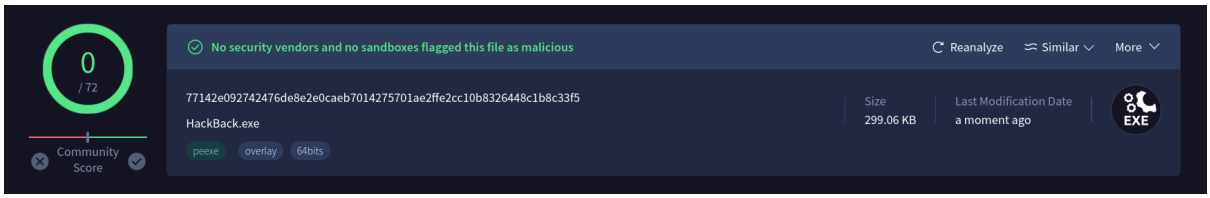
SOLVE:

https://www.embeeresearch.io/ghidra-entropy-analysis-locating-decryption-functions/
https://malware.news/t/reverse-engineering-crypto-functions-aes/55413



Open in Ghidra
Root around until we find the big main function
Label syscalls and look for one that looks like writing virtual memory

Check RDATA (with entropy listing)
Find a big ass section with given offsets
Use the reference feature of Ghidra to see that it's related to the generic Alloc-Write-Protect chain.
Use Ghidra's file offset base to get the offset of the encrypted blob from the original .exe

Now we have the encrypted shellcode!

Retrieve the AES keys from the .rdata section (you can see that the algo is AES256 from the SBOX lookups in ghidra, and you can kinda figure out where the keys are because of Ghidra's references. I left them wide open and un-obfuscated (as global variables), so this should be trivial).

Decrypt sliver implant by writing your own decryptor (see solve.py)

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

with open("Goose.exe", "rb") as file:
    content = file.read()

sc_start = 0x470cf1
sc_end = sc_start + (0x1414dffd0 - 0x140471ef0)

KEY = b"uiuctf{NO_THIS_FLAG_AIN'T_IT321}"
IV = b"YOU\?_+GITGOOSED"

shellcode = content[sc_start:sc_end]
```

```
print(len(shellcode))

def aes256_decrypt(encrypted_data, key, iv):
    if len(key) != 32:
        print("Invalid key!")
    if len(iv) != 16:
        print("Invalid IV!")

    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_data = cipher.decrypt(encrypted_data)

    return decrypted_data

with open("shellcode.recovered", "wb") as file:
    file.write(aes256_decrypt(shellcode, KEY, IV))
```

Now that you've decrypted it, you can figure out that it's a sliver implant.

Get config with:
https://github.com/Immersive-Labs-Sec/SliverC2-Forensics

Decrypt wireshark traffic (remove domain name check and rewrite sliver_pcap_parser to not hex encode everything)

python sliver_pcap_parser.py --pcap ../sliver.pcapng --filter http --domain_name trash

python sliver_decrypt.py --transport http --force ../Goose.DMP --file_path http-sessions.json

uiuctf{W1LD_G0OS3_CH4S3_8234819284901_UNGUESSABLE_1337}