WeirdSnake - picoCTF 2024

Reverse Engineering Challenge

Challenge Information

Challenge Name	WeirdSnake
Category	Reverse Engineering
Difficulty	Medium
Event	picoCTF 2024
Author	Junias Bonou

Description

I have a friend that enjoys coding and he hasn't stopped talking about a snake recently. He left this file on my computer and dares me to uncover a secret phrase from it. Can you assist?

Files Provided: snake

Hints:

- Download and try to reverse the python bytecode
- https://docs.python.org/3/library/dis.html

Solution

TL;DR

1. Analyze Python bytecode disassembly
2. Extract input_list and reconstruct key_str
3. XOR decrypt the ciphertext
4. Retrieve the flag

Initial Analysis

The challenge provides a file called `snake`, which turns out to be Python bytecode disassembly output from the `dis` module.

First, let's check the file type:

```
$ file snake
snake: ASCII text
```

Understanding the Bytecode

The bytecode disassembly reveals the program's logic:

1. Building input_list

The first section loads 40 constants and builds them into a list:

```
LOAD_CONST    0 (4)
LOAD_CONST    1 (54)
...
BUILD_LIST    40
STORE_NAME    0 (input_list)
```

This creates `input_list` containing 40 integers.

2. Constructing key_str

The bytecode builds a string character by character:

```
key_str = 'J'
key_str = '_' + key_str    # '_J'
key_str = key_str + 'o'    # '_Jo'
key_str = key_str + '3'    # '_Jo3'
key_str = 't' + key_str    # 't_Jo3'
```

Final key string: **`t_Jo3`**

3. XOR Encryption Logic

The program performs the following operations:

- Convert `key_str` to a list of ASCII values
- Extend `key_list` by repeating itself until it matches the length of `input_list`
- XOR each element of `input_list` with the corresponding element in `key_list`
- Convert the result to a string

Decryption Script

Based on the bytecode analysis, we can reconstruct the original Python code and reverse the encryption:

```
# Extract input_list from bytecode
input_list = [4, 54, 41, 0, 112, 32, 25, 49, 33, 3,
              0, 0, 57, 32, 108, 23, 48, 4, 9, 70,
              7, 110, 36, 8, 108, 7, 49, 10, 4, 86,
              43, 110, 43, 88, 0, 67, 104, 125, 9, 78]

# Build key_str (following bytecode order)
key_str = 'J'
key_str = '_' + key_str
```

```
key_str = key_str + 'o'
key_str = key_str + '3'
key_str = 't' + key_str

print(f'[*] Key string: {key_str}')

# Convert to list of ord values
key_list = [ord(char) for char in key_str]

# Extend until length matches input_list
while len(key_list) < len(input_list):
    key_list.extend(key_list)

# Truncate to match exact length
key_list = key_list[:len(input_list)]

# XOR operation
result = [a ^ b for a, b in zip(input_list, key_list)]

# Convert to string
result_text = ''.join(map(chr, result))

print(f'\n[+] Result: {result_text}')
```

Output

```
[*] Key string: t_Jo3
[*] Input list length: 40
[*] Key list length: 40

[+] Result: picoCTF{N0t_sO_coNfus1ng_sn@ke_1a73777f}
```

Flag

picoCTF{N0t_sO_coNfus1ng_sn@ke_1a73777f}

Tools Used

- **Python 3** - For bytecode analysis and XOR decryption
- **dis module** - Understanding Python bytecode disassembly

Key Takeaways

- **Python bytecode analysis** - The `dis` module can disassemble Python code into readable bytecode instructions
- **Reverse engineering workflow** - Trace bytecode instructions step-by-step to understand program logic
- **XOR encryption** - XOR is reversible: if `a ^ b = c`, then `c ^ b = a`
- **Key repetition pattern** - When key is shorter than plaintext, it repeats to match the length

References

- Python dis module documentation: https://docs.python.org/3/library/dis.html
- XOR cipher: https://en.wikipedia.org/wiki/XOR_cipher

Author

Glenvio Regalito Rahardjo

Solved: December 2024