Security Analysis of Emerging Smart Home Applications

新兴智能家居应用的安全分析

2016 IEEE Symposium on Security and Privacy

目录

[Security Analysis of Emerging Smart Home Applications 1](#_Toc37166044)

[新兴智能家居应用的安全分析 1](#_Toc37166045)

[2016 IEEE Symposium on Security and Privacy 1](#_Toc37166046)

[摘要 2](#_Toc37166047)

[一、介绍 2](#_Toc37166048)

[我们的贡献 5](#_Toc37166049)

[第二章、相关工作 6](#_Toc37166050)

[智能家居安全. 6](#_Toc37166051)

[超过特权和最低特权 7](#_Toc37166052)

# 摘要

最近，几个相互竞争的支持第三方应用程序开发的智能家居编程框架出现了。这些框架为用户提供了切实的好处，但也可能使用户面临重大的安全风险。本文首次对此类新兴的智能家居编程平台进行了深入的实证安全分析（目标：智能家居编程平台）。我们分析了三星拥有的*SmartThings*，它拥有目前最多的应用软件，支持多种设备，包括运动传感器、火灾警报和门锁。*SmartThings*将应用程序运行时托管在一个专有的、封闭源代码的云后端上，这让审查变得很有挑战性。（问题面临的挑战）我们通过对499个*SmartThings*应用程序(称为*SmartApps*)和132个设备处理程序的静态源代码分析（静态分析技术：是指不执行被测软件，可对需求分析说明书、软件设计说明书、源程序做结构检查、流程分析、符号执行来找出软件错误），克服了这一挑战，并精心设计了测试用例，揭示了该平台的许多未文档化的特性。我们的主要发现有两个方面。首先，虽然*SmartThings*实现了一个特权分离模型，但我们发现了两个固有的设计缺陷，它们导致了*SmartApps*中明显的过度特权。我们的分析显示，由于功能过于粗粒度，商店中超过55%的*SmartApps*被过度授权。此外，一旦安装了智能应用程序，它就可以完全访问设备，即使是它特别需要只需要访问有限的设备。其次，*SmartThings*事件子系统(设备使用它通过事件与*SmartApps*进行异步通信)没有足够的保护携带敏感信息(如锁码)的事件。利用框架设计缺陷，构建了四种概念验证攻击:(1)秘密植入门锁代码;(2)盗窃现有门锁代码;(3)居家残疾人度假模式;(4)伪造火警信号。最后，我们总结了新兴智能家居编程框架设计的安全经验。

# 一、介绍

智能家居技术已经超越了基本的便利功能，比如自动控制的电灯和开门器，为人们提供了一些切实的好处。例如，水流传感器和智能电表用于节能。支持*ip*的摄像头、运动传感器和连接的门锁为家庭安全提供了更好的控制。然而，攻击者可以操纵智能设备，造成身体、财务和心理上的伤害。例如，窃贼可以锁定一个连接的门锁，植入隐藏的门禁码;纵火者可以锁定一个智能烤箱，在受害者家中引起火灾.

早期的智能家居系统具有陡峭的学习曲线，复杂的设备设置程序，并且仅限于自己动手的爱好者1最近，几家公司推出了更新的系统，这些系统更易于用户设置，受云支持并提供 一个用于第三方开发人员构建可实现智能家居收益的应用程序的编程框架。 三星的SmartThings [27]，苹果的HomeKit [7]，Vera Control的Vera3 [1]，谷歌的Weave / Brillo [18]和AllSeen Alliance的2AllJoyn [3]是几个例子。

1.存在许多论坛供人们交流专门知识，例如http：// forum。 Universal-devices.com/。

我们提出的问题如下：易受攻击的新兴，可编程，智能家居以什么方式出现？这些攻击带来了什么？ 解决此问题至关重要，因为答案将在可编程智能家居普及之前启动并指导国防研究。 在单个高性能智能家居设备[17]，[19]和在这些设备之间运行的协议（例如Z-Wave和ZigBee [9]，[21]）中发现了漏洞。 但是，很少或没有先前的研究来调查智能家居应用程序或应用程序本身的编程框架的安全性。

据我们所知，我们对智能家居编程框架进行了第一次安全分析。 具体来说，我们根据经验评估一种流行的智能家居可编程框架-Samsung SmartThings的安全性设计。 我们专注于编程框架，因为它是统一应用程序，协议和设备以实现智能家居效益的基础。 攻击者可以远程和秘密地瞄准框架中的设计漏洞，以实现前面概述的紧急威胁。

我们选择SmartThings有几个原因。首先，在撰写本文时，SmartThings有越来越多的应用程序，其中有521个叫做SmartApps，排名第二的是Vera，它在iOS store[1]上有204个基于lua的应用程序。其他与之竞争的框架，如HomeKit、Weave/Brillo和AllJoyn都处于形成阶段，各自的应用程序还不到50个。（应用程序广泛）其次，SmartThings支持132种主要制造商的设备类型。（支持丰富设备）第三，SmartThings与其他框架共享关键的安全设计原则（与其他框架共享关键的安全设计原则）。对设备访问的授权和认证是确保智能家居应用程序平台安全的关键，SmartThings有一个内置的机制，通过所谓的功能来保护设备操作不受第三方应用程序的影响。事件驱动处理在智能家居应用程序[30]中很常见，SmartThings为应用程序提供了注册回调的方法为设备生成的给定事件流。其他平台也支持事件驱动处理。例如，AllJoyn支持总线信号[2]，而HomeKit提供了特征通知API[6]。因此，我们相信通过对SmartThings框架的分析所获得的经验教训，将为许多可编程智能家居框架在早期设计阶段的安全关键组件的设计提供参考。

*SmartThings*框架认识到了潜在的安全漏洞，并整合了一些安全措施。*SmartThings*有一种名为“*功能*”的特权分离机制，用于指定SmartApp可以向兼容的智能家居设备发出的一组操作。SmartApps提供了安全存储，仅该应用程序本身可以访问。开发人员在Groovy的面向安全的子集中编写SmartApp。。基于Groovy的应用程序在沙箱中运行，该沙箱拒绝诸如反射，外部JAR和系统API之类的操作。OAuth协议(OAUTH协议为用户资源的授权提供了一个安全的、开放而又简易的标准。与以往的授权方式不同之处是OAUTH的授权不会使第三方触及到用户的帐号信息（如用户名与密码），即第三方无需使用用户的用户名与密码就可以申请获得该用户资源的授权，因此OAUTH是安全的. (1). 简单：不管是OAUTH服务提供者还是应用开发者，都很易于理解与使用；(2). 安全：没有涉及到用户密钥等信息，更安全更灵活；(3). 开放：任何服务提供商都可以实现OAUTH，任何软件开发商都可以使用OAUTH；)保护与*Smart Apps*的第三方集成。SmartThings为SmartApp和设备处理程序提供了*功能* 受保护的事件子系统，以进行异步通信。

我们的安全分析探讨了*SmartThings*编程框架的上述面向安全的方面。执行安全分析很有挑战性，因为*SmartThings*平台是一个闭源系统。此外，*SmartApps*只在私有的、*SmartThings*托管的云环境中执行，这使得基于工具的动态分析变得困难。由于没有公开可用的API来获取SmartApp二进制文件，所以二进制分析技术也不适用。（面临的挑战）

为了克服这些挑战，我们将构建的静态分析工具、运行时测试和手工分析结合起来，对499个Smart Apps和132个设备处理程序的数据集进行了分析，这些处理程序都是以源代码的形式下载的。我们的分析工具可以在https://iotsecurity.eecs.umich.edu找到。

## 我们的贡献

我们在两个方面发现了安全关键的设计缺陷:SmartThings功能模型和事件子系统。

我们发现SmartApps明显被过度授权了:(a) 55%的SmartApps没有使用它们的功能所隐式要求的的所有设备操作权限;(b) 42%的SmartApps被授予了没有明确要求或使用的功能。在许多情况下，由于能力模型的设备级授权设计，不可避免地出现了特权过高的情况，而特权并非由开发人员的过错造成的（§IV-A，§V-B）。 令人担忧的是，我们已经观察到68个现有的SmartApp已在不要求相关功能的情况下利用超权限来提供额外的功能。

我们研究了SmartThings事件子系统，发现：

（a）如果授予该应用程序至少一项设备支持的功能，则该应用程序不需要任何特殊特权即可读取该设备生成的所有事件；

（b）无特权的应用程序可以仅使用一个泄漏的设备标识符读取任何设备的所有事件；

（c）可以伪造事件（第IV-B节）。

我们利用设计缺陷和框架引发的开发人员错误的组合来展示各种安全问题如何共同削弱家庭安全。 我们构造了四种概念验证攻击：

* 我们远程利用了App Store上现有的SmartApp，我们远程利用了app store上现有的SmartApp，将后门pin码编程到一个相连的门锁上(VI-A节)。我们的攻击利用了SmartApp从未要求过的lockCodes功能-由于SmartThings功能模型的设计，SmartApp自动享有特权。
* 当用户正在对一个Schlage智能锁的pin码进行编程时，我们窃听事件子系统来窥探该锁的pin码，并使用不受限制的smart things提供的SMS API来泄漏它们。我们的攻击智能应用程序宣传自己是一个电池监视器，只要求电池监测功能。
* 我们使用一个欺骗事件来停止休假模式模拟(VI-C)，从而禁用了app store上现有的休假模式Smart App。这种攻击不需要任何能力。
* 我们使用一个模拟的物理设备事件(VI-D节)发出了一个假的火灾警报。一个没有特权的智能应用程序可以升级它的特权来控制它没有授权访问的设备，从而滥用那些无害的智能应用程序。

以上所有的攻击都会使一个家庭遭受重大的伤害:入室盗窃、盗窃、提供错误信息和破坏公物。这些攻击向量不是特定于特定的设备，而是广泛适用的。

最后，在我们的前瞻性分析中，我们总结了构建安全可编程智能家居框架的关键经验。我们将这些教训与在构建这样的框架中权衡利弊的探索结合起来。我们的分析表明，尽管一些问题很容易解决，但其他问题需要多种技术的良好平衡，包括设计基于风险的能力和身份机制(VII)。

# 第二章、相关工作

## 智能家居安全.

Denning等人概述了由于智能设备[12]的快速而稳定的引入而对智能家庭产生的一系列紧急威胁。例如，有窃听的威胁和各种智能家居设备的直接危害。Denning等人还讨论了攻击的结构，包括数据销毁、非法物理入侵和侵犯隐私等。我们的工作使其中一些风险具体化，并演示了远程攻击者如何在实践中削弱家庭安全。虽然我们不是第一个认识到现代家居的安全风险，但我们首次研究了新兴智能家居应用程序及其相关编程接口的安全特性。

当前的智能家居安全分析围绕两个主题：**设备和协议**。在设备正面，MyQ车库系统可以变成一个监视工具，在房屋可能空着时通知防盗。 Wink Relay触摸控制器的麦克风可以打开以窃听对话；霍尼韦尔（中国）Tuxedo触摸控制器具有通过身份验证和跨站点请求伪造的认证功能[17]，[19]。 Oluwafemi等。导致紧凑型荧光灯快速重启，可能会导致癫痫患者癫痫发作[23]。 Ur等。研究了飞利浦Hue照明系统和Kwikset门锁等的访问控制，发现每个系统都提供了孤立的访问控制系统，该系统无法实现基本的用例，例如与其他用户（如儿童和临时工）共享智能设备[ 29]。相比之下，我们研究了智能家居编程平台的新兴应用程序和相关的攻击媒介，它们在很大程度上与家庭中使用的特定设备无关。

在协议方面，研究人员展示了ZigBee和Z-Wave协议在智能家居设备[9]、[21]上的缺陷。利用这些漏洞需要接近目标主机。我们展示了程序框架中的设计缺陷，这些缺陷可以被用于不需要物理访问的内部攻击。此外，我们的远程攻击与所使用的特定协议无关。

Veracode对包括SmartThings [32]在内的几个智能家居集线器进行了安全性分析。 安全分析的重点是基础结构保护，例如是否使用SSL / TLS，是否存在重放攻击保护以及是否强密码被使用。Veracode的研究发现，SmartThings hub已正确部署了所有研究过的基础设施安全机制，但集线器上开放的telnet调试接口除外，该接口已被修复。相反，我们对SmartThings平台及其应用程序进行了实证分析，以发现框架设计缺陷。

## 超过特权和最低特权

最小特权原则是众所周知的，应设计编程框架以使其更容易实现。 然而，在实践中，很难做到这一点，Felt等人最近在智能手机上的研究证明了这一点。 对Android应用程序进行了市场规模的超权限分析，并确定940个应用程序中的三分之一处于超权限状态[13]，并指出开发人员的困惑是导致超权限Android应用程序的主要因素。 我们的工作类似，只是分析了一个相对封闭的系统，在该系统中，应用程序仅通过专有协议与带有受SSL保护的会话的集线器一起运行在专有云后端上，并在家庭中控制设备。 我们发现，大多数特权不是由于开发人员的困惑，而是由于框架设计本身

Au等设计了PScout，这是一个用于Android源代码的静态分析框架，可以针对不同的Android版本生成完整的权限说明[8]。我们对SmartApp源代码进行了静态分析，以计算特权。 但是，与PScout不同，我们不能使用静态分析来完成功能文档，因为SmartThings运行时是开源的。 相反，我们依赖于分析SmartThings后端和客户端Web IDE之间的协议。

## 许可/功能模型设计

Roesner等。 引入了用户驱动的访问控制，使用户保持在循环中，此刻应用程序正在使用敏感资源[24]，[25]。 例如，遥控门锁应用程序应仅能够响应用户操作来控制门锁。 但是，某些设备类型和应用程序更适合安装时权限。 毛毡等。 引入了一套有关何时使用不同类型的权限的准则[14]。 我们的工作评估了SmartThings功能模型在保护敏感设备操作免受恶意或良性但笨拙的SmartApp攻击方面的有效性。 我们需要确定对未来工作的能力授予方式。