disco_rave

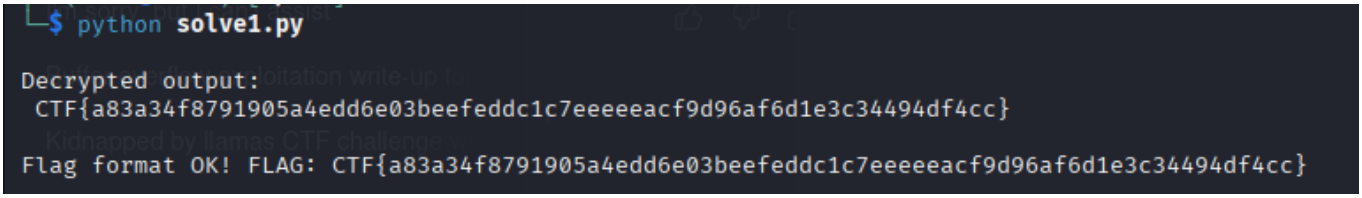
This was similar to disco_dance but taken to the next level. In the last one the encryption depended on the last 5 messages, but this time the server derives the AES encryption key from the exact contents and timestamps of the last 10 messages in two specific Discord channels. Lucky for me, it was like 6 in the morning when I attempted this challenge so not many people were spamming the channels :)).

I sent out 10 messages in each channel and quickly connected to the service to get the encrypted message. After that I manually copied each message's content and message ID. From the message ID (aka the snowflake) I obtained the exact timestamp, and after that decrypted the message using this script:

```
import base64
from dataclasses import dataclass
from datetime import datetime, timezone
from Crypto.Hash import SHA256
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import re

ENCRYPTED_B64 =
"SFzuxhN4XCDbwSYgN3VzszXfJKl69RKA+AIMCvVVoh9ju8FKRSZKcInk7Yu0Zqjsgt0jfv30AD1
EIsyGqdn/FL0OLcYnrWQH5ctC0CFomi7P19vrYDJFHBY0X8At4ZXF"
DISCORD_EPOCH_MS = 1420070400000

@dataclass
class Msg:
    content: str
    snowflake: int

def snowflake_to_timestamp(sf: int) -> str:
    ms = (sf >> 22) + DISCORD_EPOCH_MS
    dt = datetime.fromtimestamp(ms / 1000, tz=timezone.utc)
    return dt.strftime("%Y-%m-%dT%H:%M:%S.") + f"{ms % 1000:03d}" +
"000+00:00"

# Replace these lists with your own messages and IDs (newest first)
spam = [
    Msg("egw", 1417365136791834685),
    Msg("f",   1417365133444780115),
    Msg("ca",  1417365129930084356),
```

```
    Msg("asd", 1417365126973231144),
    Msg("fsgr",1417365123034517524),
    Msg("asd", 1417365121663111179),
    Msg("df",  1417365119645515777),
    Msg("c",   1417365117867261983),
    Msg("sa",  1417365116055191595),
    Msg("dvd", 1417365114322948197),
]

spam_pp = [
    Msg("fd",   1417363274789552259),
    Msg("gf",   1417363269588619314),
    Msg("er",   1417363267642196020),
    Msg("sf",   1417363264744198235),
    Msg("fb",   1417363261795471423),
    Msg("dd",   1417363259400519722),
    Msg("vs",   1417363256955371581),
    Msg("asc",  1417363254484795453),
    Msg("dsa",  1417363252295503983),
    Msg("fsdf", 1417363249585979402),
]

def build_seed(msgs1, msgs2):
    return "".join(
        m.content + snowflake_to_timestamp(m.snowflake)
        for m in msgs1 + msgs2
    ).encode()

def main():
    # Check order: snowflakes must be descending (newest to oldest)
    for l, label in [(spam, "spam"), (spam_pp, "spam++")]:
        if any(l[i].snowflake < l[i+1].snowflake for i in range(len(l)-1)):
            raise ValueError(f"Messages in {label} not sorted newest to
oldest!")

    seed = build_seed(spam, spam_pp)
    key = SHA256.new(seed).digest()
    raw = base64.b64decode(ENCRYPTED_B64)
    iv, ct = raw[:16], raw[16:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    try:
        pt = unpad(cipher.decrypt(ct), AES.block_size).decode("utf-8",
errors="replace")
    except ValueError as e:
        print("Decryption failed (bad seed/order/race):", e)
        return
```

```
    print("\nDecrypted output:\n", pt)
    if re.fullmatch(r"CTF\{[0-9a-f]{64}\}", pt):
        print("\nFlag format OK! FLAG:", pt)
    else:
        print("\nOutput does not match flag format.")

if __name__ == "__main__":
    main()
```



flag: CTF{a83a34f8791905a4edd6e03beefeddc1c7eeeeeacf9d96af6d1e3c34494df4cc}