# Reining in the Web with Content Security Policy

Sid Stamm
Brandon Sterne
Gervase Markham

**Mozilla**

# Mash-ups Anyone?

*But how do I stop malicious content?*

# Content Injection

*DOM attacks and Defacement*

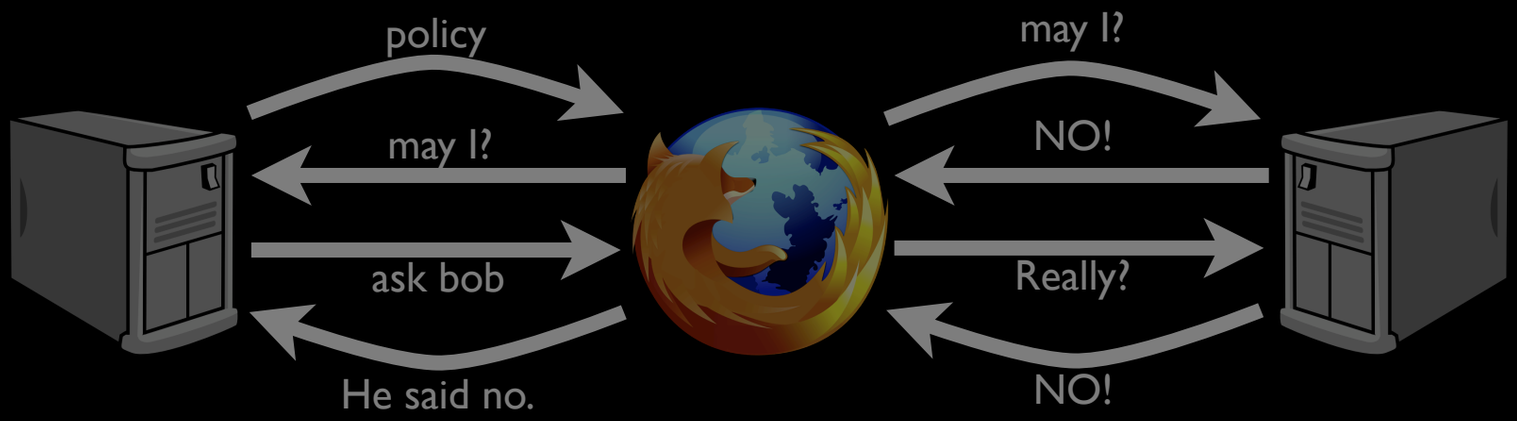
# XSS

*All your page is belong to us!*

# Filtering is Hard!

<DIV STYLE="background-image:\0075\0072\006C\0028'\006a
\006I\0076\006I\0073\0063\0072\0069\0070\0074\003a\006I\006c
\0065\0072\0074\0028.1027\0058.1053\0053\0027\0029'\0029">
<BODY onload!#$%&()*~+-_.,:;?@[/|\]^`=alert("XSS")>
<A HREF="h
tt p://6	6.000146.0x7.147/">XSS</A>

# Mutual Approval can be Expensive!




# In-Band Policies are Dangerous!

*Javascript that polices itself?*
*Is that like an application that tells you*
*if it is a virus?*

# Goals

- Control of Site Content
- Protection against XSS
- Clickjacking Avoidance
- Increased Security
- Feasible Use

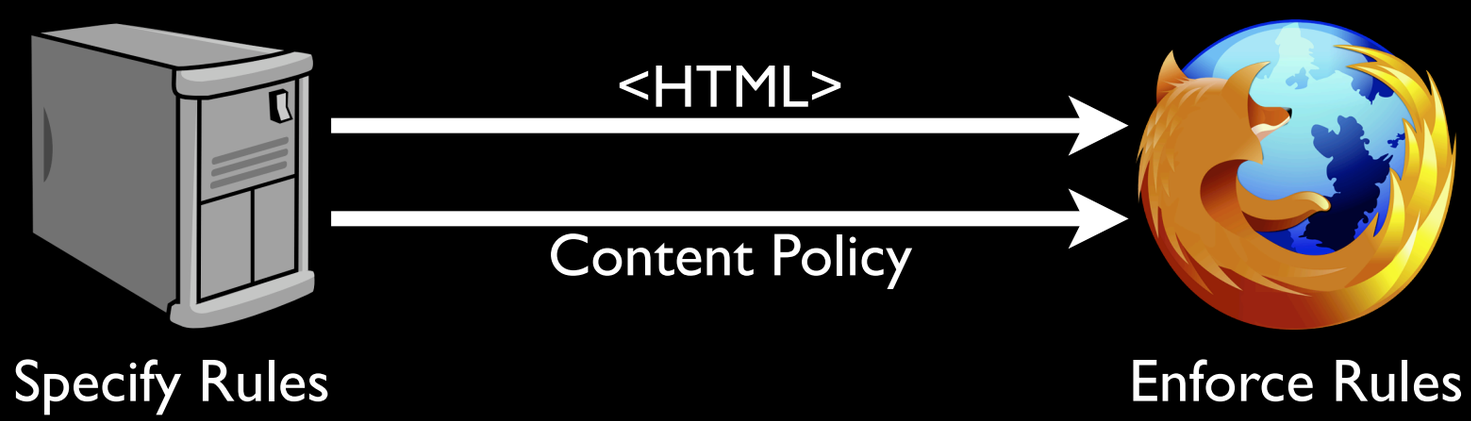
# Control of Site Content

Document “Good” behavior...

Suppress the “Bad”

# Grabbing the Reins

- Content Rules & Regulations
- Specify a "Normal Behavior" Policy
- Catch and Block Violations




# Part 1: Smooth Edges

- Scripts served in files (not inline)
  - "javascript:" URIs
  - <tag on*=...> event registration
  - text nodes in <script> tags
- Establish Code / Data Separation
  - eval("foo") and friends

# Part 2: Content Restrictions

- Block requests for all resources

... unless explicitly allowed by a policy!

# CSP: Policies

## HTTP Response Header
## `X-CONTENT-SECURITY-POLICY`

Directives to enforce listed within

# Speed Bump

`<meta http-equiv=....>?`

- Designers may not have access to HTTP
- Two entities want restrictions
- Multiple policies?

# Speed Bump

## Intersecting Policies

Given Policies P1 and P2:

$P_e = \{u \mid P_1 \text{ allows } u \text{ AND } P_2 \text{ allows } u\}$

# Speed Bump

~~<meta http-equiv=....>~~?

- policy in-band is too dangerous
- Multiple header instances!

# CSP: Directives

`report-uri`

`policy-uri`

`options`

source directives

# CSP: Source Directives

`allow` (default for these)

`img-src`

`media-src`

`script-src`

`object-src`

`frame-src`

`font-src`

`xhr-src`

`frame-ancestors`

`style-src`

# Speed Bump

- ‘self’ ... in pieces?

```
https://‘self’:443
‘self’://foo.com
          foo.com:‘self’
```

# ‘self’

```
‘self‘              -> http://foo.com:80
bar.com:8080        -> http://bar.com:8080
http://foo.com -> http://foo.com:80
bar.com             -> http://bar.com:80
```

# Speed Bump

- Redirects

```
http://foo.com
↪http://bar.com
↪http://duh.com
```

# Goals (revisited)

- Control of Site Content
- Protection against XSS
- Clickjacking Avoidance
- Only Increased Security
- Feasible Use

# Goals (revisited)

- Control of Site Content

Expressive white-list
policy language

# Goals (revisited)

- Protection against XSS

Only load scripts in
external (whitelisted) files

# Goals (revisited)

- Clickjacking Avoidance

`frame-ancestors`

# Goals (revisited)

- Only Increased Security

Declarative syntax that can
only reduce capabilities

# Goals (revisited)

- Feasible Use

(1) Built into Firefox nightlies

(2) Deployed as patch for for Mozilla Add-Ons site

(3) In progress for Wordpress

http://core.trac.wordpress.org/ticket/10237

# Beneficial Effects

- Content homogenization (mixed content control)
- Data exfiltration (and CSRF) reduction
- Violation reports = early alert

# CSP: Use Case 1

`allow 'self'`

- Site wants all content to come from the same source (scheme, host, port)

# CSP: Use Case 2

```
allow ‘self’; frame-src ads.net
```

- Site wants all content to come from the same source (scheme, host, port), except content in iframes may be served by a third-party advertising network.

# CSP: Use Case 3

```
allow ‘self’; img-src *; \
object-src *.teevee.com; \
script-src myscripts.com
```

- Auction site wants to allow images from anywhere, plugin content from a trusted media provider network, and scripts only from its server hosting sanitized JavaScript

# CSP: Use Case 4

```
allow https://*.x.com;
```

- Example site wants to force all content to be served via HTTPS on port 443, from any subdomain of example.com

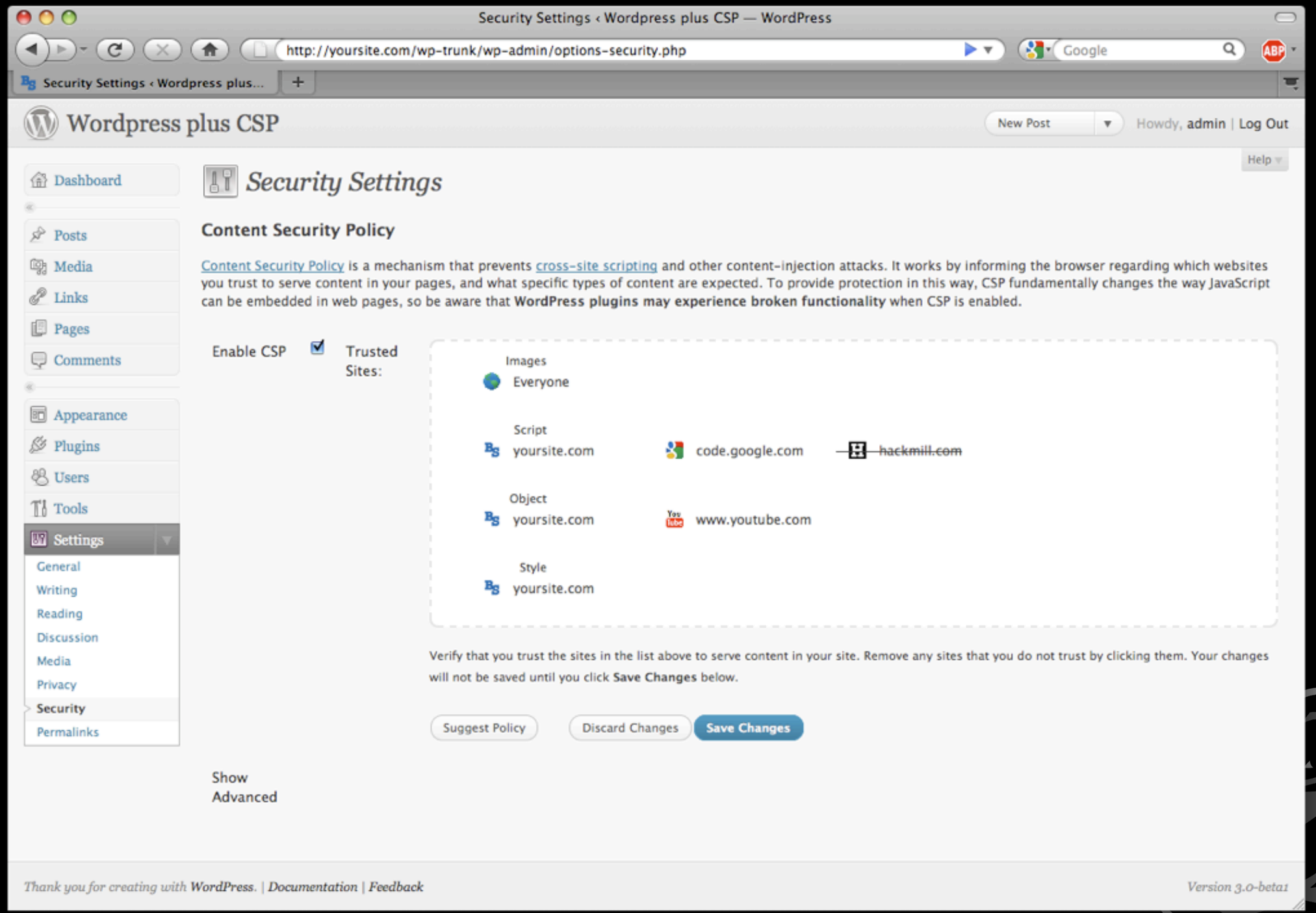
# Wait!
# That breaks my site!

- Good Option: convert your site
- Less Good Option: disable parts of CSP

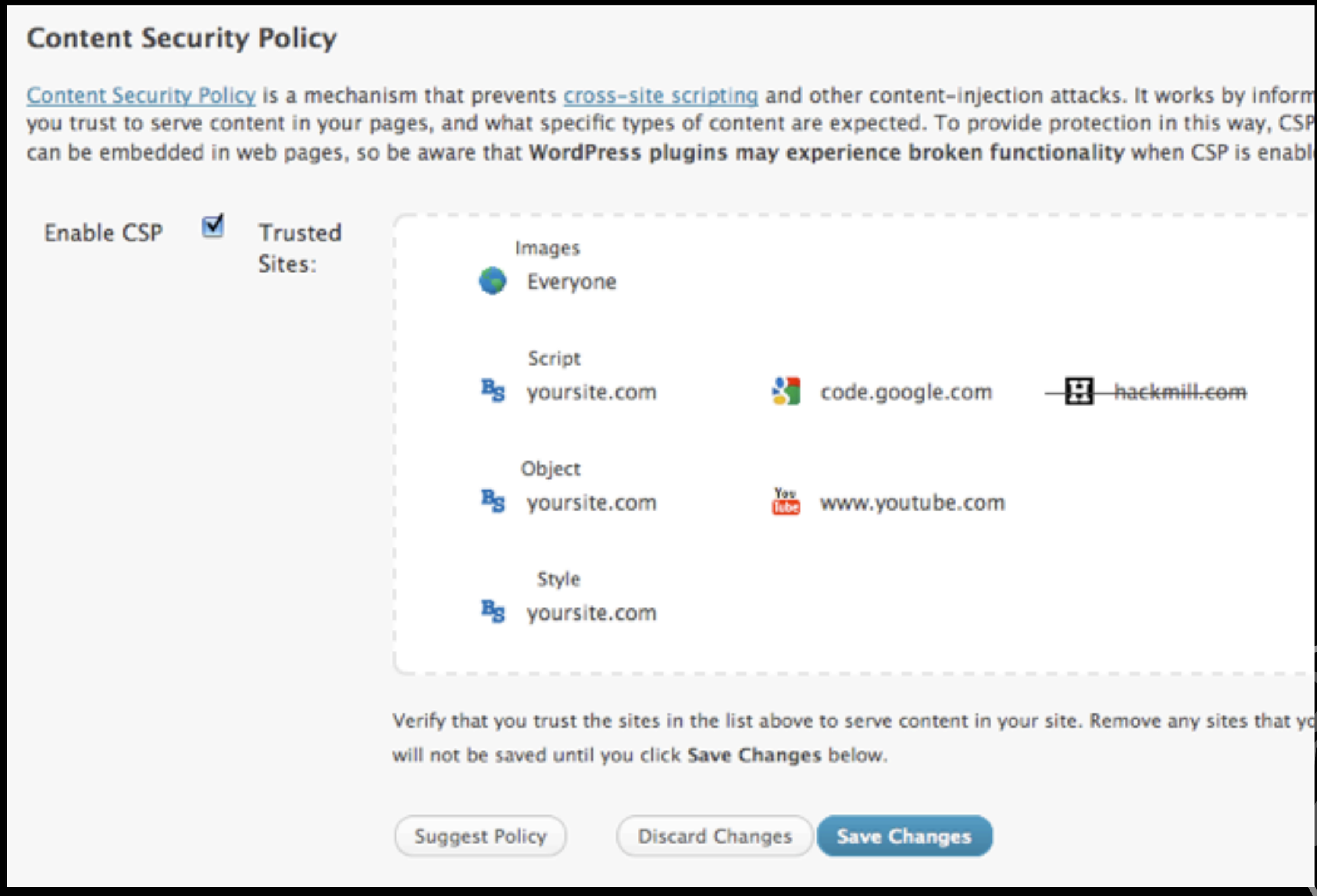
# Ramping Up

- Disable some restrictions via `options`
- Report-Only mode
- "Writing a Policy" guide
- "Converting your Site" guide
- Maybe a policy recommendation tool?

# Wordpress


Security Settings ‹ Wordpress plus CSP — WordPress
http://yoursite.com/wp-trunk/wp-admin/options-security.php
Google
Security Settings ‹ Wordpress plus...
Wordpress plus CSP
New Post
Howdy, admin | Log Out
Help
Dashboard
Posts
Media
Links
Pages
Comments
Appearance
Plugins
Users
Tools
Settings
General
Writing
Reading
Discussion
Media
Privacy
Security
Permalinks
Security Settings
Content Security Policy
Content Security Policy is a mechanism that prevents cross-site scripting and other content-injection attacks. It works by informing the browser regarding which websites you trust to serve content in your pages, and what specific types of content are expected. To provide protection in this way, CSP fundamentally changes the way JavaScript can be embedded in web pages, so be aware that WordPress plugins may experience broken functionality when CSP is enabled.
Enable CSP
Trusted Sites:
Images
Everyone
Script
yoursite.com
code.google.com
hackmill.com
Object
yoursite.com
www.youtube.com
Style
yoursite.com
Verify that you trust the sites in the list above to serve content in your site. Remove any sites that you do not trust by clicking them. Your changes will not be saved until you click Save Changes below.
Suggest Policy
Discard Changes
Save Changes
Show Advanced
Thank you for creating with WordPress. | Documentation | Feedback
Version 3.0-beta1

# Wordpress


Content Security Policy
Content Security Policy is a mechanism that prevents cross-site scripting and other content-injection attacks. It works by inform
you trust to serve content in your pages, and what specific types of content are expected. To provide protection in this way, CSP
can be embedded in web pages, so be aware that WordPress plugins may experience broken functionality when CSP is enabl
Enable CSP
Trusted Sites:
Images
Everyone
Script
yoursite.com
code.google.com
hackmill.com
Object
yoursite.com
www.youtube.com
Style
yoursite.com
Verify that you trust the sites in the list above to serve content in your site. Remove any sites that yo
will not be saved until you click Save Changes below.
Suggest Policy
Discard Changes
Save Changes

# More Stuff

- Specification: https://wiki.mozilla.org/Security/CSP/Specification
- Nightly Firefox http://nightly.mozilla.org
- Progress: https://bugzilla.mozilla.org/show_bug.cgi?id=csp