



计算机科学

COMPUTER SCIENCE

基于固件修复的工业网关仿真与测试方法

卫子涵, 麻荣宽, 李贝贝, 杨亚辉, 李卓, 宋云凯

引用本文

卫子涵, 麻荣宽, 李贝贝, 杨亚辉, 李卓, 宋云凯. 基于固件修复的工业网关仿真与测试方法[J]. 计算机科学, 2025, 52(12): 411-418.

WEI Zihan, MA Rongkuan, LI Beibei, YANG Yahui, LI Zhuo, SONG Yunkai. [Firmware Recovery Based Emulation and Testing Method for Industrial Gateway](#) [J]. Computer Science, 2025, 52(12): 411-418.

相似文章推荐 (请使用火狐或 IE 浏览器查看文章)

Similar articles recommended (Please use Firefox or IE to view the article)

[网络协议模糊测试技术研究进展](#)

Survey on Fuzz Testing Techniques for Network Protocols

计算机科学, 2025, 52(11A): 241100173-9. <https://doi.org/10.11896/jsjcx.241100173>

[结合动态分析的内存安全漏洞模糊测试方法](#)

Dynamic Analysis Based Fuzz Testing for Memory Safety Vulnerabilities

计算机科学, 2025, 52(11): 382-389. <https://doi.org/10.11896/jsjcx.241000003>

[基于神经元覆盖指标的测试用例生成优化研究](#)

Research on Optimization of Test Case Generation Based on Neuron Coverage Index

计算机科学, 2025, 52(11): 339-348. <https://doi.org/10.11896/jsjcx.240900006>

[AFL-VTest: 航天嵌入式软件模糊测试框架](#)

AFL-VTest: Fuzzing Framework for Aerospace Embedded Software

计算机科学, 2025, 52(12): 9-17. <https://doi.org/10.11896/jsjcx.250400144>

[利用精确中间污点源和危险函数定位加速固件漏洞挖掘](#)

Accelerating Firmware Vulnerability Discovery Through Precise Localization of Intermediate Taint Sources and Dangerous Functions

计算机科学, 2025, 52(7): 379-387. <https://doi.org/10.11896/jsjcx.240800052>

基于固件修复的工业网关仿真与测试方法

卫子涵¹ 麻荣宽¹ 李贝贝² 杨亚辉¹ 李卓¹ 宋云凯¹

1 信息工程大学网络空间安全学院 郑州 450001

2 四川大学网络空间安全学院 成都 610207

(q630834743@163.com)

摘要 随着智能制造产业的不断发展,以工业网关为代表的边缘计算设备被广泛应用于工业现场中。因此,工业网关中的程序漏洞也开始威胁工业网络的安全。然而,工业网关功能设计的专用性以及固件提取的低保真度问题,会导致现有安全测试方法难以适用于工业网关的测试。针对以上问题,提出了一种基于固件修复的工业网关仿真与测试方法。首先,在固件文件系统提取的基础上,采用启发式修复方法对文件系统中的重复和错误文件进行资源释放和修复,为测试提供了满足仿真运行需要的文件系统条件;其次,通过启发式干预方法对二进制程序运行时发生的错误进行修复,实现被测程序在仿真环境中运行;最后,设计实现了针对网关固件的模糊测试工具。在评估实验中,通过以上方法对4款真实的工业网关进行了固件文件系统修复,并对其中2款工业网关设备中的重要应用程序进行了仿真运行和模糊测试。实验结果表明,经过修复后的文件系统无序程度平均降低27.2%,且针对工业网关主要服务程序仿真运行效果良好,并在模糊测试中发现了真实工业网关设备中的1个未公开拒绝服务漏洞,证明了该工具的有效性。

关键词: 固件仿真; 工业网关; 物联网安全; 工控安全; 模糊测试

中图分类号 TP393

Firmware Recovery Based Emulation and Testing Method for Industrial Gateway

WEI Zihan¹, MA Rongkuan¹, LI Beibei², YANG Yahui¹, LI Zhuo¹ and SONG Yunkai¹

1 School of Cyberspace Security, Information Engineering University, Zhengzhou 450001, China

2 School of Cyberspace Security, Sichuan University, Chengdu 610207, China

Abstract With the continuous development of intelligent manufacturing industry, edge computing devices represented by industrial gateways are widely used in industrial sites. At the same time, software vulnerabilities of industrial gateways are beginning to affect the security of industrial networks. However, due to the specialized implementation of industrial gateway and the low-fidelity of firmware extraction, existing methods could not meet the security testing requirements. To address these issues, a firmware recovery based emulation and testing method for industrial gateway is proposed. Firstly, based on the extraction of the firmware filesystem, a heuristic recovery method is employed to free up and repair duplicate and erroneous system files, which provides a file access basis for emulation. Secondly, a heuristic emulation intervention method is adopted to mitigate errors occurring during emulation, which implements test-orientated emulation. Finally, a fuzzer is designed for industrial gateways that can be emulated. In evaluation part, firmware filesystem recovery is performed on four real industrial gateways. The emulations and fuzzing tests are conducted on important applications in two industrial gateways. The evaluation results reveal an average reduction of 27.2% in the degree of chaos for the recovered filesystem, and show a good result for emulation. Moreover, an undisclosed denial of service vulnerability in real industrial gateway devices is discovered during the fuzzing tests, which proves the effectiveness of the work.

Keywords Firmware emulation, Industrial gateway, Internet of Things security, Industrial control system security, Fuzzing test

1 引言

随着数字化转型的加速,物联网(Internet of Things, IoT)技术正以极快的速度融入传统的工业控制系统(Industrial Control System, ICS)中,现代IoT设备的高性能极大地增强了工业现场的数据收集和分析的能力,提供了远程监控、

预测性维护和实时数据分析等功能,促进了工业现场的智能化发展。作为一种工业物联网(Industrial IoT, IIoT)设备,工业网关是一类在工业自动化和控制系统中普遍使用的设备,它主要负责连接不同的工业设备和网络,实现数据的采集和通信。工业网关具有小型化的特点,可以部署在数据产生源(即工业现场设备等),对工业现场产生的庞大数据进行采集、

并利用这些数据进行边缘计算与数据统计。然而,工业网关的安全问题也被引入工业网络中。一旦工业网关遭到攻击者利用,就极有可能造成工业关键基础设施崩溃,甚至对国家安全构成严重威胁^[1-2]。

目前针对 IoT 设备的脆弱性发现基本可以分为静态方法与动态方法^[3]。静态脆弱性发现方法包括静态污点分析^[4]、静态符号执行^[5]等程序分析方法,还包括二进制相似性^[6]比对等方法。这类方法的优势在于,不需要执行 IoT 设备固件中的二进制文件,只需要对二进制代码进行静态分析即可发现潜在的脆弱性,因此适用于早期漏洞筛查以及大规模固件扫描。但是,静态分析在处理复杂的控制流和数据流依赖时,面临误报率高和路径爆炸的困境^[7]。相比之下,动态脆弱性发现方法通过在实际运行环境中分析程序行为来发现漏洞,包含动态符号执行^[8]、模糊测试^[9]等技术。这些方法通过观察设备在运行时的实际响应和状态变化,能够更有效地捕获利用链中的复杂行为和与特定输入相关的漏洞,特别是在涉及多路径条件和环境依赖的情况下。然而,工业网关与一般的 IoT 设备存在显著差异,主要体现在以下两个方面:1)文件系统提取保真度低;2)通常集成工业领域特有的工业协议和私有实现方法。这些特性使得工业网关在安全性评估和脆弱性分析上比一般 IoT 设备更加复杂和具有挑战性,而现有的 IoT 安全分析工具没有对以上两种问题做出响应,因此现有 IoT 安全分析工具难以应用于工业网关的脆弱性发现。

针对以上问题,本文提出了一种针对工业网关的动态脆弱性分析测试框架。该框架首先采用启发式方法对固件文件系统进行修复,通过添加间接层的方式使得工业网关固件能够适用于仿真测试工具;然后,提出了修复文件系统的方案,有效提升了仿真工具对工业网关的仿真成功率。最后,实现了模糊测试工具,能够对仿真运行的工业网关固件中主要的二进制服务程序进行测试。综上,本文的贡献包括以下 3 个方面:

1)提出了一种提升工业网关固件提取保真度的解决方案,对文件系统中的重复和错误文件进行资源释放和修复,解决了部分文件损坏导致的仿真失败问题。

2)针对工业网关特有的仿真障碍点,提出了启发式干预方案。对工业网关固件进行尝试仿真,学习导致仿真失败的障碍点并生成知识库,根据知识库内容判断使用障碍点干预措施。

3)设计实现了针对工业网关的固件仿真和测试工具 PerditioImage,并使用该工具对真实工业网关设备进行了仿真和测试。实验评估中,成功对两款工业网关中运行的关键服务进行仿真,并发现了一个未公开漏洞。

2 背景知识

2.1 IoT 设备固件

现代 IoT 设备通常将 FLASH 芯片或 eMMC 芯片作为固件的存储介质。从广义上讲,IoT 设备固件指 IoT 设备运行时所需的代码以及数据的集合,或是存储芯片本身。从狭义上讲,IoT 设备固件可以特指设备文件系统中的某个特定二进制文件。为避免产生歧义,本文定义“IoT 设备固件”为

未解包的固件镜像,或是镜像中包含的文件系统。

IoT 设备固件的类型和大小对固件分析是至关重要的,根据设备使用的操作系统类型和固件大小,本文将 IoT 设备的固件划分为 4 类。

1)A 类固件,这种固件运行在通用计算机使用的 Linux 发行版(如 Debian)或只经过少量裁剪的 Linux 系统中,占用存储空间通常在 1GB 以上,可以自由部署,如 Node-Red,Python 等组件并搭载大量附加功能。

2)B 类固件,这类固件通常使用经过大量裁剪的 Linux 系统或是 IoT 设备专用操作系统,占用存储空间通常为 16~128 MB,可以搭载一部分专用的嵌入式功能。

3)C 类固件,这类固件使用实时操作系统(Real-Time Operating System,RTOS),通常使用 16~256 MB 大小的芯片进行存储,这类固件对设备可靠性要求较高,通常用于航空航天、军事等领域。

4)D 类固件,这类固件使用裸金属(Bare Metal)应用,通常使用 8~16 MB 大小的芯片存储固件镜像,这类设备不具有操作系统,往往通过一个代码中的大型循环来周期性地完成所有任务。

2.2 提取保真度与仿真保真度

为了使用动态分析检测程序中的脆弱性,需要为固件中的二进制文件提供运行时环境,由于工业网关的文件系统较为庞大,递归解包后会占用 40~50 GB 的磁盘空间,使用全系统仿真需要读取操作系统初始化的大量文件,当文件的依赖较为复杂时,使用全系统仿真方法将会造成较低的仿真成功率。Greenhouse^[10]提出了提取保真度和仿真保真度的概念,当提取的文件系统和真实设备中的文件系统的相似程度较高时,可以认为文件系统提取保真度高。类似地,当仿真的目标二进制文件行为与运行在真实设备上的服务行为相似程度较高时,可以认为仿真保真度高。因仿真时需要读取和调用文件系统中的配置文件和库文件,所以仿真保真度强依赖于提取保真度。因此,提取完整的文件系统是保证仿真保真度的重要前提。然而,在工业网关中,设备可能挂载了多种文件系统,通常无法完整地提取出固件镜像中的所有文件,导致提取保真度非常低,因此在仿真过程中有很大可能性出现库文件缺失或软链接失效导致的仿真失败。

2.3 用户模式仿真

用户模式仿真指在不仿真操作系统内核的前提下对单个或多个二进制文件进行仿真,目前主流的用户模式仿真都使用 QEMU-User 用户空间仿真器作为底层解决方案。QEMU-User 提供的用户模式仿真能够转译仿真目标所申请的系统调用、支持信号量处理和多线程执行。QEMU-User 也存在一些缺陷,例如还存在不能转译的系统调用和部分指令集架构的仿真(如 armb),但由于 QEMU-User 是目前最成熟的开源仿真器,且 QEMU-User 支持的指令集架构能够完全覆盖主流的工业网关指令集架构,因此本文选用 QEMU-User 作为用户模式仿真的工具。

3 相关工作

固件仿真指使用通用计算机系统模拟 IoT 设备固件的方

法,这种方法不需要真实的物理设备或只需要部分物理设备就可以通过仿真固件达到测试固件中二进制程序的目的。

QEMU^[11]是一个开源的硬件虚拟化工具,它可以模拟多种硬件架构,包括 x86, ARM, PowerPC, MIPS 等,现在已经成为各仿真器开发的基准,现有的工作大多基于 QEMU 进行开发。本文按照仿真方法对近年来的代表性固件仿真工作进行了总结,如表 1 所列。

表 1 固件仿真的相关工作

Table 1 Related work on firmware emulation

| 现有工作 | 仿真类型 | 固件类型 | 技术 |
|------------|--------|---------|------------|
| FirmAE | 全系统仿真 | B 类 | 仲裁仿真 |
| Greenhouse | 用户态仿真 | B 类 | 单服务仿真 |
| Jetset | 全系统仿真 | B 类 C 类 | 符号执行引导外设建模 |
| Avatar | 硬件在环仿真 | B 类 | 硬件在环 |
| FirmPorter | 全系统仿真 | C 类 | BSP 移植 |

为了解决 IoT 设备仿真这一难题,已经有相当多的学者提出了不同的研究方案。

3.1 FirmAE & Greenhouse

FirmAE^[12]从 Firmadyne^[13]的仿真失败案例中总结经验,提出了一种全系统固件仿真器,且仿真效果严格优于 Firmadyne。FirmAE 通过仲裁仿真方法尽可能绕过 Firmadyne 在仿真过程中发生的错误。这种方法使得 FirmAE 能够避免仿真过程中发生的无关紧要的错误对仿真造成的影响,极大地提升了仿真成功率,让目标固件尽可能启动 Web 服务和网络服务,达到测试 Web 服务的目的。

Greenhouse^[10]提出了一种用户空间仿真方法,为仿真目标提供了类似“温室”的仿真环境,使用了多种方法干预导致仿真失败的原因,并使用了 kubernetes 并行执行多个容器以提升大规模固件托管和测试的效率。

FirmAE^[12],Greenhouse^[10]等工作都使用了类似于“尽力而为”的仿真思想,尽可能在不依赖具体外设的条件下,尽力修复仿真过程中所发生的错误。然而,以上两种工作均为高

提取保真度的固件设计,没有系统性地对目标固件进行文件系统恢复,导致它们很难针对低提取保真度的设备固件展开仿真和测试。

3.2 其他工作

Jetset^[14]提出了一种符号执行技术,能自动推测固件启动所需的硬件信息,使其能够在不安装硬件的前提下自动配置硬件信息,从而让二进制程序在缺少实际硬件的前提下仿真外设。GreenHouse^[10]提出了一种基于用户态的单服务仿真方法,它可以在用户空间执行单个固件服务。Avatar^[15]首次提出了硬件在环仿真方法,其核心思想是使用一个修改后的仿真器(固件代码在仿真器中执行),将与外设之间的交互转发给实际的物理设备处理,以缓解全系统仿真需要对每个外设进行仿真带来的负担。FirmPorter^[16]提出了一种利用 BSP(Board Support Package)对 RTOS 进行仿真的方法,该方法通过移植 BSP 来提供特定硬件相关操作。

综上,现有针对 IoT 设备的测试工具大多支持 B 类固件和 C 类固件,且测试过程中不重视低提取保真度下的仿真工作。然而,主流的工业网关使用的固件类型都属于 A 类固件,且固件提取保真度低,现有工作对其进行仿真测试存在诸多困难。因此,本文使用启发式规则来修复固件文件系统,并在此基础上仿真二进制文件,从而达到对工业网关进行仿真测试的目的。

4 框架设计

针对工业网关文件系统提取保真度低、服务类型特殊等问题,本文基于启发式方法的文件系统修复与仿真干预措施,通过降低文件系统无序程度,并尽可能解决仿真中遇到的障碍点问题,达到了仿真测试工业网关的目的。本文提出了一种基于 QEMU-User 的工业网关固件修复及仿真方法,设计了如图 1 所示的修复和仿真测试框架。它主要包括固件获取与提取、文件系统修复、仿真运行、模糊测试 4 个部分。

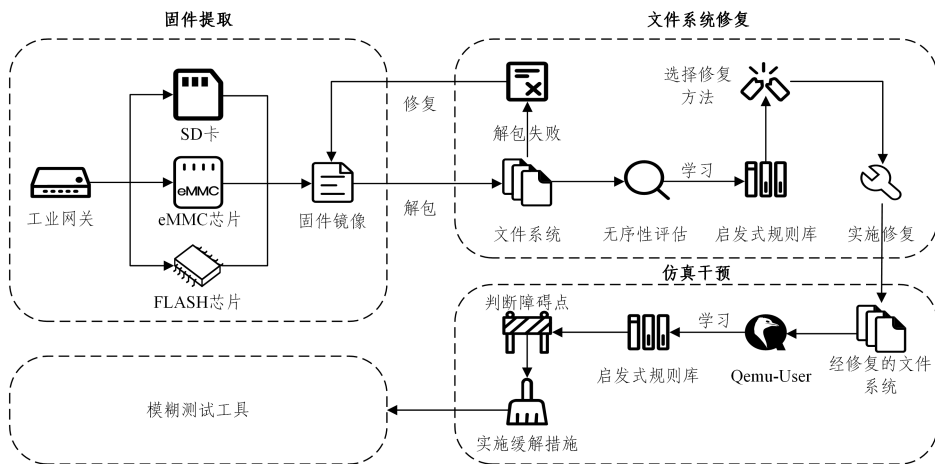


图 1 PerditioImage 框架

Fig. 1 Framework of PerditioImage

4.1 工业网关固件获取

对于传统 IoT 设备,设备固件通常可以通过互联网下载、调试接口 dump、中间人攻击窃听固件更新流量等方法获取。

然而,由于工业网关设备采取的安全措施(封禁调试接口、使用 HTTPS 加密协议等),本文通过硬件方法获取工业网关的固件,使用拆焊工具将芯片从设备主板上取下,并使用芯片编程器读取芯片原始镜像。与一般 IoT 设备固件提取不同,通

过硬件获取方法获取到的工业网关固件通常具有以下特点: 1) 固件镜像中存在大量未使用的空白块, 导致递归提取时得到了大量无用的子镜像文件, 这些文件通过多次递归提取会占用大量磁盘空间; 2) 固件镜像中可能存在多个文件系统或是存在 docker 中间镜像, 使用 binwalk 提取后可能得到多个文件系统; 3) 文件系统可能由于缺少元信息等原因没有被完整提取, 缺失了软链接和目录结构等重要信息。因此, 需要进一步分析和处理并提高提取保真度, 才能实现仿真运行。

4.2 文件系统提取与无序性评估

由于工业网关的固件与常规 IoT 设备不同, 其内部构成更为复杂, 使用直接获得的文件系统不足以支撑后续工作的展开。因此, 本文提出了一种基于文件系统无序性评估来降低文件系统无序性的方法, 通过计算目录中文件数量、文件类型、文件重复次数、损坏的软链接数量等指标来综合评估目录的无序程度, 并通过启发式方法检测多种文件损坏类型并实施修复, 降低文件系统的无序性, 从而提高文件系统的提取保真度。

4.2.1 文件系统提取

本文使用目前最流行的文件系统提取工具 binwalk 作为 PerditioImage 的文件系统提取工具, 并在提取时尽可能保留原有软链接。由于工业网关固件结构的复杂性, binwalk 可能无法解析固件镜像中的文件系统的部分元数据。这会导致目标文件系统出现目录结构丢失等错误, 需要进一步对目标文件系统进行评估和修复。

4.2.2 文件系统无序性评估方法

本文通过对几款工业网关固件提取中发生的问题进行了调查和总结, 发现了影响固件提取保真度的因素主要有以下 3 种: 软链接损坏相关问题、文件目录结构丢失和动态链接库丢失问题。这些问题的出现会严重影响仿真的成功率, 为此分别对以上 3 种问题提出了启发式的修复方案。

为了量化评估文件系统修复的必要性和有效性, 本文提出了一种目录无序性评估方法, 如式(1)所示:

$$H(dir) = \sum_{dir} \left[\frac{N_{file} \cdot \log_2(T_{file} + 1)}{\max(\ln(N_{subdir}), 1)} + 2 \cdot N_{brokenlink} + N_{duplicate} \right] \quad (1)$$

其中, N_{file} 为文件数量, N_{subdir} 为子目录数量, $N_{brokenlink}$ 为损坏的软链接数, $N_{duplicate}$ 为重复的文件数量, T_{file} 为文件种类。该方法综合考虑了目录文件数量、目录文件类型、损坏软链接数量和子目录。此外, 该方法设置了一些额外的标志位和偏移量来更好地评估固件的无序程度, 如 F_{flat} 标志位用于标志存在扁平化目录的文件系统, 当该标志位生效时, 会对无序性指数施加惩罚性的增长来表述文件系统的无序程度增加。

本文使用该方法递归地评估各级目录的无序程度, 并以此为依据对文件系统进行启发式修复。

4.3 基于无序性评估的文件系统启发式修复方法

由于从工业网关中提取的文件系统保真度较低, 需要进行修复等预处理才能进行仿真运行。考虑到工业网关设备固件获取的成本高昂, 使用数据驱动方法可能难以获得有效的训练数据集。相较于机器学习等数据驱动方法, 启发式方法可以从较小的数据样本集中收集经验, 尽可能地对固件进行修复, 此外每一款工业网关文件系统结构的相似性相对传统

IoT 设备较低, 难以达到预期的训练效果。尽管启发式规则库的生成依赖于已有经验, 不可避免地会缩小适用范围, 但影响较小。综合以上考虑, 本文选用启发式方法对文件系统修复和仿真干预。本节主要阐述针对 4 种不同文件系统损坏类型的启发式修复方法。

4.3.1 文件系统损坏类型

损坏类型 1: 软链接损坏。由于从固件镜像中提取文件系统时缺失元数据, 或从存储设备复制文件时丢失了软链接的元数据, 部分软链接文件会丢失目标或指向错误的目标。然而, 有相当一部分软链接承担了文件版本细化的职责。以 libmodbus 动态链接库为例, libmodbus.so 是一个指向了 libmodbus.so.5 的软链接, 而 libmodbus.so.5 是指向了 libmodbus.so.5.1.0 的软链接。因此, 当需要访问 libmodbus.so 时, 需要经过两次软链接重定向。当其中一个软链接损坏而某文件试图调用 libmodbus.so 时, 将会发生无法访问文件错误, 从而造成执行意外停止。

损坏类型 2: 文件目录结构扁平化。由于固件中可能存在 docker 镜像缓存层等特殊文件, 使用 binwalk 解包时, 这些文件可能缺少必要的目录信息, 导致大量文件失去目录结构并存放在文件系统根目录下, 这些文件在文件系统中存在但无法被目标二进制文件访问, 影响了仿真进程。例如, 使用 Wiztree 扫描 Inhand IG502 提取固件镜像后的根目录, 图 2 是其文件目录的扁平化展示, 该文件系统根目录下共有 1313 个文件, 其中框选部分的 1205 个文件为丢失了目录结构的单个二进制文件或配置文件, 在 1205 个没有任何目录组织的文件中包含了 230 个与动态链接库及其相关的软链接。此外, 这些文件中还包含其他仿真所需的配置文件等关键文件, 因此需要以恰当的方式将其恢复。

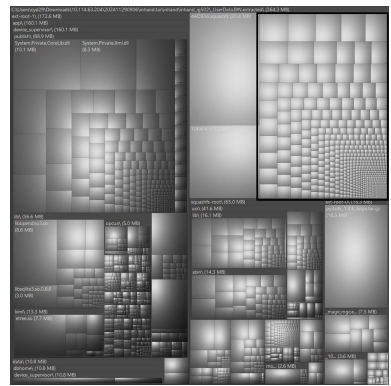


图 2 Inhand IG502 文件系统概览

Fig. 2 Overview of the Inhand IG502 filesystem

损坏类型 3: 动态链接库缺失。动态链接库缺失将导致二进制文件缺少一些组件的核心实现, 进而导致程序异常退出。

损坏类型 4: 固件镜像解析失败。NAND 是 FLASH 存储技术的一种, 由于 NAND 芯片的标准不统一, 使用编程器直接读取 NAND 得到的镜像文件中含有冗余 (Out of Band, OOB) 字段, 无法解析出相应的文件系统。

4.3.2 文件系统修复方法

针对上述文件系统损坏原因, 本文采用了启发式方法对固件文件系统修复, 该方法将自动扫描和推断文件系统

中存在的问题,并给予修复。

修复方法 1(F1):软链接损坏。根据软链接失效的原因,将需要恢复的软链接分为如下 4 种类型:

1)指向了错误地址的软链接。使用 file 命令查询这类文件类型为 broken symbolic link。

2)软链接悬空。文件系统的根目录变化,导致原有指向绝对路径的软链接指向了错误的路径。

3)识别为文本文件的软链接。从设备 SD 卡拷贝文件系统时,描述软链接的元数据可能丢失,使用 file 命令识别这类软链接文件类型为 ASCII text, with no line terminators。

4)识别为数据文件的软链接。部分固件镜像经 binwalk 解包后,由于元数据丢失,软链接文件二进制内容全部为 0x00,然而这类文件的文件大小相比于正常的软链接没有变化。文件类型为 data。

其中,第 1 类和第 2 类错误软链接通常是以绝对路径为目标的软链接,这类软链接会由于根目录的变化而使软链接错误指向或悬空,需要将固件文件系统的根目录作为软链接的实际根目录进行修复,将绝对路径修改为相对路径,从而避免固件二进制文件无法找到文件或是找到错误的文件。第 3 类软链接需要将文件中文本内容作为链接对象进行修复,如果软链接指向的文件存在,则删除文本文件重新构建软链接。第 4 类软链接则需要固件文件系统中寻找与数据文件名相近的文件进行重新链接。

修复方法 2(F2):文件目录结构扁平化。从固件镜像提取的文件系统中,目录结构丢失会导致提取文件的结构扁平化。对于这种情况,需要将 .so 类型的动态链接库文件复制到文件系统根目录中的 usr/lib 下,以供待仿真的二进制文件访问。此外,一些常见的文件(如/etc/localtime 时区文件)将从宿主机文件系统中直接拷贝获取。本文方法不会试图恢复丢失目录结构的所有二进制文件,只会恢复可能影响仿真的文件。其他可能因缺失造成仿真失败的文件,将由干预措施 1 来进行恢复。

修复方法 3(F3):.so 动态链接库缺失。如果文件系统中缺少 .so 动态链接库,则需要将其补全才能让仿真目标正常工作。由于部分动态链接库未发布源代码,无法通过交叉编译获得动态链接库的二进制文件。本文方法使用目前架构支持最丰富的 Debian 12 操作系统作为动态链接库下载来源,Debian 12 支持 10 种指令集架构,能够覆盖 aarch64, armel, armhf, mipsel, mips64el 等常见指令集架构。为了获取正确的动态链接库,本文方法使用 qemu-system 仿真器来仿真启动固件对应架构的 Debian 12.7.0,并使用 apt 下载缺少的动态链接库,最后从仿真的 Debian 系统中拷贝出固件文件系统中缺少的 .so 文件并放置在目标文件系统的/usr/lib 目录下。

修复方法 4(F4):固件镜像解析失败。一些使用了 NAND 技术的 FLASH 芯片,在使用编程器提取内容时会存储芯片的 OOB 字段一起提取,OOB 字段中包含了一些冗余信息,如错误校验位(Error Correcting Code, ECC)、块管理信息等字段,这些字段分散地嵌入在 NAND 的物理擦除块(Physical Erase Block, PEB)中,因此会影响 binwalk 对固件镜像的分析。针对此现象,对不同设备的 NAND 芯片进行了 OOB 字段分析,发现了其中一部分芯片的 OOB 字段分布及

UBI 文件系统^[17]格式与芯片手册和内核源码完全不符,这意味着生产商使用了定制的 NAND 芯片和文件系统,因此无法使用通用方法对这类芯片进行分析。然而,仍然可以从一些芯片的 OOB 字段中找到潜在的模式(如特定字节位翻转、OOB 起始字段偏移量等特征),并根据寻找的模式删除镜像中的 OOB 字段,从而达到正常解析固件的目的。

4.4 单服务仿真运行方法

4.4.1 仿真障碍点

针对低提取保真度的工业网关固件,本文发现了以下几种导致仿真失败的原因,并将其称之为仿真障碍点。

障碍点 1:配置文件或目录缺失。对于一些二进制文件而言,需要特定的配置文件或启动参数才能正确启动服务,当缺少这些参数或配置时,服务将无法启动或以非期望的方式启动。

障碍点 2:运行时参数缺失。使用 QEMU-User 工具对二进制文件进行仿真执行时,若不能提供目标二进制的启动参数,则通常无法启动二进制服务。因此,首先通过 systemd 提供的启动参数来尝试启动二进制程序。

障碍点 3:外设访问失败。外设访问失败是所有仿真器都存在的障碍点,固件中的可执行文件可能访问/dev 目录下一个不存在的外部设备,这通常会启动程序异常结束或崩溃。

障碍点 4:pyc 模块实现。工业网关需要对不同工业协议进行适配,如果工业设备供应商不希望耗费大量资源用于开发新的工业协议交互组件,则意味着供应商需要选择一种现有的开源方案进行开发。经过调查,在本研究的设备测试集中,有一款网关(Inhand)使用了 pyc 文件来实现工业协议相关功能,而 pyc 文件并非 Linux 平台的可执行文件,构成了仿真的障碍。

4.4.2 障碍点干预措施

为了能够成功仿真固件中的二进制文件,需要解决上述问题。本文采用启发式方法学习可能导致仿真失败的原因,并给出了相应的干预措施。

干预措施 1(I1):配置文件或目录缺失。当文件系统中重要的配置文件丢失时,如果能够在整个文件系统中搜索到同名文件或目录,那么就将该文件或目录放入二进制程序期望的位置。若未能找到同名文件或目录,则直接创建一个空文件或目录以供调用。

干预措施 2(I2):运行时参数缺失。当仿真的对象缺少启动参数时,本文将尝试从固件镜像中/etc/systemd 目录下的配置文件中寻找目标待仿真二进制文件的启动参数,如果不能在该目录下找到对应的服务配置文件,本文将会猜测特定服务使用的参数并应用于二进制程序启动。

干预措施 3(I3):外设访问失败。受到 Jetset^[14]工作的启发,本文认为使用基于符号执行的外设推断方法来辅助仿真二进制文件能够获得良好的效果。这种方法利用符号执行来推断固件在运行过程中期望外设返回的数据,然后基于这些推断生成外设模型,使固件能够在仿真器(如 QEMU)中运行。本文使用并修改了 Jetset 工作的部分代码,使其能够更好地运用在工业网关中。

干预措施 4(I4):pyc 模块实现。对于无法直接执行的 pyc 文件,本文使用 decompyle3 对文件系统中所有的 pyc 文

件进行反编译,从字节码文件还原到源码文件,并将其放回 pyc 文件所在目录。此时,由于测试对象由黑盒变为白盒,能够在不仿真的前提下执行 py 文件,从而达到运行程序的目的。

4.5 模糊测试

针对成功仿真的目标程序,本文针对工业网关中的重要服务实现了模糊测试工具。

针对 s7comm 等基于工控协议的信息采集功能,本文使用 Python3 实现了一个可编程逻辑控制器 (Programmable Logic Controller, PLC) 仿真交互工具,该工具可以对工业网关发出的数据收集报文做出响应,本文通过修改报文中的特定字段来达到协议模糊测试的目的。针对 Web 服务器而言,本文使用了 AFL++ 来测试目标服务。

5 实验评估

根据上文提出的用户模式仿真框架,本文实现了 PerditiImage 原型系统,本章将针对工业网关固件的文件系统修复效果、障碍点突破能力和模糊测试能力进行测试。

5.1 实验设置

首先介绍评估实验环境,并将固件仿真的结果与现有的先进工作 FirmAE 和 Greenhouse 进行对比。本文通过以下几个原则来设计测试指标,从而证明本文工作的有效性。

1) 本文的文件系统经过修复后是否比其他仿真方法获得的文件系统提取保真度更高。

2) 本文的 PerditiImage 仿真运行工业网关的大型固件能力如何。

3) 能否有效测试工业网关中的重要服务。

5.1.1 测试环境

本文提出的面向工业网关的用户模式仿真工具 PerditiImage 使用 Python3 和 bash shell 编写,测试使用安装了 Ubuntu 22.04 的通用计算机,使用 binwalk 版本为 2.3.3, docker 版本为 24.0.7, qemu-user-static 版本为 6.2.0。

5.1.2 固件测试集

为了保证测试对象的广泛性和时效性,本文选取了近 5 年发售的 4 款工业网关作为固件测试集,包含了不同的工业设备制造商,此外加入了 4 种 Greenhouse 工作中发现了 0-day 脆弱性的设备固件,且这些设备固件分别来自不同的供应商,以测试本文工作对传统 IoT 设备的通用性,所用设备如表 2 所列。

表 2 测试数据集

Table 2 Dataset for testing

| 设备型号 | 处理器 | 指令集 | 存储器 |
|-----------------------|------------------|-------|---------|
| SIMATIC IoT 2050 | AM6528BACDXEA | ARMv8 | SD Card |
| Schneider PAS 900 | Qualcomm IPQ4019 | ARMv7 | 8192 MB |
| ABB REGATE-10-52-CS07 | AM3352BZCZA100 | ARMv7 | 8192 MB |
| Inhand IG502 | AM3352BZCZ60 | ARMv7 | 8192 MB |
| NETGEAR AC1450 | BCM4708A0 | ARMv7 | 128 MB |
| D-LINK DAP-2330 | QCA9533 | MIPS | 8 MB |
| RT-AC750 | MT7620A | MIPS | 16 MB |
| TRENDNET TEW-652BRP | AR9130 | MIPS | 4 MB |

与一般的路由器不同的是,工业网关是高度定制的行业

产品,因此实际生产的工业网关实现各个工业协议的方法各不相同,包含的服务也各不相同。本文在表 3 中标注了各个工业网关对各种协议的支持,其中服务名为设备中存在服务的具体实现方法,“-”表示文件系统中不含有对应服务。值得注意的是,由于文件系统的损坏,一些设备即使支持特定协议交互(如 Schneider PAS900 支持 s7comm 协议),也无法在提取的文件系统中找到其协议的实现。本文将根据工业网关包含的服务确定测试项,对含有 Web 和 s7comm 等工控协议实现的设备(Schneider 和 Inhand 品牌设备)进行测试。

表 3 设备中的测试对象

Table 3 Test target of devices

| Service | Siemens | Schneider | ABB | Inhand |
|---------|----------|-----------|-----|--------|
| Web | - | httpd | - | httpd |
| modbus | Node-Red | modbusd | - | pyc |
| s7comm | Node-Red | - | - | pyc |

5.2 固件获取与提取

测试目标固件使用了从真实设备上获取的固件镜像。图 3 给出了设备 eMMC 芯片从设备主板上取下并放入编程器烧录座进行固件读取的过程。图 3(a)展示了光学放大镜下的 Schneider PAS 900 设备 eMMC 芯片,图 3(b)展示了使用 Xgecu T48 编程器配合烧录座读取设备 eMMC 芯片。

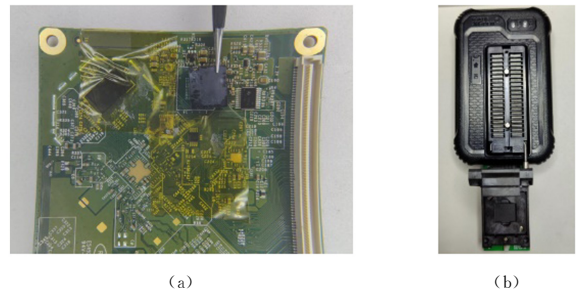


图 3 固件提取工具链

Fig. 3 Firmware extraction toolchain

使用热风枪等拆焊工具直接将芯片从主板中取下,并使用编程器读取芯片中的全部地址空间从而获得设备固件,如图 3 所示。值得注意的是,不同的编程器对芯片内部区域划分不同,实验使用的 Xgecu T56/T48 编程器会从每个 eMMC 芯片中读取到 4 个分区,分别是 BOOT1. BIN, BOOT2. BIN, ECSD_CSD. BIN 及 UserData. BIN。本文针对不同芯片的前 3 个分区进行了分析,认为其中不含有助于固件文件系统分析的部分,因此本文只对 UserData. BIN 部分进行解包。

本文使用 binwalk 作为固件提取工具,并以 binwalk 插件的形式添加 JFFS2, Cramfs, Yaffs2 等文件系统的提取工具,提取时使用 -l (--preserve-symlinks), -M (Mashotrya), -e (Extract) 参数来递归地提取固件镜像中的文件系统并尽可能保留原有软链接。经过测试,能够成功提取固件测试集中的所有固件镜像。

5.3 文件系统修复能力

本文对实验数据集集中的 4 款工业网关和 4 款传统 IoT 设备分别进行了文件系统修复,修复前后对比如图 4 所示,其中纵轴为文件系统无序程度,横轴为设备品牌。不难看出,工业网关的固件复杂度远远高于传统 IoT 设备。本文针对工业网

关中无序程度最大值和最小值进行了研究,发现 Siemens 厂商的 SIMATIC IoT 2050 由于同时搭载了 Python3,Node-Red 等多种平台,大量的库文件导致了文件系统的无序程度偏高。此外,施耐德厂商设备在提取时丢失了部分文件系统,导致无序程度偏低。经过文件系统修复后,各工业网关设备的 $H(fs)$ 之和降低了 27.2%。

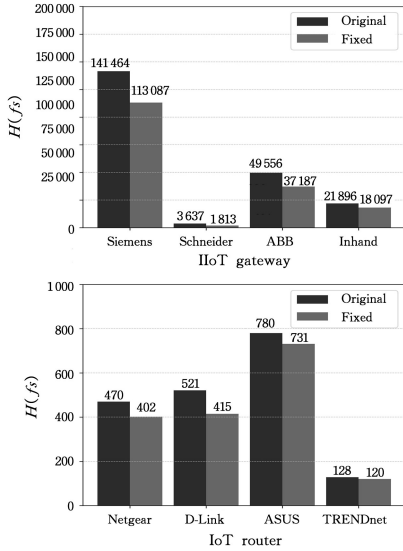


图 4 文件系统修复对比

Fig. 4 Comparison of filesystem repairs

5.4 干预措施对仿真结果的影响

以 Schneider Server Panel Box PAS 900 为例,本文设计了消融实验来研究单个不同文件系统修复方法和干预措施对仿真结果的影响。本文将分别对 httpd 和 modbusd 两个二进制文件实施用户仿真,并逐一取消文件系统修复和干预措施。此外,该设备固件镜像不是经过加密/混淆的 NAND FLASH,且未使用 pyc 实现 modbus 等功能,因此不适用 F4 修复方法和 I4 干预措施,实验结果如表 4 所列,其中“√”表示启用修复方法或干预方法,“-”表示仿真失败,“○”表示仿真成功。

表 4 修复方法与干预措施对仿真结果的影响

Table 4 Effect of fixing and mitigation measures on emulation results

Table with 10 columns: F1, F2, F3, F4, I1, I2, I3, I4, httpd, modbusd. Rows show simulation results for Schneider Server Panel Box PAS 900 under various combinations of fixing methods (F) and mitigation measures (I).

在针对 Schneider Server Panel Box PAS 900 仿真过程中,禁用任意一种文件修复方法都将导致仿真过程失败,因此本文认为在对工业网关进行仿真之前,提升文件系统的提取保真度是必要且有效的。

5.5 固件仿真结果

为目标二进制服务提供运行时环境是进行动态测试的

前提。为了更好地体现 PerditioImage 与现有工作仿真能力的不同,本文分别对测试数据集中的 2 款工业网关和其他 4 款传统 IoT 设备进行了仿真实验,并同时使用目前先进的全系统仿真工具 FirmAE 和用户空间仿真工具 Greenhouse 作为对比,测试结果如表 5 所列,其中“-”表示仿真失败,列出的服务名称为仿真成功的服务。

表 5 仿真结果的对比

Table 5 Comparison of emulation results

Table with 4 columns: 设备 (Device), FirmAE, Greenhouse, PerditioImage. Rows list devices like Schneider PAS 900, Inhand IG502, NETGEAR AC1450, D-LINK DAP-2330, ASUS RT-AC750, and TRENDNET TEW-652BRP with their respective simulation results.

由表 5 中的结果可知,在选取的 4 款传统 IoT 设备仿真测试中,传统 IoT 测试数据集来自 Greenhouse 仿真成功的设备固件,因此 Greenhouse 针对传统 IoT 设备的测试结果没有对比意义,PerditioImage 的仿真能力与 FirmAE 持平。在工业服务仿真方面,FirmAE 和 Greenhouse 均无法针对工业网关固件中的工业数据采集服务进行仿真,PerditioImage 不但能针对传统 IoT 设备进行仿真,而且 PerditioImage 针对工业网关的仿真结果显著优于其他两款工具。

经过分析,FirmAE 和 Greenhouse 均为固件解析设置了严格的驱动器大小限制和解包超时时间,其策略更偏重于如何批量、快速地对大量固件进行并行仿真,但对于仿真不成功的固件,则不会给予太多注意力。在尝试修改了 FirmAE 和 Greenhouse 源码中部分驱动器和超时参数后,仍然有其他问题阻碍它们对工业网关固件进行仿真。而本文的仿真干预方法专注于工业网关这一类设备,能够解决这类固件中独特的挑战。

5.6 脆弱性发现

基于仿真成功的工业网关 s7comm 等工控数据采集服务和 Web 服务,本文展开了模糊测试。

经过测试,成功在 Inhand IG502 的 s7comm 协议交互过程中发现一个拒绝服务脆弱性,如图 5 所示。

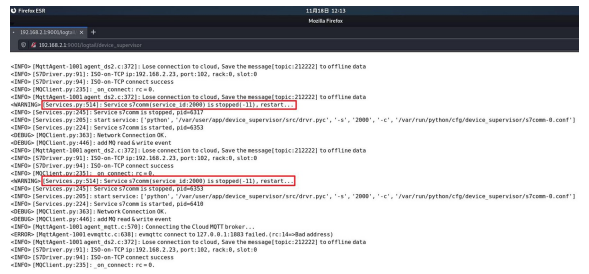


图 5 Inhand IG502 数据采集服务崩溃

Fig. 5 Data acquisition service crash on Inhand IG502

如图 5 中框选部分所示,当 IG502 向仿真的 PLC 发送 s7comm 数据采集报文时,经 PerditioImage 构造的报文能够使得 IG502 的 s7comm 数据采集服务崩溃重启。

结束语 工业控制系统在国家关键基础设施体系中扮演着关键角色,而工业网关则是工业控制系统中的智能化实现

方案,在为工业控制系统引入更多智能的同时,也引入了更多的安全风险,因此保障工业网关的安全问题成为了智能工业领域安全发展的下一个话题。针对现有仿真测试工具在工业网关固件上不能使用的不足,本文提出了一种基于文件系统修复和仿真干预的方法,降低了工业网关中文件系统的无序程度,并开发了 PerditioImage 原型系统用于仿真和测试真实工业网关固件。PerditioImage 通过使用 4 种文件系统修复方法来提升固件提取保真度,通过提供基础的文件访问保障,并在动态仿真时确定 4 种仿真失败干预方法来改善 QEMU-User 仿真时发生的故障。本文对 4 种真实工业网关展开文件系统修复评估,并对其中两款工业网关展开仿真和测试评估,实验结果表明,该工具在工业网关仿真及测试方面优于 FirmAE 和 Greenhouse,并在 Inhand IG502 上发现了一个未公开的拒绝服务脆弱性。

参 考 文 献

- [1] ANDREW L. The vulnerability of vital systems:how critical infrastructure became a security problem [M] // *Securing the Homeland*. Routledge,2020;17-39.
- [2] LI X F, DING Z G, ZHANG S K, et al. Analysis of critical cloud-native technologies and applications in the CT domain[J]. *Telecom Engineering Technics and Standardization*,2024,37(9):83-88.
- [3] ZHENG Y W, WEN H, CHENG K, et al. A Survey of IoT Device Vulnerability Mining Techniques[J]. *Journal of Cyber Security*,2019,4(5):61-75.
- [4] CHEN L, WANG Y, LINGHU J, et al. SaTC: Shared-Keyword Aware Taint Checking for Detecting Bugs in Embedded Systems [J]. *IEEE Transactions on Dependable and Secure Computing*, 2024,21(4):2421-2433.
- [5] SHOSHITAISHVILI Y, WANG R, SALLS C, et al. Sok: (state of) the art of war: Offensive techniques in binary analysis[C] // 2016 IEEE Symposium on Security and Privacy (SP). IEEE, 2016:138-157.
- [6] HAQ I U, CABALLERO J. A survey of binary code similarity [J]. *ACM Computing Surveys*,2021,54(3):1-38.
- [7] FENG X, ZHU X, HAN Q L, et al. Detecting vulnerability on IoT device firmware: A survey[J]. *IEEE/CAA Journal of Automatica Sinica*,2022,10(1):25-41.
- [8] SABBAGHI A, KEYVANPOUR M R. A systematic review of search strategies in dynamic symbolic execution[J]. *Computer Standards & Interfaces*,2020,72:103444.
- [9] ECEIZA M, FLORES J L, ITURBE M. Fuzzing the internet of things: A review on the techniques and challenges for efficient vulnerability discovery in embedded systems[J]. *IEEE Internet of Things Journal*,2021,8(13):10390-10411.
- [10] TAY H J, ZENG K, VADAYATH J M, et al. Greenhouse: Single-Service Rehosting of Linux-Based Firmware Binaries in User-Space Emulation[C] // 32nd USENIX Security Symposium (USENIX Security 23). 2023;5791-5808.
- [11] BELLARD F. QEMU, a fast and portable dynamic translator [C] // USENIX annual technical conference, FREENIX Track. 2005,41(46):10-5555.
- [12] KIM M, KIM D, KIM E, et al. Firmae: Towards large-scale emulation of iot firmware for dynamic analysis[C] // *Proceedings of the 36th Annual Computer Security Applications Conference*. 2020:733-745.
- [13] CHEN D D, MAVERICK W, DAVID B, et al. Towards Automated Dynamic Analysis for Linux-based Embedded Firmware [C] // *Network and Distributed System Security Symposium*. 2016;1-16.
- [14] JOHNSON E, BLANDM, ZHU Y, et al. Jetset: Targeted firmware rehosting for embedded systems[C] // 30th USENIX Security Symposium (USENIX Security 21). 2021;321-338.
- [15] ZADDACH J, BRUNO L, FRANCILLON A, et al. AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems Firmwares[C] // *NDSS*. 2014;1-16.
- [16] XIN M, WEN H, DENG L, et al. Firmware re-hosting through static binary-level porting[J]. *arXiv:2107.09856*,2021.
- [17] The Linux Kernel. UBI FileSystem[EB/OL]. [2025-01-12]. <https://www.kernel.org/doc/html/latest/filesystems/ubifs.html>.



WEI Zihan, born in 2000, postgraduate. His main research interests include IIoT security and reverse engineering.



MA Rongkuan, born in 1992, Ph.D, lecturer. His main research interests include program analysis, software security, ICS security and Web security.

(责任编辑:柯颖)