计算机科学
COMPUTER SCIENCE

关键字敏感的嵌入式设备固件模糊测试方法

司健鹏, 洪征, 周振吉, 陈乾, 李涛

引用本文

司健鹏, 洪征, 周振吉, 陈乾, 李涛. 关键字敏感的嵌入式设备固件模糊测试方法[J]. 计算机科学, 2024, 51(10): 196-207.

SI Jianpeng, HONG Zheng, ZHOU Zhenji, CHEN Qian, LI Tao. Keyword Sensitive Fuzzing Method for Embedded Device Firmware [J]. Computer Science, 2024, 51(10): 196-207.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230700068

关键字敏感的嵌入式设备固件模糊测试方法

司健鹏　洪　征　周振吉　陈　乾　李　涛
陆军工程大学指挥控制工程学院　南京 210007
(1183373785@qq.com)

摘　要　大部分嵌入式设备的固件提供 Web 接口,方便用户对设备进行配置和管理。然而,这些 Web 接口常常存在安全问题,给嵌入式设备的安全带来挑战。针对嵌入式设备固件中 Web 接口的漏洞检测方法误报率较高的问题,提出一种关键字敏感的嵌入式设备固件模糊测试方法 KS-Fuzz(Keyword Sensitive Fuzzing),高效地对嵌入式设备固件中 Web 接口的处理逻辑进行模糊测试。该方法通过前后端文件的关联分析,生成高质量的测试用例,在模糊测试过程中记录目标设备后端文件对前端文件关键字的引用,引导测试用例的变异,扩大模糊测试的覆盖范围。文中使用 KS-Fuzz 对多款主流品牌的嵌入式设备进行测试,以评估 KS-Fuzz 的漏洞挖掘能力,并与 SaTC,IOTScope,FirmFuzz 等现有漏洞挖掘方法进行比较。结果表明,相比现有漏洞挖掘方法,KS-Fuzz 通过对前后端文件关联性的分析,可以快速遍历目标设备的功能接口,在模糊测试过程中发现更多的安全问题。

关键词:嵌入式设备;模糊测试;灰盒测试;关联性分析;关键字敏感
中图分类号　TP309.1

Keyword Sensitive Fuzzing Method for Embedded Device Firmware

SI Jianpeng,HONG Zheng,ZHOU Zhenji,CHEN Qian and LI Tao
College of Command and Control Engineering,Army Engineering University of PLA,Nanjing 210007,China

Abstract　The firmware of most embedded devices provides a Web interface,which is convenient for the users to configure and manage the devices. However,the security problems of these Web interfaces usually bring challenges to the security of embedded devices. However,the existing vulnerability detection methods for Web interfaces in embedded device firmware have high false positive rates. This paper proposes a keyword-sensitive embedded device fuzzing method KS-Fuzz(keyword sensitive fuzzing),which efficiently performs fuzzing in the processing logic of the Web interface in the embedded device firmware. The proposed method generates high-quality test cases through the association analysis of front-end and back-end files,and records the references of keywords in the target device's back-end files to front-end files during the fuzzing process,to guide the direction of test case mutation,and improve the fuzzing coverage. In this paper,we use KS-Fuzz to test embedded devices of major brands to evaluate the fuzzing ability of KS-Fuzz,and compare KS-Fuzz with existing vulnerability mining methods,such as SaTC,IOTScope,and FirmFuzz. The results show that by analyzing the correlation of front-end and back-end files,KS-Fuzz can quickly traverse the functional interfaces of the target devices and discover vulnerabilities effectively.

Keywords　Embedded devices,Fuzzy testing,Grey box test,Correlation analysis,Keyword sensitive

1　引言

根据物联网操作系统安全白皮书[1],全球物联网连接数保持高速增长,2020 年全球物联网总连接数达到 131 亿,预计到 2025 年,连接规模将达到 246 亿。我国物联网连接数在全球占比超过 30%,产业规模突破 1.7 万亿元。作为物联网的重要技术组成部分的嵌入式设备,由于其底层架构种类繁多,软硬件资源有限,通用的安全防御机制难以适配到嵌入式设备上,导致大量缺乏有效保护的嵌入式设备暴露在互联网上,面临很大的安全风险。因此,针对嵌入设备高效准确地进行漏洞检测是亟待解决的重要问题。

嵌入式设备的固件由操作系统和驱动程序组成,负责控制硬件以及与各种软件组件进行通信,实现设备的功能。嵌入式设备的固件面向用户,提供了操作设备硬件的功能接口。多数嵌入式设备的固件提供 Web 接口,方便用户对设备进行配置和管理,而这些 Web 接口是黑客

到稿日期:2023-07-10　返修日期:2023-09-30
基金项目:智慧城市网络安全综合防控关键技术及系统(2019YFB2101704)
This work was supported by the Key Technologies and Systems for Comprehensive Prevention and Control of Cybersecurity in Smart Cities (2019YFB2101704).
通信作者:洪征(hz5215@163.com)

攻击嵌入式设备的重要途径。

Web 接口一般由前端页面、后端文件和中间件组成。前端文件构建完整的 Web 页面,并通过浏览器呈现给用户,向用户提供操作设备的交互接口;后端文件用于响应来自前端文件的 Web 请求,通过与前端页面通信来构建 Web 应用程序的完整功能;中间件指位于前端页面和后端文件之间的一层软件,用于处理 Web 请求和响应,实现 Web 接口的各项功能。

由于嵌入式设备底层架构种类较多,且受软硬件资源限制,嵌入式设备固件对 Web 请求的处理流程进行了优化,因此通用计算机平台的漏洞检测方法不能直接应用于嵌入式设备 Web 接口,需要根据嵌入式设备 Web 接口的特征进行定制设计。现有嵌入式设备处理 Web 请求的常用模型包括 Goahead 模型、mini_httpd 模型、Boa 模型、Lighthttpd 模型等。

在 GoAhead 模型中[2],Web 请求被路由到相应的控制器,控制器根据请求的内容从设备缓存中获取数据,并将数据传递给视图。视图根据模板呈现数据并生成 HTML 页面,生成的 HTML 页面被发送到客户端浏览器,呈现给用户。使用 Goahead 模型的有 Cisco,Tenda 等厂商。

mini_httpd 模型[3]是一种基于进程的架构模式,每个请求都会创建一个新进程来处理请求。这个新进程负责读取请求数据,执行相应的处理逻辑,然后生成响应数据。使用这种模型的有 Huawei,Zyxel 等厂商。

Boa 模型[4]是一种基于线程的架构模式,每个请求都会创建一个新线程来处理请求。这个线程负责读取请求数据,执行相应的处理逻辑,然后生成响应数据。使用这种模型的有 TOTOLink 等厂商。

Lighttpd 模型[5]是一种基于事件驱动的架构模式,它通过轮询操作系统的事件队列来处理请求。Lighttpd 模型在一个主线程中处理所有请求,当有请求可用时,Lighttpd 会读取请求数据,执行相应的处理逻辑,生成响应数据。使用这种模型的有 TP-Link,MERCURY 等厂商。

通过对上述模型进行分析后发现,嵌入式设备的 Web 接口的后端文件多为由 C 语言编译的 ELF 文件。C 语言可直接操作设备内存和硬件,且对变量类型的约束不严格,容易出现内存泄漏和错误访问等问题。根据已有漏洞的情况,内存溢出、越界访问、命令注入、未授权访问等高危漏洞集中存在于嵌入式设备的 Web 接口的后端文件。本文通过分析大量嵌入式设备的固件发现,设备厂商在后端文件中使用 Goahead,min_httpd 等 Web 模型来处理 Web 请求,这些模型的代码一般在互联网上开源,一直以来都是安全研究的重点对象,经过反复迭代升级后,存在零日漏洞的概率相对较小。然而,厂商对 Web 请求中用户参数的处理代码通常不对外公开,不像开源代码一样经过反复安全检测,存在漏洞的概率较大。

关键字用于在嵌入式设备固件的前端文件中标识用户提交的参数,如在用户登录过程中,用于标识用户名和登录密码的关键字是“username”和“password”。本文实现了一种关键字敏感的嵌入式设备固件模糊测试方法 KS-Fuzz。KS-Fuzz 根据前后端文件的关联关系和前端文件语义分析生成高质量测试用例,并通过动态二进制插桩方法来获取模糊测试过程中后端文件引用的前端文件关键字,引导种子变异策略。相比现有针对嵌入式设备的漏洞检测方法,KS-Fuzz 测试用例生成策略更符合嵌入式设备的固件中处理 Web 请求的流程特点。模糊测试过程中插桩策略可以引导 KS-Fuzz 关注被关键字标识的用户输入点和后端文件中对应的处理逻辑,从而可以更全面地覆盖设备固件的功能代码。本文的贡献如下:

1)提出了一种基于前后端共享关键字的测试用例生成方法。通过解析后端文件发现嵌入式设备隐藏的功能接口,并基于对前端文件的语义分析生成参数引用合法的测试用例,从而在测试过程中全面覆盖后端文件的代码,保证测试的全面性。

2)提出了一种关键字敏感的灰盒模糊测试方法。由于关键字常常用于标识用户输入点,因此关键字敏感的模糊测试结合后端文件的静态分析结果,通过动态分析方法来聚焦于处理用户输入的代码逻辑,以减小现有静态分析方法由于缺乏程序运行状态反馈支持而可能产生的误报率,提高模糊测试的效率。

3)实现了 KS-Fuzz 原型,对原型系统进行了分析测试,验证了所提方法的有效性。

2 相关工作

由于嵌入式设备的硬件资源有限,固件又缺乏操作系统和其他功能组件的支持,因此无法像通用计算机平台一样运行复杂的动态分析系统。此外,嵌入式设备的调试接口较为封闭,大部分嵌入式设备的硬件调试接口在出厂前被封闭。因此,目前针对嵌入式设备固件漏洞的动态分析方法以黑盒测试为主,但黑盒测试方法难以根据程序的实际运行状态调整测试策略,难以对目标设备进行深入的理解和分析,模糊测试效率低。

静态分析方法可以在没有目标设备动态分析环境的条件下,静态地检测设备固件中存在的安全漏洞。Redini 等[6]提出了一种静态分析方法 KARONTE,他们认为二进制文件使用进程间通信(IPC)范式进行通信,可以基于这些范式来检测用户输入被引入固件的位置,直接对相应位置进行污点分析,缩短污点传播路径。但这种方法只关注二进制文件间的数据流,忽略了 Web 接口前后端文件间的数据交互,漏洞检测效率不高。Chen 等[7]提出的静态分析方法 SaTC 使用正则匹配筛选出与用户输入相关的关键字,并在后端文件中分析这些关键字的处理流程,对处理流程进行符号执行和污点分析。这种方法通过前后端文件的映射缩短了污点传播路径,提高了静态分析的效率。上述方法采用静态分析方法对嵌入式设备的固件进行漏洞检测,由于缺少程序运行状态信息,因此难以识别文件内的隐式数据流,对关键字的识别率较低,漏洞检测效率不高。在实际应用中,上述方法需要使用大量计算资源进行符号执行和污点分析,效率较低。

通过仿真技术,利用 PC 机强大的计算资源模拟运行嵌入式设备的固件,可以获取细粒度的固件运行状态的反馈,从而引导模糊测试过程。由于嵌入式设备固件的运行需要与

外部设备的数据交互，因此如何高效地模拟外部设备与固件的交互过程是现有仿真工作需要解决的重要问题。Firmadyne[8]通过 QEMU 模拟器，复现了外部设置与固件之间常见的数据交互方式，从而对嵌入式设备的固件进行仿真。但 Firmadyne 能够复现的数据交互方式有限，因此固件仿真运行的成功率不高。Fuzzware[9]使用 Re-Hosting 仿真技术，在仿真运行过程中引入符号执行技术和模糊测试技术，对于 Re-Hosting 仿真技术中难以解决的 input overhead 问题，使用局部范围的符号执行方法，自动化地分析固件与外部设备之间的数据交互方式，提高固件仿真运行的成功率。

高质量的测试用例是保证模糊测试效果的关键。现有面向嵌入式设备的固件的测试方法主要通过静态分析嵌入式设备的固件在协议语法、逻辑处理等方面的特性，生成高质量的测试用例。FirmFuzz[10]由用户提供输入字典集，根据预设的输入字典集生成测试用例；然后使用 AFL 算法进行迭代和筛选，以生成更多有效的测试用例，提高测试过程的代码覆盖率。SIoTFuzzer[11]自动化抓取目标设备的数据包，通过机器学习对数据进行建模并生成测试用例。SRFuzzer[12]使用符号执行技术分析协议的执行路径，并通过解析路径上的符号约束生成有效的测试用例。DIANE[13]和 IOTFUZZER[14]通过分析与目标物联网设备通信的移动 APP 的输出逻辑，设计高质量的测试用例生成策略。FIRM-COV[15]通过对目标设备代码进行静态分析和动态分析，来获取代码覆盖率和程序执行路径，而后基于对覆盖率的分析，生成相应的测试用例，以尽可能多地提高代码覆盖率，发现潜在漏洞。IFIZZ[16]认为嵌入式设备中存在很多与输入无关的错误，如硬件故障、内存不足等，这些错误会导致固件在测试过程中提前崩溃，无法使固件深层的错误逻辑得到充分的测试。因此，IFIZZ 通过静态分析识别与输入无关的错误，并使用有错误状态感知的测试方法，引导测试用例的生成。

针对嵌入式设备的固件的动态分析大多是在对单一文件进行静态分析的基础上生成测试用例，并未关注 Web 处理流程中前后端文件的关联关系，因此现有的动态分析方法常常会忽视前后端文件关联关系在测试用例生成过程中的指导作用，导致无法全面覆盖嵌入式设备的功能接口。

在嵌入式设备的 Web 请求处理模型中，关键字是用户输入的参数，关键字的处理逻辑也就是用户输入的处理逻辑。但现有的基于系统仿真的灰盒模糊测试方法仅依靠代码覆盖率引导种子变异，缺乏关键字引用信息的反馈，难以对用户输入的处理逻辑进行深入的测试，模糊测试的效率相对较低。

3 系统设计

3.1 总体设计

嵌入式设备的 Web 请求一般由 URL 链接、HTTP 协议头部参数和用户提交的参数组成。多数嵌入式设备对 Web 请求的处理流程符合 Goahead 模型、mini_httpd 模型、Boa 模型或 Lighthttpd 模型。上述模型的共同点在于其后端文件多为 C 语言编译的 ELF 文件。后端文件在处理 Web 请求时，容易因对用户提交的参数检查不严格而引入内存溢出、越界访问、命令注入和未授权访问等漏洞。通过对前期 20 余款嵌入式设备固件的分析发现，后端文件的漏洞点逻辑较简单，如果仅采用静态分析方法，仍然会因为条件分支、嵌套循环、隐式数据流传播等问题而产生路径爆炸。相比静态分析方法，模糊测试通过生成大量的随机测试用例，对后端文件中的处理用户输入的逻辑进行深入的测试，可以更快地检测后端文件中的漏洞。

本文采用关键字敏感的灰盒模糊测试方法(KS-Fuzz)对嵌入式设备进行漏洞检测。关键字敏感指基于前后端文件中的关键字引导模糊测试过程中的测试用例生成和种子变异。由于在前端文件中，关键字用于标识用户输入点，而在后端文件中，引用前端文件关键字的代码逻辑常常用于处理用户输入，因此 KS-Fuzz 在测试用例生成阶段和模糊测试阶段的策略中添加关键字敏感，可以更全面地覆盖固件中处理用户输入的代码逻辑。在测试用例生成阶段，KS-Fuzz 依据前后端文件关联关系识别后端文件，在后端文件中提取 URL 字符串，并根据前端文件中对这些 URL 字符串的引用关系生成测试用例，快速地遍历后端文件的功能接口，包括难以被现有漏洞挖掘方法发现的一些隐藏的功能接口。在模糊测试阶段，如果在之前的测试过程中没有发现测试用例所触发的后端文件关键字，那么可以认为该测试用例发现了后端文件中新的用户输入点，同时也确定了这个用户输入点的处理逻辑。KS-Fuzz 会给予该测试用例更多的变异机会，从而使模糊测试充分关注后端文件内的关键字处理逻辑，提高模糊测试效率。

KS-Fuzz 主要由测试用例生成模块、基于生成的模糊测试模块和关键字拓展模块组成。测试用例生成模块生成测试用例集；模糊测试模块使用测试用例集对目标设备进行模糊测试；关键字拓展模块在模糊测试过程中监控设备运行状态，引导模糊测试过程。下文对各模块的主要功能进行了介绍。

1)测试用例生成模块。该模块通过共享关键字筛选与前端文件关联度较高的后端文件，并使用正则匹配在后端文件中搜索 URL 字符串，在此基础上，基于前端文件引用 URL 字符串的语义分析解析出以 URL 链接、关键字名、关键字值为基础的〈url，keyword，value〉三元组集，并将其作为初始测试用例集。

2)基于变异模糊测试模块。该模块解析初始测试用例集，根据 HTTP 协议的语法规范，使用基于生成的变异策略生成测试报文，并设置监控程序，用于在模糊测试过程中发现和恢复目标设备的异常运行状态。

3)关键字拓展模块。KS-Fuzz 通过关键字拓展模块动态收集后端文件对前端关键字的引用，引导种子变异，从而在模糊测试阶段实现关键字敏感。关键字拓展模块在对目标设备进行模糊测试之前，首先对后端文件中引用前端关键字的函数进行插桩，然后在模糊测试过程中记录被插桩函数的调用参数的变化，收集后端文件引用的前端文件关键字，并将其反馈给模糊测试过程。反馈主要采用两种方式：(1)当测试用例可以使后端文件处理更多的关键字时，就给予测试用例更多的变异机会；(2)当测试过程中发现了第一次被后端文件引用的关键字时，将关键字加入种子队列中，从而覆盖更多的用户输入点，提高模糊测试效率。系统组成如图 1 所示。

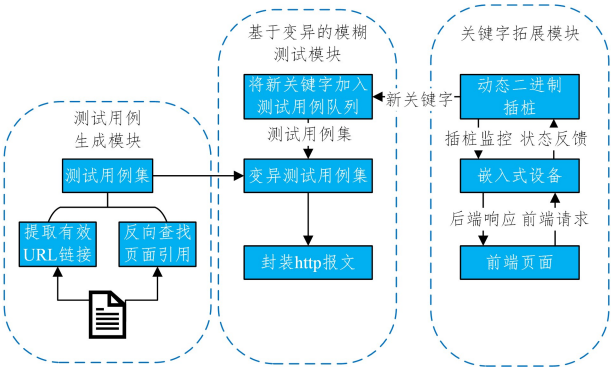

图 1 KS-Fuzz 系统结构图

Fig. 1 KS-Fuzz system structure

3.2 测试用例生成模块

根据嵌入式设备通用的 Web 请求处理流程，后端文件把 Web 请求分发给相应的处理逻辑之前，需要将 Web 请求中的 URL 链接与后端文件内嵌的 URL 进行比较。KS-Fuzz 通过提取内嵌于后端文件的 URL 字符串，生成可被测试的 URL 链接集。若提取生成的 URL 链接，通过 HTTP 请求可以使后端文件返回状态码为“200”的响应报文，则将这些 URL 链接称为有效 URL 链接。

有效 URL 链接往往是用于访问嵌入式设备特定服务的用户接口。在嵌入式设备的固件中，通过浏览器的 Web 页面向用户呈现的前端文件，通常会显式地引用有效 URL 链接。在前端文件中搜索这些 URL 链接的引用，提取 HTTP 协议头部参数和标识用户输入的参数关键字，即可与有效 URL 链接组合生成高质量的测试用例。

KS-Fuzz 通过测试用例生成模块，可以快速全面地遍历后端文件的功能接口，包括难以被现有漏洞分析方法发现的一些隐藏的功能接口，确保在测试过程中更全面地覆盖后端文件的代码。

3.2.1 后端文件识别

嵌入式设备的固件解包后的二进制文件包括响应 Web 请求的后端文件、busybox 工具箱文件和 Linux 系统内核文件等。由于嵌入式设备厂商的开发方式各不相同，后端文件在不同的嵌入式设备固件中的文件路径也不相同。根据嵌入式设备常用的 Web 请求模型，KS-Fuzz 通过文件名识别和前后端文件关联关系识别在固件中筛选后端文件。

1）文件名识别

后端文件名一般与嵌入式设备使用的 Web 请求处理模型相关，如 Tenda W15E Router 使用 Goahead 模型，其后端文件名为 goahead；Cisco RV160 VPN Router 使用 mini_httpd 模型，其后端文件名为 mini_httpd 等。结合前期对嵌入式设备的固件进行的逆向分析发现，嵌入式设备的后端文件的文件名一般包括如下字符串：httpd，boa，lighttpd，uhttpd，webs，internet，mini_httpd，sonia，switch，goahead，cgi 等。但并非文件名包括上述字符串的文件就是需要寻找的后端文件。嵌入式设备固件中存在很多脚本文件也可以响应 Web 请求。它们的文件名包含上述字符串。但 KS-Fuzz 漏洞分析的对象是后端文件中的 ELF 文件，需要对这些 ELF 文件进行二进制插桩，进而对目标设备进行动态分析。因此，脚本文件并不作为本文的分析对象。ELF 文件头部固定有 3 个字节的魔术字节“ELF”。KS-Fuzz 通过检查文件头部魔术字节的方式判断文件是否为可执行的 ELF 二进制文件。

2）前后端文件关联关系识别

Chen 提出的静态分析方法 SaTC 使用正则表达式搜索前端文件中与用户输入有关的关键字，而后在固件的二进制文件中搜索这些关键字。根据 Web 请求处理模型，二进制文件中引用前端文件关键字的位置很可能与用户输入的处理逻辑相关。若一个二进制文件中存在大量的前端关键字引用点，则该文件是后端文件的概率较大。本节基于二进制文件中引用前端关键字的次数对二进制文件进行排序，在实验过程中发现，引用前端关键字的次数较多的前 3 个文件是后端文件的概率较大。

如表 1 所列，在 DIR823G 固件的二进制文件中搜索前端文件关键字出现的次数，发现引用前端关键字次数最多的 3 个二进制为 boa，goahead 和 ppd。这 3 个文件将被视为后端文件进行下一步分析。

表 1 DIR823G 二进制文件引用关键字次数

Table 1 Times of keyword cited in DIR823G binary files

二进制文件名	前端关键字引用次数
boa	398
goahead	168
ppd	65
miniigd	61
wscd	62
iptables	56
sysconf	47
tc	44
timelycheck	44
dnsmasq	38
…	…

上述两种方法是根据嵌入式设备固件中后端文件的特征而设计实现的，但由于嵌入式设备厂商开发的习惯不同，不同设备固件中的后端文件特征并不完全相同，因此单独使用上述两种方法或者取两种方法的识别结果的或集并不能准确提取固件中的后端文件。通过实验发现，取两种方法识别结果的交集可以较准确地提取设备固件中的后端文件。

3.2.2 有效 URL 链接提取

后端文件一般将 Web 请求中的 URL 与后端文件内嵌的 URL 进行比对，根据比对结果，执行相应的处理逻辑，响应 Web 请求。

KS-Fuzz 通过静态分析技术，提取后端文件中用于比对的 URL 字符串，从而生成一组用于测试的 URL 链接，利用 URL 链接访问目标设备，根据返回报文，判断后端文件是否正常响应了 Web 请求。比如，判断返回状态码是否为“200”，若为 200，则认为是正常响应。KS-Fuzz 保留可以被正常响应的 URL 链接作为有效 URL 链接。

上述方法的具体实现主要包括 2 个步骤。

1）字符串提取

由于正则匹配可以灵活有效地匹配和处理各种不同类型的文本数据，KS-Fuzz 使用正则匹配的方法提取后端文件内用于 URL 比对的字符串。本节将 URL 链接分为两大类：

(1)访问前端文件的 URL 链接;(2)访问特定路径的 URL 链接。对于第一种 URL 链接,后端文件用于与之比对的字符串一般是以“.cgi”“.php”“.html”“.jsp”“.asp”等后缀名结尾,通过对上述后缀名进行正则匹配,可以提取后端文件中用于比对此类 URL 链接的字符串。对于第二种 URL 链接,后端文件中用于比对的字符串一般以“/”字符开头,以“/”结尾,如“/goform”,“/cgi-bin”等,本节以此为特征构建正则表达式提取 URL 链接。

2)URL 链接处理逻辑判定

后端文件内嵌入的与 URL 链接相关的字符串并不全是用于比对 Web 请求中的 URL 链接。

如图 2 所示,嵌入式设备 NetGear R7000 的固件中的 httpd 文件是一个后端文件,httpd 内的 sub_54E14 函数引用了“RST_status_dual_band.htm”字符串。根据前面的方法,该字符串会从 httpd 文件中提取出来并生成 URL 链接。但是 sub_54E14 函数仅仅是对该字符串做了一个拷贝操作,并没有对这个 URL 链接进行其他处理。

```
01. int __fastcall sub_54E14(int a1, int a2, char *dest)
02. {
03.   strcpy(dest, "RST_status_dual_band.htm");
04.   return 0;
05. }
```

图 2 NetGear R7000 的 httpd 文件 sub_54E14 函数

Fig. 2 Sub_ 54E14 function in httpd file of NetGear R7000

本文对可能是 URL 链接处理逻辑的地址进插桩。若在访问 URL 链接的过程中击中了预先插桩的地址,设备也能返回正常的响应报文,则认为该 URL 链接是有效的,否则为无效。

后端文件中分发 URL 链接到其对应处理逻辑的方法一般包括 3 种。(1)直接采用后端文件内嵌的字符串和请求的 URL 链接进行比对,比对成功后进入对应的处理逻辑对 URL 链接进行解析。(2)后端文件把文件内嵌的字符串和解析 URL 链接的函数地址作为参数显式地传入回调函数,当访问对应的 URL 链接时调用解析 URL 链接的函数。(3)后端文件会定义一个结构体数组,数组单项结构体元素中包括文件内嵌的字符串和解析 URL 链接的函数地址,后端文件会通过数组注册的回调函数,保证访问 URL 链接时可以调用对应的解析函数。

对于第一种处理逻辑,可以在后端文件中用于比对 URL 链接的地址处进行插桩。

对于第二种处理逻辑,由于后端文件内嵌的字符串和解析 URL 链接的函数地址是回调函数的参数,可以在引用 URL 字符串地址的代码上下文中搜索引用函数地址的代码,也就是引用回调函数的代码,并对引用的函数地址进行插桩。

对于第三种处理逻辑,文件内嵌的字符串和解析 URL 链接的函数的地址组成结构体数组存储于后端文件的数据段中。可以在字符串引用地址的周围搜索是否有函数地址。若存在函数地址,则对该函数地址进行插桩。

通过对上述 3 种处理逻辑的分析,可以对后端文件内可能是处理逻辑的地址进行插桩。当访问 URL 链接时,可以通过设备响应报文和插桩结果判断该 URL 链接是否是有效 URL 链接。

3.2.3 构造测试用例

测试用例是以 URL 链接、关键字名、关键字值为基础的〈url,keyword,value〉三元组集。通过 3.2.1 节和 3.2.2 节的工作可以解析出有效的 URL 链接。测试用例构造的主要工作是从前端文件中解析出构造测试用例所需要的关键字,并为每个关键字设置初始值。

开发嵌入式设备的厂商将 URL 链接作为后端文件向用户提供服务的接口,用户通过访问浏览器呈现的前端页面去使用功能接口操作设备。因此,前端文件中一般会包含提交有效 URL 链接的 form 表单。

根据 form 表单的语法规则,本文使用正则表达式搜索前端文件中引用 URL 链接的 form 表单。在 HTML 中,form 表单中的 action 标签用于指定用户提交表单后数据的目的 URL 链接。form 表单中 input 标签具有 name 和 value 属性,分别代表输入框的名称和输入框的值。当用户提交表单时,表单数据将这些输入框的名称和值配对后发送至服务器。通过正则匹配,在含有 form 表单的前端文件中,搜索引用有效 URL 链接的 action 标签,而后在 action 标签对应的 form 表单中通过正则匹配获取 input 标签中的 name 属性值,并将 name 属性值作为测试用例中的关键字。最后,随机生成一组测试数据作为关键字的初始值。

按照前面的介绍,KS-Fuzz 的测试用例生成策略如图 3 所示。

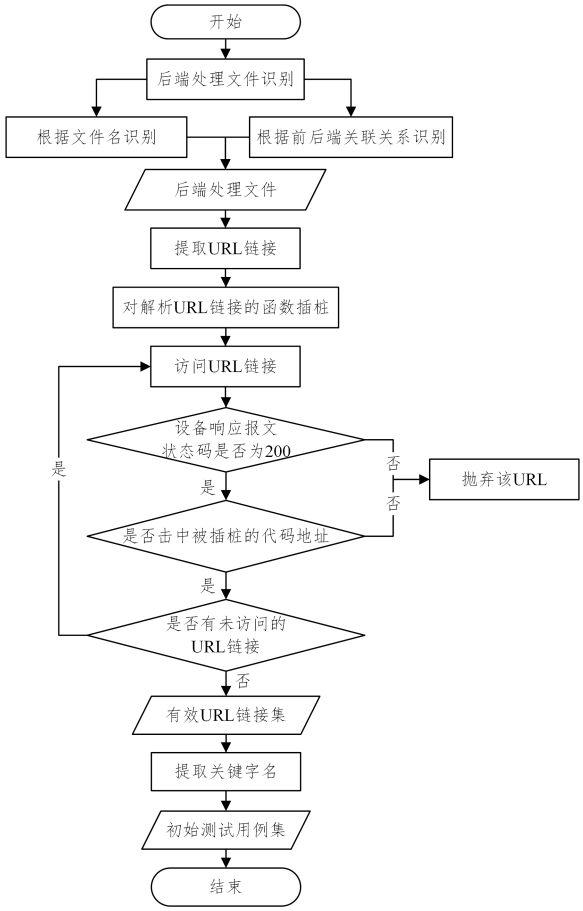

图 3 测试用例生成模块的工作流程图

Fig. 3 Workflow of test case generation module

首先通过对嵌入式设备的固件进行文件名识别和前后端

关联分析来确定后端文件，然后提取后端文件中与 URL 链接相关的字符串，生成 URL 链接集合。再对后端文件内所有可能解析 URL 链接的代码地址进行插桩，进而访问 URL 链接。根据设备响应报文和插桩反馈结果筛选可被后端文件解析的有效 URL 链接。最后，在前端文件中搜索引用有效 URL 链接的 form 表单，获取该 URL 链接可提交的参数和提交参数的方式，生成初始测试用例。

现有的测试用例生成方法并不关注前后端文件的关联关系和嵌入式设备 Web 接口处理模型的特征，难以在测试中全面地覆盖嵌入式设备的固件中后端文件的代码。KS-Fuzz 的测试用例生成方法可以生成针对后端文件的测试用例，保证了测试的全面性。

3.3 基于生成的模糊测试模块

KS-Fuzz 采用基于生成的模糊测试方式，根据 HTTP 协议规范和通过解析前端文件语义得到的〈url，keyword，value〉三元组集，引导模糊测试模块生成用于测试目标设备的报文。

基于生成的模糊测试模块主要进行生成测试报文、设置请求前的回调函数和监控目标设备的运行状态 3 部分工作。首先根据 HTTP 协议规范和〈url，keyword，value〉三元组集生成变异测试报文，而后设置模糊测试的请求前回调函数。请求前回调函数是在每次发送测试报文前 KS-Fuzz 需要执行的函数，主要用于更新访问目标设备的 HTTP 请求凭证，从而访问目标设备固件更深层的逻辑。在模糊测试过程中，需在目标设备内设置监控模块，在目标设备或服务异常时获取其状态并重启目标，提高测试效率。

3.3.1 生成测试报文

HTTP 报文一般由请求行、请求头部、请求消息正文组成。KS-Fuzz 通过抓包的方式获取 HTTP 报文的流量样本，从中提取出 HTTP 协议字段，结合〈url，keyword，value〉三元组集构造测试报文。

构造“请求方法”字段时，需要分析 URL 链接被引用的方式。如图 4 所示，在 NetGear R7000 固件的 Adv_USB. html 的 form 表单中，通过“method”字段规定了对“usb_approve. cgi”的请求方法为“POST”。

```
01. <form id="target"  name="adv_usb" method="post" action="usb_approve.cgi">
02. <input type="hidden" name="buttonHit"><input type="hidden" name="buttonValue">
03. <img class="subtop-image" src="<%77%>">
```

图 4 NetGear R7000 固件的 Adv_USB. htm 文件

Fig. 4 Adv_USB. htm file of NetGear R7000 firmware

在 HTTP 的请求头部，“Content-Length”字段用来表示请求消息正文的长度。由于嵌入式设备的固件对“Content-Length”的处理逻辑可能存在漏洞，过长的请求越界写入依据“Content-Length”字段值分配的内存块，或者在较大的“Content-Length”的控制下向较小的内存空间中写入过多的数据而导致溢出。如 Netgear R7000 的 CVE-2021-31802 漏洞就是由于后端文件缺乏对“Content-Length”字段值的安全性校验，导致缓冲区溢出漏洞。但是，如果 HTTP 报文中“Content-Length”值与请求正文的长度不一致，则该报文大概率会被目标设备抛弃，导致测试效率降低。为提高测试效率，KS-Fuzz 将“Content-Length”字段的初始值设置为与报文中请求正文长度一致，并设置变异次数阈值。当“Content-Length”字段的变异次数超过阈值后仍未触发崩溃，则放弃对“Content-Length”字段的变异，在后续测试过程中恢复“Content-Length”字段的初始值，保证模糊测试效率。

在完成对 HTTP 协议字段的设置后，需要提取〈url，keyword，value〉三元组集中的 keyword 字段作为用户提交的参数，再提取 value 字段作为 keyword 字段的默认值，并对初始的 value 值进行变异，将 keyword 字段和 value 值的变异结果拼接后，根据测试报文所对应的请求方式和 HTTP 协议规范，将其填充至测试报文。

3.3.2 设置请求前回调函数

请求前回调函数被用于更新访问目标设备的 HTTP 请求凭证，从而能够访问目标设备固件更深层的逻辑。以 NetGear R7000 为例，设备的 HTTP 请求凭证主要包括登录凭证和访问时间戳。

登录凭证用于验证设备用户身份，是授权其对设备进行管理和配置的一种凭据。通过输入正确的登录凭证，用户可以进入设备的管理页面，并进行需要身份验证的操作，如更改设备的设置、查看设备的状态、配置网络、升级固件等。在无登录凭证的情况下，只能对设备进行有限的测试。以 NetGear R7000 为例，登录凭证由“Cookie”字段和“Authorization”字段组成。设备在收到登录请求报文后，会将“Authorization”字段值做 Base64 解密处理，并将解密后的明文密码和设备内存储的密码进行比较，两者相符则设置响应报文中的“Cookie”字段的值为“XSRF_TOKEN＝”和一串 10 位的阿拉伯数字，并通过这串阿拉伯数字标识已登录的设备。NetGear R7000 在收到访问特定 cgi 的 HTTP 请求后，会检测请求报文中的时间戳是否合法，只有时间戳合法的请求才会被设备交付给对应的 cgi 处理。

请求前回调函数访问特定页面，更新访问设备的登录凭证，而后发送由 3.3.1 节构造的测试报文。KS-Fuzz 通过设置请求前回调函数，可以更深入地测试目标设备的处理逻辑，提高测试效率。

3.3.3 监控目标设备运行状态

通过监控目标设备运行状态，可以判断当前测试用例是否触发了目标设备 Web 服务的异常。本文主要监测以下 3 类异常。

1）服务重启。服务重启指当测试报文触发 Web 服务的漏洞或某些敏感接口后，引起 Web 服务的异常崩溃。但 Web 服务依然可以在守护进程的控制下在短时间内重启，不影响模糊测试进程。

2）设备重启。设备重启指当测试报文触发 Web 服务的漏洞或者某些敏感接口后，导致设备某些配置被非法篡改。目标设备将不能提供任何服务，需自动或手动重启。在实际测试过程中发现，当设备需要重启时，设备内服务的相关配置往往已被不可回滚地篡改，需要恢复出厂设置。

3）密码重置。设备密码若被测试用例重置，则 3.3.2 节中设置的请求前回调函数不能达到更新访问目标设备的 HTTP 请求凭证的效果，后续的测试报文也无法对目标设备

后端文件处理逻辑进行有效的检测。

对于上述 3 种异常情况，KS-Fuzz 设置有监控模块。当模糊测试过程中无法获取响应报文或者无法请求 Web 服务时，控制进程会尝试连接守护进程。若守护进程连接失败，则说明守护进程运行状态异常，或者设备的网络连接异常。此时控制进程可以暂停模糊测试，等待设备运行状态恢复正常。若控制进程可以正常连接守护进程，则命令守护进程重启 Web 服务。在控制进程检测到 Web 服务恢复正常后，继续实施模糊测试。

当模糊测试过程中响应报文返回“401”状态码时，表示设备认为请求报文是未授权访问报文。控制进程会暂停模糊测试过程，而后发送合法的登录请求。若可正常登录，则说明目标设备运行状态正常；若无法正常登录，则说明测试报文在未授权的条件下修改了目标设备的登录凭证。KS-Fuzz 会记录测试报文，并提示设备重置。

3.4 关键字拓展模块

关键字引入点标识了用户输入的关键字在后端文件的引用位置，一般是以关键字为参数的函数地址。嵌入式设备的 Web 接口的后端文件的关键字引入点在处理用户输入时，可能会由于对用户输入过滤规则不严格而导致内存溢出、越界访问、命令注入、未授权访问等漏洞。但现有的动态漏洞挖掘方法并未注意到上述漏洞与关键字引入点之间的关联关系，也没有对进程间的通信范式和跨文件的数据传播行为进行充分的监控。

SaTC 通过提取前后端共享关键字信息对设备固件中的后端文件的关键字引入点进行静态污点分析和符号执行，提高了分析效率。但 SaTC 是静态分析方法，在没有程序运行状态信息的条件下只能识别显式的关键字引入点，即通过显式引用关键字完成函数传参过程。但我们通过分析发现，还存在关键字的隐式引用关系。隐式引用关键字往往是通过数组、链表等指针的隐式引用关系完成函数的传参过程。与显示引用关键字不同的是，隐式引用关键字需要在程序实际运行过程中通过程序运作状态信息才能识别到，而 SaTC 则难以识别这类关键引用关系。本文方法可以通过在目标进程动态运行的过程中对关键字引入点插桩，监控每次调用过程的参数情况，获取隐式的关键字引用关系。

KS-Fuzz 的关键字拓展模块主要完成关键字引入点提取和关键字序列反馈两项工作。关键字引入点提取是通过静态分析方法在后端文件中确定关键字引入点。关键字序列反馈方法是在模糊测试开始前对关键字引入点插桩，在模糊测试过程中获取后端文件在处理测试报文时引用的前端文件关键字，从中发现被后端文件首次引用的新关键字，并将其加入〈url，keyword，value〉三元组中，而后模糊测试引擎给予能够发现新关键字的测试报文更多的变异机会，引导模糊测试变异方向。关键字拓展模块的工作流程如图 5 所示。

通过关键字拓展，KS-Fuzz 可以通过监控关键字引入点，获取后端文件在处理测试报文过程中引用的前端关键字，从而分析得到由前端关键字标识的用户输入位点，发掘后端文件中的功能接口，提升模糊测试的覆盖面。

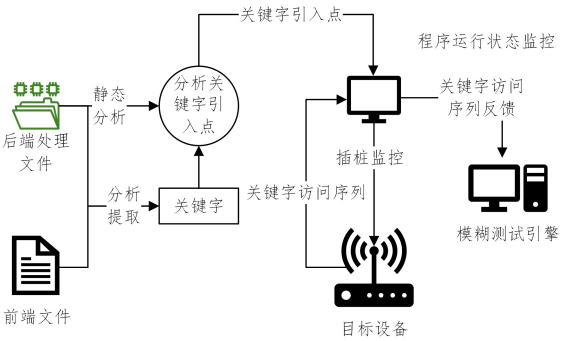

图 5 关键字拓展模块工作流程图

Fig. 5 Workflow of keyword extension module

3.4.1 关键字引入点提取

关键字引入点提取是在实施模糊测试前提取后端文件中的关键字引入点，并对关键字引入点插桩。在提取关键字引入点的过程中，主要有以下两项工作。

1)关键字识别。嵌入式设备一般通过前端文件的关键字标记用户输入，由后端文件实际处理被关键字标记的用户输入。前端文件的后缀名包括“.php”“.html”“.jsp”“.asp”“.xml”等。KS-Fuzz 选择前端文件中有关标记用户输入的标签进行解析，如输入文本、点击按钮、下拉框选取、触摸键点击。这些标签所包含的关键字一般分为参数关键字和动作关键字两种。其中，参数关键字是 KS-Fuzz 所关注的与用户输入相关联的关键字，动作关键字通常用来标记处理参数关键字的函数或后端文件。参数关键字的字段名由标签内的 id 属性值或 name 属性值描述。KS-Fuzz 使用正则表达式 r‘name=“(.*?)”’和 r‘id=“(.*?)”’提取前端文件中的关键字，并记录关键字在前端文件中被匹配时的前端文件名和前端文件路径，便于建立前端文件与后端文件的关联关系。

由于前端文件语义较为复杂，通过正则表达式 r‘name=“(.*?)”’和 r‘id=“(.*?)”’只能进行粗粒度的匹配，往往会匹配到全是数字、含有“@”“!”“%”等特殊字符的字符串或是长度过短的字符串，这类字符串并不是标识用户输入的关键字。KS-Fuzz 在过滤上述类型的字符串时，将长度阈值设置为 5，短于阈值的字符串会被过滤。阈值的选择是根据前期对固件中关键字的特征分析来确定的。

2)关键字引入点识别。关键字拓展模块引用 3.2.1 节的工作结果筛选后端文件，并在后端文件内搜索前端文件关键字的引用位置。与在前端文件中搜索关键字使用的方法相同，KS-Fuzz 使用正则匹配的方法，在后端文件中逐个搜索在前端文件中出现过的关键字。若能搜索到，则记录关键字在后端文件的引用位置和后端文件名，建立后端文件与前端文件的关联关系。

但并不是所有的关键字引用位置都是 KS-Fuzz 需要的关键字引入点。

如图 6 所示，地址 0xCCA81 是关键字“gui_region”的引用位置。但是地址 0xCCA81 位于“.rodata”数据段，说明 0xCCA81 处的数据不能被 CPU 当作指令执行，因此不能将该地址作为关键字引入点进行后续的插桩监控。

```
.rodata:000CCA81 67 75 69 5F 72 65 67 69 6F 6E+aGuiRegion DCB "gui_region",0          ; DATA XREF: sub_11530+2AEC↑o
.rodata:000CCA81 00                                                                  ; sub_11530+2B84↑o
.rodata:000CCA81                                                                     ; sub_11530:off_149E0↑o
.rodata:000CCA81                                                                     ; sub_11530+443C↑o
.rodata:000CCA81                                                                     ; sub_11530+4488↑o
.rodata:000CCA81                                                                     ; sub_11530+4508↑o
.rodata:000CCA81                                                                     ; sub_11530+4580↑o
.rodata:000CCA81                                                                     ; sub_11530+45A0↑o
```

图 6　关键字引用位置 1

Fig. 6　Keyword citation position 1

如图 7 所示，地址 0x6F2F4 是关键字“gui_region”的引用位置。虽然地址位于“. text”代码段，但该地址并非函数地址，因此地址 0x6F2F4 不是关键字引入点。但地址 0x6F2F8 调用了“acosNvramConfig_get”函数，且根据 ARM 函数传参方式可知“acosNvramConfig_get”函数的第一个参数是关键字“gui_region”。

```
.text:0006F2F4 64 02 9F E5          LDR             R0, =aGuiRegion         ; "gui_region"
.text:0006F2F8 49 81 FE EB          BL              acosNvramConfig_get
.text:0006F2F8
.text:0006F2FC 00 10 A0 E1          MOV             R1, R0                  ; src
.text:0006F300 64 02 9F E5          LDR             R0, =byte_F40628        ; dest
.text:0006F304 87 82 FE EB          BL              strcpy
```

图 7　关键字引用位置 2

Fig. 7　Keyword citation position 2

如图 8 所示，地址 0xF824 是“acosNvramConfig_get”函数地址，即关键字引入点。由图可知，“acosNvramConfig_get”函数的第一个参数是引用的关键字字符串。在模糊测试过程中插桩监控地址 0xF824，获取“acosNvramConfig_get”的第一个参数值，并将其作为关键字结果反馈至模糊测试引擎。

```
.plt:0000F824                       acosNvramConfig_get             ; CODE XREF: sub_101A8+B84p
.plt:0000F824                                                       ; sub_1030C+A04p
.plt:0000F824                                                       ; sub_1030C+B44p
.plt:0000F824                                                       ; sub_1030C+C04p
.plt:0000F824                                                       ; sub_1030C+DC4p
.plt:0000F824                                                       ; sub_1030C:loc_10450↓p
.plt:0000F824                                                       ; sub_10530+84p
.plt:0000F824                                                       ; sub_10530+184p
.plt:0000F824                                                       ; sub_10530+284p
.plt:0000F824                                                       ; sub_10530+384p
.plt:0000F824                                                       ; sub_10530+A04p
.plt:0000F824                                                       ; sub_10654+84p
.plt:0000F824                                                       ; sub_10654+184p
.plt:0000F824                                                       ; sub_10654+6C4p
.plt:0000F824                                                       ; .text:00010C084p ...
.plt:0000F824 01 C6 8F E2           ADR             R12, 0x10F82C
.plt:0000F828 10 CA 8C E2           ADD             R12, R12, #0x10000
.plt:0000F82C 04 FF BC E5           LDR             PC, [R12,#(acosNvramConfig_get_ptr - 0x11F82C)]! ; __imp_acosNvramConfig_get
```

图 8　关键字引用位置 3

Fig. 8　Keyword citation position 3

按照前面的介绍，关键字引入点识别工作首先搜索后端文件中引用前端文件关键字的位置，然后在引用位置附近搜索关键字引用函数的调用地址，根据指令集特性记录关键字引用函数被调用时关键字的参数位置，最后将关键字引用函数的地址作为关键字引入点结果返回至 KS-Fuzz。

3.4.2　关键字序列反馈方法

关键字引入点提取方法通过关键字识别和关键字引入点识别工作，获取后端文件关键字引入点的地址信息。而后，关键字序列反馈方法对关键字引入点进行插桩，并反馈在模糊测试过程中获取的关键字访问序列。

关键字序列反馈的目标是在模糊测试过程中获取后端文件在处理测试报文时引用的前端文件关键字，发现隐式关键字引用关系，引导测试用例生成策略的调整，提高模糊测试效率。该方法主要包括以下两项工作。

1）通过关键字引入点提取方法获得关键字引入点，对引入点进行插桩。现有的插桩方法根据插桩粒度可分为指令粒度、函数粒度和基本块粒度。由于关键字引入点为函数地址，因此该方法采用函数粒度的插桩。通过 Linux 系统自带的 Ptrace 系统调用对关键字引入点进行监控。

Ptrace 系统调用是在 Linux 和其他类 Unix 操作系统中使用的系统调用。它允许一个进程监控和控制另一个进程的执行，通常被用于调试（Debug）或跟踪（Trace）程序的执行流程。调试进程通过 Ptrace 系统调用可以附加被调试进程并观察它的内存和寄存器状态，同时调试进程还可以通过 Ptrace 系统调用修改被调试进程的内存数据和注入代码。

对于基于 Linux 内核的嵌入式设备来说，其固件系统可以提供 ptrace 系统调用。Ptrace 的系统调用号一般为 26，具体的系统调用号可能会因操作系统版本和架构不同而有所不同。为了通过 Ptrace 系统调用对目标进程插桩，需要获取嵌入式设备固件的调试权限。目前主要有基于仿真环境和基于实体设备调试接口两种调试固件的方法。基于仿真环境的固件调试方法的代表工作是 Firmadyne，通过模拟固件运行所依赖的硬件接口，在 Qemu 虚拟机上运行设备固件。在 Firmadyne 工作的基础上，可以在 Qemu 虚拟机的命令终端中使用 Ptrace 系统调用对目标进程进行插桩。基于实体设备调试接口的固件调试方法需要参考不同设备的软硬件特点，可以在软件层面发现固件隐藏的调试后门和口令，也可以在硬件层面识别常见的调试串口，如 UART 等，获取设备固件的调试权限，从而使用 Ptrace 系统调用对目标进程进行插桩。

采用 Ptrace 对关键字引入点进行插桩，首先通过“PTRACE_ATTACH”调试信号获取目标进程的调试权限，读取并备份所有关键字引入点的指令，然后将关键字引入点的指令修改为软件中断指令。当目标进程运行至关键字引入点处，会因软件中断指令而暂停运行，并产生异常信号。调试进程接到目标进程的调试信号后，会使用“PTRACE_GETREGS”“PTRACE_PEEKTEXT”调试信号获取当前关键字引入点的调用参数的值，而后控制被调试进程继续执行。

2）在模糊测试过程中收集调用关键字引入点的参数信息，筛选在静态分析过程中未发现的新关键字，并将其加入〈url，keyword，value〉测试用例集。

为了提高测试效率，位于目标设备上的调试进程在获得关键字引入点的调用参数的值后直接通过网络传输至模糊测试引擎，模糊测试引擎将从这些参数值中筛选出关键字。经分析后发现，参数值并不全是关键字。有些参数值含有特殊字符，例如“@”“!”“%”，这些参数值并不是关键字，本文通过正则匹配对它们进行过滤。另外，如果参数值以“=”结尾，则关键字是左边的部分，本文将截取参数值“=”左边的部分得到关键字。

KS-Fuzz 将经过筛选的关键字与已有的〈url，keyword，value〉三元组中的“keyword”进行比较，检查当前关键字是否是模糊测试过程中新发现的关键字。如果是，则将触发新关键字的 URL 链接、新关键字和新关键字的默认值加入〈url，keyword，value〉三元组集；如果不是，则抛弃。通过关键字拓展模块，KS-Fuzz 可以在模糊测试的过程中使用动态插桩的方法获取后端文件引用的前端文件中的关键字，并引导模糊测试种子变异，可以解决现有模糊测试对关键字反馈不敏感和静态分析方法难以发现隐式关键字引用关系等问题，使模糊测试覆盖更多的用户输入点，从而更全面地覆盖后端文件的代码。

4 实验设计

4.1 系统原型实现

KS-Fuzz 原型系统主要完成测试用例生成、基于生成的模糊测试和关键字拓展这 3 项工作。原型系统的测试用例生成模块和基于生成的模糊测试模块由 Python 语言开发，运行于测试机。关键字拓展模块由 C 语言开发，在测试机交叉编译完成后上传至目标设备。

测试用例生成模块主要的工作包括后端文件识别、有效 URL 链接提取和构造测试用例集。基于生成的模糊测试模块是基于 boofuzz 开发的，根据测试用例生成模块得到的〈url，keyword，value〉三元组集生成测试报文。通过目标设备的硬件调试接口上传关键字拓展模块。

4.2 实验部署方法

本文部署 KS-Fuzz 的操作系统为 Ubuntu 18.04.1 LTS，硬件配置为 Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz，2.30GH，2.0GB。

实验选择市场保有量较大的主流品牌的物联网设备作为测试目标，旨在评估 KS-Fuzz 在检测设备漏洞过程中在测试用例生成能力、关键字引入点识别能力和漏洞检测效率 3 个方面的表现，并与 SaTC，IOTScope 和 FirmFuzz 进行横向对比。测试目标列表如表 2 所列。

表 2 实验目标嵌入式设备

Table 2 Experimental target embedded devices

序号	厂商	型号	版本	CPU 指令集
1	D-Link	D-Link 823G	V1.0.2B05_20181207	MIPSEL
2	NetGear	R7000	V1.0.11.100_10.2.100	ARM
3	Tenda	AC15	V15.03.05.19	ARM
4	TOTOLINK	T10	V5.9c.1485_B20180122_ALL	MIPS
5	CISCO	RV110W	FW_1.2.2.5	MIPS
6	linksys	WRT54G	V4.4.21.5	MIPS
7	TP-Link	TL-WR740N	V3.20.1 build 200316	MIPS
8	wayos	WAM_9900	V21.10.09	MIPS

4.3 测试用例生成

KS-Fuzz 的测试用例生成模块对表 2 所列的 8 款嵌入式设备的固件进行了后端文件识别、有效 URL 链接提取和构造测试用例集等工作，完成了初始测试用例的构造。

KS-Fuzz 在构造测试用例过程中多次访问到设备的隐藏接口。这些隐藏接口一般用于设备的开发调试，或者作为设备在版本升级中被新版本弃用的功能接口，因此这些隐藏接口的权限检查一般不严格。

以 NetGear R7000 为例，KS-Fuzz 通过前后端文件关联性分析确定了 httpd 为后端文件，并从中提取出 885 个 URL 链接字符串。经过筛选，排除了 80% 与后端文件无关的 URL 链接，确定了 135 个有效 URL 链接，构成初始测试用例集，用时 68min。期间，KS-Fuzz 触发了 15 次服务重启异常，13 次设备重启异常，1 次密码重置异常。

经过异常复现分析发现，服务重启异常和设备重启异常多是 NetGear R7000 内置的服务重启和设备重启的功能接口没有做完整的权限校验造成的，可以通过简单的 http 请求使设备执行高危的重启操作。

KS-Fuzz 在第 35 分钟时触发了密码重置异常，可以在无需原密码的情况下，直接设置管理员账户密码，属于权限绕过漏洞。

通过路由器的 Web 管理页面的密码设置功能，则如图 9 所示，需要验证用户是否拥有正确的原始密码，而在该界面设置密码之前需要验证用户是否拥有正确的原始密码，只有在用户输入正确的原始密码后，才允许用户设置新密码。该界面设置管理员密码的操作是符合安全操作规范的。

KS-Fuzz 在测试用例提取过程中，从后端文件 httpd 提取出某 cgi 链接。然后根据测试用例构造策略发现，该款设备隐藏的“Password Reset”页面无需校验原始密码就可以修改管理员密码，存在权限绕过的安全问题。

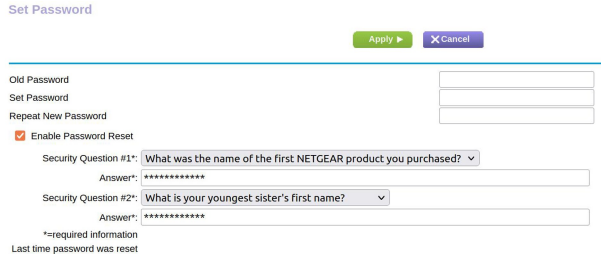

图 9 NetGear R7000 设备设置密码界面

Fig. 9 Password setting interface of NetGear R7000 device

4.4 关键字引入点识别

关键字引入点识别工作由 KS-Fuzz 的关键字拓展模块完成。本文对表 2 所列的 8 款嵌入式设备的固件进行分析，基于 4.3 节中的后端文件识别工作，进行前后端文件关联性分析，得到表 3 所列的关键字引入点列表。

表 3 目标设备关键字引入点识别

Table 3 Recognition of target device keyword introduction points

目标设备型号	后端文件	关键字引入点
DLink 823G	goahead	mxmlFindElement
NetGear R7000	httpd	sub_1A558
Tenda AC15	httpd	sub_2BA8C
TOTOLINK T10	lighthttpd system.so firwall.so	websGetVar
CISCO RV110W	httpd	get_cgi
Linksys WRT54G	httpd	nvram_get
TP-LINK TL-WR740N	httpd	httpRpmDataGet
Wayos_WAM9900	jhttd	jhl_nv_get_def

4.5 漏洞检测能力评估

本节对表 2 所列的 8 款嵌入式设备进行了模糊测试，

以此来检验 KS-Fuzz 的漏洞检测能力。与静态分析方法 SaTC 和动态分析方法 IOTScope、FirmFuzz 进行对比，比较分析 KS-Fuzz 验证已知漏洞和挖掘未知漏洞的能力。

4.5.1 已知漏洞的验证能力评估

在实验过程中，已知漏洞的验证情况如表 4 所列。KS-Fuzz 能够验证 6 个已知漏洞，SaTC 能够验证 2 个已知漏洞，IOTScope 无法验证表 4 所列的已知漏洞，FirmFuzz 可以验证表 4 中的 1 个漏洞。

以 ZDI-20-712 为例，分析 KS-Fuzz 检测已知漏洞的原理。根据有效 URL 链接的提取策略，后端文件 httpd 对 URL 链接“upgrade_check. cgi”的处理方法是比对后端文件内嵌的字符串和请求的 URL 链接来选择 Web 请求的处理逻辑。

如图 10 所示，httpd 文件使用 strstr 函数判断 Web 请求中是否包含 URL 链接“upgrade_check. cgi”，并根据判断结果选择对应的处理逻辑。

表 4 已知漏洞验证对比

Table 4 Comparison of known vulnerability verification

漏洞编号	设备型号	漏洞类型	危险关键字	KS-Fuzz 能否发现	SaTC 能否发现	IOTScope 能否发现	FirmFuzz 能否发现
CVE-2020-25367	Dlink 823G	命令注入	/HNAP1	√	×	×	√
ZDI-20-712	NetGear R7000	缓冲区溢出	mtenFWUpload	√	×	×	×
CVE-2020-10987	Tenda AC15	命令注入	devicename	√	√	×	×
CVE-2022-42525	TOTOLINK T10	命令注入	hostName	√	√	×	×
CVE-2020-3331	CISCO RV110W	缓冲区溢出	ubmit_button	√	×	×	×
CVE-2022-24355	TP-LINK TL-WR740N	缓冲区溢出	/tmp/	√	×	×	×

```
if ( stristr(s1, "Content-Disposition:")
   && stristr(s1, "Content-Length: ")
   && stristr(s1, "upgrade_check.cgi")
   && (stristr(s1, "Content-Type: application/octet-stream") || stristr(s1, "MSIE 10"))
   || stristr(s1, "Content-Disposition:") && stristr(s1, "Content-Length: ") && stristr(s1, "backup.cgi")
   || stristr(s1, "Content-Disposition:") && stristr(s1, "Content-Length: ") && stristr(s1, "genierestore.cgi") )
{
    if ( stristr(s1, "filename=\"\"") )
        goto LABEL_351;
    goto LABEL_306;
}
```

图 10 判断请求 URL 链接“upgrade_check. cgi”的处理逻辑

Fig. 10 Determine the processing logic of request URL link “upgrade_check. cgi”

根据构造测试用例的策略，KS-Fuzz 在前端文件“Modem_upgrade. htm”搜索到“upgrade_check. cgi”的引用。如图 11 所示，在对前端文件“Modem_upgrade. htm”的 form 表单的语义分析中获得请求 upgrade_check. cgi”的方式为“POST”，获得的请求参数为“mtenFWUpload”等。

然后 KS-Fuzz 使用 URL 链接“upgrade_check. cgi”、请求参数“mtenFWUpload”等和参数默认值构造〈url，keyword，value〉三元组，对 cgi 链接“upgrade_check. cgi”进行模糊测试。

```
<form method="POST" action="upgrade_check.cgi" ENCTYPE="multipart/form-data">
<table border="0" cellpadding="0" cellspacing="3" width="100%">
<tr>
 <td valign="top" align="center">
     <h1><%1258%></h1>
 </td>
</tr>
<tr> <!-- RULE -->
 <td colspan="2" background="liteblue.gif" height="12"></td>
</tr>
<tr>
 <td colspan="2" align="center"> <b><%1607%>:</b></td>
</tr>
<tr>
 <td colspan="2" align="center"><INPUT TYPE="FILE" NAME="mtenFWUpload" SIZE="40" MAXLENGTH="128" value=""></td>
</tr>
<tr> <!-- RULE -->
```

图 11 引用“upgrade_check. cgi”的前端文件 Modem_upgrade. htm

Fig. 11 Citing the front-end file Modem_ Upgrade. htm of “upgrade_check. cgi”

如图 12 所示，后端文件在处理由关键字“mtenFWUpload” 标识的用户输入时，未严格对用户输入进行检查。KS-Fuzz 在对参数“mtenFWUpload”值进行变异后，可检测到这个错误，触发服务重启异常。

```
25  v7 = *((unsigned __int8 *)src + 5);
26  v8 = *((unsigned __int8 *)src + 36);
27  *((_BYTE *)src + 37) = 0;
28  *((_BYTE *)src + 36) = 0;
29  *((_BYTE *)src + 38) = 0;
30  v9 = v5 + (v6 << 8) + (v7 << 16);
31  *((_BYTE *)src + 39) = 0;
32  memset(s, 0, 0x64u);
33  memcpy(s, src, v9);
34  calculate_checksum(0, 0, 0);
35  calculate_checksum(1, s, v9);
```

图 12 NetGear R7000 缓冲区溢出漏洞

Fig. 12 Buffer overflow vulnerability of NetGear R7000

4.5.2 未知漏洞的挖掘能力评估

未知漏洞挖掘情况如表 5 所列，KS-Fuzz 可以挖掘 11 个未知漏洞，SaTC 可以挖掘到 6 个未知漏洞。相比而言，IOTScope 和 FirmFuzz 均未挖掘到 0day 漏洞。经过人工核实分析，上述漏洞均属于缓冲区溢出漏洞。举例来看，测试 NetGear R7000 设备时，KS-Fuzz 发现设备固件在处理某个“fwSchedule. cgi”链接时，使用 sprintf 函数将某关键字的值复制到有限的栈空间前，没有进行有效的长度校验，存在栈溢出漏洞。KS-Fuzz 在对 Dlink 823G 设备的“SetWLanRadioSettings”接口进行测试时，将某关键字的值设置为超长的字符串，设备固件并没有进行有效的长度检验，直接采用 strncpy 函数进行处理，导致了内存访问异常。

表 5 未知漏洞的发现

Table 5 Discovery of unknown vulnerabilities

设备型号	危险关键字引入点	KS-Fuzz发现未知漏洞个数	SaTC发现未知漏洞个数	IOTScope发现未知漏洞个数	FirmFuzz发现未知漏洞个数
Dlink 823G	mxmlFind Element	2	0	0	0
NetGear R7000	sub_1A558	3	2	0	0
Tenda AC15	sub_2BA8C	3	4	0	0
TOTOLINK T10	websGetVar	3	0	0	0

SaTC 是静态漏洞检测方法，漏洞检测能力依赖于静态分析工具反汇编精度、符号执行工具的求解能力和污点分析工具的准确性。但由于缺乏程序动态运行状态的反馈，静态分析工具反汇编精度依然存在很大的改进空间。在对 SaTC 的静态分析工具的反汇编结果进行分析时发现，存在很多类似于“[Param “wep_key_no_an”(0x0010b1bf)，Referenced at FUN_0005b12c:0x0005cf80]≫0x0005d38c≫FUN_0005b0e4≫0x0005b29c－＞strcpy”的危险路径。该危险路径的含义是后端文件 httpd 在函数 FUN_0005b12c 的 0x0005cf80 地址处引用了关键字“wep_key_no_an”，关键字所标识的用户输入应在函数 FUN_0005b0e4 的 0x0005b29c 地址处被 strcpy 当作源地址执行内存拷贝操作。但通过人工分析发现，0x0005b29c 地址处的 strcpy 函数调用属于函数 FUN_0x0005B12C，而非函数 FUN_0005b0e4。SaTC 后续基于上述错误的反汇编结果进行污点分析和符号执行，会浪费大量计算资源，消耗较长测试时间，导致测试效率下降，甚至会产生大量误报。

IOTScope 和 FirmFuzz 是动态漏洞检测方法，可以降低对静态分析能力的依赖，但这两项工具均未关注到与用户输入相关的关键字信息，在动态漏洞检测过程中不能针对处理用户输入的代码逻辑生成测试用例，导致大量的测试用例被设备固件过早地抛弃。以对 Dlink 823G 设备的漏洞挖掘过程为例，KS-Fuzz 可以定位到后端文件 goahead 中处理“SetWLanRadioSettings”接口的代码逻辑。FirmFuzz 通过监控操作设备的流量数据获取初始测试用例，但 FirmFuzz 未能获取访问接口“SetWLanRadioSettings”的流量数据。KS-Fuzz 在模糊测试过程中监控关键字引入点的参数信息，发现了其中的关键字，然后对关键字的值构造测试用例，触发内存访问异常。经人工分析，确定该异常为缓冲区溢出漏洞。与 FirmFuzz 相比，IOTScope 关注到了前后端文件的关联关系，但并未深入分析固件的后端文件中引用前端关键字的代码逻辑，只是局限于对敏感接口的权限校验完整性的测试。因此，IOTScope 在对 Dlink 823G 的漏洞挖掘过程中并没有对核心的关键字进行模糊测试，也没有触发内存访问异常。

KS-Fuzz 采用动静结合的模糊测试方法，通过静态分析提取前端文件中的关键字和后端文件中的关键引入点，并使用动态二进制插桩方法在动态分析过程中检测和修正静态分析结果，从而避免 SaTC 静态分析方法的不足，降低误报率。相比动态分析方法 IOTScope 和 FirmFuzz，使用后端文件中关键字引用信息引导模糊测试，将资源集中到可能触发漏洞的测试用例上，提高了测试效率。

结束语 本文提出了一种关键字敏感的嵌入式设备固件模糊测试方法 KS-Fuzz，通过分析前后端文件的关联性来提取关键字，引导测试用例和模糊测试过程，解决了现有模糊测试方法生成测试用例的质量不高和模糊测试效率低下的问题。本文在测试用例生成过程中分析前后端文件关联性，提取后端文件的敏感接口，在前端文件中反查这些接口的引用关系，然后基于对前端文件语义的理解生成高质量的测试用例。在模糊测试过程中，对标记用户输入的关键字引入点进行插桩监控，获取目标设备处理测试用例时的关键字访问序列，引导模糊测试的变异策略，提高测试效率。实验结果表明，本文方法具备较强的漏洞检测能力。在进一步的工作中，我们将继续对基于前后端文件关联关系的测试用例生成策略进行研究，对 JS 脚本、HTML 脚本等前端文件的语义进行深入分析，辅助测试用例的生成。此外，我们考虑采用基于指令分支预测的动态插桩方法替换基于 Ptrace 调试的插桩方法来提高插桩效率，缩短后端文件处理测试用例的时间，进一步提高整体模糊测试的效率。

参 考 文 献

[1] China Communications Standards Association. Internet of Things Operating System Security White Paper (2022) [EB/OL]. (2022-09-08) [2023-08-03]. http://blog.nsfocus.net/wp-content/uploads/2022/09/iot-whitepaper.pdf.

[2] TO BE BETTER_MEN. GoAhead1- Basic Introduction [EB/OL]. (2022-09-08) [2023-08-03]. https://blog.csdn.net/to_be_better_wen/article/details/128749040.

[3] MY HEART. Realize HTTP server from zero－Minihttpd (IV)－semi connected and semi reactor Thread pool[EB/OL]. (2020-07-03) [2023-08-03]. https://www.jianshu.com/p/b11fabfc2c6c.

[4] ONE PORT LINUX. Building an embedded web server from scratch-boa[EB/OL]. (2022-02-27) [2023-08-03]. https://blog.csdn.net/daocaokafei/article/details/122738254.

[5] KIKILBS. Schematic diagram of CGI execution in lighttpd[EB/OL]. (2010-07-05) [2023-08-03]. https://blog.csdn.net/kikilbs/article/details/5713677.

[6] REDINI N, MACHIRY A, WANG R, et al. Karonte: Detecting Insecure Multi-binary Interactions in Embedded Firmware[C]// 2020 IEEE Symposium on Security and Privacy. 2020: 1544-1561.

[7] CHEN L, WANG Y, CAI Q, et al. Sharing More and Checking Less: Leveraging Common Input Keywords to Detect Bugs in Embedded Systems[C]// 30th USENIX Security Symposium. 2021: 303-319.

[8] CHEN 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.

[9] TOBIAS S, NILS B, MORITZ S, et al. Fuzzware: Using Precise {MMIO} Modeling for Effective Firmware Fuzzing [C]// 31st

USENIX Security Symposium. 2022：1239-1256.
[10] 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. 2019：15-21.
[11] ZHANG H，KAI L，XU Z，et al. SIoTFuzzer：Fuzzing Web Interface in IoT Firmware via Stateful Message Generation [J]. Applied Sciences，2021，11(7)：3120.
[12] ZHANG Y，HUO W，K P，et al. SRFuzzer：An automaticfuzzing framework for physical SOHO router devices to discover multi-type vulnerabilities [C]// the 35th Annual Computer Security Applications Conference. 2019.
[13] NILO R，ANDREA C，DIPANJAN D，et al. DIANE：Identifying FuzzingTriggers in Apps to Generate Under-constrained Inputs for IoT Devices[C]// 2021 IEEE Symposium on Security and Privacy. 2021：484-500.
[14] CHEN J，DIAO W，ZHAO Q，et al. IoTFuzzer：Discovering memory corruptions in IOT through APP-based fuzzing[C]// Network and Distributed System Security Symposium. 2018.
[15] KIM J，YU J，KIM H，et al. FIRM-COV：High-Coverage Grey-box Fuzzing for IoT Firmware via Optimized Process Emulation [J]. IEEE Access，2021，9：101627-101642.
[16] LIU P，JI S，ZHANG X，et al. IFIZZ：Deep-State and Efficient Fault-Scenario Generation to Test IoT Firmware [C]// 2021 36th IEEE/ACM International Conference on Automated Software Engineering(ASE). Melbourne，Australia，2021：805-816.

SI Jianpeng，born in 1996，postgraduate. His main research interest is cyber securiy.

HONG Zheng，born in 1979，Ph.D，associate professor. His main research interest is cyber securiy.

（责任编辑：喻藜）

关于 2024CCF 会士提名的说明

根据《中国计算机学会会士条例》的规定，2024 年度 CCF 会士候选人提名工作正在进行中。CCF 会士和 CCF 杰出会员具有提名权，主提名人可在 2024 年 11 月 1 日前将"CCF 会士提名表"逐项填妥后提交。

CCF 会士候选人的资格

1. 候选人在提名截止日前(2024 年 11 月 1 日)在计算机或相关领域从业 15 年以上(受高等教育期间的从业时间按如下方式计算：学士 2 年、硕士 4 年、博士 6 年，按最高学历计算，不累计)、本学会会龄 5 年以上，并在计算机及相关领域有重大发明创造及有重要贡献、或对本学会发展有重要贡献的人士。

2. 候选人须得到一名主提名人和二名附议人的提名(附议提名人仅需二名)。

提名人的资格

CCF 会士和杰出会员为有效提名人(如会员资格已失效，则提名无效)。

提名要求

1. 每位提名人作为主提名人提名的会士候选人不得超过 2 人，作为附议提名人不得超过 3 人(即主提名＋附议提名总共 5 人)。

2. 作为主提名人提名时，应保证所提名的会士候选人获得另外 2 名附议提名人的提名。

提名方式

1. 主提名人将提名表以 word 文档格式(切勿用 PDF 格式)通过邮件附件发送到会员部工作人员邮箱(xliu@ccf. org. cn)，主题为：会士提名＋被提名人姓名，邮件需同时抄送两名附议提名人，并请主提名人在邮件正文里署名，工作人员收到邮件回复后视为提交成功。

2. 由主提名人填写提名表后登录 CCF OA 系统提交，附议提名人收到提醒邮件通过链接登录 CCF OA 系统提交意见后完成附议提名。

据 CCF 微信公众号