SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov

#sf23eu

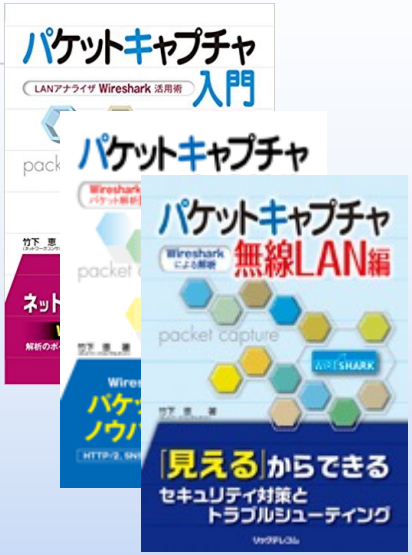
Chat GPT with Wireshark

The sample prompts and answers and trace files are
https://www.ikeriri.ne.jp/sharkfest/ChatGPTwithWireshark.zip

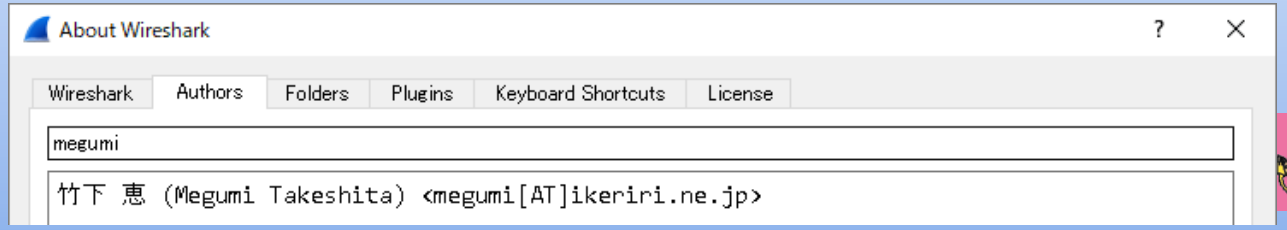
Megumi Takeshita
Ikeriri network service

Megumi Takeshita, packet otaku

SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov
#sf23eu



- Founder, ikeriri network service co., ltd
- Reseller of CACE technologies in 2008
- Worked SE/IS at BayNetwork, Nortel
- Wrote 10+ books about Wireshark
- Instruct Wireshark to JSDF and other company
- lecturer of CHUO University
- Reseller of packet capture / wireless-tools
- One of the contributors to Wireshark
  Translate Wireshark into Japanese



Session Details

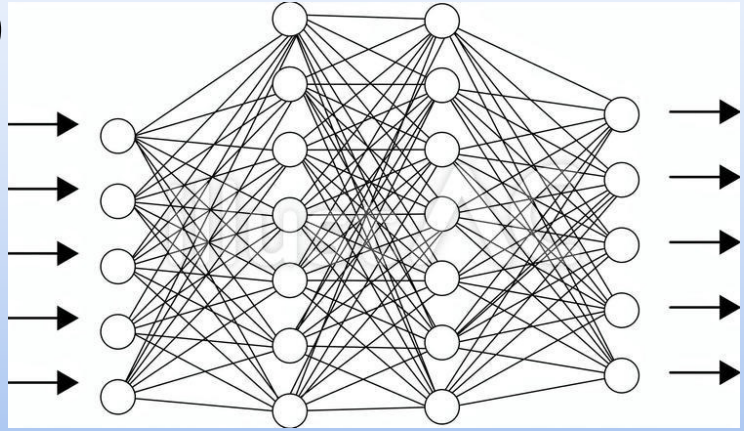
ChatGPT is trending in the IT world. Machine learning is one of the most evolved topics. Convolution / Recurrent neural network with supervised learnings are already used for many AI-based systems such as Gmail Smart Reply. ChatGPT is one of deep learning system, but we do not need difficult theory and complicated API library, we just need text/code (and URL) in any language!! How about Wireshark?

This session uses ChatGPT for Wireshark/tshark. We use ChatGPT for packet manipulation, display filtering, capture filtering, creating graphs/charts, and analysing trace files with Wireshark.

ChatGPT is not a perfect capture assistant, but it works as nice packet doctor if you indicate in a proper way. use Wireshark with ChatGPT and get better trace file analysis!!

LLM(Large Language Model) in a slide

- LLM uses Calculus and gradient descent algorithms, transform input many times, and then creates output. Then pick up interesting points (characteristics) automatically.
- We need many samples (Inputs, Outputs), then compare the current prompt (supervised learning)
- Creates a model consisting of tons parameters (ex. calculation units Input, Recurrent and Output cells) called as Recurrent Neural Network or Generative Adversarial Network



- We can use via python library sci-kit, keras and so on

ChatGPT https://chat.openai.com/

- Most popular Chatbot by OpenAI
  than Microsoft co-pilot, Google Bard, NotionAI etc.

Item	Free ChatGPT 3.5	Paid ChatGPT 4 20$/mo
Model precision	Intentionally nerfed 175 billion parameters	57 times smarter 100 trillion parameters
Response time	Slower	Priority access
Request limits	n/a (GPT-3.5)	50 times/3 hours (GPT-4)
Token limits	4096-16384 tokens About 1365-5461 chars	8192-32768 tokens About 2730-10922 chars
others		Additional plugins (web services) Advanced data analysis (Python, execution, image processing etc.)

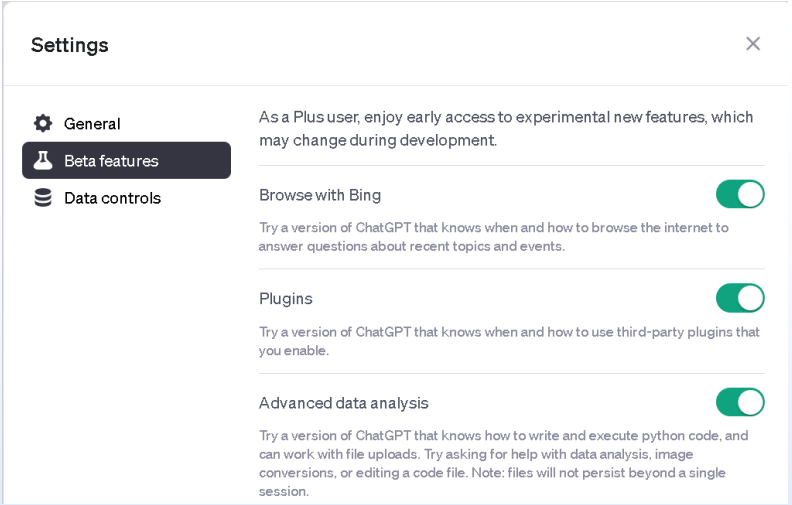
- If you want to dissect trace files, we need ChatGPT Plus.

Common knowledge for generative AI

- The answer is not always true if it seems to be true. (a.k.a Hallucination, especially for the shortage of learning)
- Answers may be different, though the prompt is the same. (because of time, place, language, learning, etc.)
- Open AI's ChatGPT is a native speaker of English, we can ask for Japanese and other languages, but the precisions of the answer are lower than English.
- Browsing functions and plugins (WebPilot, etc.) are useful. ChatGPT could use the Bing search engine and outer plugins to create the latest and correct answer.
- Iteration is important, ChatGPT may make a better one. We try to change the prompt in the sequence.

Settings of demonstration

- Use ChatGPT 4.0 (ChatGPT Plus)
- Bata features section
  Browse with Bing: on
  Plugins: on
  Advanced data analysis

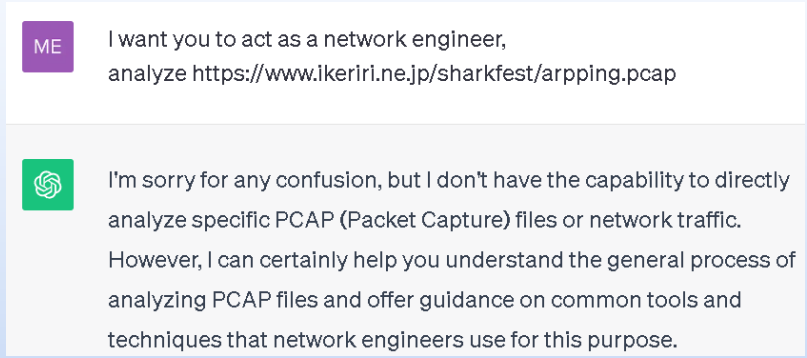


- The sample prompts and answer and trace file(pcapng,txt)

https://www.ikeriri.ne.jp/sharkfest/ChatGPTwithWireshark.zip

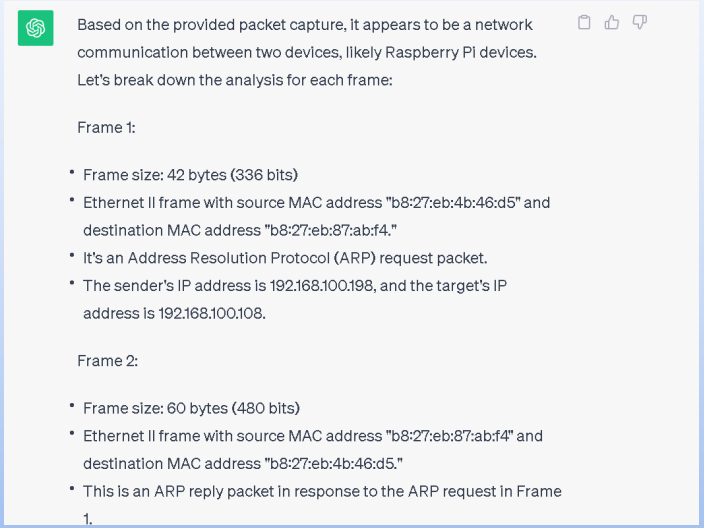
To read pcap/pcapng to GPT

- We can send text and some images into ChatGPT, but we cannot input trace files(pcap/pcapng) if we use URLs
  1:"Please analyze https://www.ikeriri.ne.jp/sharkfest/arpping.pcap"
- Creating text file
  File>Expert Packet Dissections >As Plain Text… (Wireshark)
  tshark –T text -r trace.pcapng
- There are limitations of tokens, I recommend to filter packets and limit output lines.
  Filter, Mark, and create text by omitting Packet Bytes, setting Packet details "As displayed" and so on.



To read pcap/pcapng to GPT

- tshark –r arpping.pcapng –T text -V or –O arp,ip,icmp to create the text file arpping.txt
- Open the text file and add a line at the head, 2:“I want you to act as a network engineer, analyze below”



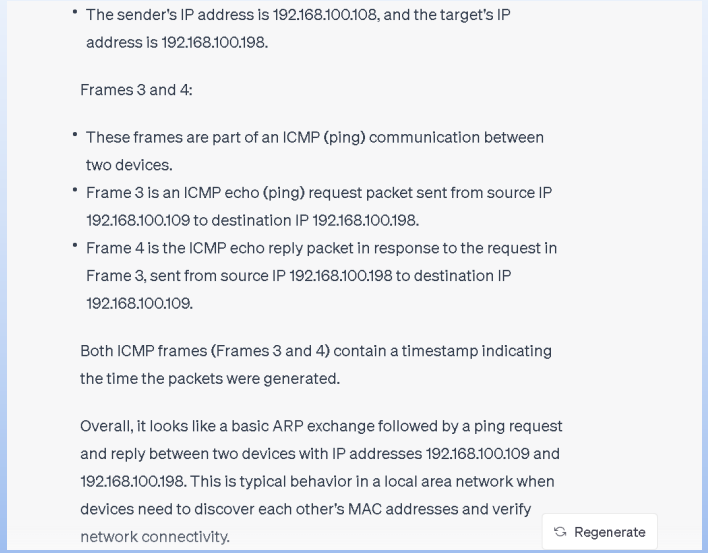
Based on the provided packet capture, it appears to be a network communication between two devices, likely Raspberry Pi devices. Let's break down the analysis for each frame:

Frame 1:

- Frame size: 42 bytes (336 bits)
- Ethernet II frame with source MAC address "b8:27:eb:4b:46:d5" and destination MAC address "b8:27:eb:87:ab:f4."
- It's an Address Resolution Protocol (ARP) request packet.
- The sender's IP address is 192.168.100.198, and the target's IP address is 192.168.100.108.

Frame 2:

- Frame size: 60 bytes (480 bits)
- Ethernet II frame with source MAC address "b8:27:eb:87:ab:f4" and destination MAC address "b8:27:eb:4b:46:d5."
- This is an ARP reply packet in response to the ARP request in Frame 1.



- The sender's IP address is 192.168.100.108, and the target's IP address is 192.168.100.198.

Frames 3 and 4:

- These frames are part of an ICMP (ping) communication between two devices.
- Frame 3 is an ICMP echo (ping) request packet sent from source IP 192.168.100.109 to destination IP 192.168.100.198.
- Frame 4 is the ICMP echo reply packet in response to the request in Frame 3, sent from source IP 192.168.100.198 to destination IP 192.168.100.109.

Both ICMP frames (Frames 3 and 4) contain a timestamp indicating the time the packets were generated.

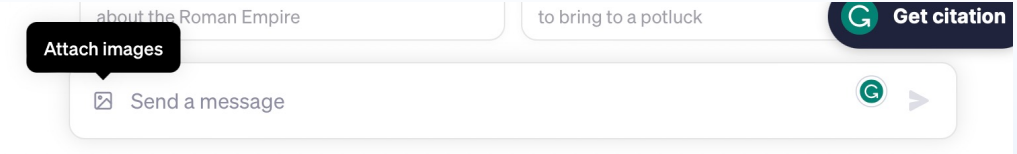
Overall, it looks like a basic ARP exchange followed by a ping request and reply between two devices with IP addresses 192.168.100.109 and 192.168.100.198. This is typical behavior in a local area network when devices need to discover each other's MAC addresses and verify network connectivity.

Regenerate

- It works and summarize the trace file.

To read pcap/pcapng to GPT

- We can upload image files instead of sending text prompt



- Open arpping.pcapng with Wireshark, take a screenshot, Click “Attach images” to upload the image of the trace file.
- And create prompt as “I want you to act as a network engineer, analyze below”

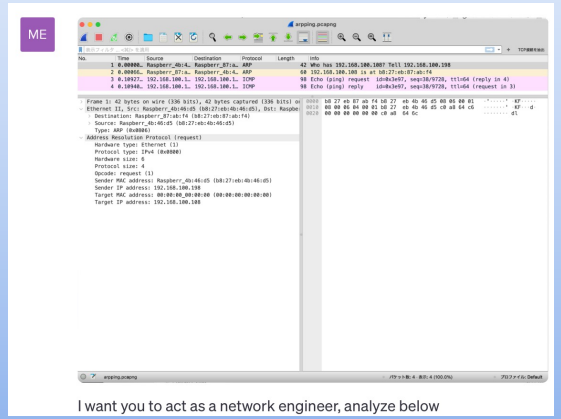
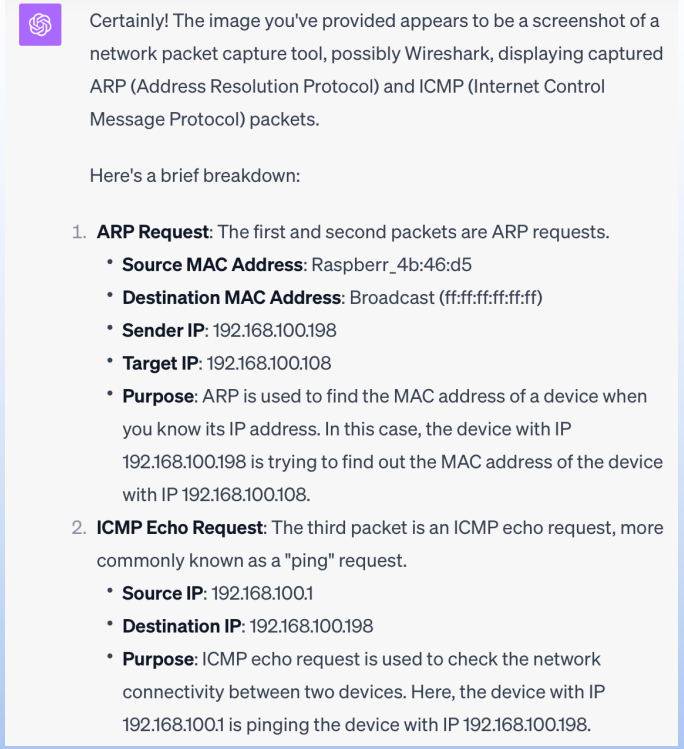


Image recognition of GPT4

- GPT4 analyze the traffic like text input



Certainly! The image you've provided appears to be a screenshot of a network packet capture tool, possibly Wireshark, displaying captured ARP (Address Resolution Protocol) and ICMP (Internet Control Message Protocol) packets.

Here's a brief breakdown:

1. **ARP Request**: The first and second packets are ARP requests.
   - **Source MAC Address**: Raspberr_4b:46:d5
   - **Destination MAC Address**: Broadcast (ff:ff:ff:ff:ff:ff)
   - **Sender IP**: 192.168.100.198
   - **Target IP**: 192.168.100.108
   - **Purpose**: ARP is used to find the MAC address of a device when you know its IP address. In this case, the device with IP 192.168.100.198 is trying to find out the MAC address of the device with IP 192.168.100.108.
2. **ICMP Echo Request**: The third packet is an ICMP echo request, more commonly known as a "ping" request.
   - **Source IP**: 192.168.100.1
   - **Destination IP**: 192.168.100.198
   - **Purpose**: ICMP echo request is used to check the network connectivity between two devices. Here, the device with IP 192.168.100.1 is pinging the device with IP 192.168.100.198.
3. **ICMP Echo Reply**: The fourth packet is an ICMP echo reply.
   - **Source IP**: 192.168.100.198
   - **Destination IP**: 192.168.100.1
   - **Purpose**: This is the response from the device with IP 192.168.100.198 to the earlier ICMP echo request. It indicates that the device is reachable and responding to network requests.

Analysis:

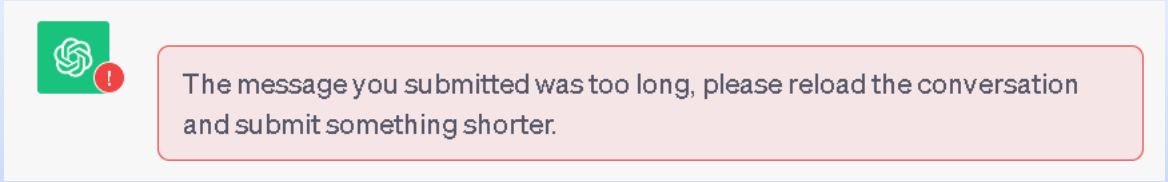
- The device with IP 192.168.100.198 (which seems to be a Raspberry Pi based on the MAC address prefix "Raspberr") is trying to communicate with the device at IP 192.168.100.108. It starts by trying to resolve the MAC address of 192.168.100.108 using ARP.
- Meanwhile, another device with IP 192.168.100.1 is pinging the Raspberry Pi to check its network connectivity, and the Raspberry Pi responds back.

The capture seems to indicate standard network communications without any signs of malicious activity. However, it's always a good idea to monitor your network for any unexpected or unauthorized traffic. If there's a specific aspect of the capture you'd like me to focus on or if you have other questions, please let me know!

- This way is useful if you analyze just a packet or compare small packets displayed in one screenshot.

To read longer trace file to GPT

- How about samplehttp.pcapng ?
  it includes 13 packets, 2132 bytes
- tshark –r samplehttp.pcapng –V –T text >>samplehttp.txt
  it contains 2604 lines, 119988 chars
- 3: So if we send this into GPT, GPT do not read this.



- We need to divide into 4096-16384 tokens, 1365-5461 chars (GPT3.5) or 8192-32768 tokens, 2730-10922 chars (GPT4)
- At first, we need to divide trace file by a packet.
  editcap -c 1 samplehttp.pcapng samplehttp

Divide and output pcapng into text

- Divide the trace file by a packet.
  editcap -c 1 samplehttp.pcapng samplehttp.pcapng
- Convert text from samplehttp_00000_20220804163133 to samplehttp_00012_20220804163134 using for loop
  for %a in (samplehttp_*) do tshark -r %a -V -T text>>%a.txt
- We can get 13 samplehttp_000XX_YYYYMMDDHHMMSS.txt
- We need to instruct ChatGPT to read this divided files add instructions sentence to read chunked files
- Modify each samplehttp_*.txt files to add symbol sentence at the first line and the end line

Edit each trace files and add Instructions

- Edit samplehttp_00000_20220804163133.pcapng.txt
- Add instructions at the first and the end line.

  “Do not answer yet. This is just another part of the text I want to send you. Just receive and acknowledge as "Part 1/13 received" and wait for the next part.
  [START PART 1/13]
  I want to act as a network engineer, analyze this trace.
  ….
  [END PART 1/13]
  Remember not answering yet. Just acknowledge you received this part with the message "Part 1/13 received" and wait for the next part.
- Edit the rest of the files like this.

Divide and output pcapng into text

- Add an instruction order sentence at first

  The total length of the content that I want to send you is too large to send in only one piece.

  For sending you that content, I will follow this rule:

  [START PART 1/10]

  this is the content of the part 1 out of 10 in total

  [END PART 1/10]

  Then you just answer: "Received part 1/10“

  And when I tell you "ALL PARTS SENT", then you can continue processing the data and answering my requests.

- Demonstration from Prompt4-0 to Prompt4-13.txt

Paste each chunk prompt to ChatGPT

ME

The total length of the content that I want to send you is too large to send in only one piece.

For sending you that content, I will follow this rule:

[START PART 1/10]
this is the content of the part 1 out of 10 in total
[END PART 1/10]

Then you just answer: "Received part 1/10"

And when I tell you "ALL PARTS SENT", then you can continue processing the data and answering my requests.

Understood! Please proceed, and I'll confirm the receipt of each part.

Paste the instruction, then paste each chunk.

part with the message "Part 1/13 received" and wait for the next part.

Part 1/13 received.

seconds]
[Time since previous frame in this TCP stream: 0.000000000 seconds]

[END PART 13/13]
ALL PARTS SENT. Now you can continue processing the request.

Part 13/13 received.

Thank you for providing all the packet capture details across the multiple parts. Based on the information provided, here are some key points and observations:

Analyze trace file with ChatGPT

1. **Devices and IP Addresses**:
   - There is communication between two devices.
   - Device with MAC address `00:1d:93:94:ea:bc` (referred to as "Modacom") has an IP address `13.73.26.73`.
   - Device with MAC address `b8:27:eb:4b:46:d5` (referred to as "Raspberry Pi") has an IP address `192.168.100.104`.
2. **Protocol**:
   - The encapsulation type is Ethernet.
   - The main protocol used for this communication is TCP over IPv4.
3. **Direction**:
   - In the first part (Part 12/13), the Modacom device is sending data to the Raspberry Pi.
   - In the second part (Part 13/13), the Raspberry Pi is sending data to the Modacom device.
4. **Ports and TCP Flags**:
   - Modacom is using port `80`, typically used for HTTP traffic, indicating it might be a web server.
   - Raspberry Pi is using a higher-numbered port `39182` which is likely a dynamically assigned port.
   - The TCP flag set in both parts is `ACK` which is an acknowledgment. This suggests a typical TCP handshake or an ongoing communication where devices acknowledge receipt of packets.
5. **TCP Options**:
   - Both packets have the Timestamps option enabled which is used for PAWS (Protection Against Wrapped Sequence numbers) and RTT calculation.
6. **Miscellaneous**:
   - The packet captures indicate there's no data payload in the TCP segments. They are just TCP ACKs.
   - Time stamps provided suggest that these captures are almost simultaneous or very close in time.
   - Both packets have a "Don't Fragment" flag set in their IPv4 headers.

Possible Scenario: The provided packet captures indicate a communication between a web server (Modacom) and a client (Raspberry Pi). The current captures only show ACK packets, likely part of an ongoing TCP communication or immediately after the three-way handshake.

Do you have any specific questions or is there anything else you would like to derive from these packet details?

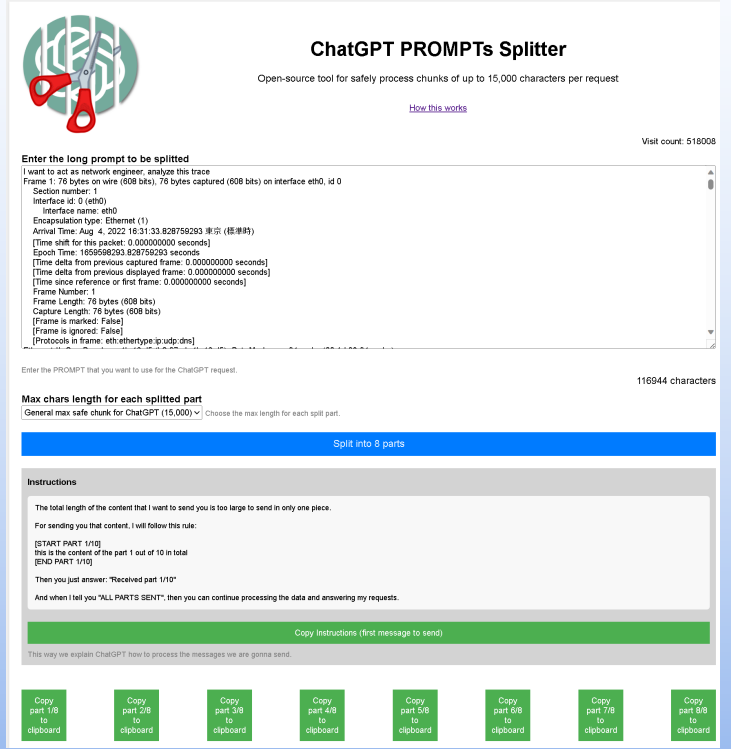
Long PROMPTs Splitter

https://github.com/jupediaz/chatgpt-prompt-splitter

SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov

#sf23eu

- The python code to split long prompt and divide into chunks with adequate instruction.
- You paste a long prompts then get each chunk with Instructions https://chatgpt-prompt-splitter.jjdiaz.dev/
- There is an another way to use API. But it is easy and simple to send big data into GPT.



Ask additional questions

- If you create a chat, drill down with additional questions again and again. For example, “summarize the traffic…”
- Sometimes not, but sometimes useful.

SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov

#sf23eu

ME

please summarize the address of the trace file

Based on the information provided in the 13 parts of the text you sent, the trace file captures network traffic between two devices:

1. Source Device:
   - IP Address: 13.73.26.73
   - MAC Address: 00:1d:93:94:ea:bc (Modacom_94:ea:bc)
2. Destination Device:
   - IP Address: 192.168.100.104
   - MAC Address: b8:27:eb:4b:46:d5 (Raspberr_4b:46:d5)

The network traffic captured is a TCP communication between these two devices. There are two frames captured in the trace file, each representing a packet in one direction of the communication:

1. Frame 1: A packet from the source device (13.73.26.73) to the destination device (192.168.100.104), with a TCP acknowledgment.
2. Frame 2: A packet from the destination device (192.168.100.104) back to the source device (13.73.26.73), also with a TCP acknowledgment.

Both packets are on the Ethernet interface eth0, with an encapsulation type of Ethernet. The packets do not carry any payload data as indicated by the TCP segment length of 0, and are primarily serving to acknowledge receipt of previous data in the TCP communication. The TCP headers include timestamp options for round-trip time measurement.

To write trace files from GPT

- GPT answers only as text format, ”text2pcap” is a nice tool for GPT to output pcap/pcapng
- For example, 5:“make icmp request and response between 1.1.1.1 11:11:11:11:11:11 and 2.2.2.2 22:22:22:22:22:22 in text2pcap format with asciidump“
  Note:ascii dump is needed for easy translation

ME

make icmp request and response between 1.1.1.1 11:11:11:11:11:11 and 2.2.2.2 22:22:22:22:22:22 in text2pcap format with asciidump

Sure! Below is an example of a text representation of an ICMP request and response packet in the format suitable for `text2pcap`, including ASCII dump. The source is `1.1.1.1` with MAC address `11:11:11:11:11:11`, and the destination is `2.2.2.2` with MAC address `22:22:22:22:22:22`.

Please note that the Ethernet type for IP is `0x0800`, and the protocol number for ICMP is `1`.

python — Copy code

```
# Packet 1: ICMP Echo Request (Ping)
0000  22 22 22 22 22 22 11 11 11 11 11 11 08 00 45 00   """""""""..E.
0010  00 54 00 01 00 00 40 01 5f 9d 01 01 01 01 02 02   .T....@._.......
0020  02 02 08 00 4d 62 00 01 00 01 61 62 63 64 65 66   ....Mb....abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76   ghijklmnopqrstuv
0040  77 61 62 63 64 65 66 67 68 69                     wabcdefghijkl

# Packet 2: ICMP Echo Response (Pong)
0000  11 11 11 11 11 11 22 22 22 22 22 22 08 00 45 00   ....""""""..E.
0010  00 54 00 01 00 00 40 01 5f 9d 02 02 02 02 01 01   .T....@._.......
0020  01 01 00 00 55 62 00 01 00 01 61 62 63 64 65 66   ....Ub....abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76   ghijklmnopqrstuv
0040  77 61 62 63 64 65 66 67 68 69                     wabcdefghijkl
```

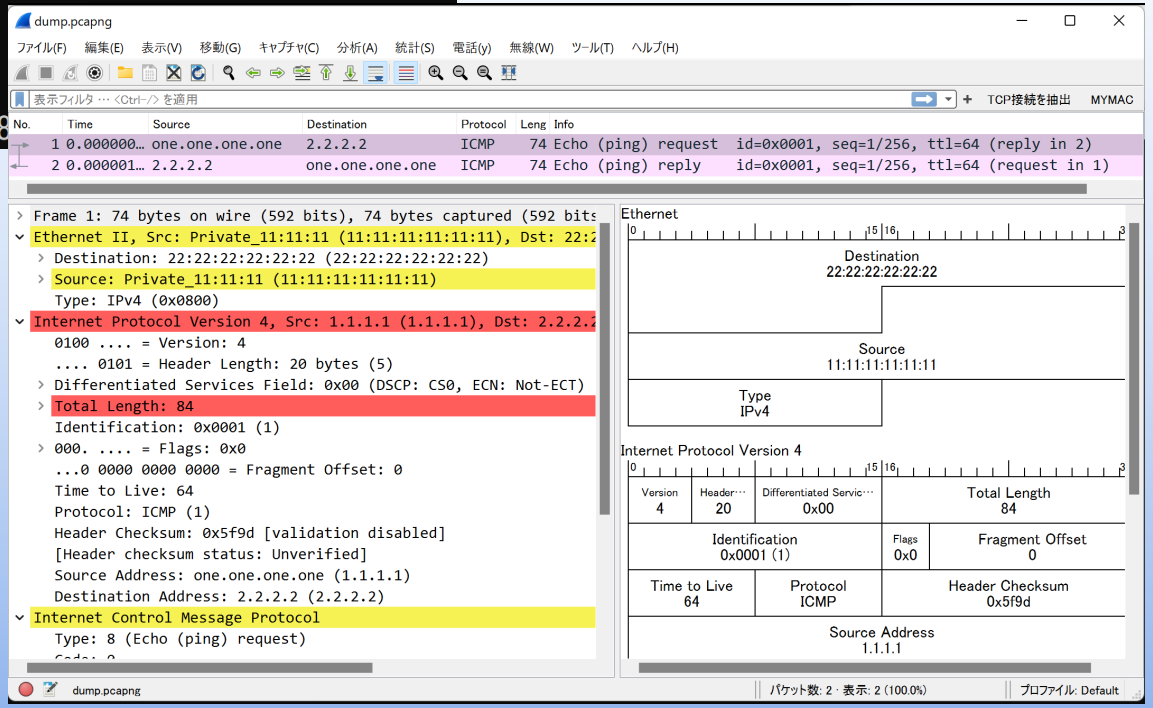
To write trace files from GPT

- Copy code and save as dump.txt
- text2pcap dump.txt dump.pcapng

```
C:¥Users¥megumitakeshita¥gpt>text2pcap dump.txt dump.pcapng
Input from: dump.txt
Output to: dump.pcapng
Output format: pcapng

--------------------------
Read 2 potential packets, wrote 2 packets (48
```

- Open dump.pcapng with Wireshark
- We can output pcapng via ChatGPT



SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov

#sf23eu

GPT best practice

https://platform.openai.com/docs/guides/gpt-best-practices/gpt-best-practices

#sf23eu

- OpenAI shows Strategies and tactics to make good prompts.

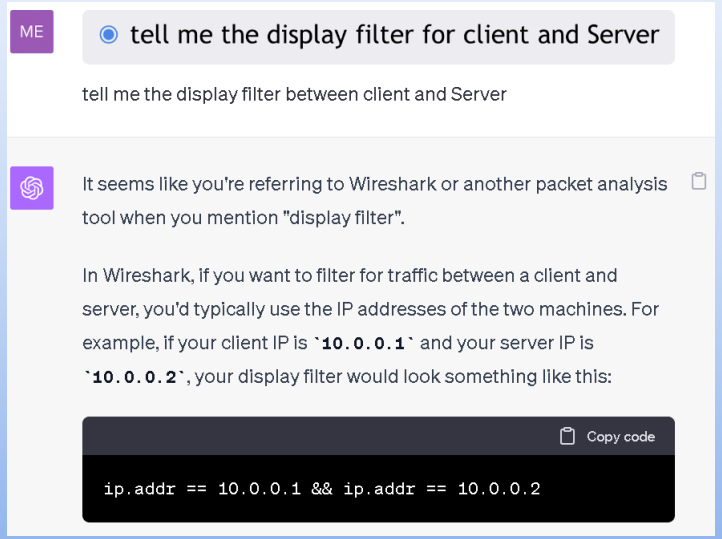
1. Write clear instructions
2. Provide reference text
3. Split complex tasks into simpler subtasks
4. Give GPTs time to “think”
5. Use external tools
6. Test changes systematically

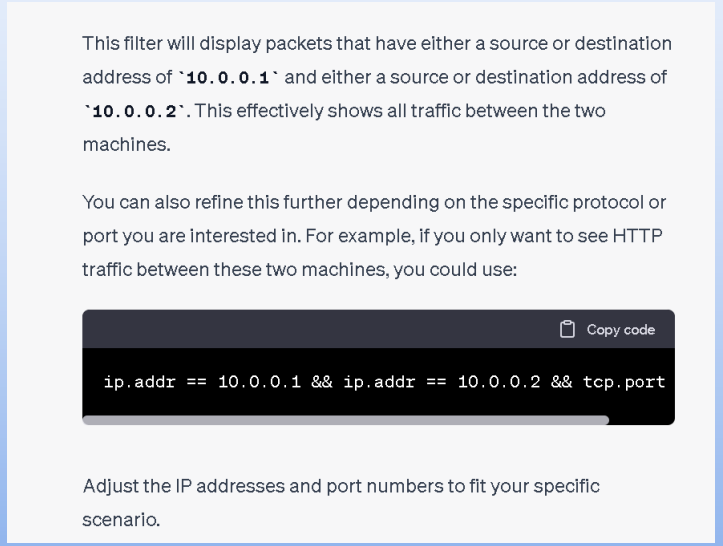
We are packet person, so Let’s create prompts for Wireshark all prompts and answers samples are below,
https://www.ikeriri.ne.jp/sharkfest/ChatGPTwithWireshark.zip

Write clear instructions

Include details in your query to get more relevant answers

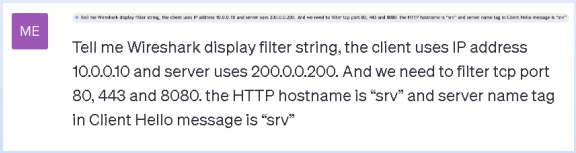
- Prompt6: Tell me display filter between the client and Server
- It works in vague, but sometimes we need port number, hostnames, and other information.

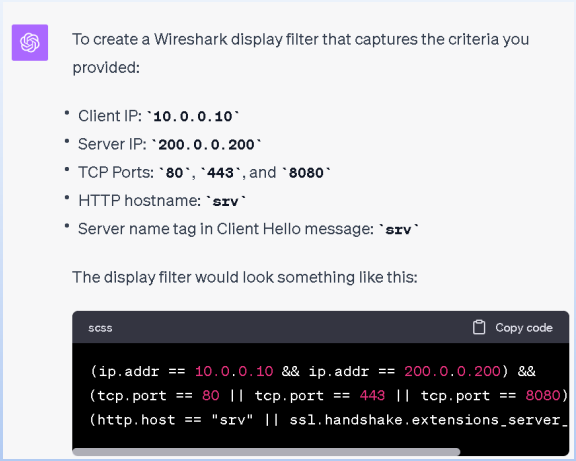


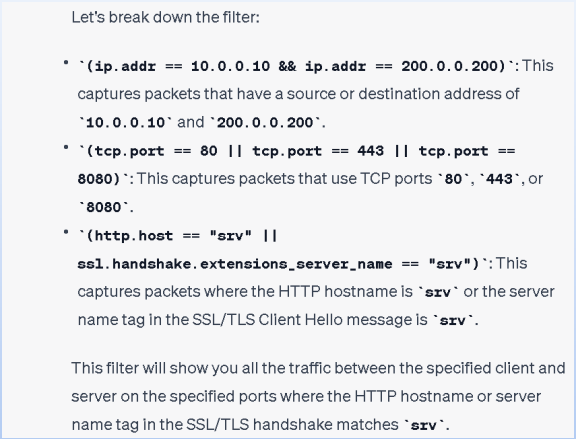


Include details in your query to get more relevant answers

- Prompt7: Tell me Wireshark display filter string, the client uses IP address 10.0.0.10 and server uses 200.0.0.200. And we need to filter tcp port 80, 443 and 8080. the HTTP hostname is “srv” and server name tag in Client Hello message is “srv”








```
(ip.addr == 10.0.0.10 && ip.addr == 200.0.0.200) &&
(tcp.port == 80 || tcp.port == 443 || tcp.port == 8080) &&
(http.host == "srv" || ssl.handshake.extensions_server_name == "srv")
```

Write clear instructions

Ask the model to adopt a persona

SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov

#sf23eu

- pwnedftp.pcapng: a part of the trace file, the attacker compromise 3COM FTP server

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.163.128	192.168.163.130	FTP	108	Response: 220 3Com 3CDaemon FTP Server Version 2.0
2	0.000001	192.168.163.130	192.168.163.128	FTP	1000	Request: USER uz^^}x45Vp0N_4uomQU[dX!{PSIMb<X*khAGi20!

- tshark -r pwnedftp.pcapng –O ip,tcp,ftp -T text>>pwned.txt
- Prompt8: Analyze trace below (and paste pwnedftp.txt)

ME

analyze trace below,
Frame 1: 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface unknown, id 0
Ethernet II, Src: VMware_94:82:d4 (00:0c:29:94:82:d4), Dst: VMware_74:9c:34 (00:0c:29:74:9c:34)
Internet Protocol Version 4, Src: 192.168.163.128, Dst: 192.168.163.130
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0×00 (DSCP: CS0, ECN: Not-ECT)
0000 00.. = Differentiated Services Codepoint: Default (0)
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 94
Identification: 0×2d27 (11559)
010. .... = Flags: 0×2, Don't fragment

The provided trace seems to capture two frames of a network communication. Let's break down the key information from each frame:

Frame 1:

1. **Ethernet II Layer**
   - Source MAC Address: VMware_94:82:d4 (00:0c:29:94:82:d4)
   - Destination MAC Address: VMware_74:9c:34 (00:0c:29:74:9c:34)
2. **IP (Internet Protocol) Layer**
   - Version: IPv4
   - Source IP Address: 192.168.163.128
   - Destination IP Address: 192.168.163.130
   - Total Length: 94 bytes
   - Don't Fragment flag is set.
3. **TCP (Transmission Control Protocol) Layer**
   - Source Port: 21 (which is typically used for FTP)
   - Destination Port: 38974 (seems to be a dynamic port)
   - Flags set: PSH and ACK
   - Sequence and Acknowledgment details provided.
   - TCP options are also provided, including two No-Operations and a Timestamp.
4. **FTP (File Transfer Protocol) Layer**
   - FTP Server Response: "220 3Com 3CDaemon FTP Server Version 2.0". This indicates that the FTP server is ready to accept a new user connection.

Ask the model to adopt a persona

- ChatGPT dissect the trace file and tell us the summary of fields and generic observations.

Observations:

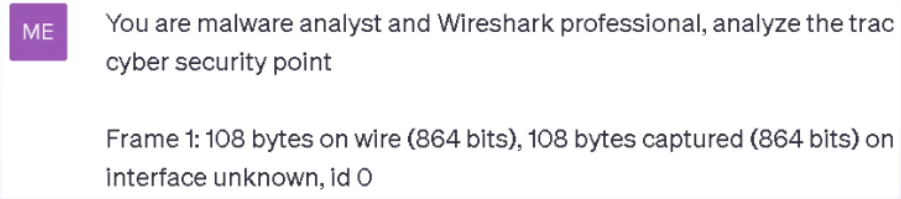
- The trace captures a communication between an FTP server and an FTP client.
- Frame 1 is from the server indicating readiness for a new user.
- Frame 2 is from the client trying to provide a username. However, the username seems unusual.
- Both the client and the server seem to be on the same local network (based on the IP addresses).
- The MAC addresses suggest that the devices or VMs are from VMware, indicating that this communication is likely occurring between two virtual machines.
- The unusual username in the FTP request in Frame 2 may be indicative of an attempt to exploit the FTP server, or it could just be non-standard behavior. Further analysis and context would be needed to draw conclusions about the nature of this activity.

It is good, but ChatGPT works as dissect fields in a generic way

We need to add a role to the GPT

Add a sentence in the first line, like
"I want you to act as a network engineer," or
"You are a malware analyst and Wireshark professional,"

Prompt9:You are malware analyst



ChatGPT act as the role you wanted

- ChatGPT dissect the trace file at first.

The provided packets show two frames of a network communication captured, seemingly from Wireshark or a similar network protocol analyzer. Let's break down the critical observations and their implications:

Frame 1:

1. **Ethernet II**:
   - Communication is between two devices with MAC addresses linked to VMware. This suggests it is a virtual environment or machines managed by VMware.
   - Src MAC: VMware_94:82:d4
   - Dst MAC: VMware_74:9c:34
2. **IPV4**:
   - Communication is between two local IPs within the same subnet.
   - Src IP: 192.168.163.128
   - Dst IP: 192.168.163.130
3. **TCP**:
   - Source port is 21, which is typically the default port for FTP (File Transfer Protocol).
   - Destination port is 38974, which seems to be a dynamically allocated port.
4. **FTP**:
   - An FTP banner reveals it's the "3Com 3CDaemon FTP Server Version 2.0". The FTP banner can be used for banner grabbing, allowing attackers to identify potential vulnerabilities associated with this version.

Frame 2:

1. **Ethernet II**:
   - It's a reply from the Dst MAC in frame 1 to the Src MAC.
   - Src MAC: VMware_74:9c:34
   - Dst MAC: VMware_94:82:d4
2. **IPV4**:
   - Reply from 192.168.163.130 to 192.168.163.128.
3. **TCP**:
   - Communication back from port 38974 to port 21, confirming the earlier assumption tha this is an FTP transaction.
4. **FTP**:
   - The command being sent is "USER", which is used to specify the username during the FTP authentication process.
   - The argument (username) in this request seems garbled or encrypted. The pattern doesn't look like typical random data or encryption; it might be some encoded or obfuscated data. In a real-world scenario, unusual or unexpected data patterns could indicate an attack or malware activity.

Ask the model to adopt a persona

- Then ChatGPT tell us the vulnerability of 3Com 3CDaemon and wrong settings of FTP server.

Cybersecurity Implications:

1. **3Com 3CDaemon FTP Server**:
   - Using an older FTP server might have vulnerabilities. Research would need to be conducted to determine if there are any known vulnerabilities associated with "3Com 3CDaemon FTP Server Version 2.0".
2. **Garbled FTP Username**:
   - The garbled username in the second frame is suspicious. It could be an attempt to exploit the FTP server using a buffer overflow, directory traversal, or another kind of attack.
3. **FTP in general**:
   - FTP is an insecure protocol as it transmits data, including credentials, in plaintext. A Man-in-the-Middle attacker can easily capture this data. Secure alternatives, like SFTP or FTPS, should be considered.
4. **Virtual Environment**:
   - Communication between virtual machines (indicated by VMware MACs) could imply testing, sandboxing, or isolated environments. However, if these are production machines, one should ensure that security measures are in place for virtual environments as well.

Next Steps:

1. Investigate the garbled username. If it's a known pattern associated with an exploit, action should be taken immediately.
2. Consider replacing or upgrading the FTP server to a version without known vulnerabilities.
3. If FTP must be used, consider securing it with VPNs or switch to secure alternatives like SFTP/FTPS.
4. Monitor the virtual environment for other suspicious activities.
5. Regularly patch and update all systems.

Finally, while these packets provide a snapshot of the network activity, a thorough investigation would require more context and potentially additional network captures to fully understand and assess the situation.

moreover, GPT tell us how to fix this problem

Write clear instructions

Use delimiters to clearly indicate distinct parts of the input

- garp.pcapng contains Gratuitous ARP, and ARP probe frames
- tshark -r garp.pcapng –V -T text>>garp. txt
- Create coloring rules as Gratuitous ARP packets are red

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	IntelCor_a0:a0:91	Broadcast	ARP	60	Who has 192.168.100.117? (ARP Probe)
2	0.990640504	IntelCor_a0:a0:91	Broadcast	ARP	60	ARP Announcement for 192.168.100.117
3	5.492377145	Dell_5d:3c:d4	Broadcast	ARP	60	Who has 169.254.14.251? (ARP Probe)
4	6.498576497	Dell_5d:3c:d4	Broadcast	ARP	60	ARP Announcement for 169.254.14.251

ME

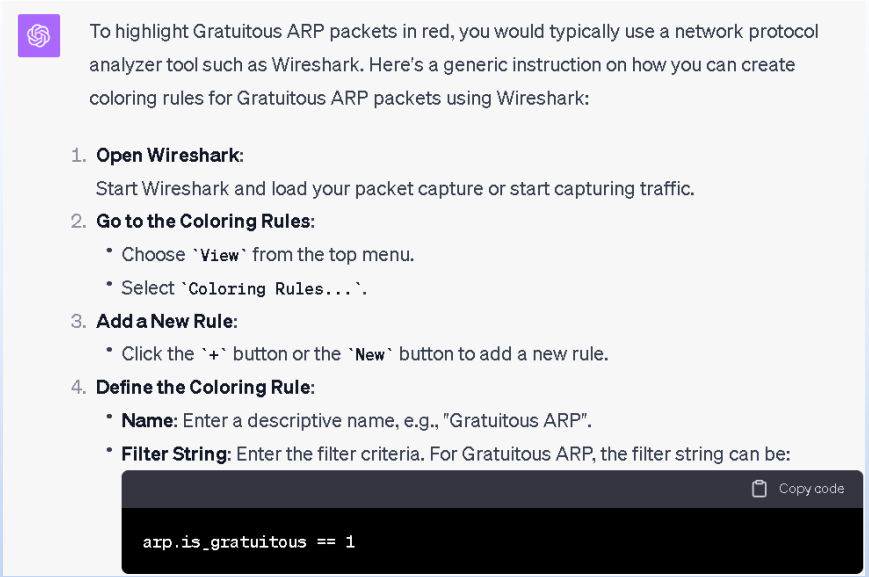
Create coloring rules as Gratuitous ARP packets are red the trace below

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0

Section number: 1

- Prompt10: Create coloring rules as Gratuitous ARP packets are red the trace below

Use delimiters to clearly indicate distinct parts of the input



To highlight Gratuitous ARP packets in red, you would typically use a network protocol analyzer tool such as Wireshark. Here's a generic instruction on how you can create coloring rules for Gratuitous ARP packets using Wireshark:

1. **Open Wireshark**:
   Start Wireshark and load your packet capture or start capturing traffic.
2. **Go to the Coloring Rules**:
   - Choose `View` from the top menu.
   - Select `Coloring Rules...`.
3. **Add a New Rule**:
   - Click the `+` button or the `New` button to add a new rule.
4. **Define the Coloring Rule**:
   - **Name**: Enter a descriptive name, e.g., "Gratuitous ARP".
   - **Filter String**: Enter the filter criteria. For Gratuitous ARP, the filter string can be:

Copy code

```
arp.is_gratuitous == 1
```

   - **Background**: Choose the color red (or any preferred color).
   - **Foreground**: Choose a color that will be readable against your background color (e.g., white or black).
5. **Save and Apply**:
   - Click `OK` to save the new rule.
   - Make sure your new rule is checked in the list of coloring rules.
   - Click `OK` again to apply the coloring rule to your packet list.
6. **Viewing the Packets**:
   Return to the main Wireshark window. Any Gratuitous ARP packets should now be highlighted in red.

Note: The filter provided is generic for Gratuitous ARP packets. Depending on the specifics of the network traffic you're analyzing or the version of Wireshark you're using, you might need to adjust the filter or the procedure slightly.

In the provided trace, Frame 2 and Frame 4 are Gratuitous ARP packets as indicated by the lines `[Is gratuitous: True]`. If you follow the above instructions, these frames will be colored red in Wireshark.

- It seems correct using GUI, but we need to tell GPT some more instructions for setting the file of “colorfilters” like @TCP@tcp@[59345,58980,65535][4718,10030,11796]

Prompt11: you are setting up Wireshark using "coloringrules" configuration file, the file format is

Tell GPT how to use delimiters ”@” and format

you are setting up Wireshark using "coloringrules" configuration file,

the file format is
@Name@DisplayFilter@[colorcode of Red, Green, Blue of character][colorcode of Red, Green, Blue of background color]

for example TCP coloring rule is
@TCP@tcp@[59345,58980,65535][4718,10030,11796]

so create coloring rule of Gratuitous ARP packet with Red character and White background

Use delimiters to clearly indicate distinct parts of the input

To create a coloring rule for Gratuitous ARP packets with Red character on a White background, we need to determine:

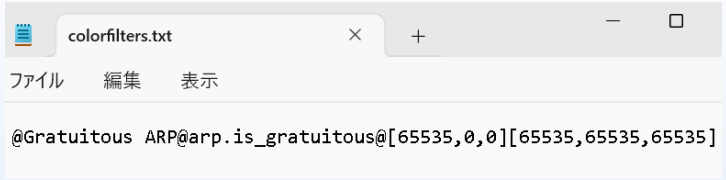
1. Display Filter: The filter that defines the traffic we're interested in. For Gratuitous ARP, this filter is typically based on the ARP opcode and the fact that the source and destination IP are the same. The display filter for a Gratuitous ARP packet in Wireshark is:
   `arp.is_gratuitous`
2. Color Codes:
   - Red character: [65535, 0, 0]
   - White background: [65535, 65535, 65535]

Using the given file format, the coloring rule for Gratuitous ARP would look like:

less — Copy code

```
@Gratuitous ARP@arp.is_gratuitous@[65535,0,0][65535,65535,65535]
```

You can add this rule to your "coloringrules" configuration file to color Gratuitous ARP packets with a Red character on a White background in Wireshark.



GPT4 creates a perfect coloring rules configuration

Write clear instructions

Specify the steps required to complete a task

- pingscan.pcapng contains some ICMP request and response
- You want to pick up the hostname that answers the request.
- tshark -r pingscan.pcapng –V -T text>>pingscan.txt

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.198	202.214.194.138	ICMP	42	Echo (ping) request id=0x95cf, seq=0/0, ttl=44 (no response found!)
2	0.000023532	192.168.100.198	180.235.36.115	ICMP	42	Echo (ping) request id=0x4bf4, seq=0/0, ttl=53 (reply in 3)
3	0.108007692	180.235.36.115	192.168.100.198	ICMP	60	Echo (ping) reply id=0x4bf4, seq=0/0, ttl=238 (request in 2)
4	0.541589729	192.168.100.198	202.214.194.138	ICMP	42	Echo (ping) request id=0xd47f, seq=0/0, ttl=55 (no response found!)

- Just ask Prompt12:“pick up the hostname and GeoIP that answer ICMP request”

ME

Prompt12

pick up the hostname and GeoIP that answer ICMP request

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0

Specify the steps required to complete a task

Based on the provided frames, here's the information regarding the hosts that answer ICMP requests:

Frame 1 (ICMP Echo Request):

- **Source Host:**
  - IP Address: 192.168.100.198
- **Destination Host:**
  - IP Address: 202.214.194.138
  - GeoIP Information:
    - Country: Japan
    - ASN: 2497
    - Organization: Internet Initiative Japan
    - Latitude: 35.69
    - Longitude: 139.69

Frame 2 (ICMP Echo Request):

- **Source Host:**
  - IP Address: 192.168.100.198
- **Destination Host:**
  - IP Address: 180.235.36.115
  - GeoIP Information:
    - City: Kobe
    - Country: Japan
    - ASN: 4685
    - Organization: Asahi Net
    - Latitude: 34.6987
    - Longitude: 135.2008

Frame 3 (ICMP Echo Reply):

- **Source Host:**
  - IP Address: 180.235.36.115
  - GeoIP Information:
    - City: Kobe
    - Country: Japan
    - ASN: 4685
    - Organization: Asahi Net
    - Latitude: 34.6987
    - Longitude: 135.2008
- **Destination Host:**
  - IP Address: 192.168.100.198

From the frames:

- The host with IP `192.168.100.198` is sending ICMP requests.
- The host with IP `202.214.194.138` associated with "Internet Initiative Japan Inc." was targeted by the ICMP request but hasn't replied (based on provided frames).
- The host with IP `180.235.36.115` associated with "Asahi Net" in Kobe, Japan, was targeted by the ICMP request and has replied, confirming it answers ICMP request

- The answer is correct, but how about making it better?

Specify the steps required to complete a task

SharkFest'23 EUROPE

Prompt13: Create the Python code that plots the IP and the region in the map using the folium package" after executing the prompt

ME

create the python code that plot the IP and the region in the map using folium package

To plot IP addresses with their regions on a map using the `folium` package, follow these steps:

1. Extract the IP addresses and their respective latitude and longitude from the data you provided (or similar data).
2. Use the `folium` package to plot the IP addresses on the map.

Here's a simple Python code to demonstrate how to achieve this:

python — Copy code

```
import folium

# Extracted data from the pcap
data = [
    {'ip': '202.214.194.138', 'lat': 35.69, 'lon': 139.69, 'region': 'Tokyo'},
    {'ip': '180.235.36.115', 'lat': 34.6987, 'lon': 135.2008, 'region': 'Kobe'
]

# Create a base map
m = folium.Map(location=[35.6895, 139.6917], zoom_start=6)  # centered around
```

```
# Add IP addresses as markers on the map
for entry in data:
    folium.Marker(
        location=[entry['lat'], entry['lon']],
        popup=f"IP: {entry['ip']}<br>Region: {entry['region']}",
        icon=folium.Icon(icon="cloud"),
    ).add_to(m)

# Save the map to an HTML file
m.save("map.html")
```

When you run the above code, it will generate an HTML file named "map.html". You can open this file in a web browser to see the IP addresses plotted on the map.

Please note:

- This example is based on the data you provided. You may need to adjust the data extraction process if your input format changes.
- You'll need to have the `folium` package installed. You can install it using pip:

Copy code

```
pip install folium
```

- For a large number of markers, consider using plugins like `FastMarkerCluster` from folium to efficiently cluster and visualize the markers.

Regen

Specify the steps required to complete a task

map.py

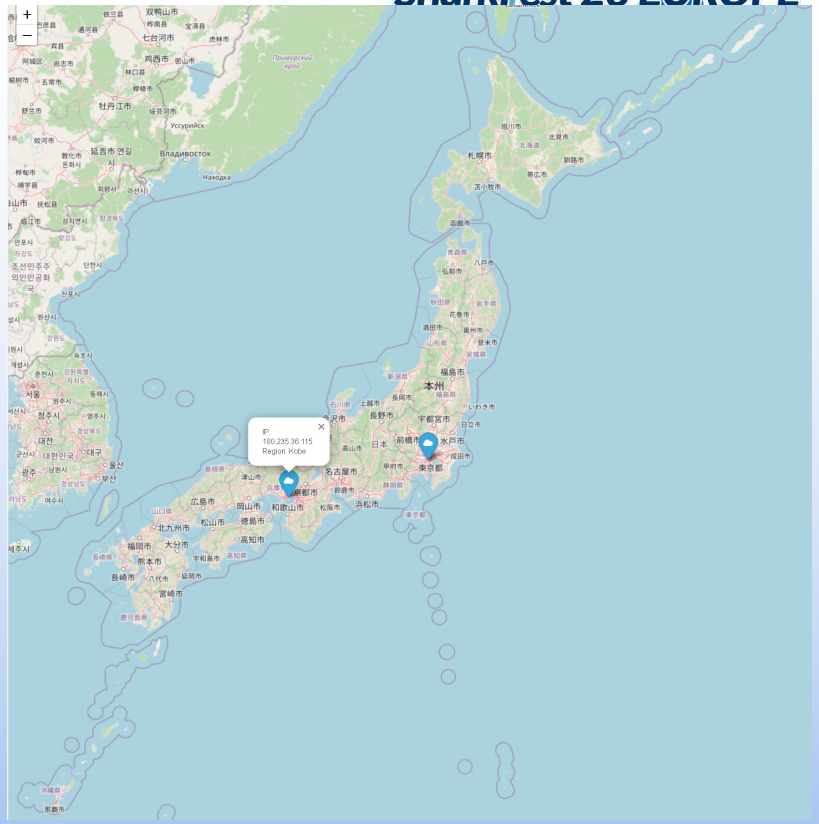
```python
import folium

# Extracted data from the pcap
data = [
    {'ip': '202.214.194.138', 'lat': 35.69, 'lon': 139.69, 'region': 'Tokyo'},
    {'ip': '180.235.36.115', 'lat': 34.6987, 'lon': 135.2008, 'region': 'Kobe'}
]

# Create a base map
m = folium.Map(location=[35.6895, 139.6917], zoom_start=6)  # centered around Tokyo for this example

# Add IP addresses as markers on the map
for entry in data:
    folium.Marker(
        location=[entry['lat'], entry['lon']],
        popup=f"IP: {entry['ip']}<br>Region: {entry['region']}",
        icon=folium.Icon(icon="cloud"),
    ).add_to(m)

# Save the map to an HTML file
m.save("map.html")
```



- pip install folium
  (python3 -m pip install folium)
- Paste the code into VS code
- Create map.html and open the browser.

Provide reference text

Instruct the model to answer using a reference text

- If you want to set the specified capture filter
- Just ask 14:"Tell me Wireshark capture filter that matches the association and disassociation of the specified host."

ME

tell me Wireshark capture filter that matches association and disassociation of the specified host

If you want to capture association and disassociation frames for a specific host in Wireshark, you'll use 802.11 Management frames, particularly the Association Request, Association Response, Disassociation, and sometimes even Reassociation Request and Reassociation Response frames.

Assuming the host in question has a MAC address of `XX:XX:XX:XX:XX:XX`, you would use the following capture filter:

less — Copy code

```
(wlan.fc.type_subtype == 0x00 || wlan.fc.type_subtype == 0x01 || wlan.fc
```

Here's a breakdown of the filter:

- `wlan.fc.type_subtype == 0x00`: This matches the Association Request frames.
- `wlan.fc.type_subtype == 0x01`: This matches the Association Response frames.
- `wlan.fc.type_subtype == 0x0a`: This matches the Disassociation frames.
- `wlan.addr == XX:XX:XX:XX:XX:XX`: This matches frames where any of the MAC addresses (destination, source, or BSSID) matches the specified MAC address.

Remember to replace `XX:XX:XX:XX:XX:XX` with the actual MAC address of the host you're interested in. If you also want to include Reassociation Request and Response frames, add the type_subtypes `0x02` and `0x03` respectively to your filter.

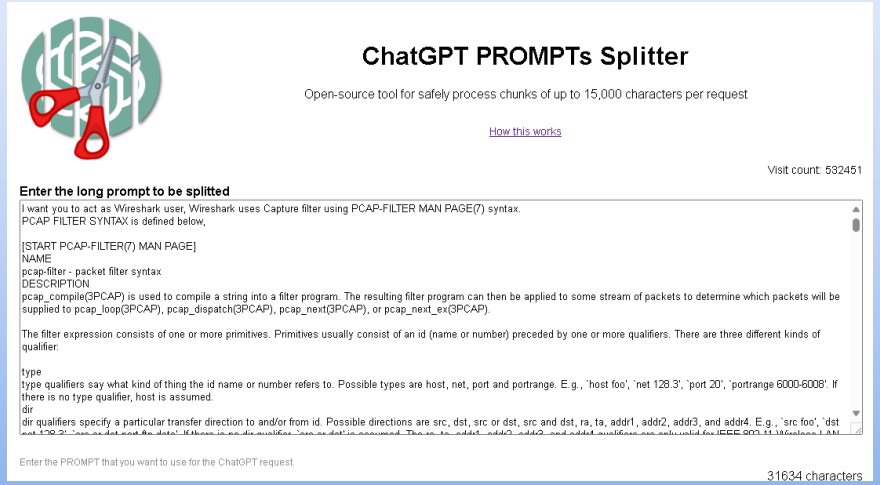
Note: This is a Wireshark capture filter, which means it should be applied when starting a capture session. If you're looking to filter already-captured data, you would use a display filter instead.

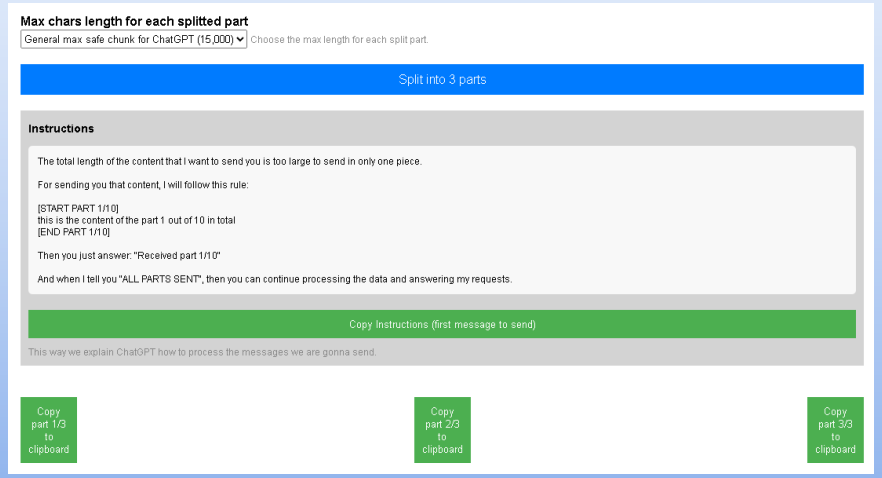
Provide reference text

- It happened Hallucination, GPT answered Display Filter
- In this case, instruct the model to answer using a reference
- There are rules of capture filter at tcpdump(libpcap) man page: https://www.tcpdump.org/manpages/pcap-filter.7.html
- We cannot refer external websites to GPT directly, copy the entire PCAP-FILTER(7) MAN PAGE, and use Long PROMPTs Splitter to create the prompt https://chatgpt-prompt-splitter.jjdiaz.dev/
- Prompt15:"I want you to act as Wireshark user. Wireshark uses a Capture filter using PCAP-FILTER MAN PAGE(7) syntax, PCAP FILTER SYNTAX is defined below, …"

Provide reference (Prompt15-0/1/2/3.txt)

I want you to act as Wireshark user. Wireshark uses a Capture filter using PCAP-FILTER MAN PAGE(7) syntax.
PCAP FILTER SYNTAX is defined below,
[START PCAP-FILTER(7) MAN PAGE]…[END PCAP-FILTER(7) MAN PAGE]
Using this PCAP FILTER SYNTAX, create a Wireshark Capture Filter that matches association and disassociation of the specified host





GPT answered correctly

SharkFest'23 EUROPE

To create a Wireshark capture filter that matches association and disassociation of a specified host using the PCAP filter syntax, we'll focus on the 802.11 Wi-Fi layer, since association and disassociation are concepts from this layer.

The 802.11 standard defines specific management frames for association and disassociation:

1. Association Request Frame
2. Association Response Frame
3. Disassociation Frame

Each of these frames has a specific subtype:

- Association Request: subtype 0
- Association Response: subtype 1
- Disassociation: subtype 10 (in binary) or 2 (in decimal)

To filter for these specific subtypes, you can focus on the Frame Control field of the 802.11 header. This field is 2 bytes in length. The subtype is defined in bits 4-7 of the first byte.

Here's a simplified representation:

[Protocol Version (2 bits)][Type (2 bits)][Subtype (4 bits)][Flags (8 bits)]

To match the specific subtypes, you'll have to consider the bit pattern:

- Association Request: 0000
- Association Response: 0001
- Disassociation: 1010

Here's a Wireshark capture filter for these specific subtypes:

```
css
(wlan[0] & 0xf0 == 0x00) or (wlan[0] & 0xf0 == 0x10) or (wlan[0] & 0xf0
```

Copy code

But you mentioned that you want to filter for a "specified host". This generally means filtering for a specific MAC address. If you know the MAC address of the host (let's assume it's `XX:XX:XX:XX:XX:XX`), then you can include this in your filter:

```
less
== 0x10) or (wlan[0] & 0xf0 == 0xa0)) and (wlan addr XX:XX:XX:XX:XX:XX)
```

Copy code

Replace `XX:XX:XX:XX:XX:XX` with the actual MAC address of the specified host.

This capture filter should match association and disassociation frames involving the specified host.

((wlan[0] & 0xf0 == 0x00) or (wlan[0] & 0xf0 == 0x10) or (wlan[0] & 0xf0 == 0xa0)) and (wlan addr XX:XX:XX:XX:XX:XX) Yes, GPT understood.

Split complex tasks into simpler subtasks

- GPT try to answer the prompt immediately, so sometimes, GPT leads to premature answers.
- So split the prompt into multiple simpler subtasks.
- Want to create a Twitter(X) Display filter, divide two tasks Step1: pick up addresses, Step2: create display filter
- 16:The task is two steps: process STEP1, then try STEP2

  Step1:There are Twitter IP address ranges described as network address / the length of subnet mask (for example 10.0.0.0/8), Pickup the Twitter IP address ranges from the document
  https://developer.twitter.com/en/blog/product-news/2022/updated-source-ip-address-range

  Step2: add “ip.addr==“ before and add “ or ” after the network addresses.

Split complex tasks into simpler subtasksovide examples

The task is two steps: process STEP1, then try STEP2
Step1 There are Twitter IP address ranges described as network address / the length of subnet mask (for example 10.0.0.0/8). Pickup the Twitter IP address ranges from the document
[START DOCUMENT]
Product News
Updated source IP address range for Account Activity API
…
202.160.130.0/24
[END DOCUMENT]

Step2 Add “ip.addr==“ before each Twitter IP address range,
Add “ or “ after each Twitter IP address range, and connect them into a sentence

Split complex tasks into simpler subtasks

Step 1: Extract Twitter IP address ranges from the document.

From the document, the Twitter IP address ranges are:

1. 199.16.156.0/22
2. 199.59.148.0/22
3. 192.133.77.0/26
4. 64.63.15.0/24
5. 64.63.31.0/24
6. 64.63.47.0/24
7. 202.160.128.0/24
8. 202.160.129.0/24
9. 202.160.130.0/24

Step 2: Add "ip.addr==" before each Twitter IP address range and " or " after each Twitter IP address range, then connect them into a sentence.

The result is:

ip.addr==199.16.156.0/22 or ip.addr==199.59.148.0/22 or ip.addr==192.133.77.0/26 or ip.addr==64.63.15.0/24 or ip.addr==64.63.31.0/24 or ip.addr==64.63.47.0/24 or ip.addr==202.160.128.0/24 or ip.addr==202.160.129.0/24 or ip.addr==202.160.130.0/24

Here's your complete sentence:

```
ip.addr==199.16.156.0/22 or ip.addr==199.59.148.0/22 or
ip.addr==192.133.77.0/26 or ip.addr==64.63.15.0/24 or
ip.addr==64.63.31.0/24 or ip.addr==64.63.47.0/24 or
ip.addr==202.160.128.0/24 or ip.addr==202.160.129.0/24 or
ip.addr==202.160.130.0/24
```

ip.addr==199.16.156.0/22 or ip.addr==199.59.148.0/22 or ip.addr==192.133.77.0/26 or ip.addr==64.63.15.0/24 or ip.addr==64.63.31.0/24 or ip.addr==64.63.47.0/24 or ip.addr==202.160.128.0/24 or ip.addr==202.160.129.0/24 or ip.addr==202.160.130.0/24

We can put it into the Wireshark display filter and check Twitter traffic.

Give GPT time to "think“

Give two trace files to compare GPT itself

- It is a good method to give GPT time to “think”
  For example, give two trace files to compare GPT itself instead of just throwing a trace file.
- There is a basic auth problem trace file: authfail.pcap
  tshark –r authfail.pcap –V –T text>>basicauthfail.txt
- Just enter Prompt17: “Why do you think this trace failed? <basicauthfail.txt>
- Use Long PROMPTs Splitter to divide the prompt
  https://chatgpt-prompt-splitter.jjdiaz.dev/
  (Prompt17-0,1,2,3,4,5.txt)

Give two trace files to compare GPT itself

The data provided is a continuation of a network packet capture (specifically, HTTP communication) and some HTML content.

Here's a summary:

1. **HTTP Data**
   - The request was made to `http://asashina.ikeriri.ne.jp/basicauth/`.
   - The response is chunked (as evident by the "HTTP chunked response"). The content is sent in chunks, and the end of the chunked encoding is signified by a chunk size of "0 octets."
   - The response contains 4483 bytes of file data.
2. **HTML Content**
   - The document type is `HTML 4.0 Transitional`.
   - There are metadata tags which provide various details about the content, including:
     - Charset: Shift_JIS (a Japanese character encoding)
     - Author: ikeriri network service
     - Manager: Megumi Takeshita
     - Keywords and Description are in Japanese and seem to mention the ikeriri network service and other related terms.
   - The body of the HTML is mainly table-based, with images from `www.ikeriri.ne.jp` and several anchor (link) elements pointing to different pages on the same domain. The content seems to be in Japanese, using a mixture of Japanese characters and HTML entities for special characters.
   - There's a copyright notice for "Ikeriri Network Service Limited" dated 1996-2004.
3. **Network Packet Details**
   - The encapsulation type for the data is Ethernet.
   - The packet was captured on `Aug 23, 2007`.
   - The packet contains `IPv4` protocol data with the source IP address `10.0.0.185` and the destination IP `210.138.35.74`.
   - Destination GeoIP data reveals that the destination is located in `Takamatsu, Japan`, belonging to the AS `Internet Initiative Japan Inc.`.

Overall, the provided information seems to detail an HTTP request to a Japanese website (`ikeriri.ne.jp`), and the subsequent response containing the website's content. The network data offers technical details about the communication, while the HTML shows the structure and content of the web page.

- GPT rushed to dissect each frame and pick up essential fields, but GPT did not answer the reason of the failure.
- GPT could not find the basic auth problem

Give two trace files to compare GPT itself

- In actual (Human) debugging / troubleshooting,
  Comparing two trace files finds us the clue we do not notice.
  Same as GPT, we need to pick up success trace and fail one,
  or trace files captured at the client side and the server side,
  or the packet of inside LAN and outside WAN,
  or the packet with settings on and off, and so on.
- To avoid GPT rushing the clue immediately,
  we add “Do not answer now” or the other phrase
- And we set emphasis on adding “compare each field by yourself” or some word to think GPT itself instead of just collecting learned information.

Prompt18: compare 2 traces, try to think GPT itself

- A Basic auth success trace file: authsuccess.pcap
  tshark –r authsuccess.pcap –V –T text>>authsuccess.txt

You are a network troubleshooter.
First, learn from the success of the basic authentication trace below:
<basicauthsccess.txt>
Do not answer now, and there is an authentication trouble
And the fail of Basic authentication trace is below:
<basicauthfail.txt>
Check both the success trace and fail trace and compare each field by yourself. What is the reason for the Authorization Required error?

- Use Long PROMPTs Splitter to divide the prompt
  https://chatgpt-prompt-splitter.jjdiaz.dev/
  Prompt18-0,1,2,3,4,5,6.txt

GPT compare 2 traces and think itself

Let's break down the information and perform a comparison of the success and fail traces based on the details provided:

1. **Ethernet Headers**:
   - Both the success and fail traces involve communication between a source with MAC address `Watsushi_94:9f:1e` and a destination with MAC address `JuniperN_41:30:d0`.
   - There's no issue detected at this layer.
2. **IPv4 Headers**:
   - Source IP: `10.0.0.185`
   - Destination IP: In the fail trace, it's `210.138.35.74`, which belongs to the "Internet Initiative Japan Inc." based in Takamatsu, Japan. This information is not available in the success trace. However, you mentioned that the destination IP was the same in both traces.
   - TTL (Time to Live), Fragmentation, and other fields are not explicitly stated in the success trace to compare with. There's no evidence of an issue at this layer either.
3. **TCP Headers**:
   - In the fail trace, the source port is `4905` and the destination port is `80`. In the success trace, you provided that the destination port is `80` (HTTP), which suggests that it's a web request.
   - There are flags set in the TCP headers: ACK in the fail trace. Without the success trace's exact flags, it's challenging to determine any differences.
   - The sequence and acknowledgment numbers are given, but without the success trace's counterpart, we can't compare.
4. **HTTP Request & Response**:
   - The fail trace contains an HTTP 401 error response, which means "Authorization Required." This typically implies that the request lacked valid authentication credentials.
   - In the success trace, the authorization credentials are provided (`Authorization: Basic`). This suggests that the client provided the correct credentials in the success trace but not in the fail trace.
5. **URL & Payload**:
   - The fail trace references a Japanese website (`www.ikeriri.ne.jp`), but the success trace's exact URL is not mentioned.
   - The fail trace has the user navigating to links related to learning, support, contact, terms, and privacy, which is not evident in the success trace. There might be different requests or interactions happening.

Conclusion:
The "Authorization Required" error in the fail trace is likely due to the absence or incorrectness of authentication credentials in the HTTP request. In the success trace, valid authorization credentials were provided, leading to a successful response.

To fully understand and validate the reason for the failure, we'd ideally need to:

- Compare the HTTP headers (especially the Authorization headers) in both traces.
- Check if the endpoints (URLs) being accessed require authentication.
- Confirm that the credentials used in the success trace are valid for the requested resource.

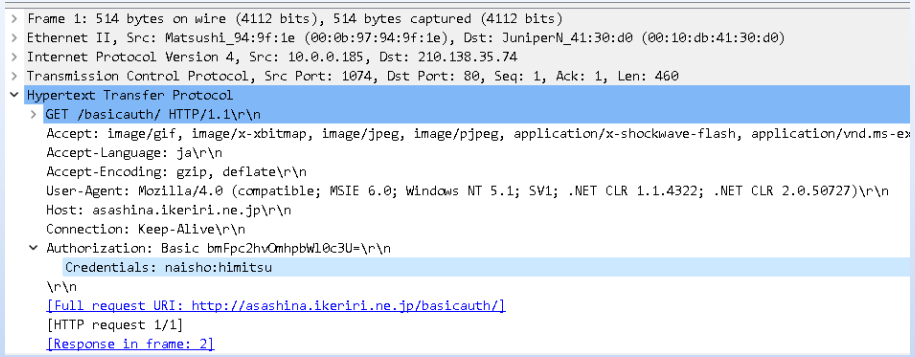
Remember, just providing authentication credentials doesn't ensure access. The credentials must be valid and have the necessary permissions for the specific resource.

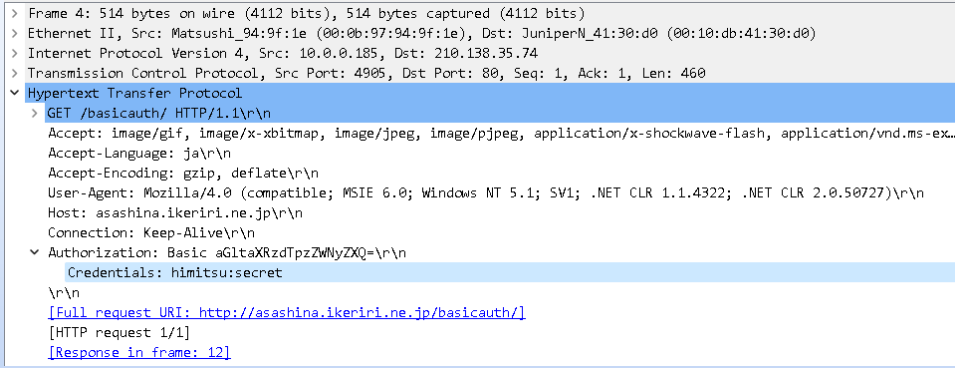
The "Authorization Required" error in the fail trace is likely due to the absence or incorrectness of authentication credentials in the HTTP request.

Give two trace files to compare GPT itself

- That is almost correct and we need to iterate the prompt to find GPT itself, the difference of credentials.

authsuccess.pcap vs authfail.pcap





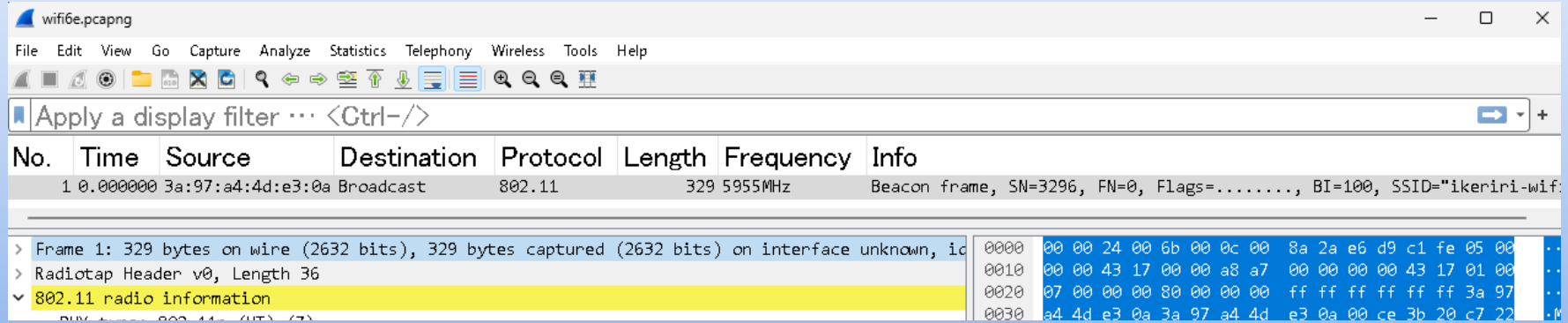
Credentials: naisho:himitsu Credentials: naisho:secret

- The clue why basic authorization failed is the wrong credentials of password (OK: himitsu NG:secret)
- This time GPT did not find this, may be next time…

Use external tool

Use code execution to perform more accurate calculations

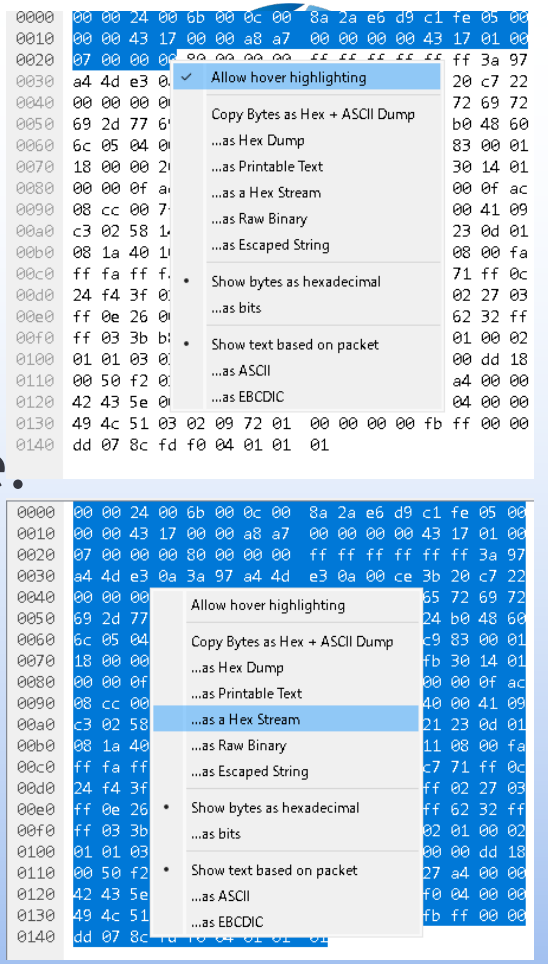
- ChatGPT can use external tools like executing Python code.
  You can write and execute Python code by enclosing it in triple backticks
  ```python print("hello")```
- We can use packet-related packages such as "scapy" allowing packet parsing using Wireshark dissectors
- Here is a wifi6e.pcapng, open this trace with Wireshark


  ```

Use external tool

Use code execution to perform more accurate calculations

- Right-click on the Packet bytes pane, check off “Allow hover highlighting”
- Select “Frame 1” in the Packet detail pane. All data is selected in the Packet bytes pane. Right-click on the Packet bytes pane, Select “…as a Hex Stream” to copy all bytes stream into a clipboard.
- Paste this hex stream into a text file named wifi6e.txt



Prompt19: Use external tool

You can write and execute Python code by enclosing it in triple backticks

```python
from scapy.all import *
from scapy.utils import *
hexdump = hex_bytes('<<paste wifi6e.txt>>')
packet = Dot11(hexdump)
packet.summary()
packet.show()
```

- "scapy" dissects Frame1(WiFi6E Beacon frame) based on Wireshark dissector.
- It is not GUI, but we utilize the Wireshark dissector on GPT.

GPT parse trace file using Wireshark dissector

You provided a hexdump that you'd like to convert into a scapy `Dot11` packet and then display its summary and detailed breakdown.

Here's the Python code using the provided hexdump:

python — Copy code

```
from scapy.all import *
from scapy.utils import *

# Convert hex string to bytes
hex_bytes_string = '000024006b000c008a2ae6d9c1fe0500000043170000a8a70000
hexdump_bytes = bytes.fromhex(hex_bytes_string)

# Convert bytes into a scapy Dot11 packet
packet = Dot11(hexdump_bytes)

# Display packet summary
summary = packet.summary()

# Show detailed packet information
packet_info = packet.show(dump=True)  # Using `dump=True` to capture the

(summary, packet_info)
```

I've enclosed the Python code in triple backticks as you suggested, so the code will be executed and the results displayed accordingly.

Here is the summary and detailed breakdown of the provided hexdump:

Summary:

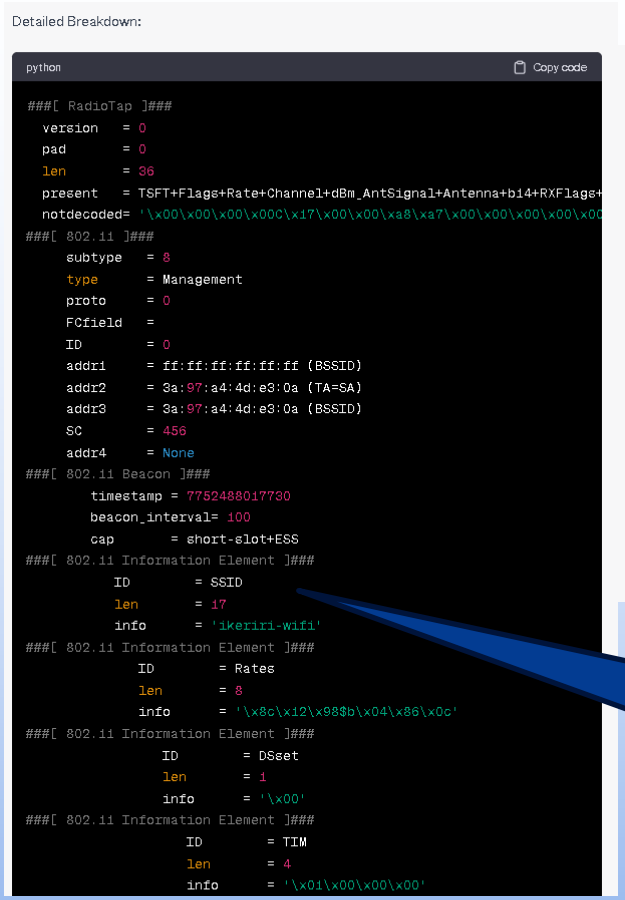
arduino — Copy code

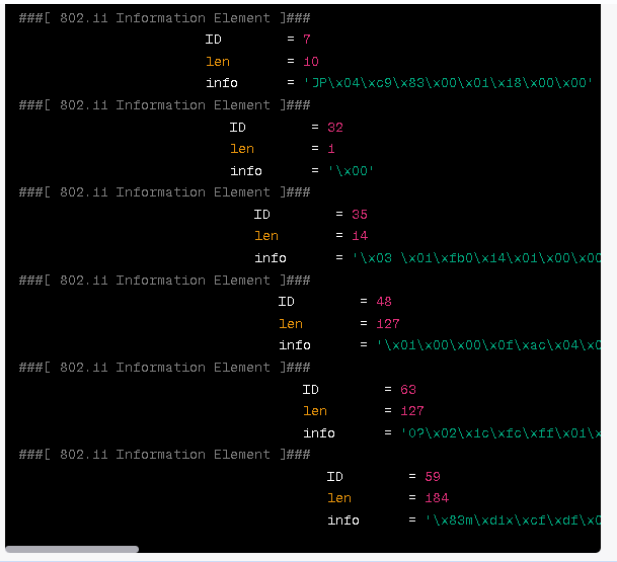
```
RadioTap / 802.11 Management Beacon 3a:97:a4:4d:e3:0a (TA=SA) > ff:ff:ff
```

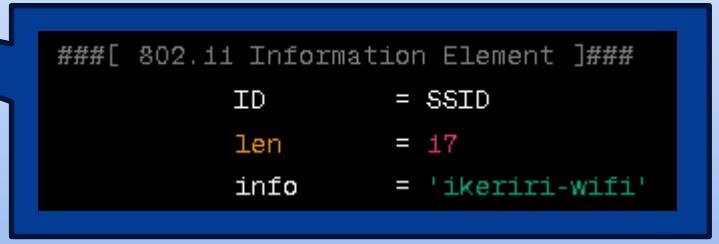
Packet Summary:

RadioTap / 802.11 Management Beacon 3a:97:a4:4d:e3:0a (TA=SA) > ff:ff:ff:ff:ff:ff (BSSID=DA) / Dot11Beacon / Dot11EltRates / Dot11EltDSset / Dot11EltTIM / Dot11Elt / Dot11EltRSN

Use code execution to perform more accurate calculations







GPT4 dissects WiFi6E beacon based on Wireshark dissector!!

Last prompt (Prompt20.txt)
Tell me the reason why we use Wireshark

SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov

#sf23eu

Wireshark is a popular open-source packet analyzer. It is used for a variety of reasons, including:

1. **Network Troubleshooting**: By capturing and analyzing network traffic, Wireshark can help identify network problems, such as connectivity issues, dropped packets, or unexpected traffic patterns.
2. **Security Analysis**: Wireshark can be used to detect security threats, such as unauthorized access attempts, Distributed Denial of Service (DDoS) attacks, or suspicious packet patterr indicative of malware.
3. **Protocol Analysis**: Engineers and developers use Wireshark to analyze how specific protoc work and interact. This can be especially useful when developing new software or trying to understand the intricacies of network protocols.
4. **Learning and Education**: Wireshark is often used in educational settings, like networking courses, to teach students about how different network protocols function and how data moves through a network.
5. **Network Forensics**: In the aftermath of a security incident, Wireshark can be used to analyze past network traffic and understand the sequence of events leading up to the incident.
6. **Performance Analysis**: Wireshark can help in determining bottlenecks and areas of inefficiency in a network. By understanding where traffic is heavy or slow, optimizations can be made.
7. **Validation**: Network administrators and engineers can use Wireshark to validate that Quality of Service (QoS) is working as expected or that network configurations are correct.
8. **Monitoring and Management**: While Wireshark is not a long-term monitoring solution, it can be useful for real-time analysis to monitor specific traffic patterns or data flows.

It's worth noting that while Wireshark is a powerful tool, using it requires a good understanding of networking concepts. Also, capturing network traffic without proper authorization can be illegal or against organizational policies, so it's essential to use Wireshark ethically and responsibly.

USE WIRESHARK

SharkFest'23 EUROPE
Brussels, Belgium · 30 Oct-3 Nov

#sf23eu

Thank you for watching.

Please complete the Google form survey

trace files and Wireshark profiles are here:
https://www.ikeriri.ne.jp/sharkfest/
ChatGPTwithWireshark.zip

ikeriri network service
http://www.ikeriri.ne.jp