

信息工程大学

毕业设计（论文）

题 目：私有工控协议逆向分析和漏洞利用

Research on the Technology of Firmware Attack against

Programmable Logic Controllers

院 别： 网络空间安全学院

专 业： 前哨班

姓 名： 孙康羽

学 号： 3032018011

指导教师： 魏强

日 期： 2022.06.06

摘 要

随着飞速发展的信息技术与工业控制系统的广泛而深入的融合，工控系统逐步从封闭转向开放。然而在信息技术赋能工业，大大提高生产效率的同时，也给工业控制系统带来了前所未有的网络安全挑战。当前，国内外均面临着严峻的工业互联网安全形势。最鲜活的例子就是，今年5月份，美国最大油气管道运营商Colonial Pipeline遭到勒索软件定向攻击的事件。

广泛应用于石油化工、冶金、水利、电力等行业的可编程逻辑控制器(PLC)，是工业控制系统的重要监控终端，遍布整个工业系统，监视并控制着管控域内的各种工业设备，直接影响着工业生产的过程和结果。也正因此，PLC成为了攻击者的重要目标，面临着巨大的安全威胁。随着攻击技术的复杂化、高级化，针对PLC的攻击焦点也逐步由上层软件向底层的固件转移。这也是本文将PLC固件攻击技术作为主要研究对象的出发点。

首先，本文依据McMinn et al提出的分层方法，从编程层、固件层和硬件层三个层面对PLC的软硬件结构进行了简单介绍。同时对PLC的运行状态和工作原理进行了简要时说明。此外还对实战攻击目标Na400 PLC进行了简明的介绍。重点介绍了编程层的上位机编程软件NaPro、固件层的操作系统vxWorks以及硬件层的模块组成，为下文详细阐述实战攻击流程奠定基础。

接着，本文总结梳理了，当前国内外针对可编程逻辑控制器固件的两大攻击技术：PLC固件篡改和伪造攻击技术和PLC固件漏洞挖掘及利用攻击技术。针对固件篡改和伪造攻击技术，本文详尽地阐述了该攻击技术的完整流程，包括固件获取、二进制分析、反汇编、校验算法推导以及篡改并实施攻击这五大步骤。针对固件漏洞挖掘及利用技术，本文重点对模糊测试、同源性分析以及符号执行这三种漏洞挖掘技术在PLC固件安全领域的应用和最新发展进行了梳理总结。在此基础上，针对现有攻击技术存在的局限，本文提出了一种新的以上位机编程软件为切入点的PLC固件漏洞挖掘及利用技术。

接下来，本文详细地阐述了运用本文提出的新攻击技术，针对Na400 PLC的系统固件vxWorks实施漏洞挖掘及利用攻击的过程。详细记录了，在固件获取阶段，利用通信流量分析和逆向分析技术，获取NaPro硬编码的FTP特权用户名和口令的过程、在固件逆向及漏洞挖掘阶段，利用binwalk和IDA等工具，定位关键函数并挖掘出栈溢出漏洞的过程以及在漏洞利用阶段，明确攻击目标并以此为指引，深入分析vxWorks用户认证机制，进而在此基础上利用栈溢出漏洞的特性，设计攻击载荷并成功实现FTP特权用户口令篡改攻击的过程。

最后，对全文工作进行总结，并对下一步有价值的研究方向进行了展望。

关键词：工业控制器，可编程逻辑控制器，固件篡改，固件漏洞挖掘

Abstract

With the extensive and in-depth integration with the information technology, industrial control systems have gradually shifted from closed to open. However, while information technology empowers industry and greatly improves production efficiency, it has also brought unprecedented network security challenges to industrial control systems. At present, both at home and abroad are facing a severe industrial Internet security situation. The most vivid example is the incident in which Colonial Pipeline, the largest US oil and gas pipeline operator, was targeted by ransomware in May of this year.

Programmable logic controller (PLC), which is widely used in petrochemical, metallurgy, water conservancy, electric power and other industries, is an important monitoring terminal for industrial control systems, covering the entire industrial system, monitoring and controlling various industrial equipment in the control domain.PLC can directly affect the process and result of the industrial production. For this reason, PLC has become an important target of attackers and faces huge security threats. As attack techniques become more complex and advanced, the focus of attacks on PLCs has gradually shifted from upper-layer software to lower-layer firmware. This is also the starting point for this article to take PLC firmware attack technology as the main research object.

First of all, according to the layered method proposed by McMinn et al, this article briefly introduces the PLC's software and hardware structure from three levels: programming layer, firmware layer and hardware layer. At the same time, the operating status and working principle of the PLC are briefly explained.Besides, this article gives a concise introduction to the actual combat target Na400 PLC. It focuses on its hardware system architecture, host computer programming software NaPro, and system firmware vxWorks, laying the foundation for the detailed description of the actual attack process below.

Next, this article summarizes and sorts out the two major attack techniques against programmable logic controller firmware at home and abroad: PLC firmware tampering and forgery attack techniques and PLC firmware vulnerability mining and exploiting attack techniques. Regarding firmware tampering and forgery attack techniques, this article elaborates the complete process of the attack technique in detail, including the five steps of firmware acquisition, binary analysis, disassembly, verification algorithm derivation, and tampering and attacking. Aiming at the firmware vulnerability mining and utilization technology, this article focuses on the application and latest development of the three vulnerability mining technologies, fuzzing, homology analysis, and symbolic execution, in the field of PLC firmware security. On this basis, in view of the limitations of the existing attack technology, this paper proposes a new PLC firmware vulnerability mining and utilization technology with the upper computer programming software as the entry point.

Next, this article elaborates the process of comprehensively using the attack technology studied in this article to implement vulnerability mining and exploiting attacks against the system firmware vxWorks of Na400 PLC. In the firmware acquisition phase, the process of using communication traffic analysis and reverse analysis technology to obtain the FTP privileged user name and password hard-coded in NaPro is recorded in detail. In the firmware reverse engineering and vulnerability mining phase, tools such as binwalk and IDA are used to locate key functions and dig out stack overflow vulnerability. During the exploit stage, the author determines the target of the attack and deeply analyzes the vxWorks’ user authentication mechanism, and on this basis, designs the attack payloads and successfully tampers with the FTP privileged user’s password.

Finally, the author summarizes the work of the full text and looks forward to the next valuable research directions.

Keywords: industrial controller, programmable logic controller, firmware tampering, firmware vulnerability mining

目录

[第一章 绪论 9](#_Toc74740387)

[1.1 研究背景及意义 9](#_Toc74740388)

[1.2 国内外研究现状 10](#_Toc74740389)

[1.3 主要研究内容 11](#_Toc74740390)

[1.4 本文结构安排 12](#_Toc74740391)

[第二章 PLC简介 13](#_Toc74740392)

[2.1 PLC软硬件结构及其工作原理 13](#_Toc74740393)

[2.2 实战目标Na400 PLC简介 15](#_Toc74740394)

[2.3 本章小结 19](#_Toc74740395)

[第三章 PLC固件攻击技术综述 20](#_Toc74740396)

[3.1 固件篡改和伪造攻击技术分析 20](#_Toc74740397)

[3.2 固件漏洞挖掘及利用攻击技术分析 22](#_Toc74740398)

[3.3 新的攻击思路 27](#_Toc74740399)

[3.4 本章小结 28](#_Toc74740400)

[第四章 PLC固件漏洞挖掘及利用实践 30](#_Toc74740401)

[4.1 固件获取 30](#_Toc74740402)

[4.2 固件逆向及漏洞挖掘 32](#_Toc74740403)

[4.3 漏洞利用与攻击测试 36](#_Toc74740404)

[4.4 本章小结 44](#_Toc74740405)

[第五章 总结与展望 45](#_Toc74740406)

[5.1 全文总结 45](#_Toc74740407)

[5.2 工作展望 46](#_Toc74740408)

[致 谢 47](#_Toc74740409)

[参考文献 48](#_Toc74740410)

第一章 绪论

1.1 研究背景及意义

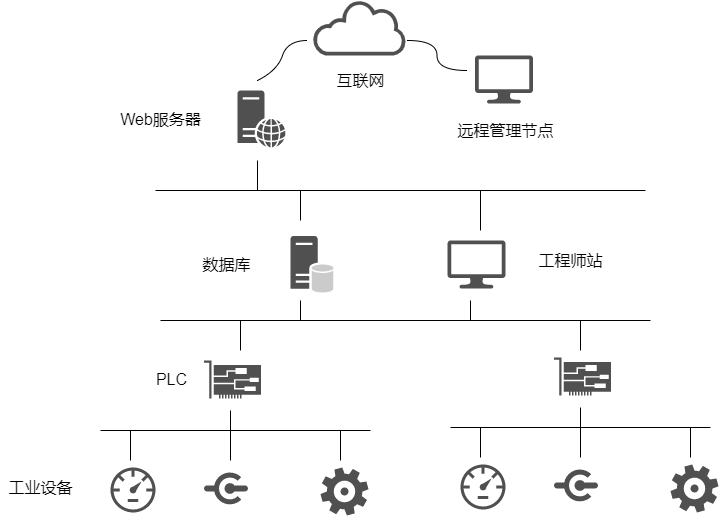
工业控制系统(Industrial Control System, ICS)由各种自动化控制组件和实时数据采集、监测的过程控制组件共同构成，其主要组件包括监控与数据采集系统(Supervisory Control And Data Acquisition，SCADA)、分布式控制系统(Distributed Control System，DCS)、可编程逻辑控制器(Programmable Logic Controller，PLC)、远程终端(Remote Terminal Unit， RTU)、智能电子设备(Intelligent Electronic Device，IED)，以及确保各组件通信的接口技术[1]。

图1.1 工业控制系统基本架构

随着信息技术的发展，工业控制系统正在被广泛运用于能源、军工、交通水利、市政等关键领域，工业控制系统的信息安全直接关系到国计民生的安全与稳定。

现阶段，工业控制系统在协议、软件、硬件层存在很多安全问题。一方面工业控制系统采用实时数据交换，依赖网络通信，为远程攻击提供了途径。且早期工业控制系统大多独立于互联网，因此很少考虑安全问题，造成通信协议的设计缺乏认证、加密等安全机制，容易留下安全隐患。另一方面，进入工业4.0时代，工业控制系统普及的同时，为了方便维护和管理，工控系统大多采用标准化的解决方案，一旦出现安全问题，影响范围将十分广泛。除此之外，工业控制系统的漏洞修复成本高，修复周期长。美国知名网络安全公司FireEye近期发布的一份名为《Overload：Critical Lessons From 15 Years of ICS Vulnerabilities 》的调查报告指出，仅仅在2016年的一年时间里，全球在工控系统上就曝出了400个0day漏洞，而其中的100余个漏洞至今仍未得到修复[2]。工控系统漏洞长时间的存在也进一步增加了工控系统安全问题的严重性。2010年爆发的“震网病毒”， 在导致伊朗核计划推迟数年的同时，也给工业控制系统安全敲响了警钟[3]。2015年恶意软件“BlackEnergy”造成乌克兰首都基辅部分地区和乌克兰西部的140万名居民停电[4]。近期欧盟通过的NIS(Network Information Service)条例，以及在波兰召开的北约峰会(Warsaw NATO summit)都再一次强调了维护工业系统网络安全的重要性，进一步提高安全级别，防止工业系统遭到黑客攻击。我国在工控安全问题上同样十分重视。2016年10月，工业和信息化部印发《工业控制系统信息安全防护指南》，指导工业企业开展工控安全防护工作[5]。

PLC作为整个SCADA和DCS系统中使用的控制系统组件，广泛应用于几乎所有的工业过程，也是针对工业控制系统攻击中常见的攻击目标，因此针对PLC的研究分析在工控安全领域就显得尤为重要。本课题将分析PLC设备协议和固件层面的原理，并对施耐德M241结合其配套控制软件Machine Expert进行协议上的逆向分析，并尝试进行漏洞挖掘及利用。通过本课题的研究，为的PLC协议分析提供更多的案例支撑，为后续工业控制系统的攻击与安全防范方面的研究打下基础。

1.2 国内外研究现状

当前，针对PLC的协议攻击方式主要有破坏可用性、破坏完整性和破坏机密性三个方面[7]。

破坏可用性是是指通过攻击手段使PLC无法正常工作，常用方法有利用协议存在的漏洞引发程序崩溃，或发送大量垃圾数据消耗PLC的计算资源使其无法处理正常业务。例如思科公司的JaredRittle[6]发现了一个应用于Schneider Electric Modicon M580 SV2.80 PLC的UMAS协议漏洞。攻击者向目标设备发送一个精心构造的UMAS协议数据包，引起目标设备无法进行远程通信，导致拒绝服务。

破坏完整性是指通过中间人攻击、报文重放攻击等方式，改变PLC正常工作流程从而造成破坏。具有代表性的攻击事件是破坏伊朗核设施的“震网”Stuxnet事件。此类攻击方式虽然没有破坏PLC的功能，但通过改变PLC的工作流程同样给工业生产造成巨大威胁。除此之外，此类攻击方式隐蔽性强，并且攻击者更能掌控攻击的效果，因此更具实用性。

破坏机密性是指通过利用协议漏洞主动请求或者监听通信非法获取系统的敏感信息。由于PLC传输的安全措施不强，现阶段PLC的通信数据大多以明文形式传输，为这种攻击方式提供了便利。虽然获取系统敏感信息并不会对工控系统造成直接的破坏，但为位下一步的攻击提供相关信息，是PLC协议攻击中的关键一步。

针对PLC固件的主流攻击手段有固件漏洞挖掘利用以及固件修改攻击。

在固件漏洞方面类似于传统软件漏洞挖掘利用，学术界和业界的研究较为成熟。而由于工业控制现场实时性的要求, PLC采用的操作系统大多是经过裁剪的实时操作系统(RTOS)。嵌入式系统历来都会使用大量专有组件, 并没有与其他系统共享太多共同电路, 这意味着当发现漏洞时，由于成本或资源限制，漏洞修复难度较大。而且，PLC 的系统更新一般只能由生产厂商完成，因此其打补丁的过程就更加缓慢。

固件篡改攻击是一类针对嵌入式系统固件更新特点的攻击方法，主要利用固件更新过程中的验证不充分。攻击者可以利用固件篡改漏洞和其它漏洞组合，对冗余控制器架构的工业控制系统进行隐蔽攻击。

1.3 主要研究内容

基于上述研究背景和现状，本文主要完成了以下工作：

（一）依据McMinn et al提出的分层方法，从编程层、固件层和硬件层三个层面对PLC的软硬件结构进行了简单介绍。同时对PLC的运行状态和工作原理进行了简单的说明。对本文的实战攻击目标Na400 PLC进行了简明的介绍。重点介绍了其编程层的上位机编程软件NaPro、固件层的操作系统vxWorks以及硬件层的模块组成，为下文详细阐述实战攻击流程奠定基础。

（二）总结梳理了，当前国内外针对可编程逻辑控制器固件的两大攻击技术：PLC固件篡改和伪造攻击技术和PLC固件漏洞挖掘及利用攻击技术。针对固件篡改和伪造攻击技术，本文详尽地阐述了该攻击技术的完整流程。对固件获取、二进制分析、反汇编、校验算法推导以及篡改和上传运行这五个阶段中涉及的具体技术记性了深入分析。针对固件漏洞挖掘及利用技术，本文重点对模糊测试、同源性分析以及符号执行这三种漏洞挖掘技术在PLC固件安全领域的应用和最新发展进行了梳理总结。

（三）基于对PLC软硬件结构、工作原理以及当前针对PLC固件的攻击技术的研究，本文针对现有攻击技术存在的局限，提出了一种能够有所突破的新攻击思路，即在厂商不提供固件在线下载的情况下，以上位机编程软件为切入点、以固件逆向分析技术为基本支撑、以固件通信功能模块为挖掘核心、以危险函数检索和分析为突破口，实现PLC固件获取并实现高效漏洞挖掘及利用的攻击方法。

（四）详细地阐述了运用本文提出的新攻击方法，针对Na400 PLC实施固件漏洞挖掘及利用攻击的过程。详细记录了，在固件获取阶段，如何以NaPro为切入点，通过通信流量分析和逆向分析，成功获取硬编码的FTP特权用户名和口令、在固件逆向及漏洞挖掘阶段，如何利用binwalk和IDA等工具，定位关键函数并挖掘出栈溢出漏洞以及在漏洞利用阶段，明确攻击目标并以此为指引，深入分析vxWorks用户认证机制，进而在此基础上利用栈溢出漏洞的特点设计攻击载荷并成功实现FTP特权用户口令篡改攻击的过程。证明了本文提出的新攻击思路的有效性。

1.4 本文结构安排

本文共五章，各章内容安排如下：

第一章，明确了本文的研究背景、意义、现状和主要内容。首先从工业互联网的发展趋势和国内外严峻的安全形势以及可编程逻辑控制器的在工业控制系统的重要地位、面临的安全威胁和攻防焦点，阐述了针对PLC固件攻击技术研究的背景和意义。接着，第一章还简要地总结梳理了当前国内外的一些研究成果。最后在此基础上，详细阐述了本文的主要研究内容。

第二章，从硬件系统结构、上位机编程软件以及系统固件三个角度，对本课题的实战攻击对象国产可编程逻辑控制器南大傲拓Na400 PLC进行了简单的介绍，为第四章实战攻击过程的介绍奠定基础。

第三章，对当前国内外针对PLC固件的两大攻击技术，固件篡改和伪造攻击和固件漏洞挖掘及利用，进行了梳理总结，对涉及的攻击技术的原理和实施流程进行了详细分析说明。并提出了一种能够突破现有攻击技术局限的新攻击思路。

第四章，详细记录了运用本文提出的新攻击方法，对Na400 PLC进行固件漏洞及利用的攻击过程，对漏洞原理进行了深入分析，并对攻击思路和攻击载荷的设计与测试进行了详细的说明。

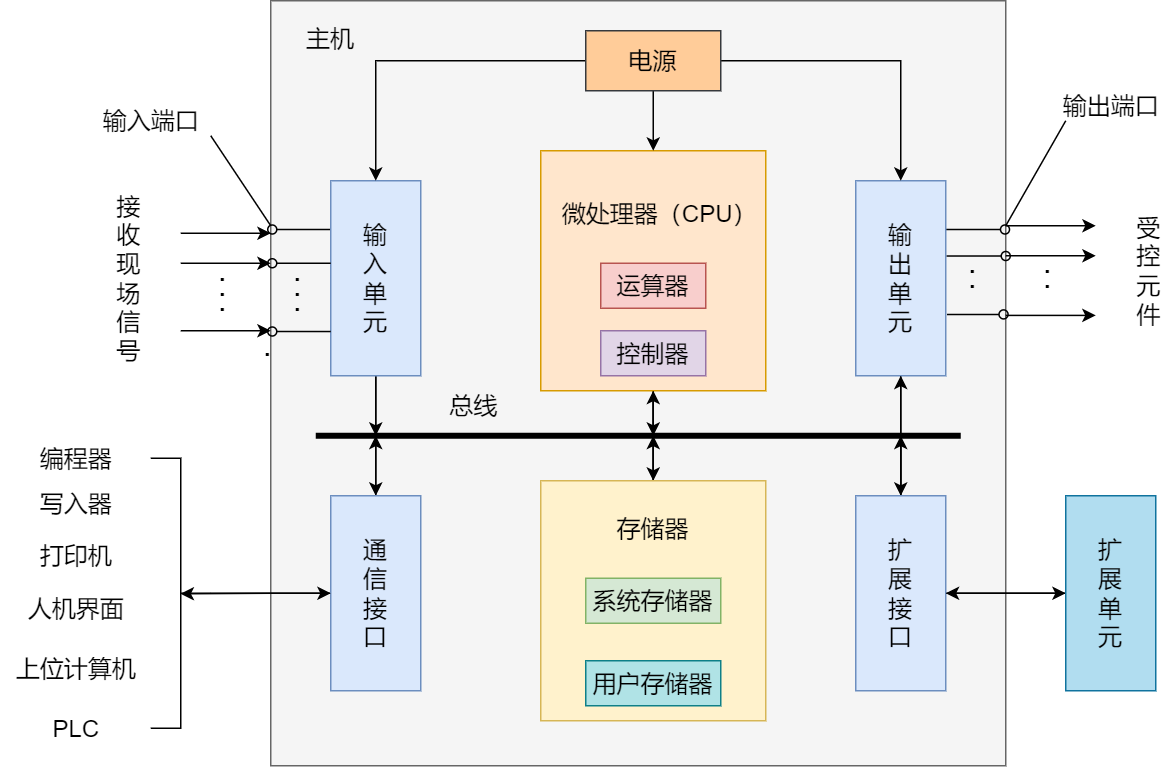
第五章，对全文的工作进行了总结，并指出了后续有进一步有研究价值的方向。

第二章 PLC简介

PLC是整个SCADA和DCS系统中使用的控制系统组件。

PLC种类繁多，功能虽然多种多样，但其组成结构和工作原理基本相同。PLC的控制系统由硬件和软件两大部分组成，硬件指PLC本身及其外围设备，软件指管理PLC的软件系统和PLC的应用程序。

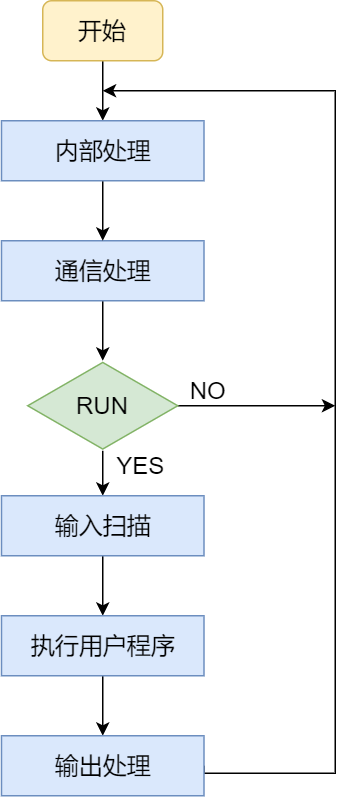
PLC实质上是一种工业控制计算机，PLC与计算机的组成十分相似。只不过它比一般的计算机具有更强的与工业过程相连接的接口，以及更直接的适应控制要求的编程语言。从硬件结构看，它由CPU、存储器、输入/输出接口、电源等组成。PLC的硬件基本组成框图如图1所示。

PLC硬件的基本组成框图

其中CPU一般由控制器、运算器和寄存器组成，这些电路都集成在一个芯片内。CPU通过数据总线、地址总线和控制总线与存储单元、输入输出接口电路相连。CPU作为PLC硬件的核心主要完成的工有执行用户程序，完成各种数据处理、传输和存储，并相应相应的控制信号，以完成用户指令规定的各种操作。

PLC的存储器主要用于存放系统程序、用户程序以及工作状态数据。PLC的存储器包括系统存储器和用户存储器。系统存储器用来存放由PLC生产厂家编写的系统程序，并固化在ROM内，用户不能更改。它使PLC具有基本的功能，能够完成PLC设计者规定的各项工作。用户存储器包括用户程序存储器（程序区）和数据存储器（数据区）两部分。用户程序存储器用来存放用户针对具体控制任务采用PLC编程语言编写的各种用户程序。用户程序存储器根据所选用的存储器单元类型可以是RAM、EPROM或EEPROM存储器，其内容可以由用户修改或增删。用户数据存储器可以用来存放用户程序中所使用器件的ON/OFF状态和数据等。

输入/输出接口是PLC与现场I/O设备或其它外部设备之间的连接部件。PLC通过输入接口把外部设备(如开关、按钮、传感器)的状态或信息读入CPU，通过用户程序的运算与操作，把结果通过输出接口传递给执行机构。

PLC控制系统的软件主要包括系统软件和用户程序。系统软件由PLC厂家固化在存储器中，用于控制PLC的运作。用户程序由使用者编制录入，保存在用户存储器中，用于控制外部对象的运行。系统软件包括系统管理程序、用户指令解释程序、标准程序模块及系统调用。用户程序即应用程序，是用户针对具体控制对象编制的程序。PLC是通过在RUN方式下，循环扫描执行用户程序来完成控制任务的，用户程序决定了一个控制系统的功能。一个完整的用户程序应当包含一个主程序、若干子程序和若干中断程序三大部分。

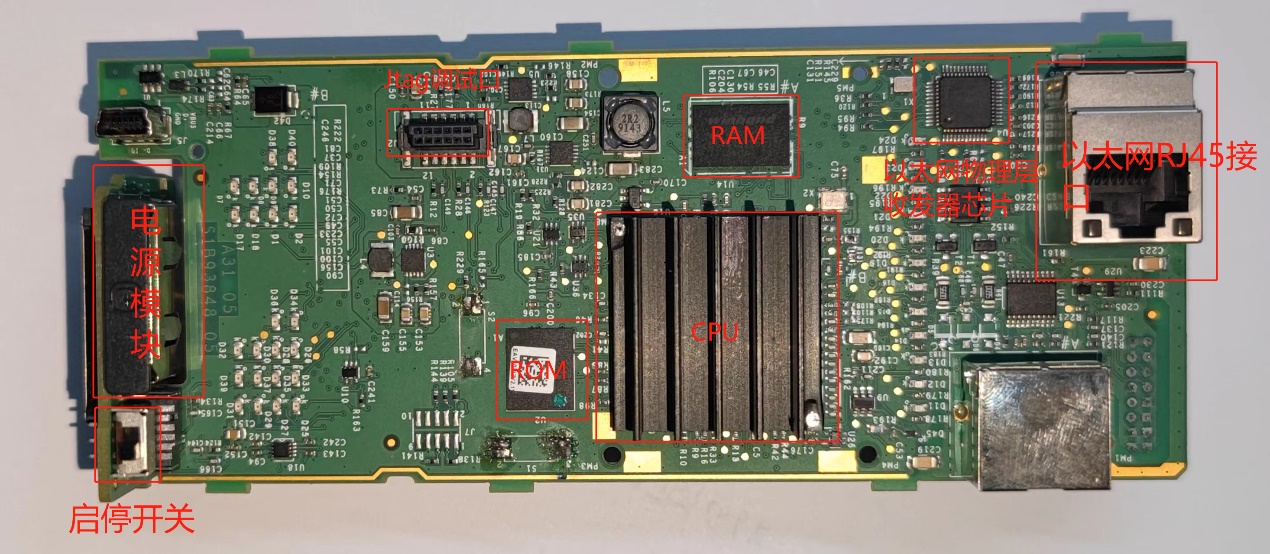
PLC的循环扫描

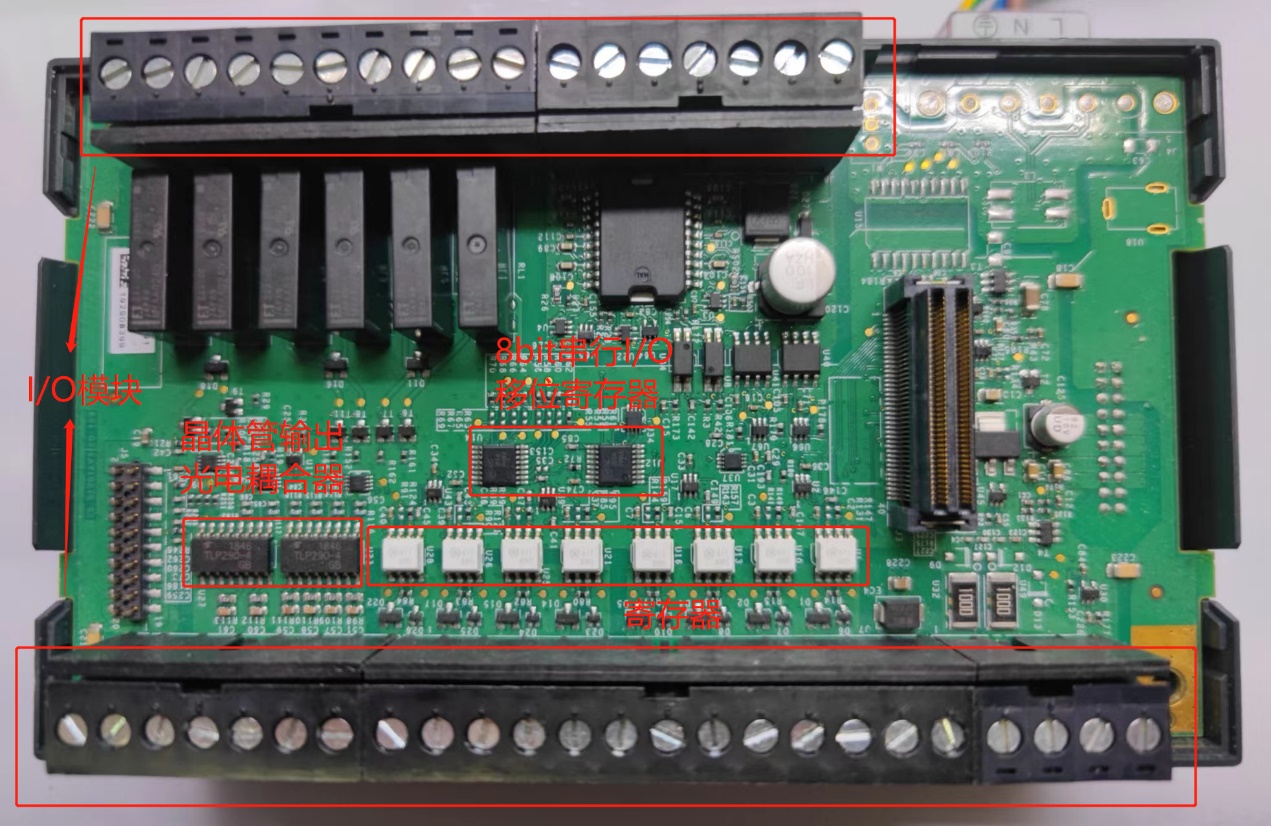
2.2 施耐德M241简介

本节主要介绍了本文的分析的具体PLC实例施耐德M241CE24R。是施耐德M241系列的其中一款产品。施耐德M241是由施耐德公司推出的一款基于SoMachine V4平台的高新能一体式可编程控制器系列。这一系列适用于具有速度控制和位置控制的高性能一体型设备，具有通信强大，定位控制，丰富扩展等优势。接下来将从硬件和软件两方面对其进行介绍。

2.2.1 硬件部分

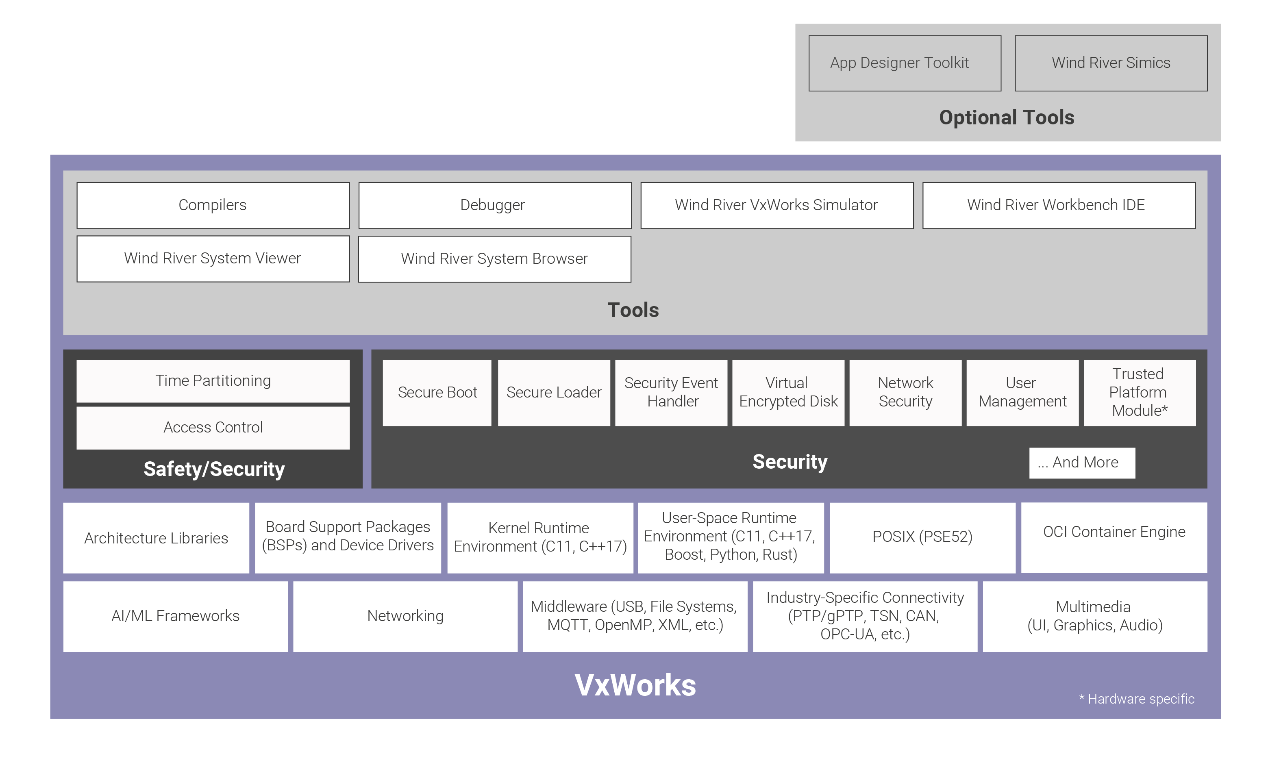
M241采用双层背板设计，这种设计方式将M241硬件上不同的功能分隔开，使得M241更加模块化，便于维修保养。

上层背板主要是CPU，存储器，通信接口，扩展接口等。

下层背板主要是输入输出单元，以及控制指示灯的相关寄存器等。

2.2.1 软件部分

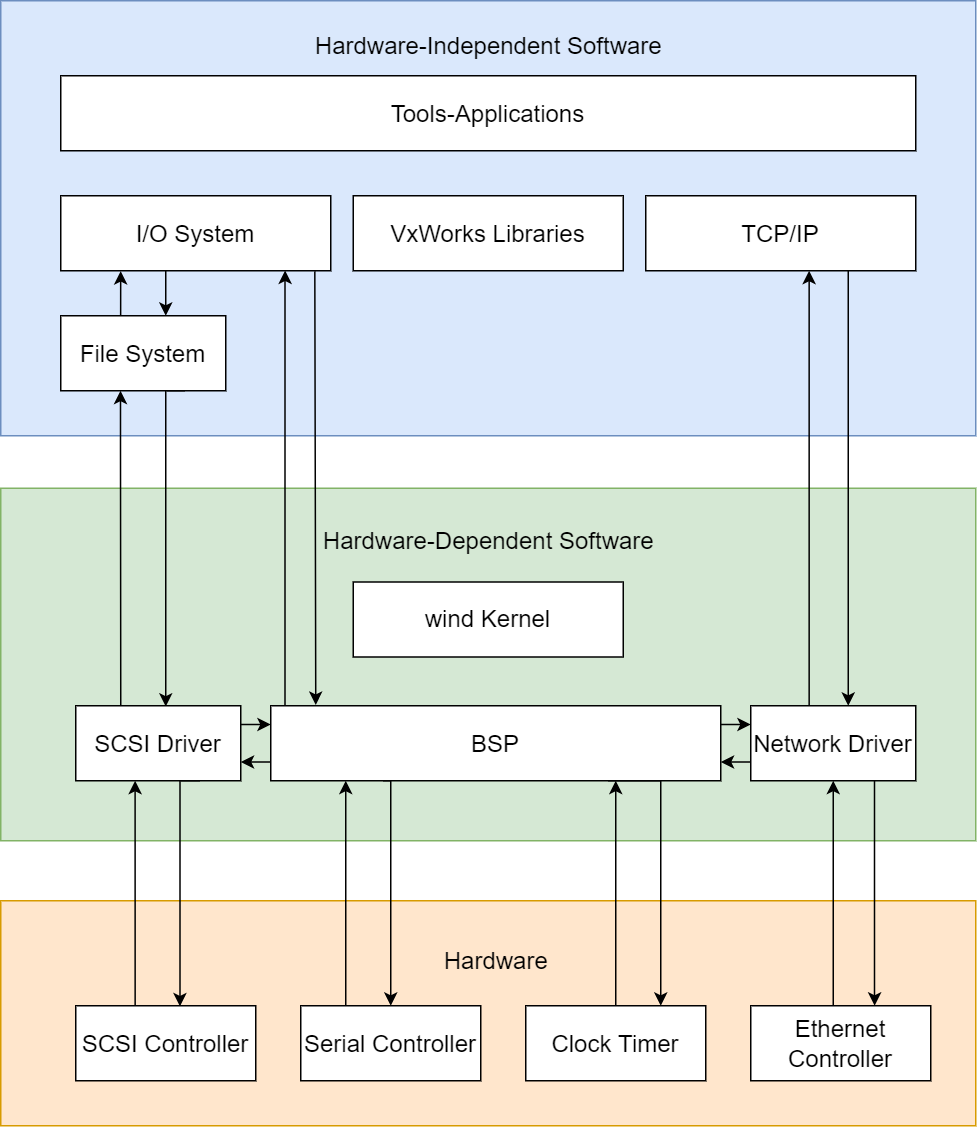
2.2.2.1 VxWorks操作系统

M241固件搭载了VxWorks操作系统。VxWorks 操作系统是美国WindRiver公司于1983年设计开发的一种嵌入式实时操作系统（RTOS），是嵌入式开发环境的关键组成部分。作为全球首个且唯一一个利用容器部署应用程序的实时操作系统，最新版VxWorks支持OCI容器，利用类似IT的传统技术，即可更高效更优质地开发和部署智能边缘软件，且不会影响确定性和性能。VxWorks是一款确定性、基于优先级的抢占式实时操作系统，具有超低延迟和最小抖动。VxWorks采用可升级架构具有很高的可扩展性。VxWorks是唯一支持C ++ 17、Boost、Rust、Python、pandas等开发语言的实时操作系统，搭载边缘优化和OCI兼容的容器引擎，便于灵活选择开发语言、工具和技术。

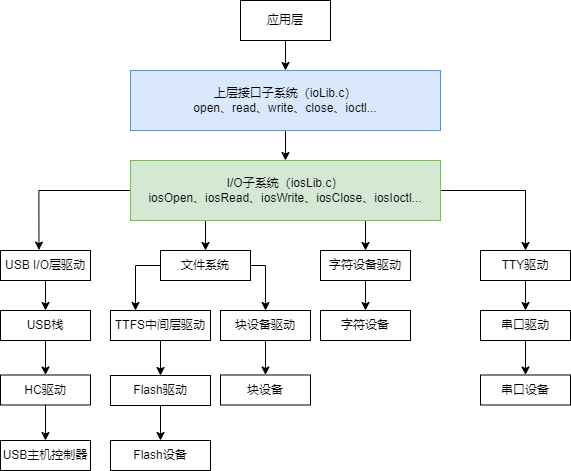
VxWorks部分组件的图表

VxWorks操作系统以其良好的持续发展能力、高性能的内核以及友好的用户开发环境，在嵌入式实时操作系统领域占据一席之地。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通讯、军事演习、弹道制导、飞机导航等。

VxWorks操作系统可以分为内核和系统运行库两部分，是一个典型的微内核结构，其架构如下图所示。

在内核部分，VxWorks拥有一个被称为Wind的高性能微内核。这一设计不仅减小了内核的体积，同时也提高了内核的可扩展性。为了提高VxWorks系统的兼容性，以适应不同的硬件环境，Wind内核内置了板级支持包BSP(Board Support Package)。板级支持包向VxWorks操作系统提供了对各种板子的硬件功能操作的统一的软件接口，它是保证VxWorks操作系统可移植性的关键，它包括硬件初始化、中断的产生和处理、硬件时钟和计时器管理、局域和总线内存地址映射、内存分配等等。每个板级支持包包括一个 ROM 启动（Boot ROM）或其它启动机制。像其它的操作系统一样，VxWorks操作系统内核支持多任务调度（采用基于优先级抢占方式，同时支持同优先级任务间的分时间片调度），任务间的同步，进程间通信机制，中断处理，定时器和内存管理机制等功能。

在内核的上一层是系统运行库，包括文件系统，I/O系统，网络模块等各种模块。

VxWorks的I/O系统分为上层接口子系统(ioLib.c)和I/O子系统(iosLib.c)两部分。上层接口子系统将系统的I/O接口封装成UNIX 标准的Basic I/O接口，是用户程序直接调用的部分。而I/O 子系统作为上层接口子系统与下层驱动系统的中间层而存在则一般对用户不可见。从图2中可以看出：I/O 子系统在整个驱动层次中起着十分重要的作用，其对下管理着各种类型的设备驱动。换句话说，各种类型（包括网络设备）的设备都必须向I/O 子系统进行注册方可被内核访问。所以在I/O 子系统这一层次，内核维护着三个十分关键的数组用以对设备所属驱动、设备本身以及当前系统文件句柄进行管理。

VxWorks文件系统在内核驱动层次中实际上是作为块设备驱动层次中的一个中间层而存在的，其向I/O 子系统进行注册，而将底层块设备驱动置于自身的管理之下以提高数据访问的效率。在这些文件系统中，dosFs 和rawFs 是最常用的两种文件系统类型，在VxWorks早期版本就包含对这两种文件系统的支持。

VxWorks的网络系统提供了对其它VxWorks系统和TCP/IP网络系统的"透明"访问，包括与BSD套接字兼容的编程接口，远程过程调用（RPC），SNMP（可选项），远程文件访问（包括客户端和服务端的NFS机制以及使用RSH，FTP 或 TFTP的非NFS机制）以及BOOTP 和代理ARP、DHCP、DNS、OSPF、RIP。无论是松耦合的串行线路、标准的以太网连接还是紧耦合的利用共享内存的背板总线，所有的 VxWorks 网络机制都遵循标准的 Internet 协议。

2.2.2.1 Machine Expert

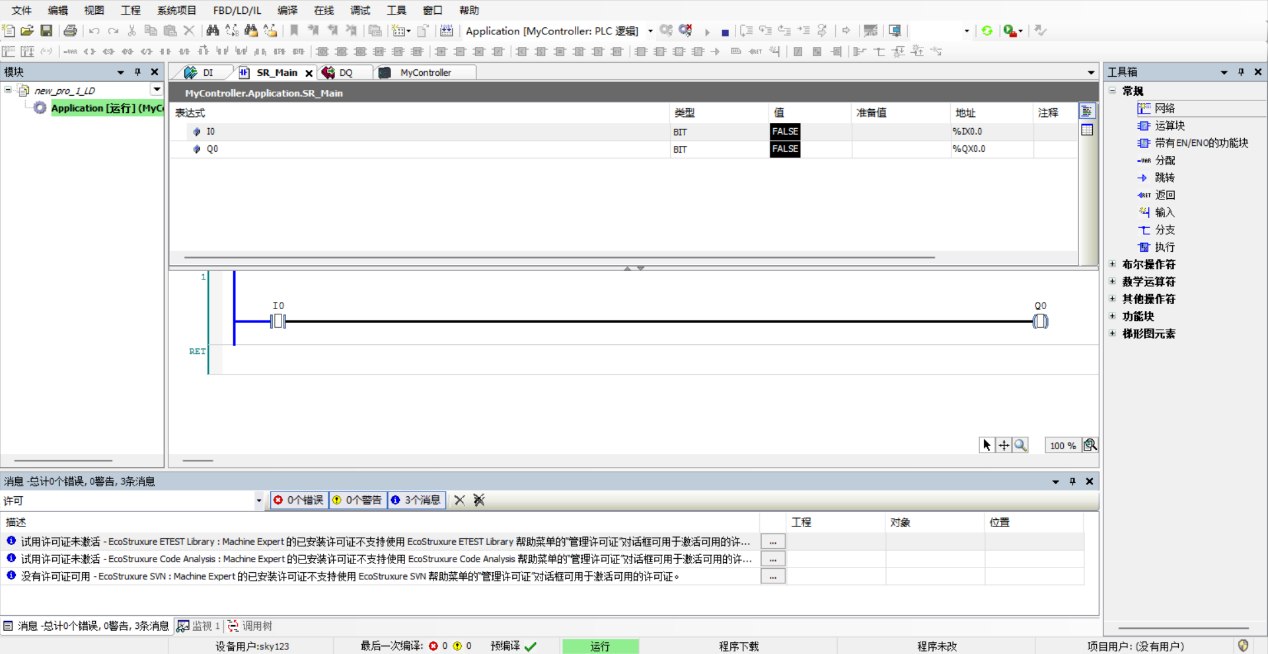
Machine Expert是施耐德M241配套的Machine Expert 是一款独特的解决方案软件，用于在单个软件环境中开发、配置和调试整个机器，包括逻辑、运动控制、HMI 和相关网络自动化功能。

Machine Expert具有以下特征。

第一，Machine Expert支持的开发语言种类丰富，支持梯形逻辑图(LD)，功能快图(FBD)，结构化文本(ST)，连续功能图(CFC)，顺序功能图(SFC)等编程语言。用户可根据自身需求为项目开发选择合适语言。

第二，Machine Expert安全性高。Machine Expert拥有完善的程序异常记录和处理机制，每一次Machine Expert的更新都会即使修复公布的漏洞和发现的程序异常。并且，Machine Expert还会强制用户为项目设置口令，并且提供一整套安全的口令验证机制。另外，Machine Expert对于权限管理也异常严格，对不同权限的操作会严格划分，避免用户出现越权操作。

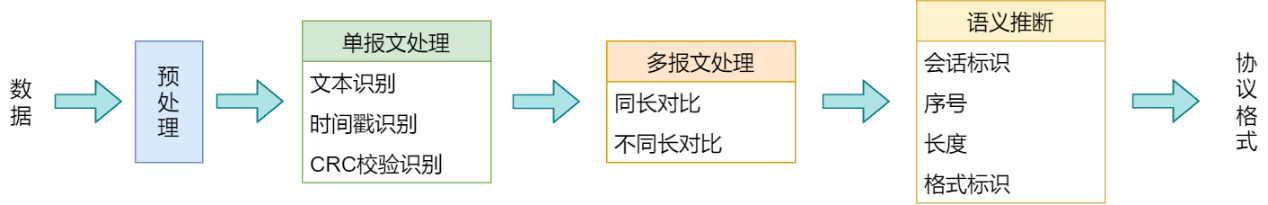
第三，Machine Expert功能完善，软件集成度高。Machine Expert集成了软件开发，调试和配置功能，并且还集成了对PLC的控制功能等。

第四，Machine Expert提供了强大的调试功能，用户可以在联机状态不仅可以下设置断点，单步执行，也可以随时查看变量和停止运行，调试起来非常方便。并且软件还支持固件仿真模拟，可以在脱离硬件的基础上进行软件的开发调试。

2.3 本章小结

第三章 私有工控协议逆向分析技术综述

3.1 基于网络流量私有工控协议逆向分析技术

由于相对于工控协议处理程序，工控网络中的数据流量相对容易获取且容易分析，因此对工控协议的逆向分析多采用基于网络流量的方式。根据工控协议的特点，基于网络流量的工控协议逆向分析过程分为预处理，单报文处理，多报文处理，语义推断四个阶段，具体过程如下图所示。

3.1.1 数据预处理

数据处理即获取工控协议报文数据，并对数据进行清洗处理的过程，主要步骤分为数据包获取，数据包过滤，数据提取三步。

首先数据包获取。通过接入工控网络捕获网络中的通信数据包。这一过程可以借助Wireshark等网络抓包工具实现。

之后是数据包过滤。对于获取的数据包，其中可能包含着其他协议的通信数据包，也可能包含着心跳包等无意义的数据包。这些数据包会对接下来的协议逆向分析产生干扰。可以根据IP地址，端口号，数据包协议和数据包长度等信息对通信数据进行初步筛选。由于工控协议在实际通信过程中可能会存在出现丢包，重传，乱序等问题，因此在数据包过滤中还需要过滤掉不含负载的报文，并丢弃校验和出错的报文。

最后是数据提取。由于私有工控协议大多是应用层协议，不需要考虑底层协议的通信过程，因此在数据预处理过程中需要将自底向上依次取出数据链路层，网络层和传输层的格式封装，得到完整的应用层报文。

3.1.2 单报文处理

单报文处理是指对与上下报文无关的字段进行识别，通过这一处理可以初步识别出报文中的一些字段的含义，并为接下来的多报文处理排除一些干扰。同时，由于同一协议的报文数据格式是固定的，因此单报文处理识别出的字段的位置和格式还有助于划分出不同协议的报文。单报文处理按照识别对象的不同可以分为文本识别，时间戳识别和CRC校验识别等类型。

文本识别是指识别出报文中的文本内容。文本识别的识别方式是筛选出报文中连续的可打印字符，然后将其划分为文本字段。为了避免将其他字段的一些个别字节错误的识别为文本字段，需要设定文本长度的最小阈值。

时间戳识别是指识别出报文中表示时间的字段。时间戳识别的方式是首先选取报文中连续的4B或8B长度的数据，然后根据时间戳(Unix timestamp)的方式进行计算得到时间值，如果该值与捕获报文的时间在一定的误差范围内，则将该字段标识为时间戳字段。

3.1.3 多报文处理

多报文处理是指根据不同报文的比较分析划分出定长和不定长的协议报文字段。由于工控协议往往具有格式相同或相近、传输信息重复度高、同类型报文长度等长等特征，因此可以首先对同长度的协议报文进行对比，初步划分出协议格式，之后再根据协议格式进行分类，对协议格式相似的不同长度报文进行比对，最终得到不同格式样本的子集。

同长度报文比对主要是划分出相同长度报文组中报文内容的不变段与变化段。为了避免将同一初始分类组中长度相同但格式不同的报文进行强行比对,影响比对的准确度,需要设定距离阈值,在距离过大时停止比对,并将之再分类。依据比对结果划分标记变化段和不变段。

不同长报文比对主要是识别出长度不同，但格式相同或相近的报文中内容数据定长段和不定长段。采用时间和空间复杂度都相对比较适合的渐进多序列比对算法。参考PI项目中的渐进比对算法,主要包括3个步骤:(1)采用Smith- Waterman算法找出任意两个样本之间的局部最佳比对,据此计算样本间的相似度,并构造样本集的距离矩阵;(2)采用非加权成对群算术平均法(Unweighted Pair Group Method with Arithmetic means , UPGMA)计算子类间的距离,逐步将距离最小的子类进行合并;(3)执行渐进多序列比对，采用Needleman -Wunsch算法进行双序列动态规划比对,对未对齐的字节进行补充。为减少算法复杂度,对同长报文比对后分类的各类中选取少数样本组合成作为样本集进行不同长报文比对。

3.1.4 语义推断

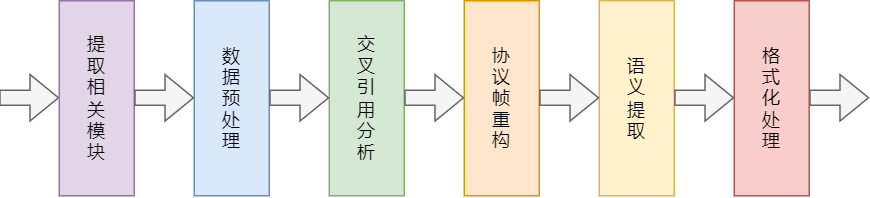
语义推断是指推断前面划分出的协议各字段的含义。由于单报文处理中已将文本字段、时间戳字段、校验和字段进行了识别，因此本阶段主要是对标识、序号、长度、地址、功能码、数据等字段进行识别。最后对提取的格式进行整理融合，最终得到协议的格式结构。

序号字段用于标识报文的前后顺序，这一字段的特征是一直在变化，且与报文截获的顺序相对应。通过判断字段的取值变化是否与报文截获呈正相关可以判断出是否为该类型字段。

长度字段用于标识某一段数据的长度，在报文中主要起界定作用。该字段的特征为对于长度相近的报文，该字段的取值相等；对于长度不同的报文，该字段取值与报文长度呈正相关。或者该字段与报文中某可变长度字段的长度呈正相关。

格式标识字符用于标识报文的格式。该字段的特征是相同格式的报文该字段相同，不同格式的报文该字段不同。

3.2 基于静态二进制分析的私有工控协议逆向分析技术

基于静态二进制分析的私有工控协议逆向分析是指通过静态分析工控相关软件的协议解析模块来对私有工控协议逆向分析。具体过程分为提取相关模块，数据预处理，交叉引用分析，协议帧重构，语义提取和格式化处理。

3.2.1 提取相关模块

通常情况下，工控软件会集成众多功能，因此软件体积较为庞大，直接对工控软件进行逆向分析不仅工作量大，还会降低分析的准确度。而协议相关的功能大多集中在通信模块中，通过提取相关模块，缩小逆向分析的范围，减少工作量。

提取相关模块的具体方式是通过读取动态链接库(Dynamic Link Library，DLL)文件的导入表和导入函数表来查找和预测DLL涉及到的操作。一般来讲，涉及到TCP协议的通信则需要导入WS2\_32.dll中的send和recv函数，UDP协议则需要导入WS2\_32.dll中的sendto和recvfrom函数。另外，通过扫描并过滤WriteFile等系统调用还可以找到工控软件自己封装的一些发包和封包函数，通过这些函数还可以解析那些不基于TCP/IP的工控协议。

3.2.2 数据预处理

在实际的分析过程中会发现，即使是提取出相关模块，缩小逆向分析的范围，但逆向的工作量依然很大。这是因为很多厂商在开发工控软件的时候没有严格遵循模块化的设计原则，协议相关的代码没有独立的封装在DLL中，而是和其它功能的代码混杂在一起封装在同一个DLL中，或者是零散分布在多个DLL中。针对这种情况，需要对DLL中的函数进行标记处理，剔除掉与协议无关的函数。

3.2.3 交叉引用分析

从软件逆向工程的角度来看，被不同的上层函数引用次数越多的底层函数通用性越好，封装程度越高，在设计协议封装和解析的软件模块中，则往往表现为在函数中处理了某一类通用性较强的帧结构，比如携带数据载荷的帧和心跳帧；而被不同的上层函数引用次数较少，甚至仅在某一处有过引用的函数，在协议封装和解析的软件模块中往往完成一些连接建立，通信对端认证等控制类的操作。通过获取交叉引用信息，根据引用数据统计可以计算推断出函数的功能类型。

3.2.4 协议帧重构

协议帧重构阶段以前两个阶段中获得的函数依赖关系以及函数类型推定为参照，找出函数中组帧的代码，根据代码构造的帧结构以及写入的常量等特征匹配收集的数据包，并将数据包按照匹配代码分类。

3.2.5 语义提取

语义提取是指根据函数中组帧的代码特征推断字段的含义。例如如果产生目标字段的代码中包含大量的移位等运算操作，则目标字段极有可能是校验码；如果产生目标字段的代码中包含着常量赋值操作，并且对于不同的函数调用常量不同，则目标字段可能是功能码。

3.2.6 格式化处理

经过以上过程的处理，有可会出现一些冗余的协议格式，比如同一个字段被识别为多个不同的含义，或者两个字段的范围出现重叠等，再或者对于不定长的帧，不同长度会会被划分为不同格式，这些错误还需要进行后期处理分析，直到得出合理的协议格式。

3.3 新的协议逆向分析思路

通过对当前私有工控协议逆向分析的综述不难发现，无论是从网络流量分析还是从静态二进制分析，虽然各具优势，但都存在着一定的缺陷和不足。无论单从哪个方向入手都无法做到对私有工控协议完整准确逆向分析。

对于基于网络流量的分析方式，虽然这种方法可以避免大量的人逆向分析工作，但是这种方法分析作用有限，只能分析一些简单的长度，校验码以及编号等字段，对于稍微复杂一点的字段就很难分析出来。而现实情况很多私有工控协议处于安全考虑，很多字段都会进行简单的变换或加密，使得分析变得非常困难。另外，由于所有字段的含义都是基于对数据包分析总结，而很多工控协议的格式灵活多变，单纯靠分析数据包不能保证字段识别的准确性。

对于基于静态二进制分析的分析方式，虽然可以根据函数功能准确划分出帧结构以及确定各字段功能，但是由于静态分析这种分析方式不能分析出完成的协议交互过程。并且仅仅从二进制分析角度出发没有结合网络流量，也缺乏合理有效的验证方法，分析结果可能存在错误。另外筛选关键代码的方式也过于依靠经验，无法保证筛选出所有的协议相关代码。

为了突破上述方法的缺陷，本文提出了一种以基于网络流量与动态调试控制软件相结合，并利用Python脚本发包验证的私有工控协议逆向方式。这种方式可以结合上述两种方法各自的优势，能够比较准确详细的获取私有工控协议的帧格式以及协议过程，并且可以极大的降低分析难度和工作量。

3.3.1 协议过程逆向分析

动态调试工程软件，利用网络流量分析工具实时检测PLC与工程软件之间的交互过程。根据PLC与工程软件之间是否发包交互，确定出生成数据包中各字段的关键代码以及处理PLC回复数据包的关键代码。根据关键代码所在函数以及其关键字段的生成与处理方式确定出数据包的功能作用。

3.3.2 帧格式及各字段功能获取

将前面协议过程逆向分析的结果与网络流量分析相结合，找出组帧的关键代码，根据反编译代码中各变量的含义及与数据包中的数据的比较，确定出帧格式及各个字段的含义。对于不易确定含义的反编译代码中的变量对应的字段以及无法找到对应帧代码的字段，则结合网络流量分析推断出其含义。最后借助Python脚本组帧发包验证，根据PLC是否正确回复确定协议帧格式及字段功能是否分析正确。

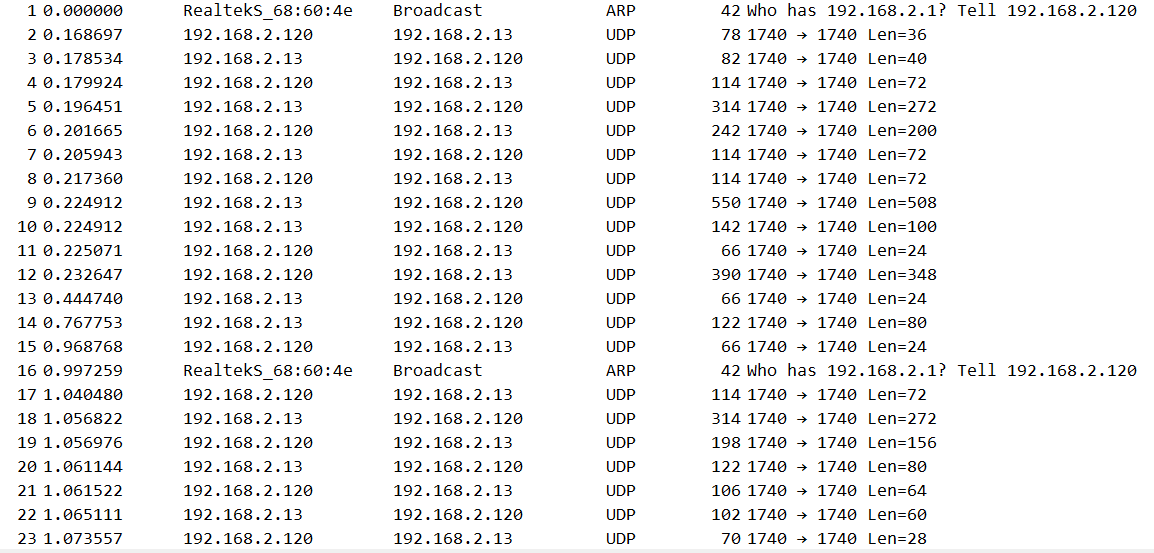
3.4 本章小结

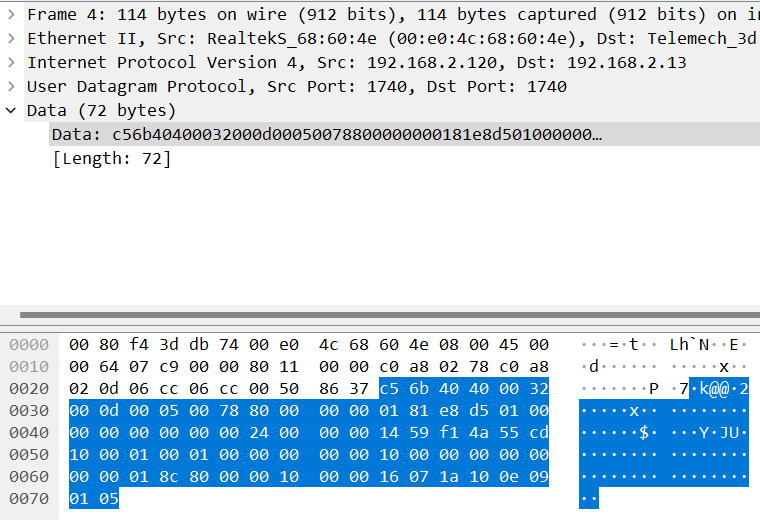
本章分析总结了当前常见两种私有工控协议逆向分析技术：基于网络流量的分析技术和基于静态二进制代码分析技术。详细阐述了两种技术的分析方法和具体过程，并且分析总结了这两种技术各自的优缺点。最后结合两种技术各自的优势，本人提出了一种新的私有工控协议的逆向分析技术，并且对帧格式及各字段功能获取以及协议过程逆向分析这两个过程进行详细的具体操作方法进行详细阐述。

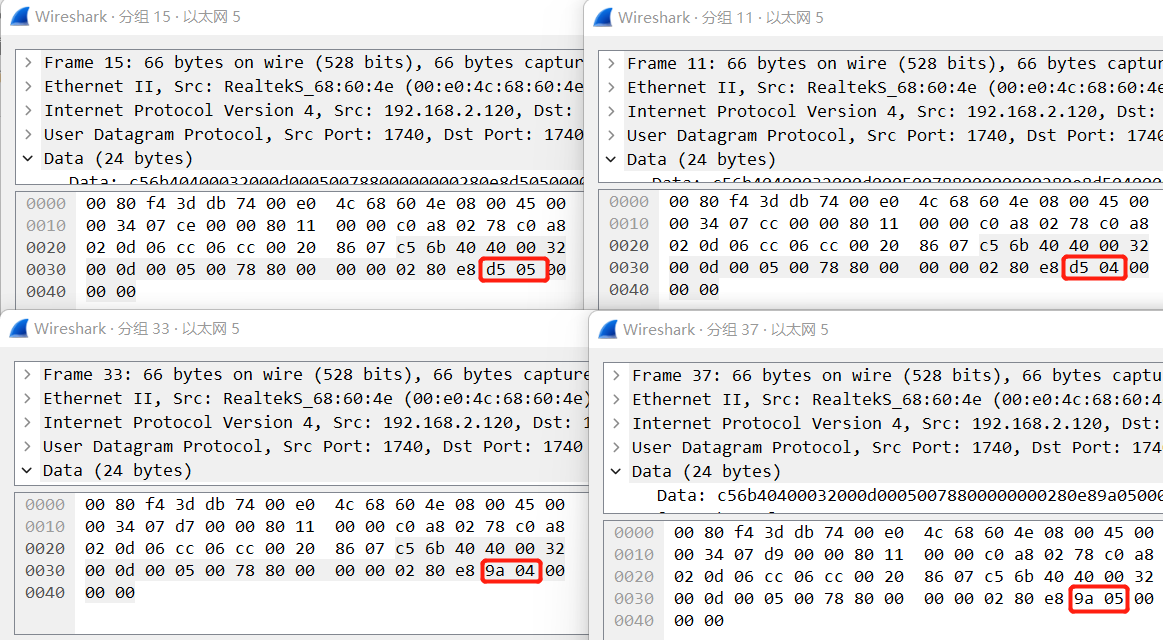
第四章 私有工控协议逆向分析实践

本文第二章对实践目标施耐德M241可编程逻辑控制器进行了详细的介绍。本章将在此基础上根据前面第三章最后提出的新的私有工控协议逆向分析技术进行实践。实践过程分为协议过程逆向分析和帧格式及各字段功能获取两部分。本文接下来将对这两个部分的具体分析过程详细介绍，其中协议过程逆向分析以登录PLC的过程为例进行介绍。最后通过python脚本实现远程登录PLC并且控制PLC启停等操作验证协议分析的结果。

4.1 通信流量初步分析

启动Machine Expert ，设置M241的IP地址为192.168.2.13，本机IP为192.168.2.120，然后登录PLC同时使用Wireshark抓取登录过程的流量数据，结果如下图。

根据流量抓取结果可以推断出该通信协议是建立在UDP协议的基础上的应用层协议。对于具体的通信数据包，Wireshark将传输的协议帧识别成一整组数据，说明该通信协议为未公开的私有协议。

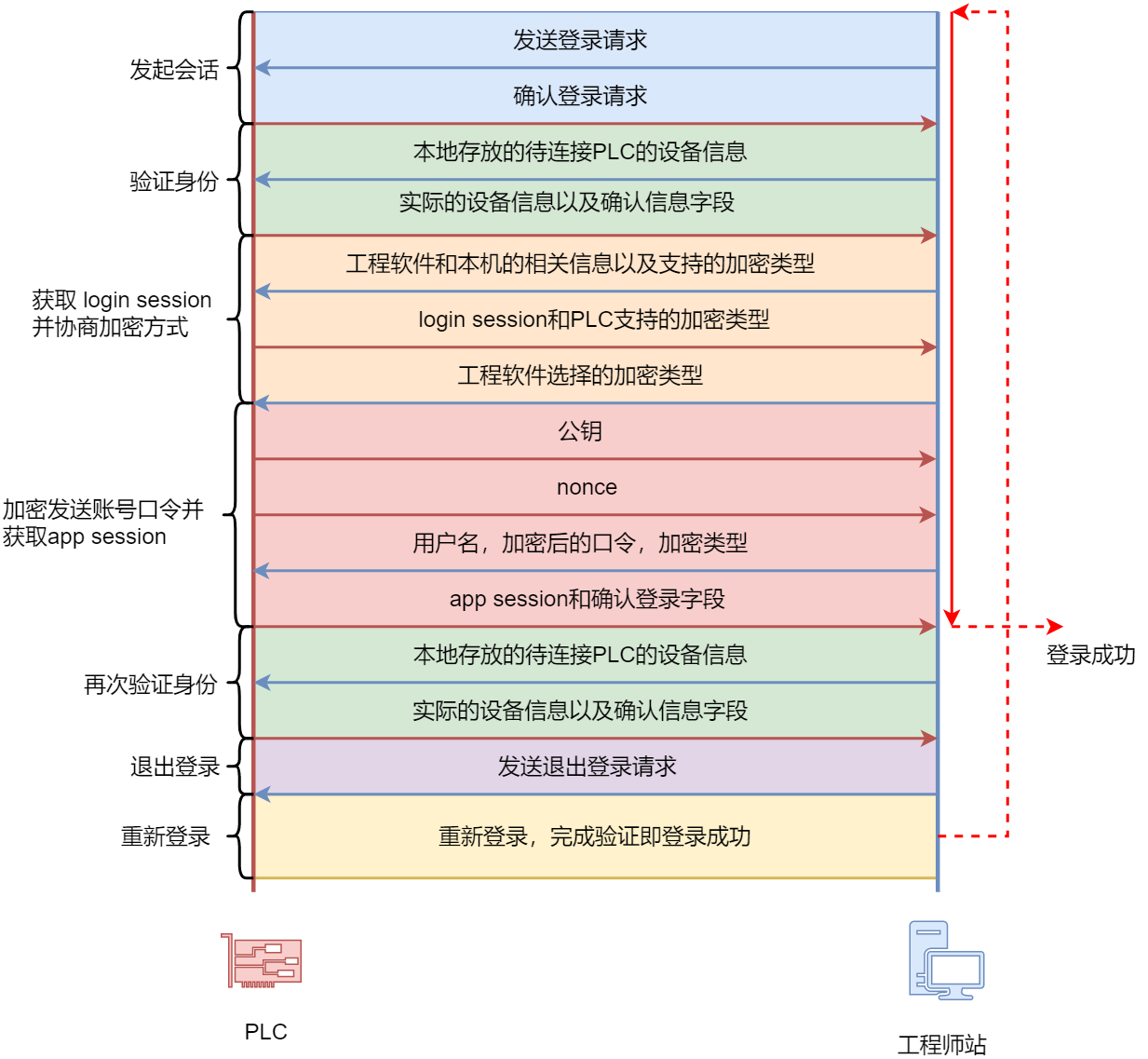
通过观察分析通信流量发现，Machine Expert与M241之间频换发送长度为66的数据包，这一数据包即使在没有任何通信相关的操作时也会发送，并且其内容大部分重复，只有个别字段呈现出一定规律的变化，因此基本可以推断出这一类的数据包是心跳包。

为了便于后续的协议分析工作，可以在Wireshark中添加如下规则来筛选出有效的数据包：

(ip.src==192.168.2.13||ip.dst==192.168.2.13)&&(ip.src==192.168.2.120||ip.dst==192.168.2.120)&&frame.len!=62

4.2 协议过程逆向分析

本节将以登录PLC的过程为例进行协议过程逆向分析。根据协议逆向分析的结果，登录过程分为发起会话，验证身份，获取 login session 并协商加密方式，加密发送账号口令并获取app session，再次验证身份以及登录五部分，过程如下图所示。

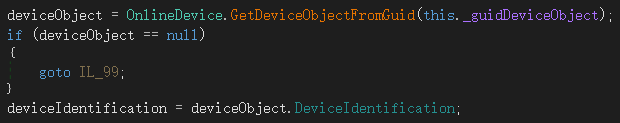


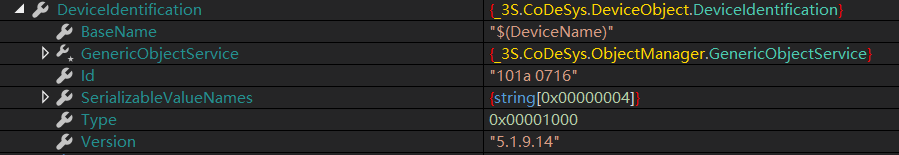
4.2.1 发起会话

通过分析数据流量可知，登录PLC的第一步Machine Expert会向M241发送一个长度固定为78的数据包发起会话。

4.2.2 验证身份

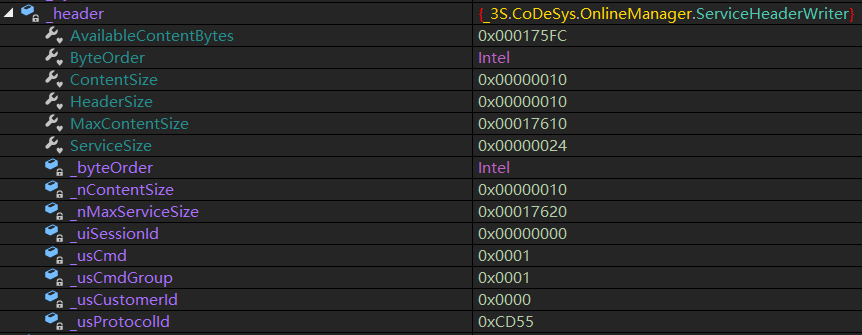
发起会话并收到M241的确认包后，Machine Expert会调用CheckTargetIdent函数，检查 PLC 的身份。

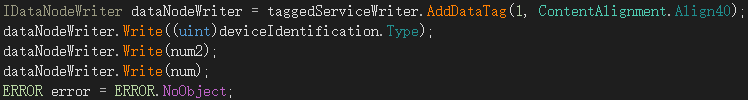
在CheckTargetIdent函数中，首先调用GetDeviceObjectFromGuid获取本地存储的待登录设备M241的相关信息。

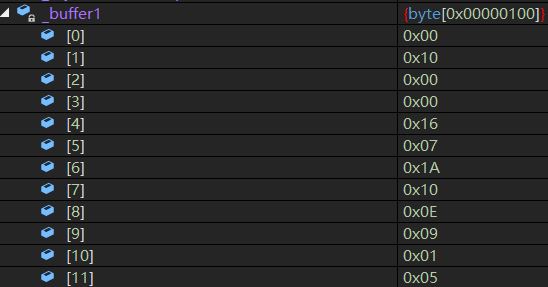
获取的信息如下：

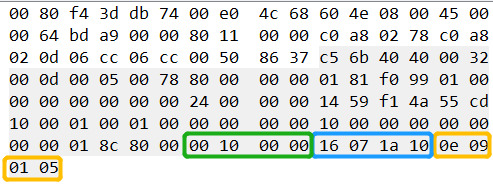
之后将获取数据进行转换为数据包中对应的字段的值。其中M241固件版本号由字符串5.1.9.14转换为数值0x0501090E；待连接的M241的设备ID由字符串101a 0716转换为数值0x101A0716。

接下来调用CreateService函数创建数据包头部类，并把这个类存放到变量taggedServiceWriter中。

taggedServiceWriter中的\_header子类中的各个变量的值对应于数据包头部的各个字段，这位后面确定协议帧格式以及各字段的含义提供了重要依据。

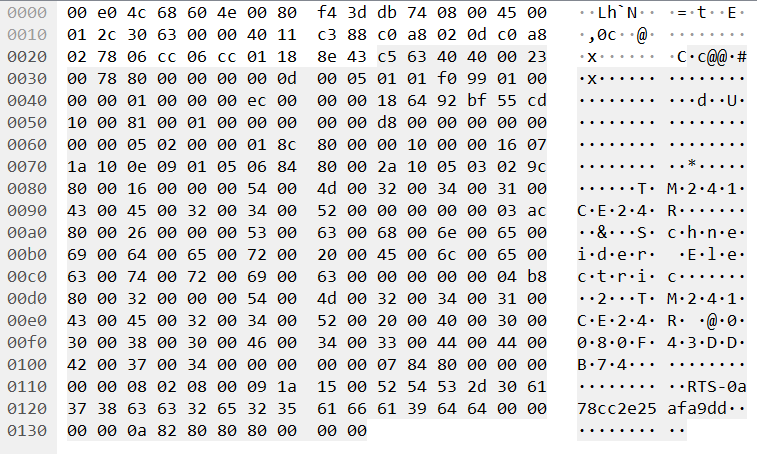
创建数据段dataNodeWriter并将前面得到的 版本号(num)、设备ID(num2)以及deviceIdentification.Type(0x00001000)写入 。

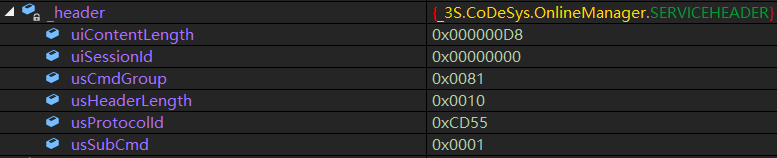
由于dataNodeWriter是一个引用，反编译器无法查看其内容。但是可以从taggedServiceWriter.\_subTags[0].\_contentBuffer.\_buffer1看到写入的内容。

最后调用ExecuteService函数将组好的数据包发送出去。从Wireshark捕获的数据包中可以找到待连接的M241的设备类型，设备ID以及固件版本对应的字段。

收到数据包之后，PLC回复报文

内容如下：

Machine Expert将报文内容识别为serviceTagReader，并且解析出其中各个字段的值。

首先serviceTagReader识别出的报文头部字段如下：

可以看出报文头部的内容与发送报文的头部结构基本一致。

之后依次解析出报文中各字段的值如下：

|  |  |
| --- | --- |
| 变量名 | 内容 |
| num5 | 0x00001000 |
| num6 | 0x101A0716 |
| num7 | 0x0501090E |
| text | TM241CE24R |
| stDeviceName | Schneider Electric |
| error | OK(0x0) |

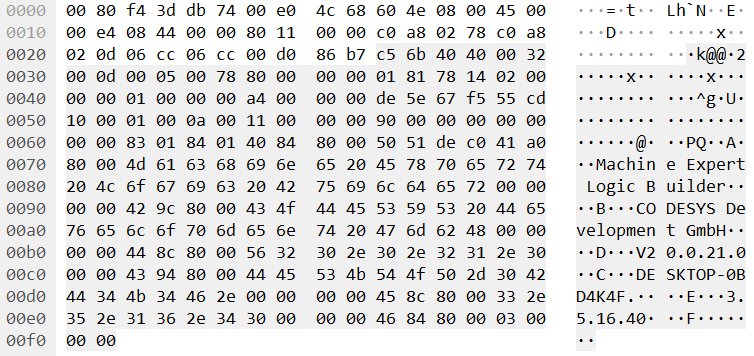
由于error字段为OK，因此因此身份验证通过。

4.2.3 获取 login session 并协商加密方式

调用Login函数，获取软件中缓存的登录账号密码，如果为空则弹出窗口让用户输入账号。之后调用DeviceLogin函数，检验this.\_dwSessionId是否初始化，然后调用DeviceLoginWithCreateSession函数。如果this.\_dwSessionId是没有始化，则DeviceLoginWithCreateSession函数的参数dwCurrentSessionId赋值为 17，作为需要获取 session 的标记。

进入dwCurrentSessionId函数由于未初始化session，因此需要获取session。Machine Expert首先收集软件主机的相关信息以及支持的加密方式如下，然后发送至PLC。

|  |  |
| --- | --- |
| 名称 | 内容 |
| ProductName | Machine Expert Logic Builder |
| Company | CODESYS Development GmbH |
| ProfileName | V20.0.21.0 |
| DomainName | null |
| MachineName | DESKTOP-0bD4K4F. |
| Version | 3.5.16.40 |
| \_supportedCryptType | 0x00000003 |

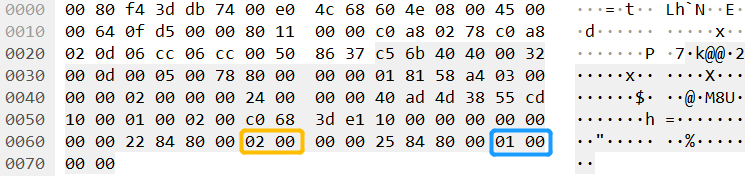
抓取的流量数据中可以找到对应字段。

从M241回复的数据包中可以获取 Login Session 和PLC支持的加密方式\_cryptTypeToUse。Machine Expert根据协商好的加密方式调用对应的函数进行接下来的过程。

1. **if** (**this**.\_cryptTypeToUse == CryptType.XOR || **this**.\_cryptTypeToUse == CryptType.Invalid)
2. {
3. result = **this**.DeviceLoginXor(dwCurrentSessionId);
4. }
5. **else** **if** (**this**.\_cryptTypeToUse == CryptType.ASYMMETRIC)
6. {
7. result = **this**.DeviceLoginAsymmetric(dwCurrentSessionId);
8. }
9. **else**
10. {
11. result = ERROR.Failed;
12. }
13. **if** (!**this**.\_reLogin)
14. {
15. **this**.\_duringLoginWithSession = **false**;
16. }
17. **return** result;

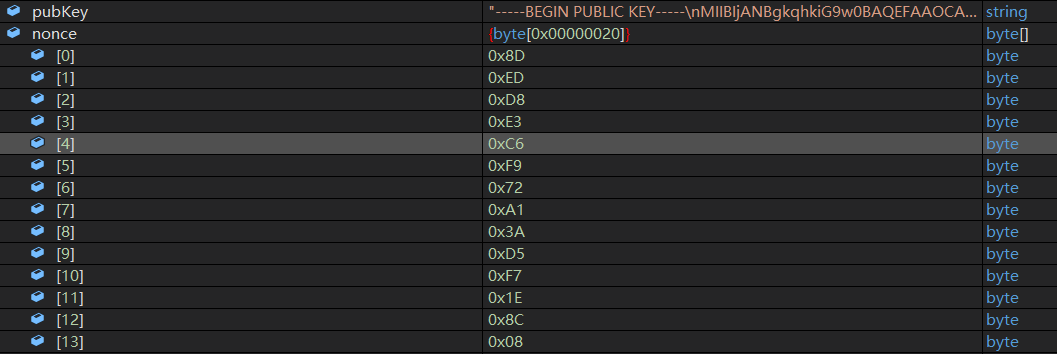
最后Machine Expert向M241回复选择的加密方式。

1. **uint** num = 1U;
2. ITaggedServiceWriter taggedServiceWriter = **new** TaggedServiceWriter(**this**.\_channel.CommunicationBufferSize, 52565, **this**.\_channel.ByteOrder, 1L, 2, dwCurrentSessionId);
3. taggedServiceWriter.AddDataTag(34, ContentAlignment.Align40).Write((**uint**)**this**.\_cryptTypeToUse);
4. taggedServiceWriter.AddDataTag(37, ContentAlignment.Align40).Write(num);
5. IServiceReader serviceReader = **this**.EndExecuteService(**this**.BeginExecuteService(taggedServiceWriter, **null**, **new** OnlineDevice.DeviceLoginState()));

数据包中可以找到对应字段。

4.2.4 加密发送账号口令并获取app session

由于前面协商的加密方式为ASYMMETRIC，M241发送加密口令所需的nonce和公钥pubKey。由于数据长度过长因此分为两个数据包发送给Machine Expert。

其中获得的nonce和pubKey内容如下：

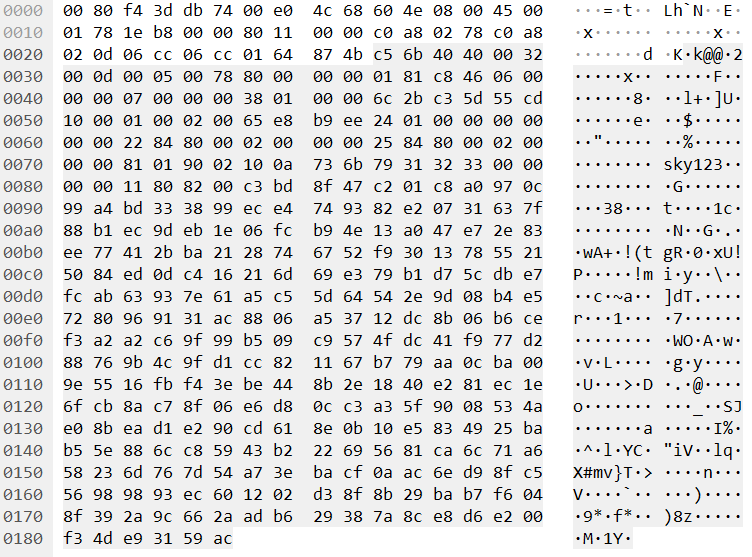
之后调用EncryptPassword函数加密口令。根据前面协商的加密方式，调用EncryptPasswordRSA对口令进行加密。

首先将password与nonce逐字节异或，结果赋值给array。

1. **byte**[] array = **new** **byte**[60];
2. **for** (**int** i = 0; i < password.Length; i++)
3. {
4. array[i] = (**byte**)password.ElementAt(i);
5. }
6. **for** (**int** j = 0; j < Math.Min(nonceBytes.Length, array.Length); j++)
7. {
8. array[j] ^= nonceBytes[j];
9. }

之后对array进行RSA加密。

1. **using** (RSACng rsacng = **new** RSACng())
2. {
3. RSAParameters paramsForPublicKey = **new** CryptoHelperService().GetParamsForPublicKey(pubKey);
4. rsacng.ImportParameters(paramsForPublicKey);
5. result = rsacng.Encrypt(array, RSAEncryptionPadding.OaepSHA256);
6. }

之后将用户名、加密后的口令以及加密类型等组包发送给 PLC 进行登录验证。

如果账号口令正确，回复的数据包中将包含app session。

4.2.4 再次验证身份以及登录

调用CheckTargetIdent函数，检查PLC的身份。过程与上一次相同。

之后初始化相关变量，然后调用Logout函数发送功能码00 01 00 03和dwSessionId退出登录。

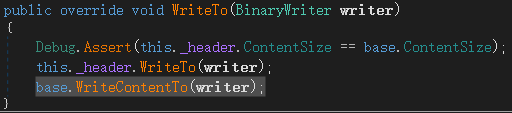
1. ITaggedServiceWriter taggedServiceWriter = **this**.CreateService(1L, 3, dwSessionId);
2. taggedServiceWriter.AddDataTag(1, ContentAlignment.Align40).Write(dwSessionId);
3. **this**.ExecuteService(taggedServiceWriter);

最后从发起会话开始重新登录 PLC 。

4.1 帧格式及各字段功能获取

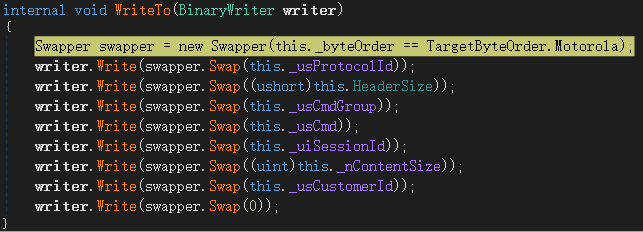
在前面协议过程逆向分析时，看到协议发包函数是ExecuteService，而传入的参数是存入了各种要发送的数据的taggedServiceWriter，因此猜测其中包含着组帧的关键代码。

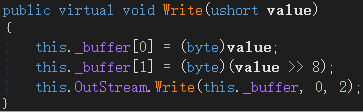


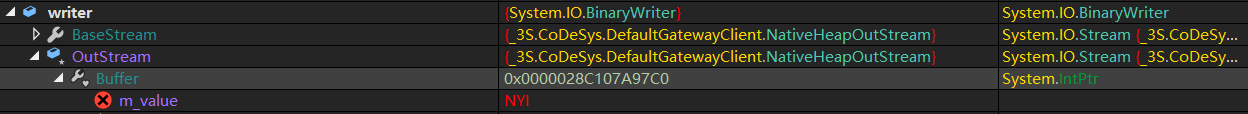
步入这个函数，然后一步步跟入，直到WriteTo函数。WriteTo函数内容如下：

WriteTo函数分别调用this.\_header.WriteTo函数和 base.WriteContentTo函数向输出缓冲区写入数据。两个函数的功能分别为写入协议头部和数据部分。

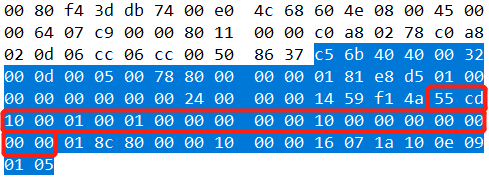
接下来依次分析这两个函数。

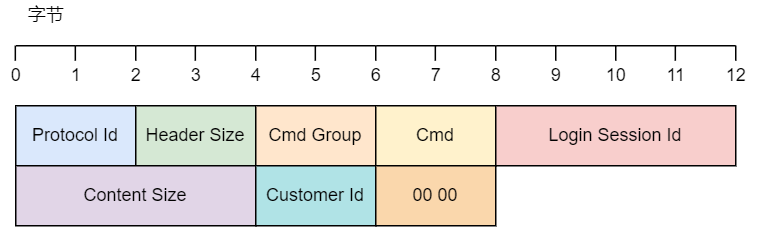
对于this.\_header.WriteTo，其函数的内容如下。该函数的作用是以小端序向输出缓冲区依次以小端序写入各个字段。

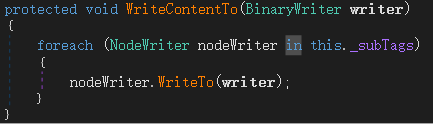
继续步入writer.Write函数，内容如下。这里需要说明的是，根据传入参数的字节大小，调用的Write是不同的，这里仅举例ushort类型的参数对应的Write。

分析Write函数的内容可知数据是先写入this.\_buffer，然后再将this.\_buffer中的数据写入到OutStream中的某个位置。具体来说是写入OutStream中的Buffer变量。

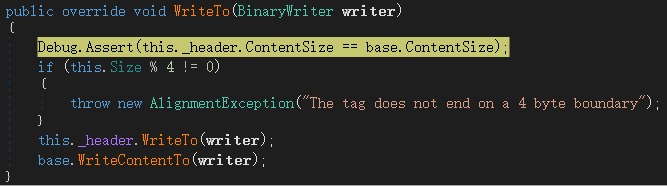
虽然无法直接查看其内容，但根据数据存放的地址，可以在内存的对应位置查看其数据：

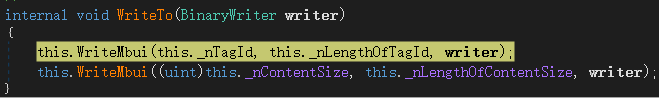
在抓取的数据包中也能找到这20字节的内容：

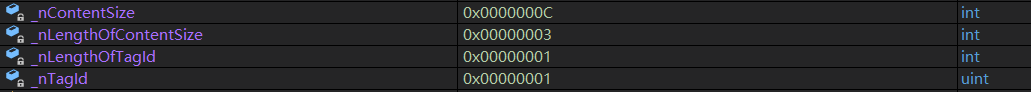
因此头部的帧格式为：

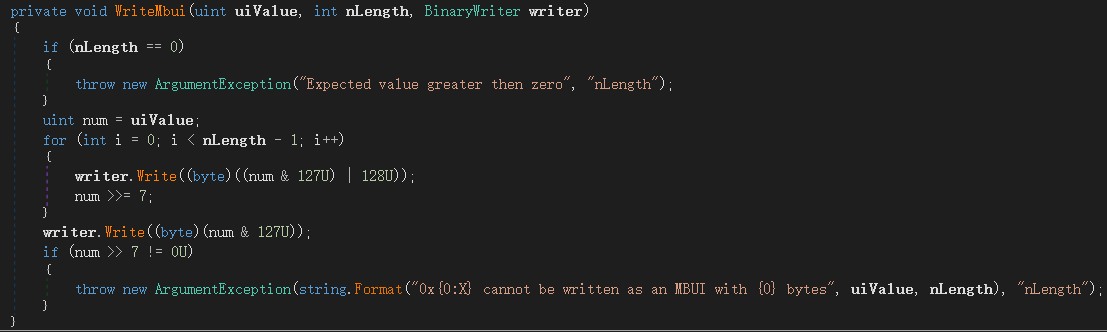
对于base.WriteContentTo，其函数内容如下：

从base.WriteContentTo函数的内容可以看出，数据包中的各个字段都在this.\_subTags中，base.WriteContentTo函数所做的操作是依次取出其中的字段写入输出缓冲区。

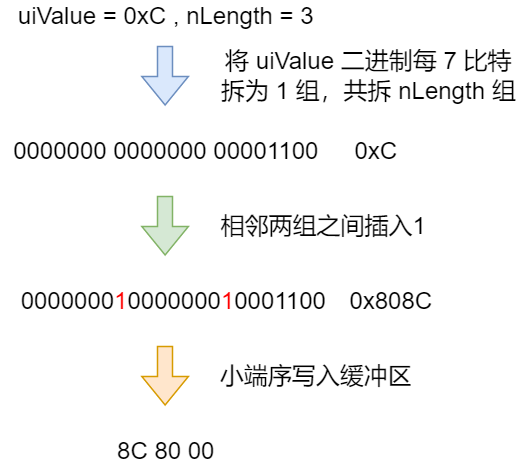
步入nodeWriter.WriteTo，可以看出每个字段都有一个独立的头部和内容。

首先是头部。步入this.\_header.WriteTo函数，

两次调用this.WriteMbui函数写数据。传入的参数如下图：

this.WriteMbui函数内容如下：

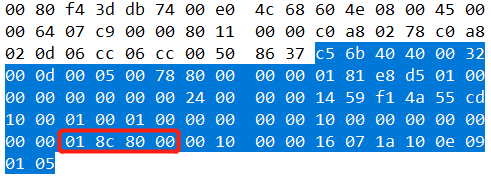
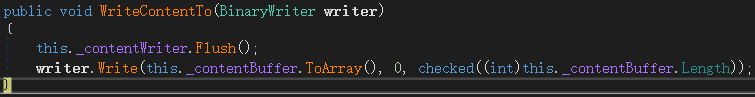
通过简单分析可知，该函数本质上是将uiValue二进制形式在高位不足时补0的前提下每7比特划分一组总共划分nLength组，然后组与组直接插入一个1使得长度为8\*nLength-1，然后按字节形式写入输出缓冲区。

以\_nContentSize为例，过程如下图所示，最终写入缓冲区的数据为8C8000。

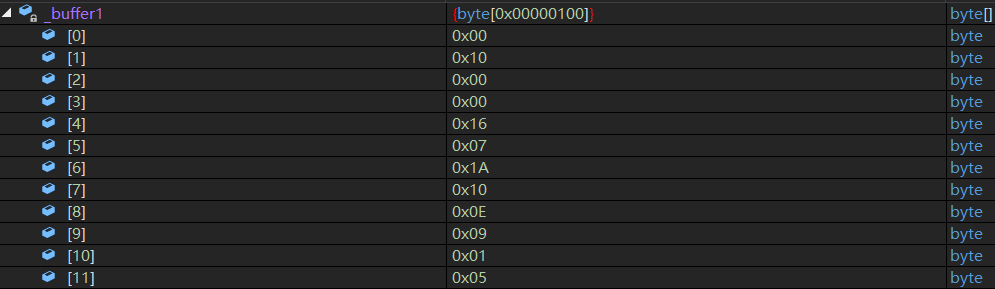
执行完this.\_header.WriteTo函数后输出缓冲区有写入了数据018C8000，与前面的分析相符合。



从抓取的数据包中也能找到对应的数据。

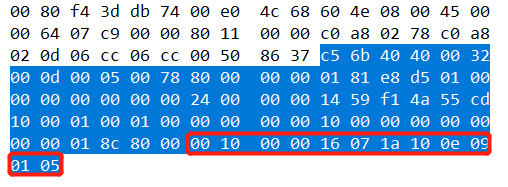
接下来是写入内容。步入base.WriteContentTo函数，

首先是刷新\_buffer缓存的数据，将还未复制到输出缓冲区的数据复制到输出缓冲区。之后将\_contentBuffer的内容写入到输出缓冲区。由于前面nodeWriter.WriteTo函数进行过校验，所以这里写入的长度this.\_contentBuffer.Length即前面加密在头部字段的\_header.ContentSize。

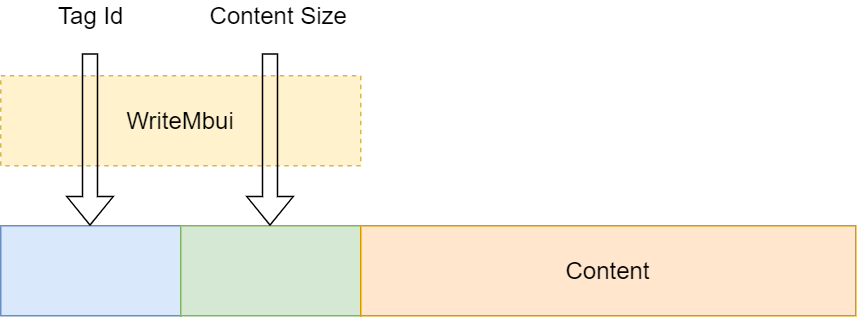
\_contentBuffer 的内容如下：

这部分内容被写入输出缓冲区。



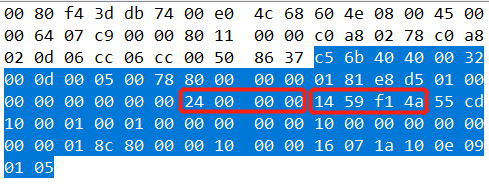
并且能在数据包中找到。

因此nodeWriter部分对应的帧格式为：

在Machine Expert中找不到关于外层的头部的组帧代码，推测在Machine Expert完成内层的组帧后又经过GatewayService封装。

不过这部分协议字段仅通过数据流量分析就可以推测出各字段的含义。

首先进行单报文处理，识别出与前后报文无关的字段。

观察报文内容推责下图所示两个字段分别为内层报文的长度和CRC32校验。

from zlib import crc32  
  
data = b"\x55\xcd" \  
 + b"\x10\x00\x01\x00\x01\x00\x00\x00\x00\x00\x10\x00\x00\x00\x00\x00" \  
 + b"\x00\x00\x01\x8c\x80\x00\x00\x10\x00\x00\x16\x07\x1a\x10\x0e\x09" \  
 + b"\x01\x05"  
  
print(crc32(data).to\_bytes(4, byteorder=**'little'**, signed=False) == b"\x14\x59\xf1\x4a")  
print(len(data).to\_bytes(4, byteorder=**'little'**, signed=False) == b"\x24\x00\x00\x00")