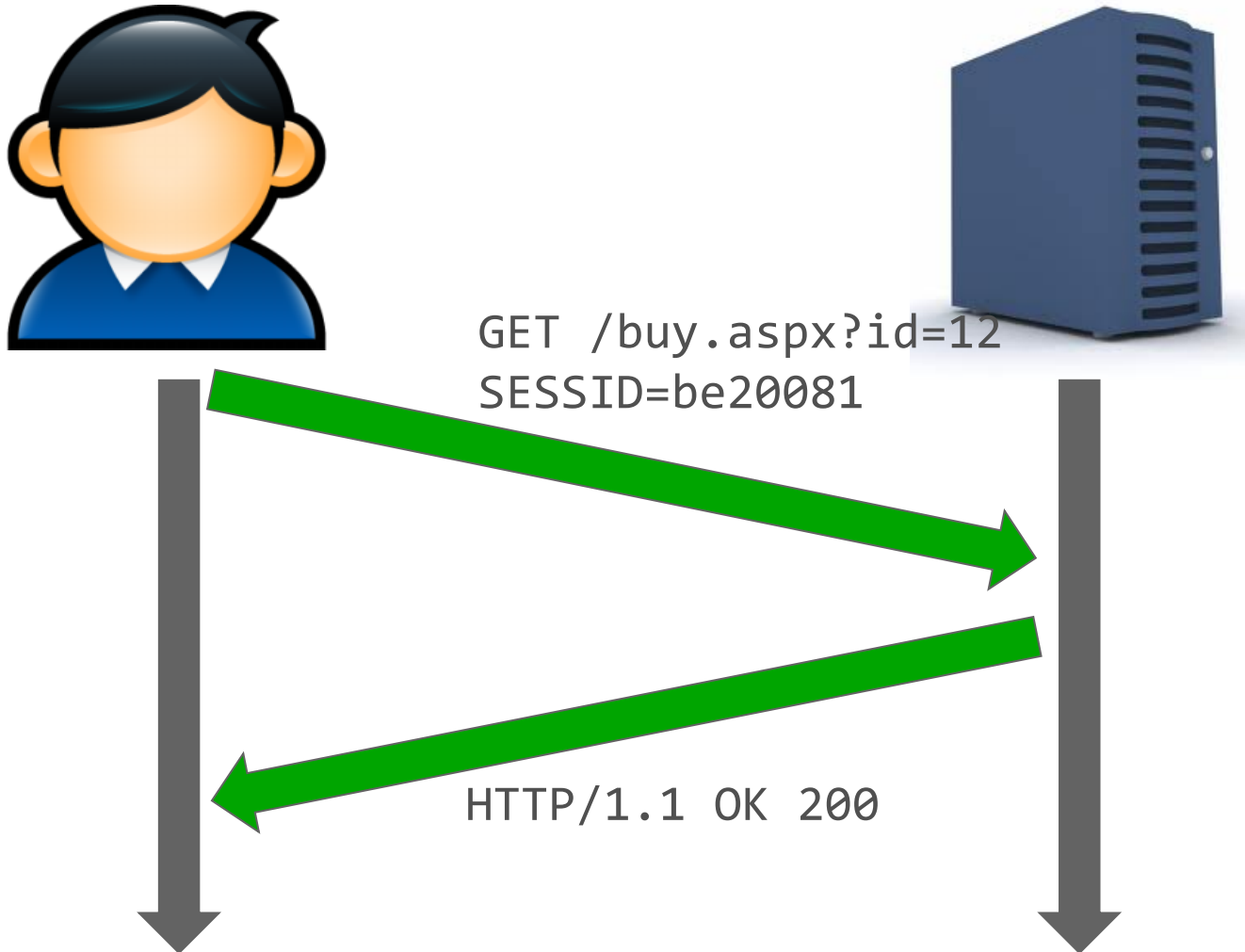
Who makes the client do the request.

The attacker has to know.

- What requests is accepted.
- The user must be authenticated.

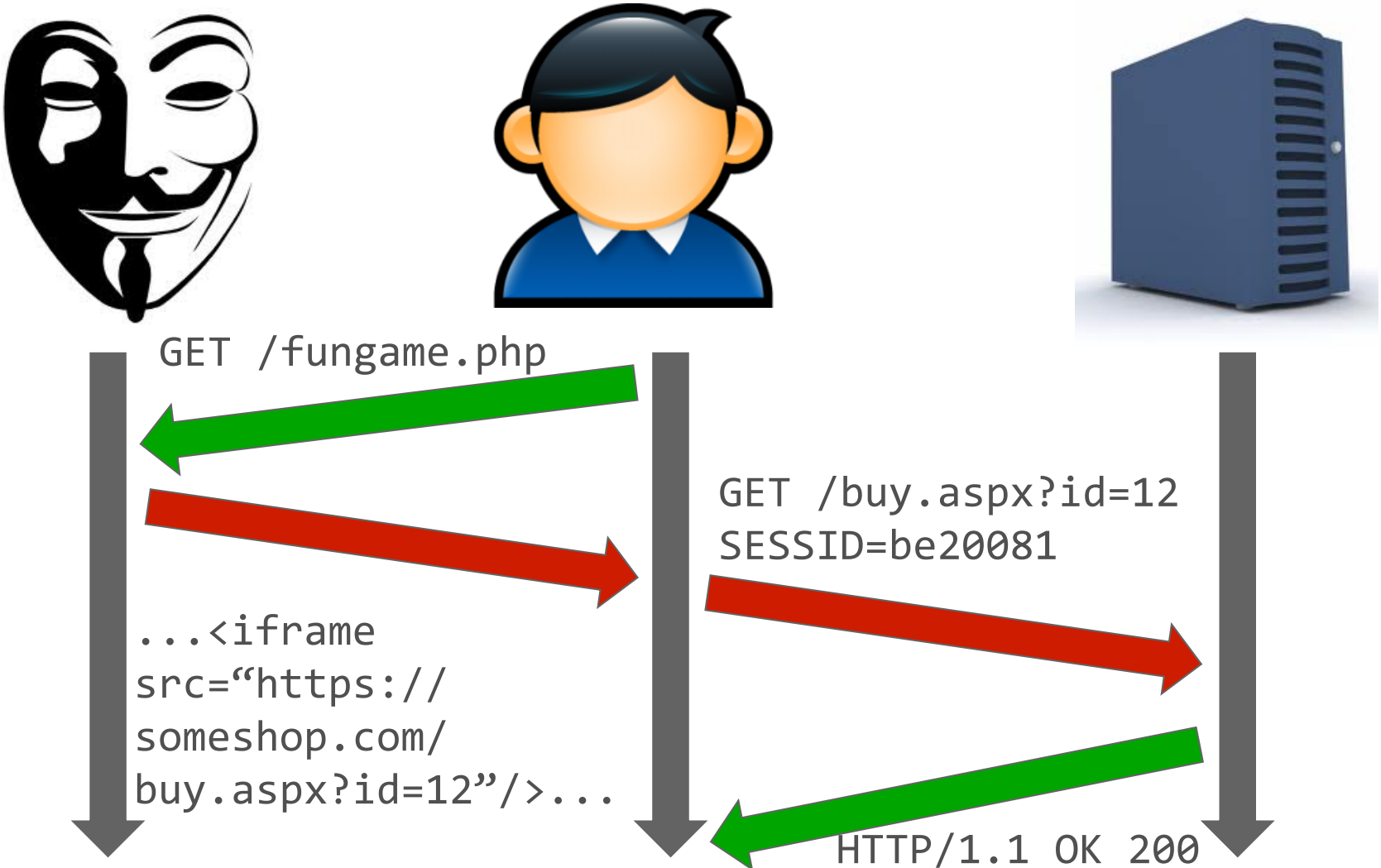
What is Cross-Site Request Forgery?

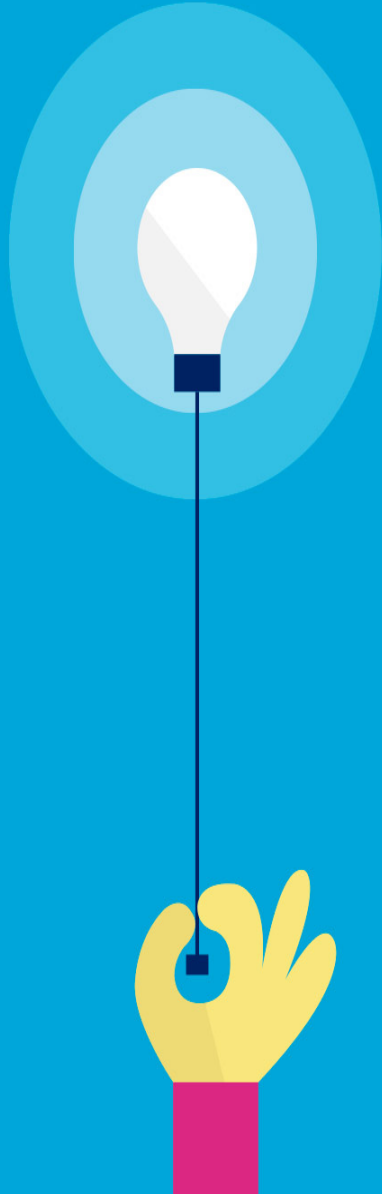
A normal request occurs when, e.g. a user press a “buy”-button.



What is Cross-Site Request Forgery?

The “buy”-request may be forced by an attacker.





Demo

Cross-Site Request Forgery Attack

Preventing Cross-Site Request Forgery

The simple solution: Make a check on the REFERER.

- You will loose visitors with no REFERER.
- Spoofing examples for old versions of Flash, Firefox and IE exists.

The best solution: Require a secret, user-s

- All sensitive requests.
- An attacker cannot guess the token and make the

```
<form action...>
```

```
<input name="product" type="hidden" value="12"/>
```

```
<input name="xsrftoken" type="hidden"  
  value="df8652852f139"/>
```

```
</form>
```

**But this can be
extracted with
XSS!**

Remember AJAX calls.

Clickjacking

What is Clickjacking?

Also known as UI redressing or “L

An attack involves:



A web server.



A user.



An attacker controlling a v

twitter blog

Clickjacking Blocked

Thursday, February 12, 2009

Some folks have noticed links from accounts they follow prefacing the link with “click” which of course people want to click right away. The link is employing a technique called clickjacking. This technique seeks to trick you into clicking on a link that you didn't intend to. We can take action on your behalf while you perform seemingly un

Clickjacking

▼ What is clickjacking?

Clickjacking is when scammers load fake buttons and icons to trick people into making unwanted actions on Facebook. For example, scammers can load an invisible Facebook Like button and put it on top of a different button. Then when you click on the visible button, you're actually clicking to Like something on Facebook you didn't intend to.

More info

[Get help for mobile apps and browsers](#) ▶

Last edited about 7 months ago

Was this answer helpful? Yes · No

[Permalink](#) · [Share](#)

▼ What should I do if I clicked on something that might be spam?

If you've clicked on something that turned out to be spam, try these steps to fix it:

- Check your [Activity Log](#) and delete any unwanted actions
- Check your [installed apps and games](#) and delete anything you don't trust
- Check your [login history](#) for any suspicious logins. If there are any, [learn how to secure your account](#).
- Install and run anti-virus software

BBC News Sport Weather Travel Future TV R

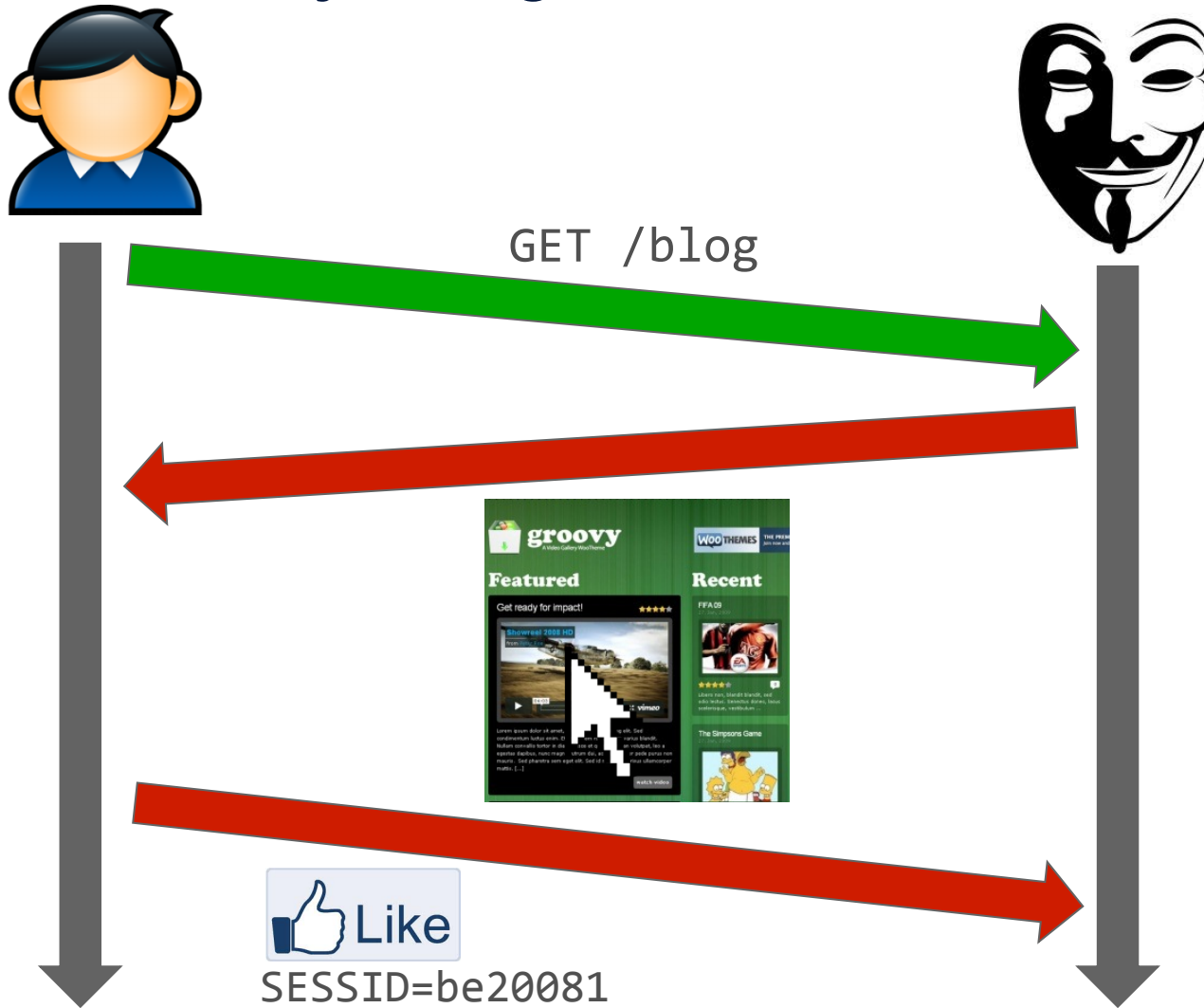
NEWS TECHNOLOGY

Home UK Africa Asia Europe Latin America Mid-East US & Canada Business Health Sci/Enviro

27 January 2012 Last updated at 22:04 GMT

Facebook sues alleged clickjacking spammer sparking row

What is Clickjacking?





Demo

Clickjacking Attack



Preventing Clickjacking

Popular Frame Breaking Scripts:

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

Limitations:

- Double Framing (top.location is a security violation)
- Exploit XSS Filter (<iframe src="http://www.victim.com/?v=<script>if">)

Preventing Clickjacking

“Best-for now” script:

```
<head><style>body { display : none;}</style></head>
<body>
<script> if (self == top) {
var theBody = document.getElementsByTagName
('body')[0]; theBody.style.display = "block";
} else {
top.location = self.location; }
</script></body>
```

Other approaches:

- HTTP Response Header “*X-FRAME-OPTIONS*”.
 - Either *DENY* or *SAMEORIGIN*.
- **Content Security Policy:**
 - Content-Security-Policy: frame-options 'deny'
 - Content-Security-Policy: input-protection tolerance=15
- Supported by recent versions of most browsers.

DNS Rebinding

What is DNS Rebinding?

If *hotmail.com* has no XSS or SQLi, how to extract (IP-filtered) user specific data?

Make IFRAME with *hotmail.com* on *hmattacker.ru* and execute:

```
document.getElementById('hmframe').contentWindow  
    .document.body.innerHTML
```

?

Or XMLHttpRequest?

Not possible because of the *Same-Origin Policy* in the browser.

What is DNS Rebinding?



Involves a web server, a user and an attacker.

A domain name resolves to an IP-address.

- E.g., *intra.net* resolves to *10.0.0.4*.

Requirements:

- The web server must accept other host names.
- E.g., if *intraattacker.ru* resolves to *10.0.0.4* the *IntraNet* web page is shown at *intraattacker.ru*.
- The user must be authorized at *intra.net* (e.g., IP-filter, or LAN application).
- The user must visit *intraattacker.ru*.

What is DNS Rebinding?

The attacker controls the name server (NS) of `intraattacker.ru`.

First, make record in the NS:

`intraattacker.ru. A 87.238.13.37 (TTL: 1 second)`

When a *intranet* user enters the site, make the change:

`intraattacker.ru. A 10.0.0.4`

- Wait for the browser's DNS cache to expire.
- Create an IFRAME with `src=//intraattacker.ru`.
 - Now the IFRAME will contain content from `10.0.0.4`.
 - The outer frame can extract content from the inner frame (user data if IP-filtered, LAN apps and Session fixation).

Preventing DNS Rebinding

Web application developers:

- Make a white-list of host headers.
- Reject all request with unknown host headers.

Paranoid Internet users:

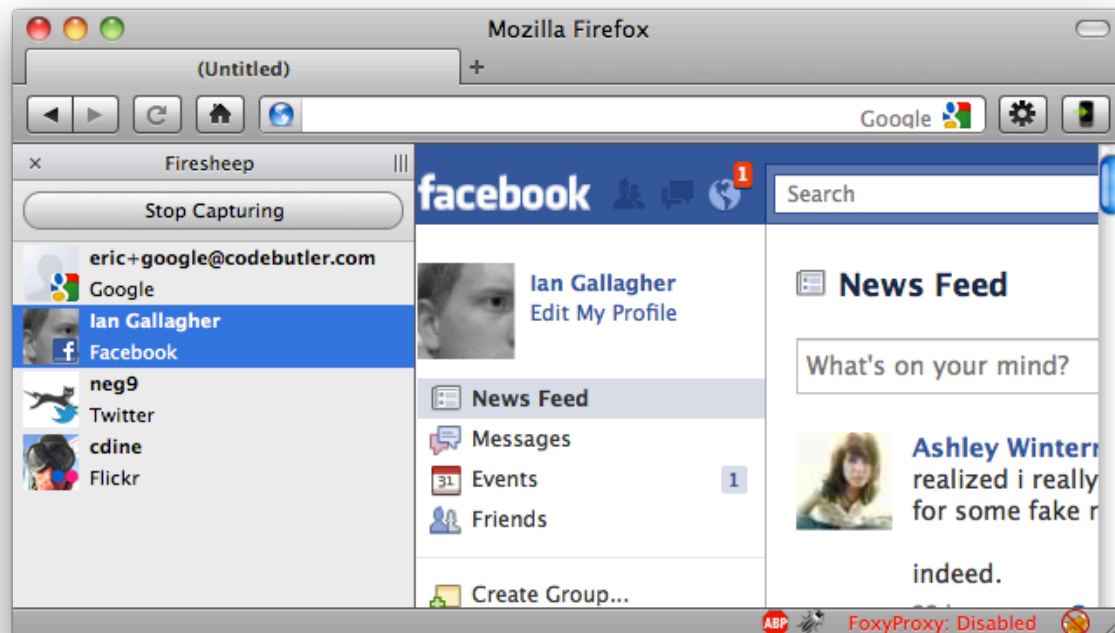
- NoScript provides protecting.
- Adjust your DNS Cache.

Mixed Recommendations

Mixed Recommendations

Use SSL

- Avoid sniffing.



- Set the HTTP Strict Transport Security (HSTS) header.
- Use SHA256 (rekey your existing SHA1 for free).
- Make sure CN matches the hostname.
- Renew before expiry.

Mixed Recommendations

Disable debug info.

ASP.NET Trace Information <http://www.domain.com/Trace.axd>

Request Details

Request Details

Session Id:	sbrysnrex3ok211c3yrr02h	Request Type:	POST
Time of Request:	12/10/2011 4:58:28 PM	Status Code:	302
Request Encoding:	Unicode (UTF-8)	Response Encoding:	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.000232113944936614	0.000232
aspx.page	Begin Init	0.000263878800550907	0.000032
aspx.page	End Init	0.000356538379371003	0.000093
aspx.page	Begin InitComplete	0.000377738283007805	0.000021
aspx.page	End InitComplete	0.000396473197849101	0.000019
aspx.page	Begin LoadState	0.000409878136917559	0.000013
aspx.page	End LoadState	0.000938465734246662	0.000529
aspx.page	Begin ProcessPostData	0.00100324543979346	0.000065
aspx.page	End ProcessPostData	0.0018416216289926	0.000838
aspx.page	Begin PreLoad	0.00189972636488016	0.000058
aspx.page	End PreLoad	0.00192501624992614	0.000025
aspx.page	Begin Load	0.00193954618388098	0.000015
aspx.page	End Load	0.00199779091913219	0.000058
aspx.page	Begin ProcessPostData Second Try	0.00201457584283708	0.000017
aspx.page	End ProcessPostData Second Try	0.0020312457670647	0.000017
aspx.page	Begin Raise ChangedEvents	0.00204484570524679	0.000014

Mixed Recommendations

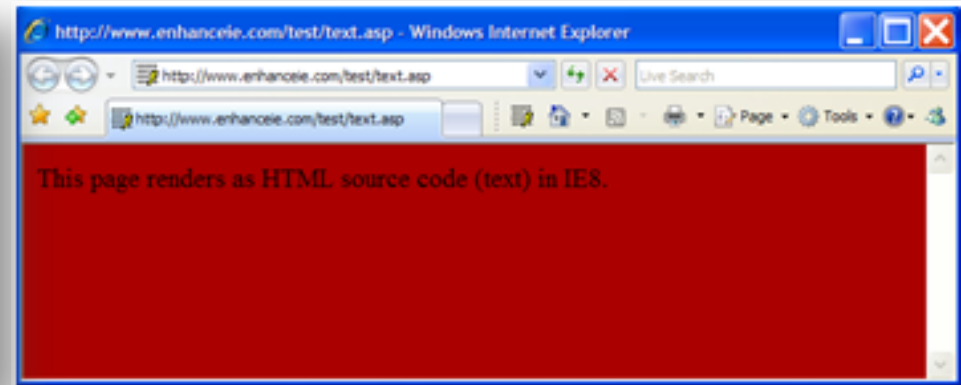
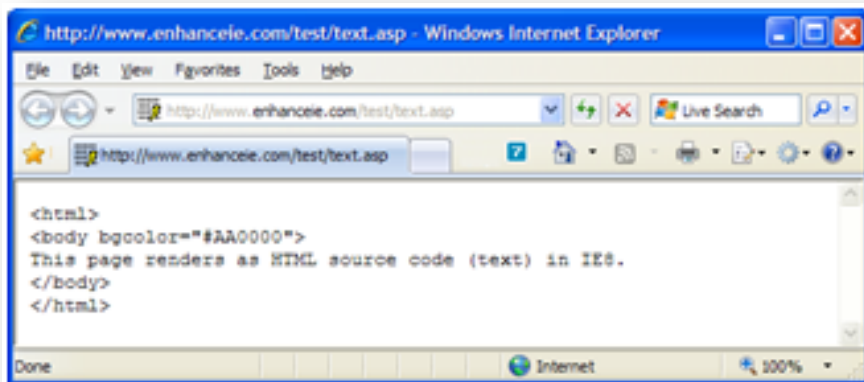
Session Hijacking

- 1. httpOnly and Secure cookies.
- 2. Use a long session identifier.
- 3. Retire sessions.
- 4. Renew session token at each login.

Mixed Recommendations

Avoid MIME-sniffing.

```
HTTP/1.1 200 OK
Content-Length: 108
Date: Thu, 26 Jun 2008 22:06:28 GMT
Content-Type: text/plain;
X-Content-Type-Options: nosniff
<html><body bgcolor="#AA0000">
This page renders as HTML source code (text) in IE.
</body></html>
```



Mixed Recommendations

Redirects

- **Avoid unvalidated redirects**
 - `https://www.example.com/redir?p=http://www.attacksite.com`
 - `https://www.example.com/redir?p=javascript:alert(document.cookie)...`
- **Redirect properly**
 - `Response.AddHeader("Location", "http://www.domain.com/page");`
 - ...
`<h1>Administration Module</h1>`
`User Credentials: ...`
 - Body is still readable.

Mixed Recommendations

- File retrieval:
`/Pdf/ShowPdf.aspx?PDfFile=/Web.config`
- Login-form brute force.
- DoS:
`/tools/getImage.php?Id=289&h=999999999&w=999999999`
- Hash password.
- Protect pages and objects (`/Administration` or `?id=1`)
- Keep your CMS and dependencies up-to-date.
 - Retire.js (<http://retire.insecurity.today/>)

What to do now?

Manually check your web application.

- Test all pages and parameters.

Use automated tools.

- Skipfish, w3af, Nessus, nmap, WebSecurify, McAfee Secure, Acunetix, Qualys.

(Install a Web Application Firewall (WAF))

More material at www.owasp.org.

- The Open Web Application Security Project.

Training: 16/17-04-2015 in Aarhus, 23/24-04-2015 in Cph.

Questions?

Anders Skovsgaard
anders@trustskills.com
www.trustskills.com



Vulnerability Scanners

Web Site Coverage

Product	Simple HTML	Basic JavaScript	Advanced JavaScript (DOM/ jQuery)
Acunetix	Yes	Yes	No
Sucuri (only malware)	Yes	No	No
WebSecurify	Yes	No	No
Nessus	Yes	No	No
W3af	Yes	No	No
Qualys	Yes	Yes	No

Vulnerability Scanners

Thorough Generics Tests

- **What you scan** for and how well you do it is important.

Product	OWASP Top 10
Acunetix	Yes
Sucuri (only malware)	No
WebSecurify	Yes
Nessus	Yes
W3af	Yes
Qualys	Yes

Vulnerability Scanners

Thorough Generics Tests

- What you scan for and **how well you do it** is important.

Product	IMG Embedded	Auto Detection	Embedded in JS
Acunetix	Yes	Yes	Yes
Sucuri (only malware)	No	No	No
WebSecurify	Yes	No	No
Nessus	No	No	No
W3af	No	No	No
Qualys	Yes	No	No

Vulnerability Scanners

The type of scanner you need.

Product	SaaS	Free scan
Acunetix	No	No
Sucuri (only malware)	Yes	Yes
WebSecurify	No	Yes
Nessus	No	Yes
W3af	No	Yes
Qualys	Yes	Yes