Every Frame You Take: A Taxonomy and Reproducible Testing Framework for Digital Surveillance Cameras

Maximilian Eichhorn*, Felix Gabel and Felix Freiling

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

ARTICLE INFO

Keywords:
digital forensics
IoT and embedded device forensics
digital surveillance cameras
hardware forensics

ABSTRACT

We present a definition and taxonomy of digital surveillance cameras and provide a structured approach for extracting forensically relevant artifacts from these devices. As part of this approach, we implemented a Python-based framework for automatically and reproducibly generating test data within a camera test environment and evaluated it using 9 cameras from various manufacturers. We show that despite considerable device diversity, some devices exhibit minimal differences upon closer examination of their forensic artifacts. Still, although some generalization across camera subgroups is possible, device-specific analysis remains necessary for a comprehensive forensic investigation.

1. Introduction

Digital cameras are used for a wide variety of applications and are integrated into different devices. These may be general camera modules in smartphones or notebooks that extend the functionality of these devices, or more specialized devices such as surveillance cameras. Due to the multitude of devices with digital camera modules, these devices are frequently encountered by law enforcement, and the images and videos captured by them can provide evidence with high probative value in criminal cases.

The diversity of "devices with a camera", however, also complicates forensic analysis. While it may seem easy to acquire video footage on such a device, some cameras may provide caches or previews that are only accessible through companion apps. Other devices may store continuous recordings on an SD card even if no event detection has triggered. Artifacts like log files may even indicate the time and circumstances of video sequences long after they have been overwritten.

Accessing all this data in a structured way required a sound methodology tailored to specific device classes. Given the diversity of devices with non-homogeneous functionality, it is terminologically unclear what category a device with a digital camera actually represents. Is it a baby monitor, a doorbell with an integrated camera, or something entirely different?

Camera-equipped devices also pose additional challenges for analysis, as we now explain. When investigating unknown devices, a usual methodology is to generate test data on an identical device while varying inputs or trigger conditions (Garfinkel et al., 2012; Eichhorn and Freiling, 2025). A comparison of this data allows to subsequently attribute forensic artifacts to the actions performed during the generation process. While for some device classes the activation of actions can be performed through APIs, reliably triggering events on devices like surveillance cameras that interact with their physical surroundings requires a controlled environment with reproducible visual and auditory inputs, free from disruptive environmental interference. And again: The analysis depends strongly on the individual camera functions, such as motion detection (for surveillance cameras) or human activation (for smart doorbells), so terminological clarity is important here too. The fact that camera-equipped devices are commonly shipped with their own companion app does not make the analysis task easier since this requires holistic ecosystem-level analyses.

Given that, to the best of our knowledge, no existing taxonomy or systematic forensic analysis exists for the device class of digital (surveillance) cameras, this paper presents a forensic examination of 9 digital cameras from 9 different brands, together with their respective companion apps and cloud APIs. Our findings, combined with a proposed taxonomy, systematically characterize the device class and establish clear boundaries with related device classes.

1.1. Related Work

Digital surveillance cameras pose substantial privacy threats and security risks in the event of unauthorized access, which is why these issues are repeatedly investigated, and vulnerabilities are identified and disclosed. Both more general investigations (Alharbi and Aspinall, 2018; Valente et al., 2019; Trabelsi, 2022) and specialized ones for individual specific devices (Seralathan et al., 2018; Abdalla and Varol, 2020) were conducted but did not focus on the extraction of forensic artifacts.

Holistic ecosystem-level forensic analyses exist for some IoT devices (Li et al., 2019; Servida et al., 2023; Stachak et al., 2024), yet remain scarce for camera equipped devices. Dragonas et al. (2023) analyzed the companion app for HIKVISION digital cameras and identified forensic artifacts in their iOS and Android apps. The authors found paths of created images, artifacts of user interactions, and user and device information. However, their study did not examine the audio/video files stored on the CCTV system itself or its log files. Dragonas et al. (2024) addressed this gap in the forensic analysis of log files by examining log entries from DAHUA CCTV systems and identifying storage locations

*corresponding author (maximilian.eichhorn@fau.de)

and log types. Additionally, Bhardwaj et al. (2023) examined the firmware of IoT cameras for potential security implications and discussed forensic implications for law enforcement purposes. They define 12 steps for a comprehensive IoT firmware security analysis, but only briefly address the forensic relevance and possible artifacts of the examined devices. Furthermore, Salem and Hamarsheh (2024) examine IoT smart cameras in a forensic context using the *Multi-level Artifact of Interest Digital Forensics Framework for IoT* (MAoIDFF-IoT) (Salem et al., 2024), and focus on extracting forensic artifacts from network traffic and companion apps or web interfaces.

So overall, prior work focused on devices from individual manufacturers or examined only specific aspects of the devices or artifact locations. We are not aware of any approach that has attempted to develop an analysis approach suitable for multiple devices.

1.2. Contributions

In this paper we present a systematic forensic analysis of nine digital cameras and their respective ecosystems, including internal memory, microSD cards, companion apps, and cloud API responses. Additionally, we propose a formal taxonomy for digital surveillance cameras, which, to the best of our knowledge, has not been established in prior work. More specifically, this paper makes the following contributions:

- We propose a taxonomy for classifying digital surveillance cameras and examine category-specific forensic artifacts.
- We present an integrated approach to analyze items from different categories. To support this, we developed a Python framework, *What is Before the Lens* (WiBtL), that supports systematic and reproducible test data generation for defined scenarios in the context of a forensic analysis of camera equipped devices.
- We evaluate our approach by applying it to a comprehensive set of 9 digital cameras from 9 different brands, examining their multi-source ecosystems (firmware, companion apps, cloud services) and identifying relevant forensic artifacts across local and remote sources.

We publish the WiBtL source code along with supplementary materials from our forensic analysis, including complete version numbers, partition tables, and file trees, available at `https://github.com/mxchhrn/wibtl`.

1.3. Outline

First, we present our taxonomy of digital cameras in Section 2, which is followed by the description of the device selection, their hardware components and the data generation and analysis in Section 3. Furthermore, we present the results of the forensic examination in Section 4. Finally, we discuss the results and the taxonomy and their implications in Section 5, and conclude the paper in Section 6.

2. Taxonomy

The existing literature lacks a consistent, formally accepted definition of digital cameras in general or of digital surveillance cameras in particular. Some works use the term IP cameras (Abdalla and Varol, 2020) to refer to cameras that use the Internet Protocol, while others use terms such as IoT cameras (Bhardwaj et al., 2023), IoT security camera Trabelsi (2022) or IoT smart cameras (Alharbi and Aspinall, 2018). However, since these terminologies partially overlap and lack clear boundaries, we propose a formal definition of the term *digital surveillance camera* and establish its boundaries within the class of digital cameras. To delineate device types and classes, inspired by Hammer et al. (2024), we define four binary properties that devices may or may not satisfy:

- *Primary Imaging (PI):* A device satisfies *Primary Imaging* if capturing images or video recordings constitutes a primary function and, accordingly, the camera module is essential to the device's core functionality.
- *Fixed Installation (FI):* A device satisfies *Fixed Installation* if it is permanently mounted and not readily relocatable (e.g., requires a dedicated power supply or tool-assisted removal).
- *Continuous Recording (CR):* A device satisfies *Continuous Recording* if continuous video recording with the integrated camera module is supported at both hardware and software levels, and data undergoes on-device processing prior to transmission to other devices.
- *Headless Operation (HO):* A device satisfies Headless Operation if it lacks direct physical I/O (displays, buttons, keyboards) beyond minimal reset capabilities.

Based on these properties, we define the device class of digital cameras:

Definition 1 (Digital Camera)**.** A device with a digital camera module (i.e., a module that digitally captures visual data) that satisfies Primary Imaging is designated a *Digital Camera*.

Definition 1 excludes devices in which the camera module is merely an additional, non-essential feature, such as notebooks. On the other hand, devices such as smart doorbells and baby monitors are deliberately included, as their primary function is to capture images and/or record video, and they count as digital cameras within the meaning of the definition. The decision not to exclude such devices is motivated by forensic considerations, as digital cameras produce camera recordings as primary data artifacts, enabling targeted forensic analysis. PI thus delimits the scope of the device class, while FI, CR, and HO differentiate device types within the class and allow conclusions about their expected forensic artifacts. With Definition 2, we offer a narrower characterization of an actual digital surveillance camera.

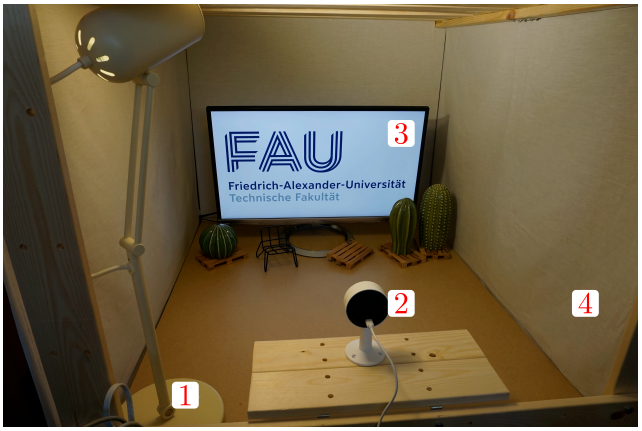


Figure 1: Controlled testing environment with light source (1), digital camera (2) and monitor for displaying images, videos and audio sequences (3), surrounded by the enclosure (4).

Definition 2 (Digital Surveillance Camera)**.** A digital camera that satisfies Fixed Installation, Continuous Recording, and Headless Operation is designated a *Digital Surveillance Camera* (DSC).

Definition 2 establishes the device class of DSCs within digital cameras, which highlights the essential characteristics of these devices independent of transmission protocols (e.g., IP). Each property has forensic implications, allowing conclusions to be drawn about expected artifacts and their types. For devices that meet FI, investigators can assume that their recordings can be assigned to the device's physical location, as otherwise there would need to be physical evidence (e.g., tool marks) or other indicators of a change of location. The CR property excludes devices that lack hardware or software support for continuous environmental recording, thereby limiting the expected artifacts to event-driven recordings. Finally, the HO property further restricts the expected artifacts when it is fulfilled, as artifacts arising from direct user-device interaction are not possible due to the lack of input interfaces. For example, endoscopic cameras (which feature displays and controls) violate HO and are thus excluded from the DSC class.

3. Methodology

In this section, we first describe our device selection methodology and the hardware components identified in these devices. Subsequently, we detail the test data generation process using the WiBtL Python framework and describe the extraction and analysis of the generated test data. To ensure reproducible, interference-free test data across all examined devices, we constructed a controlled testing environment, as illustrated in Figure 1. Our test setup enables precise control of incident light through a dedicated light source and allows predefined video or audio sequences to be systematically presented via a display.

3.1. Device Selection

Table 1 provides an overview of the digital cameras we selected and examined, along with the accompanying apps listed in their manuals. Our device selection strategy followed the taxonomy and three basic principles. First, we preferred cameras with an established market presence over the latest models, as these are more commonly used in real-world forensic investigations. Second, we deliberately included several cameras with shared cloud infrastructure, specifically several devices based on *Tuya* backend services (C_1, C_2, and C_3) and two cameras (C_4 and C_5) that use *Arenti Technology's* infrastructure, in order to evaluate the consistency of forensic artifacts obtained from the network across different brands. Third, we ensured our sample includes well-established brands such as Eufy (C_7) and Reolink (C_8) while achieving broad coverage of different brands and supported companion apps.

Regarding the taxonomy, we applied PI as a mandatory criterion; we exclusively targeted devices whose primary function is imaging. We did not use FI, CR, and HO as selection criteria, so their distribution reflects the composition of the surveillance camera market rather than a deliberate choice. FI-satisfying devices in particular dominate this market segment, as fixed-mount cameras are the most prevalent device type encountered in forensic investigations.

By selection, all examined cameras satisfy PI and qualify as digital cameras under our taxonomy. Aside from C_6, all cameras require external power, and several cameras (C_4, C_6, C_8) feature screw-mount installations. We classify all devices as satisfying FI, since C_6, despite the possibility of battery operation, is intended for fixed, permanent mounting. Furthermore, seven cameras satisfy CR, with C_6 and C_8 as exceptions, because they lack continuous recording capability. While eight cameras satisfy HO, the camera C_6 features user-facing I/O (display and button) and therefore does not. Accordingly, seven cameras (C_{1-5}, C_7, and C_9) meet all DSC criteria, while two cameras (C_6 and C_8) do not. Camera C_6 violates HO due to its I/O interfaces (integrated display and doorbell button). Additionally, C_6 and C_8 violate CR because they support only event-driven recording. This classification guides investigators regarding expected artifact types and their forensic utility. For instance, C_6 and C_8 do not provide continuous recording data, yielding only manual or event-triggered captures.

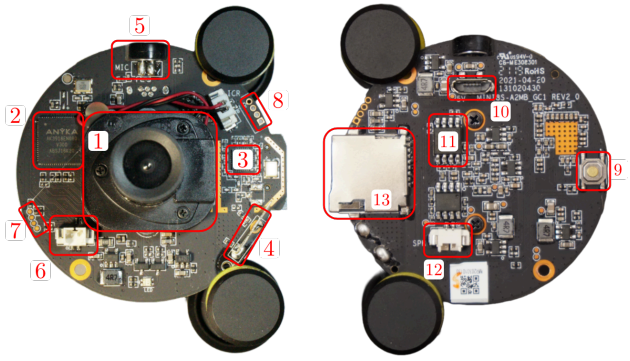
3.2. Hardware Components

Given the wide variety of digital cameras and their functions, the hardware components vary considerably from device to device. Figure 2 illustrates typical hardware components using the Arenti IN1 as a representative example. Core components include the camera module (1) and the system-on-chip (SoC) (2) with accessible UART interfaces (7 and 8). During inspection of the Arenti IN1, we identified a dedicated NOR flash chip (11), a Wi-Fi chip (3), and a microSD card interface (13). Feature-specific components vary by device capabilities and may include microphones (5) for audio recording, IR LEDs (6) for night vision, and

C_i	Brand Model	SoC	UART Shell	UART Logs	Firmware Version	Firmware Dump	Companion App Android ID	Companion App Version	Network Capture
C_1	Nikkei Smart CAM2	AK3918EN080 V300	◐[1]	◐[2]	4.0.8	🖫	Tuya Smart com.tuya.smart	6.3.1	PiRogue Tool Suite
C_2	Woox R4071	RTS3903N	○	●	2.2.5	🖫	WOOX Home com.wooxhome.smart	1.2.2	PiRogue Tool Suite
C_3	Littlelf LF-C1t	RTS3903N	○	●	1.3.8	○	littlelf smart com.littlelf.smarthome	1.4.4	mitmproxy
C_4	Arenti IN1	AK3918EN080 V300	◐[1]	◐[2]	5.6.4[3]	🖫	Arenti com.arenti.smartlife	4.5.2	PiRogue Tool Suite
C_5	Laxihub MiniCam	SSC333	○	◐	5.2.6[3]	🖫	Laxihub com.laxihub.smarthome	4.4.2	PiRogue Tool Suite
C_6	Awow J1	Hi3518 ERNCV300	◐	○	3.1.1[3]	🌐	CloudEdge com.cloudedge.smarteye	5.9.0	PiRogue Tool Suite
C_7	Eufy T8401	T31X	○	◐	2.3.1.6	🖴	eufy com.oceanwing.battery.cam	5.0.60[3]	mitmproxy
C_8	Reolink RLC-410-5MP	NV98515MBG	●	●	3.0.0[3]	⇄	Reolink com.mcu.reolink	4.53.1.0[3]	PiRogue Tool Suite
C_9	Vimtag PTZ Cloud IP Camera	NT98513MBG	◐	●	5.3.3[3]	○	Vimtag com.vimtag.vimtaga	11.9.1[3]	PiRogue Tool Suite

UART shell and logs: ● root shell/continuous debug logs, ◐ limited shell/boot logs
Firmware dump: 🖫 microSD card, 🌐 local web server, 🖴 software tool, ⇄ UART
[1]UART 1, [2]UART 2, [3]version number shortened

Table 1
Overview of the analyzed digital cameras with details regarding the UART interfaces, version numbers and the used network capture tool.



(a) Front view of the PCB (b) Back view of the PCB

Figure 2: Internal components of the Arenti IN1 digital camera: camera module (1), SoC (2), Wi-Fi chip (3), Wi-Fi antenna (4), microphone (5), IR LED connector (6), UART port 1 (7), UART port 2 (8), reset button (9), micro USB (10), NOR flash chip (11), speaker connector (12), and microSD card interface (13).

speakers (12) for 2-way audio. The Arenti IN1 also features a Wi-Fi antenna, a micro-USB port (for power supply), and a reset button. From a forensic perspective, feature-specific components determine the scope of test data generation, while storage-bearing components, the SoC, microSD card, and flash chips, constitute the primary targets for data extraction.

3.3. Data Generation and "What is Before the Lens"

Our test data generation consists of 10 distinct scenarios that cover the diverse feature sets of digital cameras and reflect realistic usage. We extract the internal storage of the cameras, where possible, initially (σ_0), after a firmware update ($\sigma_{1,2}$), and after testing privacy mode (the final scenario, $\sigma_{2,8}$). Table 2 contains a brief description of the 10 scenarios, divided into two action sets. The scenarios vary in interaction complexity, from low-interaction scenarios (e.g., user account creation, firmware updates, microSD card insertion and formatting) to high-interaction scenarios such as "Video Quality", where all quality settings are systematically evaluated with image captures and video recordings for each configuration, following predefined temporal intervals between operations.

Since the systematic, reproducible generation of such scenarios with many interactions is essential for analyzing the number of devices, we use our previously introduced test environment and a Python framework we developed, *What is Before the Lens* (WiBtL). Figure 3 shows the UI of WiBtL, which displays information about the current scenario, the actions already completed in the scenario, and the respective action currently being performed. With WiBtL and the monitor in the test environment, we can systematically test event-detection functionality by playing a video sequence on the monitor with its associated audio track in a reproducible manner. The video sequence we selected for the "Event Detection" and "Privacy Mode" scenarios is part of the WILDTRACK[1] dataset and consists of a snippet

[1] https://epfl.ch/labs/cvlab/data/data-wildtrack/

$\sigma_{i.j}$	Name	Short Description
$\sigma_{1.1}$	Setup	Creation of user account through companion app and device setup; Activation of additional functions (e.g. motion or sound detection) if necessary;
$\sigma_{1.2}$	FW Update	Update firmware if newer version is available;
$\sigma_{2.1}$	microSD Card	Inserting and formatting of microSD card;
$\sigma_{2.2}$	Idle Mode	15 minutes idling without user interaction;
$\sigma_{2.3}$	Video Quality	Recording of image snapshots and video sequences for each available quality setting; Pre-defined sleep times in between the different settings;
$\sigma_{2.4}$	Photo & Video	Recording of image and video sequences during simulated day and nighttime conditions; Pre-defined sleep times in between the different settings;
$\sigma_{2.5}$	Event Detection	Prepared video sequence from WILDTRACK dataset is played to trigger motion and sound detection;
$\sigma_{2.6}$	Doorbell/ Intercom	Scenario depends on camera type; Doorbell camera: (2x) ring doorbell and accept the message for the second try; Other camera: play audio sequence through monitor, recording & playback of the sequence, and playback of smartphone microphone input through speaker of camera;
$\sigma_{2.7}$	Formatting	Reformatting the microSD card via camera settings; Search for recordings before and afterwards;
$\sigma_{2.8}$	Privacy Mode	Activate privacy mode in camera settings; Try to capture an image and video sequence via companion app; Attempt to trigger the camera's event detection by playback of video sequence from WILDTRACK dataset;

Table 2
Description of the scenarios $\sigma_{i.j}$ for test data generation. We extracted the content of the internal storage for each digital camera (if possible) before $\sigma_{1.1}$, after $\sigma_{1.2}$, and after $\sigma_{2.8}$.

of the *Camera 5: GoPro Hero 4* file, which contains a recording of people moving in front of a building, with a corresponding audio track.

3.4. Data Extraction

Test data generation yields three storage dumps per digital camera: the initial state q_0, the state q_1 after action set $\sigma_{1.x}$, and the state q_2 after action set $\sigma_{2.x}$. Figure 4 shows a graphical representation of the three states with their respective transitions as a state machine. We selected these three extraction points because, in many cases, storage extraction is performed via the microSD card, and we must preserve the data on it before each storage extraction. For the designation of states and action sets, we follow Eichhorn and Freiling (2025) and likewise employ differential forensic analysis (Garfinkel et al., 2012) for subsequent data analysis. Table 1 provides an overview of the installed SoCs and the firmware versions present after the respective firmware update (if available), which are partially abbreviated for clarity. For a complete list of version numbers, we refer

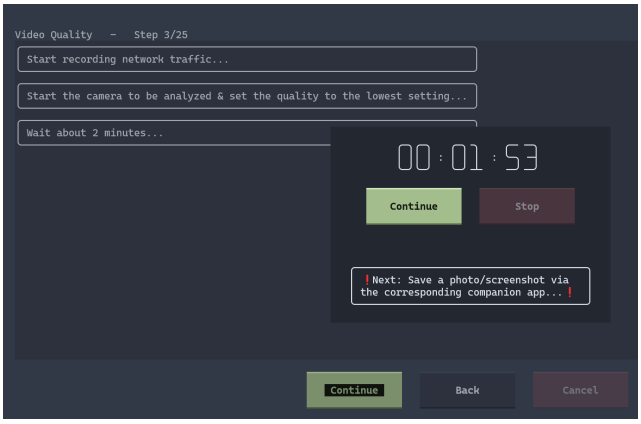


Figure 3: Cropped screenshot from the *What is Before the Lens* (WiBtL) application showing the execution of the "Video Quality" scenario.

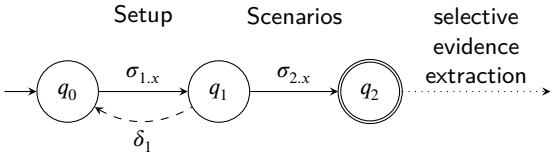


Figure 4: Representation of scenarios and extracted storage images as a state machine with the three states q_0, q_1, and q_2.

to our GitHub repository[2]. Beyond data extraction from internal storage and the microSD card, we also extract data stored by the respective companion app on a rooted Google Pixel 6a (Android 13) and record the network data for the respective cameras and companion apps.

3.4.1. Internal Storage and microSD Card

All of the examined digital cameras feature SoCs with at least one UART interface, with C_1 and C_4 providing a chip with two accessible UART interfaces.

In the two columns of the UART category in Table 1, we indicate shell availability and log message output via one (or both) UART interfaces. Cameras C_2, C_3, C_5, and C_7 do not offer any shell access, while cameras C_1, C_4, and C_9 provide shells with limited functionality, and only camera C_8 offers root shell access via UART. Regarding log messages, cameras C_1, C_4, C_5, and C_7 each display boot process messages, whereas cameras C_2, C_3, C_8, and C_9 can display continuous debug logs. Camera C_6 is the only camera we examined that does not output log messages via the UART interface. Table 1 also contains information on which of the examined digital cameras permitted dumping the internal storage and by which method. We successfully extracted storage from 7 of 9 cameras using 4 methods: microSD card, vendor software, a local web server, and UART. Cameras C_3 and C_9 resisted our attempts to extract data from them.

[2] https://github.com/mxchhrn/wibtl

```
//ppsMmcTool.txt for load address 0x81c08000
style=upgrade,,writeAddr=0,,password=nothing,,
     writeLen=0,,fileName=env;env import 81C08000;
     saveenv,,

//ppsMmcTool.txt for load address 0x20008000
style=upgrade,,writeAddr=0,,password=nothing,,
     writeLen=0,,fileName=f;dstar fw_dump.txt,,

//fw_dump.txt for load address 0x20008000
mmc info
mmc part
sf probe
sf read 0x21000000 0x0 0x1000000
mmc rescan
mmc write 0x21000000 0x3b3ec73 0x8000
```

Listing 1: File contents of `ppsMmcTool.txt` for different load addresses, along with our extraction script, which writes to RAM first and then uses `mmc write`.

MicroSD card (4/7): The most successful approach involved placing files or scripts on the microSD card that execute during boot or trigger special boot modes. Cameras C_1, C_4, and C_5 run ppstrong-based firmware, which allows extraction of internal storage via a `ppsMmcTool.txt`[3] file and *bootloader* variable modification in the `env` file. For C_5, we adapted this approach due to file length limitations of `ppsMmcTool.txt` and a different load address, using a two-stage process: copying internal storage to RAM, then writing RAM contents to the microSD card via the `mmc write` command (see Listing 1). Cameras C_2 and C_3 run tuya-based firmware on the RTS3903N SoC. For C_2, we exploited automatic script execution: placing `tyupg.config` (containing `1`) in `/tuya/upg/` triggers execution of `ty_sdcard_upgrade.sh` from the same directory, which enables storage extraction. This functionality was unavailable on C_3's older firmware version.

Vendor Software (1/7): In addition to using custom scripts on the cameras' microSD cards, for camera C_7, we used the *Igenic USB Cloner*[4] software from the SoC vendor Igenic. This software required connecting the camera via micro-USB OTG cable to enter *USB boot* mode.

Local Web Server (1/7): The digital camera C_6 did not permit full physical imaging. Therefore, we performed logical acquisition via the camera's local web server interface, extracting individual files from internal storage.

UART (1/7): Finally, for camera C_8, we extracted partition images as files via UART using the root shell.

3.4.2. Network Traffic

To record network traffic, we used a Raspberry Pi 4B as a Wi-Fi router and, in 7 of 9 cases, recorded sent and received data packets with the *PiRogue Tool Suite*[5] (PTS). Unlike classic MITM solutions, PTS extracts the TLS encryption keys from the smartphone via adb using *Frida*[6]. For the actual recording of network packets, PTS uses `tcpdump` in the background. Since PTS failed in two cases (C_3 and C_7), we used the MITM proxy *mitmproxy*[7] in conjunction with `tcpdump` for these. Since both cameras' companion apps use certificate pinning, we had to use tools beyond mitmproxy to bypass it. For camera C_3, we were able to turn off certificate pinning at runtime using Frida, and for C_7, we successfully deactivated it using the Magisk module *LSPosed Framework*[8].

4. Results

This section presents our forensic analysis results for the 9 digital cameras in two parts. First, we examine the camera-specific characteristics of network traffic and the challenges encountered during packet analysis (Section 4.1), as understanding these is essential for extracting and interpreting artifacts from other sources. We then examine the artifacts by their location: internal storage (Section 4.2), microSD card (Section 4.3), companion app data (Section 4.4), and cloud API responses (Section 4.5). Furthermore, Table 3 summarizes all identified forensic artifacts by category and location.

4.1. Network Traffic Decryption

A key methodological challenge for cameras that access the Tuya cloud infrastructure (C_1, C_2, and C_3) is the use of payload-level encryption that extends beyond standard TLS and obscures forensically relevant data in the communication between the companion app and the cloud API. In particular, the `postData` and `result` fields in JSON payloads are encrypted (see Listing 2 for examples). Our decryption approach first required extracting the `ecode`, an app-specific but static value, from each companion app using Frida; it can also be identified in network packets during app setup. With the `ecode`, we invoked the appropriate decryption methods via Frida. For the `result` variable, we leveraged the observation that opening the camera via the companion app triggers the `decryptResponse` method, allowing us to extract stored parameters with Frida and perform decryption using the `decryptResponse` method with these parameters, the `requestID`, and `ecode`. For the `postData` variable, we employed the `decryptBytesAppendedNonce2Bytes` method with the encrypted values, `requestID`, and `ecode` as parameters.

4.2. Internal Storage

We successfully acquired the internal memory of 7 of the 9 cameras (see Table 1), thereby achieving at least a logical extraction for these cameras. Partition tables and directory trees are documented in our GitHub repository[9].

[3] https://github.com/guino/Merkury1080P/blob/main/mmc/ppsMmcTool.txt

[4] https://thingino.com/cloner

[5] https://pts-project.org

[6] https://frida.re/

[7] https://mitmproxy.org/

[8] https://github.com/LSPosed/LSPosed

[9] https://github.com/mxchhrn/wibtl

C_i	Brand Model	Device: Information	Device: Configuration	Device: List	Network: Configuration	Network: Wi-Fi SSID	Network: Wi-Fi Cred.	Video/Image Files: Manual Rec.	Video/Image Files: Continuous Rec.	Video/Image Files: Event Rec.	Video/Image Files: Preview/Cached	Event Detection Info.	Cloud API Endpoints	Cloud Cred./Token	Timezone/Locale Info.	User Acc. Cred.
C_1	Nikkei Smart CAM2	◎ ☁	◎ ☁	☁	☁	◎	◎	▯	▯	▯ ☁	▯	☁	◎ ☁	◎ ☁	◎ ☁	–
C_2	Woox R4071	◎ ☁	◎ ☁	☁	☁	◎	◎	▯	▯	▯ ☁	▯	–	◎ ☁	◎ ☁	◎ ☁	–
C_3	Littlelf LF-C1t	☁	☁	☁	☁	–	–	▯	▯	▯ ☁	▯	☁	☁	☁	☁	–
C_4	Arenti IN1	☁ ◎ ▯	◎ ☁	▯ ☁	☁	☁ ◎ ▯	◎ ▯	▯	▯	☁ ▯	▯	▯ ☁	☁ ◎ ▯	◎ ▯	☁ ◎ ▯	–
C_5	Laxihub MiniCam	☁ ◎ ▯	◎ ☁	▯ ☁	☁	☁ ◎ ▯	◎ ▯	▯	▯	☁ ▯	▯	▯ ☁	☁ ◎ ▯	◎ ▯	☁ ◎ ▯	–
C_6	Awow J1	☁ ◎ ▯	◎	▯ ☁	◎ ☁	☁ ◎ ▯	◎ ▯	▯	–	☁ ▯	–	▯ ☁	☁	☁	☁ ◎ ▯	–
C_7	Eufy T8401	▯ ☁ ◎ ▯	◎ ☁	–	☁ ▯	▯ ☁ ◎ ▯	◎ ☁	▯	▯	▯	▯	▯	◎ ☁	–	▯ ☁ ◎ ▯	–
C_8	Reolink RLC-410-5MP	▯	◎ ▯	▯	▯	–	–	▯	–	▯ ▯	▯	–	☁	☁	☁ ◎ ▯	☁ ◎ ▯
C_9	Vimtag PTZ Cloud IP Camera	▯ ☁	☁	–	▯ ☁	▯ ☁	▯ ☁	▯ ☁	▯	▯ ☁	☁	☁	☁	☁	▯ ☁	▯ ☁

Artifacts found in: ◎ internal storage, ▯ microSD card, ▯ app data, ☁ cloud, – no place

Table 3
Overview of identified forensic artifacts and their location for each digital camera.

Across all 7 devices, we identified device configurations and metadata, which include timezone and regional settings. Wi-Fi credentials and device identifiers were present in 6 of 7 cases. For select devices, we identified cloud API endpoints or credentials. We detail infrastructure-specific findings below.

For cameras with Tuya infrastructure (C_1, C_2, and C_3), we obtained storage images for C_1 and C_2, each containing an encrypted key-value store `tuya_user.db` in a JFFS2 filesystem. Although the internal storage of C_1 contained the file `tuya_enckey.db` that may contain an encryption key (see Listing 3), this key could not decrypt the memory. We successfully extracted functional decryption keys from the UART debug messages of C_1, C_2, and C_3, enabling decryption for C_1 and C_2. Since C_1 and C_3 output identical keys, we infer that C_3 probably also contains `tuya_user.db` and that the key does not seem to be device-specific. The C_1 debug key corresponds to a permutation of the `tuya_enckey.db` file content (see Listing 3). For C_2, decryption required only the first 16 bytes of the debug-extracted key (see Listing 3).

We identified several partitions and JFFS2 file systems on the Arenti-based cameras (C_4 and C_5). The file structures were largely identical on both cameras. However, C_4 contained two encrypted files with the suffix `_enc`: `wifi_enc.json` and `mp_config_enc.json`. According to the analysis of C_5's unencrypted counterparts, we assume they contain Wi-Fi credentials (`wifi.json`) and device configuration (`mp_config.json`), respectively. Additionally, C_4's `mp_config.db` file contained X.509 certificates and P2P credentials absent in C_5's version.

4.3. microSD Card

We imaged all microSD cards using *FTK Imager*[10] and analyzed them for forensic artifacts. As documented in Table 3, all 9 cameras contained event recording artifacts, while 7 of 9 contained continuous recording artifacts. Cameras C_6 and C_8 lack continuous recording, which precludes the detection of such artifacts. Manual recordings initiated via the companion apps did not leave any artifacts on the microSD card on any device. In addition, the C_7's microSD

[10] https://exterro.com/digital-forensics-software/ftk-imager

```
//encrypted postData value
"postData": "lecY+HOlm99p8rEAsegVsivB74LLQJv6XPu2Oq
    SyktWlGtcutdz8wKDekUHjd+J9SQPSxM0IeQf7vfI/N4v1"

//decrypted postData value
"postData": {"limit":15,"msgType":1,"offset":0}

//encrypted result value
"result": "XGaSBgQIA9 [...] IVm2Ixpq3hliLeQwQ77"

//decrypted result value
"result": {"datas":[{"actionURL":"tuyaSmart://panel?
     devId=bfb9f555fe661a49e2n3fe","alarmType":1,"
     dateTime":"2025-3-19 14:27","hasNotRead":true,"
     homeId":230503581,"homeName":"My Home ..","icon
     ":[...],"id":833204152,"idStr":"833204152","
     isExpirePics":false,"isExpireVideos":false,"
     isHandleRosettaTemplate":true,"isNeedPullEncKey
     ":false,"msgCode":"SMART_MSG_V3","msgContent":"
     ELMARC FIXY has detected sound!","msgSrcId":"
     bfb9f555fe661a49e2n3fe","msgTitle":"Sound
     Detection","msgType":4,"msgTypeContent":"Sound
     Detection","readStatus":0,"sceneId":"5
     ee9e2140cb46c068eeb49a5","time":1742390863,"uid
     ":"eu17423815874519gUbc"}],"pageNo":0,"
     totalCount":1},"t":1742391032662,"success":true
     ,"status":"ok"}
```

Listing 2: Encrypted and decrypted example values for `postData` and `result` containing a sound detection event in the `result` payload.

```
//tuya_enckey.db
79f9526b1314e2e997a355a2be53fcf1

//Tuya debug message with encryption key
[01-01 18:12:15-- TUYA Debug][simplekv.c:952] get
     encrypt_key[<a397a25553bef1fcf9796b521413e9e2>]

//encryption key for successfull decryption
a397a25553bef1fcf9796b521413e9e2

//Tuya debug message with encryption key
[01-01 00:00:00 TUYA Err][tuya_ws_db.c:341] kvs init
      fails s_tsxH9bj8GiexUIf4xPOzEJFPImqa

//encryption key for successfull decryption
s_tsxH9bj8GiexUI
```

Listing 3: Different variations of encryption keys for `tuya_user.db` for cameras C_1, C_2, and C_3.

card contained a `/log` directory with device log files. These logs include timestamps for detected events (see Listing 4) and firmware update records. Post-format carving analysis revealed varying data retention: no files were recoverable from C_8 and C_9, partial recovery was possible for C_6, and we could completely recover them for C_{1-5} and C_7.

4.4. Companion App

Our analysis of the companion apps revealed that all 9 devices contained manual recording artifacts initiated via

```
2025-06-25 18:07:00.648 [NOTICE][
     local_storage_interface.c:2755] - event record
     or push start, trigger = 0(0:MD 1:Face 2:Cry 3:
     Sd 4.Pet), ch=0, mode=1,app_enc:1,cloud_enc:0
2025-06-25 18:07:00.651 [NOTICE][
     local_storage_interface.c:2330] - create record
      file /media/mmcblk0p1/Camera00/20250625180700.
     dat success
[...]
2025-06-25 18:11:47.631 [WARN][det_crydet_adapter.c
     :157] - current[71] abs=25815 exceed threshold
     =15677 and user_threshold=122
2025-06-25 18:11:47.631 [WARN][det_crydet_adapter.c
     :625] - sound detected!
[...]
2025-06-25 18:11:47.632 [NOTICE][
     local_storage_interface.c:2755] - event record
     or push start, trigger = 3(0:MD 1:Face 2:Cry 3:
     Sd 4.Pet), ch=0, mode=1,app_enc:1,cloud_enc:0
2025-06-25 18:11:47.632 [NOTICE][
     local_storage_interface.c:2330] - create record
      file /media/mmcblk0p1/Camera00/20250625181147.
     dat success
```

Listing 4: Excerpt of the log file `indor.2025-06-25.log` from the microSD card of C_7 containing log messages regarding detected events.

```
<?xml version='1.0' encoding='utf-8' standalone='yes
     ' ?>
<map>
  [...]
  <string name="pass_mall">[PASSWORD]</string>
  [...]
  <string name="user">[MAILADDRESS]</string>
</map>
```

Listing 5: Snippet from the file `mipca_agent_sharedPrefs.xml` from the companion app *Vimtag* of the camera C_9 containing the user account credentials.

the apps. In addition, 7 out of 9 cameras contained preview images and cached media files. Device lists were present in 4 out of 9 companion apps. Table 3 summarizes all identified artifacts by device. Despite device-specific companion apps, findings cluster by infrastructure type: Tuya, Arenti, and others. We present our findings organized by these categories below.

Companion apps for C_{1-3} communicate with the Tuya cloud infrastructure and exhibit significant data structure commonalities. A key artifact present in all three is the encrypted database `thingsmart_stat_encrypted.db`, which we could decrypt using the static key `iot-os`. Since this key succeeded across all three apps, it appears to be infrastructure-wide rather than app- or device-specific. The database's `key_value` table contains timestamps of the last app launch (key: `startupTime`). Additionally, we identified encrypted preview image files `encrypt_thing_camera.jpg` in each app's cache directory. These images are regenerated at each

app launch and exist across all three apps, though only the C_3 *little smart* app renders them. We decrypted these files using the first 16 bytes of the camera ID as a key, Frida, and a JavaScript snippet. We also identified `.xlog` files which resisted analysis.

Similar to Tuya-based cameras, Arenti infrastructure companion apps (C_4 and C_5) share common data structures. Both cameras stored image files for manual captures and automated event detection. Both apps contain the SQLite database `ppstrong.db` with two tables: `ALERTMSG_POINT` and `ALERT_MSG`. Each detected event (sound or motion) generates a new row in `ALERTMSG_POINT` including timestamp, message ID, sound level (for audio detections), and notification read status. Wi-Fi credentials are stored in cleartext in `wifiInfo.xml` within the `shared_prefs` folder.

The companion apps of the cameras $C_{6–8}$ stored temporary image captures in `cache/image_manager_disk_cache/`. The companion app for the C_6 camera is similar to the Arenti infrastructure and contains the file `ppstrong.db`, which includes a table `ALERTMSG_POINT` that is almost identical to those in C_4 and C_5. For C_7, string searches in the file `MegaAppDataBase.db` revealed Wi-Fi login credentials and timezone-specific settings. Other files appear to be encrypted or obfuscated and could not be analyzed. The companion app of C_8 yielded only cached images as forensically relevant artifacts. C_9's companion app contained the most extensive artifacts, including the SQLite database `[ACCOUNTMAIL]replay_download`, which logs video downloads from microSD to the app. Each download generates a row containing the download start time, the local path, and the video start/end timestamps. Wi-Fi credentials appear Base64-encoded in `hisSidPwd_key` and in cleartext (along with user ID and email) in `param_web_sharedPrefs.xml`. Furthermore, we could identify the user account credentials in plaintext in `mipca_agent_sharedPrefs.xml` (see Listing 5) and the public IP address of the Internet connection in the `binnet_log` file.

4.5. Cloud API

During test data generation, we analyzed requests and responses to the cloud API, and we found that no camera sent continuous recording to the cloud API endpoints in any of the 9 cases. However, event detection media (images/videos) were uploaded to cloud storage for 7 of 9 cameras. Additionally, C_9 transmitted preview images and manual recordings. Table 3 summarizes all identified cloud API artifacts, and below, we present detailed findings organized by cloud infrastructure.

Tuya infrastructure communication ($C_{1–3}$) uses HTTPS with JSON payloads. As detailed in Section 4.1, the payloads can employ additional encryption. Listing 2 shows example `postData` and `result` values, demonstrating that data related to detected events is stored in the cloud and returned in the `result` value. Information retrieval via cloud API requests uses a common JSON payload structure differentiated by the key `a`. For example, C_1 queries event detection messages using the `smartlife.m.msg.list` value. The endpoint values vary slightly between apps, but provide equivalent data:

```
"alertMsg": [{
    "msgID": "4921842247",
    "eventType": "6",
    "aiPasswordStatus": null,
    "userID": 100001114568,
    "deviceID": 10001032451,
    "decibel": "76", [...]
    "imageUrl": [IMG_URL],
    "eventTime": "20250318190157",
    "storageType": "0",
    "defaultImage": "1"},[...]],
```

Listing 6: Excerpt from the JSON payload of the response of a Cloud API request to `/v3/app/event/list` (C_4) containing a detected sound event (`eventType` 6) with it's preview image.

```
//HTTP request
"page": "SetWiFiPwd",
"time": 1750157645,
"content": "get_connected_wifi",
"response": "eyJzc2lkIjoiV [...] OiA8bm9uZT4ifQ=="

//decoded (Base64) response key
{"ssid":[SSID],"BSSID":"2c:cf:67:36:84:1b","gatewayIp
    ":"10.8.0.1","rssi":-36,"level":4,"frequency
    ":2437,"wifi_info":"SSID: [SSID], BSSID: [BSSID],
     MAC: 02:00:00:00:00:00, IP: [PUBLIC-IP],
    Security type: 2, Supplicant state: COMPLETED,
    Wi-Fi standard: 11n, RSSI: -36, Link speed: 72
    Mbps, Tx Link speed: 72Mbps, [...] Rx Link speed
    : 72Mbps, [...] isUsable: true, CarrierMerged:
    false, SubscriptionId: -1, IsPrimary: -1,
    Trusted: true, Restricted: false, [...]}
```

Listing 7: Excerpt from the JSON payload of the response from the *Vimtag* Cloud API of camera C_9 containing Wi-Fi SSID and public IP address.

device inventory queries use `m.life.my.group.device.list` (C_1), `m.life.my.group.device.sort.list` (C_2), and `tuya.m.device.get` (C_3). Unlike C_2, C_1 and C_3 encrypt uploaded images with AES-CBC before uploading. The decryption of these images requires the key exchanged during local camera-app communication and the IV stored in the first 16 bytes of each image.

For Arenti infrastructure cameras (C_4 and C_5), network traffic analysis revealed that event detection triggers cloud image uploads. Listing 6 presents the payload of a sample HTTPS response from `/v3/app/event/list` containing an image download URL and event metadata, including a timestamp and event type (`1` = motion, `6` = sound). Requests for event detection data are transmitted via unencrypted HTTP PUT. Additionally, C_4 encrypts preview images (transmitted as `.jpgx3`) while C_5 transmits unencrypted JPEGs.

For camera C_6, communication with the cloud API occurs in both directions via HTTPS, and stores event preview images. As a smart doorbell, C_6 detects doorbell presses (event type `3`) in addition to motion (`1`) and sound (`6`). Camera C_7 communicates analogously to the cameras of the Tuya

infrastructure, also via HTTPS with an additionally partially encrypted payload, which resisted Frida-based analysis due to app crashes triggered by Frida's presence. Furthermore, we identified in the network traffic from the camera to the cloud API a file named `indoor_log.tar.gz` that is transmitted and strongly resembles companion app log files, like `indoor.log` (see Section 4.4). We infer, therefore, that this is a (maybe truncated) copy of the device log, though we could not verify completeness. While C_8's local web server (disabled by default post-update) used HTTP, cloud API communication uses HTTPS. We identified no forensically relevant cloud API artifacts for C_8. However, camera C_9 transmits Wi-Fi SSID and public IP address to the cloud during account setup via a response from a request to the endpoint `/ccm/cdfs_set.js` (see Listing 7).

5. Discussion

Despite minor device-specific deviations, our results presented in Section 4 show considerable similarities between the cameras. The similarities between the respective forensic artifacts are shown in Table 3. Cameras C_4 and C_5 show identical table entries across all data categories and artifact locations for their forensic artifacts, but exhibit minimal differences upon closer examination of their extractable forensic artifacts. Although some generalization across camera subgroups is possible, device-specific analysis remains necessary for a comprehensive forensic investigation.

In Table 1, the digital cameras are not necessarily distinguishable devices and differ only with respect to their forensic artifacts in Table 3. We already demonstrated (see Section 4) that individual devices use the same cloud infrastructure and can be roughly grouped by this characteristic. If we further apply our taxonomy, we can make additional distinctions between the individual devices. To this end, we examine the presented properties and classify the devices accordingly as either fulfilled or not fulfilled.

As described in Table 3, C_6 and C_8 do not satisfy the Continuous Recording (CR) property and are consequently not classified as DSCs, whereas the remaining seven cameras could be classified as DSCs. A forensic investigator can already make these first assessments based on device specifications, which is supported by our results (see Table 3). Accordingly, no continuous recording artifacts can be found for C_6 and C_8. Thus, a classification based on our taxonomy can provide an initial indication of expected artifacts and serve as a decision-making aid to determine whether a device is likely to yield relevant artifacts or should be deprioritized in favor of other devices. Eichhorn et al. (2026) also propose an approach to estimate the utility of forensic device analysis using their *User-Device Interaction Model* (UDIM). Such a model, applied to the different device classes within the taxonomy, can provide a functional preliminary assessment of expected forensic artifacts.

In particular, each property provides a concrete indication of the expected artifact types. PI serves as a filtering criterion, narrowing the investigation to devices where audio and video recordings constitute the primary artifacts. With FI, recordings can be attributed to the camera's physical installation location, provided there is no physical evidence of relocation (e.g., tool marks). CR allows an investigator to assess, already before the analysis, whether a search for continuous recording footage is likely to be successful; as our results confirm, no such artifacts were found for C_6 and C_8 (see Section 4.3), consistent with the fact that neither device satisfies CR. If HO is not satisfied, as is the case for C_6, an investigator can additionally search for artifacts related to direct user-device interaction, which cannot exist on headless devices.

Furthermore, the Python framework presented in this work (see Section 3.3), in conjunction with our testing environment, reliably and automatically triggered events and executed scenarios reproducibly across a variety of cameras during test data generation. Without such a controlled testing setup, unshielded environmental influences could unintentionally trigger events and thus generate erroneous test data. Furthermore, the manual execution of scenarios could lead to unintended temporal deviations within the test data. Due to the focus of this work, the Python tool is limited to digital cameras and the scenarios we have defined. However, such an approach could also significantly facilitate test data generation in forensic examinations of other device classes.

6. Conclusion

In this work, we proposed a taxonomy for *Digital Surveillance Cameras* (DSC) based on four binary properties: *Primary Imaging* (PI), *Fixed Installation* (FI), *Continuous Recording* (CR), and *Headless Operation* (HO). These properties formally define DSCs and distinguish them from related device classes such as notebooks, doorbells and endoscopic cameras. Each property has possible forensic implications, enabling prediction of expected artifact types. PI scopes the investigation to devices where recordings constitute the primary artifacts; FI allows investigators to attribute recordings to a physical location; CR indicates whether continuous recording footage is to be expected; and HO limits the expected artifacts by excluding those arising from direct user-device interaction.

Additionally, we developed a Python framework *What is Before the Lens* (WiBtL) to support systematic and reproducible test data generation for digital cameras across our predefined scenarios and made it available in our GitHub repository[11].

Furthermore, our multi-device analysis (9 cameras) revealed heterogeneous artifacts distributed across internal storage, microSD card, companion app data, and cloud API responses. Identified artifacts include Wi-Fi credentials, device, and network information, preview images, and event detection records. We successfully decrypted encrypted artifacts in some cases where additional encryption was employed, including Tuya infrastructure payloads and encrypted preview images.

[11] https://github.com/mxchhrn/wibtl

Furthermore, the analysis could be extended to devices that do not satisfy PI or FI to further validate the taxonomy across a broader range of device classes. Additionally, the accuracy of event detection across different devices could be examined in future work.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback. We also thank Jonas Röckl and Julian Geus for their comments on an earlier version of this paper. This work has been supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Research and Training Group 2475 “Cybercrime and Forensic Computing” (grant number 393541319/GRK2475/2-2024).

CRediT authorship contribution statement

Maximilian Eichhorn: Conceptualization, Methodology, Validation, Formal Analysis, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization, . **Felix Gabel:** Methodology, Software, Validation, Visualization . **Felix Freiling:** Resources, Writing - Review & Editing, Supervision, .

References

Abdalla, P.A., Varol, C., 2020. Testing IoT Security: The Case Study of an IP Camera, in: 2020 8th International Symposium on Digital Forensics and Security (ISDFS). doi:10.1109/ISDFS49300.2020.9116392.

Alharbi, R., Aspinall, D., 2018. An IoT analysis framework: An investigation of IoT smart cameras’ vulnerabilities, in: Living in the Internet of Things: Cybersecurity of the IoT - 2018. doi:10.1049/cp.2018.0047.

Bhardwaj, A., Kaushik, K., Bharany, S., Kim, S., 2023. Forensic analysis and security assessment of IoT camera firmware for smart homes. Egyptian Informatics Journal 24. doi:10.1016/j.eij.2023.100409.

Dragonas, E., Lambrinoudakis, C., Kotsis, M., 2023. IoT forensics: Analysis of a HIKVISION’s mobile app. Forensic Science International: Digital Investigation 45. doi:10.1016/j.fsidi.2023.301560.

Dragonas, E., Lambrinoudakis, C., Kotsis, M., 2024. IoT forensics: Exploiting log records from the DAHUA technology CCTV systems. Journal of Forensic Sciences 69. doi:10.1111/1556-4029.15401.

Eichhorn, M., Freiling, F., 2025. Dial M for Mixer: A Methodological Approach to Forensic Analysis of Unknown Devices Using the Thermomix TM6. Forensic Science International: Digital Investigation doi:10.1016/j.fsidi.2025.301983.

Eichhorn, M., Hammer, A., Pugliese, G., Freiling, F., 2026. Udim: Formal User-Device Interaction Model for Approximating Artifact Coverage in IoT Forensics, in: Proceedings 2026 Workshop on Security and Privacy in Standardized IoT, Internet Society, San Diego, CA, USA. doi:10.14722/sdiotsec.2026.23050.

Garfinkel, S., Nelson, A.J., Young, J., 2012. A general strategy for differential forensic analysis. Digital Investigation 9. doi:10.1016/j.diin.2012.05.003.

Hammer, A., Geus, J., Nicolai, F., Schütz, P., Fein, C., Freiling, F., 2024. Fit for Forensics: Taxonomy and Common Model for Forensic Analysis of Fitness Trackers. Digital Threats 5. doi:10.1145/3687271.

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

Salem, Y., Hamarsheh, M.M.N., 2024. Forensically analyzing IoT smart camera using MAoIDFF-IoT framework. Forensic Science International: Digital Investigation 51. doi:10.1016/j.fsidi.2024.301829.

Salem, Y., Owda, M., Owda, A.Y., 2024. Towards Digital Forensics 4.0: A Multilevel Digital Forensics Framework for Internet of Things (IoT) Devices. International Journal of Wireless and Microwave Technologies 14. doi:10.5815/ijwmt.2024.02.03.

Seralathan, Y., Oh, T.T., Jadhav, S., Myers, J., Jeong, J.P., Kim, Y.H., Kim, J.N., 2018. IoT security vulnerability: A case study of a Web camera, in: 2018 20th International Conference on Advanced Communication Technology (ICACT). doi:10.23919/ICACT.2018.8323686.

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 Science International 348. doi:10.1016/j.forsciint.2023.111674.

Stachak, M., Geus, J., Pugliese, G., Freiling, F., 2024. Nyon Unchained: Forensic Analysis of Bosch’s eBike Board Computers, in: Digital Forensics Research Conference Europe (DFRWS EU’24). doi:10.48550/arXiv.2404.12864.

Trabelsi, Z., 2022. Investigating the Robustness of IoT Security Cameras against Cyber Attacks, in: 2022 5th Conference on Cloud and Internet of Things (CIoT). doi:10.1109/CIoT53061.2022.9766814.

Valente, J., Koneru, K., Cardenas, A., 2019. Privacy and Security in Internet-Connected Cameras, in: 2019 IEEE International Congress on Internet of Things (ICIOT). doi:10.1109/ICIOT.2019.00037.