LeakyPick: IoT Audio Spy Detector

Richard Mitev
richard.mitev@trust.tu-darmstadt.de
Technical University of Darmstadt

Anna Pazii
anna.pazii@inria.fr
University of Paris Saclay

Markus Miettinen
markus.miettinen@trust.tu-darmstadt.de
Technical University of Darmstadt

William Enck
whenck@ncsu.edu
North Carolina State University

Ahmad-Reza Sadeghi
ahmad.sadeghi@trust.tu-darmstadt.de
Technical University of Darmstadt

arXiv:2007.00500v1 [cs.CR] 1 Jul 2020

ABSTRACT

Manufacturers of smart home Internet of Things (IoT) devices are increasingly adding voice assistant and audio monitoring features to a wide range of devices including smart speakers, televisions, thermostats, security systems, and doorbells. Consequently, many of these devices are equipped with microphones, raising significant privacy concerns: users may not always be aware of when audio recordings are sent to the cloud, or who may gain access to the recordings. In this paper, we present the LeakyPick architecture that enables the detection of the smart home devices that stream recorded audio to the Internet without the user's consent. Our proof-of-concept is a LeakyPick device that is placed in a user's smart home and periodically "probes" other devices in its environment and monitors the subsequent network traffic for statistical patterns that indicate audio transmission. Our prototype is built on a Raspberry Pi for less than USD $40 and has a measurement accuracy of 94% in detecting audio transmissions for a collection of 8 devices with voice assistant capabilities. Furthermore, we used LeakyPick to identify 89 words that an Amazon Echo Dot misinterprets as its wake-word, resulting in unexpected audio transmission. LeakyPick provides a cost effective approach for regular consumers to monitor their homes for unexpected audio transmissions to the cloud.

1 INTRODUCTION

Consumer Internet of Things (IoT) devices have emerged as a promising technology to enhance home automation and physical safety. While the smart home ecosystem has a sizeable collection of automation platforms, market trends in the US [44] suggest that many consumers are gravitating towards Amazon Alexa and Google Home. These two platforms are unique from the other automation platforms (e.g., Samsung SmartThings, WeMo) in that they focused much of their initial smart home efforts into smart speaker technology, which allows users to speak commands to control smart home devices (e.g., light switches), play music, or make simple information queries. This dominance of Amazon Alexa and Google Home might suggest that consumers find voice commands more useful than complex automation configurations.

For many privacy-conscious consumers, having Internet-connected microphones scattered around their homes is a concerning prospect. This danger was recently confirmed when popular news media reported that Amazon [27], Google [23, 31], Apple [24], Microsoft [13], and Facebook [18] are all using contractors to manually analyze the accuracy of voice transcription. The news reports include anecdotes from contractors indicating they listened to many drug deals, domestic violence, and private conversations. Perhaps more concerning is that many of the recordings were the result of false positives when determining the "wake-word" for the platform. That is, the user never intended for the audio to be sent to the cloud.

Unfortunately, avoiding microphones is not as simple as not purchasing smart speakers. Microphones have become a pervasive sensor for smart home devices. For example, it is difficult to find a smart television that does not support voice controls via the display or the handheld remote control. Smart thermostats (e.g., Ecobee) commonly advertise that they have dual function as a smart speaker. Surveillance cameras (e.g., Ring, Wyze) are designed to notify users of events, but are in fact always recording. Perhaps most concerning was the report that the Nest security system includes a microphone [27], despite no packing material or product documentation reporting its existence. While the manufacturers of these devices might argue that users can disable microphone functionality in device software, history has repeatedly demonstrated that software can and will be compromised. Furthermore, mass collection and storage of audio recordings increases concerns over the potential for a "surveillance state" (e.g., Ring has recently been criticized for working with local police [22]).

Our research seeks to answer the question: *How can a user effectively detect if a device is using the microphone to capture the user's voice and record them without their knowledge?* We are primarily concerned with the benign, but hidden, recording and transmission of audio to cloud services, as such behaviors can potentially lead to mass surveillance. We also seek to identify the existence of compromised devices that immediately transmit audio to the network as it is recorded. We believe there is significant utility in detecting this subset of compromised devices, particularly given the limited storage capacity of many low-level IoT devices.

Existing approaches for identifying unintended data transmissions out of the user's network [11, 29] focus on other modalities (e.g., video) and rely on assumptions that do not apply to audio transmissions (e.g., some devices require an utterance of specific wake-words). Furthermore, while traffic analysis approaches targeting IoT devices have been proposed [33, 41], to the best of our knowledge there are no earlier approaches specifically targeting traffic of

Conference'17, July 2017, Washington, DC, USA
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

microphone-enabled IoT devices. Additionally, prior work attacking voice assistants and voice recognition [3, 7, 8, 14, 28, 30, 40, 46, 49–51] focuses on maliciously issuing commands or changing the interaction flow without the victim noticing.

In this paper, we present the LeakyPick architecture, which includes a small device that can be placed in various rooms of a home to detect the existence of smart home devices that stream recorded audio to the Internet. LeakyPick operates by periodically "probing" an environment (i.e., creating noise) when the user is not home (to avoid annoyance) and monitoring subsequent network traffic for statistical patterns that indicate the transmission of audio content. We evaluate LeakyPick by studying several popular voice assistants, which provide ground truth for analysis. Our evaluation shows that LeakyPick has an observed accuracy of 94% for detecting the transmission of audio. LeakyPick achieves this accuracy using a statistical approach that is generalizable to a broad selection of voice-enabled IoT devices, eliminating the need for time-consuming training required by machine learning. We also used LeakyPick to perform a wake-word fuzz test of an Amazon Echo Dot, discovering 89 words that unexpectantly stream audio recordings to Amazon. In doing so, LeakyPick provides a cost effective approach for regular consumers to monitor their homes for unexpected audio transmissions to the cloud.

This paper makes the following contributions:

- *We present the* LeakyPick *device for identifying smart home devices that unexpectedly record and send audio to the Internet.* The device costs less than USD $40 and can be easily deployed in multiple rooms of a home.
- *We present a novel audio-based probing approach for estimating the likelihood that particular devices react to specific audio-based keywords.* Our approach has a 94% accuracy for a set of devices known to transmit audio to the cloud. We also show that the approach is generalizable to different device types without the need to pre-train costly device-type-specific detection profiles.
- *We show that* LeakyPick *can identify hidden wake-words that cause unexpected audio transmission.* Our analysis of an Amazon Echo Dot identified 89 incorrect wake-words.

Finally, LeakyPick uses human-audible noises, which may be annoying to physically present users. Prior work has suggested the use of inaudible sound to control voice assistants using ultrasound audio [42, 50]. However, these approaches are specific to the technical characteristics of the targeted devices. Therefore, they are not immediately applicable to our goal of identifying unknown devices streaming audio. We leave the task of creating generic models of transmitting audio via ultrasonic sound as a topic for future work.

The remainder of this paper proceeds as follows. Section 2 provides background on devices with voice control and audio interfaces. Section 3 overviews our architecture. Section 4 describes our design and implementation. Section 5 evaluates the accuracy of LeakyPick. Section 6 discusses our approach and security considerations. Section 7 overviews related work. Section 8 concludes.

2 BACKGROUND

IoT devices increasingly use audio sensing for enabling voice-based control by the user or for other audio-based use cases. Examples of such devices include smart audio security systems [25], smart audio event-detecting IP cameras [20], vacuum cleaner robots equipped with microphones and nightvision [6], and smart fire alarms with a self-testing siren [19]. Due to the abundance of devices with audio sensing capabilities, the user may not always be aware of when a particular device will record audio and send it to the cloud. Sending audio to the cloud is frequently required for voice-control based user interfaces, as speech-to-text translation often needs more computational resources than are available on IoT devices.

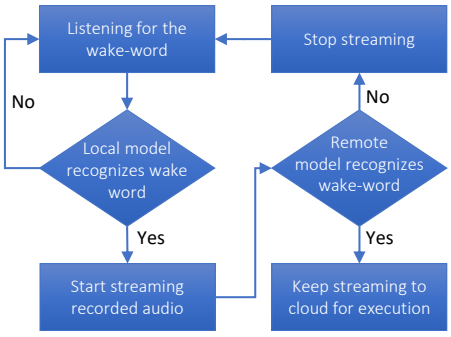

Figure 1: Overview of wake-word detection process in voice-controlled devices such as the Amazon Echo

Devices with voice-control interfaces typically use local speech recognition for detecting a specific set of "wake-words" (i.e., utterances meant to be used by the user to invoke the voice-control functionality of the device). When the local model detects the utterance of a potential wake-word, the device starts sending audio to back-end servers for voice-to-text translation. In order to not miss speech commands uttered by users, the local model needs to be configured to recognize any utterance that resembles the intended wake-word. In case of the Alexa voice assistant, it is then the task of the back-end service to verify whether the observed utterance really represents a wake-word or not, as it is equipped with a more comprehensive speech recognition model and is not limited by the potentially scarce computational capabilities of the IoT device recording the audio. Figure 1 overviews this typical approach.

Problems arise when the local or online detection model mistakenly classifies specific audio inputs as the wake-word and consequently starts sending the recorded audio to the cloud, thereby potentially leaking sensitive information. Private information may also be leaked unintentionally when the user is unaware that a device will react to specific wake-words.

Finally, attacks targeting voice assistants can use malicious audio signal injection to trick the assistant to perform actions desired by the adversary. In these attacks, the adversary either waits for the user to be asleep or not present [3, 14] or uses inaudible [30, 42, 50] or unrecognizable [7, 40, 46, 49] signals to stage the attack, making it very difficult for the victim user to realize that the device is being attacked. Our goal is to provide tools that enable users to detect audio-related transmissions by IoT devices in their network and inform the user when their audio-enabled devices transmit audio, thus making such attacks visible to the user.

Threat Model and Assumptions: In this paper, we are concerned with threats related to IoT devices that stream recorded audio over the Internet using Wi-Fi or a wired LAN connection in response to audio signals recognised by the device as potential voice commands or different sounds the device reacts to. As processing voice commands is (except for the detection of the device's wake-word) implemented on the server-side, we assume that recorded audio is transmitted instantaneously to the cloud to allow the voice-controlled device to promptly react to user commands. We consider three main threat scenarios:

(1) Audio events or utterances inadvertently recognized by the device as the device's wake-word, resulting in the *unwanted streaming of audio recordings* to the device's back-end system for further processing. The main threat in this scenario is the *unintentional disclosure* of potentially sensitive audio to the back-end system of the device's vendor.
(2) Active attacks by an external adversary as discussed above, where the adversary tricks the targeted device to initiate transmission of audio data by *injection of audio signals* to the device's environment so that the device will identify these as its wake-word. The main threat in this scenario is the *unauthorized invocation* of device or service functionalities.
(3) Application-level attacks that cause a benign IoT device to send audio without the user's knowledge. For example, the Amazon Echo contained a vulnerability [21] that allowed a malicious skill to silently listen. More recently, security researchers have reported [32] the existence of eavesdropping apps targeting Alexa and Google Assistant.

Our threat model primarily considers benign IoT devices, i.e., devices not compromised by an adversary. We do not consider scenarios in which an IoT device is modified by the adversary to transform the device to act as a bugging device that records and stores audio locally without sending it to the network. While feasible, such attacks are much less scalable, as they must be tailored for each targeted device and are only applicable for devices with sufficient storage space. We note, however, that LeakyPick is applicable to settings where an adversary has compromised an IoT device and immediately transmits audio data.

3 SOLUTION OVERVIEW

The goal of this paper is to devise a method for regular users to reliably identify IoT devices that 1) are equipped with a microphone, and 2) send recorded audio from the user's home to external services without the user's awareness. If LeakyPick can identify which network packets contain audio recordings, it can then inform the user which devices are sending audio to the cloud, as the source of network packets can be identified by hardware network addresses. This provides a way to identify both unintentional transmissions of audio to the cloud, as well as above-mentioned attacks (discussed in detail in Section 7), where adversaries seek to invoke specific actions by injecting audio into the device's environment. Achieving this goal requires overcoming the following research challenges:

- *Device traffic is often encrypted.* A naïve solution that simply looks for audio codecs in network traffic will fail to identify most audio recordings.

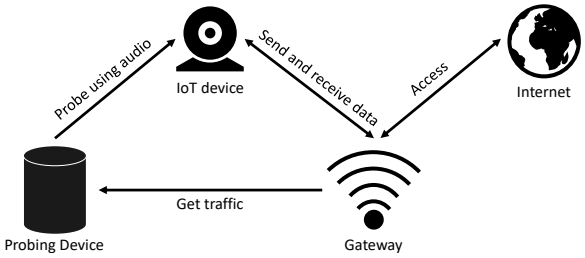

Figure 2: System set-up of LeakyPick

- *Device types are not known a priori.* Devices transmit audio in different ways. We need to identify generic approaches that work with previously unseen devices.

Due to these challenges, our solution cannot passively monitor network traffic with the goal of differentiating the transmission of audio recordings from other network traffic. While prior approaches such as HomeSnitch [34] are able to classify the semantic behavior of IoT device transmissions (e.g., voice command), they require *a priori* training for each manufacturer or device model. Since we seek to identify this behavior for potentially unknown devices, we cannot rely on supervised or semi-supervised machine learning.

At a high level, LeakyPick overcomes the research challenges by periodically transmitting audio into a room and monitoring the subsequent network traffic from devices. As shown in Figure 2, LeakyPick's main component is a probing device that emits audio probes into its vicinity. By temporally correlating these audio probes with observed characteristics of subsequent network traffic, LeakyPick identifies devices that have potentially reacted to the audio probes by sending audio recordings.

LeakyPick identifies network flows containing audio recordings using two key ideas. First, it looks for traffic bursts following an audio probe. Our observation is that voice-activated devices typically do not send much data unless they are active. For example, our analysis shows that when idle, Alexa-enabled devices periodically send small data bursts every 20 seconds, medium bursts every 300 seconds, and large bursts every 10 hours. We further found that when it is activated by an audio stimulus, the resulting audio transmission burst has distinct characteristics. However, using traffic bursts alone results in high false positive rates.

Second, LeakyPick uses statistical probing. Conceptually, it first records a baseline measurement of idle traffic for each monitored device. Then it uses an *independent two-sample t-test* to compare the features of the device's network traffic while being idle and of traffic when the device communicates after the audio probe. This statistical approach has the benefit of being inherently device agnostic. As we show in Section 5, this statistical approach performs as well as machine learning approaches, but is not limited by *a priori* knowledge of the device. It therefore outperforms machine learning approaches in cases where there is no pre-trained model for the specific device type available.

Finally, LeakyPick works for both devices that use a wake word and devices that do not. For devices such as security cameras that do not use a wake word, LeakyPick does not need to perform any special operations. Transmitting any audio will trigger the audio

transmission. To handle devices that use a wake word or sound, e.g., voice assistants, security systems reacting on glass shattering or dog barking, LeakyPick is configured to prefix its probes with known wake words and noises (e.g., "Alexa", "Hey Google"). It can also be used to fuzz test wake-words to identify words that will unintentionally transmit audio recordings.

4 LEAKYPICK DESIGN

This section describes the central aspects of LeakyPick's design. We primarily focus on audio event detection. We then describe our system implementation on a Raspbery Pi 3B.

4.1 Audio Event Detection

Due to the encryption between devices and back-end cloud systems, it is not possible to detect audio-related events by inspecting packet payloads. Instead, LeakyPick identifies audio transmissions by observing sudden outgoing traffic rate increases for specific devices, or significant changes in the device's communication behavior, both of which can indicate the transmission of audio recordings. We consider two possible complementary approaches to audio event detection: (1) a simple baseline method based on network traffic burst detection, and (2) a statistical approach for detecting apparent changes in the communication characteristics of monitored devices in response to performed audio probes.

4.1.1 Burst Detection. Our baseline approach for detecting audio transmissions is based on burst detection in the observed network traffic of devices. To do this, we need to first identify the characteristics of potential audio transmissions. We therefore analyzed the invocation process and data communication behavior of popular microphone-enabled IoT devices when they transmit audio.

Our traffic analysis was based on data of popular IoT devices with integrated virtual assistant support: (1) Echo Dot (Amazon Alexa), (2) Google Home (Google Assistant), (3) Home Pod (Apple Siri), and (4) an audio-activated home security system (Hive Hub 360). Our analysis showed that these devices do not typically send much traffic during normal standby operation. Therefore, it is possible to detect audio transmissions through the increase in traffic rate they cause. Our approach is generic in the sense that it is applicable to all devices sending audio. We chose these microphone-enabled devices, as they are popular devices produced for a broad range of use cases. To determine the parameters for burst detection, we monitored the network traffic of devices in response to audio probes emitted into the devices' environment.

We perform audio event detection by observing sudden increases in the traffic rate emitted by a device that is sustained for a specific amount of time. This is because an audio transmission will inevitably cause an increase in the data rate that will typically last at least for the duration of the transmitted audio sample. This is consistent with how most voice-controlled IoT devices utilizing cloud-based back-ends function (Section 2), where local wake-word detection causes subsequent audio to be streamed to the back-end.

Specifically, LeakyPick performs burst detection by dividing the network traffic originating from a device into time windows $W = (w_1, w_2, \ldots)$ of size s_w and calculating for each time window w_i the sum of packet payload sizes of the packets falling within the window. We then calculate the average traffic rate B_i during the time window w_i in bytes per second. If the traffic rate B_i is above a threshold B_{audio} during at least n consecutive time windows

$$W_i = (w_i, w_{i+1}, \ldots, w_{i+k-1})$$

where $k \geq n$, detection is triggered. Empirically, we found that B_{audio} = 23kbit/s is sufficient to separate audio bursts from background traffic. Therefore, it is reasonable to assume our approach will work also for systems using audio codecs with lower bandwidth requirements than Alexa.

Table 1: Parameters and packet features used by our Burst Detection and Statistical Probing approaches

Approach	Parameters	Packet Features
Burst Detection	Window size s_w Traffic rate B_{audio} Consecutive detections n	Packet size MAC/IP
Statistical Probing	Bin count k Packet sequence duration d P-value threshold t	Packet size Interarrival time MAC/IP

As shown in Table 1, LeakyPick uses predefined parameters and packet features. We extract the packet sizes, the corresponding MAC or IP (depending on the layer), and (implicitly) the direction of the packet (leaving or entering the network). To evaluate the optimal consecutive detection threshold n (Section 5.2.1), we used a fixed window size s_w = 1s, as common voice commands rarely take less than a second. For the traffic rate threshold B_{audio}, we chose 23kbit/s. This value is a sufficiently low threshold to capture any audio and sufficiently high to discard anything else, as voice recognition services need good voice recording quality (e.g., Alexa's Voice Service uses 256kbit/s, 16-bit PCM at 16kHz [4]).

4.1.2 Statistical Probing. LeakyPick uses statistical probing to refine audio transmission detection by eliminating cases where traffic bursts result from non-audio transmission. Most importantly, the approach is generic and does not need *a priori* knowledge of a device's behavior. It also can determine if a device's communication behavior changes significantly in response to audio probes.

To detect devices that react to audio, we monitor network traffic for significant changes in the device's communication behavior in response to audio probes. This is done by determining whether the distribution of the properties of communication packets transmitted by a device after the emission of an audio probe is statistically different from the distribution of packets observed before the probe injection. Specifically, LeakyPick uses a t-test [9], which is one of the most commonly used statistical tests. Given two data samples, the test computes a t-score by determining the data samples' distributions' means and standard deviations, and mapping this to a p-value. If the p-value is below a specified threshold, the distributions are considered statistically different and therefore indicate that the device reacted to the audio probe.

First, the probing device monitors idle device traffic while it is not emitting audio probes. It captures a sequence

$$T_s = (pck_1, pck_2, \ldots, pck_n)$$

of duration d seconds of data packets pck_i and calculates a packet size (or inter-arrival time) distribution vector

$$\vec{F_s} = (f_1, f_2, \ldots, f_k)$$

by binning the packets $p_i \in T_s$ into k bins based on the size (or inter-arrival time) of these packets[1] and where f_i denotes the number of packets assigned to the i-th bin.

The probing device then emits multiple audio probes and captures associated data traffic

$$T_{pr} = (pck_1, pck_2, \ldots, pck_n)$$

of duration d seconds and extracts a packet size (time) distribution vector $\vec{F_{pr}}$ using the same binning as for $\vec{F_s}$. The packet size vectors $\vec{F_s}$ and $\vec{F_{pr}}$ are then used in the t-test to determine a p-value (p) indicating the likelihood that both frequency samples originate from the same distribution (i.e., from the distribution the device produces while in idle mode).

If the p-value is below a specified threshold t (i.e., $p < t$), we assume the traffic samples are not from the same distribution. That is, the device has reacted in some way and changed its communication behavior. To further refine the results, the p-value resulting from the packet size distribution is combined with the p-value of the inter-arrival time distribution. However, as shown in Section 5.2.2, only using the p-value of the packet size distribution is sufficient.

We collected idle data samples T_s from multiple voice-controlled devices and discovered that they contained frequently occurring smaller data bursts (possibly related to, e.g., heartbeat messages) and infrequently occurring larger data bursts (e.g., state synchronization). This observation indicates it is possible to capture a large data burst in one of the two samples (T_s or T_{pr}) while missing it in the other. Therefore, when encountering a p-value indicating a possible reaction of the IoT device, the probing test must be repeated several times to ensure the observed device behavior change is caused by the audio probe and not by background traffic bursts.

Table 1 shows the parameters and packet features used for statistical probing. As with burst detection, we extract the packet sizes, the corresponding MAC or IP, and (implicitly) the direction of the packets from the recorded traffic. Additionally, we extract packet inter-arrival times. As discussed in Section 5.2.2, we set the p-value threshold t to be 0.42-0.43 to achieve an optimal precision while fixing the packet sequence duration to $d = 60$ seconds.

4.2 Wake-Word Selection

Users are not always aware of IoT devices that require a wake-word before transmitting audio recordings to the cloud. While it is possible enumerate the wake-words for known voice assistants, recent reports of third-party contractors reviewing voice assistant accuracy [13, 18, 23, 24, 27, 31] highlight the significance of false voice assistant activation. Therefore, LeakyPick identifies other wake-words that will trigger the voice detection. Note that this approach is different than using mangled voice samples [7, 46] or other means to attack the voice recognition process [8, 40, 49]. We also do not want to limit LeakyPick to words sounding similar to the known wake-words in order to confuse the voice assistant [28, 51].

Using a full dictionary of the English language is impractical. It would take roughly 40 days to test a voice assistant with the entire dictionary of 470,000 words [48] at a speed of one word every seven seconds. However, by only testing words with a phoneme count similar to the known wake-word, the subset of viable words is manageable. Our intuition is that a benign device will more likely confuse words with a similar structure. Therefore, we select all words in a phoneme dictionary [39] with the same or similar phoneme count than the actual wake-word. We also used random words from a simple English word list [26]. These words are spoken using a text-to-speech (TTS) engine.

[1] To determine the binning automatically, we use `numpy.histogram` with the bin option `auto` which uses the maximum of the Sturges and Freedman Diaconis Estimator.

4.3 System Implementation

The LeakyPick probing device injects audio probes into the user's environment and analyzes the resulting device network traffic. Our current implementation achieves this functionality using the following hardware set-up. The probing device consists of a Raspberry Pi 3B [36] connected via Ethernet to the network gateway. It is also connected via the headphone jack to a PAM8403 [15] amplifier board, which is connected to a single generic $3W$ speaker.

To capture network traffic, we use a TP-LINK TL-WN722N [45] USB Wifi dongle to create a wireless access point using `hostapd` and `dnsmasq` as the DHCP server. All wireless IoT devices connect to this access point. To provide Internet access, we activate packet forwarding between the `eth` (connected to the network gateway) and `wlan` interfaces. Alternatively the device could sniff Wi-Fi packets without being connected to the network using packet size and MAC address as the features. This approach would also work for foreign Wi-Fi networks, as it is not required to have the decrypted traffic i.e., our device does not need to be connected to that network at all: package size information is sufficient.

Finally, LeakyPick is written in Python. It uses `tcpdump` to record packets on the `wlan` interface. We use Google's text-to-speech (TTS) engine to generate the audio played by the probing device.

5 EVALUATION

This section evaluate LeakyPick's ability to detect when audio recordings are being streamed to cloud servers. Specifically, we seek to answer the following research questions.

- **RQ1** What is the detection accuracy of the burst detection and statistical probing approaches used by LeakyPick?
- **RQ2** Does audio probing with a wrong wake-word influence the detection accuracy?
- **RQ3** How well does LeakyPick perform on a real-world dataset?
- **RQ4** How does LeakyPick's statistical probing approach compare to machine learning-based approaches?
- **RQ5** Can LeakyPick discover unknown wake-words?

5.1 Experimental Setup

Our evaluation considers 8 different wireless microphone-enabled IoT devices: 3 smart speakers, 1 security system that detects glass breaking and dogs barking, and 4 microphone-enabled IoT devices, namely the audio event detecting smart IP security cameras Netatmo Welcome, Netatmo Presence and Hive View as well as the smart smoke alarm Nest Protect. Table 2 lists the specific devices. We now describe our dataset collection and evaluation metrics.

Table 2: Devices used for evaluation

Device Type	Device Name
Smart Speaker	Echo Dot (Amazon Alexa) Google Home (Google Assistant) Home Pod (Apple Siri)
Security System	Hive Hub 360
Microphone-Enabled IoT Device	Netatmo Welcome Netamo Presence Nest Protect Hive View

5.1.1 Datasets. We used four techniques to collect datasets for our evaluation. Table 3 overviews these four collection methodologies, as well as to which devices the datasets apply. The following discussion describes our collection methodology.

Idle Datasets: The idle dataset was collected in an empty office room. It consists of network traffic collected over six hours during which the device was not actively used and no audio inputs were injected.

Controlled Datasets - Burst Detection: The controlled datasets for burst detection were collected in an empty office room while injecting audio probes approximately 100 times for each of the studied devices. In all cases, the injected probe was the known wake-word for the device in question. The Hive 360 Hub device does not use a wake-word, but is activated by specific noise like dog barking and glass shattering. For this device we therefore used recordings of dog barking sounds to trigger audio transmission. For each device, three different datasets were collected by varying the wake-word invocation interval between 1, 5, and 10 minutes.

Controlled Datasets - Statistical Probing: The collection of the controlled dataset for statistical probing was performed in a way similar to the burst detection dataset. However, the experiment collected six datasets for each device. Each dataset consisted of six hours of invoking the wake-word at intervals ranging from two minutes to two hours. Thereby resulting in datasets with varying ratios of audio-related and idle background traffic.

Online Probing Datasets: Using live traffic of the 8 different devices listed in Table 2 we randomly selected a set of 50 words out of the 1000 most used words in the English language [16] combined with a list of known wake-words of voice-activated devices as possible wake-words to test. We configured our probing device to alternatingly record silence traffic T_s and probing traffic T_{pr} of one minute duration each for every wake-word in the list. T_{pr} was recorded immediately after the device started playing a word from the list repeating the word every 10 seconds in this minute.

Real-World Datasets: To create a realistic dataset for evaluation, we collected data from the three smart speakers over a combined period of 52 days in three different residential environments (houses). The times for each smart speaker are listed in Table 4. During this time period, humans used the assistants as intended by the manufacturer.

In order to evaluate the accuracy of LeakyPick, the dataset was labeled by recording the timestamps of when the device was recording audio. This was accomplished by taking advantage of the visual indicator (e.g., a light ring that glows) that Smart speakers use to alert the user when the voice assistant is activated in response to voice inputs. We therefore automated the labeling process in the real-world environment by creating a small specialized device with a light sensor to measure the visual indicator. Our device consisted of a Raspberry Pi and a Light Dependent Resistor (LDR) in conjunction with a LM393 [43] analogue-digital comparator. The LDR sensor was then attached to the smart speaker's visual indicator and protected from environmental luminosity with an opaque foil. This setup allowed the Raspberry Pi to record a precise timestamp each time the device was activated and allowed us to label periods with audio activity in the dataset accordingly.

Table 3: Datasets for Burst Detection, Statistical Probing, Online Probing and Machine Learning

Dataset	Frequency	Devices
Idle	-	Echo Dot, Google Home, Home Pod, Hive 360 Hub
Controlled - Burst Detection	1min, 5min, 10min	Echo Dot, Google Home, Home Pod, Hive 360 Hub
Controlled - Statistical Probing	2min, 5min, 10min, 30min, 1h, 2h	Echo Dot, Google Home, Home Pod, Hive 360 Hub
Online Probing	10s during probing windows	all, cf. Table 2
Real-World	real-world, cf. Table 4	Echo Dot, Google Home, Home Pod

Table 4: Duration of collected data in different residential environments (households) while used by humans

Amazon Echo Dot	Google Home	Apple Home Pod
31d	15d	15d

5.1.2 Evaluation metrics. We evaluate the performance of our detection approach in terms of true positive rate (TPR) and false positive rate (FPR). The true positive rate is defined as $TPR = \frac{TP}{TP + FN}$, where TP is true positives and FN false negatives, resulting in the fraction of audio events correctly identified as such. Similarly, false positive rate is defined as $FPR = \frac{FP}{TN + FP}$, where TN is the true negatives and FP the false positives. It denotes the fraction of non-audio events falsely identified as audio events. Ideally we would like our system to maximize TPR, i.e., the capability to identify devices sending audio to the cloud, while minimizing FPR, i.e., generating as few as possible false detections of audio transmissions.

5.2 RQ1: Detection Accuracy

In this section we evaluate the detection accuracy of our two approaches: (1) burst detection and (2) statistical probing.

5.2.1 Burst Detection. To evaluate the performance of Burst Detection for detecting audio transmissions, we used the controlled dataset for burst detection (Table 4) to determine its ability to detect audio transmissions correctly.

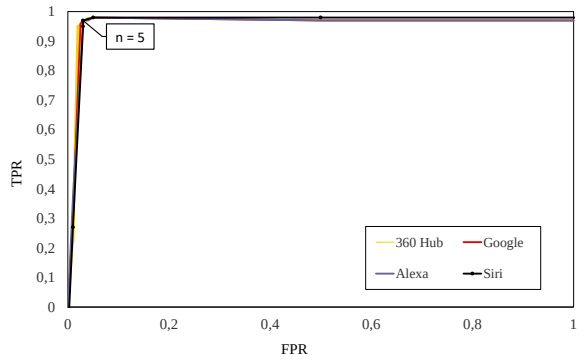

Figure 3: Results of BurstDetector using known wake-words detecting outgoing audio transmissions of Echo Dot, Google Home, Home Pod and Hive 360 Hub on the controlled data set

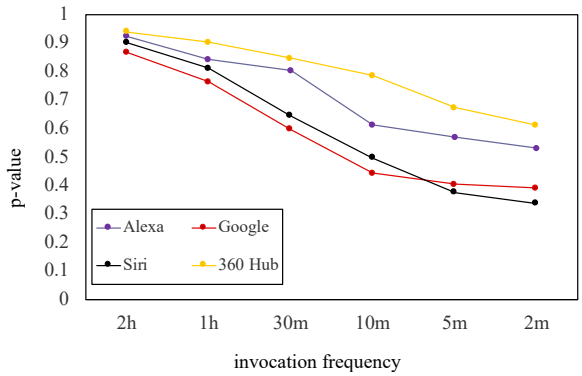

Figure 4: The resulting p-value when traffic of devices being invoked in intervals from 2 minutes to 2 hours compared to known silence, showing that the p-value decreases with an increasing number of audio bursts in the traffic

Figure 3 shows the receiver operating characteristic (ROC) curve for the burst detection approach. The ROC curve varies the parameter n, which defines the number of consecutive windows with high data throughput required for triggering detection (cf. Section 4.1.1), from $n_{min} = 1$ to $n_{max} = 8$. As can be seen, with $n = 5$ consecutive time windows, detection is triggered with a TPR of 96% and an FPR of 4% (averaged over all devices). This is explained by the fact that as mentioned in Section 3, the voice-activated devices typically send only a small amount of data unless they are active: medium bursts every few minutes and large bursts only every few hours when idle. This allows Burst Detection to identify nearly all audio invocations as they are clearly distinguishable from idle traffic, making this approach practical for devices with such behavioral characteristics.

5.2.2 Statistical Probing. To evaluate the ability of LeakyPick to detect whether a device reacts to audio events, we first determine whether the statistical properties of data traffic of IoT devices when in idle mode (i.e., not in relation to audio events) is statistically different from the devices' behavior when transmitting audio to the cloud. For this, we calculate the statistical difference of the packet size distributions in the Idle dataset to the packet distributions in the controlled datasets for statistical probing (Table 3) using the t-test as discussed in Section 4.1.2. The results are shown in Figure 4, showing the resulting p-value in dependence of the frequency of invocations of the corresponding device's wake-word. As can be seen, for all tested voice-controlled devices, the p-value decreases the more often the wake-word is injected, i.e., the more audio-transmissions the dataset contains. This suggests that the distributions of packet sizes related to audio transmission indeed are different to the distribution of packets in the background traffic and can be thus utilized to identify audio transmissions.

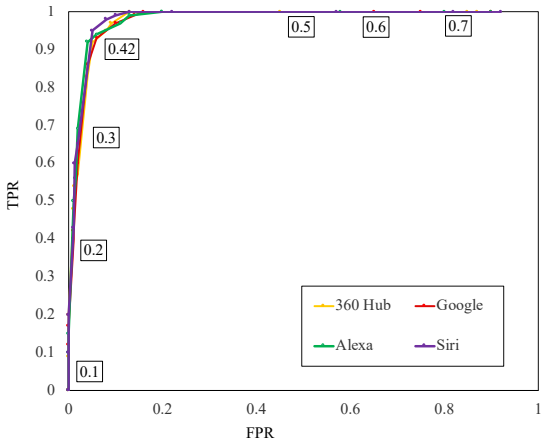

Figure 5: ROC graph of comparing consecutive windows of 30 seconds of traffic of the Controlled - Statistical probing dataset using the t-test for different p-value thresholds and comparing the output to the actual labels of the traffic

Figure 5 shows the ROC curve for our approach on the controlled dataset for statistical probing (Table 4) for different p-value thresholds. We use a sliding window approach and compare two consecutive windows of 30 seconds duration using the test, moving the window for 30 seconds to get the new window. We compare the result with the actual label of this traffic region to assess if our approach can reliably find exactly the device sending audio data. As can be seen, for a p-value threshold of 0.42 or 0.43 a True Positive Rate of 94% with a simultaneous False Positive Rate of 6% averaged over all devices can be achieved for these datasets.

5.3 RQ2: Wake-Word Sensitivity

LeakyPick is designed to detect devices reacting to a specific set of wake-words. However, as this set may be different for different device types, a relevant question is to what degree the detection accuracy is dependent on the presence of the correct wake-words in the audio probes. To evaluate this aspect, we first tested LeakyPick on the Online Probing dataset representing a live operative setting in which a number of audio probes containing actual wake-words were injected into the environment of an Amazon Echo device with the target of trying to trigger a wake-word induced audio transmission. We used the t-test as discussed in Sect. 4.1.2 to calculate the p-value between consecutive samples of packet sequences

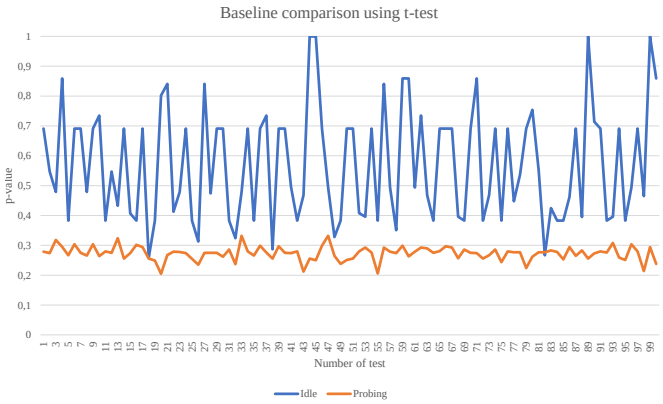

Figure 6: LeakyPick t-test p-values for probing Amazon Echo during 100 alternating windows of 1 minute of idle traffic and probing with wake-word "Alexa" at 10-second intervals, respectively

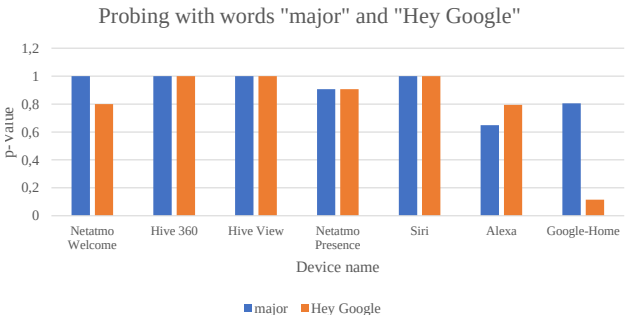

Figure 7: Representative examples of LeakyPick p-values for audio probes. None of the devices react to the non-wake-word probe "major" while only the Google Home device shows reaction for its wake-word "Hey Google"

T_s and T_{pr} of duration $d = 60$ seconds each. The result of this for 100 time window pairs is shown in Figure 6. As can be seen, the p-values for the non-probing (i.e., "idle" time windows) range between approximately 0.3 and 1, whereas the p-values for time windows containing audio probes remain mostly below 0.3. This shows that given an appropriate p-value threshold LeakyPick is able to distinguish between "idle" traffic and audio transmissions.

To further evaluate how sensitive LeakyPick is to the use of the right wake-words, we compiled a set of audio probes consisting of the 50 most used English words and a set of nine known wake-words used by the IoT devices used in our evaluation (shown in Table 2). The set of audio probes was injected into the devices' environment and the resulting p-values for each device evaluated. The resulting p-values for two representative examples of used audio probes are shown in Figure 7. The shown audio probes are the randomly-selected word "major", which does not correspond to any known wake-word of any of the tested devices and the Google Home wake-word "Hey Google". While these examples are provided to demonstrate the discriminative ability of our approach, similar results apply also to other words in the list of tested audio probes. As one can see, with a p-value threshold of, e.g., 0.5 the word "major" would not be considered to activate any of the devices, whereas one can clearly see that the p-value for "Hey Google" indicates a reaction by the Google Home device. From the results we can see that only devices responsive to a wake-word react to it which in turn can be detected using the statistical t-test employed by LeakyPick. Note that Nest Protect is not shown in Figure 7, as it was not activated by any of the examined words and therefore did not transmit any audio data.

5.4 RQ3 and RQ4: Real-World Performance

We evaluated LeakyPick on our real-world dataset containing 52 days of operation in residential environments (households) (Table 4). In addition to using this dataset to measure the accuracy of LeakyPick (**RQ3**), we also compare LeakyPick's accuracy to that of machine learning algorithms (**RQ4**). Recall from Section 3 that a key research challenge is being able to operate for unknown devices. Since machine learning algorithms require training on known devices, they are not appropriate to achieve our goals, as our approach needs to be able to handle also previously unseen device-types. That said, we use a trained machine learning algorithm as a baseline, hypothesizing that LeakyPick can perform at least as well, but without the need for training.

5.4.1 ML Implementation. We tested the performance of a number of different commonly-used machine learning (ML) algorithms for detecting audio events in the real-world dataset. We then selected the classifier with the best performance to compare against the statistical detection approach used by LeakyPick. We consider both simple ML algorithms as well as more advanced ensemble (i.e., Bagging and Boosting) and majority voting-based classifiers. The ML algorithms tested include XGboost [10], Adaboost [17], RandomForest [5], SVM with RBF kernel [47], K-NN [2], Logistic Regression, Naïve Bayes, and Decision Tree classifiers as provided by the the Scikit-learn ML package [1]. For each classifier, the used hyper-parameters were tuned using the provided Grid-search and Cross-validation processes. For constructing features for training we extracted the sequence of packet lengths (SPL) from the traffic flow and utilized the `tsfresh` tool [12] that automatically calculates a large number of statistical characteristics from a time-ordered sequence of packets. All experiments were conducted on a laptop that runs Ubuntu Linux 18.04 with an Intel i7-9750H CPU with 32 GB DDR4 Memory.

5.4.2 Evaluation. For the ML approach, we used 90% of the dataset for training and 10% for testing. In addition, we conducted a 10-fold Cross-Validation (CV) on the training data to better evaluate the performance of the ML classifiers. According to our experiments, based on CV accuracy, the Random Forest Classifier provided the best performance on our dataset, achieving 91.0% accuracy (f1-score) on test data while 10-fold CV accuracy was 90.5%.

We also evaluated LeakyPick as described in Sect. 5.2.2 on the same real world dataset in order to compare its performance to the ML-based approach. The results are displayed in Figure 8, showing the ROC curves for both approaches on the Google Home, Siri Home Pod and Alexa Echo devices. For p-value threshold 0.43 LeakyPick

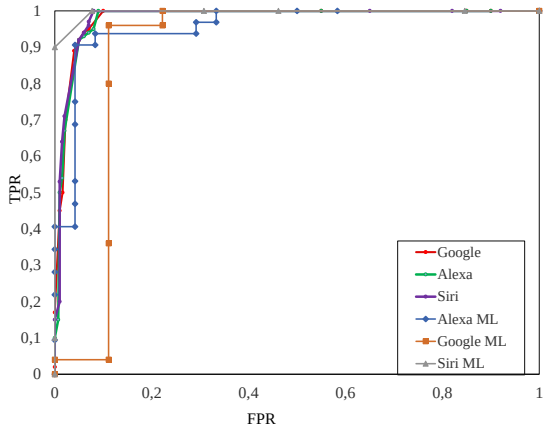

Figure 8: ROC curves of the ML-based and LeakyPick approaches on the real-world dataset

achieves a TPR of 93% with a simultaneous FPR of 7% averaged over all devices, compared to a best-case TPR of 95% and FPR of 9.7% for the ML-based classifier for Alexa Echo Dot. We also found that models are not transferable between voice assistants. For example, training on Alexa voice traffic and using the model to identify Siri voice traffic had around 35% precision.

As our evaluation results in Figure 8 show, ML-based models are indeed able to detect audio events based on the traffic the devices send out of the network. However, the evaluation also shows that similar or even better performance can be achieved using a device-agnostic approach as taken by LeakyPick.

Since applying this kind of a profiling approach requires dedicated traffic models to be trained and included in the system for each device type considered, its practicality in real-world scenarios is questionable. Due to the very large and ever-growing number of different types of voice-activated devices, this approach seems impractical. The approach adopted in LeakyPick can achieve similar performance without the need to employ pre-trained device-type-specific detection models for audio event detection, providing it much wider applicability in a wider range of environments with diverse audio-activated device types.

5.5 RQ5: Identifying Unknown Wake-Words

To demonstrate LeakyPick's ability to identify unknown wake-words, we performed a systematic experiment with Amazon's Alexa-enabled Echo Dot. As voice assistants are conceptually similar, we believe the results can be generalized to other voice-controlled devices. We configured the Echo Dot to use the standard "Alexa" wake word (other options include "Computer," "Amazon," and "Echo"). The experiment played different audio inputs, waiting for two seconds for the visual light-ring indicator of the device to light up, indicating the device reacted to the input. For each tested audio input, we recorded the number of attempts that triggered a reaction. Recall from Section 2 that Alexa-enabled devices have two states of detection: (1) an offline model on the device, and (2) an online model. We classify a word to be mistaken as a wake-word when the word triggers at least the offline model, since this transmits recorded audio to the cloud.

Results. The Alexa-enabled Echo Dot reliably reacted to 89 words across multiple rounds of testing. Table 5 (Appendix) shows the full list of words. To estimate the phonetic distance between these words and the true wake-word, we used the Metaphone algorithm [35] to convert the words into a phonetic alphabet based on their pronunciation. The resulting words were then compared with the Levenshtein distance to "Alexa." Among the 89 words, 52 have a phonetic distance of 3 or more. We found that 3 words had a phonetic distance of 0, 9 a distance of 1, 25 a distance of 2, 29 a distance of 3, 14 a distance of 4, 2 a distance of 5 and 6, 4 of 7 and one even a distance of 8. These distances shows that the Echo Dot reliably reacted to words that are phonetically very different than "Alexa." Based on these findings, it is unsurprising that Alexa-enabled devices are often triggered unintentionally, leading to private conversations and audio being transmitted outside the user's home.

6 DISCUSSION

Burst Detector: A malicious audio bug device whose sole purpose is to eavesdrop on a victim may use extensive lossy audio compression to keep the traffic rate below the detection threshold of 23kbit/s used in this paper. However, such audio may not be suitable anymore for automated voice recognition as many features of the voice are deleted or exchanged with noise which impairs the scalability of such an attack dramatically. However, our statistical probing approach would still detect a significant difference in the traffic and detect the sent audio.

Statistical Probing: As mentioned in Section 2, attacks aiming to issue commands to a victim's voice assistant can be detected by LeakyPick. To achieve that, increasing the time traffic samples are acquired as well as disabling audio probing is needed. By disabling the audio probing mechanism, every invocation of the device must be done by an external entity (e.g., the user or an attacker). By increasing the sample size, it is also possible to distinguish reliably between an actual invocation and background traffic spikes, even without the knowledge of when audio is played or not as the p-values are different for an invocation and background noise (cf. Figure 4). With this tweak, LeakyPick would also be able to warn the user of such attacks.

Countermeasures against Devices sending Audio: Depending on whether LeakyPick acts as the gateway of the home network or is sniffing passively the (encrypted) Wi-Fi traffic, there are different approaches to prevent a device from recording and sending audio without the user's permission. If our device is replacing the gateway, traffic identified as containing audio recordings can be simply dropped at the network layer. If our device can only passively sniff encrypted MAC layer traffic, inaudible microphone jamming techniques could be used to prevent the device from recording unsuspecting users private conversations [30, 37, 38, 42, 50].

Wake-Word Identification: We found that some of the identified wake-words for Alexa are only effective if spoken by Google's TTS voice, and that we were unable to replicate the effect when spoken by a human (e.g., the word "letter"). We believe this may result from features that differ between the TTS service audio file and natural

voice. However, the goal of the experiment was to demonstrate the extent to which words are incorrectly interpreted as wake-words, rather than determining the actual words triggering incorrect wake-word detection. There may also be wake-words, sounds, or noise our approach could not find. We are currently investigating whether wrongly recognized wake-words could be used to attack voice assistants and other voice recognition applications.

7 RELATED WORK

Existing works discussing detection of inadvertent data transmissions out of a user network so far have focused on IP cameras. To the best of our knowledge, there are no published approaches for detecting outgoing audio traffic for voice assistants and other audio-activated devices, in particular approaches utilizing audio probing. We are also not aware of publications utilizing audio probes to determine if devices react to audio inputs.

In the following we therefore focus on existing attacks on voice assistants as well as traffic analysis approaches for IoT device identification and IP camera detection. Moreover, we also review approaches to microphone jamming, which can be utilized by LeakyPick to prevent microphone-enabled IoT devices to record meaningful audio when the user is not aware of it.

IP Camera Detection: IP camera detection approaches usually extract features from packet headers. Wireless cameras in operation continuously generate camera traffic flows that consist of video and audio streams. The resulting traffic patterns of IP cameras are likely to be different and easily distinguishable from that of other network applications. Furthermore, to save bandwidth, IP cameras utilize variable bit rate (VBR) video compression methods, like H264. Because of the use of VBR, by changing the scene the camera monitors a change in the bitrate of the video can be enforced. Finally, by correlating scene changes and traffic bitrate changes cameras, monitoring can be identified.

Cheng et al. [11] propose using the person being monitored to change the scene by letting them move around. The resulting traffic is then classified using machine learning. Similarly Liu et al. [29] focus on altering the light condition of a private space to manipulate the IP camera's monitored scene. The resulting stream also changes its bitrate and can therefore be distinguished from non-altered streams, e.g., by using the statistical t-test. The above proposals are completely orthogonal to our approach, as they are customized for cameras. In addition, they make assumptions that are not applicable to microphone-enabled IoT devices, e.g., utilizing a variable bit rate encoding (VBR) and continuous data transmission.

Traffic Analysis: Numerous classification techniques have been proposed to learn the behavior of IoT devices, distinguishing and identifying IoT devices based on their traffic profile. Sivanathan et al. [41] use network traffic analysis to characterize the traffic corresponding to various IoT devices. They use the activity pattern (traffic rate, burstiness, idle duration) and signalling overheads (broadcasts, DNS, NTP) as features to distinguish between IoT and non-IoT traffic. However, the approach requires training.Nguyen et al. [33] propose an autonomous self-learning distributed system for detecting compromised IoT devices. Their system builds on device-type-specific communication profiles without human intervention nor labeled data which are subsequently used to detect anomalous deviations in devices' communication behavior, potentially caused by malicious adversaries. However, the proposed approaches are not applicable to our system, as these proposals focus on detecting anomalous behavior not consistent with benign device actions. In contrast, our goal is to detect benign actions in response to audio events, which may or may not be falsely detected. Also our approach does not require the system to identify IoT devices based on their traffic.

Eavesdropping Avoidance: Microphone, and more specifically, voice assistant jamming attacks have been proposed by several prior works. Roy et al. [37] present an approach for inaudibly injecting audio to jam spy microphones using ultrasonic frequencies and ultrasound modulated noise. As it is inaudible to humans, the jamming is not interfering with human conversations. Zhang et al. [50] build upon this work to inaudibly inject commands into voice assistants, demonstrating that voice assistants and possibly other commodity IoT devices are susceptible to the proposed ultrasonic control. Mitev et al. [30] further build upon these findings to precisely jam human voice and inject recorded voice into voice assistants. As discussed in Section 6, inaudible jamming approaches could be used by LeakyPick to prevent a device from recording meaningful audio when the user is not aware of it. In future work we aim to use these approaches as an additional component of LeakyPick, further increasing the privacy gains of our approach.

Voice Assistant Attacks: Voice assistants using voice recognition are fairly new and many security and privacy aspects are still to be improved.The common goal of such attacks is to control the voice assistant of a user without him noticing.Diao et al. [14] present attacks against the Google Voice Search (GVS) app on Android. A malicious app on the smart phone can activate GVS and simultaneously play back a recorded or synthesized command over the built-in speakers which is then picked up by the microphone, to control the victim's voice assistant. Alepis et al. [3] extend upon this attack. They then proceed to use use multiple devices to overcome implemented countermeasures by showing that infected devices can issue commands to other voice-activated devices such as the Amazon Echo or other smart phones.

Vaidya et al. [46] present a method to change a recording of human voice so that it is no longer comprehensible by humans but still correctly recognizable by voice recognition systems. Carlini et al. [7] extended this work by presenting voice mangling on a voice recognition system where the underlying mechanics are known, resulting in a more precise attack. Since a mangled voice may alert nearby users, Schönherr et al. [40] and Yuan et al. [49] propose methods for hiding commands inside other audio files (e.g., music files) such that they are not recognizable by humans. Similarly, Carlini et al. [8] create audio files with similar waveforms, which Mozilla's DeepSpeech interprets as different sentences.

Voice assistant extensions have also been attacked. Kumar et al. [28] showed that utterances exist such that Alexa's speech-to-text engine systematically misinterprets them. Using these findings they proposed *Skill Squatting*, which tricks the user into opening a malicious Skill. Simultaneously, Zhang et al. [51] proposed using malicious Skills with a similarly pronounced or paraphrased invocation-name to re-route commands meant for that Skill.

These attacks show that an attacker is able to manipulate the interaction flow with a voice assistant by, e.g., issuing commands without the victim noticing, turning voice assistants into a potential privacy and security risk for the user. LeakyPick can warn the user if their voice assistant is under attack without him noticing it. When combined with eavesdropping avoidance (e.g., jamming), the attacks could be mitigated or even prevented.

8 CONCLUSION

As smart home IoT devices increasingly adopt microphones, there is a growing need for practical privacy defenses. In this paper, we presented the LeakyPick architecture that enables detection of smart home devices that stream recorded audio to the Internet without the user's consent. Conceptually, LeakyPick periodically "probes" other devices in its environment and monitors the subsequent network traffic for statistical patterns that indicate audio transmission. We built a prototype of LeakyPick on a Raspberry Pi and demonstrate an accuracy of 94% in detecting audio transmissions from eight different devices with voice assistant capabilities without any *a priori* training. It also identified 89 words that could unknowingly trigger an Amazon Echo Dot to transmit audio to the cloud. As such, LeakyPick represents a promising approach to mitigate a real threat to smart home privacy.

REFERENCES

[1] 2020. *Scikit-learn Python machine learning library.* https://github.com/scikit-learn/

[2] David W Aha, Dennis Kibler, and Marc K Albert. 1991. Instance-based learning algorithms. *Machine learning* 6, 1 (1991).

[3] Efthimios Alepis and Constantinos Patsakis. 2017. Monkey says, monkey does: security and privacy on voice assistants. *IEEE Access* 5 (2017).

[4] Amazon. [n.d.]. *Alexa Built-in Products with AVS - SpeechRecognizer.* https://developer.amazon.com/de-DE/docs/alexa/alexa-voice-service/speechrecognizer.html

[5] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001).

[6] Dell Cameron. 2018. *Hack Can Turn Robotic Vacuum Into Creepy Rolling Surveillance Machine.* https://gizmodo.com/hack-can-turn-robotic-vacuum-into-creepy-rolling-survei-1827726378

[7] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *Proceedings of the USENIX Security Symposium.*

[8] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944* (2018).

[9] G. Casella and B. Roger. 199. *Statistical Inference.* Duxbury, 2nd edition.

[10] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD).*

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

[12] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing* 307 (2018). https://doi.org/10.1016/j.neucom.2018.03.067

[13] Joseph Cox. [n.d.]. *Revealed: Microsoft Contractors Are Listening to Some Skype Calls.* https://www.vice.com/en$_{us}$/article/xweqbq/microsoft-contractors-listen-to-skype-calls

[14] Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang. 2014. Your voice assistant is mine: How to abuse speakers to steal information and control your phone. In *Proceedings of the ACM Workshop on Security and Privacy in Smartphones & Mobile Devices (SPSM).*

[15] DIODES Incorporated. [n.d.]. *FILTERLESS 3W CLASS-D STEREO AUDIO AMPLIFIER.* https://www.diodes.com/assets/Datasheets/PAM8403.pdf

[16] Education First. [n.d.]. *1000 most common words in English.* https://www.ef.com/wwen/english-resources/english-vocabulary/top-1000-words/

[17] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 96.

[18] Sarah Frier. 2019. *Facebook Paid Contractors to Transcribe Users' Audio Chats.* https://www.bloomberg.com/news/articles/2019-08-13/facebook-paid-hundreds-of-contractors-to-transcribe-users-audio

[19] Google. [n.d.]. *Learn about Nest Protect's automatic Sound Check test.* https://support.google.com/googlenest/answer/9242130?hl=en

[20] Google. [n.d.]. *Learn how Nest cameras and Nest Hello detect sound and motion.* https://support.google.com/googlenest/answer/9250426?hl=en

[21] Andy Greenberg. 2017. *This hack lets Amazon Echo 'remotely snoop' on users.* https://www.wired.co.uk/article/amazon-echo-alexa-hack

[22] Caroline Haskins. 2019. *Amazon Is Coaching Cops on How to Obtain Surveillance Footage Without a Warrant.* https://www.vice.com/en$_{us}$/article/43kga3/amazon-is-coaching-cops-on-how-to-obtain-surveillance-footage-without-a-warrant

[23] Lente Van Hee, Ruben Van Den Heuvel, Tim Verheyden, and Denny Baert. [n.d.]. *Google employees are eavesdropping, even in your living room, VRT NWS has discovered.* https://www.vrt.be/vrtnws/en/2019/07/10/google-employees-are-eavesdropping-even-in-flemish-living-rooms/

[24] Alex Hern. 2019. *Apple contractors 'regularly hear confidential details' on Siri recordings.* https://www.theguardian.com/technology/2019/jul/26/apple-contractors-regularly-hear-confidential-details-on-siri-recordings

[25] Hive. [n.d.]. Hive Hub 360. https://www.hivehome.com/products/hive-hub-360 (Accessed June 2020).

[26] Infochimps.com. [n.d.]. *350000 simple english words.* http://www.infochimps.com/datasets/word-list-350000-simple-english-words-excel-readable

[27] Business Insider. [n.d.]. *Google says the built-in microphone it never told Nest users about was 'never supposed to be a secret'.* https://www.businessinsider.com/nest-microphone-was-never-supposed-to-be-a-secret-2019-2

[28] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill squatting attacks on amazon alexa. In *Proceedings of the USENIX Security Symposium.*

[29] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting Wireless Spy Cameras Via Stimulating and Probing. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys).*

[30] Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. 2019. Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS).*

[31] David Monsees. [n.d.]. *More information about our processes to safeguard speech data.* https://www.blog.google/products/assistant/more-information-about-our-processes-safeguard-speech-data/

[32] Alfred Ng. 2019. *Alexa and Google Assistant fall victim to eavesdropping apps.* https://www.cnet.com/news/alexa-and-google-voice-assistants-app-exploits-left-it-vulnerable-to-eavesdropping/

[33] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Minh Hoang Dang, N. Asokan, and Ahmad-Reza Sadeghi. 2018. DÏoT: A Crowdsourced Self-learning Approach for Detecting Compromised IoT Devices. *CoRR* abs/1804.07474 (2018). arXiv:1804.07474 http://arxiv.org/abs/1804.07474

[34] TJ OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. 2019. HomeSnitch: behavior transparency and control for smart home IoT devices. In *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec).*

[35] Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language* 7, 12 (1990).

[36] Raspberry Pi Foundation. [n.d.]. *Raspberry Pi 3 Model B.* https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[37] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. 2017. BackDoor: Making Microphones Hear Inaudible Sounds. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys).*

[38] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. 2018. Inaudible voice commands: The long-range attack and defense. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI).*

[39] Alex Rudnicky. [n.d.]. *The CMU Pronouncing Dictionary.* http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[40] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2018. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. *arXiv preprint arXiv:1808.05665* (2018).

[41] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2017. Characterizing and classifying IoT traffic in smart cities and campuses. *Proceedings of the IEEE INFOCOM Workshop on Smart Cities and Urban Computing (SmartCity)* (2017).

[42] Liwei Song and Prateek Mittal. 2017. Inaudible voice commands. *arXiv preprint arXiv:1708.07238* (2017).

[43] Texas Instruments. [n.d.]. *Low-Power, Low-OffsetVoltage, Dual Comparators.* https://www.ti.com/lit/ds/symlink/lm393-n.pdf

[44] Kevin C. Tofel. 2018. *Here's why smart home hubs seem to be dying a slow, painful death.* https://staceyoniot.com/heres-why-smart-home-hubs-seem-to-be-dying-a-slow-painful-death/

[45] tp-link. [n.d.]. *150Mbps High Gain Wireless USB Adapter - TL-WN722N.* https://www.tp-link.com/en/home-networking/adapter/tl-wn722n/
[46] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT).*
[47] Vladimir Vapnik. 2013. *The nature of statistical learning theory.* Springer science & business media.
[48] Merriam Webster. [n.d.]. .
[49] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. 2018. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. *arXiv preprint arXiv:1801.08535* (2018).
[50] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS).*
[51] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2018. Understanding and Mitigating the Security Risks of Voice-Controlled Third-Party Skills on Amazon Alexa and Google Home. *arXiv preprint arXiv:1805.01525* (2018).

A FOUND WAKE-WORDS

Full results of testing the Alexa wake-word (Alexa) with words of the English language dictionary with 6 and 5 phonemes as well as some random words, is shown in Tab. 5. The results shown are the last round of 10 tests for each word. The left column shows the probability of the device being activated while replaying 10 times the word in question.

Table 5: Full results of testing the voice assistant Alexa with English words

Probability of activating Alexa	Wake-Word
2/10	alita, baxa, elater, hexer, liker, ochna, taxer
3/10	bertha, electroceramic, excern, oxer, taxir
4/10	electrohydraulic, electropathic, wexler
5/10	blacksher, electic, hoaxer
6/10	bugsha, elatha, elator, electrodissolution, electrostenolytic, eloper, eluted, fluxer, huerta, hurter, irksome, lecher, lefter, lepre, lesser, letter, licker, lipper, loasa, loker, lotor, lyssa, maloca, maxillar, melosa, meta, metae, muleta, paxar, rickner
7/10	alexy, crytzer, electroanalytical, hyper, kleckner, lecture, likker, volupte, wexner
8/10	electroreduction, hiper, wechsler
9/10	aleta, alexa, alexia, annection, elatcha, electre, kreitzer
10/10	alachah, alexipharmic, alexiteric, alissa, alosa, alyssa, barranca, beletter, elector, electra, electroresection, electrotelegraphic, elissa, elixir, gloeckner, lechner, lecter, lictor, lxi, lxx, mixer, olexa, walesa