**WRITEUP**

**Team name:** bi0s

**Country:** India

**Username:** bi0s

**Challenge name:** ctr

**Proof of flag:**



*CTF{d6bd1954527310f3f831baa46582f553a9e780d8fa747637d25da1281c24edaf}*

**Summary:**

The main premise of the challenge is AES-CTR nonce and key reuse. We send the payload to receive from which we can decrypt the given ciphertexts using XOR ( operation. For getting the flag however, we record the counters at which our ciphertexts decrypt and convert them to ascii.

**Proof of Solving:**

*Code*

from pwn import \*

from binascii import unhexlify

from tqdm import tqdm

**HOST**='34.107.26.201'

**PORT**=32346

io=remote(**HOST**,**PORT**)

current\_dir = os.path.dirname(\_\_file\_\_)

file\_path = os.path.join(current\_dir, "ctr.txt")

with open(file\_path, 'r') as f:

    r=f.readlines()

    cts = [x[:-1] for x in r[:-1]]+[r[-1],]

cts=[unhexlify(x) for x in cts]

enc\_nonce=[]

for i in tqdm(range(255)):

    io.recvuntil(b'Give me no more than 16 bs\n')

    io.sendline(b'')

    r=io.recvline().decode().replace("Here you go ",'')[:-1]

    ct=unhexlify(r)

    enc\_nonce.append(ct)

iden=[]

flag\_var=0

capture=''

for i in cts:

    for j in enc\_nonce:

        try:

            v=xor(i,j).decode()

            iden.append(v)

            flag\_var=1

            print(f"\33[92m[\*] Worked for {i.hex()} at counter: {enc\_nonce.index(j)} --> {v}\33[0m")

            capture+=chr(enc\_nonce.index(j)+1)

            break

        except UnicodeDecodeError:

            pass

    if flag\_var!=1:

        print(f"\33[91m[!] {i.hex()} is useless... at pos {cts.index(i)+1}\33[0m")

    flag\_var=0

print(len(iden))

print(capture)

*Explanation:*

Since it is a blackbox challenge, some experimentation had to be done. From what is apparent, it is using AES-CTR mode. Upon giving the same input on multiple connections, it provides the same output, here it is clear it is reusing the nonce. Upon giving the same inputs such as ‘a’ multiple times, it gives a different output but it remains the same in multiple connections, from this we can deduce the server is essentially doing:

Where and are constant for multiple connections.

We can send an empty plaintext so we receive just . I arbitrarily chose 255 since just 69 attempts wasn’t covering all the ciphertexts which was apparent from this line I had for testing:

if flag\_var!=1:

        print(f"\33[91m[!] {i.hex()} is useless... at pos {cts.index(i)+1}\33[0m")

Now we put all the ciphertexts in a list and iteratively XOR them with all the 255 we have collected so far.

We capture the counters at which the ciphertext doesn’t give a UnicodeDecodeError and then convert this into ascii.

Hence the flag.