·体系结构与软件技术· 文章编号:1000-3428(2017)01-0072-07 文献标志码:A 中图分类号:TP18

基于同源性分析的嵌入式设备固件漏洞检测

李 登,尹 青,林 键,吕雪峰

(信息工程大学 数学工程与先进计算国家重点实验室,郑州 450001)

摘 要: 嵌入式设备的制造过程研产分离,导致不同的固件可能包含相同的第三方库,进而相同设备的不同版本甚至是不同设备的固件中都存在大量相同的已公开漏洞。针对该问题,基于第三方库同源性分析提出一种嵌入式设备固件漏洞检测方法,为固件漏洞修复提供参考,减少不必要的重复分析。通过对固件分类,并采用二进制差量分析、字符串常量匹配、模糊哈希 3 种方法分析第三方库同源性,从而检测同类固件中存在的漏洞。实验结果表明,该方法能够有效检测 D-Link 系列路由器固件中的缓冲区溢出和越界漏洞,以及 Linksys 系列路由器固件中的远程命令注入漏洞。

关键词: 嵌入式设备;同源性分析;固件;漏洞;第三方库

中文引用格式: 李 登,尹 青,林 键,等. 基于同源性分析的嵌入式设备固件漏洞检测[J]. 计算机工程,2017,43(1):72-78.

英文引用格式: Li Deng, Yin Qing, Lin Jian, et al. Firmware Vulnerability Detection in Embedded Device Based on Homology Analysis[J]. Computer Engineering, 2017, 43(1):72-78.

Firmware Vulnerability Detection in Embedded Device Based on Homology Analysis

LI Deng, YIN Qing, LIN Jian, LÜ Xuefeng

(State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University, Zhengzhou 450001, China)

【Abstract】 As development and production is separate from each other in the manufacturing process of embedded devices, different firmware may contain the same third-party libraries and thus firmware of different devices or different versions of the same device has a large number of the same vulnerabilities already disclosed. Aiming at this problem, this paper presents a firmware vulnerability detection method in embedded devices based on homology analysis to provide reference for firmware vulnerability remediation and reduce unnecessary repetitious analysis. Vulnerability detection is implemented through firmware classification and homology analysis of third-party libraries by binary differential analysis, string constant matching and fuzzy hashing method. Experimental results show that the proposed method can effectively catch buffer overflows and cross-border vulnerabilities in D-Link router firmware as well as remote command injection vulnerabilities in Linksys router firmware.

【Key words】 embedded device; homology analysis; firmware; vulnerability; third-party libraries

DOI: 10.3969/j.issn.1000-3428.2017.01.013

0 概述

嵌入式设备随处可见,手机、打印机、家用路由器、智能摄像头、网络摄像机、PLC、电脑组件及其外设都是常用的嵌入式设备。物联网技术的兴起更使其被广泛应用,越来越多的嵌入式设备连入网络,其安全问题也日益突出。

嵌入式系统与传统的 PC 机系统不同,其通常由一个称为“固件”的软件组成。固件是指写入 EEPROM 或 Flash 等存储介质中的程序,通俗的理解就是固化的软件。与传统的软件相同,嵌入式设备中的固件也存在缺陷,并且几乎所有设备的固件中都包含漏洞,这主要是由于嵌入式设备制造领域存在“客制化”的生产模式,研产分离。但生产设备

基金项目: 河南省基础与前沿技术研究项目(142300410090)。

作者简介: 李 登(1991—),男,硕士研究生,主研方向为嵌入式设备逆向分析;尹 青,教授;林 键、吕雪峰,硕士研究生。

收稿日期: 2016-01-08 **修回日期:** 2016-02-25 **E-mail:** fatldeng@163.com

时并不将整个系统随之固化,硬件厂商制造硬件设施并附带硬件驱动程序,设备厂商直接从硬件厂商购买硬件或技术方案开发特定功能的开发板。此时,设备厂商有 2 种途径生产设备:1)自己开发固件并进行调试,最后封装出厂交由供应商代购;2)设备厂商只是固化一小段引导程序,而如操作系统、文件系统等则先交由第三方分包商构建,再交给供应商代购。

设备厂商通常选用第 2)种途径,这就出现一个现象:不同的设备厂商可能选择同一个分包商,同一个设备厂商的软件可能由多个分包商开发。由于分包商依赖的开发工具、开发包或者提供的库没有统一标准,不同品牌的设备可能运行相同或者类似的固件以及包含相同的第三方库,同一品牌的设备可能存在多个不同分包商第三方库导致的漏洞,因此,嵌入式设备经常会受到一些已知漏洞的影响,同一个漏洞可能影响不同的设备厂商,甚至是一个更新的固件也会存在已公开的漏洞[1]。这使得对嵌入式设备固件的漏洞检测存在大量的重复性工作,仅针对某个特定型号的设备或某个版本固件的分析无法代表该系列所有固件的安全状态,必须找到一种合理的针对嵌入式设备固件的漏洞分析方法,有效地鉴别固件依赖库以及开发库之间的关联性,确定相似开源代码的同源性,进而对固件出现的漏洞进行较彻底的修复。

针对上述问题,本文对第三方库进行同源性分析,提出一种新的嵌入式设备固件漏洞检测方法。

1　相关理论与技术

嵌入式 Linux 设备固件映像主要包括固件头、引导程序 Bootloader、操作系统内核和文件系统等部分,具体分布情况如图 1 所示。

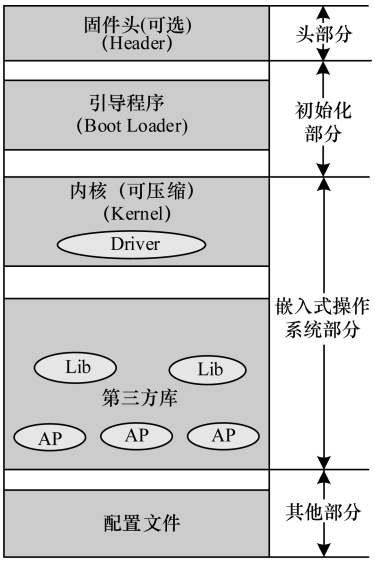

图 1　嵌入式设备固件组成

嵌入式设备中的第三方库主要包括共享库以及通用工具集。同源库是指从某一共同开源代码库祖先经趋异进化过程而形成的不同的共享库或通用软件(开源工具或某一厂商特定的软件集合)。在嵌入式系统中,常用的共享库有 glibc 和 uClibc,它们的动态链接形式保存在"/lib"目录下,文件名常为"libc. so"和"ld-uClibc. so",所有通过交叉编译编写的动态链接程序都会在运行时使用它们。常用的开源工具有集合了嵌入式系统常用命令的轻量工具集 BusyBox、用作 Web 服务器的小型工具 Boa, Httpd 等。

第三方库同源可理解为不同的固件开发库是否源自同一个开源资源库或者是否由同一个第三方开发商、团队编写,以及其是否有内在关联性、相似性。目前针对嵌入式设备固件第三方库同源性分析的研究较少,大都采用传统的漏洞分析方法来检测固件中存在的漏洞,如通过特征匹配方法的固件漏洞检测[2-3]。文献[4]提出面向统一可扩展固件的模糊测试系统,采用模糊测试技术检测固件漏洞。文献[5]通过对固件中远程升级分析控制了一台 HP 打印机,并采用静态分析方法对其中使用的第三方库进行分析,发现超过 80.4% 的固件程序存在 zlib 漏洞。文献[6]通过云计算平台实现大范围的在线固件静态分析,搭建了一个可伸缩性较好的固件分析平台,能够找到固件中的共享凭据(如硬编码的密码)、共享自签名证书以及/etc/passwd 和/etc/shadow 文件中存储的远程登录密码,并实现了一个关联引擎,扫描不同固件镜像之间存在的类似漏洞。文献[7]设计了一个专门解析嵌入式设备固件的二进制分析框架——Firmalice,采用符号执行引擎和程序切片技术,通过构造大量输入数据,检测固件中存在的认证绕过漏洞。文献[8]分析了工业控制系统(ICS)内部协议,在解析 PLC 固件的基础上提出基于数据包的固件漏洞分析方法。文献[9]实现了一个嵌入式设备固件动态分析平台——Avatar,通过注入一种特殊嵌入式设备软件代理,Avatar 能够在 QEMU 模拟器内执行固件指令,同时利用真实的物理硬件处理 I/O 操作。该平台将固件指令转换为 LLVM 中间语言表示,并采用符号执行对固件漏洞进行分析。文献[10]提出一种基于嵌入式固件的库函数识别方法,对无文件系统固件后门进行检测。

现有固件漏洞分析方法以静态分析为主,通过扫描某个固件程序中硬编码的后门标识,或采用符号执行对控制流和数据流进行跟踪,分析对象较为单一,无法对相同序列固件的整体安全性进行综合评估。由于嵌入式设备固件专用于特定的硬件,动态分析不同提供商的嵌入式设备通常需要使用其配

套的调试设备,其中具有代表性的 Avatar 部署过程不仅复杂,而且不同设备固件特定执行环境需要 QEMU 进行模拟仿真,适用性较差。文献[11]通过二进制代码克隆检测分析已发行软件是否违反 GNU 通用公共许可证,通过扫描二进制形式的软件确定其源代码所属的第三方软件包存储库并提取其许可证信息。受其启发,本文通过检测固件程序并进行第三方库的同源性分析,判断固件中是否包含已公开漏洞的第三方库,旨在为固件漏洞修复提供较为完备的参考。

2 第三方库同源性分析

本文的主要研究内容是基于第三方库同源性分析的固件家族归类及漏洞检测,包括固件家族分类、固件解包和第三方库提取、第三方库特征数据库建立以及基于二进制差量分析、字符串常量匹配和模糊哈希的第三方库近似相似性分析,最后根据分析结果对固件安全性作出评估。二进制差量计算和模糊哈希能够较为全面地分析二进制代码的变化部分,对相同架构下固件第三方库的同源性分析效果较好,缺点是依赖于架构。由于 ARM, MIPS, PowerPC 等不同架构的二进制文件差异较大,因此针对跨架构下的同源性分析效果不明显。本文考虑分析的完备性,提出字符串常量匹配方法对跨架构的固件程序进行同源性分析。一般情况下,字符串常量在编译过程中与体系结构无关,相同的源代码经过 ARM,MIPS 交叉编译工具编译,字符串常量往往保存较完整,开源代码中许多字符串在不同版本和重构操作之后几乎不变。因此,本文提出的方法能够较好地适应不同架构之间的第三方库同源性分析。

2.1 固件家族分类

由于嵌入式系统的多样性(包括嵌入式处理器类型的多样性、设备硬件电路设计的多样性、嵌入式操作系统的多样性、应用领域与方式的多样性)和嵌入式设备类型的多样性,固件打包格式差别较大,固件数目更是难计其数。如果对每个固件进行解包来分析其第三方库同源性,耗时较多,可行性也不高。图 2 和图 3 为 D-Link 路由器固件家族 DIR-505 和 DAP-1320 熵值分布图,经过分析发现同一品牌的设备固件第三方库具有较高的同源性,即使大小不一但布局大多相似。这是由于同一品牌固件一般采用相同或相似的固件头部、引导代码、操作系统、文件系统以及压缩算法。为此,本文提出基于信息熵的固件家族分类方法,首先分析整体固件布局的同源性,将熵值图类似的固件归为一类,在此基础上通过对其进行第三方库同源性分析,实现对同一家族固件漏洞的整体检测。

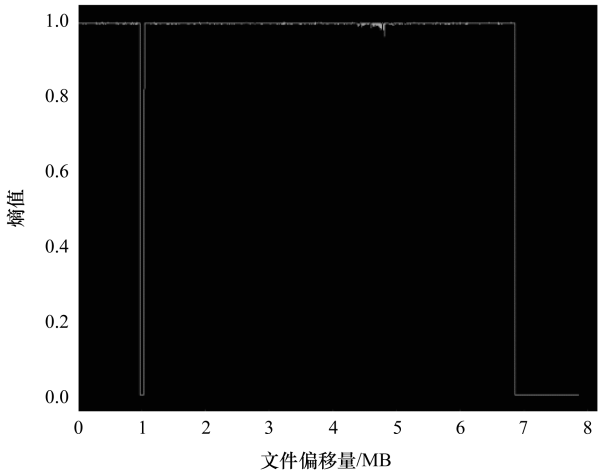

图 2 DIR-505 熵值分布图

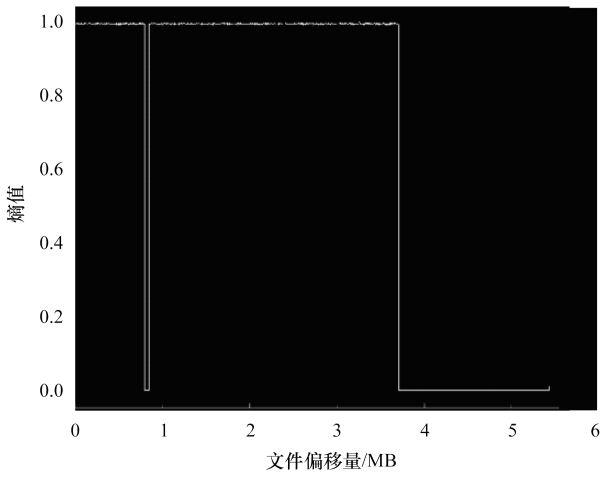

图 3 DAP-1320 熵值分布图

信息熵是一个数学或通信学的概念,它被用来描述信源的不确定性。通常,信源发送什么符号是不确定的,但可以通过该符号出现的概率来度量。概率越大,不确定性越小,蕴含的信息量越小;概率越小,不确定性越大,蕴含的信息量也越大。函数 $f(p) = -\mathrm{lb}p$ 用来描述这种不确定性,其中 p 是某个符号出现的概率。对某个信源而言,通常需要考虑的是该信源所有可能发生情况的平均不确定性,而非单个符号的不确定性。描述信源的平均不确定性即为该信源的信息熵 $H(U)$,用单个符号的统计平均值来表示,计算公式如下:

$$H(U) = E[-\mathrm{lb}p_i] = -\sum_{i=1}^{n} p_i \mathrm{lb}p_i$$

同单个符号表现出的不确定性一样,如果该信源所传递的信息量越大,其信息熵的值也越大;反之越小。

应用信息熵 $H(U)$ 对固件进行熵值统计的方法是:将固件文件分成长度为 M Byte 的数据区段,对每个数据区段进行相应 $H(U)$ 的计算。设字节数为 M 的某段数据区段 Ri,该区段的信息熵为 $H(Ri)$,其中,$i=(fi+li)/2$,fi 为数据区段在固件文件中的偏移首地址,li 为数据区段在固件文件中的偏移末地址。用二元数组 $<H(Ri),i>$ 表示该区段的熵值属性,则固件的所有数据区段构成了熵值属性集合 $SH(R)$。得到固件的熵值属性集合 $SH(R)$ 后,对集合中的每个元素,以 i 为横坐标,$H(Ri)$ 为纵坐标,建立固件熵值属性的二维坐标图。

2.2　二进制差量分析

数据压缩已经在许多归并和分类任务中得到运用,如恶意代码样本分类[12]。提取固件代码中的第三方库,使用关联压缩算法对其与数据库中的资源库进行压缩,如果关联压缩显著小于单个文件压缩的总和,则可认为是同源性的证据之一。该方法也可用来检测不同品牌设备固件的同源性,具体过程为:给定 2 个二进制文件 x 和 y 以及压缩算法 C。如果 $|C(xy)|$ 明显小于 $|C(x)|+|C(y)|$(xy 表示文件 x 和 y 的关联性,$|\cdot|$ 表示以字节为单位的文件大小),计算归一化压缩距离(Normalized Compression Distance,NCD)[13],并使用以下公式来度量第三方库之间的同源程度:

$$ncd(x,y)=1-\frac{|C(xy)|-\min(|C(x)|,|C(y)|)}{\max(|C(x)|,|C(y)|)}$$

其中,$0\leqslant ncd(x,y)\leqslant 1$。$ncd(x,y)=1$ 表示文件 x 和 y 同源;$ncd(x,y)=0$ 表示文件 x 和 y 不同源。

关联压缩算法容易受到文件大小的限制,另一种改进的方法是二进制差分计算。手机设备中 Android 应用程序版本更迭较快,升级较频繁。不断的升级更新应用程序会对用户流量造成损耗。因此,几乎所有 Android 设备都使用差分升级的方式,也称为增量升级,其原理是找出 2 个二进制文件的差异,并将其打成一个差分包,用户升级应用程序时只需要下载更新差分包,再在手机端将差分补丁与需要更新的二进制程序进行组合,得到一个新版本的应用程序。借助该设计思想计算相近固件版本中二进制程序增量,增量较小则可认为是同源性的条件之一。Bsdiff 是一款开源的二进制差量工具,用其计算 2 个二进制文件之间的增量,并使用以下公式衡量二进制 x 和 y 之间的同源性程度:

$$Hom(x,y)=1-\frac{|D(x,y)|}{|D(\in,y)|}$$

其中,$D(x,y)$ 表示 x 和 y 之间的二进制增量;$D(\in,y)$ 表示空文件和 y 之间的二进制增量;$0\leqslant Hom(x,y)<1$。$Hom(x,y)=0$ 表示 x 与 y 的同源程度较大,反之,$Hom(x,y)$ 接近 1 表示 x 与 y 同源程度较小。

本文通过分别计算归一化压缩距离和二进制增量取两者平均值,来衡量 2 个固件程序之间的同源程度。

2.3　字符串常量匹配

上述 2 种方法的局限性在于只能检测相同指令集固件第三方库同源性,下面提出基于字符串常量匹配的同源性分析方法。由于编译器在编译时不会丢弃字符串常量,可将其作为不同架构下第三方库同源性分析中的指纹标识,实现过程为:提取固件包含的二进制程序及依赖库,搜索内部标识字符串、调试信息以及依赖库的名称、版本号等信息,将提取的标识信息与数据库中保存的字符串进行比较,若与某个数据库中第三方库匹配的字符串达到某个阈值,则认为同源。如 BusyBox 是嵌入式设备常用的一个工具集,其可以被字符串常量“BusyBox V”标识。

提取带有漏洞或后门固件的特征字符串,每个字符串都以元组形式存储在数据库中,元组的定义为(*String*, *FirmwareName*, *Version*, *LibraryName*)。使用 Linux 下 strings 命令从二进制文件中提取出字符串,然后与数据库进行匹配,输出匹配的字符串元组。根据匹配的字符元组判断被检测固件中是否包含与漏洞代码同源的二进制程序块。由于第三方库在编译时可能引用其他库函数,因此相同的字符串常量可能会出现在多个第三方库中,本文针对这个特点分 2 种情况进行讨论:在字符串常量唯一的情况下,匹配的评分标准为匹配字符串的长度总和;在非唯一性的情况下,提出一种启发式的匹配方法,对每个字符串的匹配分数进行量化,计算公式为:

$$m=\frac{len(s)}{\mu^{|libs(s)|-1}}$$

其中,$libs(s)$ 表示包含字符串 s 的第三方库的个数;$len(s)$ 表示字符串 s 的长度;μ 为常量,$\mu>1$,通常取为 5,可根据 $len(s)$ 的大小进行调整。

2.4　模糊哈希

模糊哈希算法[14]又称为基于内容分割的分片哈希(Context Triggered Piecewise Hashing,CTPH)算法,可用于文件同源性比较、恶意代码检测、开源软件漏洞挖掘等。该算法使用一个弱哈希计算文件局部内容,在某个条件下切割文件,然后使用一个强哈希对每片文件计算哈希值,取这些值的一部分并连接起

来,与分片条件一起构成多个模糊哈希结果。最后使用字符串相似性对比算法判断 2 个模糊哈希值的相似度,从而判断 2 个文件的相似程度。模糊哈希对文件的部分变化(包括在多处修改、增加、删除部分内容)均能发现与源文件的相似关系,是目前判断同源性较好的一种方法。

本文使用 ssddep 程序计算不同第三方库的模糊哈希值,设置映射区间为 0 ~ 100 的整数对相似程度进行评分,以判断 2 个文件之间的同源关系,100 表示 2 个文件相似性较高,0 则表示没有相似性。

3 固件漏洞检测

嵌入式设备一般使用 GUN 项目开发固件,包括 gcc 编译器编译程序、gdb 调试器调试程序、GUN 开源库源码引用。开发商可能复制不同的开源代码或导入动态链接库嵌入或克隆第三方代码。开源库版本不断发行和更新,漏洞修复会涉及多个版本,旧版本的开源库经常被忽略。嵌入式设备制造领域存在“客制化”的生产模式,如果某个开源库出现漏洞,涉及的设备固件范围较大,修复难度大。考虑稳定性等因素,嵌入式设备使用的开源库往往版本较低,一般晚于新发行版本,一些已公开的开源库漏洞在嵌入式设备固件中不能消除。如 2005 年发现的 zlib 库漏洞,多数嵌入式设备都使用 zlib 库对固件程序进行压缩,很难确定某个厂商特定型号的某个版本固件内嵌了该 zlib 库。

本文提出第三方库同源性分析方法,找到共享代码的开源库群集,使用相关漏洞报告和开源库之间的关系推断固件中存在的漏洞。设 A 和 B 为从固件解包的 2 个第三方库,C 为 GUN 开源库,F_X 表示固件家族 X,F_Y 表示固件家族 Y。A 与 C 同源,记为 $A \approx C$,A 是固件家族 X 的一个库,记为 $A \in F_X$,推断步骤如下:

步骤 1 若 $A \approx C$,$B \approx C$,则每个出现在 C 中的漏洞 V_C,A 和 B 都可能潜在存在。

步骤 2 若 $A \subseteq B$(即 A 内嵌于 B),A 中包含漏洞 V_A,则 B 继承 A 的漏洞 V_A,反之亦然,如图 4 所示。

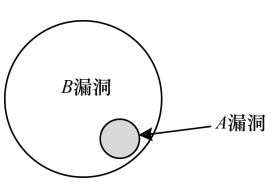

图 4 *B* 继承 *A* 漏洞示意图

步骤 3 若 A 中包含漏洞 V_A,$A \in F_X$,则固件家族 X 视为潜在存在漏洞 V_A;若 $A \approx B$,$B \in F_Y$,则固件家族 Y 视为潜在存在漏洞 V_A。

步骤 4 若 $A \approx C$,$B \approx C$,A 中存在漏洞 V_A,B 中存在漏洞 V_B,V_A 和 V_B 为 CVE 漏洞库中 2 个不同的漏洞。若 A 和 B 为同一架构下的 2 个库且 V_A 和 V_B 发现的时间不超过 2 个月,认为 V_A 和 V_B 为同一个漏洞,如图 5 所示。

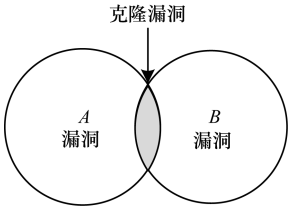

图 5 *B* 克隆 *A* 漏洞示意图

4 实验与结果分析

当一款路由器的漏洞被公布后,厂商通常不会在短时间内给出固件的更新版本,而且很少有人会去更新路由器固件,因此,路由器漏洞的危害更大。本文对 D-Link 路由器家族出现的缓冲区溢出漏洞和越界漏洞,以及 Linksys 系列路由器中频繁出现的 Web 服务器程序 HTTPD 的 apply. cgi 处理脚本漏洞进行同源性分析,实现其同序列固件漏洞检测。

4.1 D-Link 缓冲区溢出漏洞

D-Link 官方安全公告[15]公布 D-Link DIR-645 路由器存在缓冲区溢出,该漏洞存在于名为“cgibin”的网络处理程序中,未认证的攻击者传递一个超长的 Cookie 就可使程序堆栈溢出,从而获得路由器远程控制权限。从漏洞公告中得知,漏洞产生的原因是 Cookie 的值超长。通过函数 char * getenv(“HTTP_COOKIE”)函数可以获取用户输入的 Cookie 值,在 IDA 中通过搜索“HTTP_COOKIE”可以定位漏洞发生的位置,漏洞形式如图 6 所示。

```
.text:00409644          nop
.text:00409648          jalr    $t9 ; sess_get_uid
.text:0040964C          move    $a0, $s5
.text:00409650          lw      $gp, 0x4E8+var_4D8($sp)
.text:00409654          nop
.text:00409658          la      $t9, sobj_get_string
.text:0040965C          nop
.text:00409660          jalr    $t9 ; sobj_get_string
.text:00409664          move    $a0, $s5
.text:00409668          lw      $gp, 0x4E8+var_4D8($sp)
.text:0040966C          lui     $a1, 0x42
.text:00409670          la      $t9, sprintf
.text:00409674          move    $a3, $v0
.text:00409678          move    $a2, $s2
.text:0040967C          la      $a1, aSSPostxml  # "%s/%s/postxml"
.text:00409680          jalr    $t9 ; sprintf
.text:00409684          move    $a0, $s1         # s
```

图 6 D-Link DIR-645 缓冲区溢出漏洞

解包分析 19 个不同型号 D-Link 系列路由器固件发现,12 个固件中包含“cgibin”程序,占 63. 15% 。将 DIR-815,DIR-817LW,DIR-818W,DIR-803,DCHM-225,DIR-820L,DIR-890L,DIR-885,DIR-880L 固件中的“cgibin”程序分别与 DIR-645 中的“cgibin”进行同源性分析,结果如表 1 所示,可以看出,同一架构下,

DIR-815,DIR-817LW,DIR-818W 中的“cgibin”程序存在缓冲区漏洞的可能性最大;不同架构下,DIR-890L,DIR-880L 路由器字符串匹配分数较高,可能存在漏洞。通过官方公布的漏洞报告可验证,本文同源性分析方法基本能够正确识别存在漏洞的固件程序。

表 1　D-Link 各路由器缓冲区溢出漏洞同源性分析结果

路由器型号	版本	架构	二进制差量	字符串匹配	模糊哈希
DIR-815	V1.01	MIPS	0.114 1	160	67
DIR-817LW	V1.02	MIPS	0.202 3	124	75
DIR-818W	V1.01	MIPS	0.263 4	124	58
DIR-820L	V1.04	MIPS	0.454 1	68	37
DIR-803	V1.01	MIPS	0.771 2	68	22
DCHM-225	V1.08	MIPS	0.404 8	134	35
DIR-890L	V1.03	ARM	0.886 5	146	9
DIR-885	V1.00	ARM	0.648 0	49	11
DIR-880L	V1.01	ARM	0.892 1	153	12

4.2　D-Link 越界漏洞

D-Link 官方安全公告[16]公布 D-Link DIR-505 路由器存在可被利用的漏洞。该漏洞存在于名为“my_cgi. cgi”的 CGI 脚本中,造成漏洞的原因是在目的缓冲区和源缓冲区之间以字节为单位循环赋值转储时,对边界验证不合理导致程序越界访问源缓冲区,最终造成缓冲区溢出,漏洞形式如图 7 所示。溢出发生后,攻击者可以获得路由器的远程控制权。

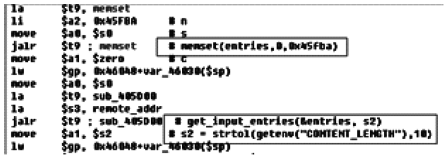

图 7　D-Link DIR-505 越界漏洞

以 DIR505A1_FW108B10. bin 固件为参照,解包 DIR-505 路由器多个版本固件,分析其“my_cgi. cgi”脚本同源性,结果如表 2 所示。

表 2　D-Link DIR-505 越界漏洞同源性分析结果

版本	架构	二进制差量	字符串匹配	模糊哈希
V1.01	MIPS	0.114 1	170	87
V1.02	MIPS	0.202 3	154	87
V1.03	MIPS	0.213 4	154	84
V1.05	MIPS	0.254 1	168	79
V1.06	MIPS	0.231 2	168	83
V1.07	MIPS	0.104 8	164	88

可以看出,DIR-505 路由器固件中的“my_cgi. cgi”脚本同源性较高,存在漏洞的可能性较大,与官方公布 D-Link DIR-505 1.01 ~ 1.08 版本受越界漏洞影响的情况基本相同。

4.3　Linksys Web 服务器漏洞

在 Linksys E1500/E2500/WRT160nv2/WRT54GL 和 Linksys WRT54G 路由器固件 Web 服务器 HTTPD 的 apply. cgi 处理脚本中,分别曝出包括远程命令注入、缓冲区溢出漏洞。命令注入漏洞源于执行 Ping 命令时没有对 ping_size 参数输入进行验证,受影响的固件有 E1500 1. 0. 00/1. 0. 04/1. 0. 05,E2500 1.0.03,WRT160nv2 2. 0/2. 03。缓冲区溢出漏洞源于对发送的 POST 请求没有设置足够的边界与内容长度检查,当未经认证的远程攻击者向路由器的 apply. cgi 页面发送内容长度大于 10 000 Byte 的 POST 请求时,就会触发缓冲区溢出,受影响的固件有 WRT54G 3.01.03/3.03.6/4.00.7 等。

解包固件提取出 HTTPD 服务器程序的 apply. cgi 脚本二进制代码,命令注入漏洞以 E1500 1.0 为参考,缓冲区溢出漏洞以 WRT54G 4.00.7 为参考,对 apply. cgi 程序进行同源性分析,结果如表 3 所示。可以看出,除 WRT160nv2 2.03 误差较大外,其他大致符合官方公布的漏洞报告。

表 3　Linksys Web 服务器漏洞同源性分析结果

路由器型号	版本	架构	二进制差量	字符串匹配	模糊哈希
E1500	V1.04	MIPS	0.143 2	264	46
E1500	V1.05	MIPS	0.260 4	267	28
E2500	V1.03	MIPS	0.154 6	264	57
WRT160nv2	V2.0	MIPS	0.357 3	248	52
WRT160nv2	V2.03	MIPS	0.527 9	219	18
WRT54G	V3.01	MIPS	0.104 8	198	35
WRT54G	V3.03	MIPS	0.176 5	146	49
WRT54G	V4.20	MIPS	0.623 5	178	11

5　结束语

现有固件安全性检测方法包括完整性检测、特征码识别、符号执行、敏感函数提取等,这些方法对小数量级的固件漏洞检测具有较好的效果。但随着智能设备的多样化,嵌入式设备固件种类急剧上升,版本更新速度加快,因此,需要找到一种能够适应一定数量级的固件漏洞检测方法。本文从嵌入式设备固件开发产业链进行考虑,针对嵌入式设备固件开

发过程中研产分离,一般采用开源代码进行二次或者多次开发,以及同一个漏洞重复出现在同一厂商或不同厂商固件程序中的特点,通过熵值分布图对固件进行分类,然后解包固件,提取其第三方库,并采用二进制差量分析、字符串常量匹配和模糊哈希分析其同源性,最后基于第三方库同源性分析结果对同类固件进行漏洞检测。实验结果表明,该方法能够准确地对同一品牌不同类型、不同架构、不同版本的嵌入式设备固件进行漏洞检测,具有一定的应用价值。

但本文提出的方法尚存不足,如对 Linksys WRT160nv2 2.03 固件的同源性分析结果存在一定误差。因此,下一步将实现具有一定启发性和关联分析特性的方法,搭建某些关键设备的动态仿真平台,结合动态分析方法开发更精确的嵌入式设备固件漏洞检测平台。

参考文献

[1] Heffner C, Yap D. Hacking the Routers: SOHO Router Security[J]. SourceSec Security Research, 2014, 55(2): 76-82.

[2] 付思源,刘功申,李建华. 基于 UEFI 固件的恶意代码防范技术研究[J]. 计算机工程, 2012, 38(9): 117-120.

[3] 周振柳. 计算机固件安全技术[M]. 北京:清华大学出版社, 2012.

[4] 马佳敏,潘 理,姚颉文. 面向新一代固件接口标准的固件模糊测试系统[J]. 计算机工程, 2014, 40(7): 277-280.

[5] Cui Ang, Costello M. When Firmware Modifications Attack: A Case Study of Embedded Exploitation[C]// Proceedings of Network and Distributed System Security Symposium. San Diego, USA: NDSS Press, 2013: 134-141.

[6] Costin A, Zaddach J, Francillon A, et al. A Large-scale Analysis of the Security of Embedded Firmwares[C]// Proceedings of USENIX Security Symposium. San Diego, USA: USENIX Association, 2014: 95-110.

[7] Shoshitaishvili Y, Wang Ruoyu, Hauser C, et al. Firmalice-automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware[C]//Proceedings of Network and Distributed System Security Symposium. San Diego, USA: NDSS Press, 2015: 145-156.

[8] Mulder J, Schwartz M, Berg M, et al. Analysis of Field Devices Used in Industrial Control Systems[M]. Berlin, Germany: Springer, 2012: 45-57.

[9] Zaddach J, Bruno L, Francillon A, et al. AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares[C]//Proceedings of Network and Distributed System Security Symposium. San Diego, USA: NDSS Press, 2014: 167-173.

[10] 忽朝俭,薛一波,赵 粮,等. 无文件系统嵌入式固件后门检测[J]. 通信学报, 2013, 34(8): 140-145.

[11] Hemel A, Kalleberg K T, Vermaas R, et al. Finding Software License Violations Through Binary Code Clone Detection[C]//Proceedings of the 8th Working Conference on Mining Software Repositories. New York, USA: ACM Press, 2011: 63-72.

[12] Bailey M, Oberheide J, Andersen J, et al. Automated Classification and Analysis of Internet Malware[M]// Kruegel C, Lippmann R, Clark A. Recent Advances in Intrusion Detection. Berlin, Germany: Springer, 2007: 178-197.

[13] Cilibrasi R, Clustering by Compression[J]. IEEE Transactions on Information Theory, 2005, 51(4): 1523-1545.

[14] Kornblum J. Identifying Almost Identical Files Using Context Triggered Piecewise Hashing[J]. Digital Investigation, 2006, 3(3): 91-97.

[15] Huntley S. Dlink DIR-645 UPNP Buffer Overflow[EB/OL]. [2016-01-07]. http://securityadvisories.dlink.com/security/publication.aspx?name=SAP10008.

[16] di Pinto A. Multiple Vulnerabilities on D-Link Dir-505 Devices[EB/OL]. [2016-01-07]. http://securityadvisories.dlink.com/security/publication.aspx?name=SAP10029.

编辑 金胡考