# When Free Tier Becomes Free to Enter: A Non-Intrusive Way to Identify Security Cameras with no Cloud Subscription


Yan He*
heyan@ou.edu
The University of Oklahoma
Norman, OK, USA

Qiuye He*
qiuye.he@ou.edu
The University of Oklahoma
Norman, OK, USA

Song Fang†
songf@ou.edu
The University of Oklahoma
Norman, OK, USA

Yao Liu
yliu21@usf.edu
University of South Florida
Tampa, FL, USA



## ABSTRACT

Wireless security cameras may deter intruders. Accompanying the hardware, consumers may pay recurring monthly fees for recording videos to the cloud, or use the free tier offering motion alerts and sometimes live streams via the camera app. Many users may purchase the hardware without buying the subscription to save money, which inherently reduces their efficacy. We discover that the wireless traffic generated by a camera responding to stimulating motion may disclose whether or not video is being streamed. A malicious user such as a burglar may use such knowledge to target homes with a "weak camera" that does not upload video or turn on live view mode. In such cases, criminal activities would not be recorded though they are performed within the monitoring area of the camera. Accordingly, we describe a novel technique called *WeakCamID* that creates motion stimuli and sniffs resultant wireless traffic to infer the camera state. We perform a survey involving a total of 220 users, finding that all users think cameras have a consistent security guarantee regardless of the subscription status. Our discovery breaks such "common sense". We implement *WeakCamID* in a mobile app and experiment with 11 popular wireless cameras to show that *WeakCamID* can identify weak cameras with a mean accuracy of around 95% and within less than 19 seconds.



## CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**.

## KEYWORDS

Security camera; traffic sniffing; paid subscription; motion detection

**ACM Reference Format:**
Yan He, Qiuye He, Song Fang, and Yao Liu. 2023. When Free Tier Becomes Free to Enter: A Non-Intrusive Way to Identify Security Cameras with no Cloud Subscription. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623083


*The first two authors contributed equally to this paper.
†Corresponding Author.




*CCS '23, November 26–30, 2023, Copenhagen, Denmark*



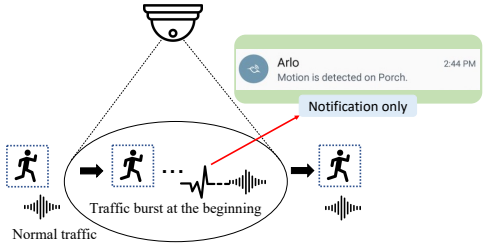
ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3623083


## 1 INTRODUCTION

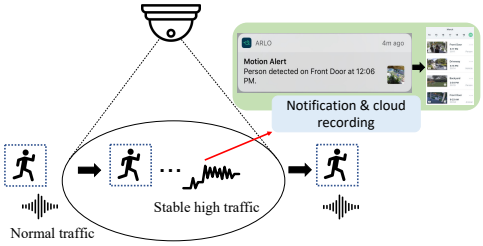
Wireless security cameras are increasingly affordable, easy to install, and multi-functional (e.g., instantly alerting the camera owner to the presence of intruders and enabling the owner to converse with visitors). They have become an essential tool in a property protection kit, as they can help with the intrusion detection and the recovery of stolen items via video footage [75]. In 2019, there were an estimated 1.12 million burglaries (i.e., the unlawful entry of a structure to commit a felony or theft) in the US, and victims suffered an estimated 3.0 billion US dollars in property losses, according to a report released by the Federal Bureau of Investigation (FBI) [29]. Meanwhile, the COVID-19 pandemic, which has changed how we interact with the outside world, has also expedited the integration of wireless security cameras into home security, since homeowners can easily use them to check and communicate with delivery persons without coming into physical contact with them.

Beyond the initial investment to buy the hardware, most wireless camera manufacturers offer consumers a paid plan to obtain more services, and offer limited functions for free users, so that users are motivated to pay for more services [43]. Usually, wireless cameras are equipped with motion sensors or microphones for enhanced protection, so that once motion or sound is detected, the camera is activated. The following behavior after the activation, however, often depends on whether the camera has an active subscription plan, which charges for services such as recording or cloud storage. For example, the latest Arlo cameras (e.g., Arlo Pro 3/4) do not actively record when events happen within their fields of view without a paid plan, and users can only get event alerts or manually stream footage to their smartphones via the Arlo app [34].

Cameras without paid subscriptions may suffer privacy issues, which have not been exploited before. We conduct a survey involving 220 participants: 213 of them believe the unpaid cameras can be used securely without privacy leakage; all users think the manufacturer guarantees that the system security is consistent across devices regardless of their subscription statuses. It is widely known that how owners safeguard their properties plays an important role when burglars select targets. A previous study reveals that in a panel made up of participants convicted of burglary, 13 out of 15 stated that they were not deterred by cameras that they believed




**(a) Camera with no subscription.**




**(b) Camera with a paid subscription.**

**Figure 1: Cameras with and without a subscription.**

were not constantly monitored [28]. Similarly, if the knowledge is available, a burglar or other malicious user will likely first target properties whose cameras do not actively record and save videos.

We give an example to illustrate the behavioral difference between cameras with and without an active subscription when they are triggered by a continuous movement. Wireless cameras are usually in sleep/standby mode until motion is detected. Figure 1a shows how a wireless camera (Arlo Pro 3) without a subscription only sends a push notification about the event and then quickly returns to sleep mode. The network traffic correspondingly exhibits a short burst when an individual enters the motion detection range of the camera, and returns to normal after that. In contrast, Figure 1b depicts the case when the camera has an active subscription. In addition to sending a push notification, the camera also records and uploads video to the cloud, which the owner can access later, until motion ceases within the detection range. The push notification content sent by a camera with a subscription is also richer than that sent by a camera without a subscription, including a still image from the event. Finally, the camera reverts to sleep mode. Corresponding to this activity, there appears a long traffic burst lasting from the moment the person enters to when they leave the motion detection range. Both cases have distinguishably different wireless traffic patterns, which can be in turn utilized to infer the camera's subscription status.

In contrast to this immediate recording and upload, owners receiving push notifications via smartphones may or may not respond quickly or at all. As motion alerts are sometimes inaccurate or irrelevant, some users may disable notifications or become desensitized to them [37]. Generally, if they turn on the live view mode, the resultant live streaming will make the camera generate more traffic until the live view mode is turned off. Such a traffic burst may be confused with the one caused by the automatic cloud recording of a camera with a subscription. Nevertheless, a human cannot initiate the video processing module instantly when a push notification is received, as there are two non-negligible delays: (1) the user needs to first access the phone and tap the camera app, depending on the user's response time; and (2) the app needs time to be launched. However, a subscribed camera can almost instantly begin cloud recording once it detects motion and sends the push notification. Consequently, the live mode and cloud recordings have different impacts on the traffic generation of the camera, and the resultant traffic pattern dissimilarity provides a clue to distinguish them.

We propose *WeakCamID*, a generalized framework to distinguish the state of a wireless camera, that is, whether or not the camera has a subscription and if its live view mode is turned on. Such an inference attack is non-trivial due to the following reasons. (1) The attacker cannot directly extract the traffic flow for the target camera, as it has neither control over the environment nor access to the WiFi network that the camera is connected to. The environment also likely contains many wireless devices and has a mixture of all flows from various devices, such as laptops, smartphones, or tablets. (2) To the best of our knowledge, previous extensive research efforts (e.g., [25, 38, 46, 50, 71, 78]) in detecting/localizing wireless cameras all assume that the cameras have the capability of recording the motion events without further distinction regarding the camera's subscription status. The existing course-grained traffic pattern identification methods can identify the existence of a camera while we cannot apply them to infer the subscription status. A novel and fine-grained traffic pattern technique is thus required. (3) Live streaming, relying on user operation, may generate comparable wireless traffic to cloud recording. Existing research either considers continuous/confirmed live streaming (e.g., [25, 39, 50, 64, 71]) or simply ignores it (e.g., [38, 48, 65]). To determine whether live streaming is on or off, human behavior then needs to be considered. (4) Traditional all-channel WiFi sniffing often requires rooted Android phones, limiting its practicality. Such an engineering challenge should also be overcome.

Almost all commodity wireless devices utilize 802.11 wireless protocols, and their use has an inherent weakness: exposure of link-layer Media Access Control (MAC) addresses [52]. A passive adversary within the radio range of a wireless camera can extract its MAC address, which tells the information of its device manufacturer via the beginning three most significant MAC bytes, i.e., the Organizationally Unique Identifier (OUI). *WeakCamID* first utilizes the motion-traffic correlation phenomenon to determine possible candidates of traffic flows belonging to a target camera, and then cross-references OUIs with publicly available manufacturer information to figure out the final candidate. By feeding motion stimuli to the camera and sniffing resultant traffic that varies with the camera state, *WeakCamID* builds a model to correlate motion-induced traffic with camera state. Such a model can be then used to map observed unlabelled traffic flows into corresponding camera states.

With *WeakCamID*, we discover that it could be counterproductive to install non-subscription wireless security cameras. The service differentiation between paying and non-paying users does not just create inequality in degrees of protection. The function restriction for non-paying users in fact introduces a serious vulnerability, which an adversary could take advantage of to identify properties with "weak" cameras. With a non-subscription camera, if the property owner does not view live streams in time, the events occurring in the area monitored by the camera will not be recorded. In such scenarios, as eye-witness descriptions and filmed recordings are not

available, malicious users may perform inappropriate or criminal activities without worrying about being identified or leaving traces. We summarize our main contributions as follows:

- We point out the vulnerability of current wireless security cameras in differentiating services for paying and non-paying users, and develop the first practical tool to successfully infer different camera states.
- We systematically explore the correlation between motion stimuli with the resultant wireless traffic generated by cameras with varying subscription statuses.
- We show that *WeakCamID* can detect user response to motion alerts by distinguishing high traffic volumes caused by cloud recording and live streaming.
- We develop an app for validating the effectiveness and efficiency of *WeakCamID*. Experimental results show that *WeakCamID* can attain a mean success rate of 95% to infer camera states within 19 seconds.

## 2 BACKGROUND

**Wireless Security Cameras:** According to the latest report published by Allied Market Research, the global wireless security camera market size was valued at 5.91 billion in 2020 and is expected to reach 18.3 billion by 2030, expanding at a mean annual growth rate of 12.4% [45]. Wireless security cameras can act as behavioral deterrents to inhibit trespassing, intrusion, theft, vandalism, and related forms of harmful activity [37], and also document what happened as evidence, especially incidents of crimes (e.g., burglary, vehicle prowl, or home invasion) [33]. Wireless security cameras are usually triggered by motion, and a few cameras (with a built-in microphone or audio line-in), can also be triggered by sound. Sound-triggered systems, however, often suffer from high false alarms via car engine sounds, barking dogs, or other noises. In this study, we focus on inferring the states of motion-activated wireless cameras.

Non-subscription cameras often have limited features such as live video streaming and motion notifications, rather than cloud recordings which allow users to save captured videos on their online storage. Almost all wireless camera companies offer video-storage plans for customers to purchase. Compared with traditional one-time revenue from the hardware sale, recurring revenue from the sale of subscription plans is predictable, sustainable, and potentially more profitable. The subscription cost normally varies with the video resolution and the number of cameras supported. For example, Arlo offers two options for multiple cameras at a single home and on the same account, $9.99 and $14.99 per month enabling recording in up to 2K and 4K video resolution, respectively [19]. The seemingly small monthly charges, however, add up, and may result in more personal debt [63]. They may thus inevitably impose a financial burden on many users. According to Arlo, it has 5.82 million registered accounts and 877 thousand paid accounts, as of January 2022 [15], meaning that as high as 85% of users still use cameras without a subscription. In this paper, we point out that using such unpaid cameras is not as safe as paid versions and may cause significant privacy concerns.

**Motion Detection:** Wireless security cameras are often battery-powered, and most of them (e.g., Blink Outdoor [5]) employ motion sensors to conserve battery, by only waking up when motion is detected. There are different types of motion sensors, including Passive Infrared (PIR), ultrasonic, microwave, tomographic, and combined types. Of these, PIR sensors are most prevalent, being small in size, cheap, and highly sensitive to motion. These are made of a pyroelectric film material sensitive to radiated heat power fluctuation [49]. This material generates electric signals when exposed to heat in the form of infrared radiation. Thus, PIR sensors can detect the presence of humans or other warm-blooded living beings from the radiation of their body heat [57], meaning that they can work even in the dark.

## 3 ATTACK MODEL AND ASSUMPTIONS

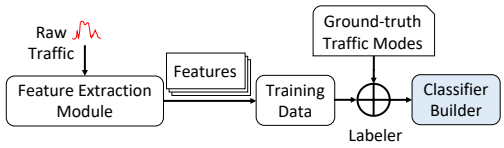
We consider a general scenario, where a wireless security camera is deployed to monitor a target area with an unknown subscription status. Once motion is detected in the camera's range, a camera without a subscription only sends a push notification to the owner while a camera with an active subscription also enables cloud recording. After receiving push notifications about motion events, the owner may or may not turn on the live view mode of the camera through the camera app. The adversary aims to employ *WeakCamID* to infer the camera state, i.e., whether the camera has a subscription and whether the live stream is opened.

We assume that the adversary has the capability to sniff wireless traffic and perform some probing motion in the target area. To avoid being exposed, the adversary can actively employ a helper or some moving robot (e.g., drone/robot/car) that emits heat to introduce movement. Additionally, she can passively monitor camera activity and rely on others triggering motion sensing. Note that it is not necessary for the attacker to know the exact location of the camera. In a common scenario, people often make wireless security cameras visible with the hope of deterring malicious users. For example, owners may post signs and stickers to warn that there is a security camera present. Such visibility, however, could help the adversary quickly determine the possible motion detection range of the camera. On the contrary, when the camera is hidden, *WeakCamID* still works, as it can recognize the existence of wireless cameras by analyzing motion-induced wireless traffic. After the attacker confirms that the camera has no subscription and no live video is turned on, she may bypass it to perform further malicious activities (e.g., burglary and intrusion) without being recorded.

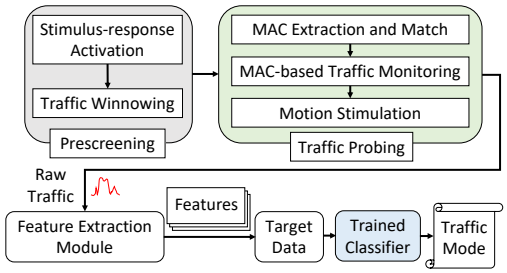
## 4 CAMERA STATE INFERENCE

### 4.1 Framework Overview

*WeakCamID* performs a two-phase process to infer camera states from observed wireless traffic: the *training* and *inference* phases. Figure 2 plots an overview of this process. Figure 2a depicts the offline training phase, in which a traffic classifier is built with the motion-induced traffic data collected from sample wireless security cameras and their corresponding states. The inference phase then uses the trained traffic classifier to recognize new traffic flows, as shown in Figure 2b. As aforementioned, there may be a variety of devices sending out wireless traffic in a new environment. Thus, in the inference phase, the adversary must first identify the traffic flow associated with the specific target camera. Toward the goals, *WeakCamID* introduces two important phases before extracting traffic features, which are *traffic prescreening* and *traffic probing*.




**(a) Training Phase: Build a traffic classifier**




**(b) Inference Phase: Recognize traffic modes**

**Figure 2: Overview of *WeakCamID*.**

The first phase coarsely determines the wireless traffic flow associated with the target camera. When provoking motion within a target area, if there is a wireless camera monitoring this area, a corresponding wireless traffic burst will be immediately observed as the triggered camera generates traffic. The burst can be short or long depending on the camera's subscription status. The traffic flow exhibiting such a distinguishable pattern is regarded as a candidate.

In the second phase, we further eliminate the inference of other wireless devices which coincidently exhibit traffic patterns similar to the target camera in the first phase. We inspect the OUI in the MAC address embedded in each candidate traffic flow to sort out the camera-generated traffic flow and then monitor the surviving traffic. We then feed manipulated motion to the camera in order to extract features from the resultant traffic.

## 4.2 Training Phase

Our model is trained via data collection, feature extraction, state labeling, and traffic classifier building steps.

**Data Collection:** To capture raw wireless packets originating from the camera, we should know the channels that the camera operates on. The wireless network interface card (NIC) of a traffic sniffing device needs to be in monitor mode to listen to all the wireless traffic nearby. Generally, the monitor mode is disabled, and the default/normal mode of an NIC is managed mode, which makes the device only capture packets with its own MAC address as the destination MAC and discard other packets.

*Sniffing with Laptop:* An intuitive way to achieve traffic sniffing is to use network sniffing software such as *WireShark* [31], but this method requires the sniffer to be able to access the same WiFi network as the target camera. The network is often secured with a password, which is unknown to the sniffer. Alternatively, if the laptop has a compatible wireless network adapter (e.g., Intel 622AN [41]) that supports monitor mode, the Aircrack toolkit [11], which is open source, can be then utilized to enable monitor mode.

*Sniffing with Android Phone:* A laptop may be bulky for a user to carry. To enable monitor mode on an Android phone, we need first to perform kernel live patching corresponding to the phone model [67] and then employ Airmon-ng tool [4], which is included in the Aircrack-ng package. For example, we enable the monitor mode on a Nexus 5 Android phone by using Nexmon [3] to patch the phone's kernel and then can run *WeakCamID* on the rooted Nexus 5. Finally, the collected traffic data are loaded into the SQLite database [8] for feature selection.

**Feature Extraction:** Different camera states usually lead to different spatiotemporal patterns in collected wireless traffic data. We then extract relevant features to construct our "feature vector", and use it to train the model.

Normally, when we continuously feed motion stimuli to a wireless camera for a period of time (e.g. 10 seconds), the camera experiences multiple phases. First, the camera sends an event notification to the owner, causing the first traffic burst. Second, the camera may or may not start recording the activity, depending on whether the camera has a subscription (i.e., cloud recording capability). If the camera has a subscription, it will immediately start to record the activity and upload the captured video to the cloud backend. As a result, another traffic burst will be generated, which is often larger than the one appearing in the first phase. However, if the camera has no subscription, the camera will not record the activity, and the traffic volume will soon become zero after sending the event notification to the property owner. Finally, the traffic flow varies according to the action taken by the property owner after she or he receives the event notification, i.e., whether to enable the live video by opening the app associated with the camera. Specifically, if the owner quickly presses the push notification after receiving it and then opens the camera app to watch the live streaming video from the camera, the traffic throughput (i.e., the rate at which the wireless camera generates packets) will abruptly increase again. Accordingly, we refer to these three phases as *event notification*, *camera response*, and *user operation*, respectively. We then extract features unique to the camera state to characterize each phase.

*Phase I - Event Notification*: When motion is detected, a camera with a paid subscription (e.g., Arlo camera [16]) often generates a rich push notification, attaching a thumbnail image of the event to the event notification, while a non-subscription camera only sends the basic event notification. Thus, the corresponding peaks of the instant traffic throughput will differ. We define the period of this phase as from the beginning of the first observable traffic peak to the next one for paid cameras (as they start to record and upload the recorded video to the cloud), and back to 0 for unpaid cameras. Accordingly, we record two features, the peak traffic throughput $T_1^p$, and the mean value $\bar{T}_1$ for the period of this phase.

*Phase II - Camera Response*: Paid cameras also perform cloud recording except for event notifications, while unpaid cameras do not and stay silent without being triggered. We regard the point from which the traffic abruptly increases or decreases as the ending point of the second phase for paid cameras; the abrupt traffic change is determined by whether the live streaming is enabled or not. A user may not always respond to a notification, e.g., when they are busy or sleeping, while if the user chooses to turn on live video streaming, it needs some time, and this delay includes two parts: (1) the interval between the time when the user receives the event notification and the time when the user opens the app, and (2) the time that the app needs to load. Empirically, this delay is at least 3 seconds. For unpaid cameras, we just consider the period

of the second phase as 3 seconds, and such a period is enough to characterize how unpaid cameras respond to motion after event notification. Similarly, we record the peak traffic throughput $T_2^p$ and the mean traffic throughput value $\bar{T}_2$ in the second phase. Obviously, for unpaid cameras, we have $T_2^p = \bar{T}_2 \approx 0$.

*Phase III - User Operation*: This phase happens only when the user turns on live view mode. For paid cameras in normal mode (i.e., no live view is enabled), the generated traffic will be nearly stable until the motion ends, while in live view mode, such traffic becomes a combination of recording and streaming traffic and would thus be higher. Also, after the motion ends, there will be only streaming traffic until the user turns the live view mode off. For unpaid cameras, no recording happens in normal mode and thus no traffic is generated for it, while they are re-triggered to generate the streaming traffic in live view mode and revert to standby mode once the user closes the live view mode. We specify the third phase starting from when traffic burst appears after the second phase until when the camera enters standby mode. Likewise, we mark the corresponding peak traffic throughput $T_3^p$ and the mean value $\bar{T}_3$ for this phase. If no live view is enabled, we set $T_3^p = \bar{T}_3 \approx 0$.

A camera's state has four possibilities, live view mode and normal mode (i.e., when live view is unopened), with and without a subscription accordingly. We refer to the four states as *Paid - Live View*, *Paid - Normal*, *Unpaid - Live View*, and *Unpaid - Normal*. The final feature-vector corresponding to the resultant wireless traffic when introducing motion stimuli to a wireless camera with one state ($S_i, i \in \{1, 2, 3, 4\}$) thus can be denoted with a 6-element vector, i.e., $FV(S_i) = [T_1^p, \bar{T}_1, T_2^p, \bar{T}_2, T_3^p, \bar{T}_3]$.

*Impact of User Behavior*: We do not assume deterministic user behaviors. The owner can make decisions arbitrarily. The success rate thus does not depend on user behavior, and *WeakCamID* works regardless of whether users respond to notifications. If the live view is off, the inference result would be 'paid-normal' or 'unpaid-normal'; otherwise, it is 'paid-live view' or 'unpaid-live view'.

**State Labeling:** Once the feature vectors are extracted from the sniffed wireless traffic, *WeakCamID* creates a training set by labeling camera states. The labeled feature vectors can be used to train the classifier in the next step.

*Dataset Splitting:* We have a dataset containing feature vectors coming from 11 different cameras that we examine for training. We perform different durations of motion from 2 to 16 seconds with increments of 2. For every motion length, we collect 70 corresponding traffic flows for each camera state of every camera, enabling us to obtain high inference accuracy. Thus, the built dataset has $11 \times 8 \times 70 \times 4 = 24,640$ feature vectors in total. We apply the common 80/20 split for training and test sets.

**Traffic Classifier Building:** The last step of the training phase consists of training a model that will be used during the inference phase to infer the camera state accordingly.

We choose a supervised learning (classification) technique over traditional statistical methods for two reasons. First, the wireless traffic flows generated by cameras with different brands/models responding to motion stimuli may be different, as different manufacturers may have proprietary configurations. For example, the patterns of the traffic generated by Ring and Arlo cameras for sending out push notifications are different; a Ring camera only sends a text notification, while an Arlo one also includes a thumbnail event image along with the text notification. It is thus difficult to build a statistical model in the form of mathematical equations to directly correlate the selected features with the camera state. Second, pre-configured video resolution for cameras may also vary across different or even the same brands of cameras. For example, the default resolutions of Arlo Pro and Arlo Ultra Camera Series are 1440p (2560×1440) and 2160p (3840×2160), respectively [17], while all Ring cameras share one same video resolution of 1080p (1920×1080) [7]. Such configuration variations can cause traditional statistical methods to generate inaccurate results over time as the data set changes. This phenomenon further increases the hurdle for us to construct a universal statistical model. Machine learning methods, however, can analyze amounts of data quickly and identify patterns that are not visible to traditional statistical methods. They can also automatically adapt to changes in the data set, ensuring that the inference can always achieve high accuracy.

With the aforementioned six parameters, we can utilize popular machine learning tools to build inference models, such as tree-based or Support Vector Machines (SVMs) [23]. Tree-based methods, e.g., decision trees (DTs) [62] and Random Forests (RFs) [40], build a tree-like structure for deciding cameras states according to the selected features, while SVMs find hyperplanes that best separate the traffic features into different domains (i.e., camera states). To build an optimal classifier, we implement and compare the following three algorithms in the scikit-learn environment [22]: DTs, RFs, and SVMs. There are four camera states, and we then use SVMs for multi-class classification. The approach we use is one-versus-one (or ovo).

*Classifier Selection:* Compared with the other two classifiers, we empirically find SVMs achieve better inference performance. Figure 3 presents the success rates for different classification algorithms applied to the test dataset. The success rate refers to the proportion of correct inference in all inference attempts. We have three key findings. First, the impact of motion duration and classifier algorithm for all four camera states are roughly consistent, and the overall success rates for cameras in the normal state are slightly higher than in the live view mode. Second, the success rates of all three algorithms increase with motion duration from 2 to 12 seconds and maintain relatively stable after the duration reaches 12 seconds. Particularly, when the motion duration is less than 8 seconds, all algorithms have success rates of less than 90%. This appears due to the lack of distinctive features in the traffic flows when the motion just lasts for a short time period. When the motion duration is 12 seconds or longer, all algorithms achieve success rates larger than 90%. In Section 5.3, we further evaluate the impact of motion length of no less than 8 seconds on the inference performance for varying cameras. Lastly, SVM shows the best performance among the three algorithms, and it can achieve success rates of higher than 97% for paying or non-paying cameras in the normal state when the motion duration is 12 seconds.

For each camera state, we also count the true positive, false positive, true negative, and false negative cases, referred to as $TP$, $FP$, $TN$, and $FN$. The corresponding success rate then equals $(TP+TN)/(TP+TN+FP+FN)$. Meanwhile, $Precision$ and $Recall$ of the model can be denoted as $TP/(TP+FP)$, $TP/(TP+FN)$. We further compute F1 score (i.e., $2/(Recall^{-1} + Precision^{-1})$), as shown in Figure 4. Similarly, we see that the SVM always achieves higher F1

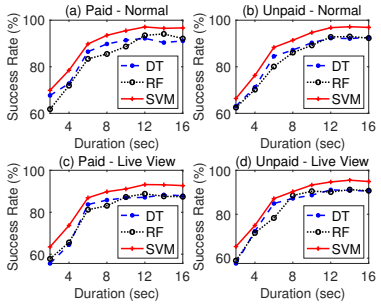



**Figure 3: Comparison of success rates.**

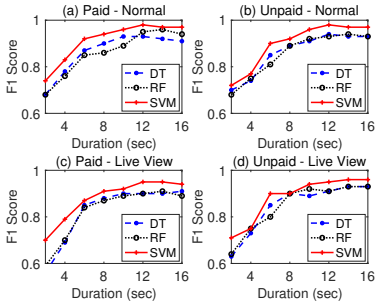



**Figure 4: Comparison of F1 scores.**

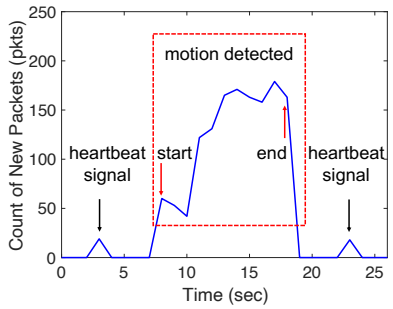



**Figure 5: Traffic volume variation.**

scores than the other two algorithms. For normal modes, the SVM obtains an F1 score of as high as 0.98, indicating its outstanding performance in both precision and recall.

## 4.3 Inference Phase

In the inference phase, the adversary needs to first determine that the target camera is a wireless motion-activated camera via two important steps, traffic prescreening and traffic probing. The following processes are performed much in the same way as the training phase has, by attempting to achieve camera state inference through data collection, feature extraction, and traffic classification.

*4.3.1 Traffic Prescreening.* Over the air, there may exist diverse wireless traffic flows generated by a myriad of Internet-of-Things (IoT) devices or applications (such as smart TVs and digital voice assistants). We thus need to first distinguish the traffic flow of the target camera from traffic flows generated by non-camera devices and other wireless cameras deployed in the environment. We propose to generate motion (e.g., walking) within the camera's monitoring area to stimulate it, and then use the resultant wireless traffic to narrow down the candidates for the target traffic flow.

**Stimulus-response Activation:** Most wireless cameras are powered by rechargeable lithium-ion batteries, either built-in or removable. They normally sit in sleep/standby mode to save power consumption, and come awake when (1) motion is detected or (2) the camera is manually turned on to live view. In standby mode, the camera usually just generates a "heartbeat signal" with a small size periodically (i.e., in order of seconds) to notify normal operation of the camera and synchronize with the base station or router.

Upon activation, the camera then sends a push notification of the motion event. If the camera has an active subscription, it also starts to record until motion stops and immediately uploads the video to the cloud for secure storage in the owner's library so that the owner can access them anytime; otherwise, if the camera has no subscription, only a push notification will be sent while no recording is initiated. Accordingly, abnormally high wireless traffic (indicating the push notification) will be generated regardless of the subscription status, and the traffic volume will soon become higher (as recording/uploading starts) for cameras with active subscriptions while decreasing to none (when heartbeat signals are ignored) for cameras with no subscription.

Therefore, to observe wireless traffic generated by the target camera, an adversary can feed the camera with activation signals by performing motion in the motion detection range of the camera.

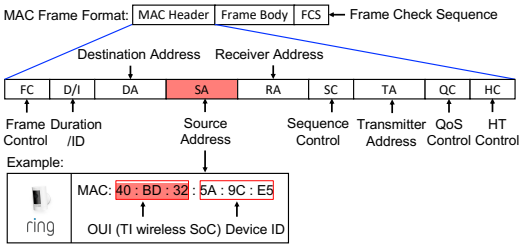



**Figure 6: MAC frame format and a source address example.**

Figure 5 depicts the traffic flow generated by a wireless camera (Ring Stick Up Cam with an active subscription) when we walk inside the motion detection range of the camera (8 ∼ 18 sec). We observe that when the camera is in sleep mode, it only sends out a heartbeat signal of a small size. When the motion event is detected, the newly generated traffic volume suddenly increases immediately for sending a push notification. Next, as the camera starts to record to the cloud, a larger traffic volume appears until the motion in the motion detection range of the camera disappears. Without motion stimulus, the camera comes back to sleep mode.

**Traffic Winnowing:** For a wireless camera in sleep mode (i.e., when there is no live view or video recording), the corresponding wireless microcontroller unit (MCU), such as TI (Texas Instruments) CC3220S [1] for a Ring Stick Up Cam, consumes low power and only listens for any trigger source. The motion sensor is integrated with the wireless MCU. Once it detects motion, it toggles the General Purpose Input/Output (GPIO) and generates an interrupt, which wakes up the camera to send a push notification and start cloud recording (if the camera has an active subscription). Consequently, the wireless traffic generated by a wireless camera has a strong correlation with the motion performed in the motion detection range of the camera regardless of the subscription status of the camera. Specifically, when a camera is wakened up by motion, a burst of wireless traffic can be immediately observed.

The distinguishable traffic pattern of the camera enables the adversary to winnow out irrelevant traffic flows, which do not show bursts according to the appearance of the artificial motion. If a monitored wireless traffic flow suddenly jumps with the motion being performed and plummets as the motion stops, we then mark it as a candidate for the traffic flow of the target camera. As the environment may have multiple motion-activated devices including the target camera, one or multiple candidates may be identified.

*4.3.2 Traffic Probing.* It is essential to determine precisely which traffic flow belongs to the target camera before collecting its traffic features. We utilize the devices' MAC addresses to pinpoint the traffic flow associated with the target camera from the obtained traffic candidates in the previous step. After that, we set up a listener to monitor the traffic transmitted from the target camera and observe the traffic change on this channel when provoking the camera with manipulated environmental motion.

**MAC Extraction and Match:** A MAC address is a unique identifier assigned to a Network Interface Controller (NIC) for every networked device. It consists of 48 bits that are typically represented as 6 pairs of hexadecimal digits separated by colons or dashes. The first half is the Organizationally Unique Identifier (OUI), indicating a manufacturer or vendor; the second half refers to the device ID.

As IEEE 802.11 wireless communication (i.e., WiFi) employs security protocols such as WEP, WPA, WPA2, and WPA3 [66], the recorded videos are encrypted in WiFi signals. A general IEEE 802.11 MAC frame consists of a header, body, and frame check sequence (FCS), as shown in Figure 6. The header holds information about the frame; the body carries data that needs to be transmitted; FCS is used for detecting errors during the transmission. However, the header is unencrypted during transmission and exposes the MAC of the device sending the traffic. For example, Ring Stick Up cameras utilize TI's chipset (i.e., CC3220S) for WiFi communication, and the OUI of their MACs starts with "40:BD:32", which indicates the SoC (System on Chip) from the manufacturer TI. The OUIs of different manufacturers are normally public [6]. We can thus build a dataset, referred to camera-tagged OUI database, containing OUIs of known vendors that manufacture wireless cameras.

*WeakCamID* first extracts the OUI in the MAC of each candidate for the traffic flow belonging to the target camera, and then checks the camera-tagged OUI dataset for a match of this OUI. If present, such a traffic flow is regarded as being generated by a wireless camera. Otherwise, it will be removed from the candidate list.

*Dealing With MAC Spoofing:* MAC addresses of NICs are hard coded in their circuit at the moment of manufacture. However, they can be changed via MAC randomization [30] or spoofing [42]. A camera may use a forged MAC with an OUI indicating a non-camera manufacturer for masquerading as a non-camera device, and similarly, a non-camera device may use a fake MAC with an OUI showing a camera manufacturer to pretend to be a camera. Since the payloads of raw WiFi packets are encrypted and the network of the target camera is inaccessible to the adversary, traditional traffic flow classification methods using a 5-tuple (source IP and port, destination IP and port, and IP protocol) or a 3-tuple (source IP, destination IP, and IP protocol) [53] do not apply. However, an attacker can launch the UUID-E (Universally Unique IDentifier-Enrollee) reversal attack [51, 76] to retrieve the original MACs for the devices with randomized or spoofed ones, as the UUID-E is derived from a device's original MAC and does not change with MAC. Alternatively, we utilize the wireless traffic pattern characteristics to uniquely identify camera devices.

The Systems on Chips (SoCs) are responsible for video/audio encoding and multimedia data transmission. Thus, the traffic patterns of a wireless camera highly depend on its SoC. However, the SoC choices are limited and most SoCs take largely identical operating flows, causing similar traffic patterns [25]. Particularly, wireless cameras follow universal standards to encode, encapsulate, and deliver video data to the cloud or users' devices. For example, Apple's HTTP Live Streaming (HLS), the most popular streaming format for the video industry according to an annual survey [21], requires that all videos must be encoded using H.264/AVC or HEVC/H.265 [13]. Accordingly, we train a Support Vector Machine (SVM) model by using the Scikit-learn libraries with Python 3.9, to distinguish traffic flows belonging to wireless cameras and non-camera devices.

An SVM classifier produces a hyperplane to best separate the input data into two classes. Since the cameras may or may not initiate video recording under different circumstances, the corresponding traffic patterns normally differ vastly. For such a multi-class case, we classify all traffic flows into three classes with the OVO (one-versus-one) approach [56]. For the cameras with no subscription and with live video mode turned off, they only generate traffic for push notifications and do not record video. We refer to such traffic as *Camera traffic 1*. For the cameras with subscriptions or with live video mode turned on, they also generate traffic for video recording, and we refer to the corresponding traffic as *Camera traffic 2*. We call the traffic generated by non-camera devices as *Other traffic*. We set a threshold according to the average data transmission rate of various wireless devices in the environment. For each traffic flow, we calculate its data transmission rate, as well as the difference between this rate and the threshold. Figure 7 depicts the outcome of running the created multiclass SVM on a data set containing 800 traffic flows coming from wireless cameras (in different modes and subscription statuses) and non-camera devices, demonstrating the success of identifying traffic flows generated by wireless cameras.

**MAC-specific Traffic Monitoring:** By setting up a packet monitor with existing tools, we can listen to the traffic coming from the device identified as a camera. Particularly, we detect if the traffic volume varies and record the count change of intercepted packets.

**Motion Stimulation:** The longer we perform motion in the motion detection range of the camera with a subscription, the more (cumulative) packets the camera may generate. We have the same observation for the live view duration. We deploy four different wireless cameras (including Arlo Pro 3, Blink Outdoor, SimpliSafe Cam, and Wyze Cam Outdoor v2) to monitor the activity in an area. We perform two groups of experiments to verify the impact of cloud recording and live view on camera traffic, respectively.

First, each camera has an active subscription and the live view mode is turned off. We collect the traffic packets generated by each camera and count the corresponding total amount of the transmitted packets when a user manually introduces motion within varying durations, as shown in Figure 8. Second, each camera has no active subscription while the live view mode is turned on for streaming the activity. Similarly, we collect the traffic packets generated by each camera and count the corresponding total amount of the transmitted packets when the live view lasts different durations, as shown in Figure 9. We see that different cameras present diverse total packet lengths changing with the motion or live view duration, due to various recording or live streaming mechanisms taken by different camera manufacturers. Overall, the obtained total packet count (denoted with $T$) consistently shows a nearly linear correlation with the duration of both the motion and the live view. For example, for every second, the corresponding packet counts for Arlo Pro 3 to record to the cloud and to stream live videos are

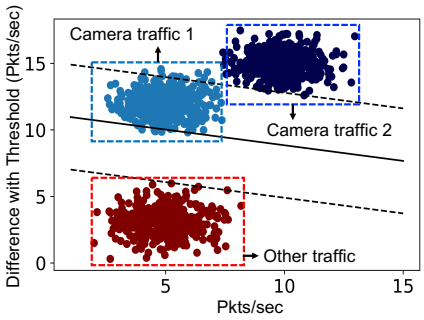



**Figure 7: SVM classification results.**

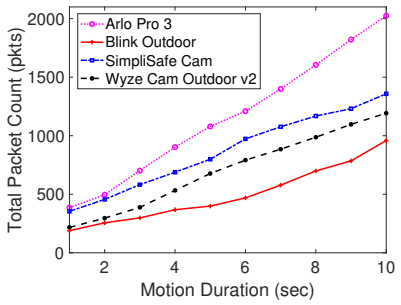



**Figure 8: Impact of motion duration.**

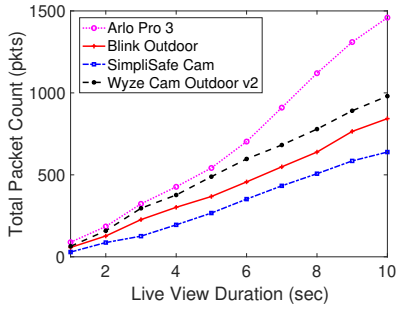



**Figure 9: Impact of live view duration.**

around 183 and 156, respectively. Accordingly, we consider a linear model to describe such as relationship, which is defined as follows,

$$T = c \cdot \Delta t + k, \tag{1}$$

where $k$ is constant, $\Delta t$ denotes either the motion or live view duration, and $c$ represents the traffic throughput, i.e., the rate at which the camera generates packets.

The model can be then utilized to determine whether the performed motion is still captured by the camera or whether the live view mode is still on. Specifically, if the observed total packet count and the motion or live view duration do not fit the linear model with a significant deviation, the cloud recording or live video streaming will be regarded as ended.

*4.3.3 Traffic Inference.* The process of traffic inference is defined much in the same way as the training phase by attempting to infer camera states via data collection, feature extraction, and traffic classification. After the traffic of the target camera responding to the motion stimuli is collected, the same features derived during training can be calculated. The obtained feature vector is then inputted into the built classifier, which outputs the camera state.

## 5 IMPLEMENTATION

We implement *WeakCamID* on commodity user devices. Appendix A shows the user interface (UI) of the developed mobile app.

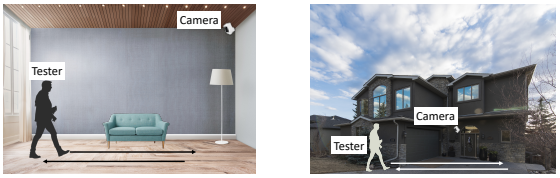
### 5.1 Experimental Setup

To achieve WiFi sniffing, existing studies usually use rooted Android phones (e.g., [9, 24, 38]) or certain models of laptops (e.g., Macbook Pro [70]), whose NICs can be set to monitor mode. It is burdensome to bring a laptop when performing *WeakCamID*. Also, smartphone vendors make it increasingly difficult to gain root access [72]. Meanwhile, apps (e.g., Google Pay) can detect root access and refuse to boot up if found [77, 79]. Instead, we design a new portable and low-cost external tool to enable WiFi sniffing, as described in Appendix B. This tool can connect with the app via Bluetooth Low Energy (BLE). Our design makes it possible to run *WeakCamID* on any factory default smartphone without rooting it.

The app first scans the possible MACs for wireless cameras. The adversary then performs motion to stimulate the camera. The app logs accelerometer readings for motion speed calculation. With observed traffic, the app outputs the current camera state and the consumed time, indicating completion of status determination. We test 11 most popular wireless cameras, as shown in Table 1. Such

**Table 1: The list of wireless security cameras we test.**

| Camera ID | Model | Cloud Recording (Unpaid) |
|---|---|---|
| 1 | Arlo Pro 3 | No |
| 2 | Arlo Pro 4 | No |
| 3 | Arlo Ultra 2 | No |
| 4 | Blink XT2 | No |
| 5 | Blink Outdoor | No |
| 6 | Ring Stick Up Cam | No |
| 7 | Ring Spotlight | No |
| 8 | Reolink Argus 2 | No |
| 9 | SimpliSafe Cam | No |
| 10 | Wyze Battery Cam Pro | No |
| 11 | Wyze Cam Outdoor v2 | No |




**(a) Indoor** **(b) Outdoor**

**Figure 10: Layout of the experimental environments.**

camera models are selected from major brands sold online on Amazon and BestBuy. Non-paying cameras only have basic functions (live video streaming and event notification) while paying ones offer cloud recording capability. Two typical scenarios are considered.

- *Outdoor:* The camera is mounted on the front outside wall (height: 10 ft; width: 17 ft) of a typical American single-family house to monitor the entryway into the house.
- *Indoor:* The camera is installed on the wall of a living room (of 372 sq. ft) to monitor the room.

Figure 10 shows the layout of the two environments, where the camera is deployed with its field of view not blocked by the wall or other obstacles and an adversary can thus feed motion stimuli to it.

**Evaluation Metrics:** We use the following three metrics.

- *Suceess rate:* this is the ratio between the number of successful camera state inference attempts and the total number of inference trials.
- *F1 score:* this is the harmonic mean of precision and recall, with its best value at 1 and worst score at 0.
- *Detection time:* this is the amount of time spent on obtaining the camera state in terms of the subscription plan and live streaming mode.

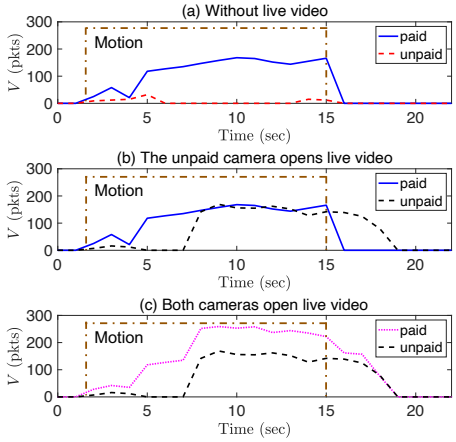



**Figure 11: Traffic volume $V$ (pkts) changes.**

## 5.2 Case Study

In this case, we let two Arlo Pro 3 cameras (one with and the other without a subscription) monitor the same area, as shown in Figure 10a. The user determines there exist wireless cameras monitoring the area, initiates motion in the area, and sniffs environmental wireless traffic. We test the following three situations.

**No Live Video is Streamed on Both Cameras:** When the user does not notice the motion notification (e.g., the phone is muted), no live stream will be opened. Figure 11 (a) shows the traffic flow generated by the two cameras. We observe a strong correlation between the traffic volume (i.e., count of newly generated packets) with the motion for the paid camera, i.e., the volume matches with the newly performed motion. However, for the unpaid camera, there is only a small amount of traffic at the beginning of the motion, corresponding to the motion notification. The paid camera not only sends a notification but also records to the cloud until the motion ends. Furthermore, we see that the traffic volume for a motion notification of the paid camera is larger than that of the unpaid one. This is because, with a subscription, the push notification information is richer and includes a thumbnail image from the recorded video [16], which is not available for the unpaid camera.

**Live Video is Streamed Only on the Unpaid Camera:** Just for the unpaid camera, we stream live video once receiving the motion notification. Figure 11 (b) compares the corresponding two traffic traces and we see clear differences. First, unlike the paid camera, which automatically records after being activated by the motion, the unpaid one re-generates the traffic burst only after the live view is turned on (at the $8^{th}$ second). To stream live video, we have to tap the notification or the app on the phone. Human reaction, tapping, and app login take time. There is thus an inevitable delay between detecting the motion and the start of the live video stream. Second, we may not end the live video exactly as the motion ends. We habitually watch until the motion ends and then close the app. Similarly, we need time to react and close the app. That's why we still observe traffic burst even after the motion ends for the unpaid camera. However, the paid camera ends recording (i.e., generating traffic bursts) precisely once the motion ends.

**Live Video is Streamed on Both Cameras:** Figure 11 (c) shows the traffic volume of both cameras streaming live videos. Unlike the unpaid camera, the paid one generates high traffic volume immediately once the motion is detected. Also, when the motion lasts and the live video is on, the traffic volume for the paid camera is apparently higher than that for the unpaid one. This is due to the fact that the paid camera streams live video and uploads the recorded video to the cloud at the same time, while the unpaid camera only streams live video. These results convincingly verify that the two cameras' traffic traces in this situation are still distinguishable.

By extracting features from the observed traffic flows, *WeakCamID* is able to successfully infer these camera states.

## 5.3 Impact of Motion Duration

Different durations of motion occurring within the motion detection area of the camera (with a subscription or in live view mode) may generate varying wireless traffic volumes. Accordingly, we vary the value of motion duration from 8 to 16 seconds, with increments of 2 seconds. For each value and camera state of every camera, we perform 10 trials and have $11\times4\times5\times10=2,200$ attempts in total.

Figure 12 shows the average success rates for different motion durations. We have the following observations. First, the success rate always maintains at a high level, i.e., ranging from 88% to 99%, regardless of motion duration and camera state. Second, with the duration increasing from 8 to 12 seconds, the success rate becomes larger. It then maintains a stable high value (above 94%) after the duration is longer than 12 seconds. Lastly, the unpaid camera in live view mode and the unpaid camera in normal mode consistently has the lowest and highest average success rates regardless of motion duration. This appears as the motion-induced traffic flows generated by unpaid cameras in live view and normal modes are the least and the most distinguishable, respectively. Figure 13 presents the F1 scores for all varying motion durations. We see that the F1 score is always above 0.9, again indicating high inference accuracy.

Figures 14 and 15 present the average success rates and F1 scores of all camera states for each camera with varying motion durations. We see that the success rates and F1 scores for all cameras are consistently high (with a minimum of 88% and 0.89), while C9 (SimpliSafe Cam) always has a higher success rate or F1 score than the rest. This appears because C9 uses differentiated video streaming quality for paid and unpaid cameras while others use the same quality for both types of cameras. The resolution of C9 is 1080p (1920×1080) with a subscription and decreases to just 480p (640×480) with no subscription. Such difference further enlarges the discrepancy between corresponding traffic volumes, facilitating camera state distinction. Also, we find for most cameras, the success rate or F1 score increases with the motion duration until the latter reaches 12 seconds, and remains relatively stable after that.

## 5.4 Impact of Movement Speed

The speed $v_m$ of motion occurring in the camera's detection range may affect its recording behavior. For example, if the speed is too slow, from the camera's perspective, the total motion may consist of multiple short activities. Compared with a quick motion which just triggers the camera once, such a slow one may cause the camera to be activated multiple times in a discontinuous way. We vary $v_m$ from 0.2 to 1.4 meters per second (m/s), with increments of 0.2. The app logs accelerometer readings for calculating the speed. For each

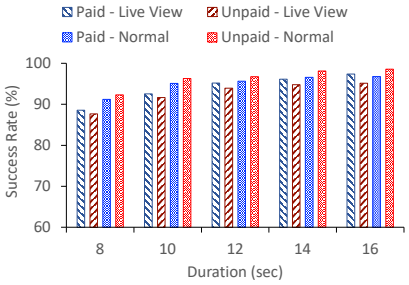



**Figure 12: Impact of motion duration.**

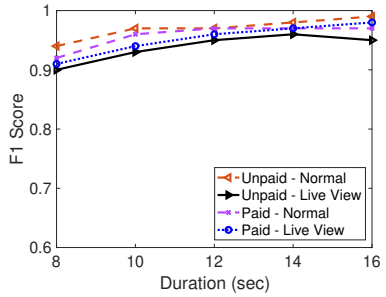



**Figure 13: F1 score vs. motion duration.**

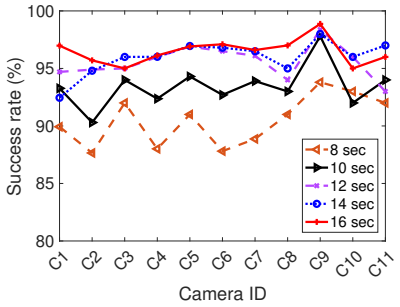



**Figure 14: Average success rates.**

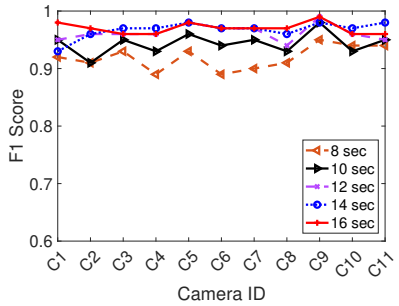



**Figure 15: Different cameras' F1 scores.**

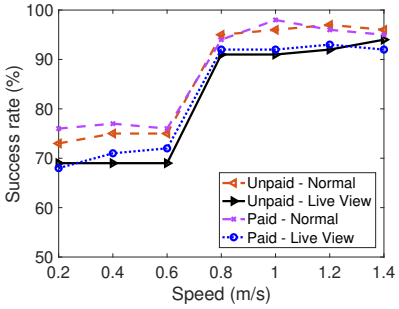



**Figure 16: Impact of movement speed.**

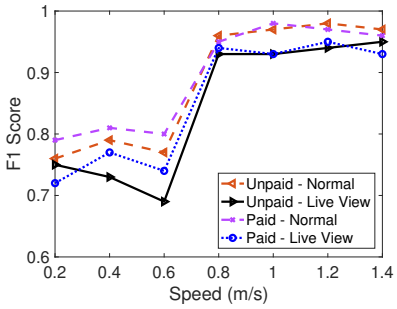



**Figure 17: F1 score vs. speed.**

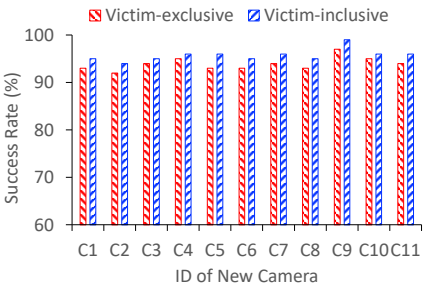



**Figure 18: Accuracy for new cameras.**

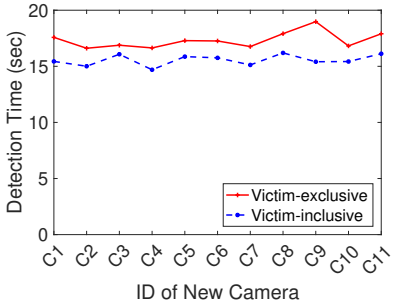



**Figure 19: Time spent for new cameras.**

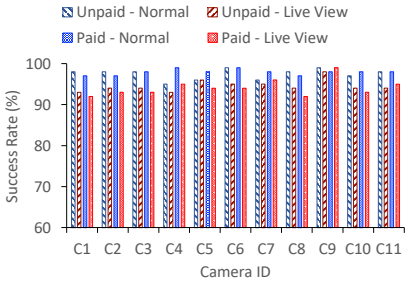



**Figure 20: Indoor success rates.**

$v_m$ and camera state, we perform 100 attempts of *WeakCamID* to infer the state of the camera (Ring Stick Up Cam).

Figure 16 illustrates the average success rates when $v_m$ varies. We observe that the success rate is below 77% when $v_m$ is no larger than 0.6 m/s. This is because the low speed may trigger the cameras multiple times and cause the camera to generate multiple notification alerts. The resultant traffic patterns become less discernible. Also, we see that once the walking speed reaches 0.8 m/s, the success rate can always be larger than 92%. Meanwhile, for the same speed, the corresponding success rates for normal mode are consistently higher than that for live view mode. Specifically, the average success rates for normal and live view modes are 96.0% and 92.5%. This appears due to the fact that the live view mode is controlled by the user, who may turn it on at a random time after receiving a motion alert, causing the traffic patterns associated with streaming live videos more diverse. Figure 17 plots the corresponding F1 scores, which always exceed 0.93 when the speed reaches 0.8 m/s. Also, the F1 scores for the normal mode are higher than that for the live view mode. The range for normal walking speed is 1.2 to 1.4 m/s for adults [32]. *WeakCamID* can thus achieve high accuracy without requiring an average user to change gait speed.

## 5.5 Impact of Previously Unknown Cameras

One concern is whether our model works for a new camera, whose brand/model is previously unknown. As aforementioned, most camera vendors take largely identical operating flows, causing their traffic variation quite consistent. *WeakCamID* can be thus applied to infer states of new cameras without retraining the model.

We specify one camera as the new camera and use the other ten (in Table 1) for training. Accordingly, we generate 11 traffic classifiers, referred to as *Victim-exclusive*. We then use each classifier to infer the state of the corresponding new camera 100 times, whose traffic data are not included in the training data set of this classifier. For comparison, we also investigate the performance of the classifier (called *Victim-inclusive*) that utilizes all 11 cameras for training, and use it to infer the state of every camera 100 times.

Figure 18 presents the comparison of the success rates for applying *Victim-inclusive* and *Victim-exclusive* classifiers. We see that the *Victim-inclusive* classifier always performs slightly better than

corresponding *Victim-exclusive* ones. Specifically, the mean success rate for all *Victim-exclusive* classifiers is 94.1% while the *Victim-inclusive* classifier achieves an average success rate of 95.8% across all cameras. Figure 19 compares the average detection time. We observe that for each victim camera, the detection time obtained from the *Victim-exclusive* classifier, ranging from 16.6 to 18.9 seconds, is always slightly longer than that obtained from the *Victim-inclusive* classifier. The small increase in detection time comes from requiring a longer time for the corresponding *Victim-exclusive* classifier to process the data. These results show that *WeakCamID* works for new cameras with a high probability and within a short period.

## 5.6 Overall Inference Performance

For each mode of every camera in Table 1, we perform 100 trials in each environment. Thus, we have $11 \times 4 \times 2 \times 100 = 8,800$ attempts in total. Figures 20 and 21 present the success rates for different cameras in the indoor and outdoor environments. We observe two major tendencies. First, the success rate is consistently high over different camera states and models, ranging from 92% to 99% and 90% to 99% for the indoor and outdoor environments, respectively. Particularly, for C9 (SimliSafe Cam) in both environments, our technique can detect all camera states with a success rate always above 98%. This again confirms that the recording quality differentiation strategy taken by C9 makes traffic flows more distinguishable. Second, a camera in normal mode can usually be detected with higher accuracy, especially when the camera has a subscription. This may be because cameras in live view mode generate higher traffic volume, causing the traffic flows to be misclassified more.

Figure 22 plots confusion matrices of the inference results. We see that *WeakCamID* has consistently high true positive rates (93.3% or above) and low false positive rates (below 2.9%). We compute the F1 scores, which are both 0.96 on average for the indoor and outdoor environments. Figure 23 plots the empirical cumulative distribution functions (CDFs) of the detection time $T_{indoor}$ and $T_{outdoor}$ under the indoor and outdoor environments. We see no apparent difference in detection time for both environments. $T_{indoor}$ and $T_{outdoor}$ are less than 17.6 and 17.5 seconds with probability 95.0%. These results convincingly demonstrate that *WeakCamID* can effectively and efficiently infer camera states.

Occasionally, multiple wireless cameras may simultaneously monitor a target area. We present the corresponding evaluation in Appendix 5.7, verifying *WeakCamID* still works in such a scenario.

## 5.7 Multiple-camera Scenario

In a multi-camera scenario, the adversary needs to infer the states of all cameras in order to determine whether there is a risk of being recorded when performing motion in the area. *WeakCamID* tracks the wireless traffic based on MAC addresses. It can monitor multiple camera-associated traffic flows at the same time. Different cameras have no interference with each other for camera state inference.

To evaluate *WeakCamID* on a multiple-camera scenario, we deploy varying numbers of cameras (1 to 6) in the testing room. We manually tweak the fields of view of the cameras and make them overlap partially. We perform *WeakCamID* for 50 attempts for each camera count. We randomly change the location and state of each camera at every attempt. As the inference error mainly comes from current wireless traffic patterns and varies with the duration of performed motion, it is thus quite consistent across coexisting cameras. We find that for each camera count from 2 to 6, *WeakCamID* always successfully infers the states of all cameras with a probability exceeding 94.5%, similar to what we achieve for inferring a single camera's state.

**Table 2: Detection time vs. camera count.**

| Camera count | Detection time (seconds) | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| 1 | 14.6 | 12.5 | 17.3 |
| 2 | 16.5 | 14.7 | 26.2 |
| 3 | 19.7 | 16.5 | 27.1 |
| 4 | 24.1 | 18.3 | 29.9 |
| 5 | 35.1 | 32.7 | 39.3 |
| 6 | 36.8 | 34.9 | 43.8 |

Table 2 presents the mean, minimum, and maximum detection time of successful trials for different numbers of cameras. We find that when the camera count is no more than 3, the detection time just slightly increases with the count in most cases. This is because the one-time motion (i.e., walking) triggers all cameras at the same time and *WeakCamID* can take advantage of it to infer the states of all cameras. Thus, an extra camera only adds data processing time. Also, we see that when the camera count exceeds 3, it is often not enough to walk one time to trigger all cameras, and we have to perform several movements instead. As a result, inferring the states of multiple cameras is equivalent to inferring the state of a single camera several times, and the detection time is almost proportional to the corresponding number of performing movements. Overall, *WeakCamID* can infer the states of up to 6 cameras within less than three-quarters of a minute, demonstrating the high efficiency of the proposed technique.

## 5.8 User Study

We recruited 11 volunteers (U1-U11; 5 self-identified as females and 6 as males) and asked each to perform *WeakCamID* to infer the state of a randomly selected camera deployed in the aforementioned indoor and outdoor environments.[1] Every participant performed 50 attempts for each camera state under each environment, and thus $50 \times 4 \times 2 = 400$ attempts in total. For each participant, the camera state appears in random order. Based on empirical results, we instructed the participants to introduce motion stimulation lasting 12 seconds or longer to achieve higher inference accuracy.

Figures 24 and 25 present the obtained success rates and F1 scores. We see that the average success rate and F1 score range from 91.0% to 95.0% and from 0.92 to 0.95, respectively. Also, regardless of the subscription status, the success rate or F1 score for the normal state is slightly higher than the live view mode. Specifically, the average success rates of the states *Unpaid - Normal* and *Paid - Normal* for all users are 95.0% and 94.9%, while that for the states *Unpaid - Live View* and *Paid - Live View* are just 90.8% and 91.1%. These results convincingly demonstrate that the performance of *WeakCamID* is robust to different camera states and users.

[1]The study has been reviewed and approved by our institution's IRB.

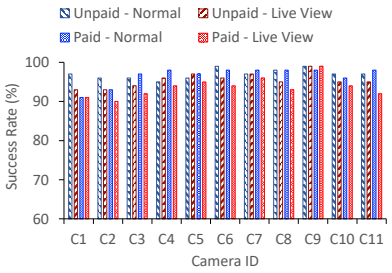



**Figure 21: Outdoor success rates.**

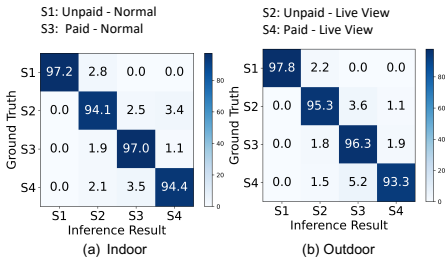



**Figure 22: Overall confusion matrix.**

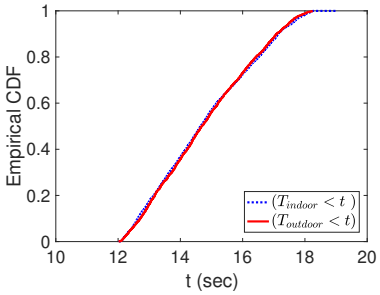



**Figure 23: CDFs of detection time.**

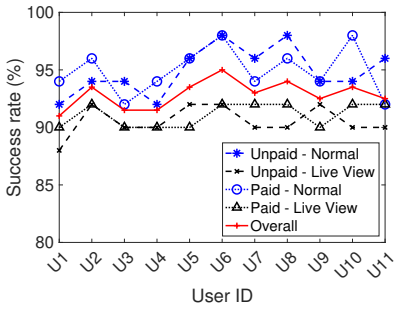



**Figure 24: Individual success rates.**

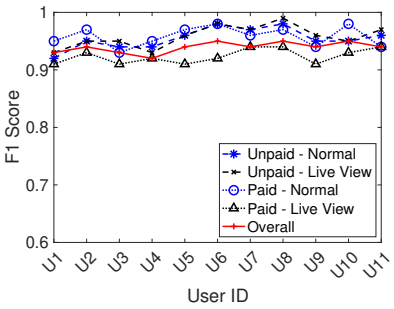



**Figure 25: Individual F1 scores.**

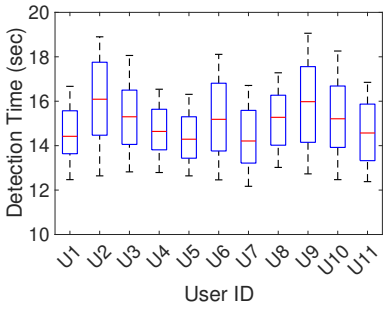



**Figure 26: Individual detection time.**

Figure 26 plots the users' detection time. We observe a consistent average detection time for all users varying from 14.3 and 16.0 seconds, indicating that a user can generally identify the camera state within a short period. This verifies the practicality of *WeakCamID*.

# 6 DISCUSSIONS

## 6.1 Limitations

**WiFi Dependency:** *WeakCamID* currently only works for cameras using WiFi connections as it cannot scan cellular networks (e.g., 4G LTE or 5G) over a licensed spectrum. In reality, some cameras (e.g., Arlo Go 2 [18]) use cellular data, especially in locations without a dedicated Internet. The general *WeakCamID* technique is expected to still work if it can sniff cellular traffic by incorporating special equipment (such as Universal Software Radio Peripheral) to receive signals in the corresponding carrier frequency range [44, 69].

**Motion Requirement:** To trigger wireless cameras, *WeakCamID* needs to generate motion stimuli, which may introduce potential risks of getting caught for the adversary. However, the adversary can perform normal and daily activities (e.g., walking) in disguise. Thus, even when such benign activities are recorded, they would not arouse suspicion and rarely harm the adversary. Alternatively, the attacker can ask a helper or control a moving object (e.g., a drone/robot) to perform movements for triggering the cameras.

**Cameras with Local Storage:** Some cameras or their base stations equip USB ports or microSD card slots for expandable local storage, which keeps recordings on local devices without sending them to the cloud. On one hand, if the local storage is enabled at the base station side (e.g., [2]), *WeakCamID* still works as there is still real-time wireless traffic between the camera and the base station for video footage. On the other hand, the local storage enabled on the camera side usually offers video storage when the WiFi goes out, such as Nest Cam [35]. In such cases, *WeakCamID* fails as the camera generates no real-time wireless traffic. It continues to work if the WiFi is on. Nevertheless, the owner of the camera often cannot view local storage via the app and has to connect it to a computer. Another downside to local storage is that if someone breaks in and steals the storage device (or the whole camera/base station), all video recordings on it would be lost.

## 6.2 Defending Strategies

*WeakCamID* exploits the correlation between the created motion and the resultant wireless traffic to infer camera states. Intuitively, to defend against such attacks, we can stop the attacker from obtaining the correct motion-traffic relationship. Accordingly, one straightforward defense is to disable motion detection of the camera, which will thus not react to any motion. To avoid losing the protection of the target area, the camera can then enable continuous video recording (CVR) that records videos continuously (24 hours a day). However, this feature can be unlocked via paid CVR subscriptions. Also, constant recording is power-consuming, making it often not supported by battery-power cameras. Instead, to support CVR, cameras are usually required to be plugged into AC power (e.g., [20]), bringing inconvenience to installation.

Alternatively, the camera can confuse the attacker by randomizing the time of uploading recorded videos. Currently, after sending the motion alert, the camera with a subscription also immediately records to the cloud. With this defense, the paying camera can postpone uploading recordings for a random delay, leading to two phenomena that increase attack difficulty. First, the attacker would only observe short wireless traffic for both paying and non-paying cameras right after performing motion in the activation zone of the camera. The corresponding traffic would be similar regardless

of subscription status. Second, there is a break between the traffic flows for video uploading and for sending a push notification. Thus, the traffic for cloud recording can be confused with that for streaming live videos. As a result, the attacker is unable to correctly infer camera states. However, this defense needs local storage at the camera side to temporarily hold the recordings.

More expediently, we can disrupt the wireless traffic patterns observed at the sniffer by padding traffic (i.e., adding traffic volume) generated by the camera. Recent studies [14, 27] have demonstrated success in applying traffic padding to reshape traffic patterns of IoT devices. Particularly, two defense strategies can be taken, i.e., *obfuscation* and *deception*. Towards obfuscation, we pad the motion-induced traffic to make it featureless so that the traffic shows a consistent pattern regardless of the camera state. In terms of deception, we intentionally generate bogus traffic flows to mislead the attacker to infer a fake camera state. However, the defense inevitably imposes communication and power overheads.

### 6.3 Camera Models Not Present in Training

Currently, our implementation and evaluation are based on testing the 11 selected wireless cameras. It is difficult to enumerate all wireless camera models and test the corresponding performance of *WeakCamID*. However, Section 5.5 shows that even if one of the chosen 11 cameras is not used for training, *WeakCamID* still works for this camera with high accuracy and within a short time. Such results show preliminary evidence of the transferability of *WeakCamID*. In general, subscriptions have become a popular revenue model for camera vendors, and *WeakCamID* works for all cameras that have no active recording without a subscription.

### 6.4 Responsible Disclosure

We reported all our findings to the camera vendors including Arlo, Blink, Ring, Reolink, SimpliSafe, and Wyze. They have acknowledged receipt of our vulnerability report. Arlo validated our identified vulnerability and awarded us a bug bounty.

## 7 RELATED WORK

**Detection and Localization of Wireless Cameras**: Nowadays, the fast adoption of wireless cameras in miscellaneous venues has boosted the privacy research associated with unauthorized recording. A malicious user may stealthily deploy a wireless camera and monitor an area without obtaining the approval of the occupants in the area. A myriad of recent studies ([24, 25, 38, 50, 65, 71, 78]) thus focus on detecting or pinpointing wireless cameras to protect user privacy. For example, [65] detects and localizes wireless cameras using the time-of-flight (ToF) sensor on commodity smartphones. Also, by exploring the causality between patterns in observable wireless traffic and a coexisting trusted sensor such as a light sensor [50], motion sensor [38], or inertial measurement unit (IMU) [71], a wireless camera can be detected or localized. On the contrary, our work is the first to explore how to infer whether a wireless security camera has a subscription plan without hacking into the camera or connecting to the same network with the camera.

**Camera Jammer:** A malicious user may utilize a WiFi signal blocker or jammer to send interference using the same radio frequency as the camera [36], so that the traffic generated by the camera will be disrupted. However, such suppression methods not only need extra high cost for jamming hardware, but also disable all WiFi connections nearby. Also, as the heartbeat signals (reporting the camera's connectivity health to the server) will be jammed, the camera app will display an offline status message and an alert accordingly. Alternatively, the study [59] selectively suppresses sensory measurements (e.g., video recording) at the network layer rather than heartbeat messages. Nevertheless, this method requires pre-compromising the wireless router that the target camera connects to. Such a requirement imposes a practical hurdle for the attacker as the target camera usually utilizes a password-protected WiFi network that is not accessible to the attacker. On the contrary, *WeakCamID* is non-invasive, having no impact on environmental WiFi devices, and requiring neither expensive hardware nor control over the router that the target camera connects to.

**Traffic Analysis:** Traditional traffic analysis is to extract and infer information from network meta-data such as the volumes and timing of network packets, even when the network traffic is encrypted [55]. In past years, traffic analysis has shown success in achieving a wide variety of applications, from classifying the traffic's nature (e.g., video streaming and p2p traffic) [61] to revealing fine-grained App usage [60, 73] or even user actions on Apps [10, 26, 74], and from unveiling streamed YouTube videos [47] to identifying IoT devices including drones [12, 58, 68], wireless cameras [24, 38, 71], and sound-triggered devices [54].

Most existing traffic analysis studies (e.g., [24, 47, 68, 73, 74]) take advantage of the fact that different applications or devices generate distinguishable traffic patterns or signatures, which in turn can be used to infer applications or devices. However, to determine the subscription and live-view mode on/off states, our work exploits the correlation between the manipulated motion stimuli fed into a wireless camera with the resultant traffic generated by the camera.

## 8 CONCLUSION

We propose *WeakCamID* for a novel and universal camera state inference technique. It is the first to point out the vulnerability of current wireless security cameras without subscription plans. An adversary may bypass such a camera without being recorded via passive WiFi sniffing. *WeakCamID* can be realized with a single smartphone and requires neither any professional equipment nor connecting to the same network as the target camera. It works by generating motion to stimulate the camera, and correlating the camera state (i.e., the statuses of subscription and live view mode) with the disclosed traffic pattern. We develop a mobile app to implement *WeakCamID*. Extensive real-world experiments on top of the developed app and 11 popular wireless cameras verify the effectiveness and efficiency of *WeakCamID*.

## ACKNOWLEDGMENTS

We would like to thank all anonymous reviewers for their insightful comments. This work was supported in part by NSF under Grants No.1948547 and No.2155181.

## REFERENCES

[1] 2022. CC3220R, CC3220S, and CC3220SF SimpleLink Wi-Fi Single-Chip Wireless MCU Solutions. https://www.ti.com/lit/ds/symlink/cc3220s.pdf.

[2] 2022. How do I set up local storage backups using my Arlo Base Station or SmartHub? https://kb.arlo.com/1146857/How-do-I-set-up-local-storage-backups-using-my-Arlo-Base-Station-or-SmartHub.

[3] 2022. The C-based Firmware Patching Framework for Broadcom/Cypress WiFi Chips that enables Monitor Mode, Frame Injection and much more. https://github.com/seemoo-lab/nexmon.

[4] 2023. Airmon-ng [Aircrack-ng]. https://www.aircrack-ng.org/doku.php?id=airmon-ng.

[5] 2023. Blink Outdoor - wireless, weather-resistant HD security camera. https://www.amazon.com/dp/B086DKSYTS.

[6] 2023. OUI. https://standards-oui.ieee.org/.

[7] 2023. Ring Security Camera Product Comparison. https://support.ring.com/hc/en-us/articles/115004687606-Ring-Security-Camera-Product-Comparison.

[8] 2023. SQLite. https://www.sqlite.org/index.html.

[9] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, Valerio Persico, and Antonio Pescapé. 2019. MIRAGE: Mobile-app Traffic Capture and Ground-truth Creation. In *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. 1–8.

[10] Fabio Aiolli, Mauro Conti, Ankit Gangwal, and Mirko Polato. 2019. Mind Your Wallet's Privacy: Identifying Bitcoin Wallet Apps and User's Actions through Network Traffic Analysis. In *Proc. of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*. ACM, 1484–1491.

[11] Aircrack-ng. 2023. Aircrack-ng - Main documentation. https://www.aircrack-ng.org/documentation.html.

[12] Anas Alsoliman, Giulio Rigoni, Marco Levorato, Cristina Pinotti, Nils Ole Tippenhauer, and Mauro Conti. 2021. COTS Drone Detection Using Video Streaming Characteristics. In *International Conference on Distributed Computing and Networking 2021 (ICDCN '21)*. ACM, 166–175.

[13] Apple Inc. 2022. Apple Developer Documentation: HTTP Live Streaming. https://developer.apple.com/documentation/http_live_streaming.

[14] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the Smart Home Private with Smart (er) IoT Traffic Shaping. *Proc. on Privacy Enhancing Technologies* 3 (2019), 128–148.

[15] Arlo. 2022. Investor Relations. https://investor.arlo.com/ir-home/default.aspx.

[16] Arlo. 2022. What are Arlo's advanced motion alerts, and how do I set them up? https://kb.arlo.com/000056585/What-are-Arlo-s-advanced-motion-alerts-and-how-do-I-set-them-up.

[17] Arlo. 2022. What is the maximum video resolution of Arlo cameras? https://kb.arlo.com/1707/What-is-the-maximum-video-resolution-of-Arlo-cameras.

[18] Arlo. 2023. Arlo Go 2 LTE/Wi-Fi Security Camera. https://www.arlo.com/en-us/cameras/go/arlo-go-2-cameras.html.

[19] Arlo. 2023. Arlo Secure Subscription Plans. https://www.arlo.com/en-us/arlosecure.html.

[20] Arlo. 2023. What is continuous video recording (CVR) and how do I use it? https://kb.arlo.com/1018425/What-is-continuous-video-recording-CVR-and-how-do-I-use-it.

[21] Bitmovin Inc. 2022. Bitmovin Video Developer Report 2021. https://go.bitmovin.com/video-developer-report-2021.

[22] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. 2013. API design for machine learning software: experiences from the scikit-learn project. In *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases*.

[23] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 1–27.

[24] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. DeWiCam: Detecting Hidden Wireless Cameras via Smartphones. In *Proc. of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18)*. ACM, 1–13.

[25] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2020. On Detecting Hidden Wireless Cameras: A Traffic Pattern-based Approach. *IEEE Transactions on Mobile Computing* 19, 4 (2020), 907–921.

[26] Mauro Conti, Luigi V. Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2015. Can't You Hear Me Knocking: Identification of User Actions on Android Apps via Traffic Analysis. In *Proc. of the 5th ACM Conference on Data and Application Security and Privacy (CODASPY '15)*. ACM, 297–304.

[27] Trisha Datta, Noah Apthorpe, and Nick Feamster. 2018. A Developer-Friendly Library for Smart Home IoT Privacy-Preserving Traffic Obfuscation. In *Proc. of the 2018 Workshop on IoT Security and Privacy (IoT S&P '18)*. ACM, 43–48.

[28] Sheena Lewis Erete. 2013. Protecting the Home: Exploring the Roles of Technology and Citizen Activism from a Burglar's Perspective. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, 2507–2516.

[29] Federal Bureau of Investigation. 2022. Burglary. https://ucr.fbi.gov/crime-in-the-u.s/2019/crime-in-the-u.s.-2019/topic-pages/burglary.

[30] Ellis Fenske, Dane Brown, Jeremy Martin, Travis Mayberry, Peter Ryan, and Erik C Rye. 2021. Three Years Later: A Study of MAC Address Randomization In Mobile Devices And When It Succeeds. *Proc. Priv. Enhancing Technol.* 2021, 3 (2021), 164–181.

[31] Wireshark Foundation. 2023. Wireshark. https://www.wireshark.org/.

[32] Stacy Fritz and Michelle Lusardi. 2009. White paper:"walking speed: the sixth vital sign". *Journal of geriatric physical therapy* 32, 2 (2009), 2–5.

[33] Global Justice Information Sharing Initiative. 2016. Video Evidence: A Primer for Prosecutors. https://bja.ojp.gov/library/publications/video-evidence-primer-prosecutors.

[34] Daniel Golightly. 2022. What is Arlo Secure? Everything You Need To Know. https://www.androidheadlines.com/what-is-arlo-secure-everything-you-need-to-know.html.

[35] Google. 2023. How Nest cameras store recorded video. https://support.google.com/googlenest/answer/9242083?hl=en.

[36] Ramakrishna Gummadi, David Wetherall, Ben Greenstein, and Srinivasan Seshan. 2007. Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In *Proc. of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '07)*. ACM, 385–396.

[37] Neilly H. Tan, Richmond Y. Wong, Audrey Desjardins, Sean A. Munson, and James Pierce. 2022. Monitoring Pets, Deterring Intruders, and Casually Spying on Neighbors: Everyday Uses of Smart Home Cameras. In *CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM, Article 617, 25 pages.

[38] Yan He, Qiuye He, Song Fang, and Yao Liu. 2021. Motioncompass: pinpointing wireless camera via motion-activated traffic. In *Proc. of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21)*. ACM, 215–227.

[39] 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 *Proc. of the 38th Annual Computer Security Applications Conference (ACSAC '22)*. ACM, 714–726.

[40] Tin Kam Ho. 1995. Random decision forests. In *Proc. of 3rd international conference on document analysis and recognition*, Vol. 1. IEEE, 278–282.

[41] Intel. 2022. Intel Centrino Advanced-N 6200, Dual Band. https://www.intel.com/content/www/us/en/products/sku/59470/intel-centrino-advancedn-6200-dual-band/specifications.html.

[42] Minhaj Ahmad Khan and Khaled Salah. 2018. IoT security: Review, blockchain solutions, and open challenges. *Future generation computer systems* 82 (2018), 395–411.

[43] Tom Kolnowski. 2020. Definitive Guide to Security Camera Subscription Plans. https://digitized.house/guide-to-security-camera-cloud-storage-plans/.

[44] Swarun Kumar, Ezzeldin Hamed, Dina Katabi, and Li Erran Li. 2014. LTE radio analytics made easy and accessible. *ACM SIGCOMM Computer Communication Review* 44, 4 (2014), 211–222.

[45] N N Kundan, Asavari Patil, and Vineet Kumar. 2021. Wireless Security Cameras Systems Market. https://www.alliedmarketresearch.com/wireless-security-cameras-systems-market-A14252.

[46] 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)*. 315–318.

[47] Ying Li, Yi Huang, Richard Xu, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Darren Webb, and Guillaume Jourjon. 2018. Deep Content: Unveiling Video Streaming Content from Encrypted WiFi Traffic. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. 1–8.

[48] Zhijing Li, Zhujun Xiao, Yanzi Zhu, Irene Pattarachanyakul, Ben Y Zhao, and Haitao Zheng. 2018. Adversarial localization against wireless cameras. In *Proc. of the 19th International Workshop on Mobile Computing Systems & Applications*. 87–92.

[49] Haili Liu, Ya Wang, Kevin Wang, and Hanbing Lin. 2017. Turning a pyroelectric infrared motion sensor into a high-accuracy presence detector by using a narrow semi-transparent chopper. *Applied Physics Letters* 111, 24 (2017), 243901.

[50] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting wireless spy cameras via stimulating and probing. In *Proc. of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18)*. ACM, 243–255.

[51] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. 2017. A Study of MAC Address Randomization in Mobile Devices and When it Fails. *Proc. on Privacy Enhancing Technologies* 4 (2017), 365–383.

[52] Jeremy Martin, Erik Rye, and Robert Beverly. 2016. Decomposition of MAC Address Structure for Granular Device Inference. In *Proc. of the 32nd Annual Conference on Computer Security Applications* (Los Angeles, California, USA) *(ACSAC '16)*. ACM, 78–88.

[53] Philip Matthews, Iljitsch van Beijnum, and Marcelo Bagnulo. 2011. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146. https://doi.org/10.17487/RFC6146 https://www.rfc-editor.org/info/rfc6146.

[54] Richard Mitev, Anna Pazii, Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi. 2020. LeakyPick: IoT Audio Spy Detector. In *Annual Computer Security Applications Conference (ACSAC '20)*. ACM, 694–705.

[55] S.J. Murdoch and G. Danezis. 2005. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*. 183–195.

[56] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.
[57] Sujay Narayana, R. Venkatesha Prasad, Vijay S. Rao, T. V. Prabhakar, Sripad S. Kowshik, and Madhuri Sheethala Iyer. 2015. PIR Sensors: Characterization and Novel Localization Technique. In *Proc. of the 14th International Conference on Information Processing in Sensor Networks (IPSN '15)*. ACM, 142–153.
[58] Phuc Nguyen, Hoang Truong, Mahesh Ravindranathan, Anh Nguyen, Richard Han, and Tam Vu. 2017. Matthan: Drone Presence Detection by Identifying Physical Signatures in the Drone's RF Communication.
[59] TJ OConnor, William Enck, and Bradley Reaves. 2019. Blinded and Confused: Uncovering Systemic Flaws in Device Telemetry for Smart-Home Internet of Things. In *Proc. of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19)*. ACM, 140–150.
[60] Eva Papadogiannaki, Constantinos Halevidis, Periklis Akritidis, and Lazaros Koromilas. 2018. Otter: A scalable high-resolution encrypted traffic identification engine. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 315–334.
[61] Eva Papadogiannaki and Sotiris Ioannidis. 2021. A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures. *ACM Comput. Surv.* 54, 6, Article 123 (jul 2021), 35 pages.
[62] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.
[63] Kate Rooney. 2019. Booming subscription services are building revenue for companies but sapping customers' wallets. https://www.cnbc.com/2019/11/18/subscriptions-building-revenue-for-companies-but-sapping-wallets.html.
[64] 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. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–27.
[65] Sriram Sami, Sean Rui Xiang Tan, Bangjie Sun, and Jun Han. 2021. LAPD: Hidden Spy Camera Detection Using Smartphone Time-of-Flight Sensors. In *Proc. of the 19th ACM Conference on Embedded Networked Sensor Systems (SenSys '21)*. ACM, 288–301.
[66] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. 2021. Let Numbers Tell the Tale: Measuring Security Trends in Wi-Fi Networks and Best Practices. In *Proc. of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21)*. ACM, 100–105.
[67] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2015. NexMon: A cookbook for firmware modifications on smartphones to enable monitor mode. *arXiv preprint arXiv:1601.07077* (2015).
[68] Savio Sciancalepore, Omar Adel Ibrahim, Gabriele Oligeri, and Roberto Di Pietro. 2020. PiNcH: An Effective, Efficient, and Robust Solution to Drone Detection via Network Traffic Analysis. *Comput. Netw.* 168, C (feb 2020), 15 pages.
[69] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert. 2016. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. In *Proc. 2016 Network and Distributed System Security Symposium*.
[70] Rahul Anand Sharma, Elahe Soltanaghaei, Anthony Rowe, and Vyas Sekar. 2022. Lumos: Identifying and Localizing Diverse Hidden IoT Devices in an Unfamiliar Environment. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1095–1112.
[71] 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)*. 1829–1846.
[72] Cameron Summerson. 2021. Rooting Android Just Isn't Worth It Anymore. https://www.howtogeek.com/335642/rooting-android-just-isnt-worth-it-anymore/.
[73] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2018. Robust Smartphone App Identification via Encrypted Network Traffic Analysis. *IEEE Transactions on Information Forensics and Security* 13, 1 (2018), 63–78.
[74] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. 2020. Packet-level signatures for smart home devices. In *Network and Distributed Systems Security (NDSS) Symposium*, Vol. 2020.
[75] Jennifer Pattison Tuohy. 2021. Why Get Home Security Cameras? https://www.usnews.com/360-reviews/services/security-cameras/why-get-security-cameras.
[76] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank Piessens. 2016. Why MAC Address Randomization is Not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms. In *Proc. of the 11th ACM on Asia Conference on Computer and Communications Security* (Xi'an, China) *(ASIA CCS '16)*. ACM, 413–424.
[77] Ryan Whitwam. 2022. Why You Should (Or Shouldn't) Root Your Android Device. https://www.extremetech.com/mobile/211314-extremetech-explains-why-you-should-or-shouldnt-root-your-android-device.
[78] Kevin Wu and Brent Lagesse. 2019. Do you see what I see? Detecting hidden streaming cameras through similarity of simultaneous observation. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–10.

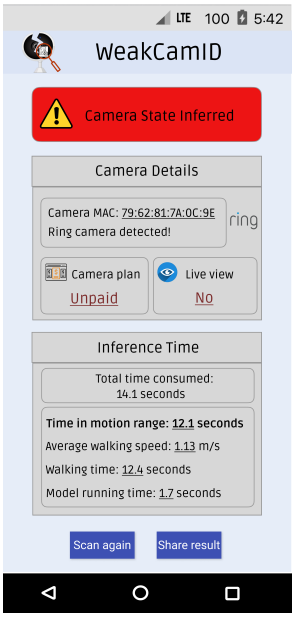



**Figure 27: The UI snapshot for the app.**

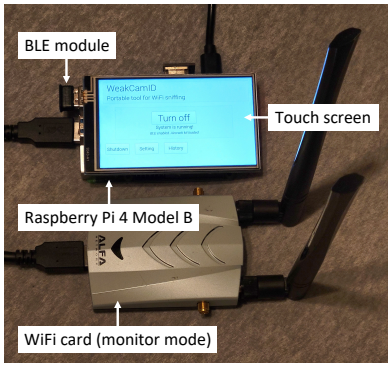



**Figure 28: Our developed external tool for WiFi sniffing.**

[79] Shichang Xu, Subhabrata Sen, and Z. Morley Mao. 2020. CSI: Inferring Mobile ABR Video Adaptation Behavior under HTTPS and QUIC. In *Proc. of the Fifteenth European Conference on Computer Systems (EuroSys '20)*. ACM, Article 33, 16 pages.

## A USER INTERFACE

Figure 27 exhibits the designed user interface (UI) of the developed mobile app *WeakCamID*.

## B EXTERNAL TOOL FOR WIFI SNIFFING

Our designed external tool consists of four components, as shown in Figure 28: a BLE module for a phone connection, a touch screen for user interaction, a WiFi adapter card (e.g., RTL8814AU chipset) in monitor mode, and a Raspberry Pi 4 Model B acting as a platform for the previous three components. The total cost is around $70.