

Basic Challenge 1

S Chowdhury

Challenge Description

A 1 byte xor key was used to encrypt the flag (using the xor operation)

The encrypted flag has been provided

encrypted hexademial string: 929895938f80918780ab9298959389

Solution

We have the ciphertext string (in hexadecimal), and we know that a 1 byte xor key was used to do the encryption. A 1 byte xor key could range from 0x00 to 0xff, or 0 to 255. So we can write a python script to perform the brute force.

```
input_string = "929895938f80918780ab9298959389"
input_string_bytearray = bytes.fromhex(input_string)
```

Here we have the input string, then we use the *bytes.fromhex()* method to convert this into a byte array. This means that instead of having the string

9298...

We would have an array of bytes.

```
for ii in range(256):
    result = encrypt(input_string_bytearray,ii)
    print(f"key: {hex(ii)}, ",end="")
    for jj in range(len(result)):
        print(f"{chr(result[jj])}",end="")
    print("")
```

We have a for loop, and *ii* can have a value from 0 to 255. We call the *encrypt* function on the byte array, and specify *ii* as the key. Please note that for xoring, if we xor a value with a key, we get the xored value. If we xor the xored value with the same key, we get the original value back. This is why we can use the *encrypt()* function for decryption too. The *encrypt* function would return an array of integers. So, we take each item in the array and use that as an input to the *chr()* function. This function converts the integer into the corresponding ASCII character that's represented by that integer.

Here's the *encrypt()* function:

```
def encrypt(data, key):
    final_encrypted = []
    encrypted_byte = 0
    for ii in range(len(data)):
        encrypted_byte = data[ii] ^ key
        final_encrypted.append(encrypted_byte)

    return final_encrypted
```

We just take a particular item in the data array, then xor it with the key. The encrypted byte gets stored into the array *final_encrypted*, which gets returned.

Here's the result, we can see the flag and the key:

```
key: 0xe9, {q|zf!xniB{q|z`
key: 0xea, xr^yej{mjAxr^yc
key: 0xeb, ys~xdkzlk@ys~xb
key: 0xec, ~ty^cl}klG~ty^e
key: 0xed, ^ux~bm|jmF^ux~d
key: 0xee, |v{}an^inE|v{}g
key: 0xef, }wz|`o~hoD}wz|f
key: 0xf0, bhec^pawp[bhecy
key: 0xf1, cidb~q`vqZcidbx
key: 0xf2, `jga}rcurY`jga{
key: 0xf3, akf`|sbtsXakf`z
key: 0xf4, flag{test_flag} ←
key: 0xf5, gm`fzudru^gm`f|
key: 0xf6, dnceyvqgv]dnce^
```

We could have also used the CyberChef website.

First, put the input into CyberChef:

Operations

Search...

Favourites

To Base64

From Base64

To Hex

From Hex

To Hexdump

From Hexdump

URL Decode

Regular expression

Entropy

Fork

Magic

Data format

Recipe

STEP

BAKE!

Auto Bake

Input

929895938f80918780ab9298959389

Output

929895938f80918780ab9298959389

Then, we need the XOR brute force operation as well as the from hex operation:

Operations

fromhex

From Hex

From Hexdump

From Hex Content

Favourites

Data format

Encryption / Encoding

Public Key

Arithmetic / Logic

Networking

Language

Utils

Date / Time

Extractors

Recipe

From Hex

Delimiter
Auto

XOR Brute Force

Key length
1

Sample length
100

Sample offset
0

Scheme
Standard

☐ Null preserving
 ☒ Print key

☐ Output as hex

Crib (known plaintext string)

STEP

BAKE!

Auto Bake

Input

929895938f80918780ab9298959389

Output

Key = f4: flag{test_flag}
 Key = f5: gm"fzudru^gm"f|
 Key = f6: dnceyvqqv]dnce.
 Key = f7: eobdxwfpw\eobd~
 Key = f8: j`mkwx1•xSj`mkq
 Key = f9: kaljvyh~yRkaljp
 Key = fa: hboiuzk}zQhbois
 Key = fb: icnht{j}|Picnhr
 Key = fc: ndios|m{|Wndiou
 Key = fd: oehnr}lz}Voeht
 Key = fe: lfkmq~oy~Ulfkmw
 Key = ff: mgjlp•nx•Tmgjlv

Python Program

```
def encrypt(data,key):
    final_encrypted = []
    encrypted_byte = 0
    for ii in range(len(data)):
        encrypted_byte = data[ii] ^ key
        final_encrypted.append(encrypted_byte)

    return final_encrypted

input_string = "929895938f80918780ab9298959389"
input_string_bytearray = bytes.fromhex(input_string)

for ii in range(256):
    result = encrypt(input_string_bytearray,ii)
    print(f"key: {hex(ii)}, ",end="")
    for jj in range(len(result)):
        print(f"{chr(result[jj])}",end="")
    print("")
```