Do You “Relay” Want to Give Me Away? – Forensic Cues of Smart Relays and Their IoT Companion Apps

By:
Maximilian Eichhorn, Gaston Pugliese

From the proceedings of
The Digital Forensic Research Conference
DFRWS APAC 2024
Oct 22-24, 2024

Forensic Science International: Digital Investigation 50 (2024) 301810

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS APAC 2024 - Selected Papers from the 4th Annual Digital Forensics Research Conference APAC

Do You "Relay" Want to Give Me Away? – Forensic Cues of Smart Relays and Their IoT Companion Apps

Maximilian Eichhorn [*], Gaston Pugliese

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

ARTICLE INFO

Keywords:
Smart relays
IoT device forensics
Smart home forensics
Hardware forensics

ABSTRACT

As IoT devices become more prevalent in everyday environments, their relevance to digital investigations increases. The product class of "smart relays", which are connected to the low-voltage grid and usually installed in sockets behind walls, has not yet received much attention in the context of smart home forensics. To close a category-specific gap in the device forensics literature, we conducted a multi-device analysis of 16 smart relays from 9 manufacturers, which support six different companion apps in total. Our examination shows that forensic artifacts can be found locally on the smart relays and in the companion app data, as well as remotely on cloud servers of the vendors. Based on our findings, we developed a Python framework to extract forensic artifacts automatically from obtained firmware dumps, from companion app data, and from captured network traffic.

1. Introduction

The number of connected Internet of Things (IoT) devices has risen steadily in the last years and is forecast to continue growing (Vailshery, 2024a). In 2022, the number of connected devices amounted to around 13.1 billion with 58.6 % of them being consumer devices (Vailshery, 2024b). This widespread use of IoT devices underlines the importance to look more closely at them from a technical perspective due to their forensic relevance as "invisible witnesses" (Urquhart et al., 2022) to digital investigations, especially in smart homes (Servida and Casey, 2019).

Unlike many prominent types of smart devices which can be visibly perceived in modern households, such as smart TVs, speakers, light bulbs, cameras, doorbells, or thermostats, *smart relays* belong to a rather invisible category of IoT devices, as they are connected to the low-voltage grid and therefore often installed in sockets behind walls. In general, relays are used to switch individual connected devices or entire circuits on and off remotely, which is why they are also called "switch actuators". Due to their intended functionality and the fact that their installation locations are typically not directly accessible, smart relays are dependent on wireless connectivity (e.g., Wi-Fi, Bluetooth, or Zigbee). The predominant form of interaction with smart relays is via companion apps, which can sometimes even be used for several products from different brands. Some models can also be controlled via a web application which is either hosted locally on-device or remotely in the vendor's cloud.

Although a selective focus on IoT forensics has been noticeable in the academic literature for some years now (see Section 2), smart relays, as an inconspicuous class of IoT devices, have received little attention so far, leading to a potential blind spot regarding the acquisition of digital evidence during investigations in smart home environments. To approach the closing of this research gap, this paper presents a forensic examination of 16 smart relays from nine brands and six respective companion apps for Android smartphones. Our findings are intended to raise awareness for the forensic artifacts of smart relays, and to support corresponding evidence acquisition in practice.

1.1. Contributions

As we are not aware of any prior work on smart relays, the contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to conduct a multi-device analysis of smart relays from a forensic perspective, covering 16 models from 9 brands as well as six companion apps.
- Facing a multi-source environment for investigation, we identified local artifacts in firmware dumps taken from smart relays as well as in the local app data of companion apps running on Android. Further, we analyzed the network traffic of smart relays and retrieved remote artifacts via cloud APIs.

* Corresponding author.
E-mail address: maximilian.eichhorn@fau.de (M. Eichhorn).

https://doi.org/10.1016/j.fsidi.2024.301810

Available online 18 October 2024

- Based on our findings, we developed a *Tool for Evidence Acquisition from Smart Relays* (TEASR) to support data acquisition in practice by enabling automated and repeatable extraction of relevant artifacts.

To facilitate subsequent work on smart relays, we publish our research artifacts at https://github.com/mxchhrn/teasr, including the source code of TEASR (see Section 5.5).

1.2. Outline

First, we give an overview of related work in Section 2, and describe the fundamentals of the smart relays and their ecosystems in Section 3. Afterwards, we present the methodology of our analysis in Section 4, which is followed by the respective results in Section 5. Finally, we discuss our findings in Section 6, before concluding the paper in Section 7.

2. Related Work

In recent years, a wide range of examinations of “connected” or “smart” devices have been covered under the term “Internet of Things”, as they are *things* that are now *connected* but once had a *non-connected* equivalent. This includes, for example, gaming consoles (Read et al., 2016; Pessolano et al., 2019; Barr-Smith et al., 2021; Eichhorn et al., 2024), smart speakers and displays with intelligent virtual assistants (Li et al., 2019; Jo et al., 2019; Youn et al., 2021; Crasselt and Pugliese, 2024), wearables and trackers (Baggili et al., 2015; Hantke and Dewald, 2020; Pace et al., 2023), CCTV surveillance systems (Dragonas et al., 2023), as well as e-scooters and e-bikes (Hilgert et al., 2021; Stachak et al., 2024).

In addition to the device-specific perspective on IoT-related forensics, various works have also been dedicated to methodological aspects, including procedure models and frameworks (Meffert et al., 2017; Bouchaud et al., 2018; Goudbeek et al., 2018), firmware extraction (Nadir et al., 2022), network traffic analysis (Wu et al., 2021; Shin et al., 2020), or side channel analysis (Sayakkara et al., 2019), while others focused on the identification of challenges (Servida and Casey, 2019; Stoyanova et al., 2020; Friedl and Pernul, 2024) as well as real-world applicability in realistic scenarios (Servida et al., 2023).

As mentioned earlier, however, we are not aware that smart relays and their forensic artifacts have been addressed in the forensic literature yet. At the time of writing, the only related work known to us whose analysis comprised a smart relay and its companion app, namely “Sonoff basic2” and “eWeLink”, respectively, is by Salzillo and Rak (2020), although the focus is primarily on the security of ESP Touch, a Wi-Fi pairing protocol by Espressif Systems.

3. Background

Smart relays are embedded into an extended ecosystem consisting of companion apps and vendor clouds, resulting in a multi-source environment for forensic analyses due to the different local and remote locations for relevant artifacts. In this section, we provide an overview on the fundamentals of smart relays to introduce them as objects of investigation.

3.1. Smart Relays

The main component of a smart relay is a switching device to switch one or more load circuits through a control circuit, an *electromagnetic relay* (Ramirez-Laboreo et al., 2016). If a voltage is applied to the coil terminals, a corresponding magnetic field is formed due to the current flow in the coil. When a voltage is applied to the coil terminals, a current flows, creating a magnetic field that exerts a force on the fixed core which overpowers the elastic force of the plastic pusher and causes the movable contact to shift. Fig. 1 shows the circuit diagram of an

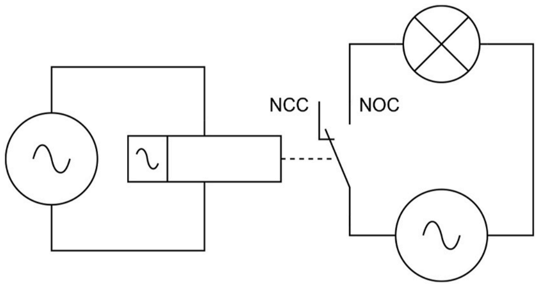

Fig. 1. Circuit diagram of an electromagnetic relay in an AC circuit with no applied voltage on the relay coil.

electromagnetic relay in an AC circuit and, in this case, the moving contact closes the *normally open contact* (NOC). If the current flow through the coil is interrupted, the magnetic field is canceled, and the magnetic force acting on the fixed core is removed. The restoring force on the fixed core is large enough to move it back to its original position under movable contact, closing the *normally closed contact* (NCC). Not all electromagnetic relays have three contact terminals; some have only two contacts, either an NOC or an NCC.

Additional components are required to expand an electromagnetic relay into a smart relay, including a System-on-Chip (SoC), flash chip, Wi-Fi antenna, and debug interfaces. Using terminal blocks, wires can be connected and disconnected conveniently on the smart relay.

Fig. 2 shows the front and back of a smart relay (Shelly 1) to illustrate the individual components. Here, the number 1 marks the terminal blocks to which wires are connected. The number of contacts depends on the range of functions of the smart relay. A minimum number of three terminal contacts is necessary. In this minimum case, two terminals are required for the power supply of the control circuit. One of these two terminals is used for the load circuit. In a standard household low-voltage network, this is the neutral conductor. The third terminal is the switched phase of the load circuit. The actual electromagnetic relay is marked with the number 2.

In addition to various resistors and capacitors, a debug interface (number 3) is located on the front of this exemplary relay. On the Shelly 1, the debug interface is already equipped with soldered sockets and cables can be plugged in directly. In addition to the functions listed in the following subsection, further accessories can be connected via these sockets on some models. The remaining essential components are located on the back. The most relevant component is the SoC (number 5), combining several parts of a computer within a single chip, such as a processing unit, a memory chip, a network module, and other interfaces. SoCs are often designed for a specific application. However, the integrated memory is only sometimes sufficiently dimensioned. In other cases, an external flash memory (number 4) is connected via the interfaces. The final component is the external Wi-Fi antenna (number 6).

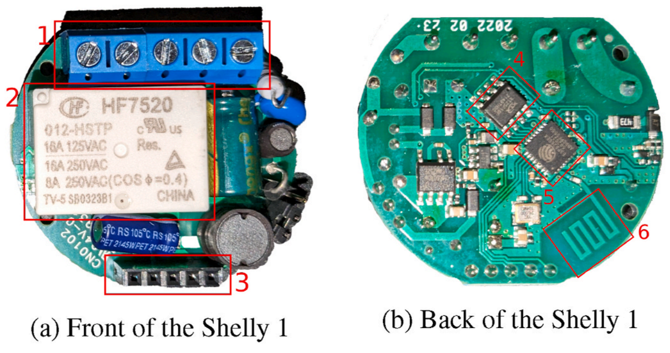

Fig. 2. Internal components of a smart relay (Shelly 1).

3.2. Companion App and Vendor Cloud

Due to their connectivity, smart relays should not be considered on their own during forensic investigations. Smart relays are controlled via the Internet or a local network using wireless protocols like Wi-Fi, Bluetooth, or Zigbee; hence, the user interaction with smart relays is via a companion app or web application, where the corresponding APIs may be hosted locally on-device or remotely in the vendor's cloud.

Depending on the relay–app combination in use, the companion app establishes a local direct connection to the relay or a remote connection to the vendor cloud. A schematic representation of the communication within this ecosystem is shown in Fig. 3 and based on observations made during our examination. The blue arrows indicates remote online communication between the vendor cloud and the smart relay, while local direct communication is shown in green. Finally, the communication between the vendor cloud and the smart relay is shown in orange, and the communication that is not directly necessary for the functionality, such as to a third-party cloud, is shown in gray.

4. Methodology

This section describes how we selected smart relays for our examination, and how we acquired forensically relevant data from the different smart relays, companion apps, and network traffic systematically. To perform repeatable measurements using a predefined circuit with a light bulb and physical switch buttons within a safety environment, we customized a Makita MAKPAC toolcase (see Fig. 4).

4.1. Test Device Selection

Since no sales figures were available to us in order to compile a set of test devices based on definitive market shares of individual models, we performed the following filtering process to select relevant smart relays for our examination. As smart relays rely on companion apps, which are partly not vendor-specific, we started by selecting those apps with at least 500,000 downloads in the Google Play Store, which resulted in the six apps (A_1–A_6) shown in Table 2. The Maxcio app (A_2) was included despite only having around 50,000 downloads, because smart relays by the eponymous vendor also support the *Smart Life* (A_5) and *Tuya Smart* (A_6) app which met the aforementioned criterion. Next, we selected smart relay models based on their support for the six companion apps and their rating on amazon.de. Where applicable, we opted to include several models per vendor to reflect a certain level of diversity in our sample regarding potential differences between products. In total, we selected 16 smart relay models (R_1–R_{16}; see Table 1).

4.2. Firmware Extraction and Analysis

An overview on our entire firmware extraction process is shown in Fig. 5. During our examination of smart relays, we read out the firmware

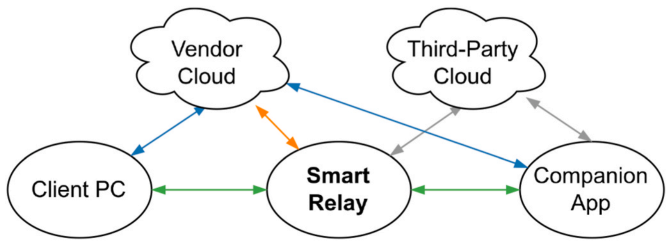

Fig. 3. Overview of a smart relay with its infrastructure and communications. An arrow indicates an observed communication between the two connected components. ↔ online communication, ↔ online communication with vendor cloud API, ↔ online communication to third party services, and ↔ local communication.

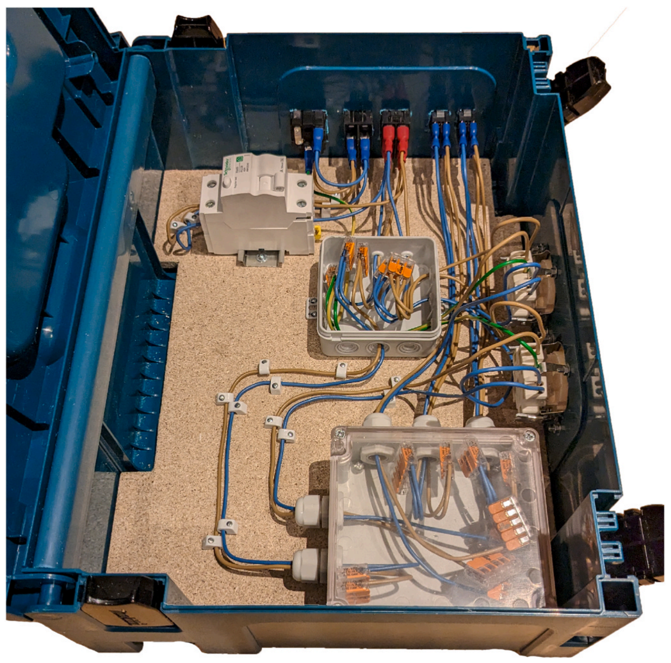

Fig. 4. Customized Makita MAKPAC for execution setup.

via the UART interfaces (universal asynchronous receiver-transmitter) provided on the respective SoCs. It is crucial to know which SoC has been installed on a smart relays to determine which chip architecture is used. In Table 1, we indicate the various SoC models and their instruction set architectures which we found in the smart relays during our analysis. Because of architectural differences between RISC-, RISC-V- and ARM-based SoCs, there is yet to be a ready-made tool for extracting the firmware of every SoC. In our case, we used existing command-line tools for the respective SoCs, namely *bk7231tools* (Clement et al., 2023) and *esptool* (Espressif Systems, 2023), to obtain the firmware of the smart relays via a USB-to-UART adapter.

In addition to the choice of tools, there are other obstacles when extracting the firmware. The contacts of the debug interface are not always directly accessible due to the arrangement of the smart relay components. In those cases of inaccessible contacts, desoldering is necessary. After regaining access to the contacts, the relevant data sheets must be consulted to determine the wiring of the contact pins. However, connecting contacts to a logic analyzer or a digital oscilloscope may still be necessary to determine the pin assignment. Not only is the pin assignment required, but also the knowledge of the different operating modes of the SoCs. For the dumping process, the relevant operating mode is what many manufacturers refer to as “flash mode”. In this mode, it is possible to read or write the flash memory of an SoC. Unfortunately, there are no general instructions for enabling this flash mode, and different steps are necessary depending on the individual SoC. In most cases, the required steps need to be looked up in the data sheets of the individual SoCs or in the instructions of existing extraction tools.

Lastly, to interpret the data of the extracted firmware, it is necessary to gain knowledge of the data structure, such as partition layouts or file systems. Nevertheless, the firmware is sometimes encrypted and needs to be decrypted first (see Table 1). We were able to decrypt the encrypted firmware images analyzed in this work using *bk7231tools* and process the encrypted dump accordingly. Finally, the data stored in the firmware dumps can be accessed and examined.

4.3. Companion App Analysis

We focused on the Android companion apps supported by the smart relays we examined (see Table 2). Persistent data generated on the

Table 1
Overview of the smart relays analyzed with details of their SoCs and notes on extracting and analyzing the firmware.

R	Model	Vendor	SoC	Chip Type	UART Accessibility	Tool	FW Dump	FW Encrypted
R_1	Shelly 1	Shelly	ESP8266EX	RISC	●	esptool	●	○
R_2	Shelly Plus 1	Shelly	ESP32 U4WD	RISC	●	esptool	●	○
R_3	Shelly Plus 1PM	Shelly	ESP32 U4WD	RISC	●	esptool	●	○
R_4	Shelly Plus 2PM	Shelly	ESP32 U4WD	RISC	●	esptool	●	○
R_5	Shelly Plus i4	Shelly	ESP32 U4WD	RISC	●	esptool	●	○
R_6	RR500W	LoraTap	BK7231T	ARMv7E-M	● [a]	bk7231tools	●	●
R_7	RR620W	LoraTap	BK7231D	ARMv7E-M	● [a]	bk7231tools	●	●
R_8	MSS710	meross	RTL8710CM	ARMv8-M	● [b]	–	○	–
R_9	MSS810	Meross	RTL8710CM	ARMv8-M	● [b]	–	○	–
R_{10}	BASICR2	Sonoff	ESP8285N08	RISC	● [a]	esptool	●	○
R_{11}	MINIR2	Sonoff	ESP8285N08	RISC	● [a]	esptool	●	○
R_{12}	SS-8839-03	eMylo	BL2028N	ARM9E	● [a]	bk7231tools	●	●
R_{13}	EWB1CH-D1	Newgoal	BL602L20	RISC-V	● [a]	–	○	–
R_{14}	QS-WIFI-S06-16A	Maxcio	BK7231N	ARMv7E-M	● [b]	bk7231tools	●	●
R_{15}	MS-105	MoesGo	BK7231N	ARMv7E-M	● [b]	bk7231tools	●	●
R_{16}	MINI Smart Mesh	SIUES	TG7220B	ARMv6-M	● [b]	–	○	–

[a] UART accessible after teardown.
[b] UART accessible after desoldering

Table 2
Overview of the companion apps, their supported relays, and Android application IDs.

A#	Companion App	Android Application ID	Supported Relays
A_1	eWeLink - Smart Home	com.coolkit	R_{10}, R_{11}, R_{13}
A_2	Maxcio	com.maxcio.smart	R_{14}
A_3	meross	com.meross.meross	R_8, R_9
A_4	Shelly Smart Control	cloud.shelly. smartcontrol	R_1, R_2, R_3, R_4, R_5
A_5	Smart Life - Smart Living	com.tuya.smartlife	R_6, R_7, R_{12}, R_{14}, R_{15}, R_{16}
A_6	Tuya Smart	com.tuya.smart	R_6, R_7, R_{12}, R_{14}, R_{15}, R_{16}

smartphone when an app is used is stored in `/data/data/<APPID>` directories (Foundation, 2024) named after the individual application IDs (Google, 2023b). The directory containing persistent data on an Android device can be accessed via the Android Debug Bridge (adb) (Google, 2023a). In our case, we accessed the local data of the companion apps via adb using a rooted Google Pixel 6a phone, as we required root permissions in the adb shell. The analysis of the generated network data by the companion apps is explained in the following section.

4.4. Network Analysis

To intercept the network communication of all involved components in the multi-source environment of our investigation (i.e., smart relay, companion app, vendor cloud), we utilized a Raspberry Pi 3 as Wi-Fi router and we performed a man-in-the-middle (MITM) attack

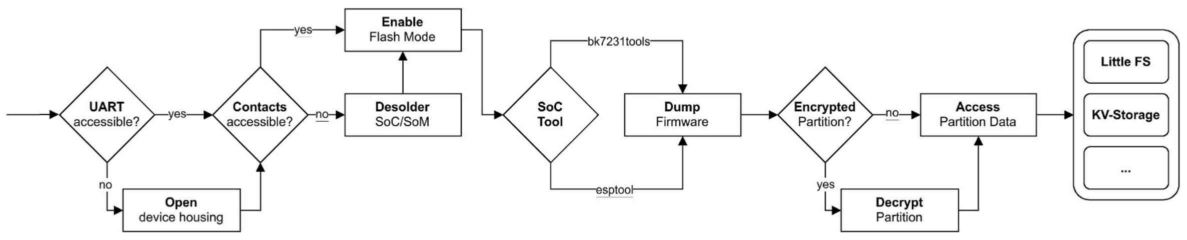

Fig. 5. Firmware extraction process for examined smart relays.

(Bhushan et al., 2017) using "mitmproxy" (Cortesi et al., 2023) as HTTPS proxy in order to bypass TLS encryption and forward re-encrypted network requests. To perform a MITM attack with decryption, the CA certificate of mitmproxy must be trusted on the respective end device whose communication shall be intercepted and decrypted. Adding certificates to the smart relays without further ado was not possible, so communication between smart relays and cloud services could not be recorded in plaintext via mitmproxy.

Two steps were required to capture and decrypt the network traffic of the companion apps. First, an Android phone with root access was required to which the mitmproxy's CA certificate was added as a system certificate. Depending on the version of Android, this step may involve varying levels of effort and obstacles. In our case, we used Android 13 and added the CA certificate using the Magisk module "OverlayFS" (Nguyen, 2023). Secondly, we utilized "objection" (SensePost, 2024) to circumvent *certificate pinning*, a mitigation against MITM attacks via hardcoded certificates in the app code to prevent apps from using untrusted certificates (Gamache and Wall, 2024). Here, the certificate *un-pinning* is achieved by modifying and repackaging the APK, which also includes the injection of a "Frida gadget" of the *Frida* toolkit (Ravnas, 2023) that allows dynamically adapting the app's control flow.

We bypassed the certificate pinning and recorded the decrypted network traffic for three of the six companion apps, namely eWeLink (A_1), meross (A_3), and Shelly Smart Control (A_4). For Maxcio (A_2), Smart Life (A_5), and Tuya Smart (A_6), however, certificate unpinning failed as these companion apps have thrown error messages regarding missing network connections.

4.5. *Creation of Test Data*

A structured procedure for generating test data is necessary to ensure a systemic forensic analysis across multiple devices. As the various smart relays differ in their range of functions, our procedure had to be designed to allow some variability. Furthermore, as incomplete captures of the network traffic of individual performed actions may be possible due to environmental influences, we repeated each action several times. If possible, the individual actions in the following list have been performed with a 10-s pause between each action:

- 10x toggle relay one (sleep 5 s in between)
- 10x toggle relay two (sleep 5 s in between)
- 10x toggle relay one and relay two after each other (sleep 5 s in between)
- 10x toggle relay one as fast as possible (no sleep in between)
- 10x if possible, set a timer for toggle relays one

In cases where a smart relay had fewer electromagnetic relays, or individual functionalities were missing, we have shortened the action set by the actions that could not be performed. Furthermore, the action set was executed once for each of the interaction possibilities which were available for each respective smart relay (i.e., companion app, local API, web server, hardware switch, and cloud API). The network traffic was saved individually in separate files for each execution of the action set.

5. Results

In this section, we present the results of our forensic examination of smart relays grouped by the following artifact categories: "firmware" (Section 5.1), "companion app" (Section 5.2), "network" (Section 5.4), and "cloud" (Section 5.3). A summary of all identified forensic artifacts and their respective locations can be found in Table 3, where each row represents a relay–app combination. Finally, we present a Python framework which we developed based on our findings in order to ease the data acquisition process for smart relays in practice (Section 5.5).

Table 3
Categories of identified forensic artifacts and locations where they have been found. As some relays could be used with multiple companion apps, the artifacts are indicated for all possible combinations of individual relays (R_i) and apps (A_j).

R_i	A_j	Device Info.	Device Conf.	Device List	Device State	State History	Network Conf.	Network State	Wi-Fi Cred.	Cloud API Endpt.	Cloud Cred./Token	Locale Info.
R_1	A_4	$\boxed{\oplus}$	$\boxed{+}$	⊕	⊕	-	□	⊕	□	○	$\boxed{\oplus}$	□
R_2	A_4	$\boxed{\oplus}$	$\boxed{+}$	⊕	⊕	-	□	⊕	□	$\boxed{\circ}$	$\boxed{\oplus}$	□
R_3	A_4	$\boxed{\oplus}$	$\boxed{+}$	⊕	⊕	-	□	⊕	□	$\boxed{\circ}$	$\boxed{\oplus}$	□
R_4	A_4	$\boxed{\oplus}$	$\boxed{+}$	⊕	⊕	-	□	⊕	□	$\boxed{\circ}$	$\boxed{\oplus}$	□
R_5	A_4	$\boxed{\oplus}$	$\boxed{+}$	⊕	⊕	-	□	⊕	□	$\boxed{\circ}$	$\boxed{\oplus}$	□
R_6	A_5	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_6	A_6	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_7	A_5	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_7	A_6	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_8	A_3	⊕	⊕	⊕	⊕	-	-	+	-	○	⊕	-
R_9	A_3	⊕	⊕	⊕	⊕	-	-	+	-	○	⊕	-
R_{10}	A_1	⊕	⊕	⊕	⊕	○	-	+	○	-	⊕	○
R_{11}	A_1	⊕	⊕	⊕	⊕	○	-	+	○	-	⊕	○
R_{12}	A_5	+	+	+	+	+	-	+	□	□	$\boxed{+}$	□
R_{12}	A_6	+	+	+	+	+	-	+	□	□	$\boxed{+}$	□
R_{13}	A_1	⊕	⊕	⊕	⊕	○	-	+	○	-	⊕	○
R_{14}	A_2	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_{14}	A_5	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_{14}	A_6	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_{15}	A_5	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_{15}	A_6	+	+	+	$\boxed{+}$	+	-	+	□	□	$\boxed{+}$	□
R_{16}	A_5	+	+	+	+	+	-	+	-	-	+	-
R_{16}	A_6	+	+	+	+	+	-	+	-	-	+	-

Artifacts found in: □ firmware, ○ app data, + cloud, - no place

5.1. *Firmware*

As mentioned in Section 4.2, SoCs have to be booted into the appropriate (flash) mode to read out the firmware, which requires different pin assignments or tools. Usually, the pin assignments is listed in the manufacturer's data sheets. However, as it is not always possible to see whether the boot process was successful in another mode, an oscilloscope or other measuring device was used to visualize the current over time. The visualization is helpful because the data sheets show that the SoC's current usually depends on the mode. At the beginning, and after we completed our actions, we were able to successfully boot into the correct mode and dump the firmware for 12 out of 16 smart relays (see Table 1). We could not extract the firmware for the remaining relays as we could not determine the requirements for a stable boot into the corresponding mode.

An entropy graph was generated for each firmware dump for an initial high-level analysis and to make first assumptions about possible partition layouts as well as encryption in use. The first two graphs in Fig. 6 are of the MINIR2 relay (R_{11}) before and after we performed our action set (see Section 4.5). Since the relay's firmware was also updated, and the first part of the entropy graph is almost congruent, it is reasonable to assume that this partition layout has two partition sets, especially since the newly written second half is similar to the first half. In contrast, the third entropy graph of the RR620W relay (R_7) shows a region with a constant high entropy, which could indicate a possible encapsulation.

Based on the documentation of the respective SoC manufacturer (Grokhotkov et al., 2023), we extracted partition tables from the firmware dumps of the four devices R_2, R_3, R_4, and R_5. Here, two partition sets are updated alternately during an update and, in some cases, LittleFS was used as file system. However, on three other devices with SoCs

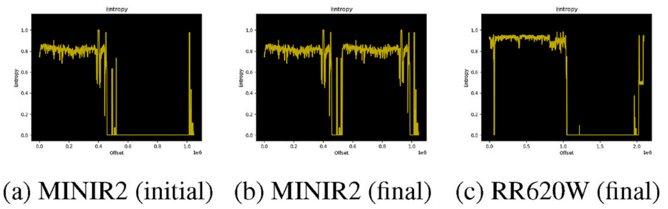
(a) MINIR2 (initial) (b) MINIR2 (final) (c) RR620W (final)

Fig. 6. Comparison of three entropy graphs from firmware dumps of R_{11} and R_7 to show influences of updates, partition layouts, and possible encryption.

from the same manufacturer (R_1, R_{10}, and R_{11}), the partition tables could not be localized, read, or interpreted according to the SoC documentation. The models R_6, R_7, R_{12}, R_{14}, and R_{15}, based on the BK7231X SoC family, had two encrypted partitions followed by a storage with key–value pairs. The tool bk7231tools extracted and decrypted the partitions using known keys for older firmware versions. The *bulk_extractor* (Garfinkel, 2013) tool was used to search for fragments on unknown file systems or on firmware dumps with unknown partitioning. Furthermore, we searched for printable strings via the command-line tool "strings" as well as a Python script to extract JSON file fragments (see Section 5.5).

Table 3 shows the artifacts found for the various firmware dumps, and includes information about the respective smart relay, its status, device and network configurations, Wi-Fi credentials, API endpoints, and localization data. Listing 1 contains an excerpt of exemplary artifacts of the devices R_6, R_7, R_{14}, and R_{15}, showing the device status and Wi-Fi credentials. Furthermore, the firmware dump of R_2 contained cloud credentials, cloud access tokens, and information such as the timezone and coordinates (see Listing 2).

5.2. Companion App

Table 3 shows that forensic artifacts were identified in the app data for three of the six companion apps, namely A_1, A_3 and A_4. The files of interest were in the paths `*/files/log/main` and had the file extension `xlog`. The corresponding entropy graph Fig. 7 shows an entropy value above 0.97 across the entire file content, which suggests encryption. We could not identify the key to decrypt these files. While interesting directories were found for the three remaining apps (A_2, A_5, A_6), they did not contain any directly readable data. It should be noted that these apps share a similar structure, GUI, and almost identical paths in their app data.

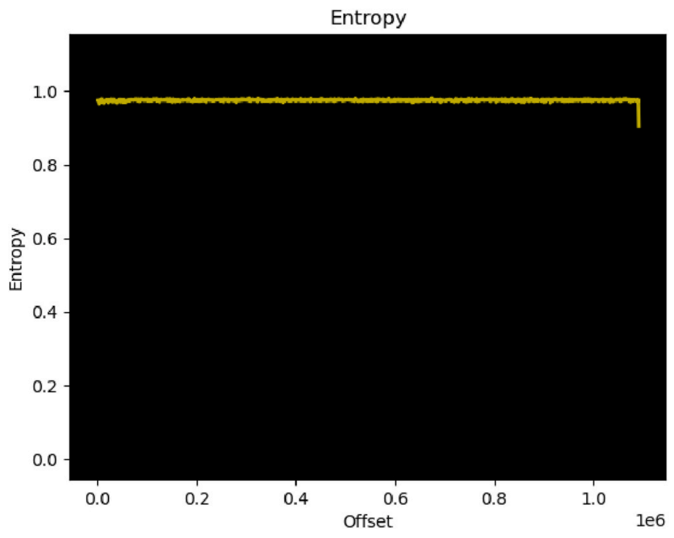

Fig. 7. Entropy graph of a log file with the extension `xlog` with a constant high entropy value for the file content.

The most interesting forensic artifact in the companion apps' data is the log file of the *eWeLink - Smart Home* app (A_1) which contains a history of switching operations during app use. The log file lists individual switching operations along with the corresponding timestamp, as well as pairing processes, and found Bluetooth devices. Furthermore, we identified small differences in the log file which can be used to determine whether the switching process was initiated via the app or otherwise. Listing 3 shows a section of this file in which two switching processes are displayed, one via the app and one external via the physical button.

The discovery of other Bluetooth devices during the pairing process is shown in Listing 4, which contains another excerpt from the log file. In addition to a history of the device status, the individual app data includes device information and configurations, lists of devices, network status, credentials, and locale information. An overview of all forensic artifacts that have been found can be seen in Table 3.

5.3. Cloud API

We extracted some of the data stored in the vendor's clouds via respective cloud APIs. The responses contained mainly data about the devices linked to the cloud account. In this way, device lists with the devices' information, configurations, and network state could be retrieved. In three of the six companion apps (A_1, A_3 and A_4), the added smart relays are connected directly to the cloud account, which can also be accessed via API with email and password credentials. To use the cloud API for Tuya-based devices, a separate developer account must first be created on the `developer.tuya.com` domain. The usage credentials in the form of an ID and a key are shown in the creation process. The devices can then be added to the account by linking the developer account either with a supported third-party application or with *Smart Life - Smart Living* (A_5). In the second case, all devices added to the app so far as well as those added in the future are automatically linked to the cloud account. The four cloud APIs of Shelly, eWeLink, meross, and Tuya can be differentiated accordingly.

Listing 5 shows a section of the Tuya cloud API output for a smart relay with highlighted forensic artifacts. The excerpt contains, among other things, information about the public IP address, time zone, device configuration, device status, coordinates, and the timestamp of the last status update of the device. In addition to a device list, the Tuya cloud API can also query a log for a device ID (see Listing 6), which lists not only switching actions but also changes in the device's online status. Two IDs, *event_from* and *event_id*, are also listed for each action. Unfortunately, we could not find manually initiated switching actions via physical buttons. Furthermore, only some switching events are listed in the received log, and we have yet to identify a pattern regarding when they are listed and when they are not. Again, an overview of further forensic artifacts is shown in Table 3.

5.4. Network

Network analysis of traffic when using companion apps, cloud APIs, and smart relays was a means to an end for the forensic artifacts explained above. The recorded network traffic provided insights into the structure of requests and responses during switching actions, update processes, or other procedures. The first thing that stood out when analyzing the data was that some relays communicate locally and others (R_6, R_7, R_{12}, R_{14}, R_{15}, R_{16}) via an Internet connection. All local communication between smart relays and companion devices or clients was transmitted unencrypted. However, three relays (R_{10}, R_{11}, R_{13}) encrypted the part of the payload that contains the transmitted commands. In all cases, communication via the Internet was carried out using TLS encryption. Below address simple successful attacks on unencrypted local communication.

A replay attack was possible in all cases of local communication, and thus, we successfully replayed payloads that had already been sent, irrespective of their encryption. The Shelly relays had an authentication option that was deactivated by default. The first-generation Shelly models used the insecure basic access authentication method of the HTTP protocol, so we received the credentials directly from sent packets and from the firmware. For second-generation Shelly devices, a hash value containing the credentials, a nonce, and the corresponding counter was created and has been sent in the request. However, even with these devices, the authentication procedure did not protect against replay attacks, as we could sent a packet multiple times despite the identical counter of the nonce.

We could also make the respective smart relay unusable in the event of local communication with DoS attacks. The only exceptions were the relays R_8 and R_9, where a successful DoS attack was impossible despite local communication. In the other cases, the devices were unusable during the ongoing attack and could only process other requests again once the attack had ended. However, even after the end of the attack, the relays of Shelly's second-generation relays were still able to process previously cached requests from the DoS attack via the WebSocket protocol after the end of the attack. In one case, we observed that the smart relay could not be used even 4.5 min after the attack ended.

5.5. Python Framework

This section describes the development of the Python framework *Tool for Evidence Acquisition from Smart Relays* (TEASR) for the automated extraction of identified forensic artifacts. It is designed to receive various input files and extract forensic artifacts based on the results in Sections 5.1, 5.2, 5.4, and 5.3. Correspondingly, the Python script `teasr.py` provides four separate commands that can be executed. The overview in Fig. 8 visualizes the framework with its most important inputs and outputs.

5.5.1. Extractions From Firmware Dumps

The first command, *firmware*, takes a given firmware dump as input or a whole directory containing different firmware dump files. The two options are mutually exclusive, and the method for extracting possible traces is called for each firmware dump file. First, the command calls the tool *binwalk* to get an entropy graph from the firmware dump. Furthermore, an attempt is made to assign the firmware to a vendor by searching for corresponding strings. If the firmware file to be analyzed is that of a second-generation Shelly or a Tuya device, the partitions are separated from the firmware dump and then processed further. For firmware from first-generation Shelly devices, Sonoff devices, or unknown device types, a partition table in CSV format can be added to the command to analyze such firmware dumps partition-wise. If no partition table is available and cannot be extracted, the firmware file is analyzed as a whole.

Strings, JSON fragments, and files are extracted, regardless of whether a firmware file or a partition is analyzed. The files are retrieved with the tools *bulk_extractor* and *mklittlefs*. However, *mklittlefs* can only successfully find files if a *LittleFS* file system is present. The output file `summary.json` contains a summary of forensic traces of the categories already presented. If the firmware has several possible sources for this summary, the sources with the higher offset are selected to use the most up-to-date data. If there are several partition sets, the summarized data of both sets is listed separately in the output.

```
{ [...]
  "gw_wsm": {
    "nc_tp": 3,
    "ssid":  [BASE64_SSID],
    "passwd": [BASE64_PASSWORD],
    "md": 0,
    "random": 0,
    "wfb64": 1,
    "stat": 2,
    "token": "XR1AzcD1",
    "region": "EU",
    "reg_key": "50Eu"
  },
  "save_off_stat": { "power": [ true ] }, [...]
}
```

Listing 1. Extracted key–value pairs of LoraTap RR500W (R_6) containing Wi-Fi credentials and the last device state.

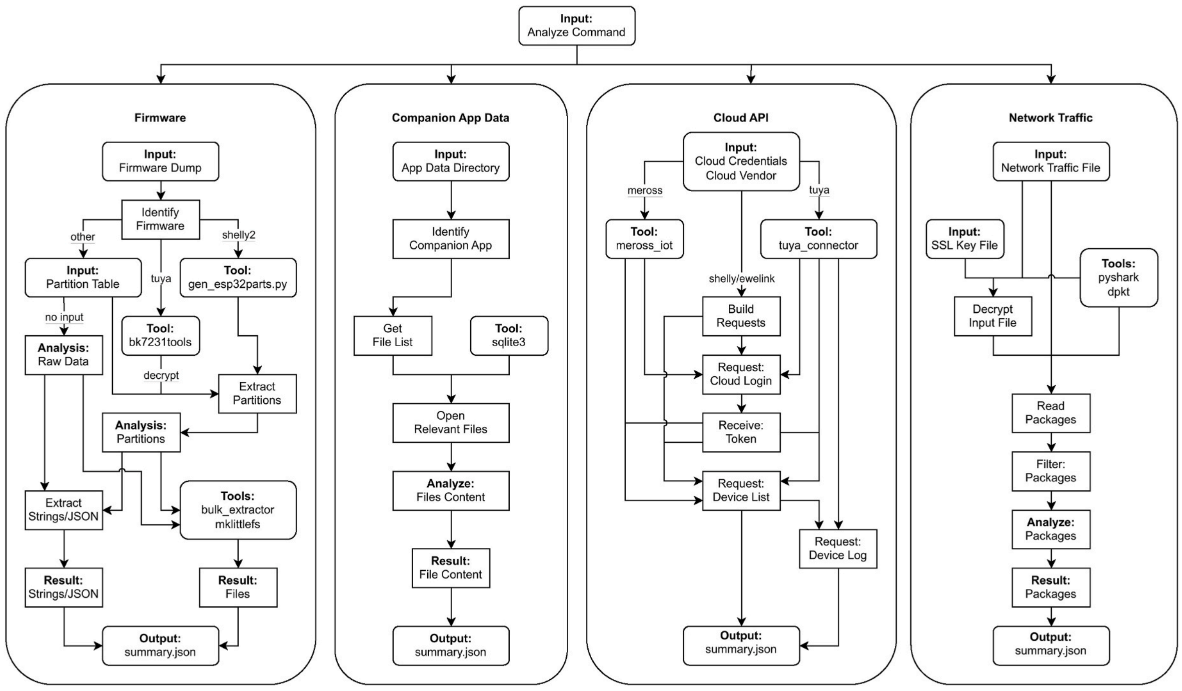

Fig. 8. Overview to visualize the framework with its four main components and their inputs and outputs.

```
{
  "sys": {
    "tz_spec": "CET-1CEST,M3.5.0,M10.5.0/3",
    "last_known_good_time": 1698404058
  },
  "device": { "location": {
      "lat": [LATITUDE],
      "lon": [LONGITUDE]
    } }, [...]
  "shelly": {
    "cfg_rev": 34,
    "cloud": {
      "enable": true,
      "server": "shelly-77-eu.shelly.cloud:6022/jrpc",
      "token": [TOKEN]
    },
    "tz": "Europe/Berlin", [...] } [...]
  "sw": {
    "cfg_version": 1,
    "auto_off_delay": 5.000000
} }
```

Listing 2. Extracted JSON files of Shelly Plus 1 (R_2) reveal cloud token, API endpoint, and timezone information.

```
// switch toggle app
$$info$$2023-11-11 21:10:44--$_onPress toggle$
$$info$$2023-11-11 21:10:44--$SWITCH_PROTOCOL jimmy
     isOn:off$
$$info$$2023-11-11 21:10:44--$_onPress toggle params
     :"switch":"on"$
$$info$$2023-11-11 21:10:44--$send toggle switch
     params :{"switch":"on"}, api is:/zeroconf/
     switch$
[...]
$$info$$2023-11-11 21:10:45--$>>>> : updateParams {"
     switch":"on"} $
$$info$$2023-11-11 21:10:45--$handleCommon key:
     switch,value:on$
$$info$$2023-11-11 21:10:45--$SWITCH_PROTOCOL jimmy
     isOn:on$

// switch toggle physical button
$$info$$2023-11-11 21:15:13--$MomokoSocket json {"
     action":"update","deviceid":"10015160a2","apikey
     ":"40820c93-38eb-4cc6-bf79-b198757702ae","
     userAgent":"device","params":{"switch":"off"},"
     seq":"46"}$
[...]
$$info$$2023-11-11 21:15:13--$>>>> : updateParams {"
     switch":"off"} $
$$info$$2023-11-11 21:15:13--$handleCommon key:
     switch,value:off$
$$info$$2023-11-11 21:15:13--$SWITCH_PROTOCOL jimmy
     isOn:off$
```

Listing 3. Extracted file `coolkit_log_1.log` of the eWeLink app reveals two different device switching actions.

5.5.2. *Extractions From Companion App Data*

The second command, *app*, calls a Python function that inputs the given app data directory and extracts artifacts from the files within this directory. The various companion apps are differentiated by the directory name. Additional companion apps can be added anytime due to the modular structure. Depending on the companion app in use, different files can be found in the app's persistent data, as described in Section 5.2. The files to be searched can be text-based files or databases. The contents of these files are searched accordingly and summarized in an output file, `summary.json`.

```
$$info$$2023-11-11 21:34:37--$BlePairingProgress
     start BlePairingProgress$
[...]
$$info$$2023-11-11 21:34:37--$[BLEPairUtils]
     onScanResult EB3A2236005094532$
$$info$$2023-11-11 21:34:37--$[BLEPairUtils]
     onScanResult [TV] Samsung 7 Series (43)$
[...]
$$info$$2023-11-11 21:35:22--$EweLnkAppNavigator
     FinishPairingScreen =>
     SharePairDeviceScreencurrentScreenParams=>{"
     addedDevice":[{"rssi":-51,"mac":"24:94:94:3D:5A
     :08","name":"Device53f1ce","deviceid":"100153
     f1ce","type":"ble","status":"reslove","uiid
     ":138,"icon":404,"startWays":"init"}],"pairInfo
     ":{"ssid":[SSID],"bssid":"b8:27:eb:d0:7c:01","
     password":[PLAINTEXT_PASSWORD],"pairType":18}}$
```

Listing 4. Extracted file `coolkit_log_1.log` of the eWeLink app reveals the pairing process of a device.

```
{
  'active_time': 1717147296,
  'biz_type': 18,
  'category': 'tdq',
  'create_time': 1717147296,
  'icon': 'smart/icon/ay1509430484182dhmed/3
     d92a1c7c005d3d18a5c9ec55dbec3d7.png',
  'id': 'bf047307b94c984cf131ie',
  'ip': [PUBLIC_IP],
  'lat': [LATITUDE],
  'local_key': '^e;&ZXJLeZ39t>(J',
  'lon': [LONGITUDE],
  'model': 'qcrr500w',
  'name': 'Smart Switch',
  'online': False,
  'owner_id': '193580224',
  'product_id': 'jjglrmqe391d5tb1',
  'product_name': 'Smart Switch',
  'status': [[...]],
  'sub': False,
  'time_zone': '+02:00',
  'uid': 'eu1717134136871A0uNH',
  'update_time': 1717163216,
  'uuid': '73ec1c5393dd8fd3'
}
```

Listing 5. Response of the Tuya cloud API endpoint `/v1.0/iot-01/associated-users/devices` containing the coordinates, public IP address, and timestamps.

5.5.3. *Extractions From Network Traffic Captures*

The third command, *network*, can be used to analyze a network traffic dump which has been captured in PCAP format. The optional inputs are files belonging to the dump with SSL/TLS keys or flow files of the *mitmproxy* tool. If the optional inputs are omitted, the command opens the PCAP file and reads the HTTP and WebSocket packets. By optionally specifying a source and a destination address, it is possible to filter the read packets. Without a TLS key, the function can only process decrypted packets. A flow file or the corresponding key file is required for network traffic via HTTPS or WSS. The transferred PCAP file is temporarily processed with a transferred TLS key file, and the network packets are decrypted if possible. Finally, relevant information about the extracted packets are saved in `summary.json`.

5.5.4. *Extractions of Remote Artifacts From Cloud*

The fourth command, *remote*, calls a Python function that extracts a device list from the corresponding cloud API endpoint, but the particular

```
// device online:
{
  "event_from": "1",
  "event_id": 1,
  "event_time": 1717401225000,
  "status": "1"
}

// relay 1 of R_7 switched to off:
{
  "code": "switch_1",
  "event_from": "2",
  "event_id": 5,
  "event_time": 1717400440537,
  "status": "1",
  "value": "false"
}
```

Listing 6. Device event log requested of the Tuya cloud API endpoint `/v1.0/devices/[DEVICE_ID]/logs` containing event log messages for the device R_7. The *event_from* value specifies the source of the event and can take values from 1 to 16 or -1. The most important values are 1 (from this device), 2 (from client), 3 (from third-party service), and 4 (from the cloud). Furthermore, the variable *event_id* specifies the event type and can take values from 1 to 10. The most important values are 1 (online), 2 (offline), 3 (activated), and 4 (reset) and refer to the device in each case. Other values can be found in the Tuya cloud API documentation (Tuya Inc, 2024).

cloud service for the given smart relay must be passed as input. Furthermore, the respective credentials are required, which can be username and password, or other access data which depends on the respective vendor and can vary accordingly. The cloud API endpoints and domains are hardcoded and must be adapted if the cloud provider changes the endpoints or domains. The command creates a request to query all devices belonging to the user account and sends this to the cloud API endpoint. The information transmitted in the response varies in scope depending on the cloud provider and is stored in the `summary.json` output file together with any access token used.

6. Discussion

6.1. Applicability and Practical Remarks

While our forensic analysis of smart relays as a device class was not based on a specific law enforcement case, the forensic artifacts which we identified may be helpful in certain investigative scenarios.

Since discarded smart devices can contain sensitive information, as pointed out by Sharma and Awasthi (2024), a disposed smart relay could be secretly confiscated to extract Wi-Fi credentials (see Table 3), which may result in expanded opportunities for the surveillance of a suspect's domestic habitat during digital investigations. From an anti-forensics perspective, but also from a security and privacy standpoint, it is therefore not advisable to dispose or resell smart relays without taking additional measures to prevent subsequent unauthorized access to locally persisted data.

For smart relays with local network communication (i.e., all devices supported by the companion apps A_1, A_3, and A_4; see Table 3), we showed that commands for switching actions are sent in plain text over HTTP. The only exception to this are the two relays R_8 and R_9 where the commands are encrypted but still sent over HTTP. The monitoring of such app-relay communication within a suspect's local network may therefore help in surveillance scenarios where it is necessary to identify whether the individual is certainly at home and currently interacting with their unlocked smartphone, or whether there are activities within rooms where smart relays are installed. If the respective smart relay vendor is willing to cooperate, these switching actions can also be monitored externally via the vendor's cloud. Without cooperation, law enforcement must obtain the cloud credentials first through other investigative or surveillance measures. However, the approach via the vendor's cloud may only be applicable if cloud usage was activated on the smart relay, or a developer account has been created (see Section 5.3).

6.2. Limitations and Future Work

Despite our best efforts to systematically generate test data during our investigation (see Section 4), the list of performed actions to create local and network artifacts cannot be considered exhaustive in light of the entire range of possible functions that (some of) the examined smart relays provide. While we covered the most essential functionalities to generate relevant traces, the number of reported artifacts may therefore rather be a lower boundary, and further actions may create additional artifacts which we could not identify due to methodological reasons. For future work, we encourage the analysis of smart relays from other manufacturers, new models from the manufacturers that we looked at, as well as of companion apps for iOS devices in order to identify further relevant artifacts on the basis of an extended scope of investigation.

The Python framework presented in Section 5.5, which we developed to ease the data acquisition process from the smart relays and their extended ecosystem consisting of companion app and vendor cloud, currently supports only those models what have been examined in this paper. Hence, adding support for newer models or further companion apps will require changes of the source code. By releasing the source code (see Section 1.1), we hope to mitigate this temporary limitation to the extent that we enable the community to adapt or extend the range of functions and devices as needed.

7. Conclusion

We forensically analyzed the IoT device class of smart relays based on 16 models from 9 vendors and six different companion apps in total. For test data generation during our examination, we used a structured approach with minimally varying action sets to create traces in a controlled manner.

We discovered a variety of forensic artifacts within the multi-source environment of our investigation which consisted of the smart relays themselves, their supported companion apps, and their respective vendor clouds. The artifacts of forensic relevance which we identified include, inter alia, Wi-Fi and cloud credentials, device and network information, as well as device lists and their status histories. Based on our findings, we developed a Python framework to facilitate evidence extraction from smart relays, their companion apps, and their respective vendor cloud APIs.

CRediT authorship contribution statement

Maximilian Eichhorn: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization, Project administration. **Gaston Pugliese:** Conceptualization, Methodology, Resources, Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. Furthermore, we thank Lukas Fischer for sharing insights into the authorization protocol of Shelly Gen 2 devices, as well as Matti Schulze for recommending TikZ for the symbols in Table 3. This work has been supported by the Bavarian Ministry of Science and Arts as part of the project "Security in Everyday Digitization" (ForDaySec).

References

Baggili, I., Oduro, J., Anthony, K., Breitinger, F., McGee, G., 2015. Watch what you wear: preliminary forensic analysis of smart watches. In: 2015 10th International Conference on Availability, Reliability and Security. IEEE, pp. 303–311.

Barr-Smith, F., Farrant, T., Leonard-Lagarde, B., Rigby, D., Rigby, S., Sibley-Calder, F., 2021. Dead Man's Switch: Forensic Autopsy of the Nintendo Switch. Forensic Sci. Int.: Digit. Invest. 36, 301110.

Bhushan, B., Sahoo, G., Rai, A.K., 2017. Man-in-the-middle attack in wireless and computer networking — A review. In: 2017 3rd International Conference on Advances in Computing,Communication & Automation (ICACCA) (Fall), pp. 1–6. https://doi.org/10.1109/ICACCAF.2017.8344724.

Bouchaud, F., Grimaud, G., Vantroys, T., 2018. IoT Forensic: identification and classification of evidence in criminal investigations. In: Proceedings of the 13th International Conference on Availability, Reliability and Security, pp. 1–9.

Clement, T., Nassar, K., Szczodrzyński, K., 2023. bk7231tools. https://github.com/tuya-cloudcutter/bk7231tools. (Accessed 19 November 2023).

Cortesi, A., Hils, M., Kriechbaumer, T., contributors, 2023. mitmproxy: a free and open source interactive HTTPS proxy. URL: https://mitmproxy.org/. (Accessed 20 November 2023).

Crasselt, J., Pugliese, G., 2024. *Started Off Local, Now We're in the Cloud:* Forensic Examination of the Amazon Echo Show 15 Smart Display. Digital Forensics Research Conference USA. DFRWS USA 2024) URL: https://dfrws.org/wp-content/uploads/2024/07/dfrws-usa-2024-echo-show-1.5.pdf.

Dragonas, E., Lambrinoudakis, C., Kotsis, M., 2023. IoT forensics: Analysis of a HIKVISION's mobile app. Forensic Sci. Int.: Digit. Invest. 45, 301560.

Eichhorn, M., Schneider, J., Pugliese, G., 2024. *Well Played, Suspect!* — Forensic examination of the handheld gaming console "Steam Deck". Forensic Sci. Int.: Digit. Invest. 48, 301688 https://doi.org/10.1016/j.fsidi.2023.301688.

Espressif Systems, 2023. esptool.py. https://github.com/espressif/esptool. (Accessed 19 November 2023).

Friedl, S., Pernul, G., 2024. IoT Forensics Readiness – influencing factors. Forensic Sci. Int.: Digit. Invest. 49, 301768.

Garfinkel, S.L., 2013. Digital media triage with bulk data analysis and bulk_extractor. Comput. Secur. 32, 56–72. https://doi.org/10.1016/j.cose.2012.09.011. URL: https://www.sciencedirect.com/science/article/pii/S0167404812001472.

Google, 2023a. Android debug bridge (adb). URL: https://developer.android.com/tools/adb. (Accessed 20 November 2023).

Google, 2023b. Configure the app module. URL: https://developer.android.com/build/configure-app-module. (Accessed 20 November 2023).

Goudbeek, A., Choo, K.K.R., Le-Khac, N.A., 2018. A Forensic Investigation Framework for Smart Home Environment. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). IEEE, pp. 1446–1451.

Grokhotkov, I., Gratton, A., Jiang, J., contributors, 2023. Partition tables. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/partition-tables.html. (Accessed 25 November 2023).

Hantke, F., Dewald, A., 2020. How can data from fitness trackers be obtained and analyzed with a forensic approach?. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, pp. 500–508.

Hilgert, J.N., Lambertz, M., Hakoupian, A., Mateyna, A.M., 2021. A forensic analysis of micromobility solutions. Forensic Sci. Int.: Digit. Invest. 38, 301137.

Jo, W., Shin, Y., Kim, H., Yoo, D., Kim, D., Kang, C., Jin, J., Oh, J., Na, B., Shon, T., 2019. Digital Forensic Practices and Methodologies for AI Speaker Ecosystems. Digit. Invest. 29, S80–S93.

Li, S., Choo, K.K.R., Sun, Q., Buchanan, W.J., Cao, J., 2019. IoT Forensics: Amazon Echo as a Use Case. IEEE Internet Things J. 6, 6487–6497. https://doi.org/10.1109/JIOT.2019.2906946.

Meffert, C., Clark, D., Baggili, I., Breitinger, F., 2017. Forensic State Acquisition from Internet of Things (FSAIoT): A general framework and practical approach for IoT forensics through IoT device state acquisition. In: Proceedings of the 12th International Conference on Availability, Reliability and Security, pp. 1–11.

Nadir, I., Mahmood, H., Asadullah, G., 2022. A taxonomy of IoT firmware security and principal firmware analysis techniques. Int. J. Critic. Infrastruct. Protect. 38, 100552 https://doi.org/10.1016/j.ijcip.2022.100552. URL: https://www.sciencedirect.com/science/article/pii/S1874548222000373.

Nguyen, S., 2023. Magisk Overlayfs. https://github.com/HuskyDG/magic_overlayfs. (Accessed 20 November 2023).

Pace, L.R., Salmon, L.A., Bowen, C.J., Baggili, I., Richard III, G.G., 2023. Every step you take, I'll be tracking you: Forensic analysis of the tile tracker application. Forensic Sci. Int.: Digit. Invest. 45, 301559.

Pessolano, G., Read, H.O., Sutherland, I., Xynos, K., 2019. Forensic Analysis of the Nintendo 3DS NAND. Digit. Invest. 29, S61–S70.

Ramirez-Laboreo, E., Sagues, C., Llorente, S., 2016. A New Model of Electromechanical Relays for Predicting the Motion and Electromagnetic Dynamics. IEEE Trans. Ind. Appl. 52, 2545–2553. https://doi.org/10.1109/TIA.2016.2518120.

Ravnas, O.A.V., 2023. Frida - Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. https://frida.re/. (Accessed 20 November 2023).

Read, H., Thomas, E., Sutherland, I., Xynos, K., Burgess, M., 2016. A Forensic Methodology for Analyzing Nintendo 3DS Devices. In: IFIP International Conference on Digital Forensics. Springer, pp. 127–143.

Salzillo, G., Rak, M., 2020. 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. Association for Computing Machinery, New York, NY, USA, pp. 37–48. https://doi.org/10.1145/3411498.3419965.

Sayakkara, A., Le-Khac, N.A., Scanlon, M., 2019. Leveraging Electromagnetic Side-Channel Analysis for the Investigation of IoT Devices. Digit. Invest. 29, S94–S103.

SensePost, 2024. Objection - Runtime mobile Exploration. https://github.com/sensepost/objection.

Servida, F., Casey, E., 2019. IoT forensic challenges and opportunities for digital traces. Digit. Invest. 28, S22–S29. https://doi.org/10.1016/j.diin.2019.01.012.

Servida, F., Fischer, M., Delémont, O., Souvignet, T.R., 2023. OK Google, Start a Fire. IoT devices as witnesses and actors in fire investigations. Forensic Sci. Int. 348, 111674 https://doi.org/10.1016/j.forsciint.2023.111674.

Sharma, P., Awasthi, L.K., 2024. Unveiling the hidden dangers: Security risks and forensic analysis of smart bulbs. Forensic Sci. Int.: Digit. Invest. 50, 301794 https://doi.org/10.1016/j.fsidi.2024.301794. URL: https://www.sciencedirect.com/science/article/pii/S2666281724001185.

Shin, Y., Kim, H., Kim, S., Yoo, D., Jo, W., Shon, T., 2020. Certificate Injection-Based Encrypted Traffic Forensics in AI Speaker Ecosystem. Forensic Sci. Int.: Digit. Invest. 33, 301010.

Stachak, M., Geus, J., Pugliese, G., Freiling, F., 2024. Nyon Unchained: Forensic Analysis of Bosch's eBike Board Computers. Digital Forensics Research Conference Europe (DFRWS EU 2024). URL: https://dfrws.org/wp-content/uploads/2024/03/Nyon-Unchained-Forensic-An.alysis-of-Boschs-eBike-Board-Computer_2024.pdf.

Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., Markakis, E.K., 2020. A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues. IEEE Commun. Surv. Tutor. 22, 1191–1221.

Urquhart, L., Miranda, D., Podoletz, L., 2022. Policing the smart home: The internet of things as 'invisible witnesses'. Inf. Polity 27, 233–246.

Vailshery, L.S., 2024a. Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide. (Accessed 14 August 2024).

Vailshery, L.S., 2024b. Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030, by vertical. https://www.statista.com/statistics/1194682/iot-connected-devices-vertically. (Accessed 14 August 2024).

Wu, T., Breitinger, F., Niemann, S., 2021. IoT network traffic analysis: Opportunities and challenges for forensic investigators? Forensic Sci. Int.: Digit. Invest. 38, 301123.

Youn, M.A., Lim, Y., Seo, K., Chung, H., Lee, S., 2021. Forensic analysis for AI speaker with display Echo Show 2nd generation as a case study. Forensic Science International. Digit. Invest. 38, 301130.

Gamache, M., Wall, K., 2024. Certificate and Public Key Pinning. https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning. (Accessed 14 August 2024).

OWASP Foundation, 2024. Android Data Storage. https://mas.owasp.org/MASTG/0x05d-Testing-Data-Storage/. (Accessed 14 August 2024).

Tuya Inc., 2024. Device management URL: https://developer.tuya.com/en/docs/cloud/device-management. (Accessed 2 June 2024).