SoK: Security Evaluation of Home-Based IoT Deployments

Omar Alrawi*, Chaz Lever*, Manos Antonakakis*, Fabian Monrose†
*Georgia Institute of Technology
{alrawi, chazlever, manos}@gatech.edu
†University of North Carolina at Chapel Hill
fabian@cs.unc.edu

***Abstract*—Home-based IoT devices have a bleak reputation regarding their security practices. On the surface, the insecurities of IoT devices seem to be caused by integration problems that may be addressed by simple measures, but this work finds that to be a naive assumption. The truth is, IoT deployments, at their core, utilize traditional compute systems, such as embedded, mobile, and network. These components have many unexplored challenges such as the effect of over-privileged mobile applications on embedded devices.**

Our work proposes a methodology that researchers and practitioners could employ to analyze security properties for home-based IoT devices. We systematize the literature for home-based IoT using this methodology in order to understand attack techniques, mitigations, and stakeholders. Further, we evaluate 45 devices to augment the systematized literature in order to identify neglected research areas. To make this analysis transparent and easier to adapt by the community, we provide a public portal to share our evaluation data and invite the community to contribute their independent findings.

I. Introduction

Security problems involving Internet of Things (IoT) continue to cause severe operational issues with high-profile attacks [1], mass exploitation of devices [2], and eye-catching headlines about "exotic" device hacking [3]. The demand for IoT devices — especially in the multi-billion-dollar residential market [4] — has created a modern-day gold rush. New and established companies are rushing to grab a piece of the IoT market. As time-to-market and production costs take priority over prudent security practices, the all-too-familiar sight of compromised IoT devices is numbing. Researchers and vendors are playing catch-up to address IoT insecurities, but much of the efforts are indistinct and ad-hoc.

Several working groups and market leaders have proposed standardizations for IoT devices [5]–[12], but unfortunately, they have not agreed on a solution. Additionally, the heterogeneity of home-based IoT devices contributes to these insecurities because although core functionalities are alike, specific features based on the device type can be vastly different. For example, an IoT vacuum cleaner and a home assistant device may use an embedded Linux operating system, but the running services on the device will be different. These differences make it difficult to analyze diverse home-based IoT products.

State-sponsored adversaries are well aware of these predicaments, and they have taken advantage to run sophisticated cyber operations [1]. To make matters worse, some vendors leave service backdoors in their devices that are later discovered and exploited by botnets [13]. Even unsophisticated criminal groups are taking advantage of the rampant insecurities to run distributed denial-of-service (DDoS) attacks [2]. Unfortunately, cleanup efforts and vulnerability patching are far from perfect, and as additional devices come online, the threats that target them become versatile, which enables them to spread even further [14]. To systematically address these security issues, researchers need to understand the landscape by conducting measurements and in-depth studies to classify and address the vulnerabilities. There are ample research efforts for home-based IoT security, but they are scattered. Our community needs an understanding of the current literature, a derivation of insights, and an identification of security gaps. The insights would allow the research community to formalize what insecurities are perpetuated, what are the proposed mitigations, and what responsibilities stakeholders bear. Further, these in-depth studies and classifications of literature can guide the community to help prioritize their efforts.

In this work, we propose a modeling methodology to study home-based IoT devices and evaluate their security posture based on component analysis, namely: the IoT device, the companion mobile application, the cloud endpoints, and the associated communication channels. Leveraging our approach, we systematize the research literature for home-based IoT devices to understand attack techniques, proposed mitigations, and stakeholder responsibilities. We use the knowledge to derive insights and identify research opportunities for our community. Additionally, we evaluate 45 home-based IoT devices that are available on the market today and provide an overview of their security properties across the IoT components.

Based on the systematization and evaluation, we compare the insights found between both approaches showing the commonalities and differences. We provide a list of mitigations for each component and propose strategies for different stakeholders to address the issues found. Most importantly, we establish a portal[1] where we invite our fellow researchers, vendors, and power-users to contribute new device evaluations and to reproduce our results using the published dataset and proposed methodology.

[1]The evaluation portal is available online at: https://yourthings.info.

Fig. 1: Typical home-based IoT setup.

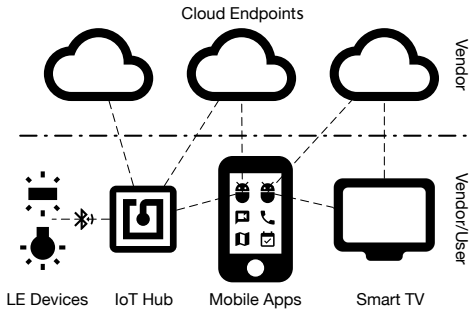


Fig. 2: Single IoT deployment.

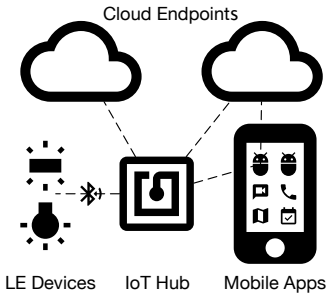
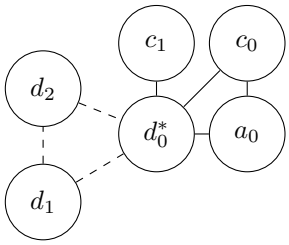


Fig. 3: IoT graph model.



II. Methodolgy

The contribution of our work is two-fold, the systematization of literature and the evaluation of home-based IoT devices. The work relies on an abstract model that segments IoT deployments into components, which we apply to the research literature and the device evaluations uniformly.

A. Abstraction Model Overview

We propose an abstract model to represent IoT deployments and their topologies. Figure 1 is an example of an IoT connected home with multiple devices. The approach involves segmenting each device into its respective topology as shown in Figure 2. Formally, we define an IoT deployment as a set of vertices V and edges E as illustrated in Figure 3. Overall, our abstract model has four main components: a set of devices (D), a set of cloud endpoints (C), a set of mobile applications (A), and a set of communication channels (E).

$$\begin{aligned} \text{where:} \quad & A, C, D \subset V; \quad D : \{d_i, i \in \mathbb{Z}\}; \\ & C : \{c_j, j \in \mathbb{Z}\}; \quad A : \{a_k, k \in \mathbb{Z}\}; \\ & E : \{e_l, l \in \mathbb{Z}\} \end{aligned}$$

For each device deployment, we construct a representative graph and examine the *security properties* for each component.

B. Security Properties

The security properties have three categories: attack vectors, mitigations, and stakeholders. Attack vectors are the methods used to circumvent the security of the IoT system. The mitigations define which measures should be taken to address the attack vectors. Lastly, the stakeholders represent the party responsible for mitigation.

Attack Vector. The device has three attack categories: *vulnerable services*, *weak authentications*, and *default configurations* that are defined as follows:

- **Vulnerable services** refers to vulnerabilities in running *services*.
- **Weak authentications** refers to weak or guessable *credentials*.
- **Default configurations** refers to the device operating with insecure *factory settings*.

The mobile application has three attack categories, *permissions*, *programming*, and *data protection* that are defined as follows:

- **Permissions** refers to a mobile application being over-privileged.
- **Programming** refers to the mobile application containing vulnerable *implementations*, including improper use of cryptographic protocols.
- **Data protection** refers to the mobile application hard coding *sensitive information*.

The communication of the components have two attack categories, *encryption* and *man-in-the-middle (MITM)* that are defined as follows:

- **Encryption** refers to lack of encryption or support of weak encryption protocols.
- **MITM** refers to the susceptibility to a man-in-the-middle attack.

The cloud endpoint shares the following attack categories with devices and communication edges: *vulnerable services*, *weak authentications*, and *encryption*, as defined above.

Mitigation. The mitigation categories, *patching* and *framework*, span all four components. Patching refers to mitigating an attack vector by patching the components through vendor updates or user attentiveness. The framework category mitigates fundamental problems that require a new approach.

Stakeholders. The stakeholder categories, *vendors* and *end-users*, span all four components. These categories indicate which stakeholder is responsible for mitigation. Figure 1 segments the IoT deployment into vendor-and-user-controlled networks. The cloud endpoint is controlled and mitigated by the vendor, while the components within the home network *may* expose configuration parameters so users can disable vulnerable features. For example, if the device has a known default password and the vendor allows users to change the default password, then the *end-user* can change the password to secure the device.

C. Systematization Approach

The systematization uses the proposed abstract model, which presents the literature uniformly across the categories discussed earlier identifying their attack techniques, proposed mitigations, and stakeholder responsibilities. Each work can fit into one or more of the IoT components. The literature for the systematization is chosen based on the following criteria:

- **Merit**: The work is unique and among the first to explore a given security predicament.

- **Scope**: The work focuses on the security (offensive and defensive) of home-based IoT systems.
- **Impact**: The work is regarded as significant based on the number of citations.
- **Disruption**: The work uncovers a new area that the community is currently investigating.

D. Evaluation Scope and Attack Model

Evaluation Scope. Our second contribution is the evaluation of home-based IoT devices using the abstract model to assess the security properties. We limit our scope to home-based IoT devices because they are relevant to the systematized work, they are readily available, and the experiment setup can be easily reproduced.

Attack Model. For the evaluation, we simplify the attack model to an Internet protocol (IP) network attacker. We recognize that there are more powerful adversaries that can attack low-energy (LE) based devices [15], but they require specialized resources that are not available in many home networks. We consider the exploitation of a hub device (communication bridge between low-energy and IP) to be equivalent to exploiting all the connected low-energy devices because a trust session exists between the hub and the low-energy devices. We exclude direct evaluation of low-energy devices but consider their hubs for evaluation. Finally, we consider the home network to be an *untrusted* network and we make no assumptions about the security state of mobile applications, modems/routers, or web browsers that have complete visibility to the home network ([16]).

III. Systemization of Knowledge

This section presents the systematization of home-based IoT research based on the abstract graph model (see Figure 3). Table I presents an overview of the systematized work and their corresponding subsections where we discuss the literature in detail. The component classification highlights the focus of the work while the attack vectors, mitigations, and stakeholders identify the approach. The systematization highlights representative work; hence it does not provide an *all-encompassing* reference to every related work.

A. Device

Most of the home-based IoT research focuses on the device because the device component is the cornerstone of an IoT deployment.

1) Attack Vector: Several works ([17]–[20]) explored IoT device configuration insecurities. Barnes [17], building on the findings of Clinton *et al.* [18], demonstrated how exposed hardware pins on a device allowed him to gain privilege access and spy on the end-users. Insecure configurations combined with weak or a lack of authentication can exacerbate the problem as shown by Chapman [21] and Rodrigues [22]. Weak or a lack of authentication in running services is a key contributor to several documented attacks [23]–[26]. These attacks demonstrate that device setup and configuration is an important process that the vendor must consider and evaluate for security flaws. Vendors should enforce strict authentication policies and for end-users to configure the device before allowing it to operate.

Max [23] assessed the security of the August Smart Lock and found that weak authentication and insecure default configuration broke the security of the lock. He found hard-coded credentials and debug configurations that allows modification and introspection of the lock. The work of Obermaier *et al.* [25] on cloud-based cameras found that although the device had what appeared to be a strong password (36 characters of alphanumeric and symbols), the password was the *MAC* address of the camera reversed and *Base64* encoded. Kavalaris *et al.* [26] showed that the Sonos device runs undocumented and unauthenticated services on high ports allowing LAN clients to fully control the device. The Sonos device was susceptible to unauthorized device pairing due to the lack of authentication. SmartAuth [24] found that the authentication problem also manifests itself in the IoT application platforms through over-privileged applications. Device pairing establishes a trusted channel between a client and their device. Further, IoT hubs bridge LE devices to IP networks, which have a pre-established trust relationship as shown in Figure 3. An attacker would exploit this specific process to circumvent the device or use it as a pivot point.

IoT application platforms expose a permission-based model to allow third-party applications to run. Fernandes *et al.* [27]–[29] showed how implicit trust to third-party applications can have major implications on the security of the device. There are many subcomponents within the device's platform, which can make securing the device difficult. Many vendors have good practices in place to ensure secure authentication and secure default configurations (as demonstrated by O'Flynn [30]), but core device services can suffer from side-channel information leakage. Ronen *et al.* [15] showed that although the Philips Hue device was reasonably secure, they were able to extract the master encryption key through a side-channel attack and combine it with a vulnerability found in the communication protocol, which resulted in a wormable exploit.

Flaws in firmware allow attackers to steal WiFi credentials [31], turn smart thermostats into spy gadgets [32], ransom them [33], run arbitrary commands on smart TVs [34], and control home assist devices covertly [35]. Costin *et al.* [36] conducted a large-scale study on firmware analysis and found an array of flaws. The literature showed that device security requires defensive approaches to secure side-channel, firmware, and hardware. The toolchain for software and hardware development has a well-defined secure development process that vendors must utilize.

2) Mitigations: To address vulnerable services, misconfiguration, and weak authentication, vendors patch through device updates, while inherent design flaws in IoT platforms are mitigated through new frameworks. Wang *et al.* [37] proposed a provenance-based framework to aggregates device activities across a deployment that can detect errors and malicious activities.

TABLE I: Systematization of the current literature using component based analysis. Each section corresponds to a graph component discussed in the methodology spanning attack vectors, mitigations, and stakeholders. The ✓implies the category of attack, mitigation, or stakeholder applies to the discussed literature.

Component	Ref	Attack Vector			Mitigations		Stakeholders	
		Vuln. Services	Weak Auth	Default Config	Patching	Framework	Vendor	End User
Device Section III-A	Ur13 [19]			✓	✓		✓	
	Costi14 [36]	✓			✓		✓	
	Chapm14 [21]		✓	✓	✓		✓	
	Kaval14 [26]	✓	✓	✓	✓		✓	✓
	Wuess15 [20]			✓	✓		✓	
	Rodri15 [22]		✓	✓	✓		✓	
	Lodge16 [31]	✓			✓		✓	
	Ike16 [18]			✓	✓		✓	
	Franc16 [33]	✓			✓		✓	
	O'Fly16 [30]	- -	- -	- -	- -	- -	- -	- -
	Ferna16 [27]	✓			✓		✓	
	Max16 [23]	✓	✓	✓	✓		✓	
	FlowF16 [28]	✓		✓	✓	✓	✓	
	Oberm16 [25]	✓	✓	✓	✓		✓	
	Barne17 [17]			✓	✓		✓	
	Herna17[32]	✓			✓		✓	
	Morge17 [34]	✓			✓		✓	
	Ferna17 [29]	✓		✓	✓		✓	
	Ronen17 [15]	✓			✓		✓	
	Dolph17 [35]	✓			✓		✓	
	Tian17 [24]	✓	✓		✓	✓	✓	✓
	Wang18 [37]	- -	- -	- -		✓	✓	
		Permissions	Programming	Data Protection				
Mobile Application Section III-B	Barre10 [38]	✓			✓		✓	
	Au12 [39]	✓			- -	- -	✓	✓
	Egele13 [40]		✓	✓		✓	✓	
	Vienn14 [41]		✓	✓	- -	- -	- -	- -
	Max16 [23]		✓	✓	✓		✓	
	Sivar16 [16]	✓		✓		✓	✓	✓
	Demet17 [42]	✓		✓		✓		✓
	IoTFu18 [43]		✓		- -	- -		✓
		Vuln. Services	Weak Auth	Encryption				
Cloud Endpoint Section III-C	Max16 [23]	✓	✓		✓		✓	
	Oberm16 [25]		✓	✓	✓		✓	
	Nandi16 [44]	✓				✓		✓
	Blaic16 [45]	✓	✓	✓	✓		✓	
	Wilso17 [46]			✓		✓	✓	✓
	Surba17 [47]	✓			- -	- -	✓	✓
	DTAP18 [48]	✓	✓	✓		✓	✓	✓
		Encryption	MITM					
Communication Section III-D	BEAST11 [49]	✓			✓		✓	
	Garci11 [50]	✓	✓		✓	✓	✓	
	LUCKY13 [51]	✓			✓		✓	
	Ryan13 [52]	✓	✓		- -	- -	- -	- -
	Foula13 [53]	✓	✓		- -	- -	- -	- -
	Alfar13 [54]	✓			✓		✓	
	Selvi14 [55]	✓			✓		✓	
	POODL14 [56]		✓		✓		✓	
	FREAK15 [57]	✓			✓		✓	
	CRIME15 [58]		✓		✓		✓	
	SMACK15[59]	✓	✓		✓		✓	
	Adria15 [60]	✓	✓		✓		✓	
	Zilln15 [61]	✓	✓		- -	- -	- -	- -
	DROWN16 [62]	✓	✓		✓		✓	
	Jasek16 [63]		✓		✓		✓	
	Kinti16 [64]	- -	- -		✓			✓
	Aptho17 [65]	✓			✓			✓
	Wood17 [66]	✓				✓		✓

SmartAuth [24] is a framework that identifies required permissions for IoT applications running on platforms like SmartThings and Apple Home. FlowFence [28] is a framework that splits application codes into sensitive and non-sensitive modules and orchestrates the execution through opaque handlers. This approach burdens developers because they must be mindful of what code operates on sensitive and non-sensitive data. Further, researchers can adapt techniques found in mobile application frameworks to address IoT platform insecurities.

3) Stakeholders: Table I shows that the main stakeholder is the vendor. Vendors are responsible for patching and updating vulnerable devices but can delegate some of the responsibilities to users through configurations. For example, users can mitigate insecurities by disabling problematic services on the device. SmartAuth [24] provides a derived authentication approach for applications on the device, but the implementation must be done by the vendor. Users gain control by having a choice about what permissions to authorize for third-party applications. Kavalaris *et al*. [26] showed how services that the Sonos device exposes create a security risk. Users can mitigate this risk through network segmentation, but it requires some technical expertise.

Not many devices allow users to fully configure running services or even disable them unless they have privileged access. Based on all the proposed mitigations, end-users can manage configuration or network segmentation residing on the home demarcation side as shown in Figure 1. End-users do not have much control and often are given a minimalistic interface, which limits the mitigation of vulnerable services. Vendors, on the other hand, bear the responsibility for keeping the device up to date.

4) Take Away: The literature addresses some aspects of device security. Devices have many components that contribute to their overall security like the platform permissions, unauthenticated services, insecure configurations, and software and hardware bugs. Further, they are amplified when combined. The device security is not purely in software, but vulnerabilities manifest themselves in hardware and side-channels as well. Embedded Linux is found in many of the devices, but there is no secure open IoT platform, which can incorporate newly proposed frameworks [24], [28], [37] by the community.

System patching addresses most of the vulnerabilities. The patching process is not perfect [32] and can be improved by good practices implemented in other areas of computing [67]. The end-users have almost no control or visibility into the operation of the device. Securely providing health telemetry and fine-grained configuration parameters can empower users to mitigate immediate risks. Users can deploy the device in many ways that go beyond the vendor's permissive assumptions, hence vendors should assume the device is Internet-facing when building security measures.

Similar problems are faced with general purpose computing systems that are publicly accessible and running vulnerable services or using weak authentication (SSH with guessable password). Adapting techniques from secure platforms and operating systems will improve the security posture of many IoT devices.

Device: Vulnerabilities in IoT systems manifest themselves in hardware, software, and side-channels and they are exacerbated when combined. There are efforts to address the security problems in IoT platforms, but common vulnerabilities across different products need a systematic analysis. Mitigating vulnerabilities relies heavily on vendors, but vendors should provide a way for users to control, inspect, and evaluate their devices. Adapting mature technology to manage IoT devices can significantly improve the security of IoT.

B. Mobile Application

Many of the home-based IoT devices have a companion mobile application to control, configure, and interface with the device. We represent the mobile application as a vertex in our abstract model (see Figure 3). Mobile applications can be leveraged as an attack surface against IoT deployments.

1) Attack Vector: Acar *et al*. [68] identified five different areas of Android mobile application issues, namely permission evolution, permission revolution, webification, programming-induced leakage, and software distribution. We adapted Acar's approach and identified three major classes of insecurities that effect IoT devices: over-privilege (permissions [38], [39]), programming errors (programming [40]), and hard-coded sensitive information (data protection [41]). Max [23] showed how programming errors leak sensitive information about the device and the cloud endpoint. Max used the sensitive information to dump credentials, escalate privileges, and circumvent the security of the August Smart Lock. Apart from Max's work, there are no **direct** attacks leveraging the mobile application to circumvent an IoT device.

Chen *et al*. [43] presented IoTFuzzer that instruments the mobile application within an IoT deployment to find bugs on the IoT device. Chen's approach is unique and leverages the semantics that the vendor programmed into the application. Although there are no reports of this technique used in the wild, theoretically an attacker can use the same approach to escalate privilege on an IoT device. Sivaraman *et al*. [16] showed how a mobile application can be used on a local network to collect information about available home devices and then reconfigure the router/modem firewall rules to make the devices Internet facing. Hanguard [42] showed how permissive security assumptions by vendors about the LAN can expose an IoT device. Companion mobile applications are an entry point to the device and vendors often assume that the deployment network is trusted and secure. These assumptions can have grave effects on the security of the device especially when devices rely on unauthenticated services or unencrypted communications.

2) Mitigation: Hanguard [42] proposed a user-space mobile application that interfaces with the router to control access through role-based access control (RBAC). Hanguard's approach will prevent the attack discussed by Sivaraman *et al*. [16] but cannot stop attacks from a compromised *companion* application. Securing the mobile application by adhering

to best practices discussed in Pscout [39], Barrera *et al*. [38], Egele *et al*. [40], and Viennot *et al*. [41], reduces the attack surface. Unfortunately, as Viennot *et al*. [41] showed, a large portion of the applications in the Google Play Store contain issues relating to permissions, programming errors, and information leakage. Mobile application platforms are mature and have built-in security facilities to promote good practices. Developers and vendors should adhere to best practices and audit their mobile applications periodically.

3) Stakeholders: The mobile application component relies on both the user and the vendor. This is partly due to the permission model that most mobile platforms provide to end-users. Hanguard [42] provides the user with a system to deploy inside the local network through routing rules (user demarcation Figure 1), which does not involve the vendor. Sivaraman *et al*. [16] proposes that users should be vigilant when running mobile applications on their networks and only use authorized stores (Google Play, Apple App Store, etc.). The vendors must address programming errors and secure information storage through updates. Vendors must familiarize themselves with the mobile platforms to deploy secure applications or use a reputable third-party developer to provide secure development expertise.

4) Take Away: The work of Acar *et al*. [68] showed the maturity of the mobile application security field. An inherent trust is given to mobile applications, which in many cases control core components of an IoT device or a cloud service. Max [23] and IoTFuzzer [43] demonstrated how to abuse the implicit trust between mobile applications and IoT devices or cloud services. IoT vendors and developers should adhere to platform development guidelines and leverage security features to ensure proper deployments. Limiting mobile application access to the device through fine-grained controls is a promising direction that can reduce the attack impact. Lastly, Hanguard's [42] approach should be further investigated to provide end-users with control to mitigate risks.

Mobile Application: Mobile applications are trusted by IoT devices and attackers have leveraged that trust as an attack point. Vendors should make conservative assumptions about the trust relationship and limit the interactions with core services. Mobile applications still suffer from over-privileged permissions, programming errors, and hard-coded sensitive data. Adhering to established secure development guidelines in mobile platforms will improve IoT security.

C. Cloud Endpoint

Cloud endpoints are the Internet components of the IoT deployment, and in a sense, they define what IoT is. They provide core services like remote administration, alerts, and digital content. The IoT devices and their mobile applications trust these cloud endpoints, which gives adversaries an additional attack point. We model the cloud endpoints as vertices in the abstract graph model (see Figure 3).

1) Attack Vector: The attack by Max [23] is a great example that touches on all components of the IoT ecosystem. The attack discovered insecure application program interface (API) on the cloud endpoint for the August Smart Lock, which escalated a guest account to an administrator account. Blaich [45] audited the Wifi Barbie doll for various vulnerabilities and found that the cloud endpoints did not authenticate firmware downloads, had multiple cross-site-scripting vulnerabilities, allowed username enumeration, had no brute force limiting, and issued never expiring cookies. Obermaier *et al*. [25] audited the cloud endpoints of surveillance cameras and showed that an attacker can inject footage, trigger false alarms, and carry out a denial-of-service attack against the camera system. These attacks were possible due to vulnerabilities introduced in the configuration of the infrastructure, vulnerable services, and insecure APIs. Zuo *et al*. [69] leveraged client-to-cloud trust to implement AutoForge, which forges requests from the mobile applications to the cloud endpoints enabling password brute-forcing, password probing, and security access token hijacking. Implicit trust between IoT components is sensitive and vendors must verify endpoints before allowing them unfettered access.

IoT integration platforms, like IFTTT [70], automate.io [71], and CloudWork [72], are third-party cloud endpoints. They use OAuth tokens to connect multiple IoT devices to perform user programmed tasks. Surbatovich *et al*. [47] studied the security implications on privacy and integrity when using recipes[2] and showed that some recipes can allow attackers to distribute malware and carry out denial-of-service attacks. Nandi *et al*. [44] reported a similar type of user-induced programming error through trigger-action programming (TAP), which led to an incorrect event triggering or a lack thereof. Fernandes *et al*. [48] pointed out that the cloud integration platforms can be compromised, which might expose the user's OAuth tokens publicly. These scenarios are likely to happen based on recent platform compromises like Equifax [73] and Orbitz [74]. The work of Wilson *et al*. [46] did not identify an attack vector on the IoT ecosystem, but it studied the privacy and trust that users place with IoT vendors. These attacks show that cloud integration services lack fine-grained control and they leak private and sensitive information that can lead to a breach.

2) Mitigation: To mitigate these attacks, Max [23], Obermaier *et al*. [25], and Blaich [45] recommend proper configuration and secure authentication mechanisms. Surbatovich *et al*. [47] offered a framework to analyze the cloud platform recipes, which motivated later work. Nandi *et al*. [44] proposed an automatic trigger generation system that analyzes user-defined triggers for errors and rectifies them by rewriting the triggers. Fernandes *et al*. [48] proposed the use of a decentralized framework for trigger-action programmable platforms called DTAP. The DTAP platform is a shim between the IoT cloud platform and the user's local network and brokers access to the IoT devices based on transfer tokens

[2]recipes are high-level programmable instructions that are used to trigger IoT device actions based on an occurrence of an event.

(XTokens). The mitigation techniques include securing cloud endpoints, offering tools to analyze third-party integration services, assisting developers in generating correct triggers for their applications, and providing short-lived tokens with constrained access to a device's functions.

Somewhat related, Wilson *et al*. [46] looked at empowering IoT users that trust the vendors with their private data. The technique is known as TLS-Rotate and Release (TLS-RaR), which requires an auditor entity collecting TLS packets to request the session key from the vendor to decrypt the communication. The vendor then rotates the TLS session key and discloses to the auditor the prior key to decrypt the collected TLS packets. The audit system must be deployed on the end-user demarcation side and collects traffic for devices that they wish to audit.

3) Stakeholders: The vendor controls the cloud endpoints (see Figure 1) and the users do not have a way to inspect or control what their device sends to the cloud endpoints [66], [75]. Additionally, third-party cloud providers offer infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) to IoT deployment. Many of the IoT devices rely on cloud-based infrastructure to run their services. Unplanned outages[76], infrastructure compromises[77], and intentional attacks[78] impact the deployment of the cloud endpoints. When it comes to cloud infrastructure configuration and API implementation ([23], [25], [45]), the vendor is responsible for the mitigation of the vulnerabilities.

Newer IoT devices are taking advantage of managed IoT platforms, which offload much of the security responsibilities to the public cloud providers. On the other hand, the majority of the proposed frameworks ([44], [46], [48]) are user-centric and give end-users visibility and control in a limited way. The work by Fernandes *et al*. [48] and Wilson *et al*. [46] is a hybrid approach and can be deployed jointly by vendors and users or by a trusted third-party. As for cloud providers, the vendor can mitigate their exposure by diversifying and over subscribing to different cloud providers.

4) Take Away: IoT cloud endpoints exhibit insecure cloud deployment through configuration and API implementation, but these vulnerabilities can be addressed with readily available tools for cloud security. Additional measurements are needed to further understand the extent of these misconfigurations in cloud deployments. The Censys Project [79] is a valuable source of data that can allow researchers to historically analyze IoT infrastructure. Further, the IoT cloud integration platforms introduce new challenges that mimic classical work like Decentralized Trust Management [80]. Integration cloud platforms offer users a way to chain multiple IoT devices to execute tasks based on an event, and they suffer from over-privilege recipes and privacy implications, which is demonstrated in the work of Surbatovic *et al*. [47].

Fernandes *et al*. [48] utilized prior techniques for the IoT cloud platforms by applying trust management systems and token authentication protocols to the IoT platforms. Vendors are adapting managed IoT cloud platforms, which shifts the security responsibility to cloud providers like Amazon IoT Core [81], Azure IoT Hub [82], and Google Cloud IoT [83]. IoT cloud endpoints are relying more on third-party infrastructure to deploy and run their services, which means vendors should consider a contingency plan for unplanned outages and infrastructure compromises. Additional studies are needed to understand the managed IoT cloud platforms and what possible weaknesses exist.

Cloud Endpoint: The cloud endpoints suffer from misconfiguration and vulnerable services that can be properly secured using industry standards. Third-party cloud providers play an important role by offering securely managed IoT platforms, which vendors are adapting. Toolchains for developing, analyzing, and deploying third-party applications via integration platforms require additional attention.

D. Communication

Communication edges (see Figure 3) in IoT deployment fall into two classes of protocols, Internet protocol (IP) and low-energy (LE) protocol. Both communications can exist on the user demarcation (see Figure 1) of the network, but only IP communication can go over the Internet. Researchers from industry and academia both are heavily invested in the security of network communication because of its applicability in other areas.

Most home-based IoT systems implement four types of communication protocols: IP, Zigbee, Z-Wave, and Bluetooth-LE (BLE). IoT devices choose to use the IP suite for communication due to its reliability and proven capability of transferring incredible volumes of global network traffic. The IP protocol is stateless and offers no security, but it can be supplemented by the use of TCP and TLS/SSL protocols to provide the security features needed. Based on the literature, we identified five popular application layer protocols that home-based IoT devices use, namely: DNS, HTTP, UPnP, NTP, and custom implementations.

1) Attack Vectors: The DNS protocol is a lightweight protocol that Internet services rely on, but inadvertently leaks private information based on the recursive and client configurations. Kintis *et al*. [64] found that open recursive DNS that enable EDNS Client Subnet feature (ECS) [84] (which embeds a truncated portion of the client's IP address) have privacy implications. Selvi [55] demonstrated how a MITM attack on NTP was used to bypass HTTP strict transport security (HSTS). The HTTP protocol gives a more reliable mode of transportation, but like DNS and NTP, it does not provide any confidentiality or integrity. Bellissimo *et al*. [85] and Samuel *et al*. [67] demonstrated how an insecure protocol like HTTP allows attackers to MITM and backdoor the system software update process.

IoT devices widely rely on UPnP protocol to offer easy configuration and control. UPnP uses the HTTP protocol, hence inherits the same flaws [86]. Garcia [50] showed how attackers abuse UPnP because it lacks authentication, validation, and logging. GNUcitizen [87] demonstrated how an UPnP

enabled device is vulnerable to cross-site scripting (XSS) vulnerabilities, while HD Moore [88] presented statistics and measurements around UPnP enabled devices on the Internet. Their work demonstrates that unauthenticated and unencrypted use of application layer protocols enables attackers to mass exploit devices, which leads to additional attacks. TLS/SSL sessions provide confidentiality and integrity, which help address the inherent flaws in these communication protocols.

Researchers have thoroughly examined the TLS/SSL protocols and uncovered severe vulnerabilities. Starting off in 2011, BEAST [49] exposed the initialization vector (IV) flaw in TLS 1.0, which allowed attackers to predict the IV of the next message in the stream. In 2012, CRIME [58] showed how TLS sessions that allow compression, like Google's SPDY protocol, were susceptible to session hijacking. In 2013, AlFardan *et al*. [51] used malformed packets to infer time delays, a side-channel attack, in the MAC verification to statistically infer the plaintext from the ciphertext. AlFardan *et al*. [54] also showed how the RC4 stream cipher weakens the security of TLS sessions. POODLE [56] exposed a downgrade flaw in SSL 3.0 that allowed for insecure communication between two parties. Beurdouche *et al*. [59] found flaws in several client and server implementations of TLS/SSL libraries that allow MITM attacks, including the FREAK [57] vulnerability.

Additional attacks disclosed by Adrian *et al*. [60] and DROWN [62] illustrated the difficulty of implementing secure communication protocols. Many IoT communications are susceptible to MITM attacks because they support older versions of TLS/SSL protocols. TLS/SSL is also widely used in managed IoT platforms to secure the communication channels. Emerging managed IoT platforms, like AWS IoT Core [81], Azure IoT Hub [82], and Google Cloud IoT [83], implement custom protocols that utilize certificates and TLS/SSL. These protocols and platforms are sparsely documented but rely on time-tested technologies to implement secure end-to-end communication.

The BLE [89], Zigbee [90], and Z-Wave [91] protocols have many security problems. Ryan [52] showed a severe flaw in the key-exchange protocol for Bluetooth, which allows an attacker to passively recover the session key. Jasek [63] demonstrated how attackers can passively and actively abuse the generic attribute profile in the GATT layer found in Bluetooth network stack. Zillner *et al*. [61] showed how the Default Trust Center Link Key defined by the Zigbee Alliance [90] is the same across all devices. Fouladi *et al*. [53] showed how a hard-coded constant in the Z-Wave firmware is used to derive session keys, which eventually became publicly known. Legacy versions of LE protocols have critical security flaws, which many home-based IoT devices implement in hardware; hence limits their mitigation options.

Aside from the inherent flaws, LE protocols offer a proximity feature that authentication systems rely on to identify geographical presence. Ho *et al*. [92] showed how relay attacks were possible against LE protocols by serializing the LE packets and relaying them over IP. Researchers have shown that MITM relay attacks against LE protocols are practical and break the geographical proximity, which authentication systems rely on. These communication channels can have privacy concerns as demonstrated by Apthorpe *et al*. [65] and Wood *et al*. [66].

2) Mitigations: For HTTP, UPnP, DNS, and NTP protocols, the suggested mitigations include disabling the ECS feature in DNS, using updated versions of the NTP protocol (NTPv4), and using TLS/SSL with insecure protocols (HTTPS). For TLS/SSL implementation flaws, upgrading the server-side and client-side libraries to the latest version should address the vulnerabilities. Further, disabling weak or vulnerable TLS/SSL versions reduces exposure but loses backward compatibility. For LE-based communication, the first generation of Zigbee and Z-Wave protocols have critical flaws and have limited mitigation options. Vendors can disable insecure portions of these protocols [93] at the expense of compatibility.

A recent direction by researchers is the work found in Apthorpe *et al*. [65] and Wood *et al*. [66]. Wood *et al*. [66] proposed a system that monitors the home network and inform users of sensitive data sent by IoT devices. Apthorpe *et al*. [65] demonstrated how traffic shaping on the home network can prevent side-channel snooping. This direction of research requires additional attention to empower consumers in protecting their networks and privacy.

Devices electing to use Z-Wave must now opt for *Z-Wave Plus*, which has improved security [94] and over-the-air (OTA) update capabilities. Also, Zigbee added a new security model to allow for secure-key distribution known as Trust Center (TC) [95]. TC is a trusted entity within the Zigbee network that is authorized to distribute keys to Zigbee client devices. TC gives each Zigbee connected device a unique encryption key, unlike the legacy key distribution schema. To mitigate relay attacks in LE protocols, Ho *et al*. [92] introduced a touch-based-intent communication approach using body-area network (BAN) for signal propagation.

3) Stakeholders: End-users cannot address the communication flaws since the implementation is on the device, the cloud endpoint, or in the mobile application. Further, vendors have limited options in addressing the communication vulnerabilities since some flaws require a hardware upgrade, but in some cases they can disable them [93]. The vendors can patch vulnerable libraries on the device, the mobile application, and the cloud endpoints.

Internet service providers (ISPs) have visibility into the utilization of IP based protocols, but they are not directly responsible for any mitigation. For ISPs to be involved, they must provide network and legal policies that define their role. As for the LE protocols, vendors can mitigate legacy devices by disabling vulnerable pairing. Users can use alternate methods for pairing LE devices with IoT hubs if such options exists. Users can buy newer devices that offer next generation secure LE protocols, like Z-Wave Plus and Zigbee.

4) Take Away: Communication channels provide essential functions for home-based IoT. Home-based IoT devices have adapted industry standards for IP and LE protocols, but they suffer from legacy libraries that in some cases cannot be fixed.

Vendors bear the responsibilities for addressing the vulnerabilities in the communication channels. Further, cloud endpoints and mobile applications can be updated by the vendor directly, but vendors must be proactive and informed about vulnerabilities affecting their software. IoT devices continue to rely on insecure protocols like UPnP and, as we will show next, rarely encrypt their communication on the LAN. End-users do not know if their device or mobile application is vulnerable to weak encryption or MITM attacks unless they analyze and test the communication traffic. An informed power-user might segment their local network into trusted and untrusted zones to limit the exposure.

TLS/SSL addresses insecure protocols that are susceptible to MITM attacks, but they also exhibit flaws in their implementation and deployment. The work of Clark *et al.* [96] provided additional analysis regarding SSL and HTTPS. ISPs can provide reports outlining best network practices and statistics about device and protocol utilization. Managed cloud IoT platforms use custom communication protocols that rely on public-key infrastructure (PKI) and TLS/SSL protocols. Further studies are required to investigate protocols used by managed cloud IoT platforms. These new platforms are not well studied and warn for further investigation to identify any weaknesses.

Communication: IoT devices rely on insecure protocols that do not offer confidentiality or integrity but mitigate them by using TLS/SSL protocols. Many devices lack encryption on the LAN, which leave them susceptible to MITM attacks. The TLS/SSL protocols exhibit flaws in implementation and deployment and require vendors to be vigilant. Managed cloud IoT platforms use custom protocols, which require further auditing. ISPs have a wealth of information that can guide vendors to secure deployments.

IV. Evaluation

We evaluated 45 devices spanning categories that include appliances, cameras, home assistants, home automation, media, and network devices. A full overview of the devices is in Appendix A Table III. We used a mix of commercial and open-source tools to conduct the evaluation; all of the commercial tools have open-source counterparts. Our methodology and evaluation require minimal technical expertise to replicate and is deliberately devised to appeal to a wide range of technical audiences allowing them to contribute to this effort. Our evaluation results are summarized in Table II and additional details of the evaluation is found in Appendix A Table IV. Specific device evaluation cases are found in Appendix B.

A. Experiment Setup

Our network setup has three main components, the IoT subnet, custom Linux gateway, and an assessment machine. The assessment machine runs all our evaluation tools and sits on the same subnet as the IoT devices. Our gateway is a Debian Jessie Linux machine, which manages the network services (DHCP, DNS, etc.) and connects the IoT subnet to the Internet. Additionally, our gateway full-packet captures all IP traffic originating from the IoT subnet. We used a 24-port switch to connect wired IoT devices via Ethernet and a wireless access point for devices that require 802.11 WLAN. All the IoT devices are assigned a static IP based on their MAC address.

B. Data

We examine different types of data generated by analyzing the device, mobile application, cloud endpoint, and network traffic. The interaction between these components generate network traffic (node interaction via edges, see Section II-A) that we capture, extract, and classify into application-level protocols to build the evaluation tables in Appendix A Table VII. We generate the scan data based on security audit tools that assess running services on the devices and cloud endpoints, which we then provide the evaluation report in Appendix A Table V and Table VI. We use mobile application audit tools to find issues related to the security properties defined earlier in Section II-B. The audit reports provide summaries of over-privileged applications, embedded sensitive data, and programming errors. We use this data summary to generate the evaluation report for each mobile application in Appendix A Table IV.

C. Challenges

We faced several challenges evaluating the IoT deployments including, but not limited to, automated device updates, cloud endpoint classification, wireless network analysis, and decryption of iOS applications. Automatic updates bias our device evaluation because the device state changes when updates are applied, which we had to disable on configurable devices. Cloud endpoint classification was involved and required manual analysis to ensure high accuracy due to increased utilization of content delivery networks (CDN). The wireless access point used WPA2 configuration, which limited our visibility into wireless-to-wireless device communication from the data collected at the egress (gateway) point of our IoT environment. We ran two different access points that forced traffic to traverse the gateway so we can gain visibility. Apple iOS applications were encrypted in the App Store and required a jailbroken iOS device to download, decrypt, and copy the iOS application locally. Once we had a copy of the iOS application, we used various open source and commercial tools to audit them.

D. Device

We used Nessus Scanner [97] to scan devices for service discovery, service profiling, and vulnerability assessment. Nessus Scanner annotates the CVE [98] information with the versions of running services and provides a summary of their security state. Nessus Scanner uses the CVSS [99] scoring system to rate the severity of the discovered vulnerability on a scale from one to ten and categorizes them into low, medium, high, and critical.

TABLE II: This table is a summary of each evaluated device per graph component in Figure 3. There are four components, namely: the device (D), mobile application (A), cloud endpoints (C), and the communication channels (E). The evaluation uses Nessus scanner to assess the device and cloud endpoints; Kryptowire, MobSF, and Qark to assess the mobile applications; Nessus Monitor, ntopng, sslsplit, and Wireshark to assess the communication protocols. The device section summarizes the number of running services and issues found. The mobile application summarizes excessive permissions, sensitive data, or incorrectly use of cryptographic protocols. The communication category summarizes the susceptibility to MITM attack and the communication channel state as fully encrypted (●), partially encrypted (◐), or not encrypted (○). For additional details see Appendix A

Device	Device Services Appendix A Table VI		Mobile Application Appendix A Table IV			Cloud Endpoints Appendix A Table V		Communication Appendix A Table VII	
	Running Services	Security Issues	Over-priviliged	Sensitive Data	Crypto Issues	SSL Issues	Service Issues	MITM	Encryption
Amazon Echo	1	0	✓	✓		✓			●
Amazon Fire TV	1	0	✓	✓			✓		◐
Apple HomePod	4	0	—	—	—	✓			●
Apple TV (4th Gen)	3	0	—	—	—	✓			●
August Doorbell	1	0	✓	✓		✓	✓	✓	◐
Belkin Netcam	1	1		✓		✓	✓	✓	◐
Belkin WeMo Crockpot	0	0		✓		✓	✓	✓	◐
Belkin WeMo Link	1	1		✓			✓	✓	◐
Belkin WeMo Motion	1	1		✓		—	—	✓	◐
Belkin WeMo Switch	1	1		✓		✓	✓	✓	◐
Bose SoundTouch 10	4	1	✓	✓	✓	✓	✓	✓	◐
Canary	0	0	—	—	—	✓			●
Caseta Wireless	2	0	✓			—	—	✓	◐
Chamberlain myQ Garage Opener	1	0			✓	✓			●
Chinese Webcam	4	1	—	—	—		✓	✓	○
D-Link DCS5009L	3	2	✓		✓	✓		✓	○
Google Home	5	2	✓		✓	—	—	✓	◐
Google Home Mini	5	2	✓		✓	—	—	✓	◐
Google OnHub	1	0	✓		✓	—	—		◐
Harmon Kardon Invoke	5	1		✓		✓	✓		●
Insteon Hub	4	6	✓	✓		✓	✓	✓	○
Koogeek Lightbulb	2	0	✓		✓	—	—		●
LIFX Virtual Bulb	0	0	✓		✓	✓		✓	◐
Logi Circle	0	0	✓			✓	✓		●
Logitech Harmony	2	1	✓			—	—		◐
MiCasaVerde VeraLite	4	6	—	—	—	✓	✓	✓	◐
Nest Cam IQ	0	0	✓	✓	✓	✓			●
Nest Camera	0	0	✓	✓	✓	✓			●
Nest Guard	0	0	✓	✓	✓	✓	✓		●
Netgear Arlo	0	0	✓	✓	✓		✓		●
nVidia Shield	2	3	—	—	—	—	—		◐
Philips HUE	2	0	✓	✓		—	—	✓	◐
Piper NV	3	0	—	—	—	✓	✓		●
Ring Doorbell	0	0	✓		✓				●
Roku 4	2	0	✓		✓	✓	✓	✓	◐
Roku TV	2	0	✓		✓	✓	✓	✓	◐
Roomba	1	0	✓	✓	✓	—	—		◐
Samsung SmartThings	1	1	✓	✓	✓	✓			●
Samsung SmartTV	4	1			✓	—	—	✓	◐
Securifi Almond	2	1	✓		✓	—	—		◐
Sonos	3	3		✓		—	—	✓	◐
TP-Link WiFi Bulb	1	0	✓		✓	—	—		●
TP-Link WiFi Plug	0	0	✓		✓	—	—		●
Wink 2 Hub	4	4	✓	✓	✓	—	—	✓	◐
Withings Home	1	0	✓				✓		●

We consider any classification of the categories high or critical by the CVSS scoring system as problematic and note it in Table II.

We evaluated 45 devices and found a total of 84 running services and 39 issues related to those running services. We found devices with running services such as SSH, UPnP, HTTP web server, DNS, Telnet, RTSP, and custom services. Many devices configure TLS/SSL for their services, but their configurations had several issues. For example, the certificates were self-signed, they supported weak to medium ciphers, they used short TLS/SSL keys, they permitted the use of vulnerable versions of SSL (v2, v3, and CBC mode), and had expired certificates. Further, some devices ran outdated and vulnerable services that allowed remote code execution, leaked sensitive information, and ran unauthenticated services.

For example, the *Insteon hub* runs a web server with TLS on port 443 and listens on port 22 for SSH connections. The certificate used for the TLS connection is expired and self-signed, while the TLS service allowed for weak ciphers like RC4 and insecure protocol like SSLv3. Similarly, the *Wink 2*, *Sonos Speakers*, *nVidia Shield*, *Google Home*, *Samsung SmartTV*, and *Samsung SmartThings* all had issues with their certificates or TLS/SSL configurations. The *Wink 2* and *Sonos* both used short SSL keys of size 1024 bits. Other devices like *D-Link DCS5009L*, *Bose SoundTouch 10*, *Chinese webcam*, and *Securifi Almond* lacked encryption for service authentication, which allows any device on the LAN to snoop.

Devices that run UPnP services have no authentication or security built in and by default are insecure. Devices like the *MiCasaVerda VeraLite*, *Wink 2*, *Sonos*, *Bose SoundTouch 10*, *Samsung SmartTV*, *Logitech Harmony*, and *Roku* all run UPnP services that allow anyone on the LAN to control the device. Specifically, the *MiCasaVerda VeraLite* uses vulnerable versions of the UPnP service libraries that have public exploits, such as *libupnp 1.6.18* (CVE-2012-5965), *dropbear 2016.72* (CVE-2012-0920), and *UPnP RunLua* (CVE-2013-4863). A complete list of CVEs with CVSS scores of high and critical are found in Appendix B Table VIII.

We found 16 devices with running services that had no issues, and ten devices that did not expose running services. For example, the *Nest* camera uses a push/pull client approach, which limits the exposure of running services.

Findings. The device evaluation found issues related to the device setup, software updates, and service configurations. Additional evaluation results for each device is found in Appendix A Table VI.

E. Mobile Application

We used MobSF [100], Qark [101], and services from Kryptowire [102] to statically and dynamically evaluate each mobile application for the IoT devices. We looked at both the Android and the iOS applications and presented the vulnerable of the two [3] in Table II. There are 42 devices that have a companion mobile application. We analyzed a total of 83 mobile applications of which 41 are Android and 42 are iOS. We found that 39 devices had one or more issues related to permissions, sensitive data, or incorrect use of cryptography. We observed 24 over-privileged mobile applications that ask for permissions on the mobile device that are not used by the application code.

As for sensitive data, we found 15 mobile applications to have hard-coded sensitive data like API keys for *Google Geocoding*, *Google Maps*, *fabric.io*, *HockyApp*, *Localytics*, *Microsoft Virtual Earth*, *Umeng*, and other credentials to cloud and device services. We found 17 mobile applications that did not implement cryptographic protocols securely or had hard-coded static keys and initialization vectors (IVs). The cryptographic implementations relied on older or broken algorithms like AES-128 and MD5 hash, respectively. Other applications did not enforce SSL and allowed for communication over unverified connections.

Findings. The evaluation identifies issues with inherent trust between mobile applications and devices that the systematized work neglects. A summary of our mobile application evaluation is provided in Table II and additional details are found in Appendix A Table IV.

F. Cloud Endpoints

We used Nessus Scanner to discover, profile, and assess running services on the cloud endpoints. On the IoT network, we observed over 4,000 cloud endpoint domains across the 45 devices. We classified each domain into one of four categories: first-party, third-party, hybrid, and unknown. First-party refers to cloud-based services that run on the vendor's infrastructure, third-party refers to subscription services like content delivery networks (CDN), hybrid refers to cloud-based infrastructures (IaaS), like Amazon AWS or Microsoft Azure, that host IoT cloud services, and Unknown refers to unclassified infrastructure due to ambiguity. We classified 950 domains as first-party, 1287 domains as third-party, 630 domains as hybrid, and 1288 domains as unknown. The unknown category includes unattributable domains for a device. For example, the Hulu application running on a Smart TV uses an AWS CloudFront domain, which gives us no indication if the domain belongs to Hulu or the Smart TV.

For each cloud endpoint, we evaluated the running services and TLS/SSL configurations, if applicable. We found 18 devices that used outdated services, leaked sensitive information, lacked encryption for authentication, or ran a vulnerable service. We found eight devices using cloud endpoints that are vulnerable and have public exploits. Additionally, seven devices authenticated with cloud endpoints in clear text. We found 26 devices using cloud endpoints that have TLS/SSL configuration issues, like self-signed certificates, domain name mismatch, and support for vulnerable versions of TLS/SSL protocol.

We found ten devices that used misconfigured cloud endpoints, which allowed for sensitive information disclosure like file paths and running processes on the server. We saw four devices use cloud endpoints that ran outdated operating

[3]The portal contains the data for both platforms iOS and Android.

systems with expired vendor support (Ubuntu 10 and Ubuntu 12).

Findings. The evaluation found issues with deployment of unsupported legacy OS and sensitive information disclosure. We summarize our findings in Table II and provide additional details in Appendix A Table V.

G. Communication

We used Nessus Network Monitor [97], ntop-ng [103], Wireshark [104], and sslsplit [105] to profile the communication edges for each device. We manually inspected traffic and tested them for MITM attack using sslsplit. IoT devices connect with their components using IP based channels, represented as edges in the model graph (see Figure 3). We classified three types of connections, device-to-cloud (D-C), mobile application-to-device (A-D), and mobile application-to-cloud (A-C). We observed 43 devices connecting to cloud endpoints (D-C), 35 mobile applications connecting to cloud endpoints (A-C), and 27 mobile applications connecting to devices through the local area network (LAN) (A-D).

We categorized these connections into five application protocols, namely: DNS, HTTP, UPnP, NTP, and custom. The custom category refers to device-specific application protocols. Smart devices utilize many protocols, but in our lab, we only observe the five listed above. We found 41 devices used the DNS protocol, where 6 of them did not respect the network configured DNS recursive server, and instead used Google's or OpenDNS's servers. We found that 38 devices used the HTTP protocol and 34 of them used TLS/SSL sessions (HTTPS). We found 21 devices that used the UPnP protocol either by sending a multicast SSDP request or responding to an SSDP request. Additionally, we saw 25 devices that used the NTP protocol for time synchronization. We observed 28 devices that used custom protocols that were specific to a device. For example, Google products (OnHub, Home, and Home mini) all sent traffic to Google's servers using a custom protocol on ports 5228 and 5223.

The majority of the devices used encryption over the Internet (D-C). We found 25 devices that encrypted all their communication, 15 devices that partially encrypted their communication, and two devices that did not encrypt their communication to the cloud endpoints. As for the mobile applications (A-C), 24 encrypted all their communication, ten partially encrypted their communication, and one did not encrypt its communication to the cloud endpoints. On the LAN (A-D) we observed five devices that encrypted their communication, two devices that partially encrypted their communication, and 20 that did not encrypt their communication. Few devices, like the Chinese webcam, did not have a companion mobile application but provided an HTTP interface that allows any device on the LAN to authenticate and interact with.

In addition to the communication analysis, we actively MITM attacked every communication edge to test their susceptibility. We found in total 20 devices had one or more of their communication edges susceptible to a MITM attack. We found four device-to-cloud (D-C) communications that were susceptible, two mobile application-to-cloud (A-C) communications that were susceptible, and 20 application-to-device (A-D) communications that were susceptible.

Findings. The evaluation finds that not all communication channels are secured and lack endpoint verification. We found devices that leak usage information by forcefully using third-party recursive DNS servers. Table II summarizes the device encryption and MITM attack and additional details are found in Appendix A Table VII.

H. Mitigations

Device. Affected devices should patch through secure channels to ensure the integrity of the update. Vendors can limit running services on IoT devices and follow a client approach where the device is managed through cloud endpoints using push/pull requests. Device configurations can be remedied using a configure-before-operable approach, where the device will not activate without proper configuration and setup. Many devices follow a configure-before-operable approach, and it should be mandated by industry standards. Finally, endpoint (cloud or mobile) verification ensures only authenticated parties can interact with the device. Vendors can limit the interaction to a sandboxed environment and assign temporal fine-grained access control for required resources. Trusted endpoints should not operate with unfettered access, and devices should enforce authentication time-outs for **all** parties. Modern home-based IoT devices are equipped with enough compute power ([106], [107]) to apply many of the suggested mitigations, contrary to the popular belief that they are underpowered and energy-constrained devices.

Mobile. Over-privileged applications can have privacy concerns regarding user's activities. Mobile platforms should implement a system to derive permissions based on functional analysis of the application and grant permissions temporarily at runtime. Further, sensitive information, such as API keys, should be derived when the application is installed on the mobile device and stored in an encrypted key store. Cryptographic protocols are difficult to implement correctly, and therefore developers should rely on mature libraries with proper implementations. Finally, developers should adhere to the recommended guidelines that accompany these libraries.

Cloud. Managed platforms and configuration management tools can alleviate the vulnerable services on the cloud endpoints. Vendors should utilize commercial platforms that are managed by experienced professionals. Similarly, automating cloud endpoint configuration through API integration can reduce the chances of misconfiguration. For example, Let's Encrypt [108] can automatically renew certificates for servers. Cloud endpoints should not support insecure protocols, but instead, they should verify both endpoint devices and mobile applications.

Communication. Network communication between all IoT components should adhere to the same security standards (LAN or Internet). Vendors must use the latest secure protocols, offer limited functionality for backward compatibility,

enforce protocol upgrade requirements, and verify endpoints. Endpoint verification will ensure MITM attacks are not successful and protect the integrity of the communication. Vendors should default to a fail state if endpoints are not verifiable. Additionally, vendors can provide an option to install custom certificates in IoT deployments for transparency.

V. Proposals

A. Stakeholders

Vendors. Vendors have to get the security requirements correct for every component at every level, including the design, implementation, and deployment of IoT systems. Our evaluation shows that many vendors strive for device security but often fail with due diligence. Realistically, many vendors do not have **all** the expertise to develop, manage, and deploy these heterogeneous technologies. Vendors that lack expertise in specific areas can outsource to specialized third-parties to develop their product.

End-Users. Home-based IoT deployments transform simple home-based networks to complex enterprise-like networks. End-users can follow good security practices by configuring devices to use encryption, disable remote administration features, and segment their network. Most importantly, consumers can influence vendors by purchasing privacy-aware and secure devices. Our portal is meant to give an objective security assessment of IoT devices and allow consumers to make informed decisions.

Other Parties. Internet Service Providers (ISPs) are not direct stakeholders, but the ubiquity of home-based IoT devices affects the operation of their networks. Much of the traffic seen by the ISPs will be encrypted, but ISPs can identify devices by destinations, service ports, and communication frequency. ISPs can potentially implement technical remedies to block certain ports, but legal policies are needed to intervene. These decisions can pose policy and compliance disputes due to the global nature of IoT and international jurisprudence [4]. ISPs can offer their expertise in running and operating residential Internet networks that can help identify implications around home-based IoT deployments.

Cloud providers offer infrastructure-as-a-service to many IoT vendors and have years of experience in developing, running, and securing cloud infrastructures and platforms. Their offerings are economical and practical for vendors, but they do suffer from outages occasionally [76]. Cloud providers are playing an important role in securing IoT deployments and should continue to offer tailored cloud services that alleviate security responsibilities from vendors.

B. Recommendations

Measurements. We recommend additional measurements for inter-device communication, mobile application-to-device interaction, and trust relationship between IoT components. Inter-device communications (device-to-device and mobile application-to-device) are not well studied within the LAN. Many IoT systems, like home assist devices, auto-discover and interact with other devices on the LAN without users' consent, which warrants further investigation to understand the security and privacy implications as a result of these communications. Further, conducting longitudinal studies can expose latent flaws that are otherwise difficult to observe without temporal analysis.

Best Practices. Best practices and guidelines for the IoT components are readily available, but their utilization is low. Some of the evaluated devices have very good practices that other vendors can benefit from, including mobile application implementation, cloud service configuration, device provisioning, and secure deployment and interaction of components. These design and implementation patterns should be evaluated in-depth to understand their cost/benefits to vendors. Government legislation can encourage economical or policy-based incentives to influence vendors in adapting best practices.

Standards. Many well-established vendors have put forth standards for IoT systems, but there is no consensus among the community. Vendors and researchers should combine their expertise to jointly draft industry standards that provide techniques to address common mistakes found in home-based IoT systems. Some home-based IoT systems have cyber-physical components, like connected ovens, fridges, and water heaters. These classes of IoT systems must be regulated by safety mandates and code standards to ensure no physical harm can result from their abuse or components failure. The government must play an active role in the development of these standards to protect consumers' safety and privacy.

VI. Conclusion

This work systematized the existing literature for home-based IoT devices through an abstract model that allowed us to derive insights. We used the same methodology to evaluate 45 IoT devices and found much of the same issues discussed in the literature exist in IoT systems today. We make available our results and the evaluation dataset on our portal and invite researchers to contribute and reproduce our work. We envision this effort to be a central pillar for evaluating home-based IoT devices, providing data for researchers, and collaborating with vendors.

VII. Acknowledgments

We thank the anonymous reviewers and Jan Werner for their insightful comments and suggestions. We thank the Kryptowire team for providing the automated mobile application security analysis platform. This material is based upon work supported in part by the US Department of Commerce grants no. 2106DEK and 2106DZD, National Science Foundation (NSF) grant no. 2106DGX, and Air Force Research Laboratory/Defense Advanced Research Projects Agency grants no. 2106DTX and 2106EHP. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Department of Commerce, National Science Foundation, Air Force Research Laboratory nor the Defense Advanced Research Projects Agency.

REFERENCES

[1] C. Cimpanu, *Over 65,000 Home Routers Are Proxying Bad Traffic for Botnets, APTs*, https://www.bleepingcomputer.com/news/security/over-65-000-home-routers-are-proxying-bad-traffic-for-botnets-apts/, 2018.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *Proc. 26th USENIX Sec.*, Vancouver, BC, Canada, Aug. 2017.

[3] O. Williams-Grut, *Hackers once stole a casino's high-roller database through a thermometer in the lobby fish tank*, http://www.businessinsider.com/hackers-stole-a-casinos-database-through-a-thermometer-in-the-lobby-fish-tank-2018-4, 2018.

[4] *Unlocking the potential of the internet of things*, http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world, 2015.

[5] *IPSO Alliance*, http://www.ipso-alliance.org/, 2016.

[6] *AllSeen Alliance*, https://allseenalliance.org/, 2016.

[7] *AllJoyn Framework*, https://allseenalliance.org/framework, 2016.

[8] *Wikipedia - Open Connectivity Foundation*, https://en.wikipedia.org/wiki/Open_Connectivity_Foundation, 2016.

[9] *Industrial Internet Consortium (IIC)*, https://www.iiconsortium.org, 2016.

[10] *Thread Group*, https://threadgroup.org, 2016.

[11] *Standard for an Architectural Framework for the Internet of Things (IoT)*, http://grouper.ieee.org/groups/2413, 2016.

[12] *IoTivity*, https://www.iotivity.org, 2016.

[13] B. Rodrigues, *LuaBot: Malware targeting cable modems*, https://w00tsec.blogspot.com/2016/09/luabot-malware-targeting-cable-modems.html, 2016.

[14] Yegenshen, *Iot_reaper: A rappid spreading new iot botnet*, http://blog.netlab.360.com/iot_reaper-a-rappid-spreading-new-iot-botnet-en/, 2017.

[15] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *Proc. 38th IEEE S&P*, San Jose, CA, May 2017.

[16] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-phones attacking smart-homes," in *Proc. of the 9th ACM WiSec*, 2016.

[17] M. Barnes, *Alexa, are you listening?* https://labs.mwrinfosecurity.com/blog/alexa-are-you-listening, 2017.

[18] Clinton, Ike and Cook, Lance and Banik, Shankar, *A Survey of Various Methods for Analyzing the Amazon Echo*, https://vanderpot.com/Clinton_Cook_Paper.pdf, 2016.

[19] B. Ur, J. Jung, and S. Schechter, "The current state of access control for smart devices in homes," in *Workshop on Home Usable Privacy and Security (HUPS)*, 2013.

[20] C. Wuesst, *How my TV got infected with ransomware and what you can learn from it*, https://www.symantec.com/connect/blogs/how-my-tv-got-infected-ransomware-and-what-you-can-learn-it, 2015.

[21] A. Chapman, *Hacking into Internet Connected Light Bulbs*, https://www.contextis.com/blog/hacking-into-internet-connected-light-bulbs, 2014.

[22] B. Rodrigues, *ARRIS Cable Modem has a Backdoor in the Backdoor*, https://w00tsec.blogspot.com/2015/11/arris-cable-modem-has-backdoor-in.html, 2015.

[23] J. Max, *Backdooring the Frontdoor Hacking a "perfectly secure" smart lock.* https://media.defcon.org/DEFCON24/DEFCON24presentations/DEFCON-24-Jmaxxz-Backdooring-the-Frontdoor.pdf, 2016.

[24] Y. Tian, N. Zhang, Y.-H. Lin, X. Wang, B. Ur, X. Guo, and P. Tague, "Smartauth: User-centered authorization for the internet of things," in *Proc. 26th USENIX Sec.*, Vancouver, BC, Canada, Aug. 2017.

[25] J. Obermaier and M. Hutle, "Analyzing the security and privacy of cloud-based video surveillance systems," in *Proc. of the 2nd ACM IoTPTS*, 2016.

[26] S. P. Kavalaris and E. Serrelis, "Security issues of contemporary multimedia implementations: The case of sonos and sonosnet," in *The International Conference in Information Security and Digital Forensics*, 2014.

[27] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. 37th IEEE S&P*, San Jose, CA, May 2016.

[28] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks.," in *Proc. 25th USENIX Sec.*, Austin, TX, Aug. 2016.

[29] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Security implications of permission models in smart-home application frameworks," in *Proc. 38th IEEE S&P*, San Jose, CA, May 2017.

[30] C. O'Flynn, *A Lightbulb Worm?* http://colinoflynn.com/wp-content/uploads/2016/08/us-16-OFlynn-A-Lightbulb-Worm-wp.pdf, 2016.

[31] D. Lodge, *Steal your Wi-Fi key from your doorbell? IoT WTF!* https://www.pentestpartners.com/security-blog/steal-your-wi-fi-key-from-your-doorbell-iot-wtf/, 2016.

[32] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, *Smart Nest Thermostat: A Smart Spy in Your Home*, https://www.blackhat.com/docs/us-14/materials/us-14-Jin-Smart-Nest-Thermostat-A-Smart-Spy-In-Your-Home-WP.pdf, 2014.

[33] L. Franceschi-Bicchierai, *Hackers Make the First-Ever Ransomware for Smart Thermostats*, https://motherboard.vice.com/en_us/article/aekj9j/internet-of-things-ransomware-smart-thermostat, 2016.

[34] S. Morgenroth, *How I Hacked my Smart TV from My Bed via a Command Injection*, https://www.netsparker.com/blog/web-security/hacking-smart-tv-command-injection/, 2017.

[35] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proc. 24th ACM CCS*, Dallas, TX, Oct. 2017.

[36] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis, "A large-scale analysis of the security of embedded firmwares.," in *Proc. 23rd USENIX Sec.*, San Diego, CA, Aug. 2014.

[37] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter, "Fear and logging in the internet of things," in *Proc. 2018 NDSS*, San Diego, CA, Feb. 2018.

[38] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in *Proc. 17th ACM CCS*, Chicago, Illinois, Oct. 2010.

[39] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "Pscout: Analyzing the android permission specification," in *Proc. 19th ACM CCS*, Raleigh, NC, Oct. 2012.

[40] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in android applications," in *Proc. 20th ACM CCS*, Berlin, Germany, Oct. 2013.

[41] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," in *Proc. of the 2014 ACM SIGMETRICS*, 2014.

[42] S. Demetriou, N. Zhang, Y. Lee, X. Wang, C. A. Gunter, X. Zhou, and M. Grace, "Hanguard: Sdn-driven protection of smart home wifi devices from malicious mobile apps," in *Proc. of the 10th ACM WiSec*, 2017.

[43] J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. C. Lau, M. Sun, R. Yang, and K. Zhang, "Iotfuzzer: Discovering memory corruptions in iot through app-based fuzzing," in *Proc. 2018 NDSS*, San Diego, CA, Feb. 2018.

[44] C. Nandi and M. D. Ernst, "Automatic trigger generation for rule-based smart homes," in *Proc. ACM PLAS*, 2016.

[45] A. Blaich and A. Hay, *Hello Barbie Initial Security Analysis*, https://static1.squarespace.com/static/543effd8e4b095fba39dfe59/t/56a66d424bf1187ad34383b2/1453747529070/HelloBarbieSecurityAnalysis.pdf, 2016.

[46] J. Wilson, R. S. Wahby, H. Corrigan-Gibbs, D. Boneh, P. Levis, and K. Winstein, "Trust but verify: Auditing the secure internet of things," in *Proc. of the 15th MobiSys*, 2017.

[47] M. Surbatovich, J. Aljuraidan, L. Bauer, A. Das, and L. Jia, "Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes," in *Proc. 26th WWW*, 2017.

[48] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Decentralized action integrity for trigger-action iot platforms," in *Proc. 2018 NDSS*, San Diego, CA, Feb. 2018.

[49] US-CERT/NIST, *CVE-2011-3389*, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-3389, 2011.

[50] D. Garcia, *Upnp mapping*, 2011.

[51] N. J. AlFardan and K. G. Paterson, "Lucky thirteen: Breaking the tls and dtls record protocols," in *Proc. 34th IEEE S&P*, San Francisco, CA, May 2013.

[52] M. Ryan, *Bluetooth Smart:The Good, The Bad, The Ugly... and The Fix*, https://lacklustre.net/bluetooth/bluetooth_smart_good_bad_ugly_fix-mikeryan-blackhat_2013.pdf, 2013.

[53] B. Fouladi, *Honey, I'm Home!! Hacking Z-Wave Home Automation Systems*, https://cybergibbons.com/wp-content/uploads/2014/11/honeyimhome-131001042426-phpapp01.pdf, 2013.

[54] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt, "On the security of rc4 in tls and wpa," in *Proc. 22th USENIX Sec.*, Washington, DC, Aug. 2013.

[55] J. Selvi, *Bypassing HTTP Strict Transport Security*, https://www.blackhat.com/docs/eu-14/materials/eu-14-Selvi-Bypassing-HTTP-Strict-Transport-Security-wp.pdf, 2014.

[56] B. Möller, T. Duong, and K. Kotowicz, "This POODLE Bites: Exploiting The SSL 3.0 Fallback," Google, Tech. Rep., 2014.

[57] US-CERT/NIST, *CVE-2015-0204*, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0204, 2015.

[58] US-CERT/NIST, *CVE-2012-4929*, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-4929, 2015.

[59] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue, "A messy state of the union: Taming the composite state machines of tls," in *Proc. 36th IEEE S&P*, San Jose, CA, May 2015.

[60] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proc. 22nd ACM CCS*, Denver, Colorado, Oct. 2015.

[61] T. Zillner and S. Strobl, *Zigbee Exploited: The good, the bad and the ugly*, https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf, 2015.

[62] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, E. Käsper, S. Cohney, S. Engels, C. Paar, and Y. Shavitt, "DROWN: Breaking TLS using SSLv2," in *Proc. 25th USENIX Sec.*, Austin, TX, Aug. 2016.

[63] S. Jasek, *GATTacking Bluetooth Smart devices*, http://gattack.io/whitepaper.pdf, 2016.

[64] P. Kintis, Y. Nadji, D. Dagon, M. Farrell, and M. Antonakakis, "Understanding the privacy implications of ecs," in *Proc. DIMVA*, 2016.

[65] N. Apthorpe, D. Reisman, and N. Feamster, "Closing the blinds: Four strategies for protecting smart home privacy from network observers," in *ConPro*, 2017.

[66] D. Wood, N. Apthorpe, and N. Feamster, "Cleartext data transmissions in consumer iot medical devices," in *IoT S&P*, 2017.

[67] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, "Survivable key compromise in software update systems," in *Proc. 24th ACM CCS*, Dallas, TX, Oct. 2017.

[68] Y. Acar, M. Backes, S. Bugiel, S. Fahl, P. McDaniel, and M. Smith, "Sok: Lessons learned from android security research for appified software platforms," in *Proc. 37th IEEE S&P*, San Jose, CA, May 2016.

[69] C. Zuo, W. Wang, Z. Lin, and R. Wang, "Automatic forgery of cryptographically consistent messages to identify security vulnerabilities in mobile services.," in *Proc. 2016 NDSS*, San Diego, CA, Feb. 2016.

[70] *About - IFTTT*, https://ifttt.com/about, 2018.

[71] *Work Super Smart - Automate.io*, https://automate.io, 2018.

[72] *Cloud Business App Integration*, https://cloudwork.com, 2018.

[73] M. Riley, A. Sharpe, and J. Robertson, *Equifax Suffered a Hack Almost Five Months Earlier Than the Date It Disclosed*, https://www.bloomberg.com/news/articles/2017-09-18/equifax-is-said-to-suffer-a-hack-earlier-than-the-date-disclosed, 2017.

[74] G. De Vynck, *Orbitz Hack May Have Compromised 880,000 Credit Cards*, https://www.bloomberg.com/news/articles/2018-03-20/expedia-s-orbitz-hack-may-have-compromised-880-000-credit-cards, 2018.

[75] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," in *DAT*, 2016.

[76] J. Novet, *Amazon scrambles to fix cloud networking issue affecting companies like Atlassian, Twilio*, https://www.cnbc.com/2018/03/02/amazon-cloud-networking-outage-affecting-atlassian-twilio-slack.html, 2018.

[77] N. Garun, *Yahoo says all 3 billion user accounts were impacted by 2013 security breach*, https://www.theverge.com/2017/10/3/16414306/yahoo-security-data-breach-3-billion-verizon, 2017.

[78] S. Moss, *Major ddos attack on dyn disrupts aws, twitter, spotify and more*, https://www.theverge.com/2017/10/3/16414306/yahoo-security-data-breach-3-billion-verizon, 2016.

[79] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and A. J. Halderman, "A search engine backed by internet-wide scanning," in *Proc. 22nd ACM CCS*, Denver, Colorado, Oct. 2015.

[80] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. 17th IEEE S&P*, Oakland, CA, May 1996.

[81] *AWS IoT Core*, https://aws.amazon.com/iot-core/, 2018.

[82] *IoT Hub Connect, monitor, and manage billions of IoT assets*, https://cloud.google.com/solutions/iot/, 2018.

[83] *GOOGLE CLOUD IOT: Intelligent IoT platform that unlocks business insights from your global device network*, https://cloud.google.com/solutions/iot/, 2018.

[84] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari, *Client subnet in dns queries*, http://www.ietf.org/rfc/rfc7871.txt, 2016.

[85] A. Bellissimo, J. Burgess, and K. Fu, "Secure software updates: Disappointments and new challenges.," in *HotSec*, 2006.

[86] CERT/CC, *Vulnerability note vu#361684*, https://www.kb.cert.org/vuls/id/361684, 2015.

[87] GNUcitizen, *Hacking the interwebs*, http://www.gnucitizen.org/blog/hacking-the-interwebs, 2008.

[88] HD Moore, "Security Flaws in Universal Plug and Play," Tech. Rep., 2013.

[89] Bluetooth SIG, *Bluetooth Low Energy - Bluetooth Technology Website*, https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy, 2016.

[90] Alliance, Zigbee and others, *Zigbee Specification*, 2006.

[91] Z-Wave Alliance, *About Z-Wave Technology*, http://z-wavealliance.org/about_z-wave_technology, 2016.

[92] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proc. 11th ACM ASIACCS*, Xi'an, China, May 2016.

[93] *Zigbee "insecure rejoin" faq*, https://support.smartthings.com/hc/en-us/articles/208201243-ZigBee-Insecure-Rejoin-FAQ, 2018.

[94] Z-Wave Alliance, *Z-Wave Transport-Encapsulation Command Class Specification*, http://zwavepublic.com/sites/default/files/command_class_specs_2017A/SDS13783-5Z-WaveTransport-EncapsulationCommandClassSpecification.pdf, 2017.

[95] Zigbee Alliance, *Zigbee: Securing the Wireless IoT*, http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/, 2015.

[96] J. Clark and P. C. van Oorschot, "Sok: Ssl and https: Revisiting past challenges and evaluating certificate trust model enhancements," in *Proc. 34th IEEE S&P*, San Francisco, CA, May 2013.

[97] tenable, *Nessus Professional*, http://info.tenable.com/rs/934-XQB-568/images/NessusPro__DS__EN_v8.pdf, 2005.

[98] MITRE, *About CVE*, http://cve.mitre.org/about/index.html, 1999.

[99] FIRST, *Common Vulnerability Scoring System SIG*, https://www.first.org/cvss/, 2005.

[100] A. Abraham, *Mobile Security Framework (MobSF)*, https://github.com/MobSF/Mobile-Security-Framework-MobSF/blob/master/README.md, 2016.

[101] Linkedin, *QARK - Quick Android Review Kit*, https://github.com/linkedin/qark/blob/master/README.md, 2016.

[102] *Kryptowire EMM+S*, http://www.kryptowire.com/enterprise.php, 2011.

[103] ntop, *High-Speed Web-based Traffic Analysis and Flow Collection*, https://www.ntop.org/products/traffic-analysis/ntop/, 1998.

[104] G. Combs, *About Wireshark*, https://www.wireshark.org, 1998.

[105] D. Roethlisberger, *SSLsplit - transparent SSL/TLS interception*, https://www.roe.ch/SSLsplit, 2009.

[106] *RASPBERRY PI ZERO*, https://www.raspberrypi.org/products/raspberry-pi-zero/, 2018.

[107] *THE JUNE OVEN*, https://juneoven.com/the-oven, 2018.

[108] *About Let's Encrypt*, https://letsencrypt.org/about/, 2018.

APPENDIX A
EVALUATION TABLES

TABLE III: An overview of the devices used in the evaluation.

Device	Category	Hub	Cloud Endpoints	Mobile Application		Communication	
				iOS	Android	IP	Low-Energy
Belkin WeMo Crockpot	Appliance		27	✓	✓	✓	
Roomba	Appliance		11	✓	✓	✓	
Belkin Netcam	Camera		79	✓	✓	✓	
Canary	Camera		22	✓	✓	✓	✓
Chinese Webcam	Camera		1	—	—	✓	
D-Link DCS5009L	Camera		4	✓	✓	✓	
Logi Circle	Camera		341	✓	✓	✓	✓
Nest Cam IQ	Camera		9	✓	✓	✓	✓
Nest Camera	Camera		7	✓	✓	✓	✓
Netgear Arlo	Camera		59	✓	✓	✓	
Piper NV	Camera		42	✓	✓	✓	✓
Withings Home	Camera		20	✓	✓	✓	✓
Amazon Echo	Home Assistant		221	✓	✓	✓	
Apple HomePod	Home Assistant		221	—	—	✓	✓
Google Home	Home Assistant		42	✓	✓	✓	
Google Home Mini	Home Assistant		221	✓	✓	✓	
Harmon Kardon Invoke	Home Assistant		128	✓	✓	✓	✓
August Doorbell	Home Automation		221	✓	✓	✓	✓
Belkin WeMo Link	Home Automation	✓	14	✓	✓	✓	✓
Belkin WeMo Motion	Home Automation		221	✓	✓	✓	
Belkin WeMo Switch	Home Automation		29	✓	✓	✓	
Caseta Hub	Home Automation	✓	221	✓	✓	✓	✓
Chamberlain myQ Garage Opener	Home Automation	✓	1	✓	✓	✓	✓
Insteon Hub	Home Automation	✓	20	✓	✓	✓	✓
Koogeek Lightbulb	Home Automation		1	✓	✓	✓	
LIFX Virtual Bulb	Home Automation		3	✓	✓	✓	
MiCasaVerde VeraLite	Home Automation	✓	74	✓	✓	✓	
Nest Guard	Home Automation	✓	14	✓	✓	✓	✓
Philips HUE	Home Automation	✓	27	✓	✓	✓	✓
Ring Doorbell	Home Automation		9	✓	✓	✓	
Samsung SmartThings	Home Automation	✓	10	✓	✓	✓	✓
TP-Link Wifi Bulb	Home Automation		11	✓	✓	✓	
TP-Link Wifi Plug	Home Automation		11	✓	✓	✓	
Wink 2	Home Automation	✓	12	✓	✓	✓	✓
Amazon Fire TV	Media		174	✓	✓	✓	
Apple TV (4th Gen)	Media		439	✓		✓	
Bose SoundTouch 10	Media		26	✓	✓	✓	✓
Logitech Harmony	Media		17	✓	✓	✓	✓
nVidia Shield	Media		261	—	—	✓	
Roku 4	Media		231	✓	✓	✓	
Roku TV	Media		226	✓	✓	✓	
Samsung SmartTV	Media		182	✓	✓	✓	
Sonos	Media		65	✓	✓	✓	✓
Google OnHub	Network		24	✓	✓	✓	
Securifi	Network	✓	938	✓	✓	✓	✓

TABLE IV: Mobile Application Evaluation.

Device	Mobile Application			Over-privileged	Sensitive Data	Crypto Issues
	Name	Platform	Version			
Securifi Almond	com.securifi.almond	iOS	3.5.6	✓		✓
LIFX Virtual Bulb	com.lifx.lifx	iOS	3.8.6	✓		✓
Ring Doorbell	com.ring	iOS	4.1.13	✓		✓
Roku TV Roku 4	com.roku.ios.roku	iOS	4.2.3	✓		✓
Netgear Arlo Camera	com.netgear.arlo	iOS	2.4.8	✓	✓	✓
TP-Link WiFi Plug TP-Link WiFi Bulb	com.tplink.kasa-ios	iOS	1.11.1	✓		✓
Chamberlain myQ Garage Opener	com.chamberlain.myq.chamberlain	iOS	6216.0.0			✓
Google Home Mini Google Home	com.google.Chromecast	iOS	1.28.508	✓		✓
Apple HomePod	—	iOS	—	—	—	—
Wink 2	com.quirky.wink	iOS	6.8.0	✓		✓
Google OnHub	com.google.android.apps.access.wifi.consumer	Android	jetstream BV10127	✓		✓
Samsung SmartThings	com.smartthings.android	Android	2.13.0	✓	✓	✓
Philips HUE	com.philips.lighting.hue2	Android	2.19.0	✓	✓	
Insteon Hub	com.insteon.insteon3	Android	1.9.8	✓	✓	
Sonos	com.sonos.acr	Android	8.3.1		✓	
Nest Camera Nest Cam IQ Nest Guard	com.nest.android	Android	5.17.0.31	✓	✓	✓
Belkin WeMo Motion Belkin WeMo Switch Belkin WeMo Link Belkin WeMo Crockpot	com.belkin.wemoandroid	Android	1.19.0		✓	
Amazon Echo	com.amazon.dee.app	Android	2.2.1615.0	✓	✓	
Belkin Netcam	com.belkin.android.android belkinnetcam	Android	2.0.4		✓	
Amazon Fire TV	com.amazon.storm.lightning.client.aosp	Android	1.0.13.18	✓	✓	
D-Link DCS5009L	com.dlink.mydlinkunified	Android	1.0.3	✓		✓
Logitech Logi Circle	com.logitech.circle	Android	2.3.2220	✓		
Canary	is.yranac.canary	Android	2.14.0	—	—	—
Piper NV	com.blacksumac.piper	Android	1.4.0	—	—	—
Withings Home	com.withings.home	Android	1.5.3	—	—	—
MiCasaVerde VeraLite	com.vera.android	Android	7.25.47	✓		
August Doorbell Cam	com.august.luna	Android	6.1.4	✓	✓	
Logitech Harmony	com.logitech.harmonyhub	Android	5.1.1	✓		
Caseta Wireless	com.lutron.mmw	Android	5.1.0	✓		
Bose SoundTouch 10	com.bose.soundtouch	Android	17.170.82	✓	✓	✓
Harmon Kardon Invoke	com.microsoft.cortana	Android	2.10.2.2135		✓	
Roomba	com.irobot.home	Android	2.3.1	✓	✓	✓
Samsung SmartTV	com.samsung.smartviewad	Android	2.1.0.100			✓
Koogeek Lightbulb	com.tomtop.home	Android	1.2.2	✓	✓	✓
nVidia Shield	—	—	—	—	—	—
Chinese Webcam	—	—	—	—	—	—

TABLE V: Cloud Endpoint Evaluation.

Device	Domains						SSL			Services			
	Total	1st Party	3rd Party	Hybrid	Unknown	Host	Self-Signed	Domain Mismatch	Vuln SSL	Outdated OS	Information Discolsure	Cleartext Auth	Exploitable Service
Amazon Echo	221	15	191	3	12	17	✓		✓	—	—	—	—
Amazon Fire TV	174	100	17	14	43	99	—	—	—				✓
Apple HomePod	182	80	6	76	20	113	✓	✓	✓	—	—	—	—
Apple TV (4th Gen)	439	170	14	188	67	38	✓		✓	—	—	—	—
August Doorbell	55	7	12	34	2	32	✓		✓			✓	
Belkin Netcam	79	13	63	1	2	12			✓		✓	✓	✓
Belkin WeMo Crockpot	27	7	15	5	0	11			✓	✓	✓		✓
Belkin WeMo Link	14	4	6	4	0	11	—	—	—	✓			✓
Belkin WeMo Motion	24	7	12	5	0	9	—	—	—	—	—	—	—
Belkin WeMo Switch	29	5	19	5	0	10			✓	✓			✓
Bose SoundTouch10	26	10	10	6	0	11			✓		✓	✓	
Canary	22	19	3	0	0	9	✓			—	—	—	—
Caseta Wireless	22	2	11	5	4	6	—	—	—	—	—	—	—
Chamberlain myQ Garage Opener	1	1	0	0	0	1	✓			—	—	—	—
Chinese Webcam	1	1	0	0	0	1	—	—	—			✓	
D-Link DCS5009L	4	4	0	0	0	3			✓	—	—	—	—
Google Home	42	29	3	0	10	14	—	—	—	—	—	—	—
Google Home Mini	40	27	3	0	10	17	—	—	—	—	—	—	—
Google OnHub	24	24	0	0	0	15	—	—	—	—	—	—	—
Harmon Kardon Invoke	128	0	108	5	15	9	✓		✓		✓		
Insteon Hub	20	2	12	5	1	5	✓				✓		
Koogeek Lightbulb	1	0	1	0	0	0	—	—	—	—	—	—	—
LIFX Virtual Bulb	3	2	1	0	0	1	✓		✓	—	—	—	—
Logitech Harmony	17	6	5	6	0	8	—	—	—	—	—	—	—
Logitech Logi Circle	341	158	5	178	0	20	✓						✓
MiCasaVerde VeraLite	74	1	30	43	0	40	✓		✓	✓	✓	✓	✓
Nest Cam IQ	9	4	5	0	0	4	✓			—	—	—	—
Nest Camera	7	6	1	0	0	5	✓		✓	—	—	—	—
Nest Guard	14	6	6	2	0	4	✓				✓		
Netgear Arlo	59	23	2	7	27	18	—	—	—				✓
nVidia Shield	261	23	177	3	58	24	—	—	—	—	—	—	—
Philips HUE	27	14	8	0	5	11	—	—	—	—	—	—	—
Piper NV	42	24	16	2	0	16	✓		✓		✓		
Ring Doorbell	9	5	3	1	0	6	—	—	—	—	—	—	—
Roku 4	231	37	177	4	13	28	✓		✓		✓		
Roku TV	226	36	144	6	40	28	✓		✓		✓	✓	
Roomba	11	2	5	4	0	5			✓	—	—	—	—
Samsung SmartThings	10	6	1	3	0	4	✓		✓	—	—	—	—
Samsung SmartTV	182	27	138	2	15	20			✓	—	—	—	—
Securifi Almond	938	9	0	0	929	6			✓	—	—	—	—
Sonos	65	13	34	7	11	1	—	—	—	—	—	—	—
TP-Link WiFi Bulb	11	3	7	1	0	4	—	—	—	—	—	—	—
TP-Link WiFi Plug	11	3	7	1	0	4	—	—	—	—	—	—	—
Wink 2	12	3	7	2	0	4	—	—	—	—	—	—	—
Withings Home	20	12	2	2	4	9	—	—	—			✓	

Device	System Services			System Setup		
	Detected OS	Running Services	Issues Found	Pairing	Config.	Upgrade
Insteon Hub	Linux 2.6	4	6	Wired+Pin	F	C
MiCasaVerde VeraLite		4	6	Cloud+Pin	D	M
Wink 2	Linux 2.6	4	4	Wired	D	M
Sonos	Linux	3	3	Wired	D	C
nVidia Shield	Linux 2.6	2	3	Wired	D	M
Google Home	Linux 3.3	5	2	Wifi	F	A
Google Home Mini	Linux 3.3	5	2	Wifi	F	A
D-Link DCS5009L		3	2	Wired	D	M
Harmon Kardon Invoke	Linux 3.3	5	1	LE	F	A
Bose SoundTouch 10	Linux 2.6	4	1	LE	F	C
Chinese Webcam	Linux 2.6	4	1	Wired+HTTP	D	N/A
Samsung SmartTV	Linux 4.8	4	1	On-Screen	D	M
Logitech Harmony		2	1	LE	F	M
Securifi Almond	Linux 2.6	2	1	Wired	D	M
Belkin Netcam	Linux 2.6	1	1	Wifi+Pin	F	C
Belkin WeMo Link		1	1	Wifi+Pin	F	C
Belkin WeMo Motion		1	1	Wifi+Pin	F	C
Belkin WeMo Switch		1	1	Wifi+Pin	F	C
Samsung SmartThings	Linux 2.6	1	1	Wired	F	C
Apple HomePod	FreeBSD 6	4	0	LE	F	C
Apple TV (4th Gen)	tvOS	3	0	Wired	F	C
Piper NV		3	0	Wifi	F	A
Caseta Wireless	Linux 2.6	2	0	Wired	F	M
Koogeek Lightbulb		2	0	LE+Pin	D	C
Philips Hue	Linux 2.6	2	0	Wired+Button	F	C
Roku 4	Linux 3.3	2	0	Wired	D	M
Roku TV		2	0	Wired	D	M
Amazon Echo	Linux	1	0	Wifi	F	A
Amazon Fire TV	Linux 2.6	1	0	Wired	D	M
August Doorbell	Linux 2.6	1	0	Wifi	F	M
Chamberlain myQ Garage Opener		1	0	Wifi	F	M
Google OnHub	Linux 4.8	1	0	Wired	F	A
Roomba		1	0	Wifi	F	M
TP-Link Wifi Bulb		1	0	Wifi	D	M
Withings Home		1	0	LE	F	C
Canary		0	0	Wifi	F	C
LIFX Bulb		0	0	Wifi	D	M
Logi Circle		0	0	LE	F	A
Nest Cam IQ		0	0	Wired	F	A
Nest Camera		0	0	LE+Pin	F	A
Nest Guard		0	0	Wired	F	A
Netgear Arlo		0	0	Wired	F	M
Ring Doorbell		0	0	Wifi	F	M
TP-Link Wifi Plug		0	0	Wifi	F	M
Belkin WeMo Crockpot		0	0	Wifi+Pin	D	C

(**F**)orced configuration change when device is setup; (**D**)efault device configuration is acceptable and allows device to operate. (**C**)onsent by the user is required for the device to upgrade; (**A**)utomatic updates are applied without user intervention; (**M**)anual device update via user request. N/A means the category is not applicable.

TABLE VI: Device Evaluation.

Device	Observed IP Communication					MITM			Encryption		
	DNS	HTTP	UPnP	NTP	Custom	D-C	A-C	A-D	D-C	A-C	A-D
Google OnHub	✓-	✓+			✓	✗	✗	—	◐	●	—
Samsung SmartThings	✓	✓+	✓			✗	✗	✗	●	●	—
Philips HUE	✓	✓+	✓	✓		✗	✗	✓	●	●	c
Insteon Hub	✓	✓		✓		✓	—	—	○	—	—
Sonos	✓	✓+	✓		✓	✗	✓	✓	◐	○	○
Securifi Almond	✓			✓	✓	✗	✗	—	◐	●	—
Wink 2 Hub	✓	✓+	✓	✓		✓	✗	✓	●	◐	○
Belkin WeMo Motion Belkin WeMo Switch Belkin WeMo Link Belkin WeMo Crockpot	✓	✓+	✓	✓	✓	✗	✗	✓	◐	◐	○
LIFX Bulb	✓			✓	✓	✗	✗	✓	●	◐	○
Amazon Echo	✓	✓	✓	✓	✓	✗	✗	—	●	●	—
Belkin Netcam	✓	✓+	✓		✓	✗	✗	✓		◐	—
Ring Doorbell	✓				✓	✗	✗	—	●	●	—
Roku TV Roku 4	✓	✓+	✓		✓	✗	—	✓	●	—	○
Amazon Fire TV	✓	✓+	✓			✗	—	✗	●	—	◐
nVidia Shield	✓	✓+			✓	✗	—	—	◐	—	—
Apple TV (4th Gen)	✓	✓+		✓	✓	✗	—	✗	●	—	●
Netgear Arlo	✓	✓+		✓		✗	✗	—	●	●	—
D-Link DCS-5009L			✓			—	—	✓	—	—	○
Logi Circle	✓	✓+		✓		✗	✗	—	●	●	—
Canary	✓	✓+				✗	✗	—	●	●	—
Piper NV	✓-	✓+		✓	✓+	✗	✗	—	●	●	—
Withings Home	✓	✓+			✓	✗	✗	✗	●	●	●
MiCasaVerde VeraLite	✓	✓+	✓	✓	✓	✗	✓	✓	◐	●	○
Chinese Webcam		✓	✓		✓	✓	—	✓	○	—	—
August Doorbell	✓	✓+				✓	✗	✓	◐	◐	○
TP-Link WiFi Plug TP-Link WiFi Bulb	✓	✓+		✓		✗	✗	✗	●	●	●
Chamberlain myQ Garage Opener	✓				✓	✗	✗	—	●	●	—
Logitech Harmony	✓	✓+	✓			✗	✗	—	◐	◐	—
Caseta Wireless	✓	✓		✓	✓	✗	✗	✓	◐	●	○
Google Home Mini Google Home	✓-	✓+	✓	✓	✓	✗	✗	✓	◐	●	○
Bose SoundTouch 10	✓	✓+	✓		✓	✗	✗	✓	●	◐	○
Harmon Kardon Invoke	✓-	✓+		✓		✗	✗	—	●	●	—
Apple HomePod	✓	✓+		✓	✓	✗	—	—	●	—	—
Roomba	✓	✓+		✓	✓	✗	✗	✗	●	●	○
Samsung SmartTV	✓	✓+	✓			✗	—	✓	◐	—	◐
Koogeek Lightbulb	✓			✓	✓	—	✗	✗	—	●	●
Nest Camera		✓+				✗	✗	—	●	●	—
Nest Cam IQ	✓-	✓+				✗	✗	—	●	●	—
Nest Guard				✓	✓	✗	✗	—	●	●	—

TABLE VII: Communication Evaluation.
✓+ (TLS/SSL) — ✓- (3rd-party recursive DNS)

APPENDIX B
EVALUATION CASES

Our evaluation shows that some devices have a better security posture than others. In this section, we take a look at three devices that we categorize based on their overall security evaluation. We propose three categories: *good*, *satisfactory*, and *needs improvement*, which highlight good security practices and short-comings.

A. Good: Withings Home

Functional Features. The Withings Home device is a camera paired with an air quality sensor. The device has a mobile companion application, integrates with cloud endpoints, and communicates over the Internet and the local network. The device exposes mDNS service, which allows zero-configuration protocols to find and configure the device (i.e. Apple's Bonjour). The device uses a low-energy protocol, Bluetooth, to configure the device initially, then switches to IP communication. Device updates are not applied automatically but require user consent.

Assessment. We found no issues with the *mDNS* service running on the device. The companion mobile application correctly utilizes secure storage facilities to store sensitive data, correctly uses cryptographic protocols, and has proper permission provisioning. The majority of the cloud infrastructure is self-hosted by Nokia and runs services to enable user notifications and control. The network communication between device-to-cloud, app-to-cloud, and app-to-device uses full encryption and is not susceptible to MITM attacks. The device did authenticate in clear-text (an insecure practice) across the Internet to associate the device with the cloud management interface that runs an XMPP server [4].

B. Satisfactory: Nest Cam

Functional Features. The Nest Cam is an indoor camera that senses motion, records video, and notifies users of activities. The device uses forced configuration, which means users have to configure and set up their device before it can operate. The camera uses the Bluetooth protocol to configure the device via the mobile application, which pairs using a pin/barcode located on the back of the camera. The camera does not utilize the local network to control the device, all of the activities and controls operate through the cloud endpoints. Finally, updates to the device are applied automatically with no user consent, ensuring the device always has the latest running firmware.

Assessment. The Nest Cam does not expose any services but uses a client model, where the device acts as a client that communicates directly with the cloud endpoints. The lack of exposed services running on the Nest Cam considerably shrinks the attack vector and limits an IP-based attacker. The Nest Cam uses certificate pinning on the device, which verifies and validates the device to cloud communication is secure. The device setup and configuration requires mobile application pairing via Bluetooth, which both ensures proximity of end-user and limits remote attack vectors. The mobile application manages all Nest products, including the Nest Cam, which requests access to the microphone, camera/photos, geolocation, and other sensitive services. The cloud endpoints fully manage the Nest Cam, which means without Internet access the device is inaccessible. The Nest products, in general, forcibly use the Google DNS recursive and ignore the DHCP configurations on the local network. A savvy user can configure static routes on their gateway to redirect DNS traffic toward their desired resolver.

[4]endpoint located at: xmpp.withings.net:5222

TABLE VIII: List of devices and their CVEs with CVSS score of Critical and High.

Device	CVE	CVSS
MiCasa Verda VeraLite	CVE-2012-5958, CVE-2012-5959, CVE-2012-5960, CVE-2012-5961, CVE-2012-5962, CVE-2012-5963, CVE-2012-5964, CVE-2012-5965, CVE-2013-4863	Critical
	CVE-2012-0920	High
Wink 2	CVE-2016-7406, CVE-2016-7407	Critical
	CVE-2016-7408	High

C. Needs Improvment: MiCasa Verde VeraLite

Functional Features. The VeraLite is a smart-home Z-Wave enabled controller that can monitor and control low-energy sensors and other devices around the home. The device pairs through a cloud portal using a pre-printed pin on the back of the device. The VeraLite requires manual updates, but the device notifies users of the availability of new updates. The device exposes four services including a web, DNS, UPnP, and SSH server. The mobile device requests excessive permissions like calling, controlling phone network state (on/off/airplane mode), and access to the camera. The VeraLite is a discontinued product and no longer offered by the vendor.

Assessment. The VeraLite device provides a hardened setting that disables many of the running services on the device, but they are on by default. The hardened mode forces the device management and monitoring from cloud endpoints. The device has several exploitable vulnerabilities as illustrated by Table VIII. The UPnP services use a vulnerable version of the *libupnp* library, and the SSH services use a vulnerable *dropbear* (2016.72) implementation.

The configuration of the SSH server supports Cipher-Block-Chaining (CBC) mode, specifically *3des-cbc*, *aes128-cbc*, and *aes256-cbc*, which an attacker can exploit to recover the plaintext from the ciphertext. The DNS service is configured to allow queries for third-party domains that do not have the recursion bit set; hence allowing attackers to snoop on the DNS cache. The mobile application requires users to establish an account with the Vera vendor, which allows end-users to manage their controller. The device does not use certificate pinning, which leaves the deployment susceptible to MITM attacks. The cloud endpoints use clear-text authentication, run exploitable services, expose sensitive information, and run unsupported operating systems.