协议栈 rp_fileter 导致收数据失败.

问题：客户环境，rsyslogd 未收到其他设备传过来的日志。
SIP 平台端设备，除了接收探针上传的日志，还能通过 syslog 接收其它设备发送的数据。linux 实现了 rsyslogd 服务，其监听 tcp 端口，作为 syslog 的服务端，接收数据

排查过程:

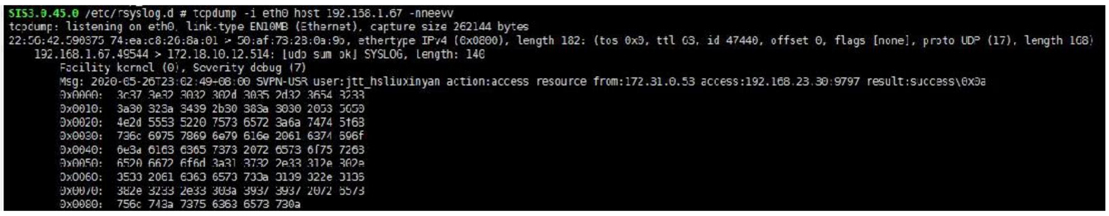
经过其他人前面的排查，问题已经缩短了一些边界：

其它设备向 eth0 网口发送时，平台的 rsyslog 没生成日志，其它设备向 eth6 网口发送时，平台的 rsyslog 能生成日志

了解一下环境:

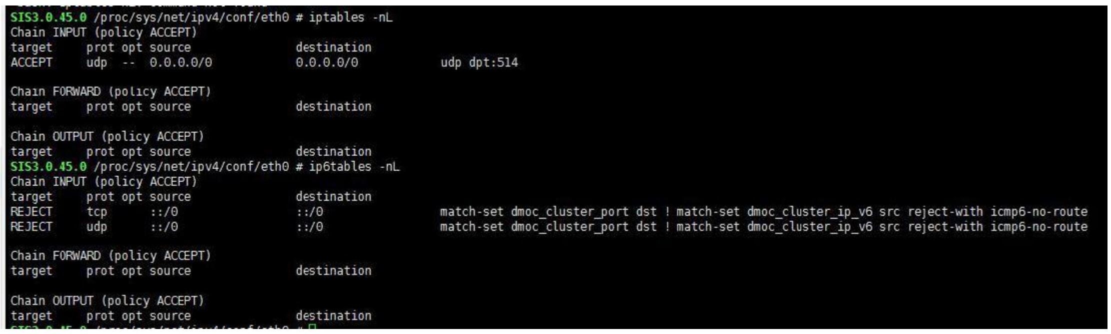
eth0	172.18.10.12
eth6	192.16.26.1
发送方	192.168.1.67　192.168.1.68
数据流	发送方--->rsyslogd--->本地文件

排查开始:
1. 抓包查看 eth0，有抓取到数据包



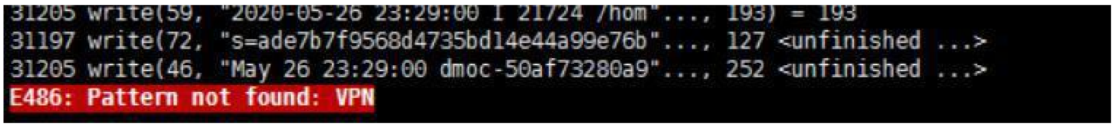
```
SIS3.0.45.0 /etc/rsyslog.d # tcpdump -i eth0 host 192.168.1.67 -nneevv
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:56:42.590375 74:ea:c8:26:8a:01 > 50:af:73:28:0a:9b, ethertype IPv4 (0x0800), length 182: (tos 0x0, ttl 63, id 47440, offset 0, flags [none], proto UDP (17), length 168)
    192.168.1.67.49544 > 172.18.10.12.514: [udp sum ok] SYSLOG, length: 140
        Facility kernel (0), Severity debug (7)
        Msg: 2020-05-26T23:02:49+08:00 SVPN-USR user:jtt_hsliuxinyan action:access resource from:172.31.0.53 access:192.168.23.30:9797 result:success\0x0a
        0x0000:  3c37 3e32 3032 302d 3035 2d32 3654 3233
        0x0010:  3a30 323a 3439 2b30 383a 3030 2053 5650
        0x0020:  4e2d 5553 5220 7573 6572 3a6a 7474 5f68
        0x0030:  736c 6975 7869 6e79 616e 2061 6374 696f
        0x0040:  6e3a 6163 6365 7373 2072 6573 6f75 7263
        0x0050:  6520 6672 6f6d 3a31 3732 2e33 312e 302e
        0x0060:  3533 2061 6363 6573 733a 3139 322e 3136
        0x0070:  382e 3233 2e33 303a 3937 3937 2072 6573
        0x0080:  756c 743a 7375 6363 6573 730a
```

2. iptables 没有规则限制



```
SIS3.0.45.0 /proc/sys/net/ipv4/conf/eth0 # iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:514

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
SIS3.0.45.0 /proc/sys/net/ipv4/conf/eth0 # ip6tables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp      ::/0                 ::/0                 match-set dmoc_cluster_port dst ! match-set dmoc_cluster_ip_v6 src reject-with icmp6-no-route
REJECT     udp      ::/0                 ::/0                 match-set dmoc_cluster_port dst ! match-set dmoc_cluster_ip_v6 src reject-with icmp6-no-route

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

3. 查看本地文件夹，无文件生，strace 跟踪 rsyslogd 进程，未看到有数据包特征
strace -fp $(pidof rsyslogd) -e trace=write -o csc.x

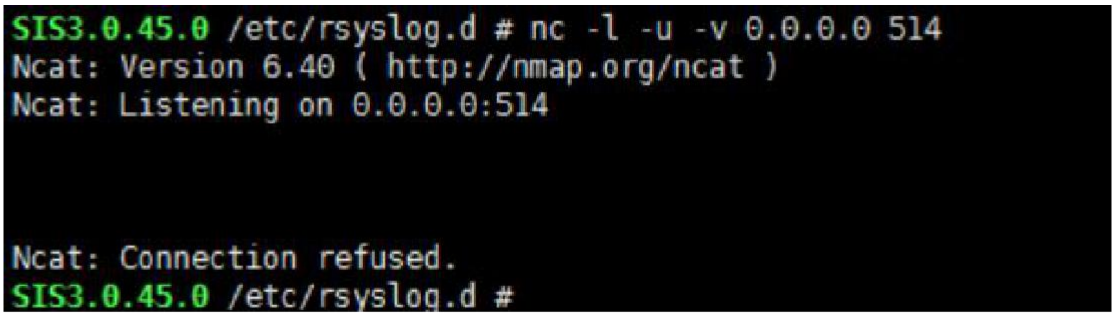


```
31205 write(59, "2020-05-26 23:29:00 1 21724 /hom"..., 193) = 193
31197 write(72, "s=ade7b7f9568d4735bd14e44a99e76b"..., 127 <unfinished ...>
31205 write(46, "May 26 23:29:00 dmoc-50af73280a9"..., 252 <unfinished ...>
E486: Pattern not found: VPN
```

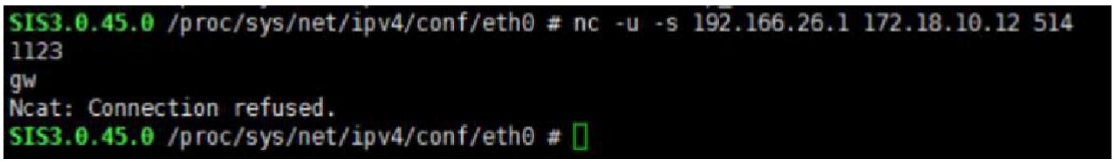
由于 rsyslogd 有过虑功能，故而还不能确认是否因为数据包被过滤掉了

4. 分析 rsyslogd 配置，找出过滤的点配置文件在/etc/rsyslog.d/, 删除过滤都配置文件，重启发现依然存在问题，排查过滤功能

5. 界定 rsyslogd 问题边界
经过上面分析，问题界定在【协议栈，rsyslogd】，不管是哪个，都会比较麻烦。总是要进一步快速排除一个。
回到问题本身，rsyslogd 只是对外提供了一个 UDP/TCP 端口服务，如果它收到了数据，那问题就发生在 rsyslogd。
选择使用一个新的程序，替代 rsyslogd，来界定 rsyslogd 是否收到数据。
修改 rsyslogd 配置，关闭其 UDP 514 端口监听
开启 netcat 监听 UDP 514 端口：nc -l -u -v 0.0.0.0 514
开启 netcat 进程，连接 UDP 514 端口：nc -u -s 192.166.26.1 172.18.10.12 514
发现无法正常收发数据包



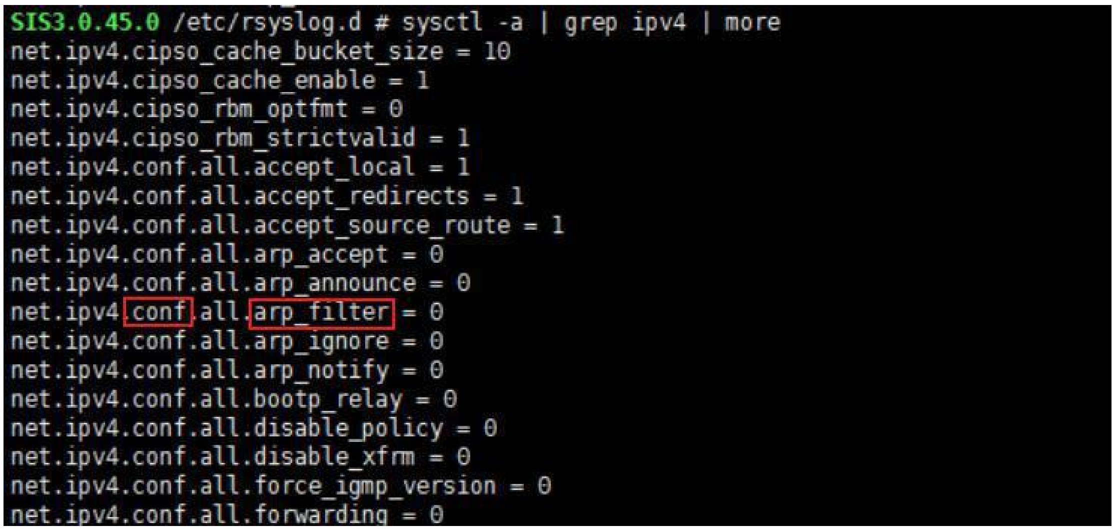
```
SIS3.0.45.0 /etc/rsyslog.d # nc -l -u -v 0.0.0.0 514
Ncat: Version 6.40 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:514



Ncat: Connection refused.
SIS3.0.45.0 /etc/rsyslog.d #
```



```
SIS3.0.45.0 /proc/sys/net/ipv4/conf/eth0 # nc -u -s 192.166.26.1 172.18.10.12 514
1123
gw
Ncat: Connection refused.
SIS3.0.45.0 /proc/sys/net/ipv4/conf/eth0 #
```

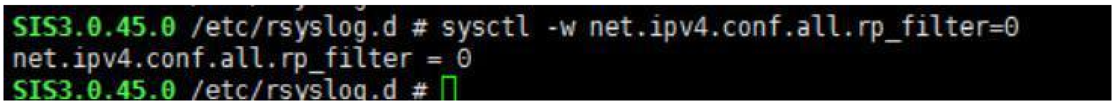
去掉源地址信息，再连接，发现有数据包，说明问题发生在协议栈!

6. google 一下呗。。。
网上虽然都没有直接都答案，都有因为这个 rp_filter 参数设置，导致网络问题
在平台也尝试修改配置，但是发现 net.ipv4.eth0.rp_filter 根本就不存在……
我只能 grep+find 了。。。。



```
SIS3.0.45.0 /etc/rsyslog.d # sysctl -a | grep ipv4 | more
net.ipv4.cipso_cache_bucket_size = 10
net.ipv4.cipso_cache_enable = 1
net.ipv4.cipso_rbm_optfmt = 0
net.ipv4.cipso_rbm_strictvalid = 1
net.ipv4.conf.all.accept_local = 1
net.ipv4.conf.all.accept_redirects = 1
net.ipv4.conf.all.accept_source_route = 1
net.ipv4.conf.all.arp_accept = 0
net.ipv4.conf.all.arp_announce = 0
net.ipv4.conf.all.arp_filter = 0
net.ipv4.conf.all.arp_ignore = 0
net.ipv4.conf.all.arp_notify = 0
net.ipv4.conf.all.bootp_relay = 0
net.ipv4.conf.all.disable_policy = 0
net.ipv4.conf.all.disable_xfrm = 0
net.ipv4.conf.all.force_igmp_version = 0
net.ipv4.conf.all.forwarding = 0
```

发现平台的 net 配置，路径上多了 conf，继续修改配置

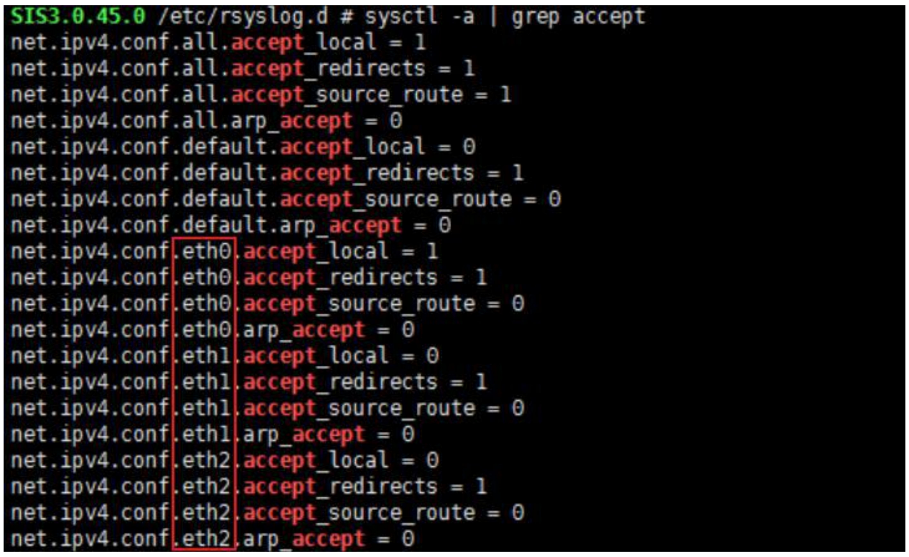


```
SIS3.0.45.0 /etc/rsyslog.d # sysctl -w net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.all.rp_filter = 0
SIS3.0.45.0 /etc/rsyslog.d #
```

改完配置，发现 netcat 依然无法收到数据

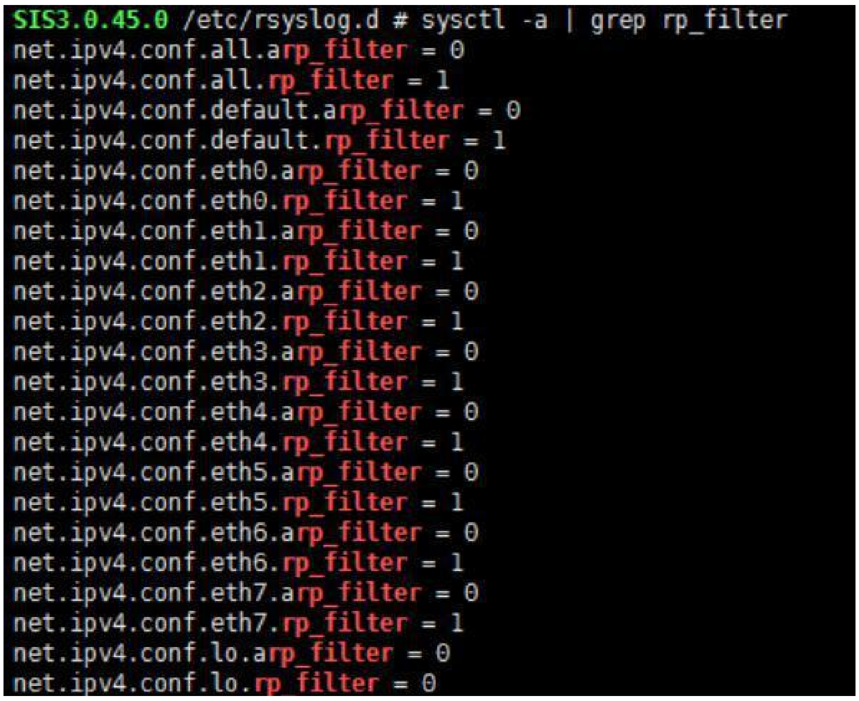
这时，大致知道是协议栈把包扔了，而且很有可能是某个参数功能的策略，然而 rp_filter 初步尝试是无效的。这又该何去何从？--------------答案：用眼睛猜

猜测，协议栈可能使用 recv、accept 之类名称来命名参数，所以试试 grep 这个关键词



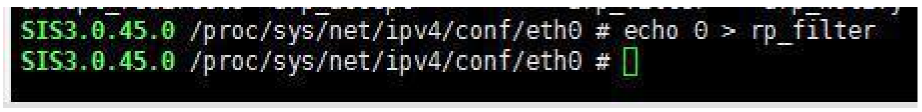
```
SIS3.0.45.0 /etc/rsyslog.d # sysctl -a | grep accept
net.ipv4.conf.all.accept_local = 1
net.ipv4.conf.all.accept_redirects = 1
net.ipv4.conf.all.accept_source_route = 1
net.ipv4.conf.all.arp_accept = 0
net.ipv4.conf.default.accept_local = 0
net.ipv4.conf.default.accept_redirects = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.arp_accept = 0
net.ipv4.conf.eth0.accept_local = 1
net.ipv4.conf.eth0.accept_redirects = 1
net.ipv4.conf.eth0.accept_source_route = 0
net.ipv4.conf.eth0.arp_accept = 0
net.ipv4.conf.eth1.accept_local = 0
net.ipv4.conf.eth1.accept_redirects = 1
net.ipv4.conf.eth1.accept_source_route = 0
net.ipv4.conf.eth1.arp_accept = 0
net.ipv4.conf.eth2.accept_local = 0
net.ipv4.conf.eth2.accept_redirects = 1
net.ipv4.conf.eth2.accept_source_route = 0
net.ipv4.conf.eth2.arp_accept = 0
```

发现，还有针对具体网口的子配置项，所以，怀疑是不是前面的 rp_filter 不对



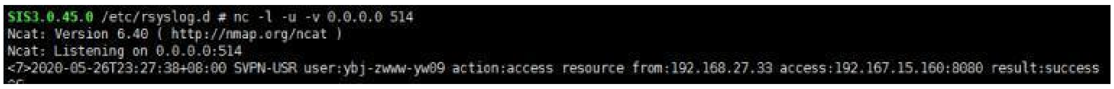
```
SIS3.0.45.0 /etc/rsyslog.d # sysctl -a | grep rp_filter
net.ipv4.conf.all.arp_filter = 0
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.arp_filter = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.eth0.arp_filter = 0
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.eth1.arp_filter = 0
net.ipv4.conf.eth1.rp_filter = 1
net.ipv4.conf.eth2.arp_filter = 0
net.ipv4.conf.eth2.rp_filter = 1
net.ipv4.conf.eth3.arp_filter = 0
net.ipv4.conf.eth3.rp_filter = 1
net.ipv4.conf.eth4.arp_filter = 0
net.ipv4.conf.eth4.rp_filter = 1
net.ipv4.conf.eth5.arp_filter = 0
net.ipv4.conf.eth5.rp_filter = 1
net.ipv4.conf.eth6.arp_filter = 0
net.ipv4.conf.eth6.rp_filter = 1
net.ipv4.conf.eth7.arp_filter = 0
net.ipv4.conf.eth7.rp_filter = 1
net.ipv4.conf.lo.arp_filter = 0
net.ipv4.conf.lo.rp_filter = 0
```

为了确保这次修改 100%有效，又想起了 sysfs 和 procfs，通过 find 找到了对应的配置目录



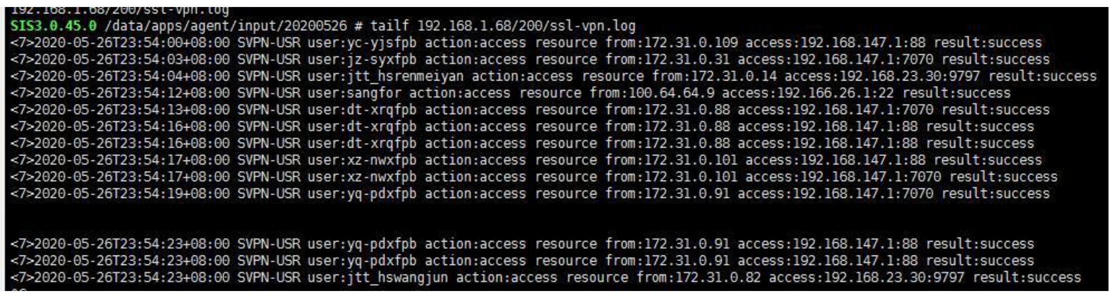
```
SIS3.0.45.0 /proc/sys/net/ipv4/conf/eth0 # echo 0 > rp_filter
SIS3.0.45.0 /proc/sys/net/ipv4/conf/eth0 # 
```

检查 netcat，收到了数据!



```
SIS3.0.45.0 /etc/rsyslog.d # nc -l -u -v 0.0.0.0 514
Ncat: Version 6.40 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:514
<7>2020-05-26T23:27:38+08:00 SVPN-USR user:ybj-zwww-yw09 action:access resource from:192.168.27.33 access:192.167.15.160:8080 result:success
```

关闭 netcat，恢复 rsyslogd 配置，查看文件，果然生成了



```
192.168.1.68/200/ssl-vpn.log
SIS3.0.45.0 /data/apps/agent/input/20200526 # tailf 192.168.1.68/200/ssl-vpn.log
<7>2020-05-26T23:54:00+08:00 SVPN-USR user:yc-yjsfpb action:access resource from:172.31.0.109 access:192.168.147.1:88 result:success
<7>2020-05-26T23:54:03+08:00 SVPN-USR user:jz-syxfpb action:access resource from:172.31.0.31 access:192.168.147.1:7070 result:success
<7>2020-05-26T23:54:04+08:00 SVPN-USR user:jtt_hsrenmeiyan action:access resource from:172.31.0.14 access:192.168.23.30:9797 result:success
<7>2020-05-26T23:54:12+08:00 SVPN-USR user:sangfor action:access resource from:100.64.64.9 access:192.166.26.1:22 result:success
<7>2020-05-26T23:54:13+08:00 SVPN-USR user:dt-xrqfpb action:access resource from:172.31.0.88 access:192.168.147.1:7070 result:success
<7>2020-05-26T23:54:16+08:00 SVPN-USR user:dt-xrqfpb action:access resource from:172.31.0.88 access:192.168.147.1:88 result:success
<7>2020-05-26T23:54:16+08:00 SVPN-USR user:dt-xrqfpb action:access resource from:172.31.0.88 access:192.168.147.1:88 result:success
<7>2020-05-26T23:54:17+08:00 SVPN-USR user:xz-nwxfpb action:access resource from:172.31.0.101 access:192.168.147.1:88 result:success
<7>2020-05-26T23:54:17+08:00 SVPN-USR user:xz-nwxfpb action:access resource from:172.31.0.101 access:192.168.147.1:7070 result:success
<7>2020-05-26T23:54:19+08:00 SVPN-USR user:yq-pdxfpb action:access resource from:172.31.0.91 access:192.168.147.1:7070 result:success


<7>2020-05-26T23:54:23+08:00 SVPN-USR user:yq-pdxfpb action:access resource from:172.31.0.91 access:192.168.147.1:88 result:success
<7>2020-05-26T23:54:23+08:00 SVPN-USR user:yq-pdxfpb action:access resource from:172.31.0.91 access:192.168.147.1:88 result:success
<7>2020-05-26T23:54:23+08:00 SVPN-USR user:jtt_hswangjun action:access resource from:172.31.0.82 access:192.168.23.30:9797 result:success
```

到这里，已经算是知道问题了

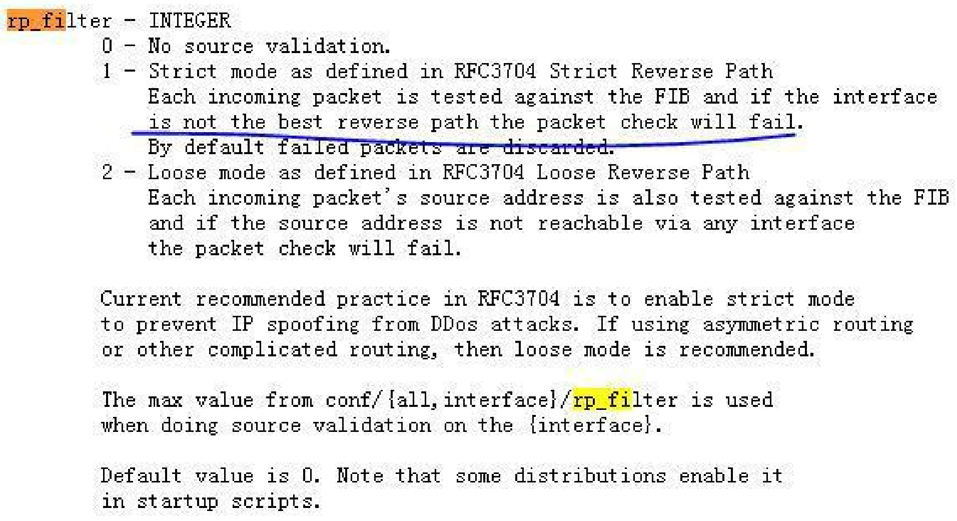
现在，修改了 rp_filter，是否就行云流水般“给客户解决掉”？

虽然修改之后收到了数据包，那为什么内核要默认拒绝数据包呢？

还是得了解 rp_filter 之后，才能正确决策

很快，找到了该选项的说明：

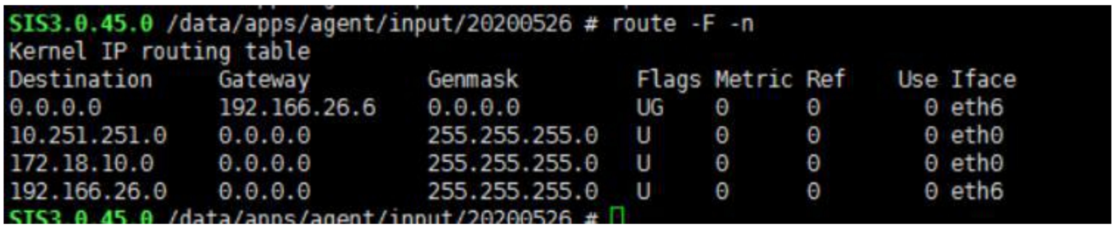
https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt



```
rp_filter - INTEGER
	0 - No source validation.
	1 - Strict mode as defined in RFC3704 Strict Reverse Path
	    Each incoming packet is tested against the FIB and if the interface
	    is not the best reverse path the packet check will fail.
	    By default failed packets are discarded.
	2 - Loose mode as defined in RFC3704 Loose Reverse Path
	    Each incoming packet's source address is also tested against the FIB
	    and if the source address is not reachable via any interface
	    the packet check will fail.

	Current recommended practice in RFC3704 is to enable strict mode
	to prevent IP spoofing from DDos attacks. If using asymmetric routing
	or other complicated routing, then loose mode is recommended.

	The max value from conf/{all,interface}/rp_filter is used
	when doing source validation on the {interface}.

	Default value is 0. Note that some distributions enable it
	in startup scripts.
```

从文档说明知道，rp_filter=1（平台默认设置 1）会经过 FIB 表（选路表）测试，如果 best reverse path（也就是选择的发送网口），那么就校验失败，丢弃数据包。

检查客户环境 FIB 表



```
SIS3.0.45.0 /data/apps/agent/input/20200526 # route -F -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.166.26.6    0.0.0.0         UG    0      0        0 eth6
10.251.251.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.18.10.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.166.26.0    0.0.0.0         255.255.255.0   U     0      0        0 eth6
SIS3.0.45.0 /data/apps/agent/input/20200526 # 
```

从 FIB 来看，第一条则为默认路由（目的地址&子网掩码是 0.0.0.0）

所谓默认路由，即是：如果数据包的目的地址，在各个网口找不到匹配的网段地址，则通过默认路由指定的网口发送。

所以，我们可以回归到通信的几个 IP：eth0 172.18.10.0 网段，eth6 192.266.26.0 网段，发送方 192.168.1 网段

发送方的数据包，从 eth0 口进入，经过 test FIB，最终会选择 eth6 口出去，自然不符合 rp_filter=1 的策略，数据包就被丢弃了。

然而，了解了 rp_filter 的含义之后，却不能在客户环境修改这个参数，因为说不定哪一天客户不小心搭错了网线，或者交换机配置安全策略，就可能导致与设备的 tcp 通信中断。

总之，如果改了这个参数，tcp 通信很容易就中途因为随意的改动导致异常。

自然不能在此挖巨坑，将来说不定还是自己来查……

解决办法

与客户协商，网络不能这么配置，eth0 这种设置，通常是要求对端也是同网段的。