CSP - the panacea for XSS or placebo?
$ whoami

Information security officer at Yandex's product security team
Web application security researcher
Yet another security blogger [www.oxdef.info](http://www.oxdef.info)
XSS
XSS

Why again about XSS?!  
Still one of the **the most common** web application security issues  
Ok, but please don't show me those alerts
Prevention

Input validation
Output escaping depending on context
httponly session cookie
Browser based solutions: IE filter, NoScript
CSP
Content Security Policy

Browser side mechanism to mitigate XSS attacks
Source whitelists for client side resources of web application
Content-Security-Policy HTTP header
W3C Candidate Recommendation
How it Works

HTML Template

<h1>Test XSS page</h1>
<h3>Hello, <i>{{ foo | safe }}!</i></h3>

Demo URL

http://127.0.0.1:5000/xss?foo=
<img src="http://www.oxdef.info/exploit.png">
Test XSS page

Without CSP
CSP in Action

Content-Security-Policy: img-src 'self'

Test XSS page

Hello,!

Refused to load the image 'http://www.originfo/exploit.png' because it violates the following Content Security Policy directive: "img-src 'self'."
Control JavaScript

Policy

Content-Security-Policy: default-src 'self'; script-src 'self' static.example.com

HTML

<!doctype html><html><head>
<meta charset="utf-8">
<script src="/js/jquery-1.10.2.js"></script>
<script src="//evil.net/evil.js"></script>...

console.log

Refused to load the script 'http://evil.net/evil.js' because it violates...
Unsafe-inline and unsafe-eval

• unsafe-inline allows:
  – Inline scripts and styles
  – onclick="..."
  – javascript:
  – You should not include it in the policy!

• unsafe-eval allows:
  – eval()
  – new Function
  – setTimeout, setInterval with string as a first argument
  – You should not include it in the policy!
Other Directives

media-src — audio and video
object-src - plugin objects (e.g. Flash)
frame-src — iframe sources
font-src — font files
connect-src — XMLHttpRequest, WebSockets, EventSource
Reporting

Policy


Log contents

```json
{
    "csp-report": {
        "violated-directive": "img-src data: ... *.example.com",
        "referrer": "",
        "blocked-uri": "https://static.doubleclick.net",
        "document-uri": "https://example.com/foo",
        "original-policy": "default-src ...; report-uri csp.php"
    }
}
```
Browser Support

Content-Security-Policy 🌐 25+ ⬆️ 23+ 
🐍 1.7+

X-Content-Security-Policy 🌐 4 - 22
🔥 10 (sandbox)

X-WebKit-CSP 🌐 14 - 25 ⬆️ 5.1+

Mobile browsers: 🌐 7.0+ 🌐 28+ ⬆️ 23+
Bypass

Manipulating HTTP response headers
Implementation bugs: MFSA 2012-36: Content Security Policy inline-script bypass
JSONP
XSS without JS
See in the Next Version: nonce-source

Policy

Content-Security-Policy: script-src 'self' nonce-Nc3n83cnSAd

HTML Code

```html
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <script src="/js/jquery.min.js"></script>
</head>
<body>
  <script nonce="Nc3n83cnSAd">
    // Some inline code here
  </script>
</body>
```
Case-study
About the Service

One of the most popular mail services in Russia
Over 12 million email messages daily
Lots of client side code and hosts to communicate with
CSP Tester

Extension for Chromium based browsers
Simple and Advanced modes
Content-Security-Policy and X-WebKit-CSP headers
Help links for directives
https://github.com/oxdef/csp-tester
### CSP Tester

<table>
<thead>
<tr>
<th><strong>URL Pattern</strong></th>
<th><em>://mail.yandex.ru/</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>default-src</strong></td>
<td>wss://xiva-daria.mail.yandex.net:* *yandex.ru *yandex.net</td>
</tr>
<tr>
<td><strong>script-src</strong></td>
<td>'unsafe-inline' 'unsafe-eval' blob: chrome-extension: *yandex:</td>
</tr>
<tr>
<td><strong>object-src</strong></td>
<td>*yandex.ru *yandex.net yandex.st</td>
</tr>
<tr>
<td><strong>style-src</strong></td>
<td>'unsafe-inline' *yandex.net yandex.st</td>
</tr>
<tr>
<td><strong>img-src</strong></td>
<td>data: *yandex.ru *yandex.net yandex.st</td>
</tr>
<tr>
<td><strong>media-src</strong></td>
<td>*yandex.net yandex.st</td>
</tr>
<tr>
<td><strong>frame-src</strong></td>
<td></td>
</tr>
<tr>
<td><strong>font-src</strong></td>
<td></td>
</tr>
<tr>
<td><strong>connect-src</strong></td>
<td></td>
</tr>
<tr>
<td><strong>sandbox</strong></td>
<td></td>
</tr>
<tr>
<td><strong>report-uri</strong></td>
<td>csp.jsx</td>
</tr>
</tbody>
</table>

- Active: checked
- Report Only: not checked

**Advanced Mode**
The Plan

1. Test it on the corporate mail
2. It's ok - let's try it on production in Report-Only mode
3. Analyze tons of logs ;-((
4. Fix bugs and improve the policy
5. Switch to block mode
6. Profit! :-(
Changes in service

Try to remove all inline code
Log Analysis

awk, grep, sort, head for **gigabytes** of logs?
Yes, but we can do it in more complex way with help of Python
Charts for directives and blocked URIs
Problems

Browser implementations differ
3rd party JS libraries
Inline styles in HTML letters
Browser extensions
What is that *** external code doing in our DOM?
From Report-Only to Block mode

Fix bugs from CSP logs
Use only standard CSP HTTP header
Allow browser extensions
unsafe-inline for style-src
unsafe-eval for script-src
Tips

Teach your front-end developers
Add CSP as security requirement for new products
Don't forget about mobile versions!
Research your core front-end components to support CSP
Assign developer responsible for CSP
CSP Based IDS

Test & Fix

Magic

XSS

XSS

XSS
CSP is not a panacea
but it's a good «yet another level» to
protect your users against XSS attacks
To be continued ;-)
Taras Ivashchenko

Information Security Officer

oxdef@yandex-team.ru

http://company.yandex.com/security

Thanks