VishwaCTF

CHALLENGE NAME :

[INTELLECTUAL HEIR]

DEV :

[ABHISHEK MALLAV]

CATEGORY :

[CRYPTOGRAPHY]

LEVEL :

[HARD]

2024

VishwaCTF

CHALLENGE NAME : [INTELLECTUAL HEIR]

Challenge Description :

You received a package, and you got to know that you are the descendant of **RIADSH.**

There are four files and a safe in the package.

You should analyze the files, unlock the safe, and prove your worth.

The safe has alphanumeric and character combinations.

The flag format is VishwaCTF{safe's combination}

PS: The safe has no lowercase buttons.

Solution :

- Firstly understand the file.py python file

```
def str_to_ass(input_string):
    ass_values = []
    for char in input_string:
        ass_values.append(str(ord(char)))
    ass_str = ''.join(ass_values)
    return ass_str
```

⬆ here we have a simple python function to convert characters to their ASCII values

```
input_string = input("Enter the Combination: ")
result = str_to_ass(input_string)
msg = int(result)
```

⬆ we are taking the input of the safe's combination and passing it to the conversion function

```
#not that easy, you figure out yourself what the freck is a & z
a =
z =

f = (? * ?) #cant remember what goes in the question mark
e = #what is usually used
```

⬆ here '***a***' is '***p***' and '***z***' is '***q***' and '***f*** ' is '***n*** '

(*f = a * z*) as (*n = p * q*)

e is 65537 (what is usually used)

```
encrypted = pow(msg, e, f)
print(str(encrypted))
```

⬆ here the converted ASCII string is encrypted using the public key by **Cryptodome** python library

pow is the function in the Cryptodome library

```
#bamm!! protection for primes
number =
bin = bin(number)[2:]
```

⬆ here '***p***' and '***q***' are converted into binary *(protection for primes)*

```
#bamm!! bamm!! double protection for primes
bin_arr = np.array(list(bin), dtype=int)
result = np.sin(bin_arr)
result = np.cos(bin_arr)
np.savetxt("file1", result)
np.savetxt("file2", result)
```

⬆ here we are applying sine to all elements in binary 'p' and cosine to all elements in binary 'q'

after applying the sine and cosine the output is exported as file1.txt and file2.txt

file2.txt has the sine output *(p)*

file1.txt has the cosine output *(q)*

- To get the flag

Here is a one version on code

```
import numpy as np
from Cryptodome.Util.number import inverse

# STEP 1
# Load the file1.txt, file2.txt
# here I know file2.txt has elemnts is of binary p with sine applied to
each one of them
# and file1.txt has elemnts is of binary q with cosine sine applied to
each one of them
# you have to try both combinations to get the proper values of 'p' and
'q'
sin = np.loadtxt('file2.txt')
cos = np.loadtxt('file1.txt')

# STEP 2
# Apply arcsin,arccos to each element in the array and round to the
nearest integer
arcsin = np.round(np.arcsin(sin)).astype(int)
arccos = np.round(np.arccos(cos)).astype(int)

# STEP 3
# Convert the arcsin,arccos to a single string of 0s and 1s without
spaces
binary_string_sin = ""
for num in arcsin:
    # Convert each integer to a string and append to the result
    binary_string_sin += str(num)

binary_string_cos = ""
for num in arccos:
    binary_string_cos += str(num)

# STEP 4
# Convert binary to integer
p = int(binary_string_sin, 2)
```

```
print("\n p:", p)

q = int(binary_string_cos, 2)
print("\n q:", q)

e = 65537

# the numbers from the 'file.txt' file
encrypted =
44000375142788892584792656252580240396364377558833770950559635604953435875537577248405704298902475097224718428882083188643045996347232835874185893478377598659140097202073654883464209492267818944720217371040986847419882157662733042476799915233970277934638 0

# STEP 5
# Calculate n, phi & d (private key)
n = (p * q)
phi = (p - 1) * (q - 1)
d = inverse(e, phi)

# STEP 6
# Now we have the encrypted message, private key & n
decrypted = pow(int(encrypted), d, n)
print("\n The decrypted ACSII values are: " + str(decrypted))

# STEP 7
# Write a functon to convert ascii values to characters
def ascii_to_string(ascii_values):
    chars = [chr(int(ascii_values[i:i+2])) for i in range(0,
len(ascii_values), 2)]
    return ''.join(chars)

# STEP 8
# Final Output
decrypted_string = ascii_to_string(str(decrypted))
print("\n The decrypted message is: " + decrypted_string)
```

The Decrypted Message is: Y0U_@R3_T#3_W0RT#Y_OF_3

The Decrypted Message is the safe’s combination

Hence the flag is **VishwaCTF{Y0U_@R3_T#3_W0RT#Y_OF_3}**