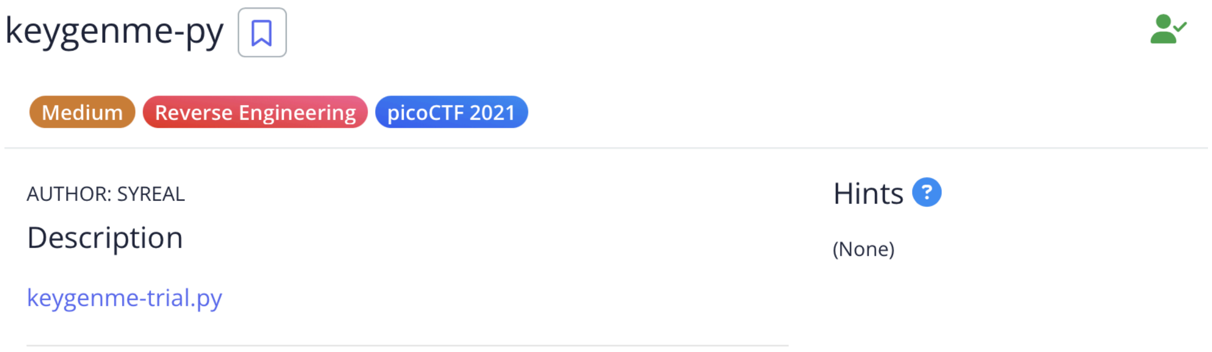


keygenme-py

Medium · Reverse Engineering · picoCTF 2021

AUTHOR: SYREAL

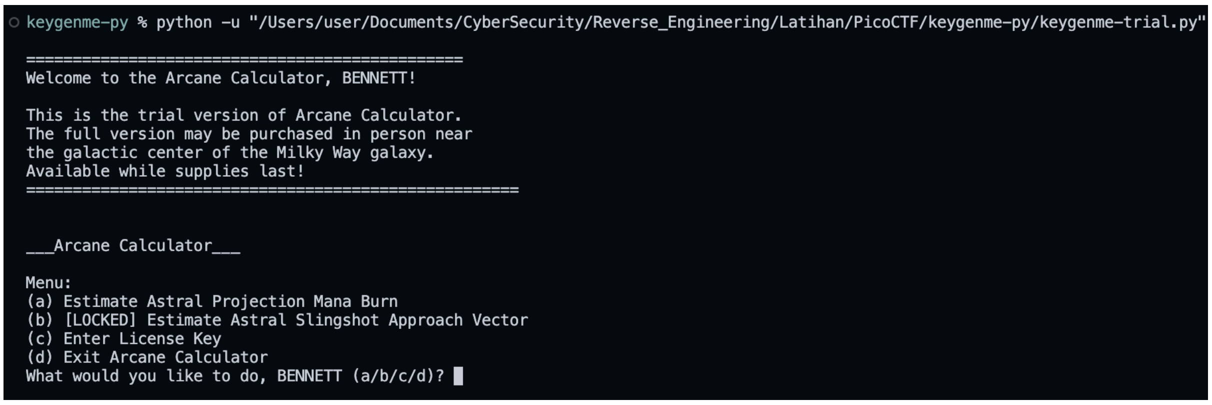
Description

keygenme-trial.py

Hints

(None)

Isi file :



```
keygenme-py % python -u "/Users/user/Documents/CyberSecurity/Reverse_Engineering/Latihan/PicoCTF/keygenme-py/keygenme-trial.py"

===============================================
Welcome to the Arcane Calculator, BENNETT!

This is the trial version of Arcane Calculator.
The full version may be purchased in person near
the galactic center of the Milky Way galaxy.
Available while supplies last!
=====================================================


___Arcane Calculator___

Menu:
(a) Estimate Astral Projection Mana Burn
(b) [LOCKED] Estimate Astral Slingshot Approach Vector
(c) Enter License Key
(d) Exit Arcane Calculator
What would you like to do, BENNETT (a/b/c/d)?
```

Source Code :

```
#============================================================================#
#============================ARCANE CALCULATOR===============================#
#============================================================================#

import hashlib
from cryptography.fernet import Fernet
import base64



# GLOBALS --v
arcane_loop_trial = True
jump_into_full = False
full_version_code = ""
```

```
username_trial = "BENNETT"
bUsername_trial = b"BENNETT"

key_part_static1_trial = "picoCTF{1n_7h3_kk3y_of_"
key_part_dynamic1_trial = "xxxxxxxx"
key_part_static2_trial = "}"
key_full_template_trial = key_part_static1_trial +
key_part_dynamic1_trial + key_part_static2_trial
# key_full_template_trial = "picoCTF{1n_7h3_kk3y_of_08c46aa4}"

star_db_trial = {
 "Alpha Centauri": 4.38,
 "Barnard's Star": 5.95,
 "Luhman 16": 6.57,
 "WISE 0855-0714": 7.17,
 "Wolf 359": 7.78,
 "Lalande 21185": 8.29,
 "UV Ceti": 8.58,
 "Sirius": 8.59,
 "Ross 154": 9.69,
 "Yin Sector CL-Y d127": 9.86,
 "Duamta": 9.88,
 "Ross 248": 10.37,
 "WISE 1506+7027": 10.52,
 "Epsilon Eridani": 10.52,
 "Lacaille 9352": 10.69,
 "Ross 128": 10.94,
 "EZ Aquarii": 11.10,
 "61 Cygni": 11.37,
 "Procyon": 11.41,
 "Struve 2398": 11.64,
 "Groombridge 34": 11.73,
 "Epsilon Indi": 11.80,
 "SPF-LF 1": 11.82,
 "Tau Ceti": 11.94,
 "YZ Ceti": 12.07,
 "WISE 0350-5658": 12.09,
 "Luyten's Star": 12.39,
 "Teegarden's Star": 12.43,
 "Kapteyn's Star": 12.76,
 "Talta": 12.83,
 "Lacaille 8760": 12.88
}


def intro_trial():
    print("\n===============================================\n\
Welcome to the Arcane Calculator, " + username_trial + "!\n")
    print("This is the trial version of Arcane Calculator.")
    print("The full version may be purchased in person near\n\
the galactic center of the Milky Way galaxy. \n\
Available while supplies last!\n\
=====================================================\n\n")


def menu_trial():
    print("___Arcane Calculator___\n\n\
Menu:\n\
(a) Estimate Astral Projection Mana Burn\n\
(b) [LOCKED] Estimate Astral Slingshot Approach Vector\n\
(c) Enter License Key\n\
(d) Exit Arcane Calculator")

    choice = input("What would you like to do, "+ username_trial +"
(a/b/c/d)? ")

    if not validate_choice(choice):
        print("\n\nInvalid choice!\n\n")
        return

    if choice == "a":
        estimate_burn()
    elif choice == "b":
        locked_estimate_vector()
    elif choice == "c":
        enter_license()
    elif choice == "d":
        global arcane_loop_trial
```

```
       arcane_loop_trial = False
       print("Bye!")
   else:
       print("That choice is not valid. Please enter a single, valid \
lowercase letter choice (a/b/c/d).")


def validate_choice(menu_choice):
   if menu_choice == "a" or \
      menu_choice == "b" or \
      menu_choice == "c" or \
      menu_choice == "d":
       return True
   else:
       return False


def estimate_burn(): #choose a
 print("\n\nSOL is detected as your nearest star.")
 target_system = input("To which system do you want to travel? ")

 if target_system in star_db_trial:
     ly = star_db_trial[target_system] # ly = 4.38
     mana_cost_low = ly**2 #4.38 ** 2 = 19.1844
     mana_cost_high = ly**3 #4.38 ** 2 = 84.027672
     print("\n"+ target_system +" will cost between "+
str(mana_cost_low) \
+" and "+ str(mana_cost_high) +" stone(s) to project to\n\n")
 else:
     # TODO : could add option to list known stars
     print("\nStar not found.\n\n")


def locked_estimate_vector(): #choose b
   print("\n\nYou must buy the full version of this software to use
this \
feature!\n\n")
```

```python
def enter_license(): #choose c
    user_key = input("\nEnter your license key: ")
    user_key = user_key.strip()

    global bUsername_trial #b"BENNETT"

    if check_key(user_key, bUsername_trial):
        decrypt_full_version(user_key)
    else:
        print("\nKey is NOT VALID. Check your data entry.\n\n")


def check_key(key, username_trial):

    global key_full_template_trial #key_full_template_trial =
key_part_static1_trial + key_part_dynamic1_trial +
key_part_static2_trial
    if len(key) != len(key_full_template_trial):
        return False
    else:
        # Check static base key part --v
        i = 0
        for c in key_part_static1_trial:
            if key[i] != c:
                return False

            i += 1

        # TODO : test performance on toolbox container
        # Check dynamic part --v
        if key[i] != hashlib.sha256(username_trial).hexdigest()[4]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[5]:
            return False
        else:
            i += 1
        if key[i] != hashlib.sha256(username_trial).hexdigest()[3]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[6]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[2]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[7]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[1]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[8]:
            return False



        return True


def decrypt_full_version(key_str):

    key_base64 = base64.b64encode(key_str.encode())
    f = Fernet(key_base64)
```

```
   try:
       with open("keygenme.py", "w") as fout:
         global full_version
         global full_version_code
         full_version_code = f.decrypt(full_version)
         fout.write(full_version_code.decode())
         global arcane_loop_trial
         arcane_loop_trial = False
         global jump_into_full
         jump_into_full = True
         print("\nFull version written to 'keygenme.py'.\n\n"+ \
                "Exiting trial version...")
   except FileExistsError:
       sys.stderr.write("Full version of keygenme NOT written to disk,
"+ \
                     "ERROR: 'keygenme.py' file already exists.\n\n"+ \
             "ADVICE: If this existing file is not valid, "+ \
             "you may try deleting it and entering the "+ \
             "license key again. Good luck")

def ui_flow():
   intro_trial()
   while arcane_loop_trial:
       menu_trial()



# Encrypted blob of full version
full_version = \
b"""
gAAAAABpRaLynVVsfWihmzH_LerSsmI7O_aFWeZRhgUuxmiqwegf8JxyPBwGgY_wpocHh5B
UxHy7qEmJcAI-BxZQeAZ5Ziiwc4UYl4ZNb4avb1vkPlQvhSVzMek4iFNsw0wuDiMrS_4Sz_
GhOHPFHp9F8OI7rwh8nteVlUfiGvymXFZQ1CdqnEC3rd-8DQ8Dc9-13ohD4QVmwfeYtRNbP
mb3qCzdL20gT064v4PpVgzzQq4o08SSvVDxZEe2ktq1YjEHddm0MMqCn4qxO2x9218HzP2W
aMatvuwFw3nWKOx4XQMV0gBzDVuPunoPJ_WOTQa3zUmI2Fin1Ija3oIIrjU6JyC796_quQk
g3hw4iV_6pERHujMI_k8Vc6tkSWyMaB5daabBKNN_iHRRb_cyM04GjmLTWY347ec8KBhNF7
DiC3VC91y3h-twvMGsbUCmGSbJsUAMiMq0GEv-SXTKcbk2YxYEYcQUDWcsaDeOQ3-P6QklV
j5sTtNq7Gu0r2xVHjRjiO4hgUAlppU7xFhYArFzrFzwSRfYGo1AHXFdkkrIoq5Igz3lOwsw
3KRK6QdkiG2ZiTr7lKqhE7RoBEx38YTx6S-Ttb7OdlGAo-x2AM0aj3uOLgT_JHU6_I-4Ehk
```

```
jmSm3BeA50jrtXpe5A033LK3sdz4b4X4hRXK9b1vQIi5R5Oz01VzLkcXMaTYTPJw7ajV10o
hnckTUKdQ_uYErpDVxi_qh1gAPU0ukO7nYNwD0mISLCaV30kq6EI9JxtrVq6d0nzBjie-Q8
Afw-MhRnnh0NCc4cON1P2dEMeTngPqIZGkE2Q7QhH21ptyYQmZFF0r7OeIigEa-40tDmVZL
k8c9oXkiSPmL63vl8IzfMFsysj6yeSYIZjFgjH77CMuTY322N_IHvCfZyK224iMU99iiOP2
tdy0IqMPgXlXf28dYmFghlc2Wu1Sj10h8hrnk_XBOEDq2NPSaoPWtVOEpB18HyZqRd6dvxC
YV9V4Sb8U5k5N314fnh0X5TYrceghtRyuYecZglPg05XGT5nsHWEt6b6AtHN8GEhzMCR6LU
ODqGohNo2_X2Vr5RRwk5lgbqQe362cy0CPnfNaEg0q3htQNx3Y_YwxfR6pT-S8GEc8g1MpS
FWb41uVxgb91FPwUOmRQWxVfxjkjXldOw_NAAR3ufhZd9sCWlOJR61sZ1t99ol5qb5NK_b-
gDymRf2G1y7jydKb4z-7XYlJWGNkvZ86VgtoUzxjh0ahAfl5rGNtlJcdk4quMhTT0PzVOUX
pSRBHAZOKhcGbawlI8WBNfmhjgqy3LHA0bydUuDvAZqDkjjeIyrvTjCPyXGoJf1F4yDYuIN
O60gCNN5RMikst7NDRiNXpkxczv237Q8LFAUQAz-kq-jRNoEKRe1Y63lezB69BqGDYp7vhc
PWlalGpCNVXzw_1hf8xa7GXX4PYnQ0iPS2obl_Oy6reieoyHd7_4MRClcR1k6tWcI6EDVTN
dI7IULCyY38T_ii2jW0C_PX7y7UKhQdReDdRflfmsrLGPQnRTO7OtcApjQiLMRKMhxLPiqF
Bg9hZ0YpuGcDazMsE0yQ-2yiGzTouzEOpnnNp_VFvLDlTO7ujXnWF0CrrCTXSnZxKYxmnIc
wtDM-65tDEyWkvbEaZTIpaU0nPLMq7y5LDuO8vcStxoDGrSDH-I5rtKhQptvvnHeobwMdat
1PqfpDX5wDhwPt36rkhaRcvCfih3kIpaNf6smxxZfFdQb66e8jkeVYD0VtfwM9H9PY64yJJ
7za1Nt-G9ipEwJCSMUHf8srxKv2uFMVRYiBaacF3UI8UZqOhj1-7KcP7frmAZ8wUxWj7Kjz
e2jrj5g90Fw8ldZalk2ab_7iCpAlnI1Qm8M7qBE4U3wAWY4hKqxxfgVkPkDUzkVAlJQhp79
MCG-Gkrkvx6CmIMkNyWj-jZHboSG8Nm2xz8Zyj_BJAObrOBUrxpiXpOHeJPMI0-2Wfr0Q0v
AlxcRoPgZe5m778cq4EV3zYp6aukbIXn5ud8G_k78m9NPRSyt84I7D5fD-RlXVOvyAxBS4Q
IuMy3QWHfBUf1rsC9JbFnduqaoxnCDLMq2GlSwIzv8IL1TH2MdRR6uEvSw3vQHir_h0Owm3
U6p57cfiSkArq81hhuVZNMzqJYcDn_RaSjGzoY1o_QrVH4oXkoBdn0axa-5DAWFqUQnlHeL
luxzQIpJyfkTPslUfy048v7vaDn9g71WwRtP7M3_nioksJYzSnCozjKa8iXwL6kXwpJwOMK
DGhGym2Y959-bE2Ihk2DzSDObDUEiTJe4zcgl9KWOIYvR5V87u7B-lUf1f2YDd__nnvBf-n
kcIcKY0sqk03IoQVnZPteDjEoDsV0qTIQRBXzt5hd_bhy5uZPVv_WweeU9-vstbkc0h1eJb
Xo5jPanmFqIpX-lWH9oaEfkYOAjDssaYgj9f6VlAez8HynM8m7_PmFNiJmhDd6Y8VmFgnF9
hepYLz0l8zyPXYn1k53oRawI3PEMbNPjeujoH2R9SFErZz6CAKHwVsQmFla6dR2UL8jDhs-
COG6nwmE-QbCBB4ureuAgKxY_CEeSIqnirL6z9lhJHlaoWqdxwlm_jg9w2y4iNhr68gSwof
9Mm62yRc-2nfZTEagxmi2847ENSEuEWWHNgYvSkQjaBgZAQsIMc3WYH9rfIFcbtb3EScgLF
05iKjhQ72ebfL-kEjF_0P6Q1eNWOXtRYXKPHoIYZ990ByUUKBERZPPcbGW7Wl8eA-q_2yXf
iI9M7Os5iAF1ODMwgYJwfj-fHKSkw5MICkeKcnW9NOnXsJeThoCCtsxotltek8QEwlOhBZF
yQRc6zZLS-gjfIJZrj5Txf9G7wS7Sn7Xe4Vfdo5fY3wPJt8mDFaj2kEEhxlcQ8m6fQdfPRb
WcqZDEARRolRgRs5js84z-pgh2GzBLRztSf9bvEuAy7oL3INRRPSUwRWNTzVIfuns3I7O4R
dVzsU7YR2gTXI7GKKYLMg_tje0uzwrVRXDDr34ridJx37yjLRo756u_PMD8WqmsLzaC3WGz
U45cHt0bQL8m9YAeEhHVVf5wEo97Wo9DsfPVozmXu8Vv-yO9t1XnCyBh-PN9-LeX_7R-Qam
Tbqm5IQtVBCO4EQkkwth-8zctCJl5XrKJbY8yTgHsmhlTRkEnyO5MQ0RCKXrLeKuQ55ghAW
rPgEXzgDPsmi3PC_abFXQVjm4tUDHl72X6Gv89IOhXpazS75UNSW2SXl5uikaZgqEIwLYxd
GofPsVExeuWaxwtrmySpoRKT4ljpSNm-fq837arq39wj-2tp7W1n2VvanL466OrcplwMxf-
lYoGxN8uEAXajGEFTHOj2djg6-RyhpRFmWKEjp2rDtK0sVEr8AiIE_eIpTQnwUM9VEkAL-A
```

```
ImbYYqK6s-aAoYqTGYwCPd7Ah7b1ZWgYQ-2WueA0tSdHpUy9ks8apmMNAl97_BYKRkUuhda
l5gzgsI0xIr24slj7COVxlIQt7KF-Z-KED5Y4sKv922K6Ts34HHpB9cHCBgJbug8mSBIB9B
c5gijMbRaQskv_2Bq2b9ifbufapOzeCuVOKgxYoDddb7R5dgqQ_NdzOtA3aWwqh5KMLa1fl
AlIR22nj_TfFUVgnAKGubolLZGwJFFXHYAQneg1WozJPH9geqp6F0AF09sHOE3MfWiKdwbg
dj6pcMm8N59NOsr0qBt5zehp03DVCPwVL2Smnm2RqWQPZke-d9o_WBlYvFRQuusXXrTMQQ4
k7qAJ47fC85ff1q60sT8PWtdSxMQxTxYUQFwGw1EuzutWOU6OFdgLowiFlvxGB73a5RdEWU
d6l25J2qEnmRM9cnP0CFBsWkCM6pIPZzZg9DA6Jg==
"""



# Enter main loop
ui_flow()

if jump_into_full:
   exec(full_version_code)
```

Analisis logika dan algoritma program :

```
username_trial = "BENNETT"
bUsername_trial = b"BENNETT"

key_part_static1_trial = "picoCTF{1n_7h3_kk3y_of_"
key_part_dynamic1_trial = "xxxxxxxx"
key_part_static2_trial = "}"
key_full_template_trial = key_part_static1_trial +
key_part_dynamic1_trial + key_part_static2_trial
```

Program memberikan kode seperti di atas yang merupakan potongan flag yang belum teridentifikasi karena masih ada string “xxxxxxxx” yang belum terdecrypt. Jadi, kita bisa coba cari dimana letak untuk mengidentifikasi string tersebut.

```
def enter_license(): #choose c
   user_key = input("\nEnter your license key: ")
   user_key = user_key.strip()

   global bUsername_trial #b"BENNETT"

   if check_key(user_key, bUsername_trial):
       decrypt_full_version(user_key)
   else:
       print("\nKey is NOT VALID. Check your data entry.\n\n")
```

Jika kita lihat dan scroll ke bawah dari program tersebut, kita bisa lihat terdapat fungsi enter_license() yang meminta kita untuk menginput key dan di dalamnya terdapat validasi key jika benar dia akan mendecrypt full versi dari key user. Jadi kita bisa identifikasi apa isi logika dari fungsi check_key() terlebih dahulu.

```
def check_key(key, username_trial):

         global   key_full_template_trial   #key_full_template_trial   =
key_part_static1_trial        +        key_part_dynamic1_trial        +
key_part_static2_trial
    if len(key) != len(key_full_template_trial):
        return False
    else:
        # Check static base key part --v
        i = 0
        for c in key_part_static1_trial:
            if key[i] != c:
                return False

            i += 1

        # TODO : test performance on toolbox container
        # Check dynamic part --v
        if key[i] != hashlib.sha256(username_trial).hexdigest()[4]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[5]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[3]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[6]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[2]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[7]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[1]:
            return False
        else:
            i += 1

        if key[i] != hashlib.sha256(username_trial).hexdigest()[8]:
            return False



        return True
```

Inilah isi dari fungsi check_key() tersebut. Pertama, fungsi ini memberikan validasi bahwa jika panjang key tidak sama dengan panjang key_full_template_trial maka akan return false sehingga bisa diasumsikan bahwa key panjangnya harus sama dengan key_full_template_trial.

Nah, jika ternyata panjangnya sama, maka akan dicek perchar dari char pertama key yang diinput user terhadap setiap char dari variabel key_part_static1_trial yang berisi string berupa “`picoCTF{1n_7h3_kk3y_of_”` . Sehingga bisa diasumsikan string key input user yang wajib ada adalah string tadi. Setelah semuanya aman dan tidak ada yang tidak sama, maka akan dicek lagi string key input user berikutnya dengan hashlib. dari hashlib tersebut jika kita print hasilnya seperti berikut :

0
8
c
4
6
a
a
4

Otomatis, setelah keynya berupa picoCTF{1n_7h3_kk3y_of_ maka kita bisa sambung menjadi picoCTF{1n_7h3_kk3y_of_08c46aa4

nah sampai disini, kita sudah bisa menyelesaikan karena penutup dari key tersebut adalah “}” sehingga flagnya adalah picoCTF{1n_7h3_kk3y_of_08c46aa4}