Smart Bulbs can be Hacked to Hack into your Household*

1st Davide Bonaventura
Dipartimento di Matematica e Informatica
Università di Catania
Catania, Italy
d.bonaventura@studium.unict.it
orcid.org/0009-0004-4463-7991

2nd Sergio Esposito
Information Security Group
Royal Holloway
University of London
Egham, UK
sergio.esposito.2019@live.rhul.ac.uk
orcid.org/0000-0001-9904-9821

3rd Giampaolo Bella
Dipartimento di Matematica e Informatica
Università di Catania
Catania, Italy
giamp@dmi.unict.it
orcid.org/0000-0002-7615-8643

arXiv:2308.09019v1 [cs.CR] 17 Aug 2023

Abstract—**The IoT is getting more and more pervasive. Even the simplest devices, such as a light bulb or an electrical plug, are made "smart" and controllable by our smartphone. This paper describes the findings obtained by applying the PETIoT kill chain to conduct a Vulnerability Assessment and Penetration Testing session on a smart bulb, the Tapo L530E by Tp-Link, currently best seller on Amazon Italy. We found that four vulnerabilities affect the bulb, two of High severity and two of Medium severity according to the CVSS v3.1 scoring system. In short, authentication is not well accounted for and confidentiality is insufficiently achieved by the implemented cryptographic measures. In consequence, an attacker who is nearby the bulb can operate at will not just the bulb but all devices of the Tapo family that the user may have on her Tapo account. Moreover, the attacker can learn the victim's Wi-Fi password, thereby escalating his malicious potential considerably. The paper terminates with an outline of possible fixes.**

Index Terms—**IoT, smart homes, smart devices, smart bulb, penetration test, vulnerability assessment.**

I. INTRODUCTION

The Internet of Things (IoT) surrounds people virtually everywhere due to the increasing number of devices that become computerised and interconnected — it may even pervade people's bodies thanks to the rise of implantable medical devices and wearable devices. IoT devices exceeded 13.8 billion in 2021 and are expected to double by 2025 [Howarth, 2023]. This world of devices leads to a massive attack surface with a significant increase in the number of entry points for hackers and, correspondingly, security and privacy challenges for researchers and engineers to face.

While the full range and inherent diversity of IoT devices cannot be overestimated, safety critical devices, such as self-driving cars, Industrial Control Systems (ICS) and their security and privacy challenges, risk drawing too much attention to the detriment of inexpensive devices, such as those for home automation. It must be stressed that any home automation issue would be highly impactful due to the enormous use of such devices worldwide, which is also favoured by a general price decrease.

We observe that similar considerations apply to smart bulbs, whose hijacking may have security implications, e.g., help a thief abuse the victim's electricity consumption, then privacy implications, e.g., help the thief profile the victim's patterns of usage hence their habits and, ultimately, safety implications, e.g., ultimately help the thief understand if a household is currently empty or not. Following these observations, this paper rests on the following research question: *What vulnerabilities affect best-seller light bulbs? What consequences do they have on the end user's security, privacy and safety?*

A. Contributions

Our experiments apply the (steps of the) PETIoT kill chain [Bella et al., 2023] to conduct a Vulnerability Assessment and Penetration Testing (VAPT) on the Tp-Link Tapo Smart Wi-Fi Light Bulb, Multicolor (L530E), currently best-seller on Amazon Italy, Fig. 1. The Tapo L530E is a cloud-enabled multicolor Smart Bulb that can be controlled through the Tapo proprietary application. The user needs to install it on an Android or iOS mobile device and then create a Tapo account. The smart bulb uses Wi-Fi technology for connectivity. So, unlike many other smart bulbs requiring a dedicated hub, the user can enjoy the L530E as is by simply connecting it to their home Wi-Fi network.

Contrarily to a potential belief that smart bulbs are not worth protecting or hacking, we found out that this model suffers four vulnerabilities that are not trivial and, most importantly, may have a dramatic impact:

1) Vulnerability 1. *Lack of authentication of the smart bulb with the Tapo app*, 8.8 CVSS score, High severity. The app does not get any guarantee about the identity of its peer. Therefore, anyone can authenticate to the app and pretend to be the smart bulb.
2) Vulnerability 2. *Hard-coded, short shared secret*, 7.6 CVSS score, High severity. The secret used by both the Tapo app and the smart bulb is short and exposed by both the code fragments run by the app and by the smart bulb.
3) Vulnerability 3. *Lack of randomness during symmetric encryption*, 4.6 CVSS score, Medium severity. The ini-

Fig. 1. The Tapo L530E on Amazon.it

tialisation vectors (IVs) used by the Tapo app and the smart bulb are static, and each communication session uses a single, fixed IV for each message.

4) Vulnerability 4. *Insufficient message freshness*, 5.7 CVSS score, Medium severity. Neither the app nor the smart bulb implements appropriate measures to check the freshness of messages that they receive.

Our exploitation experiments of such vulnerabilities demonstrate that a malicious attacker who stands in proximity of the target smart bulb, hence of the Wi-Fi access point to which the bulb is meant to be connected, can exploit the bulb in various ways. Vulnerability 1 means that the attacker impersonates the bulb and receives the user's Tapo credentials as well as the user's Wi-Fi credentials from the Tapo app. To achieve this, the bulb must be in setup mode, when it exposes its own SSID. Alternatively, if the bulb is already configured and working, then the attacker mounts a simple Wi-Fi deauthentication attack against the bulb and repeats it until the user attempts to setup the bulb again to restore it.

The attacker may also interleave another session: by leveraging the credentials just obtained, he impersonates the user through the setup of the bulb and receives a session key from the device, which he may then relay back to the user. Therefore, the attacker effectively mounts a man-in-the-middle attack. Moreover, during device setup, the Tapo app also releases the Wi-Fi credentials to the attacker, thereby causing a clear escalation of the malicious potential for other attacks requiring local access.

Vulnerability 2 means that the attacker can obtain the key that the Tapo app and the smart bulb share and use for the authentication and the integrity of the messages exchanged during the initial discovery phase, described in Section V-A. Thanks to the knowledge of the key, the attacker can violate the integrity and authentication of the messages of this phase.

Vulnerability 3 means that the attacker understands the consequences of certain encrypted messages on the target device despite the fact that he does not understand the precise cleartext inside each message. Therefore, the attacker may attempt to re-use those messages at will to operate the device as each message determines. In combination with vulnerability 4, the attacker is assured that whatever message he replays will be accepted by the bulb, hence a Denial-of-Service (DoS) attack becomes possible.

B. Ethics and Responsible Disclosure

All experiments performed on Tapo L530E only involve devices, Wi-Fi networks, accounts, emails, and passwords owned solely by the authors of this work. During the experiments, no user nor third-party data were accessed.

We dutifully contacted Tp-Link via their Vulnerability Research Program (VRP), reporting all four vulnerabilities we found. They acknowledged all of them and informed us that they started working on fixes both at the app and at the bulb firmware levels, planning to release them in due course.

C. Paper Summary

This paper continues with a short account of the related work (§II). It then unfolds the six phases of the chosen methodology, i.e., the PETIoT kill chain, on the chosen target, i.e., the Tapo L530E (§III through §VIII). Finally, it draws the relevant conclusions (§IX).

II. RELATED WORK

The security and privacy aspects of IoT devices are becoming more and more important and, correspondingly, the related work concerning such devices is very wide. Due to space constraints, this section is reduced to the literature entries that are most relevant to the core of this paper.

In 2021, it was shown that printers are common devices whose networked use may suffer at least three attacks [Bella and Biondi, 2019]. The first attack, *zombies for traditional DDoS*, shows how some printers may suffer from vulnerabilities that would transform them into exploitable zombies. The second attack, *paper DoS*, shows how a large number of printers are found to honour unauthenticated printing requests. The third attack, *privacy infringement*, shows how these devices bear a remarkable risk of data breach. Later, the same authors contributed to similar experiments against VoIP phones [Biondi et al., 2020]. Our work follows a similar methodology.

In 2022, a literature survey for understanding IoT security threats and challenges appeared [Nath and Nath, 2022]. It is divided into three parts. The first part contains an analysis of the main threats and attacks, also analysing the IoT ecosystem from four perspectives: devices, internal network services, external network services and users. The second part contains a study on recent IoT malware attacks. Finally, the general security requirements and challenges to address the devised attack categories are discussed. This work was inspirational for our experiments.

PETIoT is a recent cyber Kill Chain (KC) specifically developed to guide VAPT sessions over IoT devices [Bella et al.,

2023] with a focus on detecting their network vulnerabilities. The KC is demonstrated on the Tapo C200 IP camera by Tp-Link, enabling the discovery of three severe vulnerabilities: *Improper neutralisation of inbound packets*; *Insufficient entropy in encrypted notifications*; *Cleartext transmission of video stream*. Each of the vulnerabilities found is exploited through different attack scenarios, and appropriate remediation measures are illustrated as possible fixes. We followed PETIoT strictly through our work, as can be seen by the steps discussed below.

In 2018, a penetration testing session on the *Tradfri* smart bulb produced by *IKEA* was reported [Dalvi et al., 2018]. The article describes four different attacks. *Hacking Smart Light Bulb via Bluetooth* allows the attacker to control the bulb. *IKEA Tradfri Gateway Exploit* allows the attacker to compromise the ZigBee Tradfri gateway by managing to break the firmware update. *Denial of Service Attack* allows the attacker to perform a DoS attack against the gateway by flooding it with UDP packets. *Identity Spoofing* allows the attacker to impersonate the CoAP client and control the smart bulbs. This work is closely related to ours but revolves around the ZigBee protocol and does not achieve escalation to local Wi-Fi access.

III. EXPERIMENT SETUP

We begin defining the experiment setup, accordingly with the PETIoT kill chain. Our setup is non-invasive, as the smart bulb is not tampered with. It includes:

- A Wi-Fi switch to provide local connectivity.
- A Smart bulb Tapo series L530 with Hardware Version 1.0.0 and Firmware Version 1.1.9.
- A Samsung smartphone running Android 11 and the Tapo app Version 2.8.14.
- An Ubuntu 22.04 machine with 5.15.0-47 kernel to run all software needed for the experiments.

To be able to use all the bulb features, it is necessary to create a Tapo account and login into it. Depending on whether the smart bulb is already associated with a Tapo account or not, and the network configuration used, we identify three different setups. In all three setups, the smartphone has access to the Tapo account to which the smart bulb is associated, or to which the user wants to associate it.

A. Setup A

The context of the *Setup A* is as follows.

- The user has previously associated her smart bulb with the Tapo app on her phone. During the association process, the user connects the smart bulb to a certain network X.
- The Ubuntu device and the user's phone are both connected to the same network Y as shown in Fig. 2 — this is not necessarily the network X to which the Tapo L530E is connected.

This setup does not require the Ubuntu device to have access to the network to which the smart bulb is connected, but only to be connected to the same network as the Tapo app.
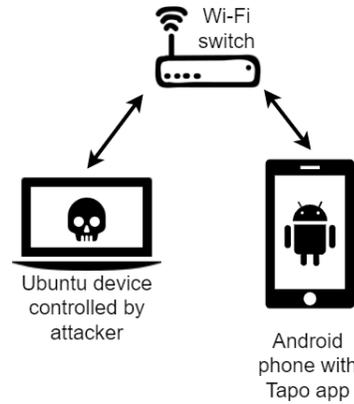

Fig. 2. Setup A, network without a local smart bulb

Therefore, the user could, for example, connect to a public network to turn off a light that may have been mistakenly left on at home.

B. Setup B

The context of the *Setup B* is as follows.

- The user has previously associated her smart bulb with the Tapo app on her phone. During the association process, the user connects the smart bulb to a certain network X.
- The Ubuntu device and the user's phone are both connected to the same network X to which the smart bulb is connected, as shown in Fig. 3.
- The Ubuntu device has complete control of the network X. It is able to carry out an *ARP spoofing attack*, i.e., to intercept data frames, modify the network traffic or prevent it from reaching its intended recipient.
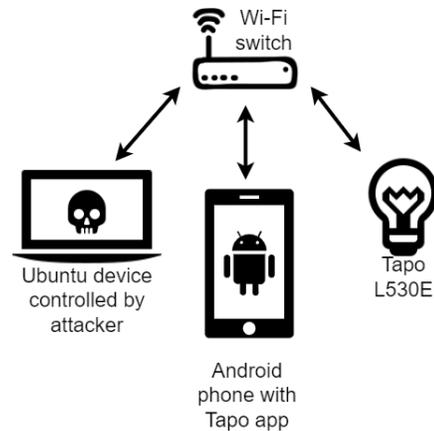

Fig. 3. Setup B, network with a configured smart bulb

This setup requires the active presence of the smart bulb. In addition, the setup requires the Ubuntu device to have access to the network to which the smart bulb is connected, typically the victim's local network. Therefore, all devices used are connected via Wi-Fi to the same access point.

C. Setup C

The context of the *Setup C* is as follows.

- The user wants to associate the newly reset or not yet configured smart bulb with her Tapo account.
- The Tapo app (hence, the user) believes to be connected to the network X created by the smart bulb, but it is actually connected to a network Y controlled by the Ubuntu device.
- The Ubuntu device is connected both to the network created by the smart bulb and to the network controlled by itself.

This setup occurs when the smart bulb has been reset or has not been configured yet. As detailed in Section V-C1, the smart bulb starts a public access point to which the user must connect to complete the setup. The network configuration is shown in Fig. 4.
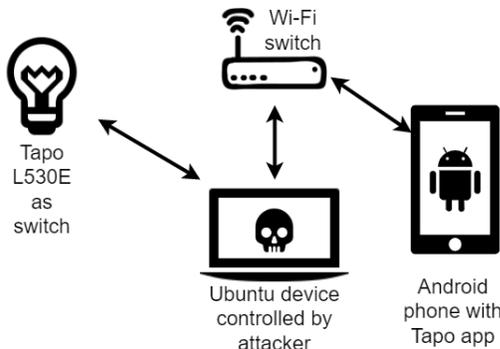

Fig. 4. Setup C, network with a non-configured bulb

The *Wi-Fi deauthentication attack* allows the attacker to easily get *Setup C*. A *Wi-Fi deauthentication attack* is a type of denial-of-service attack that targets communications between a device and a Wi-Fi wireless access point [Kristiyanto and Ernastuti, 2020]. Thanks to this attack, the attacker can deauthenticate the smart bulb from the network to which it is connected, forcing the user to repeat its setup process. The *Wi-Fi deauthentication attack* requires the access point and connected devices not to use 802.11w or WPA3. Although in literature there are other theoretically feasible methods to perform the deauthentication attack anyway [Schepers et al., 2022], we did not test them. In *Setup C*, we assume that the network to which the smart bulb is connected is a network where the deauthentication attack can be performed. One way to mount it would be to leverage *Aircrack-ng* [d'Otreppe de Bouvette, 2023], as described in a blog [Alopix, 2023]. The attack requires the adversary to know the victim device's MAC address, however, it is possible to list all MAC addresses connected to nearby access points by simply listening to the traffic, so the adversary has a short and finite list of addresses to try.

After the victim resets the smart bulb, it enters setup mode. In *Setup C*, the Ubuntu device needs the Tapo app to connect to the network it controls and not to the network started by the smart bulb — hence, the adversary performs another *Wi-Fi deauthentication attack* to deauthenticate the phone running the Tapo app from the network started by the smart bulb, arguably inducing the victim to re-attempt to connect, eventually using the adversary-controlled network with the same SSID. As the network started by the smart bulb is an unprotected Wi-Fi 4 (802.11n) network, in this latter case the deauthentication attack will always work. Intuitively, this setup assumes that the Ubuntu device is close enough to the network started by the smart bulb to hear the emitted carrier.

IV. INFORMATION GATHERING

After the setup we can proceed to the collection of information. The following tools are installed on the Ubuntu device:

- *Ettercap* [Ettercap, 2023], a suite of tools to perform MITM attacks used on the Ubuntu device to control the network.
- *Wireshark* [Wireshark, 2023], a network protocol analyser (or packet sniffer) used to capture and analyse traffic.

The smart bulb firmware source code is not publicly available. Therefore, the offensive activities are performed in black-box mode.

Thanks to the messages captured with *Wireshark* it is possible to distinguish between three different types of communications. They differ according to the sender and the receiver of the various messages belonging to them. The three different types of communications we identified are: *(a)* communications *App-Cloud*, including all messages exchanged between the Tapo App and the Cloud Server; *(b)* communications *Bulb-App*, including all messages exchanged between the smart bulb and the Tapo App; *(c)* communications *Cloud-Bulb*, including all messages exchanged between the Cloud Server and the smart bulb. When the phone running the Tapo app is not connected to the same network as the light bulb, communications happen through the cloud, combining *App-Cloud* and *Cloud-Bulb* communications – otherwise, they take place locally with *Bulb-App* communications.

All communications through the cloud, i.e., *App-Cloud* and *Cloud-Bulb*, are encrypted. They take place through the use of a secure TLS channel that ensures authenticity, integrity, and confidentiality of the messages, even though they are transmitted over the Internet. Instead, all *Bulb-App* communications are exchanged via HTTP messages. Their payloads are encrypted using the AES128 block cipher in CBC mode. The initialisation vector and the cryptographic key used for this protocol are exchanged using the Tapo "Symmetric Key Exchange Protocol" (TSKEP), which is described below in Section V-B. This protocol only generates traffic over the local Wi-Fi network, not through the Internet.

Our offensive activities only target *Bulb-App* communications, and ignore the others.

V. TRAFFIC ANALYSIS

Thanks to the information obtained during the previous phase, it is now possible to analyse the network traffic profitably.

The API exposed by the smart bulb is very similar to a *Remote Procedure Call* (RPC). The JSON sent by the app to the smart bulb contains the name of the method to be invoked and its parameters. The responses sent by the smart bulb to the app contain an *error code* representing the outcome of the operation (whether it was successfully executed or the error occurred) with an optional result. All JSONs are exchanged via the payloads of HTTP messages.

Before the Tapo app starts using the API exposed by the bulb, it must perform two preliminary steps: locate the smart bulb within the network to which it is connected, and exchange a symmetric key with the smart bulb to encrypt messages. Therefore, the communication between smart bulb and Tapo app can be summarised in three macro-steps:

- *Bulb Discovery* – it allows the Tapo app to locate the smart bulb within the local network and to get the smart bulb's current configuration.
- *Tapo Symmetric Key Exchange Protocol* – executing the TSKEP protocol allows the Tapo app and smart bulb to exchange a symmetric key.
- *Smart bulb usage* – it consists in using the actual smart bulb app protocol.

These macro-steps are described in detail below.

A. Bulb Discovery

Smart bulb configurations may change over time. For example, the IP address assigned to it by the DHCP server might have changed, it might start using a different encryption scheme after an update, or it might have just been reset. Before the Tapo app can start communicating with the smart bulb, it must locate the smart bulb within the local network and get its current configuration. To do this, the Tapo app connects to the UDP service listening on the 20002 port of the smart bulb, as represented in Fig. 5.
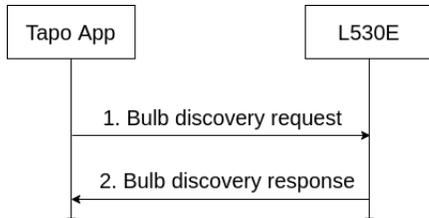

Fig. 5. Tapo (local) Device dicovery

The payload format of such UDP messages is represented in Table I. It can be explained as follows:

- Bytes in positions [0:3] and [6:7] are static. Their values never change.
- The *data length* field contains the length of the *data* field, which contains the data exchanged between Tapo app and smart bulb. When the data field is empty (i.e., when it has length 0) the bits of this field are all set to 0.
- The *nonce* field contains a 4-byte string randomly generated by the Tapo app. The string is inserted in the *bulb discovery request* message from the Tapo app and quoted

by the smart bulb in the *bulb discovery response* message from the smart bulb. This allows the Tapo app to obtain guarantees about the freshness of the response received from the smart bulb.

- Logically, the *checksum* field must be excluded from the computation of the checksum itself — instead, a secret key of the same length is used in its place for the checksum computation. This secret key is hard-coded in both the Tapo app and the smart bulb, hence, the checksum acts like a Message Authentication Code (MAC). This field allows the receiver to understand that the message arrives intact as it is sent. The algorithm used for the *checksum* calculation is the Cyclic Redundancy Check 32 (CRC-32), a checksum algorithm that hashes byte sequences to 32 bit values.
- The *data* field contains different information depending on whether it is a *bulb discovery request* or a *bulb discovery response* and whether the smart bulb is configured or not.

		Octet			
		0	1	2	3
Offset	0	0x02	0x00	0x00	0x01
	4	Data length		0x11	0x00
	8	Nonce			
	12	Checksum			
	...	Data			

TABLE I
FORMAT OF THE PAYLOAD OF UDP MESSAGES

B. Tapo Symmetric Key Exchange Protocol

This step uses the RPC API exposed by the smart bulb on port 80. All exchanged messages are HTTP messages, so at the transport layer, only the TCP protocol is used. The TSKEP, represented in Fig. 6, allows Tapo app and smart bulb to exchange a 128-bit AES key and an IV to encrypt the payload of various HTTP messages. The protocol consists of four different messages:

1) RSA public key transmission
2) AES key transmission
3) Login
4) Token transmission

At the end of this phase, Tapo app and smart bulb get a short-term shared secret to encrypt all subsequent traffic. Thanks to the credential stored during the setup phase, for the smart bulb the secret is also authenticated.

C. Smart Bulb Setup

This section focuses on the two possible configurations for the smart bulb and specifies them in depth following our traffic analysis efforts.

1) Smart bulb turned on and not configured: Before a smart bulb Tapo L530E can be used, it must be associated with a Tapo account. There are two reasons why a smart bulb may not be associated with any accounts: because it has been reset,
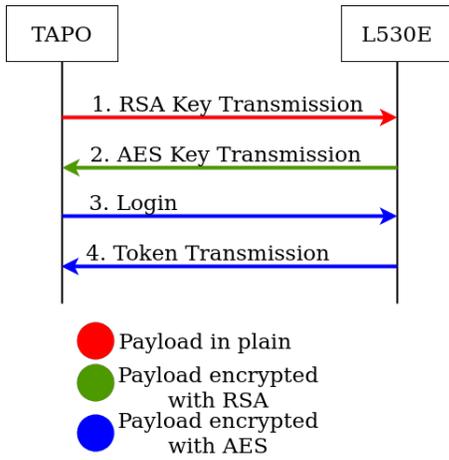
Fig. 6. Tapo Symmetric Key Exchange Protocol

or it has not been configured yet. In this Section, we discuss the process of associating the smart bulb with a Tapo account.

An unconfigured or newly reset smart bulb starts a public access point with SSID *Tapo_Bulb_XXXX*, where *XXXX* are four decimal places. The smart bulb also acts as a switch within the network it generates. In order to configure it and associate it with their Tapo account, the user must connect to the Wi-Fi network started by the smart bulb itself.

After that, the Tapo app tries to locate the smart bulb. To do this, it starts sending *bulb discovery request* messages to broadcast. In this case, the *data* field of these messages is empty. After the identification of the smart bulb, the Tapo app starts the TSKEP protocol with it. The values set in the login message as `username` and `password` are fixed values that the Tapo app uses every time it configures a new device.

Once the symmetric key is obtained, Tapo app sends to the smart bulb the SSID and the password of the Wi-Fi network to which the smart bulb must connect. The Tapo app also sends the credentials of the Tapo account to which it must be associated. The credentials are then stored by the smart bulb. Through these credentials, smart bulb is able to authenticate all subsequent requests of Tapo app. At this point, the smart bulb turns its access point off and connects to the specified Wi-Fi network. The smart bulb starts communicating with the cloud server to complete its setup. Hence, the Wi-Fi network to which the smart bulb connects must have Internet access.

2) Smart bulb already configured: Let us now consider the case where the smart bulb is associated with a Tapo account and it is ready to be used. As mentioned before, it can be controlled either locally or remotely via the Tapo cloud. To do so, the Tapo application initially tries to locate the smart bulb within the network with a *bulb discovery request* message. If it detects it (i.e., it gets a response) then the interaction happens locally via Bulb-App communications, which are the subject of our analysis. Otherwise, if the Tapo app does not receive any valid *bulb discovery response*, then it tries to check the smart bulb remotely. If the smart bulb is not detectable even remotely then it is determined offline.

VI. VULNERABILITY ASSESSMENT

The assessment following the information gathered so far highlights four vulnerabilities.

a) Vulnerability 1 – Lack of the smart bulb authentication with the Tapo app.: Improper Authentication [MITRE, 2006a] in Tapo L503E allows an adjacent attacker to impersonate the Tapo L530E with the Tapo app during the TSKEP step.

In the TSKEP step, unlike the *Bulb Discovery* step, the protocol used to exchange the session key does not give the Tapo app any evidence of its peer's identity. Hence, an attacker is able to authenticate to the Tapo app as the Tapo L530E or as another device: in fact, this vulnerability is present in all Tapo smart devices that use the TSKEP protocol.

The *CVSS v3.1* score that we calculate is 8.8, meaning *High* severity. Precisely: Attack Vector: Adjacent; Attack Complexity: Low; Privileges Required: None; User Interaction: Required; Scope: Changed; Confidentiality: High; Integrity: High; Availability: High. In particular, Attack Complexity is low because the attacker could impersonate the bulb by implementing the protocol messages to respond to the calling app. Following that, he could obtain the user password on the Tapo app, then fully impersonate the user and manipulate at will any target Tapo device of the same user. Precisely, by impersonating the bulb at setup time as explained above, the attacker would receive the victim's Wi-Fi SSID and password from the Tapo app, so that he could then impersonate the user by her password at each session with the target device, which could be any Tapo device of the user's. The attacker could also obtain the device-chosen session key, which he could then relay to the user's genuine app and effectively interpose.

b) Vulnerability 2 – Hard-coded short checksum shared secret.: Protection Mechanism Failure [MITRE, 2008] in Tapo L503E allows an adjacent attacker to obtain the secret used for authentication during the *Bulb Discovery* phase.

The shared secret used for *Bulb Discovery*'s messages authentication is short and hard-coded both in the Tapo app and in the Tapo L530E. Therefore, it can be obtained in the following ways:

1) Brueforcing, because of its shortness.
2) Decompiling the Tapo app.

The *CVSS v3.1* score that we calculate is 7.6, meaning *High* severity. Precisely: Attack Vector: Adjacent; Attack Complexity: Low; Privileges Required: None; User Interaction: Required; Scope: Unchanged; Confidentiality: Low; Integrity: High; Availability: High. The knowledge of the shared secret provides to the attacker the ability to edit and to create *Bulb Discovery* messages. Specifically, the attacker is able to generate fake *bulb discovery requests* to locate all smart bulbs, or generally Tapo devices using the same protocol, connected within the same network. Meanwhile, the ability to edit valid messages allows the attacker to edit the *bulb discovery response* messages sent from the smart bulb to the Tapo app.

c) Vulnerability 3 – Lack of randomness during symmetric encryption.: Use of a Cryptographic Primitive with a Risky

Implementation [MITRE, 2020] in Tapo L503E allows an adjacent attacker to make the cryptographic scheme deterministic.

The IV used in AES128-CBC scheme is generated together with the key and remains unchanged for the entire life of the key. Therefore, the encryption of the same messages produces the same ciphertext.

The *CVSS v3.1* score that we calculate is 4.6, meaning *Medium* severity. Precisely: Attack Vector: Adjacent; Attack Complexity: Low; Privileges Required: None; User Interaction: Required; Scope: Unchanged; Confidentiality: Low; Integrity: None; Availability: Low. When the user interacts with the device thereby generating traffic, the attacker can distinguish repeated messages without deciphering them, yet infer what messages lead to what consequences, such as turning the bulb off. He could then manipulate the bulb by repeating those messages.

d) Vulnerability 4 – Insufficient message freshness.: Predictable from Observable State [MITRE, 2006b] in Tapo L503E allows an adjacent attacker to replay messages that are considered valid both from the Tapo L530E and the Tapo app.

The smart bulb and the Tapo app do not check the freshness or the duplicity of the received messages. They only check that the session key with which the messages are encrypted is still valid, i.e., not older than 24 hours.

The *CVSS v3.1* score that we calculate is 5.7, meaning *Medium* severity. Precisely: Attack Vector: Adjacent; Attack Complexity: Low; Privileges Required: None; User Interaction: Required; Scope: Unchanged; Confidentiality: None; Integrity: None; Availability: High. Similarly to the previous vulnerability, the attacker can leverage user-generated traffic, this time to replay messages that both the bulb and the app will certainly accept because of the lack of appropriate freshness measures.

VII. EXPLOITATION

This Section shows how an attacker can exploit the vulnerabilities we found in a real environment. We show 5 attack scenarios, in which the attacker exploits one or more vulnerabilities to achieve their malicious goals. We validated each attack scenario by manually executing the steps illustrated in the forthcoming sections, hence all reported attacks are feasible in practice. As noted above through the CVSS scores, their likelihood is determined by the Adjacent Attack Vector, the Low Attack Complexity, the No Privileges Required and the Required User Interaction.

A. Attack scenario 1 - Fake Bulb Discovery messages generation

In this experiment we exploited:

- *Vulnerability 2*, with the goal of getting the ability to create fake *Bulb Discovery* messages.

This experiment can be conducted in every scenario presented in Section III. For this attack, it is first necessary to get hold of a UDP message. In our case, we have chosen to use a real *bulb discovery request* message, because they are easy to get. These messages are broadcast by the Tapo app

every time it is opened, regardless of the network to which it is connected. To get a valid one, we just capture the traffic using Wireshark and use a filter to extract all UDP messages sent in broadcast e.g., by using the filter `udp && ip.dst == 255.255.255.255`. Once a UDP message is obtained, we can perform an offline brute-force attack to get the secret shared between the Tapo app and the smart bulb. At this point, neither the smart bulb nor the Tapo app needs to be active anymore to complete the attack. In our setup, the brute-force attack always succeeded in 140 minutes on average.

This grants the adversary the ability to create fake *bulb discovery request* and *response* messages: the former allows the attacker to identify all Tapo devices that use the same protocol and the same key, on any network he connects to, while the latter allows the attacker to respond to the Tapo app's request messages with false configurations.

B. Attack scenario 2 - Password exfiltration from Tapo user account

In this experiment we exploited, in order:

- *Vulnerability 2*, with the goal of getting the ability to create fake *bulb discovery response* messages,
- *Vulnerability 1*, with the goal of getting the ability to authenticate as the Tapo L530E to the Tapo app.

The context in which we conduct the experiment is Setup A described in Section III-A. To carry out this attack, the adversary needs to obtain the *ownerId* of the victim Tapo account, and the UDP port to which the victim sends her *bulb discovery request* messages (in our case, port 20002). The attack diagram is shown in *Fig. 7*.
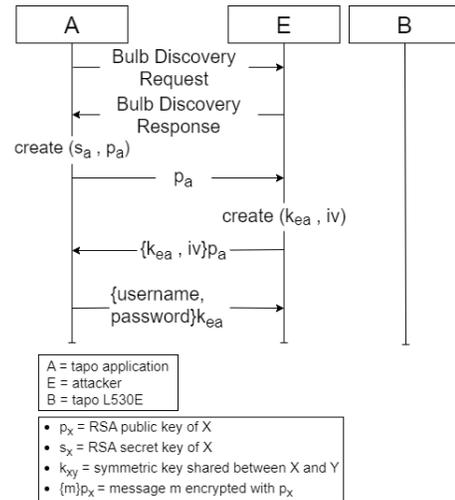

Fig. 7. Sequence chart for the attacker's impersonation of the bulb

The exploitation begins the moment the victim opens her own Tapo app. When the app is open, it starts broadcasting *bulb discovery request* messages. During the attack, the attacker exploits his ability to create fake *bulb discovery response* messages to respond to various *bulb discovery request*

from the victim. The attacker sets the *bulb discovery response* messages fields as shown in Listing 1:

- It sets the *owner* field to the *ownerId* of the victim. This is to make the Tapo app think there is a device of its own on the network to start TSKEP.
- It sets the *ip* and *port* fields to point to an adversary-controlled server.

```
{
 "result": {
   "device_id": "bd1e...9348",
   "owner": "Victim's ownerId",
   "device_type":
     "SMART.TAPOBULB",
   "device_model":
     "L530E Series",
   "ip": "Attacker's IP",
   "mac": "Attacker's MAC",
   "factory_default": false,
   "is_support_iot_cloud": true,
   "mgt_encrypt_schm": {
     "is_support_https": false,
     "encrypt_type": "AES",
     "http_port": 80
   }
 },
 "error_code": 0
}
```

Listing 1. JSON attack scenario 2

After receiving the response, the Tapo app thinks that it has successfully completed the *Bulb Discovery* phase by locating its own device within the network. Therefore, it starts the TSKEP protocol with the attacking device. Because of vulnerability 1, the TSKEP protocol does not give the Tapo app any evidence about the identity of the interlocutor. For this reason, the Tapo app assumes that the newly received key is shared with an associated device, while it is shared with the attacker instead. Hence, the adversary is able to decrypt the *Login* message of the TSKEP protocol and get the password and the hash of the email of the victim's Tapo account.

The attack can be summarised as follows:

- The attacker gets the *Bulb Discovery* shared key and creates fake *bulb discovery response* messages. Therefore, the authentication of the *bulb discovery response* message fails.
- The Tapo app executes the TSKEP protocol with the attacker instead of the smart bulb. Therefore, authentication of the smart bulb with the Tapo app fails. This results in an *integrity loss*.
- The Tapo app shares the key with the attacker, hence the distribution of the session key fails. This results in an *availability loss*.
- The attacker can violate the confidentiality of the messages and get the password and the hash of the email of

the victim's Tapo account. This results in a *confidentiality loss*.

C. Attack scenario 3 - MITM attack with a configured Tapo L530E

In this experiment we exploited:

- *Vulnerability 1*, with the goal of getting the ability to authenticate as the Tapo L530E to the Tapo app.

The context in which we conduct the experiment is Setup B described in Section III-B. The attacker makes independent connections with the victims and relays messages between them to make them believe they are talking directly to each other over a private connection. When the Tapo app starts the TSKEP with the smart bulb, the attacker intercepts the *RSA key transmission* message and blocks its reception from the smart bulb. In parallel, he starts a new session with the smart bulb from which he gets a new session key. The session key received by the attacker from the smart bulb is then encrypted with the previously received RSA public key and sent to the Tapo app. Due to vulnerability 1, the Tapo app expects to share the received key only with the smart bulb, but it is sharing it with the adversary instead. Hence, the attacker now has the key that Tapo app and smart bulb use to encrypt all subsequent communication messages. Therefore, he is capable of deciphering them and violating their confidentiality and integrity, for example by decrypting the messages, modifying their contents, re-encrypting and then forwarding them. This is summarised in the attack diagram shown in *Fig. 8*.
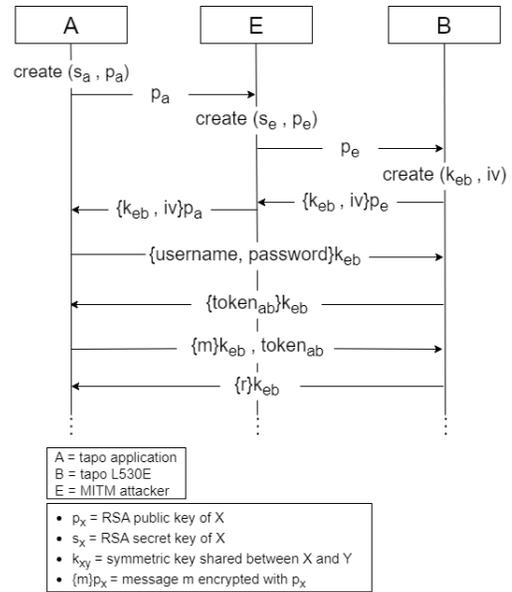

Fig. 8. Sequence chart for the attacker's MITM between app and bulb

Hence, the attack consists of the following steps:

- The Tapo app executes the protocol with the attacker instead of the smart bulb, therefore, the authentication process fails.

- The Tapo app shares the key with the attacker and not with the smart bulb, hence, the distribution of the session key fails.
- The attacker authenticates the key shared with the smart bulb thanks to the credentials received from the Tapo app, therefore, the authentication of the Tapo app with the smart bulb fails.
- The attacker relays messages between the two sessions to make Tapo app and smart bulb believe they are talking to each other, hence, the *confidentiality* and the *integrity* of messages is lost.

D. Attack scenario 4 - Replay attack with the Smart bulb as victim

In this experiment we exploited:

- *Vulnerability 4*, with the goal of getting the ability to replay both old and replicated messages.

The context in which we conduct the experiment is Setup B described in Section III-B. This attack scenario is divided into three phases. Specifically, in this case, it is also necessary that the attacker has a line of sight on the bulb to complete the attack.

During the first phase, *Wireshark* is used to *sniff* the traffic of a *Bulb-App* communication. During this communication, the app sends to the smart bulb both *get* messages, to request the value of some status parameters, and *set* messages, to request the smart bulb to change the value of some of its internal parameters. During the experiment, we are not aware of the symmetric key used by both the smart bulb and the Tapo app to encrypt the messages.

During the second phase of the experiment, we replicate all messages we previously captured. Hence, for each message we determine whether it was a *get* or *set* message simply by observing how the bulb behaves after each message. For every *set* message, we take note of the change that it caused on the light bulb.

During the third phase, we arbitrarily replicate the *set* messages to the smart bulb, managing to make changes to its internal state, without having a Tapo account associated with it. Messages continue to be accepted by the smart bulb until the session key with which they are encrypted expires.

E. Attack scenario 5 - MITM attack with an unconfigured Tapo L530E

In this experiment we exploited:

- *Vulnerability 1*, with the goal of getting the ability to authenticate as the Tapo L530E to the Tapo app.

The context in which we conduct the experiment is Setup C described in Section III-C. This attack scenario exploits the fact that not only the Tapo app, but anyone (including the attacker), can connect to the Wi-Fi network started by the smart bulb during the setup phase. It is important that the attacking device acts as a bridge between the two networks. The attacker must flow all *bulb discovery request* messages from the network it controls to the smart bulb network, and vice versa for *bulb discovery response* messages. Otherwise,

the Tapo app would not be able to detect the smart bulb and therefore would never try to start the TSKEP with it. Subsequently, *vulnerability 1* is exploited in the same way shown in Section *Attack Scenario VII-C*, so the adversary is able to violate the confidentiality of the session key between the smart bulb and the Tapo app.

At some point in the communication, the Tapo app sends the JSON shown in Listing 2:

```
{
 "method":"set_qs_info",
 "params":{
  "account"{
   "password":"Tapo password",
   "username":"Tapo email"
  },
  "extra_info":{"specs":"EU"},
  "time":{"region":"Europe/Rome","
     time_diff":60,"timestamp":1660032435
     },
  "wireless":{
   "key_type":"wpa2_psk",
   "password":"Wi-Fi password",
   "ssid":"ssid Wi-Fi"}
  },
  "requestTimeMils":1660032438365,
  "terminalUUID":"..."}'
```

Listing 2. JSON attack scenario 5

Because usernames, Tapo passwords, SSIDs, and Wi-Fi passwords are sent in base64 encoding, the attacker is able to decode and steal them.

VIII. FIXING

This Section outlines possible mitigations for the identified vulnerabilities, marking the last step of the PETIoT kill chain. Most measures consist of simple modifications to the relevant protocols, in particular to strengthen the cryptographic measures. These can be implemented via software updates to be pushed to the affected devices via Internet, so we believe these fixes to be easily deployable with the already existing update procedures

a) Fix for Vulnerability 1.: This vulnerability is the most complex and dangerous. It is not easy to find a simple fix to it because the protocol should be widely revised. Our proposed fix requires the smart bulb to sign the message of *AES key transmission* with an asymmetric, private key. The validity of that key as to belong to the bulb could be verified by the app via a digital certificate to retrieve from the Cloud Server during the association of the bulb with the app. Of course, such a certificate should chain up to a root certificate to be securely stored with the app. All this would allow the app to get evidence about the authenticity of the response, i.e., that the response really comes from the smart bulb. In consequence, the app will eventually store all certificates of the associated devices.

b) Fix for Vulnerability 2.: One possible solution to fix this vulnerability is the active presence of the cloud server. This entity should periodically assign each Tapo account a fresh key to use when calculating the checksum within *Bulb Discovery* messages. The key assigned to a Tapo account should then be communicated to all devices associated with it. The benefits of the fix can be summarized as follows:

- The key is not hard-coded, so the attacker would no longer be able to get it by decompiling the Tapo app or the firmware of a Tapo device.
- Each account has its own key, therefore, compromising a Tapo account, or a key, would not result in compromising the keys of other Tapo accounts.
- The key should be long and random enough by current standards so that brute-force attacks would not be profitable anymore.
- The key is always fresh, so even if an attacker were to get the key of a Tapo account, the latter would not be compromised forever, but only until the validity of the stolen key expires and the cloud server assigns a new key to it.

It would also be useful to use a collision-resistant cryptographic hash function for the checksum. Examples of cryptographic hash functions are SHA-224 or SHA3-224.

c) Fix for Vulnerability 3.: This vulnerability can be fixed by making the IV dynamic, i.e., using different IVs to encrypt different messages. This should be done by both the Tapo app and the Tapo L530E. The IV used to encrypt the JSON contained in the *params* field could then be included as a field in the plain part of JSON contained in *Bulb-App* communications.

d) Fix for Vulnerability 4.: The timestamp containing the message creation moment included in JSON should be verified by smart bulb and the Tapo app. Checking the creation timestamp would prevent recent messages from being passed off as fresh by an attacker. In addition, the various messages exchanged should contain a sequence number, which would prevent duplicate messages.

IX. CONCLUSIONS

We identified four vulnerabilities in the Tapo L530E, which we were able to practically exploit in five different attack scenarios with varying impacts on the users' security, privacy and safety. Of the four vulnerabilities, two are of High severity and two are of Medium severity, according to their CVSS score.

Overall, we observe that the experiment setup had to be designed with care due to the three scenarios that were possible. Following that, the information gathering step was rather large and complicated, much more than it could be reported in this paper due to space constraints. The vulnerability assessment was very surprising. For example, while deauthentication is routinely possible, we were not prepared to discover passwords in the clear and weak cryptography. Exploiting the vulnerabilities was moderately challenging but devising appropriate fixes was harder.

One way to interpret such findings could be that "small" IoT devices may have raised insufficient cybersecurity attention thus far, i.e., insufficient cybersecurity measures due to a preconception that they may not be worth hacking or exploiting. Our work pins down this preconception as wrong, at least because the scope of our attacks expands onto all devices of the Tapo family a victim may use and, most importantly, potentially onto the entire victim's Wi-Fi network, which the attacker is enabled to penetrate.

While more and more experiments will certainly follow on similar bulbs and other inexpensive devices, we argue that the evidence we have gathered thus far is sufficient to call for a fuller application of a zero trust model to the IoT domain. With dozens of years of cybersecurity experience accumulated by the international community thus far, it should be possible to find affordable ways to achieve that in due course.

REFERENCES

[Alopix, 2023] Alopix (2023). Wifi deauth attack in 2 minutes! https://systemweakness.com/wifi-deauth-attack-in-2-minutes-d1fb55112305.

[Bella and Biondi, 2019] Bella, G. and Biondi, P. (2019). You overtrust your printer. In *Proc. of the 2nd International Workshop on Safety, securiTy, and pRivacy In automotiVe systEms (STRIVE'19)*, LNCS 11699, pages 264–274. Springer.

[Bella et al., 2023] Bella, G., Biondi, P., Bognanni, S., and Esposito, S. (2023.). PETIoT: PEnetration Testing the Internet of Things. *Elsevier Internet of Things*, 22.

[Biondi et al., 2020] Biondi, P., Bognanni, S., and Bella, G. (2020). VoIP Can Still Be Exploited — Badly. In *Proc. of the 5th International Conference on Fog and Mobile Edge Computing, (FMEC'20)*, pages 237–243. IEEE.

[Dalvi et al., 2018] Dalvi, A., Maddala, S., and Suvarna, D. (2018). Threat modelling of smart light bulb. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–4.

[d'Otreppe de Bouvette, 2023] d'Otreppe de Bouvette, T. (2023). Aircrackng. https://www.aircrack-ng.org/.

[Ettercap, 2023] Ettercap (2023). Ettercap project. https://www.ettercap-project.org/.

[Howarth, 2023] Howarth, J. (2023). 80+ amazing iot statistics (2023-2030). https://explodingtopics.com/blog/iot-stats.

[Kristiyanto and Ernastuti, 2020] Kristiyanto, Y. and Ernastuti, E. (2020). Analysis of deauthentication attack on ieee 802.11 connectivity based on iot technology using external penetration test. *CommIT (Communication and Information Technology) Journal*, 14(1):45–51.

[MITRE, 2006a] MITRE (2006a). Cwe-287: Improper authentication. https://cwe.mitre.org/data/definitions/287.html.

[MITRE, 2006b] MITRE (2006b). Cwe-341: Predictable from observable state. https://cwe.mitre.org/data/definitions/341.html.

[MITRE, 2008] MITRE (2008). Cwe-693: Protection mechanism failure. https://cwe.mitre.org/data/definitions/693.html.

[MITRE, 2020] MITRE (2020). Cwe-1240: Use of a cryptographic primitive with a risky implementation. https://cwe.mitre.org/data/definitions/1240.html.

[Nath and Nath, 2022] Nath, R. and Nath, H. V. (2022). Critical analysis of the layered and systematic approaches for understanding iot security threats and challenges. *Computers and Electrical Engineering*.

[Schepers et al., 2022] Schepers, D., Ranganathan, A., and Vanhoef, M. (2022). On the robustness of wi-fi deauthentication countermeasures. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '22, page 245–256, New York, NY, USA. Association for Computing Machinery.

[Wireshark, 2023] Wireshark (2023). Wireshark project. https://www.wireshark.org/.