

实 验 报 告



课程名称 《网络攻击与防御技术》

学 院 计算机科学技术学院

专 业 信息安全

姓 名 黄 力

学 号 15307130275

开 课 时 间 2018 至 2019 学年第 1 学期

实验项目名称	基于 raw socket 的 UDP 扫描	成绩	
--------	------------------------	----	--

一、实验目的

(1) 理解 UDP 网络扫描的原理

(2) 利用 raw socket 编程实现对局域网内的主机进行 UDP 扫描

二、实验内容

(1) 采用 raw socket 方法编程，利用 UDP 协议、ICMP 协议，实现端口扫描。

(2) 验证所编软件，扫描校园网 DNS Server 的 DNS 服务端口是否开放。

(3) 通过截包工具截获扫描过程，分析扫描成功/失败的原因。

三、实验环境

(1) PC 机操作系统：macOS Mojave 10.14

(2) 虚拟机操作系统（parallels 13.1.1）：ubuntu 16.04 x86_64

(3) 开发语言：python 3.7.0

(4) 工具链：端口扫描：nmap 7.01、ICMP 截包：tcpdump

四、实验原理

(1) 通过向目标主机的某个端口发送任意 UDP 数据报文并根据该目标主机返回报文情况初步判断端口的开放情况：

- 1、若目标主机不返回任何数据报文，则该 UDP 端口为开放状态
- 2、若目标主机返回 ICMP 不可达报文，且 ICMP 头部的 type 和 code 字段值都是 3，则该 UDP 端口为关闭状态
- 3、若目标主机返回 ICMP 不可达报文，且 ICMP 头部的 type 字段值为 3，code 字段值为 1、2、9、10、13 中的一个，则该 UDP 端口为阻塞状态

(2) 为验证实验结果的准确性，同时使用 tcpdump 截获目标主机发送的 ICMP 报文，并将扫描结果与使用 nmap 工具扫描的结果做比较。

(3) 在代码实现中，为避免 icmp 接收函数阻塞，应使用一个单独的线程执行该函数，而在主线程中执行 udp 报文发送函数，当 icmp 接收函数执行完后再 join 回主线程。

五、实验步骤及结果

(1) 根据实验要求和原理编写代码，本次实验使用的开发语言为 python3.7.0，用到的库主要有 struct（数据包封装与拆解）、argparse（参数指定）、threading（线程库）、socket。

参数包含三项：

- 1、--IP：扫描的目标主机的 IP 地址，缺省值为"127.0.0.1"
- 2、--w：扫描方式，one 表示扫描一个指定端口，端口值由第三个参数—p 指定；all 表示扫描所有端口。缺省值为 all
- 3、--p：指定的某个扫描端口值，缺省值为 2333

```

21
22 parser = argparse.ArgumentParser()
23 parser.add_argument('--IP', type=str, default="127.0.0.1",
24                     help='IP you want to scan, default:127.0.0.1')
25 parser.add_argument('--w', type=str, default='all',
26                     help='select the way you want to scan:\nall: scan all the ports.\none: scan just one port gave.')
27 parser.add_argument('--p', type=int, default=2333,
28                     help='the port you want to scan(less than 65536)')
29 args = parser.parse_args()

```

代码共包含三个主要函数：udpMessageSender、icmpMessageReceiver、udpScanPort

udpMessageSender 函数实现对指定 IP 地址的主机的指定端口发送 udp 数据包：

```
35 def udpMessageSender(ip, port):
36     while True:
37         try:
38             sockUdp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
39             sockUdp.sendto("Hello world!", (ip, port))
40             sockUdp.close()
41             break
42         except:
43             print("Fail to send the udp message, try again")
44             time.sleep(1)
45             continue
46
```

icmpMessageReceiver 函数实现对指定 IP 地址的主机发送的 ICMP 报文进行监听并根据之前所述的实验原理分析报文头部的 type 和 code 字段然后打印分析结果，为使用 raw socket 截获报文，此处在建 sockIcmp 时要使用 socket.SOCK_RAW（第 53 行）：

```
51 def icmpMessageReceiver(ip, port):
52     try:
53         sockIcmp = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)
54     except:
55         print("You should run as root user")
56         return
57     sockIcmp.settimeout(5)
58     while True:
59         try:
60             packet, addr = sockIcmp.recvfrom(65536)
61             icmpHead = packet[20:28]
62             headType, code, checksum, packetID, sequence = struct.unpack(
63                 "BBHBB", icmpHead
64             )
65             break
66         except:
67             print("The port-%s on %s is opened" % (port, ip))
68             return
69     sockIcmp.close()
70     if code == 3 and headType == 3 and addr[0] == ip:
71         print("The port-%s on %s is closed" % (port, ip))
72     elif code in [1, 2, 9, 10, 13] and headType == 3 and addr[0] == ip:
73         print("The port-%s on %s is filter" % (port, ip))
74     return
75
```

udpScanPort 函数以主线程运行 udpMessageSender 函数，新开线程运行 icmpMessageReceiver 函数：

```
80 def udpScanPort(ip, port):
81     icmpReceiveThread = threading.Thread(target=icmpMessageReceiver, args=(ip, port))
82     icmpReceiveThread.daemon = True
83     icmpReceiveThread.start()
84
85     time.sleep(0.2)
86     udpMessageSender(ip, port)
87     time.sleep(0.2)
88
89     icmpReceiveThread.join()
90     return
91
```

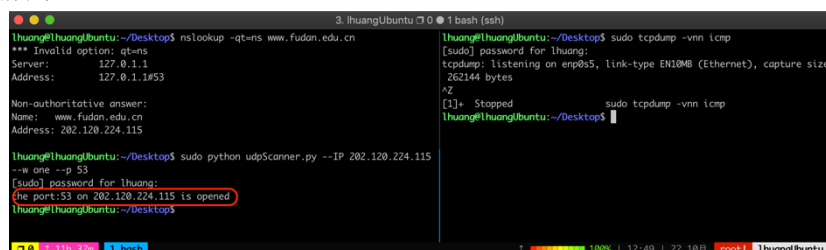
(2) 使用 (1) 中编写的程序扫描校园网 DNS Server 的 DNS 服务端口的开放情况

先使用 nslookup -qt=ns www.fudan.edu.cn 命令获得 DNS 服务器的 IP 地址（由于学校的 DNS 服务器不止一台，此处仅对负责解析 www.fudan.edu.cn 的 DNS 服务器进行扫描

在一个终端里执行 sudo tcpdump -vnn icmp 监听收到的 ICMP 报文

在另一终端里执行 sudo python udpScanner.py --IP 202.120.224.115 --w one --p 53 扫描 DNS 服务器的 DNS 服务端口 53 端口的开放情况

扫描结果如下图所示：



The screenshot shows three terminal windows. The left window shows the output of 'nslookup -qt=ns www.fudan.edu.cn', identifying the DNS server at 202.120.224.115. The middle window shows the output of 'sudo python udpScanner.py --IP 202.120.224.115 --w one --p 53', which reports 'The port:53 on 202.120.224.115 is opened'. The right window shows 'sudo tcpdump -vnn icmp' output, indicating it is listening on enp0s5.

从上图可知，DNS 服务器（202.120.224.115）的 DNS 服务端口（53 端口）是开放的，程序执行正确，同时 tcpdump 未捕获到 DNS 服务器的 ICMP 不可达报文也验证了实验结果的正确性

(3) 使用 (1) 中编写的程序扫描局域网内的某主机的端口的开放情况

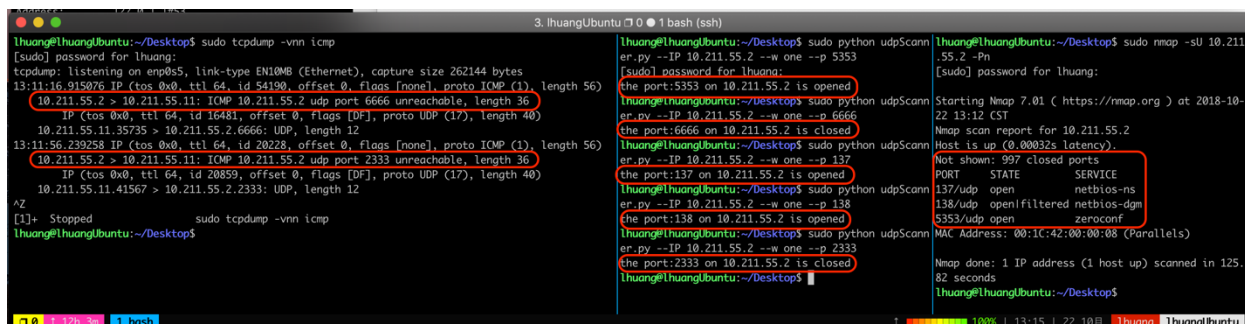
此处使用虚拟机扫描 PC 机（内网 IP：10.211.55.2），先扫描指定端口：分别是 5353、6666、137、138、2333，再扫描所有端口。同时用 nmap 和 tcpdump 验证扫描结果。

扫描指定端口：

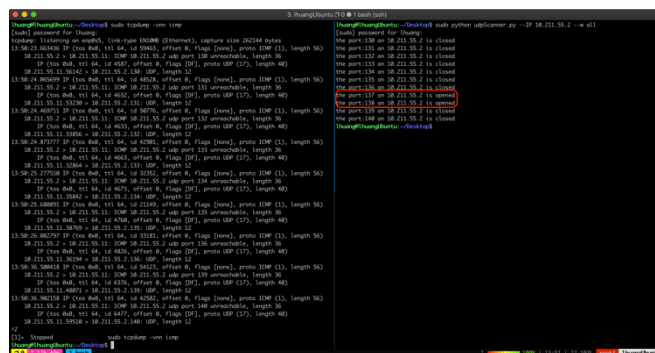
先在一个终端里执行 sudo tcpdump -vnn icmp 监听收到的 ICMP 报文

最后在第三个终端里执行 `sudo nmap -sU 10.211.55.2 -Pn` 使用 `nmap` 扫描 PC 机的 `udp` 端口开放情况以验证扫描结果

扫描结果如下图所示:



扫描所有端口（方便起见，此处仅扫描 130 到 140 之间的端口），步骤与扫描指定端口类似：
 先在一个终端里执行 `sudo tcpdump -vnn icmp` 监听收到的 ICMP 报文
 在另一个终端里执行 `sudo python udpScanner.py --IP 10.211.55.2 --w all`



从上图右边窗口可知，PC 机（10.211.55.2）的 130-140 之间的端口中只有 137 与 138 的 `udp` 端口是开放的，程序执行正确，同时上图左边窗口 `tcpdump` 也未捕获到 137、138 端口的 `ICMP` 不可达报文也验证了实验结果的正确性

通过本次实验，我理解了 UDP 端口扫描的原理和方式，成功编写了代码实现了对目标主机的 UDP 端口的扫描。此次实验过程中遇到的难点主要是 linux 需要使用 root 权限运行程序执行扫描（因为使用了 raw socket），前期不知道因此失败了几次，后来通过上网查阅解决；同时也了解了 linux 下 tcpdump 和 nmap 命令的基础使用。