PicoCTF Writeup: Quantum Entanglement Cipher

John Gahagan

Challenge Description

Name: Quantum Entanglement Cipher
Prompt: We invented a new cypher that uses *"quantum entanglement"* to encode the flag. Connect to the program with netcat:

```
$ nc verbal-sleep.picoctf.net 60759
```

The program's source code is provided.

Given Code

We are given the following Python script:

```
import sys

def exit():
  sys.exit(0)

def scramble(L):
  A = L
  i = 2
  while (i < len(A)):
    A[i-2] += A.pop(i-1)
    A[i-1].append(A[:i-2])
    i += 1

  return L

def get_flag():
  flag = open('flag.txt', 'r').read()
  flag = flag.strip()
  hex_flag = []
  for c in flag:
    hex_flag.append([str(hex(ord(c)))])

  return hex_flag

def main():
  flag = get_flag()
```

```
  cypher = scramble(flag)
  print(cypher)

if __name__ == '__main__':
  main()
```

How the Cipher Works

The cipher encodes the flag as a nested and mutated list structure. Here's the process:

1. ‘get$_flag()‘reads‘flag.txt‘, stripsit, andencodeseachcharacterintoahexstring, wrappingeachinaone-elementlist.‘scramble()‘thenperforms$:
2. 3. destructive in-place mutation by combining and popping items;
   4. nested appends, causing entangled list structures.

Finally, the scrambled structure is printed.

Downloading the Encrypted Flag

First, I used Netcat to capture the raw encoded output of the cipher into a file:

```
ncat verbal-sleep.picoctf.net 60759 > flag.txt
```

Note: Use ‘ncat‘ (from Nmap) on Windows, as ‘nc‘ may not be recognized.

Decryption Script

To reverse the structure and extract the hex values, I wrote the following Python script:

```
import ast
import sys

def extract_outer_list(array):
    # Flatten outer list to only hex string elements
    return [item for inner in array for item in inner if isinstance(item, str)]

def get_flag():
    try:
        with open('flag.txt', 'rb') as file:
            raw = file.read()
    except FileNotFoundError:
        print("flag.txt␣not␣found.")
        sys.exit(1)

    text = raw.replace(b'\x00', b'').decode(errors='ignore')

    try:
        list_string = ast.literal_eval(text)
```

```
    except Exception as e:
        print("Error␣parsing␣flag␣list:", e)
        sys.exit(1)

    flag = extract_outer_list(list_string)

    try:
        hex_flag = [chr(int(c, 16)) for c in flag]
    except Exception as e:
        print("Error␣decoding␣hex␣values:", e)
        sys.exit(1)

    return hex_flag

def main():
    flag = get_flag()
    print("Decoded␣flag:")
    print(''.join(flag))

if __name__ == '__main__':
    main()
```

Output

After running the script, the correct flag was decoded and printed:

```
Decoded flag:
picoCTF{example_flag_here}
```

Takeaways

This challenge was a fun twist on list manipulation and hex decoding. Key lessons included:

- Understanding Python list mutation and references
- Safely using `ast.literal_eval()` to parse structured text
- Using netcat/ncat effectively to dump data from remote servers