# DeWiCam: Detecting Hidden Wireless Cameras via Smartphones

Yushi Cheng[12], Xiaoyu Ji[12†], Tianyang Lu[1], Wenyuan Xu[1†]
[1] Ubiquitous System Security Lab (USSLAB), Zhejiang University
[2]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
Emails: {yushicheng, xji, 5pipi, wyxu}@zju.edu.cn

## ABSTRACT

Wireless cameras are widely deployed in surveillance systems for security guarding. However, the privacy concerns associated with unauthorized videotaping, are drawing an increasing attention recently. Existing detection methods for unauthorized wireless cameras are either limited by their detection accuracy or requiring dedicated devices. In this paper, we propose DeWiCam, a lightweight and effective detection mechanism using smartphones. The basic idea of DeWiCam is to utilize the intrinsic traffic patterns of flows from wireless cameras. Compared with traditional traffic pattern analysis, DeWiCam is more challenging because it cannot access the encrypted information in the data packets. Yet, DeWiCam overcomes the difficulty and can detect nearby wireless cameras reliably. To further identify whether a camera is in an interested room, we propose a human-assisted identification model. We implement DeWiCam on the Android platform and evaluate it with extensive experiments on 20 cameras. The evaluation results show that DeWiCam can detect cameras with an accuracy of 99% within 2.7 s.

## CCS CONCEPTS

• **Security and privacy → Privacy protections**;

## KEYWORDS

Hidden Camera Detection; Traffic Analysis; Personal Privacy

**ACM Reference Format:**
Yushi Cheng, Xiaoyu Ji, Tianyang Lu, Wenyuan Xu. 2018. DeWiCam: Detecting Hidden Wireless Cameras via Smartphones. In *ASIA CCS '18: 2018 ACM Asia Conference on Computer and Communications Security, June 4–8, 2018, Incheon, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3196494.3196509

## 1 INTRODUCTION

Surveillance cameras are widely deployed in various locations to provide protection services ranging from public safety, law enforcement, to home security. Among all types, Wi-Fi wireless cameras, i.e., Internet protocol (IP) cameras equipped with Wi-Fi modules, tend to be the first choice because no wiring is required and can provide real-time services. Not surprisingly, 8 out of top 10 surveillance cameras in Amazon are wireless cameras. Look into the future, the number of wireless cameras will continue to grow, as Technavio's analysts forecast that the global wireless video surveillance market will increase at a rate of more than 21% over 2017-2021 [29].

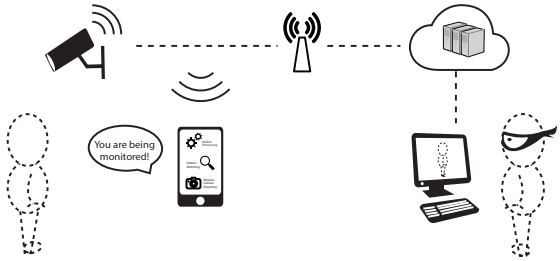



**Figure 1: Hidden wireless cameras cause privacy invasion by remote monitoring. DeWiCam detects wireless cameras over wireless channels using ubiquitous smartphones.**

However, along with this growing trend are the privacy concerns [17], especially in rental houses. Already, hidden cameras have been found inside Airbnb houses to record embarrassed moments in bed [23], videotape naked guests [7, 16] and capture guests' personal activities [13, 18]. An anonymous Airbnb host even confessed that "*I swap hidden camera sex videos with other Airbnb hosts.*" [9, 23]. The victims often feel helpless against spy cameras. They are typically wireless ones, because they can be placed anywhere in a room, e.g., in a bathroom light fixture [16], inside a smoke detector [18], even in a fan [24], or a mesh basket [13]. Although Airbnb requires all members of its community to respect guests' privacy and prohibits any surveillance devices in private spots, such news keeps being released.

Hidden camera detection is of great significance. However, a reliable solution is yet to be designed. We discover a few Apps [10, 11, 21, 25, 36] in Apple Store and Google Play that claim to be able to detect cameras. However, after evaluation, we find that they fail to work reliably. Those Apps claim to utilize one of the following methods: light-reflecting-based method that detects the tell-tale signs of tiny reflections from camera lens, or electromagnetic-interference (EMI)-based method that identifies the EMI leakage from a working camera. Both methods assume users are aware of the approximate location of cameras, which is not always the case. In addition, we find a dedicated device [28] (called wireless camera RF detector) that claims to detect hidden cameras by examining the RF signals at 2.4 GHz or 5 GHz, but it's unreliable as well.

†Corresponding faculty authors.


*ASIA CCS '18, June 4–8, 2018, Incheon, Republic of Korea*

ACM ISBN 978-1-4503-5576-6/18/06…$15.00
https://doi.org/10.1145/3196494.3196509

In this paper, we propose DeWiCam, a lightweight and effective mechanism for detecting hidden wireless cameras using smartphones. As shown in Fig. 1, we envision that when a user enters an untrusted room (e.g., an Airbnb rental), she/ he runs DeWiCam on her/ his smartphone briefly to detect whether a wireless camera is inside the room. Essentially, DeWiCam will automatically analyze wireless traffic to recognize the distinct traffic patterns of wireless cameras, if any, and inform the user of the existence as well as the exact number of wireless cameras. The underlying principle of DeWiCam is that wireless cameras in operation continuously generate camera traffic flows that consist of video and audio streams. The traffic pattern of a camera flow is likely to be different from that of other network application flows. By exploring the distinction in traffic patterns, DeWiCam shall be able to identify camera flows, and thus the cameras.

Identifying cameras via traffic patterns is promising yet challenging. Since a user has a limited control over the environment and the networks (a rental host can even dedicate a second network that is inaccessible to guest users to a wireless camera), DeWiCam can at most obtain information at the PHY (physical) and MAC (medium access control) layers, which makes the traditional classification methods that utilize 5-tuple (i.e., source IP address and port number, destination IP address and port number, and protocol) inapplicable. In addition, DeWiCam cannot extract per application flow but a mixture of all flows from a device, such as PCs and smartphones. Considering that wireless cameras are diverse in brands, hardware and application settings, the mixture of flows may appear similar as camera flows, not to mention there might be new models of cameras emerging from time to time. Besides, surrounding wireless traffic may impact the detection of wireless cameras. Last but not the least, wireless overhearing via smartphones at most reveals the existence of wireless cameras, but it cannot determine whether the cameras are inside a room, which is by far not enough.

To overcome all aforementioned challenges, we propose a camera flow identification method relying on only the PHY/ MAC layer header information. We analyze the compression and fragmentation standards, and find features that capture both the unique transient characteristics and the steady property of the traffic of wireless camera flows. Built on top of the transient features and the steady property, we are able to detect the existence of nearby wireless cameras. To determine whether a detected camera is in an interested room, we design a human-assisted method by monitoring the traffic changes caused by the human intervention. Together with the built-in on-board motion sensors in smartphones, DeWiCam can figure out whether a wireless camera is inside a room reliably. In summary, our contribution includes below.

- We propose a reliable and lightweight wireless camera detection approach using smartphones by analyzing the traffic features from PHY/ MAC layer headers.
- We investigate the distinctiveness of camera flows from non-camera ones and design an efficient detection scheme that can detect the exact number and existence of in-room cameras reliably and quickly.
- We implement DeWiCam on Android, and validate it on 20 popular wireless cameras in US and China market with 11000 evaluation traces across 30 days and 2 locations. The results show that DeWiCam can achieve above 99% detection accuracy with a detection time of 2.7 s.

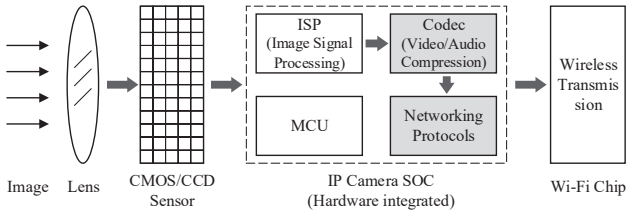



**Figure 2: Hardware modules inside a wireless camera. A scene image through a lens is first digitalized by a CMOS or CCD sensor. A dedicated video/ audio SoC chip is used for multimedia preprocessing, i.e., image signal processing (ISP), codec and transmission (networking protocols).**

## 2 BACKGROUND

In this section, we present the principle of wireless cameras from both the hardware and software perspectives.

### 2.1 Basics of Wireless Surveillance

Wireless cameras provide real-time video surveillance and are designed to stream videos through wireless networks that operate either on unlicensed frequencies (e.g., 2.4 GHz) or licensed frequencies (e.g., the 4.9 GHz public safety band). Two popular categories of wireless cameras exist: (1) Digital wireless cameras in the consumer market that are used as a low-cost solution to monitor a home or business, and the videos are typically transmitted over Wi-Fi networks; (2) Wireless video-surveillance cameras (e.g., closed-circuit television camera [33]) that are used for public safety or law enforcement agencies, and the videos are typically transmitted through government-only wireless networks. We focus on the first type, the ones used for home or business, and hereafter we call them *wireless cameras*.

Wireless cameras monitor homes or offices with the built-in image and audio sensors and generate continuous video as well as audio streams. Typically, the cameras process the video/ audio streams and then upload them through a wireless local area network (WLAN) to a cloud server, allowing remote monitoring either in real-time or aftermath. As such, videos and audios are always uploaded regardless of whether a user is watching them or not. Existing wireless cameras often support various functions, such as emitting sounds via a speaker or sending notification once motion within a field of view is detected. In addition, these cameras allow users to set up them via user-friendly interfaces, e.g., configure the resolution of videos, turn on or off the audio channel, and etc. Nevertheless, users typically use the default setup and at most configure the cameras once during installation.

### 2.2 Principle of Wireless Cameras

**Hardware modules.** IP cameras exploit integrated and standard design philosophy. The hardware modules for a wireless camera are shown in Fig. 2. CMOS/ CCD sensors capture and digitize image scenes to generate raw video frames. The raw frames are then fed into a multimedia SoC (system on chip), which contains a MCU and three additional submodules: (1) Image signal processing (ISP)

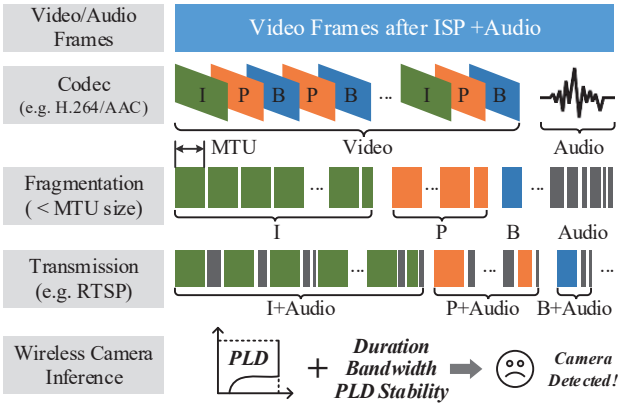



**Figure 3: Insight of DeWiCam. Video/ audio frames are first compressed. Then each frame is fragmented into a series of packets to fit in the MTU size. The fragmented video and audio packets are then combined to be transmitted. The final traffic is a series of large packets interleaved by small packets, which we define as the packet length distribution (PLD) pattern. Together with the "Duration", "Bandwidth Stability" and "PLD Stability" features, DeWiCam can detect wireless cameras.**

submodule performs functions such as noise filtering. (2) Codec submodule compresses video/ audio frames to decrease frame sizes. (3) Networking protocol submodule transmits multimedia stream via streaming protocol like RTSP (real-time streaming protocol). Popular SoC manufacturers include HiSilicon, Ambarella and TI, among which HiSilicon SoC is one of the most widely used on wireless IP cameras due to its low cost [31].

Essentially, the traffic patterns of wireless cameras depend on both the SoC chip and the camera manufacturers since the latter may configure the firmware of the SoC chip to customize additional functions. However, as the choices of SoC chips are limited as well as the main-stream manufacturers of wireless cameras, the operating flows of various wireless cameras are largely identical with minor differences, and lead to similar traffic patterns, as we reveal below.

**Compression and fragmentation.** The multimedia frames processed by the ISP submodule are compressed and fragmented before being transmitted, as illustrated in Fig 3, which causes the unique traffic patterns of wireless camera flows and sheds light on the feasibility of DeWiCam. Note that both compression and fragmentation are performed by the multimedia SoC, according to standard, thus, wireless cameras shall have similar traffic patterns.

*2.2.1 Compression.* Standard compression protocols such as H.264 [27] for video stream and AAC [5] for audio stream are utilized in the SoC. The H.264 standard utilizes redundancy between consecutive frames and outputs a series of Intra coded frames (I-frame), Predicted frames (P-frame) and Bi-directional predicted frames (B-frame), e.g., a sequence of *{I, P, B, P, B, P, B, I...}* [14, 37]. I frames are coded without reference and can be decoded by itself. P frames are coded based on the prior frames and B frames are coded based on both the prior and the subsequent frames. Thus, both P frames and B frames require other frames for decoding. Since I frames do not utilize inter-frame redundancy, they typically are a few times larger than the other two frame types, and P frames are larger than B frames. The number of P and B frames between two consecutive I frames is not fixed and depends on the video content.

An audio stream is compressed similarly. Audio signals are sampled continuously, and encoding protocols (e.g., AAC) are used on the audio samples to remove redundancy. A distinct difference between audio stream and video stream is that an audio stream occupies a much lower bandwidth compared with a video stream. Consequently, an audio stream is often lightweight and generates a lot of discrete traffic of small size, as is shown in Fig. 3.

*2.2.2 Fragmentation.* After being compressed, the video and audio stream frames will go through the streaming transmission protocol, e.g., the RTSP protocol in the application layer. Since each network interface in the link layer can only transmit a packet of a size smaller than its maximum transmission unit (MTU), when a frame is larger than MTU, it has to be fragmented. Therefore, a large frame is divided into a series of packets*.

For video streams, since I frames are typically the largest among the three types, the number of packets for an I frame is usually the largest as well. Denote the number of packets in a frame as $n$, and then the first $n-1$ packets are *full-size packets* (i.e., size of MTU) and the last one is typically smaller than the MTU size, which we call a *small-size packet*. P frames are also fragmented while they contribute to a less number of full-size packets than I frames do. For B frames, they are typically the smallest and each B frame may be encapsulated into one packet with a size less than MTU. Similarly, audio frames are also small-size packets.

## 2.3 Summary of Traffic Patterns

Compression and fragmentation over video/ audio streams have fundamental impact on the traffic patterns of wireless cameras. When passed to the network interface cards, video/ audio frames are packaged into a series of packets including both full-size packets and small-size packets as follows:

- I frames are divided into a set of full-size packets followed by a small-size packet.
- P frames are divided into a smaller set of full-size packets followed by a small-size packet.
- Most B frames and audio frames are small-size packets.

As a result, video and audio streams together generate a sequence of full-size packets with small-size packets in between. Such a pattern shall be a potential solution for wireless camera identification.

# 3 THREAT MODEL AND PROBLEM OVERVIEW

In this section, we represent the attack model, design requirement and the problem overview of DeWiCam.

## 3.1 Attack Model

We assume the attacker has full control over the camera, its surrounding and its location. To monitor the interested target in real time, the attacker tends to keep the camera on over a relatively long time period. To achieve the monitoring goal, the attacker is supposed to possess the following capabilities:

*To avoid confusion between a MAC layer frame and a video frame, we call the fragmented and encapsulated message in the link layer *a packet*.

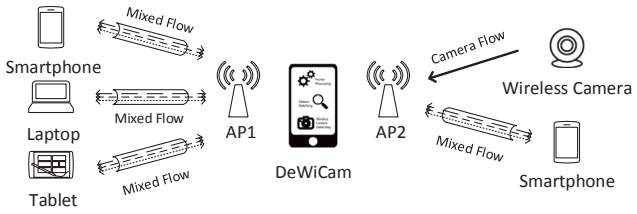



**Figure 4: DeWiCam detects a wireless camera device by examining the existence of a camera flow in the MAC layer. For non-camera devices, DeWiCam may abstract multiple flows from the same device into a virtual mixed flow.**

**Concealment of the camera.** The attacker is able to hide the camera anywhere in a room, but with the intention to monitor a large area.

**Deploying multiple cameras.** For the sake of coverage, the attacker can install any number of wireless cameras with a brand of his choices.

**Full control over networks.** The attacker has a full control over the network and can employ various protocols to enhance it, e.g., WPA/ WPA2-PSK. The attacker also can make the access point private by disabling the broadcast of its SSID (Service Set Identifier).

**Limited control over cameras.** We assume our attackers are normal Airbnb hosts. They can change the camera configuration based upon their demands but are unable to modify the bottom firmware of wireless cameras.

## 3.2 Requirements

To detect the unauthorized monitoring via wireless cameras, DeWiCam should satisfy the following requirements.

**No access to TCP/IP headers.** DeWiCam should be effective even without joining the network, because a victim may not have access to the network. Even with the Wi-Fi password, WPA/ WPA2-PSK exploit per-client, per-session keys derived from the Wi-Fi password plus the information exchanged when a client joins the network [19]. Thus, it is difficult to decrypt the network messages because the victim cannot always capture all four handshake packets of cameras to derive the keys. This stringent requirement makes DeWiCam more challenging than the existing work: existing traffic classification has the luxury of obtaining the TCP/IP headers, while DeWiCam can at most access the PHY/ MAC layer headers.

**Usability.** DeWiCam should detect the existence of wireless cameras within a short period, with a low false positive rate (FPR) and a low false negative rate (FNR). Moreover, multiple wireless cameras could be deployed in a house, DeWiCam shall be able to find all wireless cameras. Last but not least, DeWiCam should tell the victim whether the detected camera is inside the room or not.

**Lightweight.** We envision to implement DeWiCam as an App on ubiquitous smart devices like smartphones. Considering the restricted resources on smartphones, DeWiCam has to be lightweight in terms of CPU computation and memory usage.

## 3.3 Problem Overview

DeWiCam detects a wireless camera by inspecting wireless traffic. Given that only the PHY/ MAC header information is available, DeWiCam groups flows according to the MAC addresses of devices.

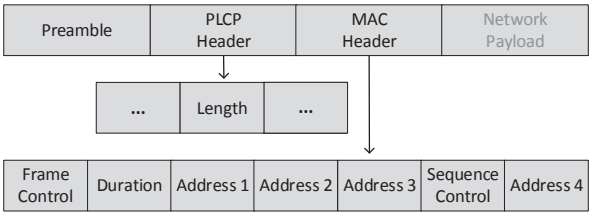



**Figure 5: The format of PHY and MAC headers.**

Therefore, *identifying a wireless camera is equivalent to determining whether there is a wireless camera flow.* As illustrated in Fig. 4, DeWiCam eavesdrops packets transmitted to or from all the nearby APs, and determines the existence of a wireless camera by identifying the camera flow. Note that a wireless camera only contains **one uplink flow** while non-camera devices may have several concurrent uplink flows, which we call the mixed flow. The key of DeWiCam is to differentiate camera flows with any other flows, including the mixed flows.

# 4 CHARACTERIZE CAMERA TRAFFIC PATTERNS

In this section, we analyze the camera traffic patterns and design features that only utilize the limited information in MAC headers yet capture the intrinsic patterns.

## 4.1 Available Frame Headers

A frame header includes the Physical Layer Convergence Procedure (PLCP) header and the MAC header, as shown in Fig. 5. The PLCP header contains the `length` of the MAC frame. The MAC header contains `Frame Control` (FC), `Duration`, and three `Address` fields [8]. Among them, `FC` indicates the state of the frame, including the direction of the packet (i.e., either uplink or downlink). The `Duration` field represents the channel reservation duration, i.e., the amount of time required to finish the transmission of the current packet and the related acknowledgment. For an uplink packet, `Address 1,2,3` are the MAC address of the AP, source MAC address, destination MAC address, respectively. Such limited information is the only basis to capture the intrinsic traffic patterns of camera flows for camera detection.

## 4.2 Applications

Below, we identify a few categories of popular applications that, by itself or in combination, might show similar traffic patterns as the one of wireless cameras.

*4.2.1 Real-time streaming applications.* They are designed for conference calls or entertainment (e.g., Skype audio/video calls, FaceTime, video gaming). Such applications typically consist of a few audio and video streams with each initiated from a participant. Each audio and video stream may utilize the same compression standards as the one of wireless cameras, but typically at a much lower resolution to satisfy the real-time requirement since Internet cannot guarantee a minimum bandwidth. Because of the real time

requirement, packets are typically transmitted as soon as they become available, resulting in a sequence of *small-size packets* with a few *full-size packets*.

*4.2.2 File sharing applications.* They can utilize a client-server architecture for a user to fetch files from a web server or to upload local files to a cloud storage server. Alternatively, they can utilize a peer-to-peer architecture to exchange files among users. Typical applications include uTorrent, YouTube and Internet surfing ones. Since file sharing has no strict timing requirements, a majority of their packets are *full-size* to reduce the overhead caused by small packets. Occasionally, small-size files or control-related messages will induce *small-size packets*.

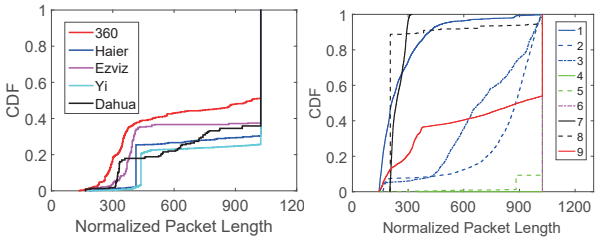
*4.2.3 Cloud-based Internet of Things (IoT) applications.* The traffic flows of IoT applications are diverse. Some of their communications are for command uploading or device status reporting, e.g., smart locks. Some may require to exchange a large amount of data that will not fit within one packet, e.g., an Amazon Echo may upload an audio clip to the server. Thus, the percentage of *full-size packets* and *small-size packets* may vary greatly among various IoT applications.

Besides the difference in packet size distribution, most applications consume an asymmetric amount of bandwidth for their upload and download link, and a few may exhibit symmetric ones (e.g., Skype calls). Given that a camera flow is typically heavy on upload link and light on download link, examining the bandwidth for upload and download links can assist in quickly eliminating obvious non-camera flows. However, applications such as a cloud file sharing and a Skype call with one party turning off his camera may exhibit similar bandwidth characteristics and therefore bandwidth by itself is insufficient.

## 4.3 Feature Overview

`DeWiCam` shall distinguish a camera flow from a non-camera flow, which could be one of following cases: (1) a non-camera flow from a single application of any aforementioned type, and (2) a non-camera flow from multiple applications (the mixed flow). The key of `DeWiCam` is to design features that characterize the distinctiveness of camera flows from any flows, including any mixed "virtual" flow. The number of features and their calculation complexity should be restricted to make `DeWiCam` lightweight.

We observe that camera flows have four characteristics. First, the flow consists of both full-size packets and small-size packets, as a result of video/ audio compression and fragmentation. Second, since cameras are typically mounted to monitor an indoor environment with less mobile objects for an extended amount of time, a camera flow has a relatively stable traffic volume and a stable packet size pattern. In particular, the number of P and B frames between consecutive I frames are more or less the same, and the sequence of packet sizes shall exhibit a unique pattern in terms of self-similarity. In addition, wireless cameras always work ceaselessly. Unless turned off, wireless cameras contribute a large amount of network traffic. Although the traffic volume may fluctuate due to network congestion and influence from other network flows, the traffic is relatively stable over time. Finally, the Wi-Fi modules of cameras are distinct from the ones of other platforms,




(a) Camera flows. (b) Non-camera flows.

**Figure 6: Packet length distribution (PLD) for camera and non-camera flows. 1-9 maps to: FaceTime, Skype video call, WeChat video call, BaiduCloud, Internet surfing, uTorrent, Awair Glow, Amazon echo and a mixed flow from a laptop.**

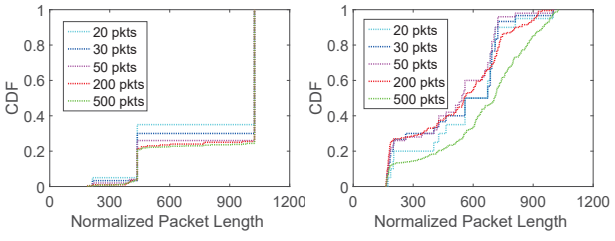
e.g., smartphones, PCs, tablets, etc. Cameras utilize the build-in Wi-Fi modules on the SoC, and other platform may use dedicated Wi-Fi modules. Thus, they may produce distinct information in the MAC headers. Based on these observations, we envision that features should include the ones model the transient characteristics caused by compression and fragmentation, stability nature as a result of one single streaming application, and hardware-related ones created by the SoC chip in cameras.

## 4.4 Feature Exploration

We introduce four features and our preliminary exploration on the effectiveness of each feature.

*4.4.1 Exploration experiment setup.* For exploration, we run a wireless camera in our lab and capture its packets with a USB Wi-Fi card and analyze the traffic pattern with Wireshark on a laptop. In addition to camera flows, we collect the following traffic flows in four categories: (1) *real-time streaming apps:* FaceTime, Skype video/ audio calls, WeChat video/ audio calls, QQ video/ audio calls and video gaming, (2) *file sharing apps:* BaiduCloud (a cloud storage service), uTorrent, YouTube, Flipboard, Amazon, Facebook, Instagram, and Weibo, (3) *cloud-based IoT apps:* a smart TV (Hisense), an Amazon Alexa echo, an air quality monitor (Awair Glow), and a smart bulb (TP-LINK) and (4) mixed flows.

*4.4.2 The transient PLD feature.* For ease of calculation, the first feature is the packet length distribution (PLD) over a short period of time, and we envision that it can distinguish camera flows from most other types of flows, but not all traffic flows, e.g., a non-camera device may consist of a mix of various applications that in combination mimics a camera flow. To make the plot readable, we plot the PLD of camera flows and a few applications in each category, as shown in Fig. 6. We observe that camera flows demonstrate a similar pattern even though they are of different brands. While for non-camera flows, most of the PLD curves vary significantly and differ from those of camera flows. More specifically, a camera flow has a large portion of full-size packets (1024 Bytes). The middle-size packets (between 500 to 1023 Bytes) occupy only a small proportion and there is a sharp increase for small-size packets (less than 500 Bytes). This is because that, for camera flows, audio streams and the B frames of video streams contribute to the small-size packets. Roughly speaking, the PLD pattern for camera flows is like a "stair" while non-camera flows show various different patterns. The PLDs




(a) PLD stability of Yi flows. (b) PLD stability of Skype flows.

**Figure 7: PLD stability. PLD curves of Yi are similar over various time periods (in terms of number of packets), while the PLD of Skype video behaves differently over time.**

of other applications all behave differently from that of camera flows, yet the mixed flow (flow 9 in Fig. 6(b)) shows a similar PLD as a camera flow. This confirms with our analysis.

## 4.5 The Stability Feature

To distinguish a camera flow from a mixed flow, we design two stability features.

*4.5.1 The bandwidth stability feature.* The first stability feature is the standard deviation of the flow bandwidth. We observe that the bandwidth of a camera flow is more stable compared with other applications. Given that a wireless camera has limited resolution choices and often stipulates a constant bitrate, the bandwidth for wireless cameras is therefore stable. We calculate the total bytes of data packets from a wireless camera within a period (i.e., bandwidth) and its deviation, as shown in Tab. 1. We observe that camera flows demonstrate a small deviation in bandwidth regardless of brands. Even in the extremely heavy network where several devices are operating in the same network, with each running several applications like Skype, BaiduCloud, etc., the bandwidth deviation of camera flows remains small. Non-camera flows such as Skype, WeChat change their bitrates based on the network condition and thus no stable bandwidth is observed in the experiment, e.g., the bandwidth standard deviation of Skype is 131.12 kbps.

*4.5.2 The PLD stability feature.* The second stability feature is the PLD stability feature. To illustrate, we use the packets from a camera of the brand Yi and compare the PLD with that of a Skype video call. Fig. 7 plots the PLDs for Yi camera and Skype video over various periods. It turns out that the PLD of Yi camera exhibits similar patterns while it changes over time for Skype video. To confirm that the stability of camera flows is better than that of non-camera flows, we quantify the self-similarity by calculating the two-sample Kolmogorov-Smirnov (KS) [20] distance and the p-value [35] of the PLD curves. The two-sample KS test is a useful nonparametric method to measure how close two probability distributions are. It is sensitive to differences in both point location and shape of the CDF curves. The p-value is the probability that used in the context of null hypothesis testing to quantify the idea of statistical significance of evidence. We use a PLD over 50 packets as reference and compare the PLDs under other conditions with it. The results are summarized in Tab. 2, from which we observe that the distance values of camera flow are much smaller than those of Skype flow, especially when the number of packets becomes larger.

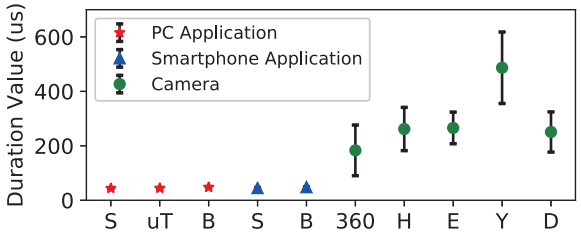



**Figure 8: Duration values of camera and non-camera flows (S: Skype, uT: uTorrent, B: BaiduCloud, H: Haier, E: Ezviz, Y: Yi and D: Dahua).**

The p-value is 1 for the camera flow, indicating that the similarity is extremely high.

## 4.6 The Hardware-Related Feature

To identify the hardware-related feature, we analyze each field in the frame headers, and we discover that the duration of a wireless packet is hardware dependent, instead of determined by the `length` of a packet according to the 802.11 specification [1, 8]. Thus, it can be utilized to reflect the hardware platform, i.e., camera, smartphone or PC. A comparison among various platforms are depicted in Fig. 8. Even though the `duration` values for camera flows vary across camera brands, the absolute values are all large compared with non-camera flows. In terms of the flows of different applications on the same device, we find that the `duration` values are application-independent. More importantly, the deviation of camera flows is remarkably larger than that of non-camera flows, which we believe is because camera applications exhibit continuous packet flows while others do not.

**Table 1: Bandwidth deviation (kbps) of camera flows.**

| Bandwidth Deviation | Brand | | | | |
|---|---|---|---|---|---|
| | 360 | Haier | Ezviz | Yi | Dahua |
| Light | 0.023 | 0.082 | 0.003 | 0.002 | 0.002 |
| Congested | 0.053 | 0.101 | 0.043 | 0.005 | 0.158 |

**Table 2: Distances and p-values of Yi and Skype flows.**

| # of packets | Yi | | Skype | |
|---|---|---|---|---|
| | Dist. | P-value | Dist. | P-value |
| 20 | 0.090 | 1 | 0.150 | 0.878 |
| 30 | 0.040 | 1 | 0.133 | 0.869 |
| 100 | 0.045 | 1 | 0.165 | 0.206 |
| 500 | 0.052 | 1 | 0.268 | 0.002 |

## 4.7 Summary

In summary, `DeWiCam` explores four fields (`length`, `duration`, `FC` and `Address`) in the PHY/ MAC header and exploits four features for camera flow identification.

- **The transient PLD feature**, which reveals the distinctiveness of the application type.
- **The hardware-related feature**, which utilizes the *duration* field and is related to the hardware platform, i.e., camera, smartphone or PC.
- **The bandwidth stability feature**, which indicates the traffic volume characteristic over time.

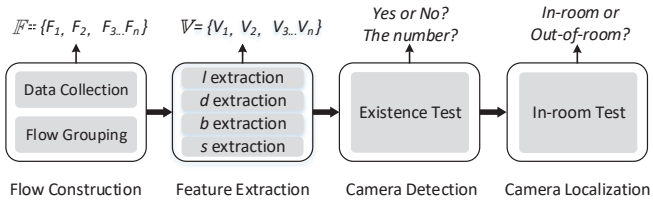



**Figure 9: The workflow of DeWiCam. DeWiCam groups collected packets into flows and extracts features for each flow. The detection module identifies camera flows, and the localization module determines the location of a camera.**

- **The PLD stability feature**, which reflects the operating characteristic of applications over time.

The first two features can differentiate most types of non-camera flows from a camera flow, i.e., all non-camera flows with a single application and a large portion of the ones from multiple applications. A mixed flow may happen to exhibit the same PLD and duration features transitorily, but it is unlikely to exhibit similar traffic patterns stably. Thus, we can extract the stability features to eliminate mixed flows.

# 5 DESIGN OF DEWICAM

## 5.1 Design Overview

In the following, we first introduce the overview of DeWiCam and then reveal the details of design.

**Workflow of DeWiCam.** DeWiCam is composed of four steps: flow construction, feature extraction, camera detection and camera localization, as shown in Fig. 9. First, DeWiCam captures a period of network traffic over wireless channels. DeWiCam removes redundant data, extracts the PHY/ MAC header information (e.g., frame lengths) for feature extraction and discards the MAC frame payload to improve processing efficiency. Then, DeWiCam groups the collected data into flows for further analysis and removes flows that are definitely non-camera flows. In the feature extraction step, DeWiCam extracts important features such as packet length, duration, bandwidth, and stability for each flow and feeds the flows with features into the camera detection classifier. The detection module examines the existence of a wireless camera flow by inspecting the flow features and finally the localization module determines whether the camera is in an interested room or not.

## 5.2 Flow Construction

The first step of DeWiCam is to construct flows associated with each device from packet traces.

The flow construction module mainly contains two phases: data collection and flow grouping. In the data collection phase, DeWiCam captures data packets over the air with filtering that discards packets that are unlikely to be from cameras. Since wireless cameras mainly generate uploading streams with a destination of an AP, we remove both non-data packets and downlink packets but record the bandwidth of downlink associated with each uplink, for further removing non-camera flows. Non-data packets include ACK, RTS, CTS and other management packets and uplink/ downlink flows are identified by reading the FC field in the MAC frame header [8]. As such, DeWiCam only collects potential camera-relevant data.

After the data collection phase, we have a set of packets $\mathbb{P} = \{P_1, P_2, P_3 \ldots P_m\}$, where $P_i$ is the $i^{th}$ packet. We group the packets in $\mathbb{P}$ according to their MAC addresses and put packets with identical source and destination MAC addresses to the same group, denoted as $F_i$. We then remove the flows with a significant downlink bandwidth, which are definitely non-camera flows, e.g., file sharing flows and Skype flows, to improve detection efficiency. In the end, the flow construction module outputs a flow set $\mathbb{F} = \{F_1, F_2, F_3 \ldots F_n\}$ containing all the candidate flows.

Note that in the grouping step, we regard all packets belonging to the same MAC pair as a flow. Each flow in $\mathbb{F}$ may be one camera flow or a mix of multiple flows from a non-camera device, e.g., a laptop or a smartphone, as is illustrated in Fig. 4.

A natural question is *"How much data should be collected?"*. A larger number of packets can increase detection accuracy at the cost of a longer delay, while fewer packets produce a faster response at the cost of a less reliable result. To strike the balance, DeWiCam chooses the required number of packets empirically, e.g., 300 packets. It is also worth mentioning that wireless networks often suffer from packet loss, but it is okay not to capture all packets. Our experiments on analyzing the detection accuracy show that as few as 300 packets suffice to identify a camera flow, which takes at most several seconds in various networks, including the ones with low-bandwidth or severe contention.

## 5.3 Feature Extraction

After grouping packets into flows, DeWiCam extracts the transient PLD feature $\vec{l}$, the hardware-related feature $d$, the bandwidth stability feature $b$ and the PLD stability feature $s$ for all the candidate flows in $\mathbb{F}$ to detect the camera flow. We build a 4-dimension vector $\vec{V} = \{\vec{l}, d, b, s\}$ for each flow and construct the feature vector set $\vec{\mathbb{V}}$ for all the candidate flows in $\mathbb{F}$, as follows.

**The transient PLD feature $\vec{l}$.** DeWiCam records the `length` of packets belonging to the same flow and calculates the PLD feature $\vec{l}$ using the cumulative distribution function (CDF) over a small number of packets (e.g.,50 packets).

**The hardware-related feature $d$.** DeWiCam calculates the hardware-related feature $d$ as the standard deviation of the `Duration` field of packets belonging to the same flow.

**The bandwidth stability feature $b$.** DeWiCam calculates the bandwidth stability feature $b$ as the standard deviation of the instantaneous bandwidth of each flow.

**The PLD stability feature $s$.** DeWiCam extracts the PLD stability feature $s$ over the PLD feature. For the $i^{th}$ flow, DeWiCam chronologically divides the collected packets into $N$ blocks (e.g., 50 packets per block in our implementation). DeWiCam extracts the PLD feature $\vec{l}_{ij}$ for each of the $j^{th}$ block, and finally obtains a PLD feature set $\vec{L}_i = \{\vec{l}_{i1}, \vec{l}_{i2}, \ldots \vec{l}_{iN}\}$ for the $i^{th}$ flow. Then, DeWiCam utilizes KS test for $\vec{L}_i$, and calculates the stability $s_i$ for the $i^{th}$ flow, as follows.

$$s_i = \frac{\sum_{j=2}^{N} \sup_x \left| \vec{L}_i[j](x) - \vec{L}_i[1](x) \right|}{N-1} \quad (1)$$

where the PLD feature of block 1 is used as reference and the KS test of $L_i[1]$ and $L_i[j]$ returns the maximum vertical distance between the two distributions.

## 5.4 Camera Detection

DeWiCam utilizes supervised learning to detect wireless camera flows with the extracted feature vector $\vec{V} = \{\vec{l}, d, b, s\}$. For the sake of high classification accuracy and robustness over a single classification algorithm, we employ an ensemble classification approach ExtraTrees [12]. ExtraTrees fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve prediction accuracy and avoid over-fitting.

**Training**. During the training process, $x$ traces (each contains 300 packets) from each camera are utilized as the positive samples and $x$ traces from non-camera applications as well as mixed flows serve as the negative samples to train the classifier.

**Testing**. To detect cameras, DeWiCam collects a number of packets (300 packets) over the air and extracts the feature vector set $\mathbb{V} = \{\vec{V}_1, \vec{V}_2, \vec{V}_3 ... \vec{V}_n\}$. Then, DeWiCam feeds it to the classifier and outputs the MAC address(es) of the detected camera(s), if any.

**Multi-camera detection.** DeWiCam can detect multiple cameras. In principle, the detection of multiple cameras is the same as detecting one camera. DeWiCam outputs the exact number of detected cameras. In Sec. 6, we provide the detection results of up to three cameras running at the same time and location.

## 5.5 Camera Localization

Identifying whether a flow is a camera flow can at most recognize the existence of cameras. Users are interested in whether there are wireless cameras inside the room that directly videotape her/ him. Thus, DeWiCam should not only identify wireless cameras nearby, but also determine whether at least one is in the room.

To address this, we ask a user to perform a simple task such that should a camera be in the room, the camera flow may change accordingly. The reason is that if a camera is inside the room and videotaping the victim, the video content of the camera flow is likely to change with the human intervention. Specifically, if users intentionally move in front of where a camera may be located, the camera may capture extra motions which increases the I, P, B frame sizes, and thus the camera flow bitrate.

DeWiCam requires the user to move, e.g. walk around the room to conduct human intervention. If a wireless camera is directly videotaping the user, the bitrate should show a significant rise when the user begins moving, and a fall when the user stops. However, due to the dynamics of network environment, the bitrate of a camera flow may fluctuate even without human intervention. To differentiate this situation, DeWiCam utilizes the accelerometer on a smartphone to extract the intervention time window automatically. By letting a user move and stop for a few times, DeWiCam shall observe the rising and falling of bitrate accordingly. An illustration is shown in Fig. 10, in which the user moves during the period from 15 s to 30 s. The accelerometer readings from the smartphone indicate that the bitrate of the camera reacts correspondingly with the user movement. As such, DeWiCam can confirm it's a flow from an in-room wireless camera. It is also worth mentioning that it takes about 1 second in our case for the camera bitrate to change after the movement, because the camera has to process the sensory data and pass it to the network interface card.

To quantify the changes of bitrate caused by human intervention, DeWiCam utilizes the CUSUM (cumulative sum control chart) [34]

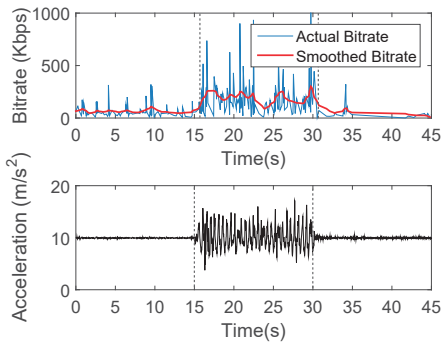



**Figure 10: Impact of human intervention on the bitrate of camera flows.**

algorithm to detect both the rising and falling edges of the bitrate sequence $r$. CUSUM is a sequential analysis technique typically used for change monitoring. The CUSUM test for edge detection is as follows:

$$U_k = \begin{cases} 0, & k = 0 \\ \max(0, U_{k-1} + r_k - w_k), & k > 0 \end{cases} \quad (2)$$

$$L_k = \begin{cases} 0, & k = 0 \\ \min(0, L_{k-1} + r_k - w_k), & k > 0 \end{cases} \quad (3)$$

$$Condition : U_k > \delta_1, L_k < \delta_2 \quad (4)$$

where $U_k$ and $L_k$ are the upper and lower cumulative sum at the time $k$. $w$ is the likelihood estimation of the bitrate sequence. $\delta_1$ and $\delta_2$ are thresholds for detecting an in-room camera. If the value of $U_k$ and $L_k$ exceed $\delta_1$ and $\delta_2$ during the moving phase identified by the accelerometer, in other words, the positive change and the negative change are larger than the thresholds, the camera is considered to be in-room.

## 5.6 Implementation

We implement DeWiCam on the Android platform. Normally, a smartphone sets its Wi-Fi card in the managed mode where packets not destined for the smartphone are discarded [6]. However, DeWiCam requires to eavesdrop all the wireless traffic in the air for analysis. To this end, the Wi-Fi card has to be changed into the monitor mode such that DeWiCam is able to capture and record all the passing packets. We implement the monitor mode function based on an open-source project named Nexmon [22]. Nexmon provides a basic API for Wi-Fi driver modification on the Android platform. We use a UDP socket to read packets from the `rawproxy` application which connects to the Wi-Fi card buffer in Nexmon. Once receiving a packet via the UDP socket, the packet header is decoupled and recorded. If enough packets are received, the headers from those packets are pushed to the detection algorithms.

# 6 PERFORMANCE EVALUATION

In this section, we evaluate the performance of DeWiCam. In particular, we evaluate (1) detection accuracy of wireless cameras, and (2) localization (in-room or out-of-room) inference accuracy of detected cameras. We conduct both offline evaluation with collected data traces from the 20 aforementioned cameras with various settings and online detection by running the DeWiCam App on a smartphone in real time.

## 6.1 Experimental Setup

We test DeWiCam in an apartment as well as in a lab with 20 cameras of various settings and network conditions. The detailed settings are as follows.

**Home scenario.** In an apartment, we place three cameras in three rooms. Fig. 16 (in the appendix) depicts the floor plan and the spots (A, B and C) where cameras are deployed. We hide cameras inside furniture, e.g., a wardrobe or a bedside table, which imitates where the attackers may place them. The AP that cameras are attached to has a bandwidth of 6 Mbps, and has 3 other connected devices. In addition, 6 neighboring APs and 19 Internet-connected devices are found. The average downlink bandwidth utilization is around 42%.

**Lab scenario.** For the lab scenario, 13 APs and 28 devices are within the detection range of DeWiCam. The AP that cameras are attached to has a bandwidth of 12 Mbps, and has 8 other connected devices. The average bandwidth utilization is 74% (downlink) and we regard network in the lab scenario heavy-load.

**Camera brands**. 20 cameras of 12 typical brands on the market from both the US and China are used, as shown in Tab. 5 (in the appendix).

**Operation platform**. We implement DeWiCam on the Android platform with an LG Nexus 5, and the PC platform, with a Lenovo ThinkPad T440p and a USB wireless adapter TL-WN722N. Given space limitations, we only demonstrate the performance of Android-based DeWiCam in this paper. Nevertheless, PC-based DeWiCam can even achieve slightly better performance.

**Camera and AP number**. We test DeWiCam with a various number of cameras and APs. Our settings include single-camera-multi-AP and multi-camera-multi-AP.

**Dataset**. We collect traces from 20 cameras under various conditions, as is shown in Tab. 5 (in the appendix). The data trace is collected during a 30-day period for all the cameras. The total size of the dataset is 11000 traces, in which 9000 are camera traces while the rest 2000 are traces without cameras.

## 6.2 Performance Metrics

**Detection accuracy metrics.** We calculate the true positive rate and the true negative rate of DeWiCam. We run DeWiCam for each collected trace, and quantify the accuracy by calculating the true positives and true negatives over the total number of traces. Hereby we define $TPR = \frac{TP}{TP+FN}$ as the true positive rate, $TNR = \frac{TN}{TN+FP}$ as the true negative rate.

**Detection time.** The detection time is the time needed to detect a wireless camera after initiating DeWiCam.

## 6.3 Camera Detection Performance

In this section, we first evaluate the appropriate size of training dataset and then the overall camera detection performance.

*6.3.1 Impact of Training Dataset.* To evaluate the appropriate size of training data, we train the system with $x$ traces from each setting of each camera as well as non-camera applications, and test it with the rest traces. $x$ is set to 5, 10, 15 and 20 respectively, and the overall TPR and TNR are plotted in Fig. 11. From the results, we can find that even with 5 training traces, the overall TPR and TNR are above 98.9%. With the increasing of training data size, both TPR and TNR are slightly improved. Given the training size 10, DeWiCam is able to achieve an overall TPR and TNR of 99.7% and 99.0%. To strike the balance between usability and accuracy, we choose 10 traces for training and use 10 traces in the rest of the evaluation.

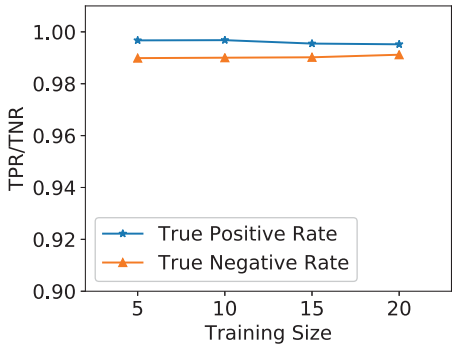


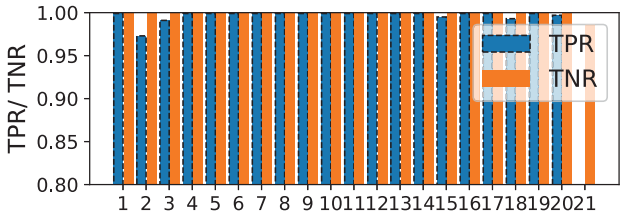

**Figure 11: The overall TPR and TNR with different training data sizes.**

*6.3.2 Single-Camera Detection Accuracy.* We evaluate the detection performance of DeWiCam in the home scenario where DeWiCam detects 7 APs nearby, including the one that wireless cameras (deployed in the spot A) are connected to. We also test DeWiCam when there is no camera at all but applications running in the background for comparison. Background applications include Skype, WeChat, Youku, etc. We show the TPR and TNR for each camera and the no-camera case in Fig. 12(a). From the figure, we have the following findings. First, TPR for all the 20 cameras are above 97.3% with an average of 99.7%. Specifically, Camera 2 (Dahua LeChange TC5) shows the worst detection accuracy with 97.3%. We examine its traces and find that its pattern is not as stable as other cameras. We suspect that since Camera 2 supports an adaptive bitrate, the traffic pattern changes over time. Camera 3 from the same manufacturer also shows the same problem. In addition, Camera 15 (Nest Cam Indoor)'s TPR is 99.2%. The reason is that it requires a high bandwidth while the bandwidth in our test scenario fails to satisfy occasionally. As a result, Nest was disconnected several times during the test. Second, the average TNR of the 20 cameras achieves 99.0%. Even the TNR of the non-camera case is larger than 98.6%. Those false alarms mainly come from the confusing mixed flows, which happened to be stable during the test.

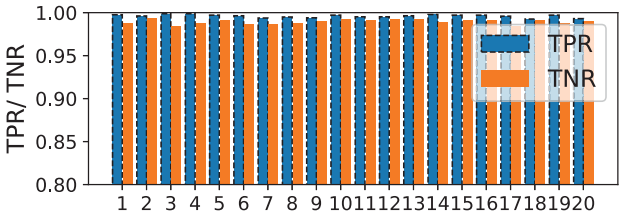
**Table 3: Multi-camera detection TPR (%).**

| # of cameras | # of detected cameras | | |
|---|---|---|---|
| | **1** | **2** | **3** |
| 2 | 99.9 | 97.8 | - |
| 3 | 99.9 | 99.9 | **75.9** |

*6.3.3 Multi-Camera Detection Accuracy.* DeWiCam can support multi-camera detection even though such cases do not happen frequently. We evaluate this with three randomly chosen cameras: Camera 1, Camera 5 and Camera 19 deployed in spot A, B and C in Fig. 16, connecting to the same AP in the home scenario. The results are shown in Tab. 3. When there are two cameras, the possibility to detect all the two cameras is 97.8% and it is 99.9% to detect any of them. For the three-camera case, the detection of all the three cameras is 75.9%, and it is 99.9% to detect two of the three cameras. We find that it is the camera 19 (Haier HC6700) that is




(a) The overall TPR and TNR performance under the single-camera-multi-AP case.




(b) The overall TPR and TNR performance with alien cameras.

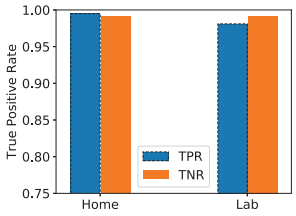
**Figure 12: Evaluation of (a) single-camera detection and (b) impact of alien cameras.**

most likely to miss the detection for both the two-camera and the three-camera cases. The reason is that in the three-camera scenario, the network is saturated, i.e., the traffic from the three cameras and other applications fully utilizes the bandwidth, which causes severe contention and packet losses. From trace analysis, we find that the bitrate of the Haier is lower and fluctuates more than the other two cameras due to contention, which in turn may influence the bandwidth feature and thus bypass the detection sometimes. Nevertheless, DeWiCam can still detect Haier camera with 75.9% possibility.

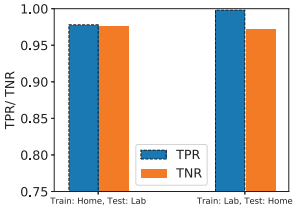
*6.3.4 Impact of Other Network Traffic.* To investigate how other network traffic affects the performance of DeWiCam, we also conduct experiments with/ without cameras (Camera 1-20) in the lab scenario where the network is congested, i.e., 13 APs and 28 devices are within the detection range of DeWiCam. The results shown in Fig. 13(a) demonstrate that even in the congested lab network, DeWiCam can still achieve 98.1% TPR and 99.2% TNR on average, and shows no significant decrease compared to that in the home scenario. The reason is that DeWiCam examines the patterns of each uplink network flow independently. Other traffics may have impact on a camera flow in terms of detection time because a minimum number of packets are required and it takes extra time to collect them in congested networks. However, they don't greatly impair detection accuracy.

*6.3.5 Ability to Detect New Cameras.* To evaluate the ability to detect alien cameras that are not part of training, we train DeWiCam with 19 cameras and test it with all 20 cameras in the home scenario, with each camera serves as an alien camera in turn (a.k.a., leave-one-out cross-validation). The overall TPR and TNR performance to detect the 19 trained cameras as well as the alien camera are plotted in Fig. 12(b). Form the results, we can observe that even with an alien camera, DeWiCam is able to achieve average TPR and TNR of 99.6% and 98.9%, showing no difference compared with the no-alien case (with TPR and TNR of 99.7% and 99.7%). In addition, the choice of alien cameras has little impact on the performance of DeWiCam, since no significant differences are found in the overall performance against different alien cameras.

*6.3.6 Performance across Environments.* To evaluate the performance of DeWiCam across environments, i.e., the ability to detect cameras in a new environment, we train DeWiCam with traces collected from the home (lab) and test it with the traces collected from the lab (home). The results shown in Fig. 13(b) reveal that even in a new environment without training, DeWiCam also shows good performance with average TPR and TNR of 98.8% and 97.4%.




(a) The overall TPR and TNR performance with different network traffic density.




(b) The overall TPR and TNR performance with new environments.

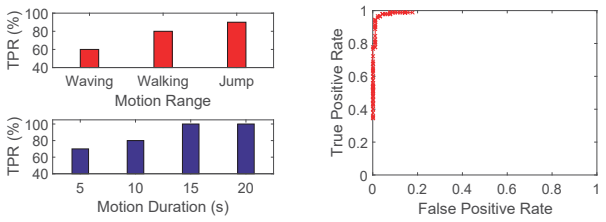
**Figure 13: The performance of DeWiCam with (a) different network traffic density and (b) untrained environments.**

## 6.4 Camera Location Inference Accuracy

In this section, we evaluate the inference accuracy of inside cameras. This set of experiments are conducted with Camera 1, 5 and 8 in the home scenario.

*6.4.1 Requirement of Motion Range.* Motion intervention is the key of the inference procedure. Thus, motion range plays an important role. We first evaluate the requirement of motion range and conduct experiments with three common motions: *waving* (wave hands), *walking* (walk around the room) and *jumping* (jump up and down). The motion range and its intensity is increased gradually. In the experiments, motion duration is set to 10 s and we repeat 10 rounds of tests for each motion. The result is depicted in Fig. 14(a). From the results, we find that the inference accuracy increases along with the growth of motion range. This coincides with the analysis that the more motion changes are captured by a camera, the larger the resulting video frames are. Specifically, waving produces an accuracy of 60% while jumping achieves the highest with 90%. However, we envision that not all users can or are willing to jump. To strike the balance between usability and accuracy, we choose walking as the recommended motion and can improve the inference accuracy by asking users to alternate between walking and halting for several times.

*6.4.2 Impact of Motion Duration.* Another factor that affects the accuracy is motion duration. Long duration is able to reduce the impact of the transient variation of bitrate caused by the dynamic network environment and thus improves the accuracy. Short duration, on the other hand, provides a fast response. To find out the proper motion duration, we conduct experiments with four time periods: 5 s, 10 s, 15 s and 20 s. The results are depicted in the bottom of Fig. 14(a). The results confirm that longer duration leads to higher accuracy. Specifically, a 5 s motion duration can only guarantee 70% accuracy. With the increase of duration, the




(a) The TPR performance with 3 body motions and 4 motion durations.

(b) The ROC curve of camera inference model.

**Figure 14: Camera location inference performance against different motion ranges and durations (a) and the ROC curve of the inference model (b).**

accuracy is promoted to 80% with 10 s and 100% with 15 and 20 s. To strike the balance between accuracy and response delay, we choose 15 s as the default duration in DeWiCam.

*6.4.3 In-room Camera Inference Accuracy.* We evaluate the overall inference accuracy with the three cameras using the recommended motion "walking" and the default duration time 15 s. Three volunteers participate in the experiments. The basic motion segment is composed of a static period (5 s), a walking duration (15 s) followed by another static period (5 s), just like the moving style shown in Fig. 10. The evaluation is conducted in the room 1 and repeated for 30 rounds for the in-room camera (deployed in the spot A) test as well as the out-of-room camera (deployed in the spot B) test. The results in Tab. 4 reveal that DeWiCam can identify an in-room camera with an average TPR of 97.8% while the out-of-room camera shows 3.3% false positive rate. It verifies that human intervention does influence the pattern of in-room cameras while have little effect on the out-of-room ones. False positives are mainly caused by the fluctuation of bitrate resulted from the dynamic of network environment. Based on the ROC curve in Fig. 14(b), one can utilize large thresholds ($\delta_3$ and $\delta_4$ in Equ. (4)) to achieve a low false positive rate at the cost of a slightly lower TPR.

**Table 4: The overall accuracy of camera location inference.**

| Camera | TPR (%) | TNR (%) |
|---|---|---|
| 1 | 100 | 93.3 |
| 5 | 96.7 | 96.7 |
| 8 | 96.7 | 100 |

## 6.5 Real-time Performance

To evaluate the real-time detection performance of DeWiCam, we choose 7 cameras and run the application on Nexus 5 to detect each of the 7 cameras for 10 rounds in the home scenario. The results show that all 70 detections succeed and the CDF curves of detection time for 7 cameras are depicted in Fig. 15. From the results, we observe that average detection time is 2.7 s and 90% of the detections complete within 6.2 s. Camera 2, 8, 15, 16 achieve a fast detection speed where all detections are finished within 3 s. Camera 11 takes the longest time and can be detected with an average time of 6.5 s.

## 7 DISCUSSION

**Attacker Defense.** A natural question about DeWiCam is *"Can DeWiCam be easily bypassed if the attacker knows how it works?"*.

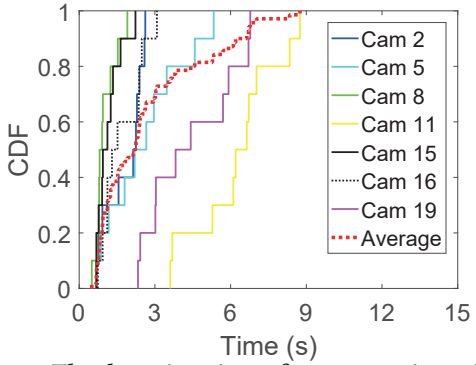



**Figure 15: The detection time of 7 cameras in real time by running DeWiCam.**

DeWiCam is built on top of the intrinsic traffic patterns of wireless cameras. If the attacker intends to bypass the detection of DeWiCam, he has to change the distinct features, i.e., the length, duration fields of each packet and the statistical bandwidth of the wireless camera flow. In other words, the attacker has to modify the codec (compression) and networking protocols. As we mentioned before, the codec and networking protocols are integrated into the SOC and the interfaces are not accessible. Even if the attacker is able to reprogram the SOC, the changes of codec and networking protocols will lead to decoding problems. Therefore, it is challenging to bypass DeWiCam, especially for normal users like Airbnb hosts.

**Root.** DeWiCam on smartphone requires rooting the phone, while DeWiCam on PCs does not. Looking into future, rooting will not be necessary if Android/ iOS can open-source the wireless driver API, or if Google recognizes the usefulness of DeWiCam, they may integrate DeWiCam to Android so that it can capture packets freely.

**Limitations.** With the elaborate design, DeWiCam achieves high performance most of the time. However, DeWiCam has the following limitations. First, DeWiCam is mainly applicable to wireless cameras such as the Wi-Fi cameras. For cameras with local-storage or wired cameras, DeWiCam is inapplicable. However, considering the booming market of wireless cameras, we believe DeWiCam can be applied in more scenarios.

Second, DeWiCam is designed based on the general principle of wireless cameras, i.e., compression and fragmentation on video and audio streams. If the camera working mode changes dramatically in the future, DeWiCam may not function well and should be updated to incorporate the new working mode accordingly.

## 8 RELATED WORK

**Hidden camera detection.** Existing approaches for hidden camera detection on smartphones can be divided into three categories. The first category detects the signs of light reflection caused by the lens of a hidden wireless camera. This approach is widely used in applications [21, 25, 36] on both the Android and iOS platforms. In reality, this approach assumes that the users are aware of the approximate location of the cameras, and can shed light towards the lens, which may not be true. The second category is based on electromagnetic waves (produced by the crystal oscillator) emitted by cameras [10, 11]. The EMI signal is claimed to be detectable with the magnetometers on smartphones. The third category detects

wireless cameras by sensing the existence of RF signals on the 2.4 GHz or 5 GHz frequency. The latter two approaches show poor performance in practice due to the interference of EMI or RF signals from other devices like computers.

**Traffic classification.** Essentially, DeWiCam belongs to the traffic classification (TC) problem. Traditional TC approaches [4, 15] mainly take the 5-tuple information as inputs and assume the traffic is not encrypted. Classification for encrypted traffic aims to identify protocols [26] and even sensitive information within encrypted streams such as VoIP calls [32]. The difference between DeWiCam and existing traffic classification work is that DeWiCam can only use the PHY/ MAC layer information instead of TCP/ IP information. This makes DeWiCam much more challenging as traffic is processed with a coarse granularity, i.e., in terms of MAC pair rather than TCP/ IP connection.

**Privacy concerns with cameras.** Recently, an increasing attention is paid to safeguard personal privacy against cameras. Ashok et al. [17] introduces an "invisible light beacon" implemented on the eye-wear to prevent unauthorized videotaping, by which the privacy preferences of photographed users are communicated to photographing cameras. The authors in [2, 30] focus on the privacy concerns caused by "first-person" wearable cameras. They propose to identify and prevent the sharing of sensitive images captured by wearable cameras. Birnbach et al. [3] detects drones carrying out privacy invasion attacks with on-board cameras. They analyze the RSS (received signal strength) of the wireless traffic from the cameras to detect the approaching of drones.

Inspired by previous work, DeWiCam exploits the intrinsic traffic patterns of wireless cameras to achieve the detection of wireless cameras.

## 9 CONCLUSION AND FUTURE WORK

In this paper, we propose DeWiCam, a lightweight and effective approach for wireless camera detection. DeWiCam analyzes the intrinsic traffic patterns of a camera flow by investigating the PHY/ MAC layer headers and detects an in-room camera with human intervention. DeWiCam is implemented on the Android platform and extensive evaluation results indicate DeWiCam can achieve over 99% detection accuracy with only 2.7 s.

Future work should investigate whether DeWiCam performs well with more unknown camera brands and explore extra functions such as resolution inference.

## ACKNOWLEDGMENTS

This work has been funded in part by NSFC 61472358, NSFC 61702451, and the Fundamental Research Funds for the Central Universities 2017QNA4017.

## REFERENCES

[1] Giuseppe Anastasi and Luciano Lenzini. 2000. QoS provided by the IEEE 802.11 wireless LAN to advanced data applications: a simulation analysis. *Wireless Networks* 6, 2 (2000), 99–100.
[2] Ashwin Ashok, Viet Nguyen, Marco Gruteser, Narayan Mandayam, Wenjia Yuan, and Kristin Dana. 2014. Do not share!: invisible light beacons for signaling preferences to privacy-respecting cameras. In *Proceedings of the 1st ACM MobiCom workshop on Visible light communication systems*. ACM, 39–44.
[3] Simon Birnbach, Richard Baker, and Ivan Martinovic. 2017. Wi-Fly?: Detecting Privacy Invasion Attacks by Consumer Drones. In *NDSS*.
[4] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. 2007. Revealing skype traffic: when randomness plays with you. In *ACM SIGCOMM Computer Communication Review*, Vol. 37. ACM, 37–48.
[5] Marina Bosi, Karlheinz Brandenburg, Schuyler Quackenbush, Louis Fielder, Kenzo Akagiri, Hendrik Fuchs, and Martin Dietz. 1997. ISO/IEC MPEG-2 advanced audio coding. *Journal of the Audio engineering society* 45, 10 (1997), 789–814.
[6] Danny Bradbury. 2011. Hacking wifi the easy way. *Network Security* 2011, 2 (2011), 9–12.
[7] Antonio Castelan and John Treanor. 2016. *Hidden cameras found inside a Las Vegas Airbnb rental recording naked people.* https://tinyurl.com/zrtysgx.
[8] ICSLMS Committee et al. 1997. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std* 802 (1997), 50.
[9] Laura Connor. 2016. *Airbnb host admits 'using hidden cameras to film guests and swaps sex videos with other users'.* https://tinyurl.com/y9kem8wm.
[10] FutureApps. 2017. *Hidden Camera Detector.* https://tinyurl.com/y7nlhbx8.
[11] GalaxyApp. 2016. *Hidden Camera Detector Pro.* https://tinyurl.com/y745nqf7.
[12] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.
[13] Jack Smith IV. 2015. *Couple Wakes Up in Airbnb to Find Hidden Camera Watching Them.* https://tinyurl.com/yd556cwm.
[14] Minqiang Jiang, Xiaoquan Yi, and Nam Ling. 2004. Improved frame-layer rate control for H. 264 using MAD ratio. In *Proceedings of the 2004 International Symposium on Circuits and Systems*, Vol. 3. IEEE, III–813.
[15] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. 2005. BLINC: multilevel traffic classification in the dark. In *ACM SIGCOMM Computer Communication Review*, Vol. 35. ACM, 229–240.
[16] Randy Kenner. 2012. *Man Sues After Finding Hidden Cam In Hotel Bathroom.* http://www.rense.com/general29/msn.htm.
[17] Mohammed Korayem, Robert Templeman, Dennis Chen, David Crandall, and Apu Kapadia. 2016. Enhancing lifelogging privacy by detecting screens. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4309–4314.
[18] KpopJoA. 2017. *Korean couple discover a hidden camera inside Airbnb guesthouse in Japan.* https://tinyurl.com/y9ju55ut.
[19] Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and Behrang Samadi. 2009. A survey on wireless security protocols (WEP, WPA and WPA2/802.11 i). In *Proceedings of the 2nd International Conference on Computer Science and Information Technology*. IEEE, 48–52.
[20] Hubert W Lilliefors. 1967. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association* 62, 318 (1967), 399–402.
[21] LLC LSC. 2016. *Hidden Camera Detector.* https://itunes.apple.com/us/app/hidden-camera-detector/id532882360?mt=8.
[22] D. Wegemer M. Schulz. 2005. *Nexmon.* https://github.com/seemoo-lab/nexmon.
[23] News Corp Australia Network. 2016. *Airbnb host admits to filming people having sex on hidden cameras.* https://tinyurl.com/j7cw2a5.
[24] Circa News. 2016. *Experts say that spy cameras in Airbnb rentals are becoming harder to detect.* https://tinyurl.com/y8q6khsm.
[25] Asher L. Poretz. 2017. *Spy hidden camera Detector.* https://itunes.apple.com/us/app/spy-hidden-camera-detector/id925967783?mt=8.
[26] Brendan Saltaformaggio, Hongjun Choi, Kristen Johnson, Yonghwi Kwon, Qi Zhang, Xiangyu Zhang, Dongyan Xu, and John Qian. 2016. Eavesdropping on Fine-Grained User Activities Within Smartphone Apps Over Encrypted Network Traffic. In *WOOT*.
[27] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the scalable video coding extension of the H. 264/AVC standard. *IEEE Transactions on circuits and systems for video technology* 17, 9 (2007), 1103–1120.
[28] Bunker Hill Security. 2017. *Wireless Camera RF Detector.* https://www.harborfreight.com/wireless-camera-rf-detector-95053.html.
[29] Technavio. 2017. *Global Wireless Video Surveillance Market 2017-2021.* https://tinyurl.com/y7u9yfl4.
[30] Robert Templeman, Mohammed Korayem, David J Crandall, and Apu Kapadia. 2014. PlaceAvoider: Steering First-Person Cameras away from Sensitive Spaces. In *NDSS*. 23–26.
[31] UNIFORE Website. 2014. *Introduction to Hisilicon IP Cameras.* https://tinyurl.com/yaolrcje.
[32] Andrew M White, Austin R Matthews, Kevin Z Snow, and Fabian Monrose. 2011. Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks. In *2011 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.
[33] Wikipedia. 2017. *Closed-circuit television camera.* https://en.wikipedia.org/wiki/Closed-circuit_television_camera.
[34] Wikipedia. 2017. *CUSUM.* https://en.wikipedia.org/wiki/CUSUM.
[35] Wikipedia. 2017. *P-value.* https://en.wikipedia.org/wiki/P-value.
[36] Workshop512. 2013. *Glint Finder-Camera Detector.* https://tinyurl.com/pbnfnur.
[37] Xin-Wei Yao, Wan-Liang Wang, Shuang-Hua Yang, Yue-Feng Cen, Xiao-Min Yao, and Tie-Qiang Pan. 2014. Ipb-frame adaptive mapping mechanism for video transmission over ieee 802.11 e wlans. *ACM SIGCOMM Computer Communication Review* 44, 2 (2014), 5–12.

# A APPENDICES

## A.1

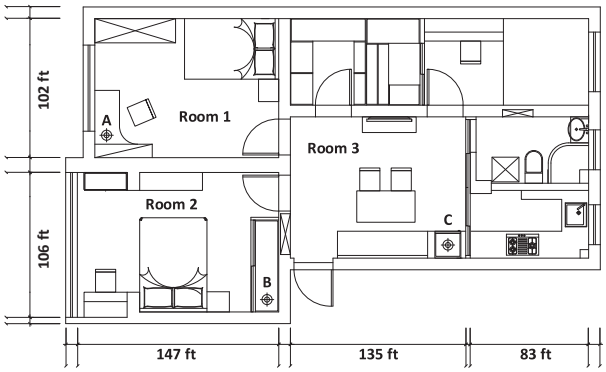



**Figure 16: Floor plan of the apartment. We set up three cameras in three rooms on spot A, B and C respectively.**

## A.2

**Table 5: The default settings of the 20 cameras and the setup we use for trace collection. We collect traces for the 20 cameras under two different resolutions besides its default resolution.**

| No. | Brand | Model | SOC Chip | Compression | Default Resolution | Resolution | | | Scenario | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | High | Mid | Low | Home | Lab |
| 1 | Dahua | LeChange TC1 | Hisilicon 3518 | H.264, AAC | 720p | ✓ | - | ✓ | ✓ | ✓ |
| 2 | Dahua | LeChange TC5 | Hisilicon 3518 | H.265, AAC | 1080p | ✓ | - | ✓ | ✓ | ✓ |
| 3 | Dahua | LeChange TC6C | Hisilicon 3518 | H.264, AAC | 720p | ✓ | - | ✓ | ✓ | ✓ |
| 4 | Dahua | LeChange TC7C | Hisilicon 3518 | H.264, AAC | 720p | ✓ | - | ✓ | ✓ | ✓ |
| 5 | Hikvision | Ezviz C2C | Hisilicon 3518 | H.264, AAC | 720p | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | Hikvision | Ezviz C2mini | Hisilicon 3518 | H.264, AAC | 720p | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | Yi | 1 | Hisilicon 3518 | H.264, AAC | 720p | ✓ | - | ✓ | ✓ | ✓ |
| 8 | Yi | 2 | Ambarella S2LM | H.264/ MJPEG, AAC | 1080p | ✓ | - | ✓ | ✓ | ✓ |
| 9 | Xiaomi | iSC5 | Sonix 9836 | H.264, AAC | 1080p | ✓ | - | ✓ | ✓ | ✓ |
| 10 | Xiaomi | SXJ01ZM | Grain 8136 | H.264, AAC | 1080p | ✓ | - | ✓ | ✓ | ✓ |
| 11 | 360 | D600 | Sonix 9836 | H.264, Opus | 720p | ✓ | ✓ | ✓ | ✓ | ✓ |
| 12 | 360 | D606 | Hisilicon 3518 | H.264, Opus | 1080p | ✓ | ✓ | ✓ | ✓ | ✓ |
| 13 | TP-LINK | TL-IPC20-2.8 | Unknown | H.264, - | 720p | ✓ | - | ✓ | ✓ | ✓ |
| 14 | TP-LINK | TL-IPC10A | Unknown | H.264, AAC | 720p | ✓ | - | ✓ | ✓ | ✓ |
| 15 | Nest | Cam Indoor | Ambarella S2LM | H.264, AAC | 1080p | ×* | ✓ | ✓ | ✓ | ✓ |
| 16 | Amcrest | IP2M-841W | Ambarella S2LM | H.264, AAC | 1080p | ✓ | - | - | ✓ | ✓ |
| 17 | D-Link | DCS-820L | Unknown | H.264/ JPEG, AAC | 1080p | ✓ | - | - | ✓ | ✓ |
| 18 | Lenovo | G2 Mini | Hisilicon 3518 | H.264/ JPEG, AAC | 720p | ✓ | ✓ | ✓ | ✓ | ✓ |
| 19 | Haier | HC6700 | Hisilicon 3518 | H.265/ JPEG, AAC | 720p | ✓ | ✓ | ✓ | ✓ | ✓ |
| 20 | Yoosee | KP-01 | Sonix 9836 | H.264, AAC | 720p | ✓ | ✓ | ✓ | ✓ | ✓ |
| 21 | No camera | - | - | - | - | - | - | - | ✓ | ✓ |

✓: Collected ×* : Not collected due to the limit of bandwidth - : Not supported