实验一 基本网络工具集使用和协议数据单元(PDU)观测

161130118 尹浚宇

**实验目的**

熟悉Linux系统中最基本的网络工具集合(包括ifconfig, route, wireshark等)的使用;

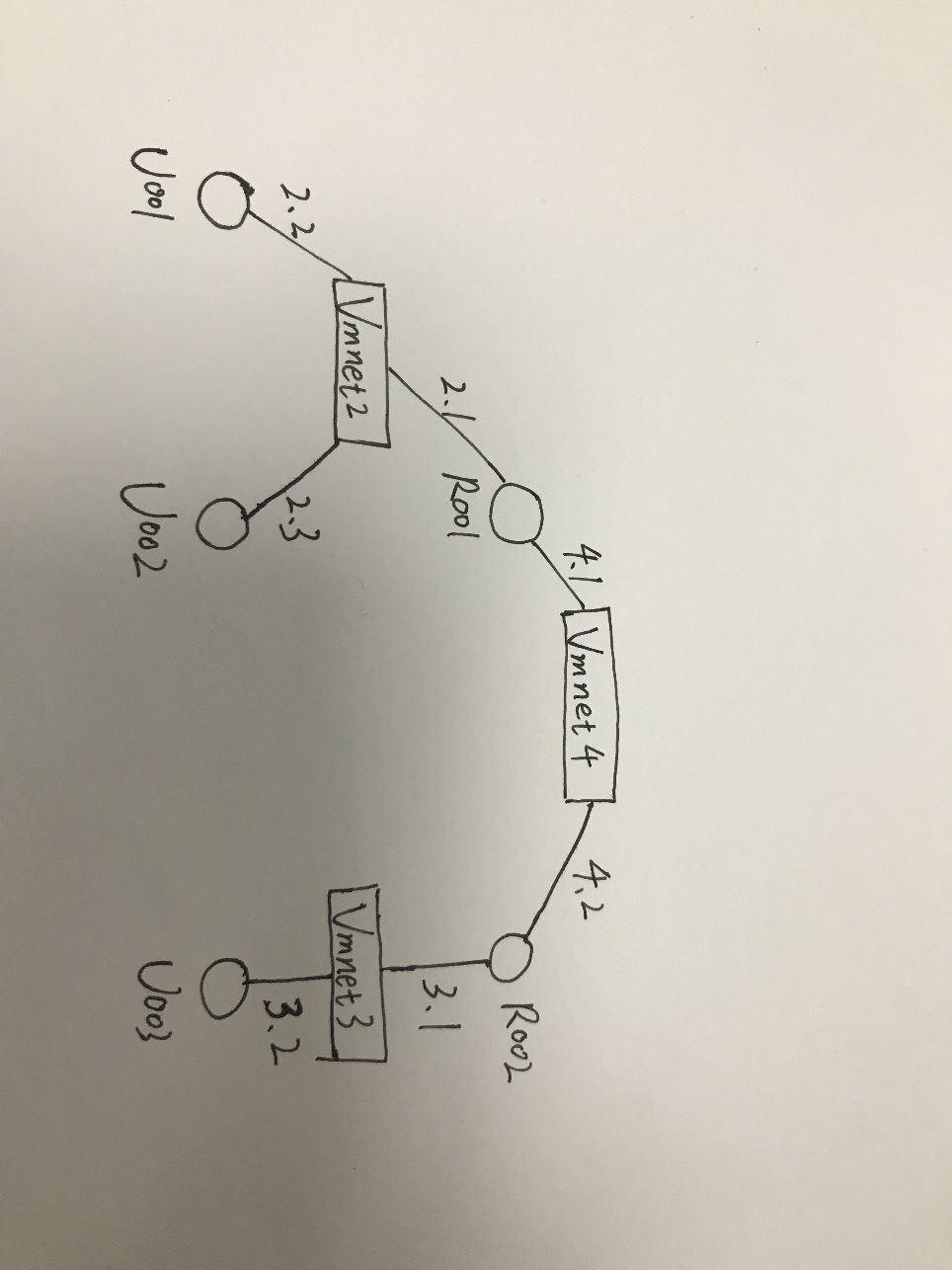
能够熟练观察和初步分析协议PDU的内容.

**网络拓扑配置**

附表一如下

|  |  |  |  |
| --- | --- | --- | --- |
| 节点名 | 虚拟设备名 | ip | netmask |
| Router0 | R001 | ens33:192.168.2.1 | 255.255.255.0 |
| ens38: 192.168.4.1 |
| Router1 | R002 | ens33: 192.168.3.1 | 255.255.255.0 |
| ens38: 192.168.4.2 |
| PC1 | U001 | ens33: 192.168.2.2 | 255.255.255.0 |
| PC2 | U002 | ens33: 192.168.2.3 | 255.255.255.0 |
| PC3 | U003 | ens33: 192.168.3.2 | 255.255.255.0 |

绘图如下:



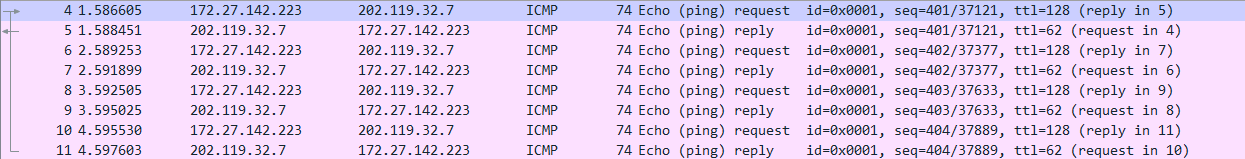
**路由规则配置**

在R001终端输入命令: sudo ip route add 192.168.3.0/24 via 192.168.4.2;

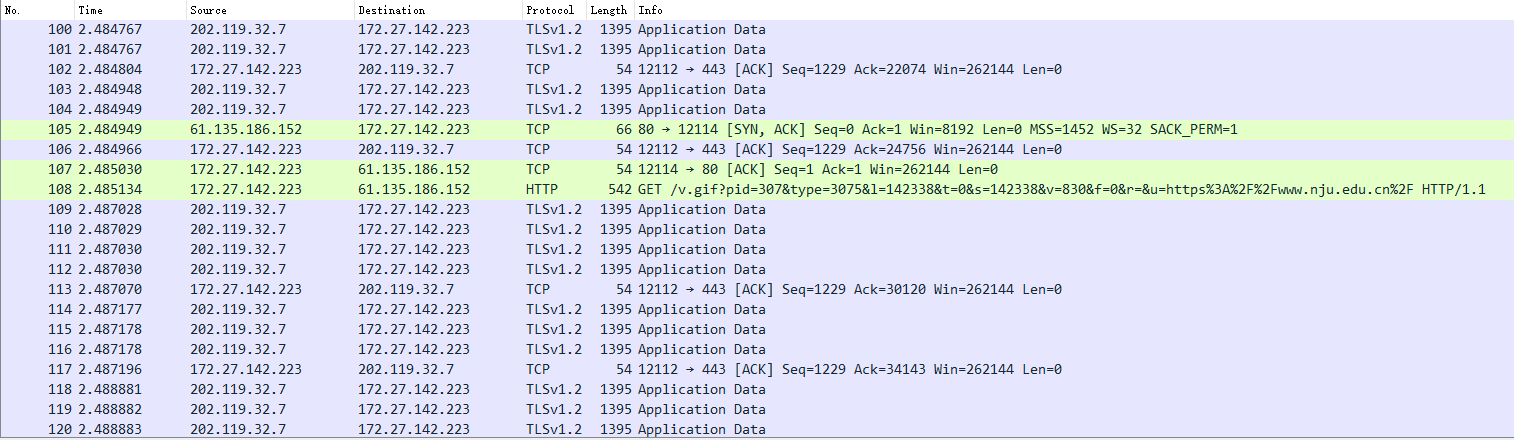
在R002终端输入命令: sudo ip route add 192.168.2.0/24 via 192.168.4.1

**数据包截图**

通过ping抓取的数据包:



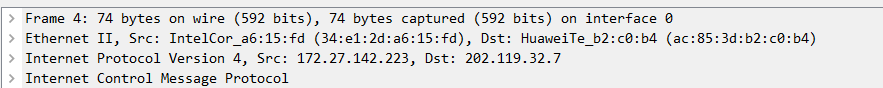
通过浏览器访问抓取的数据包:



**协议报文分析**

**对ping抓取的数据包的分析:**

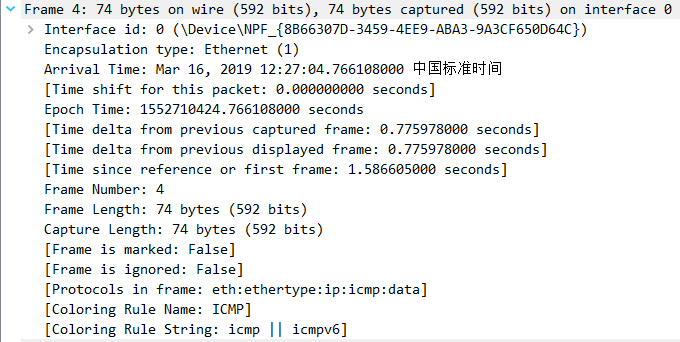
观察到捕获的8个数据包可分为4组, 每组2个数据包, 因为这8个数据包内容相近, 故选择第一个进行分析.



首先由上图可以观察到该数据包有4行内容, 第1行为物理层的数据帧概况, 第2层为数据链路层以太网帧头部信息, 第3层为互联网层IP包头部信息, 第4层为ICMP询问报文.

下面对其进行逐一分析:

物理层数据帧概况:



第1行说明这是4号帧, 线路上有74字节, 实际通过0号接口捕获了74字节;

第2行说明接口ID是0;

第3行说明封装类型是以太网;

第4行说明捕获日期和时间;

第5行说明该包的时移为0;

第6行说明了epoch time, epoch time是网络中一种被广泛使用的计时方法, 它代表了从标准世界时1970年1月1日0时0分0秒起到现在的总秒数, 不包括闰秒;

第7行说明此包与前一被捕获包的时间间隔;

第8行说明此包与前一被展示包的时间间隔;

第9行说明此包与第一帧的时间间隔;

第10行说明帧序号;

第11行说明帧长度;

第12行说明捕获长度;

第13行说明帧是否被标记;

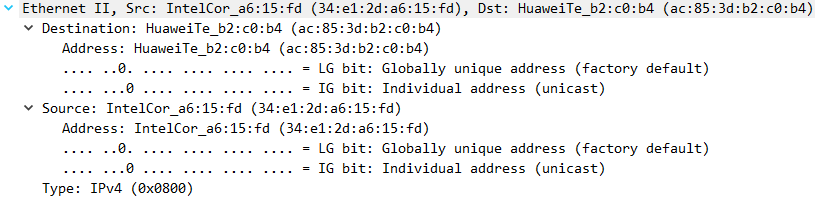
第14行说明帧是否被忽略;

第15行说明帧内封装的协议层次结构;

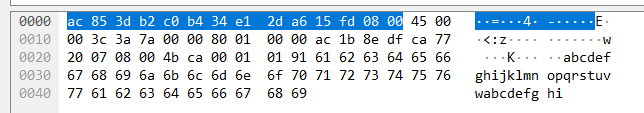
第16行说明着色标记的协议名称;

第17行说明着色规则显示的字符串.

数据链路层以太网帧头部信息:



由物理层数据帧概况可知, 该请求报文共74字节, 其中前14个字节为以太网报文的一部分 (由于网卡不会将其整个以太网帧格式数据提供出来, 因此Wireshark只能捕捉到一部分以太网的数据), 分别为本机的MAC地址和路由器的MAC地址, 对应字节如下图所示.

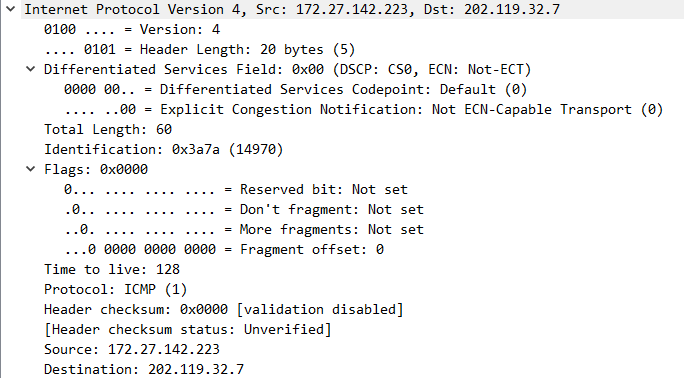


从抓包结果可以得出, 目标MAC地址为ac 85 3d b2 c0 b4, 本机MAC地址为34 e1 2d a6 15 fd.

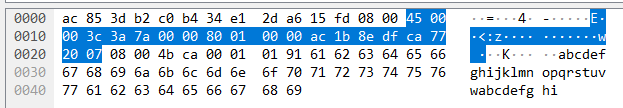
MAC地址的第1字节的最后2个bit位: IG位(individual/group, 第8个bit位)为0表示这是一个单播MAC地址(unicast MAC address), LG位(local/global，第7个bit位)为0表示这是一个厂家出厂默认的MAC地址, 为1时表示这是用户自己设置的MAC地址.

紧接着源MAC地址的是以太网封装的数据格式类型, 这里的值为0x0800, 表示接下来的数据是IPv4报文格式的数据.

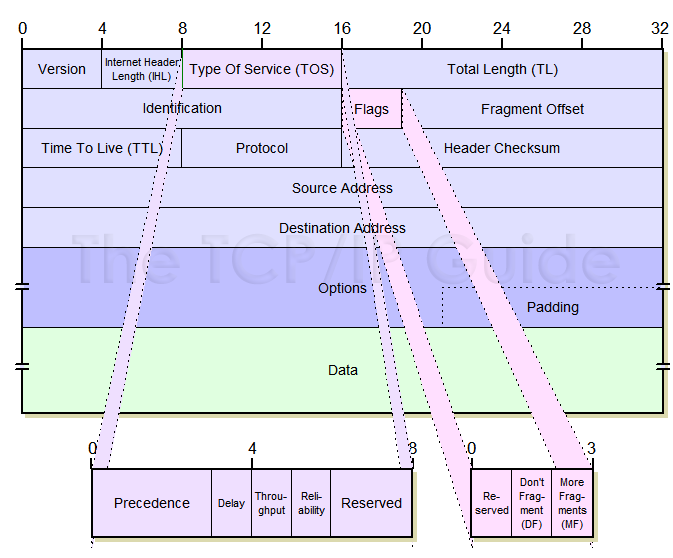
互联网层IP包头部信息:



接下来的20个字节为IP报文, 如下图所示.



ip报文格式如下:



ip报文中的第一个字节为0x45, 前一半(即0100)表示版本号为4, 后一半表示报文长度为5(此值表示报文长度为5个32位二进制数, 即5\*32/8=20个字节).

ip报文的第2字节为服务类型, 其值为0x00, 该字段声明了数据报被网络系统传输时被怎样处理. 若全为0则表示一般服务.

ip报文的第3-4字节为0x003c, 表示数据报总长度为60字节, 允许的最大长度为65535字节.

ip报文的第5-6字节为0x3a7a, 用来唯一地标识主机发送的每一份数据报. IP软件在存储器中维持一个计数器, 每产生一个数据段, 计数器就加1, 并将此值赋给标识字段. 当数据报由于长度超过网络的MTU而必须分片时, 这个标识字段的值就被复制到所有的数据报的标识字段中. 相同的标识字段的值使分片后各数据报片最后能正确的重装成为原来的数据报.

ip报文的第7-8字节为0x0000, 前3位中的2位用于管理fragmentation, 之后的比特表示片位移, 片偏移指出较长的分组在分片后, 某片在原分组的相对位置.

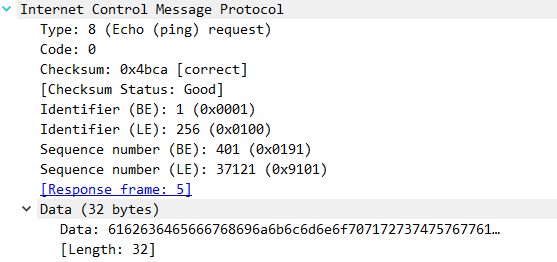
ip报文的第9字节为0x80, 值为128, 表示生存期(TTL), 用来设置数据报最多可以经过的路由器数.

ip报文的第10个字节为0x01, 表示IP层所封装的上层协议类型, 如ICMP(1), IGMP(2), TCP(6), UDP(17).

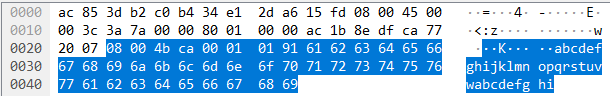
ip报文的第11-12字节为0x0000, 表示头部校验和, 其计算是将校验和的16bit位置0, 将整个IP头部字段按照16bit位来相加, 如果有进位, 则将其进位放在16bit位的最小位, 最后将加的结果取反即为校验和.

最后的13-16和17-20字节分别表示源IP地址(172.27.142.223)和目标IP地址(202.119.32.7).

ICMP询问报文:



剩余字节表示ICMP的报文信息, 如下图所示.



第1个字节0x08表示类型, 表明类型是Echo Request.

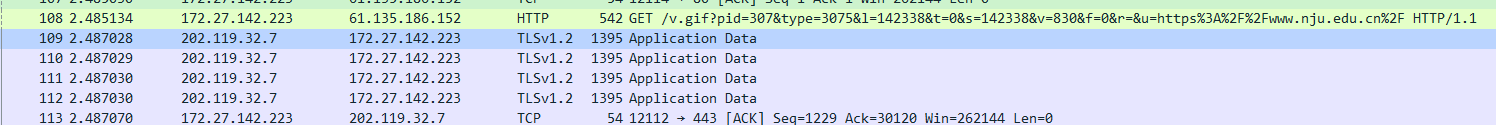
第2个字节0x00表示Code.

第3-4字节0x4bca表示检验和.

第5-6字节和第7-8字节分别表示标识码和序列码, 在报文里按大端方式储存.

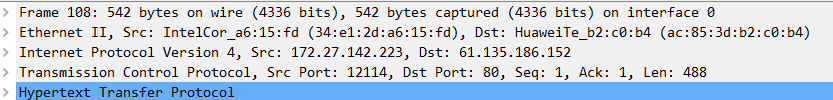
最后32字节就是我们发送的数据.

**对浏览器访问抓取的数据包的分析:**



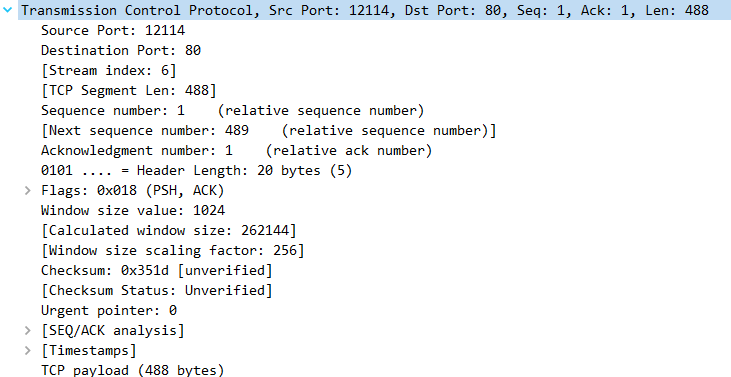
观察捕获的数据包, 发现有HTTP, TLSV1.2和TCP三种协议类型的数据包, 下面对其逐一分析.

HTTP类型:

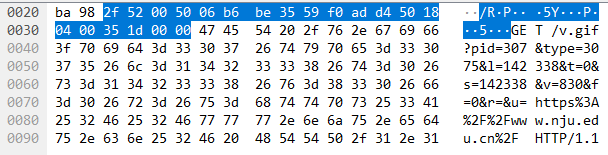


总体情况如上图所示, 该数据包大致分为5层, 前3层和ping方式抓取的数据包类似, 下面只对后2层进行分析.

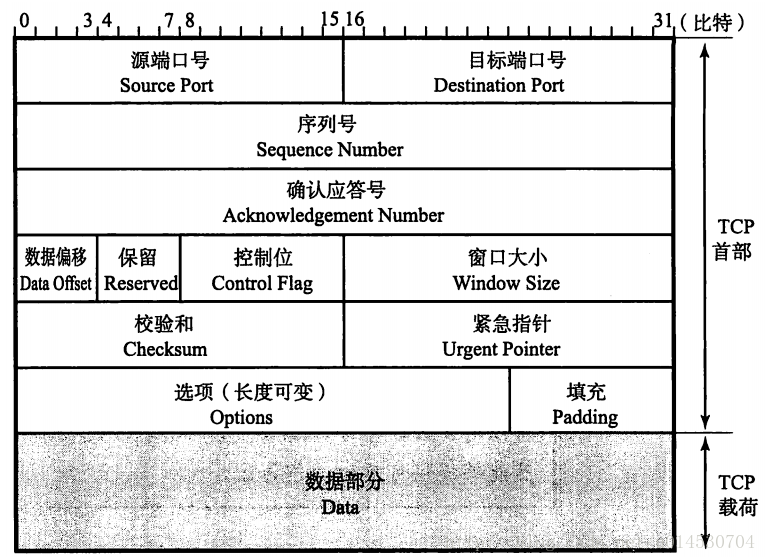
TCP协议首部字段:



上图是wireshark对该字段的解释, 该字段详细内容见下图.



TCP报文格式如下图.



网络之前的通信是不通主机之间的通信, 就windows系统而言一台主机有许多进程, 当我们发送数据时怎么知道要发送到对方主机那个进程里呢, 所以这就是端口号的作用.在TCP报文中包涵了源端口/目的端口, 源端口标识了发送进程, 目的端口标识了接收方进程. 在此报文中我们的源端口号是0x2f52 = 12114, 目的端口是0x0050 =80.

序列号用来标识从源端向目的端发送的数据字节流, 它表示在这个报文端中的第一个数据字节的顺序号, 序列号是32位的无符号类型, 当建立一个新的连接时, SYN标志为1.

确认号包涵了发送确认一端所期望收到的下一个顺序号. 因此确认号应当是上次成功接收到数据的序列号加1. 只有ACK标志为1时确认号字段才有效.

这里的偏移实际指的是TCP首部的长度, 它用来表明TCP首部中32bit字的数目. 此报文我们的偏移量是5, 所以我们的TCP报文头长为5 \* 4 = 20 bytes.

在TCP首部中有8个标志比特, 表示了“该TCP报文段”是干什么的.

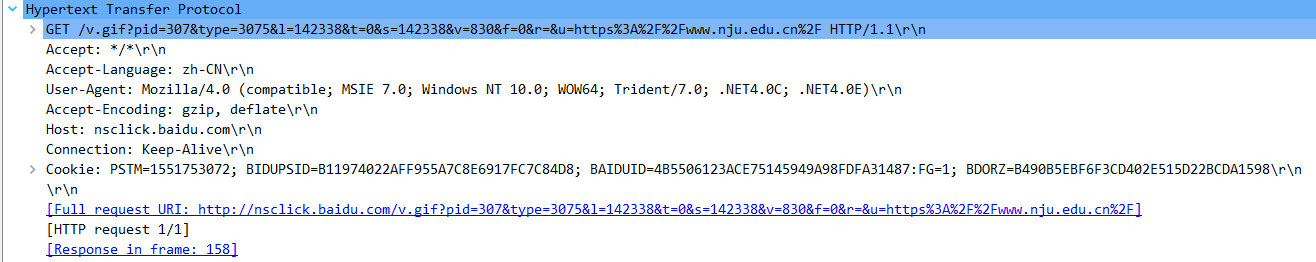
窗口大小表示源主机最大能接收多少字节.

校验和包含TCP首部和TCP数据段, 这是一个强制性的字段, 一定是由发送端计算和存储, 由接收端进行验证.

紧急指针只有当URG标志置为1时该字段才有效, 紧急指针是一个正的偏移量, 和序号字段中的值相加表示紧急数据最后一个字节的序号. TCP的紧急方式是发送端向另一段发送紧急数据的一种方式.

TCP选项是至少1个字节的可变长字段, 标识哪个选项有效.

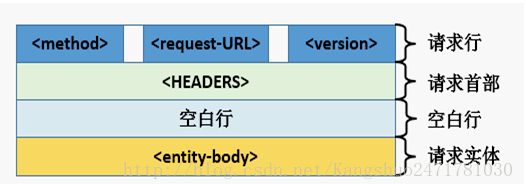
超文本传输协议:



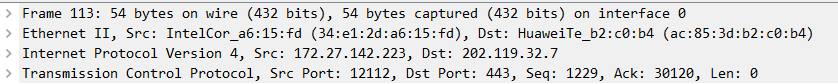
wireshark对该字段的解释如上图.

HTTP报文中存在着很多行的内容, 一般是由ASCII码串组成, 各字段长度是不确定的. HTTP的报文可分为两种: 请求报文与响应报文. 此处是一个请求报文.

请求报文的格式如下图所示.

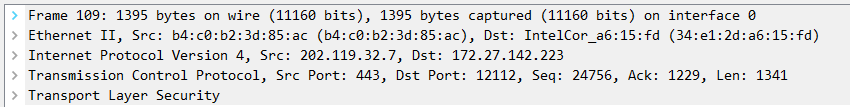


TCP类型:

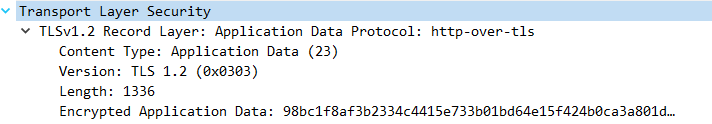


该数据包分为4层, 之前都已经分析过, 此处不在赘述.

TLSv1.2类型:



该数据包分5层, 前4层之前已经分析过, 这里只分析第5层.



TLS的设计目标是构建一个安全传输层(Transport Layer Security ), 在基于连接的传输层(如tcp)之上提供.

TLS加密了上层数据, 使得数据在中间被截获之后, 如果没有秘钥的话, 是没有用处的.