

面向物联网设备固件的硬编码漏洞检测方法

穆超¹, 王鑫¹, 杨明¹, 张恒², 陈振娅¹, 吴晓明¹

(1. 齐鲁工业大学(山东省科学院)山东省计算中心(国家超级计算济南中心)山东省计算机网络重点实验室, 山东 济南 250014;

2. 江苏海洋大学计算机工程学院, 江苏 连云港 222005)

摘 要: 随着物联网设备的普及, 越来越多有价值的数据产生, 依托物联网设备进行大数据分析和挖掘是近年来学术界和工业界关注的热点问题。然而, 由于缺乏必要的检测和防护手段, 很多物联网设备存在严重的信息安全隐患。特别地, 设备硬编码信息与系统加解密、身份认证等功能密切相关, 可为核心数据提供机密性保障, 一旦被恶意攻击者利用, 会产生敏感信息泄露、后门攻击、非授权登录等严重后果。针对该问题, 在研究物联网设备中硬编码漏洞表现特征的基础上, 提出了一种可执行文件中多类型字符识别定位和硬编码漏洞检测方法。首先, 提取固件内容并筛选所有可执行文件作为待分析源, 提出特殊格式字符、外部文件引用、密码实现3类硬编码字符的识别与定位; 然后, 根据函数调用关系对硬编码字符所在函数进行可达性分析, 采用中间表示IR模型消除指令异构性, 并利用数据流分析方法确定字符型和参数型硬编码值; 最后, 设计符号执行方法确定硬编码漏洞的触发条件, 最终输出漏洞检测结果。一方面, 所提方法在利用中间表示模型的基础上引入了符号执行的方法, 消除了指令架构依赖性, 减少了漏洞误报率; 另一方面, 该方法可融合字符、文件、密码实现3类硬编码字符的不同特征表现, 增加了漏洞检测的覆盖范围, 提升了检测方法的通用性。实验结果表明, 所提方法可有效检测多种物联网设备中的字符、文件、密码3类硬编码漏洞, 具有较好的检测精度, 可为后续安全防护技术的部署提供一定指导。

关键词: 大数据; 物联网安全; 硬编码; 漏洞检测

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2022070

Hardcoded vulnerability detection approach for IoT device firmware

MU Chao¹, WANG Xin¹, YANG Ming¹, ZHANG Heng², CHEN Zhenya¹, WU Xiaoming¹

1. Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan),

收稿日期: 2022-06-10; 修回日期: 2022-09-02

通信作者: 杨明, yangm@sdas.org

基金项目: 国家重点研发计划(2021YFF0901301-02); 山东省自然科学基金(ZR2021QF057); 国家自然科学基金(61873106); 江苏省杰出青年科学基金(BK20200049); 齐鲁工业大学(山东省科学院)计算机科学与技术学科基础研究加强计划(2021JC02023); 工业控制技术国家重点实验室(浙江大学)开放课题(ICT2022B36)

Foundation Items: The National Key R&D Program of China (2021YFF0901301-02), Shandong Provincial Natural Science Foundation (ZR2021QF057), The National Natural Science Foundation of China (61873106), The Natural Science Foundation of Jiangsu Province for Distinguished Young Scholars (BK20200049), QLU/SDAS Computer Science and Technology Fundamental Research Enhancement Program (2021JC02023), Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (ICT2022B36)

引用格式: 穆超, 王鑫, 杨明, 等. 面向物联网设备固件的硬编码漏洞检测方法[J]. 网络与信息安全学报, 2022, 8(5): 98-110.

Citation Format: MU C, WANG X, YANG M, et al. Hardcoded vulnerability detection method for firmware of IoT devices [J]. Chinese Journal of Network and Information Security, 2022, 8(5): 98-110.

Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China
2. School of Computer Engineering, Jiangsu Ocean University, Lianyungang 222005, China

Abstract: With the popularization of IoT devices, more and more valuable data is generated. Analyzing and mining big data based on IoT devices has become a hot topic in the academic and industrial circles in recent years. However, due to the lack of necessary detection and protection methods, many IoT devices have serious information security risks. In particular, device hard-coded information is closely related to system encryption and decryption, identity authentication and other functions, which can provide confidentiality protection for core data. Once this information is exploited by malicious attackers, serious consequences such as sensitive information leakage, backdoor attacks, and unauthorized logins will occur. In response to this problem, a multi-type character recognition and positioning scheme was designed and a hard-coded vulnerability detection method in executable files was proposed based on the study of the characteristics of hard-coded vulnerabilities in IoT devices. The proposed method extracted the firmware of IoT devices and filtered all executable files as the source to be analyzed. Then, a solution to identify and locate three types of hard-coded characters was provided. Further, the reachability of the function, where the hard-coded character was located, was analyzed according to the function call relationship. Meanwhile, the instruction heterogeneity was mitigated by an intermediate representation (IR) model. The character and parameter hard-coded values was obtained through a data flow analysis approach. A symbolic execution method was devised to determine the trigger conditions of the hard-coded vulnerabilities, and then the vulnerability detection result was output. On the one hand, the proposed method introduced the method of symbolic execution based on the use of the intermediate representation model, which eliminated the dependency of instruction architecture and reduces the false positive rate of vulnerabilities; On the other hand, this method can integrate characters, files, and cryptographic implementation to realize the different characteristics of three types of hard-coded characters, which increased the coverage of vulnerability detection and improves the versatility of the detection method. The experimental results show that the proposed method can effectively detect three types of hard-coded vulnerabilities of characters, files and cryptographic implementation in various IoT devices, and has good detection accuracy, which can provide certain guidance for the deployment of subsequent security protection technologies.

Keywords: big data, IoT security, hard coding, vulnerability detection

0 引言

随着大数据技术的迅猛发展，以数据为依托的大数据应用技术不断完善，同时大数据带来的隐私泄露等安全问题日益突出，数据的采集、传输、存储、应用等各个环节都需要进行数据安全的保障工作。在数据采集阶段，物联网设备拥有丰富的传感器和感知器，是获取各项指标、属性等重要参数的数据终端，亦是大数据应用的数据来源^[1]。因此，为获得真实可靠的数据资源，研究物联网终端设备安全具有重要意义。

然而，物联网设备生产厂商急于推出产品抢占市场，缺乏系统上线前产品的安全检测，导致物联网产品存在大量的漏洞；另外，厂商产品中缺

少安全防护模块，难以应对不断演化的恶意攻击行为。进一步，规模庞大的物联网设备暴露在公网中，这些设备存储着海量的用户数据，一旦被恶意攻击者控制，会导致严重的用户隐私泄露，从而引发更严重的网络攻击。例如，在2016年的Mirai僵尸网络攻击事件^[2]中，攻击者攻陷了数以万计的物联网设备，并以此为节点发动分布式拒绝服务（DDoS, distributed denial of service）攻击，最高攻击峰值达到1 Tbit/s，造成大量主流网站瘫痪。

近几年，研究者在物联网安全领域做了大量工作^[3-5]，主要研究方向包括物联网设备通信协议缺陷分析^[6-8]、物联网云平台安全^[9-11]和物联网固件漏洞检测^[12-16]，重点研究了内存泄露、代码注入、认证绕过、信息泄露等安全问题。在这些工作中，针对物联网设备中可执行文件的硬编码漏

洞研究还处于初级阶段。硬编码与系统的授权认证、密码保护等模块有重要关系,这些模块中包含保障物联网安全的核心组件,提供数据加解密、数字证书应用和身份认证等功能,对于保证物联网设备核心数据的机密性、保障关键信息的完整性和可用性至关重要。若这些组件本身存在安全问题,一旦被攻击者利用,会引发严重的安全事故,威胁人身财产安全。例如, F-Secure 公司的研究人员发现在 Foscam C2 和 Opticam I5 的网络摄像头中,存在多处硬编码漏洞^[17],可导致非授权用户远程访问,造成持续性的网络攻击。

针对硬编码漏洞的检测,通常采用漏洞特征提取与静态分析结合的技术。在现有研究中, Thomas 等^[18]提出了一种静态分析方法测量单个静态数据对固件中二进制文件控制流程的影响,识别二进制文件中的静态数据对比函数,并使用静态数据序列集标记每个函数的基础块和可达性,从而确定硬编码身份认证后门。该方案测试样本较少,无法保证在大规模测试中的适用性。 Shoshitaishvili 等^[19]研发了一种支持物联网设备运行和分析的工具 Firmalice,基于符号执行引擎 Angr^[20]和程序切片技术增加漏洞分析的可扩展性,根据攻击者的输入控制能力检测硬编码型的认证绕过漏洞,但该工具需要人工设置安全策略,对先验知识有较强的依赖性。

在现有针对硬编码漏洞检测的研究工作中,大多考虑特定的物联网设备,存在样本数量较少的问题,且在特定格式敏感字符与硬编码外部文件引用的检测方面存在较大不足,缺乏密码模块硬编码漏洞检测的深入研究。现有安全事件表明,某些特定格式字符(如 WPS PIN 码、IP 地址、邮箱)被攻击者获知,可帮助其构造更复杂更高级的攻击方案;密钥硬编码文件的泄露可造成设备的授权绕过、加密数据被解密、协议信道被攻破等问题;密码模块是物联网设备安全建设的重要支柱,也是漏洞检测的重中之重,而现有针对密码理论和密码实现中的硬编码漏洞研究较欠缺,存在严重的安全隐患。因此,亟须开展针对物联网设备硬编码漏洞的安全检测研究。

本文提出一种面向物联网设备固件的硬编码漏洞检测方法,可以自动化实现硬编码字符识别

和硬编码漏洞检测。首先,获取和解析物联网固件文件,并筛选所有可执行文件。本文考虑 3 种硬编码型漏洞,对特定格式字符串、外部引用文件和密码实现中的硬编码字符进行识别和定位,结合函数调用关系进行函数可达性分析,采用中间表示(IR, intermediate representation)形式模型解决固件异构问题,生成核心函数控制流程图(CFG, control flow graph),并利用符号执行引擎进行约束求解,最终获得硬编码漏洞触发条件。

本文的主要贡献如下。

- 1) 提出一种针对硬编码漏洞的检测方法,识别和定位特定硬编码字符,判断函数调用可达性,定位敏感函数中关键参数的数据流和分析符号执行,实现硬编码漏洞的精准识别和检测。
- 2) 设计并构建一种面向物联网设备固件的硬编码漏洞检测系统,实现 3 种类型的硬编码漏洞检测(可导致安全风险的特殊格式硬编码字符、敏感文件的外部引用、使用硬编码参数的不安全密码函数实现),该系统具备良好的可扩展性和适用性。
- 3) 基于构建的硬编码漏洞检测系统,对多类真实物联网设备固件进行了处理和安全性分析,成功地在多个固件中检测出硬编码漏洞。

1 相关工作

目前针对物联网设备固件漏洞的检测与分析已有大量的研究工作,其中与本文研究内容相关的研究方案主要包括动态分析和静态分析。

动态分析指通过将固件程序加载到实际设备或仿真环境中,融合模糊测试等技术生成测试用例,并根据程序实时运行状态来分析和挖掘漏洞的技术。FIRM-AFL^[21]是目前主流的物联网固件仿真工具,可以实现大量基于 Linux 内核的固件全系统仿真,但支持的系统架构有限,且无法支持实时操作系统。Avatar^[22]实现了软硬件混合的仿真系统,通过连接模拟器和真实硬件完成指令操作,但软硬件之间的频繁交互使得仿真时效性较差。IoTFuzzer^[23]通过分析物联网设备的 App 来挖掘内存漏洞,但仅局限于分析有对应 App 的物联网设备,且代码覆盖率不高。总之,动态分

析受制于物联网设备有限的存储和计算资源，以及需要适配复杂异构的硬件环境，这增加了动态分析的难度。

静态分析的分析对象是程序源码和二进制文件，通过逆向提取固件文件系统，搭建检测模型，结合程序分析技术等实现挖掘漏洞。Karonte^[24]首先提取二进制文件中的用户请求模块，追踪进程间通信数据流的传播过程并建立二进制依赖图(BDG, binary dependency graph)，采用跨文件污点分析方法检测不安全的数据流。SaTC^[25]通过Web服务的前端搜索用户输入的关键字，并根据该数据流定位处理输入的后端代码，利用敏感字符污点分析来发现非可信输入和危险应用。上述研究主要针对用户输入的安全检测，却忽略了系统中的硬编码会存在安全威胁。Mu等^[26]采用函数特征提取以及改进的指令统计方法来识别物联网中的密码函数，并通过调用关系图分析函数应用场景，该研究主要集中在对物联网固件中密码函数的识别，并未检测密码函数实现中是否存在缺陷。FIE^[27]通过符号化思维，建立终端统一描述进行符号执行分析，可提升分析覆盖率，但对源码有较强的依赖性，且未考虑密码实现中的安全问题。

2 背景知识

本节介绍关于物联网设备固件中硬编码字符识别和硬编码漏洞检测的背景知识。

2.1 中间表示

程序源代码需经编译器编译为二进制可执行文件，该过程与所选编译器、编译配置选项、编译优化选项等相关；由于物联网设备存在不同的系统架构，编译生成的二进制可执行文件具有完全不同的汇编代码形式，增加了调试分析程序的难度。为解决不同编译选项、不同系统架构所带来的异构难题，选择将二进制可执行文件反编译后进行中间表示转换：将不同架构汇编代码统一转化为同一种中间语言，通过对中间语言表示的优化和分析，克服异构分析协调的挑战。

对二进制可执行文件的分析主要步骤有：解码文件指令，翻译为带语义的特定格式语言。针对特定系统架构的解码器在该架构下拥有详细的

指令系统，但其具有较强的指令依赖性，难以移植；可支持多处理器体系架构的通用解码器，其对指令的表示和描述具有一致性。

本文选择Angr工具中基于Valgrind^[28]实现的PyVEX模块构建中间表示，其支持对多种架构的抽象，并具有丰富的API接口，包括寄存器、堆栈结构、常量、算术运算与逻辑运算等指令和结构的分析和调用。

2.2 数据流分析

本文采用数据流分析方法，定位已转为中间语言的待检函数参数等数据，对参数在执行过程中的赋值与调用等操作进行分析，以检测硬编码型缺陷。

数据流分析是一种针对数据在程序执行路径上的分析溯源技术，用于分析数据在程序执行过程中的流动方向和取值变化。在数据流分析中，依据不同的程序路径分析精度，语句的执行顺序可以分为流敏感性分析和流不敏感性分析；考虑程序路径的深度，根据程序中函数代码的数据传递位置不同分为过程内分析和过程间分析。

利用数据流分析规则，并按照某种路径执行顺序分析代码的中间表示过程，是检测漏洞的有效手段。按照程序执行过程自底向上、从右至左的分析顺序，在数据流分析中，与数据产生和传输直接相关且最重要的语句是赋值语句及过程调用语句。赋值语句用于改变变量的取值，且在同一函数调用中可能会发生多次变化，需要根据检测规则标注和监控变量取值的变化过程，同时观测运行至控制转移语句时的路径条件。对于过程调用语句，需根据函数的名称和定义识别各参数的语义信息，并依据函数调用关系图分析参数在各函数调用时的传递过程及赋值变化。

分析过程内数据流时，按照程序的存储位置以及程序控制流程图确定语句执行顺序。对程序流程图进行广度优先或深度优先遍历时，在基本块上保留部分分析结果，并采用前置条件或后置条件进行基本块摘要运算。分析过程间数据流时，首先保存当前执行程序段的数据环境，然后对调用过程的内部代码进行分析，并保存返回时的参数值。

2.3 符号执行

本文利用 Angr 工具的符号执行模型,对数据流分析定位的参数进行约束求解,以此生成运行至该参数的路径限制条件,从而明确硬编码漏洞的触发条件。

符号执行是一种静态分析方法,将程序执行模拟为符号表达式,程序的输出结果表示为符号之间的逻辑和算术运算,对程序流程的分析转化为对表达式的求解。符号执行根据程序的执行过程可以分为针对单个函数的过程内分析,以及考虑函数调用过程和环境信息的过程间分析。

符号执行通常用于程序全局分析和代码安全性检测,首先解析程序代码并生成中间表示,构建程序路径控制流图和调用图等,借助符号表示式完成程序执行流程的覆盖,之后进行约束求解产生测试用例,根据满足条件判断路径可达性。检查变量和符号的取值是否满足相应的约束条件或符合特定的漏洞特征,进而深入该路径分析程序的执行缺陷中。当遇到循环、迭代、递归调用时,符号执行易产生路径爆炸,采用增加限制条件来缓解分析路径的数量。

2.4 密码函数实现中的硬编码漏洞

在工程实现时,密码算法可能会出现硬编码的漏洞,通常这种硬编码以初始值、参数等形式出现。例如,在源代码中,设置关键参数为常量,编译生成的可执行文件会携带这种硬编码特征;

在可执行文件调用密码函数进行参数赋值时,安全状态下这些参数来自用户输入或系统随机生成,但由于程序硬编码漏洞的存在,相关参数可能会被直接赋值为静态常量。以 RSA 算法中的 Padding 值以及 CBC 模式中的初始化向量 (IV, initialization vector) 为例,若这些参数实现不当,则密码算法可能存在硬编码漏洞。根据 CWE-780 对该漏洞的官方描述,当 RSA 算法未采用最优非对称加密填充 (OAEP, optimal asymmetric encryption padding) 时,这些弱安全填充模式使用非随机字节,使得明文和密文之间具有映射性,进而导致明文更容易被推测。在采用 CBC 模式的密码算法中,加解密需利用一个初始 IV 值与第一个块进行异或,并将异或结果作为下一个块的 IV,若初始 IV 值设定不当,则可能受 BEAST 攻击^[29]。

3 系统设计

系统架构如图 1 所示,主要分为 3 个模块。

- 1) 预处理阶段:提取物联网识别固件文件,并筛选特定格式文件。
- 2) 硬编码字符的识别和定位阶段:对可执行文件进行逆向分析,提取所有的字符,基于预设硬编码检测方案识别满足条件的硬编码字符,并在程序中进行定位。
- 3) 硬编码漏洞检测:获取硬编码字符调用函数,验证该调用函数的可达性,并对关键参数进

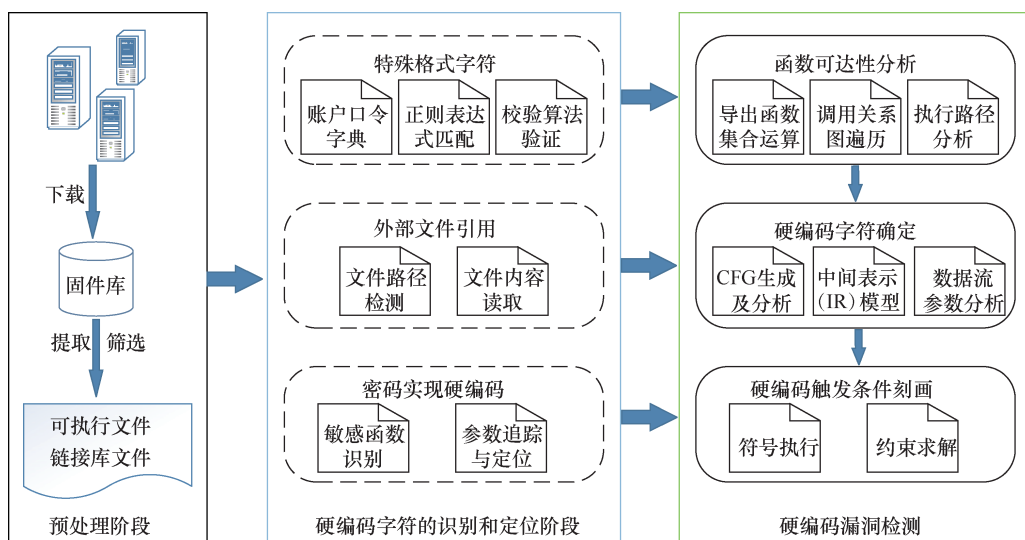


图 1 系统架构

Figure 1 System architecture

行硬编码检测。

3.1 预处理阶段

本文基于开源固件分析工具 Firmadyne^[30]开发爬虫进行固件资源下载,采用 Binwalk 和 Firmware Mod Kit 工具对 MIPS、ARM、PowerPC 等架构固件进行提取。根据文件类型筛选可执行文件和链接库文件,如 ELF 文件、SO 文件、DLL 文件、EXE 文件等。

3.2 硬编码字符的识别和定位

固件文件中存储着大量的硬编码字符(如注释信息、版本信息、打印信息、调试信息、程序中定义的常量),其中,一些字符是程序执行的辅助信息,可以被公开访问,不具备敏感性,如程序注释;另一些字符与程序的重要功能有关,具有一定敏感性,如登录账户信息。

如何界定硬编码字符的敏感性是本文面临的第一个挑战问题。从攻击者的角度出发,根据硬编码字符的泄露是否对程序运行结果产生影响、是否会进一步泄露更多的系统信息、是否会引发已知的漏洞等,将字符分为敏感字符和普通字符。

根据上述分类依据,结合敏感字符的应用场景,重点研究3类敏感字符的硬编码漏洞检测方法,并根据每类漏洞设计识别方案和模型。

3.2.1 特殊格式字符

该类漏洞的硬编码字符通常直接以字符串格式存储,字符内容具有规律性,可以使用正则表达式、字符特征匹配等方法进行识别。例如,WPS PIN 码字符,是 QSS 快速连接功能的识别码,可以代替 Wi-Fi 密码进行认证,识别后可接入网络,若某款路由器的 WPS PIN 码存在硬编码漏洞,一旦泄露会导致网络的非授权连接。PIN 码通常是满足校验规则的 8 位数字组合,具有明显的识别特征。同样地,IP 地址、邮箱地址、产品序列号、登录账号等硬编码字符都具有各自的特征,这些字符与产品的敏感信息、授权认证和版权保护有直接关联,是需要重点检测的对象。

本文提出一种固件可执行文件中特殊格式字符的定制化识别方法,支持用户添加自定义新规则,具备较强的系统扩展性。下面介绍几类硬编码字符的识别方法。

① 账户口令字符:设置常用账号口令字典实现匹配,由于本文不考虑爆破攻击的情形,因而所选字典无须规模过大。采用 top1000 字典进行匹配,同时提取口令所在的函数名称,对函数名进行口令的模糊匹配。

② IP 地址:采用正则表达式初步筛选字符,并将无效 IP 地址、广播 IP 地址、保留 IP 地址等去除,保留内网 IP 和公网 IP 等有效 IP 地址。

③ 邮箱地址:采用正则表达式进行字符初选,利用 EmailVerify^[28]等站点对邮箱进行有效性校验,保留合法合规的邮箱地址。

④ WPS PIN 码:利用正则表达式获取 8 位连续的纯数字,基于 checksum 校验算法获得符合校验的疑似 PIN 码。

⑤ 编码和 Hash 字符:利用正则表达式获取特定长度或带明显编码特征的字符,如 base64 编码、URL 编码、MD5 Hash,并对其尝试进行解码。

3.2.2 外部文件引用

程序执行会经常调用外部文件,如配置文件、源码文件,若一些高敏感文件(如密钥文件、证书文件)遭到泄露,则攻击者有更大概率解密出网络流量、登录认证过程等敏感信息,从而严重威胁系统安全。针对此类硬编码漏洞,首先检测函数中出现的路径和文件,并查看该文件是否真实存在,若存在,则识别文件类型和读取文件内容。管理并维护一个敏感文件列表白名单,主要包含以下几类。

① 账户文件: /etc/passwd、/etc/shadow 等账户相关文件。

② 数据库文件: .db、.mdb、.bak 等数据库相关文件。

③ 数字证书类文件: .key、.pem、.der、.cer 等与数字证书和密钥相关的文件。

④ 配置文件: .ini、.config 等文件。

3.2.3 密码实现硬编码

在实现密码算法时会引入数字、字符常量等硬编码,这些常量一般通过参数赋值等方式进行传递,若这些参数配置不当,则攻击者可利用密码实现上的缺陷实施破坏行为。

此类硬编码与特殊格式字符与外部文件两类硬编码不同,需要先定位函数,然后分析

函数中的参数是否存在硬编码, 本文主要检测两类密码实现的硬编码问题: 带 Padding 参数的 RSA 算法; 带 IV 值参数的 CBC 模式算法。

3.3 硬编码漏洞检测

经过以上步骤, 可以获取硬编码字符的位置和其调用函数, 仅凭这些无法断定是否存在硬编码漏洞以及该漏洞是否可被执行, 需进一步对调用函数的可达性、硬编码字符具体值以及运行至该硬编码的程序执行条件等进行全面分析, 并根据以上结果综合判定硬编码漏洞情况。

3.3.1 函数可达性分析

源代码经过预处理、编译、汇编等过程变为可执行文件, 在编译过程中, 编译器通常会对源码进行优化处理, 移除无用代码和死代码。由于编译器的优化能力和配置选项存在较大差别, 为防止可执行文件中出现未被调用函数的情况, 需对二进制可执行文件中的所有函数做调用可达性分析, 剔除未被调用的函数。

首先, 获取可执行文件中所有的导出函数, 并生成一个集合 E , 在集合 E 中找到文件的程序入口函数, 如 `main`、`entry`、`start` 等, 以入口函数为根节点, 获取此根节点函数对其他函数的调用关系图, 对图中的所有节点进行深度优先或广度优先遍历, 将遍历的所有节点函数记为调用函数集合 S_1 , S_1 中的元素表示该可执行文件运行时, 从程序入口出发可能被调用的函数集合。采用同样的方法, 对集合 E 中的其他非入口函数进行遍历和提取, 得到集合 $S_2, S_3, \dots, S_n$, 合并 n 个集合并去重生成集合, 最终得到此可执行文件的可达函数集合 S , S 中的每一个元素函数在可执行文件执行时都是调用可达的。函数可达集合 S 生成过程如图 2 所示。

最后, 根据硬编码字符所在的函数是否在可达函数集合内, 判断其调用可达性。

如图 3 所示, 存在 3 个硬编码函数 $F3()$ 、 $F7()$ 、 $F8()$, 其中, 图 3(a)中 $F3()$ 函数位于入口函数的调用关系图节点中, 是可达函数, 则说明 $F3()$ 中的硬编码字符可以被调用触发; 与之相对, 图 3(b)的 $F7()$ 函数与图 3(c)的 $F8()$ 函数虽然有硬编码字符, 但属于不可达函数, 因而

即使存在硬编码字符, 亦不会被调用触发。因此, 图 3 中存在硬编码且调用可达的函数只有 $F3()$ 。

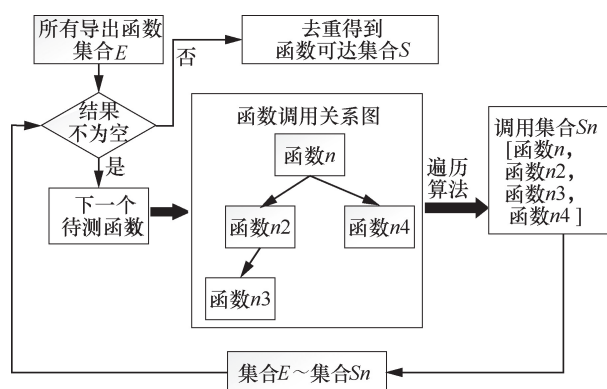
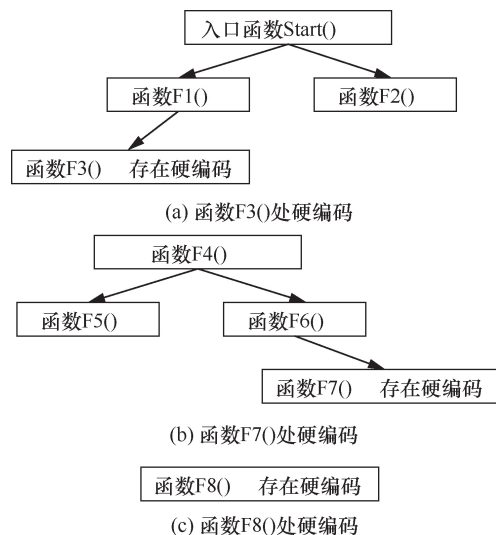


图 2 可达集合 S 生成过程
Figure 2 Reachable set S generating process



可达函数集合: `Start()`, `F1()`, `F2()`, `F3()`
不可达函数集合: `F4()`, `F5()`, `F6()`, `F7()`, `F8()`

图 3 硬编码函数可达性分析
Figure 3 Reach ability analysis of hard coded functions

3.3.2 确定硬编码字符值

硬编码字符所在的函数满足调用可达性之后, 需要对硬编码字符所在的定义位置和赋值过程进行分析, 尤其是以参数形式存储的硬编码字符, 其通常比较隐蔽。

本文首先将待分析函数生成程序控制流程图, 为后续分析提供程序执行流程的依据; 为了解决汇编指令的异构问题, 采用 `PyVEX` 模块的 `IR` 模型将待分析函数转为中间表达式; 采用数据流分析的方法回溯待分析函数的参数, 识别

参数的个数、格式，定位到参数赋值语句，最终根据参数在定义、赋值、调用等过程的情况判定其取值变化，并确定存在漏洞的硬编码字符值。

如图4所示，函数F1()调用函数F2()，且已知当函数F2()中参数 $p=2$ 时存在漏洞。对函数F2()中参数 p 是否存在硬编码的分析过程如下。

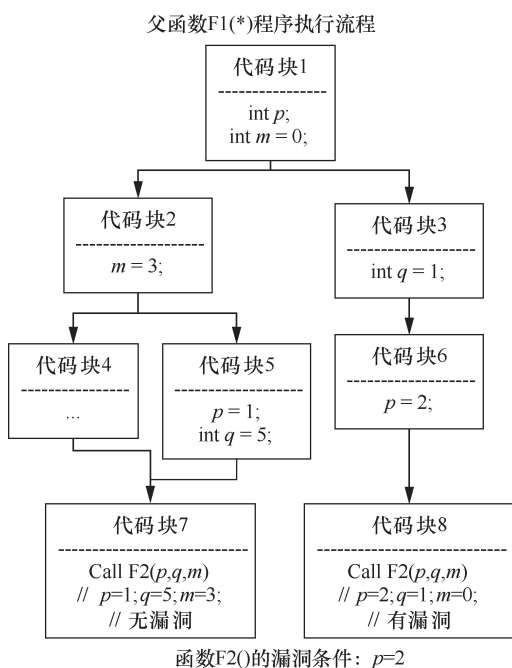


图4 参数硬编码字符值的确定

Figure 4 Determination of parameter hard coded character value

① 识别并定位函数F2()，根据调用关系找到父函数F1()，生成F1()的程序控制流程图。

② 将函数F1()生成中间表达式，结合函数F2()的定义得知有3个参数，类型均为int，采用数据流分析方法对参数进行回溯，并得到与参数定义和赋值相关的代码块为代码块1、2、3、5、6，调用过程为代码块7、8。

③ 根据函数F2()的漏洞描述，当第一个参数 $p=2$ 时是存在缺陷的，则进一步定位出参数 p 的定义和赋值代码块1、5、6，其中参数 p 赋值为2的代码块为6。

④ 分析得到函数F2()中参数 p 的定义、赋值、调用过程分别在代码块1、6、8中，且函数F2()存在漏洞。

3.3.3 硬编码漏洞触发条件分析

分析该硬编码漏洞的路径执行顺序和程序跳

转条件，在程序控制流程图中标注存在硬编码的代码块位置，采用Angr工具中的符号执行模块将执行流程转变为符号表达式，对从入口函数到硬编码代码块的路径进行约束求解，若存在满足要求的约束解，表明该硬编码漏洞可以被触发。

如图5所示，函数Fn()中存在多个疑似硬编码字符漏洞，需要分析从函数Fn()入口执行到漏洞处的触发条件，分析过程如下。

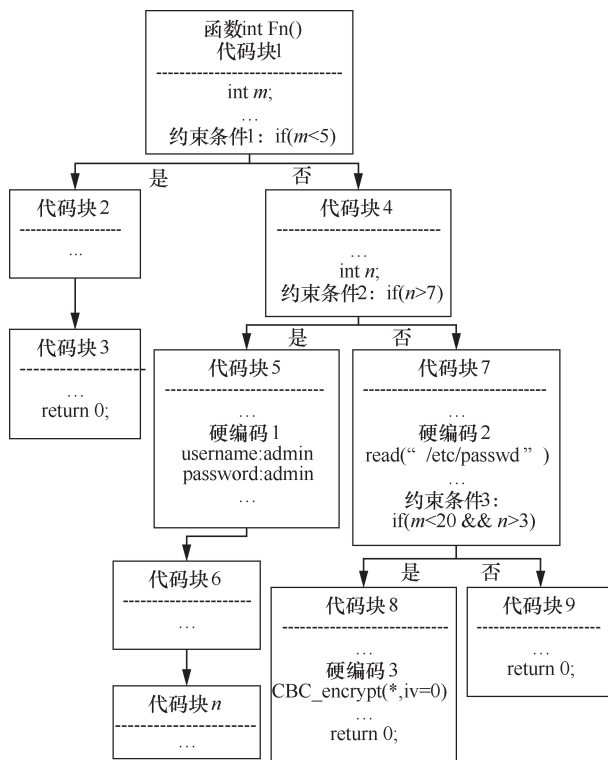


图5 硬编码漏洞触发条件

Figure 5 Hard coded vulnerability trigger conditions

① 生成函数Fn()的程序控制流程图，并在图中标注出存在硬编码字符的代码块为5、7、8。

② 使用Angr载入该文件的Fn()函数，加载符号执行模块生成各代码块间跳转的符号表达式。

③ 设置存在硬编码字符的代码块5、7、8为目标地址，并进行符号表达式的约束求解。

④ 若约束解不为空，则证明存在从函数Fn()入口到该代码块的调用路径，该硬编码字符可以被触发；若约束解为空，则表明此代码块处的硬编码字符无法被执行。

经过以上4个步骤，得到3个硬编码触发需要满足的约束条件如下。

- a. 硬编码 1 的漏洞触发条件为: $m \geq 5, n > 7$ 。
- b. 硬编码 2 的漏洞触发条件为: $m \geq 5, n \leq 7$ 。
- c. 硬编码 3 的漏洞触发条件为: $10 < m \leq 5, 3 < n \leq 7$ 。

4 实验及结果分析

在整体设计、模块功能建模的基础上, 本文进一步构建了针对物联网设备固件的硬编码漏洞检测系统, 并对多类真实物联网设备固件进行了处理和分析, 本节展示漏洞检测结果和部分实例。

4.1 预处理

使用开发的网络爬虫, 从物联网设备官方网站、社区论坛、FTP 站点等网络资源处获得 551 个固件, 包括 360、TP-Link、Tenda、D-Link、Western Digital 等厂商的物联网设备。选取主流厂商不同产品的物联网设备固件, 包括摄像头、路由器、硬盘、打印机。在这些固件中, 使用固件解析工具 Binwalk 和 Firmware-Mod-Kit 成功提取 164 个, 固件架构类型包括 MIPS、ARM、PowerPC、X86_64, 筛选出二进制可执行文件和链接库文件共计 839 个。其中可执行文件共 625 个, 主要位于 bin、sbin 等目录下, 与系统授权、指令操作、组件功能及 Web 服务等相关, 如 busybox、http、webs、ucloud、pppd 等; 链接库文件共 214 个, 主要位于 lib、libs 等目录下, 与 OpenSSL 服务、程序语言库、加解密功能及协议应用等有关, 如 libcrypto.so、libssl.so、libgcc.so 等。

各厂商固件预处理结果如图 6 所示。

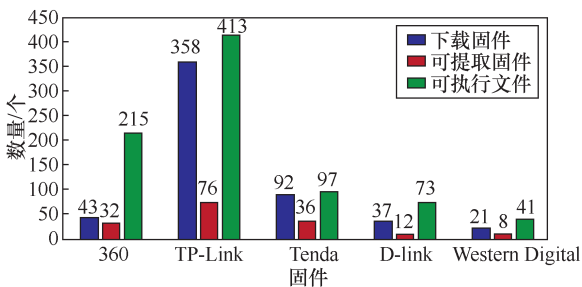


图 6 各厂商固件预处理结果
Figure 6 Firmware pretreatment results of each vendors

固件的预处理文件分布情况如表 1 所示。

表 1 预处理文件分布情况
Table 1 Pretreatment file distribution

文件类型	代表文件	应用场景	架构组成
可执行文件 (625 个)	busybox, boad, pppd, init, httpd, webs, ucloud, app_default, cgi, switch 等	命令操作, Web 服务, 初始化, 授权认证, 时间获取等	MIPS
			ARM
			PowerPC
			X86-64
链接库文件 (214 个)	libc.so, libcrypto.so, libgcc.so, libssl.so, libutil.so, libdl.so, libuClibc.so 等	OpenSSL 库, 加解密操作, 协议应用, 系统语言功能等	MIPS
			ARM
			PowerPC
			X86-64

4.2 硬编码漏洞检测结果

利用构建的物联网设备固件硬编码漏洞检测系统对上述 5 家厂商的相关产品进行测试, 共发现 81 个硬编码漏洞。其中, 每个固件至少存在特殊格式字符 (占比 64.2%) 和外部文件引用 (占比 28.4%) 硬编码漏洞, 容易引起敏感信息泄露、通信流量解密等; 而密码实现中的硬编码漏洞 (占比 7.4%) 则相对较少, 一方面是因为密码应用场景比普通场景少, 另一方面是密码硬编码漏洞只在个别密码算法中出现。各厂商硬编码漏洞检测结果如表 2 所示。

表 2 各厂商硬编码漏洞检测结果
Table 2 Hard-coded vulnerability detection results of each vendors

厂商名称	固件架构及数量	特殊格式字符	外部文件引用	密码实现硬编码
360	MIPS(32)	MIPS(13)	MIPS(5)	0
	ARM(24)	ARM(8)	ARM(4)	0
TP-Link	MIPS(52)	MIPS(12)	MIPS(7)	0
Tenda	MIPS(36)	MIPS(9)	MIPS(3)	MIPS(1)
D-Link	MIPS(7)	MIPS(5)	MIPS(2)	ARM(3)
	ARM(5)	ARM(2)		
Western Digital	ARM(3)			
	PowerPC(2)	PowerPC(1)	PowerPC(1)	ARM(1)
	X86-64(3)	X86-64(2)	X86-64(1)	X86-64(1)
合计	MIPS(127)	MIPS(39)	MIPS(17)	MIPS(1)
	ARM(32)	ARM(10)	ARM(4)	ARM(4)
	PowerPC(2)	PowerPC(1)	PowerPC(1)	
	X86-64(3)	X86-64(2)	X86-64(1)	X86-64(1)

4.3 案例分析

为深入理解物联网设备固件中的硬编码漏洞, 对 3 类硬编码漏洞的典型案例分析。

4.3.1 特殊格式字符硬编码漏洞

特殊格式字符硬编码主要检测对象包括登录

账户、IP 地址、邮箱地址、WPS PIN 码、产品序列号、后门账户等。

(1) 后门账户

在 TP-Link 的 TL-AP450C 摄像头固件中, 函数 `getDefaultWebCtrlUser()` (伪代码如图 7 所示) 位于 `uclited` 可执行文件中, 检测出与 Web Ctrl 应用相关的默认用户 `admin/admin`。

```
1.void getDefaultWebCtrlUser(char *param_1,char *
  param_2){
2.strcpy(param_1,g_default_login_name);
  //g_default_login_name= admin
3.strcpy(param_2,g_default_login_pwd);
  //g_default_login_pwd= admin
4.return; }
```

图7 函数 `getDefaultWebCtrlUser()` 伪代码
Figure 7 Function `getDefaultWebCtrlUser()` pseudo code

在 TP-Link 的 TL-IPC20 摄像头固件中, 可执行文件 `cet` 中函数 `http_digest_auth_check()` (伪代码如图 8 所示) 的作用是对摄像头 Request 请求进行配置, 并在 HTTP Header 中设置 Authorization 值完成 Digest 登录认证。检测出该函数中存在一个硬编码的默认用户 `admin/zMiVw8Kw0oxKXL0`, 该检测结果是对 CVE-2018-11482 的补充, 是在官方公告的漏洞版本之外的新发现。与官方通告的 `websys.lua` 文件不同, 本文的检测对象是可执行文件, 且识别出该硬编码是采用 Digest 方式进行登录的默认摄像头账户。同时在该函数中检测出另一个硬编码字符 `opaque = 64943214654649846565646421`。

(2) WPS PIN 码

WPS PIN 码通常由 8 位纯数字组成, 且满足前四后三加一校验位的规律, 具有明显的识别特征。对已有固件进行分析, 成功实现对 CVE-2016-10179^[31] 漏洞的精准检测。

D-Link 的 DWR-932B 路由器固件开启了

WPS 功能, 在可执行文件 `appmgr` 的函数 `wifi_get_default_wps_pin()` (伪代码如图 9 所示) 中, 检测出硬编码 WPS PIN 值 `28296607`。

4.3.2 外部文件引用

外部文件应用主要检测对象包括配置文件、密钥文件、数字证书文件、账户文件等。

```
1.undefined4http_digest_auth_check(int param_1{
2.....
3.FUN_00431b98("admin",auStack124);
4.iVar11= FUN_004a7e20(auStack124,"zMiVw8Kw0ox
  KXL0",0xf);
5.....
6.http_get_request_header(param_1,"Authorization");
7.iVar4= http_get_request_header(param_1,"Author
  ization");
8.FUN_004a85d0(*(undefined4*)(param_1+ 0x24),
  "WWW-Authenticate:Digest realm=\"%s\",algorithm=
  \"MD5\",qop=\"auth\",nonce=\"%s\",opaque=\"%s\"%s
  \r\n","TP-Link IP-Camera",local_26Q"649432146546
  49846565646421"/pcVar20);
9.....}
```

图8 函数 `http_digest_auth_check()` 伪代码
Figure 8 Function `http_digest_auth_check()` pseudo code

```
1.void wifi_get_default_wps_pin(char *param_1) {
2.iVar1= access("/config/default/factory.ini",0);
3.if (iVar1 == 0) {
4.qcfg_file_gets(&DAT_000672cc,"enrollee_pin","
  28296607",&local_1c,9,"/config/default/factory.
  ini");
5.if ((char)local_1c != '\0') goto LAB_0001fb38; }
6.qcfg_file_gets(&DAT_000672cc,"enrollee_pin","
  28296607",&local_1c,9,"/etc/qrcfg/wifi.ini");
7.strcpy(param_1,(char *)&local_1c);
8.return; }
```

图9 函数 `wifi_get_default_wps_pin()` 伪代码
Figure 9 Function `wifi_get_default_wps_pin()` pseudo code

(1) 密钥文件引用

在 360 的路由器固件中, `lib` 目录下链接库文件 `libcloudquery.so` 的 `createPostHeader()` 函数存在对路径下 `/app/devices/bin/rsa_public_key.pem` 的密钥文件引用, 如图 10 所示。

同样地, 在 `cgitest.cgi` 文件中, `sub_45EB04()`

```
000000000002918
000000000002918 loc_2918:
000000000002918 li $a1, 0
00000000000291C move $a0, $s4
000000000002920 la $t9, rsa_public_encrypt_json
000000000002924 jalr $t9, rsa_public_encrypt_json
000000000002928 addiu $a1, aAppDevicesBinR # "/app/devices/bin/rsa_public_key.pem"
00000000000292C lw $gp, 0x4048+var_4038($sp)
000000000002930 beqz $v0, loc_2DCC
000000000002934 move $s4, $v0
```

图10 `rsa_public_key.pem` 密钥文件引用
Figure 10 `rsa_public_key.pem` key file reference

```

00000000045EBFC
00000000045EBFC loc_45EBFC:
00000000045EBFC addiu $v1, $fp, 0x470+var_42C
00000000045EC00 addiu $v0, $fp, 0x470+var_438
00000000045EC04 addiu $a0, $fp, 0x470+var_440
00000000045EC08 sw $a0, 0x470+var_460($sp)
00000000045EC0C move $a0, $v1
00000000045EC10 lw $a1, 0x470+var_430($fp)
00000000045EC14 lui $v1, 0x48
00000000045EC18 addiu $a2, $v1, (aEtc360PubKey - 0x480000) # "/etc/360_pub.key"
00000000045EC1C move $a3, $v0
00000000045EC20 jal sub_45E940

```

图 11 360_pub.key 密钥文件引用
Figure 11 360_pub.key key file reference

函数引用了/etc/360_pub.key 密钥文件,如图 11 所示。

(2) 账户文件引用

在 360 路由器固件 360P4-V2 中,存在位于 bin 目录下的 busybox 文件读取/etc/passwd 文件,函数 FUN_004344fc()伪代码如图 12 所示。

```

1.undefined4 FUN_004344fc(int param_1,int param_2){
2.....
3.iVar3 = FUN_004771b0("/etc/passwd","a");
4.iVar4 = FUN_004771b0("/etc/passwd","r");
5.....}

```

图 12 函数 FUN_004344fc()伪代码
Figure 12 Function FUN_004344fc() pseudo code

在 passwd 文件中存储着 4 个系统默认用户。文件内容如图 13 所示。

```

1.admin:6HgsSsJIEOc2U:0:0:Administrator:/:/bin/sh
2.support:Ead09Ca6IhzZY:0:0:Technical Support:/:/bin/sh
3.user:tGqCt.qjxbEik:0:0:Normal User:/:/bin/sh
4.nobody:VBcXSNg7zBAY:0:0:nobody for ftp:/:/bin/sh

```

图 13 /etc/passwd 文件内容
Figure 13 Contents of /etc/passwd file

4.3.3 密码实现硬编码漏洞

密码实现硬编码主要检测对象包括 RSA 算法中 Padding 是否存在硬编码值,以及在 CBC 模式中 IV 是否存在硬编码值。

(1) RSA Padding 值

在 Tenda 的 ucloud 文件中,函数 int_rsa_verify() (伪代码如图 14 所示)存在对 RSA 算法的调用,且设置的 RSA Padding 值为 1。

(2) CBC 模式 IV 值

在 Western Digital 固件的 uams_dhx_pam.so 文件中,函数 FUN_000128e0() (伪代码如图 15

所示)调用 CAST_cbc_encrypt()函数,且 CBC 的 IV 值为硬编码静态常量 LWallace。

```

1.undefined4 int_rsa_verify(int param_1,void *
param_2,size_t param_3,uchar*param_4,int *
param_5,uchar*param_6,size_t param_7,RSA*
param_8){
2.....
3.if((param_1==0x72)&&(param_4!=(uchar*)0x0){
4.iVar2 = RSA_public_decrypt(sVar1,param_6,
param_4,param_8); // int padding = 1
5.iVar2 = RSA_public_decrypt(sVar1,param_6,to,
param_8,1); // int padding = 1
6.....}}

```

图 14 函数 int_rsa_verify()伪代码
Figure 14 Function int_rsa_verify() pseudo code

```

1.undefined4 FUN_000128e0(uintparam_1,undefined4
param_2,undefined4param_3,ushort*param_4,uint
param_5,void *param_6,undefined4param_7){
2.....
3.// void CAST_cbc_encrypt(uchar*in,uchar*out,
long length,CAST_KEY*ks,uchar *iv,int enc)
4.CAST_cbc_encrypt((uchar*)in,(uchar*)in,0x90,
(CAST_KEY*)&DAT_000239e8,(uchar*)s_LWallace_
000239b0,0); //s_LWallace_000239b0 = "LWallace"
5.....}

```

图 15 函数 FUN_000128e0()伪代码
Figure 15 Function FUN_000128e0() pseudo code

实验结果表明,本文构建的物联网设备固件可执行文件硬编码漏洞检测系统可以有效识别出特殊格式字符、外部文件引用、密码实现硬编码 3 类硬编码字符,且检测成功率较高。

5 结束语

针对物联网设备固件硬编码中存在的安全问题,本文提出了一种物联网设备固件硬编码字符识别和硬编码漏洞检测方法,并构建了检测系统。设计了一种动态更新的链接库管理方法实现链接库的维护,采用中间表示 IR 模型、数据流分析、符号执行等方法分别实现函数调用可达性分析、

硬编码字符确认和漏洞触发条件分析,对特殊格式字符、外部文件引用、密码实现3类硬编码漏洞检测进行了深入研究和大量实验验证。实验结果表明,通过对164个物联网设备固件的839个可执行文件分析,成功识别出81个硬编码漏洞,证明硬编码漏洞普遍存在于物联网设备固件中。本文提出的方法可以有效检测物联网设备固件中的硬编码漏洞,可为后续物联网设备的安全防护技术的部署提供一定指导。

然而本文依然存在部分关键挑战问题没有被完全解决。首先,在物联网固件提取方面,对于采用非常规压缩算法、非标准系统架构、使用加密保护以及搭载实时操作系统的固件,本文提出的方法在适配性方面还存在较大不足;其次,在硬编码字符和硬编码漏洞选择上,未涉及所有类型;此外,在对漏洞攻击利用和安全加固方面,本文尚未做深入研究。在今后的研究工作中,需增加固件样本种类和数量,深入研究非典型固件的提取技术,提高检测的适用范围,增加对更多类型硬编码漏洞的支持,并进一步研究漏洞利用与安全防护方案。

参考文献:

- [1] 王鑫. 面向分布式计算的隐私保护研究[D]. 浙江: 浙江大学, 2020.
WANG X. Research on privacy protection for distributed computing[D]. Zhejiang: Zhejiang University, 2020.
- [2] BURSSTEIN E, COCHRAN G J, DURUMERIC C Z, et al. Understanding the Mirai Botnet[C]//USENIX Security Symposium. 2017.
- [3] 杨毅宇, 周威, 赵尚儒, 等. 物联网安全研究综述: 威胁, 检测与防御[J]. 通信学报, 2021, 42(8): 188-205.
YANG Y Y, ZHOU W, ZHAO S R, et al. Survey of IoT security research: threats, detection and defense[J]. Journal on Communications, 2021, 42(8): 188-205.
- [4] 张玉清, 周威, 彭安妮. 物联网安全综述[J]. 计算机研究与发展, 2017, 54(10): 2130.
ZHANG Y, ZHOU W, PENG A N. Survey of internet of things security[J]. Journal of Computer Research and Development, 2017, 54(10): 2130-2143.
- [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): 61-75.
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): 61-75.
- [6] CAO X H, SHILA D M, CHENG Y, et al. Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks[J]. IEEE Internet of Things Journal, 2016, 3(5): 816-829.
- [7] WEN H, CHEN Q A, LIN Z. Plug-N-Pwned: comprehensive vulnerability analysis of OBD-II dongles as a new over-the-air attack surface in automotive IoT[C]//USENIX Security Symposium. 2020: 949-965.
- [8] CHEN J Y, ZUO C S, DIAO W R, et al. Your IoTs are (not) mine: on the remote binding between IoT devices and users[C]//2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. 2019: 222-233.
- [9] HE W, GOLLA M, PADHI R, et al. Rethinking access control and authentication for the home Internet of things (IoT)[C]//USENIX Security Symposium. 2018: 255-272.
- [10] YUAN B, JIA Y, XING L, et al. Shattered chain of trust: understanding security risks in cross-cloud IoT access delegation[C]//USENIX Security Symposium. 2020: 1183-1200.
- [11] FERNANDES E, RAHMATI A, JUNG J, et al. Decentralized action integrity for trigger-action IoT platforms[C]//Proceedings 2018 Network and Distributed System Security Symposium. 2018: 1-16.
- [12] ZHANG L, CHEN J, DIAO W, et al. {CryptoREX}: large-scale analysis of cryptographic misuse in {IoT} Devices[C]//22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019). 2019: 151-164.
- [13] ALMAKHADHUB N S, CLEMENTS A A, BAGCHI S, et al. μ RAI: securing embedded systems with return address integrity[C]//Proceedings 2020 Network and Distributed System Security Symposium. 2020: 1-18.
- [14] YAO Y, ZHOU W, JIA Y, et al. Identifying privilege separation vulnerabilities in IoT firmware with symbolic execution[C]//European Symposium on Research in Computer Security. 2019: 638-657.
- [15] ZHOU J, DU Y, SHEN Z, et al. Silhouette: efficient protected shadow stacks for embedded systems[C]//USENIX Security Symposium. 2020: 1219-1236.
- [16] MU C, YANG M, CHEN Z, et al. Research on RSA padding identification method in IoT firmwares[C]//Journal of Physics: Conference Series. 2020: 012061.
- [17] Vulnerabilities in foscaml IP cameras report[R].
- [18] THOMAS S L, CHOTHIA T, GARCIA F D. Stringer: measuring the importance of static data comparisons to detect backdoors and undocumented functionality[C]//European Symposium on Research in Computer Security. 2017: 513-531.
- [19] SHOSHITAISHVILI Y, WANG R, HAUSER C, et al. Firmalice-automatic detection of authentication bypass vulnerabilities in binary firmware[C]//NDSS. 2015: 1-8.
- [20] WANG F, SHOSHITAISHVILI Y. Angr-the next generation of binary analysis[C]//2017 IEEE Cybersecurity Development (SecDev). 2017: 8-9.
- [21] ZHENG Y, DAVANIAN A, YIN H, et al. FIRM-AFL: greybox fuzzing of IoT firmware via augmented process emulation[C]//28th USENIX Security Symposium (USENIX Security 19). 2019: 1099-1114.

- [22] ZADDACH J, BRUNO L, FRANCILLON A, et al. AVATAR: A framework to support dynamic security analysis of embedded systems' firmwares[C]//NDSS. 2014: 1-16.
- [23] CHEN J Y, DIAO W R, ZHAO Q C, et al. IoTfuzzer: discovering memory corruptions in IoT through app-based fuzzing[C]//Proceedings 2018 Network and Distributed System Security Symposium. 2018: 1-15.
- [24] 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.
- [25] 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 (USENIX Security 21). 2021: 303-319.
- [26] MU C, WANG X, YANG M, et al. Vulnerability analysis for iot devices of multi-agent systems: a cryptographic function identification approach[C]//Proceedings of 2021 5th Chinese Conference on Swarm Intelligence and Cooperative Control. 2023: 1510-1519.
- [27] DAVIDSON D, MOENCH B, RISTENPART T, et al. {FIE} on firmware: Finding vulnerabilities in embedded systems using symbolic execution[C]//22nd USENIX Security Symposium (USENIX Security 13). 2013: 463-478.
- [28] NETHERCOTE N, SEWARD J. Valgrind: a framework for heavy-weight dynamic binary instrumentation[J]. ACM Sigplan Notices, 2007, 42(6): 89-100.
- [29] DUONG T, RIZZO J. Here come the @ninjas[J]. Unpublished manuscript, 2011, 320.
- [30] CHEN D D, EGELE M, WOO M, et al. Towards automated dynamic analysis for linux-based embedded firmware[C]//Proceedings 2016 Network and Distributed System Security Symposium. 2016: 1-16.

[作者简介]



穆超（1990-），男，山东济南人，齐鲁工业大学（山东省科学院）助理研究员，主要研究方向为物联网安全与数据安全。



王鑫（1992-），男，山东邹平人，博士，齐鲁工业大学（山东省科学院）副研究员，主要研究方向为工业互联网安全与数据隐私保护。



杨明（1981-），男，山东东营人，博士，齐鲁工业大学（山东省科学院）研究员，主要研究方向为密码学、云安全、大数据安全、物联网安全。



张恒（1981-），男，江苏连云港人，博士，江苏海洋大学教授，主要研究方向为系统安全与控制和人工智能算法安全。



陈振娅（1983-），女，山东菏泽人，齐鲁工业大学（山东省科学院）副研究员，主要研究方向为数据安全与隐私保护。



吴晓明（1981-），男，山东滨州人，博士，齐鲁工业大学（山东省科学院）研究员、博士生导师，主要研究方向为物联网安全与信息安全。