TryHackMe - Cipher

Task Overview:

The task was to analyze telemetry traffic between a SCADA server and a pump controller. The traffic was encrypted using **AES-GCM** with a shared **16-byte nonce**. The objective was to decrypt the second packet and extract the hidden sabotage command containing a flag.

Step 1: Nmap Scan

I began with a quick Nmap scan to check for open ports and services. Upon scanning, I found a variety of services including HTTP, LDAP, SMB, and others.

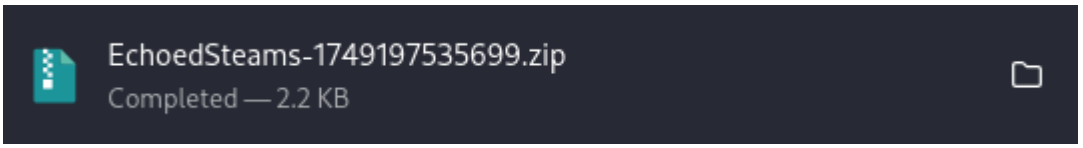
Key Observations from Nmap scan:

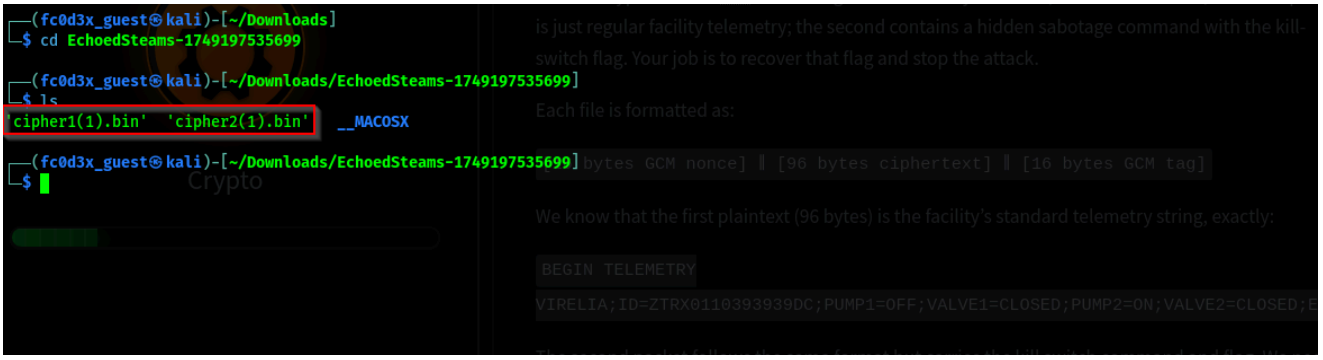
> Several services including LDAP and SMB were running on the target machine, indicating possible access points for further exploration.

Step 2: Analyzing Encrypted Telemetry Packets

I was provided with two encrypted binary files, `cipher1.bin` and `cipher2.bin`, which contained the telemetry data.

> The telemetry data was split into a 16-byte nonce, a 96-byte ciphertext, and a 16-byte GCM tag.
> The first packet (`cipher1.bin`) contained the standard telemetry data, while the second packet (`cipher2.bin`) carried the hidden kill switch command along with the flag.



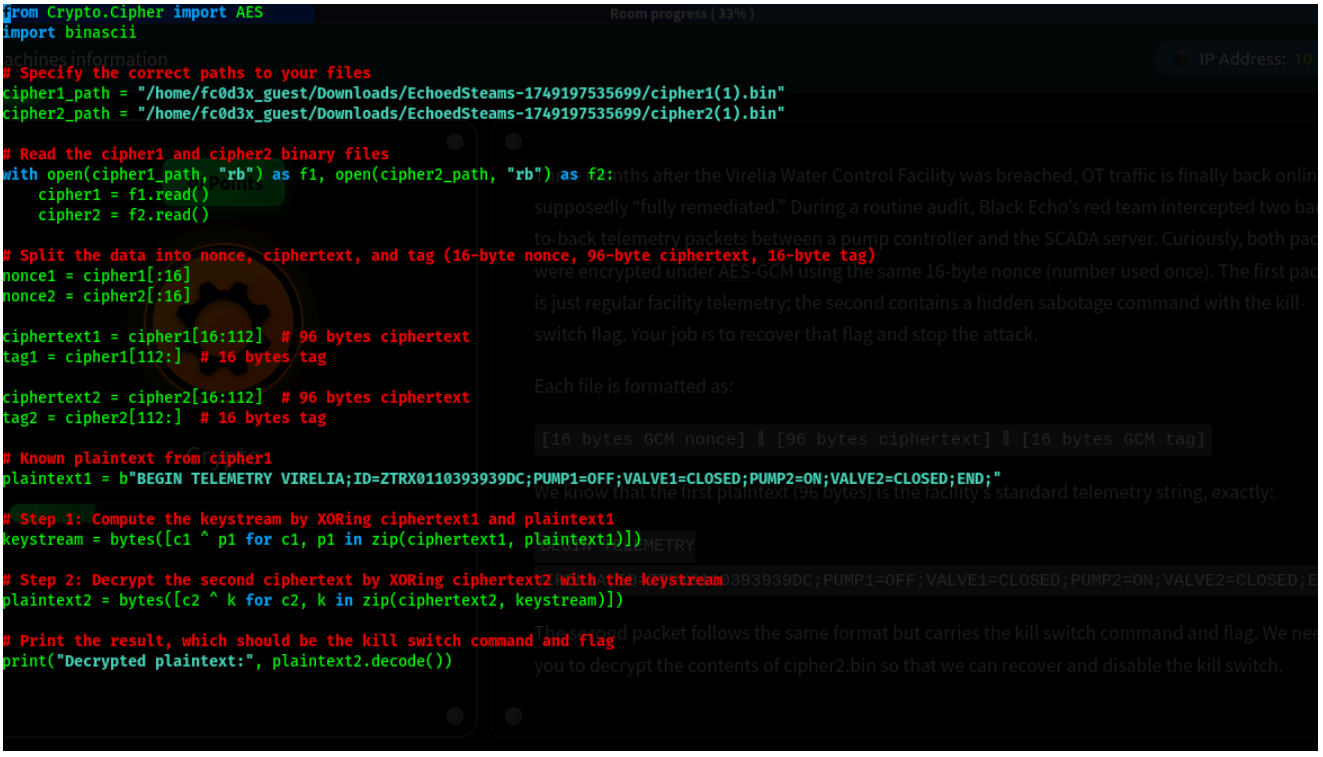


Step 3: Decrypting the Data

To decrypt the second packet, I wrote a Python script that:

- Reads both ciphertext files.
- Extracts the 16-byte nonce, 96-byte ciphertext, and the GCM tag.
- XORs the known plaintext (from the first packet) with the ciphertext to generate the keystream.
- Decrypts the second ciphertext using the keystream.

Python Code Used for Decryption:



```python
from Crypto.Cipher import AES
import binascii

# Specify the correct paths to your files
cipher1_path = "/home/fc0d3x_guest/Downloads/EchoedSteams-1749197535699/cipher1(1).bin"
cipher2_path = "/home/fc0d3x_guest/Downloads/EchoedSteams-1749197535699/cipher2(1).bin"

# Read the cipher1 and cipher2 binary files
with open(cipher1_path, "rb") as f1, open(cipher2_path, "rb") as f2:
    cipher1 = f1.read()
    cipher2 = f2.read()

# Split the data into nonce, ciphertext, and tag (16-byte nonce, 96-byte ciphertext, 16-byte tag)
nonce1 = cipher1[:16]
nonce2 = cipher2[:16]

ciphertext1 = cipher1[16:112]  # 96 bytes ciphertext
tag1 = cipher1[112:]  # 16 bytes tag

ciphertext2 = cipher2[16:112]  # 96 bytes ciphertext
tag2 = cipher2[112:]  # 16 bytes tag

# Known plaintext from cipher1
plaintext1 = b"BEGIN TELEMETRY VIRELIA;ID=ZTRX0110393939DC;PUMP1=OFF;VALVE1=CLOSED;PUMP2=ON;VALVE2=CLOSED;END;"

# Step 1: Compute the keystream by XORing ciphertext1 and plaintext1
keystream = bytes([c1 ^ p1 for c1, p1 in zip(ciphertext1, plaintext1)])

# Step 2: Decrypt the second ciphertext by XORing ciphertext2 with the keystream
plaintext2 = bytes([c2 ^ k for c2, k in zip(ciphertext2, keystream)])

# Print the result, which should be the kill switch command and flag
print("Decrypted plaintext:", plaintext2.decode())
```

Step 4: Revealing the Flag

After running the script, I decrypted the second packet successfully, revealing the kill switch command and flag:


```
THM Echo_Telemetry
```

Conclusion:

The task was a great exercise in working with AES-GCM encryption and learning to decrypt real-world industrial telemetry traffic. By using a known plaintext attack combined with Python, I was able to recover the hidden flag and disable the kill switch in the second telemetry packet.