

Homework1

目录

1 作业说明	3
1.1 实验目的	3
1.2 实验环境	3
1.3 实验要求	3
1.4 实验报告要求	5
2 安装 VirtualBox 虚拟机软件	7
2.1 下载	7
2.2 安装	8
3 在 VirtualBox 虚拟机中安装XP	8
3.1 下载	8
3.2 解压	9
3.3 导入	10
3.4 安装XP1	12
3.5 设置多重加载	17
3.6 安装XP2	18
4 在 VirtualBox 虚拟机中安装 Kali	19
4.1 下载	19
4.2 解压	20
4.3 安装Kali1	21
4.4 设置多重加载	22
4.5 安装Kali2	23
5 在 VirtualBox 虚拟机中安装安装 Debian	24

6 设置网络	25
6.1 修改名称	25
6.2 设置内部网络1	26
6.3 设置内部网络2	29
6.4 设置NatNetwork网络	32
6.4.1 选用NAT网络的原因	32
6.4.2 创建NAT网络	32
6.4.3 配置Attacker	36
6.4.4 配置网关	37
6.5 添加DCHP服务（但是似乎不需要）	37
7 网络测试	43
7.1 针机可以直接访问攻击者主机	43
7.1.1 XP1	43
7.1.2 Kali1	44
7.1.3 XP2	45
7.1.4 Debian2	46
7.2 攻击者主机无法直接访问针机	47
7.2.1 XP1	47
7.2.2 XP2	48
7.3 网关可以直接访问攻击者主机和针机	49
7.3.1 攻击者主机	49
7.3.2 XP1	49
7.4 针机的所有对外上下行流量必须经过网关	51
7.5 所有节点均可以访问互联网	55
7.5.1 攻击者	55
7.5.2 网关	56
7.5.3 XP1	57
7.5.4 Kali1	58
7.5.5 XP2	59
7.5.6 Debian2	60

1 作业说明

1.1 实验目的

- 掌握 VirtualBox 虚拟机的安装与使用；
- 掌握 VirtualBox 的虚拟网络类型和按需配置；
- 掌握 VirtualBox 的虚拟硬盘多重加载；

1.2 实验环境

以下是本次实验需要使用的网络节点说明和主要软件举例：

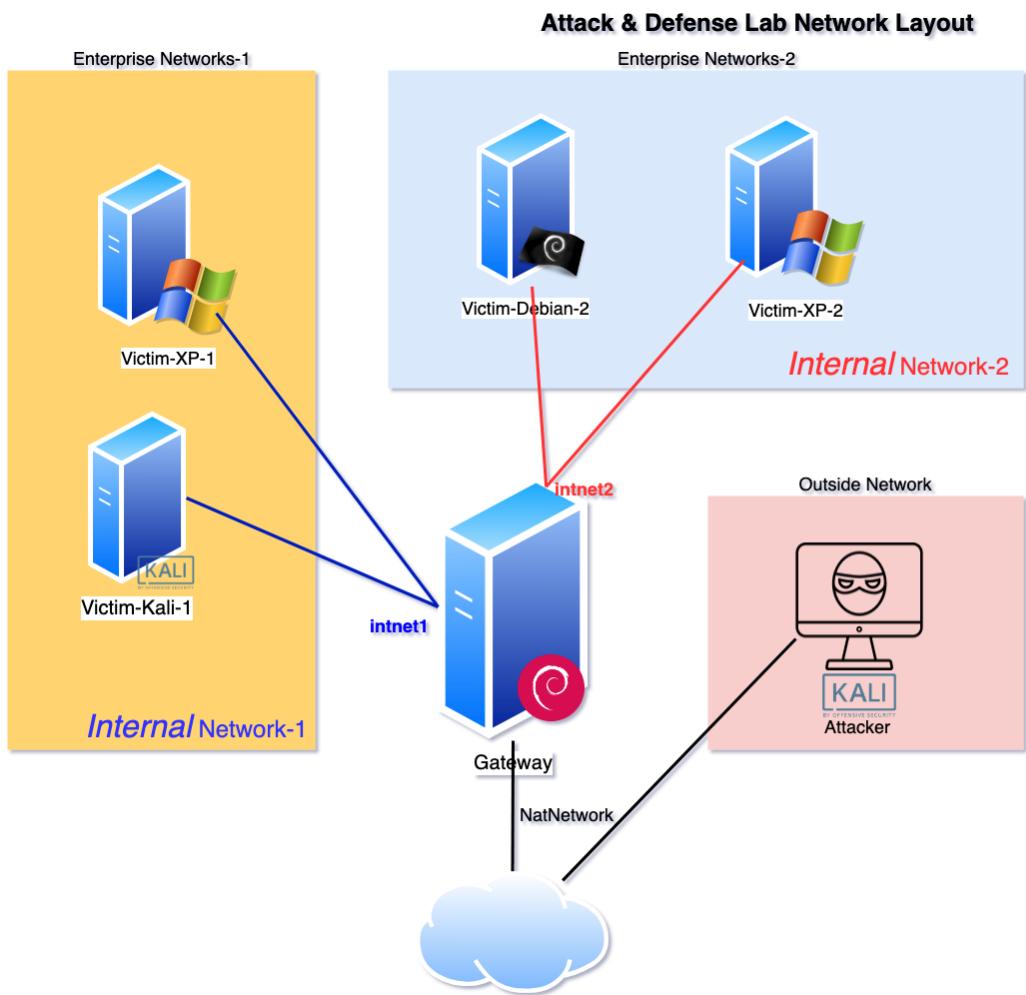
- VirtualBox 虚拟机
- 攻击者主机（Attacker）：Kali Rolling 2019.2
- 网关（Gateway, GW）：Debian Buster
- 靶机（Victim）：From Sqli to shell / xp-sp3 / Kali

1.3 实验要求

- 虚拟硬盘配置成多重加载，效果如下图所示；



- 搭建满足如下拓扑图所示的虚拟机网络拓扑；



根据实验宿主机的性能条件，可以适度精简靶机数量

- 完成以下网络连通性测试：
 - 靶机可以直接访问攻击者主机
 - 攻击者主机无法直接访问靶机
 - 网关可以直接访问攻击者主机和靶机
 - 靶机的所有对外上下行流量必须经过网关
 - 所有节点均可以访问互联网

1.4 实验报告要求

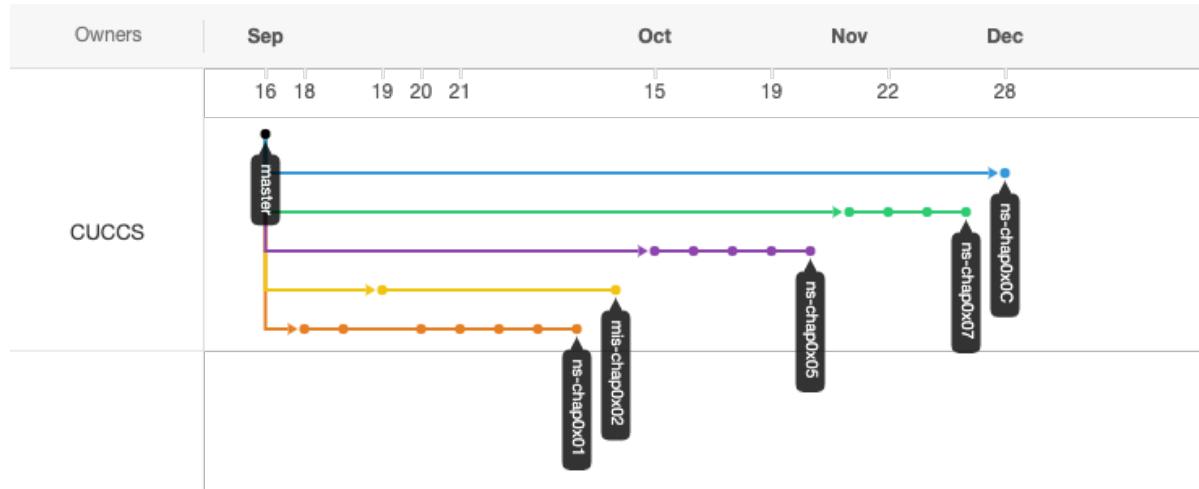
- Markdown 书写，且在 Github 上渲染出的排版效果正常，可读性强；
- 在 Github 上每次提交作业单独从 `master` 分支 **新开一个分支**；
 - 每次作业均保存到 **独立不冲突** 的子目录；
- 图文并茂证明：

- 实验关键步骤是自己做的;
- 哪些实验结果符合实验要求预期;
- 如有涉及到代码、配置文件, 请记得 commit 源代码 文件;
- 规范的 Git 工作流程:
 - 提交作业等待批改: 提交 PR 请求将作业分支合并到 master 分支;
 - * 未 PR 时的 commit 不会被批改;
 - 课程没有在教务处系统上查到分数之前, 禁止合并或关闭 已有批改记录的 PR, 可以在该 PR 对应的分支上继续提交新变更; ;
 - 每次实验报告只保留一个 Open 状态的 PR , 禁止同一次作业发起多个 PR;
 - PR 的标题应体现本次实验报告的主题;

示例作业目录（所有分支合并到 master 分支后状态）如下：

```
.  
├── .gitattributes  
├── .gitignore  
└── README.md  
├── chap0x01  
│   ├── README.md  
│   └── img  
│       ├── vb-setup.png  
│       └── vb-victim-screenshot-1.png  
├── chap0x02  
│   └── README.md  
├── chap0x03  
│   └── README.md  
├── chap0x04  
│   └── README.md  
└── chap0x05  
    ├── README.md  
    └── code  
        ├── exp.py  
        └── nginx.conf
```

示例 Git 分支结构如下：



2 安裝 VirtualBox 虛擬機軟件

2.1 下載

首先，在`https://www.virtualbox.org/`下載 VirtualBox 虛擬機的安裝包。



Figure 1: download_vbox.png

2.2 安裝

随后进行安装，安装后打开软件主页面。



Figure 2: vbox_index.png

3 在 VirtualBox 虚拟机中安装XP

3.1 下载

在老师提供的链接中找到XP的镜像：



Figure 3: xp_kod.png

点击进行下载。

3.2 解压

下载完成后使用解压缩工具进行解压。

名称	修改日期	类型	大小
MACOSX	2022/9/18 12:27	文件夹	
xp_sp3_base.vdi	2020/9/13 10:29	360压缩	1,610,752 KB

这个是吴杰的作业里的图!

Figure 4: xp_unzipped.png

3.3 导入

随后需要将这个磁盘文件导入。

首先进入虚拟介质管理:



Figure 5: vdi_mng.png

将XP的vdi文件注册：

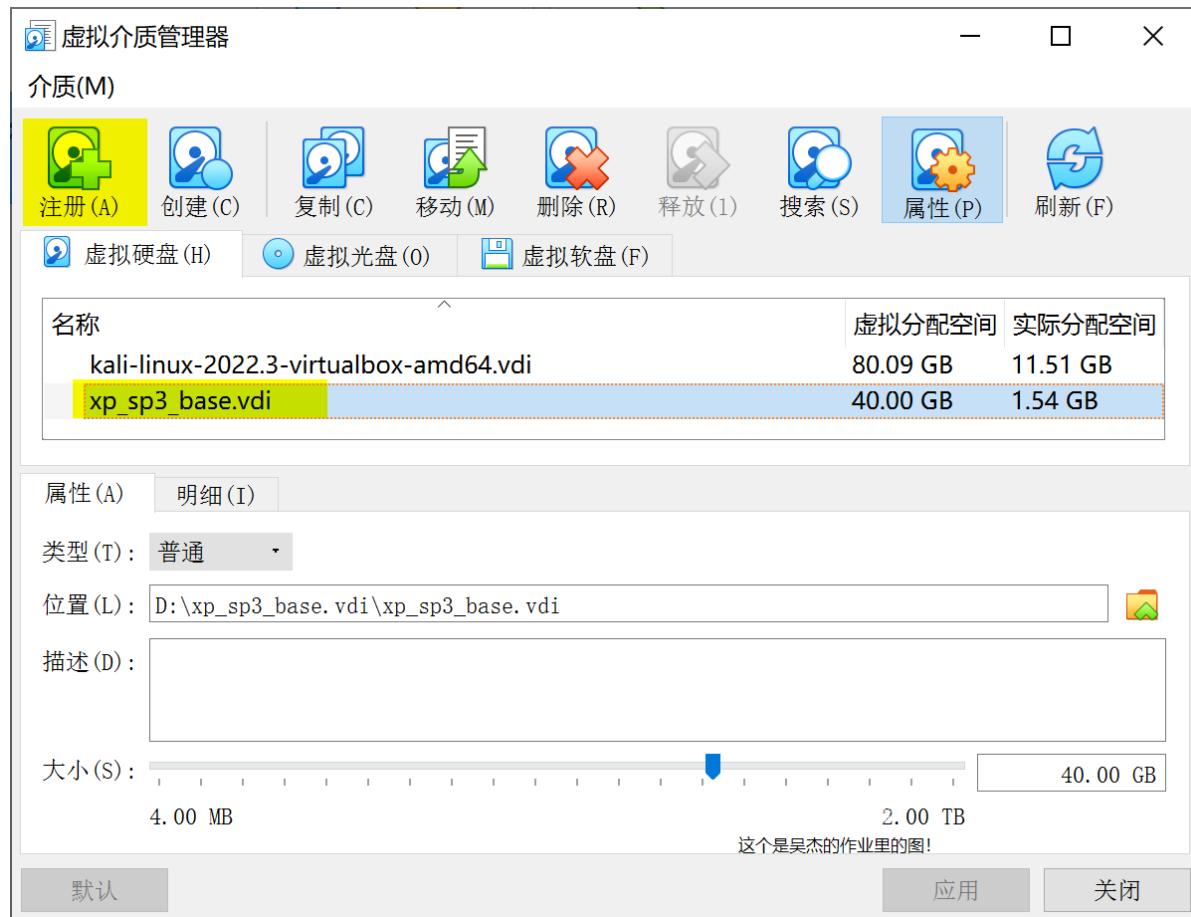


Figure 6: reg_xp_vid.png

3.4 安裝XP1

点击新建按钮:



Figure 7: new_vm.png

设定虚拟机信息：

← 新建虚拟电脑

虚拟电脑名称和系统类型

请选择新虚拟电脑的描述名称及要安装的操作系统类型。此名称将用于标识此虚拟电脑。

名称: XP1

文件夹: D:\vbox_vms

类型(T): Microsoft Windows

版本(V): Windows XP (32-bit)



这个是吴杰的作业里的图!

专家模式 (E)

下一步 (N)

取消

Figure 8: set_new_vm_info.png

设定内存:



Figure 9: set_new_vm_memory.png

将硬盘设定为刚刚下载的xp的vdi:

?

×

← 新建虚拟电脑

虚拟硬盘

你可以添加虚拟硬盘到新虚拟电脑中。新建一个虚拟硬盘文件或从列表或用文件夹图标从其他位置选择一个。

如果想更灵活地配置虚拟硬盘，也可以跳过这一步，在创建虚拟电脑之后在配置中设定。

建议的硬盘大小为 10.00 GB。

- 不添加虚拟硬盘 (D)
- 现在创建虚拟硬盘 (C)
- 使用已有的虚拟硬盘文件 (U)

xp_sp3_base.vdi (普通, 40.00 GB)

吴杰的作业的图

这个是吴杰的作业里的图!

创建

取消

Figure 10: set_new_vm_vdi.png

随后完成创建，启动XP1：



Figure 11: XP1_desktop.png

3.5 设置多重加载

由于这个镜像需要给多个虚拟机使用，所以设置多重加载。



Figure 12: set_mult_attach.png

3.6 安装XP2

与XP1安装时的设置一致，磁盘同样选择下载的xp的vdi。

这样，两台XP就都装好了：



Figure 13: xp1_xp2_desktop.png

4 在 VirtualBox 虚拟机中安装 Kali

4.1 下载

在<https://www.kali.org/get-kali/#kali-installer-images>中找到已经配置好的Kali环境：

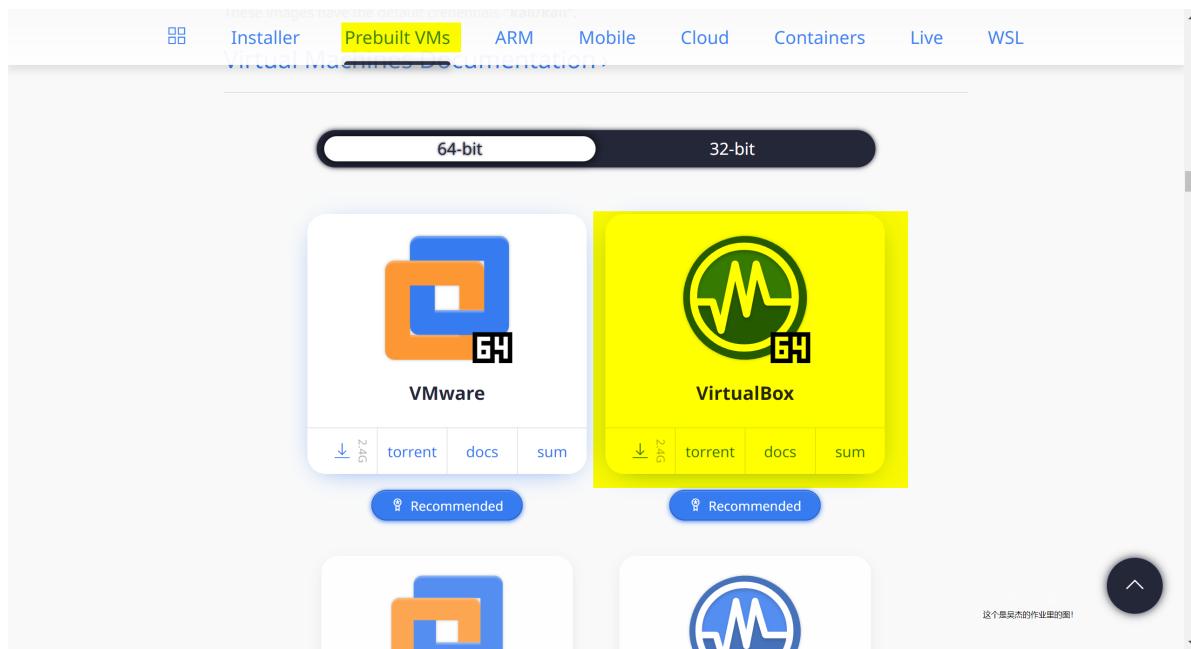


Figure 14: download_prebuild_kali_vbox.png

点击下载按钮，开始下载：

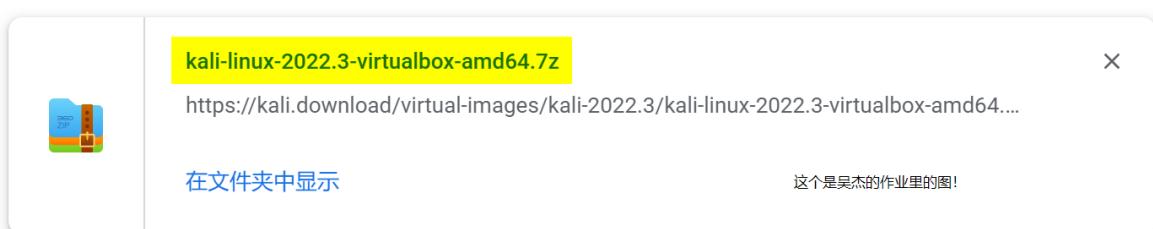


Figure 15: kali_vbox_downloaded.png

4.2 解压

随后使用压缩工具对其进行解压：

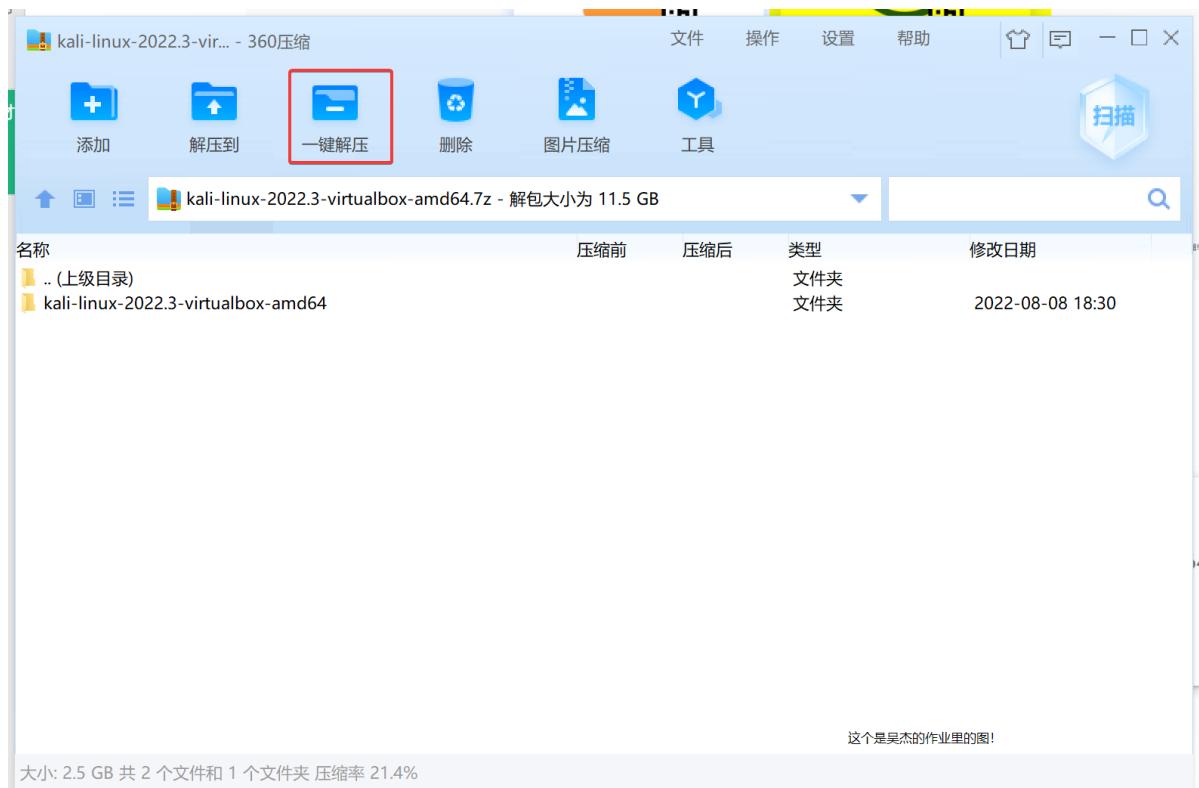


Figure 16: unzip_kali_box.png

4.3 安装Kali1

双击解压后的vbox文件，即可导入：



Figure 17: kali_imported.png

4.4 设置多重加载

同样的，因为有多台kali，需要设置多重加载：

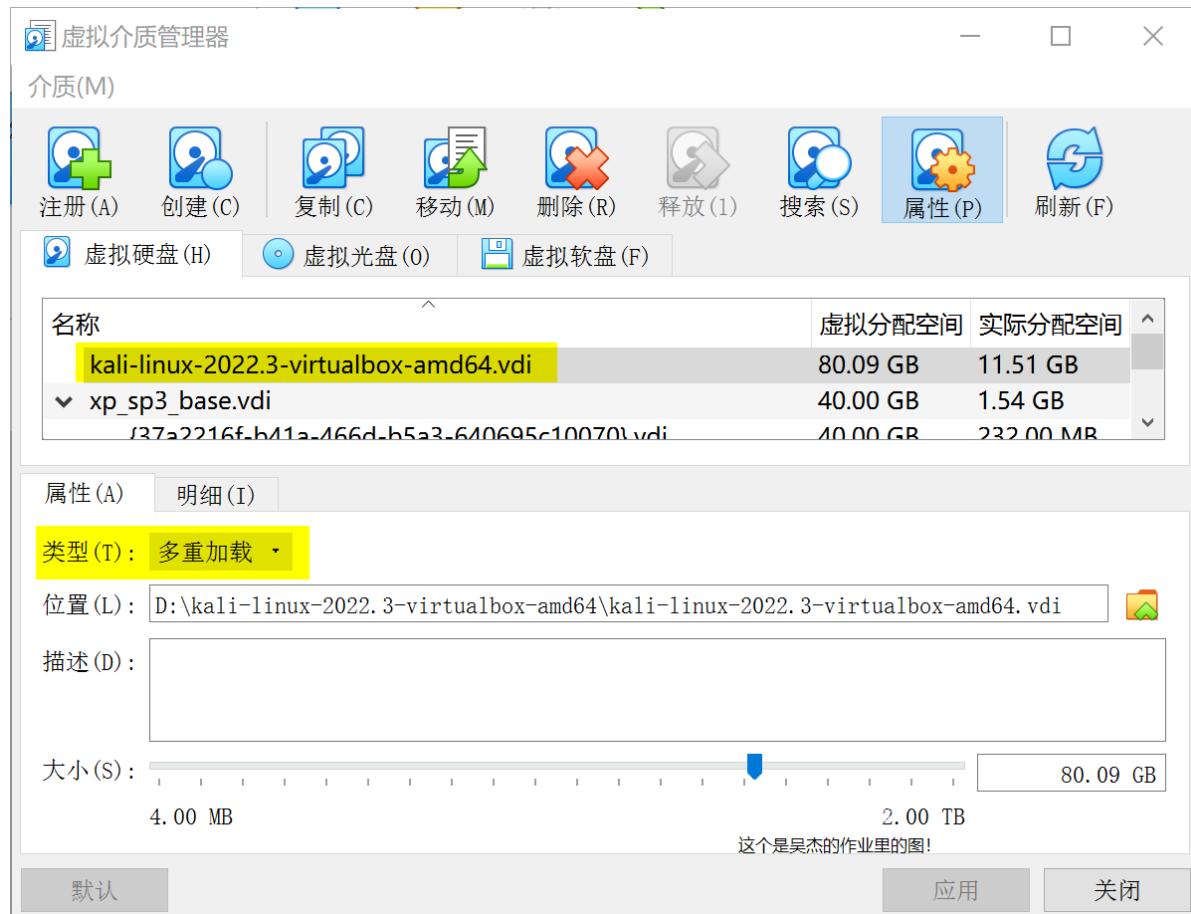


Figure 18: set_kali_mult_att.png

4.5 安装Kali2

使用和XP一样的方法安装第二台Kali

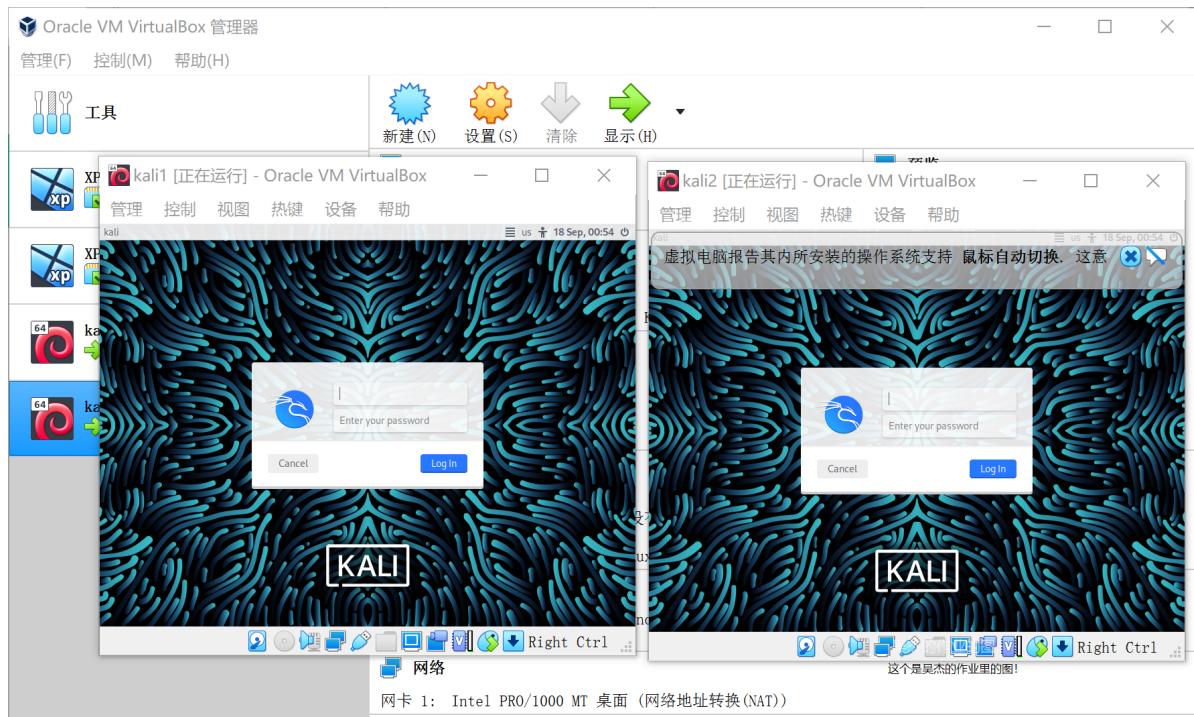


Figure 19: kali1_kali2_login.png

5 在 VirtualBox 虚拟机中安装 Debian

和XP、Kali一样：

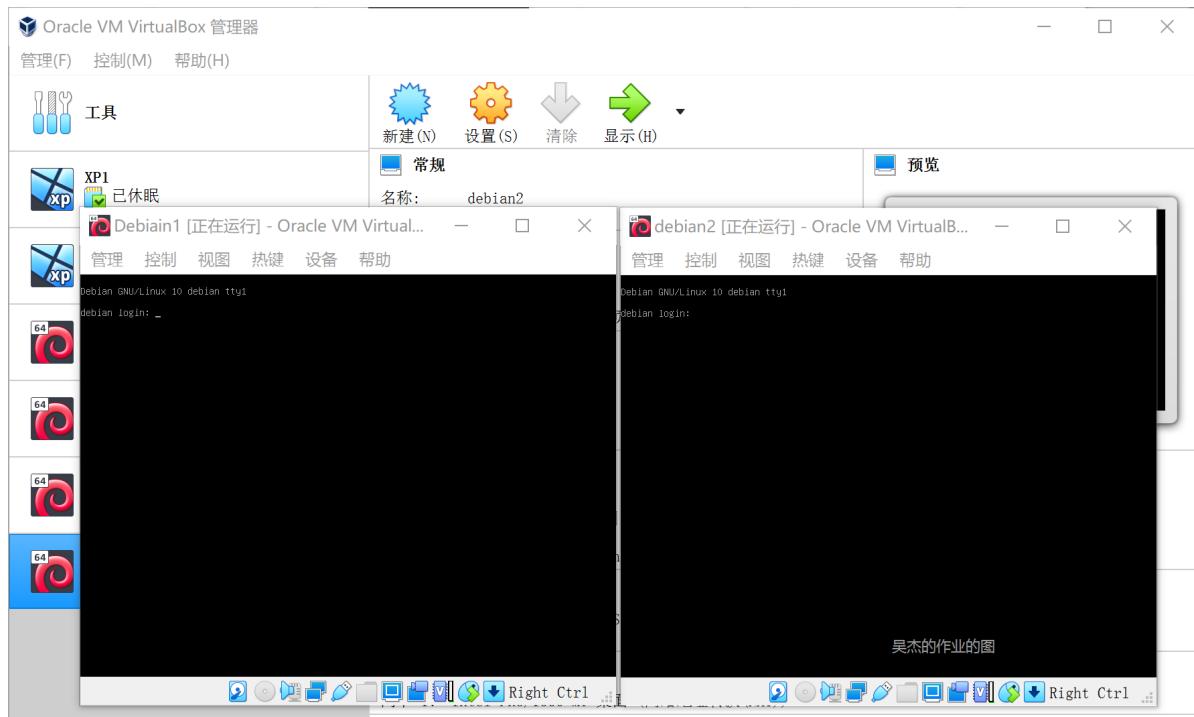


Figure 20: debian1_debian2_login.png

6 设置网络

6.1 修改名称

首先，根据网络拓扑图，修改各个虚拟机的名称：

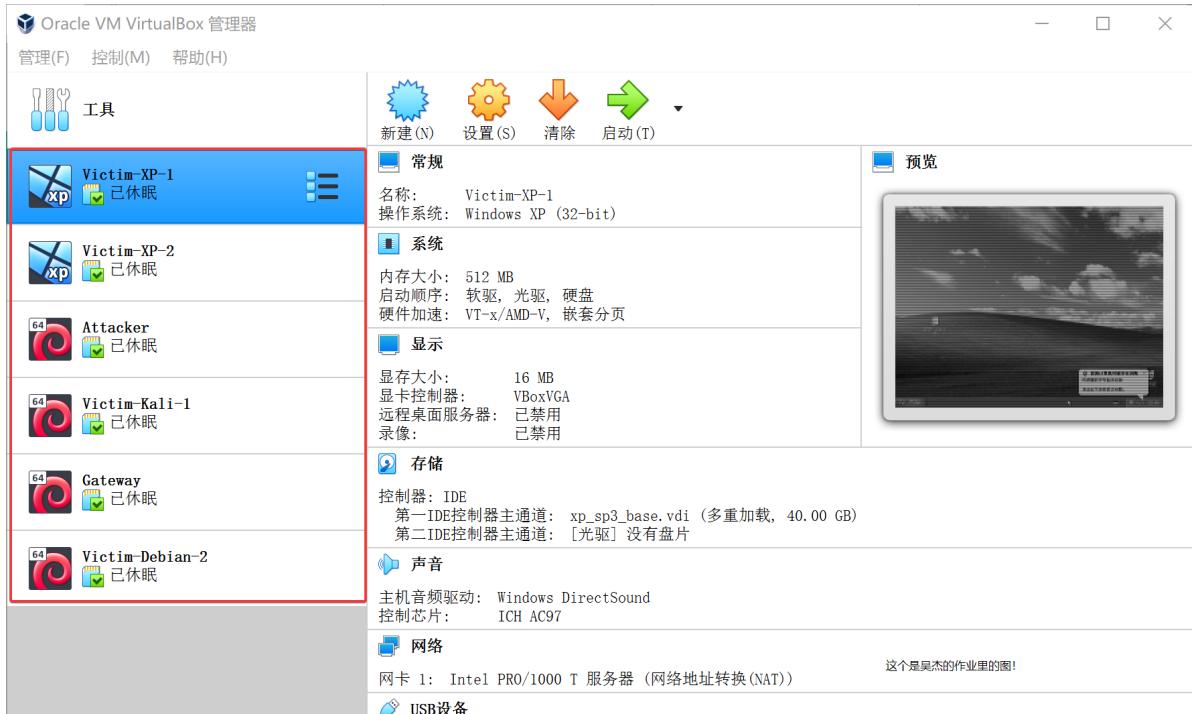


Figure 21: rename_yms.png

6.2 设置内部网络1

网络内有：

- XP1
- Kali1
- 网关

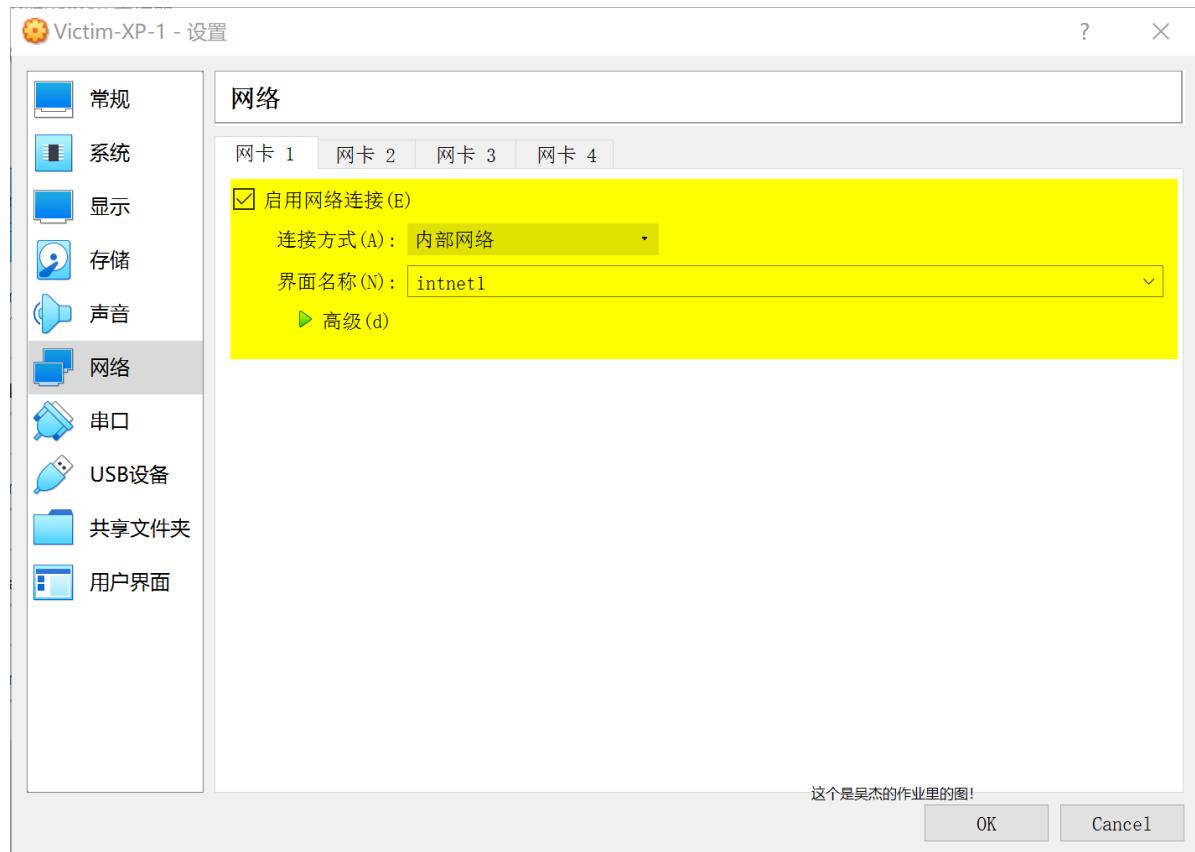


Figure 22: intnet_xp1.png

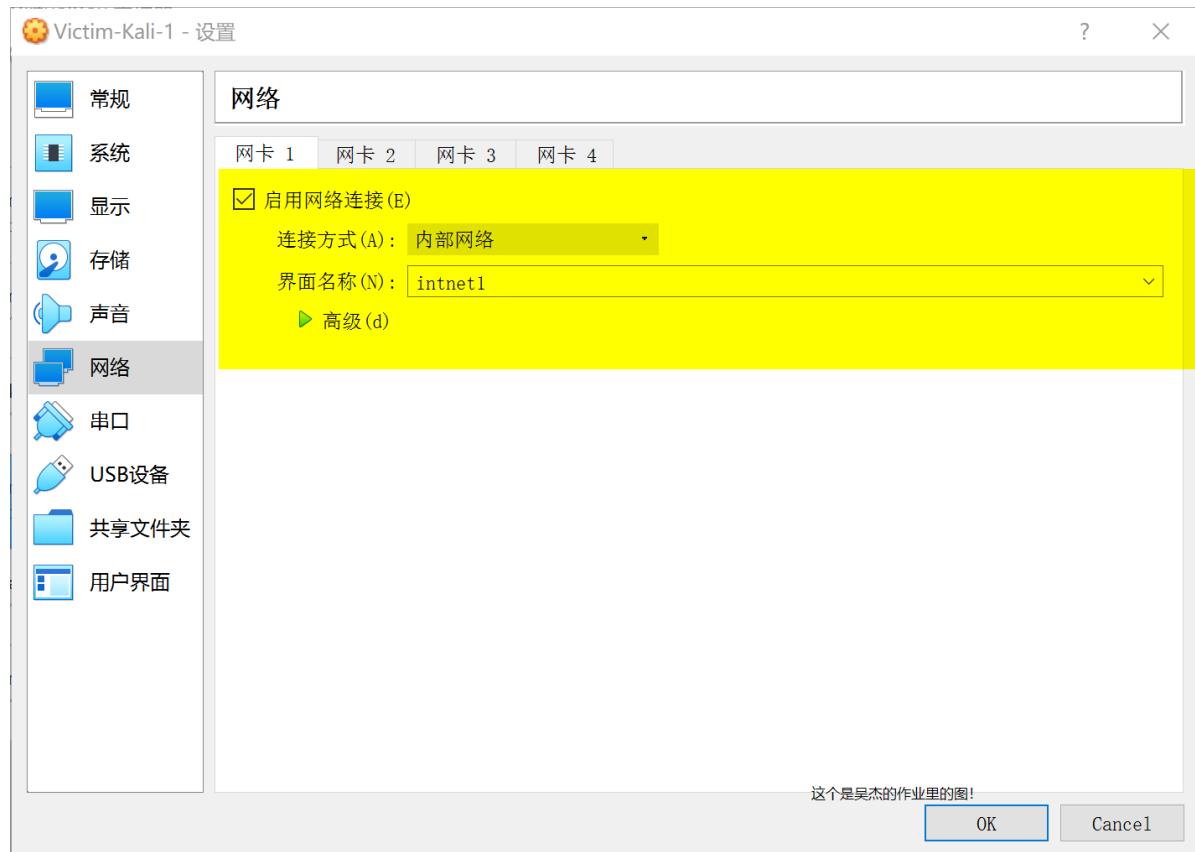


Figure 23: intnet_kali1.png

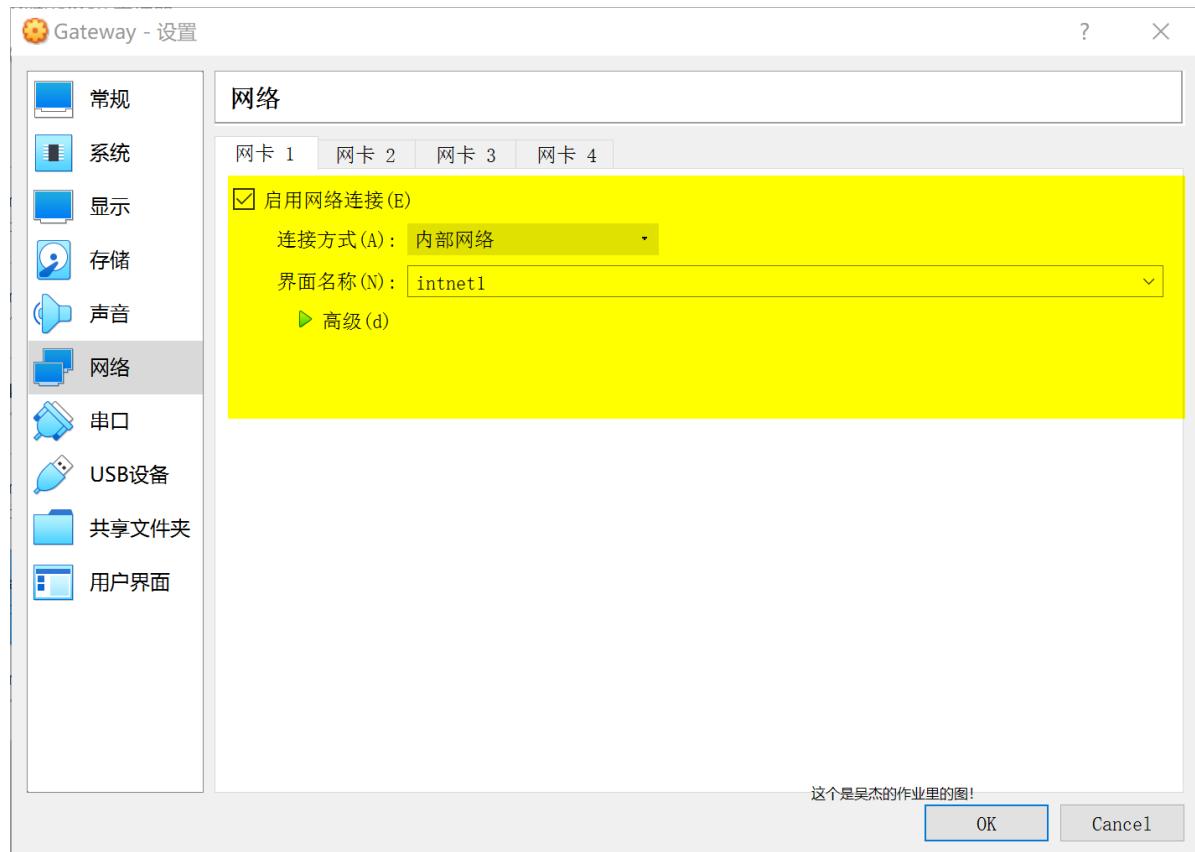


Figure 24: intnet_gw.png

6.3 设置内部网络2

网络内有：

- XP2
- Debian2
- 网关（另一块网卡）

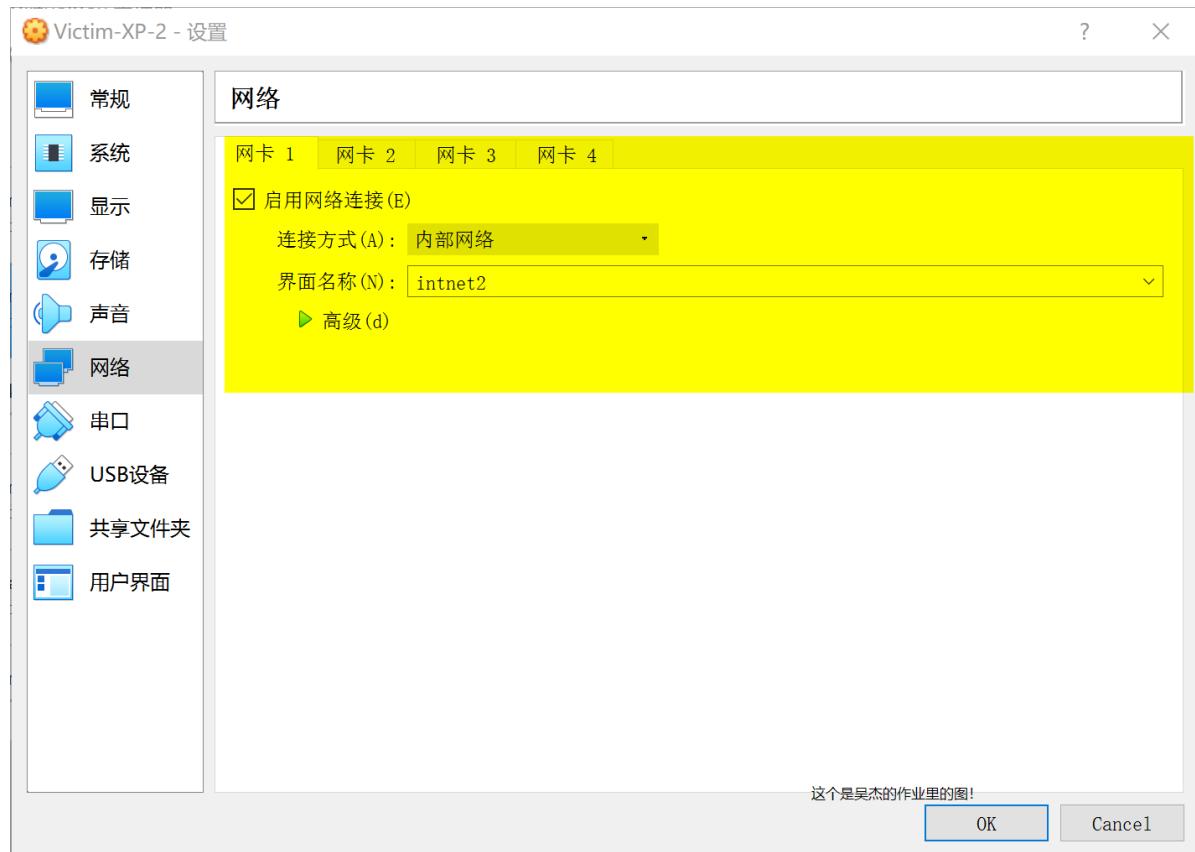


Figure 25: inetnet2_xp2.png

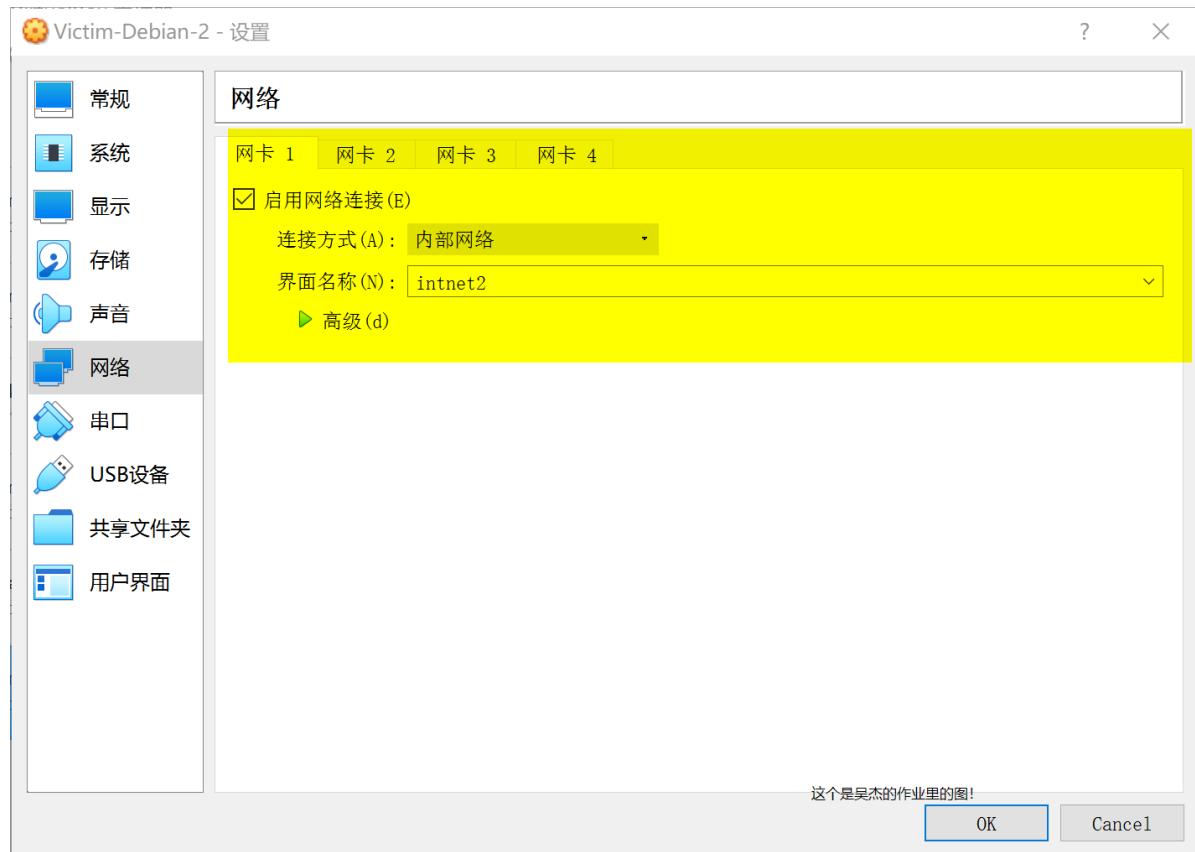


Figure 26: intnet2_debian2.png

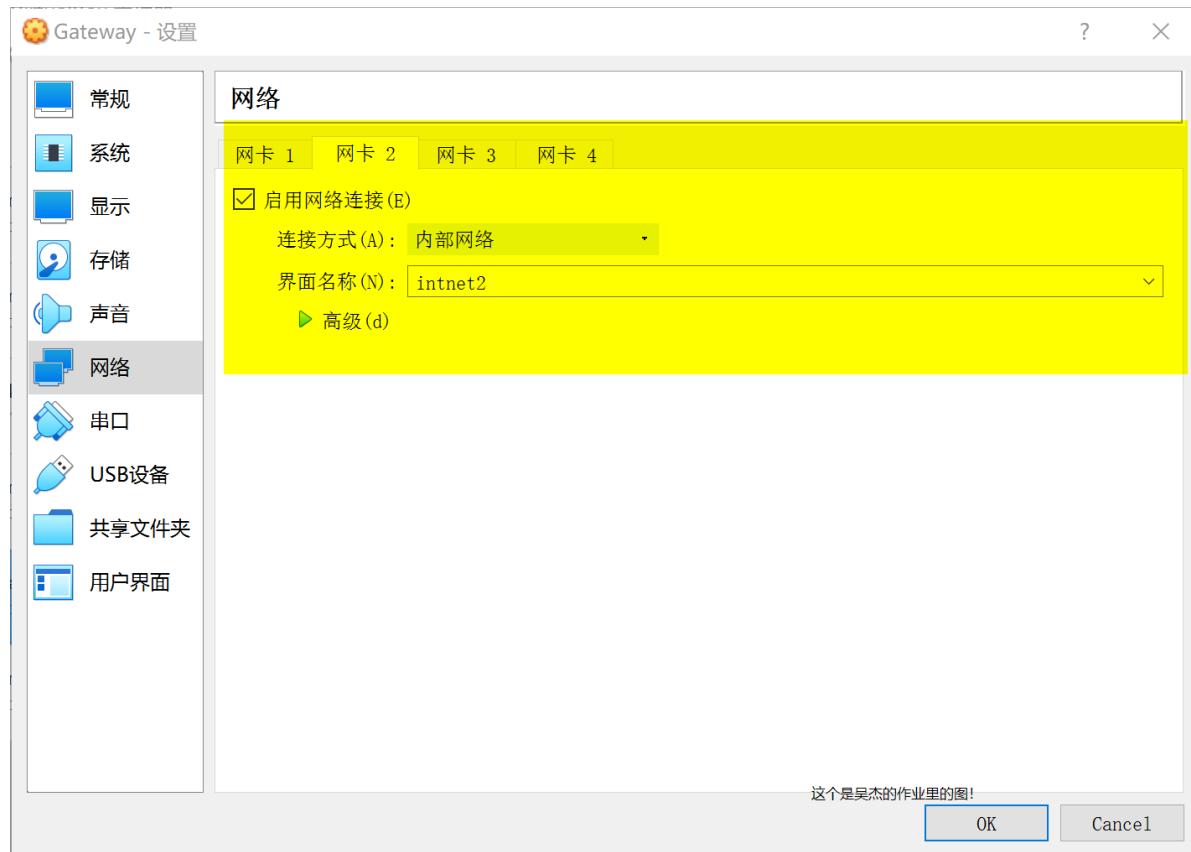


Figure 27: intnet2_gw.png

6.4 设置NatNetwork网络

6.4.1 选用NAT网络的原因

由于要求所有节点均可以访问互联网，因此需要保证网关与攻击者都能上网，内部网络机器通过网关上网。

因此有几种可能的方式：

- 桥接宿主机网卡：不可行，这样网关与攻击者无法互相访问。
- NAT地址转换：可行，但由于NAT地址转换会给每个虚拟机单独配置NAT转换器，网关与攻击者需要再添加一个内部网络才能互相访问，实际在互联网中并不是这样的。
- NAT网络：可行，且共用网关和攻击者共用NAT转换器，可以实现网关与攻击者的互相访问。

6.4.2 创建NAT网络

首先进入全局设定：

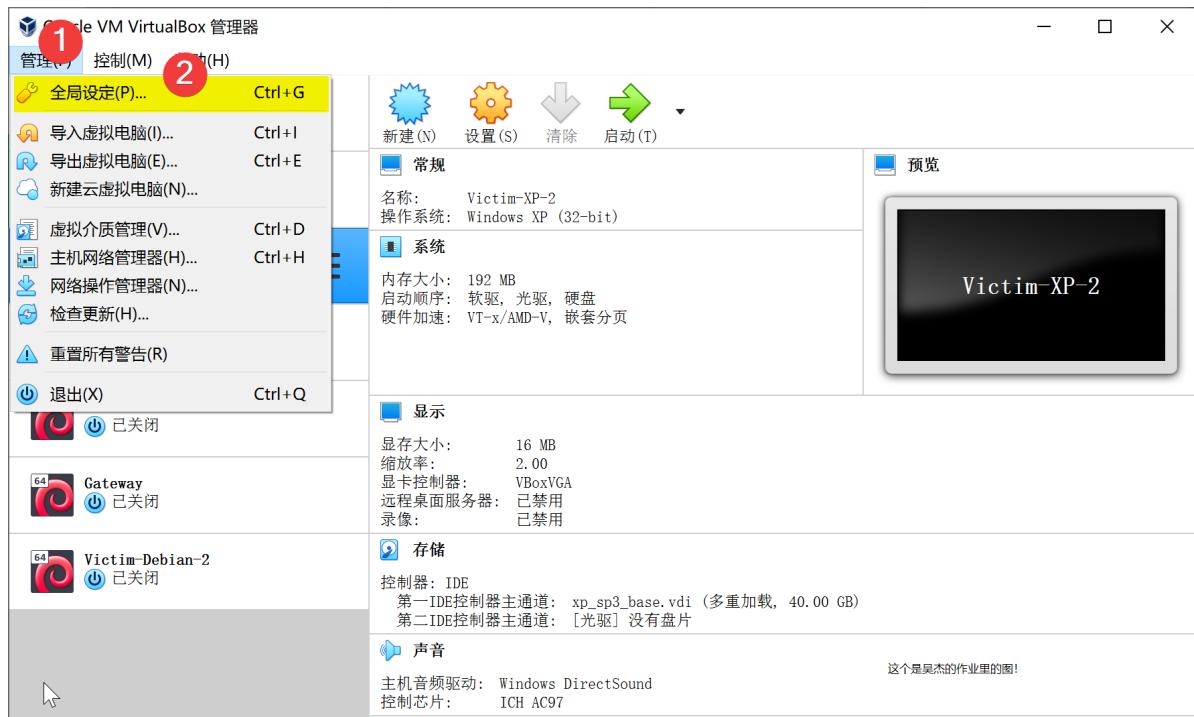


Figure 28: global_settings.png

新建一个NAT网络：

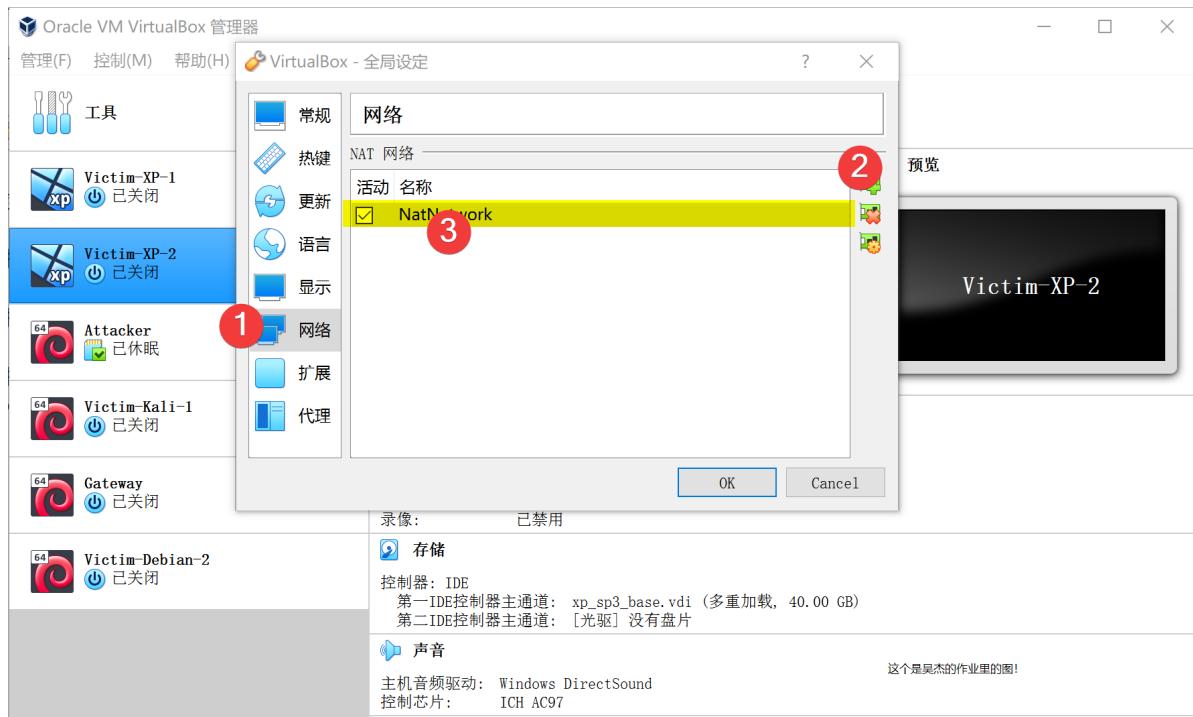


Figure 29: new_natnetwork.png

设定一个子网，不与其他网络重复，同时开启DHCP：

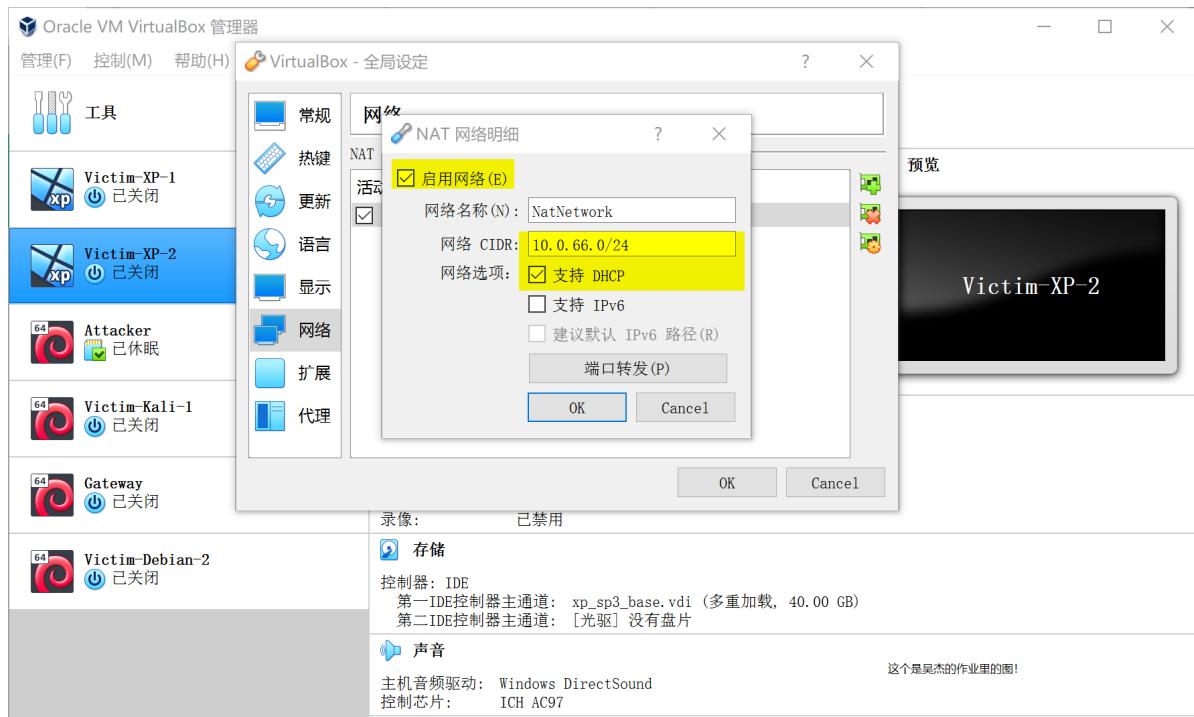


Figure 30: init_natnetwork.png

6.4.3 配置Attacker

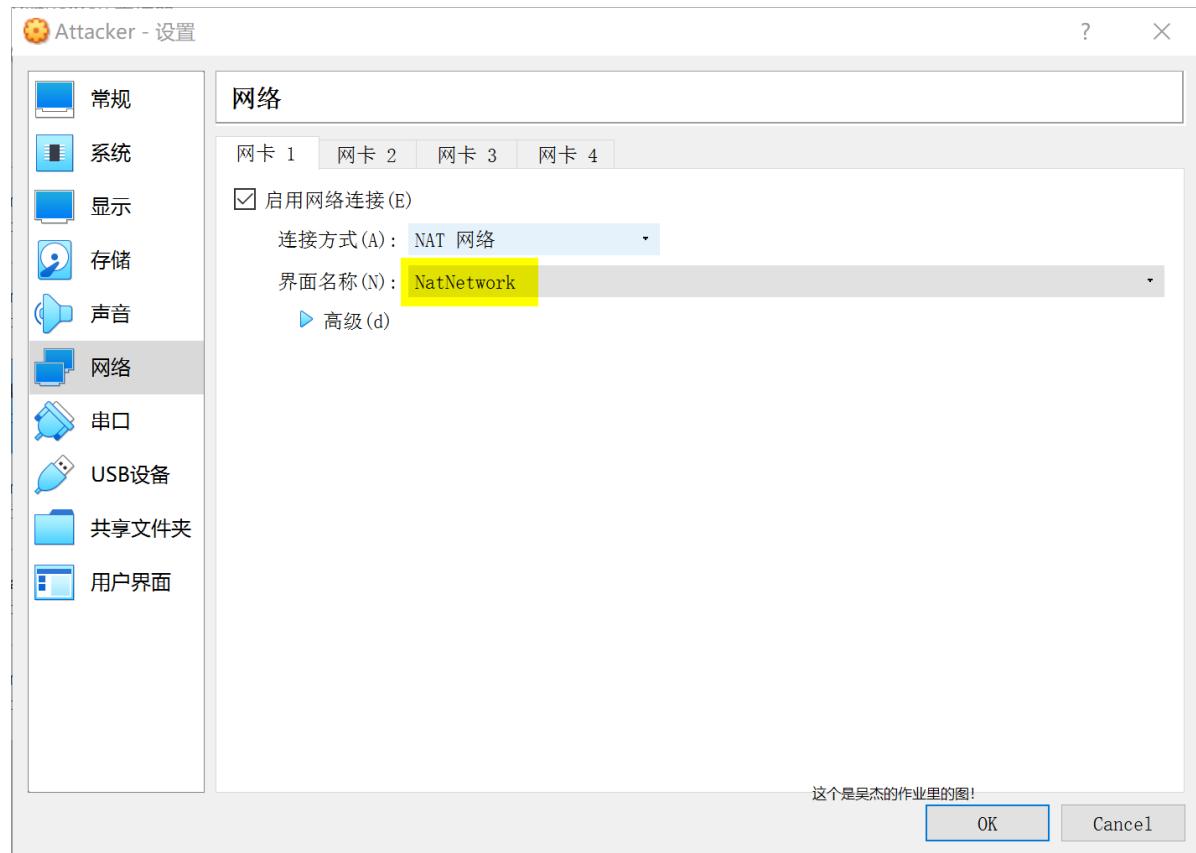


Figure 31: set_attacker_network.png

6.4.4 配置网关

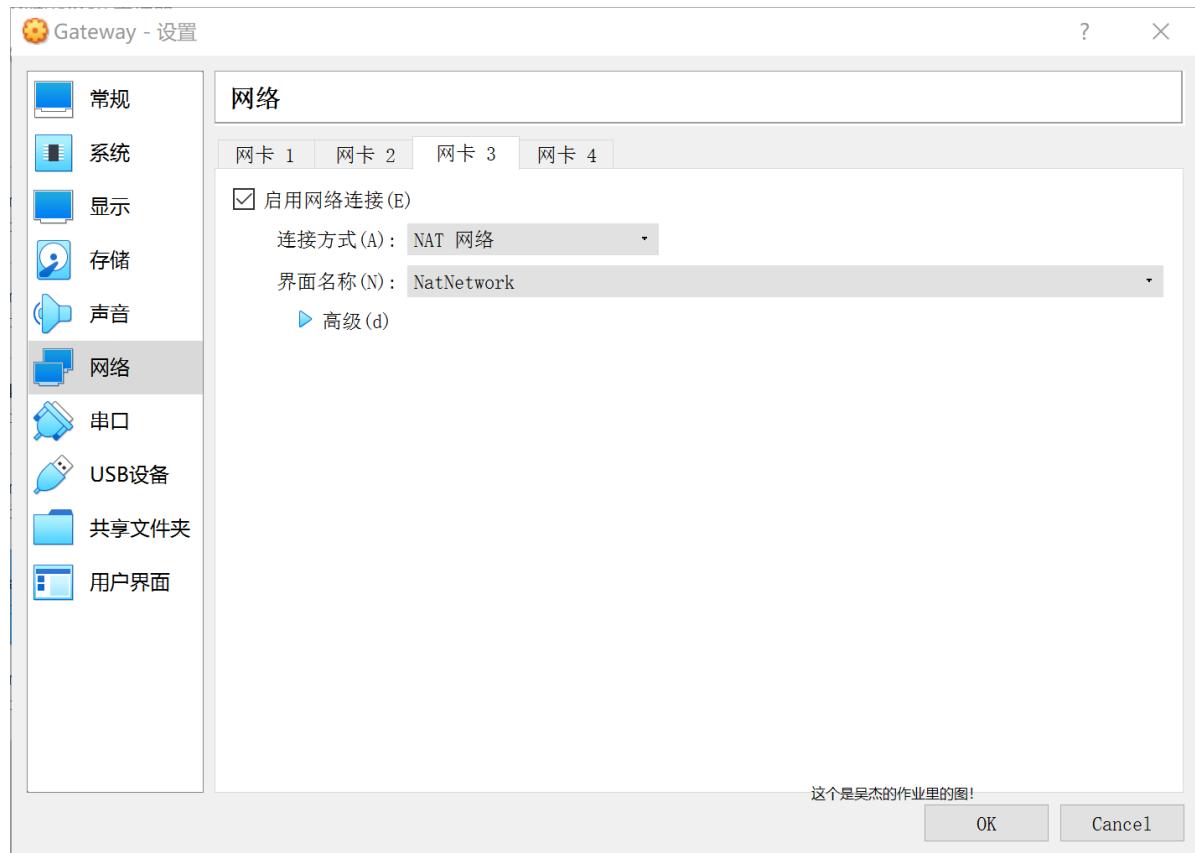


Figure 32: gw_natnetwork.png

6.5 添加DCHP服务（但是似乎不需要）

现在已经创建了内部网络1、内部网络2与NAT网络。

但是**内部网络1、内部网络2**只是简单的虚拟交换机链接，并无DHCP服务器，因此需要添加。

执行以下命令，为**内部网络1、内部网络2**添加DCHP服务：

```
VBoxManage dhcpserver add --netname intnet1 --ip 192.168.59.1 --netmask 255.255.255.0 --lowerip 192.168.59.2 --upperip 192.168.59.100 --enable
```

```
VBoxManage dhcpserver add --netname intnet2 --ip 192.168.60.1 --netmask 255.255.255.0 --lowerip 192.168.60.2 --upperip 192.168.60.100 --enable
```

添加后使用VBoxManage.exe list dhcpservers命令查看结果：

```
D:\Program Files\Oracle\VirtualBox>VBoxManage.exe list dhcpservers
```

```
NetworkName:      HostInterfaceNetworking-VirtualBox Host-Only Ethernet Adapter
Dhcpd IP:        192.168.56.100
LowerIPAddress:  192.168.56.101
UpperIPAddress:  192.168.56.254
NetworkMask:     255.255.255.0
Enabled:         Yes
Global Configuration:
  minLeaseTime:   default
  defaultLeaseTime: default
  maxLeaseTime:   default
  Forced options: None
  Suppressed opts.: None
    1/legacy: 255.255.255.0
Groups:          None
Individual Configs: None

NetworkName:      NatNetwork
Dhcpd IP:        10.0.2.3
LowerIPAddress:  10.0.2.4
UpperIPAddress:  10.0.2.254
NetworkMask:     255.255.255.0
Enabled:         Yes
Global Configuration:
  minLeaseTime:   default
  defaultLeaseTime: default
  maxLeaseTime:   default
  Forced options: None
  Suppressed opts.: None
    1/legacy: 255.255.255.0
    3/legacy: 192.168.41.1
    6/legacy: 172.16.228.9
Groups:          None
Individual Configs: None

NetworkName:      intnet1
```

```
Dhcpd IP:          192.168.59.1
LowerIPAddress: 192.168.59.2
UpperIPAddress: 192.168.59.100
NetworkMask:      255.255.255.0
Enabled:          Yes
Global Configuration:
    minLeaseTime:      default
    defaultLeaseTime:  default
    maxLeaseTime:      default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
Groups:            None
Individual Configs: None

NetworkName:      intnet2
Dhcpd IP:          192.168.60.1
LowerIPAddress: 192.168.60.2
UpperIPAddress: 192.168.60.100
NetworkMask:      255.255.255.0
Enabled:          Yes
Global Configuration:
    minLeaseTime:      default
    defaultLeaseTime:  default
    maxLeaseTime:      default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
Groups:            None
Individual Configs: None
```

可以看到，内部网络已经成功添加DHCP服务器，但Nat网络（NatNetwork）的DCHP服务器其实是Vbox的NAT模式自带的那个DHCP服务器，这样重名可能会有一些问题，所以修改一下自己新建的服务器名称和IP：

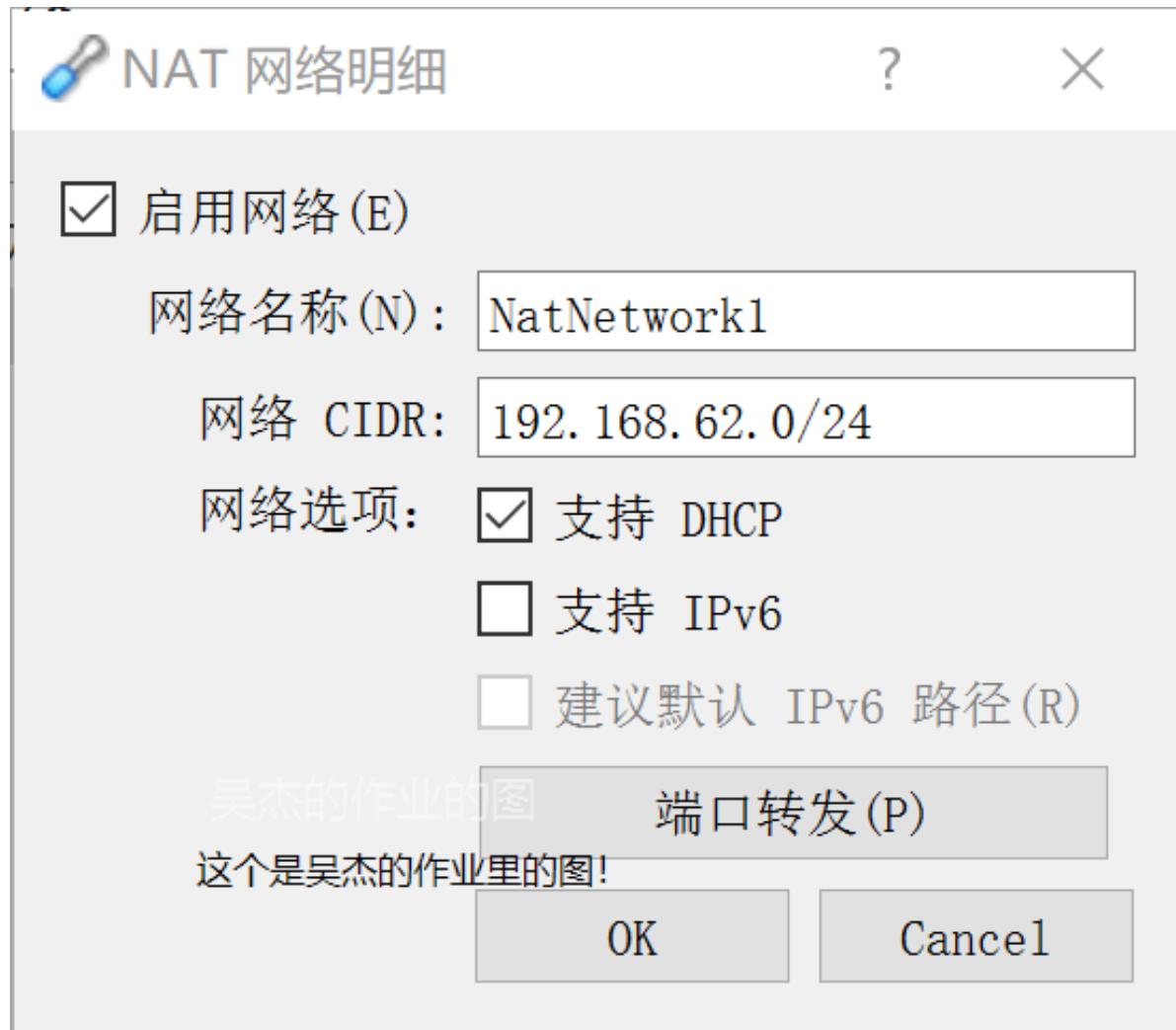


Figure 33: modify_natnetwork.png

随后运行命令，添加DHCP服务器：

```
VBoxManage dhcpserver add --netname NatNetwork1 --ip 192.168.61.1 --  
netmask 255.255.255.0 --lowerip 192.168.61.2 --upperip 192.168.61.100 --enable
```

添加结果：

```
NetworkName:      NatNetwork1  
Dhcpd IP:        192.168.62.3  
LowerIPAddress:  192.168.62.4  
UpperIPAddress:  192.168.62.254  
NetworkMask:     255.255.255.0
```

```

Enabled: Yes
Global Configuration:
  minLeaseTime: default
  defaultLeaseTime: default
  maxLeaseTime: default
Forced options: None
Suppressed opts.: None
  1/legacy: 255.255.255.0
  3/legacy: 192.168.62.1
  6/legacy: 172.16.228.9
Groups: None
Individual Configs: None

```

但是进系统之后发现网卡3、网卡4的IP是无法DHCP获取的，估计是网关系统里写成静态的了，所以调换一下顺序，网卡3和网卡4分别给内部网络1和内部网络2，这样NAT网卡就能正确DHCP、上网了。

如下图：



Figure 34: modify_gw_interface.png

随后把之前添加的DHCP删除：

```
D:\Program Files\Oracle\VirtualBox>VBoxManage.exe dhcpserver remove --network=intnet1  
D:\Program Files\Oracle\VirtualBox>VBoxManage.exe dhcpserver remove --network=intnet2
```

Figure 35: remove_dhcp_servers.png

完成后进入XP1：

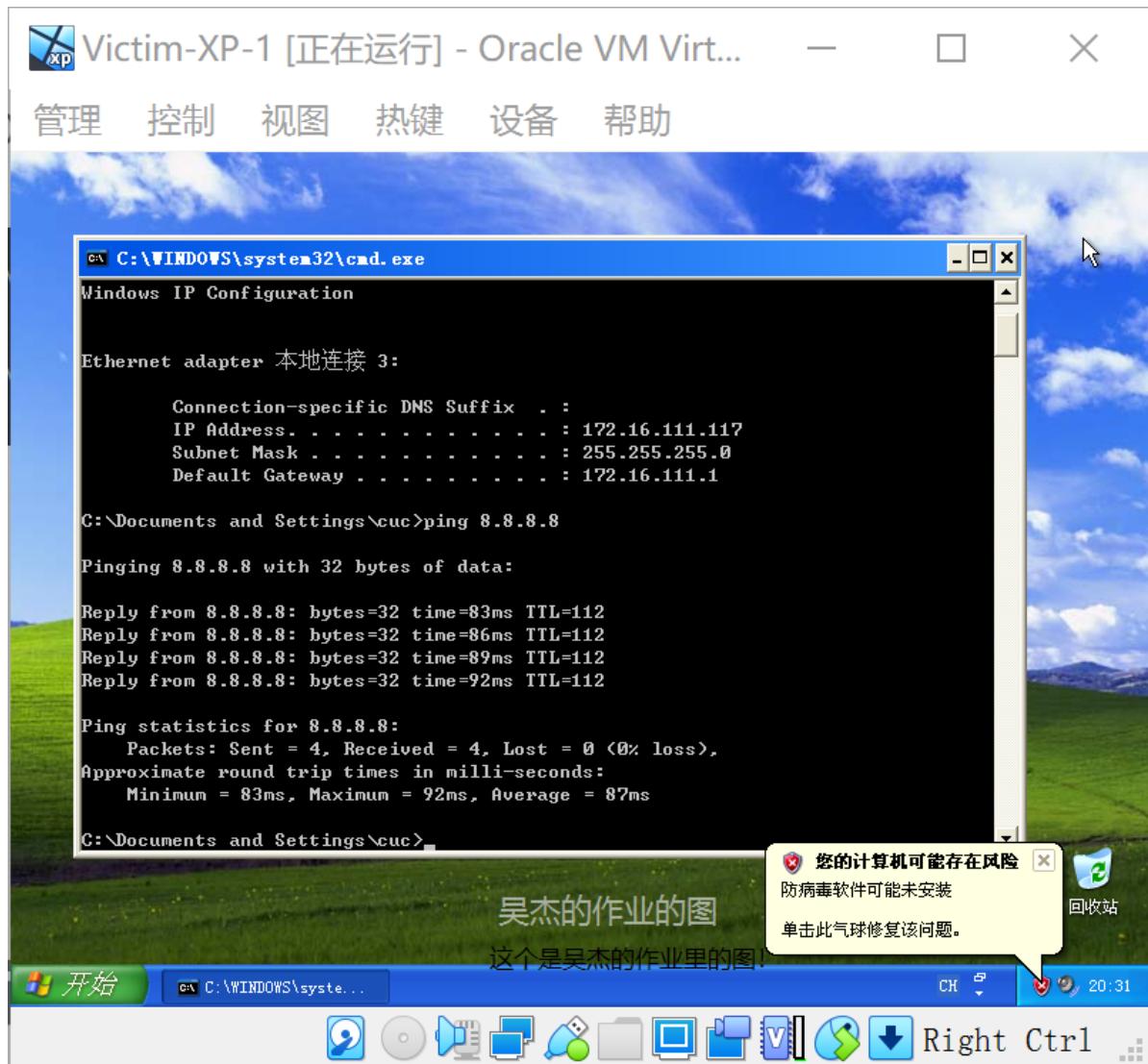


Figure 36: xp1_cmd_dhcp.png

可以看到已经获取到ip，并且也可以上网了，所以应该是这个网关是有dhcpd的。

7 网络测试

7.1 靶机可以直接访问攻击者主机

7.1.1 XP1

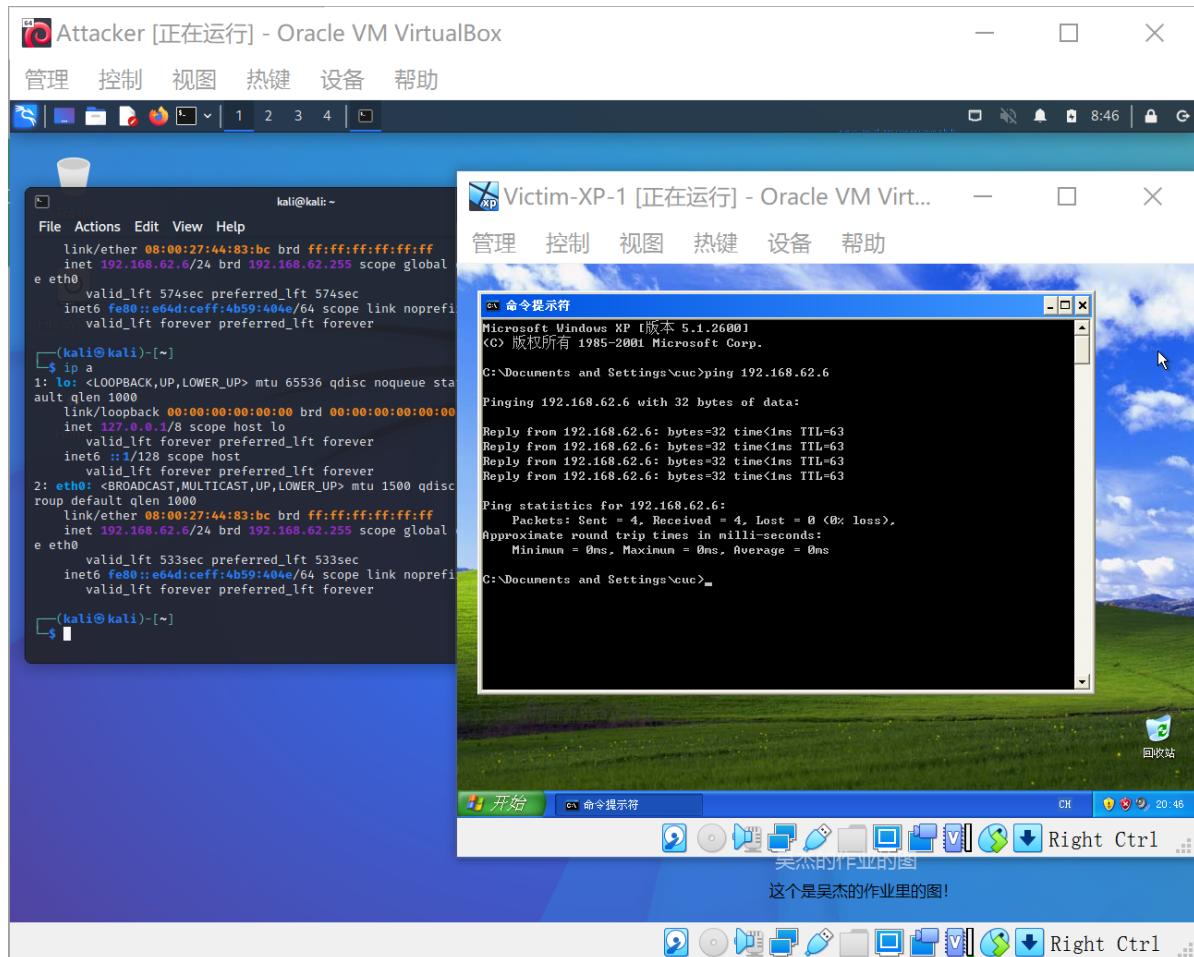


Figure 37: xp1_ping_attacker.png

7.1.2 Kali1

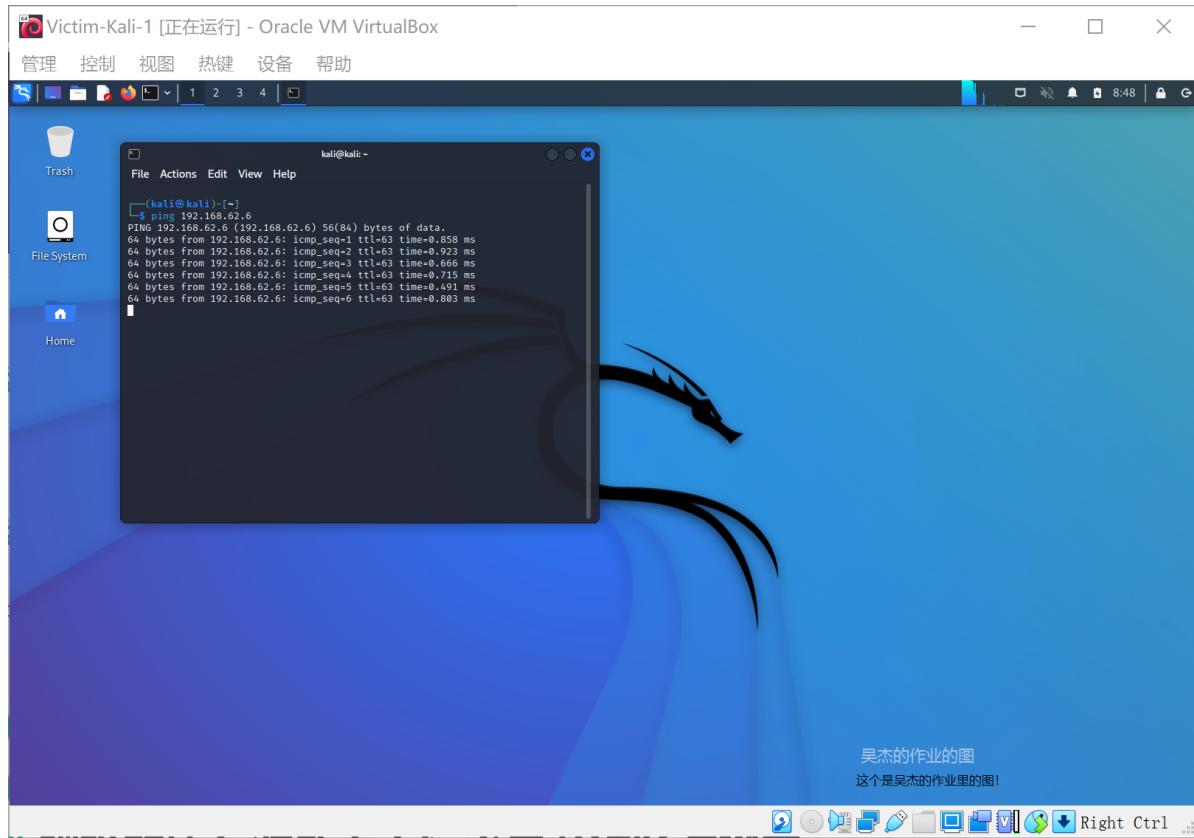


Figure 38: kali1_ping_attacker.png

7.1.3 XP2

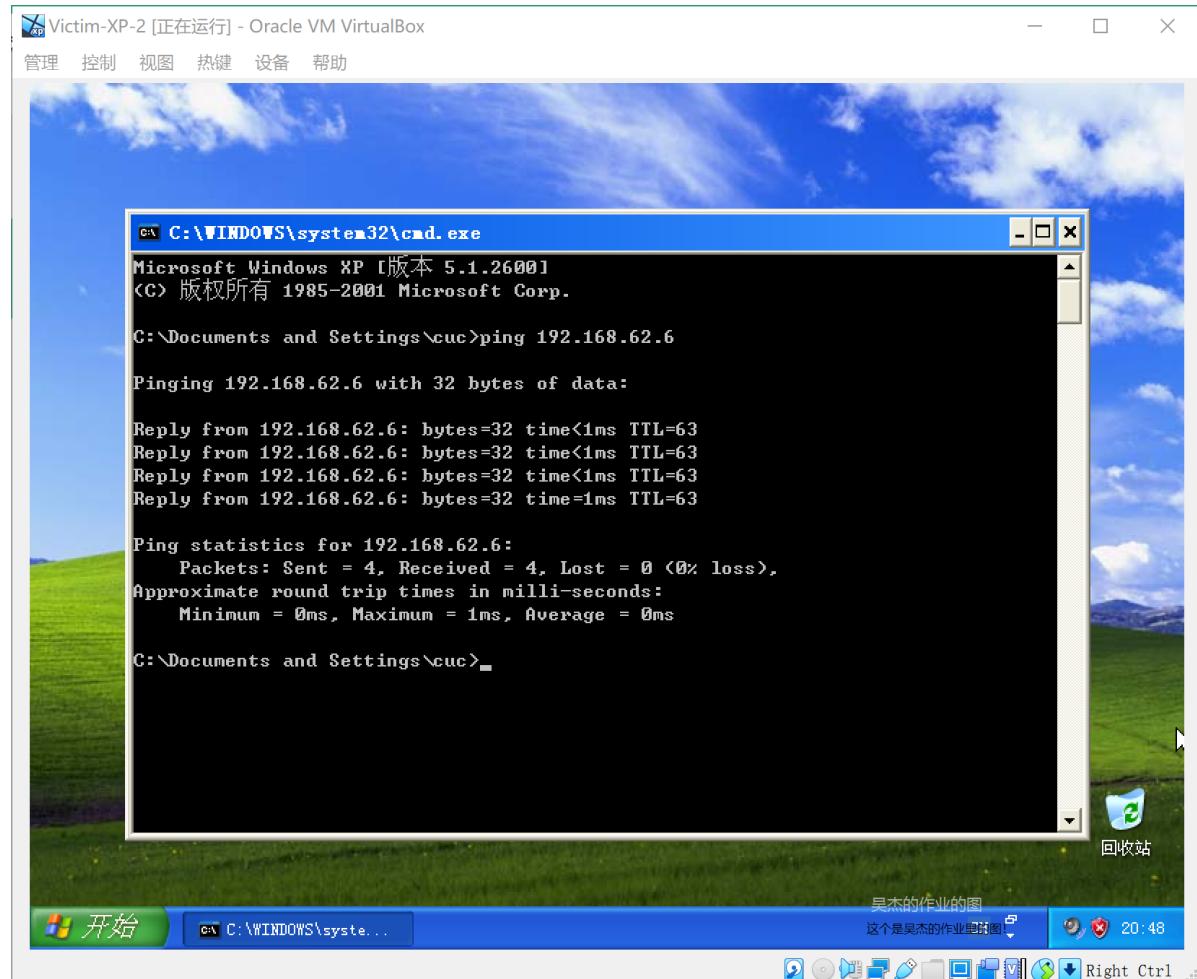


Figure 39: xp2_ping_attacker.png

7.1.4 Debian2

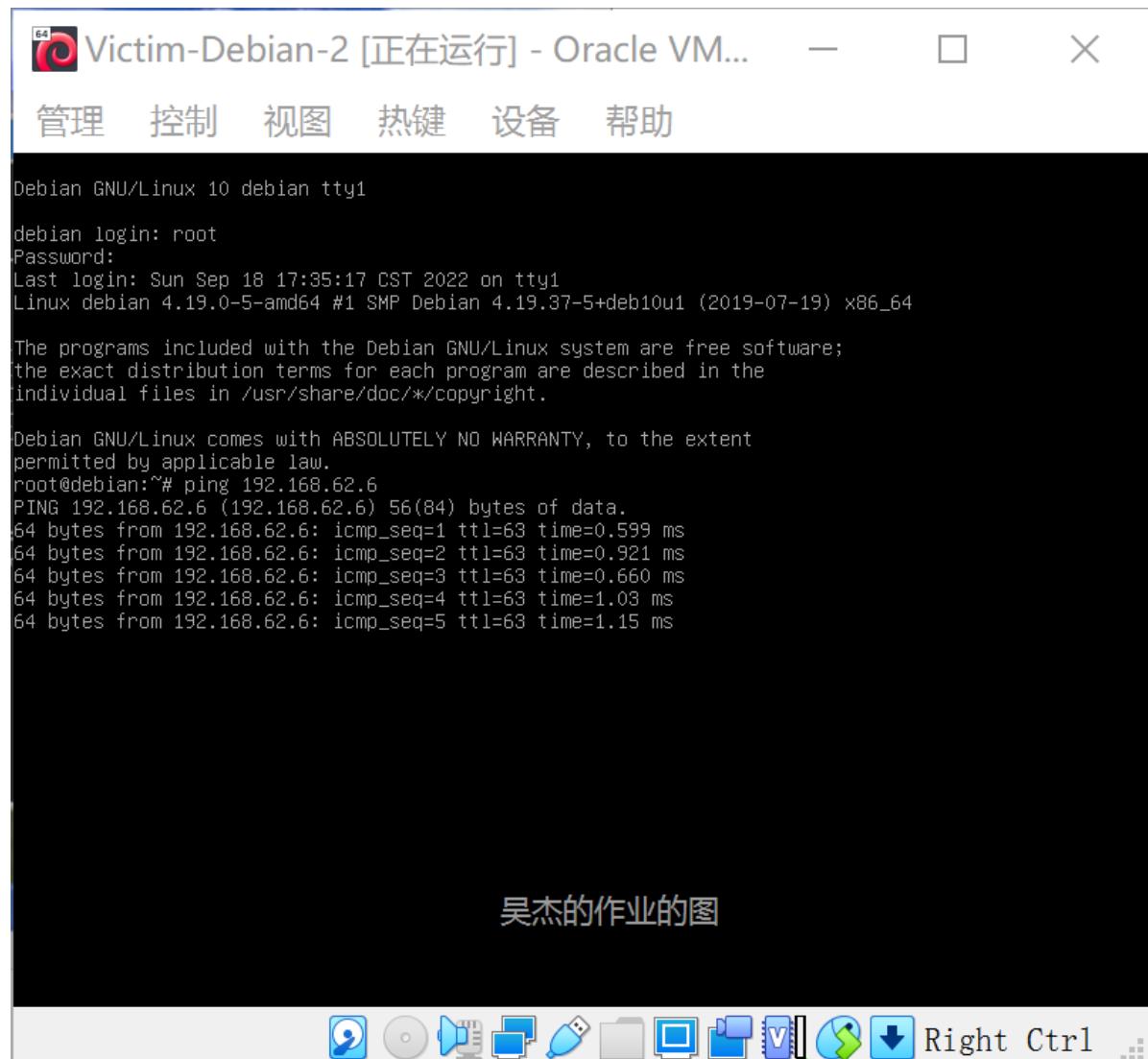


Figure 40: debian2_ping_attacker.png

7.2 攻击者主机无法直接访问靶机

7.2.1 XP1

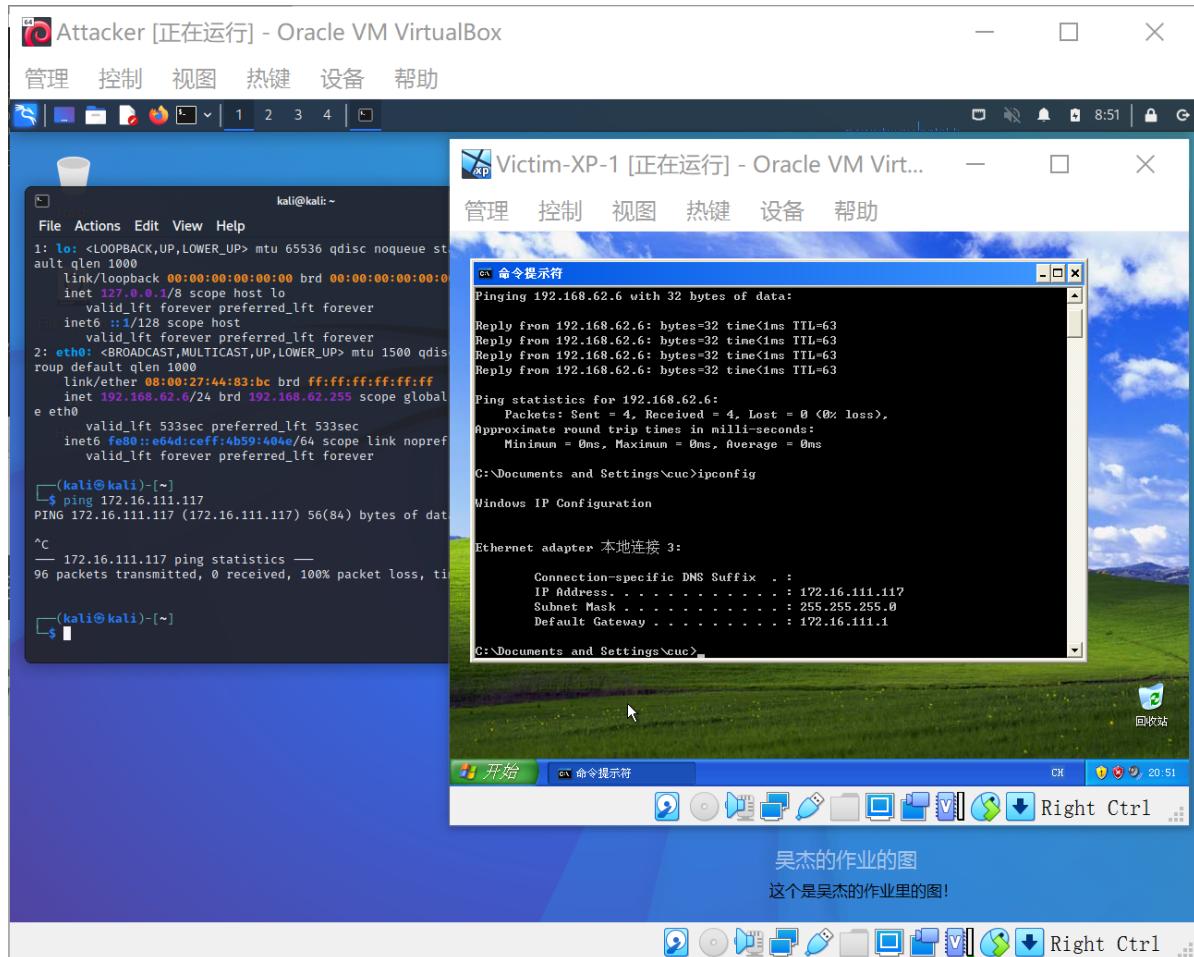


Figure 41: attacker_ping_xp1.png

7.2.2 XP2

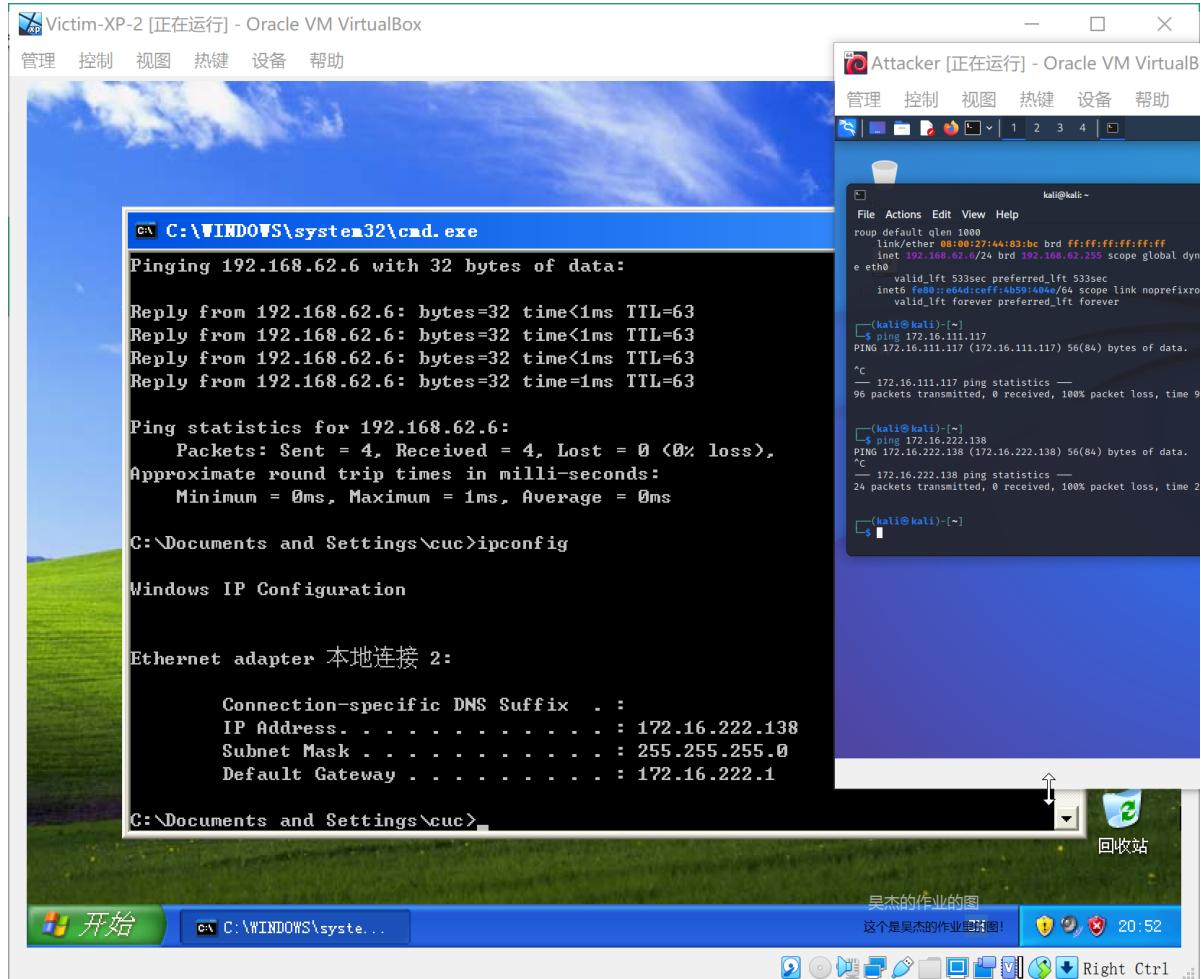


Figure 42: attacker_ping_xp2.png

7.3 网关可以直接访问攻击者主机和靶机

7.3.1 攻击者主机

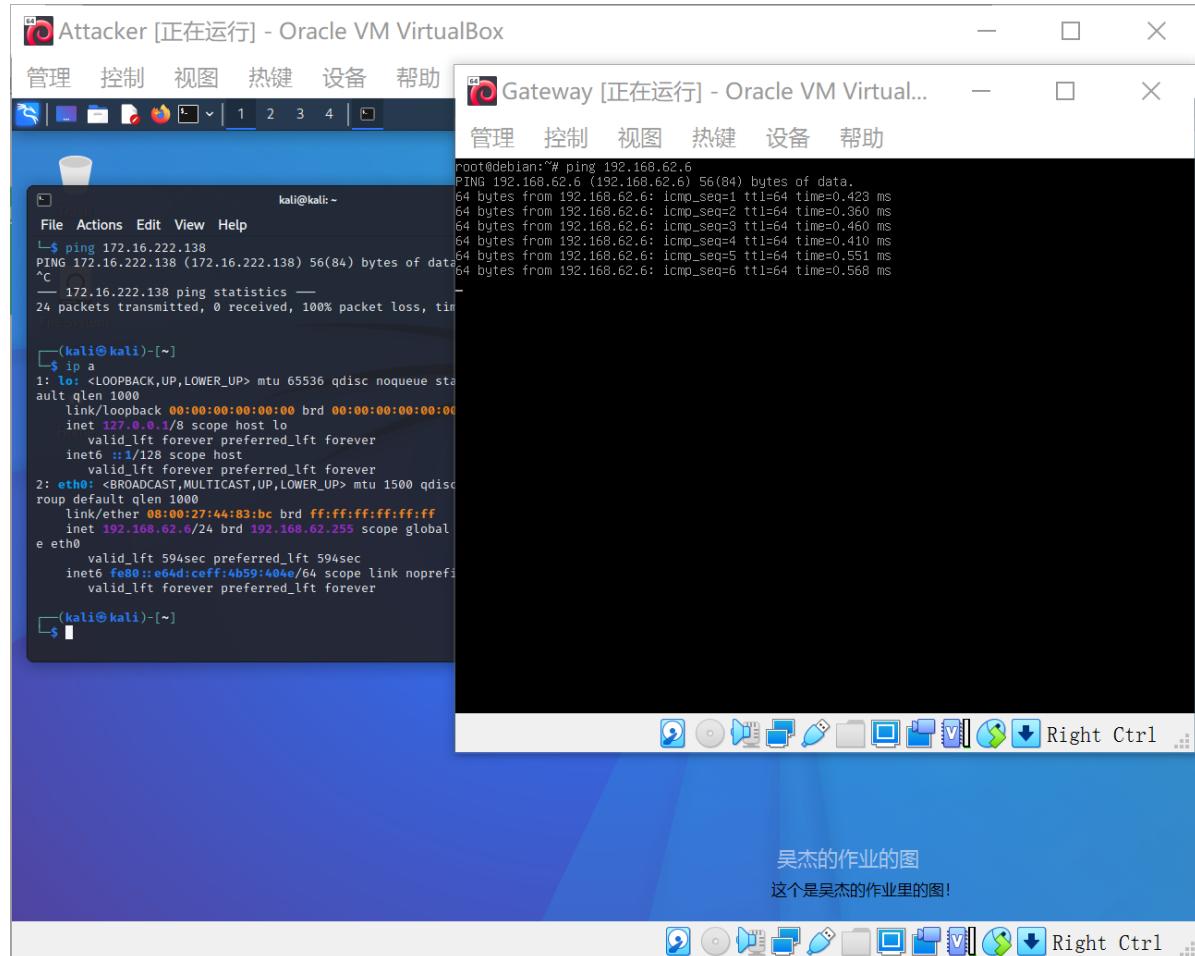


Figure 43: gw_ping_attacker.png

7.3.2 XP1

需要先关闭XP防火墙才能PING:

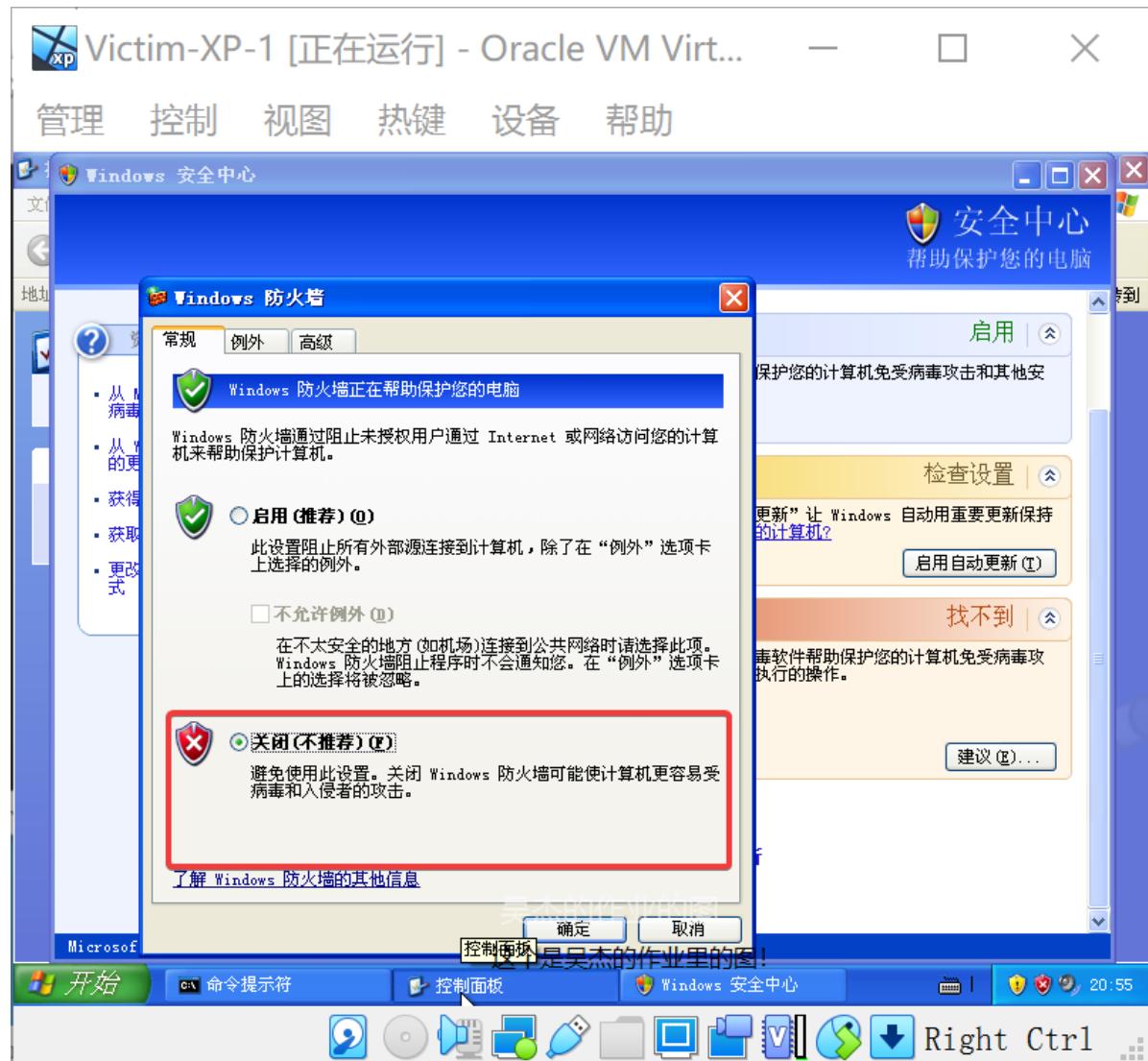


Figure 44: disable_xp1_firewall.png

```
64 bytes from 192.168.62.6: icmp_seq=17 ttl=64 time=0.568 ms
64 bytes from 192.168.62.6: icmp_seq=18 ttl=64 time=0.306 ms
64 bytes from 192.168.62.6: icmp_seq=19 ttl=64 time=0.449 ms
64 bytes from 192.168.62.6: icmp_seq=20 ttl=64 time=0.694 ms
64 bytes from 192.168.62.6: icmp_seq=21 ttl=64 time=0.506 ms
64 bytes from 192.168.62.6: icmp_seq=22 ttl=64 time=0.321 ms
64 bytes from 192.168.62.6: icmp_seq=23 ttl=64 time=0.571 ms
64 bytes from 192.168.62.6: icmp_seq=24 ttl=64 time=0.427 ms
64 bytes from 192.168.62.6: icmp_seq=25 ttl=64 time=0.304 ms
64 bytes from 192.168.62.6: icmp_seq=26 ttl=64 time=0.418 ms
64 bytes from 192.168.62.6: icmp_seq=27 ttl=64 time=0.357 ms
64 bytes from 192.168.62.6: icmp_seq=28 ttl=64 time=0.461 ms
64 bytes from 192.168.62.6: icmp_seq=29 ttl=64 time=0.431 ms
64 bytes from 192.168.62.6: icmp_seq=30 ttl=64 time=0.439 ms
64 bytes from 192.168.62.6: icmp_seq=31 ttl=64 time=0.290 ms
64 bytes from 192.168.62.6: icmp_seq=32 ttl=64 time=0.588 ms
64 bytes from 192.168.62.6: icmp_seq=33 ttl=64 time=0.376 ms
^D
--- 192.168.62.6 ping statistics ---
33 packets transmitted, 33 received, 0% packet loss, time 802ms
rtt min/avg/max/mdev = 0.281/0.436/0.694/0.101 ms
root@debian:~# ^C
root@debian:~# ^C
root@debian:~# ping 172.16.111.117
PING 172.16.111.117 (172.16.111.117) 56(84) bytes of data.
^C
--- 172.16.111.117 ping statistics ---
71 packets transmitted, 0 received, 100% packet loss, time 750ms

root@debian:~# ping 172.16.111.117
PING 172.16.111.117 (172.16.111.117) 56(84) bytes of data.
64 bytes from 172.16.111.117: icmp_seq=6 ttl=128 time=0.528 ms
64 bytes from 172.16.111.117: icmp_seq=7 ttl=128 time=0.662 ms
64 bytes from 172.16.111.117: icmp_seq=8 ttl=128 time=0.662 ms
64 bytes from 172.16.111.117: icmp_seq=9 ttl=128 time=0.389 ms
64 bytes from 172.16.111.117: icmp_seq=10 ttl=128 time=0.764 ms
```

Figure 45: gw_ping_xp1.png

7.4 靶机的所有对外上下行流量必须经过网关

以XP1为例子，尝试ping baidu.com：

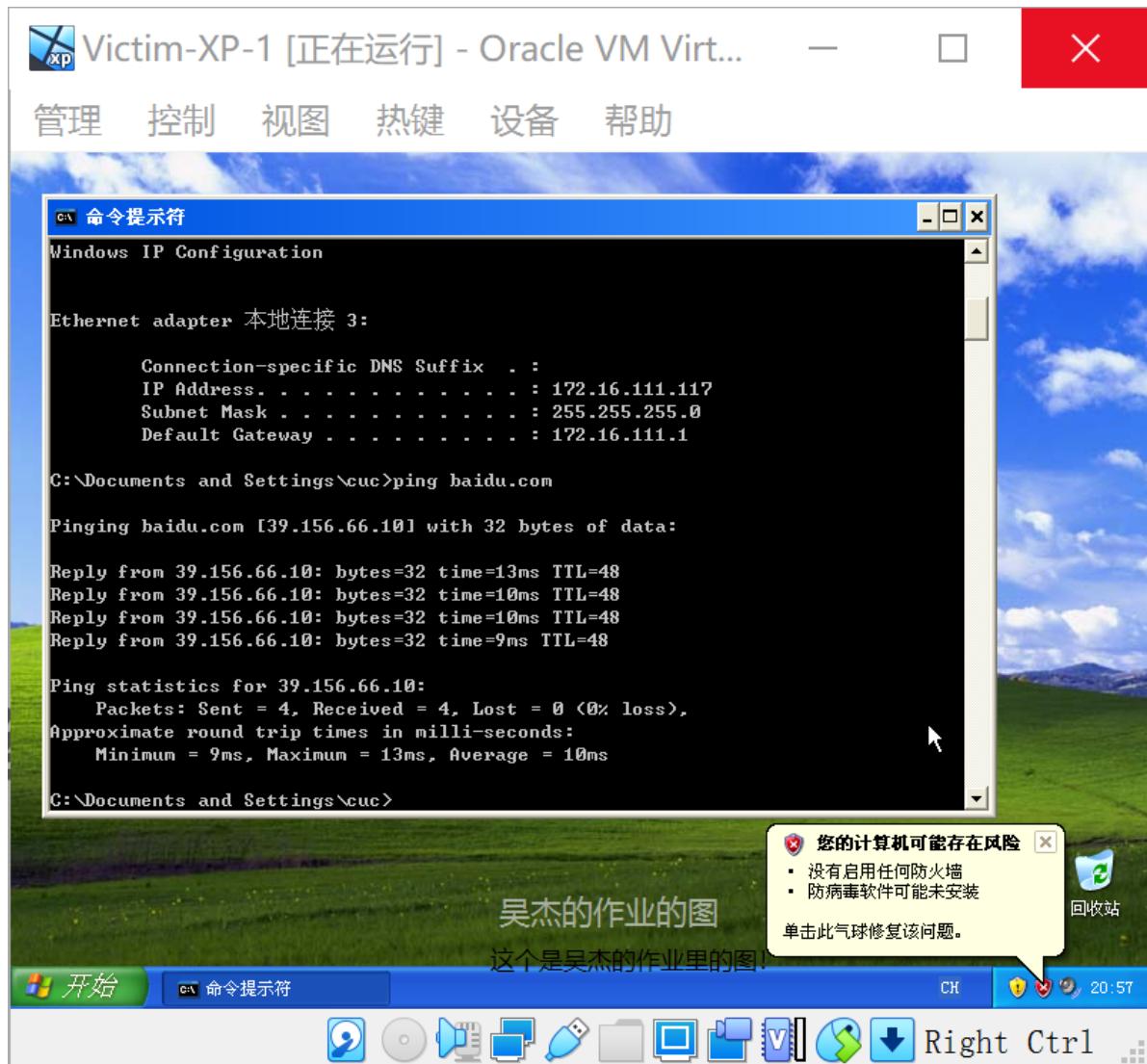


Figure 46: xp1_ping_baidu_with_gw.png

在网关上进行抓包：

64 Gateway [正在运行] - Oracle VM Virtual... — □ ×

管理 控制 视图 热键 设备 帮助

```
21:00:27.165442 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 717, length 64
21:00:28.188437 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 718, length 64
21:00:28.188893 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 718, length 64
21:00:29.189338 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 719, length 64
21:00:29.189814 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 719, length 64
21:00:30.190092 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 720, length 64
21:00:30.190370 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 720, length 64
21:00:31.194839 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 721, length 64
21:00:31.195244 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 721, length 64
21:00:32.194403 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 722, length 64
21:00:32.194653 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 722, length 64
21:00:33.209521 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 723, length 64
21:00:33.209820 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 723, length 64
21:00:34.232953 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 724, length 64
21:00:34.233168 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 724, length 64
21:00:35.256476 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 725, length 64
21:00:35.256739 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 725, length 64
21:00:36.280143 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 726, length 64
21:00:36.280525 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 726, length 64
21:00:37.303481 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 727, length 64
21:00:37.303769 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 727, length 64
21:00:38.315061 ARP, Request who-has 192.168.62.6 tell 192.168.62.5, length 28
21:00:38.315339 ARP, Reply 192.168.62.6 is-at 08:00:27:44:83:bc, length 46
21:00:38.326915 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 728, length 64
21:00:38.327176 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 728, length 64
21:00:39.350902 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 729, length 64
21:00:39.351241 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 729, length 64
^C
128 packets captured
128 packets received by filter
0 packets dropped by kernel
root@debian:~# ^C
root@debian:~# ^C
root@debian:~# tcpdump -i enp0s3 -n          吴杰的作业的图
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Figure 47: tcpdump_on.png

随后再次进行ping，可以看到tcpdump的结果：

64 Gateway [正在运行] - Oracle VM Virtual... — □ ×

管理 控制 视图 热键 设备 帮助

```
21:00:37.303769 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 727, length 64
21:00:38.315061 ARP, Request who-has 192.168.62.6 tell 192.168.62.5, length 28
21:00:38.315339 ARP, Reply 192.168.62.6 is-at 08:00:27:44:83:bc, length 46
21:00:38.326915 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 728, length 64
21:00:38.327176 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 728, length 64
21:00:39.350902 IP 192.168.62.5 > 192.168.62.6: ICMP echo request, id 59808, seq 729, length 64
21:00:39.351241 IP 192.168.62.6 > 192.168.62.5: ICMP echo reply, id 59808, seq 729, length 64
^C
128 packets captured
128 packets received by filter
0 packets dropped by kernel
root@debian:~# ^C
root@debian:~# ^C
root@debian:~# tcpdump -i enp0s3 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
Listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
21:01:25.802655 IP 192.168.62.5.57533 > 162.159.200.123.123: NTPv4, Client, length 48
21:01:25.975298 IP 162.159.200.123.123 > 192.168.62.5.57533: NTPv4, Server, length 48
21:01:44.794930 IP 192.168.62.5.2758 > 172.16.228.9.53: 28651+ A? baidu.com. (27)
21:01:44.798300 IP 172.16.228.9.53 > 192.168.62.5.2758: 28651 2/0/0 A 110.242.68.66, A 39.156.66.10
(59)
21:01:44.800373 IP 192.168.62.5 > 110.242.68.66: ICMP echo request, id 512, seq 6400, length 40
21:01:44.812804 IP 110.242.68.66 > 192.168.62.5: ICMP echo reply, id 512, seq 6400, length 40
21:01:45.798676 IP 192.168.62.5 > 110.242.68.66: ICMP echo request, id 512, seq 6656, length 40
21:01:45.811416 IP 110.242.68.66 > 192.168.62.5: ICMP echo reply, id 512, seq 6656, length 40
21:01:46.800404 IP 192.168.62.5 > 110.242.68.66: ICMP echo request, id 512, seq 6912, length 40
21:01:46.813205 IP 110.242.68.66 > 192.168.62.5: ICMP echo reply, id 512, seq 6912, length 40
21:01:47.802627 IP 192.168.62.5 > 110.242.68.66: ICMP echo request, id 512, seq 7168, length 40
21:01:47.816224 IP 110.242.68.66 > 192.168.62.5: ICMP echo reply, id 512, seq 7168, length 40
21:01:49.821678 ARP, Request who-has 192.168.62.1 tell 192.168.62.5, length 28
21:01:49.821794 ARP, Reply 192.168.62.1 is-at 08:00:27:f7:9b:1e, length 46
21:02:02.897849 IP 192.168.62.5.68 > 192.168.62.3.67: BOOTP/DHCP, Request from 08:00:27:f7:9b:1e, length 300
21:02:02.902386 IP 192.168.62.3.67 > 192.168.62.5.68: BOOTP/DHCP, Reply, length 548
21:02:07.998612 ARP, Request who-has 192.168.62.3 tell 192.168.62.5, length 28
21:02:07.999797 ARP, Reply 192.168.62.3 is-at 08:00:27:f7:9b:1e, length 46
```

是杰的作业的图

Figure 48: tcpdump_result.png

与ping baidu.com的包一致，说明经过网关。

7.5 所有节点均可以访问互联网

7.5.1 攻击者

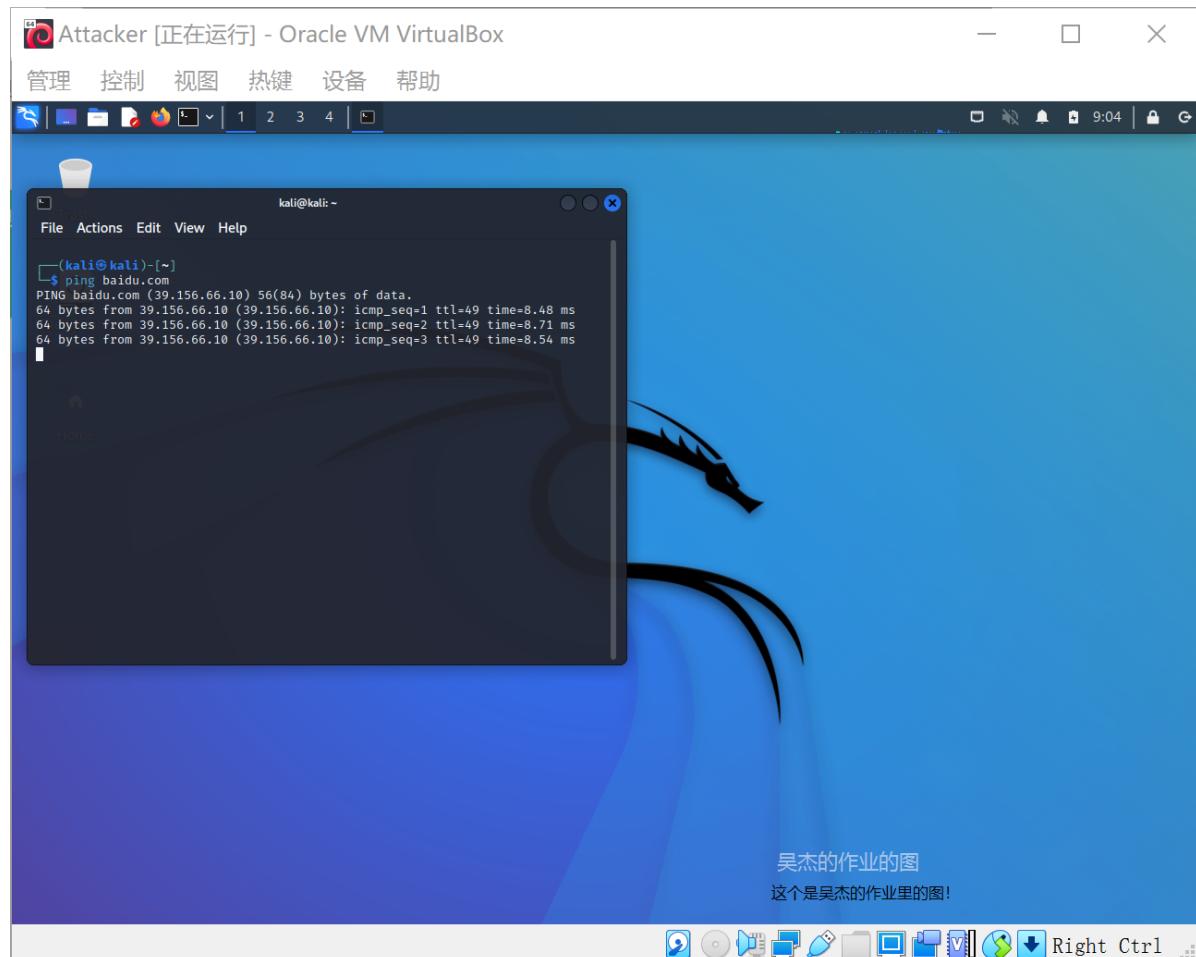


Figure 49: attacker_baidu.png

7.5.2 网关

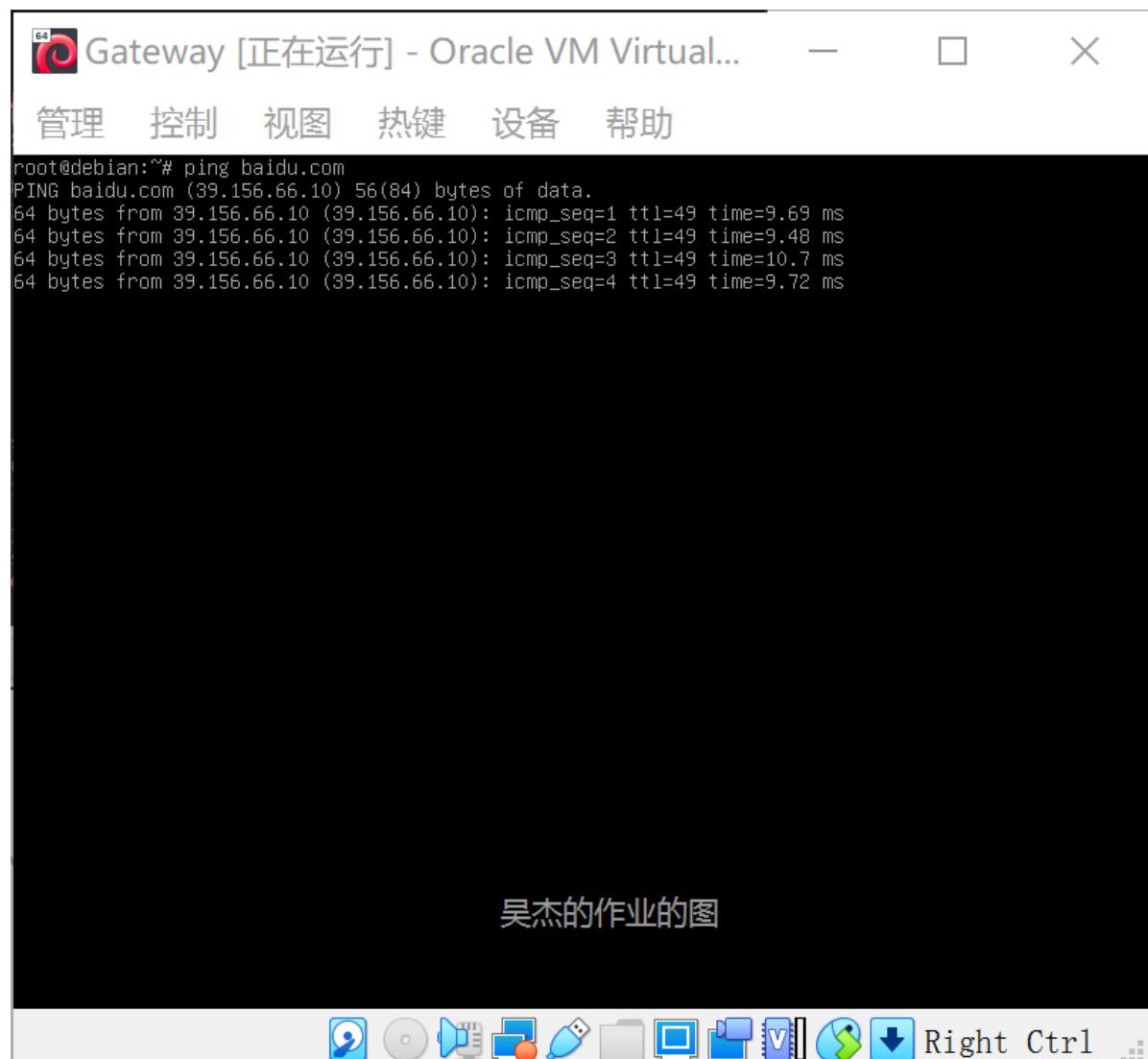


Figure 50: gw_baidu.png

7.5.3 XP1

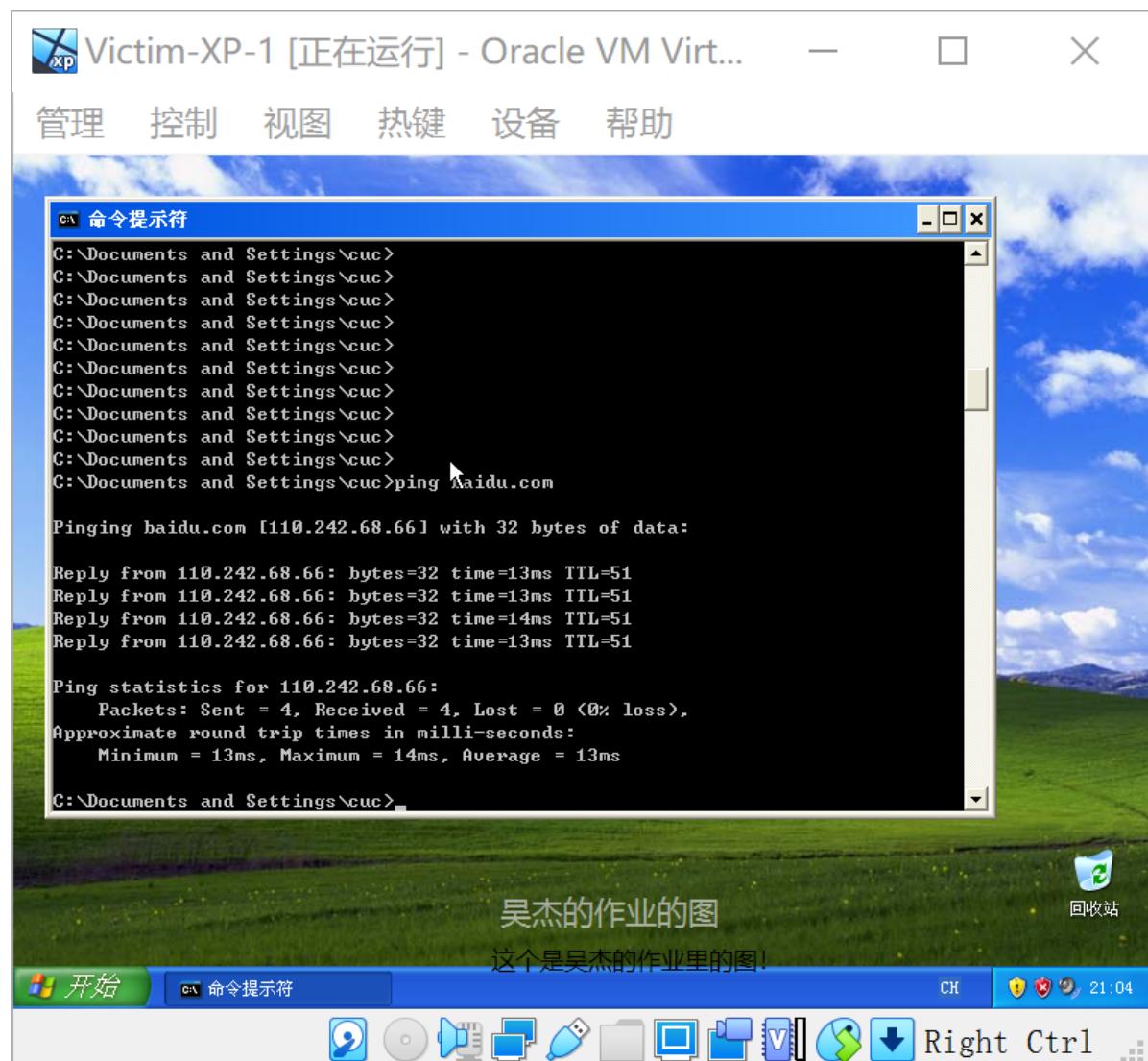


Figure 51: xp1_baidu.png

7.5.4 Kali1

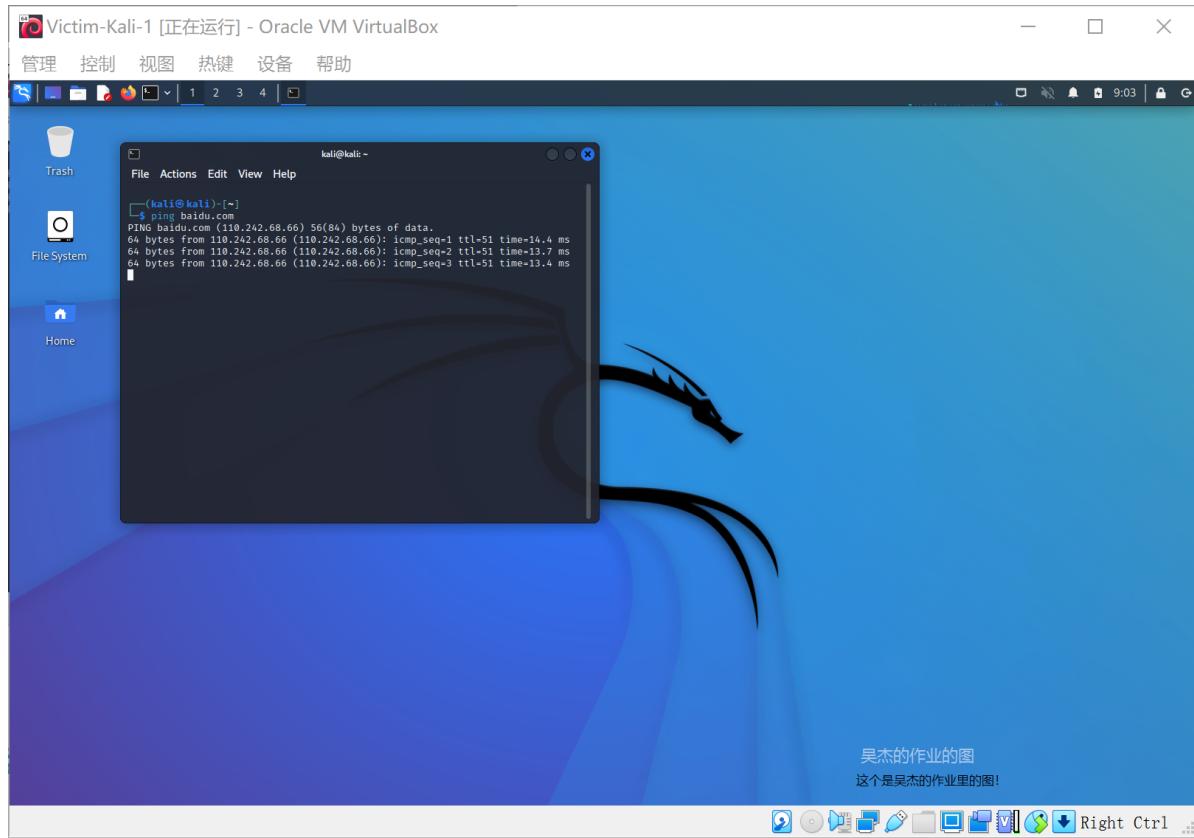


Figure 52: kali1_baidu.png

7.5.5 XP2

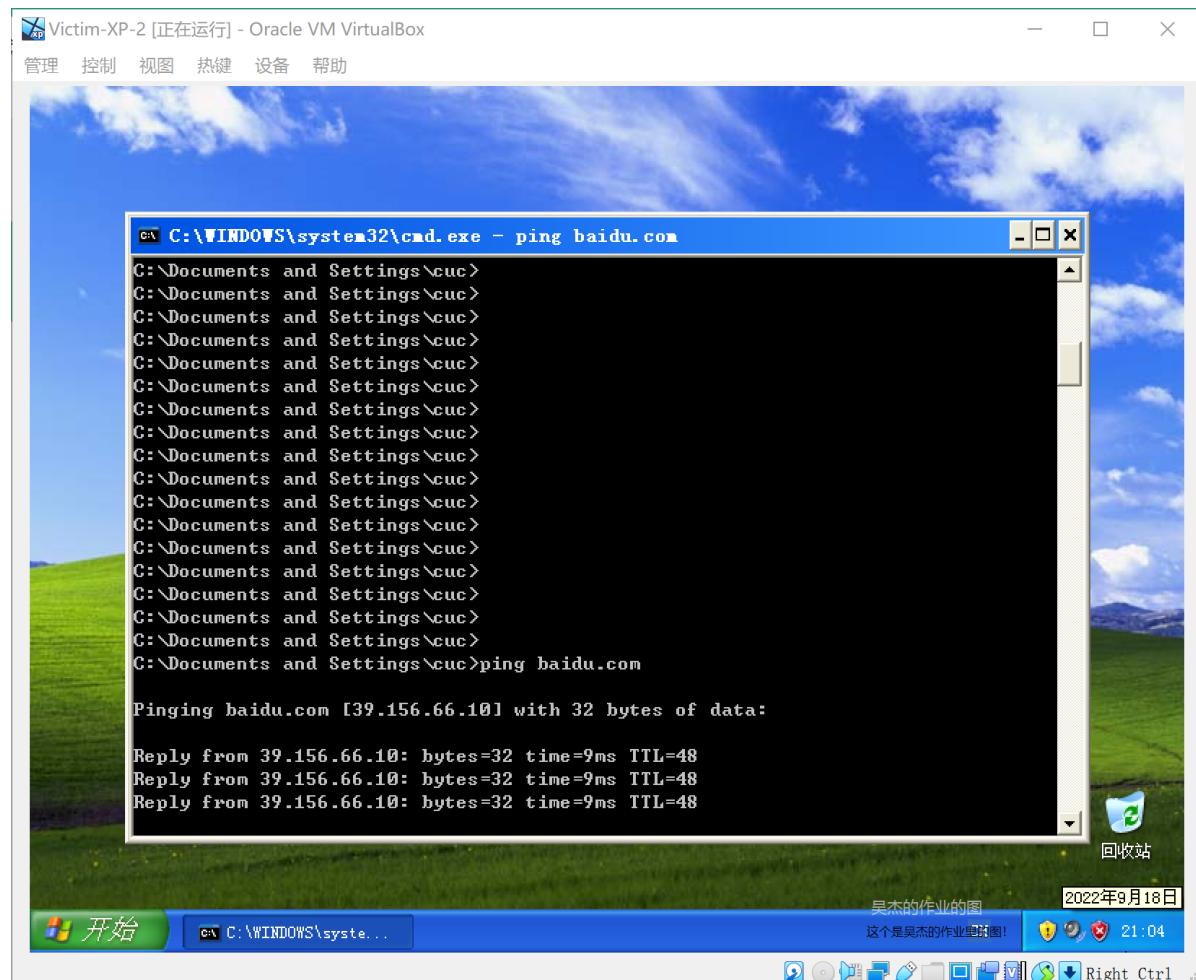


Figure 53: xp2_baidu.png

7.5.6 Debian2

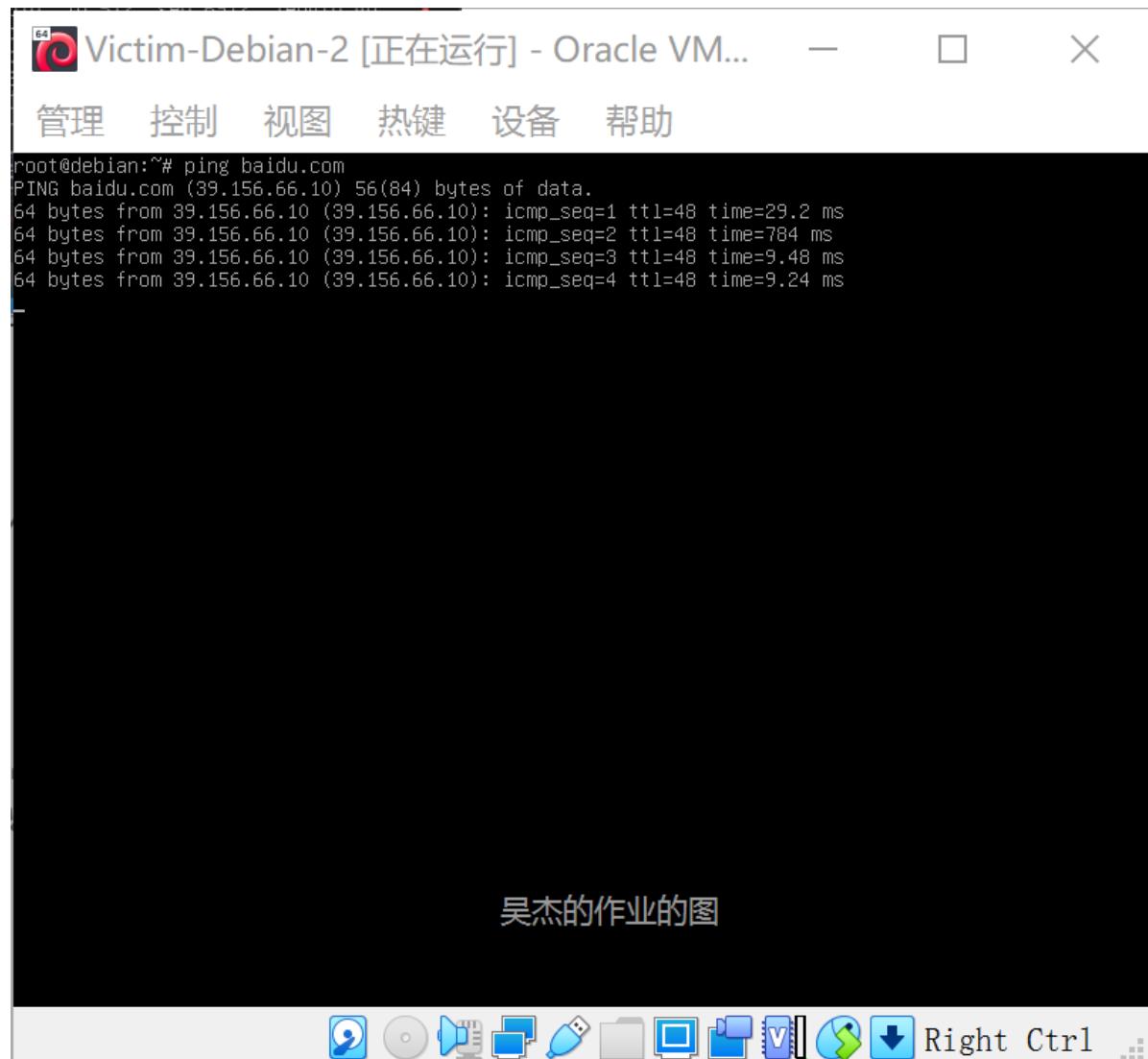


Figure 54: debian2_baidu.png