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 的防 TCPsyn 端口扫描机制	成绩	

一、实验目的

（1）理解 TCPsyn 端口网络扫描的原理

（2）利用 raw socket 编程防止主机受到 TCPsyn 扫描，利用 raw socket 对局域网内的主机进行 TCPsyn 扫描

二、实验内容

（1）采用 raw socket 方法编程，利用 TCP 协议的建立原理实现防 TCPsyn 端口扫描，即实现主机上的所有 TCP 端口均开放的假象。

（2）验证所编软件，通过虚拟机本身进行自扫，扫描本机开放的 TCP 端口。当软件运行成功时，本机的所有 TCP 端口对于扫描的机器来说都是开放的（实际未开放）

（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（paralllels 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

四、实验原理

（1）通过向目标主机的某个端口发送 TCPsyn 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、若目标主机返回 TCPsynack 报文，则该 TCP 端口为开放状态

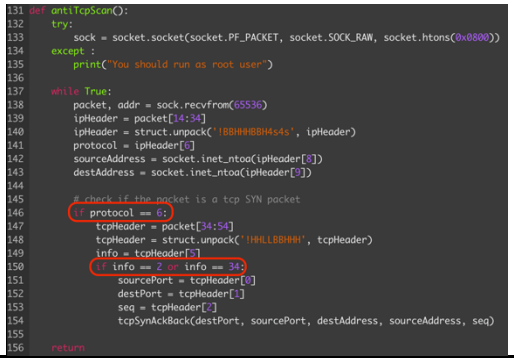
2、若目标主机返回 TCPrst 报文，则该 UDP 端口为关闭状态

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）、socket。

代码共包含三个主要函数：antiTcpScan、tcpSynAckBack、createTcpHeader、createIpHeader

antiTcpScan 函数使用 raw socket 建立套接字并通过该套接字上监听所有报文，若报文为 TCP 报文（ip 头部的 protocol 字段为 6）且为 syn 报文（tcp 头部的 syn 字段为 1）则调用 tcpSynAckBack 函数发送自己伪造的数据包：



tcpSynAckBack 函数调用 createIpHeader 函数与 createTcpHeader 函数伪造相应的 IP 头部与 TCP 头部，其中 TCP 头部字段置 syn 字段与 ack 字段为 1，并组装报文发送回去。

```
113 def tcpSynAckBack(sourcePort, destPort, sourceAddress, destAddress, seq):
114     try:
115         s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
116     except:
117         print("You should run as root user?")
118         return
119     ipHeader = createIpHeader(sourceAddress, destAddress)
120     tcpSynAckHeader = createTcpHeader(sourcePort, destPort, sourceAddress, destAddress, seq)
121     payload = ipHeader + tcpSynAckHeader
122     try:
123         s.sendto(payload, (destAddress, destPort))
124         print("succeed")
125     except:
126         print("Fail to send the tcpSynAck packet!")
127
128     return
```

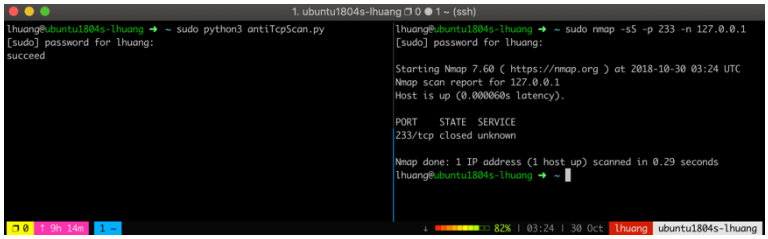
createIpHeader 函数构造 IP 头部，createTcpHeader 函数构造 TCP 头部：其中 ack 确认的序号应为收到的报文的序列号加 1，syn 和 ack 字段应置为 1。同时在计算校验和时应加入伪头部参与计算。计算方式在 checkSum 函数中。注：这 3 个函数的实现参考了两个博客的代码，在参考资料中已列出。

（2）使用（1）中编写的程序防止 TCPsyn 扫描。

先在一个终端里执行 sudo python3 antiTcpScan.py 命令开启防护程序

再在另一个终端里执行 sudo namp -sS -p 233 -n 127.0.0.1 使用 TCPsyn 扫描 233 端口

运行结果截图如下：



从上图可知，本次编写的程序未能阻挡 nmap 程序的 TCPsyn 扫描。

尝试换用其他端口（如：6666、2333）后结果仍然如上所示未能成功。

失败原因分析：

antiTcpScan.py 没有抢在内核发送 TCPrst 报文之前发送伪造的 TCPsynack 报文，从截图中可以看出防护程序成功截获并筛选出了 nmap 发送的 TCPsyn 报文，并将伪造的报文成功发送出去（打印出了 succeed 消息），但 nmap 扫描结果仍然为 233 端口关闭，这说明它已经在收到伪造报文之前收到了内核正确发出的 TCPrst 报文。

六、实验总结

通过本次实验，我理解了 TCPsyn 端口扫描的原理和方式，尝试编写代码对主机进行 TCPsyn 扫描的防护。代码的逻辑和实现方式是正确的，但因为未能屏蔽内核正常发出的 rst 报文或赶在它之前发送 synack 报文而没有成功实现防护。本次实验的难点主要有两点：一是在于需要自己构造 IP 头部和 TCP 头部，二是网络环境的复杂性可能导致实验失败。针对第一个难点解决方式是参考网上博客学习构造 IP 头部和 TCP 头部，针对第二个难点解决方式是使用虚拟机自己扫自己，避免了复杂的网络环境。

七、主要参考资料

1、https://blog.csdn.net/cheng_fangang/article/details/38709549

2、https://segmentfault.com/q/1010000012223391