Large Scale Analysis of CORS Misconfigurations

Jens Müller

http://www.nds.rub.de/
Motivation

• HTTP security headers
  – X-Frame-Options
  – X-Content-Type-Options
  – X-XSS-Protection
  – Referrer-Policy
  – CSP, HSTS, HPKP
  – …
IF THERE WAS A HTTP HEADER TO REMOVE ALL SECURITY

Yup.

WOULD TOP SITES USE IT?
Overview

1. Background
2. Misconfigurations
3. CORStest
4. Evaluation
5. Conclusions
What is CORS?

• Cross-Origin Resource Sharing
• Enables web servers to explicitly allow cross-site access to a certain resource
• Punches holes into Same-Origin Policy
Example

• **Same Origin Policy:** Scripts can only access data from the same origin (protocol, domain, port)
Cross-Origin Resource Sharing

• CORS-based web API access

Access-Control-Allow-Origin: http://good.com

Origin: http://good.com

User

good.com

site.com/api

JavaScript
CORS HTTP headers

• **Access-Control-Allow-Origin** (ACAO)
  - Which URI is allowed access?
• **Access-Control-Allow-Credentials** (ACAC)
  - Access with (session) cookies?
• Some more **Access-Control-**... headers
Overview

1. Background
2. Misconfigurations
3. CORStest
4. Evaluation
5. Conclusions
Misconfigurations

RELAXING THE SAME-ORIGIN POLICY

Credits go to James Kettle

WHAT COULD POSSIBLY GO WRONG?
Developer backdoor

- Insecure developer/debug origins allowed

A CSAO: https://fiddle.jshell.net

Origin: https://fiddle.jshell.net

User

fiddle.jshell.net

site.com/user-data
Allowing access to multiple sites

• Allow all origins
  – ACAO: *
  – but never with credentials (therefore mostly harmless)

• Invalid configurations:
  – ACAO: site1, site2
  – ACAO: *.site

• Solution:
  Dynamically return ACAO based on Origin
Subdomains allowed

- `sub.domain.com` allowed access
  - exploitable if XSS in *any* subdomain
Post/pre domain wildcard

• **notdomain.com** is allowed access
  – can simply be registered by the attacker

• **domain.com.evil.com** is allowed access
  – can be simply be set up by the attacker
Origin reflection

• The origin is simply echoed in **ACAO** header
  – any site is allowed to access the resource
Null misconfiguration

- **ACAO**: `null` to allow local HTML files
- `null` origin can be forced using an iframe
  - any site is allowed to access the resource
- `null` may be returned by software (Node.js)
Protocol-relative URLs

- **ACAO**: // returned by some websites
- How should browsers deal with this?
  - IE, Edge: deny all origins
  - FF, Ch, Sa, Op: allow all
Non-ssl sites allowed

- A `http` origin is allowed access to a `https` resource, allows **MitM** to break encryption

```
Access-Control-Allow-Origin: http://site.com
```

```
Origin: http://site.com
```

```
User
```

```
https://site.com/user-data
```

```
http://site.com
```

```
```
Overview

1. Background
2. Misconfigurations
3. CORStest
4. Evaluation
5. Conclusions
CORStest

• Simple CORS misconfiguration scanner
• https://github.com/RUB-NDS/CORStest
• Sends requests with various Origins checks for the ACAO/ACAC responses
Demo time
Overview

1. Background
2. Misconfigurations
3. CORStest
4. Evaluation
5. Conclusions
Evaluation: Alexa top 1m websites

CORS configuration
Alexa top 1 Million websites

- Developer backdoor 8
- Origin reflection 1702
- Null misconfiguration 57
- Pre-domain wildcard 159
- Post-domain wildcard 147
- Subdomains allowed 460
- Non-ssl sites allowed 1677
- Invalid CORS header 1019
- Valid CORS header 24285

= 3,750 sites
Evaluation: Alexa top 1m with credentials

Access-Control-Allow-Credentials
Alexa top 1 Million websites

- Developer backdoor: 2
- Origin reflection: 1330
- Null misconfiguration: 20
- Pre-domain wildcard: 76
- Post-domain wildcard: 93
- Subdomains allowed: 194
- Non-ssl sites allowed: 385
- Invalid CORS header: 113
- Valid CORS header: 1179

= 1,912 sites
Popular vulnerable sites

cheapstudent.de
provinzial.de
login.worldpay.com
esa.io
moneyversed.com
coinplug.com
orpay.com
wallet.baidu.com
bitcoinpay.com
coinalarm.com
ctf365.com
obamacare.com
conduits.com
transform.microsoft.com
fantrax.com
dzpay.com
payoffshore.com
crystalgraphics.com
planted.com
native-instruments.de
ft.com
abendzeitung-muenchen.de
loanframe.com
netbank.de
leonesternationalbank.com
stedty.com
staffhub.com
fedex.com
bti.click
bti.click
payscale.com
zust.de
worldpay.com
loanframe.com
paytop.com
uberall.com
porsche.com
alepay.vn
alipay.com
tu-dresden.de
dasoertliche.de
chalk.com
transferwise.com
duracell.com
metabo.com
dyol.com
alpaloan.co
playtestcloud.com
uberall.com
abendzeitung-muenchen.de
walmart.com
profile.accounts.firefox.com
helpling.de
udacity.com
9cloud.us
kaspay.com
coinplug.com
orpay.com
wallet.baidu.com
btclick
bti.click
payscale.com
zust.de
worldpay.com
loanframe.com
paytop.com
uberall.com
porsche.com
alepay.vn
alipay.com
tu-dresden.de
dasoertliche.de
chalk.com
transferwise.com
duracell.com
metabo.com
dyol.com
alpaloan.co
playtestcloud.com
uberall.com
abendzeitung-muenchen.de
walmart.com
profile.accounts.firefox.com
helpling.de
udacity.com
9cloud.us
kaspay.com
Reporting on a medium scale

• Had to notify 1,912 1,500 websites
• How to do this? Contact manually?
  – security@, support@, info@, privacy@
• About 300 websites fixed the flaw...
• Some did not want to believe:
  – Kevin has resolved your ticket: “We are fully 
    PCI-DSS compliant and have passed all scans”
  – “We use the most secured cloud servers and 
    military grade encryption to backup your data”
Causes for CORS misconfigurations

Example Nginx configuration for adding cross-origin resource sharing (CORS) support to reverse proxied APIs

```conf
# CORS header support
#
# One way to use this is by placing it into a file called "cors_support"
# under your Nginx configuration directory and placing the following

'^https?://(localhost|www\.yourdomain\.com)' -> localhost.evil.com access granted

set $cors '';
if ($http_origin ~ '^https?://(localhost|www\.yourdomain\.com|www\.yourotherdomain\.com)') {
    set $cors 'true';
```

$ missing
Causes for CORS misconfigurations

- **CORS in Action** contains examples such as
  ```javascript
  var originWhitelist = ['null', ...]
  ```
- **Rack::Cors** maps origins `'` or origins `'*'` into reflecting all origins (+ [CVE-2017-11173](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-11173))
- **crVCL** PHP Framework just checks if allowed origin string is contained in `Origin` value
Invalid headers

• Invalid (creative) ACAO values we observed:
  - `self, true, false, undefined, None, 0, (null), domain, origin, SAMEORIGIN`
Overview

1. Background
2. Misconfigurations
3. CORStest
4. Evaluation
5. Conclusions
Conclusions

• There is a lot of confusion on CORS
• It’s too easy to misconfigure CORS
• Can remove all your web security
• ACAO: * is mostly harmless
Thanks for your attention...

CORStest

- https://github.com/RUB-NDS/CORStest

Questions?
Some popular sites

• Online banking, insurance, bitcoins, payment and US state's tax filing sites vulnerable:
Non-ssl sites allowed

- A http origin is allowed access to a https resource, allows MitM to break encryption

```
Access-Control-Allow-Origin: http://site.com
```

http://site.com/user-data

Origin: http://site.com

User

Redirect to http://site.com