set-2

Basic xor

recr{X0r_4int_tough}

Based

recr{Demn_You_just_reached_the_fourth_base_XD}

Implementation

```
def extended_gcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        gcd, x, y = extended_gcd(b % a, a)
        return (gcd, y - (b // a) * x, x)

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
a = 48
b = 18
gcd_value = gcd(a, b)
extended_gcd_value = extended_gcd(a, b)

print(f"GCD of {a} and {b} is: {gcd_value}")
print(f"Extended GCD of {a} and {b} is: {extended_gcd_value}")
```

output;-
```
GCD of 48 and 18 is: 6
Extended GCD of 48 and 18 is: (6, -1, 3)
```

Hecks

->writeups for

1)great snakes

```
import sys
if sys.version_info.major == 2:
    print("You are running Python 2, which is no longer supported. Please
update to Python 3.")
ords = [81, 64, 75, 66, 70, 93, 73, 72, 1, 92, 109, 2, 84, 109, 66, 75,
70, 90, 2, 92, 79]
print("Here is your flag:")
print("".join(chr(o ^ 0x32) for o in ords))
```

flag=`crypto{z3n_0f_pyth0n}`

2)asc||

```
int_array = [99, 114, 121, 112, 116, 111, 123, 65, 83, 67, 73, 73, 95,
112, 114, 49, 110, 116, 52, 98, 108, 51, 125]

ascii_chars = ''.join([chr(num) for num in int_array])

print(ascii_chars)
```

flag=`crypto{ASCII_pr1nt4bl3}`

3)Hex

```
hex_string =
"63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f7
37472696e67735f615f6c6f747d"

flag_bytes = bytes.fromhex(hex_string)
flag = flag_bytes.decode('utf-8')
print(flag)
```

flag=`crypto{You_will_be_working_with_hex_strings_a_lot}`

4)Base64

```
import base64
hex_string = "72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf"
bytes_data = bytes.fromhex(hex_string)
base64_data = base64.b64encode(bytes_data)
```

```
base64_string = base64_data.decode('utf-8')
print(base64_string)
```

flag=crypto/Base+64+Encoding+is+Web+Safe/

---

5)bytes and big integer

```
from Crypto.Util.number import long_to_bytes
long_to_bytes(11515195063862318899931685488813747395775516287289682636499965282714637259206269).decode()
```

flag=`crypto{3nc0d1n6_4ll_7h3_w4y_d0wn}`

6)xor starter

```
given = "label"

print("crypto{", end="")
for x in given:
  print(chr(ord(x)^13), end="")
print("}")
```

flag=`crypto{aloha}`

7)xor properties

```
from pwn import xor
key1 =
bytes.fromhex("a6c8b6733c9b22de7bc0253266a3867df55acde8635e19c73313")
key1_2 = "37dcb292030faa90d07eec17e3b1c6d8daf94c35d4c9191a5e1e"
key2_3 = "c1545756687e7573db23aa1c3452a098b71a7fbf0fddddde5fc1"
flag_key123 = "04ee9855208a2cd59091d04767ae47963170d1660df7f56f5faf"
key2 = xor(bytes.fromhex(key1_2), key1)
key3 = xor(bytes.fromhex(key2_3), key2)
key1_2_3 = xor(bytes.fromhex(key1_2), key3)
flag = xor(bytes.fromhex(flag_key123), key1_2_3)
print(flag.decode())
```

flag=**`crypto{x0r_i5_ass0c1at1v3}`**

8_favourite byte

```
ciphertext =
bytearray.fromhex("73626960647f6b206821204f21254f7d694f7624662065622127234
f726927756d")

flag = ""

for num in range(256):

    results = [chr(n^num) for n in ciphertext]

    flag = "".join(results)

    if flag.startswith("crypto"):
        print(flag)
        print(num)

```

flag=`crypto{0x10_15_my_f4v0ur173_by7e}`