Unveiling the Shadows: An Approach towards Detection, Precise Localization, and Effective Isolation of Concealed IoT Devices in Unfamiliar Environments

EMILIIA GELOCZI, University of Passau, Germany
HENRICH C. PÖHLS, University of Passau, Germany
FELIX KLEMENT, University of Passau, Germany
JOACHIM POSEGGA, University of Passau, Germany
STEFAN KATZENBEISSER, University of Passau, Germany

Internet of Things (IoT) devices have been reportedly used to spy on guests in hotels or privately rented accommodations in recent years. Such attacks can be mitigated by users, especially those at higher security risk, by detecting such devices, localizing them, and subsequently disabling their intrusive functionality. Our paper introduces a system that allows a user without specific technical knowledge not only to detect and physically locate hidden IoT devices but further also to isolate devices that are perceived as threats from the network without physical intervention. For the former, the system provides a visual aid using augmented reality, for the latter we integrate de-authentication attacks on the Wi-Fi network based on the network information collected during device localization. The prototype of the system is implemented as an iOS application, connected to a Raspberry Pi, and evaluated on a set of 18 different test devices. Compared to previous research our system shows comparable or even better results. It discloses devices within 3s with a detection rate of at least 90%, localizes them in 3D space visualized using augmented reality with an approximate accuracy of 0.4m, and isolates 100% of the devices for at least 180s (and potentially longer).

CCS Concepts: • **Security and privacy** → **Privacy protections**; • **Hardware** → *Wireless devices.*

Additional Key Words and Phrases: Privacy; IoT; Spy; Hidden Devices; Detection; Localization; Isolation

1 Introduction

The growing number of Internet of Things (IoT) devices – the IoT market is set to reach 483 billion US$ between 2022 and 2027 [20] – has woven a seamless web of devices that surround us throughout our daily lives. However, the rapid growth and widespread deployment of IoT devices in everyday surroundings – the global average was 17.1 devices per home in 2022 [33] – have also introduced well-known and unprecedented challenges, especially in the context of security and privacy [7, 10, 16, 22]. As these devices are included in various aspects of our lives especially those of at-risk users, ensuring that our digital environments are as secure as expected becomes paramount. At-risk users are those users who experience elevated digital security, privacy, and safety threats because of what they do, who they are, where they are, or who they are with, for example, journalists, activists, survivors of intimate partner abuse, or children [44]. In recent years, there have been many cases of guests in hotels and apartments being spied on with physically hidden IoT devices [7, 16].

This paper addresses the pressing need for an approach to unveil the shadows of physically concealed IoT devices in unfamiliar environments, i.e., those environments that users have no full control over. We provide a system that, first, guides users by augmented reality view (ARV) to pinpoint the location of detected Wi-Fi devices. They are then able to decide if the device poses a threat to their privacy based on the device they might see or not see in the physical environment together with additional information provided by the system about the device, i.e., manufacturer.

Authors' Contact Information: Emiliia Geloczi, University of Passau, Passau, Germany, emiliia.geloczi@uni-passau.de; Henrich C. Pöhls, University of Passau, Passau, Germany, hp@sec.uni-passau.de; Felix Klement, University of Passau, Passau, Germany, felix.klement@uni-passau.de; Joachim Posegga, University of Passau, Passau, Germany, jp@sec.uni-passau.de; Stefan Katzenbeisser, University of Passau, Passau, Germany, stefan.katzenbeisser@uni-passau.de.

Moreover, it is critically important that users, especially those at-risk, are able to disarm privacy-invasive IoT devices in places where they cannot physically disarm devices without leaving traces or risking punishment. To address this issue, we include in our system the functionality to isolate suspicious IoT devices from the network, which distinguishes our methodology from existing solutions. Our approach enables users to effectively isolate concealed IoT devices, i.e., disable their network connectivity, ensuring a robust defense. This protects sensitive information, maintains privacy, and prevents live data from being sent directly over the network[1].

The novelty of our all-in-one approach lies in its capability to

- effectively isolate devices using denial-of-service attacks based on network information gathered in previous stages (see Section 4.1),
- run the localization stage iteratively based on persistent storage of previous runs to allows fine-grained detection if needed. A first rough localization gives a quick overview, and the persistent storage allows consecutive runs for more precision in the same physical location if deemed necessary by the user (see Section 4.1), and
- offer better performance compared to existing approaches (see Section 7).

The rest of this paper is organized as follows. Section 2 gives an overview of the existing literature relevant to our research. Section 3 provides a scenario description. Section 4 describes the design and workflow of the proposed system. We then detail the technical choices to constrain our prototype and the general and specific limitations of such approaches in Section 5. Section 6 presents the results of evaluating our system in a testbed of 18 commercial-of-the-shelf devices. The comparison of our approach with related work[2] is provided in Section 7. Finally, Section 8 presents the concluding remarks of the paper. Additional technical details are presented in the Appendix.

2 Related Work

In this section, we provide an analysis of the existing research related to our work.

Device Detection and Classification. There are many different approaches to detecting the presence of IoT devices in space and identifying their types, for example, the authors examine intercepted Wi-Fi traffic from devices and detect cameras with AI technologies [26, 38, 45] and without AI [12, 25, 34]. Despite the high performance of such approaches, they are mostly applicable only if the cameras are constantly transmitting video, which is not always the case, especially with battery-powered devices. Apart from traffic-based approaches, there are proposals which use camera's features, e.g., electromagnetic emanations [27] or smartphones' time-of-flight sensors [35].

Device Localization. Besides works focused only on the detection and type identification of IoT devices often focused solely on cameras, there are also many works dedicated only to the localization of different IoT devices. Some works use RF multiplexer [40], others are based on DV-Hop algorithms [46], etc. But, most of the approaches use the received signal strength (RSS) from the radio interface, e.g., [29, 32, 41].

Device Isolation. There are also solutions that allow the isolation of vulnerable, compromised, or malicious IoT devices. In [14], the authors place vulnerable devices in the sub-network to isolate them from trusted ones. Cilleruelo *et al.* mention the positioning of the device in a closed environment as a short-term solution to protect the device itself. However, besides placing the

[1]Note: Technically, a recording device could store information locally, preventing a network connection at least would prohibit live streaming of that information.

[2]We offer a direct comparison of our approach (see Table 4 in Section 6) with SCamF [19], LocCams [15], I&LComDev [17], SnoopDog [39], MotionCompass [18], AHCL [24], Lumos [37], and PanguVision [23].

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

target device in a separate environment, an attack against the compromised device itself can be performed, e.g., de-authentication attack [28], which allows to isolate the device from the network without creating a network or even being connected to the same network as a device.

Summary. Of course, the above methods could be used to provide additional support for an individual stage, like detection or localization.

Combined Approaches. In the rest of the paper, we focus on approaches that combine the stages of device detection, classification, and localization into one system. In [19], the Spy Camera Finder (**SCamF**) is proposed by Heo *et al.* SCamF analyses Wi-Fi traffic to detect the presence of a camera within a network and reconstructs video frame sizes to localize the camera. During tests, the authors achieved the following results: low false positive rates, centimetre-level distance errors during localization, 98% and 97% of classification and detection accuracy, respectively. However, SCamF can detect only live-streaming cameras and localization performance highly depends on the camera position in the room.

Gu *et al.* present an approach called **LocCams** to detect and localize hidden cameras [15]. Detection is performed by monitoring packet transmission rates. For localization, channel state information subcarriers and convolutional neural networks are used. The evaluation showed that 95.12% and 92% of detection and localization accuracy can be reached within 30s. However, this approach is effective only against Wi-Fi cameras using the Variable Bit Rate (VBR) encoding algorithm [42] and requires root access, which can cause security problems.

Identification and localization of IoT devices based on analysis of encrypted smart home traffic is described in [17] (**I&LComDev**[3]). Guo *et al.* use AI tools (i.e., convolutional neural networks) for device identification and process RSS indicators obtained during traffic monitoring for AI-based localization prediction. In experiments, the authors achieved a mean position estimation error of 1.34m and up to 98.7% accuracy for device classification. Limitations of this approach include the need for training datasets, applicability only to live streaming devices, and the ability to perform only coarse localization.

In [39], Singh *et al.* propose a framework, named **SnoopDog**, for the detection, classification, and localization of IoT devices with sensors. SnoopDog detects a correlation between the user's movements and the information transmitted by the sensor to spot the presence of sensors. Classification is based on the device's MAC address. Localization is performed through a trial-based approach for each individual sensor allowing to narrowing of a possible area of the sensor location. Performance evaluation showed that the given approach can provide the detection accuracy of the snooping devices up to 95.2% and determine the approximate location of the devices. Among the limitations of the approach, we can note that it is effective only against devices based on VBR algorithms and Wi-Fi, and the vulnerability of the approach to some attacks (e.g., noise injection, adding delays to data transmission).

He *et al.* introduce a technique called **MotionCompass** to detect and localize wireless cameras with motion sensors [18]. Detection of cameras is performed by analyzing traffic flows triggered by some movement. Then, identification is based on the device's MAC addresses which are used for MAC Spoofing for the classification of captured traffic patterns among cameras. Moreover, monitoring the number of transmitted packets allows authors to calculate the camera position with an average localization error of 5 cm within 140s. However, this approach is only applicable to Wi-Fi cameras and requires a motion trigger to activate the camera. Also, MotionCompass was tested only for default camera settings, whereas some cameras allow the user to customize activity zones for different conditions, which can negatively affect operation results. Also if the camera

[3]A short title has been assigned to this article for ease of future reference.

is wide-angle and covers the whole room, a correct correlation between trigger and transmission patterns may not be obtained.

Another approach for detecting and localizing cameras, named **AHCL**, is presented by Lee *et al.* in [24]. The authors derive video information packets from raw network traffic over an Ethernet connection. They then extract moving objects and feed them into the room classification model trained with videos of the observed rooms. This method allows the identification of the room where the camera is placed as long as the training set of the classifier contains information from that specific room, which is why the classifier model should be trained in advance. In addition, the user must have access to an Ethernet connection. Both factors limit the use of this method by users who do not have network knowledge or do not have access to the room in advance.

The general design of this work follows the idea proposed by Sharma *et al.* in [37]. The authors describe a system named **Lumos**, which is designed to detect and locate hidden Wi-Fi devices in unknown environments. This system monitors the Wi-Fi traffic to detect the devices in reach, localize them, and identify the device type. Test results show that using the given system 100% of the physically hidden Wi-Fi devices can be detected and 95% of them can be correctly classified, moreover, the achieved average localization error is 1.5m. However, Lumos has a few limitations. First, the identification of the devices performs well only if a similar device is present in a training data set. Second, this approach is limited to be used only for Wi-Fi devices. Moreover, theoretically, Lumos is applicable in many situations, as no information about the environment is required in advance. However, the Lumos mobile application prototype includes hard-coded information, which restricts it to be used only within the specific test environment.

The authors of [23] propose a demo prototype **PanguVision** for the detection and localization of physically concealed devices. Ju *et al.* combine a multi-armed slot machine approach with the greedy principle for Wi-Fi traffic monitoring to perform device identification. A multi-point localization algorithm based on RSSIs collected during monitoring is used for device localization. The authors claim that PanguVision can detect and identify all hidden devices. Furthermore, the performance evaluation shows that the given prototype can achieve an approximate average localization error of 0.8m and classification with 98% accuracy. Limitations of the proposed prototype are not mentioned.

Summary. Although the above-mentioned approaches each have their features and are quite promising in terms of different characteristics, most notably none of them provide the user functionality to isolate (i.e., disconnect temporarily) the device from the network without physically interacting with the device, to resume interrupted fine-grained localization based allowed by the presence of persistent storage, and provide the user with the ability to interact with the device in any way (sort, mark, etc.) from the application interface.

3 Scenario Description

An adversary can facilitate IoT devices to violate the privacy of users in unfamiliar locations, e.g., restaurants, hotels, apartments, and private houses [7, 16]. In the following section, we explicitly specify the peculiarities of this environment for this paper, alongside three markers: necessary equipment, access to physical environments, and access to the network. We take them into account to model both involved parties: adversaries and users. Especially, at-risk users like journalists, activists, and survivors of intimate partner abuse [44], who experience elevated digital security, privacy, and safety threats need a way to easily locate IoT devices in those physical environments and then isolate devices identified by them as a threat. In order to eliminate misunderstandings we also briefly model the adversary and the additional defensive capabilities of the user.

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

3.1 Environments

Our solution focuses on unfamiliar environments.

Unfamiliar environments: These locations are particularly vulnerable to privacy attacks as multiple parties (e.g., a host, staff, previous guests) have or have had access to them and therefore each of these parties potentially can install a spying device. So the adversary has very good access to the physical environment to place devices. Contrary, the adversary also has limited influence on the network environment, e.g., in a hotel the Wi-Fi network is not managed by guests or hotel staff. The user has very limited control over neither of them, e.g., can not physically remove cameras from high ceilings.

Familiar environments: Contrary, in these locations the user exercises control over the physical and network environment. Hence, foremost unsupervised third parties have limited access to the actual physical environment, e.g., a user's private house, and a user can simply forcefully remove unwanted devices. Moreover, users also exercise increased control over their own networks' security.

3.2 Scenario Parties: User and Adversary

In the following, we provide generalized models of the two parties involved in our scenario: a user and an adversary. For each, we state their capabilities in terms of the equipment and the access to the physical environment as well as the network (see Figure 1).

3.2.1 Adversary Model. Many different adversaries can have a fairly wide range of resources and pursue various goals. In this paper, we focus on an adversary that has the following capabilities:

Equipment. The adversary uses commercially available devices (e.g., security cameras or automatic doorbells), which are most common on well-known marketplaces such as Amazon. These devices are typically connected wirelessly to the Internet. In addition, the attacker does not modify the behavior of these devices other than changing settings or deploying new firmware versions. In our view, this is a reasonable assumption for an attacker who has no deep technical knowledge.

Access to the physical environment. The adversary has control of the environment before the user arrives including the installation and hiding of the monitoring devices, e.g., by placing them in other objects, such as smoke detectors, which are a popular choice to conceal a device [7].

Access to the network environment. It is assumed that the adversary is able to connect the hidden devices to any available Wi-Fi network or set up own networks. To disguise the network used, the adversary can either hide the network name or create multiple new Wi-Fi networks to create a distraction.

3.2.2 User Model. Similarly to the adversary, we focus on the following type of user:

Equipment. The user has access to commercially available devices that are either already in everyday use, such as smartphones, or are at least inexpensive and easily available, like a Raspberry Pi.

Access to the physical environment. The user can move freely in private and public areas of unfamiliar environments (see Section 3.1), e.g., in a hotel room or lobby. They are however not entirely capable of physically changing the environment, e.g., can not use ladders to reach high ceilings or physically remove/damage devices hidden in smoke detectors.

Access to the network environment. Finally, a user's access to the Wi-Fi network configuration is not granted. We assume the user has access to a separate "guest" network. Moreover, the user has no knowledge of how many and which devices are connected to which network and might not always already visually see where these devices are located in the unfamiliar space.

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

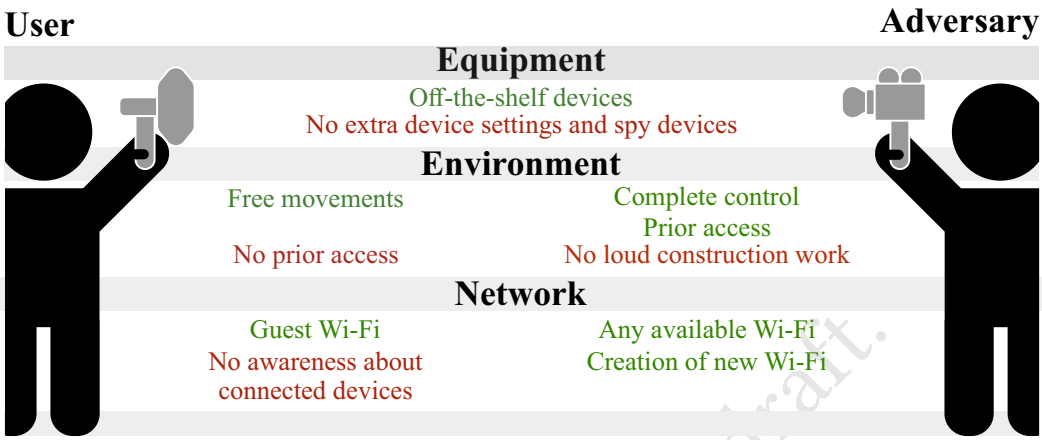
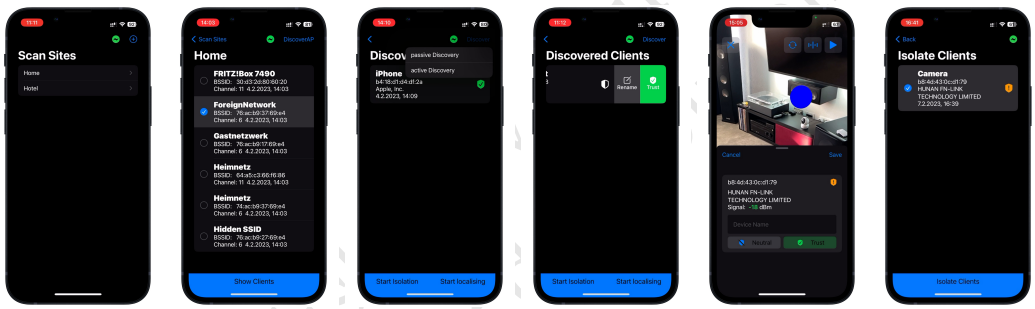


Fig. 1. Generalized user and adversary models.



(a) (b) (c) (d) (e) (f)

Fig. 2. (a) Scan-Sites View (b) Detected-APs View (c) Device Detection View - Options (d) Device Detection View - Renaming and Labeling (e) Localization View (f) Isolation View.

4 System Design and Workflow

In this section, we discuss the operation workflow and design of our proposed system. As previously described, our system allows detection of the presence of physically hidden Wi-Fi devices and their location in a room[4]. In addition, a distinctive feature of our system compared to existing ones is that the user is able to interact with the detected device, e.g., name it, mark it as a threat, and in most cases, even isolate it from the network.

4.1 Operation Workflow

In this section, we describe the operation workflow of the proposed system which consists of six stages (see Figure 3).

Stage 1: Naming the Environment. First, a user needs to choose a name for the environment to be scanned, either from any previously saved one or create a new one. We assume that the user may

[4]A video of the augmented reality view of the prototype is available: https://anonymous.4open.science/r/Detection-Localization-and-Isolation-02E5

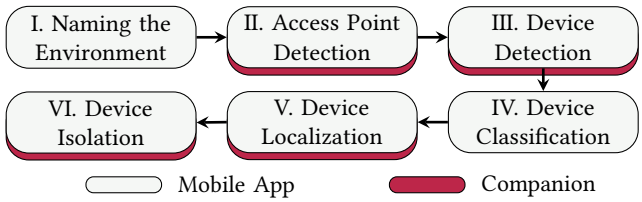


Fig. 3. Six-Stage Workflow. Companion marks a stage's need for having low-level access to wireless radio.

also want to scan multiple environments or one environment multiple times (see Figure 2(a)). The information collected during a scan is associated with the selected environment. Hence, a user can perform iterative scans.

Stage 2: Access Point Detection. During this stage, our system collects all Wi-Fi networks within range facilitating the Access Points' (AP) identifier(s) (see Figure 2(b)). To generate a list of networks that can be selected for the next stages, the following steps are performed:

(1) Capturing beacon frames, which are broadcasted as a form of advertisements by each AP at fixed time intervals.
(2) Extracting information about the network address of the APs (BSSID), network name (SSID)[5], and the used network channel from the received beacon frames.

Stage 3: Device Detection. To produce an overview of the devices that are connected to the network interested for the user (see Figure 2(c)), the following steps are performed:

(1) Capturing Wi-Fi traffic of the networks of interest to find device addresses connected to them. These addresses are used as identifiers for the detected devices. Information like the time of the last detection is stored as well.
(2) The relationship between the device and its associated AP is stored to allow the system to select the correct network channels in the subsequent steps.
(3) After detecting all connected devices, the system identifies the active ones using passive (P) and/or active (A) approaches:
   (P) Passively sniff the traffic and analyze the captured packets in order to find source or destination addresses that match any of the specified BSSIDs. This strategy is applicable when devices send or receive frequently, as the system can quickly capture a packet for each device.
   (A) Actively broadcast a spoofed de-authentication frame, which de-authenticates all active devices connected to the targeted AP. These devices should then re-authenticate to the AP to resume their connection [8]. The system then passively sniffs the generated traffic to extract the network addresses of the re-connecting devices.

   The active strategy can reduce scan time and improve the detection of devices that send less frequently, despite the offensive nature of the technique. However, both methods are offered to the user, as the disconnection during the active method can cause problems in some scenarios and the user may also wish to avoid it to stay more stealthy. Passive device detection additionally provides a valuable fallback solution if a network uses protected management frames (PMFs) and hence the active method would not work.

[5]BSSID is used as an identifier for the network, while SSID is collected to provide the user with a human-readable network name.

Note, that these steps can be repeatedly done for several networks of interest under the same environment name.

Stage 4: Device Classification. With a list of devices and Wi-Fi-related information about them from the previous stages, this stage seeks to identify and classify the device, e.g., by type like a camera, or by the device manufacturer. This step can range in its implementation from very sophisticated techniques that provide information about the device to the user, i.e., using databases and also AI [24, 37], to those that involve manual interaction with the user. In our prototype, we opted to not use AI-based methods but still offer a simple MAC-based vendor lookup[6] to the user. Thus it performs the following steps:

(1) The user assigns a custom name to each detected device.
(2) The user can also mark it as a threat (see Figure 2(d)).
(3) Attempting to identify the device manufacturer by checking the network address for a registered Organizationally Unique Identifier and providing this to the user.

Stage 5: Device Localization. During this stage, the proposed system provides the functionality to find the device positions within the physical environment (see Figure 2(e)). In order to achieve this, the position of each detected device is estimated using the Wi-Fi radio signal and reflected on ARV. This allows the user to manually search for the potentially hidden device. From the user's point of view, we can classify the localization process into types: rough and precision localization. During the first type of localization, the user is asked to go slowly around the environment, which can give a first overview of possible device positions. In the second type of localization, the user can go around the previously estimated position of the device of interest to pinpoint its position. The system performs the following steps:

(1) Signal Strength Measurement (SSM): To make localization work in unknown environments, the RSS from each device is measured at different locations in the room by sniffing the Wi-Fi packets. The place with the strongest signal is considered as a preliminary estimated device location.
(2) Position Tracking: Only SSMs do not allow the calculation of the exact estimated position of the device. In addition, it is necessary to consider the coordinates (X, Y, and Z axis) at which each SSM is made. For this purpose, we use a hybrid tracking system consisting of an inertial navigation system and a marker-less optical position tracking system. This combination of methods allows us to achieve a good tracking quality for slow and fast movements without the need for any environmental preparation [11, 13, 43].
(3) Based on the measurements collected during the first two phases, an estimated location can be formed for each device. To provide a user-friendly interpretation of the results and navigation, an ARV is used. Since augmented reality (AR) uses the same positional tracking system, the estimated device positions can be displayed in the correct locations. In addition, AR provides functionality to interact directly with the highlighted virtual objects, such as naming a device and viewing the values of the SSMs, network address, or device.

Performing the localization gives the users the option to identify all detected network devices in their physical environment, including identifying unobtrusive or hidden Wi-Fi-enabled devices.

With the use of previously acquired information that is persistently stored the user can iteratively do consecutive runs of the localization stage to gain added precision in the same physical location if deemed necessary.

[6]IEEE database like https://standards.ieee.org/products-programs/regauth/mac/

Stage 6: Device Isolation (for at-risk users). We directly included a final stage that allows a user to initiate actions to impair the Wi-Fi connectivity of a device, e.g., to prevent them from exchanging data over their Wi-Fi connection. Within our system's workflow, this capability is offered as an option to the users to initiate isolation of a selected device (see Figure 2(f)). Which devices the user sees as threats and wants to isolate have been supported in the previous stages, especially when the user is enabled to get an overview of the networked Wi-Fi-enabled devices in their actual physical environment (Stage 5), including recognizable device names and the associated privacy risk labeling (Stage 4). We base our steps to disconnect a device on the radio data extracted in previous steps (Stages 2 & 3) that is persistently stored. Most importantly, our isolation functions for isolation need to work without the need for physical interaction with the environment (see Section 3 for the motivating scenario).

For this purpose, a denial-of-service (DoS) attack using de-authentication frames is started. In this process, each targeted device is de-authenticated from the AP, hence, it has to perform the authentication process again. While the device is unauthenticated, it cannot exchange data over the Wi-Fi connection. The system repeats this DoS attack in short succession, thus the device remains in the unauthenticated state as long as the isolation process is active; this task can be performed in the background by the Companion while the user can continue to normally use the smartphone.

Note, that this relies on the use of de-authentication frames, hence, it can only be applied when these frames are not sufficiently secured in the targeted network. However, commercial off-the-shelf Wi-Fi APs deployed might not be enforcing this security yet. Hence, we integrated this vital stage into our system giving users leverage against those operating devices that pose a risk to users in situations where users have no possibility to disconnect devices physically (see Section 3).

4.2 System Design: Mobile App & Companion

When selecting the components for our system, we based our decision on the following requirements:

- **Accessibility:** The system shall be widespread and easily accessible. Hence, should be represented by commodity equipment that is either readily available or already possessed by many users.
- **Portability:** The system shall be portable in order to take it along, e.g., on travel.
- **Battery Powered:** The system shall be battery-powered to provide the required flexibility to move it freely through the room during the localization phase.

After analyzing the various options, we have selected two components a Mobile App & Companion (see Figure 4) as we believe this meets the above requirements, and in combination, they are capable of providing the required functionality to users. As a core component of the system, an iOS app is developed because smartphones and tablets are versatile and are already used by many people. Because the desired functionality requires low-level radio access, e.g., for packet capturing and injecting crafted packets, which can not be usually provided by non-rooted phones, the so-called Companion is introduced as the second component of our system. Implementation details are presented in Appendix A.

5 Constraints and Limitations

5.1 Constraints of Our Prototype

In order to limit the complexity of the prototype, we introduce several constraints regarding used devices and network settings which are listed in Table 1.

Even under the defined constraints, situations may arise where system performance is adversely affected. Such situations can either be created by the deliberate actions of an adversary, or they can

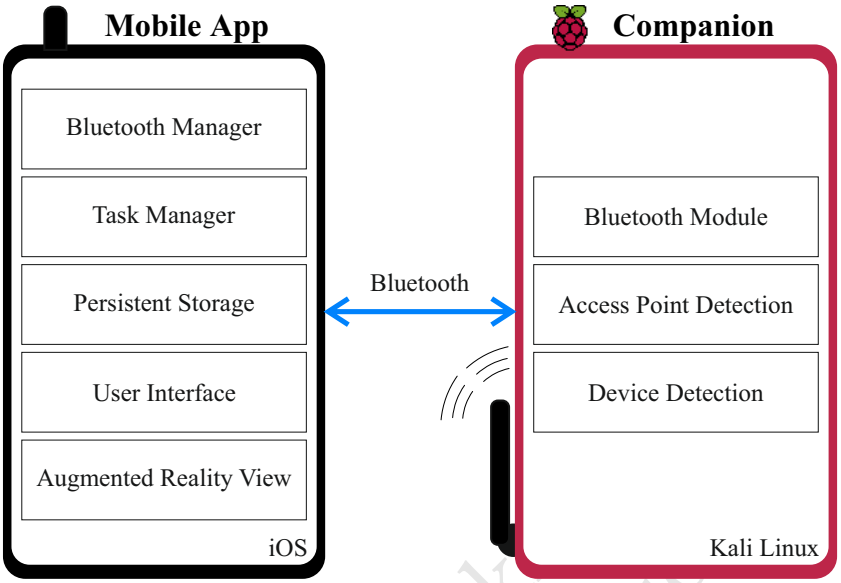


Fig. 4. Architecture of the proposed system.

form and occur naturally in network environments. The following describes the most prominent reasons that can adversely affect system performance, the first two are particularly with respect to device localization, while the last one is a general limitation on the effectiveness of isolation of recording devices with local storage.

5.2 Limitation: Highly Dense Networks

Large Number of Wi-Fi Networks. A large number of Wi-Fi networks may result in the circumstance that every network channel is occupied and has to be monitored by the system. The system has to switch between more channels at a fixed interval, so each channel is monitored at a lower frequency. This results in a lower number of measurement points for each target device and may cause sparsely transmitting devices to be missed altogether.

To address this problem, the user has the option to reduce the number of targeted channels by changing the selection of networks.

Large Number of Targeted Devices. Similar to the high density of available networks, having a large number of active devices can also create challenges for the system. As the overview for the user is crowded with entries, making it difficult to understand the environment in the ARV. Although the grouping feature can reduce the number of nodes, it still results in a long list of devices, making it problematic for the user to identify them.

Such high-density network or device environments can occur naturally in places where there are many people and their devices. It could also be deliberately created by an adversary to distract from the hidden device. Setting up multiple Wi-Fi networks and connecting devices to them allows the adversary to mask the actual hidden surveillance devices. However, this also increases the adversary's overhead.

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

Table 1. Constraints of our prototype.

Device Constrains		
Wi-Fi	802.11	The majority of common devices equipped with cameras and microphones use 802.11 WLAN.
HW Address randomization	No	Randomization of the address would prevent correct identification and traffic assignment which are based on the hardware address of a device. This only affects the usability if the address, e.g., the MAC, changes in the time between detection and localization. This might not be a problem as commercially available devices retain it for a larger time period, i.e., Apple devices can generate a random MAC address, but retain it once connected to the network [2]. Moreover, this constraint could be removed if an alternative technique for identifying the devices is used that does not use the hardware address of the devices [1].
Device Roaming	No	Devices do not switch to another AP; does not require additional hardware to listen on multiple channels simultaneously.
Location	Fixed	Assume malicious devices are stationary.
Network Constrains		
Bandwidth	2.4GHz	Absence of the need to monitor a lot of channels reduces the number of additional radio interfaces to run in parallel to uphold acceptable runtime during the scans.
Channel Width	20MHz	For simplicity the system is restricted to the 2.4GHz band with a channel width of 20MHz.
AP Roaming	No	Assume the environment can not have multiple APs with identical SSIDs and passwords, eliminating the need to store information on the AP.
PMFs	No	PMFs are disabled in order to enable the functionalities for active device detection and isolation based on de-authentication.

5.3 Limitation: Devices in Power Save Mode

Another situation that could negatively affect the device detection and localization process arises from devices using a power save mode. Some Wi-Fi devices apply different power management options to put the Wi-Fi interface to sleep for the purpose of saving battery power. During this doze state, the devices can not receive or transmit packets. Hence, they may be missed by the system.

This behavior of the devices can only be affected by the adversary using the provided settings on the device. To alter the power save behavior in such a way that makes it harder to detect and localize the device would require the adversary to change the device software. However, it is not assumed that the adversary has this capability.

5.4 Limitation: Devices with Local Storage

In case the malicious camera does not transmit a video signal but just saves it locally, the isolation function of our system would be insufficient because even after isolating the camera, it would continue to capture and save the video. However, isolation would try to suppress additional notifications or live-streaming behavior.

6 Evaluation

This section presents the results of the evaluation of the performance of the proposed system in a test environment.

6.1 Environment

6.1.1 Network Setup. Two 802.11 WLANs are provided for evaluation purposes which are created by two APs from different manufacturers. Both are configured to use WPA2 security with no PMFs enabled and the default beacon interval. Also, they operate with a bandwidth of 2.4GHz and a channel width of 20MHz. This is an artificial constraint opposed to reducing complexity as discussed in Section 5.1.

The first one is the AirPort Extreme AP. It creates the WLAN with the SSID “TestNetwork", which is set, to be included in the beacon frame broadcasts such that the prototype system can obtain the SSID. The AirPort Extreme AP provides this network only in the band on channel 11.

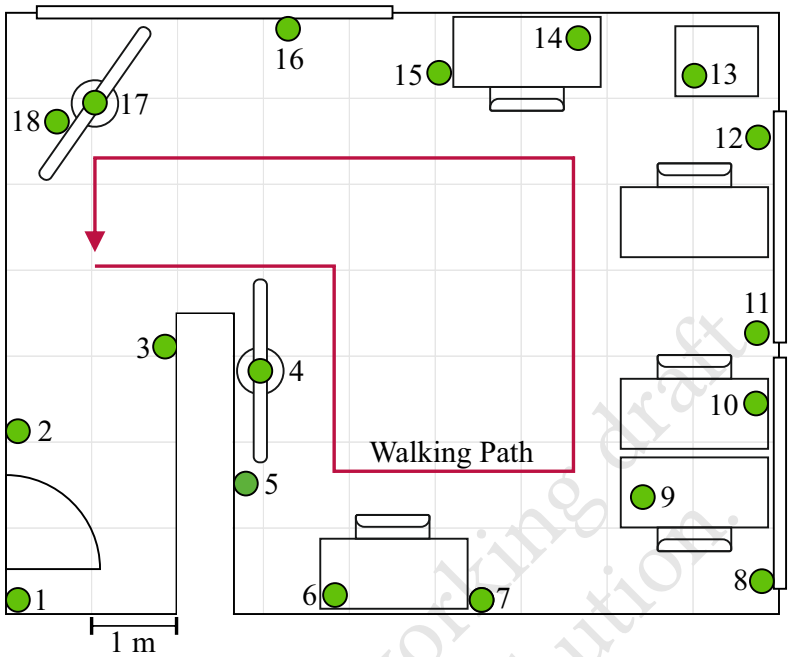
The second AP is the UniFi AC-M. It is configured to hide its SSID by not broadcasting it in the beacon frames and operates on network channel 1.

In addition to our test APs, there were two public networks and one private network available.

6.1.2 Connected Devices. 18 IoT devices with cameras and/or microphones, commonly purchased by users from online stores, were selected for evaluation and placed in the room according to the scheme shown in Figure 5.

6.2 Results

6.2.1 Network Interface Comparison. During the development of the prototype, we noticed that the system performance was better when the traffic monitoring was performed using the external USB Wi-Fi interface of the Companion rather than the internal one. To investigate this observation, we created a setup to compare RSS measurements from both Companion Wi-Fi interfaces details of which are presented in Appendix B. Based on the obtained results, we conclude, that to achieve accurate and consistent localization results, regardless of how the assembly is mounted or held by the user, the external Wi-Fi interface should be preferred.



(1) EZVIZ 1080P WLAN IP Camera
(2) TP-Link Tapo C100 WLAN IP Camera
(3) Mini Camera 1080P HD WLAN
(4) Samsung Smart TV
(5) Refoss Smart-Plug
(6) COOAU 2k WLAN Camera
(7) Little Elf 2k Babyphone with Camera
(8) ieGeek 2k WLAN Camera
(9) MacBook Pro
(10) ZUMIMALL 2k WLAN Camera
(11) TP-Link Tapo C320WS
(12) Xiaomi Mi Camera 2K
(13) DEATTI 2k WLAN Camera
(14) Smart Speaker HomePod Mini
(15) Aqara Hub
(16) Volume Knob
(17) Sony Bravia Smart TV
(18) Set-Top Box: Apple TV

Fig. 5. Test room plan.

6.2.2 AP Detection. System performance in terms of AP detection is tested using the standard scan time of 0.52s and the reduced duration of 0.11s per channel. For each duration, 10 AP detection runs are performed to observe if the available networks are reliably detected.

During the test, the two WLANs of the evaluation environment were detected by the prototype system in every run. This confirms the reliability of the AP detection and allows the system to build on the information gathered about the AP in subsequent steps without missing certain networks. The results also confirm that the reduced scan time of 0.11s per channel does not affect the AP detection performance in this particular evaluation setup thus the scan time per channel used in the prototype system could be reduced to 0.11s without missing any APs in practice.

6.2.3 Device Detection. In order to evaluate the device detection function, we count the number of successful discoveries for each device over 30 runs using active and passive approaches. During the test runs, the Companion and devices are stationary. The obtained results are presented in Figure 6.

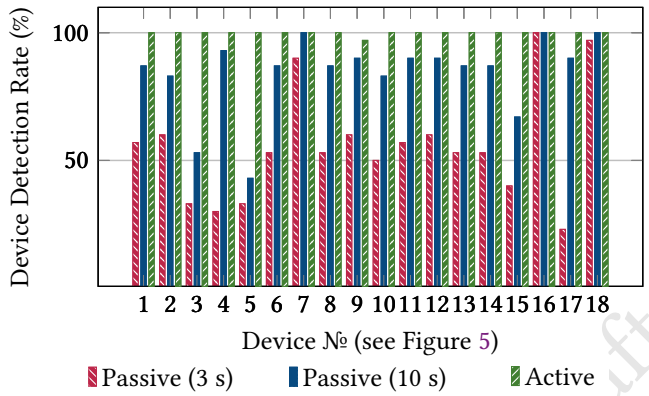


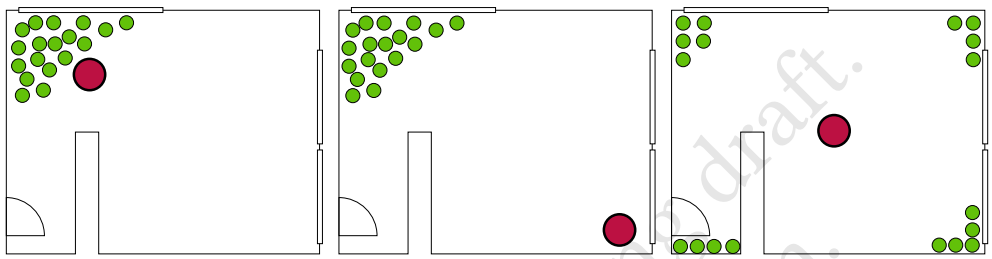
Fig. 6. Detection rates for each individual device using passive (3s, 10s) and active approaches over 30 test runs.

Comparison of the passive and active device detection approaches. First, we compare the performance of active and passive (10s) device detection. To examine whether there are differences between these two approaches, a Mann-Whitney U test is applied. The null hypothesis is that there is no difference between them and an $\alpha = 0.05$ is selected. The alternative hypothesis is that there is a difference between the two approaches. Because the resulting $2.93 * 10^{-11}$ of the test is smaller than the selected α, the null hypothesis can be rejected. Therefore the alternative hypothesis, that there is a difference between the active and passive (10s) detection approaches can be accepted. Also, according to the results, the active device detection performed 83% of the time better than the passive (10s) one. Combined with the result of the Mann-Whitney-U test, it can be concluded that the active device detection is superior to the passive (10s) one. With the reduced scan time of 3s per channel, the passive approach detected fewer devices in each run compared to the approach with 10s per channel scan. Therefore the active device detection outperforms both passive approaches under consideration.

Detection rate difference between devices. Differences between the devices can be noticed by looking at the detection rates of the passive approaches. For example, the Little Elf Babyphone (7), Volume Knob (16), and the Apple TV (18) are detected in almost all cases during both passive approaches. Hence, we can assume that these devices have frequent network traffic that allows the prototype system to capture their network address in the given time. On the other hand, for instance, the Refoss Smart-Plug (5) was only detected 43% of the time with a scan time of 10s and 33% of the time with a reduced scan time of 3s. This is related to its low network activity, which can lead to scan runs where there is no network traffic for this device, consequently, the device is not detected by the system. Reducing the scan time per channel has reduced the detection rates for the majority of devices, as a shorter interval is more likely to encounter an interval where the device has no network traffic. In contrast, during active detection, all devices in the evaluation environment were detected in all runs of the experiment successfully, with the only exception of the MacBook, which was missed three times in the 30 runs. As this approach uses device de-authentication to stimulate network traffic from all active devices, it does not depend on the natural traffic intervals of the devices. And indeed our experiments showed that the devices with low network traffic, such as the Refoss Smart-Plug (5) or Aqara Hub (15), are getting detected in every run.

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

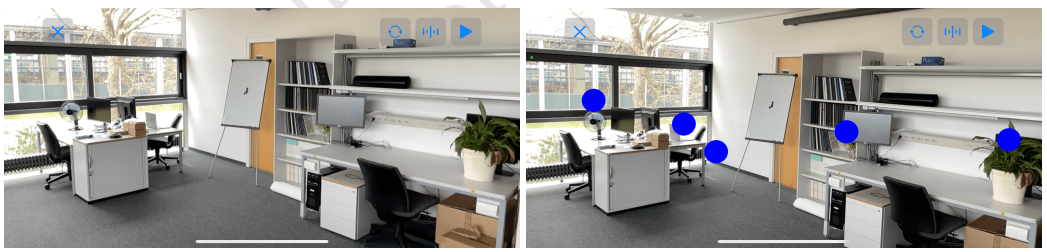
To examine the effect of different placement of devices in the environment on detection performance, we additionally conducted tests where the system and devices were located at different distances from each other (see Figure 7). Results show that if devices are within the access area of our system, detection performance is influenced neither by distance nor device density. However, localization demonstrated worse performance than in the main test environment due to the crowding of devices (see Section 5.2). This performance impact can be reduced by iterative runs of the localization, i.e., in the situation of Figure 7(c).



(a) (b) (c)

Fig. 7. Location of devices (●) in the room in relation to our system (●): (a) less than 1m, (b) more than 9m, and (c) approximately 5m from each other.

6.2.4 Device Localization. In this section, the device localization functionality of the prototype system is evaluated (see Figure 8). At first, the time synchronization between the Companion and the mobile application is examined (see Appendix C), which is fundamental for the localization task. After this, the accuracy of the device localization in the test environment is evaluated.



(a) (b)

Fig. 8. ARV: Test room (a) before and (b) after localization.

Localization Accuracy — Rough Localization. To investigate the accuracy of rough localization, we performed five slow walks along the path marked in Figure 5. During the walk, the system performs an estimation of distances to the devices. Each walk is one experiment, i.e., they are not used as consecutive iterations which would allow to increase the accuracy when demanded by the user.

Table 2 presents the measured distances between the user and estimated device locations. All measurements are taken in meters and rounded to the first decimal. The ideal value represents the goal. It is the real distance from the device to the nearest point on the walking path. The average error of the five runs to the ideal distance is given in the last column. This average error gives an

Table 2. Distances in meters between the user and estimated device locations during the rough localization.

№	Device	Ideal	Run Number					Avg. Error
			I	II	III	IV	V	
1	EZVIZ Camera	4.1	3.5	4	4	4.8	4.5	**0.4m**
2	TP-Link Tapo C100	2.2	2.2	2	2.7	1.5	2.1	**0.3m**
3	Mini Camera	1.4	2.4	1.6	1.6	1.4	2	**0.4m**
4	Samsung Smart TV	0.8	0.8	0.8	1.6	1.8	2.9	**0.8m**
5	Refoss Smart-Plug	0.9	1	1.1	1.3	2.9	1.1	**0.6m**
6	COOAU Camera	1.7	2	1.6	1.6	1.7	1.7	**0.1m**
7	Little Elf Babyphone	1.8	1.9	1.5	2.4	1.8	2	**0.2m**
8	ieGeek Camera	2.4	3	3	2.9	2.6	2	**0.5m**
9	MacBook Pro	0.5	0.5	0.5	0.5	0.7	0.6	**0.1m**
10	ZUMIMALL Camera	2.2	2.2	2.6	2.8	2	1.9	**0.3m**
11	TP-Link Tapo C320WS	2.1	2.6	2	2.1	1.7	2	**0.2m**
12	Xiaomi Mi Camera	2.1	1.7	2.2	2.1	1.3	2	**0.3m**
13	DEATTI Camera	1.7	2.3	2	2.1	1.5	1.7	**0.3m**
14	HomePod Mini	1.8	2	2.4	1.8	1.5	1.5	**0.3m**
15	Aqara Hub	0.9	1.3	1.6	2.4	2.7	1.9	**1.1m**
16	Volume Knob	1.5	1.3	1.1	2	1.5	1	**0.3m**
17	Sony Bravia Smart TV	0.6	1.9	0.6	0.6	2.4	0.6	**0.6m**
18	Set-Top Box: Apple TV	0.6	0.6	0.6	0.6	1	0.6	**0.1m**

indication of how close the localization results are for each device. For the majority of the devices, the average error is less than 0.3m. This shows that in the experiment, the short walk along the selected path was sufficient to get a rough location for each device which can then be further refined in the following precision localization step. In some cases, e.g., for the Aqara Hub (15), the estimated location was in three runs reported as more than 1m further than the goal. This deviation makes it difficult for the user to identify to which physical device the detected location belongs and makes it harder to find it. If this situation occurs, the stage can be rerun trying to localize the device. Moreover, the negative effect can be reduced by the possibility of identifying the manufacturer of the device from network information and using this to find the device in the physical space.

The results from the device detection step show that the Mini Camera (3), Aqara Hub (15), Refoss Smart-Plug (5), and TVs (4,17) had a low detection rate for the passive device detection with the shortened duration. This already indicated the infrequent network activity of these devices, which is also negatively affecting the localization of these devices.

6.2.5 Localization Accuracy — Precision Localization. In order to evaluate the accuracy of the precision localization, we measure distances between the actual and estimated device locations during five runs. Each test is performed during the time interval between 25s and 27s. In order to cover X-Y axes during the localization, we move the system in a winding motion from top to bottom. Also, it should be mentioned that the system assembly can not be moved closer than 0.2m to physical objects, as the antenna is facing forward, and a certain freedom of movement must also be preserved not to contact the objects. Again, each of the winding motions is taken as one experiment, i.e., they are not used as consecutive iterations which would allow to increase the accuracy when demanded by the user.

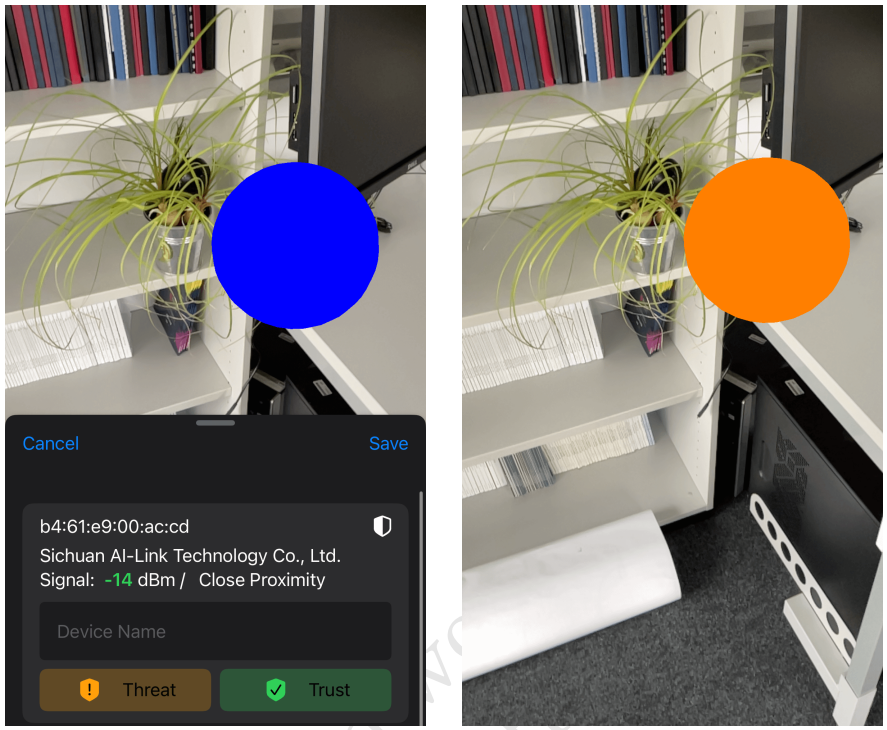
The results of the evaluation are presented in Table 3. First, we can see that for all devices, except the Refoss Smart-Plug (5), the system was able to estimate the position in each run within 0.5m or better. This precision should allow the user to identify the corresponding physical device for every

Table 3. Distance in meters between actual and estimated device locations during the precision localization.

№	Device	Run Number					Avg.
		I	II	III	IV	V	
1	EZVIZ Camera	0.4	0.5	0.5	0.5	0.5	**0.5m**
2	TP-Link Tapo C100	0.3	0.2	0.3	0.2	0.2	**0.2m**
3	Mini Camera	0.5	0.5	0.6	0.6	0.5	**0.5m**
4	Samsung Smart TV	0.3	0.2	0.2	0.2	0.9	**0.4m**
5	Refoss Smart-Plug	0.9	0.5	0.8	0.7	0.5	**0.7m**
6	COOAU Camera	0.3	0.3	0.3	0.2	0.4	**0.3m**
7	Little Elf Babyphone	0.2	0.3	0.2	0.2	0.2	**0.2m**
8	ieGeek Camera	0.4	0.3	0.3	0.4	0.4	**0.4m**
9	MacBook Pro	0.3	0.4	0.2	0.3	0.3	**0.3m**
10	ZUMIMALL Camera	0.6	0.4	0.4	0.6	0.5	**0.5m**
11	TP-Link Tapo C320WS	0.2	0.3	0.2	0.4	0.4	**0.3m**
12	Xiaomi Mi Camera	0.3	0.3	0.3	0.4	0.4	**0.3m**
13	DEATTI Camera	0.3	0.4	0.3	0.2	0.2	**0.3m**
14	HomePod Mini	0.3	0.2	0.3	0.3	0.3	**0.3m**
15	Aqara Hub	0.2	0.3	0.3	0.3	0.3	**0.3m**
16	Volume Knob	0.4	0.4	0.2	0.3	0.4	**0.3m**
17	Sony Bravia Smart TV	0.4	0.4	0.3	0.4	0.4	**0.4m**
18	Set-Top Box: Apple TV	0.3	0.4	0.3	0.3	0.5	**0.4m**

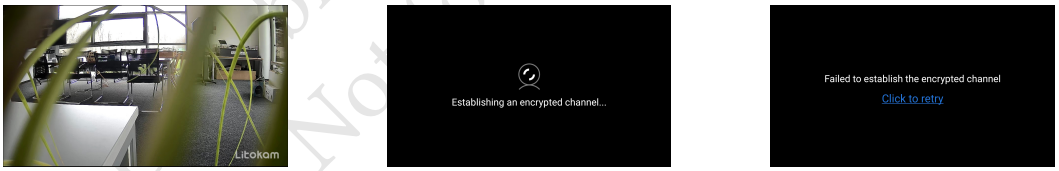
detected network device. This result might be insufficient in scenarios in which multiple devices are located close to each other. In such situations, the information about the device manufacturer can be useful to differentiate the devices, but also another localization scan can be performed to refine the estimated location further. For the Refoss Smart-Plug (5), the estimated location was, on average, 0.7m away from the actual position and even 0.9m in the first localization run. However, the absolute offset to the actual device location is essential for the user to match up the physical device with the detected network device. This is the case, especially in situations where other devices are in the immediate surroundings. For such scenarios, another localization run may be necessary to locate and identify the device correctly.

6.2.6 Device Isolation. To evaluate if the implemented device isolation of the prototype system is capable of hindering the selected device, which is marked as a threat (see Figure 9) from sending data, the network traffic during the running isolation is observed. For all devices, isolation was running for 3 min. If the isolation is successful (see Figure 10), the devices should not be sending data packets to the AP as they should not have the possibility to stay authenticated to the AP for long enough, as they get repeatedly de-authenticated by the Companion. Because the devices could be authenticated for a brief moment before the next de-authentication occurs, there exists the possibility for them to send data frames to the AP. The number of data frames sent by each isolated device to the AP is counted to look into how frequently this occurs. In the conducted experiment run no device was authenticated and associated with the AP at any point in time during the isolation. This supports the result that none of the devices in consideration was able to send a data frame. As the employed isolation technique is considered offensive their ethical impact is discussed in Section 7.2.



(a) (b)

Fig. 9. ARV: (a) Localized and (b) marked as threat device.



(a) (b) (c)

Fig. 10. View from the adversary's camera (a) during normal operation, (b)&(c) disconnected during isolation in Stage 6.

7 Discussion

This section is dedicated to discussing the performance comparison results of our approach with similar and potential ethical concerns.

7.1 Performance Comparison

Table 4 presents the comparison of our approach with similar related works (see Section 2).

First, we consider the presence of **persistent storage** for the capability to continue the process of interrupted localization (on purpose, accidentally, suddenly, etc.) at any time without losing

Table 4. Comparison of the proposed approach with existing works.

№	Approach	PS	Device								AI	Main Limitations
			Detection			Classification	Localization			Isolation		
			T	Ty	R	Type	T	D	AE			
1	SCamF [19]	✗	30s☆	P	97%	DT(98%)	≈	2D	0.18m	✗	✗	Only for live-streaming cameras, highly dependent on camera position.
2	LocCams [15]	✗	30s	P	95.12%	✗	30s	3D	92%*	✗	✓	Root authority is required, only for Wi-Fi and VBR-based cameras, only rough localization.
3	I&LComDev [17]	✗	-	P	-	DT(98.7%)	-	-	1.34m	✗	✓	Only for live-streaming cameras, only rough localization is available.
4	SnoopDog [39]	✗	40s	P	96%	DV	30s☆	2D	-	✗	✗	Only for Wi-Fi and VBR-based cameras.
5	MotionCompass	✗	-	-	-	DV	140s	3D	0.05m	✗	✓	Only for Wi-Fi-based and not wide-angle cameras, motion trigger is required.
6	AHCL [24]	✗	-	P	-	✗	-	1D✿	95.5%*	✗	✓	Only for live-streaming cameras, provides only room where a camera is located.
7	Lumos [37]	✗	40mir	P	100%	DT(95%)	30mir	3D	1.5m	✗	✓	Only for Wi-Fi-based devices, training data is required.
8	PanguVision [23]	✗	-	-	100%**	DT(98%)	-	3D	0.8m	✗	✗	-
9	Unveiling the Shadows	✓	10s	P/A	99.8%	DV	26s	3D	0.4m	100%	✗	Dense network environments and device power save mode influence performance negatively, PMF should be disabled.

(PS) Persistent Storage; (T) Time; (Ty) Detection type: **P**assive or **A**ctive; (R) Detection rate; (D) Dimensions; (AE) Average distance error; (AI) Using of AI tools;
(DT) Classification by **D**evice **T**ype, (DV) Classification by **D**evice **V**endor; (☆) The time specified is for a single device, for comparison multiply by 18 devices from our testbed;
(≈) No comparable time, as it is directly proportional to user path length; (*) Percentage of correctly defined device location areas (not at an exact physical location);
(**) No detailed experiments nor data was provided; (-) Characteristic is not mentioned; (✿) Provides only an area's name (1 - dimension) where a camera is located.

previous measurements. As can be seen, only our system provides the user with this opportunity. Note, we additionally use this information also for our integrated isolation capability.

Regarding device **detection**, we consider three characteristics: time, type, and rate. It can be observed that our active approach shows the best results with respect to time, within only 10s almost 100% detection rate can be achieved for all devices, while, e.g., SCamF requires 30s to detect one device [19]. Passive detection is also included in our solution but showed poor results, almost 15% worse than the active one. Examining the detection rates we can see that all approaches were able to achieve a detection success rate greater than 95%.

In our paper, we do not perform device **classification** by their types as some other approaches do [17, 19, 23, 37]. However, we have included a feature to identify the device manufacturer by MAC address which allows us to classify devices by their vendor.

Considering **localization** of the devices, we focus on the characteristics: time, dimensions, and average distance error. Based on our experiments, we conclude that the best results can already be achieved after about 26s of localization run, which is also the best result among the approaches under consideration. It should be noted that the measurements were taken for a standard room size of about 63sqm square meters in the scenario described. If the system is used in rooms much larger than this, such as a dance hall (e.g., 200sqm), good localization results most probably require more time. We also noticed that not all localization approaches take into account the 3 dimensions. For example, SCamF [19] and SnoopDog [39] focus only on 2 dimensions, while AHCL [24], we categorized as one dimensional because in this case the localization only indicates in which room the device is located. In most approaches, data on the average distance error (between the actual and estimated device position) are presented in meters, but in two solutions [15, 24], the authors use a percentage measure. In both cases, a kind of coarse localization is performed, where it is determined in which area the device is located without the use of exact coordinates, so the performance evaluation of the localization function is calculated as the percentage of system decisions.

From the comparison, we can see that **isolation** functionality is presented only in our system. We tested the isolation on each device in the test set for 3min and were able to achieve 100% results. During this time no packet was transmitted by an isolated device.

We also indicated the approaches that use or do not use **AI tools**. It can be seen that more than half approaches under consideration use AI tools for classification [17, 37] or/and localization [15, 17, 18, 24, 37]. In our approach, as well as in SCamF[19], SnoopDog [39], and PanguVision [23] no AI tools are used. AI usage imposes additional limitations, as it requires having data for training those AI models for a correct classification or localization.

Analyzing the **main limitations** of each approach, we noticed that all of them focus on Wi-Fi-based devices which is also valid for our approach. Some of them also target only VBR-based cameras [15, 39] or only not wide-angle cameras [18]. Also, the absence of live streaming negatively affects the detection and localization capability of existing works, and often does not allow them at all. We facilitate our isolation capabilities to allow for active detection (see Stage 3 in Section 4.1). Hence, the above limitations are not fully applicable to our approach, as our solution does not require special camera types or constant streaming. Additionally, some approaches that rely on AI require proper training data for the models, which can greatly limit their applicability [15, 17, 18, 24, 37].

7.2 Ethical Concerns

The use of offensive techniques to actively detect and isolate devices raises ethical questions.

Accidental Misuse: In our system, a user can mark devices as threats without restriction, hence, a device that is not malicious can be marked accidentally as a threat and isolated. It should be noted that the same device may or may not be a threat from the perspective of different users and situations. Given the scenario (see Section 3), an adversary may use devices that are not originally designed for espionage, and therefore from a technical point of view their transmitted signals cannot be distinguished from non-malicious ones. In order for the system to be able to determine without user input that a device is malicious, it would be necessary to observe the behavior of different devices over a long period of time and analyze the content of the information they transmit, as well as to observe the user, in order to identify possible risks and corresponding threats automatically. However, this topic is beyond the scope of our study. Therefore, in our system, we leave it to the user to decide which device is a threat to the user privacy in that single moment and which is not.

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

Adversarial Abuse: A potential adversary could abuse our system by scanning the space and isolating a device of their choice. However, all used offensive techniques are described already in the literature and other implementations are readily available for a dedicated attacker. Hence, by abusing our proposed system, no new attacks are possible.

8 Conclusion and Future Work

In this paper, we present a system that combines functionality to discover available Wi-Fi Access points (AP), to discover devices connected to an AP, to localize them, and – as a novel stage – also to finally isolate a detected device. The estimated device locations are presented to the user using augmented reality views (ARV). The persistent storage of network details and the use of unencrypted header information from captured packets, combined with position tracking and a visualization facilitating ARV makes the system usable without any knowledge of the environment. This aids users in identifying potentially harmful devices quickly and with high accuracy regarding their physical location in unfamiliar environments. Moreover, our system finally allows users to carry out denial-of-service attacks based on prior collected information and thus the ability to disconnect privacy-invasive IoT devices – with 100% reliability.

A prototype consisting of two components has been developed: The first component is an iOS mobile application that provides the user interface, main logic, position tracking, augmented reality, and persistent storage. Since the mobile application has limited low-level access to the network interface, the Wi-Fi network functionality is handled by the second component, called the Companion. The Companion is implemented on a Raspberry Pi 4 with an external Wi-Fi interface and performs the required packet capture and injection tasks for the mobile application. A two-component approach was chosen to access low-level radio functionality, even if it is not available on a user's smartphone and empowers the user to carry out the low-level DoS-attacks necessary for isolation without involvement and thus without blocking the user's smartphone.

For performance evaluation, we used a test environment containing 18 different network devices spanning a wide range of commercially available devices as well as two different APs. During the AP detection evaluation, our system detected both APs 100% of the time. For device detection, the active approach showed a 100% detection rate in the majority of cases for all 30 experimental runs, outperforming the also considered passive approaches. Localization accuracy was measured for two scenarios: rough and precise localization. On average, rough localization gives solid results (average error distance between user and device $\leq$ 0.3m) for 11 of 18 devices. For precise localization, the system was able to locate every device, except one Smart-Plug, to a range of 0.4m.
As interesting directions for future work, we see the following:

- Consider opportunities to expand the prototype and provide broader applicability even in more secure unfamiliar environments by expanding the technical functionality, e.g., to consider devices with WPA3, Bluetooth, ZigBee, etc.
- Integrate existing approaches or new solutions to enhance the performance in certain stages, e.g., add long-term scanning to unveil devices with low communication frequency; or improvements within a stage, e.g., in augmented reality.
- Provide the user study to examine the "user-friendliness" property of these approaches including the user perception of the visualization of network security. Finally, our generalized six-stage workflow could be used as a basis for classifying the focus of future approaches and indicating future open research questions in relation to existing research.

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

Acknowledgments

This work has been partially funded by the Bavarian Ministry of Science within the framework of the research cluster "ForDaySec: Security in Everyday Digitalization", as well as, by the German Federal Ministry of Education and Research – Bundesministerium für Bildung und Forschung (BMBF), as part of the Project "6G-RIC: The 6G Research and Innovation Cluster" (project number 825026).

This paper extends and builds upon the Master's thesis of Daniel Schaubschläger titled "Detect and Locate Hidden Wi-Fi Devices in Unknown Environments".

References

[1] Mnassar Alyami, Mohammed Alkhowaiter, Mansour Al Ghanim, Cliff Zou, and Yan Solihin. 2022. MAC-Layer Traffic Shaping Defense Against WiFi Device Fingerprinting Attacks. In *2022 IEEE Symposium on Computers and Communications (ISCC)*. 1–7. https://doi.org/10.1109/ISCC55528.2022.9913056

[2] Support Apple. 2023. Use private Wi-Fi addresses on iPhone, iPad, iPod touch and Apple Watch. https://support.apple.com/guide/security/wi-fi-privacy-secb9cb3140c/web.

[3] Apple Developer Documentation. 2019. Framework CoreBluetooth. https://developer.apple.com/documentation/corebluetooth.

[4] Apple Developer Documentation. 2019. Framework CoreData. https://developer.apple.com/documentation/coredata.

[5] Apple Developer Documentation. 2019. Framework SwiftUI. https://developer.apple.com/documentation/swiftui/.

[6] Apple Developer Documentation. 2019. Framework UIKit. https://developer.apple.com/documentation/uikit.

[7] Arstechnica. 2019. Airbnb guest found hidden surveillance camera by scanning Wi-Fi Network. https://arstechnica.com/information-technology/2019/04/airbnb-guest-found-hidden-surveillance-camera-by-scanning-wi-fi-network/.

[8] John Bellardo and Stefan Savage. 2003. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In *12th USENIX Security Symposium (USENIX Security 03)*.

[9] Bluez.org. 2019. BlueZ. http://www.bluez.org.

[10] David Buil-Gil, Steven Kemp, Stefanie Kuenzel, Lynne Coventry, Sameh Zakhary, Daniel Tilley, and James Nicholson. 2023. The digital harms of smart home devices: A systematic literature review. *Computers in Human Behavior* 145 (2023), 107770. https://www.sciencedirect.com/science/article/pii/S0747563223001218

[11] Julie Carmigniani and Borko Furht. 2011. Augmented reality: an overview. *Handbook of augmented reality* (2011), 3–46.

[12] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. Dewicam: Detecting hidden wireless cameras via smartphones. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 1–13.

[13] Kar Wee Chia, Adrian David Cheok, and Simon JD Prince. 2002. Online 6 dof augmented reality registration from natural features. In *Proceedings. International Symposium on Mixed and Augmented Reality*. IEEE, 305–313.

[14] Gokila Dorai, Eleason A. Williams, Hongmei Chi, and Richard A. Alo. 2020. "Is your Smart Home a Secure Home?" - Analysis of Smart Home Breaches and an Approach for Vulnerability Analysis and Device Isolation. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. 1–6. https://doi.org/10.1109/WF-IoT48130.2020.9221420

[15] Yangyang Gu, Jing Chen, Cong Wu, Kun He, Ziming Zhao, and Ruiying Du. 2024. LocCams: An Efficient and Robust Approach for Detecting and Localizing Hidden Wireless Cameras via Commodity Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 4, Article 160 (jan 2024), 24 pages. https://doi.org/10.1145/3631432

[16] Amy Gunia. 2019. Hundreds of motel guests were secretly filmed and live-streamed online. https://edition.cnn.com/2019/03/20/asia/south-korea-hotel-spy-cam-intl/index.html.

[17] Xing Guo, Jie Quan, Jiahui Hou, Hao Zhou, Xin He, and Tao He. 2022. Accurately Identify and Localize Commodity Devices from Encrypted Smart Home Traffic. In *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*. 663–670. https://doi.org/10.1109/MSN57253.2022.00109

[18] Yan He, Qiuye He, Song Fang, and Yao Liu. 2021. MotionCompass: pinpointing wireless camera via motion-activated traffic. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin) *(MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 215–227. https://doi.org/10.1145/3458864.3467683

[19] Jeongyoon Heo, Sangwon Gil, Youngman Jung, Jinmok Kim, Donguk Kim, Woojin Park, Yongdae Kim, Kang G. Shin, and Choong-Hoon Lee. 2022. Are There Wireless Hidden Cameras Spying on Me?. In *Proceedings of the 38th Annual Computer Security Applications Conference* (Austin, TX, USA) *(ACSAC '22)*. Association for Computing Machinery, New York, NY, USA, 714–726. https://doi.org/10.1145/3564625.3564632

[20] Kolya Hnatyuk. 2023. Internet of Things (IoT) Statistics: 2022/2023. https://marketsplash.com/internet-of-things-statistics.

[21] IEEE. 2023. MA-L. https://standards.ieee.org/products-programs/regauth/oui/.
[22] Sagar Joshi. 2023. 70 IoT Statistics to Unveil the Past, Present, and Future of IoT. https://learn.g2.com/{IoT}-statistics. accessed: 15.06.2023.
[23] Xiangyu Ju, Biao Han, Yitang Chen, and Jinrong Li. 2023. Demo: A Prototype for Detecting and Localizing Hidden Devices in Unfamiliar Environments. In *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing* (Washington, DC, USA) *(MobiHoc '23)*. Association for Computing Machinery, New York, NY, USA, 314–315. https://doi.org/10.1145/3565287.3617905
[24] Jihyeon Lee, Sangwon Seo, Taehun Yang, and Soochang Park. 2022. AI-aided Hidden Camera Detection and Localization based on Raw IoT Network Traffic. In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE, 315–318.
[25] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting Wireless Spy Cameras Via Stimulating and Probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services* (Munich, Germany) *(MobiSys '18)*. Association for Computing Machinery, New York, NY, USA, 243–255. https://doi.org/10.1145/3210240.3210332
[26] Yang Liu, Shi Guo, and Yanzhang Chen. 2022. Anti-candid Camera Detection System Based on Deep Learning. In *2022 China Automation Congress (CAC)*. 2601–2605. https://doi.org/10.1109/CAC57257.2022.10054963
[27] Ziwei Liu, Feng Lin, Chao Wang, Yijie Shen, Zhongjie Ba, Li Lu, Wenyao Xu, and Kui Ren. 2023. CamRadar: Hidden Camera Detection Leveraging Amplitude-modulated Sensor Images Embedded in Electromagnetic Emanations. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 173 (2023), 25 pages. https://doi.org/10.1145/3569505
[28] Karim Lounis, Steven H. H. Ding, and Mohammad Zulkernine. 2022. Cut It: Deauthentication Attacks on Protected Management Frames in WPA2 and WPA3. In *Foundations and Practice of Security*, Esma Aïmeur, Maryline Laurent, Reda Yaich, Benoît Dupont, and Joaquin Garcia-Alfaro (Eds.). Springer International Publishing, Cham, 235–252.
[29] Albert Selebea Lutakamale, Herman C. Myburgh, and Allan de Freitas. 2024. RSSI-based fingerprint localization in LoRaWAN networks using CNNs with squeeze and excitation blocks. *Ad Hoc Networks* 159 (2024), 103486. https://doi.org/10.1016/j.adhoc.2024.103486
[30] MAC Address Lookup API. 2023. https://www.macvendorlookup.com/api.
[31] David L Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on communications* 39, 10 (1991), 1482–1493.
[32] Iza S. Mohamad Hashim, Akram Al-Hourani, and Branko Ristic. 2022. Satellite Localization of IoT Devices Using Signal Strength and Doppler Measurements. *IEEE Wireless Communications Letters* 11, 9 (2022), 1910–1914. https://doi.org/10.1109/LWC.2022.3187065
[33] Plume. 2022. Plume IQ 1H 2022 Smart Home Market Report.
[34] Muhammad Salman, Nguyen Dao, Uichin Lee, and Youngtae Noh. 2022. CSI:DeSpy: Enabling Effortless Spy Camera Detection via Passive Sensing of User Activities and Bitrate Variations. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 2, Article 72 (jul 2022), 27 pages. https://doi.org/10.1145/3534593
[35] Sriram Sami, Sean Rui Xiang Tan, Bangjie Sun, and Jun Han. 2021. LAPD: Hidden Spy Camera Detection using Smartphone Time-of-Flight Sensors. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems* (Coimbra, Portugal) *(SenSys '21)*. Association for Computing Machinery, New York, NY, USA, 288–301. https://doi.org/10.1145/3485730.3485941
[36] Scapy.net. 2019. Scapy. https://scapy.net.
[37] Rahul Anand Sharma, Elahe Soltanaghaei, and Anthony et al. Rowe. 2022. Lumos: Identifying and Localizing Diverse Hidden IoTDevices in an Unfamiliar Environment. In *31st USENIX Security Symposium*. 1095–1112.
[38] Zhixin Shi, Hao Wu, Jing Zhang, Meng Zhang, and Wei-Jung Huang. 2023. FindSpy: A Wireless Camera Detection System Based on Pre-Trained Transformers. *2023 IEEE Symposium on Computers and Communications (ISCC)* (2023), 816–822. https://api.semanticscholar.org/CorpusID:261327985
[39] Akash Deep Singh, Luis Garcia, Joseph Noor, and Mani Srivastava. 2021. I Always Feel Like Somebody's Sensing Me! A Framework to Detect, Identify, and Localize Clandestine Wireless Sensors. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 1829–1846. https://www.usenix.org/conference/usenixsecurity21/presentation/singh
[40] Yang-Hsi Su, Chouchang Jack Yang, Euiseok Hwang, and Alanson P. Sample. 2023. Single Packet, Single Channel, Switched Antenna Array for RF Localization. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 2, Article 76 (jun 2023), 25 pages. https://doi.org/10.1145/3596263
[41] Piyush Tiwary, Ankur Pandey, and Sudhir Kumar. 2021. Differential d-Vectors for RSS based Localization in Dynamic IoT Networks. In *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. 82–85. https://doi.org/10.1109/COMSNETS51098.2021.9352896
[42] Geert Van der Auwera, Prasanth T. David, and Martin Reisslein. 2008. Traffic characteristics of H.264/AVC variable bit rate video. *IEEE Communications Magazine* 46, 11 (2008), 164–174. https://doi.org/10.1109/MCOM.2008.4689260

Preprint — to appear in Proceedings of 23rd Workshop on Privacy in the Electronic Society (WPES'24), Salt Lake City (US), October 14, 2024 — https://doi.org/10.1145/3689943.369504

[43] DWF Van Krevelen and Ronald Poelman. 2010. A survey of augmented reality technologies, applications and limitations. *International journal of virtual reality* 9, 2 (2010), 1–20.

[44] Noel Warford, Tara Matthews, Kaitlyn Yang, Omer Akgul, Sunny Consolvo, Patrick Gage Kelley, Nathan Malkin, Michelle L Mazurek, Manya Sleeper, and Kurt Thomas. 2022. SoK: A framework for unifying at-risk user research. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2344–2360.

[45] Zhiyuan Yu, Zhuohang Li, Yuanhaur Chang, Skylar Fong, Jian Liu, and Ning Zhang. 2022. HeatDeCam: Detecting Hidden Spy Cameras via Thermal Emissions. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 3107–3120. https://doi.org/10.1145/3548606.3560669

[46] Luping Zheng. 2024. An improved localization approach based on Sybil attack for WSN. *Physical Communication* 63 (2024), 102283. https://doi.org/10.1016/j.phycom.2024.102283

A Implementation Details

A.1 Mobile Application Implementation

The prototype was realized as an iOS application containing the main logic of the system as well as the complete UI. All components are implemented in Swift, using only native frameworks from the iOS SDK.

The Bluetooth Manager is implemented using the CoreBluetooth [3] framework and acts as the connection point between the mobile application and the Companion. While the Task Manager creates, schedules, and launches the tasks executed by the Companion, handles the received results, and makes them available to the other components of the application. Persistent storage is implemented using the CoreData [4] framework and provides persistent storage of collected information about the network environment, including data for the MAC vendor lookup functionality. Although there are many different web services that perform vendor lookup [30], using one of them would require the application to be connected to the Internet. To avoid these dependencies, the application has a local copy of the public IEEE MA-L list as a CSV file [21]. The UI and ARV are implemented using SwiftUI [5] and UIKit [6] respectively, following the Model-View-ViewModel (MVVM) pattern.

In order to provide device detection and localization, the system needs to capture all Wi-Fi packets that are transmitted in the network thus it is essential to have two functionalities: monitor mode and packet injection. The use of the mobile application for the system alone is not sufficient and is associated with the following challenges:

- The network interface cannot always be set to monitor mode because most mobile devices allow the applications only limited access to the interface. Therefore, only packets addressed to the mobile device or from multicasts are passed through to the application.
- Packet injection is required to provide the active option for device detection, and also to improve localization for infrequently transmitting devices and device isolation. Like monitor mode, packet injection is blocked on most mobile devices.

A.2 Companion Implementation

The Companion is added to the system to support the mobile application in performing the network-related tasks mentioned previously.

For the prototype, the Companion is implemented on a Raspberry Pi 4 Model B which has a small form factor, good availability, and versatility making it accessible and portable as required. Essential for the Companion is the ability to access internal and external Wi-Fi interfaces which can be set to promiscuous mode. This means that it can monitor the wireless network traffic and inject custom packets into the networks. Furthermore, the Raspberry Pi can be powered by a typical USB power bank, making it battery-powered, as required, allowing the user to walk around the room with the system. The mobile application and the Companion need a way to communicate to exchange task descriptions and collected information. For this communication, the onboard

Bluetooth modules of the Raspberry Pi and the mobile device are used. They are already integrated and no additional hardware is required. The user just powers the Companion on and all further interactions are done via the mobile application.

The Companion runs Kali Linux, which was chosen because it already includes a firmware version for the on-board Wi-Fi interface, allowing it to be set to monitor mode. As an additional network interface, an external Alfa USB Wi-Fi interface is connected to the Raspberry Pi, which supports monitor mode and frame injection. The Bluetooth module is implemented using the BlueZ Bluetooth [9] framework and acts as the connection point of the Companion with the mobile application and the user. It provides the interface to the mobile application to use the available functions of the Companion. The AP Detection functionality is implemented using the Scapy [36] framework. It captures 0.52s of network traffic on each channel to obtain a beacon frame from each AP. The ability to provide the user with information about the listed devices is also provided by Scapy through the device detection feature. In a passive approach, the process is configured to run for 10s per channel before completing the task and notifying the mobile app of the available result data. In an active approach, both Wi-Fi interfaces of the Companion are used, the internal one for channel sniffing and the external one for packet injection.

B Analysing potential interference of the combined system components

To investigate this observation, we created a setup to compare RSS measurements from both Companion Wi-Fi interfaces. For this purpose, the Companion is placed in a fixed position at a distance of 1m from the EZVIZ camera, which is in line of sight. The Companion begins sniffing the Wi-Fi traffic on both interfaces simultaneously for 20s, filtering for the camera's packets. As both interfaces remain stationary, the RSS measurements are expected to show consistent values. In practice, however, small variations are possible. In the following, we compare the results when using the Companion in close proximity, i.e., attached to the mobile device (see Figure 11(b)) and when using it standalone (see Figure 11(a)).

Note, in both scenarios we are doing the measurements for both radio interfaces of the Companion (the Raspberry Pi 4 internal & the external USB Wi-Fi device). Each is run individually, resulting in four datasets.

Figure 12(a) presents the results of the RSS measurements for the first scenario setup (see Figure 11(a)). We can see, that both interfaces perform similarly, but the RSS values are in different ranges showing that these are only relative values and therefore not directly comparable. For the localization, the dispersion of the RSS measurements is the key factor which is similar for both interfaces in the measurements taken. Looking at the outliers, the internal interface had more out-of-range RSS values than the external interface. However, the 0dBm RSS outliers do not affect the localization, as they are directly filtered out by the Companion. For the rest of the outliers, the Companion also has a good chance of filtering them out by taking the mean of the last recorded measurements for the position estimation. Hence, we can conclude that for the first scenario setup, there is no difference, in which interface is used for the measurements.

Examining the results obtained for the second scenario setup (see Figure 11(b)) presented in Figure 12(b), we can see that the measurements from the external interface have a low dispersion, comparable to the previous setup. However, the measurements from the internal interface have a higher dispersion than in the first scenario. In addition, the number of outliers increases. These outliers can also be filtered out by the Companion, but with a larger number of outliers, the implemented methods may not be able to remove all of them.

These fluctuations in RSS values can lead to inaccurate localization, as an estimated location may only be selected due to a change in RSS value caused by such a fluctuation, and not due to a device movement as assumed by the system. To achieve accurate and consistent localization results,

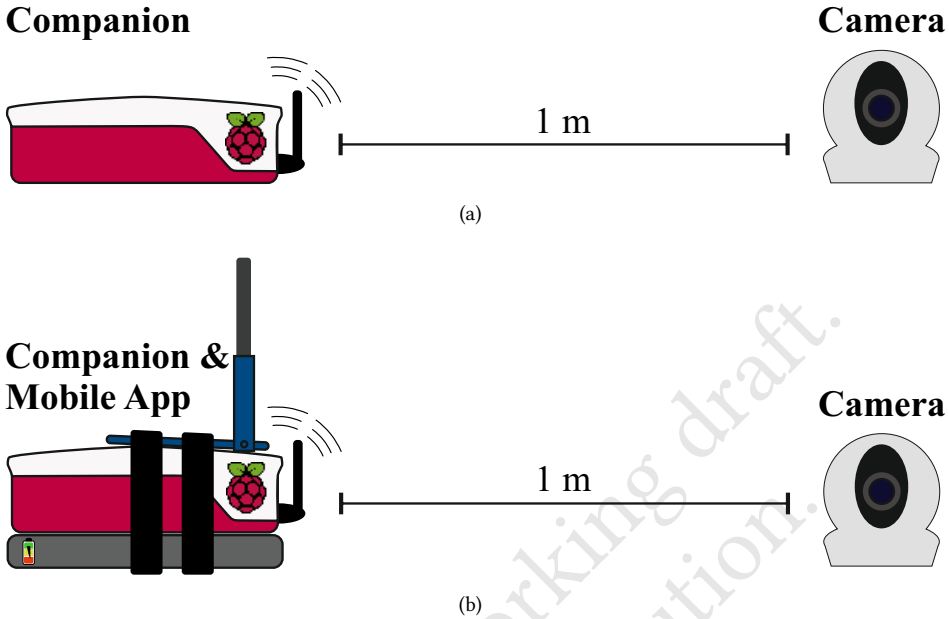


Fig. 11. Device layout scheme for comparing the performance of network interfaces using (a) the Companion only or (b) the Companion assembled to the mobile phone.

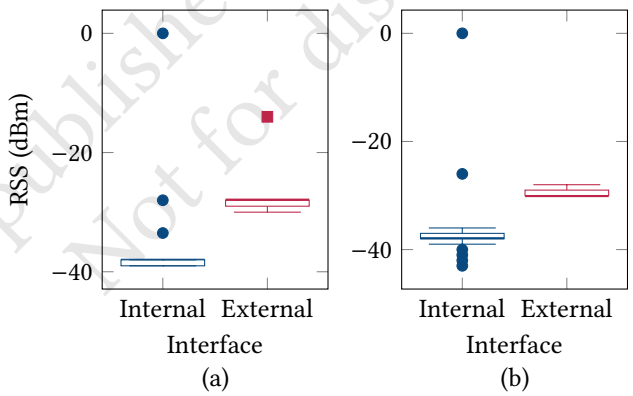


Fig. 12. RSS measurements for (a) the first and (b) the second scenarios pictured in Figure 11.

regardless of how the assembly is mounted or held by the user, the external Wi-Fi interface should be preferred.

C Evaluation of Time Synchronization during Device Localization

Localization uses timestamps to map the RSS data captured by the Companion to the positional tracking information provided by the mobile application. Since it cannot rely on synchronized time

using network time synchronization protocol (NTP), as the Companion may not have Internet access of its own, the implemented systems performs a time synchronization process before each localization [31]. To accurately determine the time difference between the two components is essential for correct mapping results and therefore a failure to do so would negatively affect the accuracy of the entire localization process. To reduce it to a minimum it the mobile application connects to the Companion via Bluetooth and executes the synchronization task. For further details, we would refer the interested readers to our code, which we can provide upon request.

To evaluate our time synchronization routine we carried out the following test: First, both components are connected to the Internet and have NTP enabled. This provides a baseline for time synchronization as the clocks on both systems should now be within 30ms of the time server [31]. Second, the mobile application is connected to the Companion via Bluetooth and performs the time synchronization task, as it would normally do before starting the localization. This task returns the measured time difference between the two systems. To assess the consistency of the results provided, the task was repeatedly executed 100 times in succession for three individual runs.

The test returned the following results: The distribution of the collected time difference measurements is shown in Figure 13 for each run. As can be seen from the box plots, the time differences in all three runs have a comparable dispersion. For all three runs, it can be said that the dispersion is narrow, in the range of about 2ms from the median. This indicates that the time difference measurement implemented in the prototype provided a consistent result in these evaluation runs. The median values from the three runs show that the first and second runs are close to 35ms while the third run has a median time difference of 45ms. Considering the consistent measurements during a run, these differences between the runs can be attributed to the differences in the NTP time synchronization of each run. Considering the NTP synchronization as the ground truth, the offset of the implemented method lies in the first two runs at around 35ms and in the third run at around 45ms. Because in the implemented positional tracking history of the system, a sample is taken only every 100ms, the deviation introduced by the implemented time synchronization process can be accepted.

Our results suggest that in the mapping process, the deviation created by the time synchronization lies around one tracked position sample before or after. Given a realistic walking speed of the user during the localization of at most 0.5m/s this relates to a deviation of 5cm, which we found sufficient for the general task of localization.

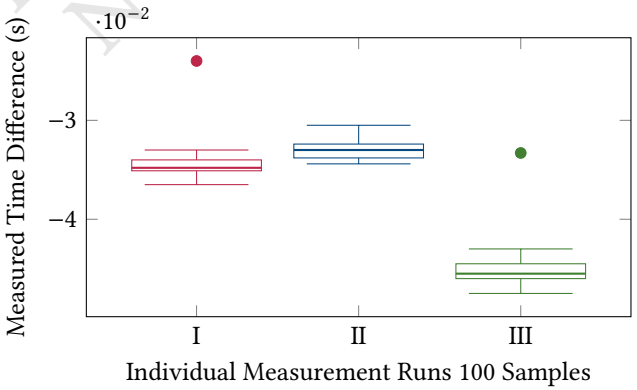


Fig. 13. Time synchronization measurements results.