

CS315 Lab 1

Name: Yitong WANG(王奕童)

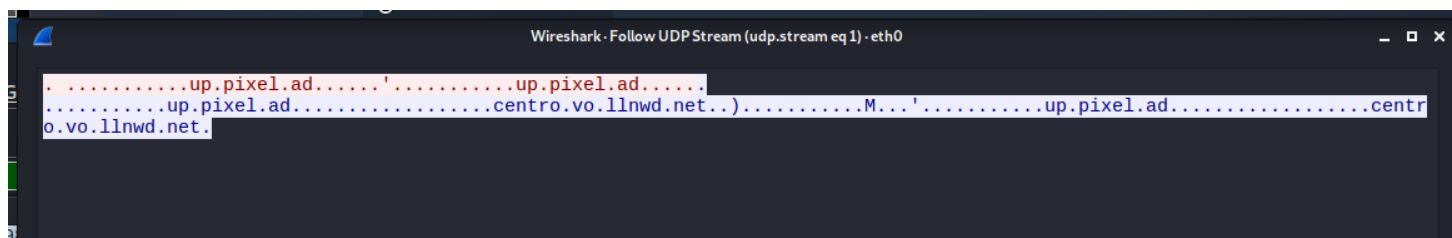
SID: 11910104

Q1

Carefully read the lab instructions and finish all tasks above.

我已经按照所有操作流程完成了实验课上的练习，以下选取部分重要截图展示：

第8-9步Follow DNS的UDP Stream：



第10步Follow HTTP请求的TCP Stream：



出现了307 Temporary Redirect，并非课件上的200。

Q2

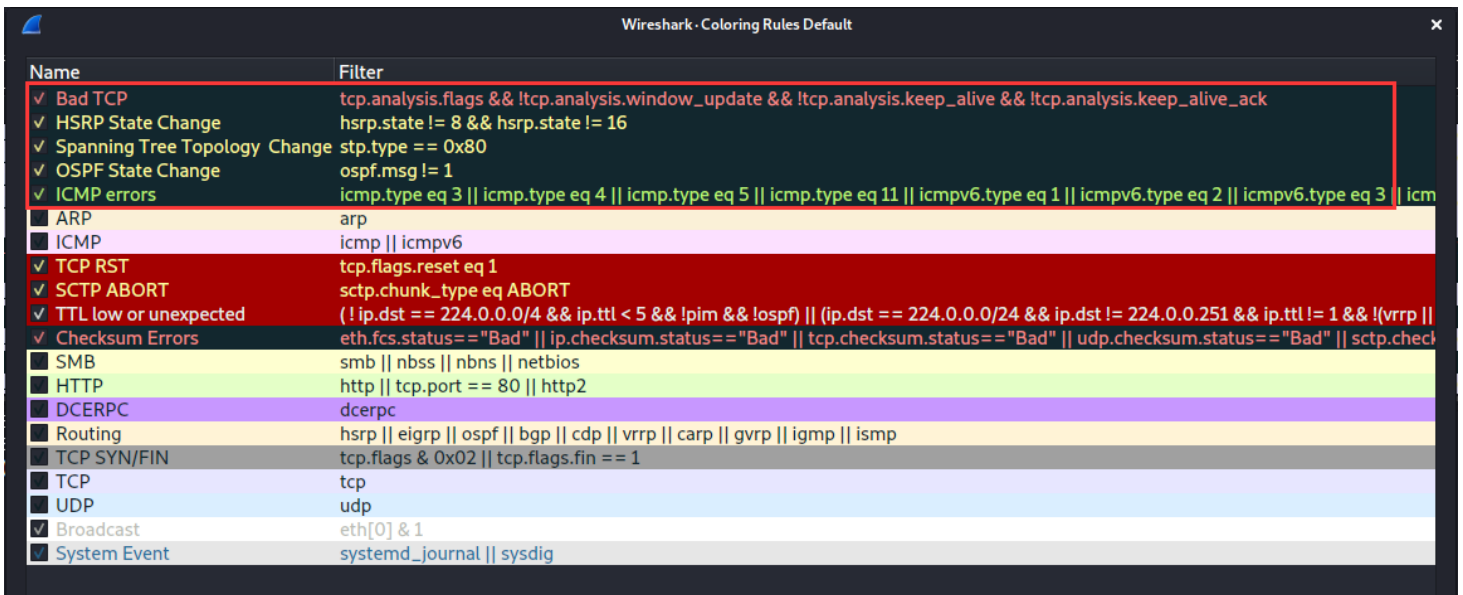
If a packet is highlighted by black, what does it mean for the packet?

根据实验课pdf中的介绍，标记为black的包是TCP传输中遇到问题的包。

5. Color Coding: You'll probably see packets highlighted in green, blue, and black. Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.

根据Wireshark自带的coloring rules, 我们可以得知标记为black的包有可能有以下情形:

- Bad TCP
- HSRP State Change
- Spanning Tree Topology Change
- OSPF State Change
- ICMP errors.



Name	Filter
✓ Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive && !tcp.analysis.keep_alive_ack
✓ HSRP State Change	hsrp.state != 8 && hsrp.state != 16
✓ Spanning Tree Topology Change	stp.type == 0x80
✓ OSPF State Change	ospf.msg != 1
✓ ICMP errors	icmp.type eq 3 icmp.type eq 4 icmp.type eq 5 icmp.type eq 11 icmpv6.type eq 1 icmpv6.type eq 2 icmpv6.type eq 3 icmpv6.type eq 4 icmpv6.type eq 5 icmpv6.type eq 11
■ ARP	arp
■ ICMP	icmp icmpv6
✓ TCP RST	tcp.flags.reset eq 1
✓ SCTP ABORT	sctp.chunk_type eq ABORT
✓ TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !ipm && !ospf) (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !vrrp
✓ Checksum Errors	eth.fcs.status == "Bad" ip.checksum.status == "Bad" tcp.checksum.status == "Bad" udp.checksum.status == "Bad" sctp.checksum.status == "Bad"
■ SMB	smb nbss nbns netbios
■ HTTP	http tcp.port == 80 http2
■ DCERPC	dcerpc
■ Routing	hsrp eigrp ospf bgp cdp vrrp carp gvrp igmp ismp
■ TCP SYN/FIN	tcp.flags & 0x02 tcp.flags.fin == 1
■ TCP	tcp
■ UDP	udp
✓ Broadcast	eth[0] & 1
✓ System Event	systemd_journal sysdig

以下部分是我按照实验操作请求 <http://www.wayne.edu> 后得到的黑色报文包:

1571	4.027007807	13.225.103.82	192.168.163.132	TCP	1414	[TCP Previous segment not captured] 443 → 36192 [PSH, ACK] Seq=599798 Ack=1275 Win=64240 Len=1360
1572	4.027026955	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#1] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1573	4.027620301	13.225.103.82	192.168.163.132	TCP	1414	443 → 36192 [PSH, ACK] Seq=601158 Ack=1275 Win=64240 Len=1360 [TCP segment of a reassembled PDU]
1574	4.027631860	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#2] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1575	4.029042164	13.225.103.82	192.168.163.132	TCP	1514	443 → 36192 [ACK] Seq=602518 Ack=1275 Win=64240 Len=1460 [TCP segment of a reassembled PDU] [TCP
1576	4.029051558	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#3] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1577	4.029074513	13.225.103.82	192.168.163.132	TCP	1514	443 → 36192 [ACK] Seq=603978 Ack=1275 Win=64240 Len=1460 [TCP segment of a reassembled PDU]
1578	4.029078199	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#4] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1579	4.029090287	13.225.103.82	192.168.163.132	TCP	1514	443 → 36192 [ACK] Seq=605438 Ack=1275 Win=64240 Len=1460 [TCP segment of a reassembled PDU]
1580	4.029093027	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#5] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1581	4.029105502	13.225.103.82	192.168.163.132	TCP	1514	443 → 36192 [ACK] Seq=606898 Ack=1275 Win=64240 Len=1460 [TCP segment of a reassembled PDU]
1582	4.029108236	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#6] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1583	4.029120738	13.225.103.82	192.168.163.132	TCP	1014	443 → 36192 [PSH, ACK] Seq=608358 Ack=1275 Win=64240 Len=960 [TCP segment of a reassembled PDU]
1584	4.029123351	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#7] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1585	4.030217752	13.225.103.82	192.168.163.132	TCP	1414	443 → 36192 [PSH, ACK] Seq=609318 Ack=1275 Win=64240 Len=1360 [TCP segment of a reassembled PDU]
1586	4.030241784	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#8] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1587	4.030848518	13.225.103.82	192.168.163.132	TCP	1414	443 → 36192 [PSH, ACK] Seq=610678 Ack=1275 Win=64240 Len=1360 [TCP segment of a reassembled PDU]
1588	4.031481709	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#9] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1589	4.031481709	13.225.103.82	192.168.163.132	TCP	1414	443 → 36192 [PSH, ACK] Seq=612038 Ack=1275 Win=64240 Len=1360 [TCP segment of a reassembled PDU]
1590	4.031489619	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#10] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1591	4.032249022	13.225.103.82	192.168.163.132	TCP	1414	443 → 36192 [PSH, ACK] Seq=613398 Ack=1275 Win=64240 Len=1360 [TCP segment of a reassembled PDU]
1592	4.032258742	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#11] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1593	4.033085917	13.225.103.82	192.168.163.132	TCP	1414	443 → 36192 [PSH, ACK] Seq=614758 Ack=1275 Win=64240 Len=1360 [TCP segment of a reassembled PDU]
1594	4.033096542	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#12] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0
1595	4.036396988	13.225.103.82	192.168.163.132	TCP	1414	443 → 36192 [PSH, ACK] Seq=616118 Ack=1275 Win=64240 Len=1360 [TCP segment of a reassembled PDU]
1596	4.036417183	13.225.103.82	192.168.163.132	TCP	54	[TCP Dup ACK 1570#13] 36192 → 443 [ACK] Seq=1275 Ack=595818 Win=65535 Len=0

查看字体颜色并与表格对照，可以发现我们在这次实验中黑色报文的包是Bad TCP类型，有可能触发的原因是触发了重传、乱序、丢包、重复响应等等[1]。

Q3

What is the filter command for listing all outgoing http traffic?

首先使用 `ifconfig` 命令，获取当前Kali虚拟机的ip地址。如下图可知是 192.168.163.132 。

```

root@kali: ~/Desktop
File Actions Edit View Help
Try: apt install <deb name>

(root@kali)-[~/Desktop]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.163.132 netmask 255.255.255.0 broadcast 192.168.163.255
    inet6 fe80::20c:29ff:fef0:4baf prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:f0:4b:af txqueuelen 1000 (Ethernet)
    RX packets 2 bytes 585 (585.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 1452 (1.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 19 base 0x2000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 28 bytes 1400 (1.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28 bytes 1400 (1.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(root@kali)-[~/Desktop]
#

```

然后获取所有http出口流量包的过滤命令模板是[2]:

```
ip.src == my_ip_address && http
```

在这里，my_ip_address是 192.168.163.132，因此命令是:

```
ip.src == 192.168.163.132 && http
```

执行命令后可以得到预期的包:

ip.src == 192.168.163.132 && http						
No.	Time	Source	Destination	Protocol	Length	Info
117	1.337652125	192.168.163.132	203.208.40.98	OCSP	425	Request
169	1.658767637	192.168.163.132	117.18.237.29	OCSP	423	Request
757	2.979123799	192.168.163.132	117.18.237.29	OCSP	423	Request
1933	6.281567185	192.168.163.132	141.217.1.160	HTTP	376	GET / HTTP/1.1
1952	9.350309681	192.168.163.132	117.18.237.29	OCSP	423	Request
2043	11.086958572	192.168.163.132	203.208.40.98	OCSP	425	Request
2196	11.731757326	192.168.163.132	203.208.40.98	OCSP	425	Request
2307	11.940553747	192.168.163.132	203.208.40.98	OCSP	425	Request
2318	11.996718616	192.168.163.132	203.208.40.98	OCSP	425	Request
3568	15.040414093	192.168.163.132	13.225.95.47	OCSP	432	Request
3605	15.288621540	192.168.163.132	13.225.95.47	OCSP	432	Request
3637	15.453994319	192.168.163.132	13.225.95.47	OCSP	432	Request
3657	15.509288166	192.168.163.132	13.225.95.47	OCSP	432	Request

Q4

Why does DNS use Follow UDP Stream while HTTP use Follow TCP Stream?

(1) DNS使用UDP连接的原因[3]:

- UDP速度更快。因为无需保持与服务端持续交互的状态，因此使用UDP可以节约连接建立与断开的
时间（无需三次握手和四次挥手）
- UDP请求体一般比较小，使用UDP会更加合适。
- UDP虽然可靠性不如TCP，但是在五层网络结构中，应用程序可以在应用层添加错误重传和超时检
测来弥补可靠性的不足。

(2) HTTP使用TCP连接的原因[4]:

- TCP有更加可靠的连接。HTTP交互的内容需要保证数据的正确性，因此需要TCP的可靠连接保
障。
- TCP支持更大的消息体传输。UDP通常传输的消息体不超过512B，而HTTP连接在保障报文完整性
的前提下，内容大小经常超过这个限制，因此需要TCP连接。

Q5

Using Wireshark to capture the FTP password.

选择 Wireshark 的 any 端口，然后在开始登录前就启动抓包。

使用命令完成一次FTP登录，用户名 csc5991-student，密码 WSU-csc5991.。

```
(root@kali)-[~/Desktop]
# ftp
ftp> open 127.0.0.1
Connected to 127.0.0.1.
220 (vsFTPd 3.0.3)
Name (127.0.0.1:root): csc5991-student
331 Please specify the password.
Password: 
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> close
221 Goodbye.
ftp> open 127.0.0.1
Connected to 127.0.0.1.
220 (vsFTPd 3.0.3)
Name (127.0.0.1:root): csc5991-student
331 Please specify the password.
Password: 
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

登录完成后，停止 Wireshark 抓包，得到以下报文：

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	37342 → 21 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=107910082 TSecr=0 WS=128
2	0.000043136	127.0.0.1	127.0.0.1	TCP	76	21 → 37342 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=107910082 TSecr=0
3	0.000064700	127.0.0.1	127.0.0.1	TCP	68	37342 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=107910082 TSecr=107910089
4	0.007033470	127.0.0.1	127.0.0.1	FTP	88	Response: 220 (vsFTPd 3.0.3)
5	0.007128092	127.0.0.1	127.0.0.1	TCP	68	37342 → 21 [ACK] Seq=1 Ack=21 Win=65536 Len=0 TSval=107910089 TSecr=107910089
6	8.691013088	127.0.0.1	127.0.0.1	FTP	90	Request: USER csc5991-student
7	8.691156306	127.0.0.1	127.0.0.1	TCP	68	21 → 37342 [ACK] Seq=21 Ack=23 Win=65536 Len=0 TSval=107918773 TSecr=107918773
8	8.691384946	127.0.0.1	127.0.0.1	FTP	102	Response: 331 Please specify the password.
9	8.691398605	127.0.0.1	127.0.0.1	TCP	68	37342 → 21 [ACK] Seq=23 Ack=55 Win=65536 Len=0 TSval=107918773 TSecr=107918773
10	24.591447518	127.0.0.1	127.0.0.1	FTP	87	Request: PASS WSU-csc5991.
11	24.591472150	127.0.0.1	127.0.0.1	TCP	68	21 → 37342 [ACK] Seq=55 Ack=42 Win=65536 Len=0 TSval=107934673 TSecr=107934673
12	24.805785067	127.0.0.1	127.0.0.1	FTP	91	Response: 230 Login successful.
13	24.805838191	127.0.0.1	127.0.0.1	TCP	68	37342 → 21 [ACK] Seq=42 Ack=78 Win=65536 Len=0 TSval=107934887 TSecr=107934887
14	24.806042560	127.0.0.1	127.0.0.1	FTP	74	Request: SYST
15	24.806052137	127.0.0.1	127.0.0.1	TCP	68	21 → 37342 [ACK] Seq=78 Ack=48 Win=65536 Len=0 TSval=107934888 TSecr=107934888
16	24.806114912	127.0.0.1	127.0.0.1	FTP	87	Response: 215 UNIX Type: L8
17	24.806120103	127.0.0.1	127.0.0.1	TCP	68	37342 → 21 [ACK] Seq=48 Ack=97 Win=65536 Len=0 TSval=107934888 TSecr=107934888

其中红框区域的报文有密码信息 WSU-csc5991.：

```
0000  00 00 03 04 00 06 00 00 00 00 00 00 00 08 00
0010  45 10 00 47 93 09 40 00 40 06 a9 95 7f 00 00 01  E . G . @ . @
0020  7f 00 00 01 91 de 00 15 f7 7d f8 55 92 40 62 bc
0030  80 18 02 00 fe 3b 00 00 01 01 08 0a 06 6e f3 d1
0040  06 6e b5 b5 50 41 53 53 20 57 53 55 2d 63 73 63  . n . PASS WSU-csc
0050  35 39 39 31 2e 0d 0a                               5991.
```

参考链接

[1] Wireshark使用教程：不同报文颜色的含义 <https://blog.csdn.net/yeyiqun/article/details/99310715>

[2] wireshark端口过滤命令(wireshark端口过滤规则) <http://www.sgjsq.com/tougao/10229.html>

[3] 为什么DNS使用UDP而不是TCP? <https://www.pridetour.com.cn/wozhidao/z9vxyego4.html>

[4] 怎么理解TCP是面向连接的，HTTP基于TCP却是无连接的？

<https://www.zhihu.com/question/51996213/answer/128583448>