LAB 2–使用 Wireshark 分析 HTTP 协议

张煌昭，1400017707，元培学院

摘要—本次 Lab 要求用浏览器访问一个网站，用 Wireshark 采集一段时间的数据并对其分析。用示例（建议采用窗口截图，配以说明文字的形式）回答以下问题：TCP/IP 协议的 5 层结构；可分析出哪些协议，以及这些协议所占百分比；指出 Web 服务器的 IP 地址；应用 Wireshark 的显示过滤器，分析你的主机与 Web 服务器之间 HTTP 协议；执行 Statistics—>Flow Graph，显示浏览器与 HTTP 服务器之间的 HTTP 协议过程；执行 Statistics—>HTTP，显示浏览器与 HTTP 服务器之间请求与响应分组总数及占比；其他问题及发现。本次报告使用 Overleaf L^AT_EX 在线平台编写[1]。

I. TCP/IP 和 ISO-OSI 协议结构

开放式系统互联协议（下称 OSI）是由国际标准化组织（下称 ISO）推出的互联系统参考模型，该模型定义了不同设备互联的标准，可以作为网络设计和实现描述的基本框架。ISO-OSI 系统结构分为 7 层，自顶向下分别为应用层，表示层，会话层，传输层，网络层，数据链路层和物理层（见表 I）。

传输控制协议/因特网互联协议（下称 TCP/IP）不仅仅指 TCP 和 IP 两个协议，而是指整个因特网网 TCP/IP 协议族。该协议由阿帕网（下称 ARPA）首先参考使用，之后被作为计算机网络标准推广使用。TCP/IP 结构分为 5 层，自顶向下分别为应用层，传输层，网络层，链路层和物理层（见表 I）。

TCP/IP 协议栈结构基本包括 ISO-OSI 协议栈中的五层，而没有涉及表示层和会话层。下面对 TCP/IP 协议分层结构和 ISO-OSI 协议独立出的两层进行较为详细的叙述。

应用层包括一些列网络应用程序和它们的协议，一般应用层协议分布于多个端系统上，其上的信息分组称作报文（message）。常见的应用层协议包括：超文本传输协议（下称 HTTP），实现 Web 文档的请求和传送，固定使用 80 端口；文件传输协议（下称 FTP），实现两个端系统间的文件传送，固定使用 20H 和 21H 端口（20H 为数据端口，21H 为控制端口）；域名解析服务（DNS），提供域名和 IP 地址的映射转换，固定使用 53 端口。

传输层将应用层信息格式化为信息流，并提供传输，其上的信息分组称作报文段（segment）。因特网中有的运输层协议有：传输控制协议（下称 TCP），提供可靠的传输，其将长报文分为短报文，提供拥塞控制机制，最重要的是其需要信源和目的地握手建立连接；用户数据报文协议（下称 UDP），提供无连接的传输，不具有可靠性，也没有流量控制。

网络层将数据报（datagram）分组从一台主机移至另一台主机，其过程需要源主机提供目的地址。网络层协议最著名也是最重要的是 IP 协议，其定义了数据报中各个字段以及端主机和路由器如何作用于这些字段。此外网络层还包括一系列路由选择协议，将数据报根据路由从源传输到目的地。

链路层将分组从一个节点沿路径移动到下一个节点，这一过程中网络层将数据报下传给链路层，链路层将数据报传递后在下一个节点上传给网络层，其上的分组称为帧（frame）。链路层协议包括以太网协议，WLAN 协议和电缆接入网的 DOCSIS 协议。

物理层将数据以物理信号的方式进行传输，链路层的帧在物理层被一个一个 bit 地移动到下一个节点。其上协议多与传输媒介相关，例如以太网物理层协议包括双绞铜线协议和光纤协议等。

表示层是 ISO-OSI 中独立的一层，它使得应用程序得以解释交换数据的含义，具体包括数据压缩，数据加密和数据描述等。**会话层**也是 ISO-OSI 中独立的一层，它提供了数据交换定界和同步功能，具体包括建立检查点和恢复等。[2]

我认为表示层和会话层独立出来，可以使得网络应用程序的独立性更好，并且开发难度会降低很多，这一概念在数据库以及其他很多领域中广泛应用。目前 TCP/IP 协议下，表示层和会话层的功能必须由网络应

[1]本报告源码可通过以下 git 命令获得，
git clone https://git.overleaf.com/14683240nvqgfbfhcynw

[2]James F. Kurose, and Keith W. Ross. 计算机网络: 自顶向下方法（原书第六版）. 机械工业出版社, 2008. 34-37.

表 I
因特网协议层级结构及功能

IOS-OSI 结构	层级功能	TCP/IP 结构	TCP/IP 栈中协议
应用层	应用程序网络服务		
表示层	数据压缩加密和数据解释	应用层	HTTP, FTP, DMS, TELNET 等
会话层	数据交换定界和同步		
传输层	格式化信息流并传输	传输层	TCP, UDP
网络层	将分组在主机间转移	网络层	IP，路由选择协议等
链路层	将分组在节点间移动	链路层	以太网协议, WLAN 协议, DOCSIS 等
物理层	通过物理媒介以 bit 传送数据	物理层	双绞铜线协议, 光纤协议等

用程序开发人员进行设计和构建。

II. Wireshark 捕获 HTTP 消息

使用 Wireshark 工具 [3] 进行之后的实验，由于我使用的 DELL XPS-13 笔记本电脑只有 WLAN 网卡，而没有内置的以太网网卡，因此以下所有试验均在 WiFi 环境下进行。对 PKU 主页，Baidu 主页，MIT 主页，以及 CMU 主页访问进行实验，它们的部分特性如表 II所示。访问四个主页所捕获的数据见附件[4]。

表 II
实验网址及其特点

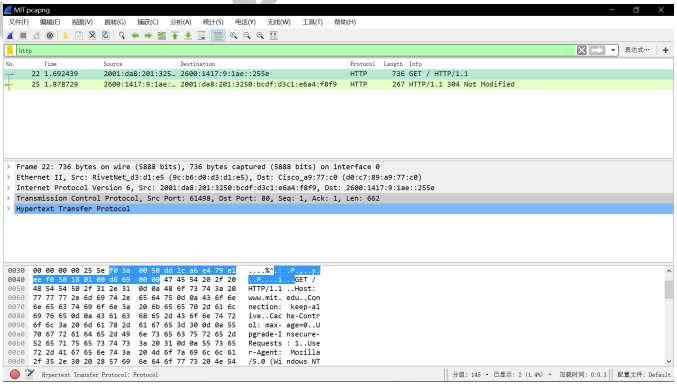
主页	网址	HTTP/HTTPS	国内/国际
PKU	www.pku.edu.cn	HTTP	国内
百度	www.baidu.com	HTTPS	国内
MIT	www.mit.edu	HTTP	国际
CMU	www.cmu.edu	HTTPS	国际

以访问 MIT 主页为例捕获 HTTP 请求，首先使用如下 ping 命令获取 MIT 主页的 IP 地址，为 [2600:1417:9:1ae::255e]。之后设置 Wireshark 过滤器为 http，并开始捕获分组。接下来使用 Chrome 浏览器访问 MIT 主页完成捕获，结果如图 1(a)所示。

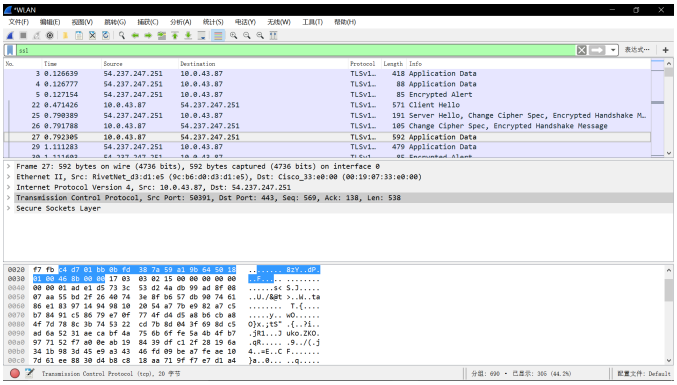
```
$> ping www.mit.edu

正在 Ping e9566.dscb.akamaiedge.net
[2600:1417:9:1ae::255e] 具有 32 字节的数据:
来自 2600:1417:9:1ae::255e 的回复: 时间=229ms
来自 2600:1417:9:1ae::255e 的回复: 时间=184ms
来自 2600:1417:9:1ae::255e 的回复: 时间=222ms
来自 2600:1417:9:1ae::255e 的回复: 时间=226ms

2600:1417:9:1ae::255e 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 184ms, 最长 = 229ms, 平均 = 215ms
```

同理，使用类似的方式，以访问百度主页为例捕获 HTTPS 请求，首先使用 ping 命令获取百度主页的 IP 地址，为 [119.75.213.61]。之后设置 Wireshark 过滤器为 ssl，并开始捕获分组。接下来使用 Chrome 浏览器访问百度主页完成捕获，结果如图 1(b)所示。

简单对比二者，发现 HTTP 请求仅有成对的请求-回应，而 HTTPS 请求更为复杂，需要用户端（下称 client）和服务器端（下称 server）通过秘钥等方式反复确认。对此的分析在第 V节进行。



(a) 访问 MIT 主页

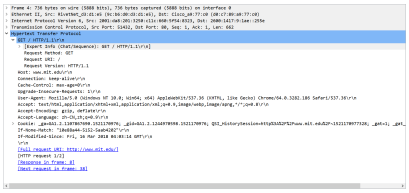


(b) 访问百度主页

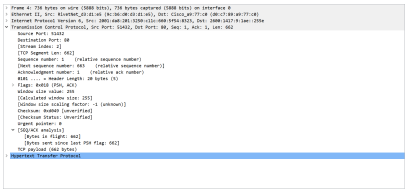
图 1. 使用 Wireshark 捕获 HTTP/HTTPS 消息。图 1(a)为 Wireshark 捕获 MIT 主页访问的 HTTP 请求结果图。图 1(b)为 Wireshark 捕获百度主页访问的 HTTPS（SSL）请求的结果图。

[3]使用 Wireshark stable release 2.4.5 64-bit 版，为目前官方最新开源版本，下载地址为 https://www.wireshark.org/#download

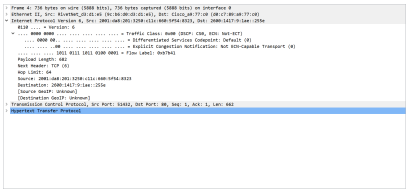
[4]捕获的数据存放地址为./wiresharkCapture/主页名称.pcapng



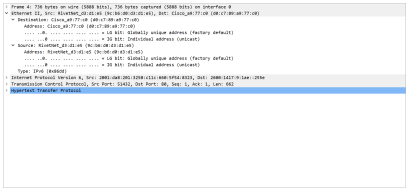
(a) 应用层报文



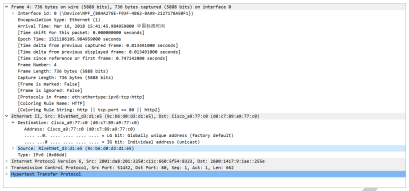
(b) 传输层报文段



(c) 网络层数据报



(d) 链路层数据帧



(e) 物理层数据帧

图 2. 分组结构，以 TCP/IP 五层结构，自顶向下分别为：图 2(a)为应用层 HTTP 请求报文，图 2(b)为传输层 TCP 报文段，图 2(c)为网络层 IPv6 数据报，图 2(d)为链路层以太网数据帧，图 2(e)为物理层数据帧。

III. Wireshark 分析 TCP/IP 协议

A. 分组结构

取访问 MIT 主页的捕获数据中的第四帧为例进行分析。根据 Wireshark 捕获的结果，分析该分组，可以得到物理层数据帧，链路层数据帧，网络层数据报，传输层报文段，以及应用层报文（见图 2）。

该分组应用层协议为 HTTP 协议，具体地，报文为一 HTTP GET 请求，对该 HTTP 请求进行解析。结果如下所示，发现其 HTTP 协议为 HTTP/1.1，该请求为两个相关的 HTTP 请求之一，其请求获取 HTML 文档和 PNG 图片等。

```
> Hypertext Transfer Protocol
    > GET / HTTP/1.1\r\n
    Host: www.mit.edu\r\n
    Connection: keep-alive\r\n
    Cache-Control: max-age=0\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/64.0.3282.186 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,
application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9\r\n

    ... ...

    \r\n
    [Full request URI: http://www.mit.edu/]
    [HTTP request 1/2]
    [Response in frame: 8]
    [Next request in frame: 38]
```

B. TCP/IP 协议及占比

取访问 MIT 主页的捕获数据，分析 TCP/IP 协议族中各个协议的占比。使用 Wireshark 中的“统计 > 协议分级”功能进行查看，该功能对捕获内容整体进行统计，最终可以获得各层网络的占比及速率等。表 III展示了对访问 MIT 主页的过程中各层协议的占比，其协议栏缩进格式表明了协议的包含关系或层级关系。

表 III
访问 MIT 主页捕获数据的 TCP/IP 各协议占比

协议	层级	按分组占比	分组	按字节占比	字节
Frame	物理层	100.0	128	100.0	49858
Ethernet	链路层	100.0	128	3.6	1792
IPv6	网络层	39.8	51	4.1	2040
UDP	传输层	18.8	24	0.4	192
QUIC	传输层	18.8	24	21.4	10687
TCP	传输层	18.8	24	34.3	17088
HTTP	应用层	3.1	4	33.3	16584
ICMPv6	网络层	2.3	3	0.4	192
IPv4	网络层	60.2	77	3.1	1640
TCP	传输层	60.2	77	32.7	16327
SSL	传输层	28.9	37	22.0	10991

表 III中 IPv6，UDP，TCP，HTTP，IPv4 都是较为熟悉的网络协议。下面对 QUIC，ICMPv6 和 SSL 进行简单的介绍：QUIC 为快速 UDP 网络连接协议，该协议在 UDP 的基础上，在两个端点间创建连接，且支持多路复用连接[5]；ICMPv6 为互联网控制消息协议，用于报告的错误消息和完成网络诊断功能[6]，该协议为

[5]Jim Roskind. QUIC, Quick UDP Internet Connections: multiplexed stream transport Over UDP. Google Docs. 2013

[6]Conta A, and Gupta M. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. 2006.

表 IV
访问 MIT 主页捕获数据的 HTTP 分组

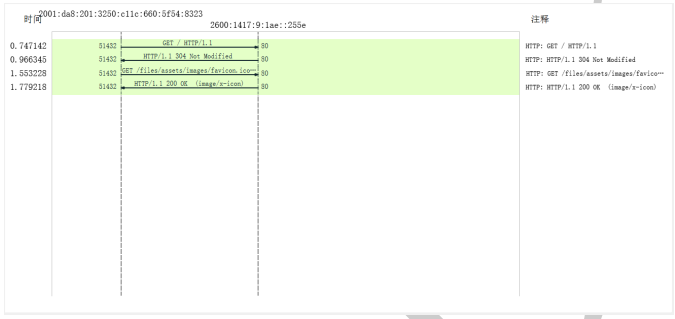
分组编号	时间	源 IP 地址	目标 IP 地址	长度	分组信息
4	0.75	2001:da8:201:3250:c11c:660:5f54:8323	2600:1417:9:1ae::255e	736	GET / HTTP/1.1
8	0.97	2600:1417:9:1ae::255e	2001:da8:201:3250:c11c:660:5f54:8323	267	HTTP/1.1 304 Not Modified
38	1.55	2001:da8:201:3250:c11c:660:5f54:8323	2600:1417:9:1ae::255e	647	GET /files/.../favicon.ico HTTP/1.1
54	1.78	2600:1417:9:1ae::255e	2001:da8:201:3250:c11c:660:5f54:8323	830	HTTP/1.1 200 OK (image/x-icon)

IPv6 体系中的一个组成部分；SSL 为安全套接层协议，和基于其构建的传输层安全协议（下称 TLS）一起，为网络通信提供安全性和数据完整性的安全协议，可以在传输层上对数据进行加密。

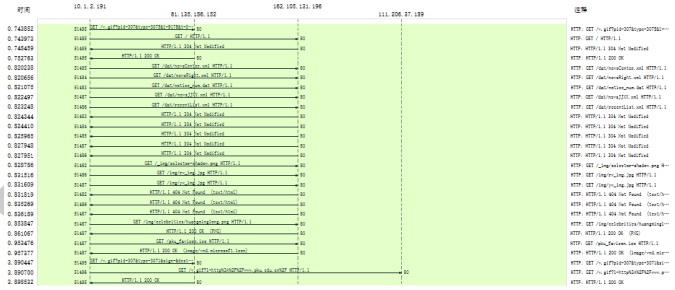
IV. 对 HTTP 的分析

对访问 MIT 主页的捕获数据中的 HTTP 协议进行分析。首先在 Wireshark 中的过滤器中设置过滤 http 协议，得到 4 个 HTTP 的分组，其基本信息如表 IV所示。

共有两组 HTTP 请求-回应对，第一个 GET 请求网页所有内容，收到服务器 304 回应，表示浏览器缓存中内容与服务器端内容一致，无需下载更新；之后第二个 GET 请求要求获取 favicon.ico 图标文件，收到服务器 200 回应，表示获取成功。通过检查第一个 GET 请求内容，发现内容为“If-Modified-Since: Fri, 16 Mar 2018 04:03:14 GMT”的一行，说明服务器通过时间戳来检查内容是否发生过更新，若不需要更新则服务器端回复“403 Not Modified”，否则将更新内容发送并回复“200 OK”。



(a) 流量图



(b) 流量图

图 3. 访问 MIT 主页的 HTTP 协议流量图。图 3(a)为访问 MIT 主页的流量图。其中显示了两对 HTTP 的 GET 请求-回应对：其中第一个 GET 请求全部内容，收到 403 回应，直接加载缓存内容；第二个 GET 请求一个 ico 文件，收到 200 回应，下载内容并将其加载。图 3(b)为访问 PKU 主页的流量图，图片截图不全，最下部还有一条由 111.206.37.198 指回 10.1.2.191 的 HTTP/1.1 200 OK 的箭头没有截入图中。

使用 Wireshark 中的“统计 > 流量图”来绘制访问 MIT 主页过程中，HTTP 浏览器客户端与服务器之间的协议过程，绘制的流量图如图 3(a)所示。该图所示的协议过程，与上述的分析描述一致。

使用 Wireshark 中的“统计 >HTTP> 分组计数器/请求/负载分配”对访问 PKU 主页的过程进行分析。该过程中共有 28 个 HTTP 分组，其对应的 HTTP 状态统计如图 5所示。

观察图 5(b)发现，整个过程中不止向 www.pku.edu.cn 发送请求，还向 nsclick.baidu.com 和 api.share.baidu.com 发送了请求，这两者均来自 baidu。猜测可能是 PKU 主页中需要加载来自 baidu 的内容，因此检查 www.pku.edu.cn 源代码（如图 4），发现一句从 baidu 获取资源的语句（图 4中红色框出），因此认为这证实了 PKU 主页需要从 baidu 加载某些资源的想法。当然也存在另一可能，即在实验过程中，由于没有严格控制，导致混入了其他网络进程的分组，使得数据被污染。

另外，对比 MIT 和 PKU 主页访问的 HTTP 分组数，MIT 主页访问总 HTTP 分组数为 4，PKU 主页访问总 HTTP 分组数为 28。考虑可能是因为 PKU 主页

图 4. www.pku.edu.cn 部分源代码，其中发现需要使用来自 baidu 的资源。

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Total HTTP Packets	28				0.0088	100%	0.2000	0.744
Other HTTP Packets	0				0.0000	0.00%	-	-
HTTP Response Packets	14				0.0044	50.00%	0.1000	0.748
???: broken	0				0.0000	0.00%	-	-
5xx: Server Error	0				0.0000	0.00%	-	-
4xx: Client Error	3				0.0009	21.43%	0.0300	0.832
404 Not Found	3				0.0009	100.00%	0.0300	0.832
3xx: Redirection	6				0.0019	42.86%	0.0600	0.748
304 Not Modified	6				0.0019	100.00%	0.0600	0.748
2xx: Success	5				0.0016	35.71%	0.0200	0.861
200 OK	5				0.0016	100.00%	0.0200	0.861
1xx: Informational	0				0.0000	0.00%	-	-
HTTP Request Packets	14				0.0044	50.00%	0.1000	0.744
GET	14				0.0044	100.00%	0.1000	0.744

(a) 分组计数

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
HTTP Requests by HTTP Host	14				0.0044	100%	0.1000	0.744
www.pku.edu.cn	11				0.0035	78.57%	0.0900	0.744
/pku_favicon.ico	1				0.0003	9.09%	0.0100	0.953
/img/yw_img.jpg	1				0.0003	9.09%	0.0100	0.832
/img/rw_img.jpg	1				0.0003	9.09%	0.0100	0.832
/img/celebrities/huangminglong.png	1				0.0003	9.09%	0.0100	0.854
/dat/recentList.xml	1				0.0003	9.09%	0.0100	0.823
/dat/notice_num.dat	1				0.0003	9.09%	0.0100	0.821
/dat/newsRight.xml	1				0.0003	9.09%	0.0100	0.821
/dat/newsIJXY.xml	1				0.0003	9.09%	0.0100	0.822
/dat/newsCenter.xml	1				0.0003	9.09%	0.0100	0.820
/_img/selector-shadow.png	1				0.0003	9.09%	0.0100	0.829
/	1				0.0003	9.09%	0.0100	0.744
nsclick.baidu.com	2				0.0006	14.29%	0.0100	0.744
/v.gif?pid=307&type=3075&l=9178&t=0&s=9178&v=6288&f=1000&r=&u=http...	1				0.0003	50.00%	0.0100	0.744
/v.gif?pid=307&type=3071&sign=&desturl=&linkid=jetn1b7om1u&apitype=1	1				0.0003	50.00%	0.0100	3.890
api.share.baidu.com	1				0.0003	7.14%	0.0100	3.891
/v.gif?l=http%3A%2F%2Fwww.pku.edu.cn%2F	1				0.0003	100.00%	0.0100	3.891

(b) 请求计数

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
HTTP Responses by Server Address	14				0.0044	100%	0.1000	0.748
61.135.186.152	2				0.0006	14.29%	0.0100	0.753
OK	2				0.0006	100.00%	0.0100	0.753
162.105.131.196	11				0.0035	78.57%	0.0900	0.748
OK	8				0.0025	72.73%	0.0600	0.748
KO	3				0.0009	27.27%	0.0300	0.832
111.206.37.189	1				0.0003	7.14%	0.0100	3.921
OK	1				0.0003	100.00%	0.0100	3.921
HTTP Requests by Server	14				0.0044	100%	0.1000	0.744
HTTP Requests by Server Address	14				0.0044	100.00%	0.1000	0.744
61.135.186.152	2				0.0006	14.29%	0.0100	0.744
nsclick.baidu.com	2				0.0006	100.00%	0.0100	0.744
162.105.131.196	11				0.0035	78.57%	0.0900	0.744
www.pku.edu.cn	11				0.0035	100.00%	0.0900	0.744
111.206.37.189	1				0.0003	7.14%	0.0100	3.891
api.share.baidu.com	1				0.0003	100.00%	0.0100	3.891
HTTP Requests by HTTP Host	14				0.0044	100.00%	0.1000	0.744
www.pku.edu.cn	11				0.0035	78.57%	0.0900	0.744
162.105.131.196	11				0.0035	100.00%	0.0900	0.744
nsclick.baidu.com	2				0.0006	14.29%	0.0100	0.744
61.135.186.152	2				0.0006	100.00%	0.0100	0.744
api.share.baidu.com	1				0.0003	7.14%	0.0100	3.891
111.206.37.189	1				0.0003	100.00%	0.0100	3.891

(c) 负载分配

图 5. 访问 PKU 主页的 HTTP 协议统计信息。5(a)统计了各个状态的 HTTP 分组的信息; 5(b)统计了对各个 IP 地址的 HTTP 请求的信息; 5(c)统计了 HTTP 分组的负载情况。

相较于 MIT 主页更为复杂，对比二者源码代码量，MIT 主页代码量为 281 行，PKU 主页代码量为 922 行。大代码量明显对应更为丰富的内容，而更为丰富的内容就必然需要更多次的 HTTP 请求。

V. 对 HTTPS 的分析

A. *HTTP* 协议的安全问题

HTTPS 与 HTTP 类似，但最大的区别在于 HTTP 明文传输报文，而 HTTPS 加密传输，这导致 HTTPS 的安全性远远强过 HTTP。以 PKU 一系列校园网站为例，很多网站包括主页均使用的是 HTTP 协议，因为明文传输网站内容一般情况下不会有很大的安全问题（被抓取/替换分组的情况仍然会有安全问题），而几乎

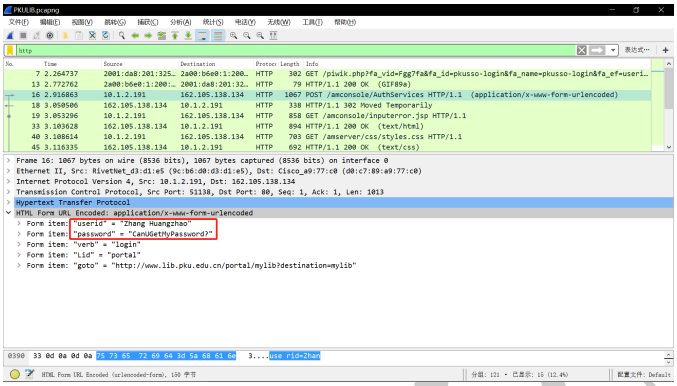


图 6. 捕获 PKU 图书馆的登录账号密码。利用 HTTP 明文传输的缺陷，直接捕获分组获取登录账号和密码。对密码中的问题，回复显然是“YesWeCan!”

所有需要密码的登录都接入 iaaa 登录系统，由 HTTPS 协议的登录系统确认身份后再转回原站。

下面通过实验说明 HTTP 会引起的的安全问题，发现北大图书馆登录页面[7]为 HTTP 协议，针对其进行实验。首先进入该页面，填写好账户和密码，准备捕获登录信息；之后开启 Wireshark，设置过滤器为 HTTP，开始捕获分组；在网站登录页面点击“登录”，查看 Wireshark 寻找带有登录信息的分组[8]。

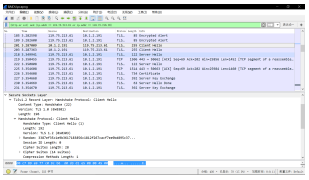
如图 6所示，通过 Wireshark 成功捕获带有账号和密码明文的 HTTP 分组。这说明 HTTP 协议是非常不安全的，在分组传递的过程中，任何一个环节都有可能将其捕获并获取其中的明文报文，包括极其重要的账号和密码等。同时，几乎绝大部分人在同一系统中的账号密码都是一样的，**这样就导致整个北大的网络系统都是不安全的——通过捕获图书馆登录账号和密码，即可获得几乎全北大所有人的北大通用的，甚至是完全通用的账号和密码信息！**

解决方案可以考虑将图书馆登录系统并入 iaaa 系统，使用 HTTPS 进行传输；或者号召全体师生更换不同的密码。显然后者是不现实的，因此更新网站协议非常重要，而且也非常紧急。

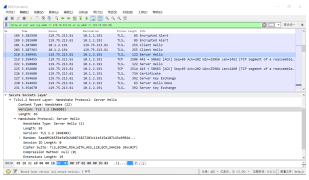
B. *HTTPS* 协议

为了解决上文提到的安全性问题，HTTPS 协议诞生。HTTPS 采用 HTTP+SSL 或 HTTP+TLS，即 HTTPS 使用 HTTP 通信，利用 SSL/TLS 来加密报文。

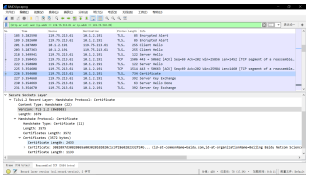
[7]http://www.lib.pku.edu.cn/portal/mylib
[8]捕获的数据存放路径为.wiresharkCapture/PKULIB.pcapng



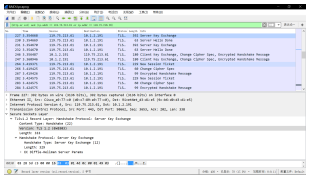
(a) Client Hello



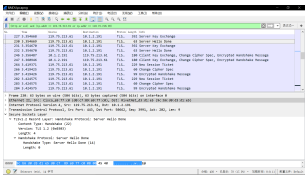
(b) Server Hello



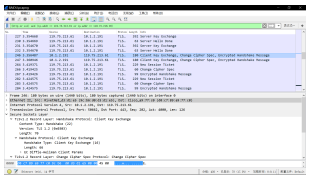
(c) Certificate



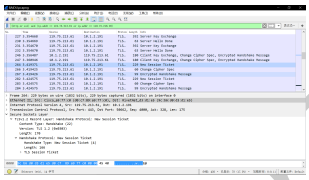
(d) Server Key Exchange



(e) Server Hello Done



(f) Client Key Exchange + Change Cipher Spec + Encrypted Handshake Message



(g) Server Key Exchange + Change Cipher Spec + Encrypted Handshake Message

图 7. 访问百度主页捕获的 HTTPS 分组。图 7(a)∼7(g)为整个 HTTPS 加密建立过程捕获的分组，整个加密通信建立的过程需要 client 和 server 多次确认，最终双方各持有一对对称秘钥中的一个进行加密的 HTTP 通信。

HTTPS 协议的通信和加密过程一般为：1. client 向 server 发送请求；2. server 返回自己的数字证书，该证书通过数字证书认证机构（下称 CA）公钥加密；3. client 使用 CA 公钥解密数字证书，如果有问题则说明具有风险，终止操作，否则继续进行；4. client 生成对称加密的随机秘钥，之后使用 CA 公钥对该秘钥进行加密，发送给 server；5. server 接受到加密的秘钥后，使用 CA 公钥对其进行解密；6. client 和 server 获得一对唯一的对称秘钥，利用该秘钥进行正常的加密通信。

这一过程当然也会存在安全问题，并且可以针对其进行非法信息的获取。比如，最为自然的方式是，劫持网关并伪造数字证书，劫持所有由 HTTP 转到 HTTPS 的请求，称为 client 和 server 的中介人，同 client 采取 HTTP 明文通信，之后同 server 采取正常的 HTTPS 加密通信，如图 8所示。如此操作便可以通过 HTTP 明文获取密码等关键信息，但暂时不在讨论之列，一般认为在协议正确配置后，HTTPS 是安全的（被劫持很多情况下是因为信任了伪造的数字证书）。

(a) 正常的 HTTPS 通信

(b) 被劫持的 HTTPS 通信

图 8. 劫持 HTTPS。正常的 HTTPS 通信如图 8(a)，client 与 server 各持有一对对称秘钥中的一个，直接加密通信；有一部分 HTTPS 由 HTTP 请求转化而来，在这一过程中可以被劫持，黑客通过劫持 client 和 server 间所有通信并伪造数字证书的方式获取秘钥。

C. 对 HTTPS 的分析

以访问百度主页的捕获数据，对 HTTPS 的过程进行分析。由于 HTTPS 为 SSL/TLS 加密后的 HTTP 通信，因此将过滤器设置为“http or ssl”，此外还在过滤器中设置了百度主页的 IP 地址。

第 201 帧，浏览器向百度主页 IP 地址服务器发送“Client Hello”，发起请求。第 218 帧，服务器回复“Server Hello”，同时发送加密的数字证书。第 226 帧，浏览器解密数字证书并通过验证，向服务器发送“Certificate”，确认证书有效，同时发送加密的对称秘钥。第 227 帧和第 230 帧，服务器回复“Server Key Exchange”和“Server Hello Done”，说明服务器获取并解密了解密对称秘钥，完成密钥交换。第 246 帧，浏览器发送“Client Key Exchange”，“Change Cipher Spec”和“Encrypted Handshake Message”，确认收到完成秘钥交换的信息，并发送加密信息检查秘钥。第 264 帧，第 266 帧和第 267 帧，服务器回复“New Session Ticket”，“Change Cipher Spec”和“Encryped Handshake Message”，回复加密消息，完成秘钥确认，完成加密通信建立。整个过程如图 7所示。

VI. 参考文献和资料

[1] James F. Kurose, and Keith W. Ross. 计算机网络: 自顶向下方法（原书第六版）. 机械工业出版社, 2008. 34-37.

[2] Jim Roskind. QUIC, Quick UDP Internet Connections: multiplexed stream transport Over UDP. Google Docs. 2013

[3] Conta A, and Gupta M. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. 2006.