Looking from the Mirror: Evaluating IoT Device Security through Mobile Companion Apps

镜像：通过移动伴侣应用程序评估物联网设备安全性

28th USENIX Security Symposium 2019

目录

[Looking from the Mirror: Evaluating IoT Device Security through Mobile Companion Apps 1](#_Toc38097063)

[镜像：通过移动伴侣应用程序评估物联网设备安全性 1](#_Toc38097064)

[28th USENIX Security Symposium 2019 1](#_Toc38097065)

[摘要 2](#_Toc38097066)

[一、介绍 2](#_Toc38097067)

[方法 3](#_Toc38097068)

[结果概述 4](#_Toc38097069)

[贡献 4](#_Toc38097070)

[二、系统设计 5](#_Toc38097071)

[2.1 总览 5](#_Toc38097072)

[2.2 APP 收集器 6](#_Toc38097073)

[2.3 APP 分析引擎 6](#_Toc38097074)

[2.3.1 设备接口分析 7](#_Toc38097075)

[2.3.2 imprints 分析 11](#_Toc38097076)

[2.3.3 模糊散列分析 12](#_Toc38097077)

[2.3.4 模块化 12](#_Toc38097078)

[2.4 Cross-App分析引擎 13](#_Toc38097079)

[2.4 设备固件收集器 14](#_Toc38097080)

[3. 数据和结果 15](#_Toc38097081)

[3.1 数据集和平台统计 15](#_Toc38097082)

[3.2 结果验证 17](#_Toc38097083)

[3.3 结果概览 19](#_Toc38097084)

[3.3.1脆弱的软件 19](#_Toc38097085)

[3.3.2设备重新品牌化 21](#_Toc38097086)

[3.3.3硬件漏洞 22](#_Toc38097087)

[3.3.4脆弱协议 23](#_Toc38097088)

[3.3.5后端服务脆弱性 24](#_Toc38097089)

[3.4结果的准确性 25](#_Toc38097090)

# 摘要

由于安全设计薄弱，智能家居物联网设备已日益成为网络犯罪分子的首选目标。 为了识别这些易受攻击的设备，现有方法依赖于对真实设备或其固件映像的分析。 不幸的是，由于固件映像的不可用以及获取用于安全性分析的真实设备的高昂成本，这些方法很难在高度分散的物联网市场中扩展。

在本文中，我们提出了一个平台，该平台可加速易受攻击的设备发现和分析，而无需存在实际的设备固件。我们的方法基于两个主要观察结果：首先，IoT设备倾向于重用和自定义其他组件（例如，软件，硬件，协议和服务） ），因此在一台设备中发现的漏洞通常在其他设备中存在。其次，可以从设备的移动配套应用程序间接推断出重用组件。因此，通过对移动伴侣应用的交叉分析，我们可以估算设备之间的相似度。我们的平台使用一套程序分析技术，分析市场上智能家居IoT设备的移动配套应用程序，并自动发现潜在的易受攻击的应用程序，从而使我们能够进行涉及4,700多种设备的大规模分析。我们的研究揭示了智能家居IoT设备之间易受攻击组件的共享（例如，共享易受攻击的协议，后端服务，设备更名），并导致发现了来自73个不同供应商的324个设备，这些设备可能会受到一组安全性的攻击问题。

# 一、介绍

智能家居物联网设备已经成为攻击者青睐的目标，因为它们的安全性设计很差。随着攻击者动机的增长(如IOT僵尸网络，个人数据盗窃)，智能家居设备的安全事件预计只会增加。保护这些设备在几个方面都面临挑战。首先，在这个领域中，有相当一部分供应商是中小型企业，它们缺乏软件质量控制和安全最佳实践的预算，导致市场上存在大量不安全的设备。第二,这些设备是相对便宜(通常不到100美元),不能支持昂贵的安全基础设施,如监控代理,加密和认证的硬件方面,当设备发现脆弱,人们很少的期待供应商发布修复。第三，高级供应商碎片化使得管理和分发软件/固件补丁非常困难。

解决这个问题的一种方法是在部署之前识别出易受攻击的设备，并采取适当的措施来保护设备。这类措施的例子可能包括升级设备固件、识别和阻止可能利用该漏洞的流量，或完全隔离该设备。为了预先识别易受攻击的设备，提出了多种方法[10、14、16、17、19、20、24、25、27、34、42、47、49、58]。有一项研究[19,27]集中在启动互联网规模的扫描，以检测公共可访问的普通易受攻击的设备(例如，具有弱密码、证书和密钥的设备)。然而,这些方法往往不能帮助识别设备和更复杂的漏洞或隐藏在NAT设备的。另一条线的研究[10,14,16,17,20,24,25,34,42,47,49,58]集中在静态或动态分析物联网设备或其固件来评估其安全性。虽然这些方法倾向于为个别设备产生更全面和准确的结果，但它们不能很好地用于大规模分析。首先，在市场上获得所有设备并不是一个可行的选择，因为在某些地区设备的可用性有限，而且它们的获取成本高得令人望而却步。类似地，由于市场高度分散，涉及到许多小型集成和分发供应商，设备厂商的软件并不总是可用的。其次，即使使用设备或其固件，分析本身也常常是冗长的、容易出错的和困难的，特别是考虑到设备供应商经常使用的“设备外壳”(例如，打包、混淆和加密)。因此，市场将受益于这样一种方法，即可以大规模快速识别易受攻击的设备，并缩小分析范围。

## 方法

在这篇文章中，我们提出了一个平台，可以加速脆弱设备的发现和分析，而不需要访问物理设备或其固件。我们的方法基于两个观察。首先，智能家居物联网设备供应商，尤其是中小型企业，往往依赖相同的组件(例如，从开源项目构建的软件，从通用供应商构建的硬件组件)来构建他们的设备。因此，相同的漏洞或不良的安全措施往往从一个物联网设备转移到另一个。因此，我们可以通过评估一个未知设备与已知的脆弱设备的相似度，将漏洞信息传播到未知设备。其次，设备的相似之处往往反映在它们的移动伴侣应用程序中，这些应用程序被广泛使用。结合这两个观察结果，我们可以构建一个平台，以一种可伸缩的方式识别脆弱的设备，而不需要物理设备本身或它们的固件映像。

在我们的平台,我们试图加快识别脆弱的过程设备,提供两种功能:(1)应用分析:找到一个设备的特点通过分析其伴侣应用,和(2)cross-app分析:发现设备的族, 即设备集群, 在应用分析中通过分析多个应用程序发现了一些相似性特征。集群有助于识别基于共享组件[8]具有类似漏洞集的应用程序。

## 结果概述

在我们的实验中，我们爬过谷歌的PlayStore[3]来寻找潜在的同伴，下载了3094个。在过滤掉一些噪音后，我们剩下了2081个应用程序的数据集(详见2.2节)

首先，我们找到了设备集群，即，设备族，包含在各个方面类似的设备，如软件或硬件组件、后端服务和网络协议。例如，在我们的分析中，我们发现19个设备族覆盖了来自122个不同供应商的139个应用程序，在一个设备族里的共享类似的软件组件。作为另一个例子，我们发现48个不同的族覆盖了460个设备共享类似后端服务。

其次，我们尝试使用已经识别的设备族来识别受给定漏洞影响的设备。在一个案例中,我们可以发现来自四个不同供应商的设备(应用估计安装超过215000用户),以前不知道的脆弱性的对于独立软件脆弱性和确认45个设备上的脆弱性的存在从四个不同的供应商（之前确认的通过其他来源）。在另一个案例中，我们识别出来自16个不同供应商的67台设备受到硬件安全问题的影响。总的来说，我们的平台已经识别了来自73个不同供应商的324个潜在的易受攻击的设备。在验证过程中，我们可以对来自43家供应商的179台设备进行判定(确认或不确认)，其中164台(91.6%)被确认为易受攻击。

## 贡献

本文的贡献如下:

* 它展示了物联网设备配套的移动应用如何为设备本身的安全性提供洞察力。
* 该方法在物理设备和固件映像不可用的情况下评估物联网设备的安全状态，证明了该方法的有效性。
* 提出了一个大规模收集、过滤、分析和聚类移动应用的平台。它通过对2000多个应用程序的分析和多维度的聚类来演示它的用途。
* 报告称，来自73个不同供应商的324个设备可能会引起安全问题。

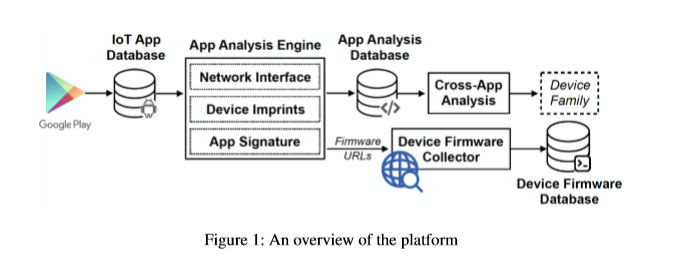
# 二、系统设计

## 2.1 总览

图1展示了我们平台的概述。我们平台的第一个组件是物联网应用数据库，它存储了从谷歌Play Store[3]抓取的智能家居物联网设备的配套应用。通过获取更多的应用程序(例如，当新的物联网设备上市或旧的应用程序更新时)，数据库不断得到扩展。

然后应用分析引擎对存储在物联网应用数据库中的应用进行分析。App分析引擎的目标是评估物联网设备(即，what the device is like)基于代码分析。具体来说，App分析引擎计算三件事:设备的网络接口、设备可能包含的独特的字符串(称为imprints)和配套应用的代码签名。App分析引擎的结果存储在App分析数据库中

跨应用程序分析引擎查询应用程序分析数据库，并识别不同设备之间的相关性，以构建设备族。设备族根据设备的相似性将来自不同供应商的不同设备分组在一起。相似性可以是在不同的维度上(例如，相似的软件、相似的硬件、相似的协议和相似的云后端服务)。设备族允许在相似的设备之间传播脆弱信息，具体地说，它允许对设备进行评估.



它允许物联网设备安全评估从一个设备或威胁的角度:1)为一个特定的设备,相似性允许快速评估设备是否脆弱,如果是是那种漏洞,和2)特定的漏洞,发现市场上设备的集合可能受影响的脆弱性。

为了方便漏洞确认，我们的平台包含一个称为设备固件收集器的额外组件。它利用应用程序分析引擎输出的代码分析结果(例如，固件URLs)和互联网搜索结果将固件映像下载到设备固件数据库中。这些固件映像使我们能够进一步确认跨应用程序分析引擎发现的漏洞。请注意，设备固件收集器不是我们平台的必要组件或必需组件。相反，它被用作帮助确认来自平台的发现的几种方法之一(更多细节见3.2节)。

在本节的其余部分中，我们将详细描述平台的每个组件。

## 2.2 APP 收集器

我们平台的第一步是收集智能家居物联网设备的移动伴侣app进行分析。为了达到这个目的，我们爬了谷歌Play Store2。我们总共下载了3094个Android应用程序，其中2081个包含在最终数据集中，由我们的平台进行分析。

在应用程序收集过程中面临的挑战是，识别哪些应用程序是物联网设备的移动配套应用程序。为了解决这个问题，我们从智能家居产品数据库SmartHomeDB[5]中手动选择281个种子应用初始化爬虫，并通过谷歌Play上种子应用与其他应用之间的连接(例如，关键字、建议和类别)，使用滚雪球采样来收集更多的应用。结果，最初下载了3094个候选应用程序。然而，我们观察到雪球取样有时会带来噪音。例如，管理手机摄像头的应用程序与管理家庭安全摄像头的应用程序是混淆的。锁定手机屏幕的应用程序与智能家居锁混淆了。为了从数据集中消除这些噪声，我们执行了过滤。过滤基于一个集群模型(亲和传播[26])，该模型根据应用程序在安装时请求的权限和应用程序在运行时可能调用的敏感Android api来对应用程序进行集群。我们将过滤部署在由相同种子样本提名的应用程序上，并且仍然是最大的集群。这种方法被证明是有效的:过滤后对200个应用程序进行随机手动检查，发现其中98.5%是真正的移动物联网伴生应用程序。在进一步的重复数据删除之后，数据集中将剩下2081个应用程序，并将它们输入到应用程序分析引擎中进行分析。注意，在研究的第一阶段，我们使用一个相对较小的数据集，并更多地关注于验证方法。我们的平台一直在运行，以收集更多的应用程序，以便在更大的范围内进行未来的分析。

## 2.3 APP 分析引擎

应用程序分析引擎分析收集到的移动伴侣应用程序，以便为单个设备构建设备配置文件。与之前专注于应用程序本身的工作不同[12,32,48,60]，这里的目标是间接地从应用程序计算设备的样子。我们通过独立应用三种方法实现了这个目标:设备接口分析计算的网络接口设备,独特的印记分析计算字符串可能有关的设备,和一个模糊散列分析计算代码签名的移动伴侣应用。在实践中,我们发现第一种方法更全面和丰富。尽管如此，剩下的两种方法在填补第一种方法不能轻易应用的空白方面仍然有用。

### 2.3.1 设备接口分析

设备接口通常很好地反映了设备的样子，例如:设备所使用的协议、设备运行的服务、设备支持的功能，有时还包括设备所使用的硬件组件。在不直接检查设备或其固件的情况下，我们通过分析其移动伙伴应用程序来估计设备接口，因为应用程序和设备在其网络接口中是互补的。应用程序和设备之间的点对点连接有助于这种估计，因为在这种情况下，应用程序接口是设备接口的直接反映;然而，这不是一个必要条件。对于涉及云或后端服务的设备，流行的后端服务(如Microsoft Azure IoT Hub[40])通常是通用的:它们倾向于在没有太多干预的情况下中继应用程序和设备之间的连接。这些设备也很好地配合了我们的方法，因为它们的应用程序界面仍然紧密地反映了设备的界面。我们对在线物联网设备数据库(SmartHomeDB)进行了一项研究，发现中小型供应商生产的大部分设备(76.3%)支持应用程序和设备之间的点对点连接。甚至像TPLink这样的大型供应商也支持云模式和点对点模式，以应对网络故障和隐私问题。这使得许多在市场上销售的物联网设备，尤其是易受攻击的设备，可以实现设备接口的一体化。

我们使用逆向方法来计算应用程序的网络接口，从应用程序可能接收到的网络响应消息开始，因为这些消息是设备输出的信息。我们首先确定应用程序中的消息处理函数，然后静态地确定响应消息的外观。然后，我们确定可能触发响应的请求。最后，我们部分实例化并执行app代码来重构请求[12,56]。图2显示了从移动伴侣app *com.Zengge.LED Wiﬁ Magic Color of Zengge Wi-Fi Bulb*提取的请求和响应示例。通过许多不同的请求和响应对(例如，使用UDP/48899、灯泡的TCP/5577和云服务器\*.magichue.net)，我们获得了对设备接口的良好估计。

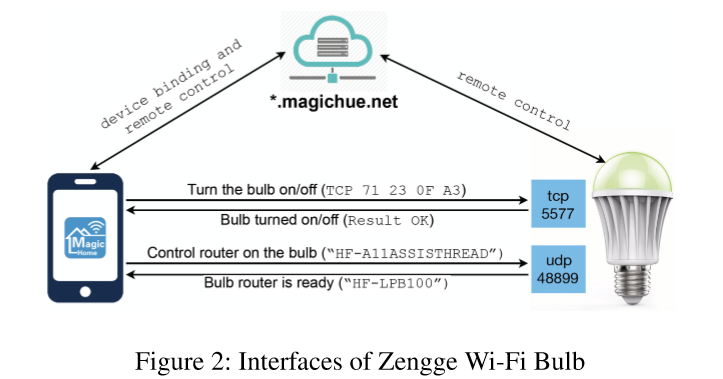


图2.Zengge WIFI 灯泡的接口

#### 提取响应

我们依赖于符号执行[33]来估计来自设备的响应消息的样子，而不需要实际运行设备。我们首先建立了一个伴随应用的控制依赖图*Control Dependency Graph (CDG)*和 数据依赖图*Data Dependency Graph (DDG)*通过Soot[51]然后我们从Android的标准网络接收功能开始（如<java.net.DatagramSocket: void receive(java.net.DatagramPacket)>)，象征性地向前执行移动伴侣app。每当我们遇到依赖于响应消息内容的分支时(例如，响应的字段将根据一个值进行检查)，我们将检查捕获为一个符号约束并派生执行。在所有执行结束后，符号约束的连接被存储为响应消息的“描述”。为了使来自两个设备的响应消息相似，它们必须满足相同的一组符号约束。

一个实际问题是决定何时终止符号执行。根据我们的经验，我们发现来自设备的有效响应(即，响应通过应用程序执行的检查)通常会触发应用程序的状态更改。这些状态更改可以是UI元素更改(例如，更新向用户显示的设备状态)，也可以是对本地注册表的修改(例如，将设备信息存储到配置文件、共享首选项或数据库中)。为了证实这一启发，我们从我们的应用程序集中随机抽取了存在于179个应用程序中的200个响应处理过程，并手动评估有效响应的影响。在这些响应中，有162个响应对UI元素有影响，有76个响应修改了本地注册表(有些响应同时改变了两个注册表);其中只有8个不会触发这样的更改，但是应用程序将响应内容(例如登录令牌)存储在全局变量中。该研究表明，状态变化可以作为有效响应处理终止的一个很好的近似。因此，我们将这些状态更改标记为终止符号执行并产生符号约束的连接点。此外，我们还观察到，无效的响应会被应用程序快速地丢弃。，只有几行代码)。因此，我们还设置了在终止执行之前要执行的过程数量的阈值。利用这两种启发，我们可以产生一组小而有意义的约束，这些约束可以很好地描述物联网设备可能产生的有效响应。

#### 成对请求和响应

下一步是识别将触发设备响应的应用程序发送的请求。在许多情况下，请求很容易识别:它与响应消息处理函数共同定位。然而，在其他情况下，由于请求可能位于不同的过程或类中，尤其是在app和设备之间的通信是异步的情况下，这就更加棘手了。在这些情况下，静态代码分析在识别匹配请求方面可能受到限制。

幸运的是，我们注意到，匹配的请求和响应常常共享它们处理功能的一个大型代码库(即，用于处理请求和响应的类和方法)。这种相似性反映在运行时的堆栈中。为了确认这一观察结果，我们检查了同一组200个响应处理过程的成对请求和响应，并评估了响应的堆栈跟踪与请求之间的相似性。图3显示了*Jaccard*相似性的累积分布函数(CDF): 81%的请求和响应共享超过61%的堆栈帧，超过一半(53%)的请求-响应对共享超过88%的帧。对于未配对的请求和响应，相似度几乎为零。因此，通过记录和比较应用程序的执行堆栈时，应用程序发出请求(即。，通过具体的执行)和处理响应(即在多个请求发送函数中，我们可以准确地确定最有可能触发目标响应的函数。例如，图4显示了*Chuango Wi-Fi*警报系统使用的请求和响应的堆栈跟踪。请求和响应是基于公共堆栈帧(例如，由同一用户单击触发的堆栈帧)进行匹配的，尽管它们位于运行在不同线程中的不同类中。

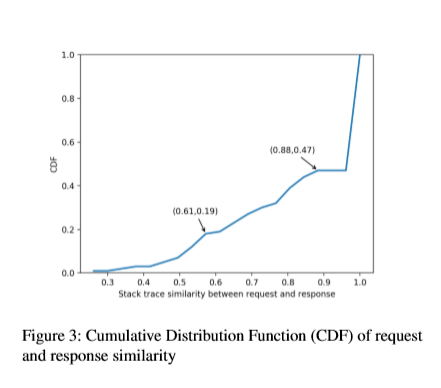


图3:请求和响应相似性的累积分布函数(CDF)

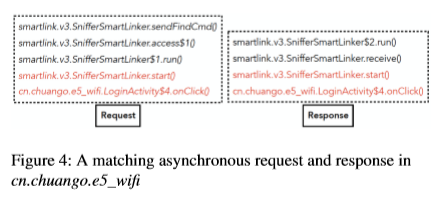


图4:一个匹配的异步请求和响应在*Chuango Wi-Fi*

#### 请求字符串重建

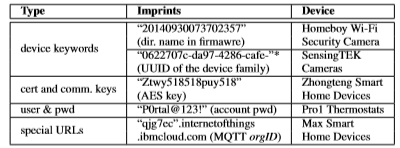
在识别匹配的请求和响应对之后，下一步是重构请求字符串。与响应不同，请求是由应用程序生成的。因此，我们可以重构一个完整的请求字符串，并将其与响应的一组符号约束进行比较。已经开发了许多通过程序切片和执行来重建程序值的技术[12,29,48,56]。在我们的平台中，我们采用了Tiger[12]中开发的实例化部分执行(IPE)技术。使用IPE的好处是，如果发现变量与请求字符串无关，它将计算并实例化变量为具体的值，从而极大地减少了需要探索的路径数量。此外，IPE还缓存代码片的输出，并在适用时重用结果，进一步降低了分析的复杂性。通过使用IPE，我们能够将重构请求所需的时间减少到一分钟以下。

设备接口分析产生的结果是一组请求和响应对。请求是完全或部分重构的请求字符串，响应是一组符号约束。一个设备据说有一个相似的接口，另一个设备可能同时接受相似的请求和输出相似的响应。

### 2.3.2 imprints 分析

设备痕迹(即。在应用程序中找到的唯一字符串可以帮助关联不同的设备。我们对应用程序和设备之间的通信中出现的印迹特别感兴趣，因为它们代表了设备的独特性。相比之下，也有应用程序印记，如应用程序开发人员的电子邮件或特殊的类名，可以识别一个应用程序或库。然而，它们对设备的指示性较差。

表1 设备印记举例



受Costinetal [16]先前工作的启发，该工作直接从嵌入式固件图像中提取烙印，我们还将重点放在四种设备烙印上：设备（后门）关键字，证书和密钥，非平凡的用户名和密码以及特殊URL。我们用来识别印记的方法很简单：我们构建应用程序的数据依赖图，然后从网络API向后检查，以找到影响这些API参数的应用程序中的常量字符串。请注意，这些API用于与设备通信。换句话说，如果唯一字符串与设备相关（例如，它们是对设备的请求或来自设备的响应的一部分），我们只会将它们用作烙印。解析器随后确定常量字符串属于哪个类别，以及它们是否经常出现（例如，忽略用户名和密码的admin）。表1显示了我们在数据集中收集的一些烙印的示例。（并且两个烙印都影响与设备的通信）时，它可以作为设备之间相似性的强有力指标。例如，通过使用烙印“ OBJ-000165-PBKMW”，我们能够关联VStarcam和OUSKI IP摄像机（后者后来被确认是前者的品牌重塑）。

尽管印记可以作为相关性的有力证据，但由于很多时候设备的印记不会在应用程序中显示出来，因此作为一种方法的印记分析通常不太适用。 例如，我们无法发现任何魔术关键字的存在，例如Constin等人报告的许多设备用于调试目的的“ xmlset\_roodkcableoj28840ybtide”（即反向由04882 joel后门编辑）关键字。 [16]。 这是有道理的，因为魔术关键字内置在固件映像中以用于调试，并且设备调试通常不是面向客户的应用程序所需的关键功能。 但是，它突出了印记分析的局限性以及我们需要全面的设备接口分析的原因。

### 2.3.3 模糊散列分析

我们使用的另一种方法是通过fuzzing（Fuzzing是指通过构造测试输入，对软件进行大量测试来发现软件中的漏洞的一种模糊测试方法）哈希来评估代码相似性。类似的移动配套应用程序通常表示类似的设备。因此，我们计算应用程序中发现的对象的*ssdeep*，在包括类、库和其他类型的资源(例如文本)，并比较应用程序之间的结果。与传统哈希算法(如SHA1)相比，使用模糊哈希的好处是我们可以将相似但不完全相同的对象关联起来。通过这种方法，我们能够识别出一些类似的设备。例如，在CHITCO和EDUP智能开关的配套应用中，发现有50.7%的对象匹配80/100的相似度，这两个设备后来被证实共享相似的软件。不过，请注意，设备之间的相似性并不一定意味着应用程序之间的相似性。我们注意到，在许多情况下，相似的设备有不同的应用程序。，应用程序是独立开发的)，因此导致模糊哈希分析失败。代码相似性对于识别明显的相关性以及其他分析方法难以应用的情况(例如，对于本机库)更有用。

### 2.3.4 模块化

在构建App Analysis Engine时，我们需要特别注意的是分析的模块化。 与为每个应用程序生成分析结果相比，我们采取这一额外步骤的原因是为了适应经常出现在IoT设备上的模块化相似性。IoT设备供应商（尤其是规模较小的设备供应商）通常由市场上许多现有模块组成其产品，例如，常见供应商的硬件组件，开源项目构建的软件，协议的二进制驱动程序代码等。 HiFlying Wi-Fi模块被许多供应商用来管理其设备的Wi-Fi连接。因此，对我们的分析也进行模块化也很重要，以便以单个设备组件的更细粒度跟踪设备相似性并检测漏洞传播（有关可跟踪组件的更多详细信息，请参阅第2.4节和第3.3节） 。

我们的设计基于这样的观察: 设备组件经常被应用程序中不同的代码模块管理(例如，类，包)。以之前的*HiFlying Wi-Fi*模块为例，使用该模块的设备(如*BeSMART*温控器)通常有两个单独的类

com.hifly.com.smartlink.v3.SnifferSmartLinkerSendAction;

com.besmart.thermostat.MyHttp

分别用于处理通过HTTP的Wi-Fi连接和用户交互。因此，我们从应用程序中推断出这种模块化(例如，基于类层次结构和调用堆栈)，并将上述分析方法应用于各个模块。

## 2.4 Cross-App分析引擎

App分析引擎输出的分析结果存储在App分析数据库中，由跨App分析引擎查询。跨应用分析引擎是为了检测不同设备之间的模块相似性而设计的。特别地，通过比较发现了四种类型的相似性:相似的软件组件、相似的硬件组件、相似的协议和相似的后端服务。

**相似的软件组件** 类似设备的接口，特别是应用程序接口，表明了不同设备的软件组件之间的强大连接。例如，我们能够关联来自16个不同供应商的72个不同的智能家居物联网设备，这些供应商可能使用相同版本的GoAhead web server4。这种关联是强大的，因为在很多时候，安全弱点表现在软件和安全的弱点上，一个设备的安全性能会直接影响到其他设备的安全性能。例如，我们能够识别出7个之前未报告的设备，这些设备容易受到已知漏洞的攻击，详见3.3节。

检测到的另一个有趣的现象是设备**重新品牌化**。在智能家居物联网行业，规模较小的供应商有时并不开发自己的产品。相反，他们从原始设备制造商定制物联网设备，然后用自己的品牌转售。从app分析结果中可以看出，重新命名的设备跨多个模块的设备接口与原始OEM（原始设备制造商）设备几乎相同。虽然设备重新品牌化本身不是问题，但它使固件更新和补丁中的安全实践复杂化。在某些情况下，例如如3.3节所示，一个漏洞由OEM的重新标记的设备继承，但是修复该漏洞的安全补丁不会继承。

**相似的硬件组件** 智能家居物联网设备可以建立在类似的硬件上(如Wi-Fi模块)。硬件组件的这种相似性有时会反映在设备配套应用程序中，因为应用程序需要配置硬件组件或与硬件组件进行交互。由于硬件的特殊性，这样的设备-应用程序接口可以是唯一的，允许使用相同硬件的不同设备之间有很强的相关性。例如，我们发现有两个Wi-Fi模块存在已知的证书泄漏安全漏洞，可能被来自35个不同设备供应商的166台设备使用。这些应用的总下载量超过27.8万次。

**类似的协议和后端服务** 特定的协议通常有自己的请求和响应格式。类似地，特定的后端服务常常公开标准的api。跨应用分析引擎可以检测网络接口的相似性，从而将使用相同协议或使用相同后端服务的设备关联起来，即使这些协议或后端服务没有文档记录。例如，我们发现来自11个供应商的39个不同的设备很可能使用SSDP协议，而众所周知，SSDP作为DDoS攻击的反射器是很脆弱的。另一个例子是，我们发现来自10个供应商的32台设备依赖于相同的云服务来管理它们的设备，而云服务有一个报告的安全弱点，允许攻击者通过设备ID和密码枚举完全控制物联网设备。

**未来工作** 物联网设备的安全性评估还可以从相似性分析中获益。例如，以前的工作[16,28]表明，相同的开发人员或分包商可能采用类似的编码方式，从而在他们的设备中构建了相同的一组糟糕的安全实践或漏洞。类似地，相同的开发工具链(例如，编译器)可能以类似的方式转换代码，从而导致相同的安全问题集[7,52,54]。作为未来的工作，我们计划将我们的分析扩展到更多的相似性维度，以获得更准确和完整的智能家居物联网设备评估。

## 2.4 设备固件收集器

我们的平台有一个额外的组件叫做设备固件收集器，它通过下载与被分析的应用程序相对应的设备固件镜像来丰富设备固件数据库。固件镜像的目的是帮助我们确认跨应用分析阶段的发现。在我们当前的平台中，我们通过两种方式收集设备固件。首先，我们利用固件下载链接内嵌在移动伴侣应用程序。因为物联网设备通常是无头的。他们经常通过配套的应用程序部署固件更新。因此，有时供应商会在应用程序中建立链接。这种联系通常是特殊的，可以通过印迹分析提取出来。其次，我们跟踪谷歌Play上的应用程序页面，这些页面通常直接指向设备供应商，以抓取潜在的固件文件。具体来说，我们使用谷歌自定义搜索API以编程方式通过供应商网站搜索固件映像文件。

对于收集到的文件，我们通过使用Binwalk[2]检查它们的格式来过滤掉非固件文件。**Binwalk**是一个著名的固件解包工具，它通过模式匹配从二进制**blob（二进制大文件格式）**中提取各种数据。一旦一个文件被确定为固件，就需要特别努力将固件版本与应用程序版本关联起来。正如我们将在3.3节中讨论的，这有助于我们确定某个特定的漏洞在哪个版本上得到了修复，以及修复后的补丁是否会对应用程序产生影响。

注意，不是所有的设备固件都可以下载。即使是我们所收集到的，仍有相当数量的固件被加密或混淆，使得分析变得困难。正如第4.2节所讨论的，这是漏洞确认中有待克服的一个限制。

## 3. 数据和结果

## 3.1 数据集和平台统计

**数据集**

我们的数据集总共包含2081个应用程序，这些应用程序是通过2.2节中描述的方法收集的。应用程序的平均大小为13MB(最小为23KB，最大为142MB)。这些应用程序在全球范围内传播(271种语言)，总下载量超过12亿。这些应用程序覆盖了1345个不同的设备制造商，估计大约4720个不同的设备型号。我们注意到这个数据集仍然是不完整的:通过比较我们数据集中某些类型的设备(如IP摄像头)和同类设备[13,21,23]，我们估计该数据集将覆盖物联网总配套应用的5-20%。

**测试台和应用程序处理**

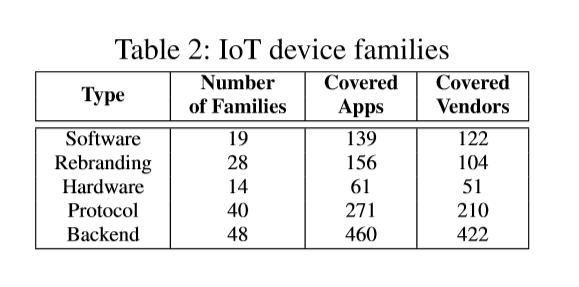
我们的app分析平台运行在一个4核，3.33GHz Ubuntu 16.04服务器上，16gb内存和1TB硬盘。Android模拟器是由Android开源项目AOSP 4.4.4编译的。

总的来说，我们的平台需要68.3个小时来处理这2081个应用程序，平均每个应用程序的处理时间为118.2秒。在我们的实验中，我们将最大处理时间设置为10分钟，大多数应用程序都在这个时间段内被成功处理。平台无法在超时窗口内完整分析73个(3.5%)应用，因此只能得到部分分析结果。此外，平台没有分析43个(2.1%)应用程序，因为我们使用的工具(即编译CDG和DDG时未能处理解释期间的应用程序字节码。总的来说，大约98%的应用程序要么被完全分析，要么被部分分析。

一个实际问题是应用程序的混淆及其对分析的影响。根据之前的研究报道，绝大多数(85.8%)的设备配套应用是由Android SDK中的标准工具(即Android SDK)生成的。它通过重命名类、方法和字段来混乱应用程序。虽然Proguard将skews引入到模糊哈希分析中，但它并不影响我们的主要分析方法(即由于它不混淆网络api，数据流和控制流。另一个关于应用程序打包的问题是，一些开发者使用打包器来加密他们的代码，这也会对网络接口分析产生影响。然而，与之前的研究[53]的观察结果相一致，恶意软件中经常会出现包装程序，而良性应用程序很少使用。在我们的数据集中，只有少数应用程序使用了商业包装。目前，我们没有对这些应用程序进行任何特殊处理。在开发更好的开箱工具方面有一条正交的研究线(例如，， DexHunter[59]和PackerGrind[57])，我们的平台可以通过这些工具进行补充。

**设备族**

表2显示了通过我们的跨应用分析检测到的所有设备族。例如，我们能够识别出19个不同的设备家族，包括122个不同的供应商和139个在家族内部使用类似软件的应用程序。作为另一个例子，我们能够检测到14个不同的设备族，它们覆盖了51个不同的供应商，这些供应商在这些设备族中使用了类似的硬件组件。注意，这些家庭并不相互排斥;一个设备可以与一个设备共享软件组件，与另一个设备共享硬件组件。我们确定的最大的设备族包括31个设备供应商，最小的设备族只包括2个设备供应商。图5更直观地展示了设备族图。



## 3.2 结果验证

我们的平台完全基于对移动伴侣应用程序的代码分析，不需要物理设备或它们的固件镜像。这是对智能家居物联网设备进行大规模安全分析的关键。然而，这种方法的缺点是结果的准确性:这种分析的输出(例如，是一种推测，指出了潜在的安全问题，需要用真实的设备进行验证。

在本文中，我们验证并报告了我们从分析中获得的一些结果，以证明该方法的价值。我们采取了混合验证的方法，同时考虑了实际的限制，比如预算。首先，我们尝试获取真实的设备并在本地环境中测试它(图6显示了我们为验证而购买的设备)。其次，如果我们尝试模拟或在某些情况下静态分析存储在设备固件数据库中的设备固件(根据第2.5节中讨论的方法构建)。第三，如果前两种方法都不适用，我们会搜索在线报告，包括供应商手册和网站、漏洞报告论坛、物联网黑客社区等。第四，我们与供应商合作，请求他们帮助验证结果。我们主要使用第二和第三种方法，因为第一种方法很贵，而第四种方法又很麻烦。，供应商无回应)。在确认我们的发现后，我们还试图通过在Shodan[4]上搜索该设备的在线状态来估计该发现的影响。

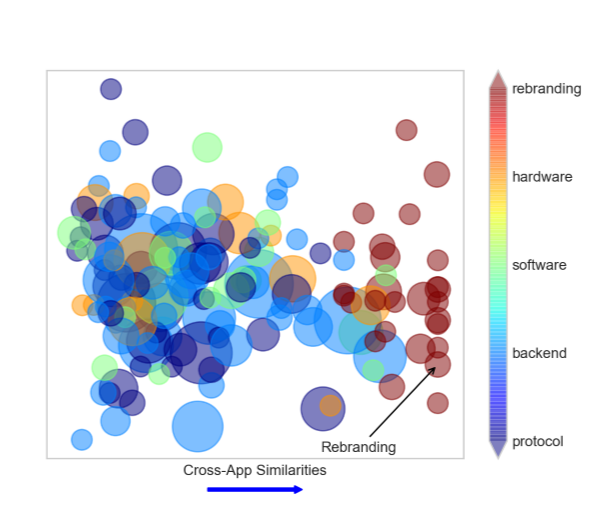


图5:设备族地图。圆圈大小表示家庭中设备供应商的数量(最大的圆圈表示31个供应商，最小的圆圈表示2个供应商)。

**道德问题** 测试漏洞和扫描现实世界的设备往往会带来严重的安全隐患。在我们的研究中，我们特别注意不要跨越法律和伦理的界限。对于真实的和模拟的设备，我们评估本地网络中只允许出站连接的设备。实验结束后，该设备将立即下线，以避免被利用和用作机器人。为了评估特定安全问题的影响，我们从Shodan的现有结果中收集数据，而不是直接扫描易受攻击的设备。这样，我们就不会引入额外的网络扫描。最重要的是，我们向所有受影响的供应商公布我们的调查结果，并避免包括任何仍未修复或正在接受调查的设备的真实名称。



图6:用于漏洞验证的智能家居物联网设备

## 3.3 结果概览

我们从威胁的角度展示了我们的发现，通过展示有多少智能家居物联网设备可能受到给定的漏洞或安全弱点的影响。然而，也可以从设备的角度来看待这些发现。，对于一个特定的设备，它可能遭受什么样的脆弱性或安全弱点。结果是令人鼓舞的:我们从73个供应商中发现了324个设备模型，这些模型可能存在安全问题。对于我们可以确认或不确认的设备，大约91%被确认为易受攻击。这些设备的用户总数估计超过1110万。

## 3.3.1脆弱的软件

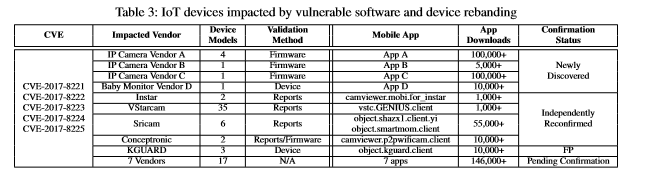
为了演示软件脆弱性如何跨设备传播，我们将分析应用到GoAhead web服务器中报告的五个高调漏洞(如表3所示)，许多智能家居物联网设备利用这些漏洞提供基于web的接口。这些漏洞的范围从认证旁路到后门帐户到远程代码执行。我们从移动伴侣app *object.liouzx.client* *NEO Coolcam* IP Camera，已知容易受到这些漏洞的攻击，并利用跨应用分析来识别软件中可能相似的设备。由于这些是相对较老的漏洞(2017年报告)，我们预计结果会更少。然而，总的来说，我们仍然确定了属于16个不同的供应商的72个设备模型，这些供应商与脆弱的设备共享类似的软件。为了验证结果，我们使用了3.2节中讨论的方法。我们能够通过在线报告确认来自四个供应商的45个设备模型确实是脆弱的。由于这些结果已经公开披露，我们将它们包含在表3中。另外，我们通过手动固件分析，确认了来自三个IP相机供应商供应商A、供应商B和供应商C的6个设备型号也受到了这些漏洞的影响。我们已经将漏洞告知了这些供应商，但还没有发布补丁。此外，我们通过真实的设备发现，来自供应商D的一个婴儿监视器设备也受到了这些漏洞的影响。卖主D已要求我们在他们进一步调查之前不要提供他们的名字。总共，我们确认了来自8个不同设备供应商的52个设备模型存在漏洞，其中*seven device models from four vendors newly discovered.*

在验证结果时，我们还遇到了这样的情况:我们的平台错误地将一个设备标记为易受攻击。平台输出的分析结果表明，*KGUARD*生产的三种设备模型与其他易受攻击的设备具有非常相似的接口。然而，我们在真实设备和模拟固件上的手动验证表明，*KGUARD*设备并不脆弱。我们检查了固件，发现*KGUARD*的软件配置确实与易受攻击的设备非常相似。具体来说，我们发现31个CGI程序和网页中有26个是相同的。但是易受攻击的代码被删除了。我们的假设是，由于这些设备相对较新的市场(即。，在漏洞被报告后)，*KGUARD*可能已经定制了软件配置，并在产品投放市场前修补了漏洞。

我们还想强调在结果验证过程中所做的两个观察。首先，尽管这些漏洞已经存在很久了，但我们仍然看到大量设备可能存在漏洞。一项Shodan搜索显示，在wild运行的设备中，仍有58,456台可能存在安全隐患，应用程序的总下载量超过28.2万次。这是一个只有2K个应用程序的数据集的结果。随着大规模的分析覆盖更多的设备模型和供应商，问题可能会变得更糟。这表明了智能家居物联网市场的一个普遍问题:市场高度分散，许多较小的供应商在出售设备后从不费心照顾设备。

另一个突出的发现是验证结果的困难，或者说是评估物联网设备安全性的普遍挑战。尽管跨应用程序分析告诉我们，16家供应商中有7家的17款设备可能也存在漏洞，但我们仍无法证实这些发现。验证没有成功有几个原因:一些设备没有针对美国市场，因此，我们无法获得，一些设备没有提供固件下载，因此我们无法评估，对于我们可以下载固件的设备，加密和包装使分析变得困难。此外，与设备供应商的合作非常困难。很多时候，我们关于潜在安全问题的电子邮件请求进入了一个黑洞(例如,从未收到供应商的回应)。我们认为，这也是市场碎片化的产物，因为较小的供应商往往不太关心他们产品的安全性。

表3:受脆弱软件和设备重组影响的物联网设备



## 3.3.2设备重新品牌化

对最近的一个漏洞(CVE-2018-11560)的调查，导致了设备重新品牌的另一个有趣的发现。该漏洞最初是在Insteon IP Camera 2864222(固件1.4.1.9)中报告的，设备上的嵌入式web服务器在解析CGI参数时丢失了边界检查，导致堆栈缓冲区溢出。我们使用Insteon IPcamera的配套应用程序作为交叉应用分析引擎的输入，来检测是否有其他设备也容易受到相同的漏洞的影响。令我们惊讶的是，我们发现主要的IP相机供应商Foscam提供了几乎相同的设备接口。我们最初怀疑相同的web服务器可能被两个供应商使用，但是后来通过研究我们发现Insteon IP camera 2864-222is实际上是Foscam IP camera FI8918W的品牌版本，它基于完全相同的硬件和软件，但是有不同的品牌名称。不足为奇的是，早期版本的Foscam IP摄像机也存在同样的漏洞，但是没有人报道过。

然而，有趣的是，Foscam实际上是在Insteon IP摄像机报道的脆弱性之前就把脆弱性拼凑起来了。我们检查了Foscam IP camera的固件历史，发现在2017年7月之前，有超过8个Foscam固件版本中都有漏洞代码，影响了至少15个Foscam型号。在2017年7月的固件更新(2.x.1.120)中，声称该漏洞被修复。然而，直到2018年该漏洞被报告后，这个补丁才出现在Insteon IP相机上。我们就此联系了福斯康，但他们的回复既没有证实也没有否认这一发现。相反，我们被建议升级到最新版本的固件。这凸显了智能家居物联网设备的另一个有趣问题。由于市场分散，较小的物联网供应商有时不开发自己的产品。相反，他们从原始设备制造商定制物联网设备，然后用自己的品牌转售。这使产品的安全管理变得复杂，并将客户置于危险之中，因为上游供应商的漏洞往往会传播到更广泛的下游供应商，但安全补丁不会。事实上，一项带有IP摄像头指纹(如服务器类型、时间戳)的Shodan搜索显示，虽然Foscam早在2017年7月就发布了补丁，但仍然有30.7%(33230个设备中有10210个)没有补丁到安全版本。

此外，我们的分析表明，重塑品牌的确并不罕见。通过2K应用程序的数据集，我们确定了其他27个重新命名的设备族，不包括Foscam示例。这些设备的例子包括来自Bayit和Orvibo的插件，来自CHITCO和EDUP的Wi-Fi插座等等。需要进一步验证，以确认这些设备是否从上游供应商处继承了任何漏洞

## 3.3.3硬件漏洞

不同的物联网设备供应商可能依赖于一个通用的硬件模块(如Wi-Fi、蓝牙)，如果该模块脆弱，可能会影响多个设备。然而，物联网设备供应商面临的挑战是，他们通常不会公布正在使用的硬件组件。因此，如果不拆开物理设备或打开固件以检查驱动程序代码，通常很难判断设备是否由于硬件组件的脆弱而容易受到攻击。

通过跨应用分析，我们确定了166个设备，它们属于35个不同的供应商，它们可能受到最近在硬件中发现的两个安全弱点的影响。例如，[36]最近的一项研究表明，Hi-Flying的Wi-Fi模块(HF-LPB100、HF-LPT100、HFLPB200)可以被敌方利用来窃取家庭网络Wi-Fi证书。Hi-Flying Wi-Fi模块是一个独立的802.11b/g/n模块，许多物联网设备使用它来提供无线接口。作为一个重要的功能，该模块支持凭据(如SSID、密码)通过SmartLink从设备伴侣app配置到物联网设备。根据[36]的研究报告，配置过程可能会泄露Wi-Fi证书:对手可以被动地监听流量并轻松收集家庭Wi-Fi网络证书。通过交叉应用程序分析，我们发现有26个应用程序(包括来自21个供应商的108个设备)可能受到这个安全漏洞的影响。这些应用程序的下载量已超过15.8万次。在另一个例子中，ESP8622是一种低成本的Wi-Fi微芯片，它出现在许多移动设备上(如移动设备)。，Wi-Ficontroller, smartplug)，据报道在它的ESP-Touch配置协议中有一个类似的安全漏洞。在我们的分析中，我们发现覆盖58devicesfrom14个不同供应商的21个应用程序可能受到安全漏洞的影响。这些应用程序总共被下载超过12万次。

在平台标记的设备中，我们能够确认来自16个供应商的67个设备确实受到了安全缺陷的影响(来自8个供应商的43个设备通过供应商响应得到了确认)。来自8个供应商的24台设备通过固件仿真、真实设备或在线报告得到确认。通过供应商响应，我们也能够识别出来自两个供应商的7台设备被平台错误地标记为易受攻击(即，9%误报率)。我们手动检查了这两个app，分析了假阳性的原因。在一个案例中，Revogi应用程序支持的14台设备被该平台标记为可能存在安全隐患。然而，其中四个(电源插头SOW324，电源带SOW321和SOW323，以及Smart Light LTW311)实际上并没有使用脆弱的硬件。该问题是由于平台进行的静态分析不够精确。由于该应用程序支持来自同一供应商的多个设备，因此控制单个设备的代码模块无法明确区分(即，有些模块是跨设备共享的，但有些是独立的)。因此，平台无法将与脆弱硬件对应的网络接口赋予特定的设备。相反，平台会输出应用程序支持的所有设备，认为它们可能存在漏洞。在另一个案例中，smanos应用程序支持的三款设备被平台错误标记。发现设备不使用脆弱的硬件,但代码模块和相应的网络接口,控制硬件包含在应用程序。这可能是由于应用程序开发人员构建应用程序在一些开源模板,包含硬件模块,或者设备供应商改变了他们的硬件配置在设备开发过程中,但是从来没有清理的程序代码。尽管如此，该平台使用的IPE方法是在静态分析的指导下构建网络接口的，只要Android活动可以访问代码片段，即使该活动实际上可能不会被实际设备触发。

## 3.3.4脆弱协议

与硬件组件类似，物联网设备供应商通常不会公布设备所讲的协议。这些协议的范围从比较开放和标准的，如UPnP、mDNS和SSDP，到专用的，如用于调试、渗透私有网络和各种其他用途的TDDP（TDDP stands for TP-Link Device Debug Protocol）。不知道设备会说哪种协议给设备管理带来了巨大的安全挑战，尤其是当协议被发现是不可攻破的，或者可以被对手利用来发动攻击的时候

通过跨app分析，我们可以识别出使用相同协议的设备，从而可能会出现类似的安全问题。例如，以前的研究[37]表明，SSDP协议可以被对手滥用，以发动DDoS攻击。SSDP查询如“SSDP:all”和“upnp:rootdevice”可能会导致响应大小数量级的增大，因此，如果可以公开访问Internet，则可以作为一个反射器来放大攻击者发送的请求。通过跨应用程序分析，我们确定了来自11个不同供应商的39台设备使用SSDP，尽管他们中很少有人清楚地记录了他们的设备使用的协议。因此，一旦这些设备在没有配置防火墙来阻止传入查询的环境中被激活，它们就可能成为DDoS攻击的反射器。我们很难知道有多少设备暴露在攻击之下，但是app的总下载量(超过1020万)表明，大量的设备可能会被攻击者利用。

通过平台对结果进行了验证。总的来说，我们能够确认来自6个供应商的18个设备确实在使用SSDP协议。其中一个设备Bixi手势控制器被平台错误地标记出来。Bixi手势控制器的情况很有趣:设备本身不会说SSDP，但它的配套应用会说，因此导致平台出现假阳性。原因是手势控制器通过手势控制设备。它不使用SSDP，而是依靠它的配套应用程序来使用SSDP来发现辅助设备，以便进行控制。在这种情况下，app的网络接口并不是设备接口的准确镜像，导致平台出现误报。

## 3.3.5后端服务脆弱性

物联网设备可能依赖于相同的物联网云后端服务来中继命令和控制(例如，渗透私人家庭网络)。当后端服务包含安全漏洞时，使用同一服务的多个IoT设备将同时受到影响。然而，如果不了解云服务的注册客户的详细信息，许多受影响的设备就会变得非常脆弱，直到独立地发现问题为止。

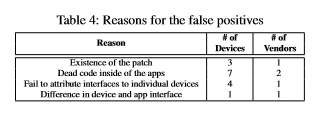
我们的跨应用分析可以帮助解决这个问题。在一个特别的案例中，安全漏洞最初是在deepsec2017[38]上被报告的，在那里，一个物联网云后端服务被发现使用非常短的设备id(即注册物联网设备。因此，任何使用该服务来中继命令和控制的物联网设备都容易受到设备ID和密码枚举攻击。成功的攻击可能使攻击者能够对设备进行身份验证，并将设备作为僵尸网络滥用。我们使用DeepSec、Yoosee中报告的易受攻击的设备作为交叉应用程序分析的种子，发现来自10个不同供应商的32台设备也依赖于相同的易受攻击的后端来中继命令和控制。虽然很难估算出在野外易受攻击的设备的实际数量，但这些应用程序的总下载量超过22.6万次。

成功的攻击可能使攻击者能够对设备进行身份验证，并将设备作为bot滥用。我们使用DeepSec、Yoosee中报告的易受攻击的设备作为交叉应用程序分析的种子，发现来自10个不同供应商的32台设备也依赖于相同的易受攻击的后端来中继命令和控制。虽然很难估算出在野外易受攻击的设备的实际数量，但这些应用程序的总下载量超过22.6万次。

## 3.4结果的准确性

总的来说，该平台标记了来自73家供应商的324台设备为潜在的易受攻击设备，我们能够确认来自38家供应商的164台设备确实易受攻击。这大约占了所有被该平台标记的设备的50.6%。在验证过程中，我们还发现来自5个供应商的15台设备被平台错误地标记为易受攻击。这占我们可以确认或不确认的所有设备的8.4%。假阳性)。表4列举了误报的原因和每个原因的实例数。假阳性的第一个原因是补丁的存在。漏洞被披露后，供应商可能会对设备进行修补。在这种情况下，应用程序-设备接口可能基本保持不变，但设备不再脆弱。这是该方法的一个基本限制，因为该平台的设计目的是只从应用程序提取信息，而不是设备。因此，如果补丁没有对应用程序造成任何影响，这种方法就无法区分脆弱的设备和补丁设备。第二个问题是应用程序内部的死代码。有时，这些应用程序可能包含设备实际上没有使用的代码(遗留代码，代码采用从其他地方没有清洁等.)。静态情况下，很难确定代码是否会在运行时被触发和执行。如果这些“死代码”与其他易受攻击的设备有相似之处，我们的平台目前可能会错误地包含这些“死代码”的分析结果，从而导致误报。假阳性的第三个原因是静态分析的不精确性。目前,静态分析技术平台不够精确属性所使用的网络接口对个别设备如果单个应用程序支持多种设备,这些设备有太多共同控制逻辑的内部应用程序。这个问题,除了上面列出的死代码的问题,不是一个基本问题的方法。相反，它们是我们用于分析平台中的应用程序的静态分析技术的产物。我们目前正在改进技术，以提高静态分析的准确性和精度。最后，我们遇到了一个异常情况，在我们的方法中，app接口不是设备接口的精确反映(即， Bixi手势控制器)。然而，由于设备的性质(即这在物联网设备中并不常见。我们正在探索谷歌播放商店，以确定是否有更多类似的设备。

表4:假阳性的原因



# 四、讨论

## 4.1各种各样的结果

在分析应用程序和设备之间的接口的过程中，我们有一些有趣的观察，在这里展示。

**令人困惑的信任模式**

我们注意到，物联网设备开发人员有时会对设备所处的本地环境产生一种令人困惑(如果不是相互冲突的话)的信任假设。一方面，他们似乎认为当地的环境(即，消费者的家庭网络)是不值得信赖的。他们使用加密和认证来保护应用程序和设备之间的通信。另一方面，他们对应用程序的信任程度过高，例如他们会在应用程序中嵌入加密密钥和认证凭证。在这种情况下，敌人在当地环境中很容易绕过设备开发人员建立的保护,只要对手访问Google Play Store,并有一个app逆向工程的基础知识.一个例子,TP-Link Smart Plug (HS110)接受命令的手机伴侣应用(LAN)内,其他地方可能从没有认证。供应商似乎担心这种设计的本地威胁，因此，加密通信。然而，使用的加密密钥(即最重要的是，它内置在应用程序中。因此，任何能够访问该应用程序的人都可以轻松地伪造通信。[50]也报告了这个问题。另一个例子是D-Link水传感器。D-Link水传感器需要其移动伙伴应用程序的认证。然而，用于认证该应用程序的凭据是固定的(即这些例子凸显了设备开发者对安全问题缺乏全面的理解。虽然在本文中我们不打算给出这个问题的解决方案，但是我们相信，考虑到安全性而开发的更标准的体系结构有助于限制提供给开发人员的自由，从而提高安全性。

**“方便”提供**

智能家居物联网设备通常是无头的，它们不提供直接的用户界面(如触摸屏、键盘)。因此，他们常常依赖移动伴侣应用来提供家庭Wi-Fi网络的凭据，以便加入网络。通过对设备接口的研究，我们发现配置方法正在从更多的用户交互方式(例如，AP模式、WPS和蓝牙等带外通道)发展为一种更加自动化和不干预的方式，在这种方式中，用户除了提供凭证外不需要做任何事情。这可能提供了便利，但很多时候是以安全性为代价的。这些较新的方法，如智能配置[30]和声波，通常会人为地在应用程序和设备之间创建一个链接，并依赖于链接到设备的链接。不幸的是，这些侧通道是公开可见的，因此允许凭据被泄漏。此外，即使不考虑侧通道的开放性，确保侧通道的安全也常常比正常的通信手段困难得多。这突出了一个长期存在的问题，即如何在可用性和安全性之间取得平衡。

## 4.2限制和未来的工作

本文讨论的方法的主要局限性是分析结果的准确性。由于我们的分析仅仅基于移动伴侣应用程序，我们从本质上局限于我们可以从应用程序获得的信息，有时我们可以获得的信息可能不是设备的准确反映。例如，一个设备可能已经修补了一个漏洞，而这个补丁根本没有改变设备接口。在这种情况下，我们的分析仍然会将设备输出为潜在的漏洞，因为我们的平台只检查应用程序，不会知道补丁的存在。然而，这是我们为了大规模研究物联网设备安全性而必须做出的权衡。我们相信，一个多阶段的解决方案可以帮助解决第一阶段的限制。我们的平台)通过识别潜在的易受攻击的设备来缩小分析范围，第二阶段通过更有针对性但更严格的分析(如固件的动态/静态分析、设备模糊化)来自动确认漏洞。

该方法的另一个局限性是，网络接口分析可能会降低目前的有效水平，比如物联网后端服务器或云将设备接口与应用程序接口显著分离。一个例子是谷歌和Amazon设备，其中大部分管理是通过云完成的。在这种情况下，我们的方法可以收集较少的设备软件信息。然而，诸如Wi-Fi证书供应模块和正在使用的后端服务等信息仍然可以在应用程序中找到，这使得平台能够预测这些组件的安全问题。

这项工作还可以从一个自动漏洞收集系统中获益。目前，这是一个手工过程:我们收集所有公开报告的设备。然后通过我们的平台将漏洞信息传播给更多的设备。一个自动漏洞收集系统可以帮助标记初始种子设备，以及从设备的角度评估安全性。，以查找给定设备可能存在的一组安全问题)。

另一个需要改进的方面是相似性分析的维度和粒度，如第2.4节所述。对App分析引擎的进一步改进可能允许平台检测设备软件栈的更精细组件(如web服务器、PHP解释器、web应用程序、OS、驱动程序)和其他维度(如web服务器、PHP解释器、web应用程序、OS、驱动程序)的相似性。，类似的开发人员，类似的开发工具链)。这将使我们能够更全面和准确地跟踪漏洞传播。我们把应用程序分析引擎的改进留给将来的工作。

一般方法，即，利用配套的应用程序分析来研究设备，也使一些有趣的应用程序，我们计划探索未来的工作。例如，通过应用程序分析，我们可能知道设备上有哪些类型的传感器，以及设备可能产生哪些类型的网络流量。这将允许家庭安全网关(市场上许多Wi-Fi路由器的默认组件)执行准确的保护配置文件，并实时检测物联网设备的异常行为。另一个例子是，我们平台中的跨应用分析引擎可以检测设备之间的差异，而不是相似性。这些差异可能使设备的指纹识别方法更加精确。

# 结论

在这篇论文中，我们提出了一个平台来加速在智能家居物联网设备市场中的脆弱设备发现。与以往研究物联网设备或固件镜像的方法不同，我们的平台通过分析设备的移动伴侣应用来间接检测设备相似性和跨设备传播的漏洞，从而为大规模分析提供了可行性。通过分析2081个移动伴侣应用程序，我们的平台能够发现73家供应商的324台设备存在潜在的安全隐患，其中38家供应商的164台设备被证实确实存在安全隐患。