Analyzing the executable reveals what we need to look for in the pcapng file: a GET /key and GET a.gif.

We can find these in TCP stream 43, respectively 78.

All we need to do is extract them and xor them, as the executable would normally do.

```python
import subprocess
import binascii

def extract_tcp_stream(stream_number):
    result = subprocess.run(
        ["tshark", "-r", "chall.pcapng", "-q", "-z",
f"follow,tcp,raw,{stream_number}"],
        capture_output=True, text=True, check=True
    )
    raw_data = "".join(result.stdout.splitlines()[5:])
    raw_data = ''.join(c for c in raw_data if c in "0123456789ABCDEFabcdef") # my
coding skills suck so this is a quickfix that works :)

    if len(raw_data) % 2 != 0:
        raw_data = '0' + raw_data  # binascii.unhexlify sucks

    return binascii.unhexlify(raw_data)

def xor_decrypt(data, key):
    return bytes(data[i] ^ key[i % len(key)] for i in range(len(data)))

def main():
    key_data = extract_tcp_stream(43)
    gif_data = extract_tcp_stream(78)

    key_data = key_data.replace(b"GET /key\r\n\r\n", b"")
    gif_data = gif_data.replace(b"GET /a.gif\r\n\r\n", b"")

    decrypted_data = xor_decrypt(gif_data, key_data)

    with open("decoded.gif", "wb") as f:
        f.write(decrypted_data)
    print("Decoded file saved as decoded.gif")

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

The original gif isn't actually a gif, it's the flag. A little bit of a bummer but challenge solved so who cares. Because I definitely didn't assume for too long that the program didn't work and gave me a corrupted gif. Definitely.

Made with love by: AndreiCat