


# Content Security Policy

## *jako ochrona przed skutkami ataków XSS*

owasp@niebezpiecznik.pl

# niebezpiecznik.pl

- **testujemy** serwisy internetowe, ludzi i sieci komputerowe pod kątem odporności na ataki (nie tylko komputerowe...)
- **doradzamy** i konsultujemy projekty IT pod kątem bezpieczeństwa
- **szkolimy** programistów i administratorów

# XSS

- Mamy i z nim nie wygramy
  - jakie są skutki?
  - jakie są payloady?

+ADw-script+AD4-alert(/xss/)+ADw-/script
+AD4-

<svg:g onload="alert(8)"/>

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]
+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:
++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],
$$_:++$,$$$:++$,$____:++$,$__$:++$};$.$_=($.$_=$
+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__
$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")
[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$
+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.
$=($.____)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]
+"")[$._$_]+$.$$$_+"\\"+$.__$+$.$$_+$._$_+$.__
+"("+$.__$+")"+"\"")())();
```

GIF89ad d ! Y, d d s ????? ?? ? ??
??????? 扦 ʁ? ??? L Ģ L* J
H j N (8HXhx iX ? ? ????? ?? ??????
????
GIF89ad.d.....!.Y
<PUBLIC:COMPONENT>
<PUBLIC:ATTACH EVENT="onclick"
ONEVENT="alert(1)" />
</PUBLIC:COMPONENT>
.,....d.d...s.....H.....L...
.....L*.....J.....j.....N.....
.....(8HXhx.....iX..;

# CSP

- Pomysł Mozilla Foundation
- Pierwsza implementacja w Fx 4.0
- Generalnie whitelista ładowanych zasobów

# Nagłówek HTTP

- **`Content-Security-Policy`**
  Oficjalny (Fx, Chrome, Safari nie wspiera):

- **`X-Content-Security-Policy`**
  Fx 4+ i preview IE 10:

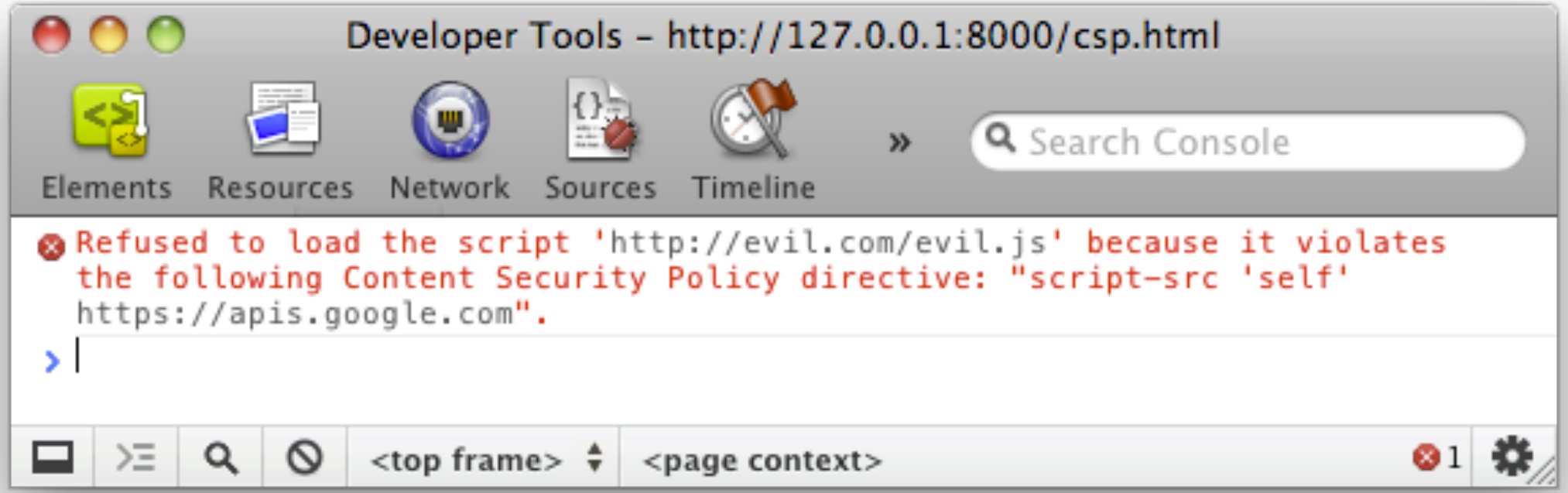
- **`X-WebKit-CSP`**
  Webkit (Safari 6+ Chrome 16+)

# składnia

`Prefix: zasób` [ *scheme* | *hostname* | *port* ];

```
Content-Security-Policy:
script-src https://foo.pl:42;
img-src *.foo.pl;
```

zadziała: ***://*.example.com:***
....ale nie dopasuje example.com :)


Developer Tools - http://127.0.0.1:8000/csp.html
Elements
Resources
Network
Sources
Timeline
Search Console
Refused to load the script 'http://evil.com/evil.js' because it violates the following Content Security Policy directive: "script-src 'self' https://apis.google.com".
<top frame>
<page context>
1

# nie ma inline

- ani skryptów `<script>trololo()</script>`
- ani eventów `onerror, onclick`
- ani urli `javascript:`
- ani "evali" `eval(), new Function(), setTimeout()`
- ani styli inline'owych

- Poprawna implementacja przez:
  - zamianę na **addEventListener(...)**
  - parsowanie **JSON.parse**

```
<script>
  function foo() {
    alert('bar');
  }
</script>
<button onclick='foo();'>trololo</button>
```

```
//index.html
<script src='foobar.js'></script>
<button id='foo'>trololo</button>

//foobar.js
function foo() {
  alert('bar');
}
document.addEventListener('DOMContentReady', function () {
  document.getElementById('foo')
          .addEventListener('click', foo);
});
```

# kontrola zasobów

**`connect-src`**
połączenia via XHR, WebSockets, and EventSource

- **`font-src`**
  skąd serwować web fonts. np. `https://themes.googleusercontent.com`
- **`frame-src`**
  co może być ramkowane? Uwaga, to nie zamiennik dla X-Frame-Options!
- **`media-src`**
  video i audio.
- **`object-src`**
  Flash i inne wtyczki
- **`style-src`**
  nie domyślicie się...
- **`img-src`**
  trololo

# data:

- dla obrazków “inline”

# ‘self’

- Obecny origin strony
- ‘ jest ważny, inaczej hostname **self**

- przeciwieństwo **‘none’** i podzbiór *****

# default-src

- jeśli nie zdefiniowano wartości dla np. **object-src**, to wszystkie obiekty dziedziczą wartość z **default-src**
    - dawniej np. *allow 'self'*
- jeśli obiekt ma ustawioną wartość, to **nie jest** ona łączona z default-src!
- domyślna wartość to *

# report-uri

- przeglądarka POST-uje JSON-a z naruszeniami polityki
- pomaga wykryć fuckup programisty
  - fuckup przepuszczający XSS
  - fuckup w implementacji CSP

```
{
  "csp-report": {
    "document-uri": "http://example.org/page.html",
    "referrer": "http://evil.example.com/",
    "blocked-uri": "http://evil.example.com/evil.js",
    "violated-directive": "script-src 'self' https://
apis.google.com",
    "original-policy": "script-src 'self' https://
apis.google.com; report-uri http://example.org/
my_amazing_csp_report_parser"
  }
}
```

# zapomnij o tym slajdzie

- `'unsafe-inline'`
- `'unsafe-eval'`

# ficzersy

- `sandbox`
  - `allow-forms`
  - `allow-same-origin`
  - `allow-scripts`
  - `allow-top-navigation`

# przykład

- Google +1 button:
  `script-src https://apis.google.com; frame-src https://plusone.google.com`

- Restrykcyjna
  `Content-Security-Policy: default-src 'none'; script-src https://cdn.foo.pl; style-src https://cdn.foo.pl; img-src https://cdn.foo.pl; connect-src https://api.foo.pl; frame-src 'self'`

# dodatki do CSM-ów

- są :)

# można tylko testować

- **Content-Security-Policy-Report-Only:**

# ale lepiej używać

- to nie boli
- nikomu nie przeszkadza
  (wyłączając programistów :>)
- jest wstecznie kompatybilne

# case Twittera (2011)

- jQuery 1.4 testuje eval()
- rozszerzenia wstrzykują inline JS
- ISP wstrzykuje JS i zmienia URL obrazków na cache serwery

- rozwiązanie: SSL :>

# case Niebezpiecznika

- 1 kwietnia można było dodać post do Niebezpiecznika
- "fałszywy" panel admina generowany w JS zapisywał dane client-side (ciasteczko)
- JS odczytywał ciastko na wejściu i pokazywał "złośliwego" posta
- Wielu próbowało wstrzykiwać JS - a my im tego nie utrudnialiśmy (brak encodingu)
- Kilka osób poznało magię CSP ;)
- Cały czas można wrzucić swój kod na dozwolone serwery (u nas np. Google, AdTaily lub Helion)

# Linki

- https://bugzilla.mozilla.org/show_bug.cgi?id=663566
- https://bugs.webkit.org/show_bug.cgi?id=53572
- http://developer.chrome.com/extensions/contentSecurityPolicy.html
- https://developer.mozilla.org/en-US/docs/Security/CSP/CSP_policy_directives
- https://developer.mozilla.org/en-US/docs/Security/CSP/Using_Content_Security_Policy
- http://people.mozilla.org/~bsterne/content-security-policy/
- http://www.w3.org/TR/CSP/
- https://dvcs.w3.org/hg/content-security-policy/raw-file/tip/csp-specification.dev.html#directives

niebezpiecznik.pl

owasp@niebezpiecznik.pl
+48 12 44 202 44
+1 42 42 **HACKME**

facebook.com/niebezpiecznik
google.com/+niebezpiecznik