



导航

博客园

首页

新随笔

联系

订阅

管理

2019年12月

日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

公告

昵称: HarrisonZhou

园龄: 5年1个月

粉丝: 4

关注: 49

+加关注

统计

随笔 - 18

文章 - 0

评论 - 0

引用 - 0

搜索

找我看

谷歌搜索

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

随笔档案

2016年11月(1)

2016年5月(1)

2016年4月(3)

2016年3月(7)

2015年12月(1)

2015年11月(2)

2015年10月(2)

2015年8月(1)

阅读排行榜

1. 用wireshark抓包分析TCP三次握手、四次挥手以及TCP实现可靠传输的机制(33516)

2. hbase运行shell时ERROR:org.apache.hadoop.hbase.PleaseHoldException: Master is initializing的解决办法(7236)

3. Tomcat中解决sql server连接失败--- java.lang.ClassNotFoundException: com.microsoft.jdbc.sqlserver.SQLServerDriver(6127)

4. Tomcat8.0.21登录时忘记密码用户名和密码(6064)

5. vs 2013 Express 无法启动程序xxx.exe,系统找不到指定文件(655)

推荐排行榜

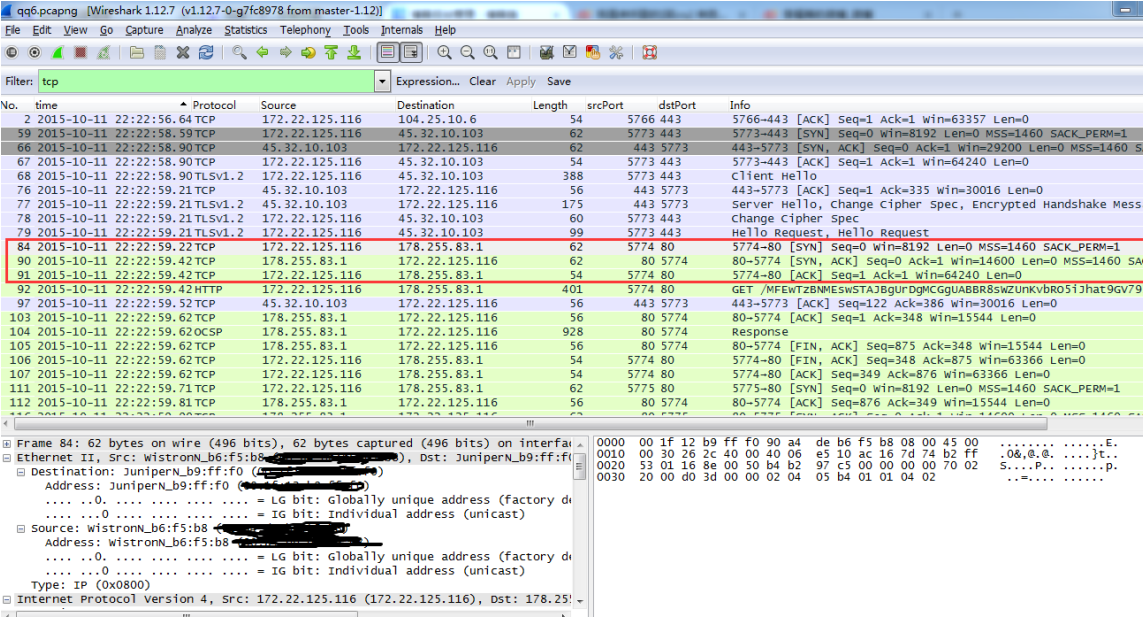
用wireshark抓包分析TCP三次握手、四次挥手以及TCP实现可靠传输的机制

关于TCP三次握手和四次挥手大家都在《计算机网络》课程里学过，还记得当时高超老师耐心地讲解。大学里我遇到的最好的老师大概就是这位了，虽然他只给我讲过《java程序设计》和《计算机网络》，但每次课几乎都动手敲代码或者当场做实验。好了不扯了，下面进入正题。

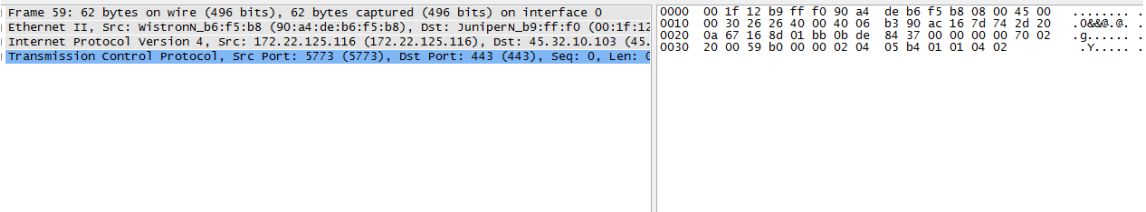
关于三次握手和四次挥手的理论部分可以在很多资料上找到，我今天动手抓了几个包验证证书上的理论，毕竟那些字段和整个通信的过程学起来很枯燥。

一、三次握手：

我用wireshark抓取的数据包如下：



观察其中红色方框内的3条数据包就是一次TCP建立连接的过程，客户端首先向服务器发一个数据包syn位置1，5774->80,嘿，哥们儿，您我想访问您的web资源，能不能把你的80端口打开；服务器向客户端返回一个数据包syn位置1，ack位置1，80->5774.可以啊，我已经把80端口打开了，但是为了保证待会儿可靠传输，你也把你的5774端口打开呗；最后，客户端会再向服务器端发送一个数据包ack位置1，5774->80，没问题我也把的5774端口打开了，好的到此一次TCP连接就此建立。下面具体分析数据包的各个字段。



在wireshark的这个界面中，左边这个框框是对数据包各个层的概述和详细信息，右边的框框是真实的数据包，我调成用16进制显示，最右边还有一堆乱七八糟的符号，其实就是16进制数据的ascii码解释，0000，0010,0020,0030就是16进制的地址，如果对汇编比较熟的话右边的框框应该很容易看明白。左边的框框第一行是数据包整体概述，第二行是以太网这一层（链路层）的详细信息，最主要的是双方的mac地址，第三行是网络层（网际层）的详细信息，最主要的是双方的IP地址，第四行是传输层的详细信息，最主要的是双方的端口号。

每一层都有一个字段指向上一层，表明上一层是什么协议。这大概是因为发包的时候会在数据上依次加上应用层、传输层、网络层、链路层的头部，但是对方收到数据包后是从最底层（链路层）开始层层剥去头部解包的，所以在每层上有一个字段指向上层表明上层的协议，对方就知道下一步该怎么解包了。说了这么多，可能又要被喷了，no picture you say a J8 a！

```

❑ Ethernet II, Src: WistronN_b6:f5:b8 (08:00:27:08:00:27:08:00:b6:f5:b8), Dst: JuniperN_b9:ff:f0 (08:00:27:08:00:27:08:00:b9:ff:f0)
  ❑ Destination: JuniperN_b9:ff:f0 (08:00:27:08:00:27:08:00:b9:ff:f0)
    Address: JuniperN_b9:ff:f0 (08:00:27:08:00:27:08:00:b9:ff:f0)
    .... 0. .... = LG bit: Globally unique address (factory default)
    .... 0. .... = IG bit: Individual address (unicast)
  ❑ Source: WistronN_b6:f5:b8 (08:00:27:08:00:27:08:00:b6:f5:b8)
    Address: WistronN_b6:f5:b8 (08:00:27:08:00:27:08:00:b6:f5:b8)
    .... 0. .... = LG bit: Globally unique address (factory default)
    .... 0. .... = IG bit: Individual address (unicast)
  Type: IP (0x0800)
  表明上层(网络层)是用的IP协议

❑ Internet Protocol Version 4, Src: 172.22.125.116 (172.22.125.116), Dst: 45.32.10.103 (45.32.10.103)
  Version: 4
  Header Length: 20 bytes
  ❑ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... 00.. = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
  Total Length: 48
  Identification: 0x2626 (9766)
  ❑ Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  表明上层(传输层)是用的TCP协议
  ❑ Header checksum: 0xb390 [validation disabled]
  Source: 172.22.125.116 (172.22.125.116)
  Destination: 45.32.10.103 (45.32.10.103)

```

由于建立TCP连接用不到应用层协议，所以传输层就没有相应的指明上层(应用层)的字段了。

下面更直观地感受一下，三次握手过程中标志位的变化情况，首先客户端发送的数据包syn位置1，然后服务器端回复的数据包syn位置1，ack位置1，最后客户端发送的数据包ack位置1.以下三幅图分别为TCP三次握手的数据包中传输层的标志位字段

```

Acknowledgment number: 0
Header Length: 28 bytes
[ ... 0000 0000 0010 = Flags: 0x002 (SYN)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Nonce: Not set
  .... 0... .... = Congestion window Reduced (CWR): Not set
  .... .0.. .... = ECN-Echo: Not set
  .... ..0. .... = Urgent: Not set
  .... ...0 .... = Acknowledgment: Not set
  .... .... 0... = Push: Not set
  .... .... .0.. = Reset: Not set
  [ ... ..1. = Syn: Set
  [ [Expert Info (Chat/Sequence): Connection establish request (SYN): server port 443]
  .... .... ..0 = Fin: Not set
Window size value: 8192
[Calculated window size: 8192]
Checksum: 0x59b0 [validation disabled]

[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Acknowledgment number: 1 (relative ack number)
Header Length: 28 bytes
[ ... 0000 0001 0010 = Flags: 0x012 (SYN, ACK)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Nonce: Not set
  .... 0... .... = Congestion window Reduced (CWR): Not set
  .... .0.. .... = ECN-Echo: Not set
  .... ..0. .... = Urgent: Not set
  [ ... ..1. .... = Acknowledgment: Set
  .... .... 0... = Push: Not set
  .... .... .0.. = Reset: Not set
  [ ... ..1. = Syn: Set
  [ [Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port 443]
  .... .... ..0 = Fin: Not set
Window size value: 20200]

```

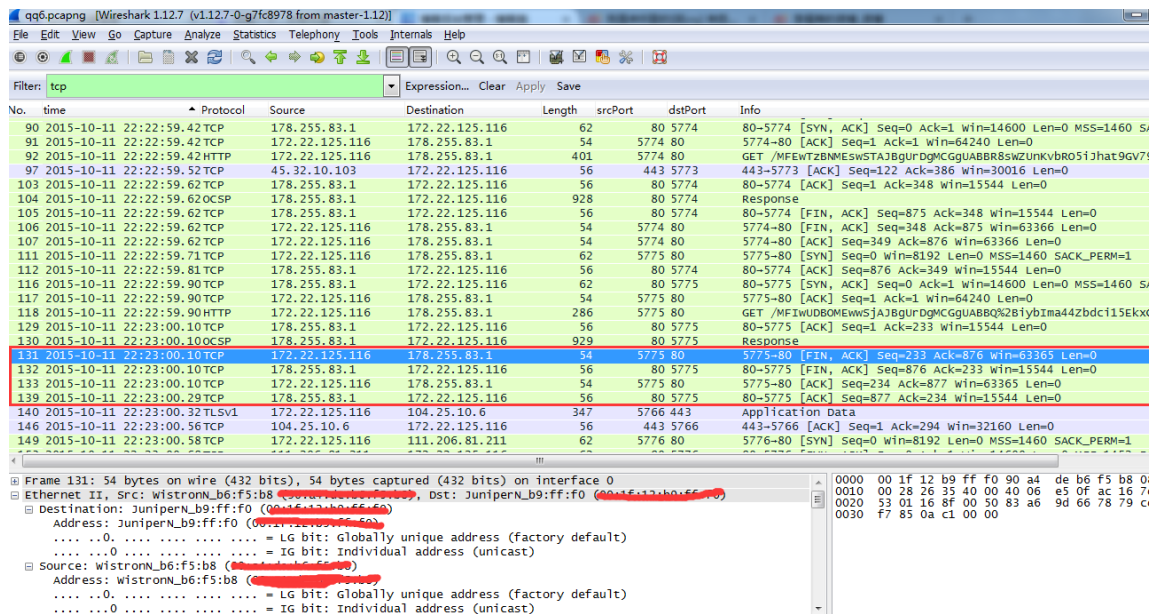
```

[TCP Segment Len: 0]
Sequence number: 1 (relative sequence number)
Acknowledgment number: 1 (relative ack number)
Header Length: 20 bytes
.... 0000 0001 0000 = Flags: 0x010 (ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... ....0... = Push: Not set
.... .....0.. = Reset: Not set
.... .... ..0. = Syn: Not set
.... .... ...0 = Fin: Not set
window size value: 64240
[calculated window size: 64240]

```

二、四次挥手：

这次抓到的包和书本上分析的过程有点不一样，过程就不赘述了，直接看图分析



关于可靠传输好像没抓到合适的包啊，下次有机会再写一篇博客，今天太晚了。

三、抓取telnet明文传输的值

众所周知，telnet在网上是明文传输，因为这样技术出现比较早，当时的人心里也没那么黑暗，想不到去盗取别人账号密码什么的，现在不同了，用telnet这种远程登录方式是很不安全的，如果黑客通过一定方式把你的流量欺骗到他的电脑，再通过抓包软件分析你的账号和密码，那就……。所以，建议使用ssh，毕竟安全一点。

我这次是用telnet远程登录美国的一台开放的路由器telnet route-server.ip.att.net。账号：rviews,密码：rviews。用wireshark抓包并过滤后发现了密码和账号。还是那句话,no picture you say a J8 a! 由于telnet是一个字符一个字符的传的，所以截的图可能会有点多。先用户名部分：

facebook.pcapng [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]

Filter: telnet

No.	Time	Source	Destination	Length	srcPort	dstPort	Info
7:42.70	TELNET	172.22.125.116	12.0.1.28	70	8489	23	Telnet Data ...
7:42.95	TELNET	172.22.125.116	12.0.1.28	57	8489	23	Telnet Data ...
7:43.29	TELNET	172.22.125.116	12.0.1.28	66	8489	23	Telnet Data ...
7:43.54	TELNET	172.22.125.116	12.0.1.28	57	8489	23	Telnet Data ...
7:43.54	TELNET	172.22.125.116	12.0.1.28	57	8489	23	Telnet Data ...
7:47.28	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:49.06	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:49.41	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:49.79	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:50.04	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:52.06	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:52.73	TELNET	172.22.125.116	12.0.1.28	56	8489	23	Telnet Data ...
7:54.66	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:55.23	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:55.57	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...
7:55.57	TELNET	172.22.125.116	12.0.1.28	55	8489	23	Telnet Data ...

Details of packet 2371:

- = Urgent: Not set
-1... = Acknowledgment: Set
-1... = Push: Set
-0.. = Reset: Not set
-0.. = Syn: Not set
-0.. = Fin: Not set
- Window size value: 63932
- [Calculated window size: 63932]
- [Window size scaling factor: -2 (no window scaling used)]
- Checksum: 0xf6ac [validation disabled]
- [Good Checksum: False]
- [Bad Checksum: False]
- Urgent pointer: 0
- [SEQ/ACK analysis]
- [RTT: 0.243335000 seconds]
- [Bytes in flight: 1]
- Telnet
- Data: r

Raw packet data (hex):

```

0000 00 1f 12 b9 ff f0 90 a4 de b6 f5 b8 08
0010 00 29 61 38 40 00 40 06 a2 f0 ac 16 7d
0020 01 1c 21 29 00 17 fa 99 9a f0 62 74 fd
0030 f9 bc f6 ac 00 00 72

```

Details of packet 2372:

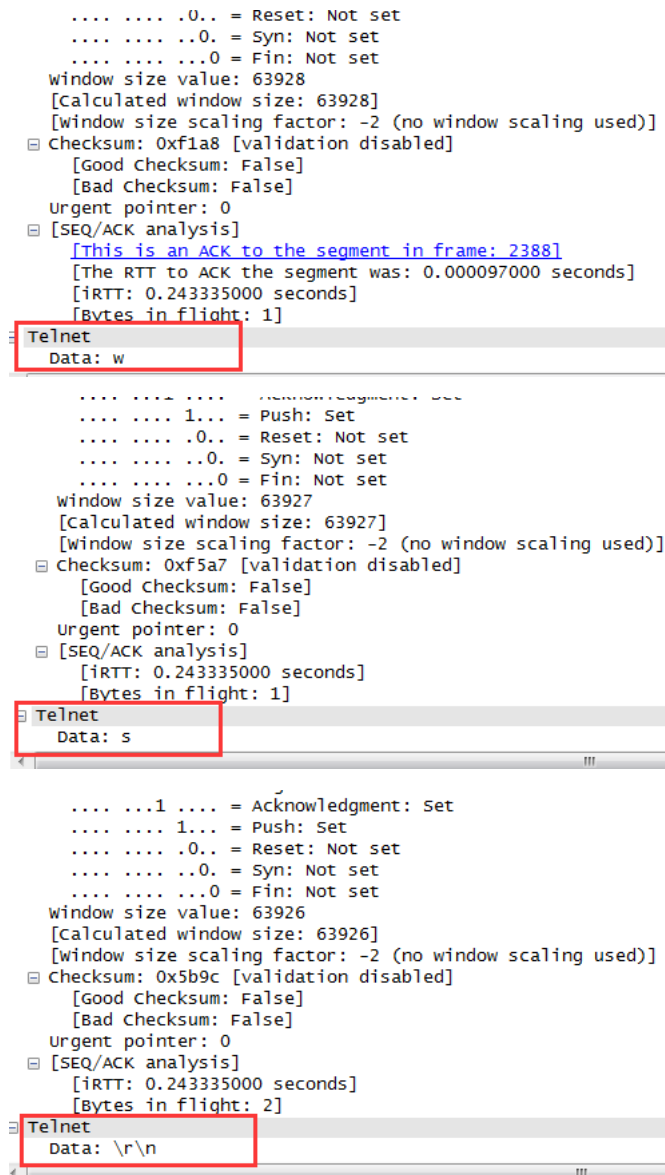
-0.. = Urgent: Not set
-1... = Acknowledgment: Set
-1... = Push: Set
-0.. = Reset: Not set
-0.. = Syn: Not set
-0.. = Fin: Not set
- Window size value: 63931
- [Calculated window size: 63931]
- [Window size scaling factor: -2 (no window scaling used)]
- Checksum: 0xf2ab [validation disabled]
- [Good Checksum: False]
- [Bad Checksum: False]
- Urgent pointer: 0
- [SEQ/ACK analysis]
- [RTT: 0.243335000 seconds]
- [Bytes in flight: 1]
- Telnet
- Data: v

Details of packet 2373:

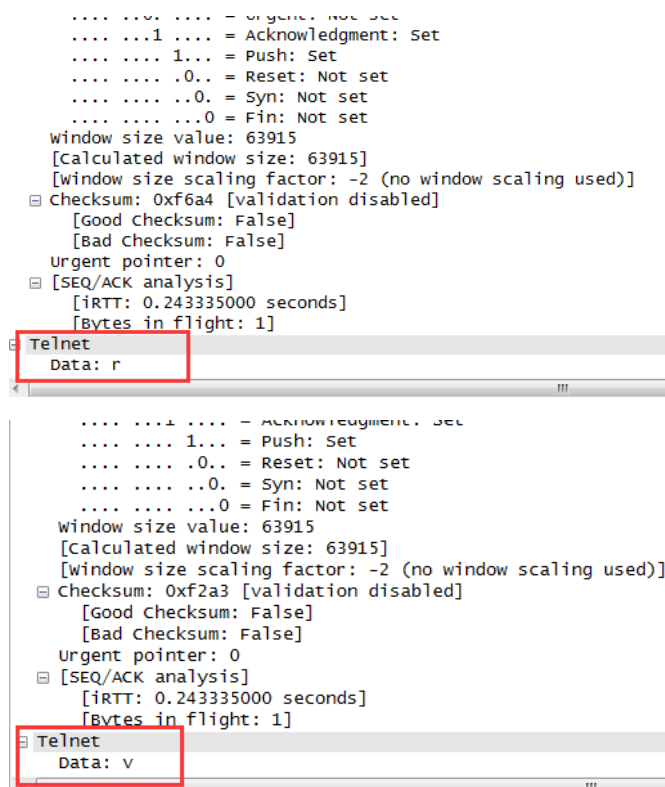
-1... = Push: Set
-0.. = Reset: Not set
-0.. = Syn: Not set
-0.. = Fin: Not set
- Window size value: 63930
- [Calculated window size: 63930]
- [Window size scaling factor: -2 (no window scaling used)]
- Checksum: 0xffaa [validation disabled]
- [Good Checksum: False]
- [Bad Checksum: False]
- Urgent pointer: 0
- [SEQ/ACK analysis]
- [This is an ACK to the segment in frame: 2371]
- [The RTT to ACK the segment was: 0.105290000 seconds]
- [RTT: 0.243335000 seconds]
- [Bytes in flight: 1]
- Telnet
- Data: i

Details of packet 2374:

-1... = Push: Set
-0.. = Reset: Not set
-0.. = Syn: Not set
-0.. = Fin: Not set
- Window size value: 63929
- [Calculated window size: 63929]
- [Window size scaling factor: -2 (no window scaling used)]
- Checksum: 0x03aa [validation disabled]
- [Good Checksum: False]
- [Bad Checksum: False]
- Urgent pointer: 0
- [SEQ/ACK analysis]
- [This is an ACK to the segment in frame: 2375]
- [The RTT to ACK the segment was: 0.132298000 seconds]
- [RTT: 0.243335000 seconds]
- [Bytes in flight: 1]
- Telnet
- Data: e



可见，将这些字符连起来就可以得到reviews,也就是用户名。而且最后传送的'\r\n'还可以推断作者用的是windows系统，因为linux下的换行是'\n',而windows下是'\r\n'。密码部分只上传部分截图：



关于可靠传输也找到了一个好点的例子，没错，就是telnet，TCP通过每个数据包的seq序列号+len长度都等于下个数据包的seq序列号，如果不等说明中间丢了比特，下次会从新的seq位开始传一定长度的数据。telnet每次好像都传1比特，传2比特的是'\n\n'字符。

```

# Frame 2445: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
# Ethernet II, Src: WistronN_b6:f5:b8 (172.22.125.116), Dst: JuniperN_b9:ff:f0 (12.0.1.28)
# Internet Protocol Version 4, Src: 172.22.125.116 (172.22.125.116), Dst: 12.0.1.28 (12.0.1.28)
# Transmission Control Protocol, Src Port: 8489 (8489), Dst Port: 23 (23), Seq: 70, Ack: 1825, Len: 1
  Source Port: 8489 (8489)
  Destination Port: 23 (23)
  [Stream index: 5]
  [TCP Segment Len: 1]
  Sequence number: 70 (relative sequence number)
  [Next sequence number: 71 (relative sequence number)]
  Acknowledgment number: 1825 (relative ack number)
  Header Length: 20 bytes
  0000 0001 1000 = Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    .... 0... = Congestion window reduced (CWR): Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    1       = Acknowledgment: Set

# Frame 2457: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
# Ethernet II, Src: WistronN_b6:f5:b8 (172.22.125.116), Dst: JuniperN_b9:ff:f0 (12.0.1.28)
# Internet Protocol Version 4, Src: 172.22.125.116 (172.22.125.116), Dst: 12.0.1.28 (12.0.1.28)
# Transmission Control Protocol, Src Port: 8489 (8489), Dst Port: 23 (23), Seq: 71, Ack: 1826, Len: 2
  Source Port: 8489 (8489)
  Destination Port: 23 (23)
  [Stream index: 5]
  [TCP Segment Len: 2]
  Sequence number: 71 (relative sequence number)
  [Next sequence number: 73 (relative sequence number)]
  Acknowledgment number: 1826 (relative ack number)
  Header Length: 20 bytes
  0000 0001 1000 = Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    .... 0... = Congestion window reduced (CWR): Not set
    .... .0.. = ECN-Echo: Not set

# Frame 2515: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
# Ethernet II, Src: WistronN_b6:f5:b8 (172.22.125.116), Dst: JuniperN_b9:ff:f0 (12.0.1.28)
# Internet Protocol Version 4, Src: 172.22.125.116 (172.22.125.116), Dst: 12.0.1.28 (12.0.1.28)
# Transmission Control Protocol, Src Port: 8489 (8489), Dst Port: 23 (23), Seq: 73, Ack: 1837, Len: 1
  Source Port: 8489 (8489)
  Destination Port: 23 (23)
  [Stream index: 5]
  [TCP Segment Len: 1]
  Sequence number: 73 (relative sequence number)
  [Next sequence number: 74 (relative sequence number)]
  Acknowledgment number: 1837 (relative ack number)
  Header Length: 20 bytes
  0000 0001 1000 = Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    .... 0... = Congestion window reduced (CWR): Not set
    .... .0.. = ECN-Echo: Not set

# Frame 2532: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
# Ethernet II, Src: WistronN_b6:f5:b8 (172.22.125.116), Dst: JuniperN_b9:ff:f0 (12.0.1.28)
# Internet Protocol Version 4, Src: 172.22.125.116 (172.22.125.116), Dst: 12.0.1.28 (12.0.1.28)
# Transmission Control Protocol, Src Port: 8489 (8489), Dst Port: 23 (23), Seq: 74, Ack: 1837, Len: 1
  Source Port: 8489 (8489)
  Destination Port: 23 (23)
  [Stream index: 5]
  [TCP Segment Len: 1]
  Sequence number: 74 (relative sequence number)
  [Next sequence number: 75 (relative sequence number)]
  Acknowledgment number: 1837 (relative ack number)
  Header Length: 20 bytes
  0000 0001 1000 = Flags: 0x018 (PSH, ACK)

```

好文要顶

关注我

收藏该文



HarrisonZhou

关注 - 49

粉丝 - 4

+加关注

0

推荐

0

反对

« 上一篇: Tomcat8.0.21登录时忘记用户名和密码

» 下一篇: vs 2013 Express 无法启动程序xxx.exe,系统找不到指定文件

posted on 2015-10-12 02:58 HarrisonZhou 阅读(33518) 评论(0) 编辑 收藏