

物联网设备固件自动化漏洞挖掘技术研究综述

刘航天^{1,2,3}, 甘水滔², 张超^{3,4}, 张红旗^{1,5}, 孙文厚³, 高子聪¹, 赵敏², 白雪¹

(1. 信息工程大学, 河南 郑州 450001; 2. 先进计算与智能工程国家级重点实验室, 江苏 无锡 214084;

3. 清华大学, 北京 100084; 4. 中关村实验室, 北京 100190; 5. 河南省信息安全重点实验室, 河南 郑州 450001)

摘要: 随着物联网技术的广泛应用, 物联网设备爆炸式增长。近年来, 物联网设备导致的安全事件频发, 使得物联网设备安全研究成为热点。首先, 对物联网设备固件的安全特性(包括黑盒特性、网络特性和定制化特性)进行了深入剖析, 这些特性给自动化漏洞挖掘带来了新的挑战。固件代码闭源、运行环境封闭、网络交互复杂、软硬件高度定制化, 这些都增加了固件安全分析的难度。针对这些挑战, 研究人员研发了一系列解决方案。通过对现有文献进行综合分析, 从黑盒模糊测试、灰盒模糊测试、静态程序分析和固件重托管4个方面, 总结了物联网设备固件自动化漏洞挖掘技术的最新研究进展。在此基础上, 分析了尚存在的挑战和不足, 并提出了未来的研究方向和思路, 包括多种技术有机组合的发展趋势、大语言模型在自动化漏洞挖掘中的应用前景, 以及物联网技术演进驱动漏洞挖掘技术的同步升级。通过对物联网设备固件自动化漏洞挖掘技术现状及发展趋势的深入剖析和总结, 为业界下一步研究及应用提供有价值的参考。

关键词: 物联网设备; 黑盒模糊测试; 灰盒模糊测试; 静态程序分析; 固件重托管; 大语言模型

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2025014

Survey on automated vulnerability mining techniques for IoT device firmware

LIU Hangtian^{1,2,3}, GAN Shuitao², ZHANG Chao^{3,4}, ZHANG Hongqi^{1,5},
SUN Wenhou³, GAO Zicong¹, ZHAO Min², BAI Xue¹

1. Information Engineering University, Zhengzhou 450001, China

2. Laboratory for Advanced Computing and Intelligence Engineering, Wuxi 214084, China

3. Tsinghua University, Beijing 100084, China

4. Zhongguancun Laboratory, Beijing 100190, China

5. Henan Key Laboratory of Information Security, Zhengzhou 450001, China

Abstract: With the wide application of IoT technology, IoT devices have exploded. In recent years, security incidents caused by IoT devices have occurred frequently, which makes the research of IoT device security become a hot spot. The security analysis of IoT device firmware has been conducted, with a focus on its black-box nature, network characteristics, and customization features. Challenges to automated vulnerability mining have been high-

收稿日期: 2024-07-30; 修回日期: 2024-11-27

通信作者: 刘航天, pydrfly@163.com

引用格式: 刘航天, 甘水滔, 张超, 等. 物联网设备固件自动化漏洞挖掘技术研究综述[J]. 网络与信息安全学报, 2025, 11(2): 26-49.

Citation Format: LIU H T, GAN S T, ZHANG C, et al. Survey on automated vulnerability mining techniques for IoT device firmware[J]. Chinese Journal of Network and Information Security, 2025, 11(2): 26-49.

lighted, such as the closed-source firmware code, closed operating environment, complex network interactions, and highly customized hardware-software. Researchers have proposed a series of advanced technologies and methods to address these challenges. The existing literature was comprehensively analyzed, and the latest research progress in automated vulnerability mining technology for IoT device firmware was summarized from four aspects: black-box fuzzing, gray-box fuzzing, static program analysis, and firmware re-hosting. Based on the analysis of the current research status, existing challenges and deficiencies were pointed out, and future research directions and ideas were proposed, including the development trend of multi-technology organically combination, the application prospects of large language models in automated vulnerability mining, and the synchronous upgrade of vulnerability mining technology driven by the evolution of IoT technology. An in-depth analysis and summary of the current status and development trends of automated vulnerability mining technology for IoT device firmware were provided, offering valuable references for future research and applications in the industry.

Keywords: IoT device, black-box fuzzing, gray-box fuzzing, static program analysis, firmware re-hosting, large language model

0 引言

物联网 (internet of things, IoT) 技术蓬勃发展, 全球物联网设备爆炸式增长^[1]。物联网是嵌入式设备与网络技术深度融合发展的产物, 因此物联网设备具有典型的嵌入式设备特点, 是嵌入式技术的继承与发展。在人们的生活中, 到处都有嵌入式设备的身影, 无论是消费电子产品, 还是工业领域的控制设备。在物联网技术的加持下, 这些设备具备了网络通信能力, 逐渐互联互通, 构建起规模巨大的万物互联网。物联网设备的广泛应用以及设备间互联互通的特性, 使其成为黑客攻击和数据泄露的高危地带。例如, 2015年12月23日, 乌克兰电力系统遭受攻击, 黑客在1天内覆写了16座变电站的关键性设备固件, 攻击了约60座变电站, 导致基辅大规模停电^[2]。再如近年持续爆发的Mirai僵尸网络攻击事件, 黑客利用漏洞入侵并控制海量物联网设备形成僵尸网络, 在全球范围内对多种网络基础设施造成了严重破坏, 其中对域名解析服务商Dyn发动的攻击造成美国东海岸地区网络大面积瘫痪^[3]。面对日益严峻的物联网安全挑战, 保护物联网安全至关重要。

物联网设备由硬件及运行其上的固件构成, 固件程序中存在大量漏洞。近年来, 物联网设备固件自动化漏洞挖掘技术研究取得了重大进展, 国内外的一些研究人员对该领域的研究成果进行了深入分析和总结。文献[4]围绕嵌入式设备固件面临的安全风险, 对相关安全分析技

术进行了综合分析和评估。文献[5]指出物联网设备固件漏洞挖掘面临的挑战主要是硬件资源受限、硬件复杂异构和代码文档不公开, 从静态分析、动态模糊测试和同源性分析3个方面总结了该领域的研究进展。文献[6]从仿真测试、静态分析、网络模糊测试和手工逆向分析4个方面总结了物联网设备固件漏洞挖掘技术研究进展。

本文与现有文献综述在分析方法、归纳方式和研究思路方面有所不同, 为该领域的研究工作提供了新的视角和思路。此外, 物联网设备固件自动化漏洞挖掘技术研究不断涌现出新的成果, 极大推动了本领域的研究进展。其中, 大语言模型 (large language model, LLM) 在程序分析领域初露端倪^[7-10], 展现出强大的潜力。本文全面整理和总结了近年来该领域的研究进展, 在此基础上深入分析了未来发展趋势。

本文首先剖析了物联网设备区别于通用软件系统的安全特性 (如黑盒特性、网络特性与定制化特性), 以及这些特性给物联网设备固件自动化漏洞挖掘 (如模糊测试、程序分析) 带来的挑战。

围绕这些挑战, 国内外研究人员提出了一系列解决方案, 本文收集了近年来该领域的相关文献资料, 梳理了相关技术的发展脉络并进行了分析和总结。具体来讲, 本文综述了物联网设备固件自动化漏洞挖掘的黑盒模糊测试、灰盒模糊测试、静态程序分析和固件重托管4种主流技术的最新研究进展。

最后, 本文指出了物联网设备固件自动化漏

洞挖掘未来的研究方向,包括多种技术的有机融合与互补发展、LLM等人工智能技术的智能化赋能,以及物联网技术演进驱动漏洞挖掘技术的同步升级。

1 背景知识

通用漏洞挖掘方法构成了物联网设备固件自动化漏洞挖掘技术的基础,模糊测试和程序分析是两种主流的漏洞挖掘技术。模糊测试技术通过对输入变异持续地生成大量测试用例,自动触发被测目标程序产生异常。根据对目标程序信息的利用程度,模糊测试技术进一步分为黑盒测试技术和灰盒测试技术。黑盒测试是在完全不了解程序内部结构的情况下进行测试,而灰盒测试利用部分程序信息指导测试。程序分析技术通过对程序代码、控制流和数据流等要素进行建模和分析,结合漏洞模式全面且深入地检测程序中潜在的危险路径和行为。固件重托管将设备固件从物理硬件中解耦出来并在模拟环境中重新运行,为漏洞挖掘提供了可控和可配置的基础环境。

图1展示了物联网设备固件自动化漏洞挖掘技术的分类及其相互关系。静态程序分析技术和模糊测试技术互为补充,静态程序分析为模糊测试提供了更加全面的程序视角,而模糊测试能够有效弥补静态程序分析因缺乏动态信息而存在的不足;固件重托管技术打破了物联网设备的黑盒特性,能够有效支撑其他漏洞挖掘技术,同时其

他程序分析技术是实现固件重托管的重要手段;借助其他的程序分析技术,黑盒模糊测试实现了有限反馈机制,逐渐趋向灰盒化。

这些技术相互关联、相互融合,共同推动着该领域的发展。

1.1 模糊测试

1988年,美国威斯康星大学的Miller Barton教授首次提出了“模糊测试生成器”概念。该生成器用于测试UNIX程序的健壮性,即用随机数据测试程序,直至程序崩溃。2001年,芬兰奥卢大学发布的PROTOS项目首次将模糊测试技术用于网络协议的安全测试,之后模糊测试技术被应用于商业产品^[11]。在模糊测试技术诞生后,很长一段时间主要以黑盒测试为主。为了解决黑盒测试的盲目性,提升测试效果,研究人员提出了多种解决方案。2004年,采用Peach工具对文件进行模糊测试,并自定义数据模型方式生成输入^[12]。2007年,SAGE^[13]利用动态符号执行技术辅助模糊测试,通过符号执行生成符合语法约束的种子,取得了不错的效果。2009年,Ganesh等^[14]使用污点分析技术替代开销巨大的符号执行技术,用来指导模糊测试变异策略。

2013年,AFL(american fuzzing loop)^[15]首次采用源优码编译插桩实现代码覆盖引导的进化式模糊测试,开启了灰盒模糊测试的新时代。随后,以代码覆盖率引导的模糊测试技术迎来了爆发式发展。如图2所示,代码覆盖率引导的灰盒

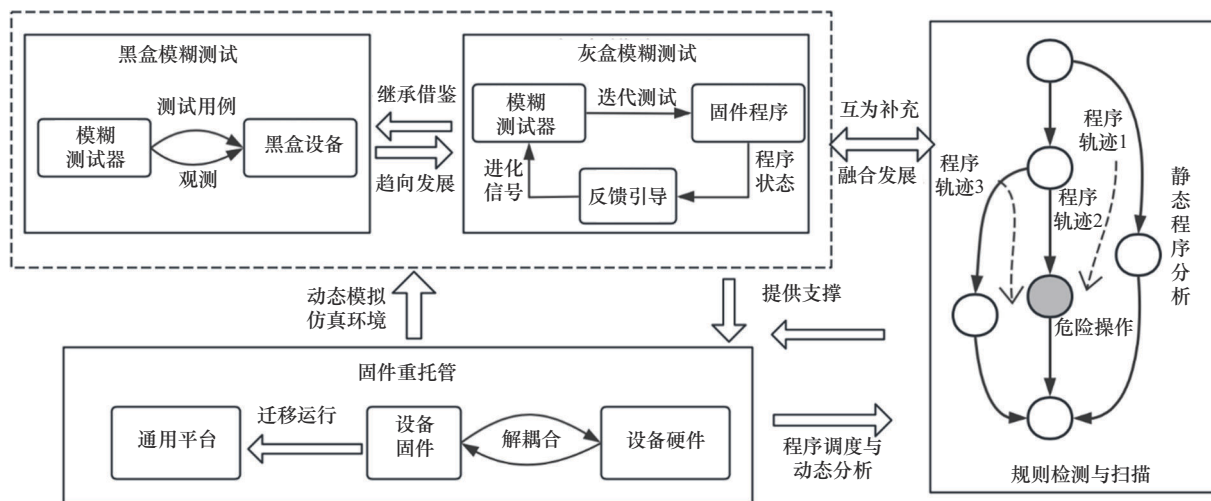


图1 物联网设备固件自动化漏洞挖掘技术的分类及其相互关系

Figure 1 Categorization of automated vulnerability mining techniques for IoT device firmware and their interrelationship

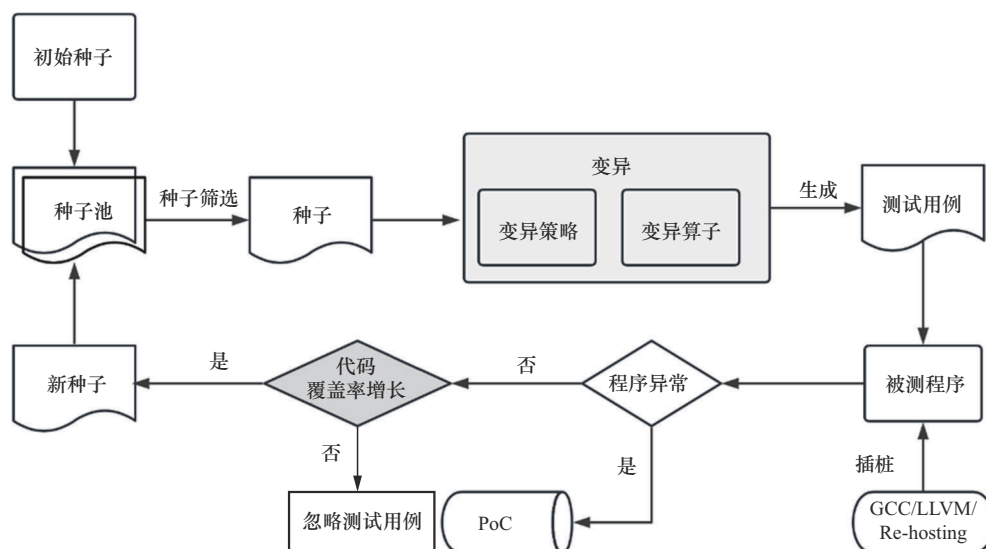


图2 代码覆盖率引导的灰盒模糊测试框架

Figure 2 Code coverage guided gray-box fuzzing framework

模糊测试在测试过程中不断监测测试用例的执行路径，如果发现执行了新的代码，就标记此测试用例，将其作为优势种子保存下来，在优势种子的基础上不断尝试触发新的代码逻辑。此后，基于AFL演进的模糊测试方案如雨后春笋般涌现出来，如CollAFL^[16]、AFLFast^[17]、Vuzzer^[18]、LibAFL^[19]等。针对不同测试目标的灰盒模糊测试工具快速发展起来。如对Linux内核开展模糊测试的kAFL^[20]和Syzkaller^[21]工具，以及面向Windows内核的WinAFL^[22]工具。为实现对闭源程序的灰盒模糊测试，程序插桩技术也获得快速发展，典型代表有ARMore^[23]、DynamoRIO^[24]、Pin^[25]。

以代码覆盖率为引导的灰盒模糊测试方案取得了巨大进步，其核心思想在于引导测试执行更多不同的代码来提高程序异常暴露的概率。但程序中容易引起程序异常的危险代码片段，它们的分布十分不均。从漏洞挖掘角度考虑，不同代码片段的危险程度不同，其测试权重也应该有所差别。Bohme等^[26]首次提出定向灰盒模糊（directed graybox fuzzing, DGF）测试技术，它首先确定程序中危险代码的目标位置，计算测试用例执行轨迹与目标代码之间的距离，通过距离反馈调节和引导变异策略，不断引导测试用例趋向执行目标代码。定向灰盒模糊测试技术近年来快速发展，取得了长足进步，其中出色的代表方案有BEACON^[27]、Hawkeye^[28]、CAFL^[29]、EcoFuzz^[30]。

1.2 程序分析

程序分析的内涵和延展非常宽泛，包括类型分析、格理论与不动点、数据流与控制流分析、程序切片、抽象解释等。类型分析是指检查程序中变量和表达式的类型，确保类型安全，即数据类型正确使用。格理论提供了一种数学框架，用于描述不同元素之间的部分顺序关系，表示数据流或控制流的不同状态。不动点是格理论的一部分，用于寻找分析算法的稳定状态，即分析结果不再变化的状态。数据流分析关注程序中数据的流动，如变量的赋值和使用。控制流分析关注程序中控制结构的执行路径，如条件分支和循环，有助于理解程序的执行流程和可能的执行路径。程序切片指根据程序中特定的需求（如变量或函数）提取出程序的一部分，其中包含所有可能影响需求的代码。这有助于理解代码的依赖关系和影响范围。抽象解释将程序的精确语义映射到一个更抽象的模型以简化分析，允许分析算法在保持一定精度的同时处理大型程序。

与模糊测试不同，程序分析建立在对代码静态分析的基础上，通过代码分析建立程序的抽象模型，结合规则检测以寻找代码中的潜在脆弱点。程序分析通常以控制流和数据流分析为基础，结合程序规范与漏洞模式进行深入分析和检测。在针对物联网设备固件的安全性分析中，污点分析和符号执行是两项常用的技术。

污点分析的核心理念在于追踪数据从其初始来源（污点源）到其在程序中的最终用途（污点汇聚点）的传播路径。这种方法可以抽象为一个三元组<污点源, 污点汇聚点, 无害处理>^[31]。污点源是指那些向系统引入不受信任数据的代码点，它们是数据潜在危害传播的起点。污点汇聚点是指那些执行安全敏感操作（如命令执行）或泄露隐私数据（如网络输出）的代码点，它们是数据传播的终点。无害处理是指使用数据脱敏等手段移除数据危害的操作，确保数据流动不会对系统安全构成威胁。污点分析示意如图3所示。污点分析的目的是检测是否存在未经无害处理的数据直接从污点源传播到污点汇聚点的路径，一旦发现这样的路径，即表明系统可能面临安全风险。

在实际应用中，污点分析面临诸多挑战，尤其是在处理复杂的控制流（如间接调用）和复杂的数据结构时，数据的追踪变得尤为困难。这影响了污点分析的准确性，导致误报和漏报。传统的污点分析技术，如TAJ^[32]、Andromeda^[33]和FlowDroid^[34]，主要依赖于静态分析方法。它们在不运行且不修改代码的情况下，通过分析程序变量间的数据依赖关系，检测数据是否能够从污点源传播到污点汇聚点。与静态污点分析技术相对的是动态污点分析技术，如TaintDroid^[35]和Flexitaint^[36]。这类技术在程序运行过程中实时监控污点数据的传播，以检测其是否能够从污点源传播到污点汇聚点。动态污点分析虽然能够提供更加精确的分析，但依赖程序插桩或专用硬件，且会额外引入巨大的资源开销，从而在一定程度上限制了

在物联网设备固件程序安全分析中的应用。

“符号执行”这一概念最早由King^[37]于1976年在ACM上提出，用符号值替代具体输入值，并使用符号表达式追踪程序中与这些符号值相关的变量状态，从而深入挖掘程序的潜在行为。在遇到程序分支时，符号执行技术会探索所有可能的分支路径，并将分支条件转化为路径约束。如果这些约束可以求解，那么对应的路径就是可达的。这种方法的优势是能够以较少的测试用例实现高覆盖率的测试，有效地揭示复杂软件程序中隐藏的深层错误。符号执行尽管在理论上具有显著优势，但在实际应用中面临着巨大的挑战：首先是路径爆炸问题，程序可能存在指数级增长的执行路径，导致分析过程变得复杂和耗时；其次是约束求解难度大，对于复杂的约束，求解器难以找到答案；最后是内存和计算资源的限制，符号执行在执行大规模程序分析时需要消耗大量的内存和计算资源，这在实际应用中是不可行的。为了打破这些限制，研究者发展了动态符号执行技术，将实际执行和符号执行相结合以实现更高效的程序分析，由此诞生了一批以SAGE^[13]、KLEE^[38]、SPF^[39]、Pex^[40]、S2E^[41]为代表的符号执行工具。与传统的静态符号执行相比，动态符号执行通过引入实际值的分析，简化了分析过程，降低了分析难度，并提高了效率。这种方法减少了对求解器能力的过度依赖，使得符号执行技术在面对现实世界的软件测试问题时更加有效。尽管动态符号执行带来了明显的优势，但在分析过程中仍然需要将程序语句准确地转换为符

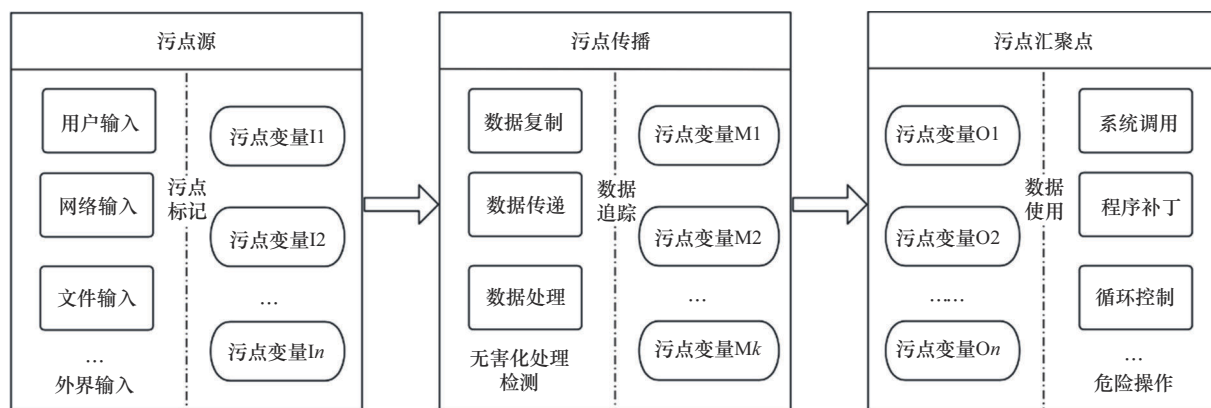


图3 污点分析示意

Figure3 Taint analysis illustration

号约束，其转换的精度与测试覆盖率和约束求解的可伸缩性密切相关。

随着程序分析技术的发展，动态执行的特点日益显著。程序分析与模糊测试的区别是，程序分析关注的是代码执行过程中的数据流，而模糊测试关注的是代码执行的结果。程序分析通常在代码执行之前就进行，对关键变量等数据进行标注，以便在执行过程中追踪和分析数据流。相比之下，模糊测试本身无须对程序进行深入分析，通常在程序运行结束后再利用逆向工程等手段对结果进行分析，以澄清漏洞成因。

1.3 固件重托管

固件重托管的核心在于构建一个能够仿真固件依赖的硬件特性的模拟环境，允许固件在虚拟机上像在真实硬件上一样运行，使得研究人员能够在一个可控的、可配置的软件环境中对固件进行深入分析，包括模糊测试、逆向工程、程序调试等。这涉及对系统硬件（包括处理器、内存、外设等）的准确模拟，以及对固件与这些硬件组件交互的精确捕获。固件重托管示意如图4所示。

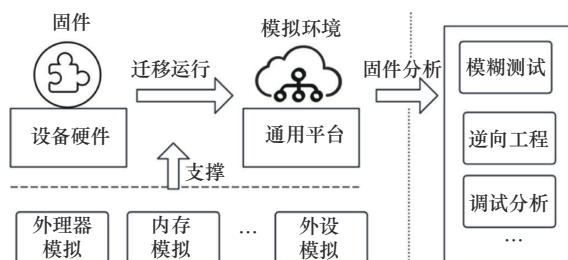


图4 固件重托管示意

Figure 4 Firmware re-hosting illustration

固件重托管的发展可以追溯到早期的硬件仿真和指令集架构级别的模拟^[42]。随着嵌入式系统复杂性的增加，传统的硬件仿真方法面临扩展性差和资源消耗大的问题。因此，研究人员开始探索更加专注于固件层面的重托管方法。这些方法通常需要对固件进行一些修改以适应模拟环境，包括对操作系统、驱动程序和应用程序层的调整。当前固件重托管技术已经发展出多种策略，包括纯软件仿真、硬件在环仿真、符号抽象等策略。每种策略都有其优势和局限性，选择哪种方法取决于分析目标和可用资源。例如，硬件在环仿真提供了高保真度的硬件行为模拟，但需要对物理硬件进行复杂的接口和调试支持。符号抽象方法使用符号值模拟硬件行为，有助于处理状态空间爆炸问题，但可能会引入虚假的分析结果。固件重托管仍然面临着创建高效的虚拟执行引擎、处理外设行为的多样性、量化重托管系统的保真度和支持复杂系统的重托管等重大挑战。

2 物联网设备固件的特点及其漏洞挖掘面临的挑战

物联网设备固件与通用软件系统相比，具有一些独特的特性，这些特性为自动化漏洞挖掘带来了一系列新的挑战，如图5所示。

2.1 物联网设备固件的特点

物联网设备固件具有黑盒特性、网络特性（安全风险主要来自网络通信接口），以及定制化特性（如资源受限且软硬件高度定制化）三大特性。

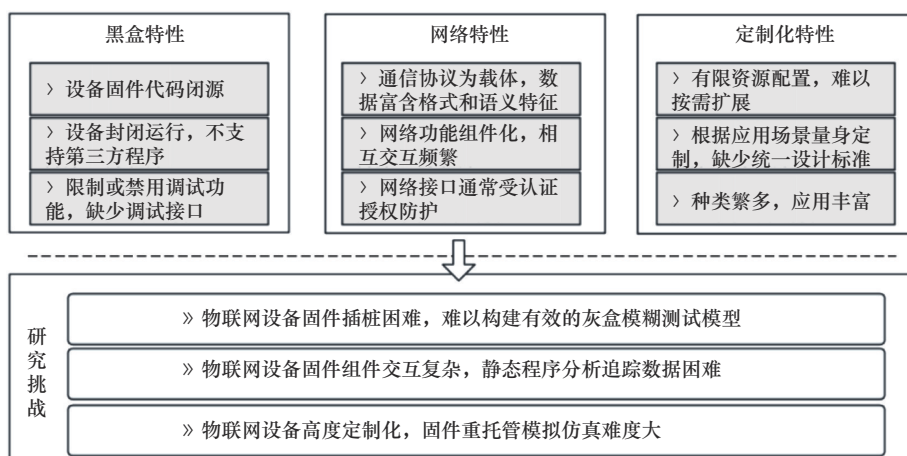


图5 物联网设备固件的特点及自动化漏洞挖掘研究挑战

Figure 5 Characteristics of IoT device firmware and challenges in automated vulnerability mining research

(1) 物联网设备固件具有黑盒特性

这主要表现在以下几个方面。

1) 设备固件代码闭源, 即设备固件源代码不对外开放, 这表现为设备固件通常为单一压缩文件, 压缩文件内部打包了设备运行所需的程序和配置文件, 固件程序多为无符号的二进制文件, 往往要结合逆向工程分析固件中的程序代码, 这限制了对设备的深入分析。

2) 设备往往在封闭的环境中运行, 不开放接口运行第三程序。物联网设备固件有相当比例是基于 Linux 系统开发的, 其本身是支持第三方程序运行的, 但缺少相关开发组件与服务, 安全研究人员往往需要在破解物联网设备固件的基础上才能运行第三方编译的程序。物联网设备面向专用领域开发, 所以无须像通用软件系统那样提供开发接口, 这也就限制了它们的灵活性和扩展性。

3) 物联网设备通常缺少调试接口, 设备在定型生产时往往会限制或禁用调试功能, 这进一步增加了提取设备内部状态的屏障。

(2) 物联网设备固件具有网络特性 (安全风险主要来自网络通信接口)

设备开放接口少, 通常仅能通过少量网络接口与设备通信。常见的网络接口包括 Web、移动 App 等, 这些接口用于设备与人之间通信, 通常用于设备管理, 在访问管理接口之前往往需要对用户的身份进行认证与授权。另外一类网络接口用于设备之间通信, 如 ZigBee 协议接口、路由协议接口、UPnP 接口, 用户通常不会直接接触这类接口。网络接口通信通常使用特定协议, 这些协议包含丰富的语法语义信息, 如广泛应用的 Web 接口使用 HTTP 通信。有限的开放网络接口限制了设备的安全威胁范围, 网络接口是其主要攻击面。物联网设备的网络接口通常具有以下几个特点。

1) 网络接口的输入/输出交互以通信协议为载体, 通信协议包含清晰的格式定义, 以及丰富的语法和语义。部分通信协议富含状态, 一次通信通常由多个状态紧密相关的报文序列构成。

2) 实现设备网络功能的程序组件化、模块化, 相互之间交互频繁, 多见进程间与线程间的

相互调用和通信。

3) 网络接口通常有认证授权机制防护, 大部分功能代码位于认证授权机制之后, 要测试功能代码则需要通过或突破认证授权机制。

(3) 物联网设备具有定制化特性 (如资源受限且软硬件高度定制化)

物联网设备配置的资源通常有限, 这些资源仅够满足其特定的应用需求。一旦设备定型, 其计算和存储资源很难进行扩展, 这进一步限制了它们在功能上的灵活性和扩展性。此外, 物联网设备是针对特定的应用领域设计的, 目前并没有统一的设计标准。不同厂商生产的设备在功能和性能上存在显著差异, 每一款设备都是根据其特定的应用场景量身定制的, 包括专用的硬件部件和软件配置。这种高度定制化的设计虽然能够满足特定需求, 但是给程序分析带来了巨大的障碍, 尤其是难以设计通用的程序分析框架, 往往需要针对不同款设备的特性进行定制化适配和开发。物联网设备种类繁多, 应用场景十分丰富。

2.2 物联网设备固件自动化漏洞挖掘面临的挑战

物联网设备固件的自动化漏洞挖掘面临下列三大挑战。

(1) 物联网设备固件插桩困难, 难以构建有效的灰盒模糊测试模型

以代码覆盖率为引导的灰盒模糊测试是先进的自动化漏洞挖掘方法之一, 在挖掘通用软件系统漏洞方面表现出了强大的能力, 但却难以直接挖掘物联网设备固件的漏洞。物联网设备的黑盒特性导致缺少引导灰盒模糊测试变异的程序状态反馈, 测试难以覆盖到固件深层的代码逻辑。灰盒模糊测试技术的关键在于实时的程序状态反馈, 这种方式依赖代码插桩, 通过在程序中植入代码, 收集动态运行时的状态信息以引导测试过程不断触发新的执行路径。在有源代码的情况下实现代码插桩比较容易, 而物联网设备固件通常是闭源的, 并且部分厂商对固件进行了如加密和完整性校验等加固措施。在这种情况下难以通过代码插桩的方式获取程序执行状态。对此, 通常采用以下 3 种方式应对。

1) 静态代码插桩。这种方式直接修改程序,

插入桩代码, 固件程序以二进制程序为主, 面临代码和数据指针的调整等问题, 在不改变程序原有功能的前提下实现对程序的精确修改是一个复杂且艰巨的任务; 此外, 还面临固件破解和重打包的问题, 进一步加剧了此方式的实施难度。

2) 基于硬件的程序追踪机制。英特尔和ARM的部分处理器分别提供了Intel-PT^[43]和ETM^[44]硬件单元来实现指令级的程序追踪功能。但这在物联网设备中鲜有应用。一方面, 这与设备的黑盒特性相关, Intel-PT和ETM主要面向程序开发者提供调试和分析功能, 而物联网设备面向专用领域并不提供开发接口, 在生产定型后会关闭与开发相关的功能; 另一方面, 物联网设备应用场景丰富, 使用的处理器类型众多, Intel-PT和ETM通常面向桌面系统等高功耗场景, 这限制了它们在此领域的应用范围。

3) 固件重托管技术。这种技术使得固件能够脱离原本的硬件环境而运行在模拟的仿真环境中, 能够打破物联网设备的黑盒屏障, 有效获取程序内部的执行状态, 为程序分析和调试提供了通道和途径; 同时还能够克服设备资源受限的困难, 通过将固件运行在资源充足的计算平台上, 实现资源的按需扩展和分配。固件重托管技术虽然有着明显的优点和广阔的应用空间, 但受限于物联网设备软硬件高度定制化的特点, 难以构建通用的模拟仿真框架, 同时面临着成功率与忠实度不足等问题。

(2) 物联网设备固件组件交互复杂, 静态程序分析追踪数据困难

静态程序分析在漏洞挖掘领域经久不衰。它不依赖于程序动态运行, 仅通过分析程序代码发现潜在问题, 自动化程度高, 具有良好的可扩展性。但静态程序分析缺少程序动态运行过程中的信息, 在静态条件下难以完整恢复程序的控制流和数据流信息, 从而大大影响了静态程序分析的准确率和精度。对于物联网设备固件, 静态程序分析面临着以下多重挑战。

1) 物联网设备固件的安全风险主要来自网络接口, 在设备中网络通信通常涉及多个组件之间的通信和调用, 多组件相互依赖形成了复杂的数

据交互。固件程序处理数据的链条长, 从数据输入到最终触发漏洞的程序片段往往经过多个环节, 甚至是跨程序间的数据处理和传播, 这大大增加了数据追踪的复杂性和难度。

2) 物联网设备固件需要处理网络通信和多种异步任务, 因而在开发过程中经常使用回调机制, 虽然这种方式能够减少代码之间的耦合性, 使得模块之间的依赖关系更加清晰, 方便代码扩展, 但增加了静态程序分析的难度。回调机制在二进制程序中通常表现为间接调用, 尤其是在涉及多个回调和嵌套回调的情况下, 其函数上下文难以明确, 这无疑增加了控制流追踪的难度。

3) 物联网设备固件的网络特性使得其程序需要处理富含语法结构的报文, 因此程序中的数据通常隐含逻辑关系, 表现为复杂的数据结构, 在缺少源代码情况下很难恢复二进制程序中的数据结构, 这进一步增加了程序分析的难度。

4) 静态程序分析产生的误报较多, 此类工具通常会生成大量警告, 但这些警告并不一定表示实际的安全问题。静态程序分析的局限性会导致错误地标记一些无害的代码片段。物联网设备固件的程序往往更加复杂和多样化, 这使得静态程序分析工具更容易产生误报。

5) 为了确定静态程序分析工具报告的警告是否真的存在安全漏洞, 通常需要安全专家进行进一步的人工分析。这通常涉及构造一个概念验证 (proof of concept, PoC) (即编写一段代码) 来实际触发和展示警告所指出的问题。在物联网设备固件程序的上下文中构造PoC特别具有挑战性。这是因为物联网设备通常运行在资源受限的环境中, 且其程序与特定的硬件紧密集成。因此, 即使是经验丰富的安全专家, 也难以在没有实际硬件的情况下准确地构造出能够触发和验证安全漏洞的PoC。

(3) 物联网设备高度定制化, 固件重托管模拟仿真难度大

固件重托管能够在缺失实际硬件的情况下对固件进行深入的安全分析, 以发现潜在的安全漏洞。这种技术能够为其他程序分析方法提供良好的环境和基础。具体来说, 实现固件重托管后为构建灰盒模糊测试框架提供了灵活的程序状态反

馈,能够有效引入和发展面向物联网设备的灰盒模糊测试模型;固件重托管为静态程序分析提供了程序调试和动态分析的条件,可以有效弥补静态条件下的信息缺失,提升静态程序分析的准确率和精度,同时为PoC生成和验证提供了良好的环境。但固件重托管用于物联网设备固件时面临着以下困境。

1) 物联网设备固件的黑盒特性使得建立准确的仿真模型具有相当的挑战性。在缺少源代码、技术开发手册的情况下,分析和理解设备内部的工作机制变得非常困难。一些非标准的硬件(如定制化外设)在通用仿真工具中并没有相应的模拟支持,因而难以实现仿真。固件与硬件之间的这种紧密耦合使得在不完全了解硬件的情况下难以准确模拟固件的行为。

2) 物联网软硬件高度定制化,难以构建通用的固件仿真模型框架和方案,通常依赖人工经验和逆向分析来对定制化的组件进行建模。例如,定制化外设需要模拟硬件抽象层(hardware abstraction layer, HAL)的行为;设备中复杂的中断和事件处理机制则需要准确地予以建模和实现来正确模拟设备行为。现实的固件重托管方案通常会在一定程度上牺牲仿真精度来提升仿真的成功率或仅实现局部仿真。虽然高保真仿真对于固件安全分析并非必需的,但这可能影响漏洞发现的准确率,在模拟环境中发现的漏洞报告通常还要在真实环境中进行验证。

3) 固件重托管并不一定能够带来性能的提升,对于一些高性能专用部件(如密码运算单元)的模拟,软件的运行效率很难与硬件相媲美。同时,尽管不需要实际硬件,但仿真环境本身可能需要较多的计算资源和存储资源,特别是当需要模拟多个设备或复杂的系统时,资源需求会显著增加。

3 黑盒模糊测试技术

3.1 研究进展

物联网设备固件代码闭源,不支持第三方程序的运行,缺少调试接口,具有很强的封闭性。这导致当前先进的灰盒模糊测试方式难以在物联网设备上应用。黑盒模糊测试依赖条件少,无须对设备进行侵入式插桩和改造,实施方便,同时

避免了插桩引入的性能损耗,因此在物联网设备固件漏洞挖掘领域依然备受青睐,研究热度颇高。

许多商用的物联网设备固件代码闭源且缺少模拟仿真环境,难以构建程序状态引导的灰盒模糊测试方案。虽然黑盒条件下难以获得精确的程序执行信息,但通过深挖设备泄露的外部信息依然能够在一定程度上推导出设备状态。文献[45]提出了一种新颖的黑盒条件下的定向模糊测试模型,其利用网络响应来推断设备的执行轨迹和代码覆盖情况,并通过IO导向的距离测量估计测试用例与潜在敏感代码的距离,进而智能指导测试用例的变异,有效提高发现漏洞的效率和准确性。文献[46]针对Android智能电视(SmartTV)的安全性问题提出了一种新颖的动态模糊测试方法,利用实时日志推断输入规范并收集反馈信息,设计了一种外部观察者机制监测电视相关的物理症状(如视觉和听觉变化)以检测潜在的物理异常。

为了提升测试效率,黑盒模糊测试更加关注种子的生成质量,变异策略对种子格式更加敏感。Snipuzz^[47]使用了一种新颖的基于消息片段的变异方法。这里的“片段”是指种子中结构上紧密相连、语义上相对完整的字节序列。它根据响应变化来自动推断消息片段,然后基于片段进行变异来提升种子的生成质量。VFuzz^[48]是一个针对Z-Wave智能家居设备设计的协议感知黑盒模糊测试框架,旨在快速检测这些设备的漏洞。通过分析目标设备能力和加密支持,指导种子选择,并结合领域知识针对性地设计算法对特定的Z-Wave帧字段变异,以确保生成的测试用例的有效性。SRFuzzer^[49]及其后续ESRFuzzer^[50]针对家用路由器进行黑盒测试,以键值对的格式生成种子数据,并按照“配置→读取”顺序对种子数据进行排序以持续有效地生成测试用例,通过电源管理自动恢复测试环境,同时设计了多种变异算子以触发多种类型的漏洞。

黑盒模糊测试的目标以网络接口为主,尤其是交互式网络协议接口。ShadowBroker^[51]对物联网设备的MQTT协议接口进行模糊测试。MQTT协议被广泛应用于物联网,用于连接用户

和设备。以往的安全研究主要集中于云端的测试，未关注设备侧接口的测试，为此，Shadow-Broker提出了中间人架构，解决了设备侧MQTT测试的难题。SioTFuzzer^[52]提出了一种物联网设备Web接口状态消息生成机制，利用Web前端推送状态检索消息的依赖性，进而设计了状态敏感的变异策略。IoTScope^[53]旨在发现物联网设备中嵌入式Web应用程序的隐藏接口。隐藏接口通常没有采用适当的保护措施，因此容易暴露与安全相关的关键操作和敏感信息给未授权用户，导致潜在的安全风险。IoTScope通过固件分析构建探测请求与设备交互，并通过差异分析过滤掉无关请求及接口，进而识别出隐藏接口。

物联网设备通常接入互联网，与云端相连，提供移动App来实现远程管理，可以间接实现与设备的远程交互。IoTFuzzer^[54]和Diane^[55]利用这个特点使用移动App来生成和发送测试数据，这样能够明显提升种子的变异质量，同时避免了对设备的逆向分析。文献[56]对物联网设备、云端与用户之间的远程绑定过程进行了系统的安全分析，提出了状态机模型以描述其生命周期，并评估了真实世界的远程绑定解决方案，揭示了认证和授权设计中的多项安全缺陷（如静态标识符的使用和薄弱的访问控制），这些缺陷可能导致敏感数据泄露、拒绝服务攻击和设备劫持等严重后果。

大语言模型的应用场景在快速发展，其在程序分析领域的应用初露端倪。LLMIF^[57]探索使用LLM增强物联网设备的模糊测试。针对现有物联网设备网络协议模糊测试中存在的问题（如复杂的消息格式、未解决的消息依赖关系以及缺乏测试用例评估），LLMIF利用协议规范中丰富的消息描述指导模糊测试过程，将LLM与规范内容相结合，使其能够执行协议信息提取和设备响应推理等任务。LLMIF模糊测试算法将增强后的LLM纳入ZigBee协议的模糊测试中，通过提取的消息格式、有趣的字段值、消息头结构和消息依赖关系，生成测试用例，并评估测试用例的质量。

表1总结了物联网设备固件黑盒模糊测试技术研究现状，直观地展示了上述黑盒模糊测试技术的主要特征。

3.2 分析评估

这些黑盒模糊测试方案在不了解测试目标内部结构和实现细节的情况下，通过向设备开放的网络端口发送畸形数据报文尝试触发程序异常，具有以下特点。

1) 测试目标的选择。黑盒模糊测试倾向选择交互式网络接口作为测试目标，其中相当一部分工作是以Web网络服务作为测试对象。在黑盒条件下，交互式网络接口能够泄露相对丰富的信息，这些信息有助于推断设备状态，捕获程序异常。

2) 信息深度利用。在黑盒测试中，由于缺乏对内部状态的直接访问，测试者会利用设备响应和日志等外部信息来推断设备的状态，进而构建反馈引导机制以降低黑盒测试的盲目性。

3) 变异策略设计。为了提高测试效率，设计高效的变异策略来生成高质量的测试用例（种子）。这些策略会考虑输入数据的格式，尽量避免破坏数据的语法结构，以提高变异的有效性。

4) 字典设计：根据不同漏洞类型精心设计测试字典，以增加触发特定漏洞的概率。

5) 异常捕获。由于物联网设备固件存在多种类型的漏洞，每种漏洞的异常表现不同。在黑盒测试中通常会使用带外（out-of-band）方式，根据漏洞类型设计针对性的异常捕获策略和机制。

综上所述，针对物联网设备的黑盒模糊测试需要综合考虑网络接口的协议规则，设计出能够提升种子生成质量的格式敏感变异策略，以减少因缺少反馈而带来的测试不确定性。同时，通过深入分析设备响应和日志等外部信息，减少测试的盲目性，并根据漏洞类型设计有效的异常捕获机制。

4 灰盒模糊测试技术

4.1 研究进展

灰盒模糊测试依赖实时的程序执行状态反馈（如代码覆盖率信息），基于遗传算法框架，利用反馈信息作为进化的导向信号，引导测试不断迭代。

固件重托管技术为获取代码覆盖率等程序执行状态信息提供了环境支撑，因此许多灰盒模糊测试框架建立在固件重托管环境基础之上。传统

表 1 物联网设备固件黑盒模糊测试技术研究现状
Table 1 Research status of black-box fuzzing for IoT device firmware

方案	测试对象	变异策略	异常监测机制	漏洞类型	特征
Labrador ^[45]	Web	Response Guided	Net Monitor	命令注入 内存破坏 跨站脚本攻击	利用检查响应中的字符串来推断执行的代码块；通过评估测试用例与目标敏感代码之间的距离来指导变异；利用上述代码覆盖率和距离反馈构建定向测试模型
SmartTVFuzzer ^[46]	SmartTV	Log Guided	Audio&Video Monitor	内存破坏 音/视频异常	利用运行日志信息推断输入规范，生成测试用例；通过重定向音频/视频输出并与正常样例进行比较，检测异常
Snipuzz ^[47]	Web	Snippet-based Random-based	Net Monitor	内存破坏	通过分析设备的响应消息，推断消息片段(snippets)，基于消息片段进行变异以提升变异质量
VFuzz ^[48]	Z-Wave	Structure-aware	Net Monitor	内存破坏	使用领域知识设计针对 Z-Wave 的变异算法，生成符合结构语义的数据包
SRFuzzer ^[49] ESRFuzzer ^[50]	Web	Dictionary-based	Net Monitor	命令注入 内存破坏	利用键值对数据模型和配置读取通信模型描述测试用例的数据格式和时序关系，指导测试用例生成
ShadowBroker ^[51]	MQTT	Dictionary-based	Net Monitor	命令注入 内存破坏	通过影子代理服务器监控网络连接，转发测试用例，并避免向真实代理发送可能有损的测试数据
SloTFuzzer ^[52]	Web	Dictionary-based AST-based	Emulator Stub	命令注入 内存破坏	使用 Web 前端分析和状态分析检索消息依赖性以指导种子生成
IoTScope ^[53]	Web	Hidden-interface Guided	Response Monitor	未认证隐藏接口	应用静态固件分析提取路径名和文件名，组合生成探测请求，通过差分测试发现未认证的隐藏接口
IoTFuzzer ^[54]	Mobile App	Structure-aware	Net Monitor	内存破坏	使用移动 App 中的程序逻辑生成测试用例，无须对协议规范进行逆向工程或显式恢复
Diane ^[55]	Mobile App	Structure-aware	Net Monitor	内存破坏	利用移动 App 控制流中位于验证逻辑和数据编码之间的函数生成满足复杂约束条件的输入
IoTBinding ^[56]	Cloud&App	Structure-aware	Net Monitor	授权问题 信息泄露	建立状态机模型用来系统性地分析物联网设备、云端和用户间远程绑定的安全性，揭示在认证和授权设计中存在的安全隐患
LLMIF ^[57]	ZigBee	LLM Guided	Response Monitor	内存破坏	将 LLM 与规范内容相结合，用来驱动 LLM 提取协议信息以指导种子生成，并根据设备响应推理程序状态。基于推理结果添加新消息丰富变异过程，有效探索协议状态空间

的全系统模拟方法效率低，而用户模式模拟虽高效但存在兼容性问题。Firm-AFL^[58]为物联网设备固件提供了高兼容性和高吞吐率的灰盒模糊测试框架，该框架通过增强进程仿真技术融合了系统模式和用户模式两者的优势。在用户模式下运行目标程序可以提高效率，仅在必要时切换到系统模式以确保程序正确执行，从而在不需要修改固件程序的情况下实现对物联网设备固件的高效模糊测试。EQUAFL^[59]设计了增强的用户模式模拟，能够自动设置执行环境使得应用程序在用户模式下运行，同时重放系统调用以满足嵌入式应

用程序的需求。在全系统模拟下观察程序行为，识别关键点，并在用户模式下重放所需环境，从而提高模糊测试的效率和兼容性。SFuzz^[60]面向实时操作系统(RTOS)开展灰盒模糊测试，利用RTOS将复杂二进制文件划分为多个独立但功能单一的任务，使用前向切片技术识别和构建与用户输入相关的代码片段；创建执行树表征不同的代码执行路径，在指令级模拟环境中运行代码片段，在此基础上实现代码覆盖率引导的模糊测试。FirmHunter^[61]面向基于Linux的物联网设备固件，在模拟的仿真环境中收集执行信息来驱动

模糊测试，利用消息状态队列解决测试用例中的依赖问题，加快漏洞发现的速度。FIRM-COV^[62]通过优化的进程仿真技术，对物联网设备固件程序进行代码覆盖率引导的灰盒模糊测试。利用字典生成算法提供结构化输入值并通过协同调度来实现高覆盖率和吞吐量。EM-Fuzz^[63]将实时内存检查与模糊测试紧密结合来检测嵌入式设备固件中的漏洞。利用内存插桩收集代码覆盖率以引导模糊测试过程，同时记录内存敏感操作对高风险区域进行的高强度测试。FirmFuzz^[64]检测基于Linux的物联网设备固件中的漏洞，利用静态分析辅助生成测试用例，通过在模拟仿真环境中植入监控桩子捕捉程序异常。

有些工作利用设备自身提供的程序追踪软硬件机制来构建灰盒模糊测试框架。NDFuzz^[65]针对虚拟化网络设备构建了非侵入式的代码覆盖率引导模糊测试框架，通过绕过虚拟化网络设备的完整性保护机制，利用开发的一种非侵入式的页面全局目录推断技术，能够区分虚拟网络设备进程和虚拟管理程序。Li等^[66]利用硬件调试机制构建了非侵入式的反馈驱动模糊测试工具 μ AFL，用来测试微控制器固件中的驱动代码。研究人员通过硬件在环（hardware-in-the-loop, HITL）方法构建了一个与AFL兼容的模糊测试框架，利用ARM ETM^[44]硬件调试机制透明地收集指令跟踪信息。GDBFuzz^[67]利用微控制器内置的调试接口和GNU调试器（GDB）来设置硬件断点，从而在不修改或重托管固件的情况下提取代码覆盖灰盒模糊测试。

LLM能够深入理解协议和规范，通过嵌入模糊测试关键环节，在指导测试用例生成、变异策略等方面表现出了显著的效果。预训练的LLM学习了数百万页人类可读的协议和规范，利用它们能够提取出可用于协议模糊测试的机器可读信息。ChatAFL^[68]通过与LLM的系统交互来构建协议的消息类型语法，增加种子消息序列的多样性，并在代码覆盖率停滞时预测生成新的消息以探索协议的新状态。文献^[69]针对物联网设备固件中常见的BusyBox程序进行测试，利用LLM生成符合目标格式的初始种子，为模糊测试提供优异的测试基础，然后重用在旧版本

BusyBox中获得的崩溃数据，以此来快速识别新版本中的潜在漏洞。

表2总结了物联网设备固件灰盒模糊测试技术研究现状，直观地展示了上述灰盒模糊测试技术的主要特征。

4.2 分析评估

这些灰盒模糊测试技术利用程序内部状态反馈引导变异策略，通过不断迭代探索更多未触发的代码区域，以更有效地发现固件中的缺陷，具有以下特点。

1) 状态反馈。灰盒模糊测试的显著特点是利用程序内部状态的反馈，尤其是代码覆盖率来引导变异过程。这与完全依赖外部输入和输出的黑盒测试形成对比。

2) 状态获取方式。大多数灰盒测试技术依赖模拟仿真环境，直接从仿真的运行态中提取所需的内部程序执行信息。另外一部分灰盒测试技术则利用设备的原生软硬件提供的调试和监控手段，提取运行态程序执行信息。

3) 测试目标选择。这些测试技术通常选择应用层功能/业务程序或直接以网络服务端口作为测试目标。这一点与黑盒模糊测试相似，主要是因为物联网设备固件的网络特性（开放的网络端口有限）直接影响了攻击面的选取。

4) 变异策略。大多数灰盒测试技术的变异策略受到AFL的影响，许多技术直接使用AFL的变异算子。然而，物联网设备的网络通信协议具有丰富的语法结构，AFL的随机变异策略并没有考虑到种子的格式要求，导致许多变异是无效的。此外，AFL的变异策略主要是比特级操作，这种粒度与物联网设备网络端口通信协议（如Web服务使用的HTTP）的文本格式不相符，比特级变异操作很难取得有效的测试效果。

总体来讲，灰盒模糊测试利用程序内部状态反馈引导外部输入变异，提高了测试的有效性。然而针对物联网设备的特定网络通信协议，需要对变异策略进行调整，以确保变异操作能够符合协议的语法结构，从而提高测试的有效性和效率。可以根据物联网设备的特定网络通信协议进行精确建模，采用机器学习技术提升模糊测试变

表2 物联网设备固件灰盒模糊测试技术研究现状
Table 2 Research status of gray-box fuzzing for IoT device firmware

方案	PUT	反馈机制	测试环境	变异策略	漏洞类型	特征
Firm-AFL ^[58]	Application Binary	Code Coverage	Augmented Process Emulation	AFL-Mutation	内存破坏	结合用户与系统模式增强进程仿真, 兼顾高兼容性和高吞吐量
EQUAFL ^[59]	Application Binary	Code Coverage	Enhanced User-mode Emulation	AFL-Mutation	内存破坏	通过系统环境重放增强用户模式仿真, 提升测试效率和兼容性
SFuzz ^[60]	RTOS	Code Coverage	Unicorn	AFL-Mutation	内存破坏	结合切片与微代码模糊测试, 面向 RTOS 的灰盒模糊测试框架
FirmHunter ^[61]	Web	System-Calls, Code Coverage	System Emulation	Dictionary-based	命令注入 内存破坏 跨站脚本攻击	从模拟环境中收集状态依赖信息指导模糊测试
FIRM-COV ^[62]	Application Binary	Code Coverage	Optimized Process Emulation	Dictionary-based AFL-Mutation	内存破坏	优化进程仿真技术以提升兼容性和吞吐量, 利用字典优化变异策略
EM-Fuzz ^[63]	Application Binary	Memory Instrumentation	User Emulation	Random-Mutation Restricted-Mutation	内存相关异常, 如缓冲区溢出、内存泄露	利用内存插桩收集代码覆盖率并捕获程序异常
FirmFuzz ^[64]	Web	Firmware Inspection	System Emulation	Dictionary-based	命令注入 内存破坏 跨站脚本攻击	静态程序分析辅助生成模糊测试用例, 固件桩程序捕获程序异常
NDFuzz ^[65]	Virtual Network Device	Code Coverage	Network Function Virtualization	Mutiny-Mutation	内存破坏	利用非侵入式的页面全局目录推断技术提取虚拟化网络程序的代码覆盖率
μ AFL ^[66]	Application Binary	Code Coverage	ARM ETM	AFL-Mutation	内存破坏	基于硬件调试接口的非侵入式代码覆盖率反馈
GDBFuzz ^[67]	Application Binary	Code Coverage	GDB	LibFuzzer-Mutation	内存破坏	结合控制流图利用 GDB 调试接口设置硬件断点来推断代码覆盖率
ChatAFL ^[68]	Network Protocol	LLM Guided	Protocol Implementation	AFL-Mutation	内存破坏	利用 LLM 对协议规范的理解能力来引导种子变异并提升代码覆盖率
BusyBoxFuzz ^[69]	Busybox	LLM Guided Crash Reuse	User Mode Emulation	AFL++-Mutation	内存破坏	通过 LLM 生成符合规范的高质量种子, 设计崩溃重用技术实现对跨不同版本的软件组件进行漏洞检测

异策略的智能性, 如利用 LLM 提升对测试用例语法结构的感知, 进而提高模糊测试的效率。

5 静态程序分析技术

5.1 研究进展

静态程序分析依赖条件少, 仅需要物联网设备固件程序即可展开分析, 方便大规模部署和测试。

污点分析是固件静态程序分析中常用的分析方法。HermeScan^[70]针对基于 Linux 的物联网设

备固件, 利用可达定义分析 (reaching definition analysis, RDA) 执行高效且有效的污点分析, 通过增强的控制流图恢复、精确的源点识别和上下文敏感的数据流分析来减少误报和漏报, 并通过按需跟踪和路径合并策略提升效率, 显著降低了时间成本。SaTC^[71]利用嵌入式设备网络前端和后端之间共享的输入关键字来精确识别后端二进制文件中的输入入口点, 然后针对性地开展数据流分析, 进而检测不信任的用户输入可能导致的危险程序路径。Karonte^[72]识别和建模嵌入式设

备固件中多个可执行文件之间的通信,利用污点分析建立和跟踪跨二进制的`数据流`,以检测不安全的交互并识别漏洞。`DTaint`^[73]在嵌入式设备固件程序中检测污点类型漏洞,通过自底向上的方式生成数据依赖关系,识别指针别名、跨进程数据流和数据结构的相似性,以减少固件复杂性对分析的影响。`EmTaint`^[74]针对固件程序静态污点分析存在的高误报率、过度近似的指针范围和无法处理间接调用等问题,采用精确的需求驱动的流、上下文和字段敏感的别名分析方法,并设计了一种新颖的间接调用解析方案,结合数据无害化规则检查来发现污点类型漏洞。

物联网设备固件程序中的调试信息和符号表通常被剥离,使得从污点源(如`recv`函数)到汇聚点(如`system`函数)的完整数据流路径分析变得非常困难,尤其是间接调用导致控制流信息缺失增加了数据追踪的难度。文献[75]提出了一种自动推断`SOHO`(`sall office home office`)路由器固件中污点源的方法,通过识别具有键值特征的函数来自动推断污点源,从而绕过间接调用并进行静态污点分析以检测漏洞。`FITS`^[76]提出了“中间污点源”概念以及自动推断的方法,中间污点源是处理用户输入并返回部分输入以供其他函数使用的自定义函数,从而显著缩短分析的数据流路径。具体而言,通过静态分析提取函数的结构和流特征,结合行为聚类 and 锚点函数评分来对自定义函数进行排名,以识别中间污点源。

物联网设备漏洞类型众多,对于不同的安全问题需要针对性设计程序分析方法。`FirmAl-ice`^[77]是一种认证绕过漏洞模型,该模型基于符号执行引擎和程序切片技术,检测是否存在路径能够绕过认证过程而直接执行特权操作。`MQT-Tactic`^[78]用于系统地分析和验证开源`MQTT`代理实现中的逻辑漏洞,特别是与授权相关的属性。现有`MQTT`代理实现频繁出现安全缺陷,这些缺陷可能被放大并影响许多供应商的物联网部署。`MQT-Tactic`综合使用静态代码分析、形式化建模和自动化模型检查来识别安全问题。首先定义`MQTT`代理的状态机模型,然后从特定代理的源代码中提取与模型相关的信息,并自动将模型信

息翻译成具体的代理模型,最后使用模型检查器验证模型是否违反预定义的安全属性。

物联网设备在与外界通信过程中通常使用加密信道,在固件程序中可能引入了密码组件。由于物联网设备的多样性和资源限制,开发者在实现安全功能时可能会错误地使用或封装密码学函数,进而出现安全漏洞。`CryptoREX`^[79]能够大规模地分析物联网设备固件中的密码学误用问题,通过将二进制代码提升到统一的中间表示(`intermediate representation, IR`),并执行静态污点分析来识别设备中的密码学误用。`CrypTody`^[80]将固件中的密码学误用问题抽象为数据流推理任务,构建数据流图和误用推理规则,基于逻辑推理分析和检测密码学API的不安全使用情况。

深度学习能够学习漏洞的潜在特征,一些研究者尝试通过构建深度神经网络模型实现对固件程序的自动化扫描和检测。文献[81]首先构建了固件程序的属性控制流图,其中不仅包含二进制代码信息,还整合了环境信息,使得神经网络模型能够深入分析二进制代码中的潜在漏洞;然后,通过图嵌入算法将此图转换为便于分析的低维特征矩阵,以处理跨架构和跨系统的二进制文件,增强模型的可扩展性;最后,利用神经网络对这些特征进行分析,以识别和预测漏洞的位置及类型。文献[82]将固件数据转换为图像形式,然后利用计算机视觉技术特别是卷积神经网络(`convolutional neural network, CNN`)来分析这些图像,并识别出表示安全漏洞的特征。作者手动收集并验证了1 450个固件样本,并将它们转换成图像数据,进一步利用局部二值模式(`local binary pattern, LBP`)、梯度直方图(`histogram of oriented gradient, HOG`)和Gabor滤波器等统计滤波算法增强图像特征,这些增强后的图像被用作输入,训练和测试了多种CNN模型,以评估它们在识别易受攻击和良性固件样本方面的准确性。

一些研究者利用LLM检测固件程序漏洞。在分析大型且复杂的代码库时,静态分析存在精度不足的问题。`LLIFT`^[83]将LLM与静态分析相结合,利用LLM对代码的理解能力来指导路径分析,特别是聚集与漏洞触发相关的约束,大幅

减少了探索路径,提高了对复杂漏洞的分析精度;通过精心设计的提示策略(包括渐进式提示、任务分解和自我验证)与 LLM 交互,提高对复杂漏洞分析的精确度,在 Linux 内核中的实证评估中成功识别出之前未被发现的未初始化使用(UBI)漏洞。Lara^[84]利用代码和数据之间的语义关系来增强对设备固件程序的污点分析。传统污点分析难以全面和准确地识别固件程序中来自用户的输入源。用户输入可以被分类为 URI 或键(数据),通过识别它们在代码中的处理方式和关系,能够更加系统和全面地识别污点源。LARA 将基于模式的静态分析与 LLM 辅助分析相结合,旨在模拟人类专家利用经验和代码语义进行分析,提高漏洞检测的准确性和效率。

由于广泛的代码重用和第三方 SDK 的大范围使用,物联网设备固件中存在大量的同源二进制文件,这些文件通常由不同编译配置的相同或相似源代码编译而成。一旦在某一固件中发现漏洞,共享相似代码片段的其他固件就会面临较高的安全风险。CVSkSA^[85]结合了支持向量机(support vector machine, SVM)和 K 最近邻(K-nearest neighbor, KNN)算法,以提高在不同架构固件中识别已知安全漏洞的准确性和效率。其首先利用 KNN 算法快速筛选出可疑函数,然后通过 SVM 进一步精细化筛选,以提高预筛选阶段的准确性。BINARM^[86]被设计用来检测基于 ARM 架构的智能电网设备固件中存在的已知漏洞。该方案中提出了从粗到细的多阶段函数匹配方法,即基于异构特征过滤不相似函数,再基于执行路径进一步过滤,最后通过模糊图匹配识别候选函数,通过这种方式减少函数匹配的计算成本。

表 3 总结了物联网设备固件静态程序分析技术研究现状,直观地展示了上述物联网设备固件静态程序分析技术的主要特征。

5.2 分析评估

静态程序分析是一种在不执行程序代码的情况下通过分析代码发现潜在安全漏洞的技术。在物联网设备固件的安全分析中,静态程序分析具有以下特点。

1) 广泛应用污点分析技术。固件静态程序分

析方案广泛采用污点分析技术来挖掘漏洞。污点分析通过追踪数据流识别可能被恶意利用的数据路径。为了提升污点分析的效果,通常结合物联网设备固件的特点优化分析算法,从而提高分析效率和准确率。

2) 可扩展性强,支持跨平台分析。静态程序分析具有强大的可扩展性,支持跨平台分析。通过将固件代码提升到统一的 IR 层进行分析,能够有效屏蔽不同架构指令的差异性,使得分析方法更加通用和灵活。

3) 控制流恢复。静态程序分析由于缺少动态执行过程中的信息,特别是在处理间接调用等复杂情况时,控制流的完整恢复是一大挑战。为了提升静态程序分析的精度,一些方案结合启发式规则来识别和恢复间接调用,增强了静态程序分析在复杂控制流场景下的分析能力。

4) 针对性分析策略。物联网设备固件存在的漏洞类型多种多样,不同类型漏洞需要设计不同的分析策略。对于缓冲区溢出、命令注入等类型漏洞,可以清楚地定义污点汇聚点,适合采用污点分析策略。对于授权相关等难以明确污点汇聚点的逻辑类型漏洞,则需要进行针对性的建模和分析,以确保分析的准确性和有效性。

综上所述,静态程序分析具有良好的跨平台特性,方便对物联网设备固件开展大规模测试。现有静态程序分析技术通过污点分析技术、控制流恢复策略以及启发式规则引导,优化分析的效率和准确率。但静态程序分析不可避免受到准确性、可扩展性和效率的制约,需要面向场景采用针对性的优化策略,通过对具体问题建模,引入专家知识构建有效的分析规则,借助 LLM 来提升静态程序分析的准确性和效率。

6 固件重托管技术

6.1 研究进展

固件重托管将固件从原本运行的设备移植到虚拟运行环境中运行,为物联网设备固件的安全性分析提供了一个有利的基础环境。

固件重托管的主要目标是可规模化,即用相对低的代价对大量不同种类固件进行安全测试。基于 Linux 的固件是固件重托管的重要目标。Firmadyne^[87]是一个面向基于 Linux 的固件自动化

表3 物联网设备固件静态程序分析技术研究现状

Table 3 Research status of static program analysis for IoT device firmware

方案	技术策略	可扩展性	漏洞类型	特征
HermesScan ^[70]	利用RDA执行高效且有效的污点分析	支持跨架构分析	污点类型	使用增强的控制流图恢复、精确的源点识别、上下文敏感的数据流分析策略提升准确率，使用按需跟踪和路径合并策略提升效率
SaTC ^[71]	粗粒度污点分析，路径合并以加速分析过程	支持跨架构分析	污点类型	利用前端和后端之间共享的输入关键字来定位后端二进制文件中的用户输入入口点
Karonte ^[72]	使用路径探索优化策略加速污点分析，跨程序追踪数据流	支持跨架构分析	污点类型	建立和跟踪固件中多个二进制文件之间的交互来检测不安全的交互
DTaint ^[73]	利用符号执行生成数据依赖	支持跨架构分析	污点类型	识别指针别名、跨进程数据流和数据结构相似性，提高数据流分析的准确性
EmTaint ^[74]	基于间接调用解析方案并结合数据无害化规则检查，提升静态污点分析效果	支持跨架构分析	污点类型	精确的需求驱动的流、上下文和字段敏感的别名分析策略
TaintSources ^[75]	通过识别具有键值特征的函数实现自动推断和识别污点源	支持跨架构分析	污点类型	能够缓解间接调用问题，通过绕过间接调用进行静态污点分析来提升静态分析效果
FITS ^[76]	自动推断中间污点源，能够显著缩短分析的数据流路径	支持跨架构分析	污点类型	提取函数的结构和流特征，利用行为聚类 and 锚点函数评分识别可能的中间污点源
FirmAlice ^[77]	基于符号执行引擎和程序切片技术，检测符合认证过的安全策略违规路径	支持跨架构分析	认证绕过	创建针对攻击者确定执行特权操作所需输入的认证绕过漏洞模型
MQTTactic ^[78]	静态代码分析、形式化建模和自动化模型检查	支持跨架构分析	授权相关的逻辑类型漏洞	设计状态机模型用来形式化验证MQTT代理实现，使用模型检查器来验证模型与预定义的安全属性
CryptoREX ^[79]	执行静态污点分析来追踪自定义加密API的误用	支持跨架构分析	密码学误用	动态更新API列表并在污点分析过程中自动追踪函数参数
CrypTody ^[80]	将数据流作为输入，将检测模式转化为逻辑规则，基于推理方式自动推导密码学API误用情况	支持跨架构分析	密码学误用	支持单个数据流中的密码学误用检测，同时使用跨流推理策略能够识别涉及多个数据流的复杂误用情况
GraphEmbed-IoT ^[81]	构建属性控制流图，用来整合环境信息和二进制代码信息，然后利用图嵌入技术将图转换为低维特征向量，最终通过深度神经网络进行分析以识别和预测漏洞	支持跨架构分析	内存相关类型	通过整合环境信息增强漏洞检测能力，利用图嵌入技术实现对代码特征的自动学习和降维，提高了模型的训练效率和预测性能
DL-Detector ^[82]	将路由器固件转换为图像数据，然后应用计算机视觉和深度学习技术特别是CNN来分析这些图像并识别出表示安全漏洞的视觉特征	支持跨架构分析	已知漏洞	将固件安全分析问题转化为图像识别问题，使得可以利用成熟的计算机视觉技术来检测设备固件漏洞
LLIFT ^[83]	利用LLM的代码理解和生成能力来辅助静态分析	支持跨架构分析	UBI	聚集漏洞相关的约束进行，路径分析大幅减少了探索路径并提高了对复杂漏洞的分析精度
Lara ^[84]	从前端提取用户输入项（如URI和键），然后分析后端代码以识别处理这些输入的函数。通过模式匹配模拟人类专家的经验，再利用LLM帮助捕捉代码的语义信息，从而更准确地识别潜在的污点源和汇聚点	支持跨架构分析	污点类型	能够识别隐藏的用户输入处理代码，这些代码往往缺乏前端对应项，导致传统方法无法检测到。此外，将模式匹配与LLM的语义理解相结合，可以显著提升对代码语义的理解能力，从而更准确地识别关键处理函数
CVSkSA ^[85]	利用KNN算法快速筛选出与已知漏洞相似的函数，减少后续处理的数据量，然后使用SVM对KNN算法筛选后的函数进行进一步分类，以提高准确性	支持跨架构分析	已知漏洞	结合KNN的高效率和SVM的高准确性，通过平衡两者的优势，实现在跨不同架构固件中快速且准确地识别已知漏洞
BINARM ^[86]	逐步细化代码相似性匹配过程，从基于异构特征的粗略匹配到基于模糊图匹配的精细匹配，以识别潜在的漏洞函数	支持跨架构分析	已知漏洞	通过多阶段检测实现从快速排除不相似函数到精确匹配的转变，可以显著减少计算资源消耗和提高处理大型固件的效率

动态分析框架,成功实现了对固件的批量自动化提取、模拟和分析。它使用全系统仿真技术,用经过插桩处理的定制内核替换原始内核,结合固件的文件系统,可以对固件中运行的全部进程进行整体性分析。FirmAE^[88]分析了大量 Firmadyne 仿真失败案例,在其基础上通过引入简单的启发式规则进一步提升了固件仿真的成功率。Greenhouse^[89]用于重托管基于 Linux 的物联网设备固件中的单一服务或应用程序,它是自包含的进程集合,在执行期间不需要与其他任何非父进程进行通信,通过用户在用户空间模拟单一服务程序,绕过了传统全系统仿真的许多挑战,并且显著降低了执行开销。

在一些实时操作系统、单体嵌入式固件中,存在固件本身解耦困难问题,难以针对重托管任务直接修改固件本身,固件模拟依赖于外设的正确模拟,此时外设模拟是固件重托管需要解决的关键问题。Avatar^[90]提出了一种硬件在环的动态模拟分析框架,通过在真实硬件和模拟器之间协调执行,允许对固件进行复杂的动态分析;在模拟器中执行固件代码的同时将 I/O 操作转发到真实硬件上,并通过一系列技术动态优化代码和数据在两种环境之间的分布。Avatar 2^[91]在 Avatar 的基础上进行了全新设计和扩展,使其能够灵活地协调任意组件,以执行复杂的二进制分析任务;通过允许分析人员组织不同工具形成复杂结构,并在这些工具之间“移动”执行二进制代码,以解决现有动态二进制分析工具之间的互操作性问题。该框架支持自动传输设备/应用的内部状态,以及可配置地转发输入/输出和内存访问到物理外设或模拟目标。P2IM^[92]通过自动化生成外设接口模型实现了与硬件无关且可扩展的固件测试。该工作为 ARM Cortex-M 架构的外设定义了抽象模型,其中包含了 4 类寄存器和响应的访问模型、访问处理规则,即模拟器应当如何识别每种外设、如何处理每种外设的访存,将寄存器类型的推断建立在固件的寄存器访问模式上,从而无须依赖具体的外设内部设计的先决知识。μEmu^[93]通过无效性引导的知识推断来学习未知外设的正确模拟方法,如果从外设寄存器中读取结果影响到条件分支,则产生一条默认的寄存器取值,如果检测到执行状态变为无效(如进

入无限循环、非法内存访问或无效程序点),则认为之前推断的外设读取值是错误的。该工具将固件作为输入,利用符号执行技术将未知外设寄存器视为符号,并在执行过程中推导出响应外设访问的规则,这些规则随后被存储在知识库中用来指导后续的动态分析。Fuzzware^[94]通过精确的内存映射输入/输出(MMIO)建模,有效地对单体固件(指一类单一的不可分割的固件镜像)进行模糊测试。通过综合使用轻量级程序分析、重托管和模糊测试,分析固件逻辑中如何使用由硬件生成的值,并自动生成模型,专注于变异关键的输入,进而提高测试的有效性。Pretender^[95]利用机器学习自动习得外设行为并建模,记录原始硬件与固件之间的交互,使用无监督的机器学习聚类算法确定外设的边界,推断中断与 MMIO 事件的关系,训练已知模型以匹配常见的内存访问模式,并采用状态近似方法处理更复杂的外设功能,这些虚拟外设最终被用于在全系统模拟器中替代原始硬件,使固件能够在模拟环境中执行。

对于单体的物联网设备固件,也有追求将固件以类似于原生状态运行在宿主系统之上的“移植”思路,将固件中相对独立的局部替换为可以直接在宿主上运行的模块,这样也能有效解决固件重托管面临的挑战。HALucinator^[96]利用硬件抽象层(HAL)作为固件重托管和分析的基础,提出了一种高级仿真(high-level emulation, HLE)方法,应用二进制相似性原理定位固件中的 HAL 函数,并在系统仿真器中对这些函数进行近似替代,从而解耦固件与硬件,避免对硬件交互的精确建模。文献[97]将嵌入式应用程序重托管为 Linux 应用程序,能够充分利用现有的动态分析技术。SAFIREFuzz^[98]采用了一种称为近原生重托管的技术,在与目标设备指令集家族相同的高性能系统上以 Linux 进程方式运行嵌入式固件,避免了传统模拟器指令翻译带来的高开销;同时利用高级仿真和动态二进制重写技术,实现在更强大的硬件上低开销地运行二进制固件,并与模糊测试工具实时联动。

表 4 总结了物联网设备固件重托管技术研究现状,直观地展示了上述物联网设备固件重托管技术的主要特征。

表4 物联网设备固件重托管技术研究现状

Table 4 Research status of re-hosting for IoT device firmware

方案	模拟对象	仿真目标	分析策略	特征
Firma-dyne ^[87]	基于 Linux 的设备固件	成功运行固件中的 Web 服务	使用 QEMU 全系统仿真启动与固件文件系统相匹配的内核；进一步推断出网络配置并动态配置模拟环境	以可扩展的方式对固件进行批量自动化模拟和分析
FirmAE ^[88]	基于 Linux 的设备固件	成功运行固件中的 Web 服务	仲裁仿真，即通过有意的干预，解决仿真过程中的失败案例，而不是严格遵循固件的原始执行行为	自动化整个固件仿真过程，并利用容器实现并行仿真，提升效率
Green-house ^[89]	Linux 类型固件中的单一服务程序	启动单一服务程序提供网络访问功能	通过迭代循环，逐渐提高服务的执行保真度。遇到障碍时，尝试应用一系列自动化的干预措施以绕过障碍	专注于在用户空间中重托管单个固件服务，而不是整个操作系统或硬件环境。这种方法减少了仿真的复杂性和资源需求
Avatar ^[90]	固件二进制程序	通过模拟器执行固件代码，同时将 I/O 操作转发到真实的硬件设备	从模拟器转发内存访问到嵌入式设备，确保固件行为的准确性；在模拟器和真实设备之间进行任意上下文切换，优化性能和分析效率	将模拟器用于固件代码的执行，同时利用真实硬件处理 I/O 操作，实现硬件与软件协同
Avatar 2 ^[91]	通过 GDB、OpenOCD、QEMU、PANDA、ANGR，实现对嵌入式设备固件、桌面应用程序以及其他类型软件的动态二进制分析	动态多目标协调框架，促进不同动态二进制分析框架、调试器、模拟器和真实物理设备之间的互操作性	在不同的分析环境之间转移执行状态，如从 GDB 转移到 ANGR，实现状态转移和同步；通过事件驱动的插件系统，提供通用代码基础和自动执行重复性任务	支持将内存访问转发到物理设备或模拟器，以实现部分仿真；允许分析人员组织不同的工具在一个复杂的拓扑中，并在这些系统之间“移动”执行二进制代码
P2IM ^[92]	微控制器单体二进制固件	实现独立于硬件的固件测试	使用自动化技术识别和模拟固件与外设之间的接口行为	抽象建模处理器与外设之间的接口，动态处理和响应接口
μEmu ^[93]	微控制器单体二进制固件	分析和测试固件的任务代码，特别是那些与 I/O 接口相关的部分	分为知识提取阶段和动态分析阶段：在知识提取阶段，通过符号执行和无效性检查，提取必要的信息，并存储在知识库中；在动态分析阶段，利用知识库指导固件的执行，保持有效状态	使用符号执行模拟固件执行，并在遇到未知外设访问时，通过无效性检查来引导知识库的构建，从而学习如何正确响应外设访问
Fuzzware ^[94]	微控制器单体二进制固件	轻量级程序分析、重托管和模糊测试的结合	使用轻量级的程序分析技术识别固件逻辑如何使用硬件生成的值；通过局部作用域的动态符号执行，分析硬件生成值对固件逻辑的意义	通过生成访问模型，模拟硬件外设的行为
Pretender ^[95]	硬件外设	提供高仿真忠实度的虚拟外设	通过自动化地从原始硬件与固件的交互中创建外设模型，实现固件重托管	不依赖于操作系统或硬件抽象层，无须人工干预地创建外设模型，模型能够响应新的输入并执行未在记录阶段观察到的代码路径
HALucinator ^[96]	单体二进制固件	通过在模拟环境中执行固件，进行隐私性和安全性分析	通过统计比较、基本块比较和上下文匹配，识别固件中与 HAL 库函数匹配的代码，为识别出的 HAL 函数提供高级仿真实现，替代执行与原函数相同的任务	通过识别和拦截固件中的 HAL 函数，对这些函数进行近似替代，从而解耦固件和硬件
ToLinux-App ^[97]	微控制器应用程序	通过在主机环境中而非实际硬件上执行固件，实现对嵌入式系统的动态分析	不同指令集架构处理、执行语义保持与外围设备交互处理	将嵌入式应用程序转换为 Linux 应用程序，使其能够在 X86-64 架构上运行
SAFIRE-Fuzz ^[98]	ARM Cortex-M 系列微控制器的固件	在高性能的 ARM Cortex-A 系统上以接近原生的性能运行 Cortex-M 固件	在运行时动态地重写固件的基本块以适应主机架构，同时插入模糊测试所需的代码	利用 ARMv8-A 核心提供的与 AArch32 和 Thumb 指令集变体兼容的用户空间执行模式，直接在强大的处理器上执行固件代码

6.2 分析评估

固件重托管技术实现了固件与硬件的解耦合,打破了物联网设备的黑盒特性,为进一步的模糊测试和程序分析提供了基础。其具有以下特点。

1) 外设模拟挑战。外设模拟是实现固件重托管的重要方面,尤其在基于微控制器的单体嵌入式固件中,需要能够精确地模拟或替代固件所依赖的硬件外设行为。现有方案除了建立外设行为模型以模拟外设,更多着眼于如何避免固件运行对外设具体行为产生依赖,通过符号执行、二进制重写等程序分析方法替代外设行为,在不需要获取真实硬件行为知识的前提下大大提升了固件的测试覆盖率,凭借优越的适用性和可扩展性受到青睐。

2) 与模糊测试紧密联动。固件重托管与模糊测试紧密联动,通过在模拟环境中插入特定的模糊测试桩代码,提取程序状态反馈。在不少研究工作中,固件模拟的直接目的就是模糊测试。这也反过来影响了固件模拟的思路,研究者注重提升模拟方法的适用性、运行的高效性,乃至可以适当牺牲模拟的保真度。这种联动能够实现灰盒模糊测试的高效化和规模化,使固件重托管成为具有前景的物联网设备固件漏洞挖掘技术支撑。

3) 有限仿真目标。固件重托管通常集中在有限的仿真目标上,这包括对设备特定的服务和组件的局部模拟,或者针对特定类型的固件进行系统仿真。这与固件重托管目标的权衡与选择有关,其面临着仿真忠实度、模拟成功率和测试可扩展性3个维度的平衡和选择,这3个方面难以同时兼顾,需要根据实际情况和需求进行平衡。

综上所述,固件重托管将固件代码从其预期运行的硬件平台转移到一个不同的执行环境,使得在受控环境中更全面、深入地测试和分析固件,有利于发现固件中潜在的安全漏洞。然而,固件重托管面临着一系列挑战,尤其是在仿真的忠实度、模拟的成功率以及测试的可扩展性方面需要精细地平衡,通常根据仿真目标进行权衡和选择。

7 结束语

物联网设备固件自动化漏洞挖掘技术尽管取

得了显著进展,但仍存在一些挑战,如黑盒模糊测试的盲目性、灰盒模糊测试的反馈机制构建、静态程序分析的准确性及固件重托管的外设模拟问题。未来的研究需要继续优化这些技术,提高其自动化程度和准确性,同时探索新的技术手段以进一步提升对物联网设备固件的安全分析能力。展望未来,该领域将呈现出以下发展趋势。

(1) 技术融合与互补发展

物联网设备固件自动化漏洞挖掘技术正朝着多种技术相互融合、互为补充的方向迅速演进。静态程序分析技术通过提供全面的程序视角,辅助模糊测试更深入地理解程序结构和潜在风险点。同时,模糊测试通过动态路径探索,可为静态程序分析提供准确的动态信息反馈,从而增强漏洞检测的深度和准确性。此外,固件重托管技术与动态模糊测试紧密结合,以实现更高效的灰盒测试,提高代码覆盖率和漏洞挖掘效率。

(2) 自动化与智能化的深度结合

物联网设备固件自动化漏洞挖掘技术正迎来智能化的新浪潮。LLM等人工智能技术在程序理解和分析方面展现出巨大潜力,为自动化漏洞挖掘技术的发展提供了全新的思路和方法。通过结合人工智能,尤其是LLM的深度学习能力,自动化漏洞挖掘技术将能够更准确地识别和理解复杂的程序逻辑,更有效地处理间接调用和多步骤交互,从而全面提升程序分析的效果。

(3) 物联网技术的演进与漏洞挖掘技术的同步创新

物联网技术的快速发展,特别是在5G通信技术的推动下,正在重塑物联网设备的组网模式、通信协议和软硬件架构。这种演进不仅为物联网设备带来了更广阔的应用场景,也对设备固件的安全分析提出了新的挑战。自动化漏洞挖掘技术需要与物联网技术的发展同步,以满足新的安全需求。例如,漏洞挖掘框架需要具备高度的可扩展性和灵活性,这意味着框架应设计为模块化以集成新的分析工具和模块,从而适应不断扩展的物联网设备特性。

参考文献:

- [1] ZAHID M, BHARATI T S. Empowering IoT networks[J]. Artificial

- Intelligence for Blockchain and Cybersecurity Powered IoT Applications, 2025, 16:99.
- [2] SLOWIK J. Crashoverride: reassessing the 2016 ukraine electric power event as a protection-focused attack[J]. Dragos, Inc, 2019.
- [3] KOLIAS C, KAMBOURAKIS G, STAVROU A, et al. DDoS in the IoT: Mirai and other botnets[J]. Computer, 2017, 50(7):80-84.
- [4] 于颖超, 陈左宁, 甘水滔, 等. 嵌入式设备固件安全分析技术研究[J]. 计算机学报, 2021, 44(5): 859-881.
- YU Y C, CHEN Z N, GAN S T, et al. Survey on the technologies of security analysis on embedded device firmware[J]. Chinese Journal of Computers, 2021, 44(5): 859-881.
- [5] 郑尧文, 文辉, 程凯, 等. 物联网设备漏洞挖掘技术研究综述[J]. 信息安全学报, 2019, 4(5):62-75.
- 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):62-75.
- [6] FENG X T, ZHU X G, HAN Q L, et al. Detecting vulnerability on IoT device firmware: a survey[J]. IEEE/CAA Journal of Automatica Sinica, 2023, 10(1): 25-41.
- [7] WANG Z, ZHANG L, CAO C, et al. The effectiveness of large language models (ChatGPT and CodeBERT) for security-oriented code analysis[J]. Available at SSRN 4567887, 2023.
- [8] LEMIEUX C, INALA J P, LAHIRI S K, et al. Codamosa: escaping coverage plateaus in test generation with pre-trained large language models[C]// Proceedings of 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). Piscataway: IEEE Press, 2023:919-931.
- [9] KANG S, YOON J, YOO S. Large language models are few-shot testers: exploring LLM-based general bug reproduction[C]// Proceedings of 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). Piscataway:IEEE Press, 2023: 2312-2323.
- [10] YAN P, TAN S, WANG M, et al. Prompt engineering-assisted malware dynamic analysis using GPT-4[J]. Arxiv Preprint Arxiv: 2312.08317, 2023.
- [11] TAKANEN A. Fuzzing: the past, the present and the future[J]. Ac-Foundation Robert Schuman, 2009: 202-212.
- [12] CHEN C, CUI B J, MA J X, et al. A systematic review of fuzzing techniques[J]. Computers & Security, 2018, 75: 118-137.
- [13] GODEFROID P, LEVIN MY, MOLNAR D A. SAGE: whitebox fuzzing for security testing[J]. Communications of the ACM, 2012, 55(3): 40-44.
- [14] GANESH V, LEEK T, RINARD M. Taint-based directed whitebox fuzzing[C]// Proceedings of 2009 IEEE 31st International Conference on Software Engineering. Piscataway: IEEE Press, 2009: 474-484.
- [15] GUTMANN P. Fuzzing code with AFL[J]. Login USENIX Mag, 2016, 41(2):11-14.
- [16] GAN S, ZHANG C, QIN X, et al. CollAFL: path sensitive fuzzing[C]// Proceedings of 2018 IEEE Symposium on Security and Privacy (SP). Piscataway:IEEE Press, 2018: 679-696.
- [17] BOHME M, PHAM V T, ROYCHOUDHURY A. Coverage-based greybox fuzzing as markov chain[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2016: 1032-1043.
- [18] RAWAT S, JAIN V, KUMAR A, et al. VUzzer: application-aware evolutionary fuzzing[C]//Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS 2017). Virginia: the Internet Society, 2017: 1-14.
- [19] FIORALDI A, MAIER D C, ZHANG D, et al. LibAFL: a framework to build modular and reusable fuzzers[C]//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2022: 1051-1065.
- [20] SCHUMILO S, ASCHERMANN C, GAWLIK R, et al. kAFL: hardware-assisted feedback fuzzing for OS kernels[C]//Proceedings of 26th USENIX security symposium (USENIX Security 2017). Berkeley: USENIX Association, 2017: 167-182.
- [21] WANG D, ZHANG Z, ZHANG H, et al. SyzVegas: beating kernel fuzzing odds with reinforcement learning[C]//Proceedings of 30th USENIX Security Symposium (USENIX Security 2021). Berkeley: USENIX Association, 2021: 2741-2758.
- [22] JUNG J, TONG S, HU H, et al. WINNIE: fuzzing windows applications with harness synthesis and fast cloning[C]//Proceedings of the 2021 Network and Distributed System Security Symposium (NDSS 2021). Virginia: the Internet Society, 2021.
- [23] DI B L, MOGHADDAS H, PAYER M. ARMORE: pushing love back into binaries[C]// Proceedings of in the 32nd USENIX Security Symposium (USENIX Security 23). Berkeley: USENIX Association, 2023: 6311-6328.
- [24] DELIA D C, INVIDIA L, PALMARO F, et al. Evaluating dynamic binary instrumentation systems for conspicuous features and artifacts[J]. Digital Threats: Research and Practice (DTRAP), 2022, 3(2): 1-3.
- [25] LUK C K, COHN R, MUTH R, et al. Pin: building customized program analysis tools with dynamic instrumentation[C]//Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation ACM Sigplan Notices. New York: ACM Press, 2005: 190-200.
- [26] BOHME M, PHAM V T, NGUYEN M D, et al. Directed greybox fuzzing[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2017: 2329-2344.
- [27] HUANG H, GUO Y, SHI Q, et al. Beacon: directed grey-box fuzzing with provable path pruning[C]//Proceedings of in 2022 IEEE Symposium on Security and Privacy (SP). Piscataway:IEEE Press, 2022: 36-50.
- [28] CHEN H, XUE Y, LI Y, et al. Hawkeye: towards a desired directed grey-box fuzzer[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2018: 2095-2108.
- [29] LEE G, SHIM W, LEE B. Constraint-guided directed greybox fuzzing[C]//Proceedings of 30th USENIX Security Symposium (USENIX Security 21). Berkeley: USENIX Association, 2021: 3559-3576.
- [30] YUE T, WANG P F, TANG Y, et al. EcoFuzz: adaptive energy-

- saving greybox fuzzing as a variant of the adversarial multi-armed bandit[C]//Proceedings of 29th USENIX Security Symposium (USENIX Security 20). Berkeley: USENIX Association, 2020: 2307-2324.
- [31] 王蕾, 李丰, 李炼, 等. 污点分析技术的原理和实践应用[J]. 软件学报, 2017, 28(4): 860-882.
- WANG L, LI F, LI L, et al. Principle and practice of taint analysis[J]. Journal of Software, 2017, 28(4): 860-882.
- [32] TRIPP O, PISTOIA M, FINK S J, et al. TAJ: effective taint analysis of web applications[J]. ACM Sigplan Notices, 2009, 44(6): 87-97.
- [33] TRIPP O, PISTOIA M, COUSOT P, et al. Andromeda: accurate and scalable security analysis of web applications[C]//Proceedings of Fundamental Approaches to Software Engineering: 16th International Conference (FASE). Berlin: Springer, 2013: 210-225.
- [34] ARZT S, RASTHOFER S, FRITZ C, et al. FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps[J]. ACM Sigplan Notices, 2014, 49(6): 259-269.
- [35] ENCK W, GILBERT P, HAN S, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smart-phones[J]. ACM Transactions on Computer Systems (TOCS). 2014, 32(2): 1-29.
- [36] VENKATARAMANI G, DOUDALIS I, SOLIHIN Y, et al. Flexitaint: a programmable accelerator for dynamic taint propagation[C]//Proceedings of 2008 IEEE 14th International Symposium on High Performance Computer Architecture. Piscataway: IEEE Press, 2008: 173-184.
- [37] KING J C. Symbolic execution and program testing[J]. Communications of the ACM, 1976, 19(7): 385-394.
- [38] CADAR C, DUNBAR D, ENGLER D R. KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs[C]//Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI 08). Berkeley: USENIX Association, 2008: 209-224.
- [39] PASAREANU C S, VISSER W, BUSHNELL D, et al. Symbolic PathFinder: integrating symbolic execution with model checking for Java bytecode analysis[J]. Automated Software Engineering, 2013: 391-425.
- [40] TILLMANN N, DE H J. Pex-white box test generation for net[C]//Proceedings of International Conference on Tests and Proofs. Berlin: Springer, 2008: 134-153.
- [41] CHIPOUNOV V, KUZNETSOV V, CANDEA G. S2E: a platform for in-vivo multi-path analysis of software systems[J]. ACM Sigplan Notices, 2011, 46(3): 265-278.
- [42] LOCKHART D, ILBEYI B, BATTEN C. Pydgin: generating fast instruction set simulators from simple architecture descriptions with meta-tracing JIT compilers[C]//Proceedings of 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Piscataway: IEEE Press, 2015: 256-267.
- [43] GE X Y, CUI W D, JAEGER T. GRIFFIN: guarding control flows using intel processor trace[C]//Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems. 2017: 585-598.
- [44] NING Z Y, ZHANG F. Ninja: towards transparent tracing and debugging on ARM[C]//Proceedings of 26th USENIX Security Symposium (USENIX Security 17). Berkeley: USENIX Association, 2017: 33-49.
- [45] LIU H T, GAN S, ZHANG C, et al. Labrador: response guided directed fuzzing for black-box IoT devices[C]//Proceedings of 2024 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2024: 127-127.
- [46] AAFER Y, YOU W, SUN Y, et al. Android SmartTVs vulnerability discovery via log-guided fuzzing[C]//Proceedings of 30th USENIX Security Symposium (USENIX Security 21). Berkeley: USENIX Association, 2021: 2759-2776.
- [47] FENG X T, SUN R X, ZHU X G, et al. Snipuzz: black-box fuzzing of IoT firmware via message snippet inference[C]//Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2021: 337-350.
- [48] NKUBA C K, KIM S, DIETRICH S, et al. Riding the IoT wave with VFuzz: discovering security flaws in smart homes[J]. IEEE Access, 2021, 10: 1775-1789.
- [49] ZHANG Y, HUO W, JIAN K P, et al. SRFuzzer: an automatic fuzzing framework for physical SOHO router devices to discover multi-type vulnerabilities[C]//Proceedings of the 35th Annual Computer Security Applications Conference. New York: ACM Press, 2019: 544-556.
- [50] ZHANG Y, HUO W, JIAN K P, et al. ESRFuzzer: an enhanced fuzzing framework for physical SOHO router devices to discover multi-type vulnerabilities[J]. Cybersecurity, 2021, 4(1): 1-22.
- [51] XU H K, YU M, WANG Y H, et al. Trampoline over the air: breaking in IoT devices through MQTT brokers[C]//Proceedings of 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P). Piscataway: IEEE Press, 2022: 171-187.
- [52] ZHANG H W, LU K, ZHOU X, et al. SIoTFuzzer: fuzzing web interface in IoT firmware via stateful message generation[J]. Applied Sciences, 2021, 11(7): 3120.
- [53] XIE W, CHEN J Y, WANG Z H, et al. Game of hide-and-seek: exposing hidden interfaces in embedded web applications of IoT devices[C]//Proceedings of the ACM Web Conference. New York: ACM Press, 2022: 524-532.
- [54] CHEN J Y, DIAO W R, ZHAO Q C, et al. IoTFuzzer: discovering memory corruptions in IoT through App-based fuzzing[C]//Proceedings of the 2018 Network and Distributed System Security Symposium (NDSS 2018). Virginia: the Internet Society, 2018.
- [55] REDINI N, CONTINELLA A, DAS D, et al. Diane: identifying fuzzing triggers in Apps to generate under-constrained inputs for IoT devices[C]//Proceedings of 2021 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2021: 484-500.
- [56] CHEN J Y, ZUO C, DIAO W R, et al. Your IoTs are (not) mine: On the remote binding between IoT devices and users[C]//Proceedings of the 49th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway: IEEE Press, 2019: 222-233.
- [57] WANG J C, YU L, LUO X P. LLMIF: augmented large language model for fuzzing IoT devices[C]//Proceedings of 2024 IEEE Sym-

- posium on Security and Privacy (SP). Piscataway: IEEE Press, 2024: 881-896.
- [58] ZHENG Y, DAVANIAN A, YIN H, et al. FIRM-AFL: high-throughput greybox fuzzing of IoT firmware via augmented process emulation[C]// Proceedings of 28th USENIX Security Symposium (USENIX Security 19). Berkeley: USENIX Association, 2019: 1099-1114.
- [59] ZHENG Y, LI Y, ZHANG C, et al. Efficient greybox fuzzing of applications in Linux-based IoT devices via enhanced user-mode emulation[C]// Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM Press, 2022: 417-428.
- [60] CHEN L B, CAI Q P, MA Z B, et al. SFuzz: slice-based fuzzing for real-time operating systems[C]// Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2022: 485-498.
- [61] YIN Q D, ZHOU X, ZHANG H W. FirmHunter: state-aware and introspection-driven grey-box fuzzing towards iot firmware[J]. Applied Sciences, 2021, 11(19): 2076-3417.
- [62] KIM J, YU J, KIM H, et al. FIRM-COV: high-coverage greybox fuzzing for IoT firmware via optimized process emulation[J]. IEEE Access, 2021, 9: 101627-101642.
- [63] GAO J, XU Y, JIANG Y, et al. EM-Fuzz: augmented firmware fuzzing via memory checking[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(11): 3420-3432.
- [64] SRIVASTAVA P, PENG H, LI J, et al. FirmFuzz: automated IoT firmware introspection and analysis[C]// Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things. New York: ACM Press, 2019: 15-21.
- [65] ZHANG Y, ZHONG N, YOU W, et al. NDFuzz: a non-intrusive coverage-guided fuzzing framework for virtualized network devices[J]. Cybersecurity, 2022, 5(1): 21.
- [66] LI W Q, SHI J M, LI F J, et al. μ AFL: non-intrusive feedback-driven fuzzing for microcontroller firmware[C]// Proceedings of the 44th International Conference on Software Engineering. New York: ACM Press, 2022: 1-12.
- [67] EISELE M, EBERT D, HUTH C, et al. Fuzzing embedded systems using debug interfaces[C]// Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM Press, 2023: 1031-1042.
- [68] MENG R, MICHEV M, BOHME M, et al. Large language model guided protocol fuzzing[C]// Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS 2024). 2024.
- [69] ASMITA, YAROSLAR O, MICHAEL S, et al. Fuzzing busybox: leveraging LLM and crash reuse for embedded bug unearthing[C]// Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24). Berkeley: USENIX Association, 2024: 883-900.
- [70] GAO Z C, ZHANG C, LIU H T, et al. Faster and better: detecting vulnerabilities in Linux-based IoT firmware with optimized reaching definition analysis[C]// Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS). 2024.
- [71] CHEN L, WANG Y, CAI Q, et al. Sharing more and checking less: leveraging common input keywords to detect bugs in embedded systems[C]// Proceedings of 30th USENIX Security Symposium (USENIX Security 21). Berkeley: USENIX Association, 2021: 303-319.
- [72] REDINI N, MACHIRY A, WANG R Y, et al. Karonte: detecting insecure multi-binary interactions in embedded firmware[C]// Proceedings of 2020 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2020: 1544-1561.
- [73] CHENG K, LI Q, WANG L, et al. DTaint: detecting the taint-style vulnerability in embedded device firmware[C]// 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway: IEEE Press, 2018: 430-441.
- [74] CHENG K, ZHENG Y W, LIU T, et al. Detecting vulnerabilities in Linux-based embedded firmware with SSE-based on-demand alias analysis[C]// Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM Press, 2023: 360-372.
- [75] CHENG K, FANG D L, QIN C, et al. Automatic inference of taint sources to discover vulnerabilities in SOHO router firmware[C]// Proceedings of IFIP International Conference on ICT Systems Security and Privacy Protection. Berlin: Springer, 2021: 83-99.
- [76] LIU P, ZHENG Y, SUN C, et al. FITS: inferring intermediate taint sources for effective vulnerability analysis of IoT device firmware[C]// Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 2023: 138-152.
- [77] SHOSHITAISHVILI Y, WANG R, HAUSER C, et al. Firmalice: automatic detection of authentication bypass vulnerabilities in binary firmware[C]// Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS 2015). Virginia: the Internet Society, 2015.
- [78] YUAN B, SONG Z, JIA Y, et al. MQTTactic: Security analysis and verification for logic flaws in MQTT implementations[C]// Proceedings of 2024 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2023: 13.
- [79] ZHANG L, CHEN J, DIAO W, et al. CryptoREX: large-scale analysis of cryptographic misuse in IoT devices[C]// Proceedings of 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019). 2019: 151-164.
- [80] WANG J N, GUO S Q, DIAO W R, et al. CrypTody: cryptographic misuse analysis of the IoT firmware via data-flow reasoning[C]// Proceedings of 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '24). New York: ACM Press, 2024: 579-593.
- [81] CHENG Y, CUI B J, CHEN C, THAR B, TAO Q. Static vulnerability mining of IoT devices based on control flow graph construction and graph embedding network[J]. Computer Communications, 2023, 197: 267-275.
- [82] ABU-MAHFOUZ A, ALRABAE S, KHASAWNEH M, et al. A deep learning approach to discover router firmware vulnerabilities[J].

- IEEE Transactions on Industrial Informatics, 2024, 20(1): 691-702.
- [83] LI H N, YU H, ZHAI Y Z, et al. Enhancing static analysis for practical bug detection: an LLM-integrated approach[C]//Proceedings of the ACM on Programming Languages. New York: ACM Press, 2024: 474-499.
- [84] ZHAO J X, LI Y K, ZOU Y Y, et al. Leveraging semantic relations in code and data to enhance taint analysis of embedded systems[C]//Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24). Berkeley: USENIX Association, 2024: 7067-7084.
- [85] ZHAO D D, LIN H, RAN L J, et al. CVSkSA: cross-architecture vulnerability search in firmware based on KNN-SVM and attributed control flow graph[J]. Software Qual, 2019, 7(3): 1045-1068.
- [86] SHIRANI P, COLLARD L, Agba B L, et al. BINARM: scalable and efficient detection of vulnerabilities in firmware images of intelligent electronic devices[C]//Proceedings of in Detection of Intrusions and Malware, and Vulnerability Assessment: 15th International Conference. Berlin: Springer, 2018: 114-138.
- [87] CHEN D D, WOO M, BRUMLEY D, et al. Towards automated dynamic analysis for Linux-based embedded firmware[C]//Proceedings of the 2016 Network and Distributed System Security Symposium (NDSS 2016). Virginia: the Internet Society, 2016.
- [88] KIM M, KIM D, KIM E, et al. Firmac: towards large-scale emulation of iot firmware for dynamic analysis[C]//Proceedings of the 36th Annual Computer Security Applications Conference. 2020: 733-745.
- [89] TAY H J, ZENG K, VADAYATH J M, et al. Greenhouse: single-service re-hosting of Linux-based firmware binaries in user-space emulation[C]//Proceedings of 32nd USENIX Security Symposium (USENIX Security 23). Berkeley: USENIX Association, 2023: 5791-5808.
- [90] ZADDACH J, BRUNO L, FRANCILLON A, et al. Avatar: a framework to support dynamic security analysis of embedded systems' firmwares[C]//Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS). Virginia: the Internet Society, 2014: 1-16.
- [91] MUENCH M, NISI D, FRANCILLON A, et al. Avatar 2: a multi-target orchestration platform[C]//Proceedings of Workshop Binary Anal. Res. (NDSS 2018). Virginia: the Internet Society, 2018: 1-11.
- [92] FENG B, MERA A, LU L. P2IM: scalable and hardware-independent firmware testing via automatic peripheral interface modeling[C]//Proceedings of the 29th USENIX Security Symposium (USENIX Security 20). Berkeley: USENIX Association, 2020: 1237-1254.
- [93] ZHOU W, GUAN L, LIU P, et al. Automatic firmware emulation through invalidity-guided knowledge inference[C]//Proceedings of the 30th USENIX Security Symposium (USENIX Security 2021). Berkeley: USENIX Association, 2021: 2007-2024.
- [94] SCHARNOWSKI T, BARS N, SCHLOEGEL M, et al. Fuzzware: using precise MMIO modeling for effective firmware fuzzing[C]//Proceedings of the 31st USENIX Security Symposium (USENIX Security 22). Berkeley: USENIX Association, 2022: 1239-1256.
- [95] GUSTAFSON E, MUENCH M, SPENSKY C, et al. Toward the analysis of embedded firmware through automated re-hosting[C]//Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019). New York: ACM Press, 2019: 135-150.
- [96] CLEMENTS A, GUSTAFSON E, SCHARNOWSKI T, et al. HALucinator: firmware re-hosting through abstraction layer emulation[C]//Proceedings of 29th USENIX Security Symposium (USENIX Security 2020). Berkeley: USENIX Association, 2020: 1201-1218.
- [97] SRINIVASAN J, TANKSALKAR S R, AMUSUO P C, et al. Towards re-hosting embedded applications as Linux applications[C]//Proceedings of the 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S). Piscataway: IEEE Press, 2023: 94-99.
- [98] SEIDEL L, MAIER D C, MUENCH M. Forming faster firmware fuzzers[C]//Proceedings of in 32nd USENIX Security Symposium (USENIX Security 23). Berkeley: USENIX Association, 2023: 2903-2920.

[作者简介]



刘航天 (1991-), 男, 河南扶沟人, 信息工程大学博士生, 主要研究方向为物联网设备安全、大模型应用。



甘水滔 (1986-), 男, 江西宜春人, 先进计算与智能工程国家级重点实验室研究员, 主要研究方向为网络安全、人工智能应用。



张超 (1986-), 男, 湖北黄冈人, 清华大学副教授、博士生导师, 主要研究方向为软件与系统安全、人工智能安全。



张红旗 (1962-), 男, 河北遵化人, 信息工程大学教授、博士生导师, 主要研究方向为网络安全、移动目标防御、等级保护和信息安全管理。



孙文厚（1998-），男，山东日照人，清华大学硕士生，主要研究方向为嵌入式固件仿真。



赵敏（1992-），女，陕西咸阳人，先进计算与智能工程国家级重点实验室助理工程师，主要研究方向为信息安全管理。



高子聪（1996-），男，江苏张家港人，信息工程大学博士生，主要研究方向为物联网设备漏洞挖掘。



白雪（1984-），女，吉林白山人，信息工程大学硕士生，主要研究方向为大数据安全管理。