**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（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、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  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：  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，此处在创建sockIcmp时要使用socket.SOCK\_RAW（第53行）：  udpScanPort函数以主线程运行udpMessageSender函数，新开线程运行icmpMessageReceiver函数：  （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：  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  在另一个终端里执行sudo python udpScanner.py --IP 10.211.55.2 --w one --p 5353(6666、137、138、2333)  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：    从上图中间窗口可知，PC机（10.211.55.2）的5353、137、138的udp端口是开放的，6666、2333端口则是关闭的，程序执行正确，同时上图左边窗口tcpdump只捕获到了6666、2333端口的ICMP不可达报文和上图右边窗口nmap的扫描结果也验证了实验结果的正确性  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  扫描结果如下图所示：  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  **六、实验总结**  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。 |