justCTF2023

Vaulted

Cùng nhìn sơ qua class FlagVault:

```
class FlagVault:
    def __init__(self, flag):
        self.flag = flag
        self.pubkeys = []

    def get_keys(self, _data):
        return str([pk.format().hex() for pk in self.pubkeys])

    def enroll(self, data):
        if len(self.pubkeys) > 3:
            raise Exception("Vault public keys are full")

        pk = PublicKey(bytes.fromhex(data['pubkey']))
        self.pubkeys.append(pk)
        return f"Success. There are {len(self.pubkeys)} enrolled"

    def get_flag(self, data):
        # Deduplicate pubkeys
        auths = {bytes.fromhex(pk): bytes.fromhex(s) for (pk, s) in
zip(data['pubkeys'], data['signatures'])}

        if len(auths) < 3:
            raise Exception("Too few signatures")

        if not all(PublicKey(pk) in self.pubkeys for pk in auths):
            raise Exception("Public key is not authorized")

        if not all(PublicKey(pk).verify(s, b'get_flag') for pk, s in
auths.items()):
            raise Exception("Signature is invalid")

        return self.flag
```

Bài này cho phép gửi một public key lên server. Sau đó có thể gửi payload chứa 3 public keys và một message đã được ký rồi, nếu verify bằng cả 3 keys này mà đúng thì trả về flag. Ý tưởng ở đây là chỉ cần 3 trong 4 public keys của server và 1 cái của user. Nhưng đoạn kiểm tra bị fail:

```
        if not all(PublicKey(pk) in self.pubkeys for pk in auths):
            raise Exception("Public key is not authorized")
```

Hàm này chỉ kiểm tra xem có key nào trong đống public keys trong payload có trên server hay không. Mà một public key lại có thể biểu diễn bằng 3 kiểu là compressed, uncompressed và hybrid. Do đó, chỉ cần tạo một Private Key ký "get_flag" rồi gửi cả 3 định dạng public keys của nó là xong.

```
from pwn import *
from coincurve import PrivateKey
import json

conn = remote('vaulted.nc.jctf.pro', 1337)

# Create a secret key
sk = PrivateKey(b'1234')
pk = sk.public_key

# Send the public key to the remote server
print(json.loads(conn.recvline().decode()))
conn.sendline(json.dumps({'method': 'enroll', 'pubkey':
pk.format().hex()}).encode())

print(json.loads(conn.recvline().decode()))
# Sign message
sig = sk.sign(b'get_flag').hex()

# Create three signatures with compressed, uncompressed and hybrid format
pk1 = pk.format().hex()
pk2 = pk.format(compressed=False).hex()
pk3 = '06' + pk.format(compressed=False).hex()[2:]

conn.sendline(json.dumps({'method': 'get_flag', 'pubkeys': [pk1, pk2, pk3],
'signatures': [sig, sig, sig]}).encode())
print(json.loads(conn.recvline().decode()))
```

Flag: `justCTF{n0nc4n0n1c4l_720378727682891992866632818189293 29}`