

# Looking through Wireshark at conversations, we can see a lot of TCP streams ( 172 ). I ran the following shell script to export all the TCP outputs in a single file ( **tcp.sh** )

```
for stream in $(tshark -r chall.pcapng -Y "tcp.stream" -T fields -e tcp.stream | sort -n |
uniq); do
    echo "=== TCP Stream $stream ==="
    tshark -r chall.pcapng -q -z follow,tcp,ascii,$stream
    echo
done
```

```
chmod +x tcp.sh
./tcp.sh > file.txt
```

# And I found nothing special. I started to inspect the second file ( **a.out** ) and I found out that it is an executable. And because I cannot run it, I decompiled it with an online tool and I found some hints. I found 2 “files” and a XOR function :

☐ angr ☒ BinaryNinja ☐ Boomerang ☐ dewolf ☒ Ghidra ☐ Hex-Rays

## BinaryNinja C

5.1.8005 (9933ffc8)

```
58
59     int32_t __fd = _socket_connect(arg2[1], _atoi(arg2[2]));
60     int64_t var_48 = _write(__fd, "GET /key\r\n\r\n", 0xc);
61     _memset(&_buffer, 0, 0x400);
62
63     while (_read(__fd, &_buffer, 0x3ff))
64     {
65         int64_t var_58_1 = __strcpy_chk(&_key, &_buffer, 0x64);
66         uint64_t rax_10;
67         rax_10 = 0;
68         int32_t var_64_1 = _fprintf(*__stderrp, "%s %d\n", &_key, _strlen(&_key));
69         _memset(&_buffer, 0, 0x400);
70     }
71
72     int32_t var_68 = _shutdown(__fd, 2);
73     int32_t var_74 = _close(__fd);
74     int32_t __fd_1 = _socket_connect(arg2[1], _atoi(arg2[2]));
75     int64_t var_80 = _write(__fd_1, "GET /a.gif\r\n\r\n", 0xe);
76     _memset(&_buffer, 0, 0x400);
77     FILE* __stream = _fopen("a.gif", "wb");
78
79     while (_read(__fd_1, &_buffer, 0x3ff))
80     {
81         int64_t (* const rax_17)();
82
```

```
while (var_30 < _strlen(&_key))
{
    void* rax_23;
    rax_23 = 0;
    _printf("%c", *(&_key + var_30) ^ *(&_test + var_30));
    var_30 += 1;
}
```

# Looking through **file.txt**, I found the 2 files ne needed ( stream 43 and 78 ) :

```
=====
=== TCP Stream 43 ===
=====
Follow: tcp.ascii
Filter: tcp.stream eq 43
Node 0: 192.168.1.12:50587
Node 1: 161.35.16.97:31337
12
GET /key

70
AAAAAA85783cc847fb84e7ba9c1c727099c9040fe086fab96857227bedc4b3967ec978
=====
```

```
=====
=== TCP Stream 78 ===
=====
Follow: tcp.ascii
Filter: tcp.stream eq 78
Node 0: 192.168.1.12:50622
Node 1: 161.35.16.97:31337
14
GET /a.gif

70
....:X.S.Y
W....U...S.SX
TT.S.Q..
W
RP...SY..UTZR[.WPQRU..
.UZ..PR[TE
```

# Now I copied the content as **raw** ( hex ) and I wrote a Python script to use the key to XOR the GIF. You cannot open the GIF because it's just text. So, when the script ends, just cat the gif. And I forgot, remove the **AAAAAA** from the key

# The Python code :

```
def hex_to_bytes(hex_string):
    return bytes.fromhex(hex_string)

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

def main():
    key_data =
hex_to_bytes("4141414141413835373833636338343766623834653762613963316337323730
3939633930343066653038366661623936383537323237626564633462333936376563393738")
```

```
gif_data =  
hex_to_bytes("040212023a78085300590a570608040f55030101530053580d54540253015  
109080d570d52500607075359000555545a525b040f575051525500050d01555a030f50525  
b5445")  
  
decrypted_data = xor_decrypt(gif_data, key_data)  
  
with open("decoded.gif", "wb") as f:  
    f.write(decrypted_data)  
    print("finish")  
  
if __name__ == "__main__":  
    main()
```

THE FLAG :

ECSC{90f7a94e0083a95671947ead3f91444bd6abca6c46cdc18ebf00df9cfc5851bc}  
~Z4que