Security Analysis of Device Binding for IP-based IoT Devices

Jiongyi Chen, Menghan Sun, Kehuan Zhang
Chinese University of Hong Kong

***Abstract*—As one of the fastest growing technologies today, the Internet of Things has profoundly changed the ways people interact with the physical world. With a mobile application on a smartphone, a user can conveniently control an IoT device and acquire the sensor data of the external environment. To enable such convenience, a critical step is to bind the user's smartphone with the IoT device and then establish a secure communication channel between them. Although various techniques have already been adopted, however, little has been done so far to systematically evaluate the security implications of those binding mechanisms in IoT.**

In this paper, we report the first systematic study on device binding mechanisms of IoT, in an attempt to understand the security implications. For this purpose, we defined a practical adversary model and systematically investigated 24 popular IoT products on the consumer market. Our investigation reveals the fact that IoT developers often mistrust the environment and do not follow best practices in device binding. As a result, we were able to launch several types of real-world attacks against the device binding process. Our research brings the insecure designs of device binding to the spotlight and shows that the threat to IoT device binding is realistic and serious.

I. Introduction

Nowadays, the emerging IoT technologies have brought great convenience to people's daily life and have enabled numerous applications such as wearables, smart home, smart grid, smart city and beyond. Using a smartphone to connect with IoT devices and visualize sensor data, users can easily understand the physical environment and operate the devices. Before the smartphone and the IoT device establish a secure communication channel, their connection should be bootstrapped through the so-called device binding process.

Unlike the client-server communication of web applications, in which a client is able to verify the identity of the server with a trusted third party like a CA (i.e., Certificate Authority), device binding does not involve other parties and can only rely on user involvement to bootstrap the communication between the user app and the intended IoT device. In this case, the main concern is the user experience (e.g., usability), rather than the security guarantees. As many protocols were proposed to minimize the user involvement [5], [15], but did not clearly define and evaluate the security boundaries.

On the other hand, as IoT devices often equip with different peripherals, such as wireless modules, speakers, and cameras, the vendors tend to adopt various binding methods based on the peripherals of the devices. For instance, with wireless modules, the device binding of smart plugs is completed by emitting a device-specific Wi-Fi signal for users to select. For IP cameras, the vendors often utilize the camera's capability: the camera can scan the QR code on the user app to uniquely bind with the user. However, given the diversity and complexity in today's IoT peripherals, less attention is paid to the security risks of those binding mechanisms in commercial IoT products.

To better understand the problem, in this paper, we take the first step to present a systematic study on IoT device binding mechanisms and to understand the security implications. Particularly, we first defined a realistic adversary model and categorized the binding mechanisms of 24 IP-based IoT products. Then, we manually reverse engineered each device and found that man-in-the-middle attacks can be realistically implemented in 20 of the devices. Our research reveals that a major misunderstanding is the mistrust of the local environment, which often causes the lack of authentication during device binding. As a result, this allows nearby attackers, such as neighbors, to stealthily launch man-in-the-middle attacks and to completely take control of the victim's device. Our findings bring such misunderstanding of emerging IoT technologies into the spotlight and contribute to a better understanding and ultimately eliminating the threat.

Contributions. We summarize this paper's contributions as below:

1) *Systematic study.* We conduct the first systematic study on device binding in IoT, by defining a practical adversary model, categorizing existing binding mechanisms into three types and evaluating their security risks.
2) *New findings and case studies.* By conducting case studies on 24 popular IoT devices, our research reveals the worrisome situation that considerable designs of device binding for real-world IoT products are insecure.

II. Related Work

In this section, we first give a high-level picture of security analysis of IoT. Then we introduce existing work on device binding and how our work is different from them.

Security Analysis of IoT. In recent years, researchers have increasing interests in the security of IoT systems [6], [4], [17], [13], [7], [12]. Most of the existing work focuses on the security analysis of IoT devices. To this end, researchers have proposed various techniques to discover implementation flaws and to explore new attack vectors. For instance, Costin et al. [6] performed a large-scale analysis of implementation flaws in 32 thousand firmware images and discovered 38 previously unknown vulnerabilities, indicating that today's firmware of IoT devices is poorly implemented. Moreover, IoTFuzzer [4] is a tool based on IoT apps to discover implementation flaws

978-1-5386-9151-9/19/$31.00 ©2019 IEEE

of IoT devices in a firmware-free way. For exploration of attack vectors, Müller et al. [17] performed a systematic study on network printers by summarizing existing attacks and designing tools to detect known attacks. Ho et al. [13] studied 5 popular smart locks and discovered several new attacks to leak information and even unlock the doors.

Another direction is to explore security problems in IoT apps and IoT clouds [9], [20], [14], [10], [8]. Those works mainly focus on tackling over-privilege misuse in IoT apps and IoT clouds. For example, SmartAuth [20] is a NLP-based framework to bridge the gap between real behaviors in code and high-level functionalities in the description of IoT apps, providing a fine-grained access control. ContexIoT [14] utilizes context information for more fine-grained access control of sensitive actions in IoT platforms. Moreover, Earlence et al. [8] performed empirical security analysis of one emerging smart home programming platform and found that the cloud-side privilege separation model could lead to significant over-privilege.

Device Binding. Device binding [1] is not a new topic and there is a large number of works (e.g., [5], [19], [16], [3], [18]) to tackle this problem. Those approaches utilize various OOB channels. such as vision, ambient sound, ambient signals, and etc. For example, BEDA [19] is a new protocol using button pressing for secure device binding. ShaCK [16] could construct a cryptographic key by matching features extracted from the motion sensor data, when the user simultaneously shakes two devices. D. Balfanz et al. [3] presented new schemes using demonstrative identification for pre-authentication over location-limited channels in ad-hoc wireless networks. UniverSense [18] is an alternative pairing method for low-resource IoT devices without an interface. Moreover, it enables more powerful devices such as cameras to pair with each other, by comparing the fingerprint of sensor data and physical motions. Perceptio [11] could pair IoT devices that equip with different types of sensors using context information autonomously.

Although above solutions can provide a certain level of usability and security in special situations, however, there is no existing work to systematically study whether the binding mechanisms of IoT devices are vulnerable in real world.

III. Security Analysis

A. Adversary Model

In our adversary model, we consider *nearby attackers* (e.g., neighbors), including *wireless attackers* and *LAN attackers*, who are near the user during device setup. Since wireless devices and wired devices use different methods to bind with apps, we consider different attacker's capabilities. In particular, binding of wireless devices is achieved via wireless channels. Therefore, we consider wireless attackers who can sniff the local Wi-Fi and emit their own Wi-Fi signals (and such signals could reach the victim's devices), since the range of Wi-Fi signals can reach over tens to hundreds meters. However, the attackers could not hear the sound from the devices (if the binding utilizes audio waves). On the other hand, binding of wired devices is achieved by first connecting the device to the local area network (LAN) with a cable and then binding with the app. Therefore, we consider *LAN attackers* who can access the local network of the user's and sniff the packets.

[1]Before two devices can transfer any data, they are required to established a virtual connection. Such a process is called device binding [5].

B. Local Binding and Attacks

In our research, we investigated local binding of the IoT devices with the user. When vendors design the binding process, in practice, they often take user experience into consideration and make use of the peripherals of an IoT device. As a result, on the consumer market, there are various binding methods based on different peripherals of products. However, to our surprise, most of the binding methods that we studied sacrifice security for usability and simply use identification instead of authentication to bind the app and the device. Even worse, in some designs, the surrounding environment is trusted by default and the app can be bound with any nearby devices that are in listening mode. In this subsection, we categorize the binding methods of our experimental devices based on their device types as in Table **??** and describe the attacks on vulnerable binding methods.

Type 1: binding with wireless devices through manual identification. In practice, there are two kinds of binding in this category: one is automatic discovery of a device, the other requires manual identification of a device hotspot. For automatic device discovery (*Type 1-1* in Figure 1), the device can be automatically bound with the app to minimize user involvement. Initially, the device is configured to the listening mode for binding and acts as a Wi-Fi hotspot to receive broadcast messages from the app. Meanwhile, on the cellphone, the user will select his home Wi-Fi from the list and input the password of the home Wi-Fi. Next, the app will *broadcast the home Wi-Fi credential to any nearby devices* that are listening. After the device receives the message, both the device and the app will connect to the same home Wi-Fi.

On the other hand, device binding can also be completed by manually identifying the SSID of the intended device (*Type 1-2* in Figure 2). At first, the device plays the role of a Wi-Fi hotspot for the app to connect, where passwords of those hotspots are not required for most of the devices. Interestingly, we found that some of the vendors implement them in a way that seems to be secure but is shown to fail in our attacks. In their design, a label of the device password is attached to each device, so that only the owner can see the password and connect with the device. We call it *PIN-based manual identification* (*Type 1-3* in Figure 2). After the user connects with the device, similar to the above approach, he will then choose his home Wi-Fi for the device to connect. Finally, the device and the app will discover each other on the home Wi-Fi network. Since the user needs to first connect with the specific device according to its SSID, the credential of home Wi-Fi will be delivered to that device only.

Attacks on Type 1. Our attack leverages another device with the same type as the user's device. When the user is setting up his device at home, an attacker also configures a malicious device outside his home. For automatic device discovery (*Type 1-1*), any nearby devices (including malicious devices) could be bound with the app, when they are in listening mode and receive the broadcast message. In the meantime, the user app will automatically and randomly choose a

TABLE I
local binding attacks

Device Type	Attacker	Binding Type	Device Authentication	Local Binding Attacks	Leaking Home Wi-Fi Credential?
Wireless Devices	Wireless Attackers	Binding Through Manual Identification (T1)	Automatic Discovery (T1-1)	Yes	Yes
			Manual Identification (T1-2)	Yes	Yes
			PIN-based Manual Identification (T1-3)	Yes	Yes
		Binding Through OOB Channels (T2)	Secret Vision Channel (T2-1)	No	No
			Secret Audio Channel (T2-2)	No	No
Wired Devices	LAN Attackers	Binding Through OOB Channels (T3)	Token Verification (T3)	Yes	Not Applicable

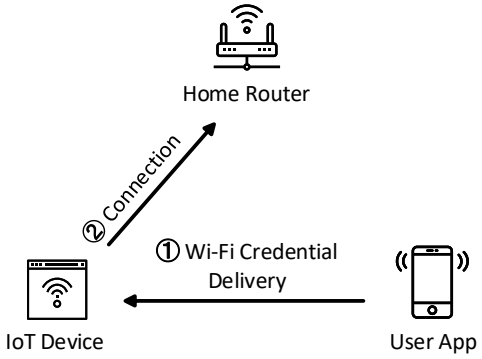

Fig. 1. Binding Type 1-1

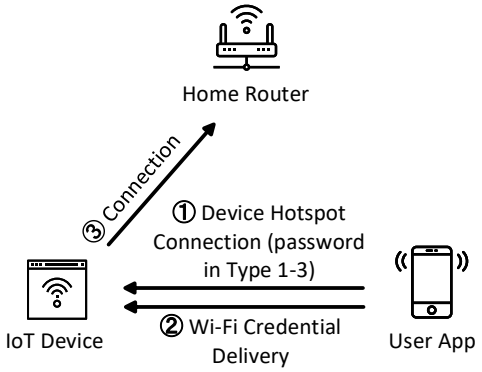

Fig. 2. Binding Type 1-2 & Type 1-3

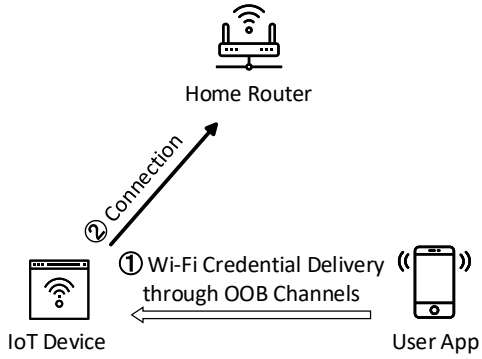

Fig. 3. Binding Type 2

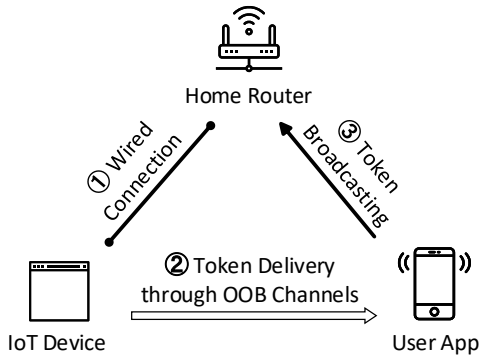

Fig. 4. Binding Type 3

device to bind, for user's convenience during setup. For manual identification based binding (*Type 1-2*), when the user chooses the device on the cellphone's Wi-Fi list, he will find two SSIDs that have the same device name like "TP-LINK SMART PLUG". Since the user is unable to differentiate the malicious device and his own device simply based on the names, he could be trapped to choose the malicious device to connect with (Figure 5). As a result, the credential of the home Wi-Fi could be leaked to the attacker because the attacker's device receives the credential (Figure 6). In the meantime, the attacker could bind his app with the user's device.

Interestingly, even if the Wi-Fi of the device is protected by a password (*Type 1-3*), the user is also trapped to input the password on the malicious device. In this case, the attacker's device is essentially a phishing device that can lure the user to input the correct device password. As a result, the attacker could then connect with the user's device with the user's password. At this point, the user is connected with the attacker's malicious device and the attacker is connected with the user's device. In order to let the user feel like he is connected with his own device (i.e. to achieve stealthiness), the attacker only needs to receive the commands at the malicious device and forward the commands to the user's device through the attacker's app.

Type 2: binding with wireless devices through OOB chan-

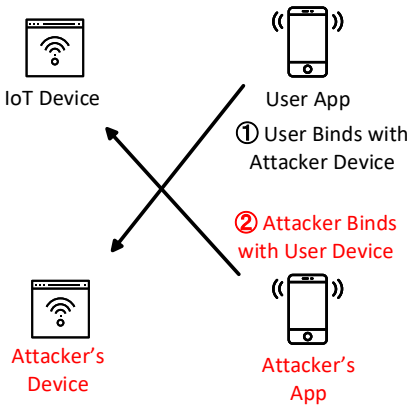

Fig. 5. Attack on Type 1-2: Step I

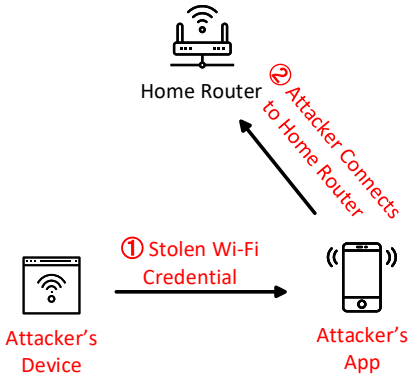

Fig. 6. Attack on Type 1-2: Step II

nels. For this type (as in Figure 3), some secret information is exchange through OOB channels that attackers are unable to access before or during binding. For a wireless device, the credential of home Wi-Fi is delivered to the device through an OOB channel such as scanning visual images or emitting sound waves. Among the devices that we studied, some smart IP cameras take such an approach. After the user inputs his home Wi-Fi credential, the credential will be encoded as a QR code (*Type 2-1*) or a sound wave (*Type 2-2*). Next, the device will scan the QR code or listen to the sound wave to receive the home Wi-Fi credential. This design could prevent attackers from obtaining the Wi-Fi credential because the information (Wi-Fi credential) is exchanged through the secret OOB channels.

Type 3: binding with wired devices through OOB channels. For some devices that do not have wireless modules such as NAS, they are first connected to the home router through a network cable (Figure 4). In the meantime, the user needs to connect the app to the same home router (Wi-Fi) and scan the QR code on the device label (the OOB channel here is vision) to get a token for binding. Later on, the app will broadcast a request with the token to discover the device. Once the device verifies such a token, it will response the app with an acknowledgment. Unfortunately, We found that this approach could still be vulnerable without careful design.

Attacks on type 3. This design suffers from replay attacks in the presence of active attackers on the local networks. An attacker could perform man-in-the-middle attacks by simply replaying on the token to bind with the victim's device, if the device token sent from the app is not random. First, the attacker captures the token and responses to the user's app with an acknowledgement. Then the attacker can simply replay (or broadcast) the captured token to bind with the user's device.

Summary and discussion. In our research, we take the first step to systemically analyze the local binding methods in various IoT devices. Launching local binding attacks on those devices, we were able to confirm that all of the *type 1-2* devices and one *type 3* devices suffer from man-in-the-middle attacks. The results reveal the attacks could occur in existing dominant designs. And that is because developers mistrust the environment and do not have a guideline for secure implementation of local binding. Our investigation on local binding brings out the following observations:

- During local binding, the user authenticates the intended device by confirming the device name that is claimed by the device. The lack of secure authentication opens the door to nearby attackers who can then stealthily take control of victim's device and obtain home Wi-Fi credentials. Unfortunately, such a fundamental weakness cannot be naively fixed via adopting a device Wi-Fi password.
- Binding through OOB channels is relatively secure but requires specific peripherals (e.g., camera, audio receiver) on the wireless IoT devices. Besides, it could still be vulnerable to the man-in-the-middle attack without careful design (e.g., the verification token is not random).

IV. Evaluation

A. Methodology

We studied 24 types of IoT devices produced by popular IoT vendors like Amazon, Philips and TP-LINK (see Table II). To conduct experiments on those IoT devices, we purchased two instances of each type of IoT devices. Below, we briefly introduce the experiment setup and the engineering efforts to perform our attacks:

- *Experiment Setup.* We created an isolated network environment by setting a home Wi-Fi and carefully conducted our experiment with the devices. The experiments are conducted locally (i.e., cloud services and other users are not affected) and, each time, only two devices of the same type are in use.
- *Traffic Analysis.* It helps us to have a basic understanding of the binding workflow. To capture and analyze the traffic/messages between the device and the user app, we setup an access point on a Ubuntu host machine with 8G RAM and Intel Core i7 2.81 GHz. Then we installed Wireshark to analyze the traffic and used a MITM proxy [2] to intercept encrypted HTTPS messages.
- *User App Reverse Engineering.* We focus on Android platforms and installed the corresponding official apps

on two Android phones (Nexus 6, Android 6.0.1, 2G RAM and 4 Cores) to bind with the IoT devices. Moreover, we use the dynamic instrumentation tool Frida [1] to intercept and modify the requests from the apps.

- *Firmware Reverse Engineering.* To obtain the stolen password on the attacker's device, we need to perform firmware reverse engineering. For those devices whose firmware images are available, we repack the firmware images. If the firmware is unavailable, alternatively, we inject a backdoor into the running systems through debugging ports.

B. Results

The overall statistics of the testing devices is shown in Table II, including vendor names, device types, local binding methods, whether local binding attacks are succeeded, and the maximum effective range of the attacks. 54% devices adopt *type 1-1* binding method and most of them suffer from local binding attacks. One exception is Philips Hue: when the user's device and the attacker's device are in listening mode, the app shows that two devices are prepared for binding. In this case, the user can check which device is his/her own device, to prevent the attack. On the other hand, all of the *type 1-2* devices suffer from local binding attacks. There are only two devices that adopt *type 3* binding method. One of them suffers from local binding attack, while the other does not. The reason of the failure is that the Samsung SmartThings hub adopts a challenge-response protocol with the cloud, in which the user needs to input the ID on the physical device to confirm the ownership.

For *type 1* binding, a major concern is the range between the user and the device. Therefore, we evaluated the maximum effective range of the attacks in an open area. In Table II, we also show the maximum range of the user and the attacker's device (which is the same as the maximum range of the attacker and the user's device). As can be seen, the average distance is around 42.25 m, which is reasonable to launch attacks.

TABLE II
statistics of local binding and the attacks

Vendor	Device Type	Local Binding		Max Effective Range (m)
		Binding Methods	Local Binding Attack	
Alibaba	Tmall Genie	*Type 1-1*	✓	60
Amazon	Echo Dot	*Type 1-2*	✓	70
Belkin	Smart Plug	*Type 1-2*	✓	30
BroadLink	Smart Plug	*Type 1-1*	✓	45
BroadLink	Remote RM	*Type 1-1*	✓	40
D-LINK	Smart Plug	*Type 1-3*	✓	30
D-LINK	Network Camera	*Type 1-3*	✓	35
KONKE	Smart Socket	*Type 1-1*	✓	25
Lightstory	Smart Plug	*Type 1-1*	✓	40
Litsped	Smart Plug	*Type 1-1*	✓	50
Mini	Smart Plug	*Type 1-1*	✓	50
Orvibo	Smart Socket	*Type 1-2*	✓	40
OZWI	IP Camera	*Type 1-1*	✓	60
Philips	Hue	*Type 1-1*	✗	60
QNAP	NAS	*Type 3*	✓	N.A.
Samsung	SmartThings Hub	*Type 3*	✗	N.A.
TECKIN	Smart Plug	*Type 1-1*	✓	30
TP-LINK	Wi-Fi Camera	*Type 1-1*	✓	50
TP-LINK	Smart Bulb	*Type 1-2*	✓	40
WD	MyCloud	*Type 1-1*	✓	20
Xiaomi	Story Teller	*Type 1-1*	✓	30
Xiaomi	Home Hub	*Type 1-2*	✓	40
YI Tech	Home Camera	*Type 2-1*	✗	N.A.
360	Smart Camera	*Type 2-2*	✗	N.A.

✓: Success
✗: Failure
N.A.: Not Applicable

V. DISCUSSION

Although our study presents the first security evaluation of IoT local binding, there are still some limitations and space for further improvement. In this section, we discuss the limitations of our work and point out the directions of the future work.

Problem scope and assumptions. In our research, we only consider man-in-the-middle attacks during local binding. However, there could be vulnerable designs in the registration process (i.e., binding with the cloud), which is often related to the local binding. Additionally, the attacker needs to know which device the user has and when the device is set up. Because the attack can only be successfully launched when the user is configuring her/her devices. Luckily, the first problem can be solved with device fingerprint techniques and the timing problem can be solved by automatic scanning.

Evaluation metrics. On one hand, although we could have evaluated more device types, we only purchased 24 types of popular devices due to limited budgets. On the other hand, in the evaluation, we only consider the setting of open area when testing the maximum effective range. However, the attacks are not evaluated in the environment with obstacles, such as inner buildings.

VI. CONCLUSION

We report the first systematic study on device binding of IoT. By evaluating 24 IoT devices, our study summarizes binding types and attack vectors in the designs of popular IoT devices. Further, we demonstrate several attacks that can take complete control of the victim's devices. Our research reveals the fact that many IoT devices are vulnerable to active attackers during local binding and the lack of security evaluation.

ACKNOWLEDGEMENTS

We thank anonymous reviewers for their insightful comments. This work was partially supported by National Natural Science Foundation of China (Grant No. 61572415), and the General Research Funds (Project No. 14208818 and 14217816) established under the University Grant Committee of the Hong Kong Special Administrative Region, China.

REFERENCES

[1] "Android frida," https://www.frida.re/docs/android/, Accessed: June 2018.

[2] "Fiddle: Web debugging proxy," https://www.telerik.com/fiddler, Accessed: June 2018.

[3] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks." in *NDSS*. Citeseer, 2002.

[4] 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," 2018.

[5] M. K. Chong, R. Mayrhofer, and H. Gellersen, "A survey of user interaction for spontaneous device association," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 8, 2014.

[6] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis, "A large-scale analysis of the security of embedded firmwares." in *USENIX Security Symposium*, 2014, pp. 95–110.

[7] A. Costin, A. Zarras, and A. Francillon, "Automated dynamic firmware analysis at scale: a case study on embedded web interfaces," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 2016, pp. 437–448.

[8] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 636–654.

[9] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks." in *USENIX Security Symposium*, 2016, pp. 531–548.

[10] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Decentralized action integrity for trigger-action iot platforms."

[11] J. Han, A. J. Chung, M. K. Sinha, M. Harishankar, S. Pan, H. Y. Noh, P. Zhang, and P. Tague, "Do you feel what i hear? enabling autonomous iot device pairing using different sensor types," in *2018 IEEE Symposium on Security and Privacy (SP)*, vol. 00, pp. 678–694. [Online]. Available: doi.ieeecomputersociety.org/10.1109/SP.2018.00041

[12] G. Hernandez, F. Fowze, D. J. Tian, T. Yavuz, and K. R. Butler, "Firmusb: Vetting usb device firmware using domain informed symbolic execution," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2245–2262.

[13] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proceedings of the 11th ACM on Asia conference on computer and communications security*. ACM, 2016, pp. 461–472.

[14] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao, A. Prakash, and S. J. Unviersity, "Contexiot: Towards providing contextual integrity to appified iot platforms," in *Proceedings of The Network and Distributed System Security Symposium*, vol. 2017, 2017.

[15] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, "Caveat eptor: A comparative study of secure device pairing methods," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–10.

[16] R. Mayrhofer and H. Gellersen, "Shake well before use: Intuitive and secure pairing of mobile devices," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 792–806, 2009.

[17] J. Müller, V. Mladenov, J. Somorovsky, and J. Schwenk, "Sok: Exploiting network printers," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 213–230.

[18] S. Pan, C. Ruiz, J. Han, A. Bannis, P. Tague, H. Y. Noh, and P. Zhang, "Universense: Iot device pairing through heterogeneous sensing signals," in *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, ser. HotMobile '18. New York, NY, USA: ACM, 2018, pp. 55–60. [Online]. Available: http://doi.acm.org/10.1145/3177102.3177108

[19] C. Soriente, G. Tsudik, and E. Uzun, "Secure pairing of interface constrained devices," *International Journal of Security and Networks*, vol. 4, no. 1-2, pp. 17–26, 2009.

[20] 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 *26th Security Symposium*. USENIX Association, 2017, pp. 361–378.