Security in IoT Pairing & Authentication protocols, a Threat Model and a Case Study Analysis

Daniele Granata, Massimiliano Rak, Giovanni Salzillo and Umberto Barbato

Università della Campania "Luigi Vanvitelli", Dipartimento di Ingegneria, Aversa (CE), 81031 Italy

Abstract

The Internet of Things has changed the way we interact with the environment around us in our daily life, and it is increasingly common to find more than one IoT device in our home. However, the current design approaches adopted by the vendors are more oriented towards customer usability than to security. This often results in more and more devices exposing serious security problems. This work focuses on the security implications, i.e. the threats and the risks, of the current IoT pairing mechanisms and represents a step forward in the definition of our automated penetration testing methodology. In addition to the general threat model for a general IoT pairing process, we present the analysis of a QR code-based pairing mechanism implemented by a class of devices taken from the real market, which led to the identification of two vulnerabilities, one of which publicly disclosed as CVE-2021-27941.

Keywords

IoT security, pairing protocols, threat model, IoT pairing

1. Introduction

Internet of Things is becoming an accepted reality. Vocal assistants, smart lamps, smart plugs are more and more common in our houses. Industrial systems include smart meters, connected robots, and drones. The health system makes large use of portable devices for continuous monitoring of critical sick persons. Cars are complex and connected computer systems. As a matter of facts, our daily life is full of interactions with several multiple surrounding devices.

Users need to continuously interact with new devices, establishing a connection and correctly using them. *Pairing* is the term adopted for describing the process of establishing a connection with a new device [1]. Pairing processes should be as simple as possible, otherwise, interactions become unacceptable for common life interactions.

However, pairing includes complex procedures and relies on the authentication of the peers connecting each other. As commonly happens, there is a clear trade-off between *usability* of pairing processes and the *security*, of the process itself [2]. Considering that users, as a general assumption, have absolutely no expertise of any kind, such trade-off is hard to consider, and often usability is the only factor considered, sometimes opening IoT devices to severe security issues.

daniele.granata@unicampania.it (D. Granata); massimiliano.rak@unicampania.it (M. Rak); giovanni.salzillo@unicampania.it (G. Salzillo); umberto.barbato1@studenti.unicampania.it (U. Barbato)
0000-0002-6776-9485 (D. Granata); 0000-0001-6708-4032 (M. Rak); 0000-0001-6491-9655 (G. Salzillo)

CEUR Workshop Proceedings (CEUR-WS.org)

To assess the security of IoT systems, our research team is working on a methodology that aims at automating IoT-based security testing. This paper, together with [3, 4, 5], is a concrete testing process made to build up such an automated procedure.

In this paper, we focus on the pairing process of IoT devices and made a systematic analysis of smart devices for home systems, building a general-purpose threat model. Thanks to our methodologies, we made a threat-based analysis of the devices pairing process, identifying the possible attacks that implement the identified threats, demonstrating their feasibility on real devices, and how some pairing systems may lead to dangerous side effects. The experiments lead to the discovery of two vulnerabilities, for one of which we requested a CVE id. Also, we contacted the vendor before the submission of this paper to leave him the time to apply some internal countermeasures to mitigate the vulnerabilities.

As a summary, the main contributions of this work are: (i) a generic Threat Model for the IoT device pairing process, (ii) a demonstration of the usefulness of our threat-based penetration testing approach, (iii) a detailed security analysis of a particular pairing scheme from the real market, (iv) the discovery of two vulnerabilities in real-world class of devices.

The remainder of the paper is organized as follows: Section 2 illustrates the state of art, focusing on the existing pairing schemes and the security analysis available in the literature on the topic. Section 3 briefly summarizes our threat-based approach to penetration testing and the Threat Model we made for the IoT pairing process. Section 4 describes the analysis on a real-world class of devices and presents one of the two discovered vulnerabilities. Section 5 summarizes the conclusions and the future works.

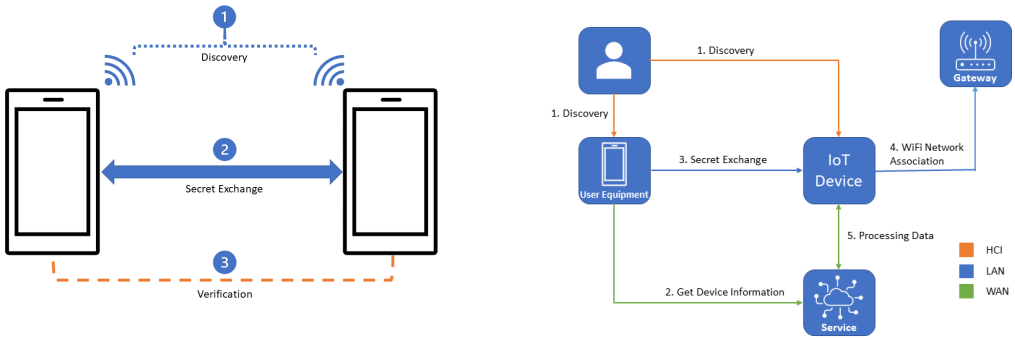
2. IoT Pairing State of the Art

In the current literature, the term *pairing* is defined as the bootstrap process that creates a communication channel between two entity, which have never met each other before [1]. A pairing process involves the *authentication* process, needed to prove each device the identity of the other. Accordingly, it is worth noticing that device authentication is only a component of the pairing scheme, even if in literature sometimes the two processes are confused. Hereafter, we briefly summarize the existing techniques and the taxonomies for device authentication available in the current state of the art.

El-Hajj et Al. [6] analyze multiple IoT authentication schemes through a multi-layered approach, i.e. through the *Perception*, the *Network* and the *Application* layer, proposing a taxonomy of IoT authentication schemes based on multiple properties, such as the authentication architecture, the authentication procedures, the number of authentication factors, the use of a token, or other hardware-based properties.

Similarly, Meneghello et Al. [7] present an extensive survey about the most common IoT authentication schemes, as well as their weakness and the possible countermeasures, even if adopting a different layering, made of the *Edge*, the *Middleware*, and the *Application* layer.

Ferrag et al. [8] address the IoT authentication schemes problem with a different approach, mainly based on the application context: Machine-to-Machine (M2M), Internet-of-Vehicles (IoV), Internet-of-Energy (IoE) and Internet-of-Sensors (IoS). The authors propose a framework to evaluate and implement a secure authentication schema.



(a) Generic Pairing Schema

(b) Case Study Pairing Schema

Figure 1: Pairing Schema

As concerning the device pairing process, and in particular, the process of pairing a device to the customer's WiFi network, very few papers are available in the literature. A survey on the available IoT pairing schemes is available in [1] by Fomichev et Al. According to the authors, a general pairing schema consists of 3 steps: (i) the *discovery*, (ii) the *secret exchange* and (iii) the *verification* step. In Figure 1a is reported a generic pairing association schema. During the discovery phase, the involved entities try to discover each other. Next, to create a secure communication channel, a secret exchange phase is needed, during which both entities agree and exchange a secret, for example, a key to encrypt messages or a password to join the customer's WiFi network. Finally, during the verification phase, the involved entities verify the shared secret and ensure that the process has not tampered.

Due to the intrinsic characteristics of the wireless communication links, those pairing procedures are often vulnerable to Man-In-The-Middle (MITM) attacks. Therefore, an auxiliary out-of-band channel (OOB) is usually involved for the key-exchange step. The authors outline that this communication channel is not necessarily a dedicated PHY (Physical) channel, but could also be a HCI (Human-Computer Interaction) channel, which is a channel where data is transmitted through human interaction.

As concerning the existing pairing techniques and their underlining technologies, multiple schemes have been proposed [9, 10, 11, 12, 13]. Therefore, device vendors stretch to develop custom solutions, relying on the security-by-obscurity paradigm and often resulting in insecure pairing mechanisms [5]. It is worth noticing that a vulnerable pairing protocol may affect not only the IoT device but also the full system in which the device resides.

Moreover, the existent pairing solutions often rely on dedicated protocols built on-top of physical channels [12, 13, 9]. From a security point of view, it is worth noticing that different physical channels have different security properties. Furthermore, physical channels are often used as out-of-band communication channels as a "verification" step, i.e. devices communicate on a network, like a WiFi network, but adopt a dedicated physical communication channel to share the data needed to authenticate each other.

As already outlined, some pairing solutions rely on particular OOB channels to securely authenticate the devices, i.e. the *Human Computer Interaction* (HCI) channel. Within an HCI

channel, the user is asked to perform a physical action, like pressing a button or scan a QR code. Nevertheless, HCI channels are subjected to the human error, so the user could become the "weakest link" of the entire pairing procedure [10, 11].

Despite the cited references, currently, there is no standard reference architecture for the pairing mechanism, and on which base a systematic security analysis approach. Accordingly, there is a clear need for a standard security solution and the definition of the general-purpose best practices for IoT-based systems.

In this paper, we focus on a specific pairing schema that uses an OOB channel, an HCI and an external service for the device configuration process. Figure 1b shows the involved parties and summarizes, at the same time, the steps required by the pairing schema.

This particular schema involves an initial HCI interaction (step 1), by which the user starts the pairing process on both the IoT device and his user equipment (UE), usually a smartphone. The HCI step could also involve the acquisition of a device secret, later used to authenticate the communication. The user equipment retrieves some device information from a remote service (step 2) and shares the network credentials to the IoT device (step 3), possibly throughout and OOB channel. Next, the device joins the user's network (step 4) and registers itself to a remote service (step 5), typically to be remotely associated with the customer's account, thus completing the configuration process.

This work analyzes the threats for this particular pairing schema and represents another little step forward in the demonstration of the versatility of our general-purpose methodology for systematic and threat-based penetration testing, already adopted in other contexts [4, 5].

3. Security Assessment Methodology

The methodology adopted in this work was previously proposed in [4, 3] and we briefly summarize here the involved steps. The approach enables less-skilled penetration testers to perform a threat-modeling-driven penetration testing security evaluation, guiding them to look for system vulnerabilities on a per threats basis. As highlighted in Fig 2, the approach relies on four phases: (i) the **System Modeling**, during which a (semi-)formal description of the system under test is produced, (ii) the **Threat Modeling**, a phase devoted to the threat identification, (iii) the **Planning phase**, for test and attack planning, and (iv) the final **Penetration Testing** phase, for the actual attack execution.

Figure 2: The steps of the used methodology

According to our approach, the *System-under-Test* (SuT) is described by the Multi-cloud Application Composition Model (MACM) formalism [14, 15], a graph-based system modelling formalism introduced to describe architectural components and security properties of cloud-based applications and IoT systems. The purpose of the model is to offer a simple and synthetic description of the system, enabling our automation techniques for threat modelling.

In MACM, system components are modelled with graph nodes, whereas relationships between system components are represented with directed links between nodes. The proposed formalism allows to properly represent system architecture components and also enable to annotate security-concerning information in a human and machine-readable format.

Thanks to the techniques described in [14, 16], it is possible to generate a full list of threats that may affect the SuT starting from a pre-organized, general-purpose catalogue. The catalogue is constructed in such a way that MACM nodes coincide with the threat asset-type and it includes threats for many software components and protocols (Ethernet, IP, TCP, TLS, XMPP, OAuth, Zigbee, Bluetooth, BLE, GSM/3G/4G/5G) and it is currently maintained and constantly updated. The goal of the threat model is to support and guide less-skilled penetration testers at identifying system security vulnerabilities. In this paper, we enriched the catalogue with a set of threats specific to IoT devices pairing process.

In the planning phase, penetration testers analyze the identified threats and select the most promising one, based on his experience and his competencies in terms of possible attacks. Currently, we are working on techniques that aim at automating this step (in [3] and in [4] there are two alternative possible solutions). Once selected the targeted component and the security objectives to be tested, penetration testers look for available vulnerabilities and, eventually, set up the appropriate tools or framework to exploit and put into effect the chosen threat.

3.1. Pairing Process: A Threat Model

As anticipated, the automated threat modelling procedure relies on a predefined threat catalogue, that collects high-level and system-independent threats, organized by asset type.

In our methodology, a threat is modelled as the triple $\{ThreatAgent, Asset, Behavior\}$, representing the possibility of a *Threat Agent*, i.e. a malicious actor, to cause damage to an *Asset*, i.e. something that has a value in the system, through a specific *Behavior*, i.e. a set of actions that implies human and/or automated interactions with the target systems [16]. All the threats in the catalogue are linked to the assets of the system under analysis, through the asset type field and are obtained in an automated way through ad-hoc queries on the threat catalogue.

The Table 1, reported in Appendix, contains an excerpt of our catalogue with the specific behaviours of the threats for the IoT pairing processes. Our catalogue contains additional information to characterize the threats (like the STRIDE classification [17]) and the information needed to customize the Threat Model for a specific system.

Also, we identified and reported in Table 1 two classes of possible threat agents: the *Malicious User* and the *Malicious Service*. A Malicious User is a hostile attacker who has internal access to the infrastructure (i.e. connected to the LAN Network) to perform the attacks with no legal and ethical constraints. The threat agent goal is to damage or steal the asset, denying, causing harm or taking control of the asset. The skills required by the attacker(s) is operational, or rather someone who understands the technology background, but is not an expert in performing the attack methods. Unlike the Malicious User, a Malicious Service can also have external access to the system and aims to steal sensitive data. To carry out a social engineering attack, the threat agent requires minimum skills, but he must register a domain that exposes the attacker to greater legal limits.

As an example of identified threats, threat *T2* of Table 1 consider the case in which a malicious

user could compromise the IoT Device by replacing it with a malicious one or by connecting a malicious User Equipment to the customer's device without the legitimate users' awareness. This would affect the IoT device integrity.

Other threats target the customer's network: a malicious user could, for example, connect a malicious IoT device to the user's WiFi network, or physically disturb the communication link to prevent the pairing process from completing (e.g. Jamming). Also, an attacker could manage to compromise the communication between the UE and the remote service. Further, the latter could be subjected to Denial of Service, an attack that aims at denying the service availability. Or worse, it could be subjected to spoofing, i.e. an attacker could replace a legitimate service with a malicious one and obtain sensitive information.

Based on the threats described in this section, the penetration tester analyzes the threat list and constructs the specific attacks in the Planning phase.

4. A Case Study: The eWelink QR Code Pairing Protocol

In this section, we present the methodology applied for the security analysis of a real-market case study, about a pairing schema matching the one presented in Figure 1b. In particular, we analyze the implementation details and the weaknesses exposed by a particular QR-code based pairing schema, tied to the *eWeLink* hub platform.

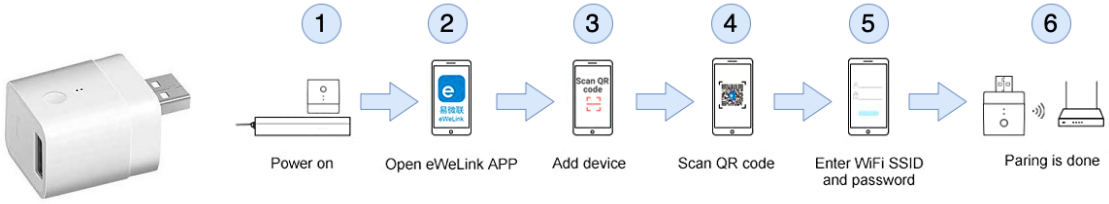
eWeLink [18] is a cooperative smart home platform, compatible with most of the home automation systems such as Apple HomeKit[19], Google Nest[20], Amazon Echo[21]. It supports a wide range of smart devices, including power switches, sockets, LEDs and several other sensors. The eWeLink application is available for free on the Google Play Store and the Apple App Store. Our tests regard the entire class of devices which are QR-code pairable with the latest versions of the eWeLink mobile application (Android version 4.9.2 and iOS version 4.9.1).

The application, depending on the device class, provides five different methods for pairing: (i) the *Quick Pairing* mode, a pairing mechanism based on the ESP-Touch protocol (which we already found vulnerable [5]), (ii) the *Sound Pairing* mode, available for camera and other devices equipped with a microphone, (iii) the *Bluetooth Pairing* mode, for devices equipped with a Bluetooth interface, (iv) the *Compatible Mode*, a fallback compatibility pairing mode based on softAP, and (v) the *Scan QR code* mode, that involves both the scan of a QR code label and the connection to the device WiFi softAP network.

Among the devices supporting the latter pairing schema, we tested the ITEAD Sonoff Micro [22], depicted in Figure 3a, which is a USB smart adapter that converts an ordinary USB power port to a smart controlled switch. It permits to schedule stop charging times, check the power-on status, or remotely turn on/off the attached devices.

The steps required to pair a supported device are reported in Figure 3b. The process starts by setting the IoT device into pairing mode (step 1), and by selecting a new device association on the eWeLink application, tapping on the QR code pairing mode (step 2 and 3). The user is asked to scan the QR code label shipped with the device (step 4) and, then, to insert his WiFi network credentials (step 5). Finally, the device joins the user's WiFi network and gets associated with the user's account (step 6).

Note that, the steps form *1* to *5* in Figure 3b directly coincide to the first step of the generic



(a) Sonoff Micro (b) QR code pairing through the eWeLink application

Figure 3: Device and Pairing Schema

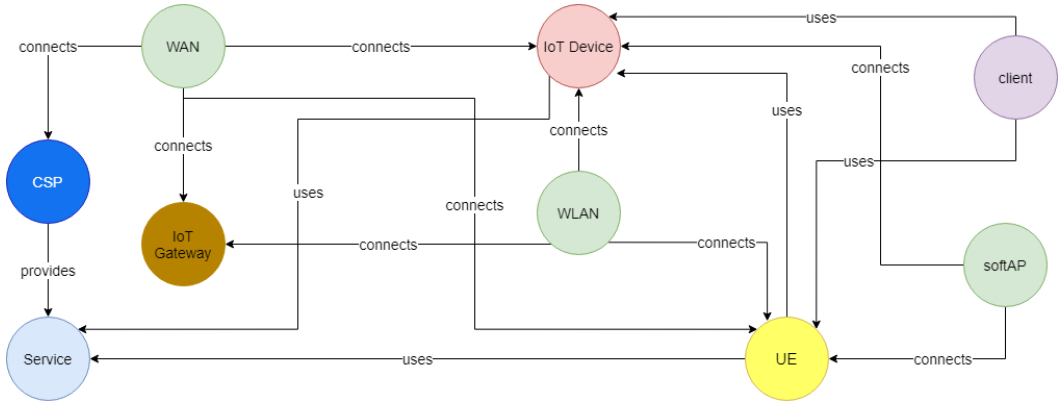


Figure 4: MACM eWeLink case study

pairing model introduced in Figure 1b, i.e. the user-device HCI interaction, described in Section 2. Conversely, step 6 in Figure 3b encloses all the remaining steps (from 2 to 5) of the pairing model of Figure 1b. This is because that the low-level device pairing operations do not require any human intervention, and are automatically executed once all the initial protocols parameters are set by the customer.

4.1. Security Evaluation

To start the security evaluation, and according to the first step of our methodology, we devised the MACM model of the entities involved in the pairing process. The graph in Figure 4 represents these assets, grouped by type in different colours, and their relationships. Among the green coloured network nodes, note the *softAP* network raised by the IoT device for its configuration. Also, note the remote *service* node, hosted by the Cloud Service Provider (*CSP*), that is used by both the *smartphone* (and the eWeLink mobile app) and the *IoT device* for the configuration. Finally, the *IoT Gateway* node represents the user's target WiFi AP.

According to the above-described approach, we derive the detailed threat model of the system from the MACM. For brevity reasons we do not report here the model, but we will refer to the threats using the information in the catalogue reported in the Appendix, in Table 1. For

each threat in the threat model, a certain number of security test plans were produced and then executed by the penetration tester (steps 3 and 4 of the methodology), looking for possible security issues. Hereafter, we report only the successful tests, i.e. the ones that led to a successful attack.

For each threat in the threat model, a certain number of security test plans were produced and then executed by the penetration tester (steps 3 and 4 of the methodology), looking for possible security issues. Hereafter, we report only the successful tests, i.e. the ones that led to a successful attack.

In our case, the *device_id* was a 10 characters long alphanumerical string representing a unique device id. After the QR code scan, the companion application extracts the SSID name and, based on the device id parameter, queries a remote service to retrieve a secret key and other device information (e.g. device model and device type). During the penetration testing steps, we found that, by having a valid platform account, it is possible to manually query the Cloud APIs and retrieve the secret key of any device just by knowing its device id, whether already associated with a network or not.

Later, the mobile application asks the user to insert both the target WiFi network SSID and passkey and finally connects the smartphone to the device WiFi AP network (unprotected by default) to perform the configuration. Accordingly, applying the attacks for the threat T4 in Table 1, we intercepted the HTTP post configuration request to the device. The intercepted JSON message contained the target WiFi network authentication details and the Cloud API endpoint for the user reference region location. This JSON message was encrypted with the AES algorithm, using a 0-pad IV and the previously retrieved secret key. The IoT device has a hard-coded copy of the secret key and, thus, can decrypt the incoming messages.

4.2. Results

According to our test plan results, an attacker could leverage this specific pairing pattern to gain full access to the client's WiFi network. In fact, the device softAP SSID also contains the device id information and, by looking at this field, a malicious attacker would be able to retrieve the related device private key. By knowing this encryption key and due to the openness of the device WiFi softAP, an attacker can sniff the HTTP configuration message sent by the mobile user equipment and decrypt the user's WiFi network credentials.

Hereafter, an example of decrypted JSON *{"ssid" : "Test-Network", "password" : "Test-Password", "serverName" : "eu-disp.coolkit.cc", "port" : 443}*, where the "Test-Network" and "Test-Password" parameters are the network credentials we chose for our testbed configuration. Also, note the presence of the *eu-disp.coolkit.cc* endpoint serving the European region.

Once received the pairing parameters, the device stops advertising its WiFi AP, joins the user's WiFi network and registers itself with the remote API Cloud platform.

Note that, this vulnerability does not expose only the IoT device, but the entire user's network, even though, for the actual exploitation, the threat agent must be located in the proximity of the pairing device.

The security of the eWeLink QR code protocol and, in particular, the confidentiality of the user's network credentials, rely on the proper AES encryption of the configuration messages. The problem is that the QR code does not encode any security information related to the IoT

device (only the public SSID is encoded into the QR label). Instead, a publicly accessible API endpoint is called to retrieve the device secret key, which is used to implement the AES security layer of the pairing protocol. This makes the HCI interaction via QR code useless from a security point of view. Currently, there isn't much users can do to prevent their network credentials from being exposed during a pairing session based on the eWeLink QR code schema. Monitoring the API endpoint for secret key retrieval, or any attempt to associate a secret key exclusively to its first calling user would be ineffective solutions, and a possible cause of denial of service to the customers, intended as the inability to complete a legitimate pairing process. One solution would be to encode the secret key inside the QR label. However, problems arise if the QR sticker is lost: it would become impossible to re-pair the device.

We responsible disclosed the vulnerabilities we found to eWeLink, which acknowledged back and defined a set of provisional countermeasures before a complete API re-design.

We reserved a CVE ID number and published the vulnerability as CVE-2021-27941 [23, 24].

5. Conclusions

"Security is the dark side of Internet of Things"[1] emphasizes that the IoT is a young technology and, as a consequence, it suffers from many concrete issues that must be addressed as soon as possible. This paper is part of a research line that aims at defining a fully automated and threat-based technique for the security assessment of IoT systems. Moreover, our analysis led to the production of a general-purpose threat model for the IoT pairing processes, that can be used for further analysis and can be integrated with our automated risk analysis process. We hope that this contribution raises the awareness of the existing security issues in home-based systems. The security analysis enabled us to disclose and report two major vulnerabilities to the vendor. We also registered a CVE number for one of them: CVE-2021-27941. We didn't disclose any detail about the second vulnerability, which has a major severity, to give eWeLink enough time to implement a correction. We are working on extending the approach to support other pairing schemes and testing more devices, and on the automation, as much as possible, of the entire process. Moreover, as anticipated, we are integrating the testing process with our risk analysis methodologies, to offer a clear definition of the level of risk associated with the adoption of new devices.

[1] https://iaonline.theiia.org/2017/Pages/The-Dark-Side-of-IoT.aspx

6. Appendix

Table 1
Pairing Threat Model.

ID	Threat	Compromised Asset	Behaviour	Threat Agent
T1	Unauthorized Netw. Access	WiFi Network	A Malicious User connects a malicious IoT device to the user's WiFi network	Malicious User
T2	Device Substitution	IoT Device	A Malicious User physically substitutes a legitimate IoT Device with a Malicious IoT Device	Malicious User
T3	Device Hijacking	IoT Device	A Malicious User connects a Malicious UE to the user's IoT Device without the legitimate user's awareness	Malicious User
T4	Eavesdropping	WiFi Network	A Malicious User retrieves valuable data from packets transmitted over the network	Malicious User
T5	Data Leakage	UE	A Malicious User retrieves valuable data about the user's UE, i.e. the user account credentials.	Malicious User
T6	Data Leakage	IoT Device	A Malicious User retrieves valuable data about the user's IoT Device, i.e. by eavesdropping on network packets or by directly querying the device	Malicious User
T7	Impersonation	IoT Device	A Malicious User forces the legitimate user's UE to pair with a Malicious IoT Device	Malicious User
T8	Device Hijacking	IoT Device	A Malicious User pairs a Malicious UE to a legitimate IoT Device	Malicious User
T9	Impersonation	WiFi network	A Malicious User connects a legitimate IoT device to a malicious WiFi network	Malicious User
T10	Jamming	WiFi network	A Malicious User disturbs the WiFi communication channel	Malicious User
T11	Message Elimination	AP WiFi Network	A Malicious User disturbs the communication on the WiFi network	Malicious User
T12	Message Elimination	Network	A Malicious User disturbs the communication with a remote service	Malicious User
T13	Device Hijacking	IoT Device	A Malicious User pairs a Malicious UE by replaying data from a previous pairing	Malicious User
T14	Exhaustion of Power	IoT Device	A Malicious User consumes power resources on IoT Device, i.e. by preventing it from being paired	Malicious User
T15	Data Leakage	Service	A Malicious User obtains valuable information for the IoT device from a remote Service	Malicious User
T16	Device Hijacking	IoT Device	A Malicious User registers the User's IoT Device to the remote Service	Malicious User
T17	Impersonation	Service	A Malicious User replaces the legitimate Service with a Malicious Service	Malicious Service
T18	Denial of Service	Service	A Malicious User makes the legitimate remote Service unavailable	Malicious User

References

[1] M. Fomichev, F. Álvarez, D. Steinmetzer, P. Gardner-Stephen, M. Hollick, Survey and systematization of secure device pairing, IEEE Communications Surveys & Tutorials 20 (2017) 517–550.

[2] S. Dutta, Striking a balance between usability and cyber-security in IoT devices, Ph.D. thesis, Massachusetts Institute of Technology, 2017.

[3] M. Rak, G. Salzillo, C. Romeo, Systematic iot penetration testing: Alexa case study 2597 (2020) 190–200.

[4] S. N. Firdous, Z. Baig, C. Valli, A. Ibrahim, Modelling and evaluation of malicious attacks against the iot mqtt protocol, in: 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017.

[5] G. Salzillo, M. Rak, A (in) secure-by-design iot protocol: the esp touch protocol and a case study analysis from the real market, in: Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy, 2020, pp. 37–48.

[6] M. El-Hajj, A. Fadlallah, M. Chamoun, A. Serhrouchni, A survey of internet of things (iot) authentication schemes, Sensors 19 (2019) 1141.

[7] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella, Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices, IEEE Internet of Things Journal 6 (2019) 8182–8201.

[8] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, L. Shu, Authentication protocols for internet of things: a comprehensive survey, Security and Communication Networks (2017).

[9] G. T. Amariucai, C. Bergman, Y. Guan, An automatic, time-based, secure pairing protocol for passive rfid, in: International Workshop on Radio Frequency Identification: Security and Privacy Issues, Springer, 2011, pp. 108–126.

[10] C. Castelluccia, P. Mutaf, Shake them up! a movement-based pairing protocol for cpu-constrained devices, in: Proceedings of the 3rd international conference on Mobile systems, applications, and services, 2005, pp. 51–64.

[11] J. Zhang, Z. Wang, Z. Yang, Q. Zhang, Proximity based iot device authentication, in: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, 2017, pp. 1–9.

[12] B. Zhang, K. Ren, G. Xing, X. Fu, C. Wang, Sbvlc: Secure barcode-based visible light communication for smartphones, IEEE Trans. on Mobile Computing 15 (2015) 432–446.

[13] C. Soriente, G. Tsudik, E. Uzun, Hapadep: human-assisted pure audio device pairing, in: International Conference on Information Security, Springer, 2008, pp. 385–400.

[14] V. Casola, A. De Benedictis, M. Rak, U. Villano, Toward the automation of threat modeling and risk assessment in iot systems, Internet of Things 7 (2019).

[15] M. Rak, Security assurance of (multi-) cloud application with security sla composition, in: Int. Conference on Green, Pervasive, and Cloud Computing, Springer, 2017, pp. 786–799.

[16] D. Granata, M. Rak, Design and development of a technique for the automation of the risk analysis process in IT Security (2021) 14.

[17] Microsoft, The STRIDE Threat Model, 2009. URL: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20).

[18] eWeLink, the eWeLink smart hub platform, last access, 2021. URL: https://www.ewelink.

cc/en/.
[19] Apple, Apple HomeKit, last access, 2021. URL: https://www.apple.com/shop/accessories/all/homekit.
[20] Google, Google Nest, last access, 2021. URL: https://store.google.com/category/connected_home.
[21] Amazon, Amazon Echo, last access, 2021. URL: https://www.amazon.it/l/15755810031.
[22] ITEAD, Sonoff Micro reference, last access, 2021. URL: https://www.itead.cc/sonoff-micro-5v-usb-smart-adaptor.html.
[23] CVE-2021-27941, MITRE CVE reference, last access, 2021. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-27941.
[24] CVE-2021-27941, CVE reference, last access, 2021. URL: https://github.com/salgio/eWeLink-QR-Code.