

# 网络丢包排查思路

原创

菜鸟别浪

于 2020-03-18 18:28:56 发布

3883

收藏 12


版权

分类专栏：

linux

网络

tcp/ip



同时被 3 个专栏收录 ▾

2 订阅

81 篇文章

订阅专栏

## 网络丢包排查思路

### 1.防火墙确认：

看防火墙是否配置了DROP特定端口范围的可能性  
方法：查看 **iptables** filter表，确认是否有相应规则会导致此丢包行为，命令：`sudo iptables-save -t filter`

### 2.连接跟踪表溢出

除了防火墙本身配置DROP规则外，与防火墙有关的还有连接跟踪表nf\_conntrack，Linux为每个经过内核网络栈的数据包，生成一个新的连接记录项，当服务器处理的连接过多时，连接跟踪表被打满，服务器会丢弃新建连接的数据包。

- 1) 问题确认：通过dmesg可以确认是否有该情况发生：  
命令：`dmesg |grep nf_conntrack`，如果输出值中有“nf\_conntrack: table full, dropping packet”，说明服务器nf\_conntrack表已经被打满。  
通过/proc文件系统查看nf\_conntrack表实时状态：  
# 查看nf\_conntrack表最大连接数  
`$ cat /proc/sys/net/netfilter/nf_conntrack_max`  
65536  
# 查看nf\_conntrack表当前连接数  
`$ cat /proc/sys/net/netfilter/nf_conntrack_count`  
7611
- 2) 解决办法：如果确认服务器因连接跟踪表溢出而开始丢包，首先需要查看具体连接判断是否正遭受DOS攻击，如果是正常的业务流量造成，可以考虑调整nf\_conntrack的参数：  
nf\_conntrack\_max决定连接跟踪表的大小，默认值是65535，可以根据系统内存大小计算一个合理值：`CONNTRACK_MAX = RAMSIZE(in bytes)/16384/(ARCH/32)`，如32G内存可以设置1048576；  
nf\_conntrack\_buckets决定存储conntrack条目的哈希表大小，默认值是nf\_conntrack\_max的1/4，延续这种计算方式：`BUCKETS = CONNTRACK_MAX/4`，如32G内存可以设置262144；  
nf\_conntrack\_tcp\_timeout\_established决定ESTABLISHED状态连接的超时时间，默认值是5天，可以缩短到1小时，即3600。  
命令行配置：  
`$ sysctl -w net.netfilter.nf_conntrack_max=1048576`  
`$ sysctl -w net.netfilter.nf_conntrack_buckets=262144`  
`$ sysctl -w net.netfilter.nf_conntrack_tcp_timeout_established=3600`

### 3.Ring Buffer溢出

排除了防火墙的因素，我们从底向上来看Linux接收数据包的处理过程，首先是网卡驱动层。物理介质上的数据帧到达后首先由NIC（网络适配器）读取，写入设备内部缓冲区Ring Buffer中，再由中断处理程序触发Softirq从中消费，Ring Buffer的大小因网卡设备而异。当网络数据包到达（生产）的速率快于内核处理（消费）的速率时，Ring Buffer很快会被填满，新来的数据包将被丢弃。

- 1) 判断方法：通过ethtool或/proc/net/dev可以查看因Ring Buffer满而丢弃的包统计，在统计项中以fifo标识：  
`$ ethtool -S eth0|grep rx_fifo`  
rx\_fifo\_errors: 0  
`$ cat /proc/net/dev`  
Inter-| Receive | Transmit  
可以看到服务器的接收方向的fifo丢包数并没有增加，说明此阶段不存在丢包。
- 2) 如果是，解决办法：  
如果发现服务器上某个网卡的fifo数持续增大，可以去确认CPU中断是否分配均匀，也可以尝试增加Ring Buffer的大小，通过ethtool可以查看网卡设备Ring Buffer最大值，修改Ring Buffer当

前设置:

查看eth0网卡Ring Buffer最大值和当前设置

```
$ ethtool -g eth0
```

Ring parameters for eth0:

Pre-set maximums:

RX: 4096

RX Mini: 0

RX Jumbo: 0

TX: 4096

Current hardware settings:

RX: 1024

RX Mini: 0

RX Jumbo: 0

TX: 1024

# 修改网卡eth0接收与发送硬件缓存区大小

```
$ ethtool -G eth0 rx 4096 tx 4096
```

Pre-set maximums:

RX: 4096

RX Mini: 0

RX Jumbo: 0

TX: 4096

Current hardware settings:

RX: 4096

RX Mini: 0

RX Jumbo: 0

TX: 4096

## 4.netdev\_max\_backlog溢出

netdev\_max\_backlog是内核从NIC收到包后, 交由协议栈(如IP、TCP)处理之前的缓冲队列。每个CPU核都有一个backlog队列, 与Ring Buffer同理, 当接收包的速率大于内核协议栈处理的速率时, CPU的backlog队列不断增长, 当达到设定的netdev\_max\_backlog值时, 数据包将被丢弃。

1) 判断方法: 通过查看/proc/net/netdev\_stat可以确定是否发生了netdev backlog队列溢出:  
其中:

```
[root@localhost diagnose-tools]# cat /proc/net/netdev_stat
00000029 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000032 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000025 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000002c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00006d0e 00000000 00000004 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000001a 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

每一行代表每个CPU核的状态统计, 从CPU0依次往下;  
每一列代表一个CPU核的各项统计: 第一列代表中断处理程序收到的包总数; 第二列即代表由于netdev\_max\_backlog队列溢出而被丢弃的包总数。

从上面的输出可以看出, 这台服务器统计中, 并没有因为netdev\_max\_backlog导致的丢包。

2) 解决办法

netdev\_max\_backlog的默认值是1000, 在高速链路上, 可能会出现上述第二列统计不为0的情况, 可以通过修改内核参数net.core.netdev\_max\_backlog来解决:

```
sysctl -w net.core.netdev_max_backlog=2000
```

## 5.反向路由过滤

反向路由过滤机制是Linux通过反向路由查询, 检查收到的数据包源IP是否可路由(Loose mode)、是否最佳路由(Strict mode), 如果没有通过验证, 则丢弃数据包, 设计的目的是防范IP地址欺骗攻击。rp\_filter提供了三种模式供配置:

0 - 不验证

1 - RFC3704定义的严格模式: 对每个收到的数据包, 查询反向路由, 如果数据包入口和反向路由出口不一致, 则不通过

2 - RFC3704定义的松散模式: 对每个收到的数据包, 查询反向路由, 如果任何接口都不可达, 则不通过

1) 如何确认

查看当前rp\_filter策略配置:

```
$ cat /proc/sys/net/ipv4/conf/eth0/rp_filter
```

0

如果这里设置为1，就需要查看主机的网络环境和路由策略是否可能会导致客户端的入包无法通过反向路由验证了。

从原理来看这个机制工作在网络层，因此，如果客户端能够Ping通服务器，就能够排除这个因素了。

## 2) 如何解决

根据实际网络环境将rp\_filter设置为0或2：

```
$ sysctl -w net.ipv4.conf.all.rp_filter=2
```

或

```
$ sysctl -w net.ipv4.conf.eth0.rp_filter=2
```

## 6.半连接队列溢出

半连接队列指的是TCP传输中服务器收到SYN包但还未完成三次握手的连接队列，队列大小由内核参数tcp\_max\_syn\_backlog定义。

当服务器保持的半连接数量达到tcp\_max\_syn\_backlog后，内核将会丢弃新来的SYN包。

### 1) 如何确认

通过dmesg可以确认是否有该情况发生：

```
$ dmesg | grep "TCP: drop open request from"
```

半连接队列的连接数量可以通过netstat统计SYN\_RECV状态的连接得知

### 2) 如何解决

tcp\_max\_syn\_backlog的默认值是256，通常推荐内存大于128MB的服务器可以将该值调高至1024，内存小于32MB的服务器调低到128，同样，该参数通过sysctl修

改：netstat -ant|grep SYN\_RECV|wc -l

```
sysctl -w net.ipv4.tcp_max_syn_backlog=1024
```

另外，上述行为受到内核参数tcp\_syncookies的影响，若启用syncookie机制，当半连接队列溢出时，并不会直接丢弃SYN包，而是回复带有syncookie的SYC+ACK包，设计的目的是防范SYN Flood造成正常请求服务不可用。



显示推荐内容

评论 1 您还未登录，请先

登录

后发表或查看评论