面向感算控智一体化融合的工业互联网新型 PLC 安全增强:趋势与展望

王雅哲[1)] 任 磊[1),2)] 冯登国[1),3)] 王文杰[1)]
张一凡[1),2)] 王 斐[1)] 孔宇升[1),2)] 吕金虎[1),2)]

[1)](中关村实验室 北京 100094)

[2)](北京航空航天大学自动化科学与电气工程学院 北京 100191)

[3)](中国科学院软件研究所 北京 100190)

摘 要 可编程逻辑控制器(PLC)作为工业控制领域的关键基础设施,由于其设计初期缺乏内生安全架构,多年来一直是针对工业互联网及控制系统展开攻击的主要目标。随着工业 4.0 及智能制造技术的迅猛发展,传统 PLC 正在向智能可编程工业控制器(IPIC)演进,集成了感知采集、数据计算、实时控制、智能决策和网络通信等功能,以满足自动化、网络化和智能化的工业互联网技术需求。然而,信息技术(IT)与操作技术(OT)的融合以及异构功能载荷的一体化集成进一步加剧了 IPIC 面临的安全挑战。本文旨在对工业互联网新型 PLC(即 IPIC)的安全增强展开研究,尤其揭示了嵌入式虚拟化隔离技术在构建 IPIC 内生安全架构中的应用前景和未来发展方向。文章首先系统性分析了 PLC 所面临的安全威胁及现有的安全防护技术;随后重点介绍了具备实时性特征的嵌入式虚拟化隔离技术,并提出了从内生安全角度重新构建 IPIC 基础安全体系架构的解决思路;接着延承虚拟化隔离思路,通过异构安全载荷的虚拟化集成为 IPIC 设计了一种异常检测与响应的安全增强架构,并讨论了该架构在实际工业系统部署中面临的技术挑战;最后对 IPIC 安全增强的未来发展趋势进行了前瞻性展望。

关键词 PLC;IPIC;嵌入式虚拟化;工业互联网;异构载荷融合部署;异常检测;安全增强

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2025.00738

Security Enhancements for New-Generation PLC of the Industrial Internet Integrating Sensing, Computing, Control and Intelligence: Trends and Perspectives

WANG Ya-Zhe[1)] REN Lei[1),2)] FENG Deng-Guo[1),3)] WANG Wen-Jie[1)]
ZHANG Yi-Fan[1),2)] WANG Fei[1)] KONG Yu-Sheng[1),2)] LÜ Jin-Hu[1),2)]

[1)](Zhongguancun Laboratory, Beijing 100094)

[2)](School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191)

[3)](Institute of Software, Chinese Academy of Sciences, Beijing 100190)

Abstract Programmable Logic Controllers (PLCs), a key infrastructure in the industrial control domain, have long been a primary target for attacks on Industrial Internet and Control Systems due to the lack of inherent security architecture in their initial design. With the rapid development of Industry 4.0 and intelligent manufacturing technologies, traditional PLCs are evolving into Intelligent Programmable Industrial Controllers (IPICs), integrating functions such as perception

收稿日期:2024-05-23;在线发布日期:2024-12-18。本课题得到国家重点研发计划(2023YFB3308000)、国家杰出青年科学基金(62225302)、国家自然科学基金(92167108,62173023)资助。**王雅哲**,博士,副研究员,中国计算机学会(CCF)会员,主要研究领域为网络与系统安全、应用与数据安全。E-mail: wangyz@zgclab.edu.cn。**任 磊**(通信作者),博士,教授,中国计算机学会(CCF)会员,主要研究领域为工业互联网、工业人工智能。E-mail: renlei@zgclab.edu.cn。**冯登国**,博士,研究员,中国科学院院士,主要研究领域为信息安全和密码学。**王文杰**,博士,助理研究员,主要研究领域为网络与系统安全,尤其关注嵌入式系统安全、工业互联网和应用密码学。**张一凡**,博士研究生,主要研究方向为神经网络与深度学习、时间序列分析和工业互联网。**王 斐**,博士,助理研究员,主要研究领域为工业物联网安全、工业智能和智能优化。**孔宇升**,博士研究生,助理工程师,中国计算机学会(CCF)会员,主要研究方向为物理工业人工智能和 IIoT 网络安全。**吕金虎**,博士,教授,中国计算机学会(CCF)会员,主要研究领域为非线性电路与系统、复杂网络、多智能体系统和大数据。

acquisition, data computation, real-time control, intelligent decision-making, and network communication to meet the technical demands of automation, networking, and intelligence in the Industrial Internet. However, the convergence of Information Technology (IT) and Operational Technology (OT), along with the integrated heterogeneous functional loads, further exacerbates the security challenges faced by IPICs. This paper aims to research the security enhancement of the new type of Industrial Internet PLCs (i.e., IPICs), especially revealing the application prospects and future development directions of embedded virtualization isolation technology in building the inherent security architecture of IPICs. Specifically, it first systematically analyzes the security threats faced by PLCs and the existing security protection technologies; then it introduces embedded virtualization isolation technology with real-time characteristics, and proposes a solution to reconstruct the basic security architecture of IPICs from the perspective of inherent security; subsequently, it follows the virtualization isolation concept and designs a security enhancement architecture for anomaly detection and response in IPIC by integrating heterogeneous security loads through virtualization, discussing the technical challenges faced in the actual deployment of industrial systems; finally, it provides a forward-looking perspective on the future development trends of IPIC security enhancement.

Keywords　PLC; IPIC; embedded virtualization; Industrial Internet; heterogeneous load consolidation deployment; anomaly detection; security enhancement

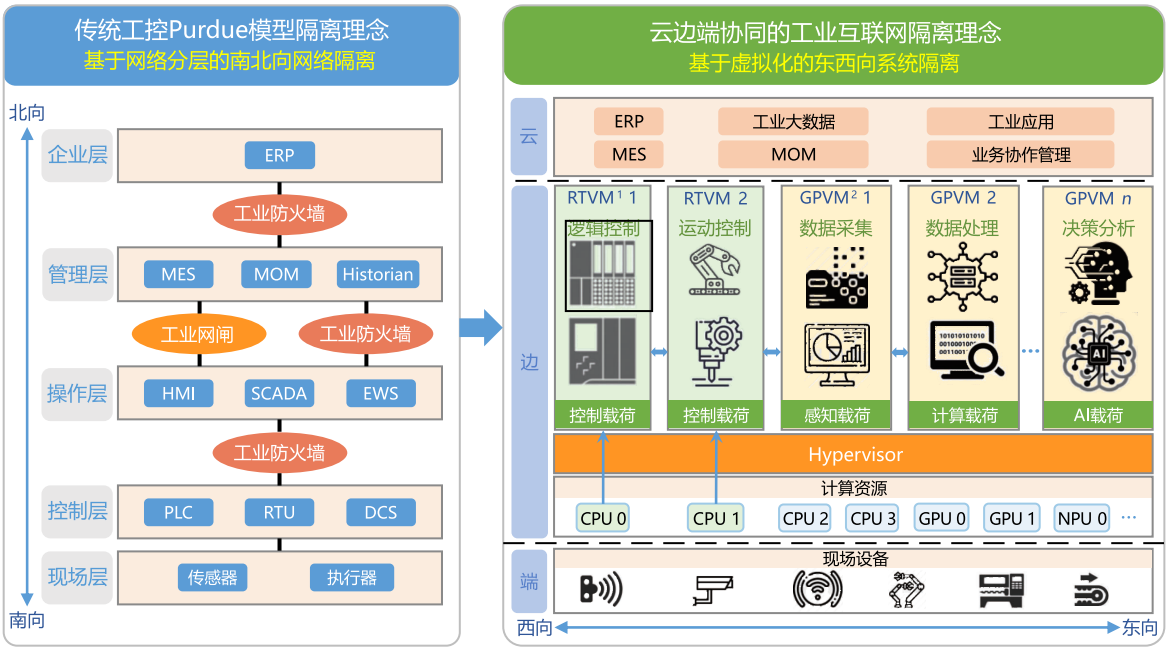
1　引　言

在传统工业控制系统(Industrial Control Systems, ICS)架构下,可编程逻辑控制器(Programmable Logic Controller,PLC)通过现场总线将自动化操作逻辑传导给产线执行设备并收集感知数据,是控制每个关键基础设施部门制造过程不可或缺的工业设备。因其缺乏针对网络空间安全攻击的原生设计,多年来 PLC 一直是针对工业控制系统的攻击焦点,Stuxnet[1]、Incontroller/Pipedream[2]等多种工控系统攻击事件都以 PLC 为目标,美国著名网络安全研发机构 MITRE 发布的网络攻击知识库 ATT &CK 中有专门的工控系统策略和技术围绕其实施[3],每年的世界顶级黑客大会 BlackHat[4-5]、DEFCON[6-7]上也多有针对 PLC 的脆弱性分析和漏洞公开。PLC 因其 IT/OT 功能结合部的业务特征,已经成为对工业信息物理系统(Cyber Physical Systems,CPS)进行网络渗透及实施攻击的关键载体和各类 ICS 防护检测的重点关注对象[8]。

随着工业 4.0、智能制造等工业互联网场景的快速发展,工业信息物理系统日益复杂[9],ISA-95 国际标准[10]提出的工业控制系统典型架构普渡模型已不能适应企业对工控系统自动化、网络化与智能化的技术需求,具体趋势表现为:跨域数据交互驱动的 IT 和 OT 网络融通开放互联、柔性制造智能工程驱动的工业控制逻辑快速灵活配置、智能化算力下沉驱动的感算控智异构载荷一体化集成。信息通信技术和自动化操作技术各自的运行边界趋向模糊,自动化控制逻辑、信息传输通信、智能决策分析为代表的工业互联网多类型计算载荷在产线侧融合增长。在这种技术发展背景下,传统可编程逻辑控制器也逐步向基于云边协同的智能可编程工业控制器(Intelligent Programmable Industrial Controller, IPIC)转变,借助多核架构通用型中央处理器(Central Processing Unit,CPU)和嵌入式虚拟化等技术演进,将感知采集、数据计算、实时控制、智能决策及网络通信等载荷在一个高集成度的硬件平台片上系统(System on Chip,SoC)统一部署[11]。例如:西门子将人工智能神经处理单元 TMNPU 模块和 S7-1500 PLC 集成,根据生产数据的实时分析自动调整控制参数,提升工厂环境下的自动化实时决策能力[12];罗克韦尔推出的 LogixAI 人工智能 PLC 能够基于内嵌运算模型向控制程序学习,进而预测和分析控制运行过程中数据变化趋势[13];欧姆龙的 NX 和 NY 系列人工智能控制器搭载特有的 AI 功能单元,通过运行上层软件功能库中 AI 模组,对控制系统对象数据开展持续采集和记录,基于预设模型实现设备行为历史趋势分析和设备异常预测[14];倍福提出基于 PC 的开放式控制技术理念,通过 TwinCAT

组件并行运行多任务操作系统以支持控制逻辑、人工智能、数据处理等多功能载荷在多核 CPU 上部署[15]；贝加莱推出的 X 系列工业控制器在 ARM 架构集成 SoC 芯片上统一部署实时系统和通用系统，达成实时高频任务与数据存储、人工智能、图像识别、人机交互等 IT 应用的高效融合[16]。上述新型 PLC 都可以纳入 IPIC 的范畴，不但能够满足工业实时控制的需求，同时也能提供智能边缘服务并与云端协同工作，满足企业对数据融通协同效率、个性化定制业务快速部署、设备预测性运维等方面的关键需求。

结合云计算、工业确定性网络、人工智能、操作系统虚拟化等新技术在工业互联网的应用场景，智能可编程工业控制器安全增强及防护需要关注以下几个安全范式改变：(1) 普渡模型依托网络分层形成的南北向网络化隔离空间在计算资源扁平化集中管理的背景下逐渐演化为依托虚拟化技术形成的东西向系统化隔离空间(如图 1 所示)；(2) 感算控智高度集成的新型工业控制器要求其伴生的多种安全防护载荷也尽可能在其软硬件一体化平台内集成部署；(3) 需要在一个系统平台下克服几个对立特性的统一，例如：OT 域控制逻辑“说到做到”的确定性与 IT 域计算逻辑“尽力而为”的不确定性、控制系统的封闭隔离优先与信息系统的开放互联优先、功能安全强调可用性优先与信息安全强调机密性优先。针对安全范式改变带来的技术挑战，新型智能可编程工业控制器安全增强和防护的维度既要确保在 OT 域功能安全前提下实时完成自动化操控这一工业领域长期不变的首要任务，又要应对工业互联网场景下众多新型信息化技术和传统自动化技术融合后安全边界和攻击面重塑带来的新增威胁。



注：1. RTVM (Real-time Virtual Machine)；2. GPVM (General-purpose Virtual Machine)

图 1 感算控智融合的 IPIC 基础安全架构演进趋势

从近年来针对 PLC 的脆弱性和攻击实现进行研究分析，攻击可针对如图 2 所示 PLC 系统结构中的多个关键组件发起，攻击类型按照被攻击组件脆弱性归因可大致分为代码缺陷攻击[17]、固件修改攻击[18-19]、控制逻辑劫持攻击[20-23]、协议缺陷利用攻击[24-26]、PLC 关联系统攻击[27]，管理门户网站 API 的非法利用攻击[28]等，各类攻击的目标组件与攻击技术总结如表 1(参考文献[29]中分类方法)所示，它们的攻击效果在 PLC 终端主要表现为远程恶意代码执行、内存破坏程序崩溃、身份认证绕过、正常功能拒绝服务、滥用管理设置等并最终在自动化操控层面造成破坏影响。

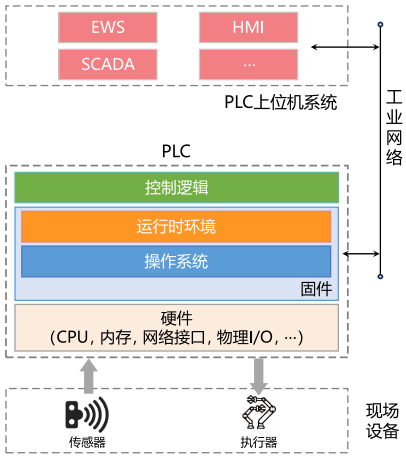


图 2 典型 PLC 系统结构

表 1　现阶段针对 PLC 攻击技术总结

攻击目标组件	攻击方案	技术路线归纳
工业网络协议或服务	Fuzzing and Breaking Security Functions[24]	针对安全通信协议的模糊测试
	The spear to break the security wall[25]	针对安全通信模式的重放攻击
	Attacking The IEC-61131 Logic Engine[26]	针对网络通信的中间人攻击
	Compromising Industrial Processes[28]	针对 PLC Web 服务的恶意利用
控制逻辑程序	On Ladder Logic Bombs[17]	针对控制逻辑的代码缺陷攻击
	Control Logic Injection Attacks[20]	针对控制逻辑的代码注入攻击
	Gadgets of Gadgets in Industrial Control Systems[21]	针对控制逻辑的 ROP 攻击
	Denial of Engineering Operations Attacks[22]	利用控制逻辑构造拒绝服务攻击
	CLIK on PLCs[23]	绕过密码认证机制对控制逻辑篡改
固件	Firmware modification attacks[18]	针对系统固件的非法更新攻击
	Hey, My Malware Knows Physics[19]	利用 rootkit 感染/修改固件
上位机系统	Rogue 7：Rogue Engineering-Station attacks[27]	伪造上位机系统与 PLC 非法通信

目前安全增强的主要研究领域包括：前置阶段利用静态分析[30]、符号执行[31]、模型检查[32-33]等技术对 PLC 控制逻辑程序的代码或二进制文件[34]开展缺陷分析及安全验证，针对控制程序及 PLC Runtime 运行环境开展自动化灰盒 Fuzzing 测试[34-37]，发现其中存在的漏洞和安全隐患；设备运行阶段开展非侵入式异常检测，核心是发现针对工业过程控制的破坏行为，包括基于 PLC 交互网络流量[38-41]、PLC 运行的上下文常量参数(扫描周期、控制常量等)[42-45]、流程行为建模及规则匹配[46-47]、关联性行为执行特征[48-49]；设备运行阶段增加内嵌安全组件提升 PLC 抗攻击能力，包括结合物理模型验证 PLC 控制逻辑程序的远程证明[50]、利用影子堆栈实现 PLC 运行环境控制流完整性检查[51]、利用标准 PLC 编程语言实现工控设备可直接运行的轻量级密码算法软件库[52]、利用自动化漏洞定位和热修复技术实现 PLC 漏洞的实时补丁更新[53-54]；其他方面，针对 PLC 的安全事件审计取证[55-56]、蜜罐诱捕[57-58]也受到一定关注。

可以看出，目前的 PLC 安全增强主要靠设备非运行期间分析源码或二进制代码，运行期间则主要依赖对流量或控制行为特征的异常检测。针对新型工业控制器依赖的异构载荷一体化集成、嵌入式虚拟化等运行场景，需要考虑前述中“确定性执行与不确定性执行、封闭隔离与开放互联、可用性优先与机密性优先”这几个现实矛盾体，回答“在计算载荷资源和系统复杂度显著提升的情况下，如何从内置系统安全结构的角度重新构建安全防护体系，实现三个矛盾体的消解统一”这个智能可编程工业控制器安全增强所蕴含的核心科学问题与底层设计挑战。

近年来，工业控制领域开始借助混合关键性系统(Mixed-Criticality Systems，MCS)[59-60]的资源安全隔离架构设计不同关键等级应用在同一 SoC 平台运行的新型工业控制器[61]。MCS 系统多应用在智能汽车、航空航天、空间探测等多任务多应用复杂工业装备，随着智能可编程工业控制器将实时操控、数据处理、智能分析等不同故障容忍等级应用集成化部署，两者的软硬件运行环境和安全理念有趋同性发展趋势，可以说基于 MCS 虚拟监控器(Hypervisor)的资源分区隔离思想已经成为智能可编程工业控制器安全增强的基础性框架并由此衍生出若干技术方案[62-64]。本文将延承这一技术脉络，梳理新型工业控制器实现嵌入式资源安全隔离的多种技术思路并分析对比，探讨嵌入式虚拟化背景下的 IPIC 安全增强实现路径。具体文章组织结构如下：第 2 节介绍目前针对 PLC 漏洞攻击的安全防护技术，按照发挥防护功效的时间阶段分为前置类型和运行时类型，其中运行时安全防护类型又可具体细分为非侵入式异常检测、内嵌安全组件及其他相关防护技术等；第 3 节介绍具备实时性特征的嵌入式安全隔离技术，具体分为基于双系统内核的隔离技术以及包括基于宏内核 Hypervisor、基于微内核 Hypervisor、基于静态资源分配 Hypervisor、基于可信执行环境、基于轻量化容器等几种类别在内的嵌入式虚拟化技术；第 4 节探讨面向嵌入式虚拟化的 IPIC 安全增强实现路径，提出了一种基于虚拟化隔离的 IPIC 异常检测安全增强架构，同时讨论了该架构在实际部署时可能面临的技术挑战；第 5 节展望智能可编程工业控制器安全增强相关的发展趋势，总结全文。

2　现阶段 PLC 安全增强与防护技术路线

2.1　前置阶段 PLC 安全防护

前置阶段 PLC 安全防护着眼于 PLC 的代码安全性、运行时保护、数据隔离以及对外界威胁防御能

力等多维度安全需求,在 PLC 正式投入业务使用的前置阶段对其编程语言的结构特性、执行环境的运行机制以及与外部设备交互方式进行安全性分析。当前,典型的前置阶段 PLC 安全防护技术包括静态分析与符号执行、模型检查、自动化逆向工程和自动化灰盒 Fuzzing 测试等。

(1)静态分析与符号执行

Zhang 等人[30]基于静态程序分析方法提出 VETPLC 系统来自动化审查 PLC 代码以发现潜在安全隐患。与传统 IT 场景中主要关注程序逻辑和数据流不同,VETPLC 系统特别考虑了工业控制系统中的事件驱动特性和时间敏感性,其运用静态分析构建时间事件因果关系图(TECG),同时挖掘工业控制系统测试床数据以提炼时间不变量,进而利用这些信息生成时间事件序列用于 PLC 代码安全审查。静态分析可能导致误报,特别是在复杂的 PLC 程序中,而符号执行技术可以通过使用符号值代替实际输入来模拟程序的执行路径,从而揭露更深层次的程序行为。Guo 等人[31]基于符号执行提出了 SymPLC 方法。该方法针对 IEC 61131-3 标准中指定的 PLC 编程语言,如结构化文本(ST)、梯形图(LAD)和顺序功能图(SFC)等,将这些标准 PLC 语言作为输入,并在应用符号执行之前将其转换为 C 语言,以便生成覆盖每个周期性任务所有路径的测试输入。SymPLC 方法可以通过消除冗余的任务间交错执行(Interleaving Execution between Tasks)来减少测试用例的数量,并且能够处理由于任务交错执行引发的故障;但对于大规模 PLC 程序,符号执行的时间成本较大。

(2)模型检查

Canet 等人[32]将 PLC 标准程序语言的形式化语义直接编码到模型检查工具 Cadence SMV 中,以自动验证 PLC 代码的行为属性。与传统 IT 场景下更关注连续的系统行为不同,该方法着重考虑了 PLC 的周期性行为,包括于每个周期内运行的输入扫描、程序执行和输出更新三个阶段,同时构建了基于转换系统的程序操作语义并利用线性时序逻辑来形式化描述行为属性,进而全面验证包括不变性、安全性和活性在内的复杂属性。此模型检查方法能够提供严格的逻辑正确性验证,但需要大量计算资源因而在处理复杂系统时面临可行性挑战。

(3)自动化逆向工程

Keliris 等人[34]针对通用 PLC 二进制代码的逆向工程提出了一种结构化方法——ICSREF。其通过对 PLC 二进制代码进行指纹识别来确定代码作者归属,同时提取了在 IT 场景逆向工程中并不关注的和物理过程控制相关的语义信息(包括物理输入/输出操作信息)来用于增强工业控制系统的安全性。该方法还可以在源代码不可用的情况下实现源代码恢复与二进制代码重用。ICSREF 提供了对 PLC 程序的深入洞察,但对于专有编译器和闭源 PLC 程序的逆向工程存在一定的复杂性和困难。

(4)自动化灰盒 Fuzzing 测试

PLC 控制程序通常在实时约束下运行,并且其输入输出(I/O)与物理过程紧密相关。这要求针对 PLC 的 Fuzzing 测试技术能够操纵 I/O 并重新利用 PLC 二进制代码,以适应这些实时约束和特定的 I/O 处理需求。Tychalas 等人[35]基于自动化灰盒测试技术 ICSFuzz 向 PLC 控制程序输入异常或随机数据来发现潜在编程错误与安全漏洞,并在 PLC 控制程序中嵌入额外的监控或控制机制以便在 Fuzzing 期间收集详细的运行时信息,进而更深入了解 PLC 控制程序在处理异常或随机输入时的行为。ICSFuzz 针对特定平台二进制文件的模糊测试存在局限性,需要复杂的逆向工程分析。继此之后,Tychalas 等人[36]又继续针对 PLC 固件中的操作系统和控制逻辑进行模糊测试,提出一种新的 Fuzzing 测试框架 IFFSET。IFFSET 通过使用 QEMU 模拟 Linux-based PLC 固件运行时环境,借助 AFL(American Fuzzy Lop)工具检测 PLC 固件中的漏洞,并与实际设备上的 Fuzzing 测试结果具有一致性,克服了实际设备无法离线测试的限制。由于二进制文件闭源性和功能信息的缺乏,IFFSET 针对特定型号 PLC 设备的 Fuzzing 测试无法直接使用动态插桩技术,仍然需要逆向工程辅助。Bytes 等人[37]通过工业网络对工业自动化系统中 PLC 的运行时进行原位黑盒 Fuzzing 测试,所提出的 FieldFuzz 方法不依赖目标 PLC 的源代码或者内部结构可见性,而是利用网络流量发送随机或异常的数据输入到正在运行的 PLC 系统中,实现了对控制逻辑程序和运行时组件的远程漏洞挖掘。FieldFuzz 通过与 PLC 运行时组件进行远程交互,并利用 Ghost 监视器进行覆盖率分析,克服了 ICSFuzz 等存在的输入交付和状态监控的限制,从而实现了一种高自动化、高可靠性、高性能的 Fuzzing 测试框架.

2.2 运行阶段 PLC 安全防护

运行阶段安全防护措施主要在 PLC 实际业务运行时确保其正常运行,按照防护技术实施与发挥作用的位置可大致分为 PLC 外置与内置。外置安全措施主要借助部署在图 2 典型 PLC 系统结构中

的工业网络与 PLC 上位机中的非侵入式异常检测系统，在不显著修改 PLC 内部程序的前提下，从工业控制系统的各类设备与网络等多个维度采集 PLC 现场数据进行分析，进而有效识别针对 PLC 的恶意攻击行为；内置安全措施则主要在图 2 中 PLC 控制程序或固件层面构建内嵌安全组件，增加代码远程证明、完整性检查、密码运算等安全功能模块，着力于从 PLC 内部提升其抗攻击能力；除了上述典型的 PLC 内外部安全防护技术，还可利用蜜罐诱捕技术捕获 PLC 攻击样本执行后续安全分析，并在安全事件发生后进行审计取证追踪定位安全威胁来源。鉴于 PLC 与物理过程的紧密耦合性及对实时性的严格要求，所提出的这些运行时安全防护措施不仅需要为 PLC 提供安全性，还要确保措施本身不会对 PLC 实时性能造成显著损耗.

2.2.1　非侵入式异常检测

(1) 基于 PLC 交互网络流量

Hadziosmanovic 等人[38]提出一种针对 PLC 网络流量进行分析与建模的语义安全监控方法。该方法的核心在于深入分析 PLC 间的通信模式与数据流，从中提取并构建工业操作过程的语义模型，进而检测出偏离预期行为的异常模式。该方法的创新性在于其并不依赖于 PLC 的具体代码或过程描述，而是通过分析网络流量中的数据来重建过程变量间的映射关系，不仅能够检测针对静态配置参数的直接过程控制攻击，还展示了通过模拟非静态变量的预期行为来检测更复杂的间接过程控制攻击的潜力。此方法的局限性在于对某些属性数据由于项目文件描述的非标准化及潜在歧义性，从项目文件中提取语义信息存在难度，这要求更精细的粒度以捕捉深层语义。此外，尽管该方法在实验室环境中评估表现出色，但在现实世界的应用中可能会遇到诸如不规则性、语义不匹配和人为干预等问题，这些问题仍需要仔细处理。Feng 等人[39]融合包级别与时间序列级别的分析策略进而提出一种多级异常检测框架。此框架首先通过分析工业控制系统中正常通信模式以初步构建基线签名数据库，随后利用 Bloom Filter 技术存储该数据库并进行数据包级别异常检测，最后借助长短期记忆(LSTM)网络开展时间序列级别异常检测。该方法的多级监控方案由于无法过滤物理过程控制变量的噪声行为导致增加了异常检测的误报率。Caselli 等人[40]通过深入建模与分析网络通信及系统变量来识别入侵行为，进而提出一种基于序列分析的工业控制系统异常检测方法。该方法采用分层架构进行细粒度的行为描述，并结合马尔可夫链与有限状态机等技术来构建模型用于执行异常检测。此研究的创新之处在于其不仅关注单一异常事件，而是分析事件序列对 ICS 设备行为的影响，通过考虑事件的时间顺序和概率，提高了对复杂攻击模式的识别能力。局限之处在于对特定类型变量进行建模以及处理随机延迟和人为操作时，可能会引入噪声从而导致检测准确性降低。Meng 等人[41]从网络流量中提取 PLC 控制变量语义信息从而为 PLC 异常检测提供了一种新的技术手段。其所提出的 SePanner 框架利用多状态比较技术，通过捕获 PLC 在不同工作状态下的网络流量，可以识别出与控制变量状态直接相关的语义字段；同时通过单状态比较和乱序消息过滤，能够精确定位并移除干扰字段来提高语义提取的准确性。SePanner 具备良好的兼容性和可扩展性，不仅支持对多种专有二进制协议的语义分析，还能够适应未经系统解析的协议。

(2) PLC 运行的上下文常量参数

Yang 等人[42]提出的 PLC-Sleuth 入侵检测方法基于控制常量概念，通过分析 PLC 上位机 SCADA 日志中记录的传感器读数与控制命令间的相关性，实现了对攻击的高准确率检测及对受损控制回路的精确定位。PLC-Sleuth 融合了物理与网络的入侵检测特征，依托于控制系统的物理诱导不变量，结合数据驱动的结构学习算法，通过控制图构建和控制常量权重监测来识别异常行为。尽管 PLC-Sleuth 在实验室环境评估中展现了优异的性能，但在实际应用时面对某些复杂攻击模式，如协同隐蔽攻击，其检测与定位的准确性可能会受到影响。Yang 等人[43]对 PLC-Sleuth 方法进一步改进，通过系统地调整和优化检测窗口大小与阈值，以在保持 PLC-Sleuth 方法较低检测误报率和延迟的基础上最大化检测准确性。但检测窗口大小调整导致了对某些参数变化攻击的敏感度降低。Ahmed 等人[44]提出一种基于时间身份验证的 PLC 异常检测机制。该机制通过被动观察 PLCs 之间的网络通信模式与数据流，检测与 PLC 正常扫描周期(Scan Cycle)时间特征不符的行为从而识别潜在的恶意攻击。特别地，Ahmed 等人还通过在控制逻辑中注入随机延迟(水印)并查看水印是否反映在预期扫描周期时间上提出了一种 PLC 水印方法(PLC Watermarking)。此方法能够对利用欺骗或重放攻击篡改 PLCs 间指令的强大网络攻击者进行检测，但需要对 PLC 进行一定程度的修改或定制，其实施过程面临实时性和兼容性方面的挑战。基于类似的时间身份验证检测

思路,Formby 等人[45]以 PLC 的实时确定性执行时间即扫描周期为基准来检测控制程序是否遭到篡改,并通过应用变化检测技术来提高检测准确性。与 Ahmed 等人被动监控网络流量提取时间特征方法不同,Formby 的方法需通过主动对 PLC 进行诊断连接或修改控制程序增加扫描周期计数器来获取时间特征,因而其实时性和鲁棒性会遭受 PLC 硬件性能与外部因素制约。基于 PLC 运行时间特征进行异常检测的各类方法思路核心在于,PLC 扫描周期时间反映了其硬件和控制逻辑的独特特征,这些特征难以被攻击者复制或伪造。

(3)流程行为建模

Yang 等人[46]针对 PLC 固件中的负载攻击,构建合法 PLC 负载程序的运行时行为模型,将其与控制系统的规范进行对照后用于监测当前 PLC 部署程序的运行时行为。该技术主要通过分析 PLC 的输入/输出、网络通信等访问模式中的异常流程来识别潜在的恶意 PLC 负载程序。在面对高度复杂或多变的攻击模式时,其检测能力会受到一定限制。Liu 等人[47]提出的 ShadowPLCs 方案基于 PLC 控制程序提取检测规则。具体而言,该方法首先通过自动分析 PLC 程序,提取包括有效地址、有效值范围以及控制逻辑规则在内的流程行为关键参数,之后通过与 PLC 主动通信或对 PLC 网络流量被动监控,从不同维度观测异常流程行为,最终实现对工业过程控制攻击的远程检测。ShadowPLCs 的最大特点在于能够从不同角度检测攻击行为,提高了检测的全面性和准确性。但是在 PLC 使用加密或私有协议时,其适用性会受到一定限制。

(4)关联性行为执行特征

Ike 等人[48]开发了一种融合 SCADA 行为与物理模型的混合检测技术 SCAPHY。该技术通过建立 SCADA 系统行为与物理世界影响之间的关联性来检测工业控制系统潜在攻击,创新之处在于利用 SCADA 系统独特的执行阶段,如初始化和过程控制,来确定控制物理世界行为的有限合法集从而区分攻击者活动。在多种工业控制系统场景与攻击案例的评估中,SCAPHY 展现了较高的检测准确率和较低的误报率。但受限于实验评估条件和所使用数据集的规模,该技术无法完全覆盖所有可能的工业控制系统攻击场景。Aoudi 等人[49]提出了一种基于数据驱动无需依赖先验知识的异常检测方法。该方法能够在工业控制系统中实时监控传感器测量数据,并在监测到物理过程行为出现结构性变化时发出警报。此外,该方法还通过分析传感器测量的时间序列行为变化来识别隐蔽攻击,相较于传统的基于规则的检测策略,展现出更高的灵活性和适应性。由于对噪声较为敏感,此方法误报率较高。

通过对上述各类非侵入式异常检测方法的进一步分析,可得出现有方法所依赖的 PLC 现场数据源主要有三大类。第一类依赖 PLC 周期性执行期间的传感器状态[49]以及基于传感器读数与随之触发的控制命令之间的相关性[42-43];第二类依赖 PLC 的实时确定性执行时间(即扫描周期)的时间特性与变化趋势[44-45];最后一类依赖 PLC 执行期间与之交互的网络流量[38-41,47]。

2.2.2 内嵌安全组件

(1)控制程序远程证明

Salehi 等人[50]提出了确保 PLC 代码完整性的远程证明技术 PLCDefender。该技术融合混合远程证明与物理模型检查,采用挑战-响应机制对 PLC 的控制逻辑程序执行远程认证。PLCDefender 平均耗时处于毫秒级,其独特之处在于整个远程证明过程无需对系统的软硬件进行修改因此独立于设备制造商,这反映了其在实际工业环境中具备广泛应用潜力。该技术主要关注对 PLC 系统行为的操纵型攻击,对于网络攻击的防护还有待进一步研究。

(2)控制流完整性检查

Abbasi 等人[51]利用影子堆栈来监控和验证 PLC 控制流,以防止恶意代码篡改 PLC 正常执行流程。所提出的异步控制流完整性方案 ECFI 与传统的执行空间保护(ESP)和地址空间布局随机化(ASLR)不同,其专注于在不牺牲实时性能的前提下,通过非阻塞的细粒度 CFI 方法来保护 PLC 免受控制流劫持攻击,因而能够满足 PLC 对实时性的严格要求。但作为一套被动保护系统,ECFI 在检测到控制流违规时无法主动终止 PLC 运行中的进程,因而无法避免恶意程序对关键基础设施造成损害。

(3)密码库构建

Yang 等人[52]开发了一套专为 PLC 设计的轻量化密码库——PLCrypto。该密码库是首次针对商业现成 PLC 的 ST 编程语言开发的,能够提供包括哈希函数、对称加密等 10 余种密码学算法,且不再需要额外的软件库和硬件辅助实现。该方案采用了硬编码等优化技巧以加速 PLC 场景下的逻辑和代数运算,为 PLC 基于专用语言独立实现内嵌密码服务提供了有益指引和探索。

(4)漏洞热修复

近年来,漏洞热修复技术逐渐受到物联网和工控设备安全更新领域的关注。Rajput 等人[53]提出

一种自动化漏洞定位和热修复方法 ICSPatch，该方法利用数据依赖图定位控制逻辑漏洞，并通过可加载的内核模块(LKM)在 PLC 内存中进行实时修复而无需中断系统的正常运行。相比于现有的热修复方法，ICSPatch 不需要重启设备，可以在运行过程中直接修补漏洞因而避免了停机时间。由于控制逻辑程序通常由专有编译器编译，ICSPatch 存在对 OT 供应商的依赖性。在 PLC 固件漏洞修复方面，Zhou 等人[54]提出了 RLPatch 框架，旨在无源代码情况下为第三方 PLC 固件部署实时补丁。该框架通过捕获 PLC 的实时条件和动态行为，识别主要不可恢复故障(MNRF)漏洞并生成热补丁。RLPatch 利用处理器的异常处理机制，在 PLC 扫描周期的空闲阶段插入补丁更新点，实现固件的实时更新。总体而言，RLPatch 的主要技术创新在于其精确的漏洞检测方法、非侵入式更新服务和自适应集成机制。不足之处在于需要对 PLC 进行逆向工程，可能违反制造商的服务条款，且固件更新时对硬件调试接口的依赖可能增加攻击面.

2.2.3　其他安全措施

(1)安全取证

PLC 设备由于异构硬件架构、专有固件和控制软件导致采用统一框架进行内存取证具有挑战性。Rais 等人[55]利用动态差分分析与字符串搜索技术，有效在 PLC 内存中提取了控制逻辑、日志和固件等关键信息。通过识别数据结构定义及应用规则，实现了在安全事件发生后对 PLC 内存转储信息的高效提取与分析，为 PLC 内存分析框架的制定奠定了基础。但由于在规则创建与验证阶段存在一定程度的主观性并依赖于先验知识，该技术针对特定型号 PLC 无法适用。

(2)蜜罐诱捕

由于 PLC 的闭源特性及供应商特定的专有固件，开发具有高交互性的基于软件模拟的虚拟 PLC 蜜罐面临重大挑战。相对而言，基于真实物理设备的 PLC 蜜罐，凭借其跨网络访问和高交互能力，有效解决了现有虚拟 PLC 蜜罐在交互性方面的不足。在此背景下，López-Morales 等人提出的 HoneyPLC 蜜罐方案[57]，通过模拟真实 PLC 设备与潜在攻击者进行高交互性通信，以收集关于攻击行为的详细信息。HoneyPLC 能够扩展支持多种 PLC 模型和制造商，但其针对特定 PLC 型号需要额外的配置文件编辑，这无疑增加了部署和操作的复杂性。HoneyPLC 在模拟特定型号 PLC 方面提供了深入的定制化能力，而 You 等人的可扩展、高交互物理蜜罐框架[58]则侧重于物理层面的交互和分布式部署。该框架旨在通过一对多机制，实现单个物理 PLC 利用多个远程代理对攻击者可见，而不在同一 IP 地址上重复暴露相同服务，从而提供了一种成本效益高、灵活且可大规模部署的物理蜜罐解决方案。

综上，图 3 全面总结了现阶段 PLC 安全增强与防护技术的具体分类情况。

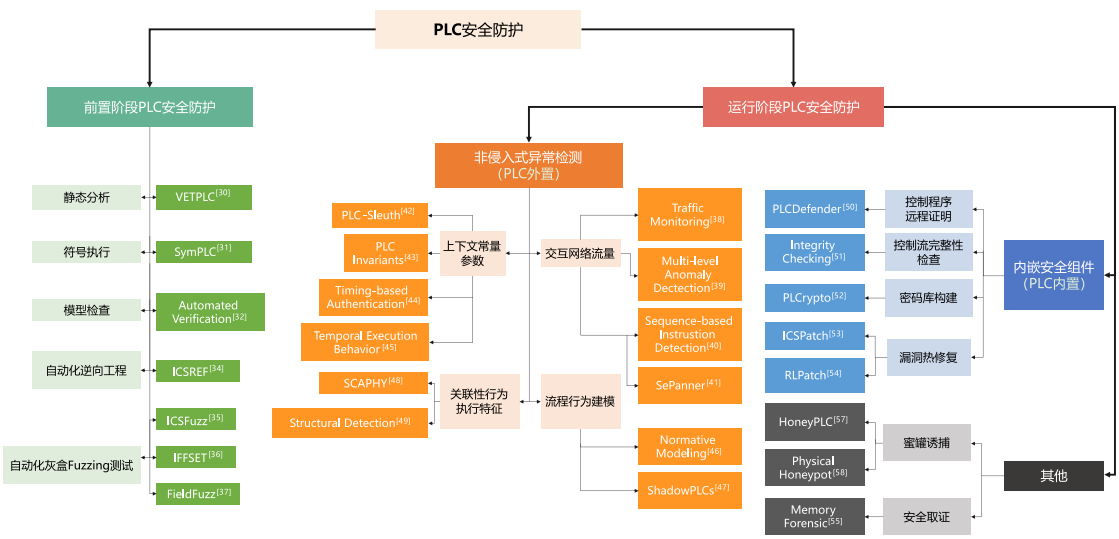


图 3　现阶段 PLC 安全增强与防护技术分类

3　具备实时性特征的嵌入式安全隔离

针对工业互联网嵌入式设备，现阶段提供实时性特征的安全隔离技术主要遵循两种设计思路。第一种思路涉及对现有通用系统内核进行实时性扩展，从而衍生出另一个独立的实时内核。在此框架下，两个系统内核实现相互隔离，并通过实施不对称的资源分

配策略,优先将 CPU、外设等资源分配给实时内核上的任务,以确保其实时性。第二种思路更为普遍,它依托于嵌入式虚拟化技术,通过在软件层面各个维度集成的包括虚拟化管理程序(Hypervisor)、容器引擎(Container Engine)及拥有类似功能的可信执行环境监视器(TrustZone Secure Monitor)等在内的专门软件层,实现对运行在这些软件层之上、目标和需求各异的多个运算负载间的相互隔离。借助嵌入式虚拟化的可扩展性与灵活定制化特性,该思路不仅能够实现运算负载的动态增减,还能为实时负载分配高优先级甚至可独占的硬件资源,从而提供其所需的实时性能。对于新兴工控领域,上述安全隔离技术已经在实时控制系统及新型工业控制器上进行了系统性测试与评估[65-66],验证了这些技术可以在优先确保工业控制任务实时性前提下提供安全隔离功能。目前已有部分新型 PLC 产品基于嵌入式安全隔离技术进行系统构建。例如,中科时代推出的 SP70 系列工业控制器[67]采用了双系统内核隔离技术;西门子 S7-1500[68]、贝加莱 Automation PC 2200/910[69]、和利时 EIC5202[70]以及汇川技术[71]与东土科技[72]的部分新型工业控制器则利用了嵌入式虚拟化技术。这些具备实时性特征的嵌入式安全隔离技术为智能可编程工业控制器 IPIC 上的异构载荷融合部署与安全增强奠定了重要技术基础。

3.1 基于双内核的操作系统内核隔离

基于通用操作系统扩展改造进而具备实时任务功能是工业界为了同时兼顾功能复杂性和任务确定性所采取的一种折中方案,双内核机制(Dual cores)是其主要技术路线之一。这种方法通常在底层部署一个小型实时内核,并将通用操作系统作为低优先级的非实时任务处理。RTLinux[73]、RTAI[74]和 Xenomai[75]是双核机制的典型代表,适用于很多工业实时控制应用(如运动控制)中的工业控制器。由于进程调度、中断处理等机制对系统时间行为造成显著影响,标准 Linux 内核的系统时间特性是不可预测的。RTLinux 在标准 Linux 内核与硬件平台之间增加了一个虚拟层(实时内核层),为实时任务提供直接访问和控制硬件的机制,以保证减少延迟;同时把标准内核的优先级设为最低,将其作为实时内核的一个进程与用户的其他实时任务一起调度,只有当实时内核上的关键任务不繁忙时标准内核才能够占用剩余 CPU 资源。RTAI 和 Xenomai 通过添加一个实时硬件抽象层 ADEOS[76]来跟 Linux 一起工作,ADEOS 对系统中断进行统一管理并作为通信媒介完成对 Linux 硬件的控制。不同之处在于,RTAI 把中断处理主导权放在自身,直接处理实时任务触发的中断,非实时中断才交给 ADEOS,这就减少了一部分实时开销,而 Xenomai 则是完全将各类中断转交 ADEOS 处理,实时性能比 RTAI 略差[77]。双内核方法的设计核心在于将传统 Linux 内核作为一个常规进程运行在另一个专门设计的实时内核之上,因此其安全隔离主要依赖通用操作系统基于 MMU(Memory Management Unit)的进程间隔离机制,双核各自的系统组件安全级别理论上没有区别,都面临主体操作系统被攻击者获得内核态权限的安全威胁.

3.2 基于宏内核嵌入式虚拟化隔离

宏内核(Monolithic Kernel)是操作系统设计中的一种类型,它将操作系统的多个核心组件和服务紧密集成到一个单一的、大型的内核进程中。在宏内核设计中,内核负责处理所有的系统服务,包括文件系统、设备驱动程序、网络堆栈、进程调度、内存管理等。虽然宏内核设计理念有悖于嵌入式虚拟化的“轻量化”“实时性”目标,但现有部分嵌入式 Hypervisor 仍基于宏内核实现。其中,按照内核功能集成度又可进一步将它们划分为两类:完全宏内核 Hypervisor 和部分宏内核 Hypervisor。完全宏内核 Hypervisor(例如 Xvisor 和 VMware ESXi Embedded 等)使用一个完全单一的软件层来负责主机硬件访问、CPU 虚拟化和客户机 I/O 模拟。部分宏内核 Hypervisor(例如 Linux KVM)通常是一个通用目的宏内核操作系统(包括 Linux、FreeBSD、NETBSD、Windows 等)的扩展。他们在操作系统内核支持主机硬件访问和 CPU 虚拟化,并通过用户空间软件支持客户机 I/O 模拟。

Xvisor[78]是一个支持全虚拟化和半虚拟化技术的 Type-1 型的完全宏内核 Hypervisor。其作为一个嵌入式系统可用的轻量级 Hypervisor,以单一软件层(即 Xvisor Hypervisor)实现了包括 CPU 虚拟化、IO 虚拟化、中断虚拟化、管理服务等所有关键组件,同时仅需很小内存占用和系统开销。Xvisor 通过为其自身分配最高特权级的 vCPU 确保客户机系统的 I/O 访问服务等能够及时得到处理。与 KVM 等依托通用操作系统的部分宏内核 Hypervisor 不同,Xvisor 的上下文切换非常轻量级。为了使 Xvisor 进一步具备为嵌入式系统提供实时虚拟化的能力,

Xvisor的实时性增强版本——Xvisor-RT[79]被提出，其通过引入支持 vCPU 实时调度的算法，能够为具有实时约束的任务/虚拟机提供确定性的执行时间，这对于需要高度可靠性和确定性执行时间的工业控制与自动化场景是非常必要的。

VMware ESXi[80]是 VMware 公司的一款企业级虚拟化产品，它同样属于 Type-1 型的完全宏内核 Hypervisor。ESXi 无需借助任何通用操作系统，直接与物理硬件进行交互，因此能够更加效率且充分地利用企业硬件资源（包括 CPU、内存、存储和网络等）。ESXi Hypervisor 的核心在于其内部集成了一个底层操作系统，称为“VMkernel”。VMkernel 是由 VMware 开发的类 posix 操作系统，提供资源调度、I/O 堆栈和设备驱动程序的核心功能。为了对嵌入式系统及设备提供虚拟化支持，VMware 还发行了 ESXi 的嵌入式版本——ESXi Embedded。ESXi Embedded 作为一个轻量级的 Hypervisor，对 ESXi 进行了部分功能优化与裁剪（包括无盘部署、启动过程优化、特定 I/O 处理等），其资源占用更小，可高度定制，因而能够确保与嵌入式设备的紧密集成和最佳性能。Frank[81]利用 MiBench 嵌入式微处理器性能基准测试套件仿真测试了将 ESXi Embedded 用于汽车和工业控制系统时可能带来的虚拟化开销。测试结果显示，在 ARMv8 ISA CPU 上运行少于 3 个虚拟机的情况下，ESXi Embedded 能提供与本机性能接近的开销。

KVM（基于内核的虚拟机）[82]是一个支持全虚拟化和半虚拟化技术的部分宏内核 Hypervisor。KVM 通过扩展 Linux 系统的两种执行模式，增加新的“客户机模式”来实现 Hypervisor 功能。该模式允许客户机操作系统与主机系统共享执行环境，同时对客户机特定指令和 I/O 进行拦截处理。KVM 依赖主机 Linux 系统在内核空间实现 CPU 虚拟化，并使用 QEMU 在用户空间为客户机提供 I/O 服务。由于使用完整 Linux 内核作为 Hypervisor，KVM 整体的上下文切换开销较大。此外，KVM 虚拟化隔离的安全性大大依赖于有较大攻击面的主机 Linux 内核的安全性。KVM 可以作为嵌入式的实时虚拟化解决方案，但需要对其进行定制和优化，以满足工业控制等系统的实时性与确定性需求。Zhang 等人[83]通过在 KVM 中应用实时性补丁，使运行在 KVM 上的 RTOS 客户机（VxWorks）实现了亚毫秒级的中断响应延迟。

3.3　基于微内核的嵌入式虚拟化隔离

微内核（Microkernel）概念主要相对宏内核最早于 20 世纪 80 年代提出，其设计理念提倡内核中的功能模块尽可能少，只包含任务调度、中断处理、进程间通信等基础能力，其他如存储器管理、文件管理等功能则被移出内核变成和用户进程同等级的服务进程。从安全角度出发，微内核秉承最小化可信计算基（Trusted Computing Base，TCB）设计理念，降低系统复杂度缩小系统受攻击面，很多面向高关键级工业应用的实时操作系统基于微内核安全理念实现，例如 QNX、PikeOS 等。嵌入式虚拟化和微内核在实时性、系统精简性、安全性等方面的需求高度契合，使得嵌入式 Hypervisor 实现方案开始借助微内核架构实施，其中大致分为两类技术路线。一类是在基于微内核架构实现的 RTOS 上做功能扩展使其具备嵌入式虚拟化管理程序（Hypervisor）能力，比较有代表性的包括：PikeOS Hypervisor[84]、QNX Hypervisor[85]、Wind River Hypervisor[86]等，这种 RTOS-based Hypervisor 都是从一种基于微内核底层架构的 RTOS 系统发展而来，其属于 type-1 类型但和 RTOS 共享微内核组件，上层实现可匹配多种资源划分策略：可以是面向实时高关键级应用的基于优先级的抢占式调度策略，也可以是面向非实时低关键级应用的基于时间轮询调度策略。另一类是从头专门设计实现基于微内核架构的 Microkernel-based Hypervisor，例如 Xen、NoVA、OKL4、ACRN 等，它们通常只提供最基本的主机硬件访问和 CPU 虚拟化功能，而依赖于用户空间组件或者一个专门的管理虚拟机来支持对整个主机的硬件访问、客户机 I/O 模拟和其他服务。

Xen[87]是一个标准 Type-1 型的轻量级的微内核 Hypervisor，其支持全虚拟化和半虚拟化。Xen Hypervisor 本身提供 CPU 虚拟化和基础的主机硬件访问，而客户机 I/O 模拟、后端处理、设备驱动等在一个称为 Dom0 的管理虚拟机当中实现。Dom0 是一个特殊类型的虚拟机，也被称为“控制域”（Control Domain），拥有管理其他虚拟机（DomU）的特殊权限和能力，其运行着一个 Linux 内核的修改版本。Dom0 的主要目的是利用 Linux 内核来提供对客户机的 I/O 虚拟化支持，同时对客户机进行管理。为了使 Xen 更契合对实时性要求较高的嵌入式虚拟化场景，专门为嵌入式实时平台设计的 RT-Xen[88]被提出。与标准的 Xen 相比，RT-Xen 进行了一系

列的优化(主要包括实时调度算法、锁机制、中断处理、确定性操作等方面),以满足工业控制设备对性能、响应时间和确定性等方面的严格要求。

NoVA[89]区别于针对传统宏内核架构 Hypervisor 的解耦合改造优化[90],从降低安全攻击面的角度提出了更轻量化的虚拟化管理框架,将资源分配管理程序、虚拟机监控器、设备驱动程序等原本在 Hypervisor 中的组件进一步分解出来在用户态实现,立足最小化可信计算基将 Hypervisor 中的高特权级组件以 microhypervisor 方式分拆独立实现,本身只包括 9000 行代码实现的内存管理单元、调度定时、中断控制、异常处理等核心功能。NoVA 支持利用可信计算和远程证明等技术对 Hypervisor 启动阶段的完整性进行防护,但对 VM 客户操作系统(GuestOS)、设备驱动以及 VMM 等组件不提供额外防护,其 microhypervisor 机制可最大程度缓解这些组件受到恶意攻击后对系统整体可用性造成的影响。

NoVA 借助 CPU 硬件辅助实现了全虚拟化方案,OKL4 Microvisor[91]则采用半虚拟化方案,旨在针对微内核架构带来的组件间高频交互负载优化系统交互响应性能,具体采用异步虚拟中断请求(vIRQ)代替同步进程间通信(IPC),半虚拟化方案不再需要实现指令模拟而是直接将特权指令映射为 Microvisor 提供的调用接口 hypercalls 进行后继处理。

ACRN[92]是英特尔针对 X86 架构完全从零开始设计的一个轻量级嵌入式虚拟化 Hypervisor,其不依赖任何已有的 RTOS 操作系统扩展实现,和已有的 type-1 类型嵌入式虚拟化 Hypervisor 比较,其充分借助了 X86 的架构资源,支持丰富的 I/O 虚拟化机制,包括完全模拟、辅助虚拟化、设备直通等。不同于宏内核 Hypervisor,ACRN 没有在 Hypervisor 中包含设备驱动管理模型(Device Model,DM)功能,而是将 DM 在一个专用虚拟机 service VM 中实现,这样即使 DM 相关功能崩溃也不会影响 Hypervisor 的健壮性和稳定性。

RTOS-based Hypervisor 和 Microkernel-based Hypervisor 在微内核架构利用方面并没有显著的技术差异,前者更多是由主流 RTOS 厂商从商业利益角度采用的一种技术捆绑策略。微内核架构对嵌入式虚拟化及单一 SoC 下实现混合关键系统具有重要影响。对于工业控制系统而言,为了确保 IPIC 中工业控制任务的实时性能,一种可被用于 IPIC 基于微内核 Hypervisor 划分差异化功能域的安全隔离架构如图 4 所示。该架构通过为执行 PLC 实时控制任务的 RTOS 分配专用 CPU 核心和直通外设等来提供实时性基础。

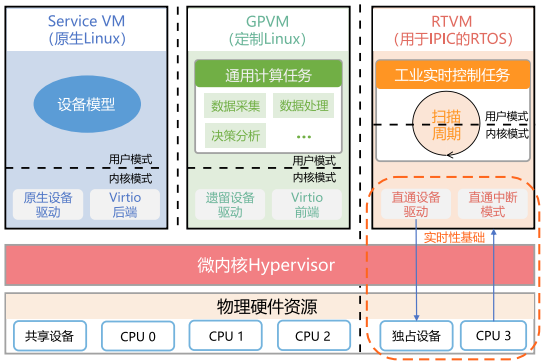


图 4 用于 IPIC 的基于微内核 Hypervisor 的安全隔离架构

3.4 基于静态资源分配的嵌入式虚拟化隔离

多核架构的集成 SoC 芯片为嵌入式虚拟化提供了更充足的资源分配条件,针对工业控制、智能汽车等对实时性和硬件隔离性有严格需求的场景,基于静态资源分配的虚拟化隔离方案(Static Partitioning Hypervisors,SPH)能够提供鲁棒性更强、多系统互影响更小的实现方式。传统嵌入式 Hypervisor 大多都提供针对客户操作系统(GuestOS)的硬件虚拟化(例如一对多的物理到虚拟 CPU 映射和调度、各类 IO 设备虚拟化映射),多个 GuestOS 共享 Hypervisors 管理的硬件资源,而 SPH 模式下不提供这些硬件资源虚拟化动态共享机制,系统在初始化阶段通过预配置方式实现物理 CPU 和虚拟 CPU 的一对一映射、多 GuestOS 对物理内存的静态分配、设备直通访问、直接中断注入等独占资源模式,且初始化后非特殊情况下严禁 Hypervisor 再介入资源调度造成额外延迟,以下介绍几个代表性 SPH 方案。

Jailhouse Hypervisor[93]是西门子提出的轻量级虚拟化工具,核心功能是借助硬件虚拟化扩展实现资源强隔离,其不提供完整的虚拟机管理和抽象功能,也不实现设备驱动的模拟仿真功能,不同客户虚拟机之间不共享任何 CPU,因此也没有复杂的调度器及调度算法设计。Jailhouse 将硬件资源进行静态分区,每个分区称为一个单元(Cell),每个 Cell 拥有自己的硬件资源(CPU、内存、外设等)且互不干扰。Jailhouse 有一个称为 Root Cell 的主控单元,这是一个特权 Cell,内部运行 Linux 系统,负责 Jailhouse Hypervisor 启动但并不具备全部硬件资源的控制权限,其他 Cell 统称为 Non-root Cell,在各自独立空间内部可运行实时操作系统或裸机应用程序(Bare Metal)。当系统启动时,Jailhouse Hypervisor 会先行加载运行,其在完成各个 Cell 单

元创建的同时相关硬件资源会被直接划分至对应 Cell 中。在系统启动后，Cell 隔离单元之间通过共享内存实现信息通信。尽管静态分区方法提供了强大的 CPU 和内存隔离，但仍无法避免一些微架构资源在分区之间保持共享，例如内存总线带宽、LLC 缓存等，另外非实时 GuestOS 的 I/O 负载增加（例如网络流量、磁盘等），会导致总线拥塞或缓存冲突，进而影响实时系统的性能和确定性。

Bao Hypervisor[63] 是一个完全从头实现的轻量化裸机应用程序模式的静态分区方案，其虽然遵循 Jailhouse 的静态分区架构，但除了执行底层平台管理的固件，不依赖任何第三方系统组件。该方案的特色在于分区时空隔离优化和微架构资源共享冲突消解，其使用两级转换静态分配内存，利用递归页表映射机制避免了对物理内存的连续完整映射，支持原生的页表着色机制，实现 LLC 缓存分区，借助虚拟到物理 CPU 的一对一映射和独占式 CPU 分配实现了完全逻辑时间隔离。中断虚拟化方面由于 ARM 架构下通用中断控制器 GIC 的限制，目前采用 Hypervisor 中断转发并在分区系统中再注入的方式实现，带来一定程度的中断延迟和性能降低。

seL4 是一个经过安全形式化验证的轻量级微内核，CAmkES（Component Architecture for Micro Kernel-based Embedded Systems）项目[94] 提供了面向微内核的嵌入式系统组件开发架构，支持在 X86 或 ARM 指令集下基于 seL4 实现虚拟化客户操作系统初始化启动和运行时管理。其虚拟机监控器（VMM）在初始化阶段为每个客户操作系统提供固定的资源分配方案（包括内存空间和 CPU 分配且仅支持虚拟到物理 CPU 的一对一映射）并支持客户操作系统直通设备。CAmkES VMM 保证每个客户机都有一个专用的 VMM 实例，如果 GuestOS 受到攻击并侵入到 VMM 中，借助内存管理中的二阶段地址转译可实现虚拟机间的完全隔离。该方案牺牲了传统虚拟化对资源动态配置的需求，借助 seL4 严格的形式化证明和硬件辅助虚拟化配置提升静态分区的安全隔离和抗攻击效果。

基于静态资源分配的 Hypervisor 通过为每个独立虚拟机分配固定的静态资源来提供实时性能，因而与工业控制 IPIC 深度兼容。使用静态资源分配的嵌入式虚拟化技术可基于如图 5 所示架构为 IPIC 划分差异化功能域并执行安全隔离，同时确保工业控制实时性。

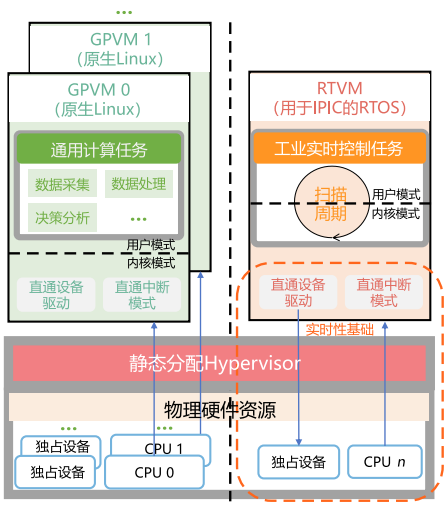


图 5　用于 IPIC 的基于 Static Partitioning Hypervisor 的安全隔离架构

3.5　基于可信执行环境的嵌入式虚拟化隔离

TrustZone 是 ARM 架构从可信执行环境（Trusted Execution Environment，TEE）视角提出的一种硬件辅助安全虚拟化技术，其针对每个物理处理器内核构造两个虚拟运行核，分别称之为安全世界（Secure World）和普通世界（Normal World）。通常情况下，GPOS 运行在普通世界，RTOS 或高关键级敏感应用运行在安全世界，SoC 上的硬件资源（例如内存和设备外设等）按两个世界进行划分，具体通过 SoC 芯片上 AXI 系统总线 NS 控制信号位、TZPC（TrustZone Protection Controller）和 TZASC（TrustZone Address Space Controller）等组件实现资源切分。其中，与基于通用 Hypervisor 的虚拟化技术利用 MMU 的两阶段映射实现内存隔离不同，TrustZone 可以基于 TZASC 划分安全世界内存区域与普通世界内存区域进而提供更强大的硬件级内存隔离机制。TrustZone 这种双系统虚拟化设置可以支持工业控制器的实时应用和非实时应用在一个 SoC 上集成部署，其突出 TEE 的安全设计为工业控制器上的高关键级应用构建了软硬结合的安全隔离方案。TrustZone 提供了 Monitor 模式负责两个世界间的切换，设计了安全世界优先的资源访问策略，Secure World 中的代码可以访问 Normal World 的系统内存但反之不允许，普通世界只能通过调用 SMC 指令（Secure Monitor Call）、FIQ 和 IRQ 中断请求、初始化配置的外部异常处理等方式进入 Monitor 模式进而和 Secure World 间接交互。作为 ARM 可信固件（ARM Trusted Firmware，ATF）的重要组件，Monitor 本身运行在安全世界且具备系统最高的

EL3 特权级，从虚拟化隔离角度 Monitor 起到了阻断和屏蔽来自 Normal World 应用对实时高关键级应用的影响干扰以及潜在的攻击路径，其上下文切换的任务和 Hypervisor 有一定的功能属性重合，因此涌现了一些基于 TrustZone 的嵌入式虚拟化隔离技术方案。

SafeG[95]针对 TrustZone 双系统架构设计明确了单核场景下安全世界通过 FIQ 中断优先抢占 CPU 资源，普通世界作为空闲任务处于低优先级执行的非对称资源调度策略。LTZVisor[96]进一步基于 ARM 硬件特性在 Monitor 模式下实现了一个完整的基于 TrustZone 的虚拟化组件，包括内存和设备分区、MMU 在两个世界的独立实现、两个虚拟核间通信交互、虚拟机控制块优化等。有些研究者提出 TrustZone 虚拟化路线不具备可扩展性[97]，仅针对单核架构提供了两个虚拟机空间，对多核场景以及多虚拟机实例缺乏明确支持路径。LTZVisor 增强版[98]针对双物理 CPU 场景，将安全世界和普通世界分别置于两个不同的 CPU，这种设计带来的优势一是突破了 TrustZone 缺乏多核模式实现路径的阻碍，二是充分利用多核带来的算力并行能力，由于两个世界分别运行在不同的物理 CPU，有效缓解单核模式下普通世界低优先级空闲抢占策略下带来的算力饥饿问题。RTZVisor[99]在单核场景下实现了任意个数实时操作系统虚拟机实例装载部署，但每次在普通世界只有一个 RTOS 虚拟机实例处于激活状态，其他未激活虚拟机实例的上下文信息保存在安全世界中。μRTZVisor[100]重构了 RTZVisor 的系统设计，增加了子系统间进程通信、可信执行环境中多安全任务间的细粒度分区隔离、多系统间上下文开销优化等。VOSYSmonitor[101]实现了符合汽车工业安全标准的混合关键系统 Hypervisor，将安全攸关的 RTOS 配置在安全世界运行，在普通世界安装 KVM 等虚拟机管理程序以支持多个虚拟机实例同时运行，从而在同一硬件平台上共同执行安全关键型实时操作系统和多个通用操作系统。TZDKS[102]实现了基于 TrustZone 的 GPOS 和 RTOS 双内核混合部署，双系统可基于时间切片共享物理多核，系统在初始化阶段进入安全世界且 RTOS 内核作为后继的系统运行主体，GPOS 作为没有实时任务发生时的空闲任务在普通世界被调用。ARM 低功耗芯片 Cortex-M 系列虽然也支持 TrustZone 机制，但其提供的硬件辅助组件和标准 TrustZone 架构有很大差异并且不提供 CPU 的 Monitor 模式，TrustZone-M Hypervisor[103]，将安全世界和普通世界分配独立 CPU 核并将 Hypervisor 划分为主从部分各自运行在安全世界和普通世界，实现了针对低功耗 SoC 的轻量级 TrustZone 嵌入式虚拟化方案，从方案扩展性上支持多核与多系统部署。TrustZone 架构侧重保护 IT 场景下的机密性和完整性，对 OT 场景下实时可用性缺乏系统性考虑。Wang 等人[104]面向高关键级应用的 CPS 系统，设计实时可信执行环境 RT-TEE 保障关键应用的实时可用性。RT-TEE 为每个外设提供两套驱动程序，一套具备全量功能的版本在普通世界运行，一套支持 OT 控制环路所需最小系统资源（安全感知和执行相关的 I/O 处理）能力的版本在安全世界执行；同时 RT-TEE 区别于传统 TEE 在普通世界实现全局任务调度器的技术路线，其在安全世界实现两级分层全局调度，实时任务相关调度在安全世界完成，其他非实时任务调度分配给普通世界执行。上述驱动消胀和分层调度机制可以在提高实时任务性能和控制 TCB 最小化间取得平衡。

整体来看，基于 TrustZone 的嵌入式虚拟化路线通过 ARM 硬件辅助体系架构，在安全隔离和信息交互两者间实现了较为高效的功能平衡，但由于其特有的双客户系统模式与多核多系统虚拟化场景融合过于复杂，因此在扩展性和灵活性方面受一定程度的能力限制。基于 TrustZone 的嵌入式虚拟化隔离通过将 RTOS 单独放置在安全世界并分配专用 CPU 核心与直通的安全外设等来确保其实时性能。为了给工业控制任务提供实时性，一种可用于 IPIC 划分多功能域的 TrustZone 协助虚拟化安全隔离架构如图 6 所示.

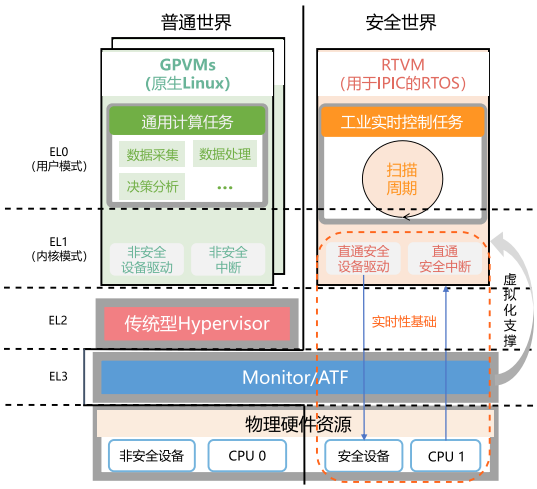


图 6 用于 IPIC 的 Trustzone 协助虚拟化安全隔离架构

3.6　基于轻量化容器的虚拟化隔离

在一些情况下，嵌入式混合关键性系统严格的内存占用要求可能需要一些更加轻量级的虚拟化方法。因此，基于轻量化容器的虚拟化解决方案开始在工业领域得到探索，尤其在工业实时控制领域，采用基于容器的虚拟化隔离成为一个活跃的技术探索方向[105]。在许多情况下，嵌入式混合关键性系统没有必要借助 VM 虚拟机来复制整个操作系统，特别是在不需要特定操作系统功能的情况下。基于容器的虚拟化目标就是利用容器替代 VM 虚拟机，在混合关键性系统间实现隔离，其占用空间更小，实时性能更高。容器作为一种轻量级的虚拟化技术，能够在单一主机上提供多个相互隔离的应用程序执行空间，其通过一系列的命名空间（例如 Linux namespaces）执行资源隔离，并通过系统资源管理组件（例如 Linux cgroups）使得每个容器都有唯一的文件系统和资源配额等。其中，对于容器间的内存隔离而言，与现有进程隔离机制一致，均是基于 MMU 实现。基于轻量化容器的虚拟化隔离在嵌入式平台当中也越来越受欢迎，目前已有部分嵌入式 Linux 内核操作系统（例如 BalenaOS、三星 Tizen 等）以及实时操作系统（例如 WindRiver 的 VxWorks）为在系统上运行轻量级的容器化解决方案进行了专门的适配。

一些通用的容器引擎技术，特别是 Docker[106] 和 LXC[107]，已经开始在工业控制系统中得到应用[108]，它们提供了轻量级的虚拟化解决方案，使得工业自动化和控制软件能够在隔离的环境中运行，同时简化了部署、管理和扩展的过程。Docker 作为一种实现了容器技术的代表项目，其优点在于启动快速、占用资源少、移植方便、可扩展性强等。Docker 里的每个容器实例都可以运行一个或多个应用程序，并且具有各自的文件系统、网络配置、进程空间等。Docker 通过 Linux namespaces 技术来进行资源隔离，并通过 cgroups 技术来进行资源分配，具有更高的资源利用率。同时，Docker 跟宿主机共用一个系统内核，不需要模拟整个操作系统，所以具有更快的启动时间。LXC 是比 Docker 更早出现的容器技术，它的设计同样是基于 Linux 的 cgroups 和 namespaces 功能，允许用户在宿主机上创建多个隔离的 Linux 系统环境。与 Docker 容器不同，LXC 更接近于传统的虚拟化技术，每个容器都提供了一个独立完整的 Linux 系统环境，可以运行任何 Linux 发行版，并且可以包含完整的 init 系统（如 systemd 或 init）。这使得 LXC 容器在功能上更加灵活，适合运行需要完整 Linux 环境的应用程序，但也意味着它们通常比 Docker 容器占用更多的资源，启动速度更慢。

Docker 等通用容器引擎虽然目前已经在工业控制场景中得到一定程度的应用，但它们的设计理念本质上更倾向于云端与传统计算架构。为了使容器虚拟化技术更好地与嵌入式和工业实时控制领域契合，一些嵌入式的软硬件厂商也陆续推出自己的容器引擎。iSulad[109] 是一个由华为自主研发的轻量级容器引擎，它被集成在华为开源嵌入式操作系统 openEuler Embedded 内部[110]。iSulad 和 Docker、LXC 等一致，都基于 Linux 内核的 cgroups 和 namespaces 特性，实现了对进程的隔离和资源限制，这也是目前大部分容器技术的核心。iSulad 作为一个轻量级的容器引擎，具有高性能、低开销的特点，尤其适合在资源受限的嵌入式设备和工业控制系统中使用。针对资源受限的嵌入式场景以及需要快速响应和实时处理的工业控制场景，iSulad 做了大量的优化。首先，iSulad 采用了 C/C++ 编程语言编写，相比使用 Go 语言编写的主流容器引擎（如 Docker 等），启动速度更快。其次，iSulad 在架构设计上进行了优化，提供了函数库调用的方式来加快运行速度。最后，iSulad 对资源占用进行了优化，针对嵌入式设备资源受限的特点，iSulad 在轻量化模式（Light Mode）下的内存占用率很低（小于 15 MB），结合特殊的轻量化镜像，能够实现极低的资源占用率。BalenaEngine[111] 是由 Balena 公司开发的一个专门为嵌入式场景设计的容器引擎。它旨在提供一个轻量级、高效且易于使用的容器解决方案，特别适合在资源受限的嵌入式设备上运行。BalenaEngine 在设计上与 Docker 引擎保持了高度的一致性，但剥离了 Docker 中一些如 Docker Swarm 等与嵌入式场景无关的功能。此外，BalenaEngine 引入了容器 Delta 的概念，这是一种计算两个镜像之间变化的二进制描述的方法。使用 Delta 可以大幅节省带宽，因为在两个镜像之间共享的任何文件都不会被重复传输。BalenaEngine 通过更有效地使用带宽，配备更小的二进制文件，并且更保守地使用硬件存储和内存，来与资源受限的嵌入式设备深度兼容。值得

一提的是，Balena 公司还提出了与 BalenaEngine 兼容的操作系统——BalenaOS，这是一个基于 Yocto Linux 的轻量级操作系统，它针对在嵌入式设备上运行容器进行了优化，尤其关注容器长时间运行的可靠性，这对于工业控制系统的稳定性和安全性至关重要。

当将容器虚拟化技术用于工业领域的嵌入式实时平台，为增强容器运行的实时性，除了容器本身需要加以修改（如嵌入式容器引擎 iSulad 和 BalenaEngine 做的努力），在大多数情况下，运行容器的主机操作系统（一般为 Linux 内核）也需要进行实时性改进，以真正达到“实时容器”的效果。当容器安装在通用操作系统上时，在容器内执行的任务最终将在尽力而为的策略下进行处理（因为它们受内核调度策略的约束），因此不符合工业控制系统的实时需求。学术界为此提出了以下两种操作系统改进方法以实现实时容器：(1) 使用实时性的协同内核与(2) 直接修改 Linux 内核调度机制使 Linux 本身具备实时性。RT-CASE[112] 便是基于协同内核思路构建的方案，它通过实施 RTAI 或 Xenomai 等双内核协同策略，实现了对宿主 Linux 内核的完全可抢占性，从而确保了实时性。在 RT-CASE 框架下，实时任务在实时容器（rt-case）中执行，由协同内核负责统一调度；而非实时任务则在非实时容器中运行，由 Linux 内核进行调度管理。在内核层面，Linux 内核及其承载的非实时容器/任务均可被协同内核及实时容器/任务所抢占。进一步地，RT-CASE 为每个实时容器都分配了关键性等级，确保关键性等级较高的实时容器和任务能够优先获得调度，并且不受关键性等级较低的实时容器干扰。协同内核在提供实时性的同时，也保留了容器引擎所提供的所有机制与工具。修改 Linux 内核调度机制是实现实时容器的另一种可行思路。两级分层调度[113] 通过修改 Linux 内核的 SCHED_DEADLINE 调度器，引入容器分层调度策略，使容器运行具备实时性。在经过修改的 Linux 内核中，第一层根调度程序负责选择每个 CPU 上要调度的容器。随后，第二层固定优先级调度程序在选定的容器中执行最高优先级任务的调度。这种基于分层的容器调度器实现具有一个显著优势：当虚拟 CPU（vCPU）的运行时间结束且 vCPU 受限时，调度器能够基于“推送”机制，将实时任务从该 vCPU 迁移至其他 vCPU，以实现任务的连续执行。

随着传统 PLC 逐步向 IPIC 转变，部分 PLC 厂商已经开始探索为他们的产品应用容器虚拟化架构。在此背景下，依据图 7 所示的架构，轻量化容器技术可以为工业自动化 IPIC 上不同关键等级的任务执行安全隔离，并确保工业控制系统的实时性。PLCnext 是由菲尼克斯电气公司提出的新一代支持工业 4.0 和智能制造实施的 PLC 架构，其代表了传统 PLC 技术与最新 IT 技术的结合，目前正在探索部署容器化解决方案在其新型的 PLC 产品当中[114]。基于轻量化容器的虚拟化技术与嵌入式和工业实时控制系统兼容，一些工作已经展示了嵌入式混合关键性系统借助容器化来执行安全隔离与实时任务的能力，但目前大部分工作仍停留在实验室环境中。在将这项技术安全地转移到工业自动化的新型 PLC 设备之前，需要进行更多的研究，特别是将容器虚拟化的所有不同方面（运行时、编排、网络）作为一个整体进行研究和验证。

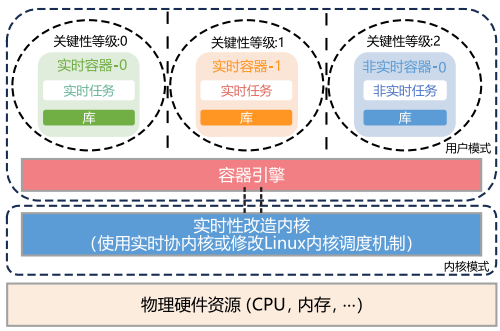


图 7 用于 IPIC 多关键级别任务隔离执行的实时容器架构

表 2 对现阶段嵌入式安全隔离方案的主要特性进行了归纳总结，包括所属类别、硬件架构支持、安全性与可靠性特征、实时性支持、内存隔离机制和硬件隔离强度等。其中，对于不同类别隔离技术的安全隔离性，双内核与轻量化容器隔离技术实现了进程级资源隔离，均面临主操作系统被攻陷后产生的安全威胁；基于 Hypervisor 实现的宏内核、微内核、静态分区等隔离技术以虚拟机为基本单位，提供了系统级资源隔离访问，减少了不同系统之间的相互干扰并阻断安全风险在系统间的传播；基于 TruztZone 协助的虚拟化隔离则借助 TZASC 与 TZPC 等安全硬件扩展实现了两个世界间资源的强隔离，在表 2 所有隔离技术类别中具备最高的硬件隔离强度。

表 2　现阶段嵌入式安全隔离技术总结

隔离技术类别	具体方案名称	支持架构	安全性与可靠性特征	实时性支持	内存隔离机制	硬件隔离强度
双内核	RTLinux[73]	ARM，x86	进程级资源隔离、优先级继承机制	采用实时协内核	MMU	***
	RTAI[74]	ARM，x86，PowerPC，m68k	进程级资源隔离、优先级继承机制、调度器实现具备原子特性			
	Xenomai[75]	ARM，x86，PowerPC	进程级资源隔离、优先级继承机制、不超过四个CPU核心情况下提供可预测实时性能			
宏内核 Hypervisor	Xvisor[78]/Xvisor-RT[79]	ARM，x86，RISC-V	系统级资源隔离	Xvisor-RT 中实时性调度机制	MMU（两阶段映射）	****
	VMware ESXi Embedded[80]	ARM，x86	系统级资源隔离、虚拟机镜像加密功能、自动重启、辅助虚拟机自动接管机制	—		
	KVM[82]	ARM，x86，PowerPC，s390	系统级资源隔离、热迁移	实时补丁[83]		
微内核 Hypervisor（RTOS扩展）	PikeOS Hypervisor[84]	ARM，x86，PowerPC，SH4，SPARCV8，LEON	系统级资源隔离、内置运行状况监控功能	对 RTOS 的扩展		
	QNX Hypervisor[85]	ARM，x86	系统级资源隔离、针对DMA攻击的 SMMU 防护机制			
	Wind River Hypervisor[86]	ARM，x86	系统级资源隔离			
微内核 Hypervisor（从头设计）	Xen[87]/RT-Xen[88]	ARM，x86，IA64，PowerPC	系统级资源隔离、热迁移、故障恢复	RT-Xen 中实时性调度机制		
	NoVA[89]	x86	系统级资源隔离、集成支持间接分支跟踪、监督者影子栈等控制流保护机制	—		
	OKL4[91]	ARM	系统级资源隔离、关键组件完整性检查与签名验证、虚拟机间加密通信支持、虚拟机运行状态监控功能	支持运行独占资源的RTVM		
	ACRN[92]	x86	系统级资源隔离			
静态分区 Hypervisor	Jailhouse Hypervisor[93]	ARM，x86	系统级资源隔离、死锁检测和自动重启	所有虚拟机各自独占静态资源		
	Bao Hypervisor[63]	ARM，RISC-V	系统级资源隔离			
	seL4 CAmkES VMM[94]	ARM，x86	系统级资源隔离、形式化验证内核、最小化 TCB、任务备份和恢复机制			
TrustZone 协助虚拟化	SafeG[95]	ARM	基于安全硬件扩展实现的强隔离、内置运行状况监控功能	实时性调度机制	TZASC	*****
	VOSYSmonitor[101]	ARM	基于安全硬件扩展实现的强隔离、故障检测与恢复			
	TZDKS[102]	ARM	基于安全硬件扩展实现的强隔离	实时性调度机制与设备直通模式		
	TrustZone-M[103]	ARM	基于安全硬件扩展实现的强隔离			
	LTZVisor[96]	ARM	基于安全硬件扩展实现的强隔离			
	RT-TEE[104]	ARM	基于安全硬件扩展实现的强隔离、I/O 参考监视器、驱动沙盒化执行与精简			
	LTZVisor-AMP[98]	ARM	基于安全硬件扩展实现的强隔离	RTOS独占资源		
	RTZVisor[99]	ARM	基于安全硬件扩展实现的强隔离			
	μRTZVisor[100]	ARM	基于安全硬件扩展实现的强隔离、I/O参考监视器、安全启动、细粒度资源访问控制、优先级继承机制			
轻量级容器虚拟化	Docker[106]	ARM，x86，PowerPC，s390x，MIPS	进程级资源隔离、容器镜像签名验证机制、容器运行状况监控与自动重启	—	MMU	***
	LXC[107]	ARM，x86，PowerPC，s390x，MIPS	进程级资源隔离、容器运行状况监控与自动重启			
	iSulad[109]	ARM，x86，SW64	进程级资源隔离、容器镜像签名验证机制、容器运行状况监控与自动重启			
	BalenaEngine[111]	ARM，x86	进程级资源隔离、容器运行状况监控与自动重启、备用容器自动接管机制			

安全与可靠性的其他方面，双内核类别中的 RTLinux、RTAI、Xenomai 均实现了优先级继承机制，能够避免优先级反转问题，确保高优先级任务能够及时获得所需资源，从而增强系统的容错能力。宏内核 Hypervisor类别中的VMware ESXi Embedded 相比 Xvisor 和 KVM 具有更加完备的安全与可靠性策略。具体的，ESXi Embedded 支持使用加密功能来保护虚拟机数据，防止敏感信息泄露；同时实现了 vSphere High Availability（HA）功能，在虚拟机故障时自动重启虚拟机；进一步地，通过 vSphere

Fault Tolerance(FT)功能可创建一个与主虚拟机同步运行的辅助虚拟机强化系统可靠性与容错能力。辅助虚拟机会实时备份主虚拟机的运行状态,并在主虚拟机发生故障时自动接管业务,因而能够为关键任务提供业务持续性和数据一致性保障。基于 RTOS 扩展的微内核 Hypervisor 类别中的 QNX 相比 PikeOS 与 Wind River 在针对恶意外设的直接内存访问(Direct Memory Access,DMA)攻击方面提供了更强的防护,其通过集成系统内存管理单元(System Memory Management Units,SMMU)管理服务模块与硬件 SMMU 同步运行,能够实现对 DMA 设备内存访问的安全限制策略。从头设计的微内核 Hypervisor 类别中的 NoVA 尤为关注系统漏洞利用和恶意软件攻击,内部集成实现了间接分支跟踪(Indirect Branch Tracking,IBT)、监督者影子栈(Supervisor Shadow Stacks,SSS)等控制流保护技术;OKL4 则更为关注系统组件完整性与系统间通信机密性,集成支持了完备的完整性和证明机制为系统关键组件提供完整性检查和签名验证,同时支持"Secure Messaging"功能为虚拟机间消息传递提供机密性保证(即使 OKL4 本身也无法访问这些通信数据)。静态分区 Hypervisor 类别中的 Jailhouse 从系统可靠性角度出发设计了虚拟机死锁检测和自动重启机制能够确保实时任务的连续性;sel4 则作为第一个在源代码级别上经过形式化安全验证的 Hypervisor,基于最低特权原则实现了最小化 TCB,能够极大减少潜在安全漏洞和攻击面。TrustZone 协助类别中的 RT-TEE 着力于构建全方位 I/O 防护框架,首先通过 I/O 参考监视器确保对安全外设的访问受到严格控制,其次借助驱动精简技术(Driver Debloating)移除复杂硬件交互过程转而使用固定 I/O 交互模板来减少 TCB,最后利用沙盒化技术(Sandboxing)防止非信任驱动行为对安全环境造成损害;μRTZVisor 则着力于提供系统完整性与细粒度资源访问控制,其实现安全启动过程以确保系统软件完整性和可信性,同时基于最小特权原则将所有资源访问通过能力(capabilities)进行控制来提供细粒度权限管理。轻量级容器类别的大多数方案均支持容器镜像签名验证、运行状况监控、自动重启等能够保证基本安全与可靠性的功能,值得注意的是,BalenaEngine 还支持故障转移机制,同步运行备用容器并在主容器实例故障时自动切换到备用实例来接管实时或关键任务,能够进一步强化系统弹性与可靠性。

4 基于虚拟化隔离的 IPIC 安全增强实现路径分析

当前,国内外诸多 PLC 公司纷纷推出了各自的 IPIC 产品,这些产品在引言部分有所概述。绝大多数 IPIC 产品都集成了旨在提升业务表现的功能增强方案,它们利用人工智能等新兴技术实现了业务流程的高效化运作。然而,针对 IPIC 安全性增强的实施方案却显得相对匮乏,这一点在现有文献和产品实践中均未获得足够的重视。用于传统 PLC 的现有安全防护技术也存在诸多"痛脚"难以满足 IT/OT 融合大背景下的 IPIC 安全增强及防护新需求。以 PLC 运行阶段的两类主要安全防护措施为例,非侵入式异常检测基于 PLC 运行过程产生的各类现场数据进行"数据驱动"的分析,却在离 PLC 相对较远的网络侧或上位机系统中部署,因此存在检测延迟和强大攻击者在 PLC 内部伪造"正常"交互数据流来对外部异常检测系统进行欺骗攻击的问题;内置安全组件在 PLC 内部集成实现能够提供及时安全功能服务,但存在与工业自动化任务抢占系统资源因而造成工业控制实时性能损耗大的问题。随着传统 PLC 逐步向 IPIC 迭代升级,嵌入式虚拟化隔离能够作为 IPIC 的基础安全架构(见图 4～图 7)兼容多种现有安全防护方案于内部进行集成化部署来为 IPIC 提供安全增强。现有 PLC 运行阶段两类安全防护技术的"痛脚"能够借助多核处理器 SoC 架构和虚拟化隔离技术得到有效解决。通过虚拟化可以为 IPIC 划分出执行工业控制载荷的业务功能虚拟机和执行安全增强载荷的安全功能虚拟机,两类虚拟机独立运行并借助虚拟化提供的高效共享内存通信实现消息传递,这一思路与当前在某些领域已经初步应用的混合关键性系统(MCS)理念"不谋而合".

4.1 混合关键性系统中的嵌入式虚拟化技术

在一个多处理器 SoC 中集成化部署多种功能载荷的背景下,混合关键性系统设计者逐步开始利用嵌入式虚拟化技术划分具有差异化关键性级别的功能域,并通过域间通信(Inter-Process Communication,IPC)机制实现安全增强的任务协作。在 DriveOS 系统[115]中,引入了基于静态资源分配的嵌入式虚拟化隔离技术,旨在构建一个集成的车辆管理系统。该系统设计允许实时域操作系统(Quest RTOS)与非实时域操作系统(Yocto Linux)在统一的车载计

算平台上共存，并通过静态资源分配策略（涵盖 CPU、内存和外设等）实施了严格的安全隔离。其中，实时域 RTOS 承担着处理关键实时任务的职责，而非实时域 Linux 则负责执行非关键性、传统型应用。同时实时域和非实时域能够借助安全共享内存通信模块（shmcomm）协同完成高级驾驶辅助系统（ADAS）任务。shmcomm 提供了一种安全、动态的域通信机制，可由 Hypervisor 根据实时需求动态地创建和销毁，并可以通过指定不同域对 shmcomm 通道的访问权限来实现细粒度访问控制。在 ADAS 任务协作框架下，RTOS 中的 CAN 网关服务实现车辆传感器数据的实时读取，并通过 shmcomm 通道将数据传输至 Linux。Linux 中的 ADAS 应用利用机器学习算法处理数据，完成障碍物检测、路径规划等，产生辅助控制命令。这些命令经 shmcomm 通道回传至 RTOS，由 ADAS 纵向控制器服务转换为对车辆执行器的控制信号。DriveOS 在硬件在环（HIL）仿真环境中进行原型设计和测试，相较于独立 Linux 系统，DriveOS 在处理车辆 CAN 消息时展现出更高的吞吐量和更低的端到端延迟，从而验证了其在实时性能方面的优势。同样，近期提出的 FlyOS 系统[116]亦采用了静态分配虚拟化技术，为多旋翼飞行器设计了一种集成模块化航空电子（IMA）飞行管理系统。在该系统中，与飞行安全和实时性能密切相关的飞行控制任务由 RTOS 实现，而对实时性要求不高的通用计算任务，如计算机视觉与图像处理、飞行数据记录等，则在 Linux 中执行。RTOS 与 Linux 之间进行了大量的任务协作与信息交互。例如，执行飞行任务期间，Linux 应用通过摄像头驱动捕获图像，并采用面部识别与跟踪算法生成控制命令，这些命令通过共享内存传输至 RTOS，以进行飞行控制实时调整。同时，RTOS 定期将飞行控制器的运行数据经共享内存复制给 Linux 的黑匣子应用以便于永久存储。在系统安全和可靠性方面，Hypervisor 实时监控 RTOS 发送的心跳信号，一旦检测到心跳信号丢失，即视为 RTOS 发生故障或受到攻击则立即触发故障恢复过程，将 RTOS 虚拟机退出并重新初始化，同时把飞行控制任务移交给 Linux 中的备用飞行控制器，备用控制器将临时执行对实时性要求不高的飞行任务，如悬停或安全着陆。FlyOS 在硬件在环仿真环境中的原型设计和测试证明了其与原始固件（Cleanflight）相比，具备更快的响应时间和更低的端到端延迟。在工业控制系统领域，Cilardo 等人[117]通过在多核 SoC 的风力涡轮装置控制器上应用嵌入式虚拟化来划分三个差异化功能域，同时通过域间协作实现了一种可用于风力发电厂的安全风险识别与预测框架。其中，三个功能域分别为执行工业控制回路的实时虚拟机，运行 AI 算法并由硬件加速器（如 FPGA）支持的硬件加速虚拟机，以及负责网络通信等的通用目的虚拟机。为了进行风险识别与预测，实时虚拟机周期性地将控制器的传感器数据通过域间共享内存传输给硬件加速虚拟机。硬件加速虚拟机利用状态监测和故障检测的深度学习算法对现场数据执行分析以自主识别潜在安全风险。一旦检测到风险情况，硬件加速虚拟机将通过共享内存向通用目的虚拟机发出预警，后者随后负责将预警信息传递给上位机，以实现风险的及时响应和处理。这种通过在非实时功能域直接嵌入一个高性能安全载荷并通过域间通信来为实时域提供安全功能服务的设计，有效地提高了风力涡轮装置的系统安全性和可靠性。上述三种系统架构的设计不仅彰显了嵌入式虚拟化技术在提升混合关键性系统实时性和安全性方面的潜力，而且通过融合嵌入式虚拟化和域间通信等技术，实现了不同关键等级任务在同一硬件平台上的安全高效协作，同时保持了系统的灵活性和可扩展性。

4.2　面向异常检测的 IPIC 安全增强框架

随着工业自动化技术的持续进步以及工业 4.0 理念的提出，混合关键性系统与智能可编程工业控制器（IPIC）的融合发展趋势日益显著[59-61]。在此背景下，混合关键性系统与 IPIC 在软硬件运行环境及安全理念上展现出显著的趋同性，进而推动了嵌入式虚拟化技术在 IPIC 中的应用。嵌入式虚拟化技术为 IPIC 提供了更为灵活的资源调配能力及更多样的任务执行空间，通过划分出不同关键级别的实时域和非实时域，可以确保实时域工业控制任务的确定性执行，同时允许其他异构载荷在非实时域按需灵活部署。从这个角度出发，可以把现有基于 PLC 各种现场数据执行的非侵入式异常检测方案作为一种安全载荷在非实时域中进行部署。之前各类安全功能需要在工业网络拓扑中加入专门的安全设备或系统（例如工业防火墙或工业入侵检测系统），这些专业系统需要通过数据探针来采集 PLC 运行产生的网络流量和历史日志数据，然后通过网络将数据传输至本地或云端的安全设备进行分析决策然后下发响应措施。这种处理方式会带来一定程度的网络开销和迟延，很多现场网络和业务运行数

据为了规避企业隐私泄露也不宜跨网上传,通过嵌入式虚拟化技术在 IPIC 非实时域中部署异常检测载荷可以有效解决这些问题。IPIC 实时域采集的现场数据可以采取本地域间通信方式直传给非实时域内的安全模块,借助多核架构比传统 PLC 多几个数量级的算力资源可以支撑异常检测载荷数据采集、清洗、分析、决策、触发响应等各环节执行。针对异常检测载荷的本地集成化部署可大幅降低网络通信开销并符合现场环境业务数据不出敏感区的隐私保护策略。

嵌入式虚拟化技术为 IPIC 中非侵入式异常检测方案的高效统一部署奠定了技术基础。然而,现有检测方案在异常发现后大多只采用告警机制,缺乏进一步的处置响应措施。这是因为 PLC 负责关键的实时工业控制任务,若在检测到异常后贸然进行硬件或系统重置,可能会严重干扰工业过程,导致生产中断或损失。针对 PLC 异常的即时处理策略是借助故障转移(Failover),采用冗余手段在主要 PLC 硬件或主要软件组件发生故障时自动切换到备用 PLC 硬件或软件组件来接管实时控制任务。例如,Yang 等人[118]通过四个硬件 PLC 实现异构冗余,这些 PLC 运行功能一致但异构的控制程序,两个一组构建了一主一备的 PLC bank。在每个扫描周期内,为所有 PLC 提供一致的物理输入并通过硬件选择器比较主要 PLC bank 的输出一致性,输出不一致时视为出现异常或故障,立即切换至备用 PLC bank 来控制工业过程,从而实现故障转移。然而,此方案需对硬件进行改造,增加了成本和实施难度,限制了其在工业控制系统中的大规模部署。为克服这一问题,Liu 等人[119]提出了 HRPDF 方法,该方法通过在 PLC 固件中增加运行时管理器(Runtime Manager),在每个扫描周期内,基于共享内存为多个隔离的异构控制程序的运行时提供相同物理输入,并在执行完毕后比较输出结果,采用"少数服从多数"原则确定最终输出,实现故障转移。这些冗余方法利用异构特性,如代码混淆和地址随机化策略,确保同一攻击针对所有硬件或软件组件不会同时有效,从而通过输出结果比对来发现故障组件并触发故障转移过程。尽管异构冗余架构能有效进行故障转移,但其实施过程中需要对 PLC 的常规扫描周期进行修改,增加了额外的对比决策和输入/输出数据拷贝阶段,对工业控制的实时性造成不小影响。此外,异构冗余思路适用的攻击范围有限,部分攻击利用控制逻辑程序[17]、工业网络协议[24-26]和 PLC Web 应用程序[28]的功能逻辑缺陷发起,具备同时针对所有冗余组件发起有效攻击的能力,能够使这种依赖输出结果比对的故障转移机制失效。因此,如何平衡实时性、安全性和实施成本,仍是 PLC 故障转移领域面临的重要挑战。嵌入式虚拟化技术同样为 PLC 故障转移所面临的挑战提供了新的解决途径。通过虚拟化技术所提供的备份与恢复能力,能够为 IPIC 实时域虚拟机创建功能一致但软件异构的备用虚拟机。当非实时域执行的异常检测载荷检测到当前实时域虚拟机发生异常或故障时,利用 Hypervisor 可以无缝切换至备用虚拟机,迅速接管实时控制任务,从而在异常响应阶段实现有效的故障转移。这一思路类似于 FlyOS 系统在实时域 RTOS 出现故障或受到攻击时,通过 Hypervisor 将飞行控制任务临时托管至非实时域 Linux。然而,对于工业控制任务而言,由于其对实时性和可靠性的严格要求,必须采用具有相同设计目标和一致功能特性的备用实时域虚拟机来托管这些任务。此外,备用虚拟机的异构性能够使当前利用关键参数特定内存地址发起的攻击(如代码注入攻击[20]和 ROP 攻击[21])失效.

4.3 技术挑战与问题

基于上述,嵌入式虚拟化可以为 IPIC 提供安全增强的异常检测架构,优先确保工业控制的实时性和确定性,提供低延迟和闭环的异常检测安全功能。然而,要完全实现这一设计蓝图并将其应用于实际的工业控制系统,如图 8 所示当前仍需攻克以下技术挑战:

(1)非实时域中的异常检测模块需基于实时域 PLC 的现场数据进行分析决策。挑战在于如何在不损害 IPIC 实时域安全隐私前提下,将现场数据提供给非实时域。利用现有域间通信机制在非实时域与实时域之间建立共享内存,可以实现现场数据的直传。然而,这一思路存在两方面的安全问题:首先,从内存攻击的角度来看,直接建立共享内存的数据交换机制可能使高关键级别的实时域直面来自低关键级别的非实时域的内存攻击,如针对共享内存发起的缓冲区溢出攻击和拒绝服务攻击;其次,某些实时域现场数据,如敏感的传感器读数,可能涉及工业生产过程中有关制造工艺的关键机密信息,不宜提供给低关键级别的非实时域,而当前基于共享内存的数据直传机制缺乏必要的安全审查,无法有效识别和拦截交互数据中的敏感信息。

(2)如何在不增加 PLC 实时性损耗的前提下,将 IPIC 实时域的现场数据提供给非实时域的异常检测载荷。根据前节所述,现有的非侵入式异常检

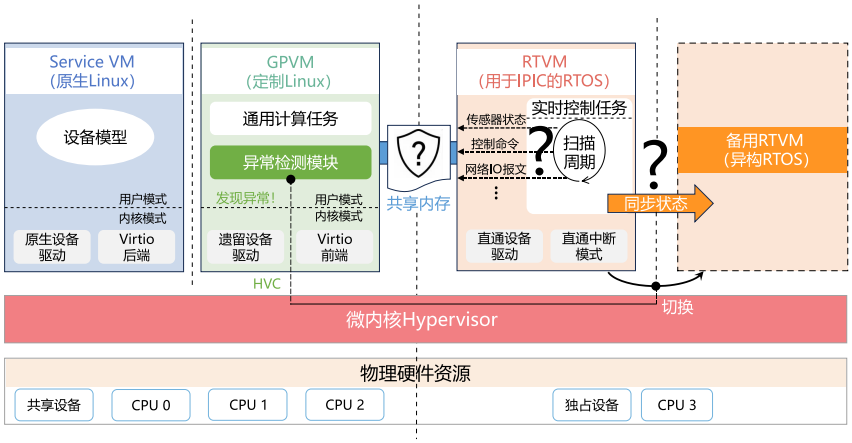


图 8　基于虚拟化隔离的 IPIC 异常检测安全增强架构面临的挑战

测方案依赖于 PLC 现场数据，这些数据包括传感器状态、控制命令、扫描周期时间以及网络数据包。其中，传感器状态和控制命令分别存储于实时域 PLC 内存的输入映像表和输出映像表中，而网络数据包则存储在为网络通信专门构建的 I/O 内存缓冲区中。在 PLC 执行工业自动化控制过程中，实时控制任务遵循扫描周期规律性地运行。若通过实时控制任务在每个扫描周期的适当时刻，将前述现场数据从实时域内存复制到共享内存，则必然需要在 PLC 常规扫描周期中增加额外的数据拷贝阶段，这将对 PLC 的实时性能造成较大损耗；而若采用另一个独立的进程任务来执行现场数据拷贝工作，为了避免损害实时性，同样必须确保该任务与实时控制任务于扫描周期内的频繁内存数据访问不产生任何冲突。

(3) 虽然虚拟化技术允许将实时域备用虚拟机的上下文直接保存在另一段隔离内存中，并通过 Hypervisor 在异常发现后迅速进行切换，但此方法与 FlyOS 系统故障发生时实现的飞行控制任务在两个虚拟机间的直接切换(无状态保留)有所区别。实时域 PLC 中某些状态信息需要在多个扫描周期内持续维护，例如，PLC 控制逻辑程序或固件配置的定时器中断会在预定时间点触发中断处理过程。因此，如何将实时域中用于 PLC 持续运行的系统连续性状态安全地同步至备用虚拟机，以确保实现真正的“无缝”切换，仍然是一个亟待解决的问题。

基于虚拟化隔离的 IPIC 安全增强实现路径不仅适用于非侵入式异常检测技术，现有 PLC 内嵌安全组件防护技术也可以将存在较大计算资源需求的子模块在非实时域灵活部署，既能降低计算(网络)延迟又能避免对实时域工业控制任务造成延时损耗，如 PLC 控制程序远程证明方法可将原本由远程主机执行的认证过程放置在非实时域中，漏洞热修复技术可在非实时域中执行数据驱动的 PLC 控制行为动态分析与自动化漏洞定位等。但正如 IPIC 异常检测安全增强所呈现的，修改 PLC 自有协议和架构来进行基于虚拟化隔离的安全融合增强实现仍是一个复杂问题，既涉及域间数据的安全交互，也涉及如何将协议架构改造引入的实时性开销最小化等诸多因素。

5　总结与展望

针对工业互联网场景下智能可编程工业控制器面临的“基础安全架构由南北向网络化隔离空间向东西向系统化隔离空间转变、感算控智高度集成的控制器 SoC 系统要求伴生的安全载荷尽可能一体化集成部署”等安全范式转变，本文全面梳理了 IPIC 当前的安全威胁与防御路线，并重点阐明了嵌入式虚拟化隔离作为 IPIC 基础安全架构的未来发展方向；随后延承嵌入式虚拟化技术脉络为 IPIC 设计了一种具备安全参数本地采集、异常检测融合决策、实时域热备份动态切换等功能的 IPIC 安全增强架构，同时探讨了该架构在实际工业系统部署中可能面临的技术挑战。

随着人工智能技术、新一代通信(SDN、TSN)等 IT 技术和 OT 自动化控制运营技术的加速融合，智能可编程工业控制器安全增强未来将呈现以下几个趋势：(1) 将感知采集、分析计算、运行控制、智能决策在 SoC 片上系统集成化部署已成为工业控制器新一轮技术和产品形态升级方向，信息安全和功能安全的实施载体和威胁传导路径在网络空间层面趋于一致，确定性和实时性保障成为安全增强方案的先决条件；(2) 对计算和网络资源的灵活配置需

求已逐渐延伸到工业控制网络场景，虚拟化技术不仅可以支撑终端侧和边缘侧新型工业控制器多功能异构载荷东西向安全隔离边界的构建，同时可以通过软件定义网络、网络功能虚拟化、容器集群等技术实现工业网络侧资源和工业云侧资源的虚拟化协同，进而实现工业云网边端信息交互和操控指令传递通路的精准管控；(3) 云网边端各层级的虚拟化部署使得传统固定工业计算场景和工业控制器网络结构向动态可调转变，可以探索利用近年来出现的若干动态安全技术(如弹性安全、零信任)提升新型工业控制器的纵深防御水平，加强开放场景下工业互联网的抗攻击容忍和自愈合恢复能力。

参 考 文 献

[1] Falliere N, Murchu L O, Chien E, et al. W32. Stuxnet dossier. Symantec Security Response, Version 1.4, 2011, 5(6): 29

[2] Dragos. Pipedream: Chernovite's emerging malware targeting industrial control systems. https://www.cyberdaily.au/commercial/7747-chernovite-s-pipedream-malware-targeting-ics, 2022, 4, 14

[3] Strom B E, Applebaum A, Miller D, et al. MITRE ATT&CK: Design and philosophy. https://attack.mitre.org/, 2018

[4] Tokarev V. CoDe16: 16 zero-day vulnerabilities affecting CODESYS framework leading to remote code execution on millions of industrial devices across industrie. https://i.blackhat.com/BH-US-23/Presentations/US-23-Tokarev-Code16-16-zero-day-vulnerabilities.pdf, 2023

[5] Bitan S, Dankner A. sOfT7: Revealing the secrets of the Siemens S7 PLCs. https://i.blackhat.com/USA-22/Wednesday/US-22-Bitan-Revealing-S7-PLCs.pdf, 2022

[6] Cheng M, Yang S. Taking apart and taking over ICS-SCADA ecosystems: A case study of mitsubishi electric. DEF CON. https://infocondb.org/con/def-con/def-con-29/taking-apart-and-taking-over-ics-scada-ecosystems-a-case-study-of-mitsubishi-electric, 2021

[7] Wylie J. Analyzing pipedream: Challenges in testing an ICS attack toolkit. DEF CON 30. https://www.cct.lsu.edu/lectures/analyzing-pipedream-challenges-testing-ics-attack-toolkit, 2022

[8] Wang Z, Zhang Y, Chen Y, et al. A survey on programmable logic controller vulnerabilities, attacks, detections, and forensics. Processes, 2023, 11(3): 918

[9] Yuan Y, Tang X, Zhou W, et al. Data driven discovery of cyber physical systems. Nature Communications, 2019, 10: 4894

[10] ISA. ISA95, Enterprise control system integration. https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95, 2024

[11] Azarmipour M, Elfaham H, Gries C, et al. PLC 4.0: A control system for industry 4.0//Proceedings of the 45th Annual Conference of the IEEE Industrial Electronics Society on IECON. Lisbon, Portugal, 2019, (1): 5513-5518

[12] Siemens. SIMATIC S7-1500 TM NPU. https://www.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1500/simatic-s7-1500-tm-npu.html, 2018

[13] Rockwell Automation. Revolutionizing predictive maintenance: Rockwell automation's FactoryTalk analytics LogixAI, powering next-gen data modeling and machine learning for operational technology. https://www.rockwellautomation.com/en-us/support/documentation/overview/factorytalk-analytics-logixai-machine-learning-and-logix.html, 2024

[14] OMRON. IPC machine controller. https://www.fa.omron.com.cn/data_pdf, 2024

[15] BECKHOFF. TwinCAT automation software. https://www.beckhoff.com.cn/zh-cn/products/automation/twincat/, 2024

[16] B&R Automation. Chip I/O and control system. https://www.br-automation.com/zh/products/plc-systems/x20-system/, 2024

[17] Govil N, Agrawal A, et al. On ladder logic bombs in industrial control systems//Proceedings of the Conference on Computer Security: ESORICS 2017 International Workshops, CyberICPS 2017 and SECPRE 2017. Oslo, Norway: Springer, 2018: 110-126

[18] Basnight Z, Butts J, Lopez Jr J, et al. Firmware modification attacks on programmable logic controllers. International Journal of Critical Infrastructure Protection, 2013, 6(2): 76-84

[19] Garcia L, Brasser F, Cintuglu M H, et al. Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit//Proceedings of the Conference on Network and Distributed System Security Symposium. San Diego, USA, 2017: 1-15

[20] Yoo H, Ahmed I. Control logic injection attacks on industrial control systems//Proceedings of the ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference (SEC 2019). Lisbon, Portugal, 2019, (34): 33-48

[21] Ayub A, Zubair N, Yoo H, et al. Gadgets of gadgets in industrial control systems: Return oriented programming attacks on PLCs//Proceedings of the Conference on 2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). San Jose, USA, 2023: 215-226

[22] Senthivel S, Dhungana S, Yoo H, et al. Denial of engineering operations attacks in industrial control systems//Proceedings of the 8th ACM Conference on Data and Application Security and Privacy. Tempe, USA, 2018: 319-329

[23] Kalle S, Ameen N, Yoo H, et al. Clik on PLCs! attacking control logic with decompilation and virtual PLC//Proceedings of the 26th Conference on Network and Distributed System Security Symposium. San Diego, USA, 2019: 1-12

[24] Jian G. Fuzzing and breaking security functions of simatic PLCs. https://i.blackhat.com/EU-22/Thursday-Briefings/EU-22-Jian-Fuzzing-and-Breaking-Security-Functions-of-SIMATIC-PLCs.pdf, 2022

[25] Cheng Lei, Li Donghong, Ma Liang. The spear to break the security wall of S7CommPlus. Blackhat USA, 2017, (17): 1-12

[26] Qasim S A, Auib A, Johnson J, et al. Attacking the IEC 61131 logic engine in programmable logic controllers//Critical Infrastructure Protection XV: 15th IFIP WG 11.10 International Conference. Virtual, 2022, (15): 73-95

[27] Biham E, Bitan S, Carmel A, et al. Rogue 7: Rogue engineering-station attacks on S7 simatic PLCs. Black Hat, USA, 2019

[28] Pickren R, Shekari T, Zonouz S, et al. Compromising industrial processes using web-based programmable logic controller malware//Proceedings of the 31st Annual Conference on Network and Distributed System Security Symposium. San Diego, USA, 2024

[29] López-Morales E, Planta U. Rubio-Medrano C, et al. SoK: Security of programmable logic controllers//Proceedings of the 33rd Conference on the USENIX Security. Philadelphia, USA, 2024: 7103-7122

[30] Zhang M, Chen C Y, Kao B C, et al. Towards automated safety vetting of PLC code in real-world plants//Proceedings of the Conference on the IEEE Symposium on Security and Privacy(SP). San Francisco, USA, 2019: 522-538

[31] Guo S, Wu M, Wang C, et al. Symbolic execution of programmable logic controller code//Proceedings of the 11th Joint Conference on Foundations of Software Engineering. Paderborn, Germany, 2017: 326-336

[32] Canet G, Couffin S, Lesage J J, et al. Towards the automatic verification of PLC programs written in instruction list//Proceedings of the Conference on Systems, Man and Cybernetics. Cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions. Nashville, USA, 2000: 2449-2454

[33] Bruttomesso R. Rockwell PLC logic analysis using model checking. https://uploads-ssl.webflow.com/645a4534705010e2cb244f50/ 64947c5d3907aade26a9a049_Nozomi-Networks-WP-Rockwell-PLC.pdf, 2022

[34] Keliris A, Maniatakos M. ICSREF: A framework for automated reverse engineering of industrial control systems binaries//Proceedings of the 26th Annual Conference on Network and Distributed System Security Symposium. San Diego, USA, 2019

[35] Tychalas D, Benkraouda H, Maniatakos M, et al. ICSFuzz: Manipulating I/Os and repurposing binary code to enable instrumented fuzzing in ICS control applications//Proceedings of the 30th Conference on USENIX Security Symposium. 2021: 2847-2862

[36] Tychalas D, Maniatakos M. IFFSET: In-field fuzzing of industrial control systems using system emulation//Proceedings of the Conference on Design, Automation & Test in Europe Conference & Exhibition (DATE). Grenoble, France, 2020: 662-665

[37] Bytes A, Rajput P H N, Doumanidis C, et al. FieldFuzz: In situ blackbox fuzzing of proprietary industrial automation runtimes via the network//Proceedings of the 26th Conference on International Symposium on Research in Attacks, Intrusions and Defenses. New York, USA, 2022: 499-512

[38] Hadziosmanovic D, Sommer R, Zambon E, et al. Through the eye of the PLC: Semantic security monitoring for industrial processes//Proceedings of the 30th Annual Computer Security Applications Conference on ACSAC 2014. New Orleans, USA, 2014: 126-135

[39] Feng C, Li T, Chana D, et al. Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks//Proceedings of the 47th Conference on Dependable Systems and Networks (DSN). Denver, USA, 2017: 261-272

[40] Caselli M, Zambon E, Kargl F, et al. Sequence-aware intrusion detection in industrial control systems//Proceedings of the 1st Conference on Cyber-Physical System Security. Singapore, 2015: 13-24

[41] Meng J, Yang Z, Zhang Z, et al. SePanner: Analyzing semantics of controller variables in industrial control systems based on network traffic//Proceedings of the 39th Conference on Annual Computer Security Applications Conference. New York, USA, 2023: 310-323

[42] Yang Z, He L, Cheng P, et al. PLC-sleuth: Detecting and localizing PLC intrusions using control invariants//Proceedings of the Conference on International Symposium on Recent Advances in Intrusion Detection. Boston, USA, 2020, (9): 333-348

[43] Yang Z, He L, Yu H, et al. Detecting PLC intrusions using control invariant. IEEE Internet of Things Journal, 2022, 9(12): 9934-9947

[44] Ahmed C M, Ochoa M, Zhou J, et al. Scanning the cycle: Timing-based authentication on PLCs//Proceedings of the Conference on 2021 ACM Asia Computer and Communications Security. Hong Kong, China, Virtual, 2021: 886-900

[45] Formby D, Beyah R. Temporal execution behavior for host anomaly detection in programmable logic controllers. IEEE Transactions on Information Forensics and Security, 2020, 15: 1455-1469

[46] Yang H, Cheng L, Chuah M C, et al. Detecting payload attacks on programmable logic controllers//Proceedings of the Conference on 2018 IEEE Conference Communications and Network Security. Beijing, China, 2018: 1-9

[47] Liu J, Lin X, Chen X, et al. ShadowPLCs: A novel scheme for remote detection of industrial process control attacks. IEEE Transactions on Dependable and Secure Computing, 2020, 19: 2054-2069

[48] Ike M, Phan K, Sadoski K, et al. SCAPHY: Detecting modern ICS attacks by correlating behaviors in SCADA and PHYsical//Proceedings of the Conference on 2023 IEEE Symposium on Security and Privacy. San Francisco, USA, 2022: 20-37

[49] Aoudi W, Iturbe M, Almgren M, Truth will out: Departure-based process-level detection of stealthy attacks on control systems//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. Toronto, Canada, 2018: 817-831

[50] Salehi M, Bayat-Sarmadi S. PLCDefender: Improving remote attestation techniques for PLCs using physical model. IEEE Internet of Things Journal, 2021, 8: 7372-7379

[51] Abbasi A, Holz T, Zambon E, et al. ECFI: Asynchronous control flow integrity for programmable logic controllers// Proceedings of the 33rd Conference on Annual Computer Security Applications Conference. Orlando, USA, 2017: 437-448

[52] Yang Z, Bao Z, Jin C, et al. PLCrypto: A symmetric cryptographic library for programmable logic controllers. IACR Transactions on Symmetric Cryptology, 2021, (3): 170-217

[53] Rajput P H N, Doumanidis C, Maniatakos M, et al. ICSPatch: Automated vulnerability localization and non-intrusive hot-patching in industrial control systems using data dependence// Proceedings of the 32nd Conference on USENIX Security Symposium. Anaheim, USA, 2023: 6861-6876

[54] Zhou M, Wang H, Li K, et al. Save the bruised striver: A reliable live patching framework for protecting real-world PLCs //Proceedings of the 19th Conference on Computer Systems on Association for Computing Machinery. Athens, Greece, 2024: 1192-1207

[55] Rais M H, Awad R A, Lopez Jr J, et al. Memory forensic analysis of a programmable logic controller in industrial control systems. Digital Investigation, 2022, 40: 301339

[56] Ahmed I, Obermeier S, Sudhakaran S, et al. Programmable logic controller forensics. IEEE Security & Privacy, 2017, 15: 18-24

[57] López-Morales E, Rubio-Medrano C, Doupe A, et al. HoneyPLC: A next-generation honeypot for industrial control systems//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. Virtual, USA, 2020: 279-291

[58] You J, Lv S, Zhao L, et al. A scalable high-interaction physical honeypot framework for programmable logic controller //Proceedings of the Conference on 2020 IEEE the 92nd Vehicular Technology Conference. Victoria, Canada, 2020: 1-5

[59] Burns A, Davis R I. A survey of research into mixed criticality systems. ACM Computing Surveys(CSUR), 2017, 50: 1-37

[60] Esper A, Nelissen G, Nelis V, et al. An industrial view on the common academic understanding of mixed-criticality systems. Real-Time Systems, 2018, 54(3): 745-795

[61] Simó J, Balbastre P, Blanes J F, et al. The role of mixed criticality technology in industry 4.0. Electronics, 2021, 10(3): 226

[62] Cinque M, Cotroneo D, De Simone L, et al. Virtualizing mixed-criticality systems: A survey on industrial trends and issues. Future Generation Computer Systems, 2022, 129: 315-330

[63] Martins J, Pinto S. Bao: A lightweight static partitioning hypervisor for modern multi-core embedded systems// Proceedings of the 1st Conference on NG-RES@HiPEAC. Bologna, Italy, 2020, (3): 1-14

[64] Martins J, Pinto S. Shedding light on static partitioning hypervisors for arm-based mixed-criticality systems//Proceedings of the 2023 IEEE 29th Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS). San Antonio, USA, 2023: 40-53

[65] Queiroz R, Cruz T, Simoes P, et al. Testing the limits of general-purpose hypervisors for real-time control systems. Microprocessors and Microsystems, 2023, 99: 104848

[66] Schade F, Barton D, Fleischer J, et al. Evaluation of a hypervisor-based smart controller for industry 4.0 functions in manufacturing//Proceedings of the 2021 IEEE 7th Conference on World Forum on Internet of Things(WF-IoT). New Orleans, USA, 2021: 680-685

[67] Sinsegye. SP7000s series ICP type machine. https://www.sinsegye.com.cn/Products/info.aspx?itemid=532&lcid=25#menuny, 2024

[68] Siemens. Be open and independent. https://assets.new.siemens.com/siemens/assets/api/uuid:f088231a33215e94cb4a9a0f39cd56dadfa765b0/dffa-b10006-03-7600-ws-simatic-s7-1500-software-controller-en.pdf, 2017

[69] B&R Automation. Two operating systems on one device. https://www.br-automation.com/zh/about-us/customer-magazine/2018/20189/two-operating-systems-on-one-device/, 2018

[70] Intel. A quick guide to building a new generation of smart industrial controllers. https://www.intel.cn/content/www/cn/zh/internet-of-things/next-gen-industrial-controllers-guide.html, 2023

[71] Shenzhen Huichuan Technology Co. Virtualization Systems, Work Methods, Work Devices and Readable Storage Media, China: CN202311727371.5, 2023(in Chinese)
(深圳市汇川技术股份有限公司. 虚拟化系统、工作方法、工作设备及可读存储介质, 中国: CN202311727371.5, 2023)

[72] Kyland. Intewell-h edge operating system. https://www.kyland.com.cn/products/IntewellH.html, 2020

[73] FSM Labs. Getting started with RTLinux. http://cs.uccs.edu/~cchow/pub/rtl/doc/html/GettingStarted/, 1995

[74] Sarolahti P. The real-time application interface. Research Seminar on Real-Time Linux and Java. Citeseer, 2001

[75] Xenomai. Xenomai 3 Real-time Linux. https://v3.xenomai.org/, 2015

[76] Yagcmour K. Adaptive domain environment for operating systems. Opersys Inc, 2001

[77] Koh J H, Choi B W. Real-time performance of real-time mechanisms for RTAI and Xenomai in various running conditions. International Journal of Control and Automation, 2013, 6(1): 235-246

[78] Patel A, Daftedar M, Shalan M, et al. Embedded hypervisor Xvisor: A comparative analysis//Proceedings of the 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. Turku, Finland, 2015: 682-691

[79] De Bock Y, Mercelis S, Broeckhove J, et al. Real-time virtualization with Xvisor. Internet of Things, 2020, 11: 100238

[80] Hitepaper. The Architecture of VMware ESXi. White Paper, 2008

[81] Frank C. Arm virtualization using VMware ESXi hypervisor for embedded devices. KI-Verirauenswurdig, Erklarbar, Fair, 2021, (45): 1-73

[82] Kivity A, Kamay Y, Laor D, et al. KVM: The Linux virtual machine monitor. Linux Symposium, 2007: 225-230

[83] Zhang J, Chen K, Zuo B, et al. Performance analysis towards a KVM-based embedded real-time virtualization architecture//Proceedings of the 5th International Conference on Computer Sciences and Convergence Information Technology. Seoul, Republic of Korea, 2010: 421-426

[84] Sysgo. PikeOS: Certifiable RTOS with Hypervisor. https://www.sysgo.com/fileadmin/user_upload/data/flyers_brochures/SYSGO_PikeOS_Product_Note.pdf, 2024

[85] BlackBerry. QNX hypervisor virtualization software. https://blackberry.qnx.com/en/products/foundation-software/qnx-hypervisor, 2024

[86] Lauterbach. Hypervisor awareness manual wind river hypervisor. https://www2.lauterbach.com/pdf/hv_windriver.pdf, 2024

[87] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization//Proceedings of the Conference on ACM SIGOPS Operating Systems Review. New York, USA, 2003, 37(5): 164-177

[88] Xi S, Wilson J, Lu C, et al. RT-Xen: Towards real-time hypervisor scheduling in Xen//Proceedings of the 9th ACM International Conference on Embedded Software(EMSOFT'11). Taipei, China, 2011: 39-48

[89] Steinberg U. The NoVA microhypervisor. https://hypervisor.org/nova-fosdem-2013.pdf, 2013

[90] Murray D G, Milos G, Hand S, et al. Improving Xen security through disaggregation//Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. Seattle, USA, 2008: 151-160

[91] Heiser G, Leslie B. The OKl4 microvisor: Convergence point of microkernels and hypervisors//Proceedings of the 1st ACM SIGCOMM Asia-Pacific Conference on Systems. ApSys New Delhi, India, 2010: 19-24

[92] Li H, Xu X, Ren J, et al. ACRN: A big little hypervisor for IoT development//Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. Providence, USA, 2019: 31-44

[93] Sinitsyn V. Jailhouse. Linux Journal, 2015, (252): 2

[94] Kuz I, Liu Y, Gorton I, et al. CAmkES: A component model for secure microkernel-based embedded systems. Journal of Systems and Software, 2007, 80(5): 687-699

[95] Sangorrin D, Honda S, Takada H. Dual operating system architecture for real-time embedded systems. http://www.artist-embedded.org/docs/Events/2010/OSPERT/slides/1-1_NU_OSPERT2010.pdf, 2010, (6): 1-24

[96] Pinto S, Pereira J, Gomes T, et al. LTZVisor: TrustZone is the key//Proceedings of the 29th Euromicro Conference on Real-Time Systems. Dubrovnik, Croatia, 2017, (4): 1-22

[97] Pinto S, Santos N. Demystifying arm TrustZone: A comprehensive survey. ACM Computing Surveys, 2019, 51(6): 1-36

[98] Pinto S, Oliveira A, Pereira J, et al. Lightweight multicore virtualization architecture exploiting arm TrustZone//Proceedings of the 43rd Annual Conference on the IEEE Industrial Electronics Society. Beijing, China, 2017: 3562-3567

[99] Pinto S, Pereira J, Gomes T, et al. Towards a TrustZone-assisted hypervisor for real-time embedded systems. IEEE Computer Architecture Letters, 2016, 16(2): 158-161

[100] Martins J, Alves J, Cabral J, et al. μRTZVisor: A secure and safe real-time hypervisor. Electronics, 2017, 6(4): 93

[101] Lucas P, Chappuis K, Paolino M, et al. VOSYSmonitor, a low latency monitor layer for mixed-criticality systems on ARMv8-A//Proceedings of the 29th Euromicro Conference on Real-Time Systems. Dubrovnik, Croatia, 2017

[102] Dong P, Burns A, Jiang Z, et al. TZDKS: A new TrustZone-based dual-criticality system with balanced performance//Proceedings of the 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications. Hakodate, Japan, 2018: 59-64

[103] Pinto S, Araujo H, Oliveira D, et al. Virtualization on TrustZone-enabled microcontrollers? Voilà!//Proceedings of the Conference on 2019 IEEE Real-Time and Embedded Technology and Applications Symposium. Montreal, Canada, 2019: 293-304

[104] Wang J, Li A, Li H, et al. RT-TEE: Real-time system availability for cyber-physical systems using arm TrustZone//Proceedings of the Conference on 2022 IEEE Symposium on Security and Privacy (SP). San Francisco, USA, 2022: 352-369

[105] Zhang Y, Hao Z, Hu N, et al. A virtualization-based security architecture for industrial control systems//Proceedings of the 2022 7th IEEE International Conference on Data Science in Cyberspace(DSC). 2022: 94-101

[106] Docker. Use containers to build, share and run your applications. https://www.docker.com/what-docker, 2020

[107] LinuxContainers. Container and virtualization tools. https://linuxcontainers.org/, 2019

[108] Goldschmidt T, Hauck-Stattelmann S, Malakuti S, et al. Container-based architecture for flexible industrial control applications. Journal of Systems Architecture: Embedded Software Design, 2018, 84: 28-36

[109] Openeuler. iSulad. https://www.openeuler.org/zh/other/projects/isula/,2020

[110] Openeuler. Openeuler embedded. https://docs.openeuler.org/zh/docs/24.03_LTS/docs/Embedded/Embedded.html, 2024

[111] Balena. Balena engine. https://www.balena.io/, 2022

[112] Cinque M, Della Corte R, Eliso A, et al. RT-CASEs: container-based virtualization for temporally separated mixed-criticality task set//Proceedings of the 31st Euromicro Conference on Real-Time Systems Stuttgart. Germany, 2019, (5): 1-22

[113] Abeni L, Balsini A, Cucinotta T. Container-based real-time scheduling in the Linux kernel. ACM SIGBED Review, 2019, 16(3): 33-38

[114] Plcnext Marcel. Docker, balena, and containerization in industrial automation. https://www.plcnext-community.net/news/docker-balena-and-co-containerization-in-industrial-automation/, 2021

[115] Sinha S, West R. Towards an integrated vehicle management system in DriveOS. ACM Transactions on Embedded Computing Systems (TECS), 2021, 20(5s): 1-24

[116] Farrukh A, West R. FlyOS: Rethinking integrated modular avionics for autonomous multicopters. Real-Time Systems, 2023, 59(2): 256-301

[117] Cilardo A, Cinque M, De Simone L, et al. FlyOS: Rethinking integrated modular avionics for autonomous multicopters. Computer, 2022, 55(10): 35-47

[118] Luo J, Kang M, Bisse E, et al. A quad-redundant PLC architecture for cyber-resilient industrial control systems. IEEE Embedded Systems Letters, 2020, 13(4): 218-221

[119] Liu K, Wang J Y, Wei Q, et al. HRPDF: A software-based heterogeneous redundant proactive defense framework for programmable logic controller. Journal of Computer Science and Technology, 2021, 36: 1307-1324

WANG Ya-Zhe, Ph. D., associate professor. His research interests include network and system security, application and data security.

REN Lei, Ph. D., professor. His research interests include Industrial Internet and industrial artificial intelligence.

FENG Deng-Guo, Ph. D., professor, Academician of the Chinese Academy of Sciences. His research interests include information security and cryptology.

WANG Wen-Jie, Ph. D., assistant professor. His research interests include network and system security, particularly in embedded system security, Industrial Internet, and applied cryptography.

ZHANG Yi-Fan, Ph. D. candidate. His research interests include neural networks and deep learning, time-series analysis, and Industrial Internet.

WANG Fei, Ph. D., assistant professor. Her research interests include Industrial Internet of Things security, industrial intelligence, and intelligent optimization.

KONG Yu-Sheng, Ph. D. candidate, assistant engineer. His current research interests include physical industrial artificial intelligence and IIoT cyber security.

LÜ Jin-Hu, Ph. D., professor. His research interests include nonlinear circuits and systems, complex networks, multiagent systems, and big data.

Background

The advent of Industry 4.0 and intelligent manufacturing concepts has revolutionized the industrial field, emphasizing the need for automation, networking, and intelligence within the Industrial Internet. Traditional Programmable Logic Controllers (PLCs), once the cornerstone of industrial automation, are evolving to meet these new technical demands. The emergence of Intelligent Programmable Industrial Controllers (IPICs) exemplifies this progression, utilizing embedded virtualization technologies and multi-core SoC architectures to integrate functions such as perception acquisition, data computation, real-time control, intelligent decision-making, and network communication. However, the convergence of diverse functional loads potentially reshapes security boundaries and attack surfaces, thereby exposing IPICs to a broader spectrum of threats than those encountered by traditional PLCs. Therefore, it is imperative for IPICs to establish an intrinsic secure system architecture and to implement advanced security enhancement measures for robust protection.

This paper outlines the emerging trends and prospects for IPIC security enhancement. Initially, the authors analyze the security threats confronting PLCs and systematically summarize the current security defense mechanisms, categorized by the pre-deployment and operational stages of PLC lifecycles. They then introduce a range of research works on embedded security isolation endowed with real-time features, shedding light on how to construct a secure system architecture paradigm for IPICs. These research works are categorized based on technological attributes into dual-kernel isolation techniques and a spectrum of embedded virtualization technologies, including monolithic kernel Hypervisor, microkernel Hypervisor, static partitioning Hypervisor, virtualization utilizing trusted execution environments, and lightweight container-based virtualization. Furthermore, they propose an anomaly detection and response architecture for security enhancement in IPIC, based on virtualization isolation, and discuss the technical challenges anticipated in its practical deployment. Finally, they offer a forward-looking perspective on the future trends in IPIC security enhancement.

This work was supported by the National Key R&D Program of China (2023YFB3308000), the National Science Foundation for Distinguished Young Scholars of China (62225302), and the National Natural Science Foundation of China (92167108, 62173023).