# NullCon CTF Write-up: USBNET

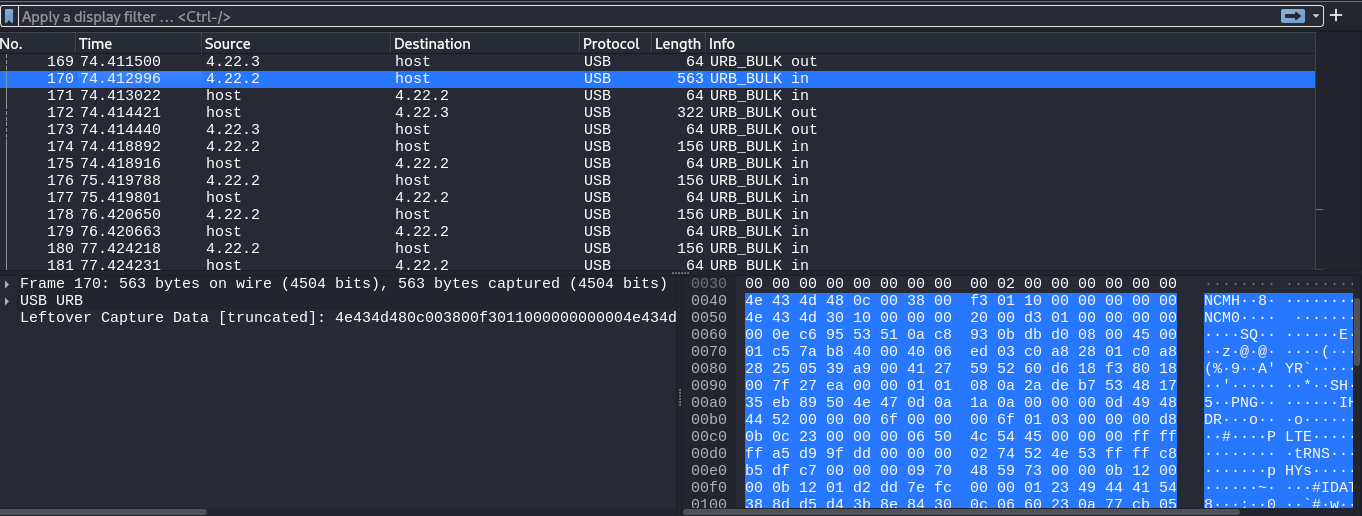
## Challenge Overview

In this challenge, we were provided with a `.pcapng` file that contained USB traffic. Our task was to analyze the packets and extract useful data from the capture. Upon investigating the capture, we discovered that it contained USB Ethernet traffic, and some packets had meaningful data embedded in them.

## Initial Analysis

We opened the `.pcapng` file in \*\*Wireshark\*\* and noticed several packets containing data with `NCMA` and `NCM0` identifiers. This suggested that the packets were part of an \*\*Ethernet over USB\*\* communication.

Upon further inspection, we found a packet larger than 500 bytes that contained recognizable \*\*PNG file signatures\*\*:



- `IHDR` (Image Header)

- `PLTE` (Palette)

- `IDAT` (Image Data)

- `IEND` (End of Image)

This indicated that a \*\*PNG image\*\* was embedded within the USB data transfer.

## Extracting the PNG

To automate the extraction, we wrote a \*\*Python script\*\* using \*\*Scapy\*\* to parse the `.pcapng` file and extract the PNG image. The script searches for the PNG magic bytes (`\x89PNG\x0D\x0A\x1A\x0A`), extracts the data until the `IEND` marker, and saves it as a valid image file.

### Extraction Script

```python

from scapy.all import rdpcap

# Path to your pcapng file

pcap\_file = "usbnet.pcapng"

output\_png\_file = "extracted\_image.png"

def extract\_png\_from\_pcap(pcap\_file, output\_png\_file):

# Read the pcap file

packets = rdpcap(pcap\_file)

# Loop through each packet and search for the PNG signature

for packet in packets:

# Ensure the packet contains raw payload (USB communication with data)

if packet.haslayer("Raw"):

payload = bytes(packet["Raw"].load)

# Search for the PNG start marker (The PNG signature header: 89 50 4e 47)

png\_start = payload.find(b'\x89\x50\x4e\x47') # PNG header (89 50 4e 47)

if png\_start != -1:

# Search for the PNG end marker (IEND chunk: '49 45 4e 44')

png\_end = payload.find(b'\x49\x45\x4e\x44', png\_start)

# Check if the end of the PNG (IEND) was found and ensure there's data after the start

if png\_end != -1 and png\_end > png\_start:

# Extract the PNG image data (from PNG header to IEND)

png\_data = payload[png\_start:png\_end + 4] # Include the IEND marker

# Write the extracted PNG data to a file

with open(output\_png\_file, 'wb') as f:

f.write(png\_data)

print(f"PNG image extracted and saved to {output\_png\_file}")

return

print("No valid PNG data found in the pcap file.")

# Run the extraction function

extract\_png\_from\_pcap(pcap\_file, output\_png\_file)

```

## Results

After running the script, we successfully extracted `extracted\_image.png`. Opening the image revealed a hidden flag inside the picture, completing the challenge.



This gave the flag:

ENO{REDACTED}

## Conclusion

This challenge demonstrated:

- \*\*Analyzing USB packet captures\*\* using \*\*Wireshark\*\*.

- \*\*Recognizing embedded files\*\* in network traffic.

- \*\*Extracting raw binary data\*\* using \*\*Scapy\*\*.

This method can also be applied in \*\*forensics and reverse engineering\*\* where file extraction from network traffic is required.

### Tools Used

- \*\*Wireshark\*\* (Packet analysis)

- \*\*Scapy\*\* (Packet manipulation in Python)

- \*\*strings\*\* (Checking extracted file contents)

### Final Thoughts

This was an interesting challenge that tested our ability to work with \*\*USB traffic and embedded data extraction\*\*. Understanding protocols like \*\*USB Ethernet (NCM)\*\* and identifying file signatures in network captures can be crucial in real-world \*\*digital forensics and cybersecurity investigations\*\*.

Let me know if you have any questions or suggestions!

---

\*\*Author: SPECT3R\*\*

\*\*Event: NullCon CTF\*\*