三菱PLC MELSOFT通信协议浅析

CISRC 发布于 2022-07-19 14:47:28

1. 概述

三菱 MELSOFT 协议为三菱 PLC 私有组态协议，用于编程软件与三菱 PLC 通信。针对该协议公开资料并不多，本次主要对该协议进行分析，并基

于分析结果对三菱 PLC 进行 MELSOFT 协议模糊测试。

2. 环境配置

GX WORKS2：三菱 PLC 编程软件，适用于 Q、QnU、L、FX 等系列可编程控制器，并且支持 PLC 仿真。

L61P-CM：L06 系列 CPU，默认使用 TCP/5007 或 UDP/5008 端口与编程软件进行通信。该通信协议为三菱私有协议 melsoft。

逆向工具：IDA 7.5

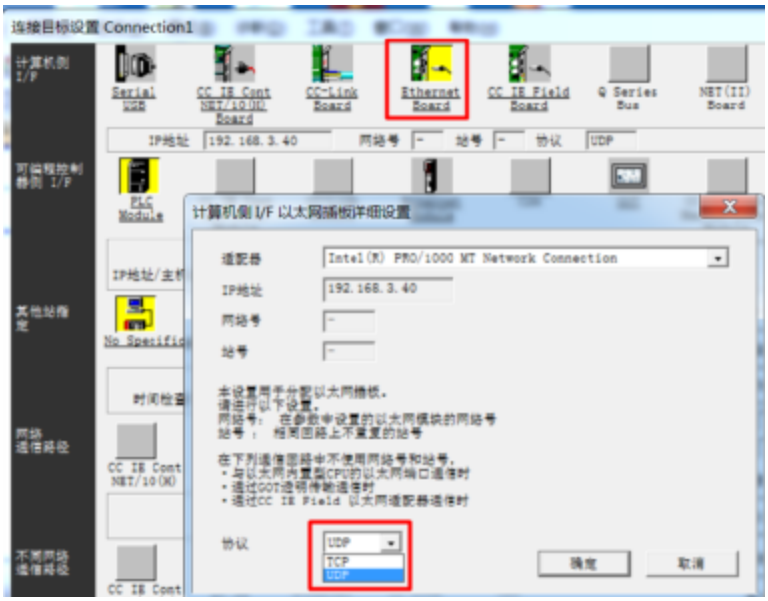
调试工具：x64dbg

3. 分析过程

3.1 信息搜集及分析

3.1.1 通信端口

GX WORKS2 默认以 UDP 广播方式与 PLC UDP 5008 端口进行组态协议通信，在 GX WORKS2 上可通过配置连接目标，以 TCP 方式与 PLC TCP 5007 端口进行组态协议通信。

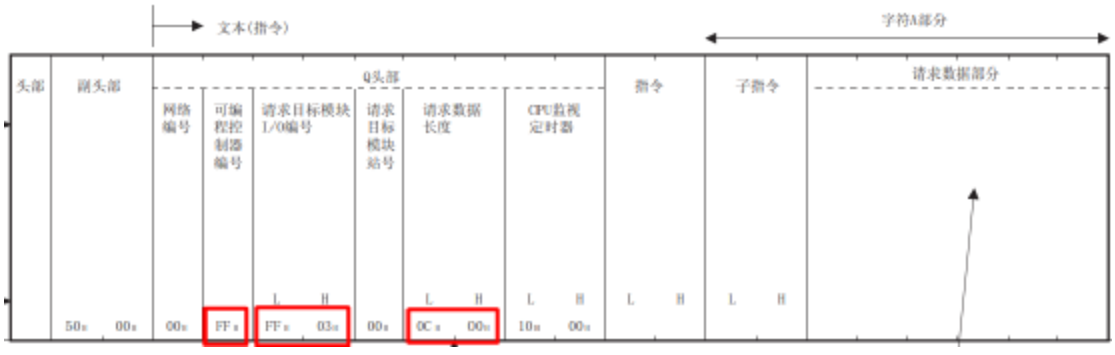


1	0.000000	192.168.3.43	255.255.255.255	UDP	46	52892 → 5008 Len=4
2	0.001041	192.168.3.39	255.255.255.255	UDP	60	5008 → 52892 Len=14
3	0.368583	192.168.3.43	255.255.255.255	UDP	46	52893 → 5008 Len=4
4	0.369528	192.168.3.39	255.255.255.255	UDP	60	5008 → 52893 Len=14
5	0.398421	192.168.3.43	255.255.255.255	UDP	46	52894 → 5008 Len=4
6	0.400437	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

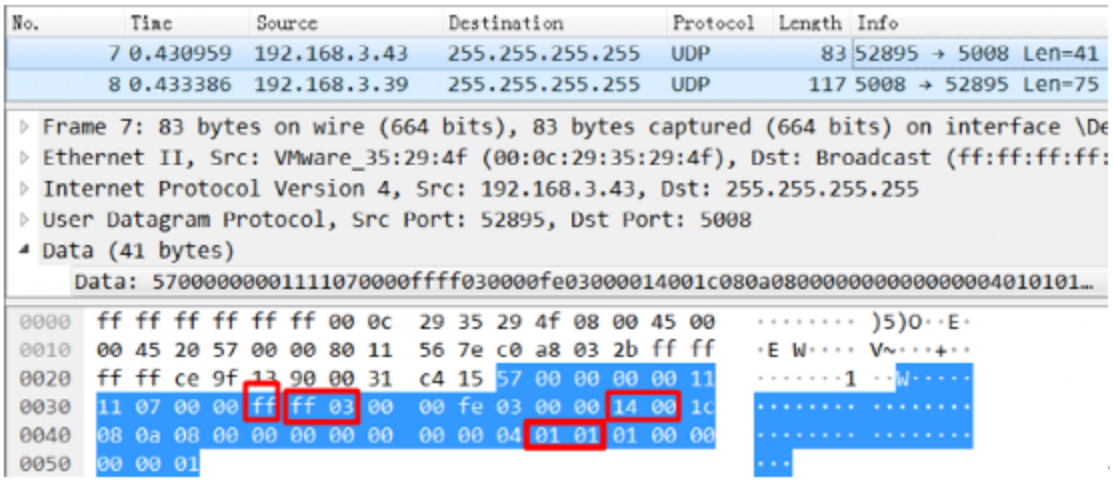
```
13 2.558853  192.168.3.43  192.168.3.39  TCP   66 49164 → 5007 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
15 2.560192  192.168.3.39  192.168.3.43  TCP   60 5007 → 49164 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
16 2.560456  192.168.3.43  192.168.3.39  TCP   54 49164 → 5007 [ACK] 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
```

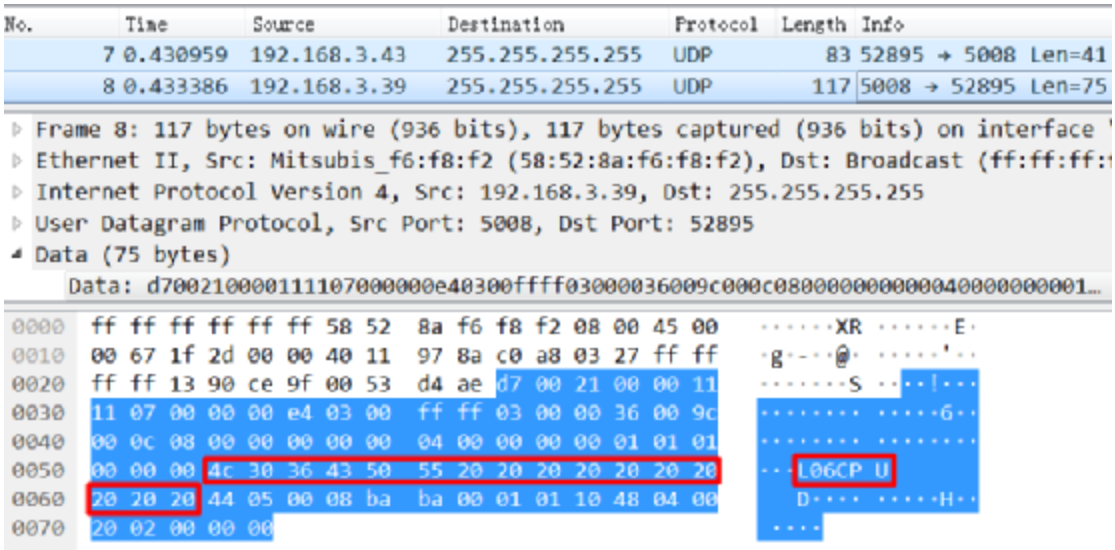
3.1.2 协议相似性

经过网上的一些资料以及对真实 PLC 流量进行分析，发现 MELSOFT 协议与三菱 MC 协议有一些相似之处（三菱 MC 协议是一个公开协议，可直接从三菱官网下载通讯协议参考手册）。例如报文格式（MC 协议 3E 帧）、“CPU 型号读取”功能。



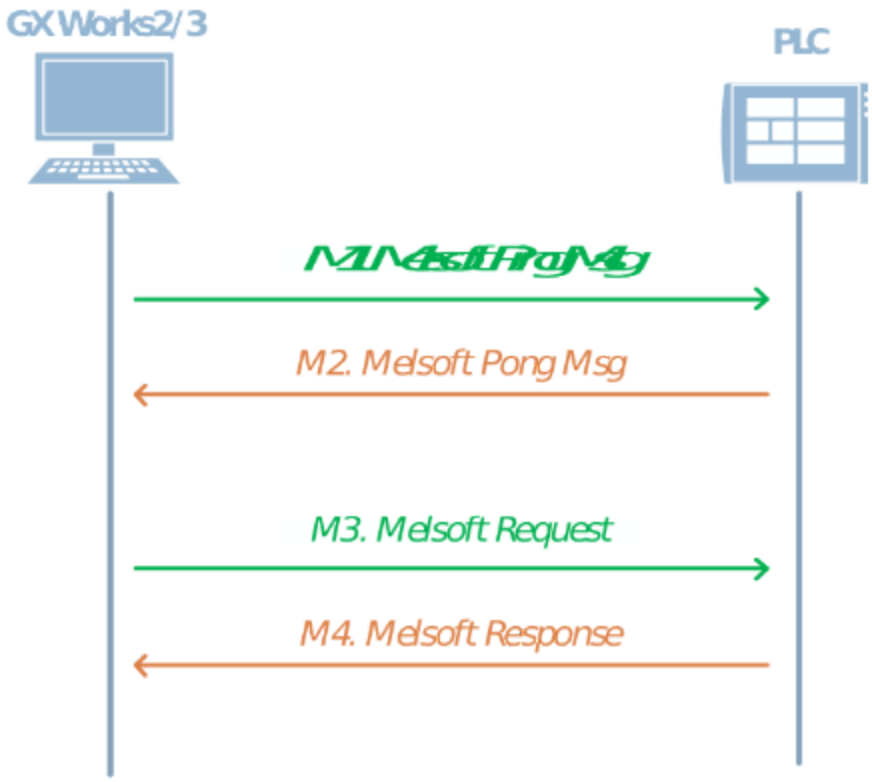
功能	指令(*1) (子指令)	处理内容
远程 RUN	1001 (0000)	进行远程 RUN(执行运算)请求。
远程 STOP	1002 (0000)	进行远程 STOP(停止运算)请求。
远程 PAUSE	1003 (0000)	进行远程 PAUSE(停止运算)请求。 (保持输出状态)
远程锁存清除	1005 (0000)	STOP 状态时，进行远程锁存清除(软元件存储器的清除)请求。
远程 RESET	1006 (0000)	STOP 状态时，进行远程 RESET(开始执行运算)请求。
CPU 型号读取	0101 (0000)	进行可编程控制器 CPU 的型号读取请求。





3.1.3 协议交互流程整理

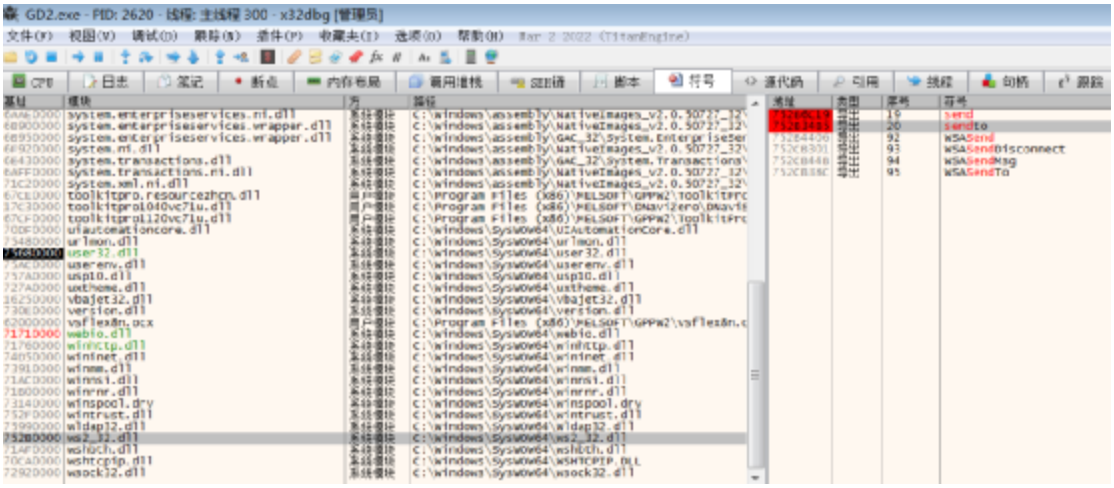
根据网络搜集的资料，初步整理出协议的交互流程如下：

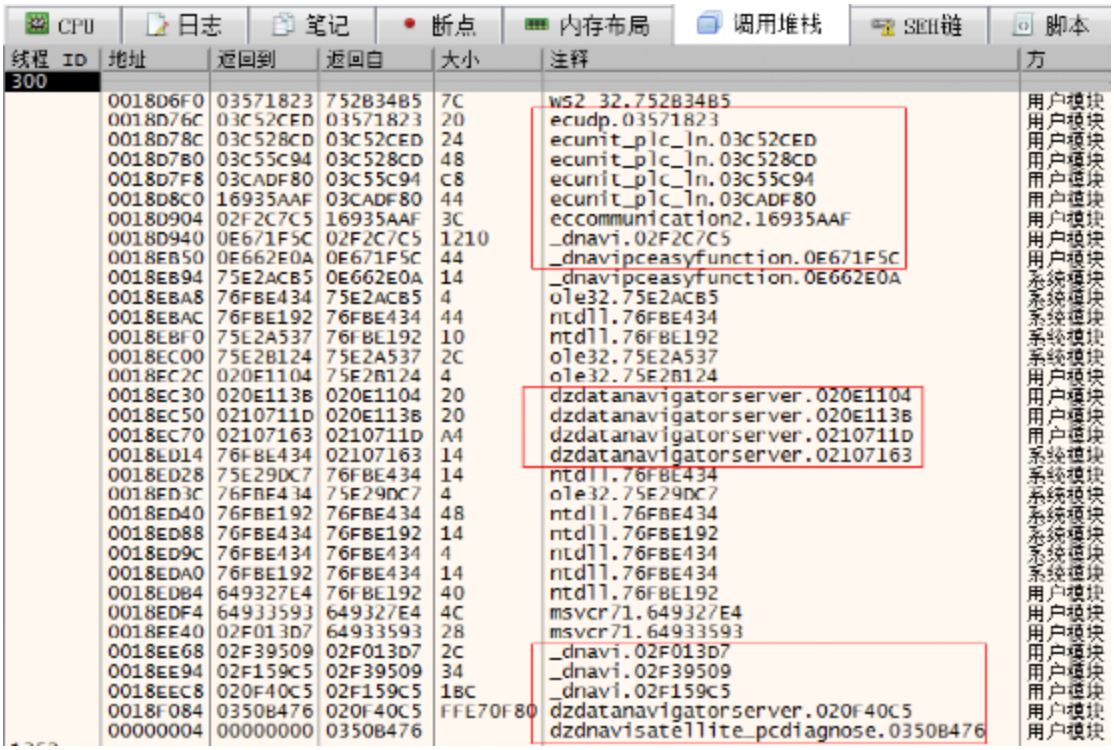


3.2 动态调试 GX Works2 编程软件

在 GX WORKS2 运行后，使用 x64dbg 附加到程序，在 “ws2_32.dll” 的 send/sendto 函数设置断点。在 GX WORKS2 中执行 “数据读取”、“远程操

作”等操作，查看调用堆栈，分析函数调用关系。





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 编程软件 DLL 调用分析

3.3.1 调用堆栈分析

从调用堆栈中可查找到相关的函数调用，以发送“5A 00 00 01”这条报文为例，主要函数调用顺序如下（部分函数已重命名，括号中为相对 dll 基址的偏移地址），可以看到函数调用主要集中于 ECUNIT_PLC_LN.dll。

ws2_32.dll 是用于执行 TCP/IP 网络通信的操作系统动态链接库。

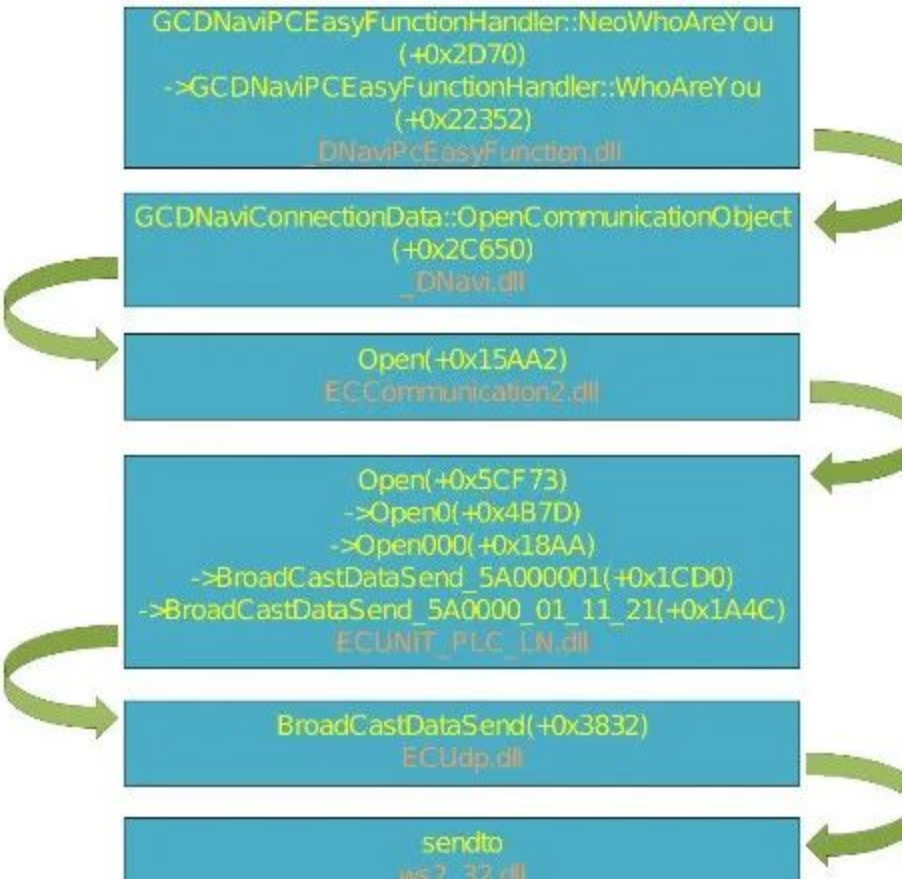
ECUdp.dll 是对 ws2_32.dll 的简单封装，主要作用是发送和接收 UDP 数据包。

ECUNIT_PLC_LN.dll 主要负责的数据封包和解包，然后调用 ECUdp.dll 中的相关函数。

ECCommunication2.dll 对 ECUNIT_PLC_LN.dll 进行封装。

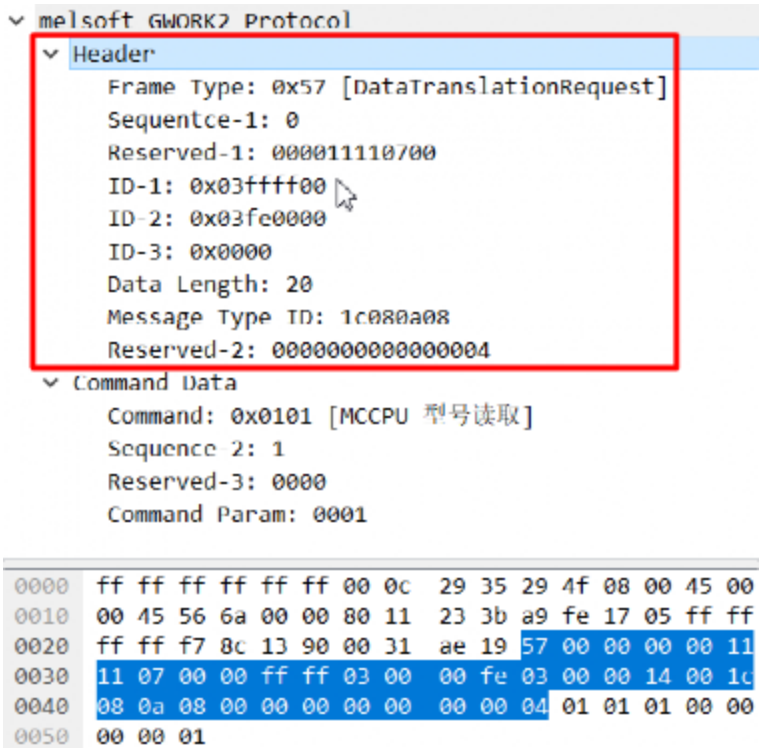
_DNavi.dll： 调用 ECCommunication2.dll 中的相关函数。

_DNaviPcEasyFunction.dll： 调用_DNavi.dll 中的相关函数。



3.3.2 协议数据封装分析

3.3.2.1 数据头封装函数



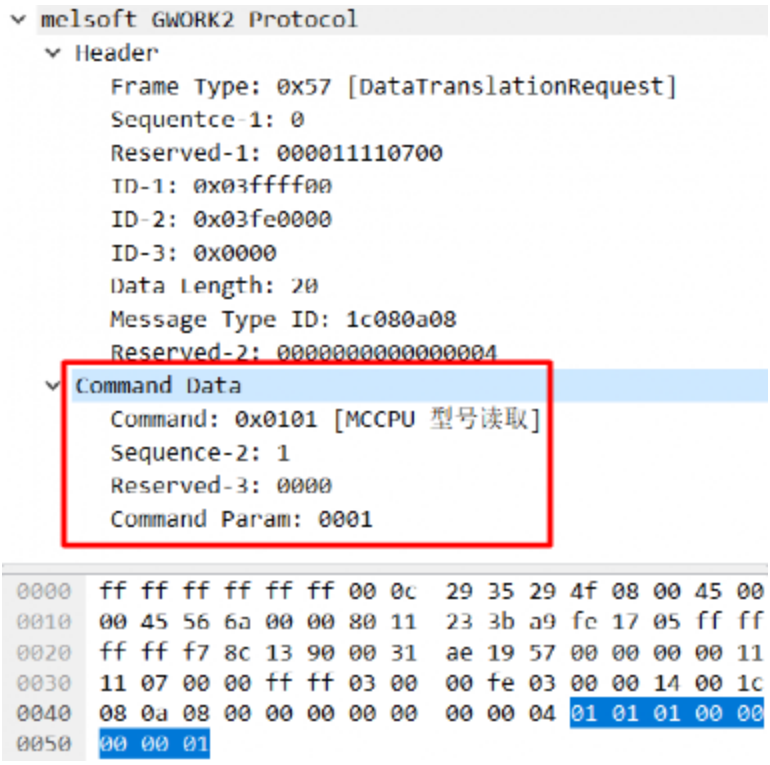
Packet 函数位于（ECUNIT_PLC_LN.dll + 0x6C2F)

```
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 位于 (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 命令数据封装函数



以 0101 功能码为例，BroadCastDataSend_fn57_0101 位于 (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 构造数据包头尾及序列号自加
    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 授权码计算

位于 (UNIT_PLC_LN.dll + 0x16461)，根据 CPU 型号生成密钥，然后对 challenge_code 进行编码，最后生成授权码。

```
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];
  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
```

challeng_code编码

3.4 PLC 模拟器分析

分析过程与编程软件类似，此处不再赘述，分析对象为 QnUDSimRun2.exe。

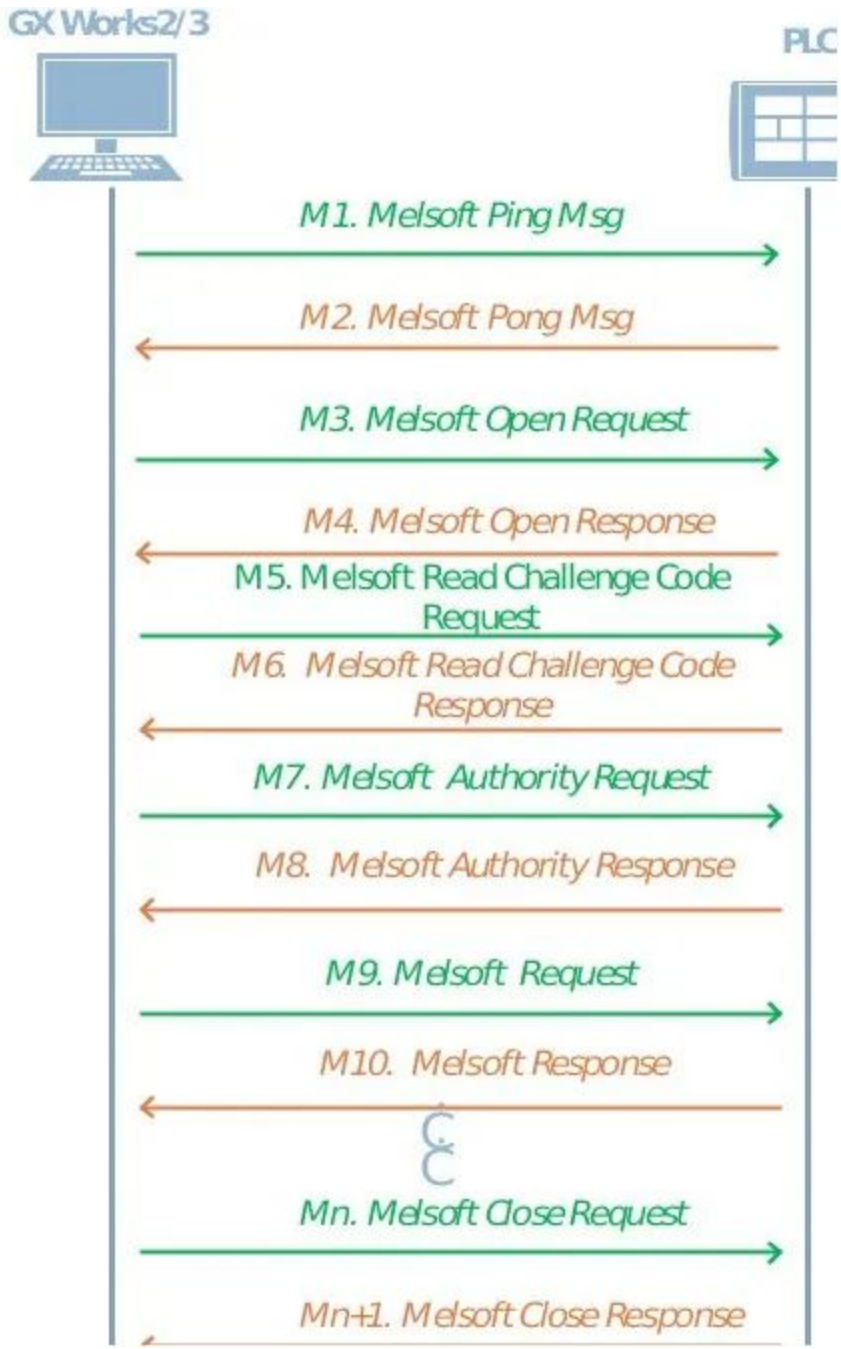
模拟器通信使用的通信协议与 Melsoft 略有不同，主要是数据包头部不相同，命令数据部分基本一致。通分析其命令处理函数，可推测数据包中命令数据的字段类型及作用。所有命令处理函数基本都在一个函数中统一进行注册（QnUDSimRun2.exe+0x9070），然后再进行调用。

```
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 == 0x260 || (_WORD)cpu_type == 0x360 || (_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;
    v64 = (int)this + 2120;
    v68 = (int)this + 2120;
    cmd_pfn_array0 = 1;
    cmd_pfn_array1 = 1;
    cmd_pfn_array6 = pkt_cmd_process_0101;
```

4. 分析结果

4.1 通信交互流程

根据分析结果，推测 Melsoft 协议通信交互流程。



4.2 数据包结构示例

表 1 数据帧类型

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

表 2-1 读取 CPU 型号请求报文 header

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

表 2-2 读取 CPU 型号请求报文 data

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

表 3-1 读取 CPU 型号响应报文 header

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

表 3-2 读取 CPU 型号响应报文 data

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 协议解析插件

以读取 CPU 型号功能码 0101 为例，wireshark 解析插件效果如下：

```
melsoft GWORK2 Protocol
  Header
      Frame Type: 0x5a [NetLinkRequest]
      Sequentce-1: 0
      NetLinkRequest Type: 11 [Open]
```

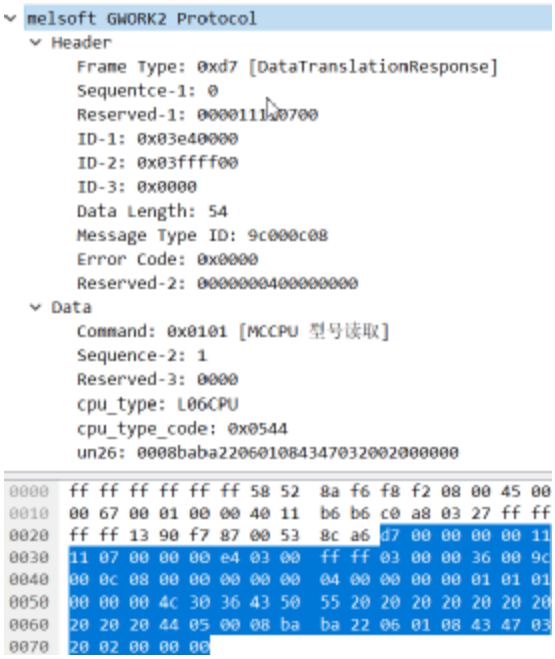
```
0000  ff ff ff ff ff ff 00 0c  29 35 29 4f 08 00 45 00
0010  00 20 56 61 00 00 80 11  23 69 a9 fe 17 05 ff ff
0020  ff ff f7 85 13 90 00 0c  d9 ab 5a 00 00 11
```

```
melsoft GWORK2 Protocol
  Header
      Frame Type: 0xda [NetLinkResponse]
      Sequentce-1: 0
      NetLinkRequest Type: 11 [Open]
      un27: 0300
      IP Address: 169.254.23.5
```

```
0000  ff ff ff ff ff ff 58 52  8a f6 f8 f2 08 00 45 00
0010  00 2a 00 00 00 00 3c 11  ba f4 c0 a8 03 27 ff ff
0020  ff ff 13 90 f7 85 00 16  92 c7 da 00 00 11 03 00
0030  a9 fe 17 05 00 00 00 00  00 00 00 00
```

```
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: 20
      Message Type ID: 1c080a08
      Reserved-2: 0000000000000004
  Data
      Command: 0x0101 [MCCPU 型号读取]
      Sequence-2: 1
      Reserved-3: 0000
      Command Param: 0001
```

```
0020  ff ff f7 87 13 90 00 31  ae 1e 57 00 00 00 00 11
0030  11 07 00 00 ff ff 03 00  00 fe 03 00 00 14 00 1c
0040  08 0a 08 00 00 00 00 00  00 00 04 01 01 01 00 00
0050  00 00 01
```



5. 协议 Fuzz

本次 fuzz 工具选择 Fuzzowski，该工具由 python3 编写。根据逆向出 melsoft 协议格式定义出数据模型，然后进行 fuzz 测试。主要难点在于获取 PLC 响应数据，然后计算授权码，再发送到 PLC，需要对工具中的 checksum 模块进行扩展。

5.1 获取响应数据

将 Challenge Code 响应报文中偏移 16 位置开始的 10 个字节存储到全局字典当中，key 为 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 扩展算法

在 checksum 模块新增 MelsoftAuth 算法

```
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'])
```

在数据块定义中调用扩展的 MelsoftAuth 算法

```
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. 总结

目前分析的报文中仍有部分字段语义不明，需要做进一步的分析。协议分析结果基于当前配置，不同配置环境下 MELSOFT 协议报文略有不同（如 GX Works3 和 FX5U）。后续还需在其他配置下进行研究，以完善 MELSOFT 协议解析。本次研究仅完成了 Melsoft 协议的部分解析，希望对同行研究者有所帮助，不当之处还请批评指正。

参考资料：

1.Taking Apart and Taking Over ICS-SCADA Ecosystems A Case Study of 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%20Electric.pdf

2. 三菱 Q 系列 PLC 安全分析报告

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

原文来源：网络安全应急技术国家工程研究中心

撤稿纠错

顶级网络犯罪论坛曝光10万黑客身份，数据泄露冲击全球

最近的一份报告披露了黑客论坛中的一系列数据泄露事件。据Hudson Rock报道，这些漏洞已经影响了超过12000...

因网络安全事件中存在违规行为，一证券公司被警示！

二是变更重要信息系统前未制定全面的测试方案。该行为违反了《办法》第十六条第一款的规定。四是事件调查...

关于网络安全网格，企业应该了解的十个问题

“网络安全网格（CyberSecurity Mesh）”是国际研究机构Gartner 提出的一个创新网络安全技术发展理念，近两年...

2023中国国际数字经济博览会网络和数据安全产业大会将召开

由中国国际数字经济博览会组委会主办，工业和信息化部网络安全产业发展中心（工业和信息化部信息中心）承...

国际风向标：将网络安全纳入公司管理层薪酬考核指标

安全内参9月5日消息，一些公司开始将首席执行官和其他高层领导的奖金与网络安全指标挂钩。治理专家表示，...

奖金高达900万美元，美国能源部在电力等领域发起网络安全竞赛

为改善小型电力公用事业的网络安全状况，美国能源部最近宣布了一项具有高额奖金的网络安全竞赛。这项名为A...

中国网络安全硬件市场，启明星辰集团第一！

近日，IDC发布《中国网络安全硬件市场份额，2022》报告。报告显示，启明星辰集团以10.2%的市场份额登顶榜...

2023年国家网络安全宣传周将于9月11日至17日在全国范围举行

2023年8月31日，2023年国家网络安全宣传周新闻发布会在京举行。中央网信办网络安全协调局局长高林，福州...

中国电信首届网络安全宣传月启动仪式在京举行

9月4日下午，中国电信在北京举行网络安全宣传月启动仪式。该活动旨在以2023年国家网络安全宣传周为契机，...

公安部：严厉打击网络违法犯罪 切实维护国家网络和数据安全

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

给出了一种产生认证标签的方法，使得该工作模式可提供数据加密和报文完整性检验功能。有些轻量级分组密码...