Consumer IoT Security and Comfort Product Case Studies

Jonathan Derenge, *Dakota State University*, Jonny.Derenge@trojans.dsu.edu

***Abstract*—The market for household IoT has substantially grown in recent years [22] [12]. Advances in chip technology as well as adoption of the Web of Things framework have enabled manufactures to create a growing variety of products with diverse capabilities [5]. This paper examines three household smart appliances. These include the WiFi Table Top Heater model AT1482 by Atomi, The Onelink Safe and Sound model 1039102, a smoke and carbon monoxide detector, and the PTZ ZS-GQ4 outdoor security camera by IoGeek. A testing environment was created consisting of all these devices connected to one access point—a Hak5 Wi-Fi Pineapple (Mark VII model). The methodology for testing was black box. Reconnaissance techniques revealed Multiple attack vectors and attack surfaces to investigate. Upon the conclusion of the research, the vendors will receive a copy of the final report highlighting and security issues affecting their product.**

***Index Terms*—IEEE, IEEEtran, journal,**

I. Introduction

IN April of 2022, Instron, a company specializing in smart-lighting technologies, abruptly stopped maintaining their services without warning. As reported by the Wired Magazine, smart household gadgets used by Instron's 1.3 million customers had lost connectivity with the company's servers. As a result, users could no longer control their lights remotely. Although some of the light switch models still functioned as traditional switches, the ones which have no override no longer functioned at all [1]. While similar problems occur from time to time, what stands out about this case was the lack of notice. As Blake Kozak, a consultant from consulting firm commented "It has happened before, but this was next level." [1]. To date Instron has declined to comment. This raises an important question about the risk associated with allowing third party control of basic appliance function. The demand for convenient and affordable technology designed to make life easier is increasing drastically, as the market continues to produce various products with even more capabilities [2].

While Insteon's case was referred to as an avoidable "black eye" it will certainly not be the last debacle in the market [1]. Industry players face challenges, including inadequate security protections, limited consumer demand, market fragmentation, to name a few [5]. Despite this, IoT is expected to grow at an alarming rate [21]. Despite the current trends, it should be noted that IoT is still in infancy [15]. Due to the many competing standards competing across several platforms, interoperability remains a challenge [16] [15]. The majority IoT devices today still use proprietary vertical technology stack technology, designed to communicate with a central server (see Fig.5) [14]. There is not a predominant voice among handful of industry leaders, and no overarching governing body for how these devices ought to be designed and what standards are acceptable [15].

A. Background

The term, Internet of Things (IoT), first coined in 1999, by Kevin Aston, to describe a system where the internet interacts with the physical world through various sensors. Aston's work with standardizing RFID highlights the utility of extending such sensors over the internet [3]. Since then, IoT includes a vast array of modern technologies which operate on various wireless standards for exchanging information. When it comes to Smart-Home technologies, there are two competing architectures. They include centralized and autonomous. The Centralized model relies on an outside processing device, typically in the cloud, which are responsible for many of the devices functions [4]. Whereas autonomously designed IoTs are capable of managing their own processes [17]. One advantage of the central model is lower processing power of the IoT device, which translates to less power consumption for the end user. On the other hand, as demonstrated by the Instron incident, reliance on a third party services implies it's own set of risks.

The majority of and IoT devices today are designed with this central model [15]. This architecture utilizes proprietary vertical stack technology designed to communicate with their respective platform infrastructure (see Fig. 4) [15]. Because of the proprietary nature of the design, devices from one platform cannot easily communicate with devices from other vendors [15]. Enter the Web of Things (WoT) framework. The WoT is a project organized by the World Wide Web Consortium (W3C) which aims to make IoT ecosystem more interoperable. They provide guidelines and standards for Web-based IoT applications, which aims at facilitating a more open and secure internet [15] [17].

Importantly, WoT standardized the following: Thing descriptors, scripting APIs, and guidelines for security and privacy Thing descriptors are used to define a devices' identifying characteristics and functionality. A scripting API is an API that not is responsible for extending functionality of the device through scripts. Common APIs, such as web servers, use get, post, put and delete to update resources. Scripting APIs work work the same, only instead of a traditional Web interface to handle requests, requests are sent by the device itself, or by user using the mobile application. The receiving server, known as a publisher, performs the calculation. then sends the result to back to the end-device, known as the subscriber [17]. This way

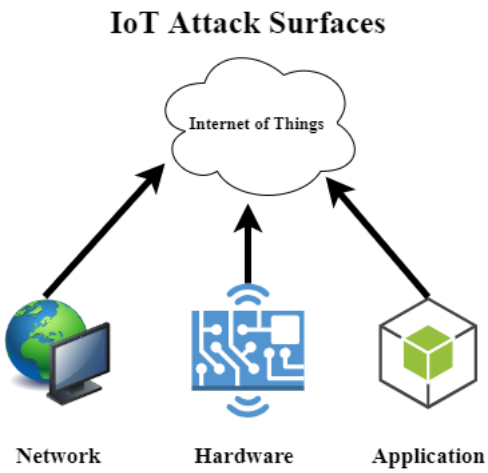

Fig. 1. Internet of Things Attack Surfaces

devices can outsource processing threads and management interfaces to a variety of platforms. WoT endorses using Restful API, which maximizes interoperability across platforms and on the application level [16]. This affords smart-home devices compatibility with additional features from different platforms i.e. digital assistants (DAs) from both Amazon and Google. WoT also recommends the current best practices for security and privacy [14] [15].

B. Methodology

The devices observed consisted of a Onelink Safe and Sound three in one Smoke and Carbon Monoxide Alarm, model 101, the Atomi Smart Table-top Heater, and PTZ Security Camera model ZS-GQ4. All of which had 802.11 compatibility. Specific standards were not mentioned in the Atomi product packaging or user documentation, but did state that it could not connect to the 5 GHz band, implying specifically 802.11b/g/n compatibility. The Onelink additionally advertised an embedded Bluetooth speaker, and Amazon Alexa capabilities. Using a black-box testing methodology, a scheme of tests was created to enumerate the devices. To conduct these tests, a miniature home network was configured by connecting the devices to a separate WLAN, provisioned by a Wi-Fi Hak5 Pineapple Mark VII model. To provide Internet access, the pineapple was bridged to the larger Wireless testing network. Essential network reconnaissance included packet sniffing, port scanning, and protocol analysis. The Android applications used to interface with each device were also examined for vulnerabilities. Internal hardware-level investigation was performed only if it could be done non-destructively.

C. Methods

1) Network Reconnaissance: Nmap scans were initiated from a local computer connected to our pineapple (see figure) Due to the limited hardware and network capabilities of these devices, the scans used the -T1 option for the slowest scan speed. These identified numerous TCP ports running on the devices [see table]. For instance, Fig 2 represent a series of scans for detecting known vulnerabilities. By default Nmap this scans ports 1 through 1000.

a) Port Scanning: Some limitations of utilizing this technique for service detection are twofold. One, assuming proper firewall rules are configured, attempts to communicate with this port can be blocked, and second, services running on non-standard ports cannot be reliably identified.

Device	Nmap Command
PTZ Camera	nmap -oN ptz.txt --script vuln 172.16.42.228
Atomi Smart Heater	nmap -oN heater.txt --script vuln 172.16.42.245
First Alert Onelink	nmap -oN onelink.txt --script vuln 172.16.42.188

Fig. 2. Scanning for vulnerabilities

b) Service Detection: To identify services, packet inspection was utilized. Packet sniffing is commonly used to accomplish this by setting a Network interface card in Promiscuous mode, which allows the card to capture any traffic, irrespective of its assigned destination. The WiFi pineapple had a module available for dumping traffic with TCPdump. These files were then analyzed using Wireshark version 3.6.6.

c) Host Detection: Nmap offers was ineffective at detecting the host operating system using TCP/IP stack fingerprint. This is most likely because these device use a proprietary custom embedded Linux operation system. Therefore each device had to be classified manually, based on the characteristics of the stack, the Onelink and the Atomi were both running a modified version of a light-weight Linux distribution, likely OpenWRT. The PTZ OS fingerprint matched too many profiles to be classified accurately.

II. Application Security Testing

All three of the IoT appliances had respective mobile applications which allowed a user to interact with the device from their smartphone. The x86 version of Android 8.1 was used to run the applications on the testing host computer. The Internet connection was bridged to the virtual machine (VM). The Android Device Debugger (ADB) was then connected over TCP/IP. This allowed the client application to be installed onto the Android VM.

A. Static APK Analysis

As aforementioned, each device has a respective mobile application to grant functionality remotely. For Android devices, the application is the Android Package Kit (called APK for short) [10]. A examination of the APK files was performed on all three application. To maintain consistency, copies of the same APKs were later installed on the Android device. Each APK file contained the AndroidManifest.xml file. This indicated the permissions requested by the app. Much attention was paid to the intents and receivers listed in the file. Intents are simply the actions an application wants to perform. Notably, they don't necessarily have to dictate a specific process or application to perform the action [9].

Anything from generating a notification to taking a picture can be declared with intents. Receivers on the other hand, indicate locations where the application can retrieve data. This includes data obtained through an implied intent, as well as specific calls to other APIs [9]. This explained much of how the application could enhance it's own functionality by passing information to and from the various sources.

B. Proxy interception

Communication between the mobile phone and the central servers were intercepted forwarding traffic, a technique known as proxying. Most applications mitigate the risk associated of intercepted traffic by implementing TLS encryption. To defeat TLS, a certificate was generated on the intercepting computer using MiTMproxy, and then pushed into the /system/storage/security/cacerts directory. using ADB. The difficulty in this technique includes making the system volume read-write (rw). The option to make the system volume writable, was available when installing android, and subsequently booting into it. Another methods included using a Magisk module, which imports user certificates into the system automatically on reboot.

III. Physical Security Testing

A. Internal Atomi Anaylsis

The chassis of the Atomi device was dismantled by unscrewing six screws on the back panel. Four behind the removable ventilation panel, and two visible from the inside the concave handle. Then thirteen additional screws on the underbelly of the device. The first seven of which were were covered with rubber stoppers. The next two allowed the base panel to slide down the power cable, to allow the four remaining screws securing the back shell to be removed. Upon inspection of the inside, one could identify open sockets, labeled TYJW2S. Upon further reading, this was discovered that this WiFi interface was well documented, module made by a company called Tuya [23]. With enough time and the right tools, an attacker could reprogram this module entirely and even burn their own firmware to the card.

IV. Related Works

A. MQTT injection

Wong and Luo, Researchers from the University of Montana, evaluated the security of MQTT protocol. First, they wrote an MQTT parser in python. Using two raspberry pies, one for the subscriber and one for the publisher, a WiFi pineapple and an access point, where able to successfully inject malicious MQTT packets and measure how effectively the end nodes filtered malicious packets [11].

B. OWASP Top 10 IoT Vulnerabilities

In 2018, The Open-source foundation of application security, (OWASP) compiled a list of the top ten vulnerabilities related to IoT. Among the top five include: weak, guessable or hard-coded passwords, insecure network services, insecure ecosystem interfaces, lack of secure update mechanism, and use of insecure or outdated components [14].

C. Mirai Botnet

Mirai is small piece of malware, which identifies vulnerable IoT devices, using a small table of known factory credentials, it authenticates itself on any public facing devices and attempt to repudiate itself from new hosts. These new hosts make up the Miraibotnet, which has famously used to execute a number of DDoS attacks. Despite being around for over five years the malware remains prevalent due to the number of devices with insecure passwords [13].

V. Results

The Nmap and TCPdump traffic analysis revealed that OneLink and the Atomi Heater used MQTT protocol [port numbers]. This protocol is a used for lightweight messaging, so that IoT devices which do not require high bandwidth can communicate. There is no encryption built into this protocol, but the payloads are encrypted. Additionally, Nmap revealed several TCP-wrapped ports, which indicated each device had a host-based firewall configured. Using the MiTM tool allowed application data to be decrypted. Presently, all AWS cryptographic services are at minimum, FIPS 140-2 compliant [20]. This means, at a minimum, the communication (requests and responses) between the IoT device and the cloud service will be encrypted. In the case of the Onelink, these requests, known as invocations, required a signature validating the request. Meaning requests could not be forged without generating without also forging an authorization token. Replay attacks were possible, as long as the original request/response occurred before the signature of the original token expired. The expiration date, or time-to-live values of certain tokens varies between the devices. For instance, the PTZ camera's authentication token, which authenticates the the user to the account, was observed to have a TTL value of 604800 seconds, or seven days (see Fig. 3). Although the authentication methods varied between devices, all of them called upon a separate API, usually Google-based, to authenticate the user. One issue seen was that server replies often contained the user email. Given that the user email can generally uniquely identifier a single account, this is unsurprising. However, an attacker utilizing a a similar approach such as MiTM would only have to find the password to which corresponds to that email. An in-depth enumeration of platform vulnerabilities could not be conducted Due to lack of authorization.

Interestingly, none of the three IoT applications examined utilized certificate pinning. A technique wherein, a server only allows select certificates, generally from reputable Certificate Authority (CAs). This means that that a user would be unable to connect to the server. There are a multitude of reasons applications do not implement this protection. Necessary changes in configuration could render users unable to connect. Root detection is an essential security check for many applications. Only one of the three apps had a security notification, which notified the user that they were logged in as root. For critical application wanting to maximize obfuscation, and unexpected user behavior [18]

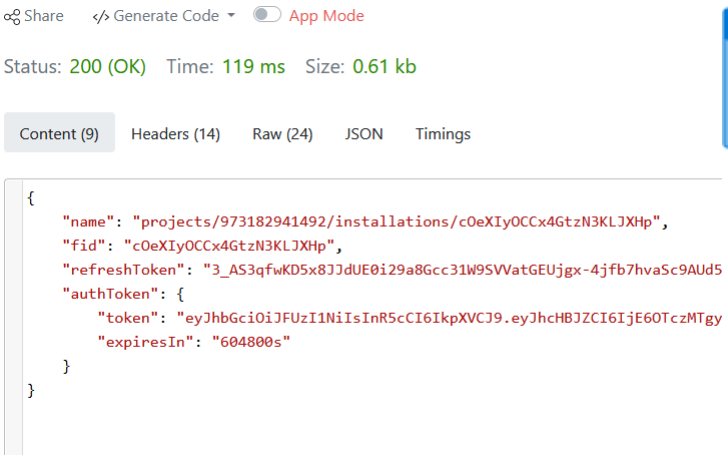

Fig. 3. Response to Authorization Request Using Curl (credit to ReqBin.com)

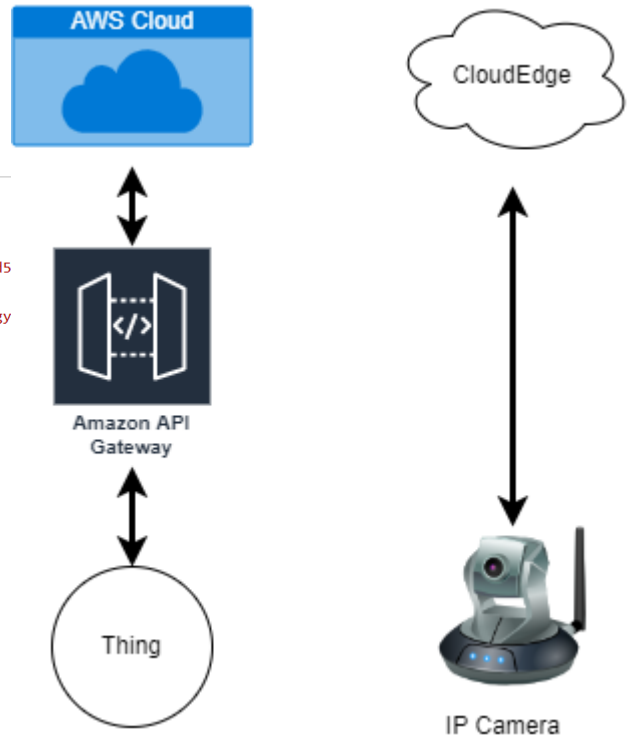

Fig. 4. IoT Gateway Models

Company	Google	Microsoft	Amazon	Apple	Samsung
Cloud Services	Nest Cloud/ Google Cloud	Azure IoT	AWS IoT	iCloud	ARTIK Cloud/ SmartThings
Application Protocols	Weave	AMQP	MQTT	HomeKit	MQTT
Network Protocols	Wifi/Thread	WiFi	WiFi	WiFi/BLE	Wifi/Zigbee/ BLE/Thread
Operating Systems	Linux/Android Things	Windows IoT	Linux/AWS Greengrass	iOS	Linux/ARTIK

Fig. 5. Proprietary IoT Stack Implementation

VI. User Risk Mitigation

A. 802.11 considerations

Use Strong Authentication and Encryption. Passwords should always be changed from the default, and should not be easily guessed or brute-forced. This includes passwords to log into gateway (router) management interfaces. While not always the case, generally devices on the home network have a private IP address. Advanced techniques are necessary if an outside attacker wanted to identify them outside the LAN [6]. Using WPA2 with Encryption makes deauthentication and ARP spoofing/redirection attacks more difficult as management frames are encrypted. Providing IoT devices with their own segment of the network. This is done by partitioning a part of the network address space exclusively for embedded smart systems. This increases privacy, and mitigates the risk to the rest of the network in case of unauthorized access [19].

B. Using TLS/SSL

TLS/SSL ensures that communication between two end points is confidential. Notably, when setting up MiTM proxy on different versions of android, the application data was encrypted using a different version of TLS. Currently the best standard for encrypting application traffic is TLS 1.3. Because this standard was released in after Android 8.1, it is not supported by default. One way a user can protect one's self from the vulnerabilities in TLS 1.2 is by configuring one's system the newer version. TLS 1.3 is supported naively on systems running Android 10 and above [8].

Appendix

IoT Gateway Models (see Figure 4.)

Acknowledgment

The author would like to thank the following individuals for their contributions and assistance in this project. Angel Gamboa, Aiden Schramm, Riley Basaran.

References

[1] B. Ashworth, "Smart Home Company Insteon Shut Down—and Left Customers in the Dark," Wired, Apr. 20, 2022. https://www.wired.com/story/insteon-shutdown/ (accessed Jul. 31, 2022).

[2] A. Jain, "Why smart homes are popular," The Hindu, Jun. 03, 2016. Accessed: Aug. 03, 2022. [Online]. Available: https://www.thehindu.com/features/homes-and-gardens/Why-smart-homes-are-popular/article14384601.ece

[3] K. Ashton, "That 'Internet of Things' Thing In the real world, things matter more than ideas," Jun. 2009. Accessed: Jul. 31, 2022. [Online]. Available: http://www.itrco.jp/libraries/RFIDjournal-That

[4] P. Pereria, "Edge computing vs. central IT: Where do your workloads belong?," HPE, Feb. 14, 2017. https://www.hpe.com/us/en/insights/articles/edge-vs-central-it-where-do-my-apps-and-services-belong-1702.html (accessed Jul. 31, 2022).

[5] H. Bauer, M. Patel, and J. Veira, "Internet of Things: Opportunities and challenges for semiconductor companies," McKinsey and Company, Feb. 18, 2015. https://www.mckinsey.com/industries/semiconductors/our-insights/internet-of-things-opportunities-and-challenges-for-semiconductor-companies (accessed Aug. 03, 2022).

[6] Y. Meidan, V. Sachidananda, H. Peng, R. Sagron, Y. Elovici, and A. Shabtai, "A novel approach for detecting vulnerable IoT devices connected behind a home NAT," Computers and Security, vol. 97, p. 101968, Oct. 2020, doi: 10.1016/j.cose.2020.101968.

[7] A. Faisal and M. Zulkernine, "A secure architecture for TCP/UDP-based cloud communications," International Journal of Information Security,

vol. 20, no. 2, pp. 161–179, Jul. 2020, doi: 10.1007/s10207-020-00511-w.

[8] "Behavior changes: all apps," Android Developers, Jun. 21, 2022. https://developer.android.com/about/versions/10/behavior-changes-all (accessed Aug. 04, 2022).

[9] "Common Intents," Android Developers, Jun. 21, 2022. https://developer.android.com/guide/components/intents-common

[10] N. Montegriffo, "What is an APK file and how to install APKs on Android?," NextPit, Jan. 17, 2022. https://www.nextpit.com/android-for-beginners-what-is-an-apk-file

[11] K. Storey, "Smart Houses and Smart Technology: Overview and Implications for Independent Living and Supported Living Services," Intellectual and Developmental Disabilities, vol. 48, no. 6, pp. 464–469, Dec. 2010, doi: 10.1352/1934-9556-48.6.464.

[12] H. Wong and T. Luo, "Man-in-the-Middle Attacks on MQTT-based IoT Using BERT Based Adversarial Message Generation," 2020.

[13] "What is the Mirai Botnet? — Cloudflare," Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/

[14] "OWASP top 10 list 2018," Owasp, Dec. 2018. https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf (accessed Aug. 04, 2022).

[15] B. Francis, "Building the Web of Things – Mozilla Hacks - the Web developer blog," Mozilla Hacks – the Web developer blog, Jun. 28, 2017. https://hacks.mozilla.org/2017/06/building-the-web-of-things/

[16] "Solution for IoT Interoperability - W3C Web of Things (WoT)," www.w3.org. https://www.w3.org/2020/04/pressrelease-wot-rec.html.en (accessed Aug. 04, 2022).

[17] B. Chan, "IoT Subscription Models - 6 Key Considerations," IoT For All, Aug. 22, 2017. https://www.iotforall.com/iot-subscription-strategy (accessed Aug. 04, 2022).

[18] "What is Root Detection?," Verimatrix. https://www.verimatrix.com/knowledge-base/application-security/what-is-root-detection/

[19] A. Dua, V. Tyagi, and N. Patel, "IISR: A Secure Router for IoT Networks," Nov. 2019.

[20] "FIPS - Amazon Web Services (AWS)," Amazon Web Services, Inc. https://aws.amazon.com/compliance/fips/

[21] S. Sinha, "State of IoT 2021: Number of Connected IoT Devices Growing" IoT Analytics, Sep. 22, 2021. https://iot-analytics.com/number-connected-iot-devices/ (accessed Aug. 04, 2022).

[22] "The rise of consumer IoT: Smart devices that are becoming a part of the average household," clockwise.software. https://clockwise.software/blog/consumer-iot-smart-devices-become-a-part-of-average-household/ (accessed Aug. 05, 2022).

[23] "Wi-Fi module introduction–TYJW2S-5V-Tuya IoT Development Platform-Tuya Developer," developer.tuya.com. https://developer.tuya.com/en/docs/iot/wifijw2s5vmodule?id=K9605srhjahvz (accessed Aug. 05, 2022).