0day Wireless Penetration Tester CTF Exam

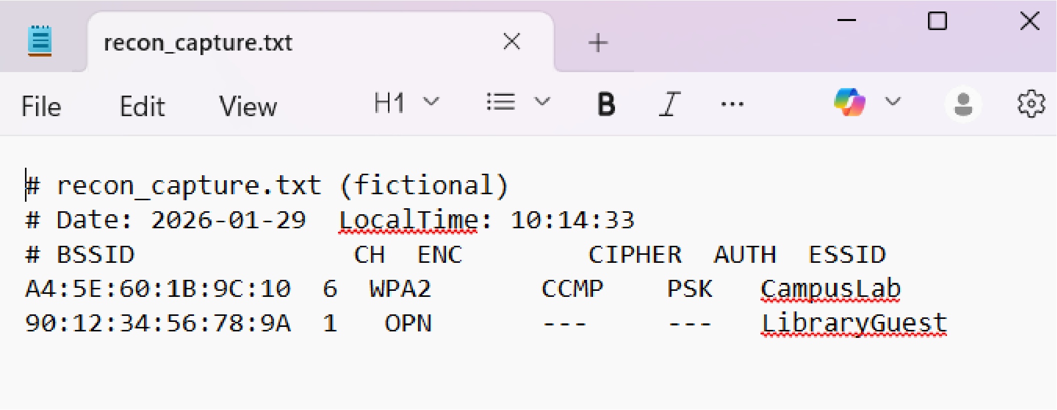
Team: M41N2

Table of Contents

1.0 Easy Peasy Lemon Squeezy

1.1 I See You

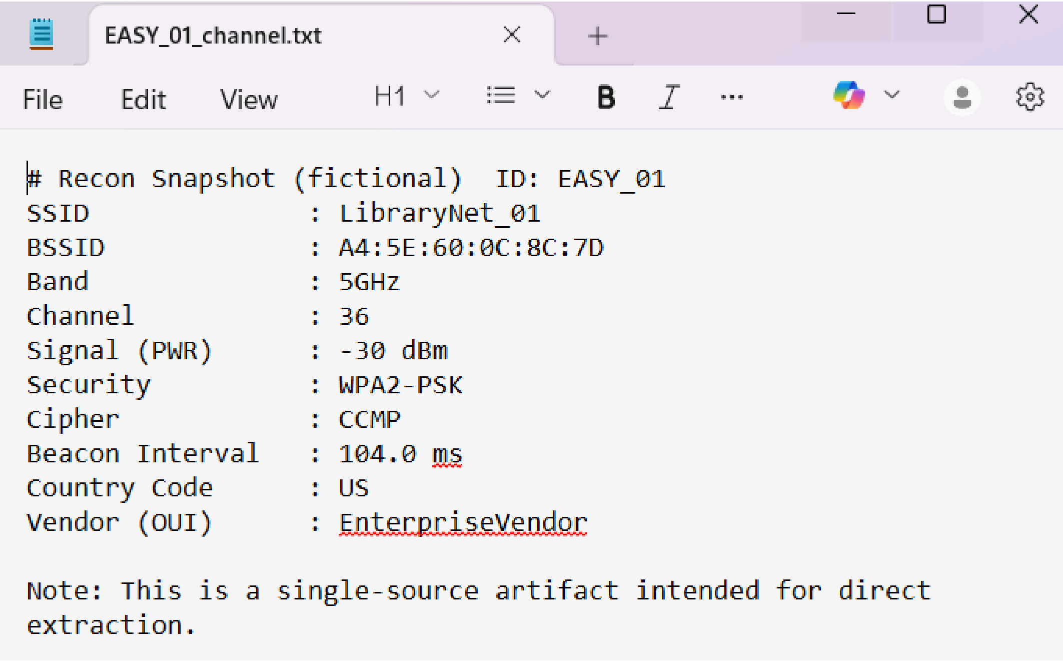
This challenge is to determine the BSSID of the CampusLab, which is A4:5E:60:1B:9C:10.



```
# recon_capture.txt (fictional)
# Date: 2026-01-29  LocalTime: 10:14:33
# BSSID              CH  ENC       CIPHER  AUTH  ESSID
A4:5E:60:1B:9C:10  6  WPA2      CCMP     PSK    CampusLab
90:12:34:56:78:9A  1   OPN      ---      ---    LibraryGuest
```

1.2 I See You 2

This challenge is to determine the channel of this SSID, which is 36.

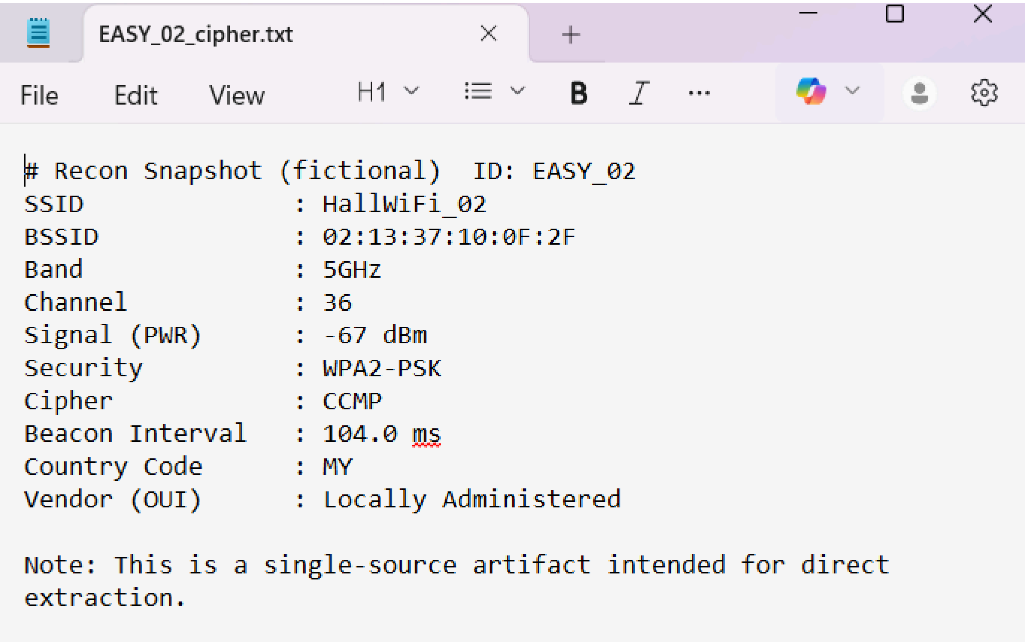


```
# Recon Snapshot (fictional)  ID: EASY_01
SSID                 : LibraryNet_01
BSSID                : A4:5E:60:0C:8C:7D
Band                 : 5GHz
Channel              : 36
Signal (PWR)         : -30 dBm
Security             : WPA2-PSK
Cipher               : CCMP
Beacon Interval      : 104.0 ms
Country Code         : US
Vendor (OUI)         : EnterpriseVendor

Note: This is a single-source artifact intended for direct
extraction.
```

1.3 I See You 3

This challenge is to determine the cipher used for this SSID, which is CCMP.

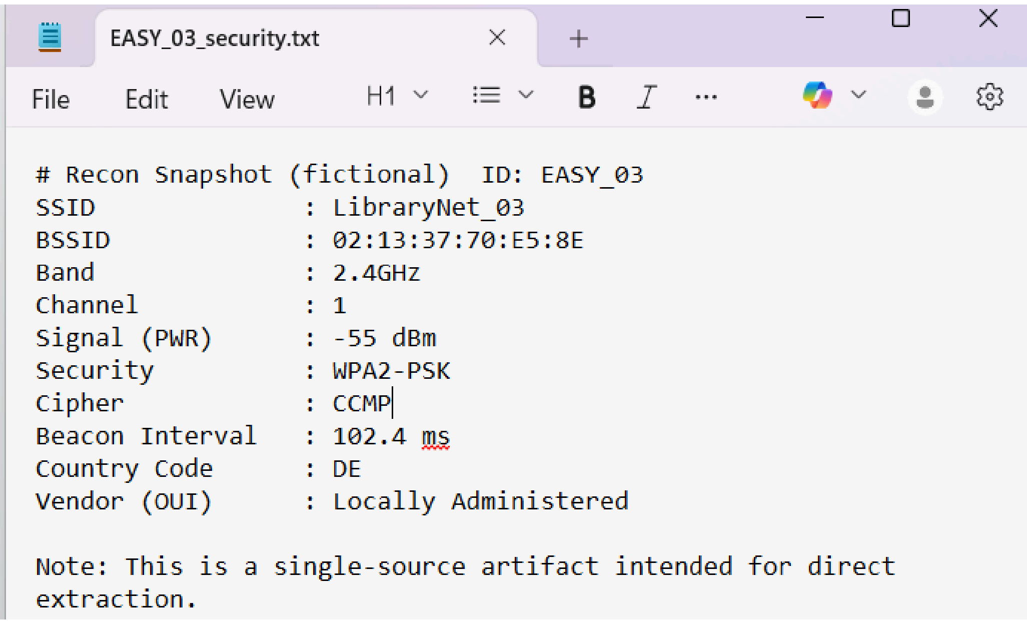


```
# Recon Snapshot (fictional)  ID: EASY_02
SSID                 : HallWiFi_02
BSSID                : 02:13:37:10:0F:2F
Band                 : 5GHz
Channel              : 36
Signal (PWR)         : -67 dBm
Security             : WPA2-PSK
Cipher               : CCMP
Beacon Interval      : 104.0 ms
Country Code         : MY
Vendor (OUI)         : Locally Administered

Note: This is a single-source artifact intended for direct
extraction.
```

1.4 I See You 4

This challenge is to determine the security mode used by this SSID, which is WPA2-PSK.

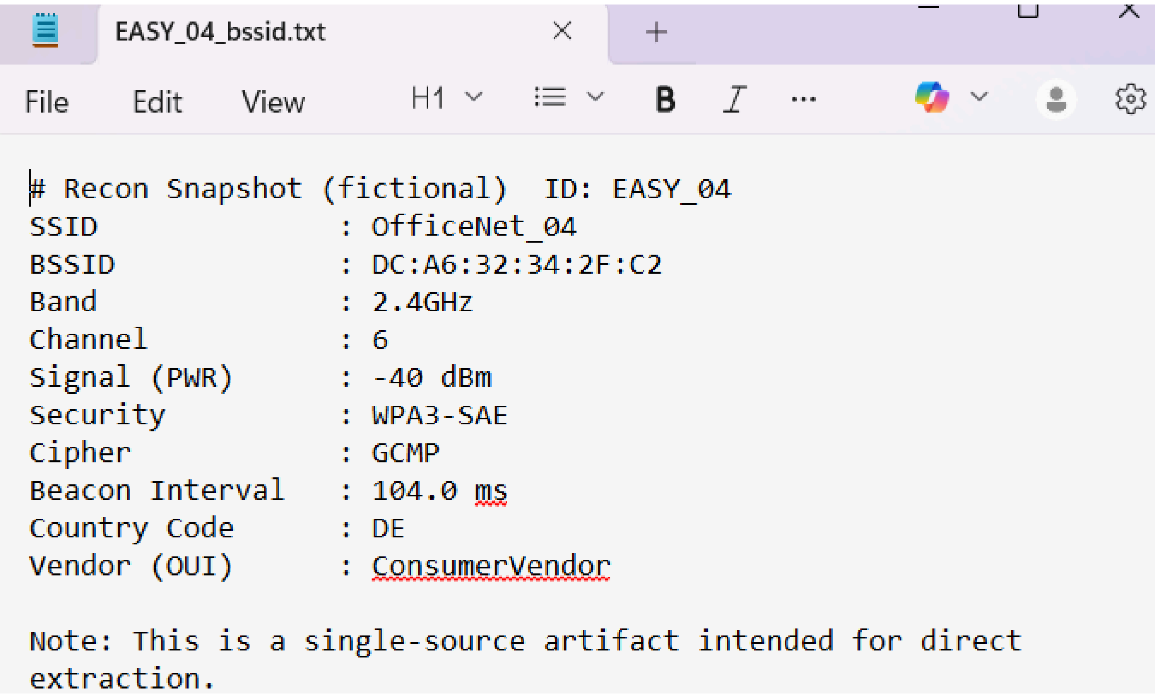


```
# Recon Snapshot (fictional)  ID: EASY_03
SSID                 : LibraryNet_03
BSSID                : 02:13:37:70:E5:8E
Band                 : 2.4GHz
Channel              : 1
Signal (PWR)         : -55 dBm
Security             : WPA2-PSK
Cipher               : CCMP
Beacon Interval      : 102.4 ms
Country Code         : DE
Vendor (OUI)         : Locally Administered

Note: This is a single-source artifact intended for direct
extraction.
```

1.5 Big Mac

This challenge is to determine the MAC address (BSSID) of this SSID, which is DC:A6:32:34:2F:C2.



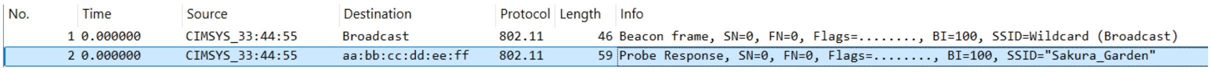
```
# Recon Snapshot (fictional)  ID: EASY_04
SSID                  : OfficeNet_04
BSSID                 : DC:A6:32:34:2F:C2
Band                  : 2.4GHz
Channel               : 6
Signal (PWR)          : -40 dBm
Security              : WPA3-SAE
Cipher                : GCMP
Beacon Interval       : 104.0 ms
Country Code          : DE
Vendor (OUI)          : ConsumerVendor

Note: This is a single-source artifact intended for direct
extraction.
```

1.6 Hidden in Plain Sight

Challenge description: One network is "Hidden," but a client just connected to it, revealing its name in the probe response.

The revealed network name is: Sakura_Garden.



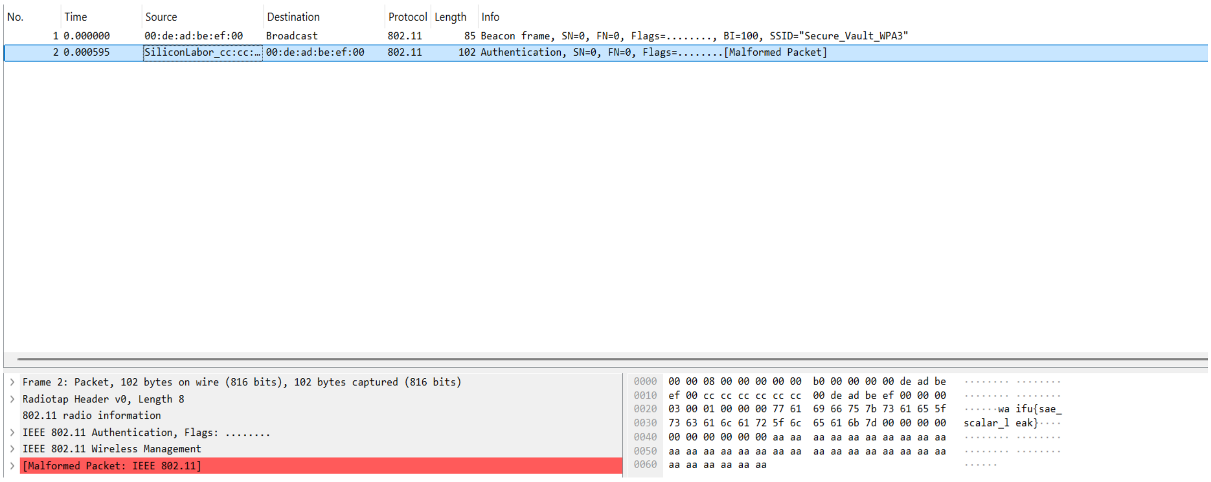
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	CIMSYS_33:44:55	Broadcast	802.11	46	Beacon frame, SN=0, FN=0, Flags=........, BI=100, SSID=Wildcard (Broadcast)
2	0.000000	CIMSYS_33:44:55	aa:bb:cc:dd:ee:ff	802.11	59	Probe Response, SN=0, FN=0, Flags=........, BI=100, SSID="Sakura_Garden"

2.0 Insane in the Membrane!!!

2.1 Dragon's Egg

Challenge description: Instead, look at the math being sent over the air. What is the secret hidden in the exchange?

By inspecting the Authentication packet, the flag is waifu{sae_scalar_leak}.



```
No.     Time       Source                 Destination          Protocol  Length  Info
      1 0.000000   00:de:ad:be:ef:00      Broadcast            802.11        85  Beacon frame, SN=0, FN=0, Flags=........, BI=100, SSID="Secure_Vault_WPA3"
      2 0.000595   SiliconLabor_cc:cc:…   00:de:ad:be:ef:00    802.11       102  Authentication, SN=0, FN=0, Flags=........[Malformed Packet]
```

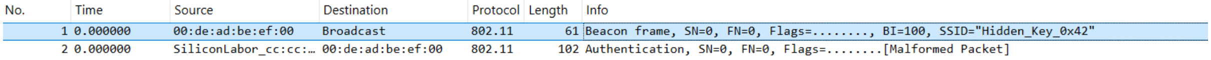
```
> Frame 2: Packet, 102 bytes on wire (816 bits), 102 bytes captured (816 bits)
> Radiotap Header v0, Length 8
  802.11 radio information
> IEEE 802.11 Authentication, Flags: ........
> IEEE 802.11 Wireless Management
> [Malformed Packet: IEEE 802.11]
```

```
0000  00 00 08 00 00 00 00 00  b0 00 00 00 00 de ad be   ········ ········
0010  ef 00 cc cc cc cc cc cc  00 de ad be ef 00 00 00   ········ ········
0020  03 00 01 00 00 00 77 61  69 66 75 7b 73 61 65 5f   ······wa ifu{sae_
0030  73 63 61 6c 61 72 5f 6c  65 61 6b 7d 00 00 00 00   scalar_l eak}····
0040  00 00 00 00 00 00 aa aa  aa aa aa aa aa aa aa aa   ········ ········
0050  aa aa aa aa aa aa aa aa  aa aa aa aa aa aa aa aa   ········ ········
0060  aa aa aa aa aa aa                                  ······
```

2.2 Dragon's Heart

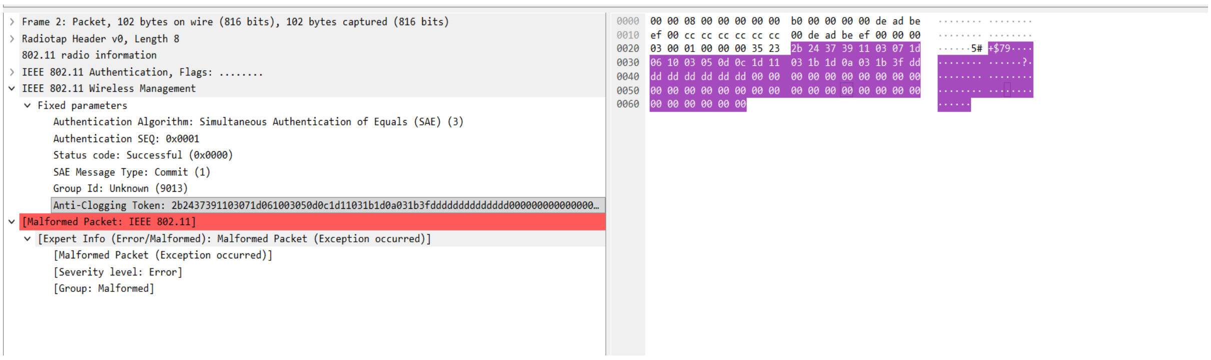
Challenge description: We've captured a single connection attempt. The handshake looks standard at first glance, but the SAE Commit contains more than just cryptographic noise. Find the key, solve the scalar, and recover the architect's secret.

The key is 0x42, as shown in SSID.



```
No.     Time       Source                 Destination          Protocol  Length  Info
      1 0.000000   00:de:ad:be:ef:00      Broadcast            802.11        61  Beacon frame, SN=0, FN=0, Flags=........, BI=100, SSID="Hidden_Key_0x42"
      2 0.000000   SiliconLabor_cc:cc:…   00:de:ad:be:ef:00    802.11       102  Authentication, SN=0, FN=0, Flags=........[Malformed Packet]
```

Now, look for scalar. In the hex dump, look at line 0040. You see a sequence starting with dd dd dd dd dd dd dd....



```
> Frame 2: Packet, 102 bytes on wire (816 bits), 102 bytes captured (816 bits)
> Radiotap Header v0, Length 8
  802.11 radio information
> IEEE 802.11 Authentication, Flags: ........
v IEEE 802.11 Wireless Management
  v Fixed parameters
      Authentication Algorithm: Simultaneous Authentication of Equals (SAE) (3)
      Authentication SEQ: 0x0001
      Status code: Successful (0x0000)
      SAE Message Type: Commit (1)
      Group Id: Unknown (9013)
      Anti-Clogging Token: 2b2437391103071d061003050d0c1d11031b1d0a031b3fdddddddddddddd0000000000000000…
v [Malformed Packet: IEEE 802.11]
  v [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
      [Malformed Packet (Exception occurred)]
      [Severity level: Error]
      [Group: Malformed]
```

```
0000  00 00 08 00 00 00 00 00  b0 00 00 00 00 de ad be   ········ ········
0010  ef 00 cc cc cc cc cc cc  00 de ad be ef 00 00 00   ········ ········
0020  03 00 01 00 00 00 35 23  2b 24 37 39 11 03 07 1d   ······5# +$79····
0030  06 10 03 05 0d 0c 1d 11  03 1b 1d 0a 03 1b 3f dd   ········ ······?·
0040  dd dd dd dd dd dd 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00                                  ······
```

The Red Flag: Standard cryptographic scalars are random-looking (high entropy). A repeating pattern like dd dd dd... is a classic CTF indicator that this is where the "backdoor" or the secret message is hidden.

Copy the selected bytes as a Hex Stream and paste it in the following code.

```
scalar_hex =
"0000080000000000b000000000deadbeef00cccccccccccc00deadbeef000000030001
00000035232b2437391103071d061003050d0c1d11031b1d0a031b3fdddddddddddddd0
000000000000000000000000000000000000000000000000000000000000000"
key = 0x42

scalar_bytes = bytes.fromhex(scalar_hex)
# XORing the scalar with the hidden key
flag = "".join(chr(b ^ key) for b in scalar_bytes)

print(f"Result: {flag}")
```

This Python script is a classic "XOR Cipher" decoder. In this CTF challenge, the "backdoor in the math" refers to the fact that the architect didn't use a random number for the scalar; instead, he took a secret text (the flag) and masked it using a simple bitwise operation with the key 0x42.

Encryption: flag XOR key = scalar
Decryption: scalar XOR key = flag

Flag: waifu{SAE_DRAGON_SAY_HAY}

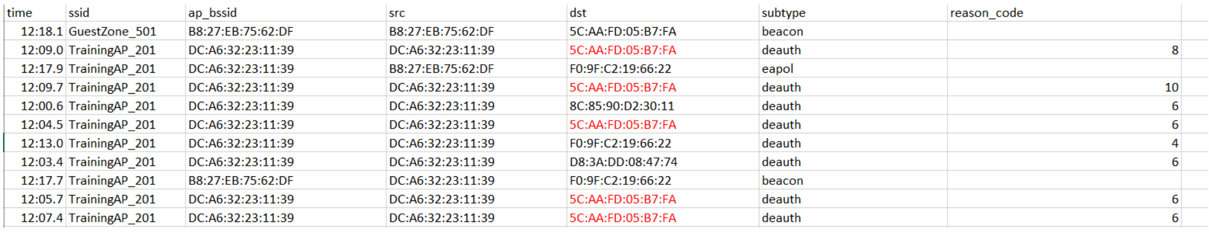
3.0 Now We’re Cooking

3.1 Target Lock In Sight

Challenge description: From the raw events, determine which client STA is targeted most frequently by DEAUTH for the primary SSID and legit BSSID.

Primary SSID: TrainingAP_201
BSSID: DC:A6:32:23:11:39

From a portion of the log file, I can observe that the deauth frame was sent frequently to the client with MAC address 5C:AA:FD:05:B7:FA.



time	ssid	ap_bssid	src	dst	subtype	reason_code
12:18.1	GuestZone_501	B8:27:EB:75:62:DF	B8:27:EB:75:62:DF	5C:AA:FD:05:B7:FA	beacon	
12:09.0	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	5C:AA:FD:05:B7:FA	deauth	8
12:17.9	TrainingAP_201	DC:A6:32:23:11:39	B8:27:EB:75:62:DF	F0:9F:C2:19:66:22	eapol	
12:09.7	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	5C:AA:FD:05:B7:FA	deauth	10
12:00.6	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	8C:85:90:D2:30:11	deauth	6
12:04.5	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	5C:AA:FD:05:B7:FA	deauth	6
12:13.0	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	F0:9F:C2:19:66:22	deauth	4
12:03.4	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	D8:3A:DD:08:47:74	deauth	6
12:17.7	TrainingAP_201	B8:27:EB:75:62:DF	DC:A6:32:23:11:39	F0:9F:C2:19:66:22	beacon	
12:05.7	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	5C:AA:FD:05:B7:FA	deauth	6
12:07.4	TrainingAP_201	DC:A6:32:23:11:39	DC:A6:32:23:11:39	5C:AA:FD:05:B7:FA	deauth	6

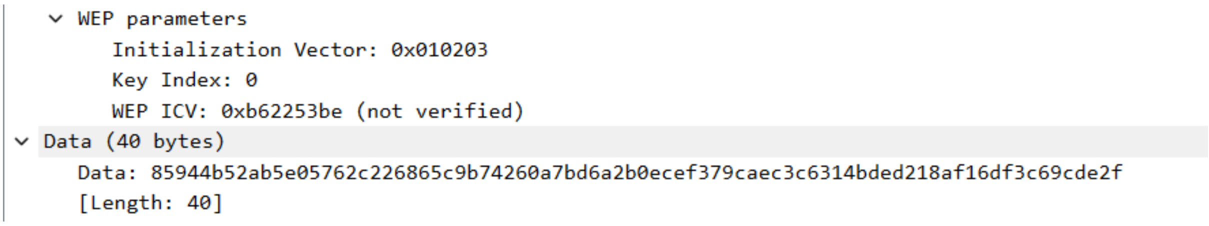
3.2 Counting from 1234567890

Challenge description: Identify the 40-bit WEP key. Decrypt the transmission to reveal the flag hidden within the data.

Attack overview:
RC4 key = IV (3 bytes) + Secret key (5 bytes)
ciphertext = plaintext XOR RC4_keystream
plaintext = ciphertext XOR RC4_keystream

Step 1: Extract the data

40-bit WEP key (secret_key): 1234567890
Initialization Vector(IV): 010203
Ciphertext:
85944b52ab5e05762c226865c9b74260a7bd6a2b0ecef379caec3c6314bded218af16df3c69cde2f



```
  ⌄ WEP parameters
        Initialization Vector: 0x010203
        Key Index: 0
        WEP ICV: 0xb62253be (not verified)
⌄ Data (40 bytes)
      Data: 85944b52ab5e05762c226865c9b74260a7bd6a2b0ecef379caec3c6314bded218af16df3c69cde2f
      [Length: 40]
```

Step 2: Build RC4 key

WEP uses:
RC4 key = IV + secret_key
RC4 key = 0102031234567890

Step 3: Decrypt using Python

```
def rc4(key, data):
    S = list(range(256))
    j = 0

    # Key Scheduling Algorithm
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) % 256
        S[i], S[j] = S[j], S[i]

    # Pseudo-Random Generation
    i = j = 0
    output = []

    for byte in data:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        k = S[(S[i] + S[j]) % 256]
        output.append(byte ^ k)

    return bytes(output)

# ====== INPUT ======
iv = bytes.fromhex("010203")
secret = bytes.fromhex("1234567890")
ciphertext =
bytes.fromhex("85944b52ab5e05762c226865c9b74260a7bd6a2b0ecef379caec3c63
14bded218af16df3c69cde2f")

full_key = iv + secret

plaintext = rc4(full_key, ciphertext)

print("Decrypted:")
print(plaintext)
print("\nAs ASCII:")
print(plaintext.decode(errors="ignore"))
```

Flag: waifu{WEP_is_dead_long_live_WPA}