Author: Aman Gupta (aman.gupta@iitgn.ac.in) (23210122)

Assignment: Crypto

Mind your Ps and Qs

- Given the value of N is weak
    - Try to factorize `N` into `p` and `q`.
    - Used the following resources to get `p` and `q`:
        - https://www.alpertron.com.ar/ECM.HTM (was able to fatorize N in 6m 16s)
        - http://factordb.com/ (contains a list of already factorized numbers and out N was there)
- Given we have the valus of `p` and `q`, we can use the following to get the value of `d` and correspondingly `m` with the following block of code:

```
 phi = (p-1)*(q-1)
d = inverse(e, phi)
m = pow(c,d,n)
print(long_to_bytes(m))
```

  Full code can be referenced on the following link: https://github.com/afoolinthefetter/CNS/blob/main/crypto/one.py

Easy Peasy

- The `otp.py` file given to us shows that the 50000 length long key is used in parts to xor with the message to progue the encrypted value.
- We plan to cross over the entire 50000 length and then reach the start of the key, then `xor` it with a particular message to produce the encrypted message.
- Since we konw the key that was given to us int the beginning has been xored with the same portion of the key, we will perform the following command to get us the actual key:

```
 ef = 0x0345376e1e5406691d5c076c4050046e4000036a1a005c6b1904531d3941055d
ea = 0x0346303d1902033d1959003d1903553d1951553d1907593d1951511a3d190505
aa = 0x6161616161616161616161616161616161616161616161616161616161616161
decrypted = ef ^ ea ^ aa
print("{:x}".format(decrypted))
```

- Running this python script will give use the hex value as: `0x61626632663764356564663038323032383037366266643761346366653961`39
- Converthing this hex value into ASCII characters is the flag we need.

New Ceaser

- We can cnage the given script into a solution. By soing the operation for all the possible 16 letters in the ALPHABET.

- Add the follwoing to do the opposite of what b_16 encoder is doing:

```
 def decode(enc):
dec = ""
for i in range(0, len(enc), 2):
    binary = "{0:04b}{1:04b}".format(ALPHABET.index(enc[i]), ALPHABET.index(enc[i+1]))
    dec += chr(int(binary, 2))
return dec
```

- After which we perform the following operation:

```
 encFlag = "ihjghbjgjhfbhbfcfjflfjiifdfgffihfeigidfligigffihfjfhfhfhigfjfffjfeihihfdieiei"
for c in ALPHABET:
    desc = ""
    for k in encFlag:
        desc += shift(c,k)
    print(decode(desc))
```

- In the output, we get something which looked like the key and that is our flag.

Dachshund Attacks

- The attack only specified that the value of `d` is small.
- For small value of d, we perform `Wiener's Attack`.
- The original code for the attack was from this particulat repositorty https://github.com/pablocelayes/rsa-wiener-attack
- The `RSAwienerHacker.py` file was modified to contain `e`, `n` and `c` from the picoctf server as:

```
 e= 65094732460576022293604611776806183225886976548576082075367737380579670217785287220856158758112315941996848460533502689179
n= 833051951938975916107481402269792333913436217783752294315538809214469042223010312854877355140702174040686275672389530482472
c= 617359711304686032557099023326470301824424725146259186614848396471683634489679860681775128897030422399642409274997761168032

d = hack_RSA(e,n)
m = pow(c,d,n)
plainM = "{:x}".format(m)
print(plainM)
```

- This will give us the message in hex format. We change it into characters to get out flag.

credstuff

- The files `usernames.txt` and `passwords.txt` are there and we want the password for user `cultiris` which is on line 378 in the users list. Corresponding the user's id in the password's file there is `cvpbPGS{P7e1S_54I35_71Z3}`.
- Since I can guess that the flag will start with picoCTF, I know that characters have been replaced with some other character and these have equal offset in between them.
- Replacing the characters with the specific characters will give us the flag we need.

rail-fence

- The given cypher text is passed to a python program that decrypts the rail-fence. Reference to the used code snippet is: https://www.geeksforgeeks.org/rail-fence-cipher-encryption-decryption/
- The output of the script is the flag we need.

spelling-quiz

- The texts and also the code made no sense, so I used https://www.boxentriq.com/code-breaking/cipher-identifier#monoalphabetic-substitution-cipher to analyze the cypher.
- Form the analyzer, we get to see that this is Monoalphabetic Substitution Cipher.
- There was a solver for this cypher and using this, I was able to decode the wordds with some confidence.
- Passing the flag along with these gibbrish words, the analyzer gave a meaningful sentence, which is out flag.

vigenere

- The description showed that this is Vigenere cipher, gave the encrypted message and the key.
- Used this online solver for Vigenere at: https://dcode.fr/vigenere-cipher, passed the message and the key.
- Output was the flag.

Double DES

- `netcat` picoctf server to get the flag, choose one plaintext and an encrypted text.
- We are going to use known plaintext attack on this to ger the flag.
- Double DES is susceptible to a meet-in-the-middle assault. This explanation from StackExchange elucidates the method succinctly. Essentially, it begins with the plaintext, then exhaustively tests every conceivable key, encrypting the plaintext and recording the outcomes in a dictionary. Subsequently, the original encrypted data (in this instance, `a0c8972feadffda4`) undergoes brute-force decryption with every possible key, with the results being stored concurrently. The process identifies the overlap between the encrypted and decrypted values, revealing the two keys employed in the Double DES scheme.
- The usage of 6 bit and the remaining 2 bites padded makes this attack easier.
- The script bruteforces the first and second key using the aforementioned exploit. Then it finds the intersection using Python's set class. Finally, now that both keys are known, the encrypted flag is decrypted. Reference: https://github.com/afoolinthefetter/CNS/blob/main/crypto/double_des/ddes.ipynb

Scrambled RSA

- From hit and trial, we can figure out that the rsa is done character wise based on some account of the index of each character.
- We hit and trail for each character at a particular place and check if the portion of reply is in the encrypted flag.
- If it is there, we take it out of the encrypted falg and keep doint this until there is nothing left in the encrypted flag.
- We are using a program to do this. Reference can be found: