CTF Group 5 Intended Solution

Guo Yuchen 1004885
Meng Fanyi 1004889
Xiang Siqi 1004875
Zach Lim 1002141

0x00: Overview

The attacker has a stripped version of the code, and an encoded image "mona_lisa_modified.pgm".
From the code, the attacker can see that the flag is firstly encoded using base64, then encrypted using a variant Vigenere table by a secret key, and then hidden in the last bit of the pgm image.
Thus what the attacker needs to do now is to extract the ciphertext of the flag, and then brute-force the key which is used for Vigenere encryption.

0x01: Recover ciphertext from image

From the given code, an attacker should be able to see how the image that has been provided to them was generated. They should quickly realize that there is some sort of message embedded in the body of the image.

Examining the functions "embed()" and "embed_one()", they should then be able to discover that a cipher has been embedded into the image, with the last bit of each line replaced by a bit from a ciphertext. The number of lines that are changed depends on the length of the ciphertext.

Looking further up at the code, they should be able to find that the length of the ciphertext is 64 bytes, meaning that the first 64 lines of the image were altered.

The final conclusion they should come to is that the first 64 lines of the image file have been altered. For each of those lines, the last bit of the first 8 digits should be combined to form a byte, which can then be converted to its ASCII representation. Combining these 64 bytes will give them the ciphertext.

0x02: Retrieve the flag by breaking the ciphertext

After combining the bits, convert them into base64 encoding format. Then the attacker needs to guess the key used for Vigenere encryption by making use of the first 6 fixed characters of the flag “fcs22{“. This will give the first 8 characters of the key “counters”. Then the attacker may choose different ways to guess the rest of the characters, such as brute force, guessing by trying different words with a length of 13, or by observing patterns from a partially decrypted flag and manually trying out the rest.

0x03: Intended solution code:

(For the full attack scipt, please see the attack folder)

```
from EmbedAndExtract import *
from image_split_combine import *
from VariantVigenere import *

variantVigenere = VariantVigenere()

# Step 1: extract text from the image
split_image("mona_lisa_modified.pgm", "header_orig.txt",
"body_modified.txt")
ciphertext_recovered = extract("body_modified.txt")
print("ciphertext_recovered = ", ciphertext_recovered)
#1O7a5B0nKhnM4iJWBz/TGyob/VxHHNqTGS+K/q/B/kAZ2BIOz0pV2urWIUMbIhfh

# Step 2: guess the key
# by mapping the table, we can get the first 8 characters of the key
is "counters"
variantVigenere1 = VariantVigenere()
# since base64 encode 6 bits at a time instead of 8 bits
encoded_prefix = variantVigenere1.string_to_base64("fcs22{")
key_prefix =
variantVigenere1.guess_key(ciphertext_recovered[:len(encoded_prefix)]
,encoded_prefix)
print("key_prefix =",key_prefix) #counters

# Step 3
# Method 1: bruteforce the rest
for a in range(97,123):
    for b in range(97,123):
        for c in range(97,123):
            for d in range(97,123):
                for e in range(97,123):
                    key_test =
key_prefix+chr(a)+chr(b)+chr(c)+chr(d)+chr(e)
                    try:
                        variantVigenere.decrypt(ciphertext_recovered,
key_test)
                        flag_recovered = variantVigenere.get_plain()
                        print("flag_recovered = ", flag_recovered)
                    except:
                        continue

# Method 2: stop brute forcing after certain pattern is found
# by luck can find some word that can reveal part of the key
# e.g. "found_"
encoded_part = variantVigenere1.string_to_base64("found_")
print(encoded_part)
for start in range(len(ciphertext_recovered)):
    try:
        key_part =
variantVigenere1.guess_key(ciphertext_recovered[start:start+len(encod
ed_part)],encoded_part)
        print("key_part =",key_part) #key_part = unterstr
    except:
        break
# bruteforce/googling fill the rest : counterstrike

# Method 3: guessing certain part of the plaintext and map back
manually
```