如何利用网络回溯分析系统分析压测过程中的异常

应用审计 域间策略/安全域 ASPF 会话同步 **孔凡安** 2024-06-12 发表

组网及说明

不涉及

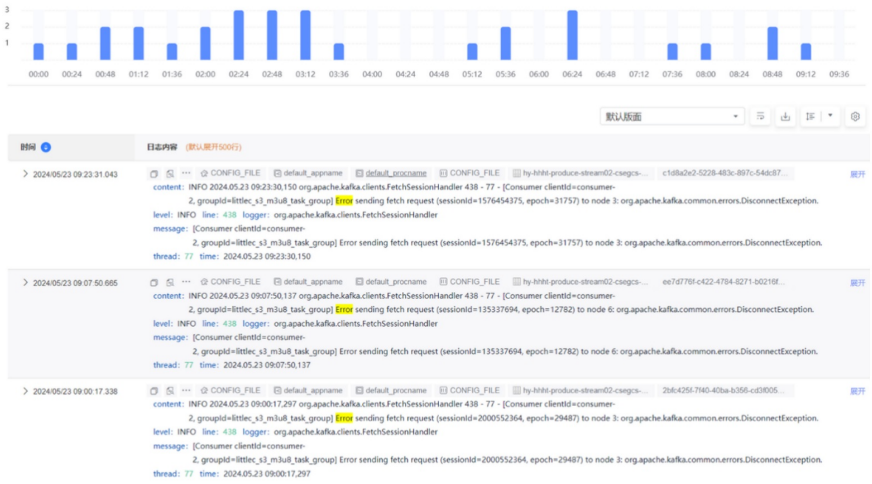
告警信息

不涉及

问题描述

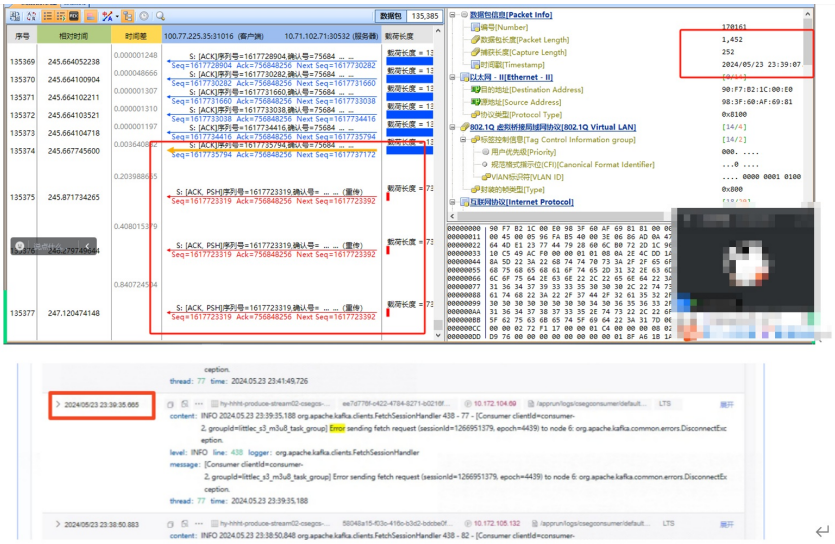
案例背景：某局点云主机进行Kafka服务器压测，测试过程中发现不定时有Kafka消息响应超时问题。

补充：云主机即客户端为100.X.X.X网段，Kafka服务器为10.X.X.X网段。

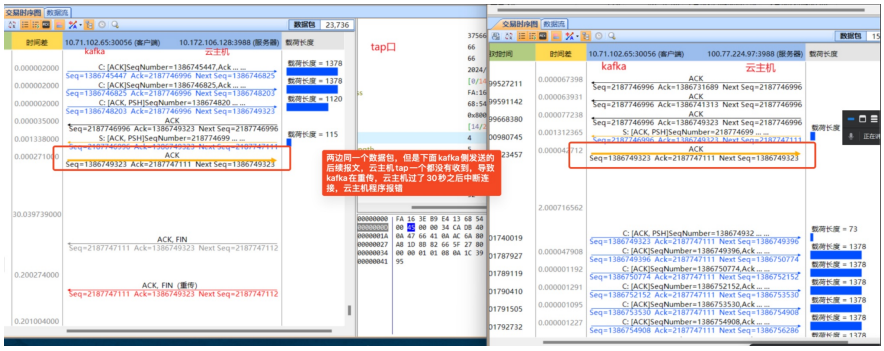


过程分析

通过对Kafka服务器侧的报文进行分析发现，双方传输数据的期间，Kafka侧在传输报文的时候，发现客户端（云主机侧）突然对传输的报文不响应，Kafka侧在重传多次后，结束了这个会话，同时客户端也报错，没有取到这个数据。



观察其他时间点的报文也是类似情况：



经过以上的分析，初步分析问题出现在中间一台安全设备（FW）上。
通过前面的分析，FW丢弃的都是ACK报文，怀疑是故障时候FW上没有对应的会话。
压测过程中存在端口复用，即在快拆快建的过程中，第1条流量的会话还没老化删除，第2条流量就来了；第2条流量跑了一会后第1条流量的会话老化了，然后第2条流量就断了。
遵循这个思路，让现场放通反向的安全策略，会话删除之后反向ACK报文到达设备也可以正常建立会话。
设备对于TCP协议报文处理可以参考案例：https://zhiliao.h3c.com/theme/details/223790
配置反向安全策略放通后，故障果然不再复现。证实了我们的猜想。

查看抓包并没有发现有拆链报文，即fin置位的报文。说明配置反向安全策略只是误打误撞的解决了问题，实际原因不是前期的猜想。
没办法，只能取消反向策略复现继续分析了。。。
复现的时候，在设备上收集会话信息，以及debug的相关信息。观察流量在防火墙上的变化。

故障复现后，根据故障时的debug发现会话被异常删除，导致后续kafka服务器发给客户端的PUSH-ACK报文被异常删除。

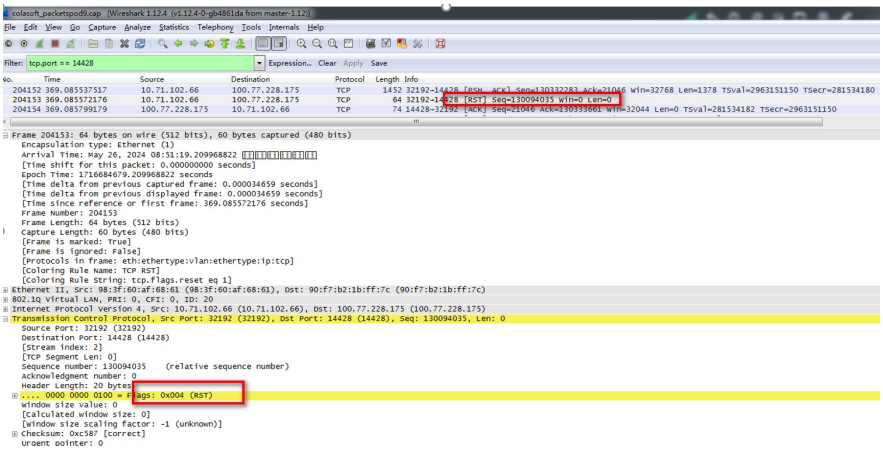
```
*May 26 08:52:08:125 2024 HHHT-PSC-P10F2-POD1-S-FW-M9016-1 SESSION/7/TABLE: -Slot=0.1;
 Tuple5(EVENT): 100.77.228.175/14428-->10.71.102.66/32192(TCP(6))
 Session entry was deleted.

*May 26 08:52:08:126 2024 HHHT-PSC-P10F2-POD1-S-FW-M9016-1 IPFW/7/IPFW_PACKET: -Slot=0.1
;
Receiving, interface = Route-Aggregation1.234
version = 4, headlen = 20, tos = 0
pktlen = 125, pktid = 44131, offset = 0, ttl = 59, protocol = 6
checksum = 55697, s = 10.71.102.66, d = 100.77.228.175
channelID = 1, vpn-InstanceIn = 0, vpn-InstanceOut = 0.
prompt: Receiving IP packet from interface Route-Aggregation1.234.
Payload: TCP
  source port = 32192, destination port = 14428
  sequence num = 0xce7075c3, acknowledgement num = 0xb9e48675, flags = 0x18
  window size = 32768, checksum = 0x80b5, header length = 32.

*May 26 08:52:08:126 2024 HHHT-PSC-P10F2-POD1-S-FW-M9016-1 ASPF/7/PACKET: -Slot=0.1; The
first packet was dropped by packet filter or object-policy. Src-ZOne=POD_3, Dst-ZOne=Untr
ust;If-In=Route-Aggregation1.234(7000), If-Out=Route-Aggregation104.21(7017); Packet Info:Src-IP=10.7
1.102.66, Dst-IP=100.77.228.175, VPN-Instance=none, Src-Port=32192, Dst-Port=14428. Protocol=TCP(
6). Flag=0x18. Seq=3463476675.
```

查看收集的会话信息，发现故障会话在删除前由正常的TCP_ESTABLISHED状态变成了TCP_CLOSE状态。

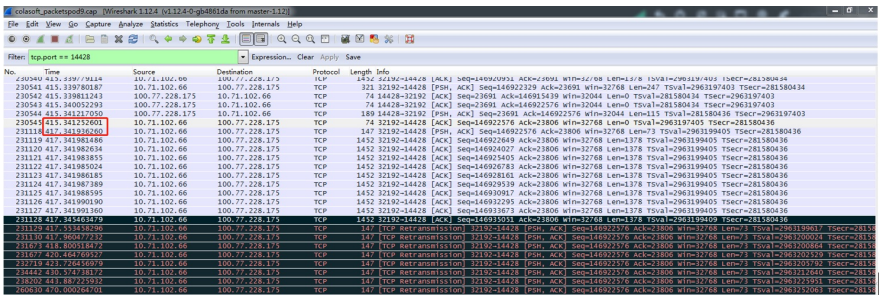
```
08:51:49 Beijing Sun 05/26/2024
Time Zone : Beijing add 08:00:00
Initiator:
  Source      IP/port: 100.77.228.175/14428
  Destination IP/port: 10.71.102.66/32192
  DS-Lite tunnel peer: -
  VPN instance/VLAN ID/Inline ID: -/-/-
  Protocol: TCP(6)
  Inbound interface: Route-Aggregation103.21
  Source security zone: Untrust
Responder:
  Source      IP/port: 10.71.102.66/32192
  Destination IP/port: 100.77.228.175/14428
  DS-Lite tunnel peer: -
```

VPN instance/VLAN ID/Inline ID: -/-/-
Protocol: TCP(6)
Inbound interface: Route-Aggregation1.234
Source security zone: POD_3
State: TCP_CLOSE
Application: GENERAL_TCP
Rule ID: 99
Rule name: 005-20221102-0027
Start time: 2024-05-24 20:20:45 TTL: 1s
Initiator->Responder: 2105909 packets 189915897 bytes
Responder->Initiator: 35632670 packets 3710039534 bytes

再次检查抓取的报文发现在会话删除前防火墙收到了10.71.102.66---100.77.228.175的RST报文，导致会话跳转到TCP_CLOSE状态（老化时间2秒）。



但此时Kafka并没有真正关闭连接，而是继续发送ACK报文（此时不会跳转会话状态，只有收到syn包才会跳转到稳态），后续正常报文的报文交互可以刷新TCP_CLOSE老化时间并转发，但是报文230545和231118的间隔超过了2秒，导致会话被删除，造成后续Kafka服务器发给客户端的ACK报文重传被丢弃的现象。（安全策略只放通了客户端->服务器方向，反向策略未放通）



看起来需要排查Kafka发送RST的原因，你以为事情到这就结束了吗，其实并没有：）
查看该RST报文的ip.ttl发现为63，正常Kafka发送给客户端的报文ip.ttl为62。怀疑Kafka和防火墙之间可能存在代理设备发送了该报文。
因此问题最终落在代理设备上，并非防火墙问题。
代理设备阻断正常业务的场景之前有过案例介绍：SSLVPN拨号登录提示“与VPN网关建立连接失败”问题处理过程

解决方法

排查异常RST报文的来源。
防火墙侧可以先暂时更改TCP_CLOSE老化时间规避，对应命令：

session aging-time state命令用来设置各协议状态的会话老化时间。
undo session aging-time state命令用来恢复缺省情况，如果不指定任何参数，则将所有协议状态的会话老化时间都恢复为缺省情况。

【命令】

session aging-time state { **fin** | **icmp-reply** | **icmp-request** | **icmpv6-reply** | **icmpv6-request** | **rawip-open** | **rawip-ready** | **syn** | **tcp-close** | **tcp-est** | **tcp-time-wait** | **udp-open** | **udp-ready** } *time-value*

undo session aging-time state [**fin** | **icmp-reply** | **icmp-request** | **icmpv6-reply** | **icmpv6-request** | **rawip-open** | **rawip-ready** | **syn** | **tcp-close** | **tcp-es**

t | **tcp-time-wait** | **udp-open** | **udp-ready**]

【缺省情况】

各协议状态的会话老化时间如下，单位为秒：

- FIN：30
- ICMP-REPLY：30
- ICMP-REQUEST：60
- ICMPv6-REPLY：30
- ICMPv6-REQUEST：60
- RAWIP-OPEN：30
- RAWIP-READY：60
- SYN：30
- TCP-CLOSE：2
- TCP-EST：3600
- TCP-TIME-WAIT：2
- UDP-OPEN：30
- UDP-READY：60

【视图】

系统视图

【缺省用户角色】

network-admin

mdc-admin

vsys-admin

【参数】

fin：表示TCP协议FIN-WAIT状态的会话老化时间。

icmp-reply：表示ICMP协议REPLY状态的会话老化时间。

icmp-request：表示ICMP协议REQUEST状态的会话老化时间。

icmpv6-reply：表示ICMPv6协议REPLY状态的会话老化时间。

icmpv6-request：表示ICMPv6协议REQUEST状态的会话老化时间。

rawip-open：表示RAWIP-OPEN状态的会话老化时间。

rawip-ready：表示RAWIP-READY状态的会话老化时间。

syn：表示TCP协议SYN-SENT和SYN-RCV状态的会话老化时间。

tcp-close：表示TCP协议CLOSE状态的会话老化时间。

tcp-est：表示TCP协议ESTABLISHED状态的会话老化时间。

tcp-time-wait：表示TCP协议TIME-WAIT状态的会话老化时间。

udp-open：表示UDP协议OPEN状态的会话老化时间。

udp-ready：表示UDP协议READY状态的会话老化时间。

time-value：指定的老化时间，其中**tcp-close**、**tcp-time-wait**状态取值范围为0～100000，其他状态会话老化时间取值范围为1～100000，若设备上存在支持硬件快速转发的业务板，则**tcp-close**状态取值范围为0～63，单位为秒。

【使用指导】

会话进入稳态后，如果该会话属于**session aging-time application**命令中指定的一种应用层协议，则此会话的老化时间为指定的应用层协议老化时间；否则为四层协议状态的老化时间（由**session aging-time state**命令配置）。

对TCP会话来说，如果会话符合长连接会话的规则，那么该会话稳态的老化时间为长连接老化时间（由**session persistent acl**命令配置）。

【举例】

设置TCP协议SYN-SENT和SYN-RCV状态的老化时间为60秒。

```
<Sysname> system-view
[Sysname] session aging-time state syn 60
```

【相关命令】

- **display session aging-time state**
- **session aging-time application**
- **session persistent acl**