The First, Toughest and Messiest XSS Filter *Ever*

Giorgio Maone
giorgio@maone.net

OWASP
The Open Web Application Security Project

about **Giorgio Maone** (@ma1)

- Full time dad

about **Giorgio Maone** (@ma1)

OWASP
The Open Web Application Security Project

- Full time dad
- **NoScript** creator & maintainer
- #9 Most Dangerous People on the Internet
- **Hackademix** breaker + builder
- **Mozilla** contributor & Sec. Group member
- **W3C** WASWG invited expert

noscript.net

about **NoScript**

- JavaScript permission manager
- Embedded content blocker
- Application Boundaries Enforcer (ABE)
- ClearClick (Clickjacking protection)
- HTTPS enhancements
- Usability helpers
- **Cross Site Injection Checker**

noscript.net

OWASP
The Open Web Application Security Project

Injection Checker basics

- Hooks **cross-site** HTTP requests
- Checks **document** loads
- If triggered, **transforms the request**
- Sanitizes the **document rendering context** if needed
- Notifies user with analyze/bypass options

The Injection Checker module

OWASP
The Open Web Application Security Project

Hooks **cross-site** HTTP requests

https://a.net → https://b.com	YES
http://b.com → https://b.com	YES
https://b.com/a → https://b.com/b	NO
https://a.net → ftp://b.com/	NO
Navigation bar → https://b.com	YES
External application → https://b.com	YES

The Injection Checker module

OWASP
The Open Web Application Security Project

Hooks **cross-site** HTTP requests

- Pages reloaded
  on Javascript activation! YES

The Injection Checker module

OWASP
The Open Web Application Security Project

Checks **document** loads

- HTML pages
- SVG objects
- (I)Frames
- Generic <OBJECT> inclusions

The Injection Checker module

OWASP
The Open Web Application Security Project

If triggered, **transforms** the request

- Strips **POST payloads** from untrusted origins (rudimentary CSRF protection)
- Sanitizes syntactically **valid JavaScript** (when the document to be loaded is allowed)
- Sanitizes potentially **dangerous HTML**
- Turns suspect POSTs into GETs

The Injection Checker module

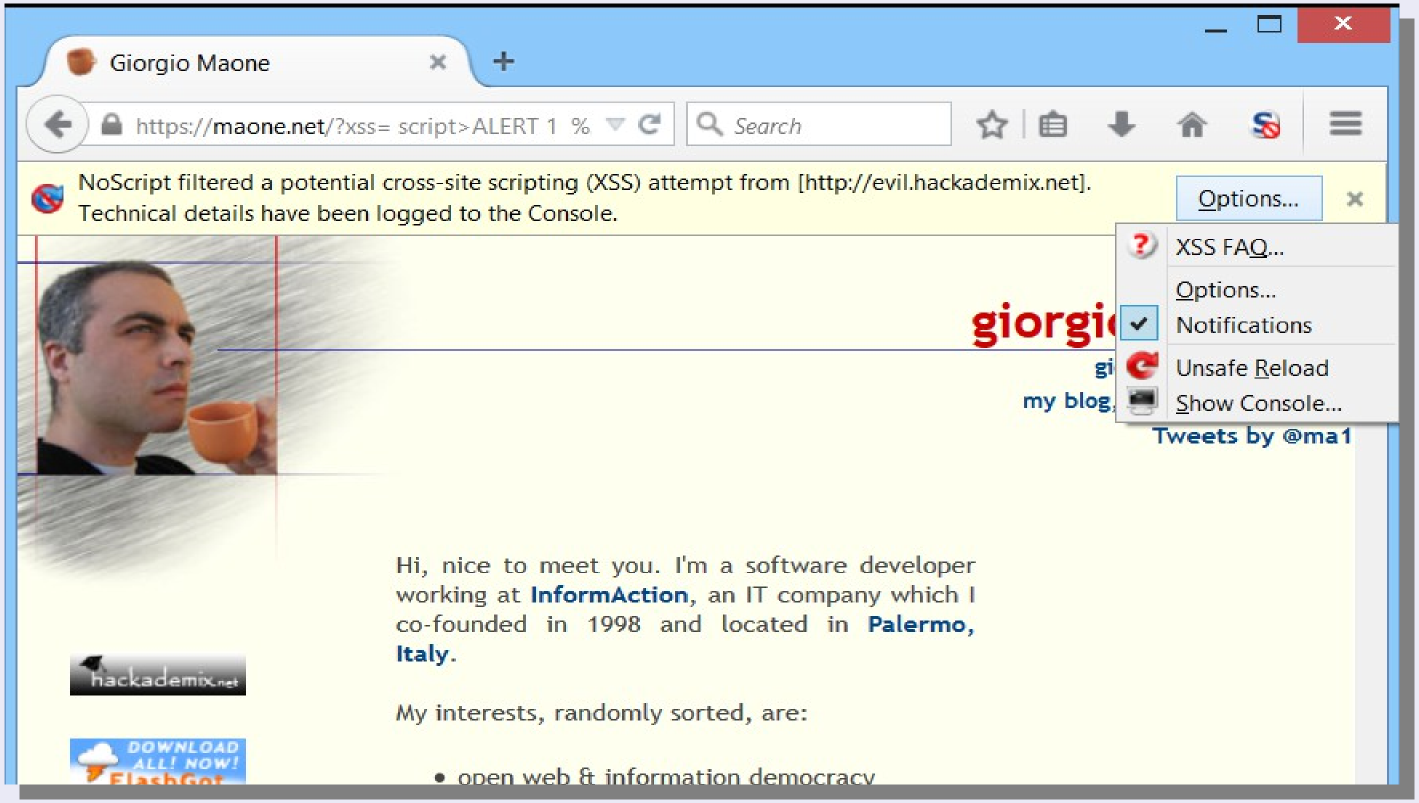
OWASP
The Open Web Application Security Project

Sanitizes the **document rendering context** if needed

- Forces UTF-8 if a potentially dangerous and unusual char-set is found
- Removes potential injections from `window.name`

The Injection Checker module

Notifies user with analyze/bypass options



NoScript filtered a potential cross-site scripting (XSS) attempt from [http://evil.hackademix.net]. Technical details have been logged to the Console.

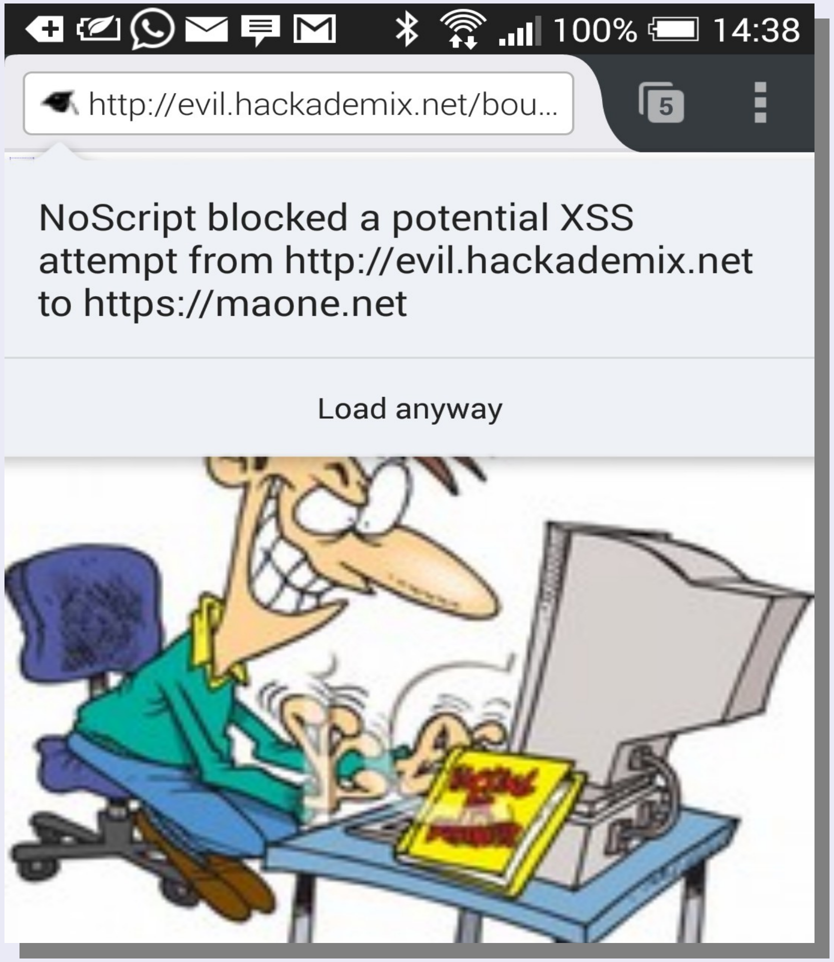
Hi, nice to meet you. I'm a software developer working at InformAction, an IT company which I co-founded in 1998 and located in Palermo, Italy.

My interests, randomly sorted, are:

- open web & information democracy

OWASP
The Open Web Application Security Project

Notifies user with analyze/bypass options



NoScript blocked a potential XSS attempt from http://evil.hackademix.net to https://maone.net

The Injection Checker module

OWASP
The Open Web Application Security Project

Notifies user with analyze/bypass options

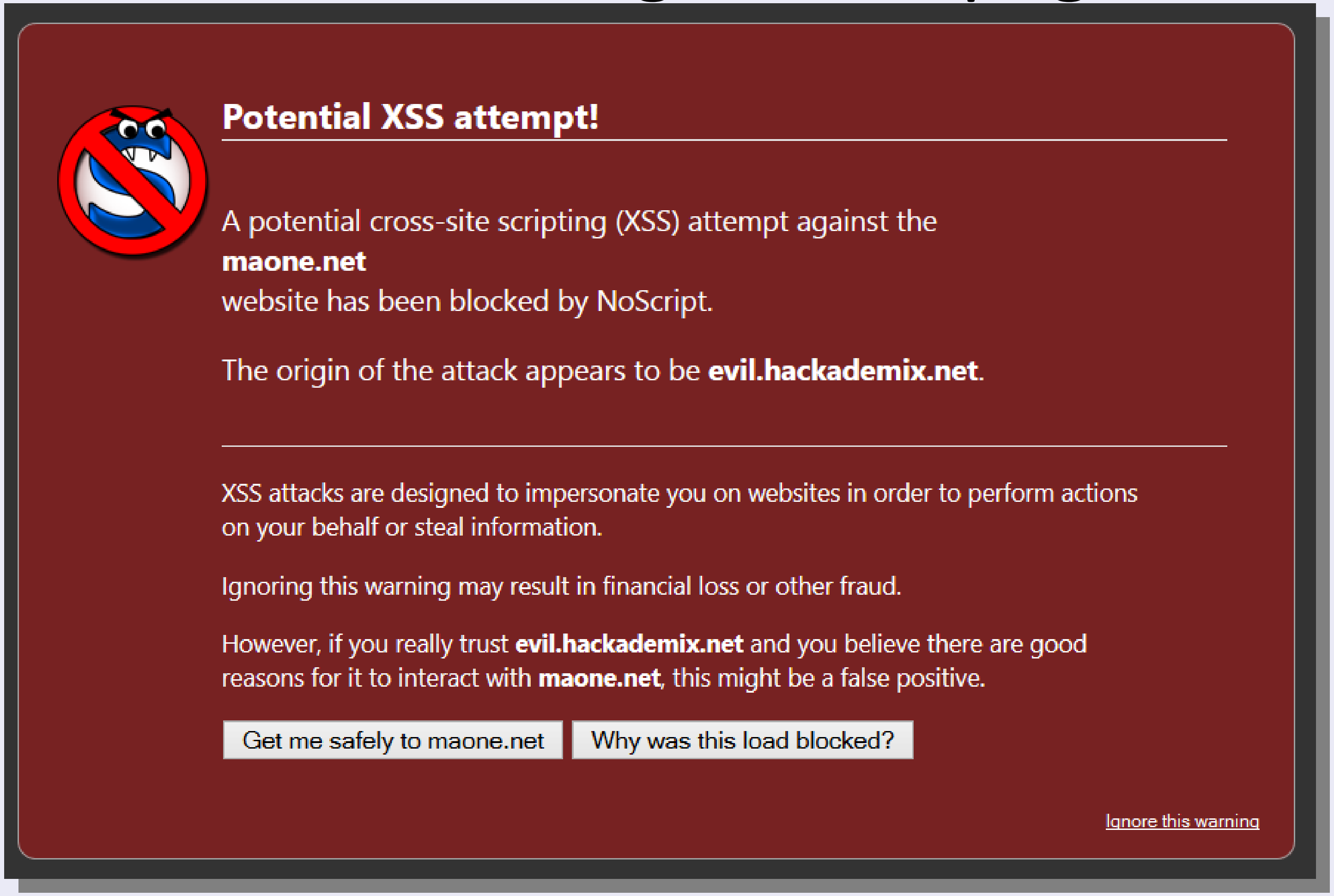
IN MEDIO STAT VIRTUS

THOU SHALL
NOT REINVENT THE WHEEL

ETC. ETC.

OWASP
The Open Web Application Security Project

Hard blocking + Error page



Potential XSS attempt!

A potential cross-site scripting (XSS) attempt against the
maone.net
website has been blocked by NoScript.

The origin of the attack appears to be **evil.hackademix.net**.

XSS attacks are designed to impersonate you on websites in order to perform actions on your behalf or steal information.

Ignoring this warning may result in financial loss or other fraud.

However, if you really trust **evil.hackademix.net** and you believe there are good reasons for it to interact with **maone.net**, this might be a false positive.

OWASP
The Open Web Application Security Project

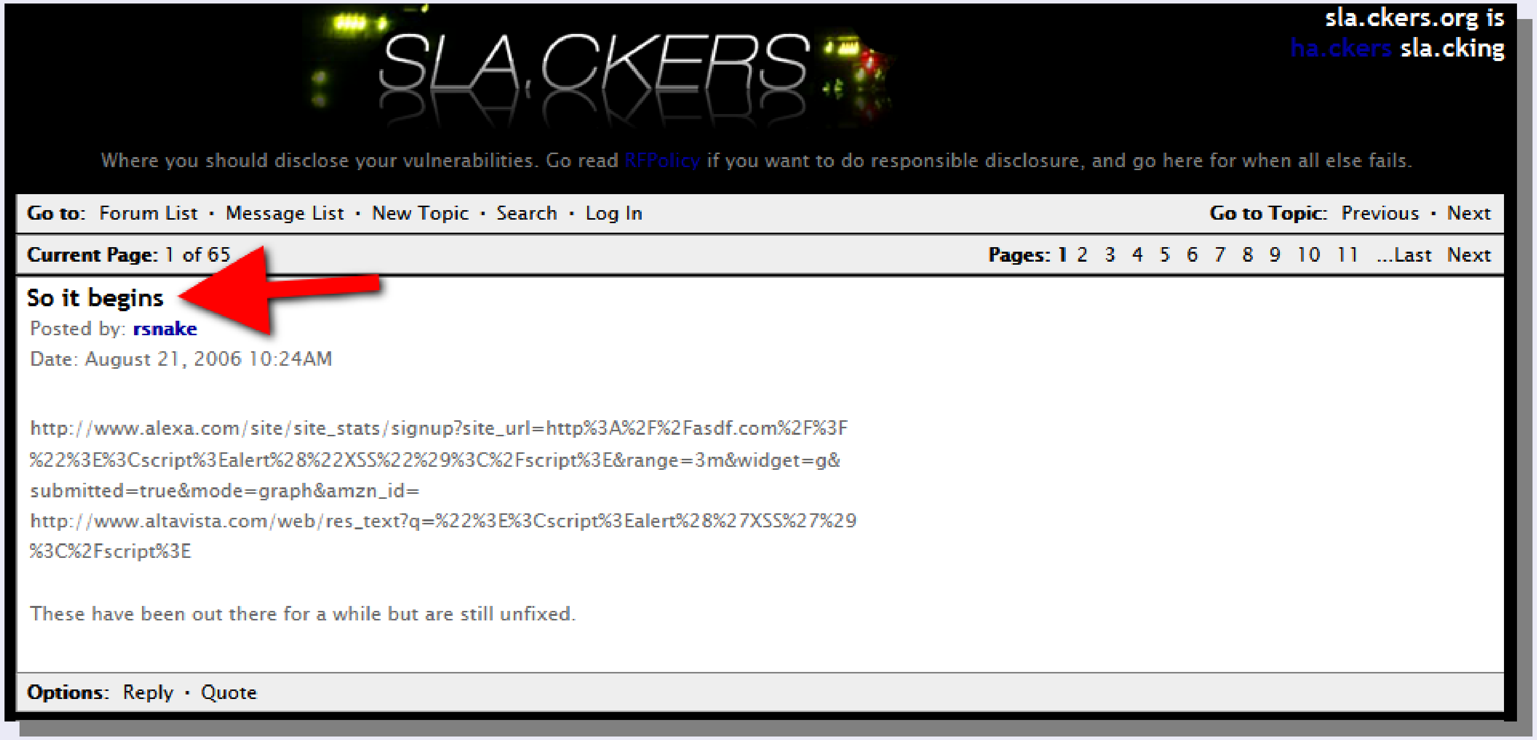
Imminent changes

- Hooks **cross-site** HTTP requests
- Checks **document** loads
- If triggered, ~~**transforms**~~ **suspends the request**
- ~~**Sanitizes the document rendering context if needed**~~
- Notifies user ~~**with analyze/bypass options**~~ using a “Safe Browsing-like” page

Origins

OWASP
The Open Web Application Security Project

Once upon a time...



Where you should disclose your vulnerabilities. Go read RFPolicy if you want to do responsible disclosure, and go here for when all else fails.

Go to: Forum List · Message List · New Topic · Search · Log In **Go to Topic:** Previous · Next

Current Page: 1 of 65 **Pages:** 1 2 3 4 5 6 7 8 9 10 11 ...Last Next

So it begins
Posted by: **rsnake**
Date: August 21, 2006 10:24AM

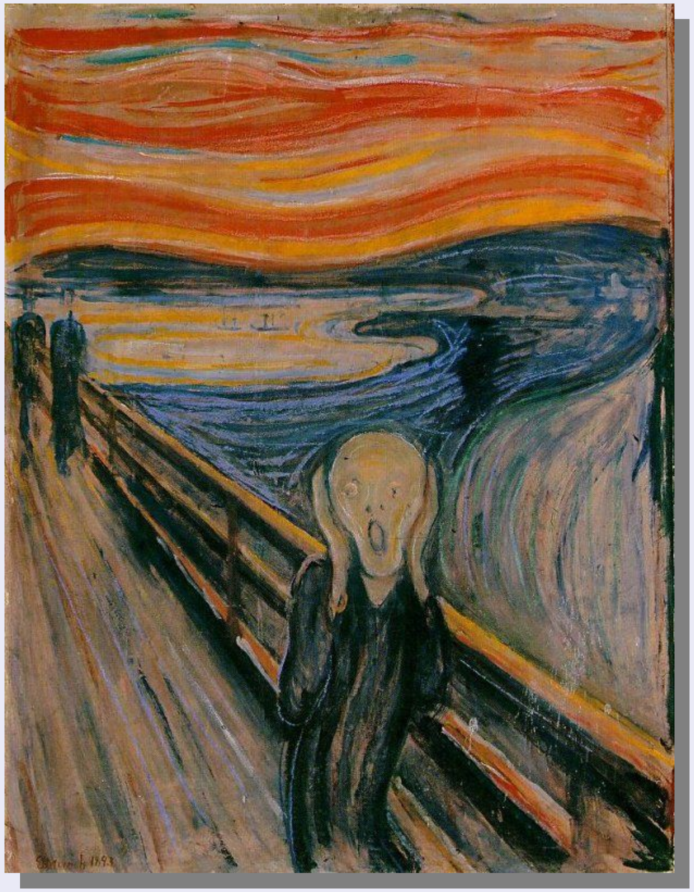
```
http://www.alexa.com/site/site_stats/signup?site_url=http%3A%2F%2Fasdf.com%2F%3F
%22%3E%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E&range=3m&widget=g&
submitted=true&mode=graph&amzn_id=
http://www.altavista.com/web/res_text?q=%22%3E%3Cscript%3Ealert%28%27XSS%27%29
%3C%2Fscript%3E
```

These have been out there for a while but are still unfixed.

Options: Reply · Quote

Origins

OWASP
The Open Web Application Security Project

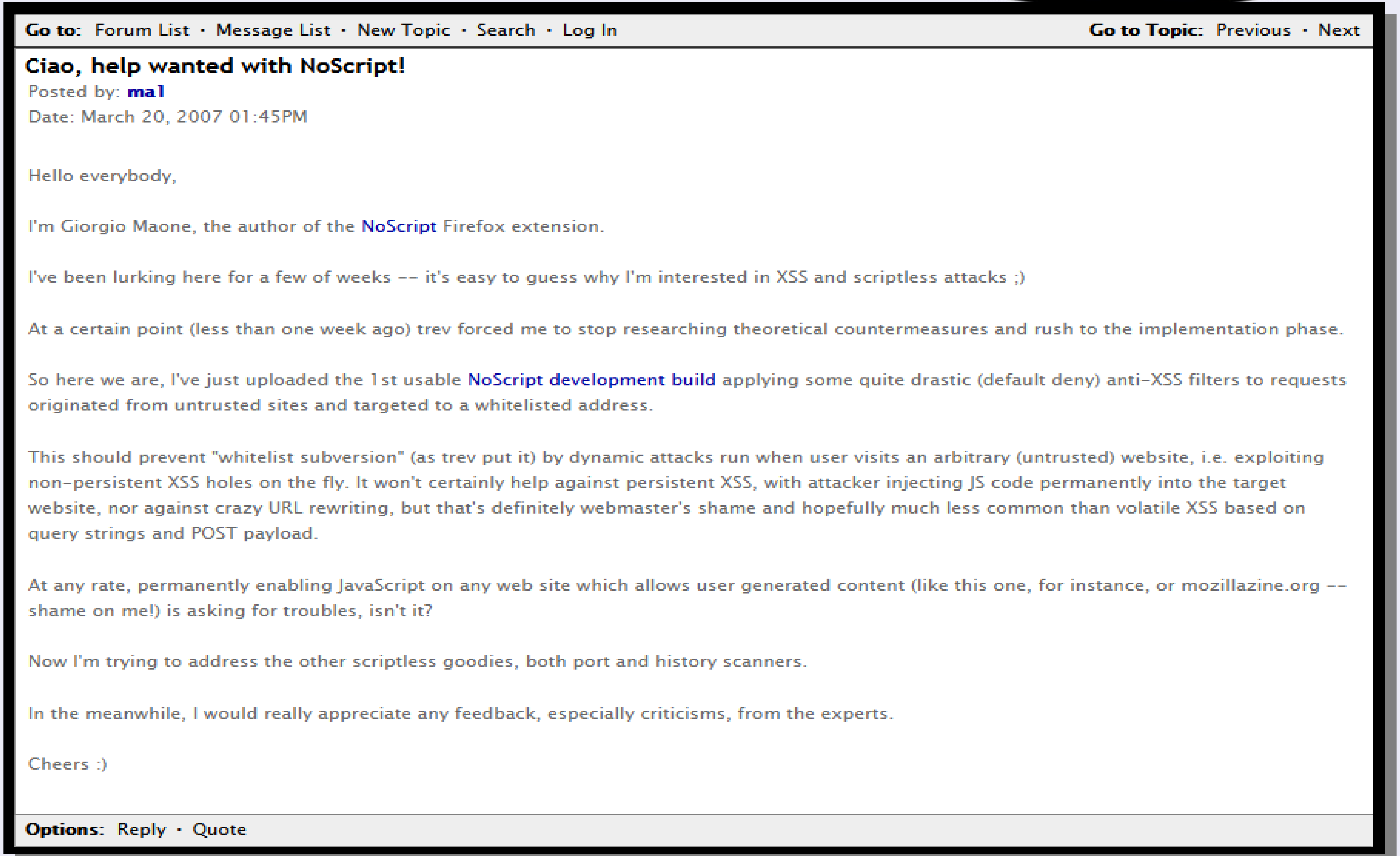


Whitelist + XSS =

No NoScript

!!!

Origins



Go to: Forum List · Message List · New Topic · Search · Log In   Go to Topic: Previous · Next

Ciao, help wanted with NoScript!
Posted by: ma1
Date: March 20, 2007 01:45PM

Hello everybody,

I'm Giorgio Maone, the author of the NoScript Firefox extension.

I've been lurking here for a few of weeks -- it's easy to guess why I'm interested in XSS and scriptless attacks ;)

At a certain point (less than one week ago) trev forced me to stop researching theoretical countermeasures and rush to the implementation phase.

So here we are, I've just uploaded the 1st usable NoScript development build applying some quite drastic (default deny) anti-XSS filters to requests originated from untrusted sites and targeted to a whitelisted address.

This should prevent "whitelist subversion" (as trev put it) by dynamic attacks run when user visits an arbitrary (untrusted) website, i.e. exploiting non-persistent XSS holes on the fly. It won't certainly help against persistent XSS, with attacker injecting JS code permanently into the target website, nor against crazy URL rewriting, but that's definitely webmaster's shame and hopefully much less common than volatile XSS based on query strings and POST payload.

At any rate, permanently enabling JavaScript on any web site which allows user generated content (like this one, for instance, or mozillazine.org -- shame on me!) is asking for troubles, isn't it?

Now I'm trying to address the other scriptless goodies, both port and history scanners.

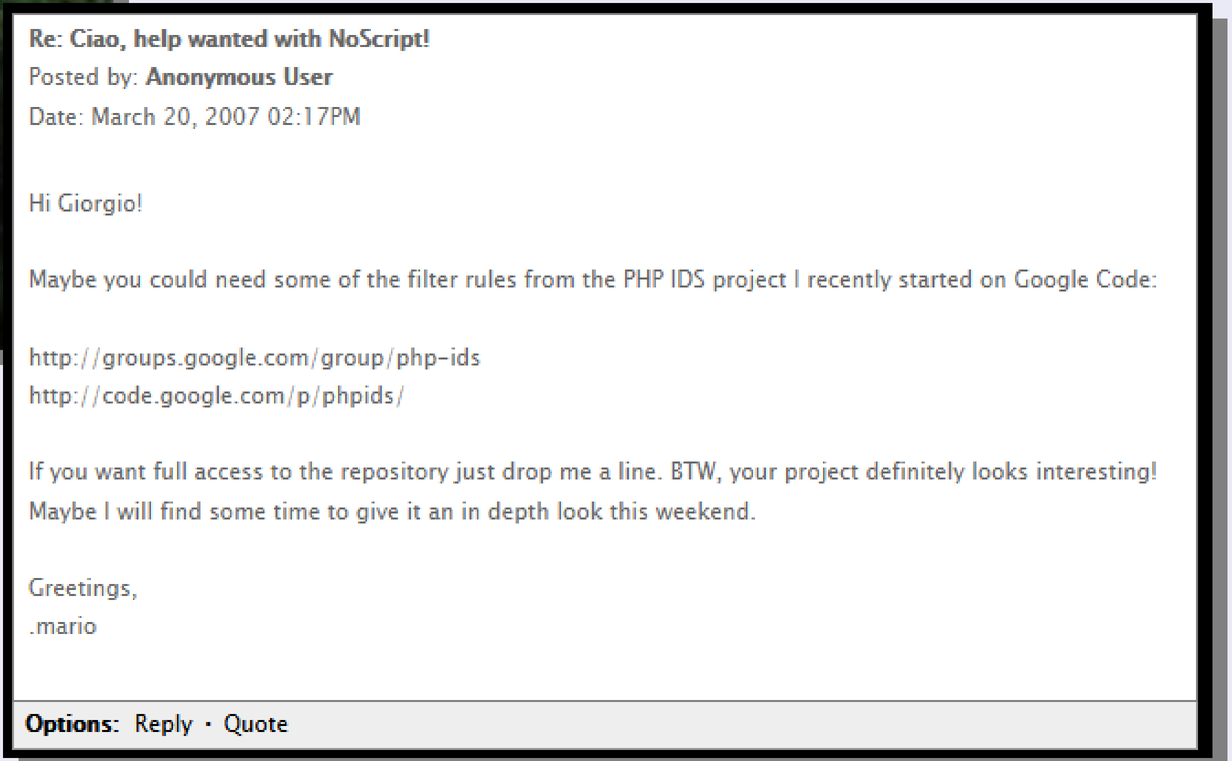
In the meanwhile, I would really appreciate any feedback, especially criticisms, from the experts.

Cheers :)

Options: Reply · Quote

Origins

dr.-ing. mario heiderich


Re: Ciao, help wanted with NoScript!
Posted by: **Anonymous User**
Date: March 20, 2007 02:17PM

Hi Giorgio!

Maybe you could need some of the filter rules from the PHP IDS project I recently started on Google Code:

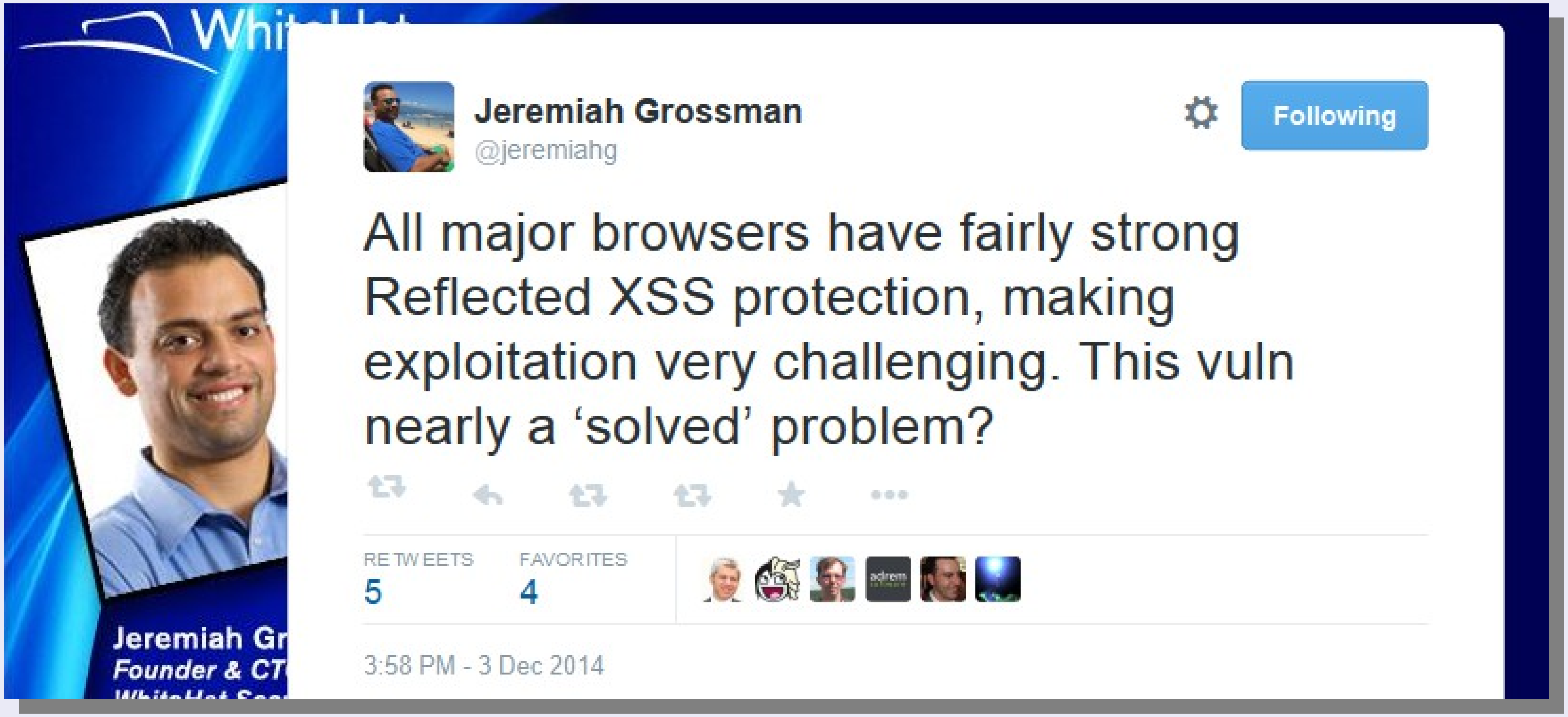
http://groups.google.com/group/php-ids
http://code.google.com/p/phpids/

If you want full access to the repository just drop me a line. BTW, your project definitely looks interesting! Maybe I will find some time to give it an in depth look this weekend.

Greetings,
.mario

OWASP
The Open Web Application Security Project

7 years later...


Jeremiah Grossman
@jeremiahg

Following

All major browsers have fairly strong Reflected XSS protection, making exploitation very challenging. This vuln nearly a 'solved' problem?

RETWEETS 5
FAVORITES 4

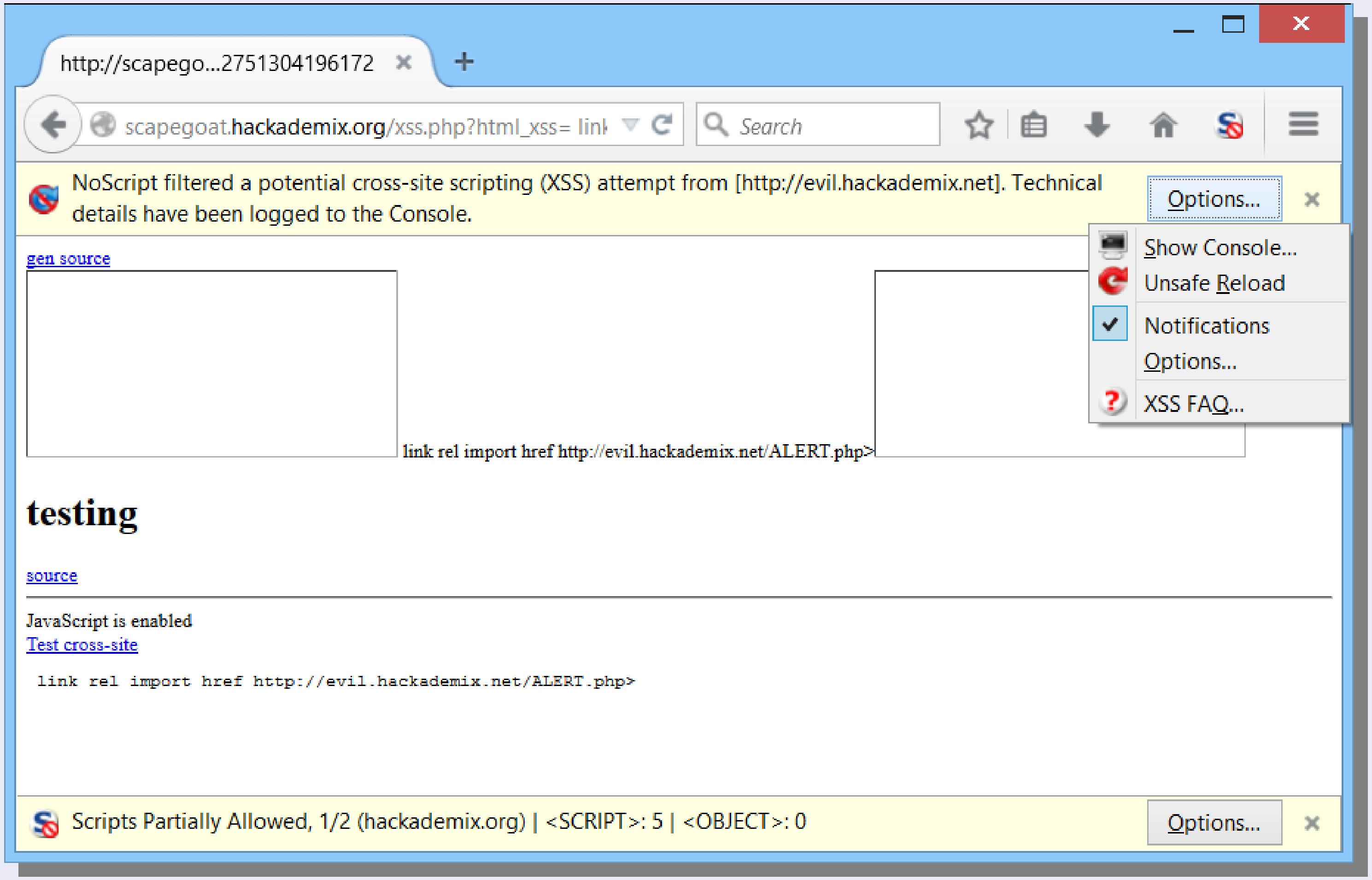
3:58 PM - 3 Dec 2014

Legacy

Yeah, right.

OWASP
The Open Web Application Security Project

Firefox has no native protection yet...



NoScript filtered a potential cross-site scripting (XSS) attempt from [http://evil.hackademix.net]. Technical details have been logged to the Console.

gen source

link rel import href http://evil.hackademix.net/ALERT.php>

testing

source

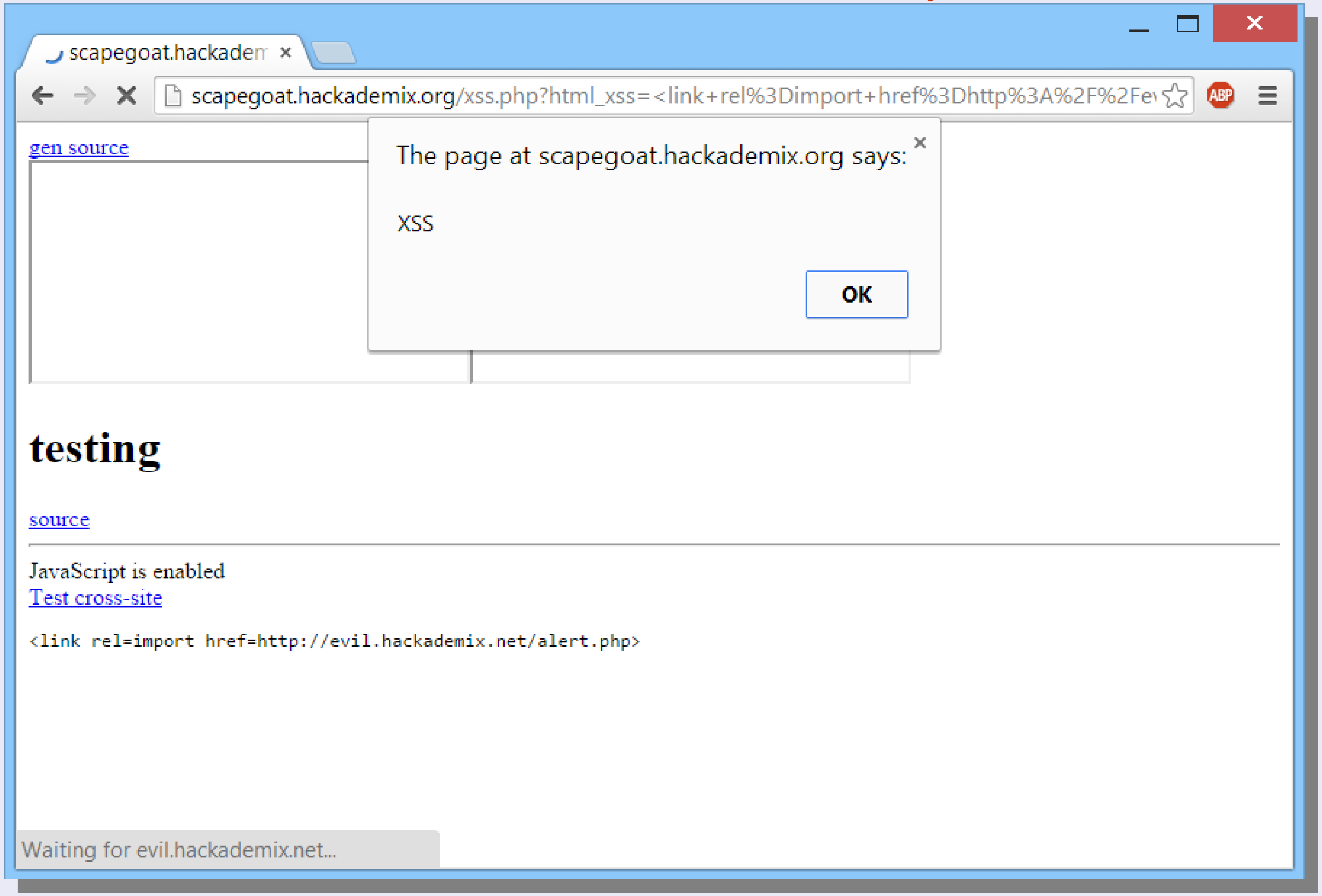
JavaScript is enabled
Test cross-site

```
link rel import href http://evil.hackademix.net/ALERT.php>
```

Scripts Partially Allowed, 1/2 (hackademix.org) | <SCRIPT>: 5 | <OBJECT>: 0

OWASP
The Open Web Application Security Project

.. nor has Chrome ;-)



scapegoat.hackademix.org/xss.php?html_xss=<link+rel%3Dimport+href%3Dhttp%3A%2F%2Fe

The page at scapegoat.hackademix.org says:

XSS

testing

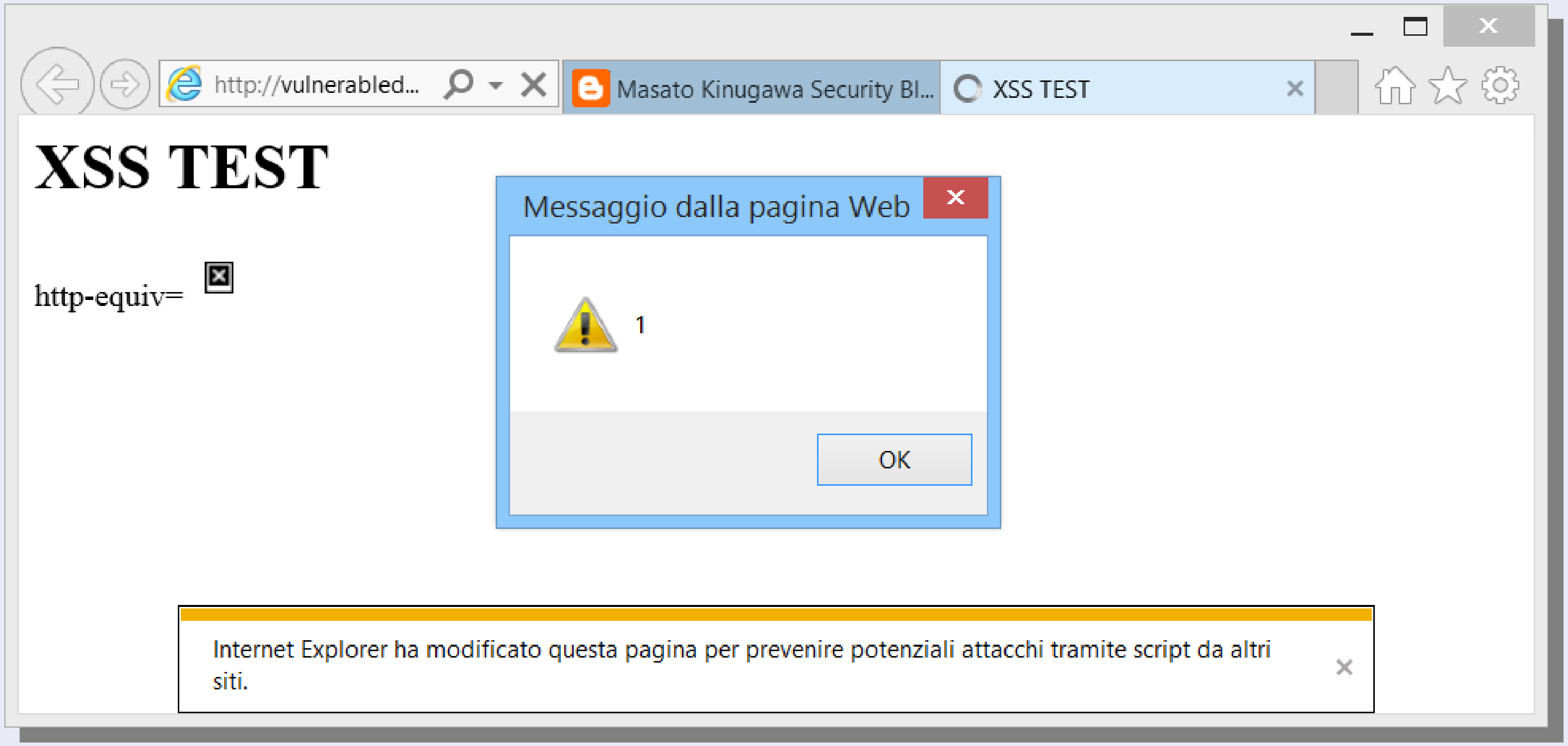
source

JavaScript is enabled
Test cross-site

```
<link rel=import href=http://evil.hackademix.net/alert.php>
```

OWASP
The Open Web Application Security Project

did you say MSIE?



XSS TEST

http-equiv=

Messaggio dalla pagina Web

1

OK

Internet Explorer ha modificato questa pagina per prevenire potenziali attacchi tramite script da altri siti.

Legacy

OWASP
The Open Web Application Security Project

So where we are, really?

- 2007: NoScript demonstrated client-side XSS protection was viable
- 2008: MSIE 8's XSS filter (effective against many attacks but causes vulnerabilities of its own)
- 2010: Chrome's XSS Auditor (weak)
- ????: Firefox's Heuristics to block reflected XSS (like in IE8) (TODO, Bug 528661)

experience counts

OWASP
The Open Web Application Security Project

NoScript XSS Trainers Hall of Fame

```
Object.keys(
  document.querySelector("#changelog")
  .textContent.match(/\n(?:[x+]) .*(\n {2}.*)*/g)
  .map(s => let (m = s.match(
              /(?:XSS|Inj)[\s\S]*\bthanks\s+(?:to\s+)?\s*([\s\S]+)?\b(?:(?:,\s*)see|for|\))/
          )) m && m[1].replace(/\s+/g, ' ').replace(/\s*(?:\bfor\b|\))[\s\S]*|\s+$/g, ''))
  .filter(s => !!s)
  .reduce((o, s) => s.split(/\s*(?:\band|&|,)\s+/).reduce((o, s) => o[s] = o, o), {})
).sort((a,b) => a.localeCompare(b)).join(", ")
```

The Open Web Application Security Project

NoScript XSS Trainers Hall of Fame

.mario, ableeker, Aditya K Sood, Aerik, Ahamed Nafeez, Aicke Schulz, al_9x, Alan Baxter, Alejandro Rusell, Alex Inführ, Ashar Javed, boris, Bueller007, Chris Lonsberry, Colling Jackson, Daethian, Dan Loomis, Daniel Holbert, dave b, Dixie, dondado, dood_97, Edward C. Kim, File Descriptor AKA XSS Jigsaw, Gareth Heyes, Gavin H, gazer75, Gunnar, Gunnar Scherf, Harry, HeikoAdams, hi_RAM, Jamie Cox, Janne Maekelae, jerriy, John Danfort, John Dwyer, JonCage, Jussi Lahtinen, Kostas, Krzysztof Kotowicz, Kuza55, LeeB, Logos, LouiseRBaldwin, Lucas Malor, Luigi, m_c, Markus Wienand, Martin Focke, maryadavies, Masato Kinugawa, MaZe, Mirko Tasler, MysticOrchid, Nick Fnord, niko322, NoRelationToNed, Olaf Schweppe, Pepe Vila, Phil Purviance, Philipp Gühring, PrinceofWeasels, RAJAH235, Roman Vock, RSnake, Salim, sharpie, Silvana, Sirdarckcat, skl, Soroush Dalili, Stefano Di Paola, Stephen F., Stuart Young, Sylvia Oberstein, the JoshMeister, therube, Thomas, Trupti Chaudhari, WHK, yahoo mail user, Zoiz

Credits

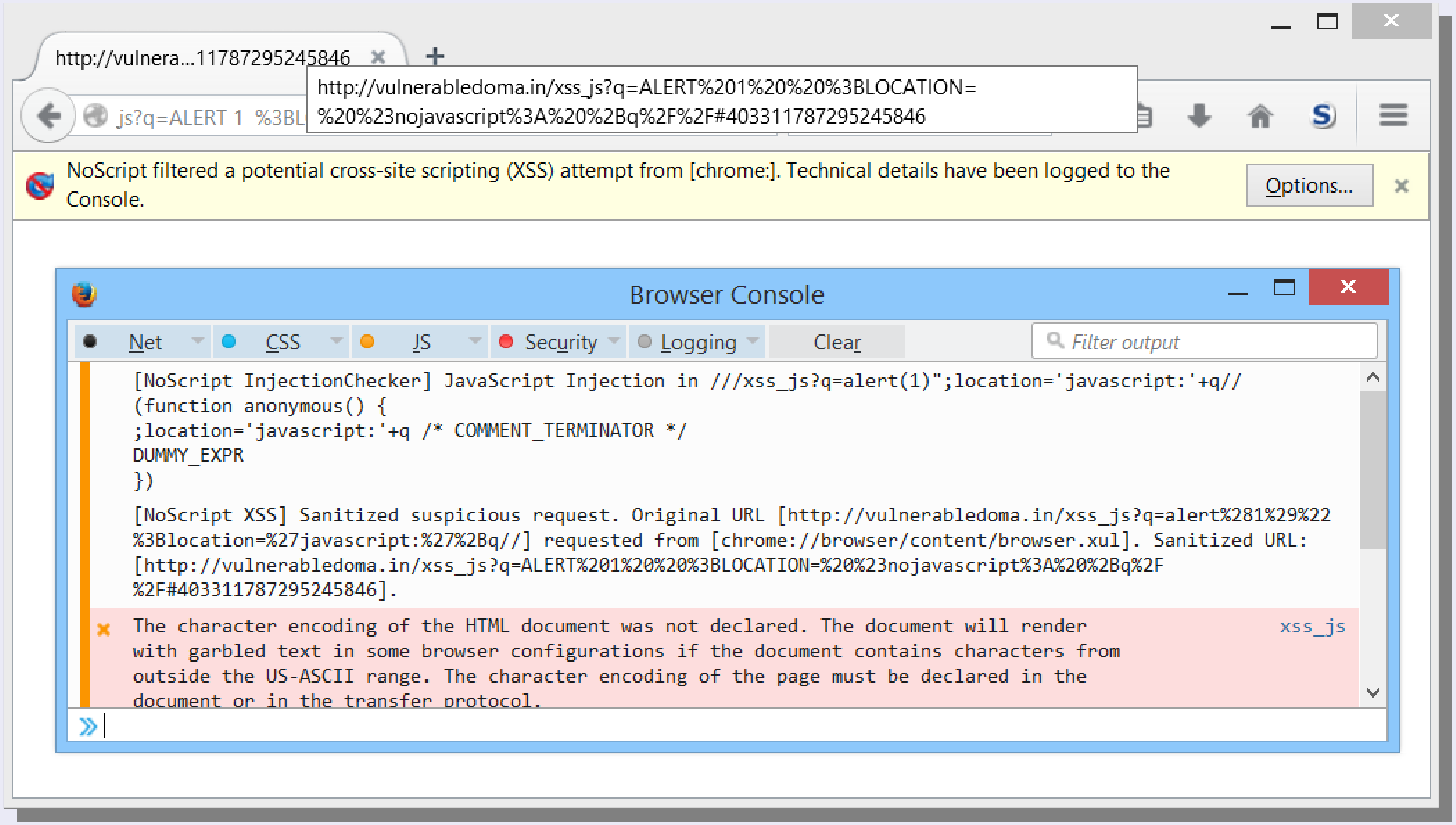
- Most of these researchers use NoScript daily and depend on its security
- Their findings get always full aknowledged
- **A fix is usually released in less than 24 hours**

OWASP
The Open Web Application Security Project

wanna help?

OWASP
The Open Web Application Security Project

CTRL+SHIFT+J



```
http://vulnerabledoma.in/xss_js?q=ALERT%201%20%20%3BLOCATION=
%20%23nojavascript%3A%20%2Bq%2F%2F#403311787295245846
```

NoScript filtered a potential cross-site scripting (XSS) attempt from [chrome:]. Technical details have been logged to the Console.

```
[NoScript InjectionChecker] JavaScript Injection in ///xss_js?q=alert(1)";location='javascript:'+q//
(function anonymous() {
;location='javascript:'+q /* COMMENT_TERMINATOR */
DUMMY_EXPR
})

[NoScript XSS] Sanitized suspicious request. Original URL [http://vulnerabledoma.in/xss_js?q=alert%281%29%22
%3Blocation=%27javascript:%27%2Bq//] requested from [chrome://browser/content/browser.xul]. Sanitized URL:
[http://vulnerabledoma.in/xss_js?q=ALERT%201%20%20%3BLOCATION=%20%23nojavascript%3A%20%2Bq%2F
%2F#403311787295245846].

The character encoding of the HTML document was not declared. The document will render
with garbled text in some browser configurations if the document contains characters from
outside the US-ASCII range. The character encoding of the page must be declared in the
document or in the transfer protocol.
```

```
$ wget https://noscript.net/betas/noscript-2.6.9.6rc3.xpi

$ unzip noscript-2.6.9.6rc3.xpi

$ unzip chrome/noscript.jar

$ vi content/noscript/RequestWatchdog.js
$ find ./ -name "*.js" | xargs cat \
   | sed '/^\s*$/d' | wc -l

22300

$ cat content/noscript/RequestWatchdog.js \
     | sed '/^\s*$/d' | wc -l

2437
```

OWASP
The Open Web Application Security Project

back to the origins

Origins

The Open Web Application Security Project

Where do we come from?

Hard question for Humans and HTTP requests

Origins

OWASP
The Open Web Application Security Project

- Referrer is good, but not dependable
- Privileged does not (always) mean safe
- Sometimes you need to examine the call stack
- You always need to walk back redirections

Exceptions

OWASP
The Open Web Application Security Project

- In an ideal world we shouldn't need them :(
- User can define his own (regexp-based)
- Built-in are fine grained up to skip individual requesst parameters (GET or POST)

Escaping escapings

- (un)escape VS (d)encodeURI(component) VS form encoding
- Base64
- XML and HTML entities
- CSS escapes
- ASCII & Unicode escapes in string literals
- Unicode escapes in JavaScript source

↖ ADDITIVE OMG!!! ↗

Escaping escapings

OWASP
The Open Web Application Security Project

The kinky stuff...

- PHP overdecoding
- ASP HomoXSSuality
- ASP parameter collapse
- Flash escaping
- Ebay escaping

Ignoring the noise

- JSON, even in URL parameters
- XML!
- Common URL subpatterns
- Other expensive distractions

Looking for injections

OWASP
The Open Web Application Security Project

- HTML injections
- Attribute breaking/insertion
- CSS injections
- JavaScript injections

Looking for injections

maybeJS()

Looking for injections

The Open Web Application Security Project

```
Regular expressions +
         DOM Parser +
JavaScript interpreter =
                    WIN!
```

Sanitizer

OWASP
The Open Web Application Security Project

- Blacklist of characters and constructs
- Regexp-based, replaces with spaces
- Triggered on InjectionChecker match
- Affects URLs and referrers, POST payloads get entirely erased
- It works, but needs to go away

False positives

OWASP
The Open Web Application Security Project

Please post data, not code!

- Avoid fancy cross-site POSTs (and GETs!)
- JSON & XML are OK
- JavaScript & HTML are bad
- Base64 != “obfuscation”

OWASP
The Open Web Application Security Project

back to the future

Future plans

- Refactoring (less regexps, more parser)
- Remote (out-of-process) Request Watchdog (ABE + InjectionChecker)
- Request suspension and resuming
- Safe Browsing – like error page
- False positive reporting (like ClearClick)

ThanXSS!

OWASP
The Open Web Application Security Project

- giorgio@maone.net
- hackademix.net
- @ma1
- noscript.net