开题报告

基于覆盖率反馈的PLC安全性分析方法研究

陈张萌

2024年6月1日

目录

研究背景和相关工作

工业控制系统与PLC

工业控制系统（Industrial Control Systems, ICS）在与民生密切相关的基础行业中广泛部署，如能源、交通、水利、电力、石油等。随着互联网的快速发展，越来越多的工业控制系统通过通用协议连接到互联网。然而，这些系统在设计之初并未考虑连接到互联网的需求，因此其采用的协议缺乏加密、授权、认证等机制，存在许多漏洞。

PLC的具体结构

PLC（Programmable Logic Controller）是工业控制系统的核心，其硬件由CPU、电源模块、I/O模块、内存和扩展模块组成，结构与传统计算机基本相同。
循环扫描是PLC执行其控制程序的基本工作方式。在这种模式下，PLC按照以下步骤重复执行一系列操作：

- 输入扫描：PLC读取所有输入端口的状态，这些输入可能来自传感器、按钮或其他设备。
- 程序执行：PLC根据输入值和预设的逻辑来执行存储在内存中的程序。这包括执行所有的逻辑运算、定时器、计数器和其他控制指令。
- 输出更新：程序执行完成后，PLC更新输出端口的状态，这些输出控制着执行器、指示灯或其他设备。
- 循环重复：PLC返回第一步，再次读取输入，形成一个连续的循环。

PLC编程语言

PLC编程语言是专门为编写控制逻辑而设计的，以便管理和监控工业自动化过程。根据国际电工委员会制定的工业控制编程语言标准（IEC1131-3），PLC有五种标准编程语言：

- **梯形图语言（Ladder Diagram, LD）**：这是**最常用的PLC编程语言**，它使用类似于电路图的图形符号，通过视觉化的方式来表示控制逻辑。
- 指令列表语言（Instruction List, IL）：这种语言使用一系列的指令来编程PLC，类似于汇编语言。
- 功能块图语言（Function Block Diagram, FBD）：这种语言使用功能块来表示操作，并通过图形化的方式将它们连接起来，以创建控制逻辑。
- 顺序功能流程图语言（Sequential Function Chart, SFC）：也称为SFC图，它用于描述顺序控制过程，通过一系列的步骤和转换来表示过程的流程。
- PLC ST（Structured Text）：以类似于Pascal或C语言的文本格式出现，是一种结构化的编程语言，非常适合复杂的算法和数据处理任务。

PLC安全漏洞

PLC本质上是一台简单的计算机，执行主机计算机的指令，处理用户程序，产生输出以控制现场设备的操作，并将现场设备的数据传输给主机计算机。在此执行过程中，存在用户代码漏洞、固件篡改和网络攻击等安全问题。在绘制梯形图的过程中，由于人员的疏忽可能会引入一些潜在的安全漏洞。

- **常见的代码安全问题**：包括逻辑错误、定时器条件竞争、无条件转移、代码无法到达、无限循环、语法错误、作用域和链接错误、隐藏跳转器、重复对象和未使用对象等问题。
- **代码安全问题的后果**：例如定时器振荡、某些分支无法访问、循环超时错误、易于受到攻击和改变过程、恶意程序嵌入、程序执行过程改变、输出失败、攻击者利用等。

PLC安全分析方法

1. 静态分析方法

代码安全的形式化分析方法 该方法将PLC梯形图或指令列表语言等转换成中间语言，以便于模型构建。然后考虑环境和其他因素构建模型，并通过符号执行来解决状态空间爆炸问题，并最终检查各种代码缺陷。

包括中间语言转换、模型检测、符号执行等步骤。

静态分析的局限性：使用符号执行虽然可以解决状态空间爆炸问题，但会消耗大量时间和内存，因此对于大型程序的检测时间较长。此外，由于它使用静态分析方法，无法检测分配给PLC的虚假传感器值。

PLC安全分析方法

2. 动态分析方法

可运用Fuzzing等测试手段。Fuzzing是一种自动化的软件测试技术，通过自动生成大量异常、随机或边界测试数据并输入到程序中，观察程序是否能够妥善处理这些输入，以此来发现潜在的错误或安全漏洞。

PLC安全分析方法

2. **动态分析方法**

对梯形图的安全分析方法，例如：**Sizzler**

1. **梯形图转换**：Sizzler首先将PLC的梯形图（Ladder Diagrams, LD）转换成ANSI C代码，以便在不同的微控制器单元（MCUs）上执行和测试。
2. **仿真环境搭建**：利用QEMU和Avatar2等工具，Sizzler创建了一个仿真环境，模拟PLC的固件行为，包括GPIO和I2C接口等硬件抽象层。
3. **变异策略**：Sizzler采用了基于SeqGAN（序列生成对抗网络）的变异策略，通过一系列操作符（如位翻转、字节翻转、算术运算和值替换）来修改输入，以触发不同的代码路径，生成有效的测试用例
4. **局限性**：对特定厂商PLC固件的依赖和仿真过程中的性能开销

PLC安全分析方法

2. 动态分析方法

对二进制文件的安全分析方法，例如：**ICSFuzz**

利用KBUS子系统向控制应用程序提供输入，并通过物理PLC进行测试，这限制了其可扩展性。

该工作研究的对象时Runtime。

研究目标

本研究旨在通过Fuzzing技术对PLC（Programmable Logic Controller）的安全性进行分析和检查，尝试进一步挖掘Fuzzing在PLC应用程序安全性分析方面的能力。

相关工作

OpenPLC OpenPLC是一个开源的软件项目，旨在提供一个成本低廉且实用的替代方案，以取代传统的硬件可编程逻辑控制器（PLC）。

- 开源性质：作为开源软件，OpenPLC提供了自由访问的源代码，鼓励社区参与和贡献。
- 兼容性：它支持多种硬件平台，不局限于特定供应商的硬件，提供了更大的灵活性。
- 符合国际标准：OpenPLC遵循国际电工委员会（IEC）61131-3标准，支持多种编程语言，如梯形图（LD）、功能块图（FBD）、顺序功能图（SFC）、指令列表（IL）和结构化文本（ST）。
- 应用广泛：它已被应用于多种场景，包括家庭自动化、交通控制、水处理、现场网络以及暖通空调（HVAC）系统的控制等。

相关工作

LibAFL：一个轻量级的AFL（American Fuzzy Lop）库，用于Fuzzing。

- LibAFL是一个新颖且完全可扩展的模糊测试框架，它允许研究人员和开发人员构建自定义的模糊测试工具。LibAFL的设计基于模块化原则，提供了高度的灵活性，允许用户轻松地扩展核心模糊测试流程并共享新组件。该框架用Rust语言编写，利用了Rust的性能和安全性特性。
- LibAFL的核心库提供了基础的模糊测试构件，包括输入处理、语料库管理、调度器、变异器、执行器和观察器等。它还包括了与不同执行引擎和仪器化后端的集成，如LLVM、QEMU用户模式和Frida。

整体设计

step1. 将PLC梯形图编译为C代码，以便进行Fuzzing
step2. 使用QEMU进行模拟，减少对真实硬件的依赖
step3. 探索处理外设输入的方法，包括转发真实外设请求或模拟外设输入

具体实现

step1. 从梯形图到C代码：使用OpenPLC

使用OpenPLC可以将梯形图代码翻译成C代码，（以及可以编译为可以运行的二进制文件）这样就可以使用afl工具来做fuzzing。

具体实现

step2. 使用QEMU对硬件进行模拟

非通用架构软件fuzzing时的几种常用解决方案及其对比：

1. On-device Analysis：这种方法需要真实设备来进行测试。它得到了最真实可信的结果，但可扩展性较差，且缺乏可见性。因为在裸机上收集执行信息较为困难。
2. Full Emulation：为了克服真实硬件上的性能和可扩展性问题，研究人员提出了使用如QEMU这样的完整仿真器来模拟固件执行。主要面临的挑战是对外设的方针难以模拟。
3. Peripheral Forwarding：将外设访问转发到真实设备，并将固件在仿真器内运行。由于依赖真实硬件，性能和可扩展性问题仍未解决。
4. Semi-rehosting：固件的主要逻辑仍然在仿真器内执行，外设相关的处理被识别出来，在主机上运行。

具体实现

step3. 探索处理外设输入的方法，包括转发真实外设请求或模拟外设输入

在现有的研究中：采用了外设映射的方法。当遇到fuzzing框架没有处理过的外设时，就会crash，影响对程序结构的进一步探究

考虑采用模拟外设输入的方式完成

预期成果

- 开发一个基于覆盖率反馈的PLC Fuzzing系统
- 发表一篇高水平会议论文
- 提供PLC安全漏洞样本

研究计划

1. **2024.8 系统设计**：设计Fuzzing系统架构，包括代码编译、模拟环境设置和输入处理
2. **2024.11 系统实现**：开发Fuzzing系统，集成QEMU模拟和外设输入处理
3. **2024.12 测试与评估**：对系统进行测试，评估其发现安全漏洞的能力
4. **2025.3 论文撰写**：撰写毕业论文

参考文献

1. A Systemic Review of Kernel Fuzzing
2. Armor PLC: A Platform for Cyber Security Threats Assessments for PLCs
3. From Library Portability to Para-rehosting: Natively Executing Microcontroller Software on Commodity Hardware
4. Fuzzing of Embedded Systems: A Survey
5. Fuzzing the Internet of Things: A Review on the Techniques and Challenges for Efficient Vulnerability Discovery in Embedded Systems
6. Fuzzing: a survey
7. Fuzzware: Using Precise MMIO Modeling for Effective Firmware Fuzzing
8. Hydra: Finding Bugs in File Systems with an Extensible Fuzzing Framework
9. ICS3Fuzzer: A Framework for Discovering Protocol Implementation Bugs in ICS Supervisory Software by Fuzzing
10. ICSFuzz: Manipulating IOs and Repurposing Binary Code to Enable Instrumented Fuzzing in ICS Control Applications

参考文献

11. IEC 61850 Compatible OpenPLC for Cyber Attack Case Studies on Smart Substation Systems
12. Investigating the Security of OpenPLC: Vulnerabilities, Attacks, and Mitigation Solutions
13. kAFL
14. KernelGPT: Enhanced Kernel Fuzzing via Large Language Models
15. LibAFL QEMU: A Library for Fuzzing-oriented Emulation
16. LibAFL: A Framework to Build Modular and Reusable Fuzzers
17. OpenPLC: An IEC 61,131–3 compliant open source industrial controller for cyber security research
18. OpenPLC: An Open Source Alternative to Automation
19. Review of PLC Security Issues in Industrial Control System
20. Security Challenges in Industry 4.0 PLC Systems

参考文献

21. SHiFT: Semi-hosted Fuzz Testing for Embedded Applications
22. Sizzler: Sequential Fuzzing in Ladder Diagrams for Vulnerability Detection and Discovery in Programmable Logic Controllers
23. SoK: Enabling Security Analyses of Embedded Systems via Rehosting
24. Tardis: Coverage-Guided Embedded Operating System Fuzzing
25. What You Corrupt Is Not What You Crash: Challenges in Fuzzing Embedded Devices
26. μAFL: Non-intrusive Feedback-driven Fuzzing for Microcontroller Firmware