**Writeup LycoReco**

**National Cyber Week 2022 - Final**



**Anggota tim:**

**we love aseng**

**azuketto**

**chaerla**

# **Daftar Isi**

[**Daftar Isi**](#_jtobumv3ewy0) **2**

[**Free Flag**](#_dx0lkflrke9q) **3**

[**🚩 Well Played**](#_yvfftzql2h85) **3**

[Flag: NCW22{final\_family\_friendly}](#_2bgcgjq5b6dq) 3

[**Rev**](#_b9or4c5td3i8) **4**

[👨‍💻 199 passcode](#_dbwcng2mg9az) 4

[Flag: NCW22{congratz\_you\_have\_successfully\_decompiled\_the\_pyc\_then\_solving\_the\_tree\_traversal\_problem\_and\_find\_this\_flag\_damn\_you\_really\_deserve\_it\_champ}](#_t25gvr6jvdgp) 16

# Free Flag

| 🚩 Well Played 100  Welcoming Corpse Party Time!  These flags are valid because we are friendly!  ┬─┬ノ( º \_ ºノ)  NCW22{final\_family\_friendly}  Looking for Crypto and Forensics? Try in hology5!  hology5{final\_family\_friendly} |
| --- |

### 

Tinggal copas dapet flag horeeeee

### 

### Flag: NCW22{final\_family\_friendly}

# 

# **Rev**

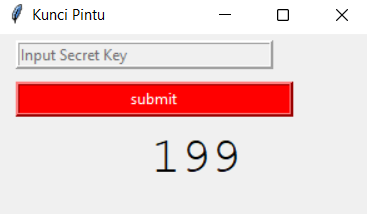
| 👨‍💻 199 passcode 436  There is a lesson you must learn  No matter how much you have earned  When you think the game is over  Mistakes will bring you back to where you were  Author: kisanak#5303 |
| --- |

### 

Diberikan sebuah file pyc chall. Saat dijalankan menggunakan python 3.10, keluar warning bahwa magic number pyc broken.

| from Crypto.Util.number import \*  f = open("199passcodehahahoho.pyc", "rb")  buf = f.read()  print(int.from\_bytes(buf[:2], byteorder='little')) #3413 |
| --- |

Kemudian, setelah mengambil magic number (3413) dan mengeceknya pada dokumentasi python, kami mendapatkan bahwa pyc dicompile menggunakan python 3.8. Kami kemudian mencoba menjalankan file dengan python 3.8, dan pyc berhasil jalan.



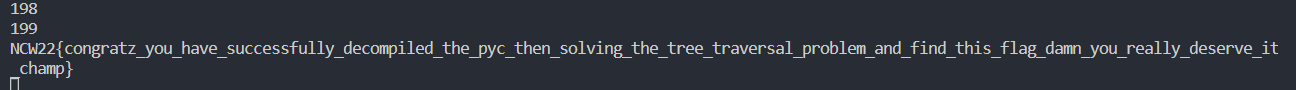
Setelah itu, kami melakukan decompile pada pyc dengan mencari decompiler di internet yang dapat mendecompile python 3.8. Berikut merupakan hasil chall dekompilasi:

| # decompyle3 version 3.9.0  # Python bytecode version base 3.8.0 (3413)  # Decompiled from: Python 3.8.10 (default, Jun 22 2022, 20:18:18)  # [GCC 9.4.0]  # Embedded file name: soal.py  # Compiled at: 2022-11-07 15:36:17  # Size of source mod 2\*\*32: 4776 bytes  import tkinter as tk  from tkinter import \*  from Crypto.Util.number import \*  from tkinter.messagebox import \*  import random  turn = 0  random.seed(0)  def main():  global turn  class Node:  def \_\_init\_\_(self, data):  self.data = data  self.l = None  self.r = None  self.height = 1  class AdelsonVelskiiLandis:  def insert(self, root, key):  if not root:  return Node(key)  if key < root.data:  root.l = self.insert(root.l, key)  else:  root.r = self.insert(root.r, key)  root.height = 1 + max(self.getHeight(root.l), self.getHeight(root.r))  b = self.getBal(root)  if b > 1:  if key < root.l.data:  return self.rRotate(root)  if b < -1:  if key > root.r.data:  return self.lRotate(root)  if b > 1:  if key > root.l.data:  root.l = self.lRotate(root.l)  return self.rRotate(root)  if b < -1:  if key < root.r.data:  root.r = self.rRotate(root.r)  return self.lRotate(root)  return root  def lRotate(self, z):  y = z.r  T2 = y.l  y.l = z  z.r = T2  z.height = 1 + max(self.getHeight(z.l), self.getHeight(z.r))  y.height = 1 + max(self.getHeight(y.l), self.getHeight(y.r))  return y  def rRotate(self, z):  y = z.l  T3 = y.r  y.r = z  z.l = T3  z.height = 1 + max(self.getHeight(z.l), self.getHeight(z.r))  y.height = 1 + max(self.getHeight(y.l), self.getHeight(y.r))  return y  def getHeight(self, root):  if not root:  return 0  return root.height  def getBal(self, root):  if not root:  return 0  return self.getHeight(root.l) - self.getHeight(root.r)  def check(self, state, root, n, x):  state = root  for i in n:  if i == '0':  state = state.l  else:  if i == '1':  state = state.r  if state == None:  showwarning(title='error lur', message='Error invalid node!\nResetting level...')  return False  else:  if state.data == x:  return True  showwarning(title='yah', message='Wrong answer! \nResetting level...')  return False  def decrypt(key, plain):  dec = ''  key = long\_to\_bytes(int(''.join(key), 2))  for i in range(len(key)):  dec += chr(key[i] ^ plain[i])  else:  return dec  def initialization():  tr = AdelsonVelskiiLandis()  root = None  for i in init:  root = tr.insert(root, i)  else:  return (  tr, root)  def submit(root):  global turn  try:  u = inp.get()  if not tr.check(0, root, u, target[turn]):  turn = 0  validatedkey.clear()  else:  showinfo(title='anjay', message='Correct!')  validatedkey.append(u)  inp.delete(0, 'end')  turn += 1  if turn == len(target):  showinfo(title='kelazzz', message=(decrypt(validatedkey, FLAG)))  except SyntaxError:  showerror(title='Error', message='Invalid input!')  else:  text.set(199 - turn)  num = [i for i in range(1, 201)]  init = num.copy()  random.shuffle(init)  FLAG = [70,106,196,124,8,66,39,192,6,86,222,245,244,101,138,58,30,27,51,31,63,175,0,3,25,58,24,225,209,18,7,253,185,174,197,236,7,171,127,126,232,243,65,171,144,237,160,22,105,213,23,12,35,20,105,144,235,96,74,96,37,207,95,111,24,156,0,165,123,211,243,141,213,104,71,106,157,252,198,22,19,73,6,154,31,47,157,200,255,246,161,214,226,97,196,87,61,201,204,192,130,73,143,58,243,190,72,9,131,29,20,89,235,149,143,178,154,47,102,141,11,158,96,28,34,168,62,204,204,74,9,205,209,133,2,58,20,108,206,224,125,223,66,21,143,157,21,203]  validatedkey = []  tr, troot = initialization()  target = init.copy()  target.remove(troot.data)  random.shuffle(target)  def on\_focus\_in(entry):  if entry.cget('state') == 'disabled':  entry.configure(state='normal')  entry.delete(0, 'end')  def on\_focus\_out(entry, placeholder):  if entry.get() == '':  entry.insert(0, placeholder)  entry.configure(state='disabled')  root = tk.Tk()  root.geometry('300x150')  root.title('Kunci Pintu')  root.maxsize(300, 150)  root.minsize(300, 150)  inp = Entry(root, width=33, borderwidth=3, relief=RIDGE)  inp.grid(pady=5, row=0, sticky='w', padx=15)  inp.insert(0, 'Input Secret Key')  inp.configure(state='disabled')  x\_focus\_in = inp.bind('<Button-1>', lambda x: on\_focus\_in(inp)  )  submitbutton = Button(root, text='submit', width=30, command=(lambda: submit(troot)  ), bg='red', fg='white', borderwidth=3, relief=RIDGE)  submitbutton.grid(row=1, sticky='w', padx=15, pady=5)  text = StringVar()  text.set(199 - turn)  textbox = Label(root, textvariable=text, justify='center', width=13)  textbox.config(font=('Courier', 30))  textbox.grid(pady=5, row=2, sticky='w')  root.mainloop()  if \_\_name\_\_ == '\_\_main\_\_':  main()  # okay decompiling 199passcodehahahoho.pyc |
| --- |

Terlihat pada chall bahwa terdapat data structure self balancing binary search tree yang diinisialisasi menggunakan permutasi acak (dengan random.seed(0)) dari 1 sampai 200. Kemudian, pada array target yang merupakan permutasi dari permutasi acak awal, value root dihilangkan, dan kita harus menjawab sejumlah pertanyaan yang akan divalidasi menggunakan fungsi check dari class self balancing BST. Fungsi check sendiri akan melakukan search pada BST berdasarkan input biner yang kita masukkan, dimana “0” berarti search berlanjut ke child kiri, dan “1” berarti search berlanjut ke child kanan. Kemudian, value hasil search tersebut harus sama dengan value target pada tiap iterasi, mulai dari value target pertama. Setelah menjawab pertanyaan dengan benar, kita akan diberikan flag hasil decrypt dari key yang terbentuk selama searching, dan flag encrypted yang hardcoded. Karena random di-seed dengan seed 0, kita sebenarnya dapat merekonstruksi BST, permutasi awal, dan permutasi target.

| # decompyle3 version 3.9.0  # Python bytecode version base 3.8.0 (3413)  # Decompiled from: Python 3.8.10 (default, Jun 22 2022, 20:18:18)  # [GCC 9.4.0]  # Embedded file name: soal.py  # Compiled at: 2022-11-07 15:36:17  # Size of source mod 2\*\*32: 4776 bytes  import tkinter as tk  from tkinter import \*  from Crypto.Util.number import \*  from tkinter.messagebox import \*  import random  turn = 0  random.seed(0)  def main():  global turn  class Node:  def \_\_init\_\_(self, data): #create Node without child  self.data = data  self.l = None  self.r = None  self.height = 1  class AdelsonVelskiiLandis:  def insert(self, root, key):  if not root: #not None = true, create Node on initialization  return Node(key)  if key < root.data:  root.l = self.insert(root.l, key)  else:  root.r = self.insert(root.r, key)  root.height = 1 + max(self.getHeight(root.l), self.getHeight(root.r))  b = self.getBal(root)  if b > 1:  if key < root.l.data:  return self.rRotate(root)  if b < -1:  if key > root.r.data:  return self.lRotate(root)  if b > 1:  if key > root.l.data:  root.l = self.lRotate(root.l)  return self.rRotate(root)  if b < -1:  if key < root.r.data:  root.r = self.rRotate(root.r)  return self.lRotate(root)  return root  def lRotate(self, z):  y = z.r  T2 = y.l  y.l = z  z.r = T2  z.height = 1 + max(self.getHeight(z.l), self.getHeight(z.r))  y.height = 1 + max(self.getHeight(y.l), self.getHeight(y.r))  return y  def rRotate(self, z):  y = z.l  T3 = y.r  y.r = z  z.l = T3  z.height = 1 + max(self.getHeight(z.l), self.getHeight(z.r))  y.height = 1 + max(self.getHeight(y.l), self.getHeight(y.r))  return y  def getHeight(self, root):  if not root:  return 0  return root.height  def getBal(self, root):  if not root:  return 0  return self.getHeight(root.l) - self.getHeight(root.r)  def check(self, state, root, n, x): #self, 0, root, input, turn  state = root  for i in n:  if i == '0':  state = state.l  else:  if i == '1':  state = state.r  if state == None:  return False  if state.data == x:  return True  return False    def decrypt(key, plain):  dec = ''  key = long\_to\_bytes(int(''.join(key), 2))  for i in range(len(key)):  dec += chr(key[i] ^ plain[i])  else:  return dec  def initialization():  tr = AdelsonVelskiiLandis()  root = None  for i in init: #init = random 1-200  root = tr.insert(root, i) # Create BST  else:  return (  tr, root)  def submit(root):  global turn  try:  u = inp.get()  if not tr.check(0, root, u, target[turn]):  turn = 0  validatedkey.clear()  else:  showinfo(title='anjay', message='Correct!')  validatedkey.append(u)  inp.delete(0, 'end')  turn += 1  if turn == len(target):  showinfo(title='kelazzz', message=(decrypt(validatedkey, FLAG)))  except SyntaxError:  showerror(title='Error', message='Invalid input!')  else:  text.set(199 - turn)    def try\_submit(root, guess):  global turn  try:  u = guess  if not tr.check(0, root, u, target[turn]):  return False  else:  validatedkey.append(u)  turn += 1  if turn == len(target):  print((decrypt(validatedkey, FLAG)))  return True  except SyntaxError:  return False  else:  return False  f = open("values.txt", "w+")  num = [i for i in range(1, 201)] #1-200  init = num.copy()  random.shuffle(init) #random 1-200  f.write(",".join([str(i) for i in init]))  f.write("\n")  FLAG = [70,106,196,124,8,66,39,192,6,86,222,245,244,101,138,58,30,27,51,31,63,175,0,3,25,58,24,225,209,18,7,253,185,174,197,236,7,171,127,126,232,243,65,171,144,237,160,22,105,213,23,12,35,20,105,144,235,96,74,96,37,207,95,111,24,156,0,165,123,211,243,141,213,104,71,106,157,252,198,22,19,73,6,154,31,47,157,200,255,246,161,214,226,97,196,87,61,201,204,192,130,73,143,58,243,190,72,9,131,29,20,89,235,149,143,178,154,47,102,141,11,158,96,28,34,168,62,204,204,74,9,205,209,133,2,58,20,108,206,224,125,223,66,21,143,157,21,203]  validatedkey = []  tr, troot = initialization()  target = init.copy()  target.remove(troot.data)  random.shuffle(target)  f.write(",".join([str(i) for i in target]))  ans = []  for t in range(1,200):  #brute turns  i = 0  f = False  print(t)  for i in range(1,15):  for j in range(pow(2, i)):  ask = "{0:b}".format(j).zfill(i)  ans = try\_submit(troot, ask)  if (ans):  break  if (ans):  break  def on\_focus\_in(entry):  if entry.cget('state') == 'disabled':  entry.configure(state='normal')  entry.delete(0, 'end')  def on\_focus\_out(entry, placeholder):  if entry.get() == '':  entry.insert(0, placeholder)  entry.configure(state='disabled')    root = tk.Tk()  root.geometry('300x150')  root.title('Kunci Pintu')  root.maxsize(300, 150)  root.minsize(300, 150)  inp = Entry(root, width=33, borderwidth=3, relief=RIDGE)  inp.grid(pady=5, row=0, sticky='w', padx=15)  inp.insert(0, 'Input Secret Key')  inp.configure(state='disabled')  x\_focus\_in = inp.bind('<Button-1>', lambda x: on\_focus\_in(inp)  )  submitbutton = Button(root, text='submit', width=30, command=(lambda: submit(troot)  ), bg='red', fg='white', borderwidth=3, relief=RIDGE)  submitbutton.grid(row=1, sticky='w', padx=15, pady=5)  text = StringVar()  text.set(199 - turn)  textbox = Label(root, textvariable=text, justify='center', width=13)  textbox.config(font=('Courier', 30))  textbox.grid(pady=5, row=2, sticky='w')  f.close()  root.mainloop()    if \_\_name\_\_ == '\_\_main\_\_':  main()  # okay decompiling 199passcodehahahoho.pyc |
| --- |

Karena kami malas, ketiga hal tersebut langsung dilakukan pada chall hasil dekompilasi, dan untuk tiap iterasi target, kami melakukan bruteforce untuk mencari traversal BST yang dapat menghasilkan nilai target (angry CP noises).



### Flag: NCW22{congratz\_you\_have\_successfully\_decompiled\_the\_pyc\_then\_solving\_the\_tree\_traversal\_problem\_and\_find\_this\_flag\_damn\_you\_really\_deserve\_it\_champ}