Burglars’ IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds

窃贼的物联网天堂:理解并降低物联网云上通用信息协议的安全风险

目录

[Burglars’ IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds 1](#_Toc36237667)

[窃贼的物联网天堂:理解并降低物联网云上通用信息协议的安全风险 1](#_Toc36237668)

[摘要 3](#_Toc36237669)

[第一章．介绍 4](#_Toc36237670)

[MQTT中的安全风险 5](#_Toc36237671)

[攻击 5](#_Toc36237672)

[影响 6](#_Toc36237673)

[贡献 7](#_Toc36237674)

[第二章．背景 7](#_Toc36237675)

[A.基于云的物联网通信 7](#_Toc36237676)

[体系结构 7](#_Toc36237677)

[MQTT 及其在IOT中的应用. 8](#_Toc36237678)

[B.MQTT对物联网云的保护 9](#_Toc36237679)

[客户端身份验证. 9](#_Toc36237680)

[客户授权. 10](#_Toc36237681)

[C.威胁模型 10](#_Toc36237682)

[第三章．定制的MQTT物联网的安全性分析 10](#_Toc36237683)

[A.未经授权的MQTT消息 11](#_Toc36237684)

[未授权的Will消息. 11](#_Toc36237685)

[通过Will消息进行攻击. 12](#_Toc36237686)

[讨论. 13](#_Toc36237687)

[未经授权的保留信息. 13](#_Toc36237688)

[POC 攻击. 14](#_Toc36237689)

[负责任的披露. 14](#_Toc36237690)

[B.管理MQTT会话的错误 14](#_Toc36237691)

[未更新的会话订阅状态 14](#_Toc36237692)

[未更新的会话生命周期状态 15](#_Toc36237693)

[攻击 16](#_Toc36237694)

[负责任披露 17](#_Toc36237695)

[C.未经身份验证的MQTT身份 17](#_Toc36237696)

[*ClientId*劫持 17](#_Toc36237697)

[*ClientId识别* 18](#_Toc36237698)

[讨论 20](#_Toc36237699)

[负责任披露 20](#_Toc36237700)

[D．MQTT主题的授权之谜 20](#_Toc36237701)

[保护MQTT主题的不安全快捷方式 21](#_Toc36237702)

[攻击 22](#_Toc36237703)

[MQTT的表达语法 22](#_Toc36237704)

[POC实验 23](#_Toc36237705)

[负责任披露 24](#_Toc36237706)

[四、测量 24](#_Toc36237707)

[A.范围和幅度 24](#_Toc36237708)

[B.泄露的MQTT消息的隐私含义 26](#_Toc36237709)

[五．缓解 28](#_Toc36237710)

[A.管理协议标识和会话 29](#_Toc36237711)

[B.面向消息的访问控制模型 29](#_Toc36237712)

[C. 执行与评估 31](#_Toc36237713)

[六、讨论与未来工作 32](#_Toc36237714)

[七、相关工作 33](#_Toc36237715)

[八、结论 34](#_Toc36237716)

[附录 35](#_Toc36237717)

[A.从泄漏的MQTT消息推断用户的例程 35](#_Toc36237718)

# 摘要

随着物联网(*IoT*)的日益普及，许多物联网云平台应运而生，帮助物联网制造商将设备连接到用户。服务于设备-用户通信的是部署在平台上的*通用消息传递协议*（*general messaging protocol*）。然而，不太清楚的是，这样的协议是否会带来新的风险。在本文中，我们报告了对主要物联网云(如AWS、微软、IBM)的保护的首次系统研究，该研究针对可以说是最流行的消息传递协议*MQTT*。我们发现这些协议中增加的平台安全都是脆弱的，允许对手控制设备，发动大规模的拒绝服务攻击，窃取受害者的秘密数据，伪造受害者的设备状态进行欺骗。我们成功地对这些受欢迎的物联网云进行了端到端攻击，并进一步进行了测量研究，这表明我们的攻击对安全的影响是真实、严重和广泛的。我们向有关方面报告了我们的调查结果，他们都认识到其重要性。我们进一步提出了新的设计原则和增强的访问模型*MOUCON*。我们在一个流行的开源*MQTT*服务器上实现了我们的保护。我们的评估显示了它的高有效性和可忽略的性能开销。(针对于MQTT协议在主要物联网云平台上的脆弱性研究，对其进行端到端攻击，并针对发现的安全问题，设计出新的增强性的访问模型,并验证该模型在MQTT服务器上)

# 第一章．介绍

物联网(IoT)设备的普及以及对其方便部署和控制的需求，催生了一种新型服务——物联网云平台。这样的平台已被物联网设备制造商广泛部署到使用户能够远程控制各种设备，例如智能设备随时随地的锁、开关、恒温器等（图.1）。如今许多领先的云服务提供商和物联网设备制造商提供此类云服务。例子包括AWS IoT Core [1]，Microsoft的Azure IoT中心[2]，和三星的SmartThings [3]。这些服务的核心是在物联网设备和用户之间协调通信(例如控制命令和消息)的机制。这种沟通建立在现有的*general messaging protocols*上，尤其是MQTT（消息队列和遥测运输）。MQTT是OASIS标准[4]，并且设计用于支持弱计算力的通信设备通过不可靠和低带宽的通道，因此非常适合物联网用户交互。因此主流物联网云广泛采用[5]，包括那些由AWS，Microsoft，IBM，阿里巴巴，Tuya，Google等提供。

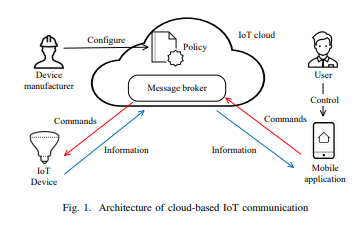


图1

### MQTT中的安全风险

不幸的是MQTT并未以在对抗环境中运作而设计，因此它无法保护自己免受来自于物联网系统的潜在威胁。更具体地说该协议几乎没有内置的认证和授权机制，强制云平台提供商制定自己的保障措施。（核心问题）鉴于定制通用的适用于各种物联网应用场景的协议的复杂性，和有效保护其通信过程是具有挑战性。在缺乏适当的安全分析的情况下，很难确定用户-设备交互是否得到了有效的保护。MQTT协议扮演的关键角色，任何安全弱点被暴露，可能会造成严重后果，例如丧失设备控制，敏感用户数据的公开等。但是，由于问题的重要性，到目前为止，还没有做多少工作来了解MQTT及其支持的通信机制是否得到了充分的保护。

### 攻击

在我们的研究中，我们首次系统地研究了在主流物联网云上集成通用消息传递协议（*general messaging protoco*）的安全威胁。(工作的创新性)*STEP1*作为第一步，我们将重点放在*MQTT*上，它可能是采用最广泛的跨越物联网云平台协议[5]。我们的研究表明，云对MQTT的安全添加常常是脆弱的(第三节)。利用这些漏洞对手可以(1)控制受害者的设备(如开锁门窗)，(2)推断敏感的用户信息，如她的日常生活，健康状况，地点，同居者，(3)对大量设备实施拒绝服务(DoS)攻击，(4)冒充目标设备到云上操作传递给用户的消息。这种攻击的后果是重大的，有时甚至是毁灭性的。例如，我们发现苏宁智能生活的弱授权使得远程敌手可以收集由云管理的所有设备产生的所有事件，包括可识别用户信息(如电子邮件、电话号码)和设备信息。

展望这些缺陷的根源, 我们发现大多数物联网云今天无法意识到安全运行在一个相对简单的影响一般消息传递协议在复杂的物联网环境中,敞开了大门时新的风险协议和实际设备使用方式场景之间的差距并没有被得到适当的保护。特别是，我们发现MQTT中没有考虑到的撤销过程对其在物联网通信中的应用提出了严重的安全挑战。根据[6]-[10]最近的研究，如今，物联网设备(如智能锁)越来越多地在用户之间共享（威胁产生的应用场景），例如，酒店居住者、Airbnb公寓租户、家庭访客(如保姆等)。一旦一个人对一个设备的访问权被终止，没有适当的授权，他不应该被允许再次连接到它。然而，这种保护很难在主流平台上实施，因为以实用主义设计为导向的协议为用户提供了各种访问设备的方式:我们的研究表明，通过一个主题（topic）、一个会话(session)和其他实体(entities)，恶意的前用户可以保留对其访问权限已经过期的设备的完全控制，甚至可以接收当前用户的私人信息，（工作的发现）如健康状况、同居关系等(章节III-A、III-B和III-D)。还发现原始协议中的安全敏感状态和状态转换还没有完全确定，并且在敌对环境中运行该协议时没有得到保护。例如，MQTT包含一个ClientId字段作为每个客户端的唯一标识符，当出现具有相同ClientId的新客户端时，云将断开现有客户端的连接。这个实体在原始协议中不受保护，因为它是为在一个良好的环境中运行而设计的。然而,当协议运行在一个物联网云,没有涉及实体的身份验证,我们发现对手可以滥用协议状态断开客户端通过声称他们的ClientIds在大规模下,导致一个DoS攻击甚至MQTT会话劫持(III-C节)。（会话劫持：会话 ，两台主机间的一次通讯；分类：中间人，注射式攻击；步骤：嗅探正在运行的会话，猜测正确的序号seq,发送包给目标主机目标主机与攻击者建立会话 断开现有会话）

### 影响

为了了解这些安全漏洞的影响，我们进行了一项测量研究，分析了8个领先的物联网云平台(包括AWS、IBM、微软等提供的云平台)。我们的研究揭示了安全风险的普遍性和重要性:大多数这些平台包含如此关键的缺陷在他们定制的*MQTT*协议,让成千上万的设备制造商和数以百万计的用户遭受上述危害(第四节)。我们所有发现的问题报告给受影响的各方,包括AWS、微软、IBM、阿里巴巴,百度,Eclipse Mosquitto,等等,并与他们合作来解决这些问题。最重要的是，我们的发现已经引起了MQTT技术委员会(TC)的注意，该委员会正在通过现在的[11]公开讨论寻求解决方案。我们的攻击视频被发布在[12]上。

鉴于所发现问题的广泛性和重要性，我们认为有必要对物联网云的通用消息传递协议的认证和授权保护进行标准化。作为第一步，我们提出了一组安全设计原则，并在一个流行的开源MQTT服务器上进行了实现。（对于问题的解决方案）我们的评估表明，这种保护只会产生微不足道的开销，并且有效地解决了我们发现的所有问题。我们正在与MQTT TC就解决方案进行沟通，并帮助他们加强协议的保护。我们还在网上发布了我们的技术代码[14]。

### 贡献

论文的贡献如下:

**对安全物联网通信的新理解.**

我们首次对物联网设备-用户通信通用消息传递协议的安全风险进行了系统研究。我们的研究揭示了在简单和良好的环境下运行的协议与复杂的环境下运行之间的差距,敌对的物联网使用场景, 以及用适当的安全手段(通过扩展协议来覆盖新的场景，如撤销和保护现有的安全敏感状态和转换)来弥补缺陷的挑战。从这项研究中得到的教训将有助于更好地设计和实现在敌对环境中工作的其他协议。

**措施**.

我们展示了领先的物联网云平台和流行的开源MQTT服务器的安全风险的普遍性，并确定了攻击所发现的漏洞的严重后果。

**安全的设计原则和实现**.

我们提出了定制物联网云通用消息传递协议和实现端到端保护的新安全原则。我们的方法是有效和高效的，可以方便地集成到当今的物联网平台。通过与MQTT TC和物联网云平台的通信，这第一步可以更好地保护现实世界中的用户设备交互。

# 第二章．背景

## A.基于云的物联网通信

### 体系结构

基于云的物联网系统通常包括三个组件:**云(本文也称云平台)、物联网设备和控制设备的用户管理控制台(特别是移动应用程序)**，如图1所示。核心系统的云管理设备和应用程序之间的通信,通过应用程序发送控制消息(命令)和设备(如锁智能门)从设备得到信息(例如,从一个恒温器温度,一个锁的打开或关闭状态)。为了保护这种交互，云认证**设备和应用程序**(代表用户)，并决定是否允许用户命令设备或从设备接收消息。为此，云提供商提供实现特定消息传递协议(如MQTT)的sdk，这些协议由物联网制造商集成到他们的设备和移动应用程序中通过云端进行通信。这一模式已被大多数物联网云提供商(如AWS、微软、

IBM、图亚、阿里巴巴)和设备制造商[15]、[16]。

### MQTT 及其在IOT中的应用.

一个发布-订阅消息传递协议允许发送方根据消息的主题或其内容将消息传递到由一组接收方订阅的类中[17]。一个突出的例子是MQTT1，它是一个应用层协议(基于OSI模型[19])，运行在TCP/IP或其他有序的、无损的、双向的连接(如WebSocket[20])上。MQTT以其轻量级设计而闻名，它适用于低带宽或不可靠网络中的资源紧张设备，因此非常适合支持物联网生态系统。

基于MQTT的物联网通信的核心是MQTT message broker(本文简称为broker)，如图1所示。broker代理在其服务器端点托管MQTT主题，每个主题的结构类似于分层文件路径，如/doorlock/[UUID-8JH…S9P]/status。以代理作为连接轴心。MQTT利用订阅发布模式[17]实现通信：MQTT客户端(例如物联网设备或管理应用程序)将消息发布到承载特定主题的代理，然后代理将消息路由到订阅了该主题的其他客户端。客户端可以订阅主题层次结构中的特定级别，也可以使用通配符(#)连接到多个级别。

在MQTT通信中，客户机向代理发送三种基本类型的消息:CONNECT、PUBLISH和SUBSCRIBE，如图2所示。（Step1）首先，MQTT客户机(例如智能空调或应用程序)向代理发送连接消息，用于建立MQTT会话(如果代理接受连接)。会话和客户机由一个ClientId字段(嵌入在CONNECT消息中)惟一标识，该字段类似于web会话cookie。在已建立的会话中，物联网设备订阅其相关主题(例如/DeviceId/cmd)通过向代理发送的SUBSCRIB消息(包括其主题)。代理为每个会话维护订阅状态，并将发布到主题的MQTT消息传递给它的订阅者。通过这个渠道，应用程序可以代表其用户对设备进行操作，发布对设备订阅的主题的评论(例如，开始或停止)。此外，该设备可以定期将其状态信息更新到某个主题，比如当前温度，订阅该主题的所有应用程序都会接收到当前温度。

在此过程中，整个MQTT通信依赖于四个实体:标识(ClientId)、消息、主题和会话。因此，这些实体是否得到了良好的保护对于协议在物联网环境中的安全应用至关重要。

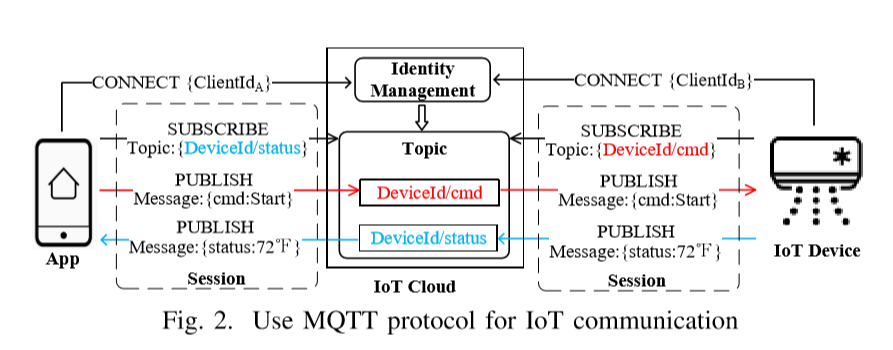


图2

## B.MQTT对物联网云的保护

作为一种通用的消息传递协议，MQTT并不是为在敌对环境中工作而设计的:例如，它缺少内置的身份验证和授权。保护涉及敏感物联网通信设备(如门锁、心脏设备,安全摄像头,火灾探测器),尤其是这类设备收集私人信息,一个物联网云往往有其自定义安全机制MQTT客户机进行身份验证和授权当事人资格发布或订阅MQTT主题。在此，我们通过对8种物联网领先云的研究，总结了在物联网领先云上部署的这种保护措施。

### 客户端身份验证.

MQTT连接通过WebSocket（HTML5中的websocket 一种基于单个TCP的全双工通信协议）和TLS[21]，它们使用部署在不同物联网云上的各种机制进行身份验证。例如，AWS IoT支持Amazon用户名/密码、单点登录(SSO在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。)（单点登录）[22]到谷歌/Facebook、Amazon Cognito[23](一种跨AWS云服务的登录方案)。另一个例子是，在TLS连接中，客户端使用TLS客户端身份验证模式[24]向云提供一个密码证书（基于PKI），这是对设备进行身份验证的典型方法，其中可能包含内置的证书.

### 客户授权.

物联网云平台旨在确保每个用户只能向被允许使用的设备发送命令和接收消息。为此，云强制执行一组安全策略。示例包括允许客户端访问的主题和消息，以及它可以采取的操作(例如，发布或订阅)

## C.威胁模型

我们假设对手可以通过物联网设备制造商和物联网云开放用户账户，并且能够收集和分析自己控制的物联网云、物联网设备和app之间的网络流量。另一方面，他不能窃听其他用户的设备和应用程序的通信。(前提假设)

在我们的一些攻击中，我们考虑了今天变得普遍的设备共享情况。酒店、Airbnb、公寓和其他度假租赁房屋越来越多地配备了物联网设备，他们的客人通常可以临时使用这些设备[7]-[10]。最近的一项研究[6]进一步表明，大多数人愿意分享他们的设备(如智能锁)，例如，与家庭访客或保姆等。在这种情况下，物联网接入控制模型有望以安全可靠的方式处理撤销，这完全超出了最初的MQTT协议本意。注意，除了对撤销的研究外，我们不假设在其他攻击中攻击者被授予对目标设备的临时访问权。

SECURITY ANALYSIS OF MQTT IOT CUSTOMIZATION

# 第三章．定制的MQTT物联网的安全性分析

In this section, we report a security analysis on the MQTT based communication mechanisms operated by leading IoT cloud platforms. Our study systematically inspected individual protocol entities, including identity, message, topic and session, in an attempt to understand whether their related security-sensitive protocol states have been properly guarded or whether they could be abused to circumvent the IoT protection MQTT does not support. The study shows that such security gaps do exist and oftentimes, these entities have not been properly authenticated or authorized to cover the gaps. To understand the security implications of the ﬁndings, we further implemented end-to-end attacks on all the problems discovered, with their demos posted online [12]. It is important to note that all such experiments were conducted in an ethical way: we always aimed at our own devices, never putting the cloud services, platforms and other users in danger. Also importantly, we reported all our ﬁndings to the manufacturers and IoT cloud providers, sharing our PoC attacks with them and helping them improve their security protection. This effort has been well received and acknowledged, and we received 6,700 USD in total from those vendors’ bug bounty programs.

在本节中，我们将报告对领先的物联网云平台运行的基于MQTT的通信机制的安全性分析。我们的研究系统地检查了单个协议实体，包括**身份、消息、主题和会话**，试图了解它们相关的安全敏感协议状态是否得到了适当的保护，或者它们是否可能被滥用来规避MQTT不支持的物联网保护。研究表明，这种安全漏洞确实存在，而且通常这些实体没有得到适当的身份验证或授权来弥补这些漏洞。为了理解这些发现的安全含义，我们进一步对发现的所有问题实施端到端攻击，并将它们的演示发布到网上[12]。值得注意的是，所有这些实验都是以道德的方式进行的:我们总是以自己的设备为目标，从未将云服务、平台和其他用户置于危险之中。同样重要的是，我们向制造商和物联网云提供商报告了我们的所有发现，与他们共享我们的PoC攻击，并帮助他们改善安全保护。这个努力已经被很好地接受和承认了，我们从那些供应商那里总共收到了6700美元的bug奖励计划。

A. Unauthorized MQTT Messages

## A.未经授权的MQTT消息

As mentioned earlier, MQTT was not designed to work in the diverse scenarios of IoT communication. A prominent example is device sharing and revocation, in which a party (hotel dweller, Airbnb tenant, babysitter, etc.) is only trusted with temporary access to an IoT device and not allowed to get information from or interfere with the activities of prior and future users of the same device. This potentially adversarial situation is completely outside the MQTT protocol. Without in-depth understanding of the problem, the current IoT cloud platform just cannot handle it in a secure way. Following we present the new security risks we discovered in the scenario due to the insecure management of MQTT messages ,including Will message and Retained message, which leads to unauthorized control of IoT devices.

如前所述，MQTT的设计并不适用于物联网通信的各种场景。一个突出的例子是设备共享和撤销，其中一方(酒店住客、Airbnb租户、保姆等)只被信任可以临时访问物联网设备，不允许从之前和未来使用同一设备的用户那里获取信息或干扰他们的活动。这种潜在的敌对情况完全不属于MQTT协议。没有对问题的深入了解，目前的物联网云平台无法安全处理。接下来，我们将介绍在该场景中发现的新的安全风险，这些风险是由于MQTT消息(包括Will消息和retain消息)的不安全管理导致未授权控制物联网设备。

### 未授权的Will消息.

**Unauthorized Will Message**. Based on the MQTT speciﬁcation, the client can register with the broker a special Will Message (in a CONNECT type MQTT message, see Section II) for a topic. Once the client is accidentally disconnected (i.e., not sending a DISCONNECT message to the broker), the broker will publish the Will Message to all subscribed clients of the topic, allowing them to take corresponding actions. Like other MQTT messages, Will Message can include either control commands or just text.

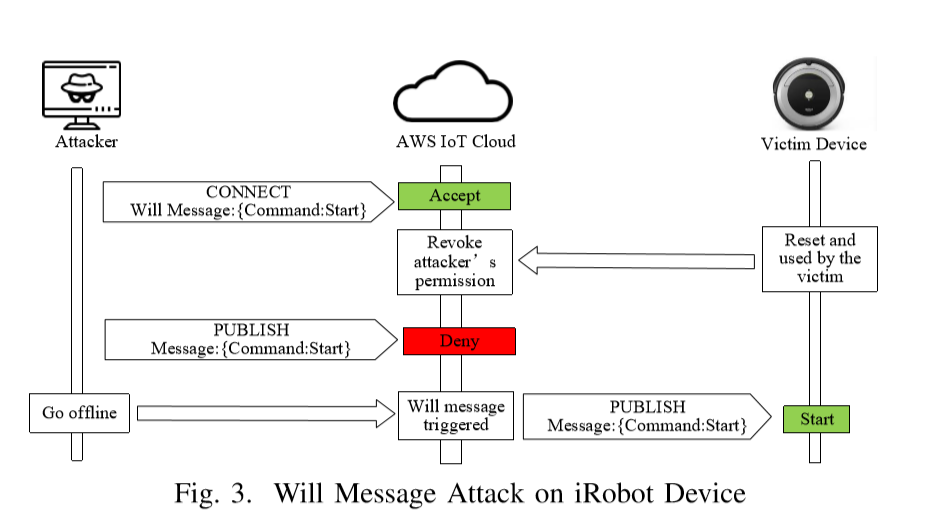
根据MQTT规范，客户机可以向代理注册一个特殊的Will消息（在CONNECT类型的MQTT消息，见第二部分）为一个topic.一旦客户端突然断开连接(即，而不是向代理发送断开连接的消息)，代理将向topic的所有订阅客户端发布will消息，允许它们采取相应的操作。与其他MQTT消息一样，Will消息可以包含控制命令，也可以只包含文本。

However, this exception handling feature was not meant to work in the adversarial environment, when the access right on a device is transferred from one party to another. In this case, a malicious ex-user can strategically register a Will Message to trigger it later when he no longer has the access privilege, to stealthily issue commands when the device is serving other users: for example, a babysitter or a repairman given access to a smart lock once could open the door for others later. Speciﬁcally, we found that even after the attacker’s privilege on a device expires, so he is prevented from direct communicating with the device by the broker based upon the owner-speciﬁed policy, on the protocol level, the attacker’s Will Message will still be issued as soon as his client is disconnected from the broker. In this way, he can decide when to unlock the door in the above example by choosing the right time to go ofﬂine. Note that, the attacker can launch the attack multiple times leveraging multiple clients to register multiple Will Messages (e.g., 10 times in our PoC attack below).

但是，当设备上的访问权限从一方转移到另一方时，这种异常处理特性并不适合在敌对环境中工作，在这种情况下,恶意ex-user可以战略性的注册一个Will消息来在之后触发当他不再有访问权限时,暗地里发出命令当设备服务其他用户:例如,一个保姆或者修理工获得智能锁一次为别人打开大门。具体来说,我们发现,即使攻击者的特权在设备上到期,他被代理阻止直接与设备通信在owner-specified政策层次上,在协议层次,攻击者will消息仍将尽快发给他的客户与代理。通过这种方式，他可以通过选择在离线状态下合适的时间来决定何时打开上面例子中的门。注意，攻击者可以利用多个客户机来注册多个Will消息，从而多次发起攻击(例如，下面的PoC攻击中有10次)（PoC(全称: Proof of Concept), 中文可译作“观点验证程序”，它主要用于证明提出者的观点是否正确，在信息安全领域这种观点一般是我们对于漏洞的判断，即目标系统是否存在漏洞。）

The problem can be addressed by removing the Will Message registration when one’s access right is revoked. This, however, is complicated since the IoT cloud platform can no longer just operate on top of MQTT, by controlling whether a user should communicate with a device based upon her platform ID, but needs to get into the protocol to expand its state machine with the capability to handle revocation (e.g., ﬁnding out each state related to the party like registering a Will Message and cleaning up each of them). This has never been thought of, up to our knowledge, and does not exist on all leading IoT clouds we studied (see Section IV).

当一个人的访问权限被撤销时，可以通过删除Will消息注册来解决这个问题。然而，这很复杂，因为物联网云平台不能再仅仅在MQTT上操作，即通过控制用户是否应该基于其平台ID与设备通信，而需要进入协议以扩展其状态机，使其具有处理撤销的能力,但是需要进入的协议扩大状态机处理能力撤销(例如,发现每个州政党登记等相关消息和清理他们每个人)。据我们所知，这一点从未被考虑过，也不存在于我们所研究的所有主要物联网云上(参见第四节)。



### 通过Will消息进行攻击.

我们使用我们的iRobot Roomba 690在AWS的物联网云上实现了一个利用Will消息的PoC攻击(图3)。具体来说，我们编写了一个脚本来注册一个Will消息，包括一个command (command:start)来启动我们的设备和代理。然后,当受害者用户(像后续客人的在同一酒店房间)重置设备和使用机器人并且我们的客户特权完全撤销(例如,拒绝发布消息到设备), 我们表明，当脚本离线时，Will消息中的命令仍然被传递并叫醒机器人。这样的攻击可以通过多次利用该问题而被放大:我们还尝试了10个恶意客户端同时注册10条Will消息，每个客户端都能够独立地发起攻击。

同样的问题已经在AWS、IBM、百度、Tuya Smart等物联网云上得到了证实。由于云中没有适当的安全检查，设备制造商也没有安全提示，这个问题可能会影响到云中的所有设备，如门窗锁、视频门铃、心脏设备、安全摄像头、火灾探测器等，这些设备具有很高的安全性、私密性，甚至对安全性很敏感。

### 讨论.

通过与MQTT技术委员会的对话和对MQTT规范的检查，我们发现问题来自于Will消息的契约类属性:一个主题上的已注册的Will消息有权交付给订阅该主题的所有客户端;消息是否违反订阅客户机的安全要求不是MQTT关心的问题。这样的协议假设MQTT是为一个更加可信的环境而设计的，而不是物联网通信的敌对环境(例如，前一个设备用户可能攻击后一个设备用户)。我们发现的问题表明，物联网供应商应该理解这一差距，并扩展该协议，以解决其安全影响。我们的发现第一次表明，弥补这一差距的重要性在很大程度上被低估了，安全风险是非常现实的。我们还发现这个问题不仅限于Will消息，还包括其他MQTT特性，稍后将对此进行详细说明

### 未经授权的保留信息.

当MQTT客户端将消息发布到某个主题，并且没有客户端订阅该主题时，代理将简单地丢弃该消息。然而，这种简单的处理还会在客户端暂时离线时干扰发布者与其订阅客户端的通信。为了解决这个问题，MQTT客户机可以注册带有主题的保留消息(通过在常规MQTT消息中设置保留标志)，这允许代理将最后保留的消息保留在主题上，并立即将其发布给主题的任何未来订阅者。

就像Will Message一样，这个功能不是设计来在对抗的物联网环境下工作的，我们再次发现它可以被一个恶意的前用户利用来暗中命令一个他不再能够访问的设备。例如，在Airbnb房间里使用过某台设备的恶意前用户可以发布一条保留的消息，其中包括任意的控制命令(例如，在凌晨3点开门)，在他仍然拥有访问权限的情况下，发布到智能锁的相关主题。稍后，当他退房并因此失去特权时，他可以等待设备重新联机。当这种情况发生时，锁将再次订阅旧主题并接收unlock命令。一旦执行，门将在凌晨3点打开，窃贼可以进入。

### POC 攻击.

我们执行PoC攻击和证实，百度的物联网云(一个最高的公共云服务提供商在中国[25]和与世界上第四大网站[26])和Eclipse Mosquitto(一个著名的开源MQTT代理[13],这是复制fork950次[27]和部署在流行的开源物联网平台[28],[29])受到保留消息攻击。在这两个平台上，我们首先运行一个恶意客户机(代表前用户)来将保留的消息注册到一个主题上，然后撤销它的常规权限。之后，当受害客户端(代表设备被不同用户使用时)订阅同一主题时，两个平台都将攻击消息转发给受害客户端，这可能导致对设备的未授权控制。

### 负责任的披露.

我们向AWS物联网、IBM、百度、Tuya Smart、Eclipse Mosquitto等受影响方报告了问题，这些都承认了问题的重要性。AWS与我们进行了一次在线会议，讨论减轻物联网用户风险的可能解决方案.

B.Faults in Managing MQTT Sessions

## B.管理MQTT会话的错误

As introduced earlier, the MQTT communication is through the established session between the client and the broker server, and each session is associated with an MQTT client. Therefore, when a client has a state change (e.g., his/her access to a device is revoked), the states of its established session should be updated, which is particularly important to security sensitive ones such as subscription state (which topics are subscribed to) and the lifecycle state (whether the session should last or be terminated) in particular. However, this expected security property is often not in place on realworld IoT platforms, as elaborated below.

如前所述，MQTT通信是通过客户机和代理服务器之间建立的会话进行的，每个会话都与MQTT客户机相关联。因此,当一个客户端有一个状态改变(例如,他/她访问设备被吊销),建立了会话的状态应该更新,这是特别重要的安全等敏感的订阅状态(主题订阅)和生命周期状态(会话是否应该延迟或终止)。然而，这种预期的安全特性通常在现实物联网平台上并不存在，如下所述。

### 未更新的会话订阅状态

**Non-updated session subscription state** MQTT speciﬁcation suggests that the server authorizes particular actions of the client[18].With this guidance in place, IoT platforms typically enforce a security policy to govern the client’s operations. For example, when a device is reset to completely remove all privileges of an ex-user on the device, in any established session, the user’s client becomes no longer permitted to take any proactive action, such as to SUBSCRIBE to the device’s topic. However, when we inspect the session’s state management, we found that the MQTT speciﬁcation provides no guidance in updating session states in response to the client’s privilege change. Likely due to such lack of guidance, session management in real-world IoT systems, particularly when it comes to a session’s subscription state, was found to have privacy-critical defects. Speciﬁcally, as long as a client establishes a session that ever subscribed to a topic (e.g., the topic of a smart speaker in a hotel room), even when the user is no longer permitted to subscribe to the topic (e.g., after checking out), we found that the broker continuously delivers messages to the client through the established session. That is, the subscription state of the session lasts even after the subscriber lost his privilege, which effectively enables the malicious ex-user of a device to continuously receive all messages generated by the device for the current user (victim), such as personal buying history and habits, health conditions and data (e.g., heartbeats), etc. Such an unsound session state management is conﬁrmed on major IoT clouds (e.g., IBM, Tuya, Alibaba, Baidu, see responsible disclosure below).

MQTT规范建议服务器授权客户端[18]的特定操作。有了这个指导，物联网平台通常执行一个安全策略来管理客户端操作。例如，当设备被重置为完全删除前用户在设备上的所有特权时，在任何已建立的会话中，用户的客户端将不再被允许采取任何主动操作，例如订阅设备的主题。但是，当我们检查会话的状态管理时，我们发现MQTT规范在更新会话状态以响应客户机的特权更改方面没有提供任何指导。可能由于缺乏指导，现实物联网系统中的会话管理，特别是会话的订阅状态，被发现有隐私关键缺陷。具体地说,只要客户端建立一个会话,订阅一个主题(例如, 酒店房间里智能扬声器的topic),甚至当用户不再允许订阅主题(例如,检查后),我们发现通过建立会话代理不断传递消息到客户端。即订阅状态的会话持续即使在用户失去了特权,这有效地使设备的恶意ex-user不断收到所有5消息生成的设备为当前用户(受害者),如个人购买历史和习惯,健康状况和数据(如心跳),等等。这种不健全的会话状态管理在主要的物联网云上得到了证实(例如，IBM、Tuya、阿里巴巴、百度，见下面负责任的披露)。

**Non-updated session lifecycle state.**

The MQTT client in the IoT environment can represent two different roles, the device (when the client is authenticated through a device’s credential) and the user (when the client is authenticated through a user’s credential). The two roles are managed differently by IoT clouds from the security perspective: the device is treated as the resource to access, and the user is regarded as the principle to authenticate and authorize. Such a difference is found to have security implications in the scenarios of IoT device sharing and revocation. Particularly, when a device is reset by a new user (for removing exusers’ access), permissions of the ex-user (and his/her client) for accessing the device are revoked (i.e., publish/receive messages through the device’s topic). In contrast, there is no concept of revoking the permission of a device for accessing its topic. Hence, a possible attack is: the ex-user (attacker) obtains the device credential when he/she is permitted to use the device (obtaining device credentials is oftentimes trivial, such as through trafﬁc analysis or reverse engineering, as demonstrated by a recent study [30] [31]), then even after the new user removes ex-users, the attacker can always leverage the device credential to impersonate the device and publish fake messages to the device’s topic Such an attack , however is found to be mitigated by leading IoT platforms. For example, Tuya’s IoT cloud ensures (leveraging its device SDK provided to manufacturers) that device credentials under its cloud are forced to expire if the owner changes (e.g., when the new user resets the device). Hence, with an expired device

### 未更新的会话生命周期状态

物联网环境中的MQTT客户端可以代表两个不同的角色:设备(当客户端通过设备的凭据进行身份验证时)和用户(当客户端通过用户的凭据进行身份验证时)。从安全角度看，物联网云对这两个角色进行了不同的管理:将设备视为要访问的资源，将用户视为进行身份验证和授权的源泉。这种差异被发现在物联网设备共享和撤销的场景中具有安全含义。特别是，当一个新用户重置一个设备时(用于删除ex-users访问)，前用户(和他/她的客户端)访问设备的权限将被撤销(即，通过设备的主题发布/接收消息)。相反，没有撤销设备允许访问其主题的权限的概念。因此,一个可能的攻击:ex-user(攻击者)获得设备证书时,他/她是允许使用设备(设备获取凭证通常是微不足道的,比如通过流量分析或逆向工程,根据最近的一项研究证明[30][31]),那么即使在新用户删除既往使用者,攻击者可以利用设备证书模拟设备和发布虚假的信息给设备的主题。然而，人们发现领先的物联网平台缓解了这种攻击。例如，Tuya的物联网云可以确保(利用其提供给制造商的设备SDK)，如果所有者发生变化(例如，新用户重置设备)，其云下的设备凭据将被迫过期。因此，对于过期的设备凭据，前用户不能再模拟该设备。

然而，在我们的研究中，当我们从MQTT会话管理的角度来看这个问题时，我们发现了允许攻击继续进行的领先物联网平台的一个新弱点。特别是通过获得的设备凭据，只要前用户(攻击者)在凭据过期之前建立了一个会话并保持会话在线，他/她就可以代表设备通过会话向设备的主题发布虚假消息。请注意，攻击可以在凭据过期后继续，因此不能再用于验证新客户端或建立新会话。这对安全性甚至是安全都有严重的影响:例如，窃贼或罪犯可以利用虚假信息向用户显示门是锁着的，或者煤气阀门是关着的，尽管它不是;虚假信息可以通过触发物联网平台(如IFTTT[32])触发受害者的其他敏感设备(如:开门)。

在这种攻击中，平台无意中造成的错误是:当设备被重置以清除其所有现有用户以及更基本的现有状态时，所建立的MQTT会话的生命周期状态没有更新。为了安全，会话的状态应该被清理，例如，完全终止已建立的会话。但是，MQTT规范[4]没有考虑在用户特权发生变化时会话状态更新的安全性必要性，这可能是由于它假定的使用环境比物联网简单得多，并且对抗性更小。该问题得到了IBM、Tuya、阿里巴巴、百度等公司的确认和认可(见第四节)，并在下面实现了端到端攻击，展示了真正攻击的可行性。

更重要的是，上面的两个攻击场景表明了安全的MQTT会话状态管理，即在美国，针对物联网用户权限变化而对会话状态更新进行管理的健全机制还处于研究阶段，并面临着物联网环境中新的安全风险。这些风险被成熟的MQTT开发指南和现实世界的物联网供应商严重低估。

### 攻击

利用上述弱点，我们利用我们的MiKO智能插座在Tuya智能[33](托管超过1亿台智能设备)的云上实现了PoC攻击。Step1首先，为了利用会话订阅状态，充当恶意用户，我们建立了一个会话，它能够连续接收来自设备的消息，即使在第二个用户重置设备以删除所有以前的用户之后也是如此。Step2其次，利用生命周期管理的不安全，通过逆向工程设备流量，我们能够获得设备凭据，并建立一个恶意会话。Step3之后，该设备被另一个用户(受害者)重置，以删除前用户的访问权限，但恶意会话能够持续向订阅该设备主题的受害者应用程序发布虚假的设备消息(如设备状态on)。通过这样的攻击，一个真正的窃贼或罪犯可能会闯入一个家庭，同时向受害者/警方发布虚假的设备更新(冒充家庭安全设备)，并表明这个家庭仍然是安全的。

### 负责任披露

我们向包括Tuya、阿里巴巴、百度、IBM等在内的受影响供应商报告了我们的发现，他们都承认存在问题。

## C.未经身份验证的MQTT身份

如前所述(第二部分)，物联网云平台使用其自己的平台层身份(例如AWS物联网云上的Amazon帐户)对其MQTT客户端进行身份验证。同时，每个客户端还通过其自己的协议级标识ClientId进行标识。这两个身份之间的关系可能很复杂:一个帐户可以有多个设备，每个设备都有自己的ClientId，而一个设备可以在多个帐户之间共享。如果处理不好，这种关系可能会使MQTT通信受到攻击，正如我们在研究中发现的那样。

### *ClientId*劫持

MQTT协议要求代理在观察到具有相同*ClientId*的新客户机时断开联机客户机。在一个敌对的环境中，人们会期望相关的MQTT协议状态和转换(例如，在检测到冲突的*ClientId*下将客户端移动到离线状态)通过适当的身份验证(包括*ClientId*)加以保护，并且令牌是保密的。然而，我们的研究表明，这种保护实际上并不存在于主要的物联网云平台上。因此，攻击者可以利用自己经过身份验证的平台身份，通过任意一个*ClientId*(包括属于目标设备的*ClientId*)连接到物联网云，从而迫使云脱离目标。在我们的*PoC*攻击中，我们展示了威胁是真实存在的，并且可以是大规模的，跨越我们研究的云(第四部分)，例如AWS、IBM、百度。

对*ClientId*的攻击实际上超越了*DoS*。具体来说,*MQTT*协议允许代理和客户端恢复前一会话如果客户机与先前*ClientId*(用干净的会话标记检查连接消息[4]),它允许客户端迅速恢复之前的状态(例如,订阅和等待消息应该收到),以避免配置的麻烦。然而，一旦将其*ClientId*公开给攻击者，这种恢复机制就可能被滥用来针对目标客户端工作。注意，这里的攻击者可以携带一个完全合法的平台身份。在缺乏将这两个身份连接起来并规范对相关对象的访问权限的安全策略的情况下，恶意云用户可以利用目标*ClientId*恢复其会话并窃取其消息(例如，健康状况、位置、个人习惯等)。在我们的研究中，我们成功地对*IBM* *Watson*物联网和百度云物联网实施了*PoC*攻击，攻击者虽然从未订阅过受害者的主题，但仍然可以接收到受害者的消息。我们发现这是一个现实的问题，因为在真实的物联网云上的身份管理不当，只要*MQTT*客户端证明了它的平台身份，就不会对*ClientId*进行身份验证。

### *ClientId识别*

前面提到的攻击是基于攻击者对目标*ClientId*的知识，这是完全现实的。具体来说，我们的研究表明，很明显，设备制造商和物联网云仍然没有意识到保持*ClientId*安全的重要性，允许我们通过至少两种方式获取受害者的*ClientId*，如下所述。

* 做一个猜想*ClientId*是一个序列，它的格式和语义由设备制造商决定。客户端使用它来与代理建立连接，而代理要么根本不检查它的合法性，要么只检查它的基本格式(例如，只允许使用字母和数字)。当两个客户机连接到相同的*ClientId*时，第一个连接将根据*MQTT*的要求从代理中断。利用这个属性，我们开发了一个简单的攻击来测试*ClientId*的存在性，方法是尝试使用令牌连接到代理:如果客户端(已连接的应用程序或设备)已经使用了它，连接将不断失败，因为客户端正在与攻击者竞争。否则，连接总是通过。

攻击的关键是有效地搜索已经分配的*ClientIds*。我们发现这是可行的，因为*ClientId*从来就不是一个秘密。实际上，MQTT规范只要求*ClientId*是惟一的[4]。因此，受欢迎的物联网云平台公开向客户(设备制造商)提出的*ClientId*架构应该易于管理。例如，IBM (MQTT的创建者)主张使用设备的48位MAC地址作为其*ClientId*，以确保其唯一性[34]。甚至设备制造商也没有意识到*ClientId*的安全含义:他们倾向于选择诸如序列号[35]、[36]这样的不安全格式，而这些格式通常是增量地分配给[37]的。例如，苹果产品序列号的格式是众所周知的，它的主要部分指定了制造工厂、日期、产品型号等属性，只有三个数字用于区分具有相同属性的设备[38]。

在我们的研究中，我们发现格式信息可以帮助我们轻松地发现指定的序列。例如，对于MAC地址，它的前24位与特定的制造商相关，而其余的24位可以被枚举来识别已经在使用的序列[30]。同样，当序列涉及到序列号时，我们可以从已知的ClientId开始搜索它的邻域，这可能会命中连接到物联网平台的设备使用的其他令牌

* 攻击一个共享设备

一旦一个设备被用户访问(例如，通过酒店住宿、公寓租金等)，它的ClientId将永远暴露给用户(通过查看其MAC、序列号和嗅探网络流量等)。如果用户是恶意的，他可以迫使设备不时离线(例如，来自不满的Airbnb租户的报复)，甚至恢复当前用户的MQTT会话来窃取她的敏感消息(例如，健康状况和统计数据、私人习惯等)。注意，这可以通过目标设备的云平台进行远程操作。

具体来说，从我们自己的设备中，我们发现iRobot实际上使用了它的16位序列号(例如，3147C60043211234)作为它的ClientId，这使得我们可以不断地增加或减少我们的设备的编号来生成潜在的ClientId。在AWS云上评估这些令牌可能会对合法用户造成干扰。所以我们采取一个Web服务发现从我们的分析交通产生的移动应用。服务是在https://disc-prod.iot.irobotapi.com/robot/discover上运行,这有助于应用程序找出其设备年代经纪人定位在世界各地不同的AWS服务器,ClientId。通过查询带有200K 个id的服务，我们能够在6个小时内确认超过10K的id是已经部署的设备的clientid。注意，我们的实验只使用了一个线程来进行查询，而使用更多的线程和更长的时间可以帮助我们在一天内找到成千上万个真正的clientid。

为了验证攻击确实可以在真实的云平台上发生，我们构建了一个简单的脚本来连接到AWS IoT (https://a2uowfjvhio0fa.iot)上的7 iRobot s代理。(us -east-1.amazonaws.com)，它使用我们的平台凭据(我们注册的合法iRobot帐户)向代理进行身份验证，但与目标设备(我们的另一个iRobot设备)的ClientId进行连接。攻击立即导致目标离线。我们进一步测试了攻击的可伸缩性，成功地运行了我们自己的客户端(通过单一平台身份)，与我们自己的AWS IoT代理进行了2000个并发连接。这表明，只要多一点资源(脚本和平台身份)，影响数万台或更多设备的DoS攻击是完全可行的。

我们发现的DoS弱点是普遍存在的，这一点已经在包括IBM、Tuya Smart等在内的其他领先的物联网云上得到了证实。除了iRobot，我们还对Tuya Smart和苏宁Smart Living的设备发起了*PoC*攻击。在后面的章节III-D中，我们进一步展示了利用另一个漏洞，攻击者可以很容易地在苏宁物联网云上获得100多家制造商[39]的设备*ClientIds*。

### 讨论

我们认为，从根本上讲，安全缺陷来自于管理*ClientIds*的不安全实践。MQTT规范[4]从一开始就没有将*ClientId*视为一个秘密，甚至建议它可能只有一个字节那么短。更严重的是，在将该协议应用于可能具有对抗性的物联网通信时，这个问题从未得到确认和解决。作为证据，尽管*MQTT*将*ClientId*指定为代表客户端的身份，但没有任何物联网云认真地将令牌视为与用户ID类似的身份，它使用各种身份验证方案进行保护。此外，在*MQTT*中，*ClientId*还充当用于恢复会话的*web*会话*cookie*。然而，正如我们在研究中发现的，物联网平台和制造商都没有做什么来保密。对于物联网云平台来说，最让人困惑的可能是平台身份的共存，平台身份的确涉及到各种认证和授权过程(如双因素认证、AWS cognito、SSO等)。然而，这种保护只能确保只有经过身份验证的平台用户才能建立MQTT连接，而不是只有授权用户才能声明连接的*ClientIds*。因此，平台上任何经过身份验证的用户都可以使用任何*ClientId*。

### 负责任披露

我们向所有受影响的供应商报告了我们的发现，如iRobot, Microsoft, IBM, Tuya，百度和苏宁，他们都承认了这个问题。特别是，微软通过他们的漏洞奖励计划奖励了我们。

## D．MQTT主题的授权之谜

由于缺乏指导和物联网独特的竞争环境，保护消息(III-A部分)、会话(III-B部分)和ClientId (III-C部分)的必要性对物联网供应商来说并不明确。然而，即使对于众所周知需要保护的MQTT实体，例如MQTT主题，我们也发现它们的保护是不够的:物联网云在保护中采取了不安全的捷径，这也是由于缺乏安全采用MQTT的指导，特别是在物联网的敌对环境中;受保护的MQTT主题的不合理描述也可能授予恶意用户过多的访问权。在实践中，这些问题被发现对安全性和隐私都有极大的损害。

### 保护MQTT主题的不安全快捷方式

如前所述(第二部分)，物联网云增加了MQTT中缺少的安全控制。特别是，物联网云控制用户可以发布或订阅哪些MQTT主题(与特定设备相关联)。现实世界中的物联网云需要管理来自数千家制造商的数百万用户和设备，授权可能非常复杂，尤其是考虑到常见的用户权限变化。为了简化MQTT的采用，我们发现物联网云在构建授权时走了捷径。例如，苏宁(财富500强企业[40])的物联网云服务于100多个品牌的[39]、[41]，允许用户订阅他/她知道的任何MQTT主题，基于MQTT主题是机密的隐式假设。

然而，这种假设在物联网的敌对环境中并不成立。正如II-C节所介绍的，物联网对手通常能够至少一次使用目标设备(例如，在酒店房间甚至私人住宅中)，因此很容易知道它的主题(例如，通过流量分析)。此外，我们发现设备制造商倾向于使用设备的一个已建立的唯一标识符作为其MQTT主题，例如设备序列号或MAC地址，这些都属于暴力枚举(Section III-C)。因此，在呈现目标设备的主题时，对手可以在没有用户同意的情况下订阅它的所有消息。根据设备的不同，这些信息可能包括高度敏感或私人信息，如健康统计数据和状况、购买偏好和历史、个人习惯、家庭关系等。

这种隐私风险是在物联网的对抗环境中引入的，甚至可能在MQTT的原始使用场景中都不存在;例如，工业专用设备通过卫星[42]传输遥测数据。同样，我们的发现表明，原始和更可信的MQTT环境与物联网环境之间的安全差距以前没有得到充分的评估，而且在很大程度上被低估了。此外，考虑到市场[43]中物联网平台的数量庞大，在物联网环境中安全采用MQTT缺乏指导和标准的情况下，供应商特有的、不健全的保护(如苏宁s)往往会在现实世界中发生。

### 攻击

我们使用HONYAR智能插件IHC8340AL实现了PoC攻击，通过它我们反向工程了苏宁Smartlive移动应用程序和云之间的通信，并找到了它的MQTT主题。为了呈现主题，我们创建了一个脚本，并成功订阅了设备的所有后续消息，即使设备被重置，然后与另一个用户帐户(即用户名)一起使用。，表示下一个使用该装置的人是受害人)。

### MQTT的表达语法

一个设备可能有8个相关的主题(例如，/deviceID/cmd用于发送命令，/deviceID/status用于更新状态)。为了便于使用，设备的允许用户可以使用通配符#或+订阅设备的多个主题，甚至多个设备。在我们的研究中，我们发现MQTT的这种表达性语法为物联网云留下了巨大的空间，在不知情的情况下将用户隐私置于高风险之下。例如，流行的物联网云苏宁智能生活(Suning Smart Living)未能正确授权使用通配符订阅MQTT主题。具体来说，平台的任何用户都可以订阅物联网云的通用#主题，根据其在MQTT中的定义，这意味着代理上的所有MQTT主题(实际上是物联网云下的所有设备)。该漏洞会通过物联网设备在云平台上泄露所有物联网用户的各种高度关键的隐私信息，如个人身份信息(PII)、健康状况、家庭关系、个人习惯等(见下文我们的PoC实验)。

令人惊讶的是，这样的问题一般都是通用的，因为灵活的MQTT语法使得在实践中正确地描述和解释要保护的资源变得非常困难。安全策略中不健全的资源描述和解释实际上允许对手访问未经授权的资源。在上述苏宁的例子中，苏宁很可能只允许用户访问其安全策略中描述的允许主题，如deviceId/cmd;但是，请求的通配符-topic #在技术上可以与策略中允许的主题匹配，从而导致策略绕过。此类问题也在AWS IoT上得到了确认。具体来说，设备制造商为其用户和设备配置安全策略，然后由AWS强制执行。我们发现，即使策略明确地拒绝用户访问描述的主题(如deviceId/cmd)， AWS也不能很好地解释受保护的目标，并启用未经授权的访问:恶意用户可以订阅deviceId/#并从受保护的主题接收消息。显然，AWS未能很好地解释包含通配符的资源描述，并将其与应受保护的主题关联起来。

物联网云不仅没有很好地处理灵活的资源描述语法，设备制造商也犯了类似的错误。特别是，为了方便用户访问其设备的多个主题，我们发现iRobot在AWS IoT上配置了一个(拖尾)通配符，如/[deviceModel]/[deviceId]/+。这样的主题描述政策过于包容甚至允许用户恶意iRobot公司创建一个任意主题相匹配的路径前缀,如/ [deviceModel] /[的deviceId] /攻击(注意的deviceId和deviceModel恶意用户的Id和模式s的设备在袭击中)。这实际上提供了一个隐藏的C&amp;C通道:恶意用户可以向这个隐藏的主题发布消息，所有机器人都可以订阅该主题并接收控制命令，从而使用无害的MQTT流量来逃避检测。这个问题是通过我们对iRobot移动应用程序和AWS物联网之间的流量进行逆向工程推断出来的，并通过iRobot进行了确认，尽管出于道德原因，我们没有实施端到端攻击实验。同样有趣的是，通过主题的所有通信都将由AWS支付给iRobot，因为承载该主题的代理是由iRobot部署的。更重要的是，这样一个实用的，隐藏的，省钱的C&amp;C频道可能有重大的现实世界的影响，考虑到最近毁灭性的僵尸网络，如Mirai[44]。

我们的研究结果首次表明，在实践中，对MQTT主题进行准确的描述和解释非常容易出错，而且在物联网的敌对环境中，这些错误可能会导致灾难性的安全性、隐私甚至直接的财务违约。同样，我们的研究表明，如果没有一个充分考虑物联网风险和微妙的MQTT资源的标准的指导，从受保护较少的MQTT协议安全地构建物联网通信比先前预期的更具挑战性。

### POC实验

我们在苏宁物联网云上进行了PoC实验，验证了上述问题。MQTT通信是在苏宁智能生活云终端的1885端口上进行TLS加密的。通过一个简单的脚本，我们在云上验证了我们的苏宁账号，并订阅了topic #(平台的通用主题)。通过订阅，我们收到了大量来自智能锁、摄像头、家庭安全监视器等的隐私关键信息。通过3周的信息收集(IRB批准)，我们发现潜在的对手能够推断出苏宁用户的家庭/同居关系，行为习惯，甚至个人可识别的信息(详见第四- b节)。同样令人惊讶的是，泄露的信息包括所有云下设备的ClientIds。利用III-C节中介绍的基于clientid的DoS攻击，潜在的对手可以轻松地将任意用户的任意设备踢出离线;考虑到用户和设备类型(例如，健康或安全监视器)可以从消息中识别出来，潜在的安全甚至安全风险是显著的。

### 负责任披露

我们向苏宁、AWS和iRobot报告了我们的发现，他们都承认这些问题及其严重的安全、隐私和财务影响。

# 四、测量

为了了解在我们的研究中发现的设计缺陷的严重程度，我们对8个领先的物联网云进行了测量研究。这项研究揭示了这些问题的普遍性及其影响，包括执行这些攻击的成本(如收集设备消息)。此外，我们发现物联网云提供的误导性指导可能导致物联网制造商方面的安全设计和实现出现问题。

## A.范围和幅度

我们测量集中在领先的物联网云调解许多知名厂商的设备,例如,AWS 9物联网核心[1],IBM Watson物联网[45],[46]阿里巴巴云计算物联网平台,微软Azure物联网中心[2],等等(见表1),所有依靠MQTT(3.1.1版本,当时我们的研究)的物联网通信。评估每一个物联网云平台,我们注册自己的账户,利用公共开发文档和sdk,并建造了演示设备和应用程序行使MQTT-based物联网通信并验证其安全保护每个问题我们发现身份管理(ClientId劫持,参见III-C),消息授权(将消息和保留信息,参见III-A),会话管理(III-B节)和主题授权(III-D节)。请注意，我们研究的两个云平台并不发布它们的sdk(仅对付费设备制造商可用)，即图雅、苏宁;因此，我们购买了一些物联网设备来支持我们的测试。我们的测量结果如表一所示，具体阐述如下:

**•Identity management.**

The majority of tested cloud platforms suffered from the ClientId hijacking attack, including AWS, IBM, Microsoft, Tuya, etc. (Table I), which demonstrates the problem is general. As discussed in Section III-C, very likely the problem is due to the misguidance of MQTT speciﬁcation and the lack of secure and standardized practices in managing MQTT ClientId in the IoT systems. In such unguided development environment, each platform implements its own version of IoT communication, and interestingly, some platforms’ implementation renders the problem harder to exploit. In particular, each device under IBM IoT cloud has to use the ClientId pre-assigned under its factory setting, in the format d:orgId:deviceType:deviceId), in which the deviceId ﬁeld must match the recorded ID of the device on the cloud. Hence, the attacker cannot exploit his/her device’s MQTT connection to claim a victim’s ClientId. In contrast, on the same cloud, the user client follows another format a:orgId:appId, in which the appId ﬁeld can be arbitrarily set by the user. Hence, the attacker can exploit his/her user connection to claim a victim user’s ClientId – our ClientId hijacking attack. Since the two types of MQTT clients (device and user) on IBM cloud have different security implications, we assessed them separately in our study (see the two sub-columns under the IBM column in Table I). Although IBM’s device connection is hard to exploit, this is likely attributed to its ClientId deployment choice for devices – not an intentional security design that restricts the ClientId per MQTT client. Otherwise, IBM may have applied the same restriction to protect its user client.

* 身份管理

大多数被测试的云平台都遭受了ClientId劫持攻击，包括AWS、IBM、Microsoft、Tuya等(表一)，这说明问题是普遍存在的。正如在第三- c节中讨论的，问题很可能是由于MQTT规范的误导以及在物联网系统中管理MQTT ClientId方面缺乏安全和标准化的实践。在这种不受引导的开发环境中，每个平台都实现了自己版本的物联网通信，有趣的是，一些平台的实现使得这个问题更难被利用。特别是，IBM IoT云下的每个设备都必须使用其工厂设置下预先分配的ClientId，格式为d:orgId:deviceType:deviceId)，其中deviceId字段必须与云上记录的设备ID匹配。因此，攻击者不能利用他/她的设备的MQTT连接来声明受害者的ClientId。相反，在同一云中，用户客户机遵循另一种格式a:orgId:appId，在这种格式中，用户可以任意设置appId字段。因此，攻击者不能利用他/她的设备的MQTT连接来声明受害者的ClientId--我们的ClientId劫持攻击。自两种类型的MQTT客户机(设备和用户)在IBM云有不同的安全问题,我们在研究中分别评估(见下两个sub-columns IBM列在表I)。虽然很难利用IBM s设备连接,这可能归因于其ClientId部署选择设备不是一个故意安全设计,限制了ClientId / MQTT客户机。否则，IBM可能会应用相同的限制来保护其用户客户机。

同样有趣的是，我们注意到阿里巴巴(全球第四大云提供商)没有遵循MQTT规范:在其物联网平台上的两个MQTT客户可以主张相同的ClientId，没有冲突，这使得我们的利用无效。值得一提的是，AWS物联网允许设备制造商指定高度灵活的定制安全策略[47]，并支持限制客户可以索赔的内容。然而，我们从未向设备制造商建议过这样的限制，而且在AWS开发人员文档中推荐的模范策略甚至受到了我们的攻击。附录B部分报告了我们对不安全的AWS物联网安全政策在野外创建的进一步研究:开发者在Github上创建的大部分是不安全的，这进一步强调了消除威胁和构建安全的物联网生态系统的艰难挑战。我们的研究表明，ClientId管理是一个普遍的问题，更重要的是，每个云都为MQTT通信添加了定制的但无效的安全保护，这样的生态系统是高度不安全的，令人担忧的，迫切需要标准化的安全指南和实践。

* 消息授权

所有的物联网云平台，如AWS、微软、Tuya等，除了少数例外，在我们研究时，其实现并不支持两种类型的面向效用的消息(Will Message和retain Message)

* 会话管理

在我们的研究中发现，大多数物联网云受到不安全的会话管理(订阅和生命周期状态，参见III-B部分)的影响。有趣的是，为了安全地管理订阅状态的变化，AWS添加了自己的许可策略，而MQTT规范和其他物联网平台没有考虑到这一点。尽管如此，AWS还是受到了会话生命周期攻击。再次，这一观察表明，迫切需要一致和标准化的安全指南，以保护许多物联网平台的MQTT通信。

* 主题授权

主要的平台AWS和苏宁公司都受到了这类攻击。我们所研究的其他特性则不是，因为它们的当前实现没有积极利用MQTT主题中的通配符，这是一个方便的特性，使用户能够轻松订阅她的设备的多个主题或她的设备的多个主题。考虑到当前系统设计[48]中面向可用性的趋势，我们认为这样的特性对这些平台很有吸引力。一旦被采用，安全隐患在实践中可能是不可避免的，AWS、苏宁、以及设备制造商(如iRobot，参见III-C部分)所犯的错误就是明证。

* 结论

我们向相关供应商报告了所有发现的安全问题，这些供应商都承认存在这些问题。我们的测量显示，物联网云每个都构建自己的MQTT通信和自组织安全保护，这是不有效和安全的。这突出了在缺乏系统分析、深思熟虑的设计原则和安全指南的情况下，如何将MQTT(一种在更可信的环境中设计的通用消息传递协议)安全地引入敌对而复杂的物联网环境在实践中面临的挑战。

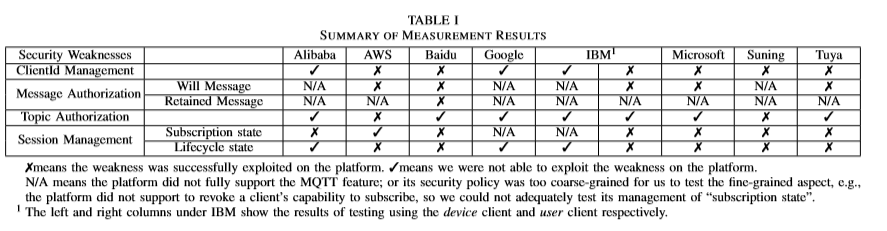
## B.泄露的MQTT消息的隐私含义

III-B和III-D部分的弱点使攻击者能够秘密收集由物联网受害者设备发布的MQTT消息。了解现实世界中的消息泄漏是否确实会导致严重的后果和隐私的影响，我们进行了一个实验仔细批准了我们大学的IRB办公室。

具体来说，由于苏宁云平台上MQTT主题订阅中的通配符处理不当(第III-D节)，任何实际的攻击者都可以很容易地(通过通配符/#)订阅来自云下所有设备的消息。在我们的实验中，我们运行了一个简单的脚本来订阅苏宁的代理端点(通过反向工程我们自己的设备的流量很容易获得的URI)，并收集了三个星期的MQTT消息(参见下面的IRB批准)。

总的来说，我们成功地从云中收集了8亿条实际的MQTT消息。从如此容易收集到的信息中，我们的发现是非常惊人的。特别是个人发布的信息物联网设备包含丰富的信息,包括设备id和类型(例如,门锁,空调,相机,等),设备状态(关闭,加热、洗涤),时间戳、设备位置(家里、办公室、客厅等)和捕获的信息设备(如室内温度,空气质量,是否一个人路过一个相机,等等)。收集到的一些消息包括用户PII(例如，电子邮件、电话号码、昵称和姓名等，由用户在其设备上配置)。同样有趣的是，从设备ID和用户ID，潜在的攻击者可以推断出三周内活跃用户和设备的数量，这可能是我们决定在论文中不披露的平台商业秘密。

表一测量结果汇总



×表示该漏洞在平台上被成功利用。意味着我们不能利用平台的弱点。√表示平台不完全支持MQTT特性;或者它的安全策略太粗粒度，我们无法测试细粒度方面，例如，平台不支持撤销客户机的订阅功能，因此我们无法充分测试它对订阅状态的管理。IBM下面的左右两列分别显示了使用设备客户机和用户客户机进行测试的结果。

更重要的是，当这些信息结合在一起进行纵向分析(在我们的研究中进行了三周)时，潜在的攻击者可能会推断出可识别的人(通过他们的电子邮件地址、电话号码)的私人习惯、日常行为、同居关系等。例如，图4显示了门锁的三周状态:用户通常在工作日呆在家里，因为门经常从9:40开到19:10，而在我们的研究期间，由于门在两个周末完全不开，用户通常在周五离开家。此外，附录中的图5显示了一台空调的三周状态，它也泄露了用户通常不在家的信息。因此，我们设想这样的日常信息(从门锁，空调，冰箱，灯泡，咖啡机，烤箱)的受害者可以帮助一个潜在的窃贼找出最好的时间闯入一个家庭。此外，我们还发现了来自家居锁的信息，如“[用户设置的人名]通过指纹开门”，这可能泄露了用户的私人同居关系(如与电影明星同居)。

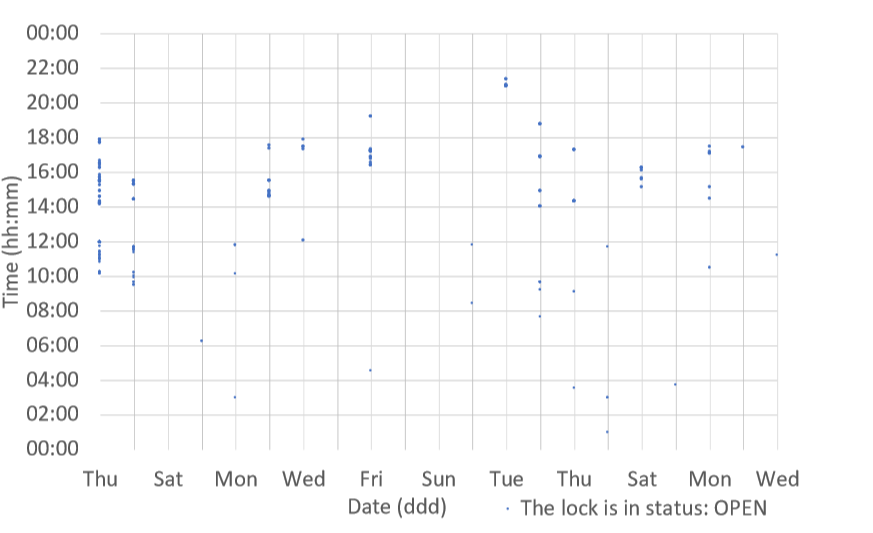


图4所示苏宁云下智能锁的“打开”状态

考虑到数以百万计的消费者和医疗物联网设备在市场上和他们敏感的功能,比如怀孕测试和监控/血压、抑郁、血糖,心跳,帕金森病,和用户位置,潜在的隐私泄漏显示在我们的研究中值得严重关注,并呼吁紧急保护。

**IRB批准**为了避免潜在的隐私侵犯现实世界的用户，我们的实验设计是由我们大学的IRB批准的。特别是，我们收集的所有数据都是在传输过程中加密的(通过支持tls的苏宁代理服务器1885端口);消息中的任何个人身份信息(PII)，例如电子邮件、电话号码，在rest时通过SHA-512进行散列。

# 五．缓解

从我们的安全性分析中，我们得出结论，当今的物联网云无法弥补MQTT的面向可用性设计与复杂的用户设备交互的安全性需求之间的差距。要降低这些缺陷带来的风险，需要在协议中加强对实体的保护，以确保一方只能在其授权期间访问物联网设备。然而，我们的研究结果表明，这种保护虽然是物联网环境中迫切需要的，但却超出了当前MQTT协议的安全模型，例如身份、消息和会话状态通常不受保护或不建议保护。因此，保护的关键是添加缺少的安全模型和控制关键协议实体的设计原则。因此，在本节中，我们将介绍保护协议实体的设计原则和增强的访问控制模型。我们还报告了蚊虫保护的实施情况及其效果评价。

## A.管理协议标识和会话

在复杂的、对抗性的物联网系统中采用消息传递协议的一个关键设计原则是，协议层身份(例如，ClientId)(如果有的话)应该经过验证;此外，如果标识用作安全令牌(例如，会话令牌)，则应该保证其机密性。

我们的研究表明，物联网云平台通常通过平台层身份(如Amazon帐户，参见III-C部分)对MQTT连接进行良好的身份验证。因此，对典型的物联网平台的一种轻量级且有效的保护是，将MQTT客户机的ClientId严格限制为逐字匹配客户机平台上的身份p\_user\_id (或客户机的平台层身份验证令牌的消息摘要，例如OAuth[49]令牌或证书)。例如，一个用户的ClientId应该从她的平台身份p用户id开始。这样的设计也考虑了物联网应用程序的并发性:一个用户可以有多个不同ClientId的客户端，每个客户端通过其前缀进行限制。然而，这种设计可能在某些情况下不能工作，例如，设备客户端可能有硬编码的ClientId。为了解决这种情况，一种更通用的方法是维护平台标识与其允许的标识之间的映射。任何试图主张未经授权的ClientId都应该被拒绝。我们将把这种映射的完整设计和评估留给我们未来的研究。

此外,会议后,在消息传递协议应该谨慎原则:在一个敌对的环境中,主体的存在(例如,一个用户)预计将有特权的变化,会话状态,包括协议无关的状态(例如,生命周期状态)和协议的特殊状态(例如,订阅状态),应该相应地保持更新。

## B.面向消息的访问控制模型

保护物联网系统上的消息传递协议的关键是保护其消息通信:该系统应管理主体发送/接收消息的权利，并根据接收者的安全需求管理接收消息时的安全影响。

在我们的研究中发现，前者通常被认为是强制执行的，而后者，尤其是在设备共享和特权撤销时保护消息接收者(II-C节)，则完全不在MQTT当前考虑的范围内。为了弥补这一不足，我们提出了面向消息的使用控制模型(MOUCON)，这是一种基于UCON[50]的物联网通信增强访问模型。MOUCON模型的核心思想很简单，它建立在我们熟悉的概念之上:它将消息作为资源(对象)，并根据主题和对象的属性检查主题的访问权限。它的定义如下:

通信中的客户端集合，如设备和用户。主体由其属性定义和表示。

主题的属性被指定为

并且包含识别信息(*id*),的集合[51](例如*topic*), 允许发送消息到(),并且URI允许接受消息从

主体拥有权限的一组消息。

对象的属性被指定为

它包括*content*应用程序层信息(例如，消息内容)、*URI*(表示消息通道例如，消息发布到哪个*MQTT*主题或来自哪个*MQTT*主题)、*source*表示对象源的*URI*(例如，消息内容)即，创建消息的主题。

权限是主体可以持有并在对象上行使的特权。一般有两类权限，读*(R)*(例如，接收消息)和写*(W)*(例如，发布消息)。

*Authorizations.*授权功能评估和请求的权限，以及一组用于访问决策的授权规则。例如，通过检查消息发送方(源)和接收方的属性来针对消息接收的授权规则如下

*(1)*

在式(1)中，表示允许*Subject* *s*的权利*Right* *R* 到*Object* *o*(即，接收人s被允许接收一条消息o).这个决定是通过检查右边的条件做出的。在物联网场景中，这条规则确保，从消息接收者的角度来看，如果消息的source(o.source，例如，发送消息的用户客户端)已经失去了访问消息来源的主题的权限(通过检查不在中)，此消息(例如，*Will*消息)应该被拒绝。此外，此规则还通过检查消息的主题()是否在允许的主题集中来检查是否允许接收消息。这样的规则弥补了考虑消息接收者的安全影响方面的不足，并且可以击败我们的消息相关攻击(Section III-A和Section III-B)。

**讨论**MOUCON还支持其他安全策略。例如，我们可以根据消息发送者的属性定义授权消息发送者的规则，该属性指定允许将消息发送到的主题。此外，一个更完整的解决方案应该考虑如何在物联网系统运行期间正确地更新发送方和接收方的特权:例如，如何动态地撤销客户机的读/写特权，如何通过云授权此类撤销请求，等等。未来，我们还计划12个扩展模型，以保护其他消息传递协议，如CoAP，当它们在物联网环境下运行时。例如通过扩展属性集，定义相应的授权规则，增强对这些协议的保护。

## C. 执行与评估

我们对*Mosquitto*实施了上述保护(1.5.4版，原来没有我们的保护)。具体来说，我们修改了与其消息相关的数据结构*(struct mosquitto\_msg\_store*)，以添加与安全性相关的属性(例如，消息的源)，并向其代理添加了授权功能(例如，在发送消息前检查消息源)。此外，我们还在用于建立会话的代理现有访问控制函数[52]中添加了建议的*ClientId*限制。

我们在实验室环境中部署了安全版的*Mosquitto*，并评估了它的有效性和开销性能。具体来说，我们的安全*Mosquitto*和原始的无保护的*Mosquitto*分别部署在一个工作站上，使用Intel i7-6700 CPU、15.6G内存和475.3GB磁盘。我们还在充当MQTT客户端的笔记本电脑(2.2 GHz Intel Core i7、16 GB内存、251GB SSD和2.3 GHz Intel Core i5、16 GB内存、251GB SSD)上运行脚本。为了评估我们的保护效果，我们在第三节中发起了所有攻击:我们的安全*Mosquitto*击败了所有攻击，而未受保护的服务器没有捕获任何攻击。为了评估性能开销，针对服务器，我们使用并发客户端来发布消息，并使用相同数量的订阅者来接收消息。对于每个服务器，我们记录了消息发布和接收之间的平均延迟(以ms为单位)，以及在此期间的平均CPU和内存使用情况。我们对不同数量的并发发布客户机(从1000到8000)重复了这个实验。结果(详见表II)表明，我们增加的保护只会带来微不足道的消息传递延迟(最多0.63%)和内存开销(最多0.16%)。尽管CPU使用率更高(大约10%)，但这是正常的预期:添加关键的安全检查通常是有代价的，特别是与不受保护的实现相比。

# 六、讨论与未来工作

**经验教训**从我们的研究中得到的最重要的教训是，在将一个实用的、通用的协议应用到可能涉及恶意方的领域时应该谨慎。在这种情况下，既需要评估协议不涉及的场景，也需要评估协议的各个状态，以确定协议可以保护的内容与需要保护的内容之间的差距。更具体地说一般消息传递协议的使用物联网设备用户沟通, 我们的研究强调，不仅应该通过适当的身份验证和授权来保护*ClientId*及其相关状态，而且还应该在适当的保护下将整个撤销过程(其安全性至关重要)添加到协议中。注意，虽然我们的研究集中在MQTT上，但是由于它的广泛部署，其他消息传递协议，如Firehose[53]、CoAP[54]、AMQP[55]、JoyLink[56]和Alink[57]也可能有类似的问题。事实上，我们在三星的Artik物联网云[53]上的Firehose WebSocket中发现了一个类似的撤销问题，三星承认:云未能终止客户端接收消息的会话，即使在其特权过期之后也是如此。

减轻此类身份验证/授权缺陷需要协议设计者和物联网制造商的共同努力。如前所述，尽管*MQTT TC*开始针对我们发现的威胁增强协议，但是制造商也需要尽自己的职责。为此，能够通过不同的消息传递协议将设备和用户安全地连接到云的统一接口是有价值的。我们的方法为此迈出了第一步，并在网上发布了[14]。

**缺陷的自动发现** 基于我们对应用通用消息协议来支持物联网管理的安全隐患的理解，我们认为更系统和自动的安全分析和漏洞发现是可行的。以MQTT为例，一种可能的方法是半自动地构造其操作的状态转换，特别是由协议的IoT定制引入的状态转换，并利用模型检查和/或指导手动分析来检测安全缺陷。更具体地说，MQTT的有限状态机(FSM)可以通过比较MQTT规范和IoT云文档来使用自然语言处理(NLP)恢复，就像我们在之前的工作[58](关于支付服务分析)中所做的那样。在FSM上，我们可以手动识别所需的安全属性，例如访问特权撤销的会话状态更新，并使用线性时态逻辑(LTL)[59]指定它们。FSM上属性的存在可以使用模型检查器来评估，例如，SAL[60]。通过在系统上运行示例，检查器报告的问题可以作为发现*MQTT-IoT*系统安全缺陷的反例。然而，在现实中，由于协议文档中的限制和/或不完整或不准确的规范，*NLP*或许不能完全捕获物联网协议的所有细节。此外，复杂的*FSM*可能使正式的验证难以成功。在这两种情况下，模糊的、不完整的描述或复杂的协议逻辑都可以指导我们的分析将重点放在寻找安全弱点的相关实现上。请注意，这一步也可以在某种程度上自动化，例如，通过对协议实体(例如，*clientID*、*topic*、*Will* *Message*等)的向导模糊测试。未来，我们将在这个方向上进行一系列研究，自动发现物联网系统中的复杂安全漏洞，特别是那些建立在现有消息协议之上的漏洞。

***MQTT5.*** *MQTT*规范的新版本*MQTT5*在2019年3月成为*OASIS*标准。据我们所知，在我们研究期间，没有公共物联网云支持*MQTT5*。此外，新标准保留了以前版本中的所有功能，因此我们所有的攻击都应该是有效的在物联网云上使用没有额外保护的*MQTT5*。而且，新规范没有提供协议层解决方案来解决我们发现的安全风险。即使它的非规范化部分提到客户端需要授权使用*ClientId*和应该提供一个实现访问控制限制客户端发布或订阅特定主题的能力,它没有提供任何建议应该如何在这种保护迫使我们研究物联网的场景。我们向*OASIS* *MQTT* *TC*报告了我们的发现，该*TC*承认我们的发现同时影响版本5和3.1.1，现在正在公开讨论[11]。

# 七、相关工作

**关于MQTT的安全性研究**行业报告显示，通用的MQTT协议缺乏基本的身份验证和授权。在没有任何保护的情况下(如今天的物联网云已经添加的保护)，[61]表明攻击者能够连接到任意的MQTT服务器而无需身份验证和订阅任何主题。另一份行业报告[62]扫描了Internet上暴露的MQTT服务器端点和MQTT库和代理实现中的实现错误(处理字符编码)。与这些研究完全不受保护的MQTT实现的工作相比，我们关注的是当今主流的物联网云在采用MQTT时的保护以及它们在消除物联网环境特有的新风险方面的不足。

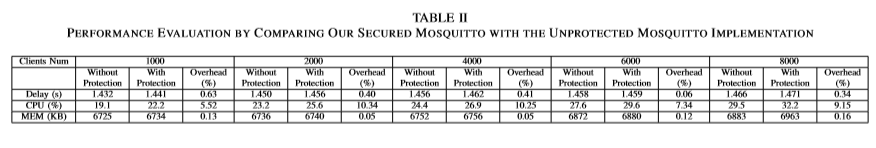
由于在协议设计中缺乏基本的安全考虑，学术界提出了加强MQTT安全模型的建议。[63]提出了发布临时keys to mqtt clients的新机制;[64]提出了通过考虑客户端上下文信息(如IP地址)来增强安全保护;[65]提出了通过监控可变属性进行访问决策来装备MQTT的动态控制模型;[66]提出在MQTT环境中引入OAuth作为安全性的补充。与之前的工作相比，我们的研究集中在现实的物联网平台，以及它们在MQTT部署中增加的安全保护的弱点，并分析了当前物联网供应商面临的实际安全挑战和陷阱**。**

**IOT平台安全**物联网平台的安全性已经得到了广泛的研究，如[30][67][72][70]首次对三星SmartThings平台进行安全分析，发现其粗粒度的访问控制设计。[30]研究了物联网设备、移动应用和云之间的交互，重点研究了这三个实体的状态转换图中的弱点。也有人提出了保护物联网系统的工作。[73][75]研究了如何限制云、移动应用和设备的能力;[76]跟踪和保护敏感的物联网信息;[77]提出了一个基于源的框架来聚合设备活动，以检测错误和恶意活动。与此形成鲜明对比的是，我们对物联网的研究侧重于底层消息传递协议的独特方面，即以及在复杂的设备-用户对抗交互中采用MQTT时，物联网云所增加的安全性保护不足。

# 八、结论

我们首次对使用物联网设备用户通信通用消息传递协议的安全风险进行了系统研究。我们的研究揭示了为在简单和良好的环境下运行而设计的协议与复杂的、敌对的物联网场景之间的差距，以及用适当的安全手段来弥补这一差距所面临的挑战。从这些发现中，我们归纳出新的设计原则，并提出了一个增强的访问模型。我们的保护措施得到了实施，并对其有效性和效率进行了评估。我们的新发现和保护将导致在现实世界中更好地保护用户-设备交互。

表二绩效评估通过比较我们的安全*MOSQUITTO*和无保护*MOSQUITTO*的实施情况



# 附录

## A.从泄漏的MQTT消息推断用户的例程

图5为空调三周状态，其中红色的点表示空调加热的时间。从图表中，一个潜在的攻击者可以推断出，空调通常在工作日的早上6:45，周末的早上7:45，晚上10点到午夜之间停止，这大概表明了用户早上什么时候起床，晚上什么时候睡觉。同样有趣的是，除了第二周的星期四和第一个星期六到第二个星期五的下午12点到4点之间，空调不工作期间，用户很可能大部分时间在家。

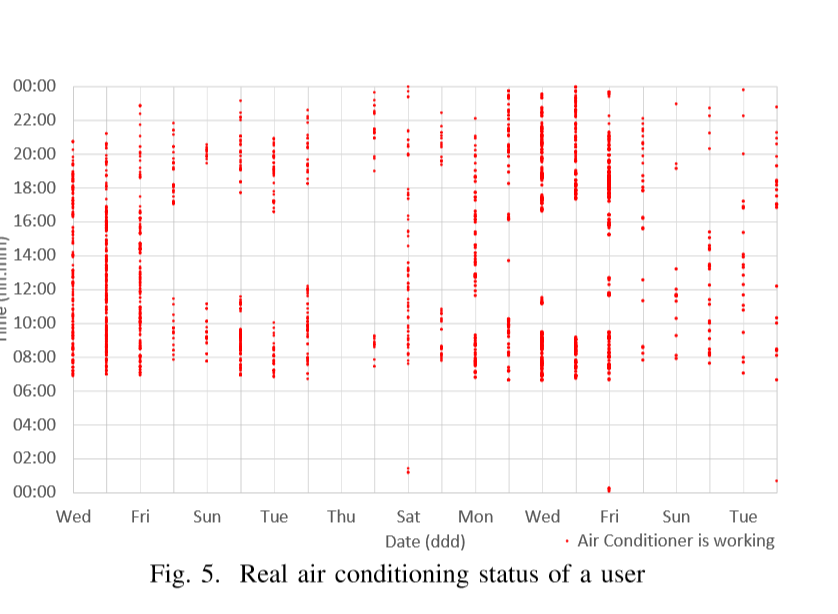


图5. 真实的空调用户状态

## B．误导开发指南

在AWS物联网上，设备制造商可以定制AWS物联网策略(一个安全策略模板，在运行时填充并分配给每个客户端)来限制客户端的ClientId，从而减轻我们基于ClientId的攻击(章节III-C)。然而，设备制造商并不清楚这样做的安全方式，因为我们发现AWS开发人员文档和其他流行的物联网开发指南中推荐的典型安全策略甚至会受到我们的攻击。此外，我们发现大多数AWS物联网安全政策都是由开发者在野外创建的。(例如，在GitHub上)很容易受到我们的攻击，如下所述。

通过手动检查AWS官方开发者指南和博客，我们收集了AWS提供的38条堪称典范/最佳实践的AWS物联网政策。不幸的是，AWS从未建议限制ClientId，在我们发现的38个推荐策略中有26个是脆弱的。例如，清单1演示了来自AWS官方博客[78]的物联网策略，其中资源字段指定了客户端连接时允许使用的ClientId。在类似模板的策略中，变量${iot:ClientId}将在运行时被填充为客户端实际声明的ClientId。这个模板特性可以帮助设备制造商避免在其物联网政策中硬编码特定的ClientId，这可以在运行时应用于广泛的客户端。然而，这样的策略并没有限制客户端可以使用什么，因此容易受到攻击。我们报告了那个有误导性的、不安全的行政命令，他们承认了这个问题。

为了了解AWS提供的有问题的政策样本是否确实误导了野生物联网开发者，我们在Github上找到了89个开源项目[79]，其中包括AWS的物联网政策，并评估了它们的安全质量。不幸的是，85.4%(76/89)的项目包括一项物联网政策，很容易受到我们的攻击

此外，虽然AWS物联网政策支持指定客户的允许范围，但在实践中，设备制造商和开发人员仍然不清楚如何正确定义安全政策。在我们的研究中，我们彻底检查了AWS开发人员指南，并构建了允许制造商减轻我们的攻击的物联网策略，如清单2所示。具体来说，当该策略在运行时被填充时，它的变量*${Cognito -identity.amazonaws.com:sub}*将被实例化为客户端的Amazon Cognito ID[80]，即客户端的平台层身份。实际上，这限制了客户机的*ClientId*等于其经过身份验证的平台层标识，从而挫败了我们的攻击。然而，这个策略有一个可用性问题，这很可能阻碍它的广泛采用。具体来说，一个物联网用户往往有多个客户端(例如，在她的手机、平板电脑和笔记本电脑等)，所有这些客户端都必须呈现相同的ClientId- the Amazon Cognito ID 作为用户账户，因此，会在离线状态下互相攻击(参见III-C部分的*ClientId*冲突)。为了有效地降低安全风险，我们设想了一个符合V-A部分的解决方案:客户端可以使用的*ClientId*必须从客户端平台层身份开始(使用后缀来区分不同的合法客户端)

Listing 2. A Secure AWS IoT Policy Example

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Action": [ "iot:Connect" ],

"Resource": [ "arn:aws:iot:\*:\*:client/${cognitoidentity.amazonaws.com:sub}"

]

}]

}

Listing 1. An Example of Vulnerable AWS IoT Policy

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Action":["iot:Connect"],

"Resource":["arn:aws:iot:us-east -1:000000000000:client/${iot:ClientId}"]

}]

}