Writeup: Reto de Reversing "Pajarito"

Análisis inicial

El reto comienza con un ejecutable llamado "pajarito" y una secuencia de números con la pista: "Un pajarito me ha contado algo sobre ti... Pero no se descifrarlo... Qué querrá decir ese pajarito????????"

```
431 510 670 730 430 140 070 450 410 520 640 450 310 740 640 500 270 760 601 370 320 140 550 260 700 450
370 530 370 611 350 450 150 350 540 070 310 370 660 300 010 340 360 760 510 640 140 250 250 060 350 230
710 201 070 760 521 760 640 600 700 750 540 000 110 450 750 370 220 721 650 000 120 230 070 040 700 140
000 200 500 230 660 060 210 101 330 200 200 750 040 370 410 170 670 600 600 231 050 330 610 240 250 500
500 070 660 600 520 770 121 300 130 340 460 550 300 570 160 500 220 440 531 340 500 570 540 050 720 370
150 350 420 360 540 370 110 440 060 050 001 621 400 000 600 760 050 530 610 240 270 500 230 750 030 100
500 760 270 370 420 340 440 370 401 150 140 550 720 050 760 340 130 140 560 200 500 230 010 000 100 130
710 070 110 131 230 200 501 231 670 670 021 450 200 330 500 560 670 370 620 750 010 000 100 130 710 070
230 740 060 500 220 721 670 100 600 520 450 340 500 540 631 200 321 750 260 370 420 740 410 300 200 760
401 070 000 170 650 550 510 450 740 060 230 160 250 500 500 070 660 600 520 760 160 070 530 031 060 000
```

Análisis del código decompilado

Tras examinar el ejecutable, obtenemos el siguiente código decompilado:

- **Función main**:
    - Solicita un secreto al usuario: "Cuentame tus secretos, sin miedo: "
    - Lee la entrada con fgets y elimina el salto de línea final
    - Llama a xor_encode con la entrada y un buffer de salida
    - Llama a reverse_string con el buffer resultado
    - Imprime el contenido final: "El pajarito me dijo %s\n"
- **Función xor_encode**:
    - Toma la entrada y un buffer de salida
    - Define la clave: "awanabumbambamwiyobadiou" (longitud 24)

- Para cada carácter de la entrada:
    - Aplica XOR con el carácter correspondiente de la clave (cíclicamente)
    - Formatea el resultado como número octal de 3 dígitos (%03o)
    - Añade al buffer de salida seguido de un espacio

- **Función reverse_string**:
    - Invierte completamente la cadena de entrada

Proceso de decodificación

Para recuperar el mensaje original, debemos invertir el proceso:

1. **Revertir la inversión de la cadena completa**:

```
encoded_reversed_string = "431 510 670 730 430 140 070 450 410 520 640 450 310 740 640 500 270 760 601 370 320 140 550 260 700 450 370 530 370 611 350 450 150 350 540 070 310 370 660 300 010 340 360 760 510 640 140 250 250 060 350 230 710 201 070 760 521 760 640 600 700 750 540 000 110 450 750 370 220 721 650 000 120 230 070 040 700 140 000 200 500 230 660 060 210 101 330 200 200 750 040 370 410 170 670 600 600 231 050 330 610 240 250 500 500 070 660 600 520 770 121 300 130 340 460 550 300 570 160 500 220 440 531 340 500 570 540 050 720 370 150 350 420 360 540 370 110 440 060 050 001 621 400 000 600 760 050 530 610 240 270 500 230 750 030 100 500 760 270 370 420 340 440 370 401 150 140 550 720 050 760 340 130 140 560 200 500 230 010 000 100 130 710 070 110 131 230 200 501 231 670 670 021 450 200 330 500 560 670 370 620 750 010 000 100 130 710 070 230 740 060 500 220 721 670 100 600 520 450 340 500 540 631 200 321 750 260 370 420 740 410 300 200 760 401 070 000 170 650 550 510 450 740 060 230 160 250 500 500 070 660 600 520 760 160 070 530 031 060 000"

# Revertir la cadena completa
encoded_string_unreversed = encoded_reversed_string[::-1]
```

2. **Decodificar la operación XOR**:

```
key = "awanabumbambamwiyobadiou"
key_len = len(key)

# Dividir la cadena revertida en números octales
octal_numbers = encoded_string_unreversed.strip().split()
```

```
# Decodificación XOR
decoded_chars = []
for i, octal_str in enumerate(octal_numbers):
    try:
        # Convertir la cadena octal a un entero (byte)
        xor_result_byte = int(octal_str, 8)

        # Obtener el byte correspondiente de la clave
        key_byte = ord(key[i % key_len])

        # Realizar XOR para obtener el byte original
        original_byte = xor_result_byte ^ key_byte

        # Convertir el byte original a un carácter
        decoded_chars.append(chr(original_byte))
    except ValueError:
        print(f"Error: La cadena '{octal_str}' en el índice {i} no es un número octal válido.")
        continue
    except Exception as e:
        print(f"Error inesperado procesando '{octal_str}' en índice {i}: {e}")
        continue

# Unir todos los caracteres decodificados
decoded_result = "".join(decoded_chars)
```

Al ejecutar este código, obtenemos una cadena codificada en Base64:

```
aG9sYSBxdWUgdGFsIHNlIGVuY3VlbnRyYSB1c3RlZCwgeW8gdGFtYmllbiBtZSBlbmN1ZW50cm8gYmllbiwgcXVpZXJvIHF1ZSBzZXBhcyBxdWUgdGUgdm95IGEgZGFyIHVuIHJlZ2FsaXRvLCBwb3JxdWUgdGUgbG8gbWVyZWNlcy4gRWwgcmVnYWxpdG8gZXMgaGZjdGZ7VU5fcDRKNHIxVDBfbTNfZDFKMF9RdTNfVDNfZ1VTdDRfNGxnMUVufQo=
```

3. **Decodificar la cadena Base64**:

```
import base64

base64_encoded_string =
"aG9sYSBxdWUgdGFsIHNlIGVuY3VlbnRyYSB1c3RlZCwgeW8gdGFtYmllbiBtZSBlbmN1ZW50cm8gYmllbiwgcXVpZXJvIHF1ZSBzZXBhcyBxdWUgdGUgdm95IGEgZGFyIHVuIHJlZ2FsaXRvLCBwb3JxdWUgdGUgbG8gbWVyZWNlcy4gRWwgcmVnYWxpdG8gZXMgaGZjdGZ7VU5fcDRKNHIxVDBfbTNfZDFKMF9RdTNfVDNfZ1VTdDRfNGxnMUVufQo=
```

```
wgcmVnYWxpdG8gZXMgaGZjdGZ7VU5fcDRKNHIxVDBfbTNfZDFKMF9RdTNfVDNfZ1VTdDRfNGxnMUVufQo="

decoded_bytes = base64.b64decode(base64_encoded_string)
decoded_string = decoded_bytes.decode('utf-8')

print(decoded_string)
```

Resultado final

Al decodificar la cadena Base64, obtenemos el siguiente mensaje:

```
hola que tal se encuentra usted, yo tambien me encuentro bien, quiero que sepas que te voy a dar un regalito, porque te lo mereces. El regalito es hfctf{UN_p4J4r1T0_m3_d1J0_Qu3_T3_gUSt4_4lg1En}
```

La flag es: `hfctf{UN_p4J4r1T0_m3_d1J0_Qu3_T3_gUSt4_4lg1En}`