

Tutorial 8: Traffic Analysis and Network Layer Attacks Based on Wireshark

IS309 Network Security 2021 Spring

Chunyu Xue, Juntao Shen, Qianfei Ren

Student ID {518021910698, 518021911057, 518030910018}

Dicardo@sjsu.edu.cn, sjt166@sjsu.edu.cn, cordial.Daniel@sjsu.edu.cn

Department of Computer Science,
Shanghai Jiao Tong University, Shanghai, China

Abstract. A packet analyzer or packet sniffer is a computer program or computer hardware such as a packet capture appliance, that can intercept and log traffic that passes over a computer network or part of a network. Packet capture is the process of intercepting and logging traffic. As data streams flow across the network, the analyzer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to the appropriate RFC or other specifications¹. Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education². The network layer attacks consist of IP spoofing, hijacking, smurf, wormhole, blackhole, sybil and sinkhole.

Keywords: packet sniffer, Wireshark, traffic analysis, network layer attacks

1 Problem 1: Traffic Analysis

1.1 Problem Description

Please use “Wireshark” to perform a traffic analysis on www.cs.sjtu.edu.cn and tell us your finding.

Solution. In section 1.2, we briefly introduce a simple tutorial on how to use ”Wireshark” to operate traffic analysis on a certain website. In section 1.3, we discuss and analyse our findings when operating traffic analysis on www.cs.sjtu.edu.cn, including ICMP protocol used by Ping command and the entire process of establishing TCP connection between client and server.

1.2 The Wireshark Tutorial

When you open Wireshark without starting a capture or opening a capture file it will display the “Welcome Screen,” which lists any recently opened capture files and available capture interfaces. Network activity for each interface will be shown in a sparkline next to the interface name. It is possible to select more than one interface and capture from them simultaneously.³

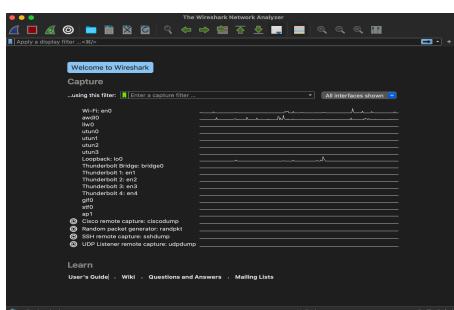


Fig. 1. Welcome screen of Wireshark

¹ Packet Analyzer: https://en.wikipedia.org/wiki/Packet_analyzer

² Wireshark: <https://en.wikipedia.org/wiki/Wireshark>

³ Wireshark Tutorial: https://www.wireshark.org/docs/wsug_html_chunked/ChCapInterfaceSection.html

Wireshark isn't limited to just network interfaces — on most systems you can also capture USB, Bluetooth, and other types of packets.

When you select Capture → Options (or use the corresponding item in the main toolbar), Wireshark pops up the “Capture Options” dialog box as shown in Fig 2, “The “Capture Options” input tab”. If you are unsure which options to choose in this dialog box, leaving the defaults settings as they are should work well in many cases.

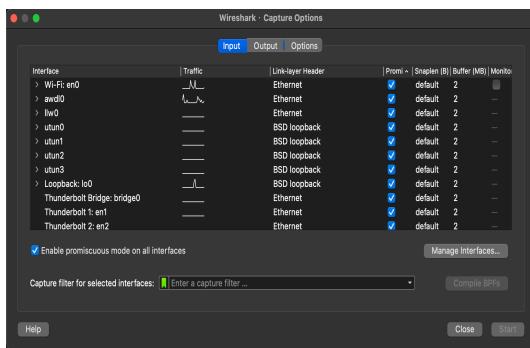


Fig. 2. “Capture Options” dialog box of Wireshark

The description of input/output(option) parameters is as follows:

Input:

- **Interface:** name of interface
 - **Traffic:** A sparkline showing network activity over time.
 - **Link-layer Header:** The type of packet captured by this interface. In some cases it is possible to change this.
 - **Promiscuous:** Lets you put this interface in promiscuous mode while capturing. Note that another application might override this setting.
 - **Snaplen:** The snapshot length, or the number of bytes to capture for each packet. You can set an explicit length if needed, e.g. for performance or privacy reasons.
 - **Buffer:** The size of the kernel buffer that is reserved for capturing packets. You can increase or decrease this as needed, but the default is usually sufficient.
 - **Monitor Mode:** Let you capture full, raw 802.11 headers. Support depends on the interface type, hardware, driver, and OS. Note that enabling this might disconnect you from your wireless network.
 - **Capture Filter:** The capture filter applied to this interface. You can edit the filter by double-clicking on it.

Output: Seen in Fig 3.

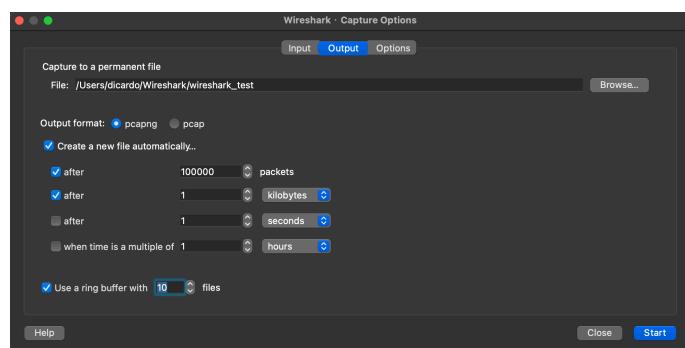


Fig. 3. Output options of Wireshark

Options: Seen in Fig 4.

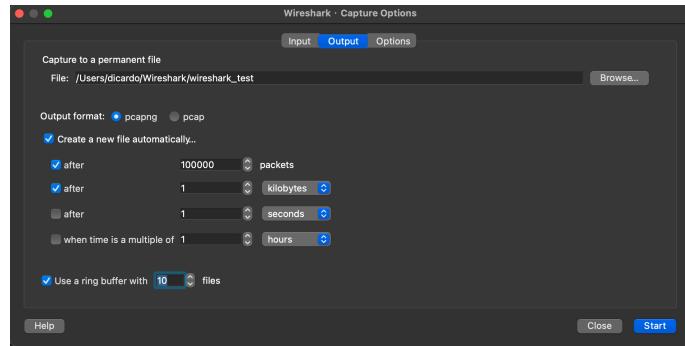


Fig. 4. Option settings of Wireshark

– Display Options:

- **Update list of packets in real-time:** Updates the packet list pane in real time during capture. If you do not enable this, Wireshark will not display any packets until you stop the capture. When you check this, Wireshark captures in a separate process and feeds the captures to the display process.
- **Automatically scroll during live capture:** Scroll the packet list pane as new packets come in, so you are always looking at the most recent packet. If you do not specify this Wireshark adds new packets to the packet list but does not scroll the packet list pane. This option is greyed out if “Update list of packets in real-time” is disabled.
- **Show capture information during capture:** If this option is enabled, the capture information dialog will be shown while packets are captured.

– Name Resolution

- **Resolve MAC addresses:** Translate MAC addresses into names.
- **Resolve network names:** Translate network addresses into names.
- **Resolve transport names:** Translate transport names (port numbers).

1.3 Traffic Analysis

In this section, we introduce two common ways to visit www.cs.sjtu.edu.cn (including Ping command and visit through Chrome browser) and discuss the protocols used in these processes based on the experimental results got from Wireshark.

Visit by Ping command We use the command ”ping -c 1 www.cs.sjtu.edu.cn” to ping the target website only once, and use Wireshark to listen to the messages exchange, the result is seen in Fig 5.

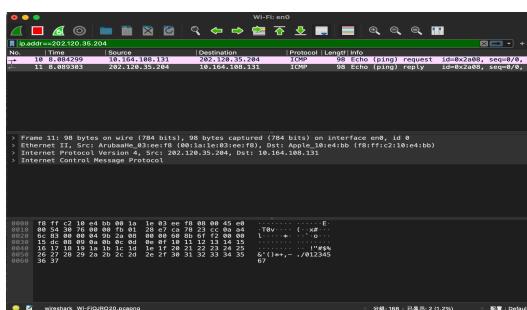


Fig. 5. Messages exchange in ping operation

We can know that the local computer (ip: 10.164.10.131) only has two ICMP messages exchange with target website (ip: 202.120.35.204), and these two messages are marked as Echo (ping) request / reply in Info segment. The reason is that Ping command is implemented based on ICMP protocol¹.

ICMP means **Internet Control Message Protocol**, which is a sub-protocol in TCP/IP group. ICMP is used in message exchange between IP hosts and routers.

The mechanism of the ping command is to echo the request and echo the response message. Specifically, it sends an ICMP message to another host on the network. If the specified host receives the message, it will send the message back.

Visit through Chrome browser We directly use the Chrome browser to visit www.cs.sjtu.edu.cn and use Wireshark to listen to the messages exchange, the result is seen in Fig 6 and Fig 7.

No.	Time	Source	Destination	Protocol	Length Info
338	16.559521	172.20.10.3	202.120.35.204	TCP	78 58425 - 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1468 WS=64 TSval=685767544 TSeср=0 SACK_PERM=1
341	16.634899	202.120.35.204	172.20.10.3	TCP	66 443 - 58425 [SYN, ACK] Seq=1 Win=1310 Len=0 MSS=1330 WS=256 SACK_PERM=1
342	16.634938	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1 Win=1310 Len=0
343	16.635686	172.20.10.3	202.120.35.204	TLSv1.2	571 Client Hello
346	16.698415	202.120.35.204	172.20.10.3	TLSv1.2	155 HelloRetryRequest
347	16.700009	202.120.35.204	172.20.10.3	TLSv1.2	64 ChangeCipherSpec
348	16.700054	172.20.10.3	202.120.35.204	TLSv1.2	54 58425 - 443 [ACK] Seq=510 Ack=109 Win=262816 Len=0
349	16.698994	172.20.10.3	202.120.35.204	TLSv1.2	577 ChangeCipherSpec, ClientHello
351	16.753582	202.120.35.204	172.20.10.3	TCP	54 443 - 58425 [ACK] Seq=100 Ack=1841 Win=68384 Len=0
352	16.754495	202.120.35.204	172.20.10.3	TLSv1.2	214 ServerHello
353	16.754498	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [PSH, ACK] Seq=268 Ack=1401 Win=68384 Len=1330 [TCP segment of a reassembled PDU]
354	16.754686	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1401 Ack=1598 Win=266068 Len=0
355	16.754684	202.120.35.204	172.20.10.3	TLSv1.2	126 ApplicationData
356	16.754684	202.120.35.204	172.20.10.3	TLSv1.2	281 ApplicationData
357	16.754748	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1401 Ack=2964 Win=260736 Len=0
358	16.768238	172.20.10.3	202.120.35.204	TLSv1.2	128 ApplicationData
359	16.761008	172.20.10.3	202.120.35.204	TLSv1.2	485 ApplicationData
378	16.893412	202.120.35.204	172.20.10.3	TCP	54 443 - 58425 [ACK] Seq=2964 Ack=1115 Win=68384 Len=0
376	16.868168	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=2964 Ack=1546 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
377	16.868238	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=4294 Win=268080 Len=0
378	16.868798	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=4294 Ack=6954 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
379	16.869018	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=6954 Ack=5234 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
380	16.869018	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=5034 Ack=1546 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
381	16.869966	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=6954 Win=268080 Len=0
382	16.869457	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=6954 Ack=1546 Win=68128 Len=1330 [TCP segment of a reassembled PDU]

Fig. 6. Messages exchange in Chrome browser(1)

No.	Time	Source	Destination	Protocol	Length Info
301	16.859618	172.20.10.3	202.120.35.204	TCP	1384 443 - 58425 [ACK] Seq=624 ACK=1546 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
301	16.859606	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=624 ACK=1546 Win=268080 Len=0
382	16.869457	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=6954 Ack=1546 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
383	16.869529	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=8234 Win=268080 Len=0
384	16.869938	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=8234 Ack=1546 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
385	16.869982	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=9614 Win=268080 Len=0
386	16.870289	202.120.35.204	172.20.10.3	TLSv1.2	1310 ApplicationData
387	16.870333	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=10878 Win=686864 Len=0
388	16.870734	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=1546 Ack=10878 Win=268080 Len=1330 [TCP segment of a reassembled PDU]
389	16.871111	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=12298 Win=268080 Len=0
390	16.871111	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=12298 Ack=1546 Win=68128 Len=1330 [TCP segment of a reassembled PDU]
391	16.871143	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=13538 Win=68080 Len=0
392	16.871597	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=13538 Ack=1546 Win=680840 Len=1330 [TCP segment of a reassembled PDU]
393	16.871643	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=14868 Win=680800 Len=0
394	16.872893	202.120.35.204	172.20.10.3	TCP	1384 443 - 58425 [ACK] Seq=14868 Ack=1546 Win=680640 Len=1330 [TCP segment of a reassembled PDU]
395	16.872182	202.120.35.204	172.20.10.3	TLSv1.2	394 ApplicationData
396	16.872124	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [ACK] Seq=1546 Ack=16528 Win=268416 Len=0
397	16.872124	202.120.35.204	172.20.10.3	TLSv1.2	78 ApplicationData
110	64.354312	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [FIN, ACK] Seq=1570 Ack=16578 Win=262144 Len=0
110	64.356605	172.20.10.3	202.120.35.204	TCP	18 [TCP Retransmission] 58425 - 443 [FIN, PSH, ACK] Seq=1546 Ack=16528 Win=262144 Len=0
112	64.360933	202.120.35.204	172.20.10.3	TLSv1.2	78 ApplicationData
113	64.360941	202.120.35.204	172.20.10.3	TCP	54 443 - 58425 [FIN, ACK] Seq=16552 Ack=1571 Win=81928 Len=0
114	64.360218	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [RST] Seq=1571 Win=0 Len=0
115	64.401234	202.120.35.204	172.20.10.3	TCP	66 [TCP Dup ACK 110#1] 443 - 58425 [ACK] Seq=16553 Ack=1571 Win=81920 Len=0 SLE=1546 SRE=1571
116	64.401334	172.20.10.3	202.120.35.204	TCP	54 58425 - 443 [RST] Seq=1571 Win=0 Len=0

Fig. 7. Messages exchange in Chrome browser(2)

Based on the experimental result above, we can conclude the entire process when using browser to visit website:

- **DNS.** After inputting www.cs.sjtu.edu.cn in the browser, DNS server will parse the domain name, find the corresponding IP address: 202.120.35.204, and find the path from client to server.
- **Three-way handshake connection.** The client (172.20.10.3) and the server (202.120.35.204) establish a TCP three-way handshake connection ²:

¹ Ping command based on ICMP protocol: https://blog.csdn.net/bian_cheng_ru_men/article/details/81476998

² Three-way handshake connection of TCP: <https://www.cnblogs.com/kaleidoscope/p/9701117.html>

- The client takes the initiative to open, and sends a connection request segment (No. 336), marking SYN as 1, and setting Sequence Number to x (TCP regulates that SYN = 1 cannot carry data, x is a randomly generated value), and then enter the SYN_SEND state
- The server receives the SYN message segment and confirms it (No. 341), and sets the SYN identification position to 1, ACK to 1, and Sequence Number to y, Acknowledgment Number is set to x + 1, and then enters SYN_RECV state (semi-connected state)
- The client confirms again (No. 342), sets ACK to 1 (no SYN is used at this time), Sequence Number to x + 1, Acknowledgment Number to y + 1 and sends it to the server. Finally, both the client and the server enter ESTABLISHED status
- **Hello messages.** After completing the establishment of the TCP connection, the client sends a **request attempt message "Client Hello"** (No. 343) to the server, and the server replies to the client with a **response attempt message "Hello Retry Request"** (No. 346)
- **Change Cipher Spec Message**¹. After completing the verification of the request and the attempt message (ie the above hello message), the server sends a Change Cipher Spec message (No. 347) to the client. Note that the current status is that the client and server have previously established an SSL Session, otherwise the Change Cipher Spec message is placed in the SSL handshake message.
- **Completion of Change Cipher Spec.** The client first returns an ACK (No. 348) to the server, and then returns a Client Hello Change Cipher Spec message (No. 349), indicating that the Cipher Spec has been modified.
- **Transmission of data.** The server first returns an ACK (No. 351) to the client, then returns a **hello message "Server Hello"**, and then begins the formal transmission of data.

Notice that many of the messages transmitted by the server are labeled with "TCP segment of a reassembled PDU". The reason is that when we transmit messages based on TCP, if for some reasons (such as exceeding MSS) TCP Segment cannot contain all application layer PDUs at one time, to divide a complete message into multiple segments, all other segments except the last segment will be marked with "TCP segment of a reassembled PDU".

Also, when the same ip visits the target website multiple times, the above message exchange will not occur. The guess is that the website content has been stored in the local cache.

¹ Change Cipher Spec Protocol: https://blog.csdn.net/weixin_43364172/article/details/83860247

2 Problem 2: Network Layer Attacks

2.1 Problem Description

Please try to test at least three network layer attacks, including **Wi-Fi jamming attack**, **IP spoofing attack**, **MAC spoofing attack**, **DHCP spoofing attack** and so on. (Note: please record the technical details and evidences.)

Solution. In section 2.2, we introduce the implementation of **DHCP Spoofing Attack** by using **bettercap**, a Man-in-the-MiddleAttack tool. In section 2.3, we discuss the implementation of **Mac Spoofing Attack (ARP Spoofing Attack)** by using bettercap as well. In section 2.5, we implement **IP Spoofing Attack** with **sendip** tool. Finally, in section 2.5, we introduce how to opreate **Wi-Fi Jamming Attack** with **ESP-8266-01S**.

2.2 DHCP Spoofing Attack

In this section, we introduce the implementation of DHCP Spoofing Attack by using **bettercap**, a Man-in-the-MiddleAttack tool¹.

Installation of Bettercap By operating ”**brew install bettercap**” command, we can successfully install bettercap in MacOS environment.

Run Bettercap By running ”**sudo bettercap**” command and enter the password for root, we can enter the **interface of bettercap**, seen in Fig 8. By running ”**help**” command in current interface, we can check the help messages and the situation of modules, seen in Fig 9.

```

dicardo@xuechunyudeMacBook-Pro ~ % sudo bettercap
Password:
bettercap v2.30.2 (built for darwin amd64 with go1.16.2) [type 'help' for a list of commands]
192.168.31.0/24 > 192.168.31.229 > |

```

Fig. 8. Interface of bettercap

```

Modules
any.proxy > not running
api.rest > not running
arp.spooft > not running
ble.recon > not running
c2 > not running
caplets > not running
dhcp6.spooft > not running
dns.spooft > not running
events.stream > running
gps > not running
hid > not running
http.proxy > not running
http.server > not running
https.proxy > not running
https.server > not running
mac.changer > not running
mdns.server > not running
mysql.server > not running
net.probe > not running
net.recon > not running
net.sniff > not running
packet.proxy > not running
syn.scan > not running
tcp.proxy > not running
ticker > not running
ui > not running
update > not running
wifi > not running
wol > not running
192.168.31.0/24 > 192.168.31.229 > |

```

Fig. 9. Modules situation in bettercap

¹ Bettercap Tutorial: <https://blog.csdn.net/u012570105/article/details/80561778>

Target Machine Settings Set the DNS server address of the target machine (which will be attacked) to the IP address of our attack host, which will fake our attack host to the DNS server. Note that the target machine must be in the same network segment as our attack host.

Get Real IP Address in Target Machine for Website Let our target machine be a computer with Win10 operating system. We first test the IP address of www.koudai8.com by using ping command on target machine, seen as Fig 10. We can know that the real IP address of this website is **49.232.222.175**.

```
C:\Users\15531>
C:\Users\15531>ping www.koudai8.com

正在 Ping www.koudai8.com [49.232.222.175] 具有 32 字节的数据:
来自 49.232.222.175 的回复: 字节=32 时间=41ms TTL=110
来自 49.232.222.175 的回复: 字节=32 时间=37ms TTL=110
来自 49.232.222.175 的回复: 字节=32 时间=35ms TTL=110
来自 49.232.222.175 的回复: 字节=32 时间=38ms TTL=110

49.232.222.175 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 = 35ms, 最长 = 41ms, 平均 = 37ms
```

Fig. 10. Real IP from ping command

Fake IP Address with DHCP Spoofing Attack Operate following commands on attack host to implement DHCP Spoofing:

- set dns.spoof.domains www.koudai8.com
- set dns.spoof.address 1.1.1.1
- dns.spoof on

When we run ping command again on target machine, we can notice the following message shown in attack host in Fig 11. The output on target machine is in Fig 12, we can know that the fake IP address from DHCP Spoofing Attack is **1.1.1.1**. Therefore, **the DHCP Spoofing Attack is successfully completed!**

```
[92.168.31.0/24 > 92.168.31.229] [22:59:56] [syslog] [inf] dns_spoof sending spoofed DNS reply for www.Koudai8.com (->1.1.1.1) to 92.168.31.251 : 3c:95:09:df:34:89 (Liteon Technology Corporation) - LAPTOP-F10PKTFC.local.
```

Fig. 11. Output in attack host when operating DHCP Spoofing Attack

```
C:\Users\15531>
C:\Users\15531>ping www.koudai8.com

正在 Ping www.koudai8.com [1.1.1.1] 具有 32 字节的数据:
来自 1.1.1.1 的回复: 字节=32 时间=202ms TTL=41
来自 1.1.1.1 的回复: 字节=32 时间=201ms TTL=41
来自 1.1.1.1 的回复: 字节=32 时间=203ms TTL=41
来自 1.1.1.1 的回复: 字节=32 时间=200ms TTL=41

1.1.1.1 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 = 200ms, 最长 = 203ms, 平均 = 201ms
```

Fig. 12. Fake IP address from ping command

2.3 Mac Spoofing Attack (ARP Spoofing Attack)

In this section, we discuss the implementation of Mac Spoofing Attack (ARP Spoofing Attack) by using **bettercap** as well.

Initialization of Bettercap Same as section 2.2.

Get Real Mac Address of Router Let our target machine be a computer with Win10 operating system. We first query the IP address of the router in our LAN by using "arp -a" command. From Fig 13 we can know that the real MAC address of the router (192.168.31.1) is 9c-9d-7e-51-3f-2f.

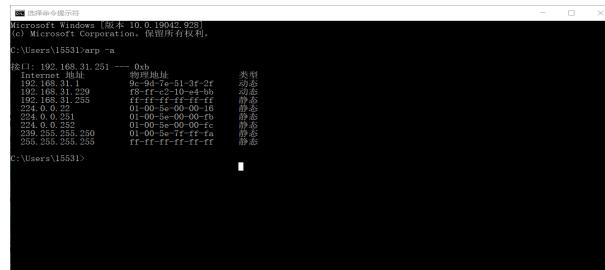


Fig. 13. Real MAC address of the router in LAN

Fake MAC Address of Router We use the MAC address of our attack host to fake. First, we use command "`net.recon on`" to search the IP address of our target machine in Fig 14, here is **192.168.31.251**. Then, we use the command "`set arp.spoof.targets 192.168.31.251`" to set the target of our ARP Spoofing Attack (if not specified, all hosts in LAN by default). After that, we enter "`arp.spoof on`" to start the attack, the result is in Fig 15. The fake MAC address of the router seen by target machine is **f8-ff-c2-10-e4-bb**, same as the attack host (192.168.31.229). Therefore, **the Mac Spoofing Attack (ARP Spoofing Attack) is successfully completed!**

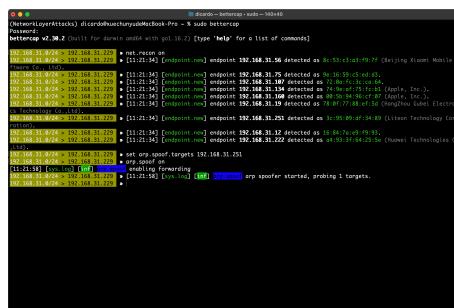


Fig. 14. Get IP Address of Target Machine

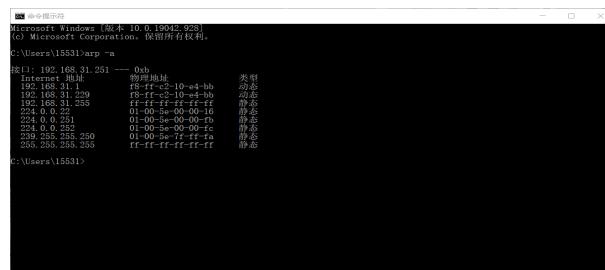


Fig. 15. Fake MAC Address of the router in LAN seen by target machine

2.4 IP Spoofing Attack

In this section, we implement IP Spoofing Attack with sendip tool.

Preparations Create two virtual machines, then check their IP address and MAC by using "ifconfig" command.

- VM1: IP address: **192.168.56.102**, MAC: **08:00:27:98:ab:71**
- VM2: IP address: **192.168.56.103**, MAC: **08:00:27:31:7e:0b**

Ping with Real IP Address We first ping VM2 from VM1 by using "sudo sendip -v -p ipv4 -is 192.168.56.102 -id 192.168.56.103 -p icmp -d 0x66666666 192.168.56.103" command, then check messages exchange in VM2 with Wireshark, the result is in Fig 16 and Fig ???. Note that there exists ICMP exchange (two messages) between VM1 and VM2.

```
sjt@sjt-VirtualBox:~$ sudo sendip -v -p ipv4 -is 192.168.56.102 -id 192.168.56.103
[sudo] password for sjt:
Added 25 options
Initializing module ipv4
Initializing module icmp
Finalizing module icmp
Finalizing module ipv4
Final packet data:
45 00 00 1C  E...
25 20 00 00  % ...
FF 01 A4 A2  ....
C0 AB 38 66  ..8f
C0 AB 38 67  ..8g
08 00 2B 33  ...3
66 66 66 66  ffff
Sent 28 bytes to 192.168.56.103
Freeing module ipv4
Freeing module icmp
```

Fig. 16. Output of sendip command

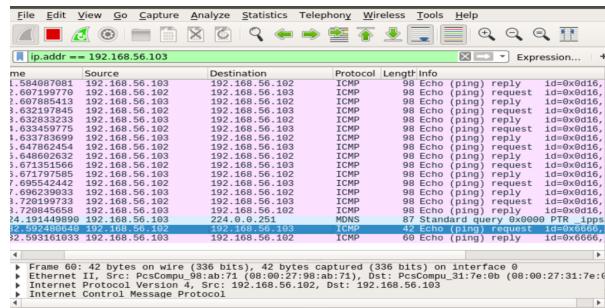
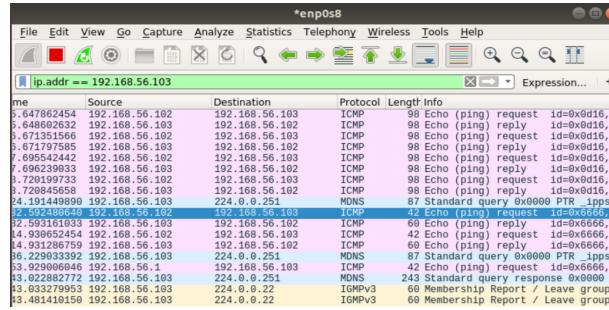


Fig. 17. Check messages exchange in VM2 with Wireshark

Ping with Fake IP Address We modify the source IP address to **192.168.56.1** and test the command again, the result is in Fig 18 and Fig 19. Note that there doesn't exist ICMP exchange, since the response message from VM2 is sent to the fake IP address, not VM1 itself. Therefore, **the IP Spoofing Attack is successfully completed!**

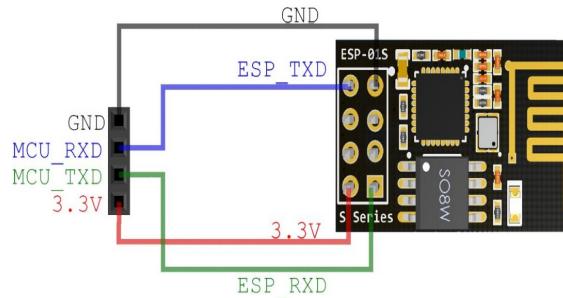
```
sitesjt-VirtualBox:~$ sudo sendip -v -p ipv4 -is 192.168.56.1 -td 192.168.56.103
3 -p icmp -d 0x66666666 192.168.56.103
Added 25 options
Initializing module ipv4
Initializing module icmp
Finalizing module icmp
Finalizing module ipv4
Final packet data:
45 00 00 1C   E...
88 AA 00 00   ...
FF 01 41 7D   ..A}
C0 A8 38 01   ..8.
C0 A8 38 07   ..8g
08 00 2B 33   ...+
66 66 66 66   ffff
Sent 28 bytes to 192.168.56.103
Freeing module ipv4
Freeing module icmp
```

Fig. 18. Output of sendip command**Fig. 19.** Check messages exchange in VM2 with Wireshark

2.5 Wi-Fi Jamming Attack

In this section, we introduce how to operate Wi-Fi Jamming Attack with ESP-8266-01S.

Preparations We bought a ESP-8266-01S for our attack and given the fact that it only has pin instead of plugins, we also deployed a USB-TTL transmitter. After adapting the workload to 3.3V, we could connect it as Fig 20.

**Fig. 20.** Overlook of ESP-8266-01S

Drivers and Connections When we connect ESP-8266 to our laptop, we need to use a serial communication tool to test whether it is successfully connected. In this experiment, our ESP-8266 supports AT commands, so after connecting, we send the following AT commands to test the connection status. After normal communication, the computer still needs to install the driver to correctly identify ESP-8266. The driver installation can be found in <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>.



Fig. 21. AT commands between laptop and ESP-8266

Deploy on the Arduino If you have not installed arduino, you need to install the software first, and then add the ESP-8266 library manager in the preferences. This part of the settings can be found in the arduino open source community at http://arduino.esp8266.com/stable/package_esp8266com_index.json. Next, we can find the ESP-8266 driver from the library manager, so click to install.

Code Part In this part of the code, we borrowed SpacehuhnTech's main frame and made some changes to our machine model. In the code, we created a new WIFI node to connect to a mobile phone or computer, and control the ESP-8266 by logging in to the local IP. After compiling and burning into the chip, when the program is executed, we can see the newly added network called "pwned" in the WIFI network.

Then enter 192.168.4.1, log in to our control menu homepage, and you can see some tips from SpacehuhnTech on security.



Fig. 22. Log in pwned WIFI

After that, scan all available WIFI networks and select the target we want to attack.

After selecting the attack mode, the corresponding WIFI can be attacked, and the network will be disconnected.



Fig. 23. Scanning



Fig. 24. Attack

Practical Deployment In this part, we show the wiring and switching details of our chip in the actual environment. It should be noted that for the 01S type ESP-8266, we need to connect pin 3 to the point source to enable it.

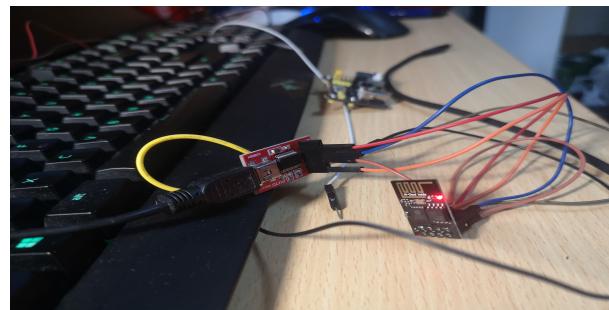


Fig. 25. Devices

3 Contributions

- **Problem 1**
 - **Installation and simple usage of Wireshark:** Chunyu Xue, Juntao Shen
 - **Traffic Analysis:** Chunyu Xue
- **Problem 2**
 - **DHCP Spoofing Attack:** Chunyu Xue
 - **MAC Address Spoofing Attack:** Chunyu Xue
 - **IP Spoofing Attack:** Juntao Shen
 - **Wi-Fi Jamming Attack:** Qianfei Ren
- **Latex report:** Chunyu Xue