



# 边界渗透中的小技巧

---

R3START@白帽汇安全研究院

# 个人简介

ID: R3start@白帽汇安全研究院

白帽汇高级打字工程师兼职初级渗透测试人员

希望能多结交一些志同道合的大佬

Blog: <http://r3start.net>

Github : <https://github.com/r35tart>



白帽汇安全研究院  
白帽汇安全研究院



## 案例分享

四月份的时候Github有一个项目名为：  
openXXXX

我在其中发现了多个内部域名，最后通过  
这些内部域名，结合接下来要讲的方法，  
成功发现了多个漏洞。



# 渗透流程



白帽汇安全研究院  
白帽汇安全研究院



# 资产收集

## 1. 目标主业务二级域名、三级域名等...多级域名收集

- ✓ 通过FOFA语法收集
- ✓ 通过子域名爆破、反查收集
- ✓ 通过JS接口收集
- ✓ 通过Github信息泄露
- ✓ ...

## 2. 业务强关联子公司资产收集

- ✓ 多级域名资产
- ✓ Github信息泄露
- ✓ 员工信息、管理后台
- ✓ ...

## 3. 目标IP资产、内网域名收集

- ✓ 线上测试环境
- ✓ Github信息泄露
- ✓ 历史漏洞信息
- ✓ JS代码
- ✓ ...

## 4. ...

但大部分都是....

 401 Unauthorized

 403 Forbidden

 404 Not Found

 500 Internal Server Error

## 资产收集





# 如何渗透401、403、404、500?

那么...我们应该怎么对这些这些页面开展渗透工作呢?

其实很多时候这些IP、域名往往都是一些脆弱的、高价值的又容易被突破的站点，但大部分人看到这些响应码后的操作最多也就扫扫端口、扫扫目录有发现就继续搞搞，没发现就丢掉，从而错失了打入内网的大好机会。

HOSTS碰撞



白帽汇安全研究院

# Hosts碰撞

很多时候访问目标资产响应多为：401、403、404、500，但是用域名请求却能返回正常的业务系统，因为这大多数都是需要绑定host才能正常请求访问的（目前互联网公司基本的做法），那么我们就可以通过收集到的目标的内网域名和目标资产的IP段组合起来，以IP段+域名的形式进行捆绑碰撞，就能发现很多有意思的东西。这一操作可以通过脚本自动化来访问：

[https://github.com/r35tart/Hosts\\_scan](https://github.com/r35tart/Hosts_scan)

```
import itertools
import signal
import threading
from multiprocessing.dummy import Pool
from time import sleep

from requests.packages import chardet
import requests
import re
from lib.processbar import ProcessBar

def host_check(host_ip):
    host_ip = host_ip
    schemes = ["http://", "https://"]
    for scheme in schemes:
        url = scheme + ip
        headers = {'Host': host.strip(), 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62'}
        try:
            r = requests.session()
            requests.packages.urllib3.disable_warnings()
            res = r.get(url, verify=False, headers=headers, timeout=30)
            charset = chardet.detect(res.content)["encoding"]
            res.encoding = charset
            title = ""
            try:
                title = re.search('<title>(.*?)</title>', res.text).group(1) # 获取标题
            except Exception as ex:
                title = u"获取标题失败"
            info = u'%s\t%s -- %s 数据包大小: %d 标题: %s' % (ip, host, scheme + host, len(res.text), title)
            if lock.acquire():
                try:
                    success_list.append(info)
                    pbar.echo(info)
                    pbar.update_suc()
                    with open('hosts_ok.txt', 'a+') as f:
                        f.write(info.encode("utf-8") + "\n")
                    f.close()
                finally:
                    lock.release()
```



# 脚本原理

在发送http请求的时候，对域名和IP列表进行配对，然后遍历发送请求（就相当于修改了本地的hosts文件一样），并把相应的title和响应包大小拿回来做对比，即可快速发现一些隐蔽的资产

```
for iplist in open("ip.txt"):
    ip = iplist.strip('\n')
    #读取host地址
    http_s = ['http://', 'https://']
    for h in http_s :
        for hostlist in open("host.txt", 'r'):
            host = hostlist.strip('\n')
            headers = {'Host': host, 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.97 Safari/537.36'}
            try:
                r = requests.session()
                requests.packages.urllib3.disable_warnings()
                rhost = r.get(h + ip, verify=False, headers=headers, timeout=5)
                rhost.encoding='utf-8'
                title = re.search('<title>(.*?)</title>', rhost.text).group(1) #获取标题
                info = '%s -- %s 协议: %s 数据包大小: %d 标题: %s' % (ip, host, h, len(rhost.text), title)
                lists.append(info)
                files.write(info + "\n")
                print(info)
            except Exception :
                error = ip + " --- " + host + " --- 访问失败! ~"
                print(error)
```

碰

某金融



## 需要用到的知识点

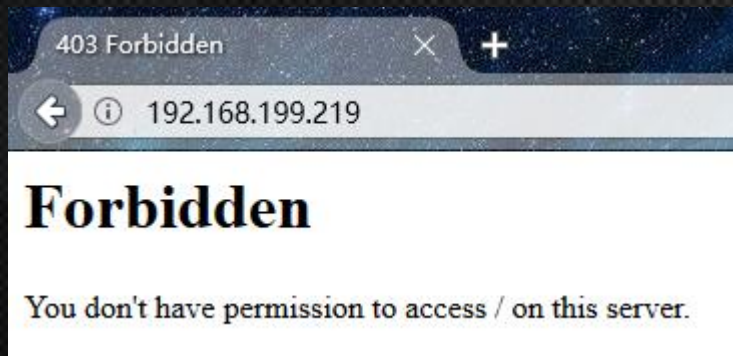
懂点网站搭建  
大概了解DNS解析过程





# 漏洞原理

如果管理员在配置apache或nginx的时候禁止了IP访问，那么我们直接访问IP将会回显403页面



(直接IP访问)

这时候访问网站则需要使用Apache的httpd.conf配置中的ServerName里指定的值才能够正常访问



(使用域名访问)

(apache\_httpd.conf配置)

```
<VirtualHost 192.168.199.219>
    ServerName 192.168.199.219
    <Location />
        Order Allow,Deny
        Deny from all
    </Location>
</VirtualHost>
<VirtualHost 192.168.199.219>
    DocumentRoot "C:\phpStudy\PHPTutorial\WWW"
    ServerName woshihoutai.r3start.baidu.com
</VirtualHost>
```



白帽汇安全研究院



# 漏洞原理

如果管理员在配置的时候ServerName域名写的是内网域名怎么办？  
(公网DNS服务器无法解析内部自定义域名)

大概了解一下DNS解析过程

- 1.在浏览器内部中查看是否有缓存
- 2.在本机hosts文件中查看是否有映射关系
- 3.本地DNS缓存 (ipconfig /displaydns)
- 4.本地DNS服务器
- 5.跟域服务器

通俗点讲：

当用户在浏览器中输入一个需要访问的网址时，浏览器会查看自身是否有缓存，没有系统则会检查自己的Hosts文件中是否有这个域名和IP的映射关系。如果有，则直接访问这个IP地址指定的网络位置，如果没有，再向的DNS服务器提出域名解析请求。**也就是说Hosts的IP解析优先级比DNS要高。**



# 漏洞原理

那么我们只需要知道目标的IP和域名即可通过修改本机Hosts访问到目标系统 ✓

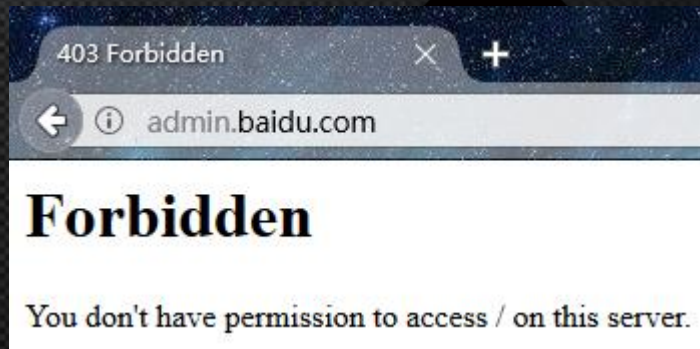
(本机Hosts添加映射关系)

```
hosts - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com           # source server
#       38.25.63.10        x.acme.com              # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1          localhost
#       ::1                localhost
#
192.168.199.219 woshihoutai.r3start.baidu.com
192.168.199.219 admin.baidu.com
```

(IP域名正确匹配 访问成功✓)



(绑定的域名不正确 访问失败✗)







不好意思 还没结束

# 资产收集

## 1. 目标主业务二级域名、三级域名等...多级域名收集

- ✓ 通过FOFA语法收集
- ✓ 通过子域名爆破、反查收集
- ✓ 通过JS接口收集
- ✓ 通过Github信息泄露
- ✓ ...

## 2. 业务强关联子公司资产收集

- ✓ 多级域名资产
- ✓ Github信息泄露
- ✓ 员工信息、管理后台
- ✓ ...

## 3. 目标IP资产、内网域名收集

- ✓ 线上测试环境
- ✓ Github信息泄露
- ✓ 历史漏洞信息
- ✓ JS代码
- ✓ ...

## 4. ...

 请输入账号

 请输入密码

 请输入验证码

登录

 请输入企业帐号

 请输入员工帐号

 请输入密码 

☐ 两周内自动登录 [忘记密码?](#)

立即登录



# 资产收集

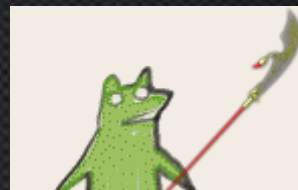
但尝试登陆后大部分都是....

密码需要八位以上必须包含大小写、数字和特殊符号

登陆账号为11位手机号码

请通过短信验证码登陆

请通过二维码登陆



然后你突然心血来潮，要爆破六位数验证码、爆破11位手机号来登陆，然后发现....



验证码: 065544, 密码。如非本人操作, 请忽略本短信。该验证码五分钟内有效。

```
$txtCapt.val(""); // 清空
$imgCapt.attr('src', '/page/jcaptcha.jpg?time=' + new Date().getTime());
});

// 登录异常信息处理
var MSG_JSON = {
  '1': '用户名或密码错误!',
  '2': '用户名或密码错误!',
  '3': '验证码错误!',
  '4': '该用户已被禁用!',
  '5': '帐号已锁定, 请在30分钟之后再试!',
  '6': '用户名或密码错误, 你还有1次机会!',
  '7': '用户名或密码错误, 你还有2次机会!',
  '8': '用户名或密码错误, 你还有3次机会!',
  '9': '用户名或密码错误, 帐号已锁定!',
};
```

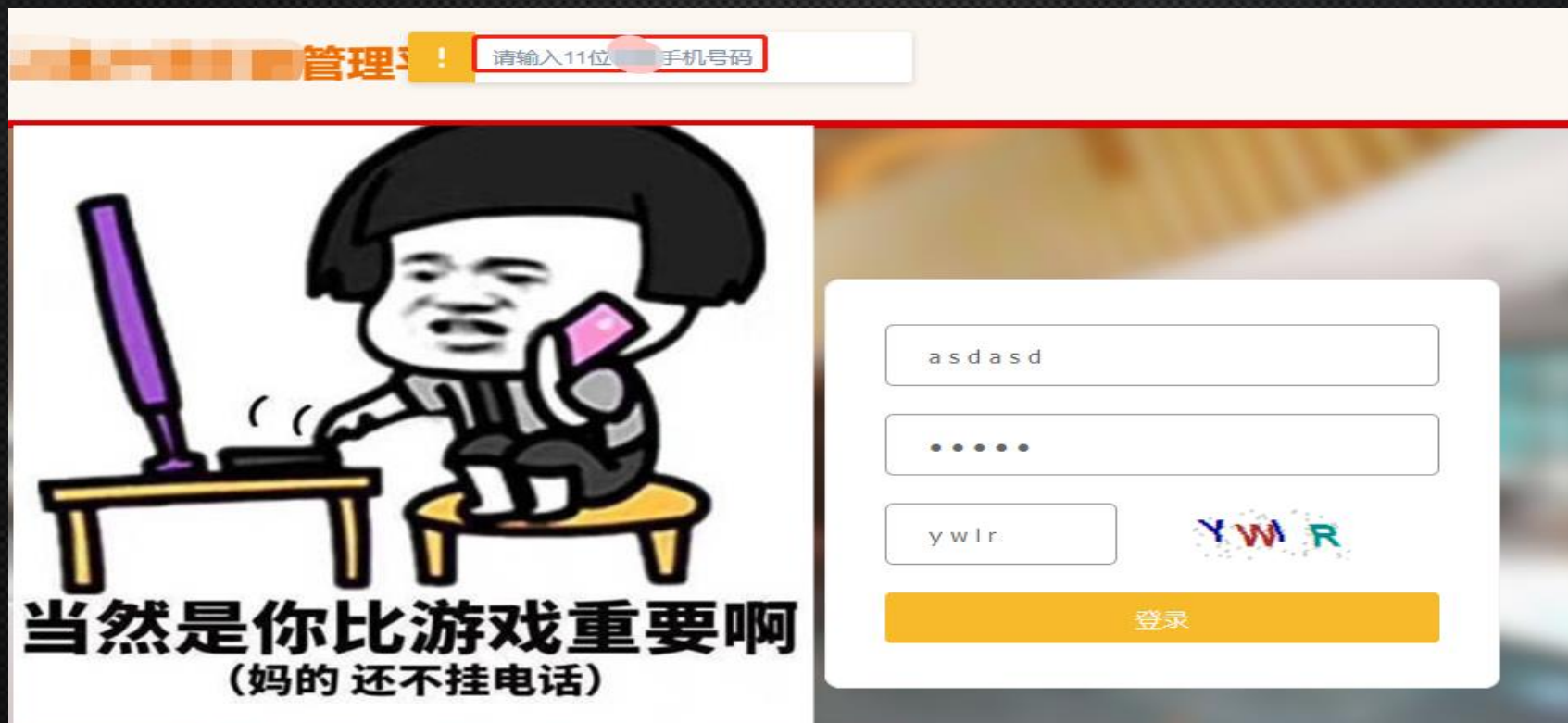
## 资产收集





## 案例分享

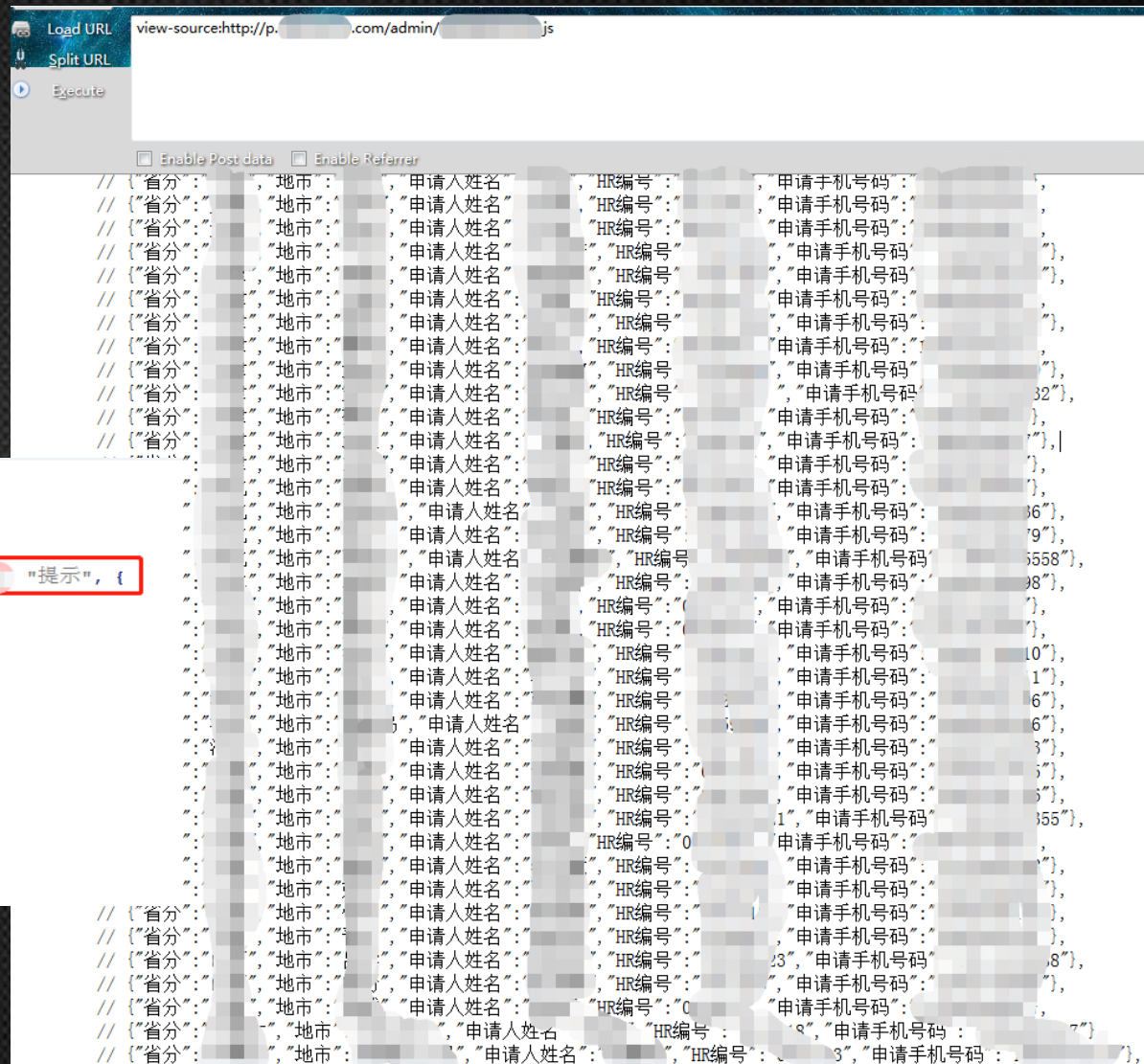
某目标系统后台登陆账号为11位手机号码，端口情况只开了80，443，看到账号是11位手机号码，我直接放弃了识别验证码爆破的想法，因为动作太大，可能性太小



## 通过JS寻找可用信息

**每当渗透进入死胡同的时候，不要放过任何可能有用的信息，可尝试通过查看js源码寻找可用信息。**

```
60689      })
60690      })
60691    },
60692    resetPwdFunc: function(t) {
60693      var e = this;
60694      e.$confirm("确定重置密码?密码恢复为账号后四位数字+ "提示", {
60695        confirmButtonText: "确定",
60696        cancelButtonText: "取消",
60697        type: "warning"
60698      }).then(function() {
60699        config.reqPost({
60700          url: "/mgt/resetpwd",
60701          params: {
60702            code: t
60703          },
60704          success: function(t, i) {
60705            e.$message({
60706              type: "success",
60707              message: i
60708            })
60709          }
60710        })
60711      })
60712    }
60713  },
60714  data: {
60715    title: "重置密码",
60716    code: ""
60717  },
60718  methods: {
60719    resetPwd: function() {
60720      this.resetPwdFunc(this.code)
60721    }
60722  }
60723 }
60724 
60725 
60726 
60727 
60728 
60729 
60730 
60731 
60732 
60733 
60734 
60735 
60736 
60737 
60738 
60739 
60740 
60741 
60742 
60743 
60744 
60745 
60746 
60747 
60748 
60749 
60750 
60751 
60752 
60753 
60754 
60755 
60756 
60757 
60758 
60759 
60760 
60761 
60762 
60763 
60764 
60765 
60766 
60767 
60768 
60769 
60770 
60771 
60772 
60773 
60774 
60775 
60776 
60777 
60778 
60779 
60780 
60781 
60782 
60783 
60784 
60785 
60786 
60787 
60788 
60789 
60790 
60791 
60792 
60793 
60794 
60795 
60796 
60797 
60798 
60799 
60800 
60801 
60802 
60803 
60804 
60805 
60806 
60807 
60808 
60809 
60810 
60811 
60812 
60813 
60814 
60815 
60816 
60817 
60818 
60819 
60820 
60821 
60822 
60823 
60824 
60825 
60826 
60827 
60828 
60829 
60830 
60831 
60832 
60833 
60834 
60835 
60836 
60837 
60838 
60839 
60840 
60841 
60842 
60843 
60844 
60845 
60846 
60847 
60848 
60849 
60850 
60851 
60852 
60853 
60854 
60855 
60856 
60857 
60858 
60859 
60860 
60861 
60862 
60863 
60864 
60865 
60866 
60867 
60868 
60869 
60870 
60871 
60872 
60873 
60874 
60875 
60876 
60877 
60878 
60879 
60880 
60881 
60882 
60883 
60884 
60885 
60886 
60887 
60888 
60889 
60890 
60891 
60892 
60893 
60894 
60895 
60896 
60897 
60898 
60899 
60900 
60901 
60902 
60903 
60904 
60905 
60906 
60907 
60908 
60909 
60910 
60911 
60912 
60913 
60914 
60915 
60916 
60917 
60918 
60919 
60920 
60921 
60922 
60923 
60924 
60925 
60926 
60927 
60928 
60929 
60930 
60931 
60932 
60933 
60934 
60935 
60936 
60937 
60938 
60939 
60940 
60941 
60942 
60943 
60944 
60945 
60946 
60947 
60948 
60949 
60950 
60951 
60952 
60953 
60954 
60955 
60956 
60957 
60958 
60959 
60960 
60961 
60962 
60963 
60964 
60965 
60966 
60967 
60968 
60969 
60970 
60971 
60972 
60973 
60974 
60975 
60976 
60977 
60978 
60979 
60980 
60981 
60982 
60983 
60984 
60985 
60986 
60987 
60988 
60989 
60990 
60991 
60992 
60993 
60994 
60995 
60996 
60997 
60998 
60999 
61000 
61001 
61002 
61003 
61004 
61005 
61006 
61007 
61008 
61009 
61010 
61011 
61012 
61013 
61014 
61015 
61016 
61017 
61018 
61019 
61020 
61021 
61022 
61023 
61024 
61025 
61026 
61027 
61028 
61029 
61030 
61031 
61032 
61033 
61034 
61035 
61036 
61037 
61038 
61039 
61040 
61041 
61042 
61043 
61044 
61045 
61046 
61047 
61048 
61049 
61050 
61051 
61052 
61053 
61054 
61055 
61056 
61057 
61058 
61059 
61060 
61061 
61062 
61063 
61064 
61065 
61066 
61067 
61068 
61069 
61070 
61071 
61072 
61073 
61074 
61075 
61076 
61077 
61078 
61079 
61080 
61081 
61082 
61083 
61084 
61085 
61086 
61087 
61088 
61089 
61090 
61091 
61092 
61093 
61094 
61095 
61096 
61097 
61098 
61099 
61100 
61101 
61102 
61103 
61104 
61105 
61106 
61107 
61108 
61109 
61110 
61111 
61112 
61113 
61114 
61115 
61116 
61117 
61118 
61119 
61120 
61121 
61122 
61123 
61124 
61125 
61126 
61127 
61128 
61129 
61130 
61131 
61132 
61133 
61134 
61135 
61136 
```





# 通过JS寻找可用信息

使用低权限账号登陆后，还可以通过js寻找接口信息，大部分接口很可能存在越权

管理平台 / 会员管理

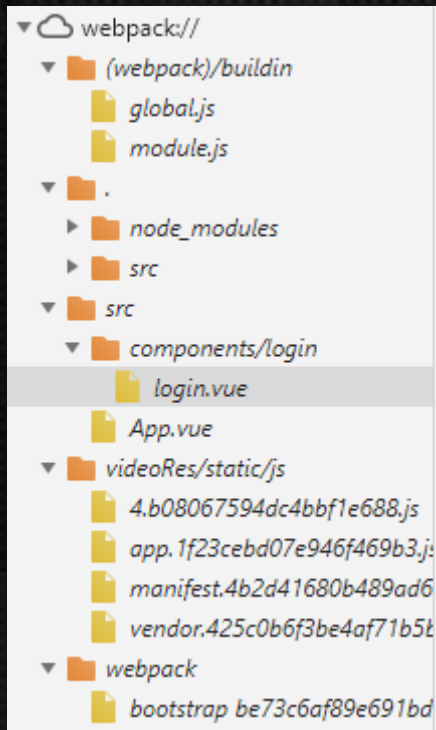
手机号码  省分  查询

账号	昵称	省分	注册时间	上次登录	状态	操作
		广东-广州	2017-7-05	2019-4-17	正常	<a href="#">查看用户兴趣标签</a>
		广东-广州	2017-7-05	2019-5-29	正常	<a href="#">查看用户兴趣标签</a>
		广东-广州	2017-7-05	2019-5-30	正常	<a href="#">查看用户兴趣标签</a>
		广东-广州	2017-7-05	2019-4-02	正常	<a href="#">查看用户兴趣标签</a>
		广东-广州	2017-7-05	2019-5-31	正常	<a href="#">查看用户兴趣标签</a>
		湖北-武汉	2017-7-05	2019-5-06	正常	<a href="#">查看用户兴趣标签</a>
		广东-广州	2017-7-06	2019-4-01	正常	<a href="#">查看用户兴趣标签</a>
		贵州-贵阳	2017-7-06	2019-5-24	正常	<a href="#">查看用户兴趣标签</a>
		内蒙古-呼和浩特	2017-7-06	2019-5-15	正常	<a href="#">查看用户兴趣标签</a>
		天津	2017-7-06	2019-5-20	正常	<a href="#">查看用户兴趣标签</a>
		浙江-杭州	2017-7-06	2019-5-20	正常	<a href="#">查看用户兴趣标签</a>
		天津	2017-7-06	2019-3-27	正常	<a href="#">查看用户兴趣标签</a>

共 26744555 条 1 2 3 4 5 6 ... 1782971 前往 1/1 页

# 通过JS寻找可用信息

部分VUE站点，还可以通过F12查看webpack打包前的前端代码，可从注释中获取敏感信息



```
//alert(this.userName+" "+this.userPwd)
//axios.post(this.$store.state.baseUrl.auth + 'oauth/token?username='+this.userName+'&password='+this.userPwd+'&client_id='+client1+'&grant_type='+password+'&client_secret='+123456')
//console.log("登录参数: "+this.userName+" "+this.userPwd)
//let loginUrl = this.$store.state.baseUrl.auth + 'Login/form?username='+this.userName+'&password='+this.userPwd+'&v='+new Date().getTime()
//console.log("登录URL: "+loginUrl)
axios({
  method: 'post',
  url: this.$store.state.baseUrl.auth + 'oauth/token',
  // headers: {
  //   // 'Content-type': 'application/json;charset=UTF-8'
  //   // 'Content-type': 'application/x-www-form-urlencoded'
  // },
  params: {
    'grant_type': 'password',
    // 'client_id': 'client1',
    // 'client_secret': '123456',
    'username': this.userName,
    'password': this.getAES(this.userPwd),
    'code': this.userCode,
    'randomStr': this.randomStr
  }
})
.then(res => {
  //console.log("登录结果: "+JSON.stringify(res))
  if (res && res.status === 200) {
    setToken(res.data.access_token);
    setReToken(res.data.refresh_token);
    this.$router.push("/selectSystem");
    //this.$router.push('/monitor')// 审核花销
  } else {
    this.refreshCode();
    // ElementUI.Message({
    //   type: "error",
    //   //message: Base64.encode('账号或密码不正确')
    //   message: "账号或密码不正确"
    // });
    //$("#error-notice").show()
    //next()
  }
})
})
```

```
52 setUserInfo,
53 getUserInfo,
54 getToken,
55 setToken,
56 setReToken,
57 getReToken
58 } from "@/utils/auth";
59 // let Base64 = require('js-base64').Base64; -- 未使用
60 axios.interceptors.request.use(
61   config => {
62     //config.headers.Authorization = 'Basic Z3JhbnRfdHlwZS50Zm9udF9zZW50ZXQxMjM0NTY=';
63     //注释 17
64     // config.headers.Authorization = "Basic Y2xpZW50MT0xMjM0NTY=";
65     config.headers.Authorization = "Basic bWptaDptaWZsdDFqZm9udF9zZW50ZXQxMjM0NTY=";
66     //config.headers['Content-Type'] = 'application/x-www-form-urlencoded';
67     return config;
68   },
69   err => {}
70 );
71
```



## 总结

渗透中需要养成不放过查看任何文件的习惯，有时候右键查看JS源码、习惯性查看F12，你可能会发现... 被注释的账号密码、接口、token、真实IP、开发环境地址等....

永远不知道程序员会在JS中给你留下了什么样的惊喜。

比如：

玫瑰金手铐一对

+

精美囚服一套

+

保镖全方位保护体验

一切渗透工作均需在得到授权的情况下开展



白帽汇安全研究院  
白帽汇安全研究院



THANKS

感谢观看