计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(11):68−78 [doi: 10.15888/j.cnki.csa.008765]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 MQTT 协议扩展的 IoT 设备完整性监控①

齐　兵[1,2], 秦　宇[2], 李敏虹[3], 谢　宏[3], 尚科彤[1,2], 冯　伟[2], 李　为[1,2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 软件研究所 可信计算与信息保障实验室, 北京 100190)
[3](深圳供电局有限公司, 深圳 518028)
通信作者: 秦　宇, E-mail: qinyu@iscas.ac.cn

摘　要: 随着物联网飞速发展, 设备数量呈指数级增长, 随之而来的 IoT 安全问题也受到了越来越多的关注. 通常 IoT 设备完整性认证采用软件证明方法实现设备完整性校验, 以便及时检测出设备中恶意软件执行所导致的系统完整性篡改. 但现有 IoT 软件证明存在海量设备同步证明性能低、通用 IoT 通信协议难以扩展等问题. 针对这些问题, 本文提供一种轻量级的异步完整性监控方案, 在通用 MQTT 协议上扩展软件证明安全认证消息, 异步推送设备完整性信息, 在保障 IoT 系统高安全性的同时, 提高了设备完整性证明验证效率. 我们的方案实现了以下 3 方面安全功能: 以内核模块方式实现设备完整性度量功能, 基于 MQTT 的设备身份和完整性轻量级认证扩展, 基于 MQTT 扩展协议的异步完整性监控. 本方案能够抵抗常见的软件证明和 MQTT 协议攻击, 具有轻量级异步软件证明、通用 MQTT 安全扩展等特点. 最后在基于 MQTT 的 IoT 认证原型系统的实验结果表明, IoT 节点的完整性度量、MQTT 协议连接认证、PUBLISH 报文消息认证性能较高, 都能满足海量 IoT 设备完整性监控的应用需求.

关键词: 物联网安全; 完整性度量; MQTT 协议安全扩展; 软件证明; 可信计算

引用格式: 齐兵,秦宇,李敏虹,谢宏,尚科彤,冯伟,李为.基于 MQTT 协议扩展的 IoT 设备完整性监控.计算机系统应用,2022,31(11):68–78. http://www.c-s-a.org.cn/1003-3254/8765.html

Integrity Monitoring for IoT Device Based on MQTT Protocol Extension

QI Bing[1,2], QIN Yu[2], LI Min-Hong[3], XIE Hong[3], SHANG Ke-Tong[1,2], FENG Wei[2], LI Wei[1,2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[3](Shenzhen Power Supply Co. Ltd., Shenzhen 518028, China)

Abstract: With the rapid development of the Internet of Things (IoT), the number of IoT devices has grown exponentially, which is accompanied by the increasing attention to IoT security. Generally, IoT devices adopt software attestation to verify the integrity of the software environment, so that system integrity tampering caused by the execution of malicious software can be detected timely. However, the existing software attestation suffers from poor performance in the synchronous attestation of massive IoT devices and the difficulty in extending the general IoT communication protocol. To address these problems, this study proposes a lightweight asynchronous integrity monitoring scheme. The scheme extends the security authentication message of software attestation on the general message queuing telemetry transport (MQTT) protocol and asynchronously pushes the integrity information of devices. It improves not only the security of IoT systems but also the efficiency of integrity attestation and verification. The following three security functions are realized: device integrity measurement in a kernel module; lightweight authentication extension of device identity and integrity based on MQTT; asynchronous integrity monitoring based on MQTT extension protocol. This

① 基金项目: 国家重点研发计划 (2020YFE0200600); 国家自然科学基金 (61872343); 中国科学院青年创新促进会
收稿时间: 2022-02-24; 修改时间: 2022-03-15; 采用时间: 2022-03-21; csa 在线出版时间: 2022-07-07

scheme can resist common software attestation attacks and MQTT protocol attacks and has the characteristics of lightweight asynchronous software attestation and general MQTT security extension. The experimental results of the prototype system of IoT authentication based on MQTT show the high performance of the integrity measurement of IoT nodes, MQTT protocol connection authentication and PUBLISH message authentication, which can meet the application requirements of integrity monitoring of massive IoT devices.

Key words: IoT security; integrity measurement; message queuing telemetry transport (MQTT) protocol security extension; software attestation; trusted computing

1 引言

设备节点在 IoT 环境中, 运行程序完成特定工作, 如: 石油管道传感器节点负责采集管道的状态信息, 以及时发现管道的异常状态; 智慧城市中, 智能交通信号灯、智能电力装置等众多 IoT 设备共同提供基础设施服务[1]. 在物联网领域, 针对具体的应用场景通常有两种模式, 一种是大量的同构低端设备同时工作, 典型代表有工控 IoT 设备、智能门锁等家庭物联网设备[2]; 一种是多类型异构设备共同提供服务, 主要代表有智慧城市中提供基础服务的物联网设备, 包括: 气象监测设备、交通管理设备、电力控制设备等.

物联网设备数量的飞速增长, 带来了新的安全威胁和挑战. 由于 IoT 设备通常不具备 PC、服务器的计算、存储能力, 往往也没有专用的安全设备, 其更容易受到安全威胁. 智能家居、可穿戴设备在被攻击后将泄露用户隐私信息. 关键的医疗设备、智能汽车在受攻击时甚至会直接威胁使用者的生命安全. 2015 年的黑客远程控制汽车事件, 攻击者通过吉普车的联网系统侵入电子系统, 远程控制车的行驶速度, 操纵空调、电台等设备. 2021 年 5 月 9 日, 美国最大的输油管道网络遭到攻击, 造成了大量经济损失.

因此, IoT 设备远程证明和远控监控技术是必要的, 即: 由专门的云服务器验证节点对其他功能设备节点进行安全验证、管理.

1.1 国内外研究现状

1.1.1 MQTT 协议

MQTT 协议[3,4] 是一种轻量级的 M2M 物联网传输协议. 适用于空间占用和带宽使用非常严格的远程连接, 满足资源受限、低带宽、高延迟和低可靠性等应用场景, 如: 物联网 M2M 通信、安卓消息推送、车联网通信、石油电力等能源行业.

MQTT 协议的报文由 3 部分组成. 第 1 部分为固定报头, 用 1 字节定义报文类型和标志位, 最小 1 字节记录剩余长度. 第 2 部分为可变报头, 由报文类型决定该部分的构成. 第 3 部分为载荷 (payload), 用于承载数据部分. 这种开销极小的报文构成, 非常适合资源受限的物联网环境, 以降低计算、存储、传输开销.

MQTT 采用发布/订阅模式. 该协议用户有两种身份, 分别是: 发布者和订阅者. 订阅者向服务器 broker 订阅一个主题 topic 后, 每当发布者发布该主题的消息时, 就会经由 MQTT broker 转发给订阅者. 该协议可以将消息分配给多个订阅用户, 有效地实现一对多消息发布.

为了保证可靠性需求, MQTT 提供服务质量管理 (QoS), 设计了 3 种不同的级别以应对不同应用环境. “至多一次”只发送一次, 适用于周期性频繁传输的推送, 即丢失也可以接受; “至少一次”保证消息顺利到达, 如果没有成功就会重复发送; “只有一次”同样保证消息到达, 但与“至少一次”不同的是, 该 QoS 确保消息只会到达一次, 不会出现重复.

MQTT 协议的 V3 版本实现了上述功能, V5 版本在 V3 的基础上改进了客户端与服务端之间通信的反馈, 并增加了报文属性, 如: 会话到期间隔、用户属性. V5 版本提供了共享订阅功能, 用于客户端订阅消息的负载均衡, 在同一订阅组中的客户端交替接受消息推送. 在大量设备频繁发送短消息的场景下, V5 提供了主题别名, 减少重复主题名的空间占用.

由于 MQTT 协议本身只提供了 username、password 字段来提供基础的身份认证, 安全风险极高, 协议的现有设计已经无法满足需求[5]. 因此, 有效的安全方案是需要解决的问题. 在文献 [6] 中展示了目前的解决方案的缺点, 如: 使用通用的加密算法引入较高的计算开销; 使用 TLS 会显著影响性能; 物联网环境往往具有大量的节点, 产生大量的数据, 因此需要实现灵活的数据共

享和访问控制. 在文献 [7] 中, Harsha 等提出了 MQTT 面临的多种安全挑战, 如: 缺乏有效的身份认证技术, 任何客户端都可以发布和或订阅所有的主题. 这将对数据的机密性构成威胁; 无法阻止客户端订阅某一主题.

本方案利用 MQTT 的发布订阅模式来设计完整性监控协议, 无需验证者与每一个终端设备建立、维护连接. 对 MQTT 进行安全拓展, 由 broker 保证通信的设备节点合法性以及完整性数据的可信性, 验证者只需订阅相关主题, 即可获得完整性数据的推送, 验证设备的可信状态.

1.1.2 软件证明

在可信物联网体系中, 软件的完整性是重要的安全需求[8], 因此引入软件证明来保证运行环境的可信以及验证软件的可信. 软件证明是指软件在嵌入式设备上运行时, 要检验软件是否缺失、软件是否被恶意篡改、真实的软件是否被执行. IoT 设备大多资源受限, 因此该领域的证明多采用软件或轻量级的软硬件协同方案, 如: 基于 Puf 的 AAoT[9].

软件证明过程一般由验证者发起, 对设备中的内存数据、处理器状态等进行特定形式的验证[10]. 通用的证明模型采用挑战应答方式, 验证者 V 产生随机数 nonce, 发送证明请求给设备节点, 设备节点 P 进入度量过程, 使用随机数 nonce 计算校验和, 并将结果发送回验证者 V, 验证者提取自身存储的基准值或软件副本进行校验比对, 以确定设备的软件状态.

1.1.2.1 单设备证明方案

现已提出了 3 种类型的单设备解决方案: 一种是基于响应时间的软件证明. 一种是随机构造证明函数的软件证明, 一种是单向的软件证明. 前两种方案由验证者发起, 再由节点计算结果并反馈给验证者, 第 3 种是由设备节点自行发起, 依赖于可靠的时钟硬件, 向验证者发起证明.

(1) 基于响应时间的软件证明

基于时间的软件证明将证明的响应时间纳入判断, 从验证节点发起证明请求开始, 到验证节点收到度量值为止, 记录响应时间. 若超出可接受的时间范围即认为是证明失败. 典型的方案有 SWATT[11]、SCUBA[12]、VIPER[13]、SBAP[14].

SWATT 方案是一种软件实现的证明方案, 能够以较高概率地检测内存数据的修改. SWATT 采用了挑战应答的机制, 随机生成要访问的内存地址发送给设备节点, 设备节点基于该内存地址进行度量, 并返回结果. 攻击者必须在证明请求到来时, 监测是否检验已经篡改的部分, 若访问到则必须欺骗验证过程访问正确的位置. 验证节点根据响应时间可以判断设备的软件状态. 为了防止攻击者在篡改了软件的同时在空闲内存中保留合法副本, 基于时间的方案可采用内存随机填充. 当证明请求到达时, 在设备节点空闲内存中加入伪随机噪声, 不给攻击者留下可利用空间.

(2) 基于随机构造证明函数的软件证明

这类方案采用证明函数随机化的方式来实现软件证明, 常规方案多采用固定的证明函数嵌入到设备固件中, 攻击者可以静态分析获取此类过程函数, 如: hash 函数. Shaneck 等[15] 提出了一种新的证明方案来检测恶意代码, 不依赖于证明过程的时间延迟, 保证每次证明请求发送给设备的验证过程都是不同的. 这种方案有效防止了重放攻击, 并防止对证明函数的分析破解.

(3) 单向软件证明

前述两种方案采取双向交互证明的方式, 验证节点发起软件证明请求, 设备节点中断当前任务进入证明计算流程, 将计算结果传输给验证节点, 验证节点进行相同计算验证设备软件的合法性. 这样的设计可以让验证节点控制证明开始的时间, 但一旦攻击者有能力伪造证明请求, 就可以实现对设备节点的拒绝服务攻击. 因此, 产生了由设备端发起的证明方案.

SeED 方案[16] 需要依赖硬件来实现. 可靠的实时时钟和证明触发电路辅助证明的进行, 设备节点在随机时间点触发证明过程, 去除验证者发送验证请求的过程, 帮助资源受限的设备抵抗拒绝服务攻击, 证明设备的软件可信性.

1.1.2.2 多设备证明

在物联网大规模设备的应用场景下, 对单一设备进行验证的软件证明是无法满足需求的. 现今提出了多种集群远程证明的方案, 但往往需要额外安全硬件的支持, 典型的方案有 SEDA[17], 其为设备数量大的应用环境提供了可扩展的证明协议, 但只获得证明成功的设备个数, 无法得知失败设备的具体信息; SANA[18] 在 SEDA 的基础上进行设计, 使验证阶段可以获取失败节点的 ID 与软件配置信息, 同时考虑了硬件篡改攻击; AID[19] 利用缺席检测机制来验证网络的完整性; ESDRA 协议[20] 利用分布式证明来验证每个节点的完

整性, 并应用指控机制来报告被入侵的节点, 但方案假设每个设备至少拥有 3 个以上的邻居设备. 另一种思路是基于设备分组的证明方案[21], 将设备划分给不同的管理节点, 由管理节点负责对所属设备的验证, 验证者只需验证管理节点的完整性.

1.2 现有方案的问题

在 IoT 环境下, 影响验证的因素有 3 个方面: 验证节点的同步计算开销、多验证节点连续或同时触发证明对设备节点的可用性影响、系统结构中验证节点的调整难度.

(1) 验证节点的计算开销

现有的验证方案一般采用挑战应答的方式, 由验证节点主动发起证明请求, 请求中附带随机挑战值, 终端设备节点利用收到的挑战值计算内存校验和并返还给验证节点, 验证节点采用同样的方式进行计算, 并比对收到的校验和, 以此判断终端设备节点的完整性状态. 在大量终端设备的 IoT 环境中, 若采用这种挑战应答的同步方式, 验证节点需要维护大量的网络连接来实现对终端设备的远程证明. 这会给验证节点造成巨量的计算开销.

(2) 系统结构的更新调整难度

针对 (1) 中的问题, 一种改进的方式是采用分层的集群证明方案, 由低级验证节点负责一部分 IoT 终端设备的完整性校验, 由高级验证节点对所有的低级验证节点进行校验证明, 即高级验证节点证明低级验证节点的可信性, 低级验证节点证明 IoT 终端节点的可信性. 这种改进方案存在的问题是, 需要维护复杂的管理体系, 系统网络中节点规模越大就需要越庞大的管理结构层级, 并且当某验证节点故障、失效、网络堵塞将影响其下属的全部节点, 难以有效进行系统网络中节点的添加、删除、修改以及控制权的转移.

(3) 设备节点的可用性影响

在证明过程中设备节点同步的开销是无法忽视的. 当验证节点向终端设备发送证明请求后, 终端设备会产生中断进入度量状态, 完成度量后才可以回到正常的程序状态. 在大规模 IoT 设备环境中, 若多个验证者同时或连续向设备发送证明请求, 则设备可用性将受到极大影响. 另一方面, 现有的证明方案常依赖于响应时间判断设备是否可信, 大批量的度量结果流向验证节点造成时间延迟, 进一步会导致基于时间的方案失去作用.

1.3 方案简介

为了解决上述问题, 本文采用 MQTT 协议进行新的可信认证体系的构建, 利用其发布订阅模式解决上述问题, 但由于 MQTT 本身存在的安全缺陷, 本方案将对 MQTT 协议进行安全拓展, 并基于改进后的协议设计实现完整性监控协议.

本方案的工作包含如下 3 个方面:

(1) 对 MQTT 协议进行了改进, 进行了轻量级的身份认证安全拓展, 保证验证者与 MQTT broker、终端设备与 broker 之间通信的可信性.

(2) 在设备节点插入内核模块, 进行设备软件完整性的度量. 内核模块周期性对设备的进程信息进行采集度量.

(3) 基于改进的 MQTT 协议进行完整性监控方案的设计, 保证传输给验证者的设备度量信息未经篡改且来自合法节点, 有效应对大批量 IoT 设备的应用场景. 本方案采用异步证明方式, 利用 MQTT 协议的发布订阅模式, 提供 IoT 节点的完整性监控功能.

本文提供一种轻量级的异步完整性监控方案, 以内核模块方式实现设备完整性度量功能, 设计实现了基于 MQTT 的设备身份认证扩展, 并基于扩展后的 MQTT 协议实现异步完整性监控功能. 有效解决了现有软件证明存在海量 IoT 设备同步证明验证性能低下、通用 IoT 通信协议难以扩展等问题, 缓解了同步开销, 提升 IoT 系统安全性的同时提高了设备软件证明验证效率, 并降低了系统结构调整的难度. 在安全性方面, 本方案能够抵抗常见的软件证明和 MQTT 协议攻击.

2 MQTT 完整性监控方案设计

2.1 威胁模型

本方案的目标是对物联网设备进行完整性监控, 设备度量结果经 MQTT broker 转发给所有订阅了该主题的验证者. 因此主要的攻击来自设备系统自身以及网络通信, 系统面临的攻击点主要有物联网设备节点本身、节点与 MQTT broker 建立连接的过程、设备向 MQTT broker 推送的过程.

(1) 设备节点网络入侵. 攻击者直接攻击 IoT 设备节点, 在节点系统中运行恶意程序或篡改合法程序.

(2) 破解设备节点口令和密钥. 利用 IoT 设备的节点口令强度不够, 密钥存储和密钥分发存在的缺陷, 破解设备, 采用伪造系统状态数据欺骗服务器验证节点

的完整性监控检查.

(3) 伪造设备节点数据包, 企图欺骗 MQTT broker. 由于原本的 MQTT 协议只提供了用户名、密码的明文校验, 伪造数据包的攻击非常容易欺骗 broker.

(4) 中间人攻击. 采用拦截篡改 MQTT broker 与终端节点的通信数据, 如当攻击者在设备上运行了恶意程序, 但在发往 broker 的数据包种填充了正常的度量数据.

(5) 重放攻击. 攻击企图利用之前的合法完整性数据跳过验证节点的验证过程, 以防止恶意程序被验证者识别.

2.2 安全假设

本文有如下假设: (1) 终端设备节点、验证节点与 MQTT broker 的时钟是同步且可信的. (2) MQTT broker 与验证节点是可信的, 且验证节点已知所属的设备节点的软硬件配置. (3) 假设系统网络中, 所有的节点都有一个静态非易失写保护内存区域 A, 该区域存储终端设备的核心校验代码, A 区域代码不可写, 不可中断执行. (4) 假设节点与 MQTT broker 约定的会话密钥、初始的身份密钥是安全存储的, 可以由 TCM 加密存储实现或系统中专门受保护的内存区域. 不考虑身份密钥的分配机制. (5) 敌手不能修改硬件设备.

2.3 符号说明

表 1 列出方案所有使用的符号.

2.4 系统体系结构

MQTT 完整性监控系统主要由 3 方面参与者组成: 设备节点、MQTT broker、验证节点. 系统结构如图 1.

(1) 设备节点是指实际应用中负责专用功能的终端 IoT 设备, 他们可以是温度传感器、信息采集传感器等. 同时他们在完整性监控领域中是被验证的对象, 作为消息发布者, 通过内核模块采集自身完整性信息, 并通过 MQTT broker 转发给验证者, 以确定其运行在可信状态.

(2) MQTT broker 负责维持整个网络的通信, 负责节点间消息的中转传输, 订阅者节点向 broker 订阅主题, 发布者节点向 broker 推送信息, 并由 broker 进行转发. 在本方案中, broker 节点需对主题订阅和消息推送的节点进行身份认证, 阻止攻击者伪造数据包欺骗 broker, 保证节点到 broker 通信的可信性, 最重要的是保证完整性度量信息来自于可信的终端设备节点.

表 1 标识符与函数标识

标识符与函数标识	说明
Device	终端设备节点
Broker	MQTT broker节点
Verifier	方案的验证节点
Kb	MQTT Broker的公钥
$Kprib$	MQTT Broker的私钥
Kd	设备节点的公钥
$Kprid$	设备节点的私钥
generateNonce( )	生成随机数
getPubkey( )	获取身份密钥
genKey(p, q)	使用两个随机数生成会话密钥
SignSignature(m, K_{pri})	使用私钥K_{pri}计算消息m的签名
VerifySignature(m, K_{pub}, S)	使用公钥K_{pub}验证消息m的签名S
$ENC_{K_{pub}}(m)$	用公钥K_{pub}对消息m进行非对称加密
$DEC_{K_{pri}}(m)$	用私钥K_{pri}对消息m进行非对称解密
getDigest( )	进行设备完整性度量值计算
getTime( )	获取可信时间
VerifyTime(CurTime, T)	验证当前时间CurTime是否在时间T的合法范围内
Hash(m)	计算消息m的hash值
ComputeCheckSum(K, m)	计算消息m的hash值Hash(m), 用密钥K对其进行加密计算认证码
CheckCheckSum(K, m, t)	用密钥K计算消息m的认证码并与t进行比对验证
getACK( )	获得PUBLISH报文推送的结果

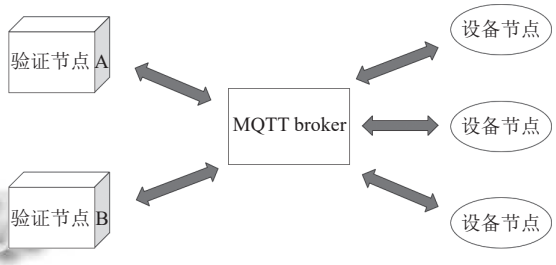

图 1 完整性监控协议系统结构

(3) 验证节点负责对设备节点的完整性度量值进行校验比对. 验证节点首先向 broker 订阅完整性校验主题, 设备节点推送的度量信息经 broker 转发至验证节点后, 在验证节点本地进行完整性校验, 比对后确认终端设备节点的安全状态. 当网络中一个验证节点出现问题, 如: 网络堵塞、超时失联等, 可以很容易设置一个新的验证节点, 因为验证节点只需经由 MQTT broker 检验权限后即可订阅完整性主题, 获得设备完整性信息的推送.

2.5 方案流程

在大规模物联网设备的应用场景中, 验证节点需要处理数量巨大的并发验证和完整性检验. 对于大量

IoT 设备采用常规的挑战应答模式的验证方式会造成巨大的同步开销, 导致功能无法正常执行. 本方案对现有的 MQTT 协议进行软件证明安全扩展, 加入轻量级身份认证功能, 并让设备节点周期性度量设备进程的完整性状态, 发送给验证方验证, 最后基于 MQTT 协议进行 IoT 节点完整性的异步传输和监控.

方案流程分为 5 个主要部分: 初始化配置阶段、设备度量阶段、CONNECT 阶段、推送阶段、订阅验证阶段. 下面介绍方案的具体流程, 参与身份包括: 验证者 V、设备节点 P、MQTT broker.

2.5.1 初始化配置阶段

终端设备节点、验证节点、MQTT broker 节点配置各自身份密钥对. 在 MQTT broker 中注册终端设备节点与验证节点的密钥信息, 便于后续的身份认证流程. 由于本方案不关注密钥分配的流程, 因此所有身份密钥已经在初始化阶段设置完成, 即 MQTT broker 与设备节点、MQTT broker 与验证节点都已知交互双方的公钥.

2.5.2 设备度量阶段

本方案采用内核模块的方式实现对设备上程序的完整性度量, 如图 2. 终端设备节点插入内核模块, 周期性地触发对系统内进程的度量计算. 度量可分为静态度量、动态度量两种形式.

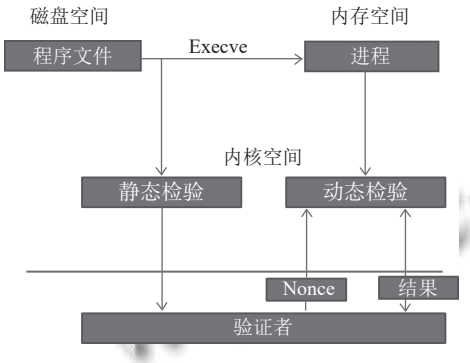

图 2 设备度量流程

静态度量采用 Linux 的 IMA 架构, 利用 Hook 机制修改了程序执行流程, 每次执行程序时, 在将程序的 ELF 文件加载进内存空间形成进程前, 内核对整个 ELF 文件进行度量计算, 并安全存储开机后运行的所有进程度量信息.

动态度量依赖于 DIMA 架构[22], 在系统运行时周期性触发, 对系统内活动进程进行度量, 采用的方法是提取内核维护的全部进程结构体, 进而获取全部进程的代码段、参数信息进行度量计算. 两种方式的度量信息安全存储可以由 TPM/TCM 芯片本身提供, 或者采用预置的身份密钥进行加密存储, 以抵抗针对度量信息的篡改.

2.5.3 CONNECT 阶段

终端设备节点在进行度量信息推送发布前、验证者节点在向 MQTT broker 订阅主题前, 首先应被 MQTT broker 身份认证.

MQTT 协议头中只提供了 username 和 password 字段, 用于简单的身份认证, 其实现逻辑依赖于用户名和密码的匹配. 本方案借助这两个字段进行身份认证方案的改造, 在 CONNECT 报文和 CONNACK 报文中设计了新的轻量级认证方案, 为后续的完整性监控提供身份合法性的认证.

首先, 节点和 broker 各自生成自身的随机数, 用对方的公钥加密后发送给对方, 这一过程通过 CONNECT 与 CONNACK 报文实现加密随机数的传输. 设备节点生成随机数 p, 用 broker 公钥加密后发送给 broker. broker 节点生成随机数 q, 用终端设备的公钥加密后发送给设备节点. 双方用这两个随机数 p 和 q 生成会话密钥 SK, 用于后续的度量信息的完整性、可靠性证明. 双方各自生成随机数共同生成密钥的方法可以有效防止中间人攻击, 当合法节点发送 CONNECT 数据包时, 被攻击者拦截. 攻击者利用该数据包的 username 和 password 字段企图通过 broker 的身份认证, 并与 broker 建立连接. 但由于产生的两个随机数都由对方密钥加密处理, 因此攻击者无法获知随机数, 因此无法生成约定的会话密钥 K, 在后续的通信时将轻易被 broker 发现.

第二, 除了引入随机数生成会话密钥, 本方案在 password 字段中加入数字签名, 并修改 broker 的处理逻辑. 通过终端设备的 TCM 芯片与被保护的密钥计算数字签名, 将签名放入 MQTT CONNECT 报文的 password 字段中. CONNECT 报文发送到 broker 后, broker 找到用户名 username 对应的节点的公钥, 验证签名是否来自合法的节点, 并决定是否建立连接.

下面描述该阶段的具体细节.

节点构造 CONNECT 报文. 在 MQTT CONNECT 报文的基础上, 针对 password 字段进行身份认证的扩展. 节点生产随机数 p, 使用 broker 的公钥 Kb 加密获得密文 $C1$, 对 $C1$ 进行 hash 计算并使用自身私钥 $Kprid$ 计算签名得到 S=SignSignature(Hash($C1$), $Kprid$). 将

$C1$ 与 S 拼接构造新的身份认证信息, 放入 password 字段. CONNECT 报文格式如图 3(a).

MQTT broker 节点收到 CONNECT 报文后, 首先获取 user 字段内容, 取得对应公钥 Kd. 拆分 CONNECT 报文中的 $C1$ 与 S, 验证 VerifySignature(Hash($C1$), Kd, S) 以确定签名是否来自合法节点. 当验证失败后拒绝连接, 并发送失败信息给节点; 当验证通过后, 使用自身私钥 $Kprib$ 解密 $C1$ 获得随机数 p. broker 生成随机数 q, 使用节点的公钥 Kd 对 q 进行加密获得 $C2$. 由于 MQTT 协议本身的设计中, CONNACK 报文只含有协议头和可选头两部分, 无位置存储 $C2$, 因此对协议的 CONNACK 报文进行了调整, 加入身份认证头存储 $C2$, 构造新的报文发送给节点, CONNACK 报文结构如图 3(b). 同时, broker 利用两个随机数 p 和 q 计算会话密钥 SK, 用于后续的完整性监控.

节点收到 CONNACK 报文后, 首先检查连接结果. 确认连接成功后, 提取出 $C2$, 使用自身私钥 $Kprid$ 进行解密获得随机数. 节点同样利用两个随机数 p 和 q 计算出会话密钥 SK, CONNECT 阶段结束. 身份认证阶段流程如图 4.

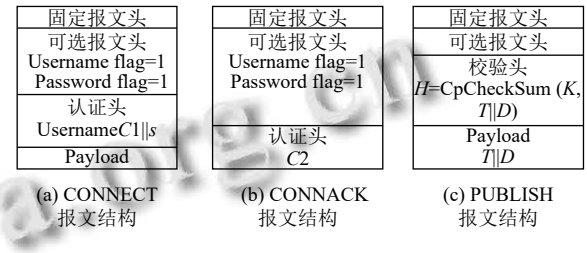

图 3 报文结构

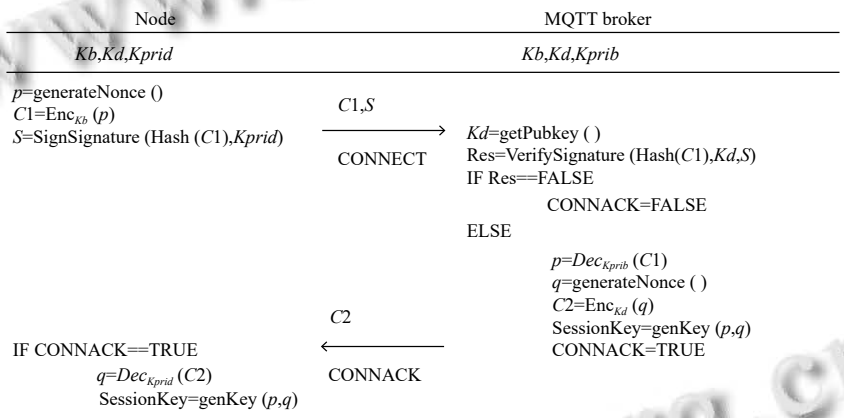

图 4 身份认证流程

在原始的 MQTT 协议中, CONNECT 阶段的身份认证只依赖于用户名 username 和密码 password 的匹配. 数据包在网络传输过程中若不采用 TLS, 将被攻击者轻易破解. 本方案依赖于这两个字段进行安全扩展, 只对报文结构进行小范围的修改, 阻止伪造数据包的攻击、中间人攻击, 以及重发攻击. 每次连接都会约定新的会话密钥, 用于安全传输与完整性监控.

2.5.4 PUBLISH 阶段

在该阶段, 设备节点将自身收集到的完整性度量信息发送给 MQTT broker, broker 确保数据来自合法节点, 但不会对收集的度量信息进行验证, 验证过程将由下一阶段的验证节点完成, 从而将完整性验证与身份认证功能分离, 有效缓解常规证明方案的同步开销以及计算瓶颈.

本方案支持第 2.5.2 节中的内核模块方式实现系统中进程信息的完整性度量, 设备插入内核模块, 从内核态对系统中运行程序的内存数据进行完整性计算获得度量值 D. 采用 MQTT 协议传输度量信息, 设定周期性校验时间, 每固定时间后执行一次完整性检验, 并采用修改后的 MQTT 协议向 broker 推送 PUBLISH 报文. IoT 设备往往针对专用应用场景, 因此运行环境中进程固定, 且数量有限, IoT 设备的完整性校验无需较大时间开销.

设备节点也可采用 SMART[23] 等混合架构软件证明方式, 将校验代码与信息传输代码放入非易失写保护内存区域 A, A 中代码周期性进行进程度量值计算得到 D. 在度量值计算完成后, 由改进后的 MQTT 协议负责度量数据包的传输工作.

本阶段对 MQTT 协议的 PUBLISH 报文字段进行了修改, 在现有报文头和可选头结构的基础上加入了校验头, 将度量值 D 与当前时间 T 进行拼接计算 Hash 值, 并使用 SM4 算法和会话密钥 SK 进行了对称加密获得了校验值 H, 表示如下:

H=ComputeCheckSum(K, $T||D$)

将校验值 H 放入新加入的校验头, 将度量值 D 与时间 T 放入报文 payload 中, 构造新的 PUBLISH 报文. 将新报文推送给 broker, 等待 broker 验证该报文是否来自合法节点, 并向验证者异步推送消息内容. PUBLISH 报文结构如图 3(c).

当使用上述两种方式推送度量信息后, broker 收到并验证校验值 H 与进程度量值 D、时间 T, 方式如下:

CheckCheckSum(K, $T||D$, H)

该步骤主要用于判断度量数据包是否来自可信的合法节点, 以及数据包是否完整, 并不会验证终端节点度量值, 度量值的校验会交给后续的验证节点进行. broker 收到数据包后将时间 T 与当前时间进行比对, 防止重放攻击. 如果攻击者劫持了终端设备节点, 向 broker 重复发送之前的合法数据包, broker 通过比对当前时间与数据包中时间 T 即可认定节点处于不可信状态. broker 记录收到节点数据包的最新时间, 一旦两次数据包时间间隔超出容忍范围, 可认为节点为不可信状态 (可能遭到攻击, 也可能网络堵塞超时).

Broker 验证完成后, 去除 PUBLISH 包中校验头. 将 payload 数据加入 broker 维护的各验证者的消息队列, 等待 broker 异步推送给网络中验证者. 这一步骤也可以使用 broker 与验证者约定的会话密钥计算新的校验头, 保证 broker 与验证者之间的数据完整性.

PUBLISH 数据报文推送与校验完整流程如图 5.

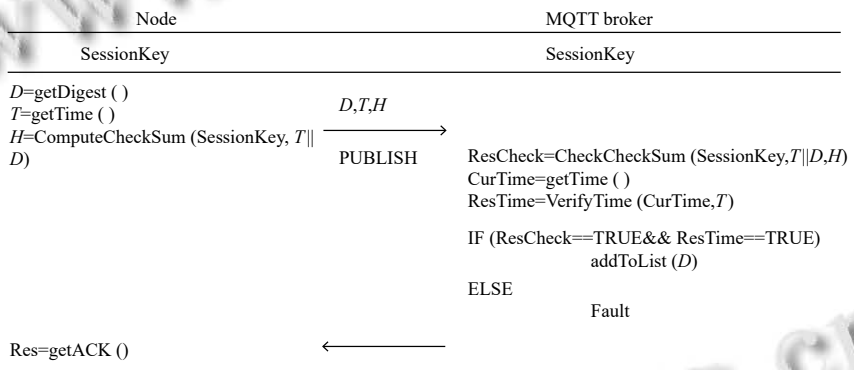

图 5 PUBLISH 数据报文推送与校验

2.5.5 验证阶段

验证节点首先向 MQTT broker 订阅需要进行完整性度量的设备的主题, 当终端 IoT 设备的 PUBLISH 包到达后, broker 异步向验证节点推送 PUBLISH 中的完整性校验信息, 由于对时间的约束已经在终端节点与 broker 的交互中进行, 验证节点只需关注完整性度量值的计算与比对.

验证节点负责最后的度量值验证工作, 实现完整性监控功能. 其上存储了 IoT 设备中的进程副本, 在验证过程中进行与 IoT 终端节点相同的度量值计算, 得到度量值 $D1$, 比对 broker 推送过来的 D, 确定终端节点是否处于可信状态, 进程是否被恶意篡改.

验证阶段流程如图 6 所示. 本方案对验证节点的控制有较大灵活性, 验证节点可以存在多个, 可随时离线、上线、添加、删除, 订阅度量相关主题后即可获得 broker 转发的度量结果, 只要该度量节点存在合法副本即可完成完整性校验的过程.

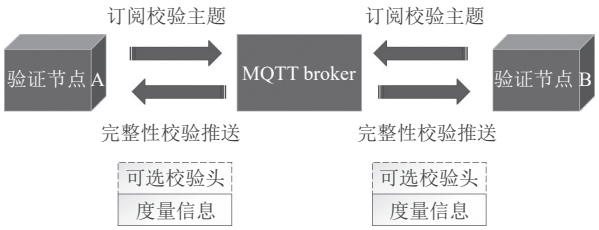

图 6 验证节点验证流程

3 方案实现和安全性分析

3.1 安全性分析

当攻击者篡改了设备节点的进程或运行了不可信程序后, 验证节点计算的完整性校验必然与合法状态存在不同. 当校验数据包经 broker 转发到达验证节点

后, 验证节点经比对就会判断出设备节点处于非安全状态, 因为进程数据被篡改后其校验结果必然不同.

当攻击者企图实施中间人攻击时, 由于无法获知节点端 (验证者、设备节点) 的私钥, 因此无法计算签名, 也无法解密得到 broker 的随机数. 因此攻击者无法生成会话密钥, broker 通过验证签名也可以检测到非法身份.

当攻击者企图伪造 PUBLISH 数据包欺骗 broker 企图跳过完整性检测时, broker 通过 PUBLISH 包中的校验头部确定数据包是否来自合法节点, 实现身份认证功能. 因为校验头采用会话密钥计算, 该密钥只有合法的约定双方持有, 攻击者无法或者约定的随机数, 进而计算出会话密钥.

当攻击者企图利用之前的数据包进行重放攻击时, broker 可比对数据包中时间, 若不在延迟容忍的范围内, 认为数据包为非可信的状态, 可能是重放攻击, 也可能是网络延迟过大.

3.2 方案实现与评估评价

在本节对方案涉及到的 3 个模块进行性能测试, 包括 CONNECT 阶段的身份认证流程、内核模块完整性度量流程, 以及完整性监控的校验头生成和验证流程.

3.2.1 CONNECT 阶段身份认证的性能开销

在我们的方案中, CONNECT 阶段包含 3 个主要部分: 设备节点生成校验头, MQTT broker 校验并构造 CONNACK 报文的校验内容, 设备节点解析安全信息. 3 个步骤之后, 设备节点与 MQTT broker 成功进行身份认证, 并协商交互通信的会话密钥. 开销数据如图 7(a).

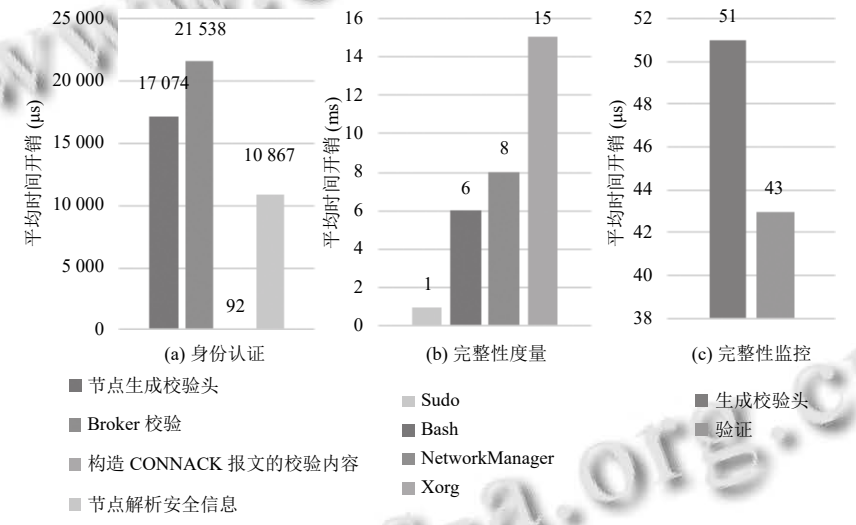

图 7 性能开销

(1) 设备节点生成校验头

该过程中, 设备节点加密自身随机数, 并计算签名, 生成 CONNECT 报文的校验头. 经过 1 000 次测试, 本过程的平均时间开销为 17 074 μs.

(2) MQTT broker 校验并构造 CONNACK 报文的校验内容

该过程中, MQTT broker 收到了设备节点发送的 CONNECT 报文, 截取校验头使用终端节点公钥验证校验头签名. 验证通过后, broker 使用自身私钥解密设备节点的随机数. 经过 1 000 次测试, 本过程的平均时间开销为 21 538 μs.

MQTT broker 构造 CONNACK 报文, 使用设备节点公钥加密自身随机数, 构成 CONNACK 报文发送给设备节点, 并计算出会话密钥. 经过 1 000 次测试, 本过程的平均时间开销为 92 μs.

(3) 设备节点解析安全信息

设备节点收到 MQTT broker 的 CONNACK 报文后, 解密出 broker 的随机数, 并生成会话密钥. 经过 1 000 次测试, 本过程的平均时间开销为 10 867 μs.

(4) 总体时间

在 CONNECT 认证阶段, 进行了 100 次总体测试, 得到的平均时间开销为 61 106 μs. 不考虑网络传输的时间影响, 设备节点端的开销约为 27 ms, broker 端开销约为 21 ms. 而无认证方案的 MQTT CONNECT 开

销为 452 μs.

3.2.2 IoT 节点完整性度量性能

对系统内进程进行完整性度量, 度量结果如表 2. 并选择了不同大小的程序分别进行 1 000 次测试, 以其中 4 种为例, 大小为 149 128 字节的 sudo, 964 220 字节的 bash, 2 271 600 字节大小的 Xorg 平均测量开销如图 7(b).

表 2 完整性度量结果

进程名	度量值
cat	8d0762778010ba4109bc8800bc3035208d12938d
sudo	6d4e7c82a931b33b4264e021a92cb677ae506373
bash	62079ee60b2f70c51c40a913b07a18e65517c734
Xorg	d56b5e52102746bdc6a10d8442debe1c73898b2d
mission-control	a4aff032a3650e94e42cd87a1a32bffa9caa3567
cupsd	216bbc0eb8cb579a8a3bd847dedcdea45e223194

ELF 文件大小为 1.1 MB 的程序 NetworkManager 如图 7(b) 所示, 其代码段大小为 1 130 108 KB, 进行 1 000 次度量测试平均用时 8 ms. 度量结果信息占用空间 634 字节. 综合考虑平衡性与可用性, 本方案建议设定周期性检查时间为 3 min.

3.2.3 PUBLISH 阶段校验头生成与验证性能

设备节点在与 MQTT broker 节点建立连接后, 向 broker 推送 PUBLISH 报文. PUBLISH 报文在度量内容的基础上加入了校验头, 使用 CONNECT 阶段约定的会话密钥进行 MAC 校验. 设备节点对度量内容进行 SM4 哈希计算, 并使用会话密钥加密产生 MAC 认证码. 经过 1 000 次测试, 本过程的平均时间开销为 51 μs.

Broker 收到 PUBLISH 报文后对度量内容进行相同 MAC 认证码计算, 与报文中校验头内容进行比对验证, 确认度量内容来自合法节点. 经过 1 000 次测试, 本过程的平均时间开销为 43 μs. 测量结果如图 7(c).

4 讨论

在本方案中, 验证节点负责对全部终端设备节点进行完整性校验与比对, 尽管本方案采用了异步校验的方式, 对时间没有较大的约束性, 但海量终端设备数量会限制验证节点的验证效率.

因此, 我们设计了基于 MQTT 的分层验证扩展, 后续将会进一步的实验改进, 由每个一级验证节点负责一种类型或一定范围终端设备的完整性验证过程, 这一流程跟基础方案一致. 二级验证节点负责对一级验证节点进行完整性校验, 即二级验证者订阅一级验证者相关主题, 一级验证者对下属终端设备节点进行验证后, 计算自身完整性校验值并附带对下属节点的验证结果, 发布给 MQTT broker. MQTT broker 将该消息包转发给二级验证者. 二级验证者只需验证下属的一级验证者, 确保其可信状态, 并获得所有的终端设备节点的校验结果.

5 结论

本方案在通用 MQTT 协议的基础上扩展 IoT 设备身份和完整性证明, 实现了轻量级的 IoT 设备软件证明功能. 在设备节点系统中采用内核模块度量方式提供了系统级完整性度量, 并依赖 MQTT 协议安全扩展设计实现了轻量级的完整性监控协议. 本方案具有轻量级异步软件证明、通用 MQTT 安全扩展等特点. 它能够有效应对海量 IoT 设备的远程证明应用需求, 减少了同步证明开销和验证节点计算开销, 避免了多验证节点并发证明请求对设备造成的可用性影响, 更利于 IoT 网络的动态调整与升级.

参考文献

1 Liyanage M, Braeken A, Kumar P, *et al*. IoT Security: Advances in Authentication. Hoboken: John Wiley & Sons, 2020.

2 He WJ, Golla M, Padhi R, *et al*. Rethinking access control and authentication for the home Internet of Things (IoT). Proceedings of the 27th USENIX Conference on Security Symposium. Baltimore: USENIX Association, 2018. 255–272.

3 OASIS. MQTT version 3.1. 1 plus errata 01. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html. (2015-12-10).

4 OASIS. MQTT version 5.0. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. (2019-03-07).

5 Jia Y, Xing LY, Mao YH, *et al*. Burglars' IoT paradise: Understanding and mitigating security risks of general messaging protocols on IoT clouds. Proceedings of 2020 IEEE Symposium on Security and Privacy (SP). San Francisco: IEEE, 2020. 465–481.

6 Hintaw AJ, Manickam S, Karuppayah S, *et al*. A brief review on MQTT's security issues within the Internet of Things (IoT). Journal of Communications, 2019, 14(6): 463–469.

7 Harsha MS, Bhavani BM, Kundhavai KR. Analysis of vulnerabilities in MQTT security using Shodan API and implementation of its countermeasures via authentication and

ACLs. Proceedings of 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI). Bangalore: IEEE, 2018. 2244–2250.
8 冯登国, 刘敬彬, 秦宇, 等. 创新发展中的可信计算理论与技术. 中国科学: 信息科学, 2020, 50(8): 1127–1147.
9 Feng W, Qin Y, Zhao SJ, *et al*. AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS. Computer Networks, 2018, 134: 167–182. [doi: 10.1016/j.comnet.2018.01.039]
10 Spinellis D. Reflection as a mechanism for software integrity verification. ACM Transactions on Information and System Security, 2000, 3(1): 51–62. [doi: 10.1145/353323.353383]
11 Seshadri A, Perrig A, van Doorn L, *et al*. SWATT: Software-based attestation for embedded devices. Proceedings of the 2004 IEEE Symposium on Security and Privacy. Berkeley: IEEE, 2004. 272–282.
12 Seshadri A, Luk M, Perrig A, *et al*. SCUBA: Secure code update by attestation in sensor networks. Proceedings of the 5th ACM Workshop on Wireless Security. Los Angeles: ACM, 2006. 85–94.
13 Li YL, McCune JM, Perrig A. VIPER: Verifying the integrity of PERipherals' firmware. Proceedings of the 18th ACM Conference on Computer and Communications Security. Chicago: ACM, 2011. 3–16.
14 Li YL, McCune JM, Perrig A. SBAP: Software-based attestation for peripherals. Proceedings of the 3rd International Conference on Trust and Trustworthy Computing. Berlin: Springer, 2010. 16–29.
15 Shaneck M, Mahadevan K, Kher V, *et al*. Remote software-based attestation for wireless sensors. Proceedings of the 2nd European Workshop on Security in Ad-hoc and Sensor Networks. Visegrad: Springer, 2005. 27–41.
16 Ibrahim A, Sadeghi AR, Zeitouni S. SeED: Secure non-interactive attestation for embedded devices. Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks. Boston: ACM, 2017. 64–74.
17 Asokan N, Brasser F, Ibrahim A, *et al*. SEDA: Scalable embedded device attestation. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. Denver: ACM, 2015. 964–975.
18 Ambrosin M, Conti M, Ibrahim A, *et al*. SANA: Secure and scalable aggregate network attestation. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna: ACM, 2016. 731–742.
19 Ibrahim A, Sadeghi AR, Tsudik G. US-AID: Unattended scalable attestation of IoT devices. Proceedings of the 37th Symposium on Reliable Distributed Systems (SRDS). Salvador: IEEE, 2018. 21–30.
20 Kuang BY, Fu AM, Yu S, *et al*. ESDRA: An efficient and secure distributed remote attestation scheme for IoT swarms. IEEE Internet of Things Journal, 2019, 6(5): 8372–8383. [doi: 10.1109/JIOT.2019.2917223]
21 杜变霞, 秦宇, 冯伟, 等. 面向物联网的高效集群证明机制. 计算机系统应用, 2018, 27(10): 22–32. [doi: 10.15888/j.cnki.csa.006626]
22 刘孜文, 冯登国. 基于可信计算的动态完整性度量架构. 电子与信息学报, 2010, 32(4): 875–879.
23 Eldefrawy K, Tsudik G, Francillon A, *et al*. SMART: Secure and minimal architecture for (establishing dynamic) root of trust. Proceedings of the 19th Annual Network and Distributed System Security Symposium. San Diego: The Internet Society, 2012. 1–15.

(校对责编: 孙君艳)