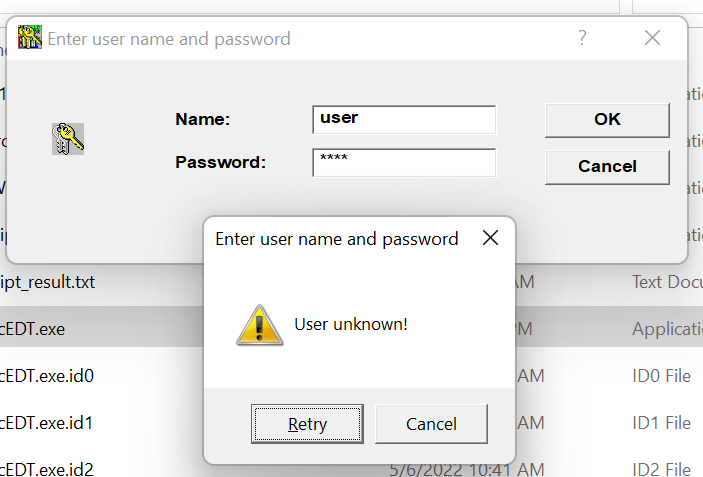
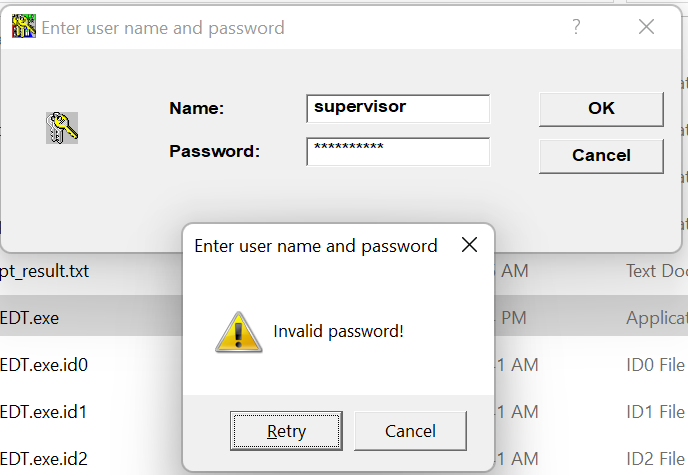
1. **对于SecEDT.exe的部分逆向**

偶然间看到该程序描述为security editor，是一个独立运行的程序，同时似乎可以更改应用的一些安全配置，故想着是去挖掘一下看是否真的会存在一些问题，也是通过发现一些安全配置的脚本.txt，之后发现里面有一个supervisor的用户名，而且经过验证，该用户的确存在：

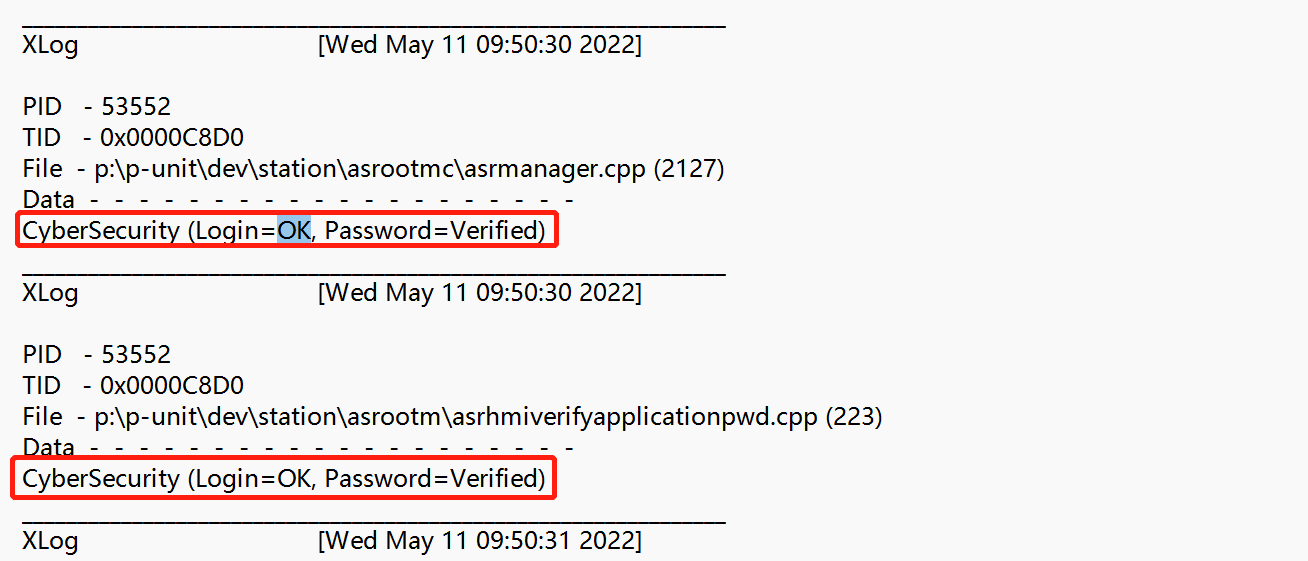
 

在IDA下对于该程序进行反编译，最后通过查找，在sub\_4617C0中找到了用户名片段，但是目前并不知道有什么作用，也在一步步的调试过程中然后逐渐发现问题：

但是这里始终没有找到具体的内容在哪？😭

1. **对于Control\_expert.exe的逆向**
   1. 对于口令破解的逆向过程

通过log里面得到的结果，有一个看似跟口令相关的记录的结果里面有一些内容需要特别注意：



这一些重要的内容是Control Expert.exe所加载的ASRootM.dll（整个函数名在IDA中为sub\_1000A320）所输出的内容：



而且其实整个验证过程似乎就是针对control expert来说的。

通过log也可以发现，在使用control expert打开有密码的工程的时候其整个过程就是下面的几条记录内容：

**XLog [Wed May 11 09:47:41 2022]**

**PID - 53552**

**TID - 0x0000C8D0**

**File - p:\p-unit\dev\station\stmsdk\istsbasicactions\_opensave.cpp (1470)**

**Data - - - - - - - - - - - - - - - - - - - -**

**Opening E:\M340\TEMPLATE\WITHPASSWORD.STU #打开文件**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**XLog [Wed May 11 09:47:43 2022]**

**PID - 53552**

**TID - 0x0000C8D0**

**File - p:\p-unit\dev\onlinegui\plccontrol\plccomctrl.cpp (7235)**

**Data - - - - - - - - - - - - - - - - - - - -**

**CPLCComCtrl::STGOnStateChangeApp() call to GetAppliInfo() New state: FLGV\_UCM\_APPLI\_BUILT.**

**# 检查好app的状态后开始getappliinfo**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**XLog [Wed May 11 09:47:43 2022]**

**PID - 53552**

**TID - 0x0000C8D0**

**File - p:\p-unit\dev\onlinegui\plccontrol\plccomctrl.cpp (8065)**

**Data - - - - - - - - - - - - - - - - - - - -**

**CPLCComCtrl::GetAppliInfo() called. # 检查工程的info**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**XLog [Wed May 11 09:50:30 2022]**

**PID - 53552**

**TID - 0x0000C8D0**

**File - p:\p-unit\dev\station\asrootmc\asrmanager.cpp (2127)**

**Data - - - - - - - - - - - - - - - - - - - -**

**CyberSecurity (Login=OK, Password=Verified) # 输入口令验证**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**XLog [Wed May 11 09:50:30 2022]**

**PID - 53552**

**TID - 0x0000C8D0**

**File - p:\p-unit\dev\station\asrootm\asrhmiverifyapplicationpwd.cpp (223)**

**Data - - - - - - - - - - - - - - - - - - - -**

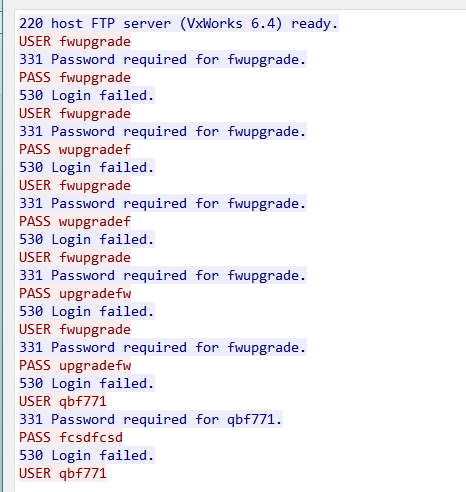
**CyberSecurity (Login=OK, Password=Verified)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

但是如果是回到一般过程，针对PLC中有密码的工程，会记录下怎样的结果呢？

1. **对于OSloader的一些研究**

OSloader是用于PLC固件更新的工具，但是由于其对于整个过程中应该是会有严格的把控，所以这里不好处理，但是对于里面protocol select时，我使用过是FTP模式，但是不知道为啥，总感觉这个OSloader有点老旧，因为在进行固件选择更新时，没有进行FTP账号和口令的选择（**但是这里的确是之前留下的一个问题，口令账号硬编码**）：

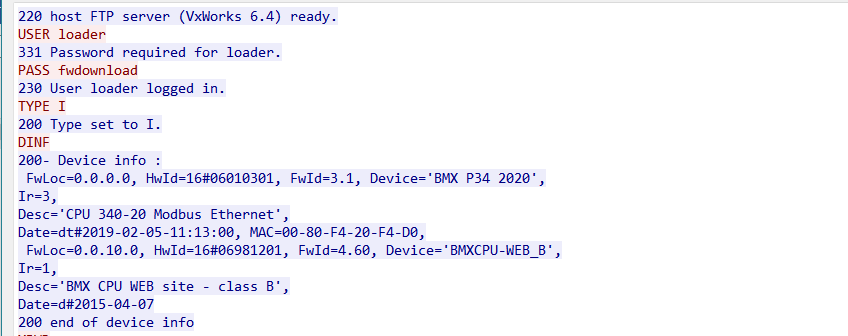


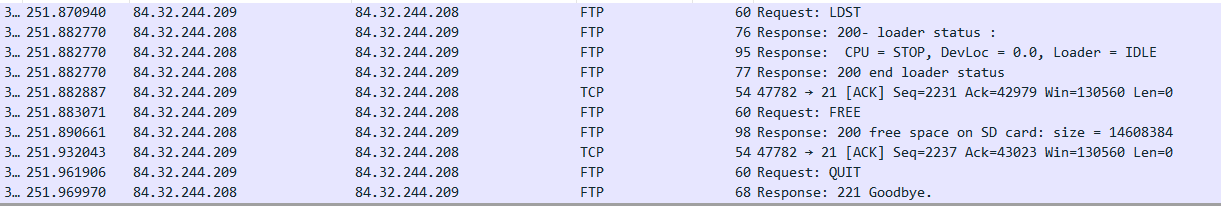
所以这里为了能够到达后面的步骤，使用fiddler或者是Burpsuite进行抓包更改，但是有一个问题就是没有实现这样的结果，两个工具下对于这个都是不允许的。

而这些过程中使用的所有用户名和密码都是在该程序中，硬编码而存在，所以没有太多的内容，**后面经客服确认，该款软件无法给M340刷写固件**。

1. **对于Unity loader的逆向**

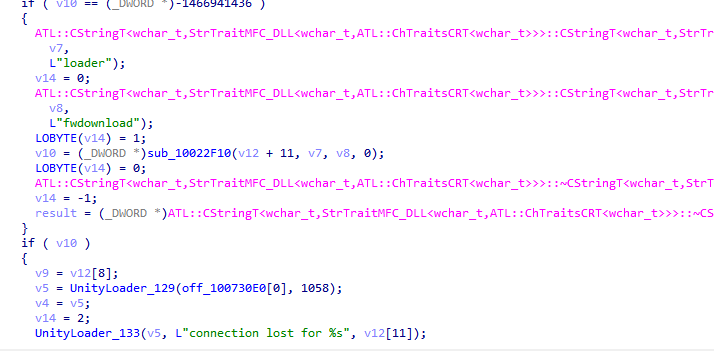
由于control expert的内容真的是过于庞大，所以转换对象，使用Unity loader进行逆向，可以发现，有ftp的口令在对于使用Unity loader的所有设备中都普遍存在：



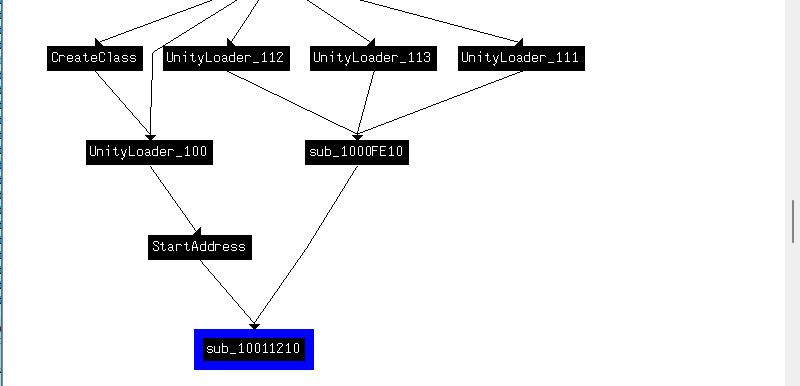


但是也通过这个FTP链接发现了一些新的内容，也许可以用于帮忙解释之前使用modbusTCP抓包的结果报文内容。

找到该FTP连接部分的代码内容：



下面是该函数的调用关系，因为目前在动调过程中只有UnityLoader命名的函数：



对于sub\_100393D0（Func. Code都存在的那个）也有去查找在发包过程中进行的选择，但是由于其函数的调用关系太复杂，所以这里不进行过多的探讨，同时由于对于整个协议的理解要求其实并没有那么高。

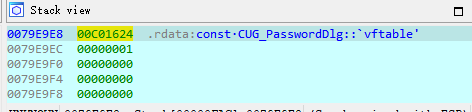
然后Unity\_100就是实现modbusTCP连接建立的位置，这一步之后会进行抓包得到下面的一串base64编码的内容：

fw3jTmgCty8=

dxCEGHWjlUdEz2BucUD2EKqaREYosuVCDT4Okqgslt4=，所以可以证明这个也许跟口令本身没有太多的关联性。

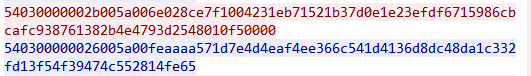
同时该response报文是由query：00200114000c0100003d00（读取特定的存储块）请求得到的：

然后进入口令验证的过程：



这是跳出来的弹窗里面应该显示的内容字段，但是看似这些内容都是经过处理过的，所以不那么好识别，也就不太好破解。（下一次找的时候需要盯紧这一块的.rdata字段的内容，然后看看提前下断点，知道在哪）

然后在原来的过程中发现还有一组base64的编码不见了，工程密码验证通过同时还成功了，但是也发现了部分不一样的内容状态：



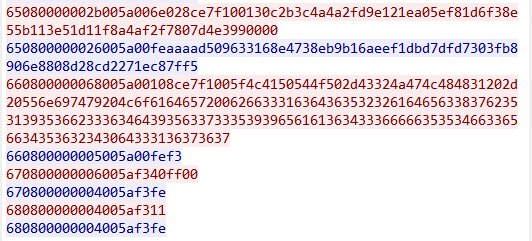
该功能码也是在之前的过程中没有发现的功能码。

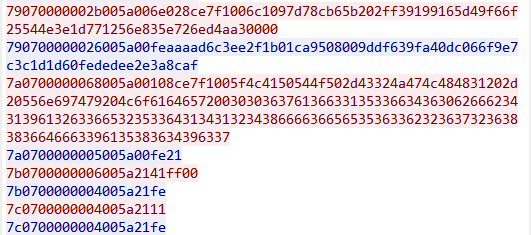
重复一遍：

54030000002b005a006e028ce7f1004231eb71521b37d0e1e23efdf6715986cbcafc938761382b4e4793d2548010f50000

540300000026005a00feaaaa571d7e4d4eaf4ee366c541d4136d8dc48da1c332fd13f54f39474c552814fe65

但是有一个非常大的问题跟原来的过程中不同的是，目前好像没有抓到之前功能码为启动的报文，就是一个报文似乎就启动了。





同样的，固件更新之后设备里面的工程是不变的，还可以继续使用，这一点也是比较有吸引力的。

1. **Something Else**

在Unity Loader目录下，从Unityloader.dll中获得sub\_100393D0，再次验证了之前的功能码的功能

 switch (BYTE1(a4))

    {

    case 1:

        sub\_10036E90(&String, L"get COM info");

        break;

    case 3:

        sub\_10036E90(&String, L"get application info");

        break;

    case 4:

        sub\_10036E90(&String, L"get PLC state");

        break;

    case 0xA:

        sub\_10036E90(&String, L"mirror, data length = %d", v19 - 4);

        break;

    case 0x10:

        sub\_10036E90(&String, L"take reservation");

        break;

    case 0x11:

        sub\_10036E90(&String, L"release reservation");

        break;

    case 0x12:

        sub\_10036E90(&String, L"keep reservation");

        break;

    case 0x20:

        sub\_10036E90(

            &String,

            L"read memory block: BlockId=%d, offset=%d, length=%d",

            \*(unsigned \_\_int16 \*)((char \*)&a4 + 3),

            \*(\_DWORD \*)((char \*)&a4 + 5),

            \*(unsigned \_\_int16 \*)((char \*)&a4 + 9));

        break;

    case 0x21:

        sub\_10036E90(

            &String,

            L"write memory block: BlockId=%d, offset=%d, length=%d",

            \*(unsigned \_\_int16 \*)((char \*)&a4 + 3),

            \*(\_DWORD \*)((char \*)&a4 + 5),

            \*(unsigned \_\_int16 \*)((char \*)&a4 + 9));

        break;

    case 0x24:

        v8 = (int)&a4 + 3;

        v9 = (int)&a4 + 5;

        sub\_10036E90(

            &String,

            L"read var list: type=%d, address=%d, NumberOfElements=%d",

            BYTE4(a4),

            \*(\_DWORD \*)((char \*)&a4 + 5),

            \*(unsigned \_\_int16 \*)((char \*)&a4 + 9));

        break;

    case 0x25:

        v10 = (int)&a4 + 3;

        v11 = (int)&a4 + 5;

        sub\_10036E90(

            &String,

            L"write var list: type=%d, address=%d, NumberOfElements=%d",

            BYTE4(a4),

            \*(\_DWORD \*)((char \*)&a4 + 5),

            \*(unsigned \_\_int16 \*)((char \*)&a4 + 9));

        break;

    case 0x30:

        sub\_10036E90(&String, L"begin download");

        break;

    case 0x31:

        sub\_10036E90(&String, L"download packet, handle=%u, nr=%u, len=%u", WORD1(a4), WORD2(a4), WORD3(a4));

        break;

    case 0x32:

        sub\_10036E90(&String, L"end download");

        break;

    case 0x33:

        sub\_10036E90(&String, L"begin upload");

        break;

    case 0x34:

        sub\_10036E90(&String, L"upload packet, handle=%u, nr=%u", WORD1(a4), WORD2(a4));

        break;

    case 0x35:

        sub\_10036E90(&String, L"end upload");

        break;

    case 0x40:

        sub\_10036E90(&String, L"start PLC");

        break;

    case 0x41:

        sub\_10036E90(&String, L"stop PLC");

        break;

    case 0x53:

        sub\_10036E90(&String, L"get selected blocks: kind=");

        String += wcslen(String);

        v7 = BYTE2(a4);

        if (BYTE2(a4) == 4)

        {

            sub\_10036E90(&String, L"FB data");

        }

        else if (v7 == 6)

        {

            sub\_10036E90(&String, L"unloc data");

        }

        else

        {

            sub\_10036E90(&String, L"%d", BYTE2(a4));

        }

        break;

    case 0xFD:

        sub\_10036E90(

            &String,

            L"negative response: error=%04x(%d), source=%08x(%d), value=%08x(%d)",

            WORD1(a4),

            WORD1(a4),

            DWORD1(a4),

            \*(\_QWORD \*)((char \*)&a4 + 4),

            DWORD2(a4));

        break;

    case 0xFE:

        sub\_10036E90(&String, L"positive response");

        break;

    default:

        break;

    }