Phân tích ngắn gọn về giao thức truyền thông Mitsubishi PLC MELSOFT

CISRC Xuất bản vào ngày 2022-07-19 14:47:28

1. Khái quát chung

Giao thức Mitsubishi MELSOFT là giao thức cấu hình riêng của Mitsubishi PLC, được sử dụng để

liên lạc giữa phần mềm lập trình và PLC Mitsubishi. Không có nhiều thông tin công khai về giao thức này, lần này chúng tôi chủ yếu phân tích giao thức và tiến hành kiểm tra fuzz giao thức MELSOFT trên PLC Mitsubishi dựa trên kết quả phân tích.

2. Cấu hình môi trường

GX WORKS2: Phần mềm lập trình PLC của Mitsubishi, phù hợp với Q, QnU, L, FX và các dòng bộ điều khiển khả trình khác và hỗ trợ mô phỏng PLC.

L61P-CM: CPU dòng L06 sử dụng cổng TCP/5007 hoặc UDP/5008 để giao tiếp với phần mềm lập trình theo mặc định. Giao thức truyền thông là giao thức độc quyền của Mitsubishi melsoft.

Công cụ đảo ngược: IDA 7.5

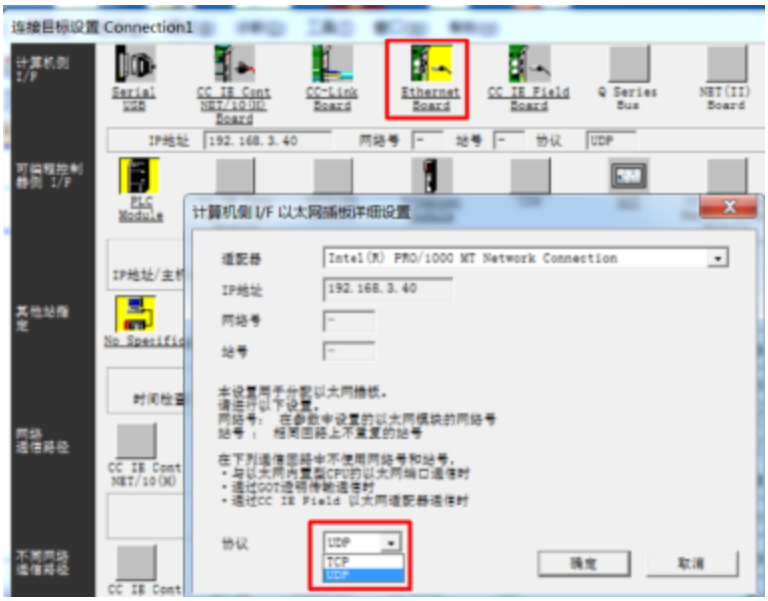
Công cụ gỡ lỗi: x64dbg

3. Quy trình phân tích

3.1 Thu thập và phân tích thông tin

3.1.1 Cổng giao tiếp

Theo mặc định, GX WORKS2 sử dụng chế độ phát sóng UDP để giao tiếp với cổng PLC UDP 5008 cho giao thức cấu hình.Trên GX WORKS2, bạn có thể định cấu hình mục tiêu kết nối để sử dụng chế độ TCP để giao tiếp với cổng PLC TCP 5007 cho giao thức cấu hình.



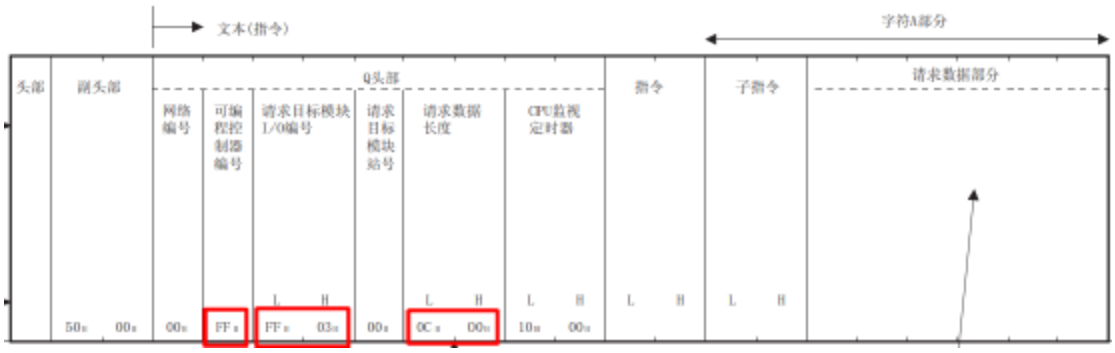
1	0.000000	192.168.3.43	255.255.255.255	UDP	46	52892 → 5008 Len=4
2	0.001041	192.168.3.39	255.255.255.255	UDP	60	5008 → 52892 Len=14
3	0.368583	192.168.3.43	255.255.255.255	UDP	46	52893 → 5008 Len=4
4	0.369528	192.168.3.39	255.255.255.255	UDP	60	5008 → 52893 Len=14
5	0.398421	192.168.3.43	255.255.255.255	UDP	46	52894 → 5008 Len=4
6	0.400137	192.168.3.39	255.255.255.255	UDP	70	5008 → 52894 Len=28
7	0.430959	192.168.3.43	255.255.255.255	UDP	83	52895 → 5008 Len=41
8	0.433386	192.168.3.39	255.255.255.255	UDP	117	5008 → 52895 Len=75

13	2.558653	192.168.3.43	192.168.3.39	TCP	66	49164 → 5007 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
15	2.560192	192.168.3.39	192.168.3.43	TCP	60	5007 → 49164 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
16	2.560456	192.168.3.43	192.168.3.39	TCP	54	49164 → 5007 [ACK] Seq=1 Ack=1 Win=64240 Len=0
18	2.562100	192.168.3.43	192.168.3.39	TCP	58	49164 → 5007 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=4
20	2.564120	192.168.3.39	192.168.3.43	TCP	82	5007 → 49164 [PSH, ACK] Seq=1 Ack=5 Win=5840 Len=28
21	2.566195	192.168.3.43	192.168.3.39	TCP	95	49164 → 5007 [PSH, ACK] Seq=5 Ack=29 Win=64212 Len=41
23	2.568964	192.168.3.39	192.168.3.43	TCP	129	5007 → 49164 [PSH, ACK] Seq=29 Ack=46 Win=5840 Len=75

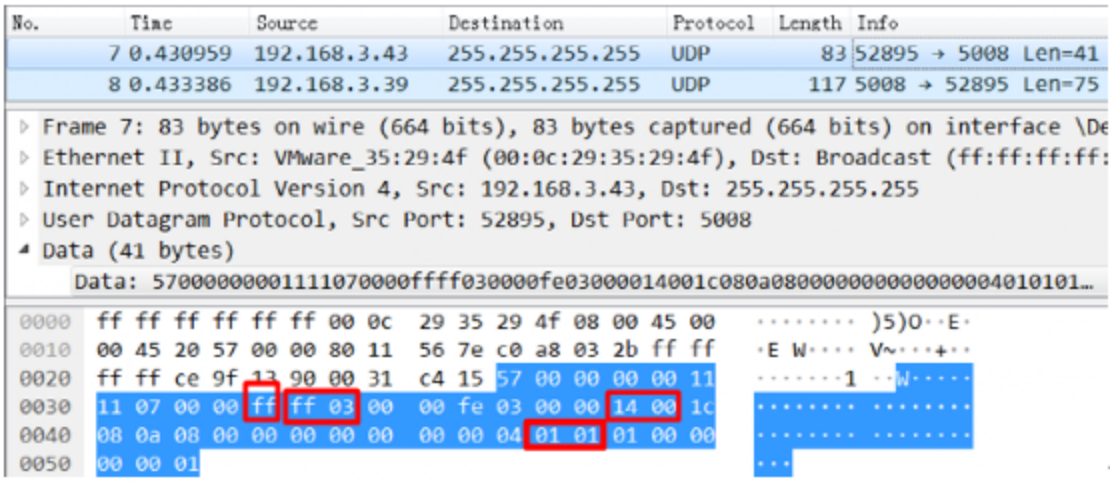
3.1.2 Tương tự giao thức

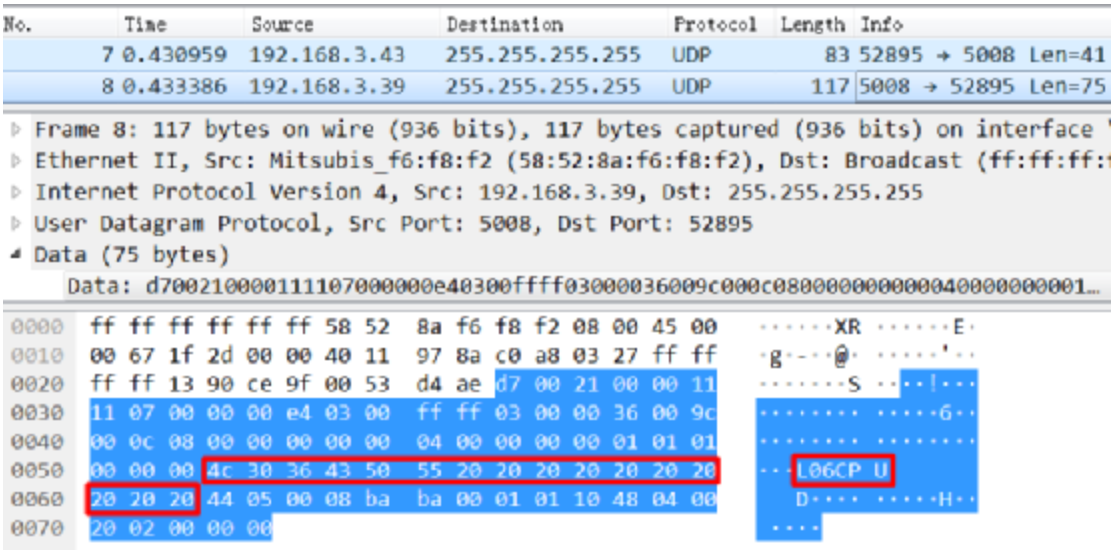
Sau một số thông tin trên Internet và phân tích lưu lượng PLC thực, người ta thấy rằng giao thức MELSOFT có một số điểm tương đồng với giao

thức Mitsubishi MC (giao thức Mitsubishi MC là giao thức công cộng và có thể tải xuống hướng dẫn tham khảo giao thức truyền thông trực tiếp từ Mitsubishi MC). Trang web chính thức). Ví dụ: định dạng tin nhắn (khung giao thức MC 3E), chức năng "đọc mô hình CPU".



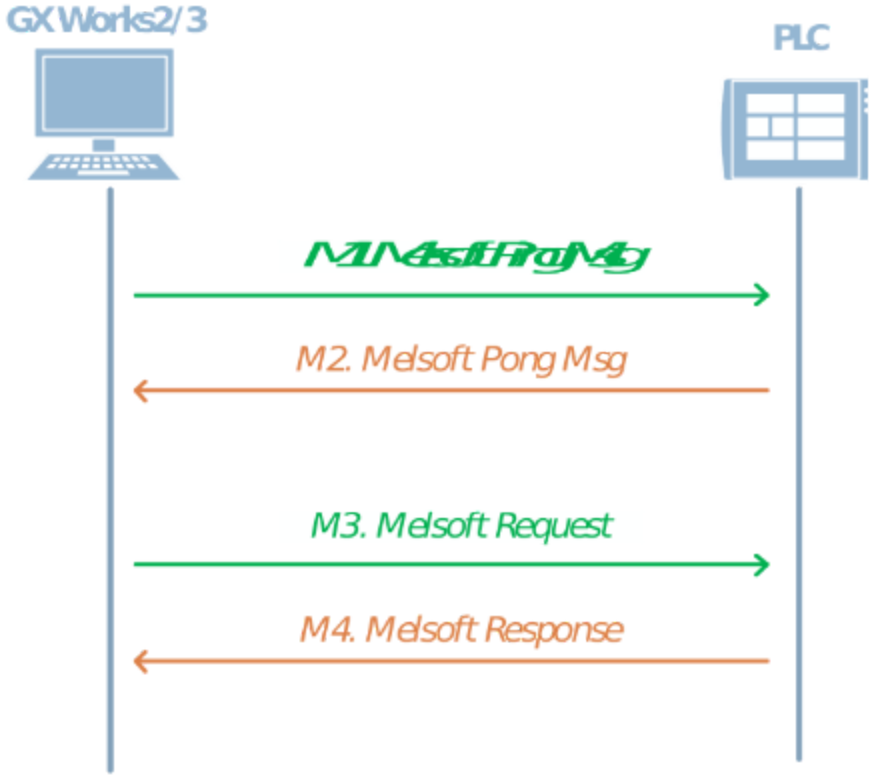
功能	指令(*1)(子指令)	处理内容
远程 RUN	1001 (0000)	进行远程 RUN(执行运算)请求。
远程 STOP	1002 (0000)	进行远程 STOP(停止运算)请求。
远程 PAUSE	1003 (0000)	进行远程 PAUSE(停止运算)请求。(保持输出状态)
远程锁存清除	1005 (0000)	STOP 状态时，进行远程锁存清除(软元件存储器的清除)请求。
远程 RESET	1006 (0000)	STOP 状态时，进行远程 RESET(开始执行运算)请求。
CPU 型号读取	0101 (0000)	进行可编程控制器 CPU 的型号读取请求。





3.1.3 Bố trí quá trình tương tác giao thức

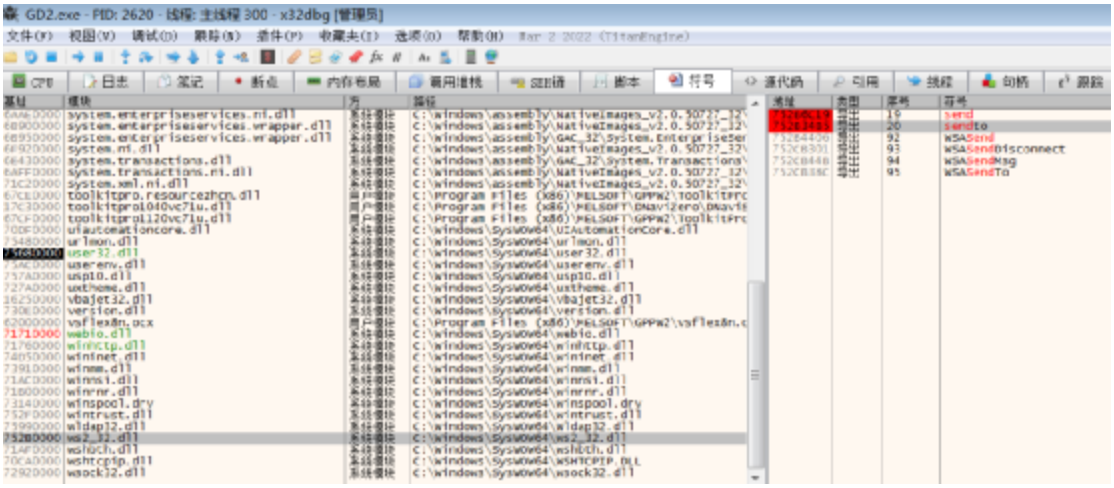
Dựa trên thông tin thu thập được từ Internet, quá trình tương tác của giao thức bước đầu được sắp xếp như sau:

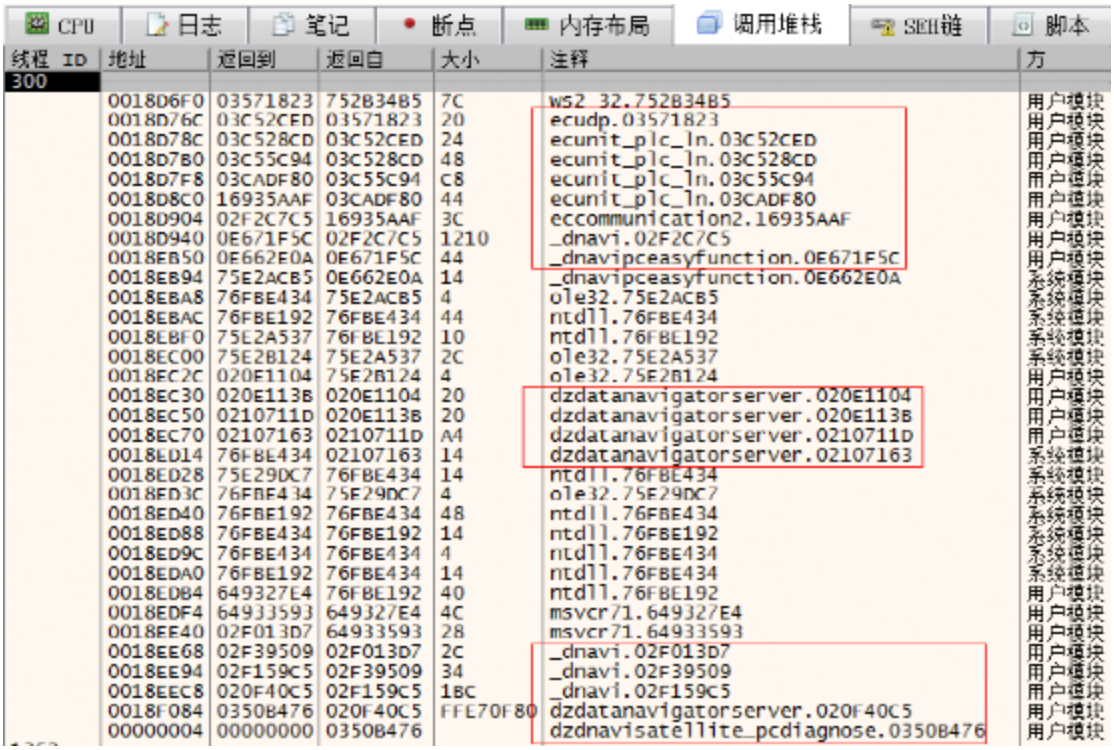


3.2 Gỡ lỗi động Phần mềm lập trình GX Works2

Sau khi GX WORKS2 đang chạy, hãy sử dụng x64dbg để đính kèm vào chương trình và đặt điểm ngắt trong chức năng gửi/sendto của "ws2_32.dll". Thực hiện các hoạt động như "đọc dữ liệu" và "thao tác từ xa" trong GX WORKS2,

xem ngăn xếp cuộc gọi và phân tích các mối quan hệ gọi chức năng.





线程 ID	地址	返回到	返回自	大小	注释	方
300						
	0018D6F0	03571823	752B34B5	7C	ws2_32.752B34B5	用户模块
	0018D76C	03C52CED	03571823	20	ecudp.03571823	用户模块
	0018D78C	03C528CD	03C52CED	24	ecunit_plc_ln.03C52CED	用户模块
	0018D7B0	03C55C94	03C528CD	48	ecunit_plc_ln.03C528CD	用户模块
	0018D7F8	03CADF80	03C55C94	C8	ecunit_plc_ln.03C55C94	用户模块
	0018D8C0	16935AAF	03CADF80	44	ecunit_plc_ln.03CADF80	用户模块
	0018D904	02F2C7C5	16935AAF	3C	eccommunication2.16935AAF	用户模块
	0018D940	0E671F5C	02F2C7C5	1210	_dnavi.02F2C7C5	用户模块
	0018EB50	0E662E0A	0E671F5C	44	_dnavipceasyfunction.0E671F5C	用户模块
	0018EB94	75E2ACB5	0E662E0A	14	_dnavipceasyfunction.0E662E0A	系统模块
	0018EBA8	76FBE434	75E2ACB5	4	ole32.75E2ACB5	系统模块
	0018EBAC	76FBE192	76FBE434	44	ntdll.76FBE434	系统模块
	0018EBF0	75E2A537	76FBE192	10	ntdll.76FBE192	系统模块
	0018EC00	75E2B124	75E2A537	2C	ole32.75E2A537	系统模块
	0018EC2C	020E1104	75E2B124	4	ole32.75E2B124	用户模块
	0018EC30	020E113B	020E1104	20	dzdatanavigatorserver.020E1104	用户模块
	0018EC50	0210711D	020E113B	20	dzdatanavigatorserver.020E113B	用户模块
	0018EC70	02107163	0210711D	A4	dzdatanavigatorserver.0210711D	用户模块
	0018ED14	76FBE434	02107163	14	dzdatanavigatorserver.02107163	系统模块
	0018ED28	75E29DC7	76FBE434	14	ntdll.76FBE434	系统模块
	0018ED3C	76FBE434	75E29DC7	4	ole32.75E29DC7	系统模块
	0018ED40	76FBE192	76FBE434	48	ntdll.76FBE434	系统模块
	0018ED88	76FBE434	76FBE192	14	ntdll.76FBE192	系统模块
	0018ED9C	76FBE434	76FBE434	4	ntdll.76FBE434	系统模块
	0018EDA0	76FBE192	76FBE434	14	ntdll.76FBE434	系统模块
	0018EDB4	649327E4	76FBE192	40	ntdll.76FBE192	用户模块
	0018EDF4	64933593	649327E4	4C	msvcr71.649327E4	用户模块
	0018EE40	02F013D7	64933593	28	msvcr71.64933593	用户模块
	0018EE68	02F39509	02F013D7	2C	_dnavi.02F013D7	用户模块
	0018EE94	02F159C5	02F39509	34	_dnavi.02F39509	用户模块
	0018EEC8	020F40C5	02F159C5	1BC	_dnavi.02F159C5	用户模块
	0018F084	0350B476	020F40C5	FFE70F80	dzdatanavigatorserver.020F40C5	用户模块
	00000004	00000000	0350B476		dzdnavisatellite_pcdiagnose.0350B476	用户模块

3.3 Lập trình phần mềm phân tích cuộc gọi DLL

3.3.1 Phân tích ngăn xếp cuộc gọi

Các lệnh gọi hàm liên quan có thể được tìm thấy từ ngăn xếp cuộc gọi. Lấy thông báo "5A 00 00 01" làm ví dụ, trình tự gọi hàm chính như sau (một số hàm đã được đổi tên và địa chỉ offset liên quan đến địa chỉ cơ sở dll là trong ngoặc đơn.), bạn có thể thấy rằng các lệnh gọi hàm chủ yếu tập trung ở EUNIT_PLC_LN.dll.

ws2_32.dll là thư viện liên kết động của hệ điều hành được sử dụng để thực hiện truyền thông mạng TCP/IP.

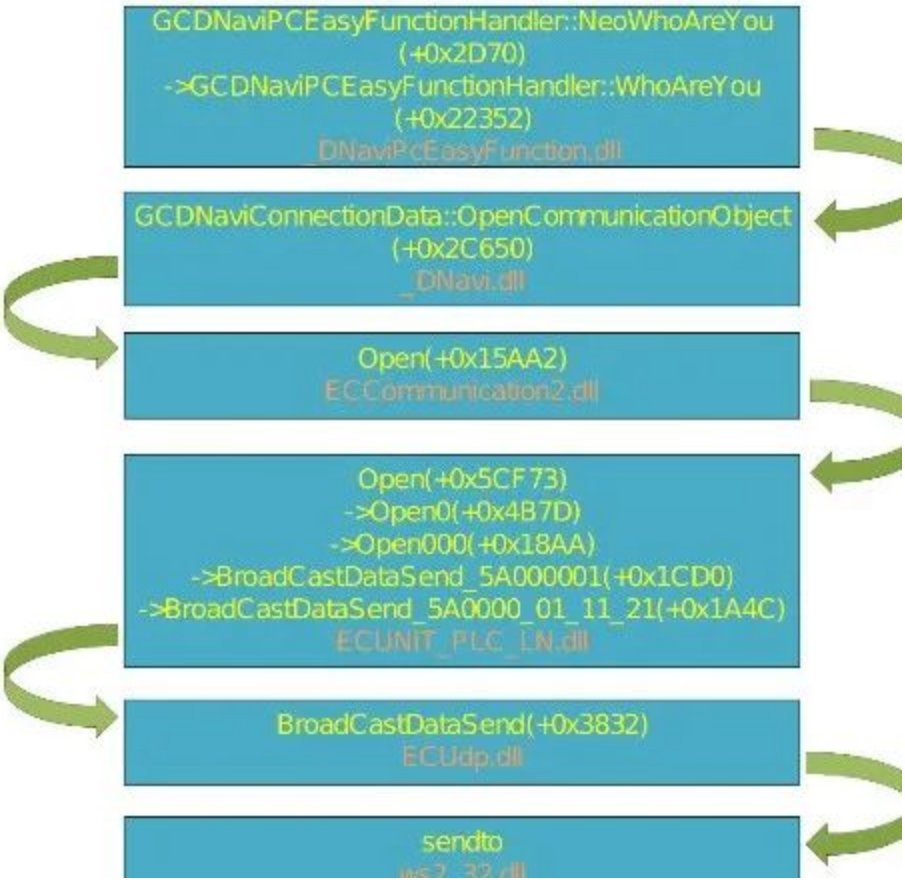
ECUdp.dll là một gói đơn giản của ws2_32.dll, chức năng chính của nó là gửi và nhận các gói dữ liệu UDP.

ECUNIT_PLC_LN.dll chủ yếu chịu trách nhiệm đóng gói và giải nén dữ liệu, sau đó gọi các chức năng liên quan trong ECUdp.dll.

ECCommunication2.dll đóng gói EUNIT_PLC_LN.dll.

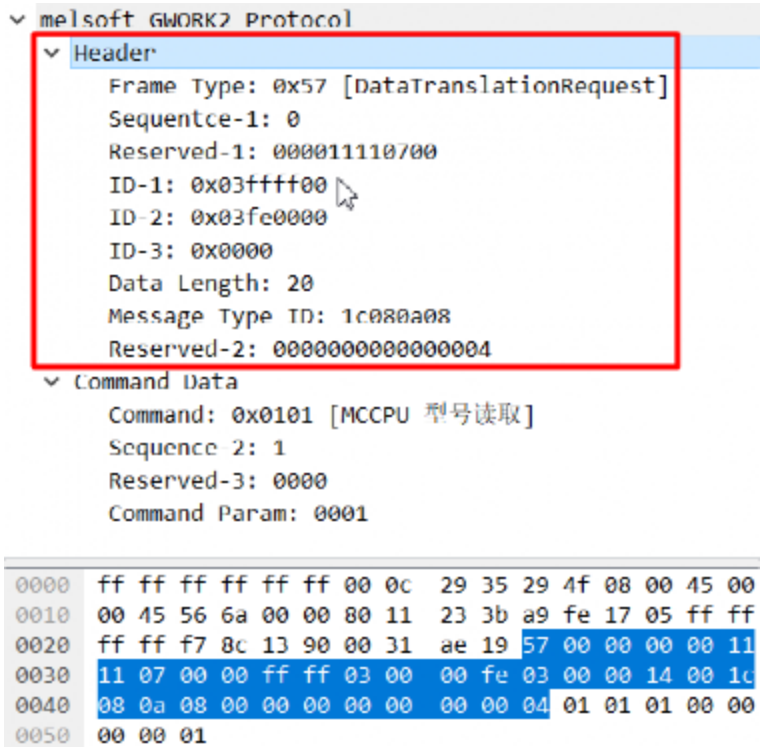
_DNavi.dll: Gọi các hàm liên quan trong ECCommunication2.dll.

_DNaviPcEasyFunction.dll: Gọi các hàm liên quan trong _DNavi.dll.



3.3.2 Phân tích đóng gói dữ liệu giao thức

3.3.2.1 Chức năng đóng gói tiêu đề dữ liệu



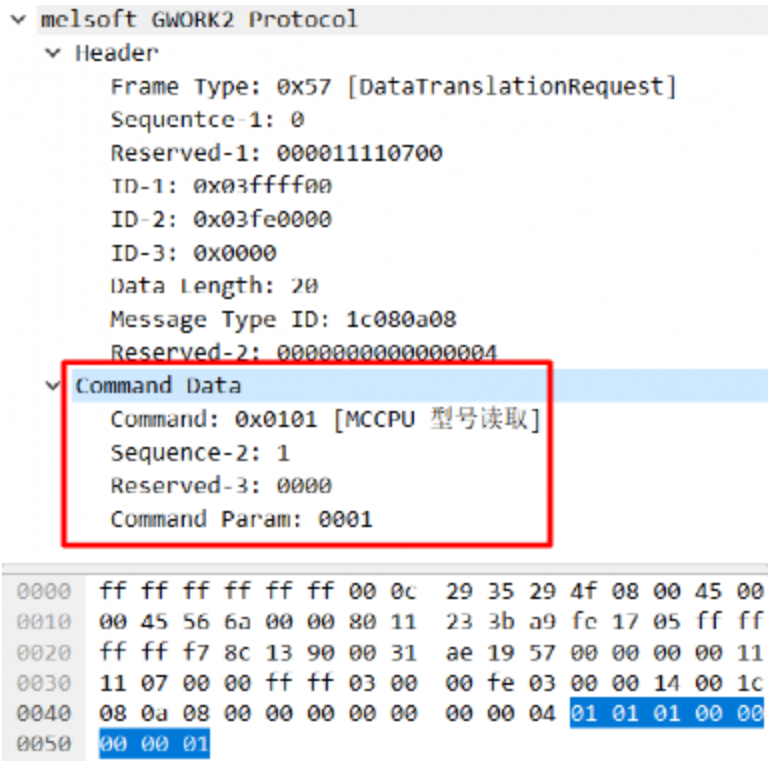
Chức năng gói được đặt tại (ECUNIT_PLC_LN.dll + 0x6C2F)

```
char __thiscall Packet(char *this, int headersize, int cmd_data_size, int ftsize, _DWORD *pkt_size)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  cmd_data_size00 = cmd_data_size;
  hdsize = headersize;                          // 一般为33个字节
  cmd_data_size0 = cmd_data_size;
  *(_DWORD *)(this + 107) = ftsize;             // 一般是0
  pfn1 = *(_DWORD *)this;
  *(_DWORD *)(this + 99) = hdsize;
  *(_DWORD *)(this + 103) = cmd_data_size00;
  v9 = (*(int (__thiscall **)(int, int, int))(pfn1 + 0x12E8))((int)this, hdsize, cmd_data_size0);// pfn1 PackCmdParam
  if ( v9 > 0 )
    cmd_data_size00 += v9;
  HeaderSizeGetQn(*((_DWORD *)this + 2), (int)&headersize);// header size为0x15
  app = (_BYTE *)(hdsize + *((_DWORD *)this + 4));
  if ( (*app & 0x20) != 0 )                     // 判断功能码
  {
    *app &= 0xDFu;
    v15 = 0x3C;
  }
  else
  {
    v15 = 0x1C;
  }
  (*(void (__thiscall **)(char *, int, int))(*(_DWORD *)this + 0xCD4))(this, cmd_data_size00, v15);// pfn1 HeaderMake0 生成数据包头部
  if ( (unsigned __int16)APP_SEQ < 255u )       // 序号，大于255，重置为1
    ++APP_SEQ;
  else
    APP_SEQ = 1;
  *(_WORD *)(hdsize + *((_DWORD *)this + 4) + 2) = APP_SEQ;
  pkt_size0 = pkt_size;
  *(_WORD *)(this + 57) = APP_SEQ;
  *pkt_size0 = hdsize + cmd_data_size00;
  buf0 = *((_DWORD *)this + 4);
  this[183] = *(_BYTE *)(buf0 + hdsize);
  result = *(_BYTE *)(buf0 + headersize);
  this[2037] = result;
  return result;
}
```

HeaderMake00 được đặt tại (ECHEADER_ETHER_PLC_LN.dll + 0x5E3)

```
int __stdcall HeaderMake00(int a1)
{
  int v1; // ebx
  int pkt; // esi

  v1 = *(_DWORD *)(a1 + 1);
  HeaderMake_21bytes(a1);                       // pkt[0:20]
  pkt = *(_DWORD *)(a1 + 9);
  memcpy((void *)(pkt + 21), &unk_3A34024, 12u);// pkt[21:33] 00 00 0A 08 00 00 00 00 00 00 00 00
  *(_BYTE *)(pkt + 21) = *(_BYTE *)(a1 + 13);
  *(_BYTE *)(pkt + 22) = 8;
  if ( !*(_DWORD *)(v1 + 124) )
    *(_BYTE *)(pkt + 22) = 40;
  if ( !*(_DWORD *)(v1 + 120) )
    *(_BYTE *)(pkt + 22) |= 0x80u;
  *(_WORD *)(pkt + 25) = *(_WORD *)(v1 + 100) & 0x3FF;
  *(_BYTE *)(pkt + 27) = *(_BYTE *)(v1 + 108) & 0xF;
  *(_BYTE *)(pkt + 32) = 4;
  *(_WORD *)(pkt + 19) = *(_WORD *)(a1 + 5) + 12;// 数据长度
  return 0;
}
```

3.3.2.2 Chức năng đóng gói dữ liệu lệnh



Lấy mã hàm 0101 làm ví dụ, BroadCastDataSend_fn57_0101 nằm ở (ECUNIT_PLC_LN.dll + 0x257BF)

```
int __thiscall BroadCastDataSend_fn57_0101(_DWORD *this, char cmd_parse, void *a3, int a4, int a5, int a6, void *a7, int a8, int a9, int a10, void *a11)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  result = (*(int (__thiscall **)(_DWORD *, int *))(*this + 0x87C))(this, &hdsize);// pfn1 + 0x87c DcHeaderSizeGet
  if ( !result )
  {
    v13 = this[2];
    hdsize0 = hdsize;
    v15 = v21;
    pos = v21;
    ftsize = FooterSizeGet1(v13);
    pkt = this[4];
    cmd_data = pkt + hdsize0;
    v17 = pkt + v15;
    memcpy((void *)(pkt + hdsize0), &cmd_0101_temp, 8u);// 01 01 00 00 00 00 00 00 初始化命令数据
    *(_BYTE *)(cmd_data + 7) = cmd_parse;
    (*(void (__thiscall **)(_DWORD *, int, int, int, int *))(*this + 0x844))(this, hdsize0, 8, ftsize, &size);// pfn1 + 0x844 Packet 构造数据包头部及序列号自加
    result = (*(int (__thiscall **)(_DWORD *, int, int))(*this + 2128))(this, pos, size);// BroadCastDataSend0 发送并接收返回数据
    if ( !result )
    {
      memcpy(a3, (const void *)(v17 + 6), 16u);
      *((_BYTE *)a3 + 16) = 0;
      *(_WORD *)a4 = *(_WORD *)(v17 + 22);
      *(_WORD *)a5 = *(_WORD *)(v17 + 24);
      *(_WORD *)a6 = *(_WORD *)(v17 + 26);
      memcpy(a7, (const void *)(v17 + 28), 8u);
```

3.3.3 Tính mã ủy quyền

Tọa lạc tại (UNIT_PLC_LN.dll + 0x16461), key được tạo theo model CPU, sau đó challenge_code được mã hóa và cuối cùng là mã ủy quyền được tạo.

```
melsoft GWORK2 Protocol
  Header
      Frame Type: 0xda [NetLinkResponse]
      Sequentce-1: 0
      NetLinkRequest Type: ff [Challenge Code]
      un27: 4405
      Postion of Challenge Code: 12
      un26: 0100440500100202
      Challenge Code: a98798534befde74c38a
      un26: 1003
```

```
melsoft GWORK2 Protocol
  Header
      Frame Type: 0x57 [DataTranslationRequest]
      Sequentce-1: 0
      Reserved-1: 000011110700
      ID-1: 0x03ffff00
      ID-2: 0x03fe0000
      ID-3: 0x0000
      Data Length: 50
      Message Type ID: 1c080a08
      Reserved-2: 0000000000000004
  Command Data
      Command: 0x0114 [MS Authentication]
      Sequence-2: 3
      Reserved-3: 0000
      Auth Code: bcc27e5b9940ea65c311556dd5864a9ebdf8d63789217b12b26040c6667e544a
```

```
0000  ff ff ff ff ff ff 00 0c  29 35 29 4f 08 00 45 00   ........ )5)O..E.
0010  00 63 00 88 00 00 80 11  12 ff a9 fe 7d 05 ff ff   .c...... ....}...
0020  ff ff eb 3e 13 90 00 4f  84 cd 57 00 00 00 00 11   ...>...O ..W.....
0030  11 07 00 00 ff ff 03 00  00 fe 03 00 00 32 00 1c   ........ .....2..
0040  08 0a 08 00 00 00 00 00  00 00 04 01 14 03 00 00   ........ ........
0050  00 bc c2 7e 5b 99 40 ea  65 c3 11 55 6d d5 86 4a   ...~[.@. e..Um..J
0060  9e bd f8 d6 37 89 21 7b  12 b2 60 40 c6 66 7e 54   ....7.!{ ..`@.f~T
0070  4a                                                J
```

```
int __thiscall calc_auth_0114_payload(int *this, int arg_0, int cpu_type_code, _BYTE *challenge_code)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  init_out_buf_60_63(v18);
  v5 = *this;
  v19 = 0;                                                    根据CPU型号生成密钥
  (*(void (__thiscall **)(int *, int, int, int *))(v5 + 0x115C))(this, arg_0, cpu_type_code, key_hmac);
  LOBYTE(tmp_buf0_1) = MELSEC_Q[7] ^ challenge_code[7];
  HIBYTE(tmp_buf0_1) = MELSEC_Q[3] ^ challenge_code[3];
  LOBYTE(tmp_buf2_3) = MELSEC_Q[0] ^ *challenge_code;
  HIBYTE(tmp_buf2_3) = MELSEC_Q[6] ^ challenge_code[6];
  LOBYTE(tmp_buf4_5) = MELSEC_Q[5] ^ challenge_code[5];
  HIBYTE(tmp_buf4_5) = MELSEC_Q[2] ^ challenge_code[2];       challeng_code编码
  LOBYTE(tmp_buf6_7) = MELSEC_Q[4] ^ challenge_code[4];
  HIBYTE(tmp_buf6_7) = MELSEC_Q[1] ^ challenge_code[1];
  v6 = challenge_code[9];
  LOBYTE(challenge_code_sum) = challenge_code[8];
  HIBYTE(challenge_code_sum) = v6;
  if ( tmp_buf6_7 + tmp_buf4_5 + tmp_buf0_1 + tmp_buf2_3 == challenge_code_sum )
  {
    data_hmac[0] = tmp_buf6_7 * tmp_buf2_3;
    data_hmac[3] = tmp_buf6_7 * tmp_buf6_7;
    data_hmac[1] = tmp_buf6_7 * tmp_buf0_1;
    data_hmac[2] = tmp_buf6_7 * tmp_buf4_5;
    hmac_sha256((int)key_hmac, (int)data_hmac, 16, (int)out_hmac);   计算授权码
```

3.4 Phân tích mô phỏng PLC

Quá trình phân tích tương tự như quá trình lập trình phần mềm và sẽ không lặp lại ở đây, đối

tượng phân tích là QnUDSimRun2.exe.

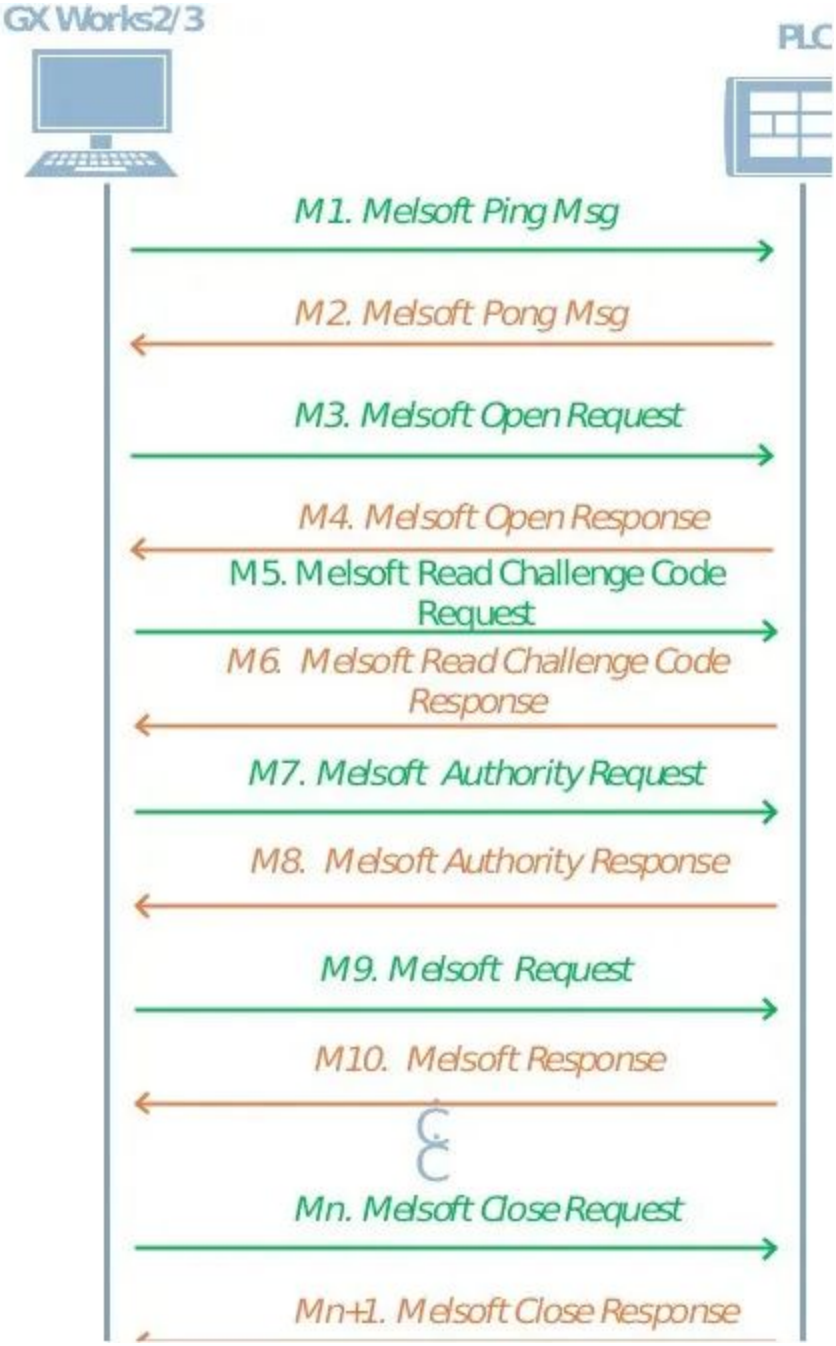
Giao thức truyền thông được sử dụng để liên lạc mô phỏng hơi khác so với Melsoft, lý do chính là tiêu đề gói dữ liệu khác nhau và phần dữ liệu lệnh về cơ bản giống nhau. Bằng cách phân tích chức năng xử lý lệnh của nó, có thể suy ra các loại trường và chức năng của dữ liệu lệnh trong gói dữ liệu. Về cơ bản, tất cả các hàm xử lý lệnh đều được đăng ký trong một hàm (QnUDSimRun2.exe+0x9070) và sau đó được gọi.

```
int __thiscall register_pkt_cmd_process_function(void *this, int a2, int a3)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v535 = 0;
  v536 = 0;
  v537 = 0;
  v538 = 0;
  v539 = 0;
  v540 = 0;
  v541 = 0;
  v542 = 0;
  v543 = 0;
  cpu_type = CPU_TYPE_CODE & 0xFFF0;
  Destination = 0;
  if ( (_WORD)cpu_type == 0x200 || (_WORD)cpu_type == 0x300 || (_WORD)cpu_type == 0x540 )
  {
    v20 = (int)this + 2119;
    v24 = (int)this + 2119;
    v28 = (int)this + 2119;
    v32 = (int)this + 2119;
    v36 = (int)this + 2119;
    cmd_pfn_array2 = (int)this + 2118;
    v12 = (int)this + 2118;
    v16 = (int)this + 2118;
    v40 = (int)this + 2120;
    v44 = (int)this + 2120;
    v48 = (int)this + 2120;
    v52 = (int)this + 2120;
    v56 = (int)this + 2120;
    v60 = (int)this + 2120;
    v54 = (int)this + 2120;
    v58 = (int)this + 2120;
    cmd_pfn_array0 = 1;
    cmd_pfn_array1 = 1;
    cmd_pfn_array6 = pkt_cmd_process_0101;
```

4. Phân tích kết quả

4.1 Quá trình tương tác truyền thông

Dựa trên kết quả phân tích, quy trình tương tác giao tiếp giao thức Melsoft được suy ra.



4.2 Ví dụ về cấu trúc gói dữ liệu

Bảng 1 Các loại khung dữ liệu

类型值（hex）	请求		响应	
	57	5A	D7	DA
描述	数据传输帧	连接控制帧	数据传输帧	连接控制帧
举例	读取 CPU 信息	打开、关闭连接	读取 CPU 信息	打开、关闭连接

Bảng 2-1 Đọc tiêu đề thông báo yêu cầu mô hình CPU

类型	header								
长度	1 字节	2 字节	6 字节	4 字节	4 字节	2 字节	2 字节	4 字节	8 字节
字段	Frame Type	Sequentce-1	Reserved-1	ID-1	ID-2	ID-3	data length	Msg Type ID	Reserved-2
典型值	57	/	11110700	0000 e403	00ff ff03	0	1400	1c08 0a08	4

Bảng 2-2 Đọc dữ liệu thông báo yêu cầu mô hình CPU

类型	data			
长度	2 字节	2 字节	2 字节	2 字节
字段	Command	Sequentce-2	Reserved-3	Command Param
典型值	101	/	/	1

Bảng 3-1 Đọc tiêu đề thông báo phản hồi của mô hình CPU

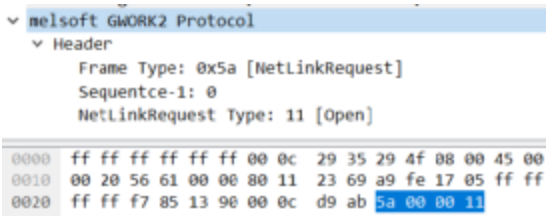
类型	header									
长度	1字节	2字节	6字节	4字节	4字节	2字节	2字节	4字节	2字节	8字节
字段	Frame Type	Sequentce-1	Reserved-1	ID-1	ID-2	ID-3	data length	Msg Type ID	Error Code	Reserved-2
典型值	d7	/	11110700	0000 e403	00ff ff03	0	3600	1c08 0a08	0000	00000004 00000000

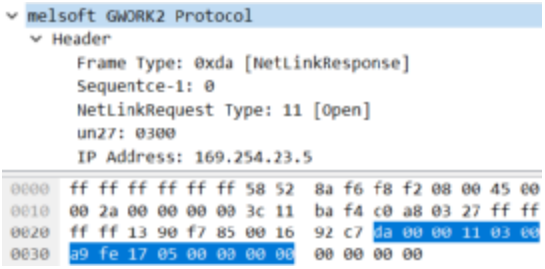
Bảng 3-2 Đọc dữ liệu thông báo phản hồi mô hình CPU

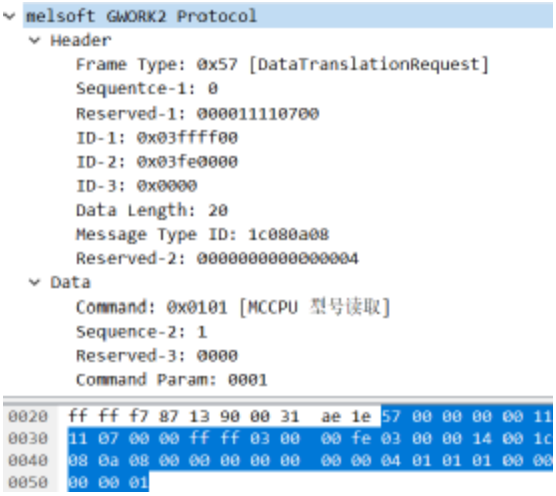
类型	data					
长度	2字节	2字节	2字节	16字节	2字节	16字节
字段	Command	Sequentce-2	Reserved-3	CPU Type	CPU Type Code	unknown
典型值	101	/	/	/	/	/

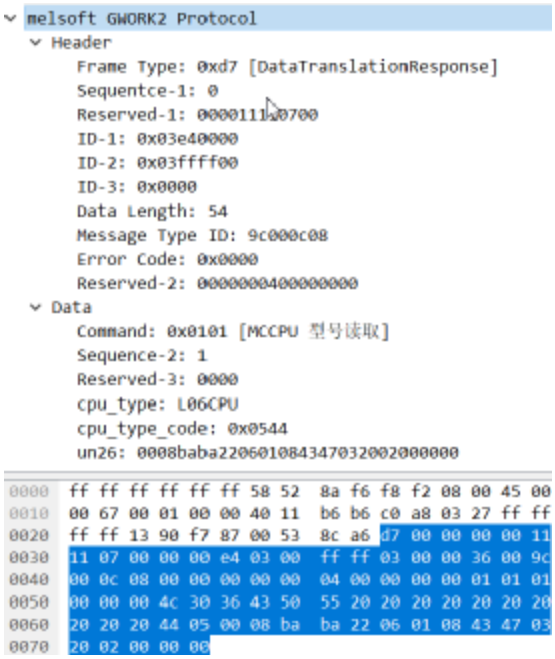
4.3 Trình cắm phân tích giao thức

Lấy việc đọc mã chức năng mô hình CPU 0101 làm ví dụ, hiệu ứng plug-in phân tích cú pháp của Wireshark như sau:









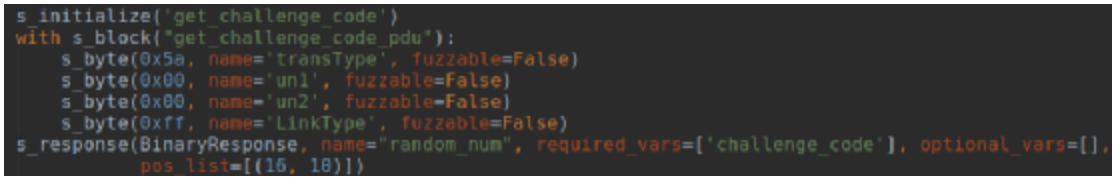
5. Giao thức Fuzz

Lần này công cụ fuzz là Fuzzowski, được viết bằng python3. Xác định mô hình dữ liệu dựa trên định dạng giao thức melsoft được thiết kế ngược và sau đó thực hiện kiểm tra fuzz. Khó khăn chính nằm ở việc lấy dữ liệu phản hồi của PLC, sau đó tính toán mã ủy quyền rồi gửi đến

PLC, mô-đun tổng kiểm tra trong công cụ cần được mở rộng.

5.1 Nhận dữ liệu phản hồi

Lưu trữ 10 byte bắt đầu từ offset 16 trong thông báo phản hồi Mã thách thức vào từ điển chung, với khóa là challenge_code.



```
s_initialize('get_challenge_code')
with s_block("get_challenge_code_pdu"):
    s_byte(0x5a, name='transType', fuzzable=False)
    s_byte(0x00, name='un1', fuzzable=False)
    s_byte(0x00, name='un2', fuzzable=False)
    s_byte(0xff, name='LinkType', fuzzable=False)
s_response(BinaryResponse, name="random_num", required_vars=['challenge_code'], optional_vars=[],
           pos_list=[(16, 10)])
```

5.2 Thuật toán mở rộng

Đã thêm thuật toán MelsoftAuth trong mô-đun tổng kiểm tra

```
elif self._algorithm == "MelsoftAuth":
    field_dict = {}
    for FieldName in ['challenge_code', ]:
        for FieldItem in self._request.variables:
            if FieldName == FieldItem:
                field_dict[FieldName] = self._request.variables[FieldName]
    check = melsoft_hmac(field_dict['challenge_code'])
```

Gọi thuật toán MelsoftAuth mở rộng trong định nghĩa khối dữ liệu

```
with s_block('cmd_data'):
    s_word(0x1401, name='command', fuzzable=False)
    s_word(0x0001, name='sequence', fuzzable=False)
    s_word(0x0000, name='un23', fuzzable=False)
    s_checksum(block_name='app', algorithm='MelsoftAuth', fuzzable=False)
```

6. Tóm tắt

Vẫn còn một số trường trong các tin nhắn hiện được phân tích mà ngữ nghĩa của chúng chưa rõ ràng và cần được phân tích thêm. Kết quả phân tích giao thức dựa trên cấu hình hiện tại. Các thông báo giao thức MELSOFT hơi khác nhau trong các môi trường cấu hình khác nhau (chẳng

hạn như GX Works3 và FX5U). Nghiên cứu tiếp theo cần được tiến hành theo các cấu hình khác để cải thiện phân tích giao thức MELSOFT. Nghiên cứu này mới chỉ hoàn thành một phần phân tích của giao thức Melsoft. Chúng tôi hy vọng nó sẽ hữu ích cho các nhà nghiên cứu. Hãy phê bình và sửa chữa bất kỳ điều gì không phù hợp.

Người giới thiệu:

1.Tách rời và tiếp quản hệ sinh thái ICS-SCADA Một nghiên cứu điển hình về Mitsubishi Electric

https://hitcon.org/2021/agenda/8335bbd7-5072-4fca-aae5-

b657cbf60336/Taking%20Apart%20and%20Taking%20Over%20ICS%20_%20SCADA%20Ecosystems_%20A%20Case%20Study%20of%20Mitsubishi% 20Điện.pdf

2. Báo cáo phân tích an toàn PLC dòng Q của Mitsubishi

http://plcscan.org/blog/2014/08/mitsubishi-electric-melsec-q-series-plc-analysis-report/

Nguồn gốc: Trung tâm nghiên cứu kỹ thuật quốc gia về công nghệ khẩn cấp an ninh mạng

Rút lại và sửa chữa

đến bài viết này phải được ghi chú khi in lại

Diễn đàn tội phạm mạng hàng đầu vạch trần danh tính 100.000 hacker, rò rỉ dữ liệu toàn cầu

Một báo cáo gần đây đã tiết lộ hàng loạt vụ vi phạm dữ liệu trên các diễn đàn hacker. Theo Hudson Rock, các lỗ ...

Một công ty chứng khoán bị cảnh cáo do vi phạm trong sự cố an ninh mạng !

Thứ hai, một kế hoạch thử nghiệm toàn diện đã không được phát triển trước khi thay đổi các hệ thống thông tin ...

Mười câu hỏi doanh nghiệp nên biết về lưới an ninh mạng

" CyberSecurity Mesh" là một khái niệm phát triển công nghệ an ninh mạng đổi mới do tổ chức nghiên cứu quốc...

Hội nghị Công nghiệp Bảo mật Dữ liệu và Mạng lưới Kinh tế Kỹ thuật số Quốc tế Trung Quốc năm ...

Hội nghị Công nghiệp An ninh Dữ liệu và Mạng năm 2023 do Ban tổ chức Triển lãm Kinh tế Kỹ thuật số Quốc tế ...

Tiêu chuẩn quốc tế: Tích hợp an ninh mạng vào các chỉ số đánh giá lương quản lý công ty

Security Insider đưa tin vào ngày 5 tháng 9 rằng một số công ty đã bắt đầu liên kết tiền thưởng của các CEO và l...

Với giải thưởng lên tới 9 triệu USD, Bộ Năng lượng Hoa Kỳ phát động cuộc thi an ninh mạng trong...

Để cải thiện tình trạng an ninh mạng của các công ty điện lực nhỏ, Bộ Năng lượng Hoa Kỳ gần đây đã công bố m...

Tập đoàn Venus đứng đầu trong thị trường phần cứng an ninh mạng Trung Quốc !

Gần đây, IDC đã công bố báo cáo " Thị phần phần cứng an ninh mạng Trung Quốc năm 2022". Báo cáo cho thấy ...

Tuần lễ nâng cao nhận thức về an ninh mạng quốc gia năm 2023 sẽ được tổ chức trên toàn quốc t...

Vào ngày 31 tháng 8 năm 2023, cuộc họp báo Tuần lễ An ninh mạng Quốc gia năm 2023 đã được tổ chức tại Bắc...

Lễ ra mắt Tháng công khai về an ninh mạng đầu tiên của China Telecom đã được tổ chức tại Bắc Ki...

Chiều 4/9, China Telecom đã tổ chức lễ phát động Tháng Nhận thức về An ninh mạng tại Bắc Kinh. Sự kiện nhằm...

Bộ Công an: Triệt để trấn áp tội phạm mạng , bảo vệ hiệu quả an ninh mạng và dữ liệu quốc gia

2023年7月6日，公安部召开“公安心向党 护航新征程”系列主题新闻发布会。其中，公安部牵头建立的网络安全等...

高通违规获取用户隐私信息，一直在秘密收集私人用户数据

关键词个人数据据称，一家制造无线电信硬件的跨国高通公司一直在秘密收集私人用户数据。大约三分之一的安...

分组密码的隐秘密文分组链接模式

Phương pháp tạo nhãn xác thực được đưa ra để chế độ làm việc này có thể cung cấp chức năng mã hóa dữ liệu ...