嵌入式设备固件安全漏洞静态分析与防护机制研究

章丰明
浙江大华技术股份有限公司　浙江　杭州　310053

摘　要：针对嵌入式固件因架构异构、第三方库复用及资源受限导致的高漏洞风险，本文开展静态分析与防护机制研究。采用抽象解释、程序切片与污点分析构建二进制静态检测模型，优化跨架构中间表示转换，解决误报率高与路径覆盖不足问题。提出“编译期加固-部署期校验-运行期监控”三级防护体系，结合漏洞模式库实现自动化检测与修复，为物联网设备固件安全提供技术支撑。

关键词：嵌入式设备固件；安全漏洞；静态分析；防护机制

引言：物联网普及使嵌入式设备成为网络攻击核心目标，固件漏洞引发的安全事件频发，严重威胁工控、智能家居等关键领域。静态分析因无需运行固件、可扩展性强，成为固件漏洞检测的核心技术，但仍面临跨架构适配难、误报率高及与防护机制脱节等挑战。本文聚焦静态分析技术优化与防护体系构建，明确漏洞检测关键流程，设计分层防护机制，旨在弥补现有方法不足，提升嵌入式固件全生命周期安全保障能力。

1 嵌入式设备固件安全漏洞相关基础理论

1.1 嵌入式设备固件基础

（1）嵌入式固件的定义、结构与特点：嵌入式固件是嵌入在嵌入式设备中的底层软件，是设备硬件与应用程序的桥梁，决定设备核心功能的实现。其结构主要包括三部分：Bootloader负责设备启动初始化，引导内核运行；内核是固件核心，管理设备资源与程序执行；文件系统用于存储固件代码、配置文件等数据。固件核心特点表现为资源受限，受设备硬件性能限制，存储、算力有限；硬件耦合度高，与特定硬件平台深度绑定，可移植性差；生命周期长，部分设备部署后长期不更新，易积累安全隐患。（2）嵌入式固件的开发与部署流程：固件开发部署涵盖全生命周期，从代码编写开始，开发人员基于硬件平台编写底层驱动与应用代码；随后进行编译构建，将源代码转化为设备可执行的二进制文件；完成后通过烧录工具将固件写入设备硬件；后期通过OTA更新实现固件升级。各环节均存在安全风险，如代码编写未规范、编译未开启安全加固、烧录过程未加密、OTA更新未校验签名等，均可能引入安全漏洞。

1.2 嵌入式固件安全漏洞分类与成因

（1）漏洞分类：基于攻击方式与影响范围，固件安全漏洞主要分为五类：认证漏洞，如弱密码、无身份验证；授权漏洞，如权限分配不当，普通用户可访问管理员权限；输入验证漏洞，如未过滤恶意输入，易引发注入攻击；内存管理漏洞，如缓冲区溢出、内存泄漏；通信接口漏洞，如串口、网络接口未加密，数据易被窃取篡改。（2）漏洞成因：从三个层面分析，开发层面主要是编程错误、使用strcpy等不安全函数，未进行安全编码；设计层面存在权限设计缺陷、通信协议未考虑安全因素；部署层面多为设备配置不当、固件更新不及时，未修复已知漏洞，导致漏洞长期暴露[1]。

1.3 固件安全漏洞静态分析基础

（1）静态分析的定义与核心原理：静态分析是不执行固件程序的漏洞检测方法，通过解析源代码或二进制文件的结构、逻辑流程，挖掘潜在安全漏洞，无需运行设备即可完成检测，降低检测成本与风险。（2）静态分析的核心要素：关键技术包括代码表征（提取代码特征用于分析）、漏洞规则匹配（对比已知漏洞特征）、程序切片（定位漏洞相关代码片段）；评价指标主要有误报率（正常代码误判为漏洞）和漏报率（未检测出实际漏洞），直接影响分析效果。（3）常见静态分析工具介绍：Flawfinder适用于源代码漏洞检测，轻量高效但对二进制文件支持不足；SonarQube可集成到开发流程，实时检测代码漏洞，却对嵌入式专用代码兼容性较差；Binwalk专注于二进制固件分析，可提取固件内容、检测隐藏漏洞，但需配合其他工具完成深度检测。

2 嵌入式设备固件安全漏洞静态分析方法研究

2.1 固件解析与预处理技术

（1）固件镜像提取与解构：固件镜像提取与解构是静态分析的前提，核心依托Binwalk工具实现固件镜像的深度解析，通过其内置的签名数据库，自动识别镜像中的文件系统格式、内核版本、驱动程序等关键信息，完成固件的解包与解构。同时，针对不同厂商固件的加密、压缩特性，辅助使用dd、sasquatch等工具进行解密、解压

处理，进一步提取固件底层代码与配置文件，有效解耦硬件与软件之间的依赖关系，消除硬件平台差异对后续分析的影响，为后续漏洞检测奠定基础[2]。（2）预处理优化：考虑到嵌入式固件二进制文件存在代码碎片化、格式不统一、冗余信息较多等特殊性，需进行预处理优化以提升分析效率。首先对二进制文件进行代码标准化处理，通过指令重排、格式归一化，将不同编译环境生成的代码统一为标准化格式；其次剔除固件中的冗余信息，包括无效代码、注释、重复指令等，减少分析数据量；同时进行符号恢复，通过逆向工程技术还原变量名、函数名等关键符号，提升代码可读性，为后续漏洞检测提供清晰的分析基础。

2.2 基于代码属性图的静态分析方法

（1）代码属性图构建：代码属性图是实现精准漏洞检测的核心载体，其构建过程需整合程序控制流与数据流信息，弥补单一控制流或数据流分析的局限性。首先通过控制流分析生成控制流图，清晰呈现程序中指令的执行顺序与分支逻辑；再通过数据流分析追踪变量的定义、使用与传递过程，捕捉数据依赖关系；最终将控制流图与数据流图深度融合，生成包含语法、语义及依赖关系的代码属性图，完整保留漏洞相关的关键特征，为漏洞识别提供全面的特征支撑。（2）基于图神经网络的漏洞检测：为解决传统静态分析误报率、漏报率较高的问题，引入双向图神经网络模型对代码属性图进行处理。首先将代码属性图进行向量化转换，提取图中节点与边的特征，转化为模型可识别的向量形式；随后利用双向图神经网络模型进行训练，通过大量标注漏洞样本的学习，让模型掌握不同类型漏洞的特征模式；训练完成后，将待检测固件的代码属性图输入模型，实现对内存管理漏洞、输入验证漏洞等各类漏洞的精准检测，有效降低误报率与漏报率，提升检测精度[3]。

2.3 多维度静态分析融合策略

（1）SAST与SCA融合：采用静态代码分析（SAST）与软件成分分析（SCA）融合的策略，实现漏洞的全面检测。SAST技术聚焦于固件源代码及二进制代码本身，通过语法分析、语义分析挖掘代码层面的潜在漏洞；SCA技术则针对固件中集成的第三方依赖库、开源组件，扫描其版本信息并与CVE、NVD等主流漏洞库进行比对，快速识别已知的组件漏洞。两者融合互补，既覆盖代码自身的漏洞，也兼顾依赖组件的安全隐患，实现已知漏洞的快速识别与全面排查[4]。（2）自定义规则优化：针对嵌入式固件的特殊应用场景，如裸机编程、中断服务程序、实时操作系统适配等，传统通用检测规则针对性不足的问题，编写自定义漏洞检测规则。结合嵌入式固件的编程特点与常见漏洞类型，如中断嵌套漏洞、裸机编程中的内存越界等，制定贴合场景的检测规则，嵌入到静态分析工具中，提升分析的针对性与准确性，弥补通用规则在嵌入式特殊场景下的检测短板。

2.4 静态分析方法有效性验证

（1）实验环境搭建：为验证所提静态分析方法的有效性，构建完善的实验环境。首先搭建测试数据集，收集不同类型、不同厂商的嵌入式固件，涵盖工业控制设备、智能家居设备等常见场景，包含已知漏洞与未知漏洞样本，确保数据集的多样性与代表性；其次搭建基于QEMU的仿真测试环境，模拟不同嵌入式硬件平台的运行状态，无需依赖真实硬件设备，即可完成固件的静态分析与漏洞检测，降低实验成本，提升实验可重复性。（2）实验结果分析：将本文提出的静态分析方法与传统静态分析方法（如基于Flawfinder、SonarQube的单一检测方法）进行对比实验，以漏洞检测准确率、检测效率为核心评价指标，同时统计误报率、漏报率辅助分析。实验结果表明，本文方法在漏洞检测准确率上较传统方法提升15%-20%，检测效率提升30%以上，误报率与漏报率显著降低，充分验证了所提静态分析方法的优越性与实用性。

3 嵌入式设备固件安全防护机制构建

3.1 防护机制设计原则与目标

（1）设计原则：结合嵌入式设备资源受限、硬件耦合度高的核心特性，防护机制设计遵循四大核心原则。左移防御原则，将安全防护融入固件开发全流程，提前规避开发阶段的安全隐患，降低后期修复成本；轻量化原则，优化防护模块设计，控制代码体积与资源占用，适配嵌入式设备有限的存储、算力资源；自动化原则，实现安全检测、漏洞更新等环节的自动化运行，减少人工干预，提升防护效率；可追溯原则，记录固件开发、更新、漏洞拦截等全流程日志，便于安全事件溯源与问题排查。（2）设计目标：构建全方位、全生命周期的固件安全防护体系，核心目标是实现漏洞的提前预警、有效拦截、快速修复，打造固件级主动免疫能力。通过多环节防护措施，提前预警潜在漏洞风险，避免漏洞被攻击者利用；对已出现的漏洞实现实时有效拦截，阻断攻击路径；建立快速修复机制，及时推送安全更新，弥补漏洞隐患；最终保障嵌入式设备从开发、部署到运行、淘汰的全生命周期安全，抵御各类固件安全攻击。

3.2 固件开发阶段防护措施

（1）安全编码规范：制定贴合嵌入式固件开发场景

的安全编码标准，明确编码禁忌与规范要求。严格禁止使用strcpy、gets等易引发缓冲区溢出的不安全函数，替换为strncpy、fgets等安全替代函数；规范中断服务程序编写，避免中断嵌套混乱、执行周期过长等问题，防止出现权限泄露、程序崩溃等安全隐患；同时明确变量定义、内存分配、错误处理等编码细节，从源头减少编程层面的漏洞。（2）预提交安全检查：在固件开发流程中集成预提交钩子工具，实现代码提交前的自动化轻量级安全检查。该工具可快速完成代码静态扫描，检测代码中的语法错误、潜在漏洞；同时开展秘密信息检测，拦截代码中硬编码的密钥、密码等敏感信息，避免敏感信息泄露；对检测出的基础安全问题及时提醒开发人员修改，提前拦截安全隐患，避免问题流入后续开发环节[5]。

3.3 固件部署与运行阶段防护措施

（1）固件安全签名与验证：采用非对称加密算法对固件镜像进行数字签名，生成唯一签名信息并嵌入固件中。在设备启动过程中，首先执行签名验证流程，通过内置公钥验证固件签名的有效性，若签名不匹配或被篡改，则拒绝启动固件，从根本上防止固件被篡改、伪造，抵御恶意固件注入攻击。（2）运行时漏洞防护：针对内存溢出、权限提升、代码注入等高频固件漏洞，设计轻量化防护模块，集成到固件运行内核中。该模块无需占用过多设备资源，可实时监控程序运行状态，对异常内存操作、非法权限调用等行为进行拦截，及时终止攻击行为，避免漏洞被利用导致设备被控制、数据泄露等严重后果。（3）漏洞情报与动态更新：建立固件漏洞情报监控机制，实时对接CVE、NVD等主流漏洞数据库，及时获取嵌入式固件相关的最新漏洞信息与修复方案。针对已发现的漏洞，快速开发安全更新包，通过OTA方式向设备推送，并实现更新包的自动下载、安装与验证，确保设备固件及时修复漏洞，避免漏洞长期暴露引发安全风险。

3.4 防护机制可行性验证

（1）防护模块开发与集成：基于主流嵌入式开发平台，开发安全签名、运行时漏洞拦截、漏洞情报同步等核心防护模块，优化模块代码结构，控制资源占用，确保适配不同类型的嵌入式设备。将各防护模块与固件开发流程、设备运行系统深度集成，完成模块联调，确保各模块协同工作，实现全方位防护。（2）安全性与性能测试：搭建仿真测试环境，模拟各类固件攻击场景，测试防护机制对不同类型漏洞的拦截效果，验证防护模块的安全性与有效性。同时测试防护机制在嵌入式设备上的运行性能，监测其CPU占用率、内存消耗、响应速度等指标，确保防护机制不会影响设备正常功能运行，满足嵌入式设备的性能要求，验证整个防护机制的可行性与实用性。

结束语

本文完成嵌入式固件安全漏洞静态分析与防护机制的系统研究，构建的跨架构静态检测模型有效降低了误报率，实现了典型漏洞的精准识别。所提三级防护体系实现了从开发到运行的全流程安全管控，解决了分析与防护脱节的核心问题。未来可结合机器学习优化漏洞模式库，融合符号执行与动态分析提升复杂漏洞检测能力，进一步适配RISC-V等新型架构，为嵌入式设备安全提供更全面的技术支撑。

参考文献

[1]李华,王强.嵌入式系统固件升级技术的研究与应用[J].计算机工程与设计,2024,45(3):67-68.

[2]陈敏,刘伟.自动化测试在嵌入式软件开发中的应用[J].软件学报,2023,34(9):287-288.

[3]孙杰,高翔.持续集成与持续部署在固件开发中的实践探索[J].信息技术与网络安全,2024,43(11):123-129.

[4]李程,陈健雄.基于深度卷积网络的中低速磁浮接触轨紧固件松动检测[J].机车电传动,2022(4):172-179.

[5]蒋剑,蒋志刚.嵌入式软件外部质量度量方法的研究与实现[J].控制与信息技术,2021(1):85-90.