Міністерство освіти і науки України Національний технічний університет «ХПІ»

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерної інженерії та програмування

3BIT

з лабораторної роботи № 12

з дисципліни «Сучасні технології безпечного програмування»

«ДОСЛІДЖЕННЯ ЗАСОБІВ ОБФУСКАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Виконав:

студент гр. КН-Н922б

Кулик Д.І.

Перевірив:

Бульба С. С.

Мета роботи: Дослідити існуючі утиліти обфускації програмного забезпечення.

Індивідуальне завдання

Для обраної мови програмування слід обфускувати будь-який проект.

Слід використовувати декілька утиліт для обфускації. Мінімальна кількість обфускаторів - 4. У звіті навести порівняльну характеристику обраних обфускаторів, що складатиметься з:

- інформації щодо використовуваних методів обфускації
- плюсів та мінусів кожного з обфускаторів.

Навести опис використання кожного з обраних обфускаторів.

Хід роботи

```
def generate_hash(message: bytearray) -> bytearray:
   """Return a SHA-256 hash from the message passed.
   The argument should be a bytes, bytearray, or
   string object."""
   if isinstance(message, str):
       message = bytearray(message, 'ascii')
   elif isinstance(message, bytes):
       message = bytearray(message)
   elif not isinstance(message, bytearray):
       raise TypeError
   # Padding
   length = len(message) * 8 # len(message) is number of BYTES!!!
   message.append(0x80)
   while (len(message) * 8 + 64) % 512 != 0:
     message.append(0x00)
   message += length.to_bytes(8, 'big') # pad to 8 bytes or 64 bits
   assert (len(message) * 8) % 512 == 0, "Padding did not complete properly!"
```

Рисунок 1 – Початковий код (лабораторна робота 01)

Проведемо обфускацію за допомогою утілити руагтог

```
(stbp) C:\Users\Daniil\PycharmProjects\stbp>pyarmor obfuscate LABS\kulyk12\main.py
        PyArmor Trial Version 7.7.3
INFO
        Python 3.8.6
INFO
        Target platforms: Native
INFO
        Source path is "C:\Users\Daniil\PycharmProjects\stbp\LABS\kulyk12"
INFO
       Entry scripts are ['LABS\\kulvk12\\main.py']
     Use cached capsule C:\Users\Daniil\.pyarmor\.pyarmor_capsule.zip
INFO
      Search scripts mode: Normal
INFO
        Save obfuscated scripts to "dist"
INFO
        Read product key from capsule
INFO
      Obfuscate module mode is 2
INFO
     Obfuscate code mode is 1
INFO
       Obfuscate string value is False
INFO
        Wrap mode is 1
INFO
       Restrict mode is 1
INFO
       Advanced value is 0
INFO
      Super mode is False
INFO
       Super plus mode is not enabled
INFO
        Generating runtime files to dist\pytransform
INFO
       Extract pytransform.key
INFO
       Generate default license file
INFO
       Update capsule to add default license file
        Copying c:\users\daniil\pycharmprojects\stbp\lib\site-packages\pyarmor\platforms\windows\x86_64\_pytransform.dll
INFO
        Patch library dist\pytransform\_pytransform.dll
INFO
       Patch library file OK
INFO
      Copying c:\users\daniil\pycharmprojects\stbp\lib\site-packages\pyarmor\pytransform.py
INFO
       Rename it to pytransform/__init__.py
INFO
        Generate runtime files OK
TNFO
        Start obfuscating the scripts..
              main.py -> dist\main.py
INFO
     Patch this entry script with protection code
INFO
        Insert bootstrap code to entry script dist\main.py
INFO
        Obfuscate 1 scripts OK.
(stbp) C:\Users\Daniil\PycharmProjects\stbp>
```

Рисунок 2 – Виконання команди для обфускації



Рисунок 3 – Обфускація коду за допомогою руагтог

```
(stbp) C:\Users\Daniil\PycharmProjects\stbp\LABS\kulyk12>python opy.py
Opy (TM) Configurable Multi Module Python Obfuscator Version 1.1.28
Copyright (C) Geatec Engineering. License: Apache 2.0 at <a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>
Obfuscated words: 180
(stbp) C:\Users\Daniil\PycharmProjects\stbp\LABS\kulyk12>
```

Рисунок 4 – Виконання команди для обфускації

```
import sys
  lall11_opy_ = sys.version_info [0] == 2
l1lll1_opy_ = 2048
l1llll1_opy_ = 7
  def l1l1lll_opy_ (l1lll1l_opy_):
                Global lillil_opy_
lillil_opy_ = ord (lillil_opy_ [-1])
lillil_opy_ = lillil_opy_ [:-1]
lillil_opy_ = lillil_opy_ % len (lilli_opy_)
                lill_opy_ = lill1_opy_ [:lilll1_opy_] + lil11_opy_ [lilll11_opy_:]
if lill1_opy_:
                                 l11111_opy_ = ll_opy_ () .join_([l111l_opy_ (ord_(char) - l11111_opy_ - (l11_opy_ + l11111_opy_) % l111111_opy_) for l11_opy_, char in enumerate_(l111_opy_)])
                                  l11111_opy_ = str_() .join_([chr_(ord_(char) - l1lll1_opy_ - (l11_opy_ + l1l111_opy_) % l1llll1_opy_) for l11_opy_, char in enumerate_(l1ll_opy_)])
                 return eval (l11111_opv_)
0x4026a2749, 0x71374491, 0x00c0T0CT, 0x402000a3, 0x390c0250, 0x59711111, 0x2376224, 0x80165803, 0x408074899, 0x12855041, 0x24618504, 0x560e01fe, 0x9buc06047, 0x1091174, 0xe49b09c1, 0xefbe4786, 0x9fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da, 0x98856152, 0xa831c66d, 0xb0032768, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967, 0x27b70a85, 0x2e1b2138, 0x4d2codfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85, 0xa02bf8e1, 0x8d6a64b, 0xc2db0b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa670, 0x19a4c116, 0x16376c08, 0x2746774c, 0x3d0b0b5, 0x591c0c03, 0x4ed8aa4a, 0x509cca4f, 0x682e6f75, 0x106aa670, 0x76c6160, 0x76c6160, 0x76c6160, 0x76c6160, 0x662e6f75, 0x4668a64a, 0x509cca4f, 0x682e6f75, 0x76c6160, 0x76c616
                 0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
 def l11ll1_opy_(message: bytearray) -> bytearray:
                    l1l1lll_opy_ (u"00000
                                                                                                                                                        ("\texttt{some one consideration of the constant o
                 if isinstance(message, str):
                                   message = bytearray(message, l1l1lll_opy__(u"00000000"))
                 elif isinstance(message, bytes):
```

Рисунок 5 – Обфускація коду за допомогою ору

Висновки: в результаті виконання лабораторної роботи було досліджено існуючі утиліти обфускації програмного забезпечення.