[CTF] Special Caesar

Created	@September 25, 2025 4:05 PM
Class	CTF
Text	Problem Link: https://play.picoctf.org/playlists/17?m=136

- We got this Encryption code and this message `"lkmjkemjmkiekeijiiiglJlhilihliikiliginliljimiklligljiflhiniiiniiihlhilimlhijil"`:

```python
import string

LOWERCASE_OFFSET = ord("a")
ALPHABET = string.ascii_lowercase[:16]

def b16_encode(plain):
        enc = ""
        for c in plain:
                binary = "{0:08b}".format(ord(c))
                enc += ALPHABET[int(binary[:4], 2)]
                enc += ALPHABET[int(binary[4:], 2)]
        return enc

def shift(c, k):
        t1 = ord(c) - LOWERCASE_OFFSET
        t2 = ord(k) - LOWERCASE_OFFSET
        return ALPHABET[(t1 + t2) % len(ALPHABET)]

flag = "redacted"
key = "redacted"
assert all([k in ALPHABET for k in key])
assert len(key) == 1

b16 = b16_encode(flag)
enc = ""
for i, c in enumerate(b16):
        enc += shift(c, key[i % len(key)])
print(enc)
```

- Code explanation:
  - `b16_encode` function:
    - Each letter “c” in plaintext string is changed into ASCII code, then presented as 8-bit binary.
    - Separate 8-bit in to 2 groups of 4-bits:
      - `binary[:4]` (high bit) → be changed into number → mapping with an character in `ALPHABET`
      - `binary[4:]` (low bit) → do the same as high bit

    ⇒ Each letter will be mapped with 2 others letters in ALPHABET
    - For example: “A” converted into ASCII code is 65 = “01000001”
      - “0100” = 4 → “e”
      - “0001” = 1 → “b”

      ⇒ A → “eb”
  - `shift` function:
    - `c` and `k` is the characters, and `t1` and `t2` is the position
      - Code to get the position of a character: `pos = ord(c) - LOWERCASE_OFFSET`
    - The main purpose is move `c` right `t2` steps in the alphabet.
  - Main function:
    - `flag` is the plaintext (string need to be encrypted)
    - `b16` is the **encoded string** of flag.
    - For each letter in `b16`, encrypt with the key word: `key[i % len(key)]`.
    - Push the encrypted letters back of `enc`.
- **SOLUTION: Reverse the process and Brute Force all 16 keys**
- Decryption Code:

```python
import string

LOWERCASE_OFFSET = ord('a')
ALPHABET = string.ascii_lowercase[:16]  # 'a' to 'p'

#decode function
def b16_decode(encoded_str):
    decode_string = " "
    for char in range(0, len(encoded_str), 2):
        step = ""
        first_char = "{0:04b}".format(ALPHABET.index(encoded_str[char]))
        step += first_char
        second_char = "{0:04b}".format(ALPHABET.index(encoded_str[char + 1]))
        step += second_char
        byte_value = int(step, 2)
        decode_string += chr(byte_value)
    return decode_string

def unshift (char, key):
    t2 = ord(char) - LOWERCASE_OFFSET
    t1 = ord(key) - LOWERCASE_OFFSET
    t3 = (t2 - t1) % len(ALPHABET)
    return ALPHABET[t3]

def decrypt(encrypted_str, key):
    decrypted_str = ""
    for i, char in enumerate(encrypted_str):
        decrypted_str += unshift(char, key[i % len(key)])
    decrypted_str = b16_decode(decrypted_str)
    return decrypted_str

encryped_message = "mlnklfnknljflfmhjimkmhjhmljhjomhmmjkjpmmjmjkjpjojgjmjpjojojnjojmmkmlmijimhjmmj"
for key in ALPHABET:
    decrypted_message = decrypt(encryped_message, key)
    print(f"Key: {key}, Decrypted Message: {decrypted_message}")
    # print("picoCTF{",decrypted_message,"}")
```

- Code explanation:
  - `b16_decode` function (reverse process of `b16_encode` function above):
    - Get 2 consecutive characters in the encoded string → get their index
    - Convert each index into 4-bit and merge into 2 group (4-bit + 4-bit = 8-bit = 1 byte)
    - Change into ASCII code
  - `unshift` function: `unshift(c,k) = (c - k) mod 16`
  - `decrypt` function:
    - For each characters in `encrypted_str`, using unshift function to eliminate Vigenère cipher
    - Using `b16_decode` to get the plaintext (not be ASCII encoded)