

## Kali 安全渗透高级工程师

### 学神 IT 教育: 从零基础到实战, 从入门到精通!

#### 版权声明:

本系列文档为《学神 IT 教育》内部使用教材和教案, 只允许 VIP 学员个人使用, 禁止私自传播。否则将取消其 VIP 资格, 追究其法律责任, 请知晓!

#### 免责声明:

本课程设计目的只用于教学, 切勿使用课程中的技术进行违法活动, 学员利用课程中的技术进行违法活动, 造成的后果与讲师本人及讲师所属机构无关。倡导维护网络安全人人有责, 共同维护网络文明和谐。

#### 联系方式:

学神 IT 教育官方网站: <http://xuegod.ke.qq.com>

学神 IT 教育-Kali 安全技术交流 QQ 群: 869961923



学习顾问: 乔老师

学习顾问: 李老师

学神微信公众号

微信扫码添加学习顾问微信, 同时扫码关注学神公众号了解最新行业动态, 获取更多学习资料及答疑就业服务!

## 第五章 WireShark 抓包及常用协议分析

### 本节所讲内容:

- 5.2 实战: WireShark 抓包及快速定位数据包技巧
- 5.3 实战: 使用 WireShark 对常用协议抓包并分析原理
- 5.4 实战: WireShark 抓包解决服务器被黑上不了网



### 5.2 实战: WireShark 抓包及快速定位数据包技巧

#### 5.2.1 常见协议包

本节课主要分析以下几种协议类型。

ARP 协议

ICMP 协议

TCP 协议

UDP 协议

DNS 协议

HTTP 协议

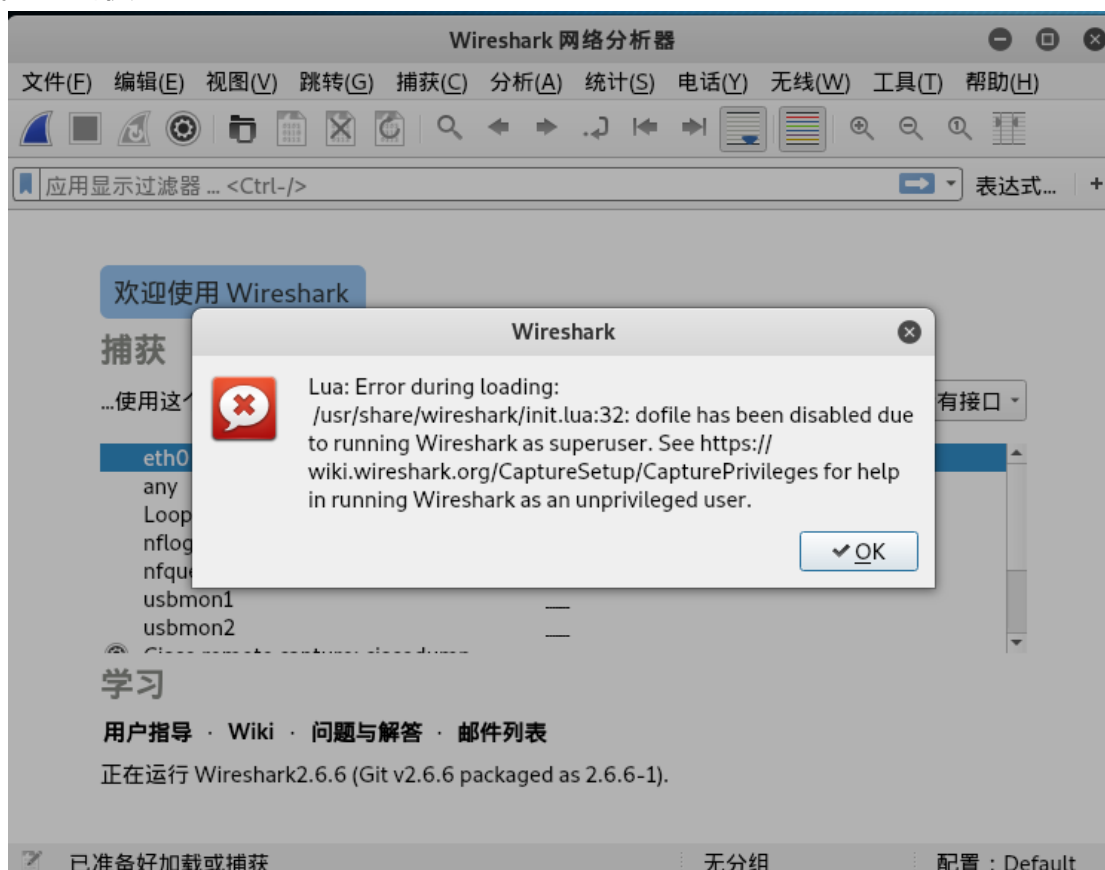
#### 5.2.2 使用 WireShark 进行抓包

启动 WireShark

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料



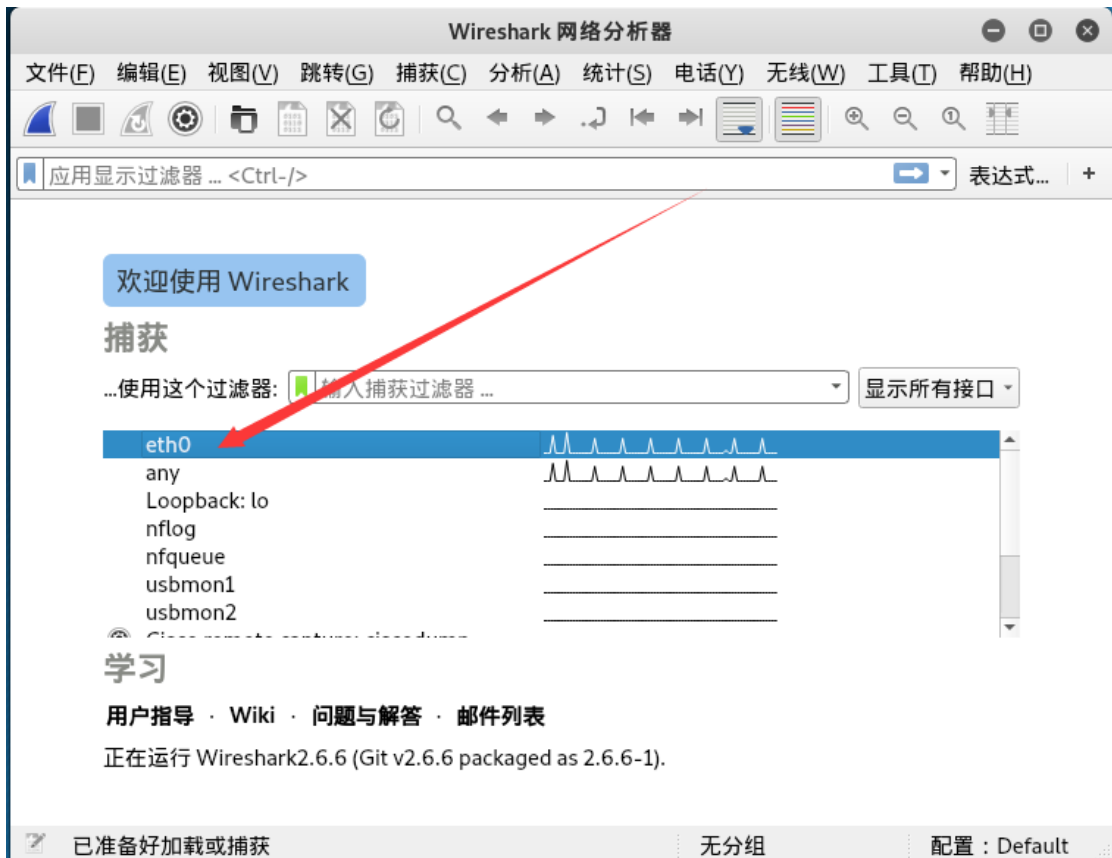
会有一个报错信息，是不建议我们使用 root 用户运行。我们直接点击 OK 就行，这个报错信息不影响我们任何的使用。



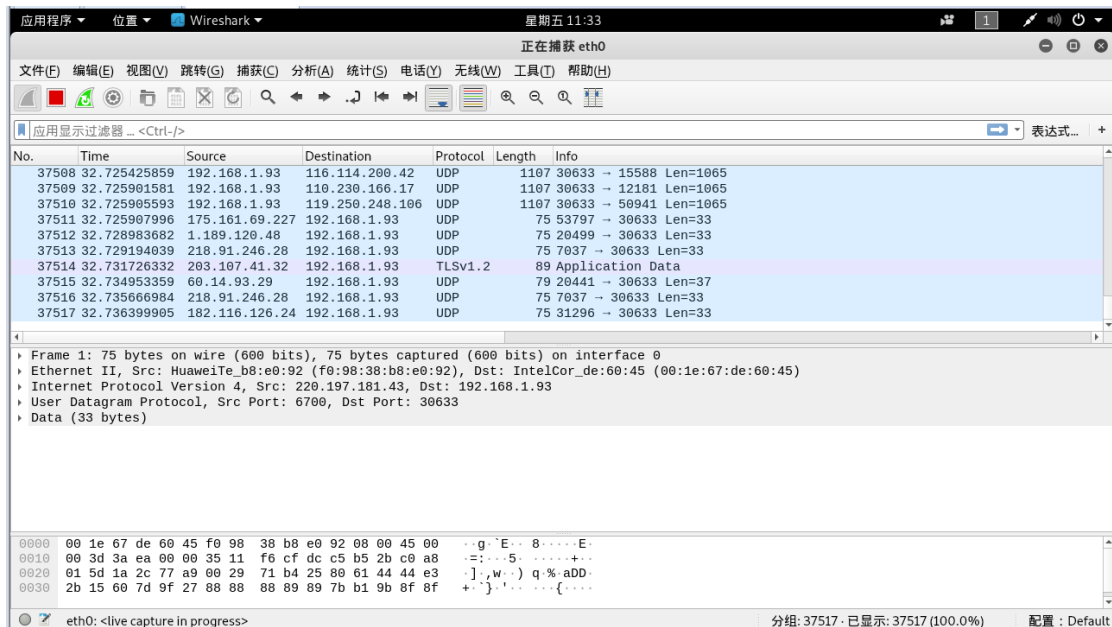
请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

### 选择我们的网卡



### 双击网卡之后就会自动进行抓包



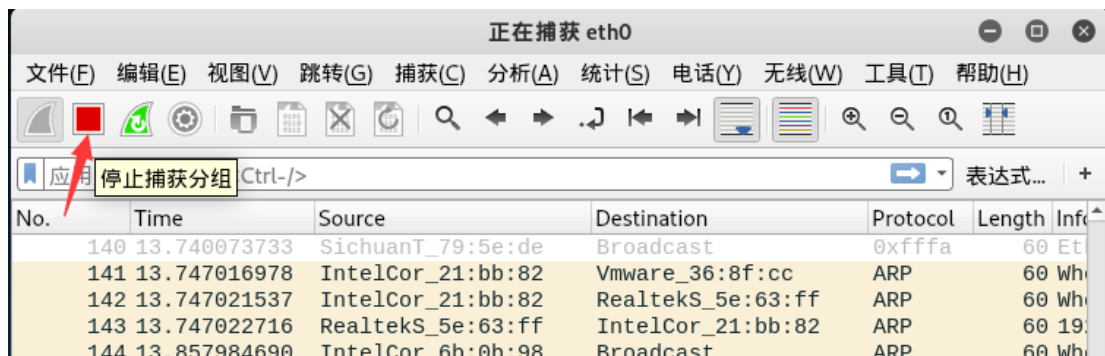
## 5.2.3 混杂模式介绍

- 1、混杂模式概述：混杂模式就是接收所有经过网卡的数据包，包括不是发给本机的包，即不验证 MAC 地址。普通模式下网卡只接收发给本机的包（包括广播包）传递给上层程序，其它的包一律丢弃。一般来说，混杂模式不会影响网卡的正常工作，多在网络监听工具上使用。

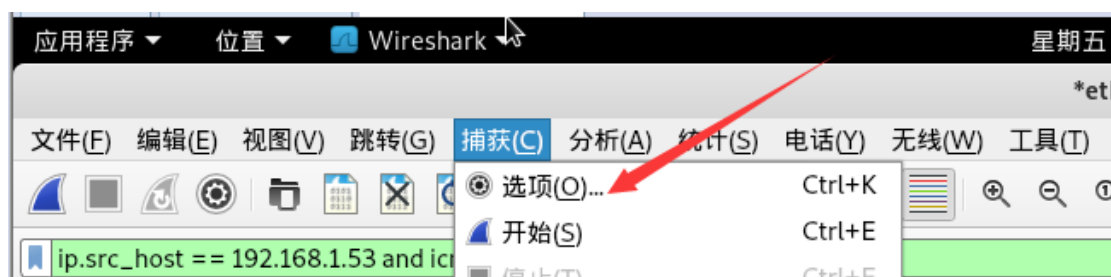
请加学神 IT 教育官方 QQ 群: 869961923 或李老師 QQ: 3147296050 领取更多资料

## 2、关闭和开启混杂模式方法

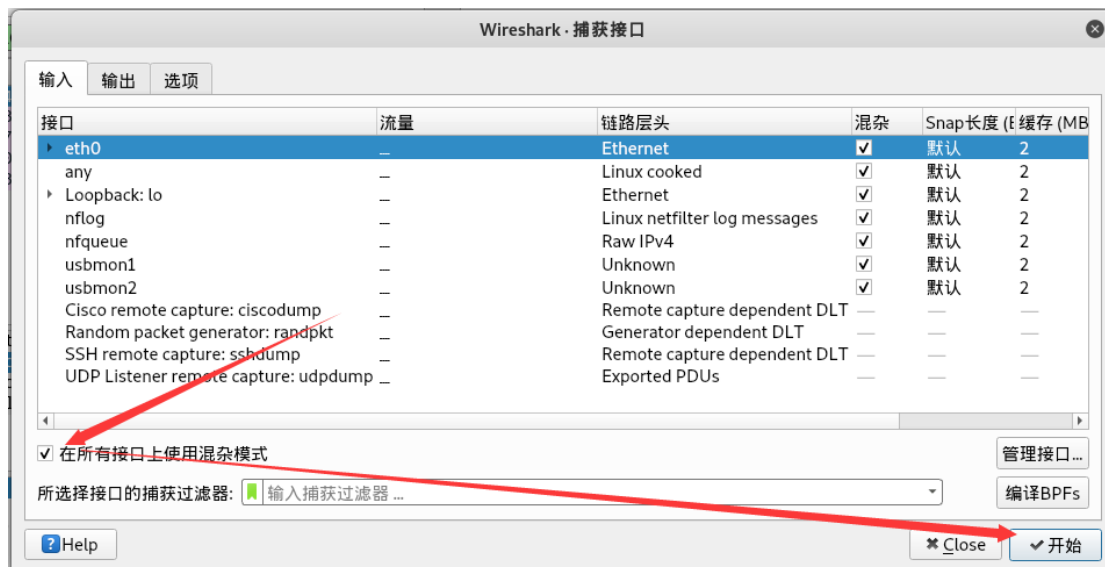
关闭和开启混杂模式前，需要停止当前抓包。如下，停止捕获。



在程序的工具栏中点击 “捕获” ---> “选项”



在选项设置界面中的 “输出” 设置栏的左下方勾选 “在所有接口上使用混杂模式”

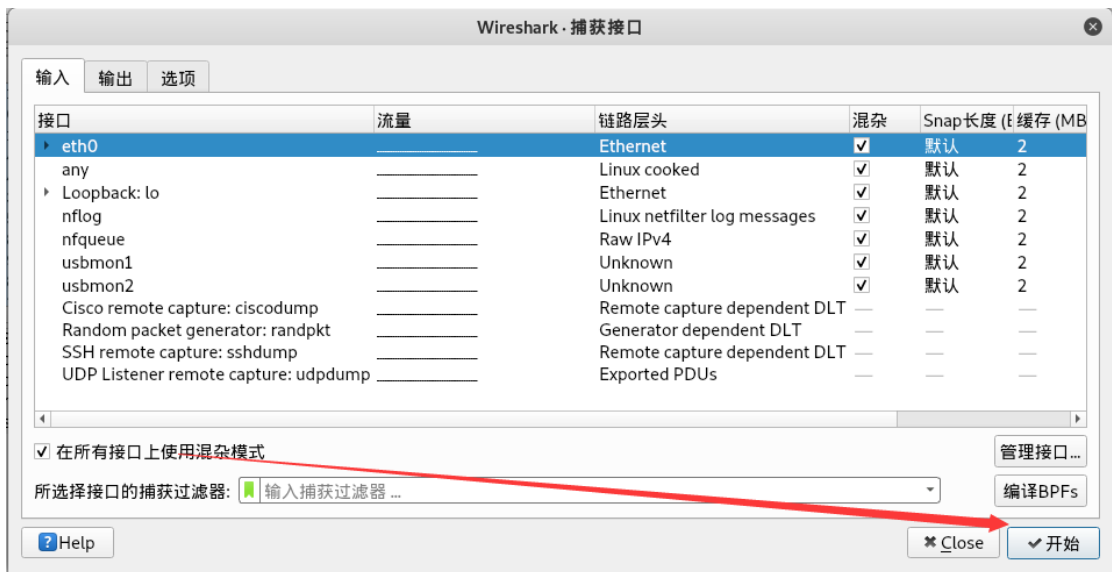


这样就开启了。默认就是开启混杂模式。

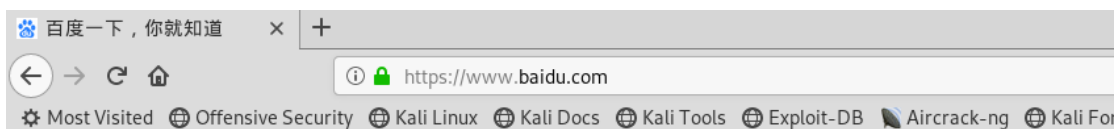
### 5.2.4 WireShark 的过滤器使用

我们开启混淆模式来做一下感受，我们再次捕获---在所有接口上使用混杂模式就可以直接进行抓包了

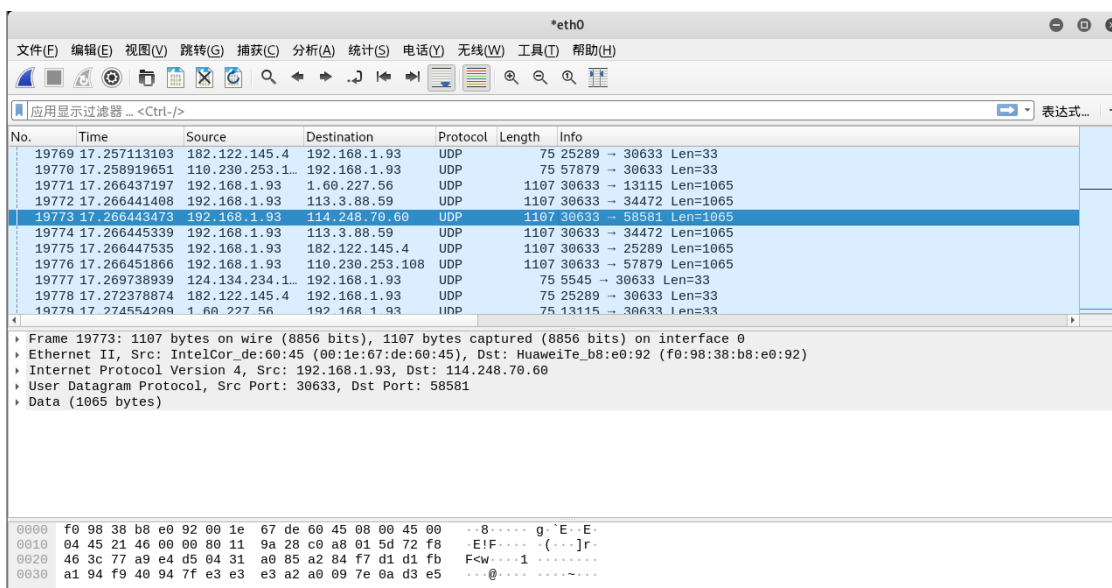
请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料



下面我们打开浏览器访问以下百度。



访问完成后点击停止抓包即可, 我们不需要抓太多的数据包。

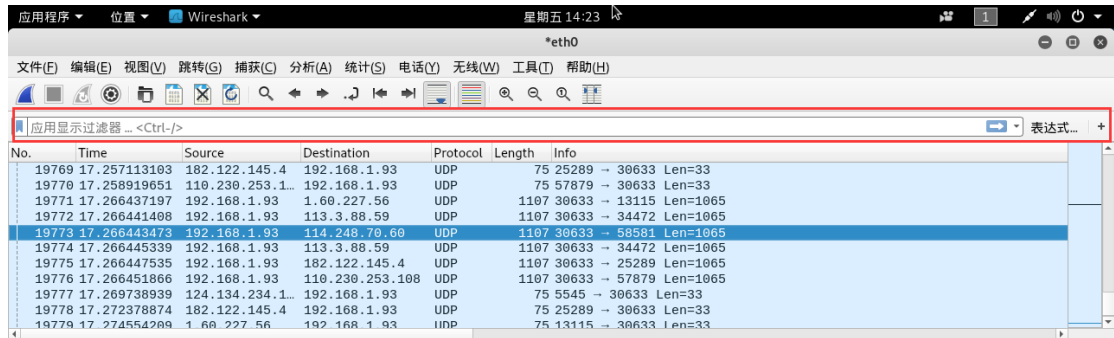


我们可以看到有很多数据包但是我们怎么才能找到对应的数据包类型呢?

请加学神 IT 教育官方 QQ 群: 869961923 或李老師 QQ: 3147296050 领取更多资料



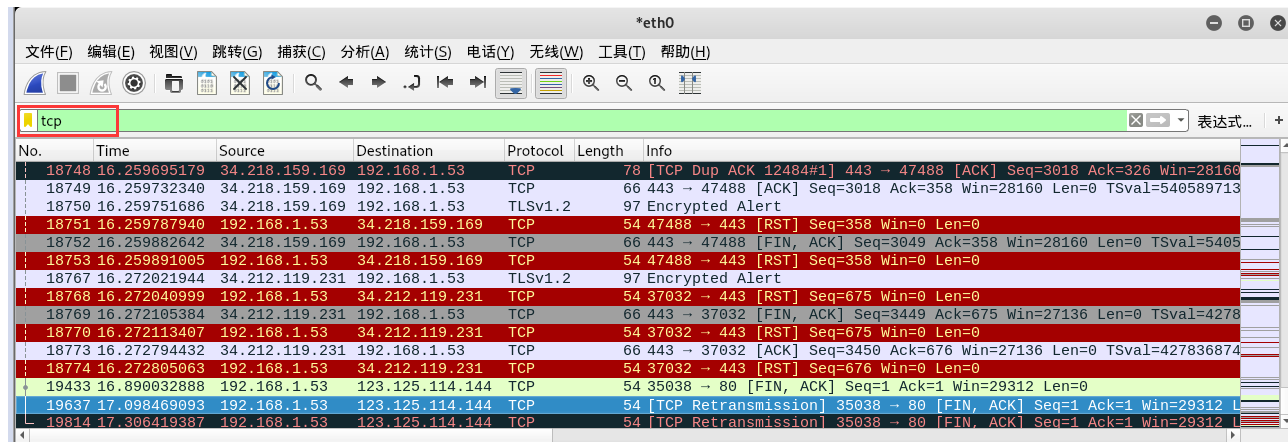
请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料



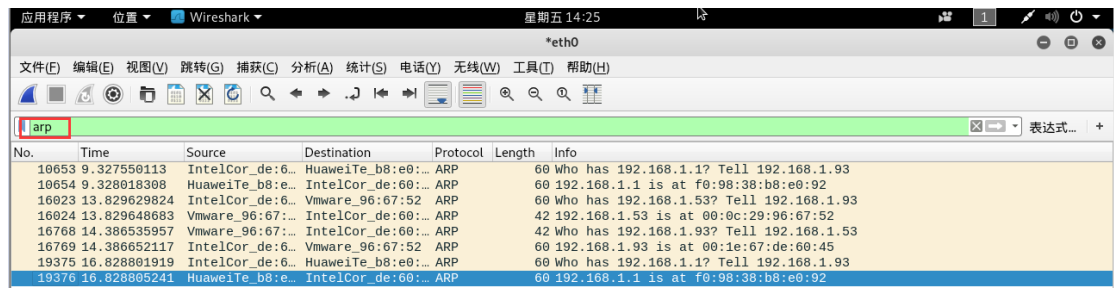
这里就是我们的过滤器，我们可以根据自己的条件筛选自己想要的数据包。

例 1：使用过滤器筛选 TCP 的数据包

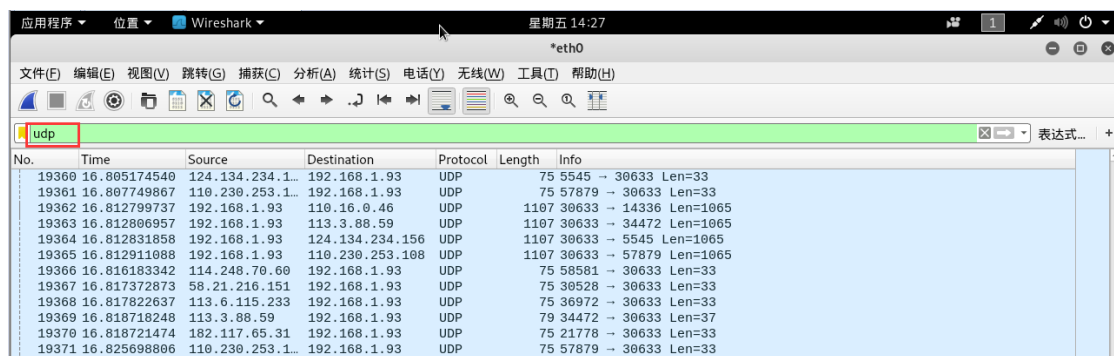
注意：筛选条件我们都使用小写就好了，大写的话会不识别。



例 2：使用过滤器筛选 arp 的数据包



例 3：使用过滤器筛选 udp 的数据包



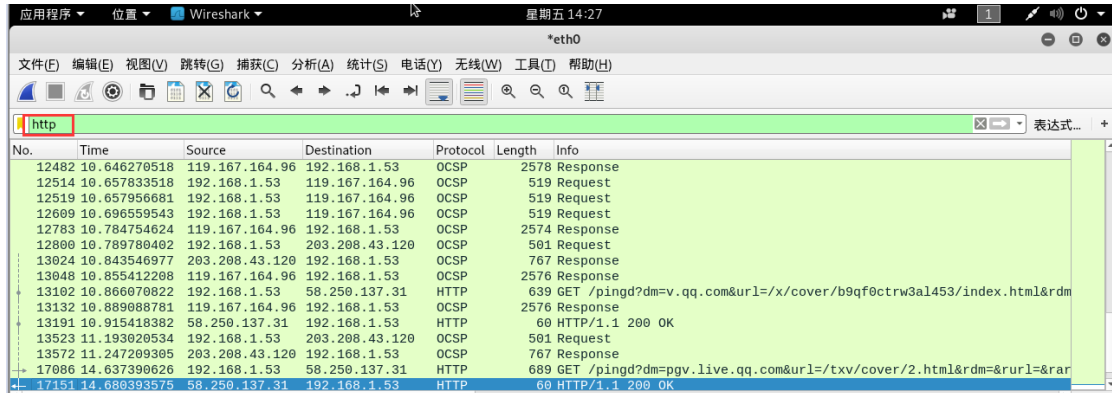
我们使用过滤器输入“udp”以筛选出 udp 报文。但是为什么输入 udp 之后出现那么多协议呢？原因就是 oicq 以及 dns 都是基于 udp 的传输层之上的协议

请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

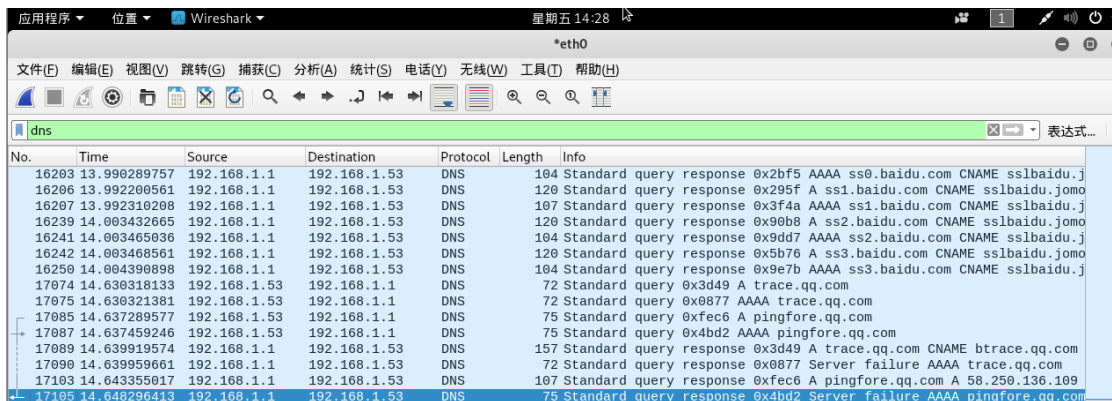
扩展: 客户端向 DNS 服务器查询域名, 一般返回的内容都不超过 512 字节, 用 UDP 传输即可。不用经过三次握手, 这样 DNS 服务器负载更低, 响应更快。理论上说, 客户端也可以指定向 DNS 服务器查询时用 TCP, 但事实上, 很多 DNS 服务器进行配置的时候, 仅支持 UDP 查询包。

#### 例 4: 使用过滤器筛选 http 的数据包



No.	Time	Source	Destination	Protocol	Length	Info
12482	10.646270518	119.167.164.96	192.168.1.53	OCSP	2578	Response
12514	10.657833518	192.168.1.53	119.167.164.96	OCSP	519	Request
12519	10.657956681	192.168.1.53	119.167.164.96	OCSP	519	Request
12609	10.696559543	192.168.1.53	119.167.164.96	OCSP	519	Request
12783	10.784754624	119.167.164.96	192.168.1.53	OCSP	2574	Response
12800	10.789780402	192.168.1.53	203.208.43.120	OCSP	501	Request
13024	10.843546977	203.208.43.120	192.168.1.53	OCSP	767	Response
13048	10.855412298	119.167.164.96	192.168.1.53	OCSP	2576	Response
13102	10.866070822	192.168.1.53	58.250.137.31	HTTP	639	GET /pingd?dm=v.qq.com&url=/x/cover/b9qf0ctrw3a1453/index.html&rdm=
13132	10.889088781	119.167.164.96	192.168.1.53	OCSP	2576	Response
13191	10.915418382	58.250.137.31	192.168.1.53	HTTP	60	HTTP/1.1 200 OK
13523	11.193020534	192.168.1.53	203.208.43.120	OCSP	501	Request
13572	11.247209305	203.208.43.120	192.168.1.53	OCSP	767	Response
17086	14.637390626	192.168.1.53	58.250.137.31	HTTP	689	GET /pingd?dm=pgv.live.qq.com&url=/txv/cover/2.html&rdm=&url=&rar
17151	14.680393575	58.250.137.31	192.168.1.53	HTTP	60	HTTP/1.1 200 OK

#### 例 5: 使用过滤器筛选 dns 的数据包



No.	Time	Source	Destination	Protocol	Length	Info
16203	13.990289757	192.168.1.1	192.168.1.53	DNS	104	Standard query response 0x2bf5 AAAA ss0.baidu.com CNAME sslbaidu.j
16206	13.992200561	192.168.1.1	192.168.1.53	DNS	120	Standard query response 0x295f A ss1.baidu.com CNAME sslbaidu.jomo
16207	13.992310208	192.168.1.1	192.168.1.53	DNS	107	Standard query response 0x3f4a AAAA ss1.baidu.com CNAME sslbaidu.j
16239	14.003432605	192.168.1.1	192.168.1.53	DNS	120	Standard query response 0x90b8 A ss2.baidu.com CNAME sslbaidu.jomo
16241	14.003465036	192.168.1.1	192.168.1.53	DNS	104	Standard query response 0x9d77 AAAA ss2.baidu.com CNAME sslbaidu.j
16242	14.003468561	192.168.1.1	192.168.1.53	DNS	120	Standard query response 0x5b76 A ss3.baidu.com CNAME sslbaidu.jomo
16250	14.004390898	192.168.1.1	192.168.1.53	DNS	104	Standard query response 0x9e7b AAAA ss3.baidu.com CNAME sslbaidu.j
17074	14.630318133	192.168.1.53	192.168.1.1	DNS	72	Standard query 0x3d49 A trace.qq.com
17075	14.630321381	192.168.1.53	192.168.1.1	DNS	72	Standard query 0x0877 AAAA trace.qq.com
17085	14.637289577	192.168.1.53	192.168.1.1	DNS	75	Standard query 0xfec6 A pingfore.qq.com
17087	14.637459246	192.168.1.53	192.168.1.1	DNS	75	Standard query 0x4bd2 AAAA pingfore.qq.com
17089	14.639919574	192.168.1.1	192.168.1.53	DNS	157	Standard query response 0x3d49 A trace.qq.com CNAME btrace.qq.com
17090	14.639959661	192.168.1.1	192.168.1.53	DNS	72	Standard query response 0x0877 Server failure AAAA trace.qq.com
17103	14.643355017	192.168.1.1	192.168.1.53	DNS	107	Standard query response 0xfec6 A pingfore.qq.com A 58.250.136.109
17105	14.648296413	192.168.1.1	192.168.1.53	DNS	75	Standard query response 0x4bd2 Server failure AAAA pingfore.qq.com

其实我们不仅可以对协议类型进行筛选, 我们还有跟多的筛选条件, 比如源地址目的地址等等...

#### 例 6: 筛选源地址是 192.168.1.53 或目的地址是 192.168.1.1

在终端 ping 192.168.1.1



```
root@xuegod53: ~  
root@xuegod53:~# ping 192.168.1.1
```

No.	Time	Source	Destination
317	56.887503890	192.168.1.53	192.168.1.1

然后修改筛选器条件为:

**ip.src\_host == 192.168.1.53 or ip.dst\_host == 192.168.1.1**

这个判断条件是什么意思呢?

**ip.src\_host == 192.168.1.53** 表示源 IP 地址

**ip.dst\_host == 192.168.1.1** 表示目的地址

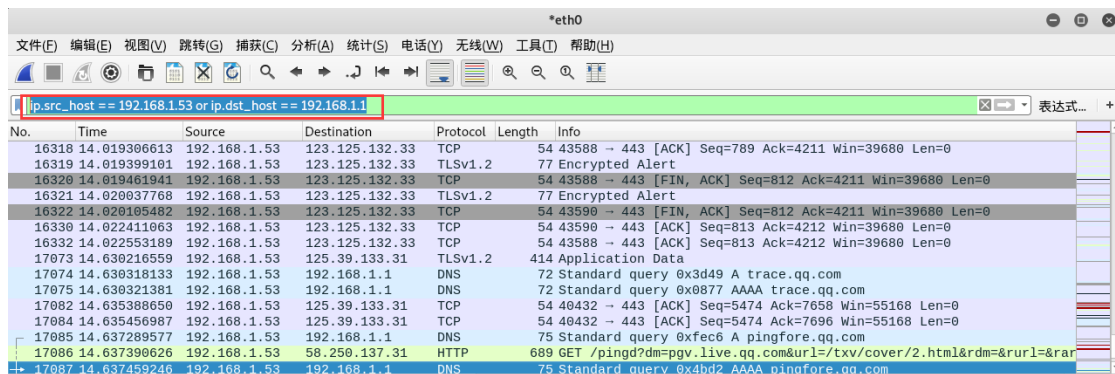
请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料



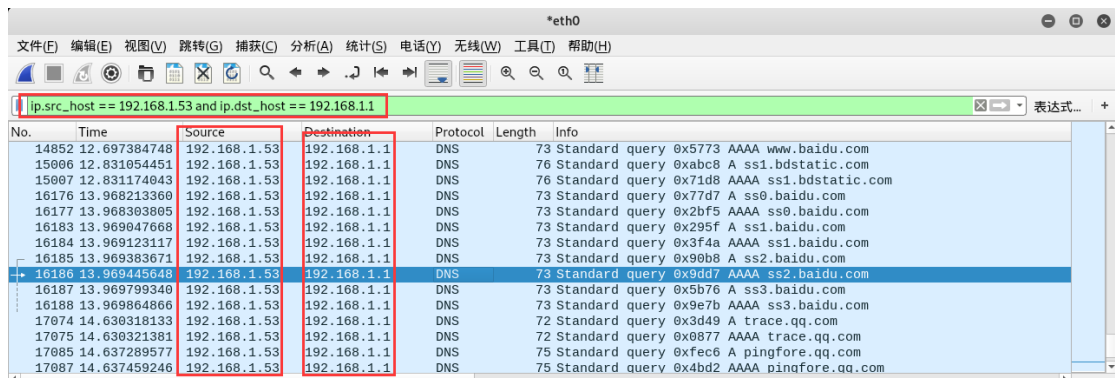
请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

我们中间用 or 进行了拼接, 表示或 当然我们也可以使用 and 表示与, or 表示满足左右其中一个条件就会显示符合条件的数据包, and 表示左右 2 个条件都满足才会显示。

or



and



我们可以很直观的看到结果的不同。

## 5.3 实战：使用 Wireshark 对常用协议抓包并分析原理

协议分析的时候我们关闭混淆模式, 避免一些干扰的数据包存在。

### 5.3.1 常用协议分析-ARP 协议

地址解析协议 (英语: Address Resolution Protocol, 缩写: ARP) 是一个通过解析网络层地址来寻找数据链路层地址的网络传输协议, 它在 IPv4 中极其重要。ARP 是通过网络地址来定位 MAC 地址。

开始抓包---过滤 arp



我们使用 nmap 来基于 ARP 协议进行扫描

```
root@xuegod53:~# nmap -sn 192.168.1.1
```

请加学神 IT 教育官方 QQ 群: 869961923 或李老師 QQ: 3147296050 領取更多資料

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

151 Vmware_8b:2b:b8	74.304928302	Broadcast	ARP	42 Who has 192.168.1.1? Tell 192.168.1.53
152 Fenglian_3b:4b:03	74.305860057	Vmware_8b:2b:b8	ARP	60 192.168.1.1 is at a4:56:02:3b:4b:03

我们看一下我们抓取到的数据包

分析第一个请求包

- › Frame 151: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
- › Ethernet II, Src: Vmware\_8b:2b:b8 (00:0c:29:8b:2b:b8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- › Address Resolution Protocol (request)

查看 Address Resolution Protocol (request) ARP 请求包内容:

```
▼ 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_8b:2b:b8 (00:0c:29:8b:2b:b8)
  Sender IP address: 192.168.1.53
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.1.1
```

**Address Resolution Protocol (request)**      #ARP 地址解析协议 request 表示请求包

**Hardware type: Ethernet (1)**      #硬件类型

**Protocol type: IPv4 ( 0x0800 )**      #协议类型

**Hardware size: 6**      #硬件地址

**Protocol size: 4**      #协议长度

**Opcode: request ( 1 )**      #操作码, 该值为 1 表示 ARP 请求包

**Sender MAC address: Vmware\_96:67:52 (00:0c:29:8b:2b:b8)**      #源 MAC 地址

**Sender IP address: 192.168.1.53 .**      #源 IP 地址

**Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)**      #目标 MAC 地址

**Target IP address: 192.168.1.1**      #目标 IP 地址

我们来分析第二个数据包 ARP 的应答数据包

- › Frame 152: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
- › Ethernet II, Src: Fenglian\_3b:4b:03 (a4:56:02:3b:4b:03), Dst: Vmware\_8b:2b:b8 (00:0c:29:8b:2b:b8)
- › Address Resolution Protocol (reply)

查看: Address Resolution Protocol (reply) ARP 地址解析协议

```
▼ Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: Fenglian_3b:4b:03 (a4:56:02:3b:4b:03)
  Sender IP address: 192.168.1.1
  Target MAC address: Vmware_8b:2b:b8 (00:0c:29:8b:2b:b8)
  Target IP address: 192.168.1.53
```

**Address Resolution Protocol (reply)**      #ARP 地址解析协议 reply 表示回复包

**Hardware type: Ethernet (1)**      #硬件类型

**Protocol type: IPv4 ( 0x0800 )**      #协议类型

**Hardware size: 6**      #硬件地址

**Protocol size: 4**      #协议长度

**Opcode: reply ( 2 )**      #操作码, 该值为 2 表示 ARP 回复包

**Sender MAC address: XXXXXXXXXXXX (a4:56:02:3b:4b:03)**      #源 MAC 地址

请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

Sender IP address: 192.168.1.1 . #源 IP 地址  
Target MAC address: 00:00:00\_00: 00:00 (00:0c:29:8b:2b:b8) #目标 MAC 地址  
Target IP address: 192.168.1.53 #目标 IP 地址

总结: 我们可以看到到应答包补全了自己的 MAC 地址, 目的地址和源地址做了替换  
我们再来看两个数据包的需求和过程

```
Who has 192.168.1.1? Tell 192.168.1.53
192.168.1.1 is at f0:98:38:b8:e0:92
```

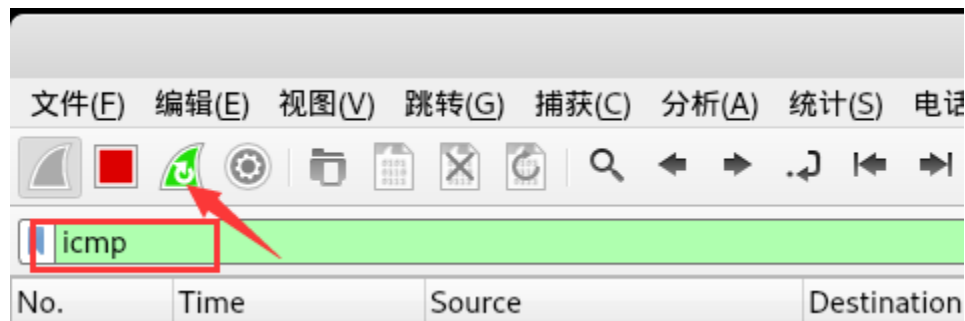
192.168.1.53 广播: 谁有 192.168.1.1 的 MAC 地址?

192.168.1.1 应答: 192.168.1.1 的 MAC 地址是 xxxxxxxxxxxx

很有趣的一个过程不是吗?

### 5.3.2 常用协议分析-ICMP 协议

我们把之前的数据包清空掉然后筛选 ICMP 协议的数据包



打开一个终端

```
root@xuegod53:~# ping xuegod.cn -c 1
```

我们只发送一个 ping 包, 方便我们分析发送完之后停止抓包即可。

No.	Time	Source	Destination
12	192.168.1.53	4.213770065	101.200.128.35
13	101.200.128.35	4.243491546	192.168.1.53

我们先看请求包的内容我们可以看到这是个 4 层的协议包

```
> Frame 51: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
> Ethernet II, Src: Vmware_96:67:52 (00:0c:29:96:67:52), Dst: HuaweiTe_b8:e0:92 (f0:98:38:b8:e0:92)
> Internet Protocol Version 4, Src: 192.168.1.53, Dst: 101.200.128.35
> Internet Control Message Protocol
```

下面我们开始分析 ICMP 协议包:

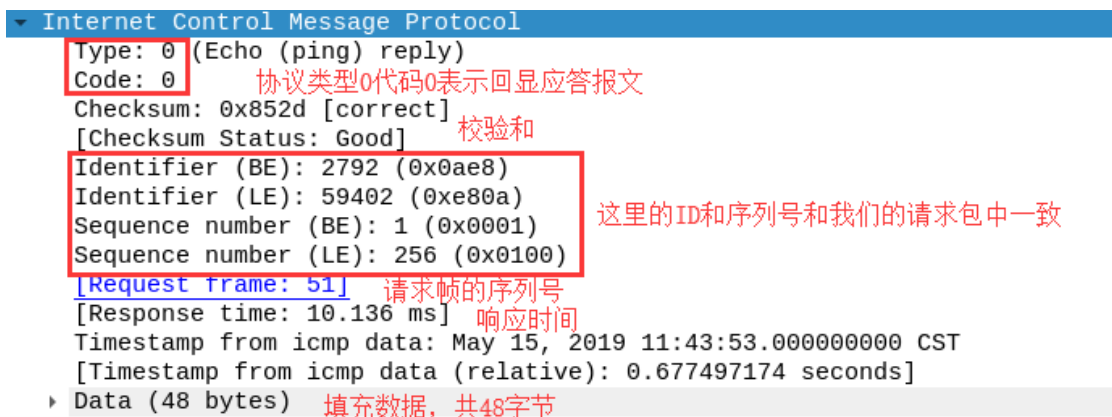
ICMP 协议分析请求包

```
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0 协议类型8, 代码0 表示回显请求, 即Ping请求
Checksum: 0x7d2d [correct] 校验和, 用于检查错误的数据
[Checksum Status: Good] 校验状态 Good
Identifier (BE): 2792 (0x0ae8)
Identifier (LE): 59402 (0xe80a) ID值, 在应答包中返回该字段
Sequence number (BE): 1 (0x0001)
Sequence number (LE): 256 (0x0100) 序列号依旧在应答包中返回该字段
[Response frame: 52] 响应帧的序列号: 52
Timestamp from icmp data: May 15, 2019 11:43:53.000000000 CST
[Timestamp from icmp data (relative): 0.667360808 seconds]
Data (48 bytes) 填充数据, 共48字节
```

ICMP 协议分析应答包

请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料



工作过程:

本机发送一个 ICMP Echo Request 的包

接受方返回一个 ICMP Echo Reply, 包含了接受到数据拷贝和一些其他指令

### 5.3.3 常用协议分析-TCP 协议

首先是清空数据包然后筛选 tcp 开始抓包



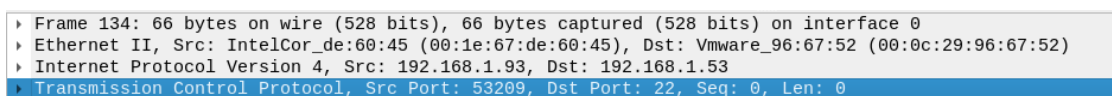
我们模拟一下 tcp 会话建立, 那最简单的方式是什么呢?

我们通过 Xshell 远程连接 Kali Linux 就会捕获到完整的 TCP3 次握手的链接。



抓完数据包之后我们就停止抓包, 接下来我们开始分析 TCP 的数据包

TCP 协议最核心的概念无非就是 3 次握手 4 次断开, 我们先讲 TCP 的 3 次握手



查看 TCP 协议:

我们先来看第一个数据包 SYN 数据包

请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料

```
Transmission Control Protocol, Src Port: 53209, Dst Port: 22, Seq: 0, Len: 0
Source Port: 53209 源端口
Destination Port: 22 目的端口
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number) 序列号
[Next sequence number: 0 (relative sequence number)] 确认序列号
Acknowledgment number: 0
1000 .... = Header Length: 32 bytes (8) 头部长度
Flags: 0x002 (SYN) 标志位SYN
Window size value: 64240
[Calculated window size: 64240] Windows窗口大小
Checksum: 0xdafb [unverified] 校验和
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
[Timestamps]
```

下面这样图是打开标志位的详细信息

```
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 SYN=1表示发起一个链接请求
.... ....1. = Syn: Set
.... .......0 = Fin: Not set
[TCP Flags: .....S.] TCP Flags = S 表示SYN类型
```

我们从以上信息就可以看出这是一个 SYN 数据包, SYN=1 表示发送一个链接请求。这时 Seq 和 ACK 都是 0

我们分析第二个数据包

```
Transmission Control Protocol, Src Port: 22, Dst Port: 53209, Seq: 0, Ack: 1, Len: 0
Source Port: 22 源端口
Destination Port: 53209 目的端口
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number) 序列号
[Next sequence number: 0 (relative sequence number)] Seq=0
Acknowledgment number: 1 (relative ack number) Ack=1
1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK) 标志位SYN/ACK 这表示是TCP三次握手的第二个包
Window size value: 29200
[Calculated window size: 29200] 窗口大小
Checksum: 0x8409 [unverified] 校验和
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted
[SEQ/ACK analysis] 这里是Seq和ACK的数据
[Timestamps]
```

Flags 位信息

```
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 ACK=1 确认序列号有效
.... ....0 = Push: Not set
.... .....0.. = Reset: Not set
.... ....1. = Syn: Set SYN=1
.... .......0 = Fin: Not set
[TCP Flags: .....A..S.] 类型为SYN/ACK
```



▼ [SEQ/ACK analysis]

[This is an ACK to the segment in frame: 134] 表示ACK响应第134帧的请求  
[The RTT to ACK the segment was: 0.000064117 seconds]  
[iRTT: 0.000282739 seconds]

我们可以看到服务端收到 SYN 连接请求返回的数据包 SYN=1,ACK=1 表示回应第一个 SYN 数据包。

我们看第三个数据包

▼ Transmission Control Protocol, Src Port: 53209, Dst Port: 22, Seq: 1, Ack: 1, Len: 0

Source Port: 53209 源端口  
Destination Port: 22 目的端口

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 1 (relative sequence number) Seq=1 等于上一帧的确认序列号

[Next sequence number: 1 (relative sequence number)]

Acknowledgment number: 1 (relative ack number) ACK=1 确认序号1, 上一帧的序号+1

0101 .... = Header Length: 20 bytes (5)

▸ Flags: 0x010 (ACK) Flags=ACK

Window size value: 8212

[Calculated window size: 2102272]

[Window size scaling factor: 256]

Checksum: 0x8277 [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

▸ [SEQ/ACK analysis]

▸ [Timestamps]

▼ 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 ACK=1 确认序列号有效

.... .... .0.. = Reset: Not set

.... .... ..0. = Syn: Not set

.... .... ...0 = Fin: Not set

[TCP Flags: .....A....] A=ACK

▼ [SEQ/ACK analysis]

[This is an ACK to the segment in frame: 135] 响应第135帧

[The RTT to ACK the segment was: 0.000218622 seconds]

[iRTT: 0.000282739 seconds]

到这里三次握手过程就结束了。

我们生成一个图表来观察数据交互的过程



请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

Wireshark Statistics menu options:

- 捕获文件属性 (Ctrl+Alt+Shift+C)
- 已解析的地址
- 协议分级(P)
- 对话
- 端点
- 分组长度
- I/O 图表(I)
- 服务响应时间
- DHCP (BOOTP) Statistics
- ONC-RPC Programs
- 29West
- ANCP
- BACnet
- Collectd
- DNS
- 流量图
- HART-IP
- HPFEEDS
- HTTP
- HTTP2
- Sametime
- TCP 流图形

Packet Details for Transmission Control Protocol:

- Source Port: 53209
- Destination Port: 22
- [Stream index: 0]
- [TCP Segment Len: 0]
- Sequence number: 0 (relative sequence number)
- [Next sequence number: 0 (relative sequence number)]
- Acknowledgment number: 0
- 1000 .... = Header Length: 32 bytes (8)
- Flags: 0x002 (SYN)
- Window size value: 64240
- [Calculated window size: 64240]

点击显示过滤器

Wireshark - 流 - eth0

时间: 192.168.1.93, 192.168.1.53

2 节点, 34 项目

限制显示过滤器

流类型: All Flows

地址: 任何

Help, Close, Save As...

前面 3 个就是 TCP 建立链接的过程, 后面的就是相互通信的过程了这个时候 seq 就会根据数据包的大小改变。

时间	192.168.1.93	192.168.1.53
241.586583474	53209	53209 → 22 [SYN] Seq=0 Win=64240 Len=0
241.586647591	53209	22 → 53209 [SYN, ACK] Seq=0 Ack=1 Win=2920
241.586866213	53209	53209 → 22 [ACK] Seq=1 Ack=1 Win=2102

请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

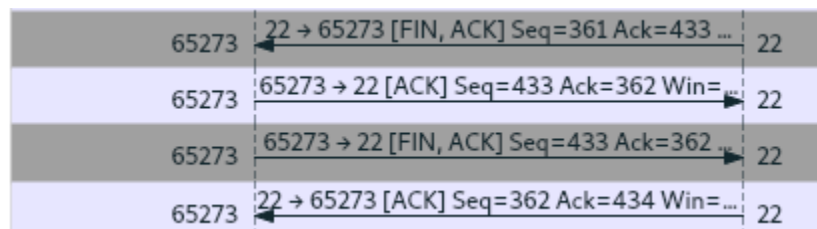
我们清空一下数据包来看一下断开链接是一个什么样的过程.



我们在 Xshell 窗口中输入 exit 退出

```
root@xuegod53:~# exit
注销
Connection closing...Socket close.
Connection closed by foreign host.
Disconnected from remote host(kali-xuegod53) at 14:00:25.
Type 'help' to learn how to use Xshell prompt.
[!C:\~]$
```

我们重新到 Wireshark 生成图标



请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料

我们分析一下过程, 我们在终端输入 EXIT 实际上是在我们 Kali 上执行的命令, 表示我们 SSHD 的 Server 端向客户端发起关闭链接请求。

第一次挥手: 服务端发送一个[FIN+ACK], 表示自己没有数据要发送了, 想断开连接, 并进入 FIN\_WAIT\_1 状态

第二次挥手: 客户端收到 FIN 后, 知道不会再有数据从服务端传来, 发送 ACK 进行确认, 确认序号为收到序号+1 (与 SYN 相同, 一个 FIN 占用一个序号), 客户端进入 CLOSE\_WAIT 状态。

第三次挥手: 客户端发送 [FIN+ACK] 给对方, 表示自己没有数据要发送了, 客户端进入 LAST\_ACK 状态, 然后直接断开 TCP 会话的连接, 释放相应的资源。

第四次挥手: 服务端收到了客户端的 FIN 信令后, 进入 TIMED\_WAIT 状态, 并发送 ACK 确认消息。服务端在 TIMED\_WAIT 状态下, 等待一段时间, 没有数据到来, 就认为对面已经收到了自己发送的 ACK 并正确关闭了进入 CLOSE 状态, 自己也断开了 TCP 连接, 释放所有资源。当客户端收到服务端的 ACK 回应后, 会进入 CLOSE 状态并关闭本端的会话接口, 释放相应资源。

### 5.3.4 常用协议分析-HTTP 协议

我们还是筛选 TCP 协议因为 HTTP 是 TCP 的上层协议, 所以我们过滤 TCP 的数据会包含 HTTP 协议的数据包



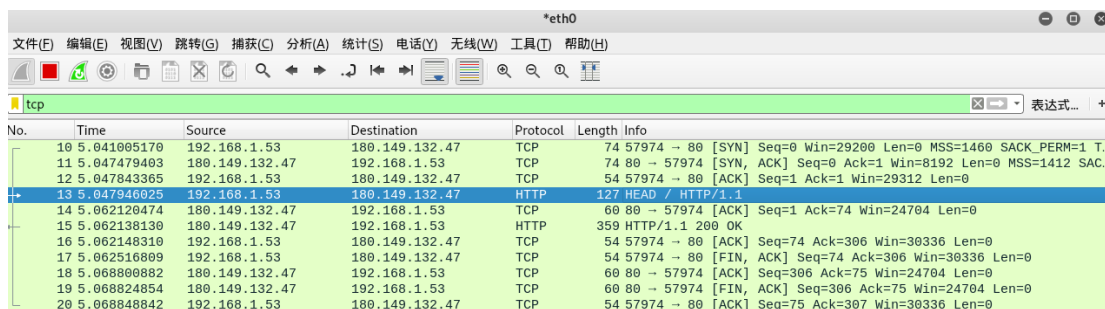
我们打开一个终端输入下面命令。

```
root@xuegod53:~# curl -I baidu.com
```

```
root@xuegod53:~# curl -I baidu.com
HTTP/1.1 200 OK
Date: Mon, 21 Dec 2020 03:07:56 GMT
Server: Apache
Last-Modified: Tue, 12 Jan 2010 13:48:00 GMT
ETag: "51-47cf7e6ee8400"
Accept-Ranges: bytes
Content-Length: 81
Cache-Control: max-age=86400
Expires: Tue, 22 Dec 2020 03:07:56 GMT
Connection: Keep-Alive
Content-Type: text/html
```

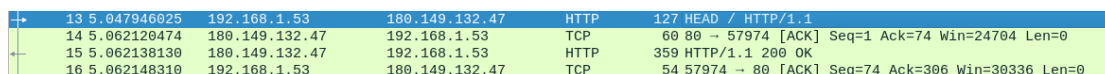
curl 是一个在命令行下工作的文件传输工具, 我们这里用来发送 http 请求  
-I 大写的 i 表示仅返回头部信息。

我们可以看到我们抓到了 TCP 的 3 次握手 4 次断开



No.	Time	Source	Destination	Protocol	Length	Info
10	5.041005170	192.168.1.53	180.149.132.47	TCP	74	57974 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 T...
11	5.047479403	180.149.132.47	192.168.1.53	TCP	74	80 → 57974 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1412 SAC...
12	5.047843365	192.168.1.53	180.149.132.47	TCP	54	57974 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0
13	5.047946025	192.168.1.53	180.149.132.47	HTTP	127	HEAD / HTTP/1.1
14	5.062120474	180.149.132.47	192.168.1.53	TCP	60	80 → 57974 [ACK] Seq=1 Ack=74 Win=24704 Len=0
15	5.062138130	180.149.132.47	192.168.1.53	HTTP	359	HTTP/1.1 200 OK
16	5.062148310	192.168.1.53	180.149.132.47	TCP	54	57974 → 80 [ACK] Seq=74 Ack=306 Win=30336 Len=0
17	5.062516809	192.168.1.53	180.149.132.47	TCP	54	57974 → 80 [FIN, ACK] Seq=74 Ack=306 Win=30336 Len=0
18	5.068800882	180.149.132.47	192.168.1.53	TCP	60	80 → 57974 [ACK] Seq=306 Ack=75 Win=24704 Len=0
19	5.068824854	180.149.132.47	192.168.1.53	TCP	60	80 → 57974 [FIN, ACK] Seq=306 Ack=75 Win=24704 Len=0
20	5.068848842	192.168.1.53	180.149.132.47	TCP	54	57974 → 80 [ACK] Seq=75 Ack=307 Win=30336 Len=0

#### 第 4 个和第 6 个是我们的 HTTP 数据包



No.	Time	Source	Destination	Protocol	Length	Info
13	5.047946025	192.168.1.53	180.149.132.47	HTTP	127	HEAD / HTTP/1.1
14	5.062120474	180.149.132.47	192.168.1.53	TCP	60	80 → 57974 [ACK] Seq=1 Ack=74 Win=24704 Len=0
15	5.062138130	180.149.132.47	192.168.1.53	HTTP	359	HTTP/1.1 200 OK
16	5.062148310	192.168.1.53	180.149.132.47	TCP	54	57974 → 80 [ACK] Seq=74 Ack=306 Win=30336 Len=0

第一步: 我们发送了一个 HTTP 的 HEAD 请求

第二步: 服务器收到我们的请求返回了一个 Seq/ACK 进行确认

第三步: 服务器将 HTTP 的头部信息返回给我们客户端 **状态码为 200 表示页面正常**

第四步: 客户端收到服务器返回的头部信息向服务器发送 Seq/ACK 进行确认

发送完成之后客户端就会发送 FIN/ACK 来进行关闭链接的请求。

## 5.4 实战: Wireshark 抓包解决服务器被黑上不了网

场景: 服务器被黑上不了网, 可以 ping 通网关, 但是不能上网。

模拟场景

修改主机 TTL 值为 1, 下面的方式是我们临时修改内核参数。

```
root@xuegod53:~# echo "1" > /proc/sys/net/ipv4/ip_default_ttl
```

拓展:

TTL: 数据报文的生存周期。

默认 linux 操作系统值: 64, 每经过一个路由节点, TTL 值减 1。TTL 值为 0 时, 说明目标地址不可达并返回: Time to live exceeded

作用: 防止数据包, 无限制在公网中转发。我们测试结果

```
root@xuegod53:~# ping 192.168.1.1 -c 1
```

```
root@xuegod53:~# ping 192.168.1.1 -c 1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.58 ms

--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.584/1.584/1.584/0.000 ms
```

```
root@xuegod53:~# ping xuegod.cn -c 1
```

```
root@xuegod53:~# ping xuegod.cn -c 1
PING xuegod.cn (101.200.128.35) 56(84) bytes of data.
From bogon (192.168.1.1) icmp_seq=1 Time to live exceeded

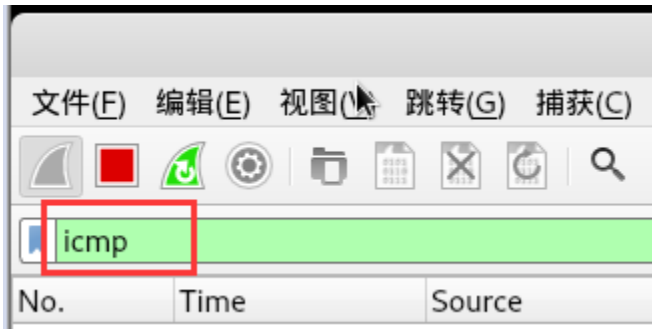
--- xuegod.cn ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

我们可以看到提示我们 Time to live exceeded 这表示超过生存时间,

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

我们判断和目标之间经过多少个网络设备是根据目标返回给我们的 TTL 值来判断的, 因为我们发送的数据包是看不到的。

下面我们来实战抓包分析数据包



开启抓包, 过滤 icmp 协议

```
root@xuegod53:~# ping xuegod.cn -c 1
```

然后回到 WireShark 中查看数据包

No.	Time	Source	Destination	Protocol	Length	Info
12	22.483128354	192.168.1.53	101.200.128.35	ICMP	98	Echo (ping) request id=0x0f0d, seq=1/256, ttl=1 (no response found!)
13	22.484264442	192.168.1.1	192.168.1.53	ICMP	126	Time-to-live exceeded (Time to live exceeded in transit)

由于图片较长下面给大家截取重要部分的信息提示截图的是 info 字段内容

Info
Echo (ping) request id=0x0f0d, seq=1/256, ttl=1 (no response found!)
Time-to-live exceeded (Time to live exceeded in transit)

我们可以看到第一个包是发送了一个 ping 请求包 ttl=1

然后呢我们收到了 192.168.1.1 返回给我们的数据包告诉我们超过数据包生存时间, 数据包被丢弃。

那我们把 TTL 值修改成 2 会有什么效果呢?

```
root@xuegod53:~# echo "2" > /proc/sys/net/ipv4/ip_default_ttl
```

```
root@xuegod53:~# ping xuegod.cn -c 1
```

No.	Time	Source	Destination	Protocol	Length	Info
9	11.044938098	192.168.1.53	101.200.128.35	ICMP	98	Echo (ping) request id=0x0f13, seq=1/256, ttl=1 (no response found!)
10	11.046094077	192.168.1.1	192.168.1.53	ICMP	126	Time-to-live exceeded (Time to live exceeded in transit)
17	14.798767635	192.168.1.53	101.200.128.35	ICMP	98	Echo (ping) request id=0x0f14, seq=1/256, ttl=2 (no response found!)
18	14.841159108	112.103.140.1	192.168.1.53	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

我们对比数据包发现返回我们数据包被丢弃的源地址变成了 123.115.0.1, 这证明了数据包在网络中已经到达了下一个网络设备才被丢弃, 由此我们还判断出我们的运营商网关地址为 123.115.0.1 但是我们并没有到达目标主机。

我们恢复系统内核参数

```
root@xuegod53:~# echo "64" > /proc/sys/net/ipv4/ip_default_ttl
```

```
root@xuegod53:~# ping xuegod.cn -c 1
```

```
root@xuegod53:~# ping xuegod.cn -c 1
PING xuegod.cn (101.200.128.35) 56(84) bytes of data.
64 bytes from xuegod.cn (101.200.128.35): icmp_seq=1 ttl=52 time=31.9 ms

--- xuegod.cn ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 31.878/31.878/31.878/0.000 ms
```

请加学神 IT 教育官方 QQ 群: 869961923 或李老師 QQ: 3147296050 领取更多资料

请加学神 IT 教育官方 QQ 群: 869961923 或乔老师 QQ: 2217978235 领取更多资料

目标返回给我们的 TTL 值为 52, 这表示我们的 TTL 值需要大于  $64-52=12$  才可以访问 xuegod.cn

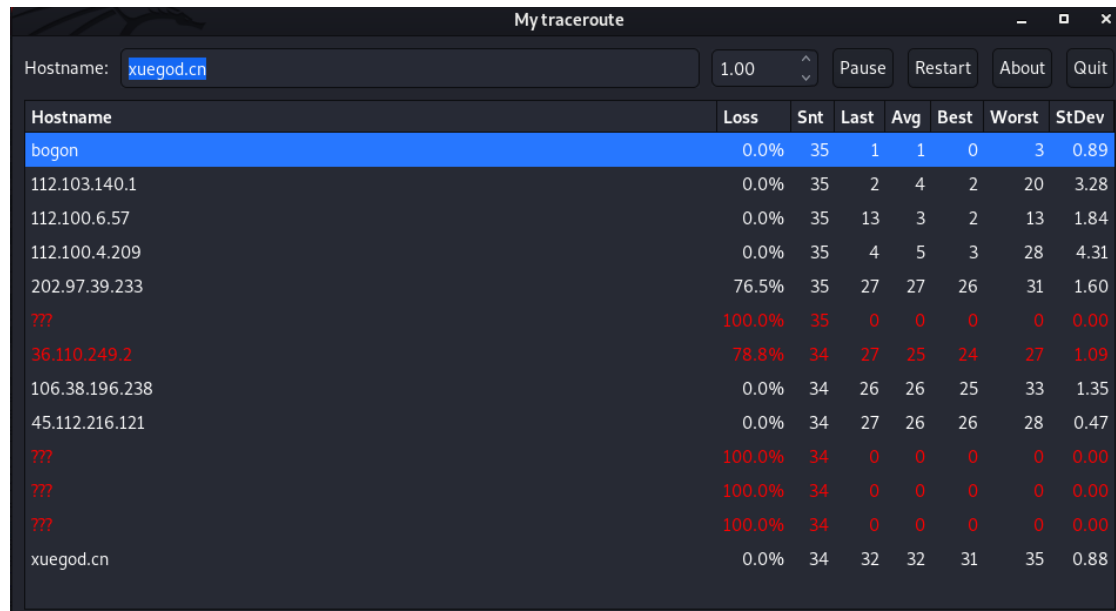
MTR 可以检测我们到达目标网络之间的所有网络设备的网络质量,

默认系统是没有安装 MTR 工具的我们手动安装一下

```
root@xuegod53:~# apt install -y mtr
```

例: 检测到达 xuegod.cn 所有节点的通信质量

```
root@xuegod53:~# mtr xuegod.cn
```



Hostname	Loss	Snt	Last	Avg	Best	Worst	StDev
bogon	0.0%	35	1	1	0	3	0.89
112.103.140.1	0.0%	35	2	4	2	20	3.28
112.100.6.57	0.0%	35	13	3	2	13	1.84
112.100.4.209	0.0%	35	4	5	3	28	4.31
202.97.39.233	76.5%	35	27	27	26	31	1.60
???	100.0%	35	0	0	0	0	0.00
36.110.249.2	78.8%	34	27	25	24	27	1.09
106.38.196.238	0.0%	34	26	26	25	33	1.35
45.112.216.121	0.0%	34	27	26	26	28	0.47
???	100.0%	34	0	0	0	0	0.00
???	100.0%	34	0	0	0	0	0.00
???	100.0%	34	0	0	0	0	0.00
xuegod.cn	0.0%	34	32	32	31	35	0.88

我们可以看到从我当前主机到目标主机之间经过 12 跳。

## 总结:

- 5.1 WireShark 简介和抓包原理及过程
- 5.2 实战: WireShark 抓包及快速定位数据包技巧
- 5.3 实战: 使用 WireShark 对常用协议抓包并分析原理
- 5.4 实战: WireShark 抓包解决服务器被黑上不了网

请加学神 IT 教育官方 QQ 群: 869961923 或李老师 QQ: 3147296050 领取更多资料