



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

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



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，从而错过了打入内网的大好机会。

HOSTS碰撞



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>

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

from requests.packages import chardet
import requests
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
        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 = ""
        except:
            title = re.search('<title>(.*?)</title>', res.text).group(1) # 获取标题
            title = "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_succ()
                    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 ip in open("ip.txt"):
    ip = ip.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/62.0.3181.150 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)
```

碰撞案例

融金某

[illegible]

漏洞原理

需要用到的知识点

懂点网站搭建

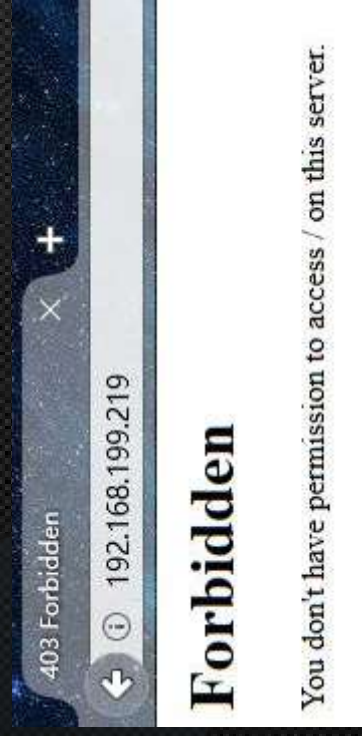
大概了解DNS解析过程



自帽汇安全研究院

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)

(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>
```

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



(使用域名访问)



自帽汇安全研究院

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地址时，浏览器会查看自身是否有缓存，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，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

那么我們只需要知道目标的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

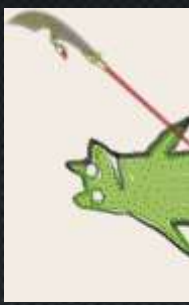
但尝试登陆后大部分都是....

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....

请点击下图中的所有 相声演员 刷新



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
$ingCapt.attr('src', '/page/jcaptcha.jpg?time=' + new Date

});

// 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



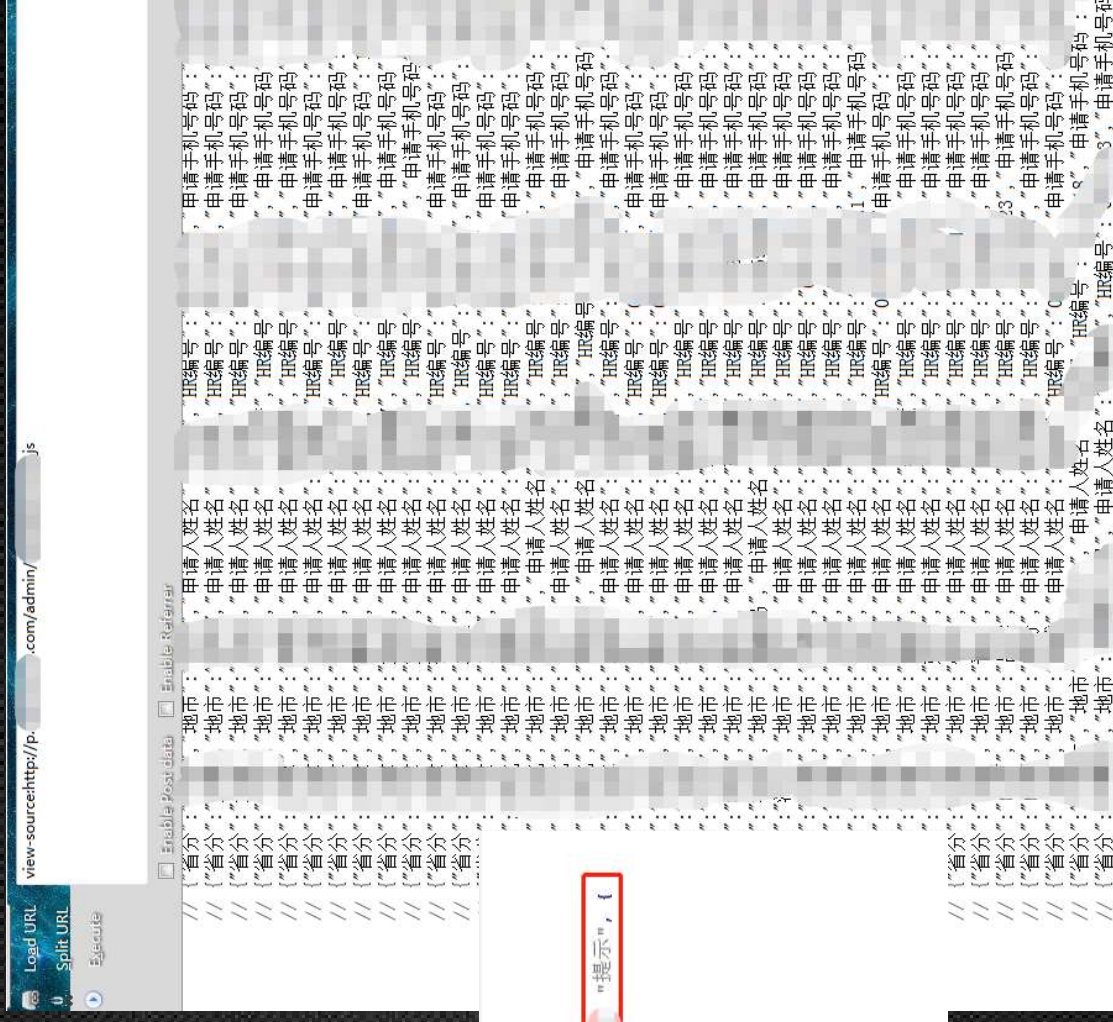
案例分享

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  
60693  
60694  
60695  
60696  
60697  
60698  
60699  
60700  
60701  
60702  
60703  
60704  
60705  
60706  
60707  
60709  
  
    },  
    resetPwdFunc: function(t) {  
        e.$confirm("确定重置密码?密码恢复为账号后四位数字+  
            "提示", {  
                cancelButtonText: "确定",  
                type: "warning"  
            }).then(function() {  
                config.reqPost({  
                    url: "/mgt/resetpwd",  
                    params: {  
                        code: t  
                    },  
                    success: function(t, i) {  
                        e.$message({  
                            type: "success",  
                            message: i  
                        })  
                    }  
                })  
            })  
        }  
    }  
}
```

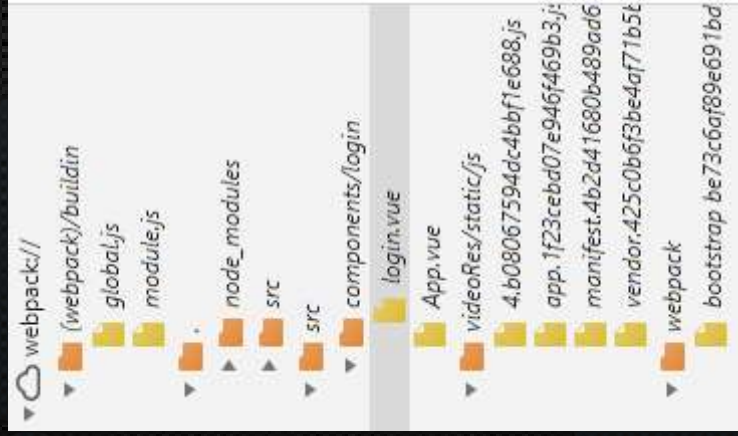

通过JS寻找可用信息

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



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.$store.state.baseUrl.auth +
//console.log("登录参数: "+this.userName+" "+this.userPwd)
//let loginUrl = this.$store.state.baseUrl.auth + '
//console.log("登录URL: "+loginUrl)
axios({
  method: 'post',
  url: this.$store.state.baseUrl.auth + '
  // headers: {
  // // 'Content-type': 'application/json;charset=UTF-8'
  // // 'Content-type': 'application/x-www-form-urlencoded'
  // },
  params: {
    'grant_type': 'password',
    // 'client_id': 'client1',
    // 'client_secret': '123456',
    'username': this.userName,
    'password': this.getAES(this.userPwd),
    'code': this.userCode,
    'randomStr': this.randomStr
  }
})
.then(res => {
  //console.log("登录结果: "+JSON.stringify(re
  if (res 88 res.status === 200) {
    setToken(res.data.access_token);
    setRefreshToken(res.data.refresh_token);
    this.$router.push("/selectSystem"); // 修改路径
  } 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()
  }
})
})
```

```

  setUserInfo,
  getUserInfo,
  getTokens,
  setToken,
  setRefreshToken,
  getRefreshToken,
  } from "@utils/auth";
  // Let Base64 = require('js-base64').Base64; -- 未使用
  axios.interceptors.request.use(
    config => {
      //config.headers.Authorization = 'Basic Z3JhbnRfZm9udGVudF99NmMsaWVudF99
      //config.headers.Authorization = "Basic Y2xpZW50MT0xYjM0NTY=";
      config.headers.Authorization = "Basic bWp0aDp0a1ZsdFpZm9hNGdhc2Rm";
      //config.headers['Content-Type'] = 'application/x-www-form-urlencoded';
      return config;
    },
    err => {}
  );
}
```

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





THANKS

感谢观看