Picker III

Medium | Reverse Engineering | picoGym Exclusive | Python

AUTHOR: LT 'SYREAL' JONES

Description

Can you figure out how this program works to get the flag?

Connect to the program with netcat:

```
$ nc saturn.picoctf.net 54785
```

The program's source code can be downloaded here.

This challenge launches an instance on demand.

Its current status is: RUNNING

Instance Time Remaining: 9:12

Restart Instance

Disini, untuk running program harus melalui netcat, tidak bisa langsung dari source code karena jika running program dari SC, maka bisa terjadi error terutama ketika mendapatkan flagnya.

Isi program python sebagai berikut :

```
import re
```

```
USER_ALIVE = True
FUNC_TABLE_SIZE = 4
FUNC_TABLE_ENTRY_SIZE = 32
CORRUPT_MESSAGE = 'Table corrupted. Try entering \'reset\' to fix it'
```

```
func_table = ''
```

```
def reset_table():
 global func_table
```

```
 # This table is formatted for easier viewing, but it is really one line
 func_table = \
'''\
print_table                     \
read_variable                   \
write_variable                  \
getRandomNumber                 \
'''
```

```
def check_table():
 global func_table
```

```
 if( len(func_table) != FUNC_TABLE_ENTRY_SIZE * FUNC_TABLE_SIZE):
   return False
```

```
 return True
```

```
def get_func(n):
 global func_table
```

```
 # Check table for viability
  if(  not  check_table()  ):  #  Kalo  check_table()  true  justru  malah
corrupt massage
   print(CORRUPT_MESSAGE)
   return
```

```
 # Get function name from table
 func_name = ''
 func_name_offset = n * FUNC_TABLE_ENTRY_SIZE
          for          i          in          range(func_name_offset,
func_name_offset+FUNC_TABLE_ENTRY_SIZE):
   if( func_table[i] == ' '):
     func_name = func_table[func_name_offset:i]
     break
```

```
 if( func_name == '' ):
                                                 func_name                =
func_table[func_name_offset:func_name_offset+FUNC_TABLE_ENTRY_SIZE]
  return func_name
```

```
def print_table():
 # Check table for viability
 if( not check_table() ):
   print(CORRUPT_MESSAGE)
   return
```

```
 for i in range(0, FUNC_TABLE_SIZE):
   j = i + 1
   print(str(j)+': ' + get_func(i))
```

```
def filter_var_name(var_name):
 r = re.search('^[a-zA-Z_][a-zA-Z_0-9]*$', var_name)
 if r:
   return True
 else:
   return False
```

```
def read_variable():
 var_name = input('Please enter variable name to read: ')
 if( filter_var_name(var_name) ):
   eval('print('+var_name+')')
```

```
 else:
   print('Illegal variable name')
```

```
def filter_value(value):
 if ';' in value or '(' in value or ')' in value:
   return False
 else:
   return True
```

```
def write_variable():
 var_name = input('Please enter variable name to write: ')
 if( filter_var_name(var_name) ):
   value = input('Please enter new value of variable: ')
   if( filter_value(value) ):
     exec('global '+var_name+'; '+var_name+' = '+value)
   else:
     print('Illegal value')
 else:
   print('Illegal variable name')
```

```
def call_func(n):
 """
 Calls the nth function in the function table.
 Arguments:
   n: The function to call. The first function is 0.
 """
```

```
 # Check table for viability
 if( not check_table() ):
   print(CORRUPT_MESSAGE)
   return
```

```
 # Check n
 if( n < 0 ):
   print('n cannot be less than 0. Aborting...')
   return
 elif( n >= FUNC_TABLE_SIZE ):
    print('n cannot be greater than or equal to the function table size
of '+FUNC_TABLE_SIZE)
   return
```

```
 # Get function name from table
 func_name = get_func(n)
```

```
 # Run the function
 eval(func_name+'()')
```

```
def dummy_func1():
 print('in dummy_func1')
```

```
def dummy_func2():
 print('in dummy_func2')
```

```
def dummy_func3():
 print('in dummy_func3')
```

```
def dummy_func4():
 print('in dummy_func4')
```

```
def getRandomNumber():
 print(4)  # Chosen by fair die roll.
           # Guaranteed to be random.
           # (See XKCD)
```

```
def win():
  # This line will not work locally unless you create your own
'flag.txt' in
 #   the same directory as this script
 flag = open('flag.txt', 'r').read()
 #flag = flag[:-1]
 flag = flag.strip()
 str_flag = ''
 for c in flag:
   str_flag += str(hex(ord(c))) + ' '
 print(str_flag)
```

```
def help_text():
 print(
 '''
This program fixes vulnerabilities in its predecessor by limiting what
functions can be called to a table of predefined functions. This still
puts
the user in charge, but prevents them from calling undesirable
subroutines.
```

```
* Enter 'quit' to quit the program.
* Enter 'help' for this text.
* Enter 'reset' to reset the table.
* Enter '1' to execute the first function in the table.
* Enter '2' to execute the second function in the table.
* Enter '3' to execute the third function in the table.
* Enter '4' to execute the fourth function in the table.
```

```
Here's the current table:
 '''
 )
```

```
print_table()
```

```
reset_table()
```

```
while(USER_ALIVE):
 choice = input('==> ')
 if( choice == 'quit' or choice == 'exit' or choice == 'q' ):
   USER_ALIVE = False
 elif( choice == 'help' or choice == '?' ):
   help_text()
 elif( choice == 'reset' ):
   reset_table()
 elif( choice == '1' ):
   call_func(0)
 elif( choice == '2' ):
   call_func(1)
 elif( choice == '3' ):
   call_func(2)
 elif( choice == '4' ):
   call_func(3)
 else:
   print('Did not understand "'+choice+'" Have you tried "help"?')
```

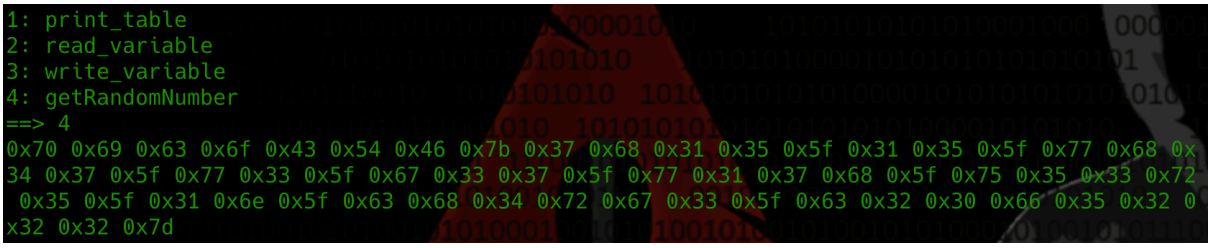
Intinya, pada kode di atas, terdapat sebuah fungsi yang menyimpan flag yaitu fungsi win(). Lalu melalui netcat, kita bisa tahu maksud dari soal ini yaitu untuk menampilkan output dari fungsi win(), tapi sayangnya dalam netcat tersebut tidak ada nomor spesifik untuk menampilkan output win(). Jadi kita bisa lihat sejenak apa perintah dari netcat.

```
~ % nc saturn.picoctf.net 54785
==> help

This program fixes vulnerabilities in its predecessor by limiting what
functions can be called to a table of predefined functions. This still puts
the user in charge, but prevents them from calling undesirable subroutines.

* Enter 'quit' to quit the program.
* Enter 'help' for this text.
* Enter 'reset' to reset the table.
* Enter '1' to execute the first function in the table.
* Enter '2' to execute the second function in the table.
* Enter '3' to execute the third function in the table.
* Enter '4' to execute the fourth function in the table.

Here's the current table:

1: print_table
2: read_variable
3: write_variable
4: getRandomNumber
```

Terdapat 7 pilihan, 4 di antaranya untuk fungsi tabel. Aku terpacu pada perintah nomor 3 yaitu write table sehingga aku coba untuk mengulik.

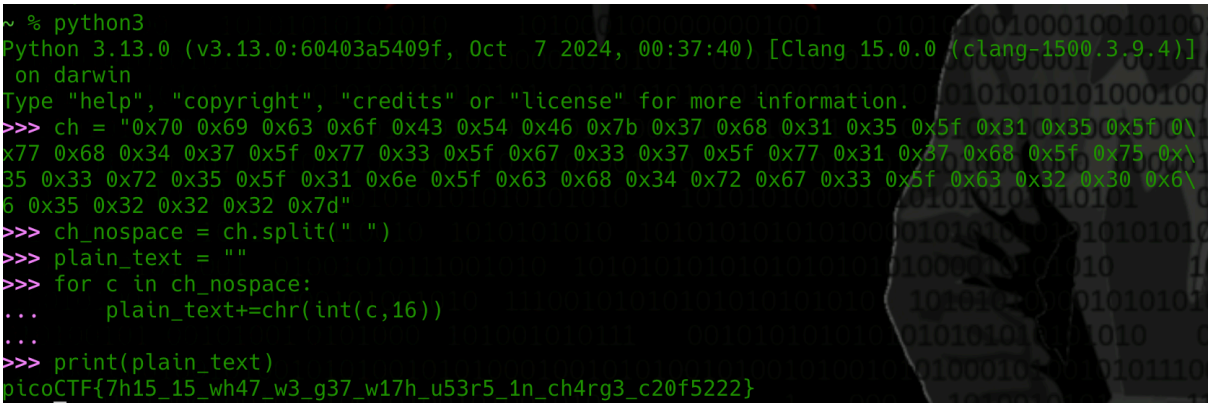
Disini, opsi no 3 adalah untuk mengganti nama variabel pada tabel, langsung saja aku ganti getRandomNumber menjadi win tanpa ‘(‘ dan ‘)’, karena aku lihat pada fungsi filter_value(), jiak terdapat tanda kurung maka akan langsung false. setelah itu aku enter dan aku coba untuk print table (opsi 1) ternyata hasilnya sama saja.

Tanpa basa-basi aku coba pilih langsung opsi ke 4 sehingga menampilkan seperti berikut.



Bisa dilihat bahwa output dari no. 4 merupakan string kumpulan heksadesimal.

Script solver.py



Dari kode diatas, string heksadesimal aku simpan di variabel ch, lalu aku coba split spasinya dan disimpan ke dalam ch_nospace seperti berikut :

ch_nospace = [“0x70”, “0x69”, “.....”, “……”]

setelah itu aku coba inisiasikan plain_text untuk hasilnya nanti.

Masuk ke looping, dimana plain_text nanti akan diisi hasil dari decoding. Decoding ini seperti berikut berdasarkan penjelasan ChatGPT (aku pakai karena malas menjelaskan wkwkwk)

3️⃣ Fokus utama: `int(c, 16)`

📌 Fungsi `int()`

```python
int(value, base)
```

Artinya:

> **Konversi string angka ke integer dengan basis tertentu**

📌 Kenapa pakai `16`?

Karena:

- `0x70` adalah **hexadecimal**
- Hex = **basis 16**

Jadi:

```python
int("0x70", 16)
```

hasilnya:

```python
112
```

Samplenya.
Yah seperti itulah.
Sehingga outputnya adalah flag :

picoCTF{7h15_15_wh47_w3_g37_w17h_u53r5_1n_ch4rg3_c20f5222}