After connecting to the remote server, we notice 3 prompts: 2 prompots for “starting points” and 1 prompt for password. After a bit of messing around, we notice the “starting points” prompots leak a png and the maximum leak at one time is 10. We need to make a program to recover the png which probably contains the password needed, or in any case crucial information.

After a few attempts at programming, some observations need to be made:

1) For some reason, the only way to recover bytes is to ask for them one-by-one. Asking for more causes the server to skip some of them, for some reason.

2) The response is almost always in this format: \x89, byte, /r/n. This means we need to slice the response received to uncover the byte we need

3) After a failed attempt, I noticed sometime the response doesn’t contain the byte mentioned at 2). After looking at a normal png via hexedit, I saw that when the server wants to send a LF (line feed) or byte 0A, it doesn’t. This means we need to insert it ourseleves if the byte is missing.

4) Since it takes a while to get the bytes, I implemented threading. I noticed for high values the server seems to crash or something, so I kept 5 threads. After some testing, this makes the program significantly faster, so it’s a good practice to use threads in cases like this.

The final code is as follows:

```
from pwn import *
from concurrent.futures import ThreadPoolExecutor
import threading

host = "35.246.152.131"
port = 31427

output_file = "a.txt"
file_lock = threading.Lock()

def handle_connection(n):
    context.log_level = 'error'
    conn = remote(host, port)
    conn.recvuntil(b": ")
    conn.sendline(str(n).encode())
    conn.recvuntil(b": ")
    conn.sendline(str(n + 1).encode())
    conn.recvline()
    response3 = conn.recvuntil(b"\n")[1:-2]
    conn.close()
    return response3

def main():
    with open(output_file, "w+b") as file:
        file.write(b'\x89')
```

```
        n = 1
        with ThreadPoolExecutor(max_workers=5) as executor:
            while n <= 10000:
                futures = {executor.submit(handle_connection, i): i for i in
range(n, n + 5)}
                for future in futures:
                    response3 = future.result()
                    if not response3:
                        response3 = b'\x0A'
                    with file_lock:
                        if file.tell() >= 4:
                            file.seek(-4, 2)
                            last_four_bytes = file.read(4)
                            if last_four_bytes == b'IEND':
                                print("IEND chunk received. Exiting...")
                                return
                        file.write(response3)
                n += 5

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

The image we recover is a qr containing the password we needed to recover the flag.

Made with love by: AndreiCat