CORS Attacks

Author: Milad Khoshdel
Blog: https://blog.regux.com
Email: miladkhoshdel@gmail.com

Contents

What is CORS?

CORS or Cross-Origin Resource Sharing use in modern browsers to check the permission of remote access to web resources and services. For example, it's not possible to use font files from external URL in a website or it is impossible to send an Ajax request from an external domain. CORS is a tools for eliminating this limitation.

Let's look at the example below to make things more clear:

Site A requires an AJAX service that exists on site B. Normally, sending AJAX requests in this case is not possible due to browser restrictions. Modern browsers that support CORS send AJAX requests to a site B with an extra header called Origin.

```
Origin: http://foo.bar
```

After reviewing the Origin in Site B, it Origin value is valid, it will respond with Access-Control-Allow-Origin in header response and finally the Ajax request complete successfully.

```
Access-Control-Allow-Origin: http://foo.bar
```

The **Origin** header always sent in a CORS request by the browser and indicates the origin of the CORS request and **Access-Control-Allow-Origin** is a response header that used by a web server to indicate which domains are allowed to access the CORS response.

How to Test?

Now we should look for insecure configurations. For example If you send set a value for **Origin** header in request (for example foo.bar) and get a '*' wildcard as value of the Access-Control-Allow-Origin header in response, that means all domains are allowed to access the server response and it is a security vulnerability.

I can show you how to do that by example bellow:

The Request:

```
GET http://target.domain/file.php HTTP/1.1
Host: target.domain
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8;)
Gecko/20100101 Firefox/24.0
Accept: text/html
Accept-Language: en-US
Referer: http://foo.bar/
Origin: http://foo.bar/
Connection: keep-alive
```

The Response

```
HTTP/1.1 200 OK
Date: Fri, 23 NOV 2018 18:57:53 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.4-14+deb7u3
Access-Control-Allow-Origin: *
Content-Length: 44
Keep-Alive: timeout=18, max=89
Connection: Keep-Alive
Content-Type: application/xml

[Response Body]
```

As you see, we have a '*' wildcard as value of the Access-Control-Allow-Origin header in response and it means all domains are allowed to access the server response and it is an insecure configuration for CORS.

There is another type of CORS attack. If you send a random domain as value of origin header in request and you get the same domain name as value of the Access-Control-Allow-Origin header in response, it mean you successfully trusted your random domain to get the CORS responses.

It is a security vulnerability with high security (Cross-origin resource sharing: arbitrary origin trusted).

I can show you how to do that by example bellow:

The Request:

```
GET http://target.domain/file.php HTTP/1.1
Host: target.domain
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8;)
Gecko/20100101 Firefox/24.0
Accept: text/html
Accept-Language: en-US
Referer: http://foo.bar/
Origin: http://foo.bar/
Connection: keep-alive
```

The Response

```
HTTP/1.1 200 OK
Date: Fri, 23 NOV 2018 18:57:53 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.4-14+deb7u3
Access-Control-Allow-Origin: http://foo.bar/
Content-Length: 44
Keep-Alive: timeout=19, max=59
Connection: Keep-Alive
Content-Type: application/xml

[Response Body]
```

For doing it more accurate, I wrote a java script that send a URL as value of Origin in request header and check the value of Access-Control-Allow-Origin in the response header.

CORS Checker Script

For start testing you need to install java on your computer.

It is my installed java version:

```
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```

You can see your java version by bellow command on Linux/Windows.

```
Java –version
```

If you don't have installed java on your computer, you can download the latest version from oracle website:

```
https://www.oracle.com
```

Download Script Jar file from bellow URL:

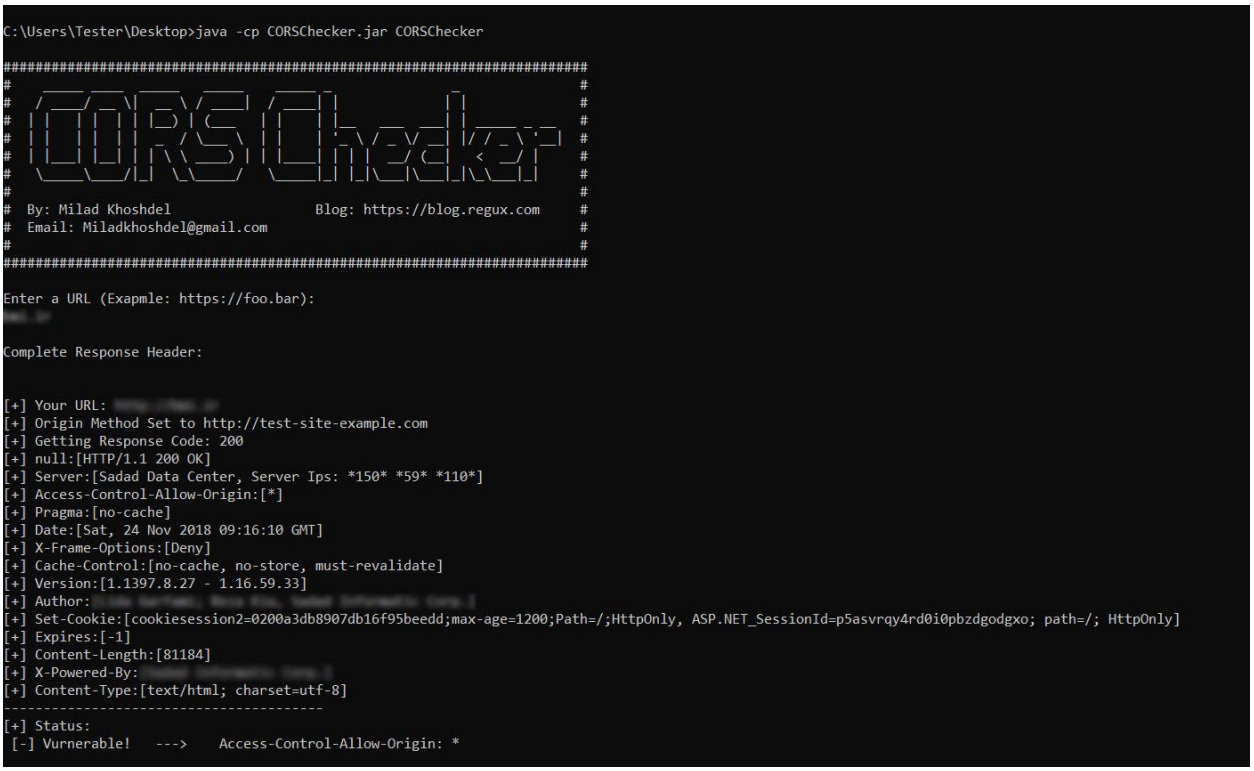
```
https://github.com/Miladkhoshdel/corschecker
```

Run it with bellow command:

```
java -cp CORSChecker.jar CORSChecker
```

Now enter your target URL here:

Script will send a website as value of Origin in request header and check the value of Access-Control-Allow-Origin in the response header.



In this example, our target return * as value of Access-Control-Allow-Origin header in response. So it means that this website is vulnerable to CORS attacks.

Have Fun
Milad Khoshdel

References

1. https://www.owasp.org/index.php/Test_Cross_Origin_Resource_Sharing_(OTG-CLIENT-007)
2. https://www.owasp.org/index.php/CORS_OriginHeaderScrutiny