

计算机网络第二次实验作业——Web服务器配置，HTTP报文捕获

计算机网络第二次实验作业——Web服务器配置，HTTP报文捕获

实验要求

Web服务器的搭建

通过Wireshark捕获与Web服务器的交互过程

网页访问概览

TCP

HTTP

连接的建立与断开

获取网页文字信息

获取图片信息

获取视频信息

在文本框输入信息并回显

总结

实验要求

- (1) 搭建Web服务器（自由选择系统），并制作简单Web页面，包含简单文本信息（至少包含专业、学号、姓名）。
- (2) 通过浏览器获取自己编写的Web页面，使用Wireshark捕获与Web服务器的交互过程，并进行简单分析说明。

Web服务器的搭建

- 系统：windows xp 虚拟机
- 软件：apache (phpnow)
- 虚拟机IP：192.168.141
- 端口：默认端口80
- 访问方式：主机浏览器访问虚拟机ip

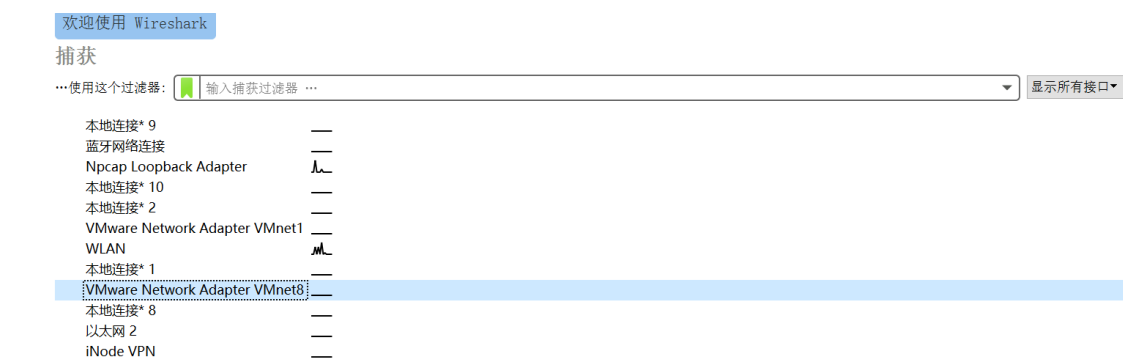
安装软件后，在安装路径 C:\phpnow\htdocs\ 下添加页面，本次实验所需的html文件将统一放置在 `netword` 子路径下

具体代码内容将在下一部分（数据包捕获）详述

通过Wireshark捕获与Web服务器的交互过程

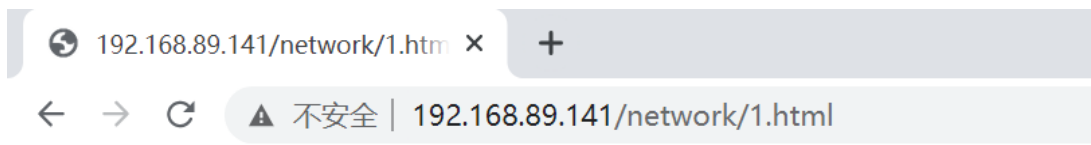
网页访问概览

打开Wireshark，选择需要捕获的网卡，这里由于是使用NAT模式下的虚拟机搭建的服务器，所以选用 VMnet8

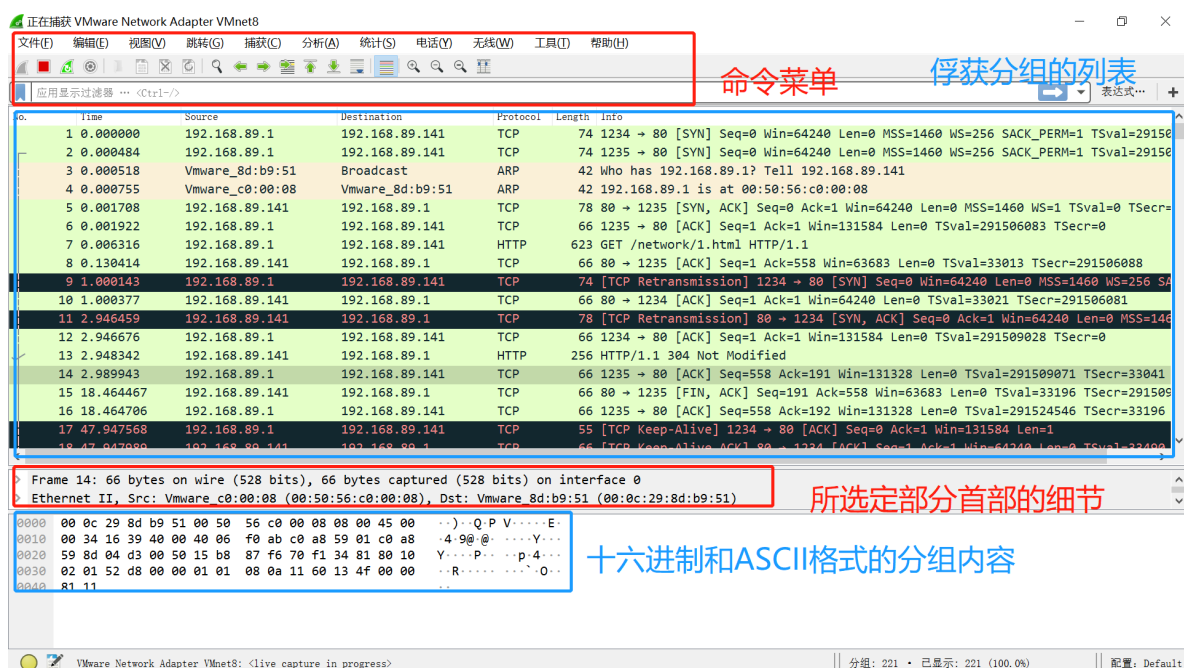


为了观察连接的建立和页面访问的基本过程，使用一个空的文件 1.html

访问

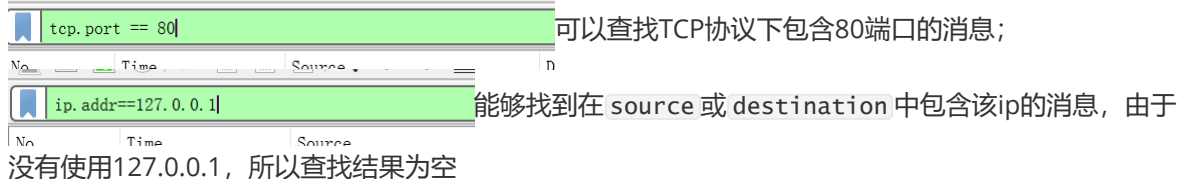


可以看到Wireshark捕获到的内容。



界面大致分为四个区：命令菜单区、俘获分组列表区、选定分组首部细节区、十六进制和ASCII格式分组内容区。

其中，命令菜单区的**应用显示过滤器**部分可以筛选显示的分组，例如



TCP

对于第一条消息，为TCP原型

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.89.1	192.168.89.141	TCP	66	3164 → 80 [FIN, ACK] Seq=1 Ack=1 Win=514 Len=0 TSval=327001868 TSecr=57607

消息分为4个部分

- > Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
- > Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
- > Internet Protocol Version 4, Src: 192.168.89.1, Dst: 192.168.89.141
- > Transmission Control Protocol, Src Port: 3164, Dst Port: 80, Seq: 1, Ack: 1, Len: 0

1. Frame, 指的是物理层的数据帧概况

- ▼ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 - > Interface id: 0 (\Device\NPF_{7F205D8E-42A7-46FB-BBBC-C102D4E54C72})
 - Encapsulation type: Ethernet (1)
 - Arrival Time: Nov 12, 2020 09:58:02.074467000 中国标准时间
 - [Time shift for this packet: 0.000000000 seconds]
 - Epoch Time: 1605146282.074467000 seconds
 - [Time delta from previous captured frame: 0.000000000 seconds]
 - [Time delta from previous displayed frame: 0.000000000 seconds]
 - [Time since reference or first frame: 0.000000000 seconds]
 - Frame Number: 1
 - Frame Length: 66 bytes (528 bits)
 - Capture Length: 66 bytes (528 bits)
 - [Frame is marked: False]
 - [Frame is ignored: False]
 - [Protocols in frame: eth:ethertype:ip:tcp]
 - [Coloring Rule Name: HTTP]
 - [Coloring Rule String: http || tcp.port == 80 || http2]

0000	00 0c 29 8d b9 51 00 50 56 c0 00 08 08 00 45 00	..).Q.P V....E.
0010	00 34 46 6b 40 00 40 06 c0 79 c0 a8 59 01 c0 a8	.4Fk@.@.y.Y...
0020	59 8d 0c 5c 00 50 3a c6 b3 e1 7c 7e b5 09 80 11	Y..\.P:.. ~....
0030	02 02 78 6d 00 00 01 01 08 0a 13 7d a7 0c 00 00	..xm.......}....
0040	e1 07	..

2. Ethernet II, 第0-13个字节, 表示数据层以太网帧头部信息, 包含目的地址、源地址和协议类型

- ▼ Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
 - ▼ Destination: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
 - Address: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
 -0. = LG bit: Globally unique address (factory default)
 -0 = IG bit: Individual address (unicast)
 - > Source: Vmware_c0:00:08 (00:50:56:c0:00:08)
 - Type: IPv4 (0x0800)

0000	00 0c 29 8d b9 51 00 50 56 c0 00 08 08 00 45 00	..).Q.P V....E.
0010	02 61 46 71 40 00 40 06 be 46 c0 a8 59 01 c0 a8	.aFq@.@.F.Y...

源地址为00:50:56:c0:00:08

0000 00 0c 29 8d b9 51 00 50 56 c0 00 08 08 00 45 00 在数据包的第6-11字节表示发送端的物理地址, 看到是主机网卡中的信息, 源ip信息与源网卡信息相对应, 可以看到是主机向“服务器”发送的网络请求

以太网适配器 VMware Network Adapter VMnet8:

```

连接特定的 DNS 后缀 . . . . . :
描述. . . . . : VMware Virtual Ethernet Adapter for VMnet8
物理地址. . . . . : 00-50-56-C0-00-08
DHCP 已启用. . . . . : 否
自动配置已启用. . . . . : 是
本地链接 IPv6 地址. . . . . : fe80::587a:cee0:6981:e706%12(首选)
IPv4 地址. . . . . : 192.168.89.1(首选)

```

目的地址为00:0c:29:8d:b9:51

0000	00 0c 29 8d b9 51 00 50 56 c0 00 08 08 00 45 00
------	-------------------------------------------------

在数据包的前六个字节, 是虚拟机对应的网卡, 表示消息发送至虚拟机网卡

```

Ethernet adapter 本地连接:

    Connection-specific DNS Suffix  . : localdomain
    Description . . . . . : VMware Accelerated AMD PCNet Adapter
    Physical Address. . . . . : 00-0C-29-8D-B9-51
    Dhcp Enabled. . . . . : yes
    Autoconfiguration Enabled . . . . : Yes
    IP Address. . . . . : 192.168.89.141

```

协议类型为IPV4，用0x0800表示

Type: IPv4 (0x0800)															
000	00	0c	29	8d	b9	51	00	50	56	c0	00	08	08	00	45 00

整体格式符合定义

```

1 #pragma pack(1)//以1byte方式对齐
2 typedef struct FrameHeader_t { //帧首部
3     BYTE DesMAC[6]; //6字节目的地址
4     BYTE SrcMAC[6]; //6字节源地址
5     WORD FrameType; //帧类型，解析时注意大小端转换
6 }frameHeader_t;

```

3. Internet Protocol Version 4，第14-33个字节，互联网层IP包头部信息

Internet Protocol Version 4, Src: 192.168.89.1, Dst: 192.168.89.141															
0100 = Version: 4															
.... 0101 = Header Length: 20 bytes (5)															
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)															
Total Length: 52															
Identification: 0x466b (18027)															
Flags: 0x4000, Don't fragment															
Time to live: 64															
Protocol: TCP (6)															
Header checksum: 0xc079 [validation disabled]															
[Header checksum status: Unverified]															
Source: 192.168.89.1															
Destination: 192.168.89.141															
0000	00	0c	29	8d	b9	51	00	50	56	c0	00	08	08	00	45 00
0010	00	34	46	6b	40	00	40	06	c0	79	c0	a8	59	01	c0 a8
0020	59	8d	0c	5c	00	50	3a	c6	b3	e1	7c	7e	b5	09	80 11

对应内容为

```

1 typedef struct IPHeader_t { //IP首部
2     BYTE Ver_HLen;
3     BYTE TOS;
4     WORD TotalLen
5     WORD ID;
6     WORD Flag_Segment
7     BYTE TTL;
8     BYTE Protocol;
9     WORD Checksum; //校验和
10    ULONG SrcIP; //源ip地址
11    ULONG DstIP; //目的ip地址
12 }IPHeader_t;

```

例如源IP地址为192.168.89.1，以十六进制显示在消息中为c0.a8.59.01

Source: 192.168.89.1															
Destination: 192.168.89.141															
0000	00	0c	29	8d	b9	51	00	50	56	c0	00	08	08	00	45 00
0010	00	34	46	6b	40	00	40	06	c0	79	c0	a8	59	01	c0 a8
0020	59	8d	0c	5c	00	50	3a	c6	b3	e1	7c	7e	b5	09	80 11

4. Transmission Control Protocol, 第34个字节开始，传输层的数据段头部信息

<															
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0															
> Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)															
> Internet Protocol Version 4, Src: 192.168.89.1, Dst: 192.168.89.141															
> Transmission Control Protocol, Src Port: 3164, Dst Port: 80, Seq: 1, Ack: 1, Len: 0															
0000	00	0c	29	8d	b9	51	00	50	56	c0	00	08	08	00	45 00
0010	00	34	46	6b	40	00	40	06	c0	79	c0	a8	59	01	c0 a8
0020	59	8d	0c	5c	00	50	3a	c6	b3	e1	7c	7e	b5	09	80 11
0030	02	02	78	6d	00	00	01	01	08	0a	13	7d	a7	0c	00 00
0040	e1	07													

可以看到，消息是由主机的3164端口发送至虚拟机的80端口

Transmission Control Protocol, Src Port: 3164, Dst Port: 80, Seq: 1, Ack: 1, Len: 0															
Source Port: 3164															
Destination Port: 80															
[Stream index: 0]															
[TCP Segment Len: 0]															
Sequence number: 1 (relative sequence number)															
[Next sequence number: 1 (relative sequence number)]															
Acknowledgment number: 1 (relative ack number)															
1000 = Header Length: 32 bytes (8)															
0000	00	0c	29	8d	b9	51	00	50	56	c0	00	08	08	00	45 00
0010	00	34	46	6b	40	00	40	06	c0	79	c0	a8	59	01	c0 a8
0020	59	8d	0c	5c	00	50	3a	c6	b3	e1	7c	7e	b5	09	80 11
0030	02	02	78	6d	00	00	01	01	08	0a	13	7d	a7	0c	00 00
0040	e1	07													

HTTP

选取一条HTTP消息

11	0.799305	192.168.89.1	192.168.89.141	HTTP	623	GET /network/1.html HTTP/1.1
----	----------	--------------	----------------	------	-----	------------------------------

消息格式为在原有TCP格式的基础上，增加超文本传输协议部分

Hypertext Transfer Protocol															
> GET /network/1.html HTTP/1.1\r\n															
Host: 192.168.89.141\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/86.0.4240.183 Safari/537.36\r\n															
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n															
Accept-Encoding: gzip, deflate\r\n															
Accept-Language: zh-CN,zh;q=0.9\r\n															
If-None-Match: "c448-0-b51e179a"\r\n															
If-Modified-Since: Wed, 11 Nov 2020 15:47:06 GMT\r\n															
\r\n															
[Full request URI: http://192.168.89.141/network/1.html]															
0040	00	00	47	45	54	20	2f	6e	65	74	77	6f	72	6b	2f 31
0050	2e	68	74	6d	6c	20	48	54	54	50	2f	31	2e	31	0d 0a
0060	48	6f	73	74	3a	20	31	39	32	2e	31	36	38	2e	38 39
0070	2e	31	34	31	0d	0a	43	6f	6e	6e	65	63	74	69	6f 6e

可以看到十六进制的消息和对应的字符

连接的建立与断开

- 找到IP对应的mac地址

```
Vmware_8d:b9:51 Broadcast ARP 42 Who has 192.168.89.1? Tell 192.168.89.141
```

broadcast为设置目的地址为全1（十六进制下的全f），类型为ARP，用0x0806表示

```
> Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▼ Ethernet II, Src: Vmware_8d:b9:51 (00:0c:29:8d:b9:51), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
    Type: ARP (0x0806)
▼ Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 0c 29 8d b9 51 08 06 00 01 .....Q.....
0010 08 00 06 04 00 01 00 0c 29 8d b9 51 c0 a8 59 8d .....Q..Y.
0020 00 00 00 00 00 00 c0 a8 59 01 .....Y.
```

目的地址全零表示未知地址

```
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
  Sender IP address: 192.168.89.141
  Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.89.1
```

```
0000 ff ff ff ff ff ff 00 0c 29 8d b9 51 08 06 00 01 .....Q.....
0010 08 00 06 04 00 01 00 0c 29 8d b9 51 c0 a8 59 8d .....Q..Y.
0020 00 00 00 00 00 00 c0 a8 59 01 .....Y.
```

```
Vmware_c0:00:08 Vmware_8d:b9:51 ARP 42 192.168.89.1 is at 00:50:56:c0:00:08
```

主机向虚拟机回复ip地址对应的MAC

```
> Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
▼ Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: Vmware_c0:00:08 (00:50:56:c0:00:08)
  Sender IP address: 192.168.89.1
  Target MAC address: Vmware_8d:b9:51 (00:0c:29:8d:b9:51)
  Target IP address: 192.168.89.141
```

```
0000 00 0c 29 8d b9 51 00 50 56 c0 00 08 08 06 00 01 .....Q.PV.....
0010 08 00 06 04 00 02 00 50 56 c0 00 08 c0 a8 59 01 .....P.V.....Y.
0020 00 0c 29 8d b9 51 c0 a8 59 8d .....Q..Y.
```

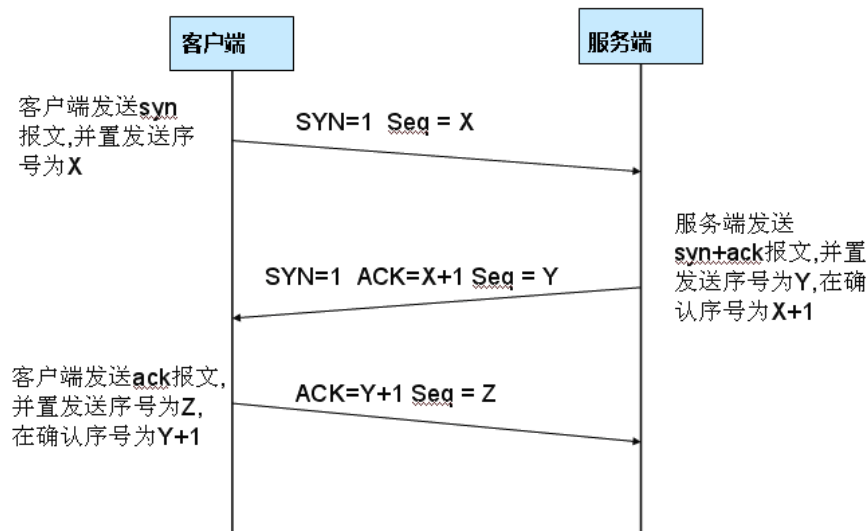
• 三次握手

1	0.000000	192.168.89.1	192.168.89.141	TCP	74	11536 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=3537
2	0.000214	192.168.89.141	192.168.89.1	TCP	78	80 → 11536 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 WS=1 TSval=0 TSecr
3	0.000260	192.168.89.1	192.168.89.141	TCP	74	11537 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=3537
4	0.000368	192.168.89.1	192.168.89.141	TCP	66	11536 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=353795537 TSecr=0
5	0.000512	192.168.89.141	192.168.89.1	TCP	78	80 → 11537 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 WS=1 TSval=0 TSecr
6	0.000794	192.168.89.1	192.168.89.141	TCP	66	11537 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=353795537 TSecr=0
7	0.007746	192.168.89.1	192.168.89.141	HTTP	597	GET /network/1.html HTTP/1.1

由第7条消息可以看到，采用的是HTTP/1.1，使用双端口进行连接，防止头阻塞

对于每个端口，由主机向服务器发送连接建立请求SYN，服务器向主机回复SYN消息并携带确认消息ACK，主机收到服务器的回复并再次向服务器发送ACK

TCP 三次握手



四次挥手

10	8.001044	192.168.89.1	192.168.89.141	TCP	66 11537 → 80 [FIN, ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=353803537 TSecr=0
11	8.001203	192.168.89.1	192.168.89.141	TCP	66 11536 → 80 [FIN, ACK] Seq=532 Ack=191 Win=131328 Len=0 TSval=353803537 TSecr=0
12	8.001483	192.168.89.141	192.168.89.1	TCP	66 80 → 11537 [ACK] Seq=1 Ack=2 Win=64240 Len=0 TSval=195243 TSecr=353803537
13	8.001581	192.168.89.141	192.168.89.1	TCP	66 80 → 11536 [ACK] Seq=191 Ack=533 Win=63709 Len=0 TSval=195243 TSecr=3538035
14	8.001730	192.168.89.141	192.168.89.1	TCP	66 80 → 11537 [FIN, ACK] Seq=1 Ack=2 Win=64240 Len=0 TSval=195243 TSecr=353803
15	8.001946	192.168.89.141	192.168.89.1	TCP	66 80 → 11536 [FIN, ACK] Seq=191 Ack=533 Win=63709 Len=0 TSval=195243 TSecr=35
16	8.002184	192.168.89.1	192.168.89.141	TCP	66 11537 → 80 [ACK] Seq=2 Ack=2 Win=131584 Len=0 TSval=353803538 TSecr=195243
17	8.002217	192.168.89.1	192.168.89.141	TCP	66 11536 → 80 [ACK] Seq=533 Ack=192 Win=131328 Len=0 TSval=353803538 TSecr=195

同样，使用双端口，对于每个端口有以下过程

1. 主机向服务器发送连接断开请求，服务器回复确认，
2. 服务器关闭与服务器的连接，并发送FIN和ACK。客户端收到消息，回复确认

获取网页文字信息

网页

```
1 <html>
2   <head>
3     <title>Michelle</title>
4     <h1>1811494 刘旭萌 </h1>
5   </head>
6   <body>
7     信息安全专业
8   </body>
9 </html>
```

← → ↻ ⚠ 不安全 | 192.168.89.141/network/2.html

1811494 刘旭萌

信息安全专业

使用过滤器进行查找

http contains "1811494" tcp contains "1811494"						
No.	Time	Source	Destination	Protocol	Length	Info
+	113	222.856214	192.168.89.141	192.168.89.1	HTTP	501 HTTP/1.1 200 OK (text/html)

在这个区找到我们写的html内容，可以看到我们的文件一共由九行，与上述代码相对应

> Line-based text data: text/html (9 lines)

Line-based text data: text/html (9 lines)

\357\273\277<html>\r\n<head>\r\n<title>Michelle</title>\r\n<h1>1811494 \345\210\230\346\227\255\350\220\214 </h1>\r\n</head>\r\n<body>\r\n\344\277\241\346\201\257\345\256\211\345\205\250\344\270\223\344\270\232\r\n

0160	68 74 6d 6c 3b 20 63 68 61 72 73 65 74 3d 55 54	html; ch arset=UT
0170	46 2d 38 0d 0a 0d 0a ef bb bf 3c 68 74 6d 6c 3e	F-8..... <html>
0180	0d 0a 3c 68 65 61 64 3e 0d 0a 3c 74 69 74 6c 65	<head> <title
0190	3e 4d 69 63 68 65 6c 6c 65 3c 2f 74 69 74 6c 65	>Michell e</title
01a0	3e 0d 0a 3c 68 31 3e 31 38 31 31 34 39 34 20 e5	><h1>1 811494 .
01b0	88 98 e6 97 ad e8 90 8c 20 3c 2f 68 31 3e 0d 0a </h1>..
01c0	3c 2f 68 65 61 64 3e 0d 0a 3c 62 6f 64 79 3e 0d	</head> <body>
01d0	0a e4 bf a1 e6 81 af e5 ae 89 e5 85 a8 e4 b8 93
01e0	e4 b8 9a 0d 0a 3c 2f 62 6f 64 79 3e 0d 0a 3c 2f</b ody>..</
01f0	68 74 6d 6c 3e	html>

中文由 \xxx 八进制编码显示，在字符区由16进制表示，例如使用URL解码工具

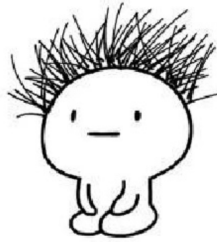
%e5%88%98

UrlEncode编码UrlDecode解码

刘

获取图片信息

```
1 <html>
2   <head>
3     <title>Michelle</title>
4     <h1>1811494 刘旭萌 </h1>
5   </head>
6   <body>
7     <img src=/1.jpg>
8   </body>
9 </html>
```

No.	Time	Source	Destination	Protocol	Length	Info
5	1.095846	192.168.89.1	192.168.89.141	HTTP	460	GET /network/3.html HTTP/1.1
6	1.096814	192.168.89.141	192.168.89.1	HTTP	497	HTTP/1.1 200 OK (text/html)
8	1.149870	192.168.89.1	192.168.89.141	HTTP	408	GET /1.jpg HTTP/1.1
28	3.289528	192.168.89.1	192.168.89.141	HTTP	414	GET /favicon.ico HTTP/1.1
43	3.301572	192.168.89.141	192.168.89.1	HTTP	131	HTTP/1.1 200 OK (JPEG JFIF image)
45	3.306647	192.168.89.141	192.168.89.1	HTTP	712	HTTP/1.1 404 Not Found (text/html)

> [Expert Info (Chat/Sequence): GET /1.jpg HTTP/1.1\r\n]

Request Method: GET

Request URI: /1.jpg

0040	cb 25 47 45 54 20	2f 31 2e 6a 70 67	20 48 54 54	%GET /1 .jpg HT
0050	50 2f 31 2e 31 0d	0a 48 6f 73 74 3a	20 31 39 32	P/1.1..Host: 192
0060	2e 31 36 38 2e 38	39 2e 31 34 31 0d	0a 55 73 65	.168.89. 141..Use
0070	72 31 41 67 65 63	22 31 6f 70 6a 63		Agent: Mozilla

JPEG File Interchange Format															
0120	67 65 2f 6a 70 65 67 0d 0a 0d 0a ff d8 ff e0 00	ge/jpeg.													
0130	10 4a 46 49 46 00 01 01 01 00 48 00 48 00 00 ff	.JFIF... ..H.H...													
0140	e1 00 22 45 78 69 66 00 00 4d 4d 00 2a 00 00 00	.."Exif. .MM.*...													
0150	08 00 01 01 12 00 03 00 00 00 01 00 01 00 00 00													
0160	00 00 00 ff db 00 43 00 02 01 01 02 01 01 02 02C.													
0170	02 02 02 02 02 02 03 05 03 03 03 03 06 04 04													
0180	03 05 07 06 07 07 07 06 07 07 08 09 0b 09 08 08													
0190	0a 08 07 07 0a 0d 0a 0a 0b 0c 0c 0c 0c 07 09 0e													
01a0	0f 0d 0c 0e 0b 0c 0c 0c ff db 00 43 01 02 02 02C....													
01b0	03 03 03 06 03 03 06 0c 08 07 08 0c 0c 0c 0c 0c													
01c0	0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c													
01d0	0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c													
01e0	0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c ff c0 00													
01f0	11 08 01 ce 02 b2 03 01 22 00 02 11 01 03 11 01"													
0200	ff c4 00 1f 00 00 01 05 01 01 01 01 01 01 00 00													

Frame (131 bytes)

Reassembled TCP (26089 bytes)

获取视频信息

```
1 <body>
2   <video width="300"height="240"controls="controls">
3     <source src="video.mp4",type="video/mp4"/>
4   </video>
5   </body>
6 </body>
7 </html>
```

← → ↻ 🔒 不安全 | 192.168.89.141/network/4.html

1811494 刘旭萌 信息安全专业



捕捉到一系列数据包

```
192.168.89.141 HTTP 470 GET /network/video.mp4 HTTP/1.1
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=647 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802135 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=2095 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802135 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1266 80 → 32886 [PSH, ACK] Seq=3543 Ack=781 Win=63460 Len=1200 TSval=290270 TSecr=366802135 [TCP segment of a reassembled PDU]
192.168.89.141 TCP 66 32886 → 80 [ACK] Seq=781 Ack=4743 Win=131584 Len=0 TSval=366802136 TSecr=290270
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=4743 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802136 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=6191 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802136 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1514 80 → 32886 [PSH, ACK] Seq=7639 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802136 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=9087 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802136 [TCP segment of a reassembled PDU]
192.168.89.141 TCP 66 32886 → 80 [ACK] Seq=781 Ack=10535 Win=131584 Len=0 TSval=366802136 TSecr=290270
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=10535 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802136 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1018 80 → 32886 [PSH, ACK] Seq=11983 Ack=781 Win=63460 Len=952 TSval=290270 TSecr=366802136 [TCP segment of a reassembled PDU]
192.168.89.141 TCP 66 32886 → 80 [ACK] Seq=781 Ack=12935 Win=131584 Len=0 TSval=366802137 TSecr=290270
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=12935 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802137 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1514 80 → 32886 [ACK] Seq=14383 Ack=781 Win=63460 Len=1448 TSval=290270 TSecr=366802137 [TCP segment of a reassembled PDU]
192.168.89.1 TCP 1266 80 → 32886 [PSH, ACK] Seq=15831 Ack=781 Win=63460 Len=1200 TSval=290270 TSecr=366802137 [TCP segment of a reassembled PDU]
192.168.89.141 TCP 66 32886 → 80 [ACK] Seq=781 Ack=17031 Win=131584 Len=0 TSval=366802137 TSecr=290270
```

1. TCP包中的**win**代表接收窗口的大小，即表示这个包的发送方当前还有多少缓存区可以接收数据
2. **TS**(Timestamps) Timestamps在tcp选项中包括两个32位的timestamp: **TSval**(Timestamp value)和**TSecr**(Timestamp Echo Reply)。如果设置了**TS**这个选项，发送方发送时，将当前时间填入**TSval**，接收方回应时，将发送方的**TSval**填入**TSecr**即可(注意发送或接收都有设置**TSval**和**TSecr**)。
3. “**TCP segment of a reassembled PDU**”，指TCP层收到上层大块报文后分解成段后发出去
4. **PSH**所表达的是发送方通知接收方传输层应该尽快的将这个报文段交给应用层


通过观察可以发现消息显示出明显的**周期性**，即服务器发送三条消息，最后一条带有psh标志，客户端回应一条ack消息，且len=0

最后，返回状态码206，表示部分请求成功，视频获取完毕

```
192.168.89.1 HTTP 637 HTTP/1.1 206 Partial Content (video/mp4)
```

在文本框输入信息并回显

```
1 <html>
2   <head>
3     <title>1811494</title>
4   </head>
5   <body>
6     <script language="javascript">
7       function getLoginMsg(){
8         loginMsg=document.loginForm;
9         alert("name:"+loginMsg.userName.value+"\n");
10      }
11    </script>
12    <form name="loginForm">
13      name:<input type="text" name="userName"/><br/><br/>
14      <input type="button" value="submit" onclick="getLoginMsg()"/>
15    </form>
16  </body>
17 </html>
18
```



输入姓名，点击submit

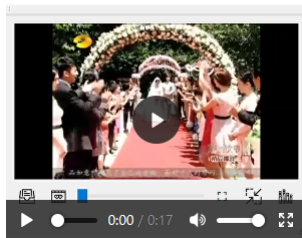
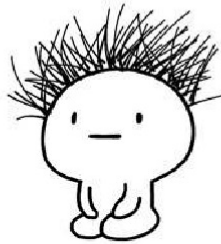


页面回显，但是在wireshark中没有找到回显内容相关数据包，猜测是因为使用了alert的函数是由浏览器直接处理的，并不将数据提交至服务器



总结

1811494 刘旭萌 信息安全



1. 在实验过程中，我们发现，当第一次请求页面并请求成功时，页面会返回状态码200表示请求成功，并同时返回页面的html内容；当再次请求且页面没有修改时会返回304表示页面未修改可以直接使用浏览器缓存的内容
2. 传输流程整体为：
 1. 客户端发出请求，服务器在局域网内（因为使用的是本地的虚拟机）发送查找网卡请求并找到ip对应的物理地址
 2. 三次握手建立连接
 3. 进行页面请求，服务器返回html内容
 4. 请求并返回图片和视频等内容
 5. 四次挥手断开连接

参考：

- https://www.wireshark.org/docs/wsdg_html_chunked/
- <https://blog.csdn.net/chenpuo/article/details/108186444>
- <https://blog.csdn.net/lixiangminghate/article/details/83024865>
- <https://blog.csdn.net/chenlycly/article/details/52402945>
- <https://blog.csdn.net/asdfsadfasdfa/article/details/88090027>