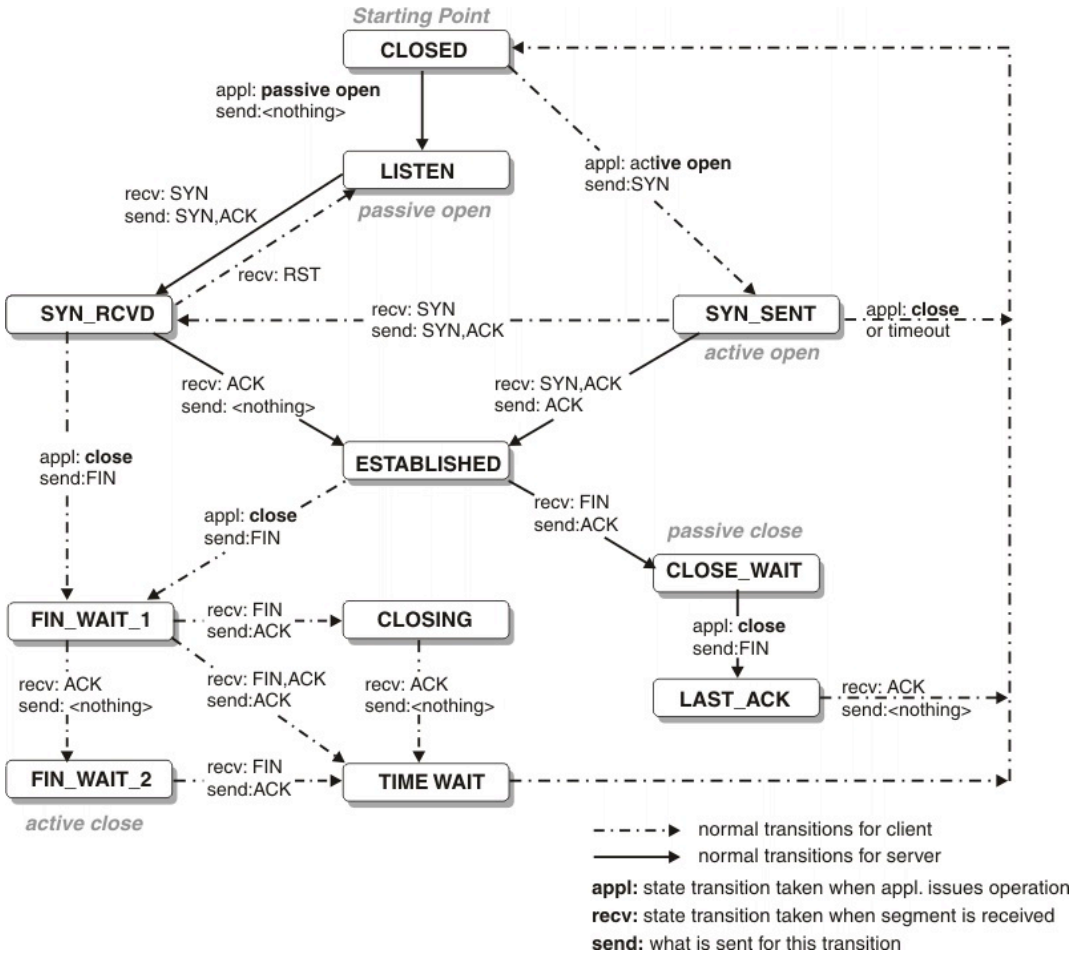
TCP的状态

gaccob

2012 年 8 月 14 日

1. TCP的流转状态图

一图胜万言，先看一下来自UPN的一张图：



- CLOSED: 无连接状态.
- LISTEN: server监听端口, 等待连接, 这个叫passive_open.
- SYN_RCVD: server收到SYN, 发出SYN, ACK, 等待client ACK.
- SYN_SENT: client发出SYN, 请求建立连接, 等待server ACK, 这个叫active_open.
- ESTABLISHED: 连接建立.
- FIN_WAIT_1: 主动方发送FIN, 主动关闭, 等待server ACK, 这个叫active_close.
- FIN_WAIT_2: 主动方收到了对方ACK, 继续等待对方FIN.
- CLOSING: 主动方在active_close的同时, 收到对端的FIN, 这表明了双方同时请求关闭, 这时主动方发出ACK, 继续等待对方ACK. 一般比较罕见.
- TIME_WAIT: 主动方收到被动方的FIN并发出ACK之后的状态, 一般会有2MSL, 是为了保证最后一个ACK包不丢失.
- CLOSE_WAIT: 被动方收到FIN发出ACK之后就进入这个状态, 等待数据传输完, 到发出FIN之前, 都处于这个状态. 这个叫passive_close.
- LAST_ACK: 被动方在CLOSE_WAIT状态中发出FIN, 则进入LAST_ACK, 等待对端最后一次ACK确认.

需要注意的一点是: 图中的关闭连接的状态流转, 不限定于client或者server, 而是解释为主动方(发起active_close)或被动方(passive_close).

server从监听->建立连接->被动关闭的典型状态流转: CLOSED->LISTEN->SYNC_RCVD->ESTABLISHED->CLOSE_WAIT->LAST_ACK->CLOSED.

client从建立连接->主动断开的典型状态流转: CLOSED->SYN_SENT->ESTABLISHD->FIN_WAIT1->FIN_WAIT2->TIME_WAIT->CLOSED.

2. FIN_WAIT_2和CLOSE_WAIT

当FIN_WAIT_2时, 如果因为对端的程序bug, 或者对端的网络出现故障, 会导致主动方的状态超时, 这时候会直接进入CLOSED状态, 超时时间与tcp_fin_timeout参数有关. 特别是当网络中有大量的FIN_WAIT_2或者CLOSE_WAIT时 (一个在主动方, 一个在被动方, 两者是相互对应的), 要注意检查为何收不到FIN, 是不是程序存在bug (一般不太可能是攻击, 因为已经建立了连接, 对攻击者消耗也很大).

3. TIME_WAIT和2MSL

一般我们会发现主动关闭方有很多TIME_WAIT的状态, 这是正常的. TIME_WAIT的超时时间长达2个报文的存活周期, 这有两个原因: 1. 保证最后一个ACK传输到对端(没有重传请求); 2. 保证网络中不会有残留的报文, 被新连接接收而产生数据错乱.

对于主动关闭的短连接服务器来说, 可能产生大量的TIME_WAIT状态, 带来一定程度上资源浪费, 一种方法是设置socket的SO_LINGER标志位来使socket调用close()之后发送RST来强制终止TCP连接, 但是并不推荐.

如果有root权限的话, 推荐的做法是调整内核参数: tcp_tw_reuse=1, tcp_tw_recycle=1. tcp_tw_reuse开启重用, 允许TIME_WAIT的socket重新用于tcp连接 (只要五元素不一样就不会有问题, 但是如果这时候对端以相同的端口请求建立连接, 则会收到FIN); tcp_tw_recycle打开了TIME_WAIT的socket的快速回收开关, 目测回收时间在1s左右, 一般来说, 单开启tcp_tw_recycle已经基本OK了.

4. SYN_RCVD与syn-flood

当server中出现大量的SYN_RCVD状态时, 代表了tcp负载过大来不及处理新的连接, 也许是syn-flood攻击(是指攻击方估计发送syn包而不发送ack, 导致tcp连接建立不起来, 从而使服务器长时间停留在SYN_RCVD状态而消耗服务器资源). 一般碰到这种情况, 可以从/var/log/messages中看到日志, 如果是负载跟不上需要做扩容或者调整负载均衡策略, 如果是异常的攻击行为, 可以通过调整内核参数或者防火墙的黑名单策略来缓解.

内核参数tcp_max_syn_backlog: 代表处于SYN_RCVD状态的socket在队列中的最大数量, 一般默认是1024, 调高这个值可以有效的缓解(注意, 不是解决)小规模的syn-flood. 在负载繁忙的服务器上, 也应该调高这个参数.

现代的unix系统中, 普遍采用多队列处理方式, 即一个基本队列维护已经ESTABLISH的socket, 其他半打开的连接, 例如SYN_RCVD的socket, 在另一个队列. 这样可以很大程度上缓冲syn-flood这种攻击.

内核参数tcp_syncookies: 只有在内核编译时选择了CONFIG_SYNCOOKIES时才会发生作用, 当出现syn等候队列出现溢出时象对方发送syncookies, 目的是为了防止syn-flood攻击. 这个选项严重违背了tcp协议, 而且会对某些服务导致严重的性能影响(例如smtp转发), 一般不推荐打开.

5. BSD socket API与tcp状态

listen(): 进入LISTEN;

accept(): 从已经ESTABLISHED的队列中取出一个来, 队列的大小取决于listen()的参数;

close(): 开始进入FIN_WAIT_2状态, 等待到TIME_WAIT;