Test Cross Origin Resource Sharing (OTG-CLIENT-007)

From OWASP

This article is part of the new OWASP Testing Guide v4.

Back to the OWASP Testing Guide v4 ToC:
https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents
Back to the OWASP Testing Guide Project:
https://www.owasp.org/index.php/OWASP_Testing_Project

Summary

Cross Origin Resource Sharing or CORS is a mechanism that enables a web browser to perform "cross-domain" requests using the XMLHttpRequest L2 API in a controlled manner. In the past, the XMLHttpRequest L1 API only allowed requests to be sent within the same origin as it was restricted by the same origin policy.

Cross-Origin requests have an Origin header, that identifies the domain initiating the request and is always sent to the server. CORS defines the protocol to use between a web browser and a server to determine whether a cross-origin request is allowed. In order to accomplish this goal, there are a few HTTP headers involved in this process, that are supported by all major browsers and we will cover below including: Origin, Access-Control-Request-Method, Access-Control-Request-Headers, Access-Control-Allow-Origin, Access-Control-Allow-Credentials, Access-Control-Allow-Methods, Access-Control-Allow-Headers.

The CORS specification mandates that for non simple requests, such as requests other than GET or POST or requests that uses credentials, a pre-flight OPTIONS request must be sent in advance to check if the type of request will have a bad impact on the data. The pre-flight request checks the methods, headers allowed by the server, and if credentials are permitted, based on the result of the OPTIONS request, the browser decides whether the request is allowed or not.

How to Test

Origin & Access-Control-Allow-Origin

The Origin header is always sent by the browser in a CORS request and indicates the origin of the request. The Origin header can not be changed from JavaScript however relying on this header for Access Control checks is not a good idea as it may be spoofed outside the browser, so you still need to check that application-level protocols are used to protect sensitive data.

Access-Control-Allow-Origin is a response header used by a server to indicate which domains are allowed to read the response. Based on the CORS W3 Specification it is up to the client to determine and enforce the restriction of whether the client has access to the response data based on this header.

From a penetration testing perspective you should look for insecure configurations as for example when the server returns back the Origin header in the Access-Control-Allow-Origin without any additional checks AND returns Access-Control-Allow-Credentials: true, which can lead to access of sensitive data. Note that this configuration is very insecure, and is not acceptable in general terms, except in the case of a public API that is intended to be accessible by everyone.

Access-Control-Request-Method & Access-Control-Allow-Method

The Access-Control-Request-Method header is used when a browser performs a preflight OPTIONS request and let the client indicate the request method of the final request. On the other hand, the Access-Control-Allow-Method is a response header used by the server to describe the methods the clients are allowed to use.

Access-Control-Request-Headers & Access-Control-Allow-Headers

These two headers are used between the browser and the server to determine which headers can be used to perform a cross-origin request.

Access-Control-Allow-Credentials

This header as part of a preflight request indicates that the final request can include user credentials.

Input validation

XMLHttpRequest L2 (or XHR L2) introduces the possibility of creating a cross-domain request using the XHR API for backwards compatibility. This can introduce security vulnerabilities that in XHR L1 were not present. Interesting points of the code to exploit would be URLs that are passed to XMLHttpRequest without validation, specially if absolute URLS are allowed because that could lead to code injection. Likewise, other part of the application that can be exploited is if the response data is not escaped and we can control it by providing user-supplied input.

Other headers

There are other headers involved like Access-Control-Max-Age that determines the time a preflight request can be cached in the browser, or Access-Control-Expose-Headers that indicates which headers are safe to expose to the API of a CORS API specification, both are response headers specified in the CORS W3C document.

Black Box testing

Black box testing for finding issues related to Cross Origin Resource Sharing is not usually performed since access to the source code is always available as it needs to be sent to the client to be executed.

Gray Box testing

Check the HTTP headers in order to understand how CORS is used, in particular we should be very interested in the Origin header to learn which domains are allowed and if credentials are allowed.

Tools

- **OWASP Zed Attack Proxy (ZAP) -** https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing. ZAP provides automated scanners as well as a set of tools that allow you to find security vulnerabilities manually.

References

OWASP Resources

- **OWASP HTML5 Security Cheat Sheet: https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet**

Whitepapers

- **W3C -** CORS W3C Specification: http://www.w3.org/TR/cors/

Retrieved from "https://www.owasp.org/index.php?title=Test_Cross_Origin_Resource_Sharing_(OTG-CLIENT-007)&oldid=247831"

Categories: OWASP Testing Project | Test
