Vasilios
Mavroudis*, Shuang Hao, Yanick Fratantonio, Federico Maggi, Christopher Kruegel, and Giovanni Vigna

On the Privacy and Security of the Ultrasound Ecosystem

Abstract: Nowadays users often possess a variety of electronic devices for communication and entertainment. In particular, smartphones are playing an increasingly central role in users' lives: Users carry them everywhere they go and often use them to control other devices. This trend provides incentives for the industry to tackle new challenges, such as cross-device authentication, and to develop new monetization schemes. A new technology based on ultrasounds has recently emerged to meet these demands. Ultrasound technology has a number of desirable features: it is easy to deploy, flexible, and inaudible by humans. This technology is already utilized in a number of different real-world applications, such as device pairing, proximity detection, and cross-device tracking.

This paper examines the different facets of ultrasound-based technology. Initially, we discuss how it is already used in the real world, and subsequently examine this emerging technology from the privacy and security perspectives. In particular, we first observe that the lack of OS features results in violations of the principle of least privilege: an app that wants to use this technology currently needs to require full access to the device microphone. We then analyse real-world Android apps and find that tracking techniques based on ultrasounds suffer from a number of vulnerabilities and are susceptible to various attacks. For example, we show that ultrasound cross-device tracking deployments can be abused to perform stealthy deanonymization attacks (e.g., to unmask users who browse the Internet through anonymity networks such as Tor), to inject fake or spoofed audio beacons, and to leak a user's private information.

Based on our findings, we introduce several defense mechanisms. We first propose and implement immediately deployable defenses that empower practitioners, researchers, and everyday users to protect their privacy. In particular, we introduce a browser extension and an Android permission that enable the user to selectively suppress frequencies falling within the ultrasonic spectrum. We then argue for the standardization of ultrasound beacons, and we envision a flexible OS-level API that addresses both the effortless deployment of ultrasound-enabled applications, and the prevention of existing privacy and security problems.

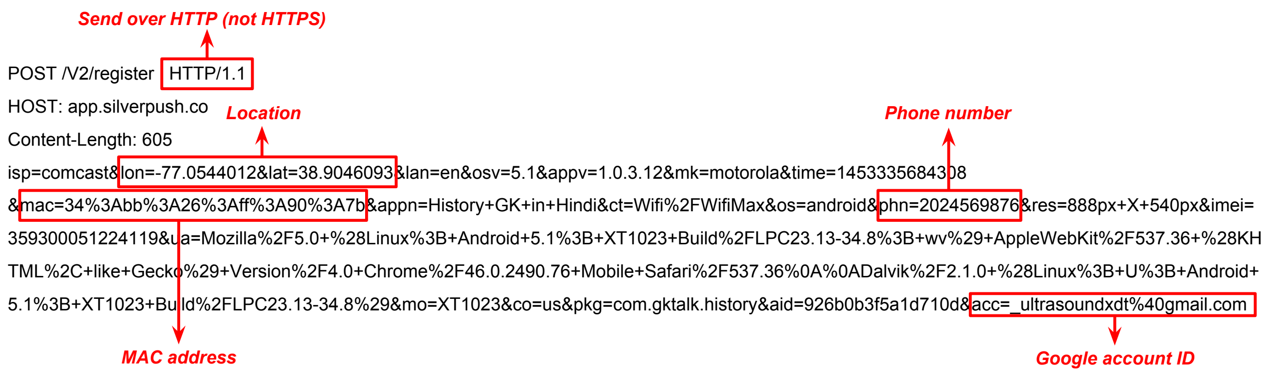
Keywords: Ultrasounds, Deanonymization, Privacy Violation, Cross-device Linking.

1 Introduction

Modern users have been increasingly relying on multiple devices and frequently switch between them in their daily lives: Browsing the Internet from computers, playing mobile games on tablets, or watching shows on televisions. A recent survey has shown that a single user carries 2.9 electronic devices on average [44]. Among the others, smartphones play a prominent role since users carry them everywhere they go, and they are often used to control other devices. The trend of engaging with multiple digital platforms brought to the demand for new technologies to efficiently link multiple devices of a user.

To meet this demand, a novel set of technologies based on ultrasounds have recently emerged. These technologies leverage the ultrasonic frequency range to transmit information in the form of inaudible sound *beacons* to identify and link two (or more) devices belonging to the same users. The ultrasounds are inaudible to humans, and they can be emitted and captured by regular speakers and microphones, needing no extra requirement for deployment. The benefits of ultrasounds make the technologies gain increasing popularity. For example, the Google Cast app [3] uses ultrasounds to pair the user's mobile device to her Google Chromecast, without requiring the devices to have access to the same wireless network. The ultrasound-based techniques are also widely used to track the in-store posi-

***Corresponding Author: Vasilios Mavroudis:** University College London, E-mail: v.mavroudis@cs.ucl.ac.uk
Shuang Hao: UC Santa Barbara, E-mail: shuanghao@cs.ucsb.edu
Yanick Fratantonio: UC Santa Barbara, E-mail: yanick@cs.ucsb.edu
Federico Maggi: Politecnico di Milano, E-mail: federico.maggi@polimi.it
Christopher Kruegel: UC Santa Barbara, E-mail: chris@cs.ucsb.edu
Giovanni Vigna: UC Santa Barbara, E-mail: vigna@cs.ucsb.edu



```
POST /V2/register HTTP/1.1
HOST: app.silverpush.co
Content-Length: 605
isp=comcast&lon=-77.0544012&lat=38.9046093&lan=en&osv=5.1&appv=1.0.3.12&mk=motorola&time=1453335684308
&mac=34%3Abb%3A26%3Aff%3A90%3A7b&appn=History+GK+in+Hindi&ct=Wifi%2FWifiMax&os=android&phn=2024569876&res=888px+X+540px&imei=
359300051224119&ua=Mozilla%2F5.0+%28Linux%3B+Android+5.1%3B+XT1023+Build%2FLPC23.13-34.8%3B+wv%29+AppleWebKit%2F537.36+%28KH
TML%2C+like+Gecko%29+Version%2F4.0+Chrome%2F46.0.2490.76+Mobile+Safari%2F537.36%0A%0ADalvik%2F2.1.0+%28Linux%3B+U%3B+Android+
5.1%3B+XT1023+Build%2FLPC23.13-34.8%29&mo=XT1023&co=us&pkg=com.gktalk.history&aid=926b0b3f5a1d710d&acc=_ultrasoundxdt%40gmail.com
```

Fig. 1. An example of an insecure, non-standardized implementation of the ultrasound-enabled applications. HTTP POST request sent by an ultrasound-enabled app that we found on the Google Play Store, violating the user's privacy and security practices: (1) The framework surreptitiously collects the user's private information, including the Google account ID, phone number, geolocation, the device MAC address, and other information; (2) The data is sent insecurely over HTTP (instead of HTTPS).

tion and behavior of the customers to serve relevant ads; this approach is used by a number of companies, such as Lisnr [30], ShopKick [39], and CopSonic [18]. Another important application is cross-device tracking, used in the context of advertisement: in this case, ultrasound beacons are embedded into websites or TV ads and they get *picked up* by advertisement SDKs embedded in smartphone apps. This technique allows the ad company to identify the different devices of an audience and subsequently push targeted ads. Such advertising techniques are at the core of the business model of several recently established companies, such as SilverPush [42], Signal360 [40], and Audible Magic [14]. Even though these companies are relatively recent, they already received several million dollars of funding, and some of these technologies have already been adopted by many Android apps (available at the Play Store, and installed by millions of users [15] [20]), for example to increase user engagement in large sport events [10].

This paper explores the security and privacy implications of these technologies. First, we point out that all these technologies clearly violate the principle of least privilege. In fact, while these technologies would only require access to the ultrasound spectrum, they are forced to ask for full microphone access. Moreover, many design choices can lead to devastating violations of the users' privacy. To make things worse, ad companies are not only using these emerging technologies, but they are *combining* them with existing tracking techniques. Just to give an example, Figure 1 shows an HTTP POST request that we captured from an ultrasound-enabled app currently hosted on the Play Store, History GK [6]. Figure 1 shows that the framework surreptitiously collects a substantial amount of the user's private information, including the Google account ID, phone number, geolocation, and the device MAC address, and, in addition, it transfers the data insecurely over HTTP (instead of using HTTPS).

In addition to the privacy issues raised in a recent workshop hosted by the Federal Trade Commission (FTC) [23], our work also uncovers a variety of security shortcomings in the ultrasound-based technology, and identifies the possibility to carry on stealthy attacks. In particular, we show that third parties can launch attacks based on inaudible beacons: (1) *deanonymization attacks*, to unmask users of anonymity networks (such as Tor or virtual private networks) and provide a new way to connive at surveillance; (2) *beacon injection attacks*, to pollute the results of advertising algorithms; and (3) *information leakage attacks*, to infer other users' on-line activities and interests, sabotaging the users' privacy.

These attacks exploit inherent vulnerabilities of current deployments of the ultrasound-based technology, and can thus be leveraged to launch stealthy attacks against unsuspecting users, even if they did not install any malicious app. Consequently, these attacks can affect a much larger number of users when compared to traditional malware (e.g., malicious eavesdropping apps), which has to both evade the detection of automatic analysis systems and lure the user into installing the malicious app.

To mitigate the risks posed to users, in this paper we further propose and develop several security enhancements. First, we propose a browser extension to allow a user to selectively suppress all ultrasound frequencies. We also developed an analogous solution for the Android framework: we developed a new permission that allows for fine-grained control in providing app access to the ultrasound spectrum. The user is thus empowered with an opt-out option through a system-level mechanism.

Finally, we argue for the standardization of the ultrasound beacons format, and we envision a new OS-level API (similar with the Bluetooth low energy beacons [4] [5] [2] in Android) that implements in a single, trusted place the functionality to detect and decode beacons. This API would be implemented by a privileged process (like a system process on Android). The existence of such API would provide several important benefits: an app that wants to implement ultrasound-based mechanisms only needs to require access to this API, and it does not require anymore access to the device's microphone. Thus, the user's sensitive data is not exposed to third-party apps in the first place. Another advantage of such API is that it would act as a central place to better and more easily detect and monitor abuses based on ultrasounds.

In summary, our paper makes the following contributions:

– We perform the first detailed study to examine the security and privacy implications of emerging ultrasound-based technologies.
– We point our attention to a clear violation of the principle of least privilege and we introduce and demonstrate a series of attacks that can be performed against ultrasound-enabled systems. These attacks include user deanonymization, beacon injection, and information leakage.
– We design a set of countermeasures, such as a new browser extension and a new Android permission that creates a system-enforced opt-out mechanism for privacy-concerned users.
– We argue and propose that the format of ultrasound beacons should be standardized, which would allow the implementation of an OS-level API to make these emerging technologies secure by design.

We will open-source the attack prototypes and the countermeasures that we have proposed and implemented.

2 Key Concepts

This section discusses some key concepts and techniques commonly used in the advertising industry to track users and monetize the collected information.

User Profiling. Behavioural targeting is a common practice in the advertisement industry, where the user's past activity is leveraged to build a personalized profile. This profile, along with other demographic data (e.g., gender, location), is then used by ad networks to serve targeted advertisements. For instance, they are often used in *real-time bidding* (RTB), which enables ad buyers to compete on a per-impression basis, and adjust their bids depending on the specific characteristics of an individual user (e.g., jewellery ads could have different prices depending on the user's gender) [34]. RTB is becoming the industry standard when buying and selling ads, and a recent study shows that more than 80 percent of North-American advertisers have switched to RTB when buying ad impressions since 2011 [51]. However, this fine-grained data collection comes also with severe security implications, and even though the data are not directly provided to the ad buyers, it has been shown that information leakage [34][17][16] is very hard to prevent.

Device Pairing. Device pairing enables the establishment of a link between two devices. Such links can be used for data transmission (e.g., bluetooth pairings), or simply to prove device proximity (e.g., Google Chromecast). A pairing can be performed in a number of ways, including Bluetooth (e.g., headset pairing), ultrasounds (e.g., Google Chromecast), and simple PIN re-entry.

uBeacons. Ultrasound beacons (uBeacons, in short) are high-frequency audio tags that can be emitted and captured by most commercial speakers and microphones, and are not audible by humans. These beacons encode a small sequence of characters and symbols, which in most cases serve as an identifier used to fetch content from an external server or to pair two devices together. Currently, there is no standard or specification, and hence each company designs its own beacon encoding formats and protocols. As a result, there are multiple incompatible frameworks, providing varying levels of security.

From a technical perspective, an ultrasound beacon has a duration of only few seconds, usually around 5. However, the exact method of encoding the information varies greatly depending on the requirements of the application (e.g., range requirements, information volume). In the great majority of cases, the spectrum between 18,000Hz and 20,000Hz is divided in smaller chunks, and each one corresponds to a symbol or character. For instance, for an application using a chunk size of 75Hz, an amplitude of 18,000Hz could correspond to 'A', while 18,075 to 'B'. The spectrum plot of such a beacon can be seen in Figure 2. The beacon emitter then plays each high-frequency tone for a predetermined amount of time, with one second being the most common setting. Moreover, beacons are usually built following a number of error-prevention rules. For example, two rules often seen are: (1) each beacon must start with a specific pre-defined character, and (2) subsequent characters must not be the same. For the decoding of a beacon, frameworks usually apply a fast Fourier transform

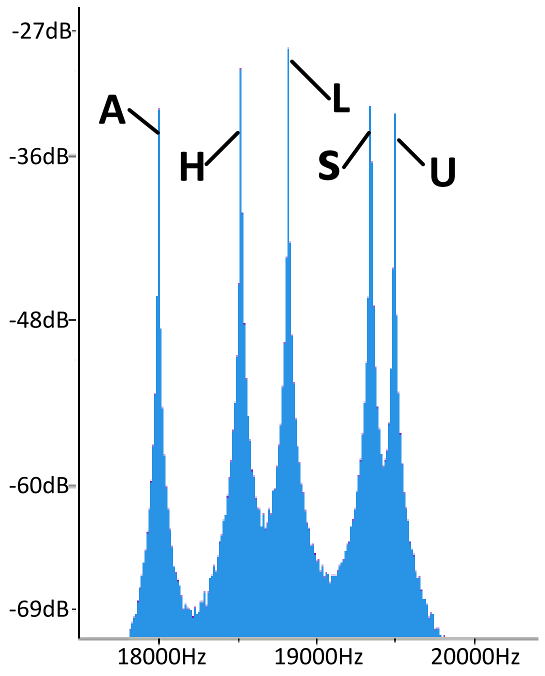


Fig. 2. Spectrum plot of beacon encoding five characters.

and Goertzel's algorithm to each incoming signal. This allows them to distinguish the individual frequencies and to then retrieve the original characters/symbols. Once a valid beacon is found, the identifier is extracted and submitted to the company's backend.

Ultrasound Tracking Framework. Ultrasound tracking frameworks are software components released by tracking service providers. These frameworks enable their clients (e.g., an Android app) to perform user/device tracking based on (ultrasound) beacons. Such a framework is usually provided as an Android library to be incorporated in the app owned by the client. This library provides proprietary methods for supporting all uBeacon-related operations such as discovery, parsing, and validation. The exact realization of uBeacons and the operations differ between the frameworks, and the only common requirement is the need for access to the device's microphone. The next section discusses several examples of these frameworks and their particular use cases.

3 Ecosystem Overview

In this section, we present the ultrasound techniques implemented in actual products, and we discuss their technical details and capabilities. We organize the discussion by grouping these techniques in two sections, depending on what their objective is: proximity tracking techniques, and cross-device tracking techniques.

3.1 Proximity Tracking

Proximity tracking is an emerging approach of determining the location of a user, and has numerous applications in *device pairing* and *proximity marketing*. More specifically, for implementing device pairing, a device A may use a uBeacon to broadcast a random PIN to nearby devices, and trigger a pairing once a device B submits the correct PIN back (usually through the Internet). On the other hand, proximity marketing is used for serving location-specific content and tracking in-store user behaviour. It is used in various places (e.g., casinos, museums, retail, airports) and is based on small distance transmission technologies such as Wifi, bluetooth, or ultrasounds. While the exact implementation varies, the main idea is that the user installs the company's application on her mobile device and receives, for example, notifications and discounts for products. Proximity marketing techniques relying on ultrasounds recently gained a tremendous traction in the last few years, since an increasing number of companies is developing supporting frameworks [3, 14, 18, 30, 40, 47].

Google Cast. Google Cast [3] is a popular app developed by Google, and it reportedly utilizes ultrasound beacons to facilitate device pairing between mobile devices and Google Chromecast [26]. Chromecast is a digital media player that, among others, enables mobile devices to stream content on a television or on an audio system. In this case, ultrasound beacons are used to prove physical proximity to Google Chromecast, as they do not penetrate through walls. The pairing process takes place in four steps: Initially, Google Chromecast broadcasts a Wifi beacon that makes itself visible to all nearby mobile devices running the Google Cast app (note that the devices do not need to be connected to the same Wifi to receive such Wifi beacons). Subsequently, the user sends a pairing request through the app, and Chromecast responds with a uBeacon carrying a 4-digit code. Then the Google Cast app gains access to the device's microphone, captures the uBeacon, and authenticates to Chromecast by using the 4-digit code as the password [26]. It should be noted that the pairing process is always initiated by the user, and an alternative manual pairing method is also supported.

Lisnr framework. The Lisnr framework [30] is another ultrasound framework aimed towards proximity marketing and location-specific content. This framework has been incorporated in various applications already. For instance, an American football team uses the framework in its official app (with hundreds of thousands downloads [7]) to deliver content to fans in its home stadium [41]. Another such app is "Made in America Festival" Android App [1] (also available for iPhone), whose core objective is to act as a real-time information stream for the audience of a music festival. In both cases, ultrasounds were chosen since they make use of the existing audio infrastructure and require no additional equipment.

From a technical perspective, the Lisnr framework implements all the necessary methods to capture uBeacons, and fetches location-specific content from the service provider. Upon execution, the apps run on the background and periodically access the device's microphone to listen for uBeacons. This is because the user is not expected to keep the app active for the whole duration of the events (e.g., games, concerts). Once a uBeacon is captured, the framework parses it, and extracts its content. If the content is a message, it will get displayed on the device screen, while if it is an identifier, then an online server is reached to fetch the resource the identifier corresponds to. This is due to the nature of uBeacons, which is prone to transmission errors and cannot reliably transfer large volumes of data. For users that decide that they do not want to receive constant notifications, some applications provide an option to deactivate the default listening behaviour. However, this is an application-specific feature, and is not mandated by the Lisnr framework.

Shopkick. Another real-world deployment of proximity marketing using ultrasounds is the shopping application Shopkick [39] (available on Google Play Store). Shopkick aims to provide incentives to its users to visit and purchase products from specific stores and brands. To realise this, the app is listening for ultrasound beacons (i.e., walk-in tones) emitted by speakers installed in businesses and stores cooperating with Shopkick [38]. When a user visits such a store, the app gets notified and reward points are credited to the user's account. These points can be later spent by the user to receive discounts or products at a reduced price.

To realise this functionality, Shopkick has developed its own framework and implementation of uBeacons. In particular, Shopkick uBeacons contain a unique identifier of the store and the exact in-store location. When the app captures such a beacon, it extracts the encapsulated identifiers, and submits them to the Shopkick servers, along with the user information. Then, the company's backend verifies the validity of the "walk-in" (see Section 5.2), and credits the user with the points. In contrast to the majority of the other implementations, the user must manually trigger the app to gain access to the microphone and listen for uBeacons. This is possible in this particular case because the user is strongly incentivized to use the app, but would result in very low usage rates in scenarios where the user is not directly rewarded.

3.2 Cross-device Tracking

Various cross-device tracking (XDT) techniques are currently employed by many major advertisement networks to track users across different devices. These techniques offer various degrees of precision, depending on the application and the available resources. For instance, when few resources are available, *probabilistic XDT techniques* are usually applied, which, in most cases, involve fingerprinting and have mediocre accuracy. On the other hand, *deterministic XDT techniques* achieve much higher accuracy, but are often cumbersome. For example, the most common deterministic technique requires the user to sign in the account of the advertiser's service from all the devices that the user owns.

Despite falling into the deterministic category, ultrasound cross-device tracking (uXDT) does not suffer from the applicability problems that other deterministic techniques come with. uXDT is the most sophisticated use of uBeacons found in products currently available on the market. Figure 3 illustrates the entities participating in the ultrasound-based mobile advertising ecosystem and their interactions. At first, the *advertising client* sets up a new advertising campaign and provides the ads to the *uXDT provider* (❶). The advertising client is a company or individual that is interested in promoting services or products (e.g., a supermarket chain), while the *uXDT provider* is a company that provides the infrastructure for user-tracking. Subsequently, the uXDT provider generates a unique inaudible *beacon* $\boldsymbol{b}$ and associates it with the client's campaign (❷). Once the beacon is generated, the uXDT service provider forwards it along with the client's ads to the *content provider* (❸), which produces content that the user is interested in, and can take many different forms (e.g., TV station, news website). Note that, for efficiency, the same beacon can be pushed to multiple content providers.

Next, the user uses one of her devices (e.g., a laptop) to request a resource that is returned by the *content provider* along with the beacon $\boldsymbol{b}$ (❹). Once the resource is rendered by the user's browser, the beacon $\boldsymbol{b}$ is emitted, and it is then captured by the uXDT-enabled device (e.g., a smartphone) owned by the user (❺). In the next stage, the uXDT framework embedded in the uXDT-enabled device reports the beacon ***b*** to the backend of the uXDT service provider. At this point, the advertisement framework is able to show the user an ad that is targeted to the user's interests (❻). During the last steps, the uXDT device displays those ads that match exactly the interests of the user (❼), attracts the user to the advertising client's store, and eventually converts it to a purchase (❽).

As said, uXDT is the most advanced use of uBeacons we have seen so far, as it requires both sophisticated infrastructure (e.g., profiling algorithms processing millions of submissions) and a network of publishers who incorporate beacons in their ads/content. Hence, very few companies are able to provide uXDT services. In addition to this, since the current form of uXDT techniques are controversial (e.g., [24][15]), companies are very secretive and very rarely publicly advertise that they are using the technology or share their APIs and SDKs.

SilverPush uXDT framework. Here we examine one of the most widespread uXDT frameworks, developed by SilverPush, which as of April 2015 was tracking more than 18 million devices [15]. Since the framework and its specifications were never made public, we reversed-engineered the History GK application [6], which is one of the apps incorporating the SilverPush uXDT framework.

The framework comprises of two modules: the beacon detection module and the data reporting module. The *beacon detection module* listens for ultrasound beacons with duration of 2 seconds, every 20 seconds. Any captured uBeacon is then decoded to a string of characters and its integrity is verified. To increase the efficiency of this module, the app is added to the boot sequence of the device, so that it can actively listen through the microphone for beacons (in the background), even when the application has not been "manually" started by the user. Once a valid beacon is captured, the *data reporting module* creates an event report and submits it to the company's backend using an unencrypted HTTP request. This report contains the Android ID of the device and the received uBeacon. If no Internet connection is available at the time, the event is stored and reported later. The data reporting module is also responsible for registering new devices in the uXDT ecosystem. This process takes place when the app is executed for the first time: the framework extracts a wealth of identifiers including the user's phone number, longitude and latitude, the IMEI of the device, the Android ID, and the Google account ID (email address), and reports them to the company's backend (Figure 1). These information are used to (1) build and maintain the user profile, (2) display targeted ads, and (3) as parameters for real-time bidding auctions (e.g., targeting users in a specific city). The other "actor" of the uXDT ecosystem, the advertising client, can then choose to cross-target these users and display follow-up ads on all their devices. It is noteworthy that, from our analysis, the only way to opt-out of uXDT is to completely remove the app. Moreover, the user is not notified that this framework is installed and actively used, and so she is not aware that the microphone is repeatedly activated to scan for uBeacons.

To better understand the market penetration of the specific framework, we conducted a broad search and, in combination with other sources (e.g., [20]), identified a few dozen applications using it. The download counts for the most popular of these ranged from several hundreds thousands to few millions (e.g., [9]). However, in subsequent searches we noticed that the number of applications incorporating the specific framework reduced. This likely happened for two reasons: due to the community backslash forcing developers to remove the framework [15, 23, 36, 37], and due to Silverpush reportedly withdrawing its product from the US market.

4 Privacy & Security Considerations

This section describes a few high-level considerations that affect the security and privacy of the ultrasound ecosystem. We postpone the discussion of actual vulnerabilities and attacks to the next section.

One main concern that we identified is that any app that wants to employ ultrasound-based mechanisms needs to require full access to the device's microphone. This is an obvious violation of the least privilege principle, as the app has also access to all audible frequencies. We note that this is the case not because the developers of the ultrasound frameworks were not cautious, but because there is currently no *safe* way to implement this. In fact, current versions of the Android framework do not expose any mechanism that allows fine-grained control over the device's microphone.

This aspect has several negative repercussions. A malicious developer could claim access to the microphone for ultrasound-pairing purposes, and then use it to spy on the user (e.g., to record the audio). On the other hand, developers of benign apps are also negatively affected: developers who want to use ultrasound-based technologies risk to be perceived as "potentially malicious" by the users. For this reason, users would hesitate to install apps requiring unconstrained access to the device's microphone. To make things worse, there are wide discrepancies between the

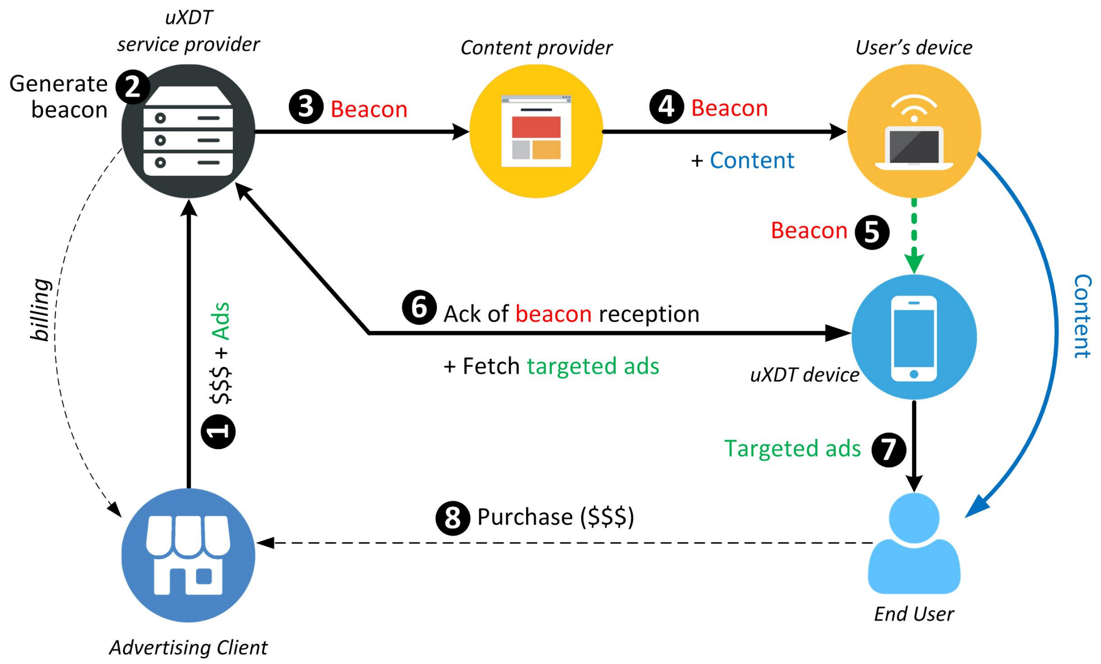


Fig. 3. Interaction between the entities of the mobile advertising ecosystem during a typical cross-device tracking scenario.

practices followed by companies when it comes to informing the users and providing opt-out options. For instance, in some cases (e.g., SilverPush) no notice or opt-out option is given to the user (apart from the mandatory microphone permission request).

There are also several practicality concerns related to this technology. Currently, each app featuring ultrasound-based capabilities performs a periodic *polling* on the device's microphone. These attempts at decoding uBeacons heavily strain the battery, and they prevent other apps from using the microphone. For instance, during our experiment, the official Android camera crashed several times when another app was listening for uBeacons in the background (e.g., SilverPush framework). To make things worse, a device running two or more ultrasound-enabled apps suffers from even heavier battery drainage, and the ultrasound frameworks race each other for access to the microphone, making it essentially unusable.

We believe that all these factors hurt the ultrasound ecosystem and that they significantly limit the diffusion and impact these emerging technologies can have.

5 Vulnerabilities & Attacks

This section introduces a number of attacks against one or more actors of the ultrasound ecosystem. These attacks exploit vulnerabilities inherent to ultrasound beacons that are not just simple, easily fixable implementation bugs. The attacks, summarized in Table 1, are relevant in the scenario where a user has an ultrasound-enabled app installed on her device and the ultrasound frameworks (and the other parties in the ecosystem) provide the functionalities they advertise.

Beacon Traps. We now define the concept of *beacon traps*. A beacon trap is a technique that an attacker can use to inject a uBeacon so that it is captured by the user's ultrasound-enabled device. A naïve way to do this is to play the uBeacon while the user is in physical proximity. However, relying on physical proximity is not always practical. To overcome the limitation, an attacker can use a small snippet of code that, when loaded in a browser, would automatically reproduce one or more attacker-chosen ultrasound beacons. For this technique to be effective, the adversary needs to attach the snippet to a resource that is accessed by the user. For example, an attacker could set up an innocuous-looking web page that, once visited, would play an audio beacon in the background. An attacker could also use existing XSS vulnerabilities present in a benign website to redirect the user to an attacker-controlled web page. Alternatively, an attacker could inject beacon-playing JavaScript (or HTML) in the users' traffic by mounting a man-in-the-middle attacks or by setting up a (malicious) Tor exit node [32, 48]. As a last example, an attacker could also send the user an audio message that stealthily embeds a uBeacon—when the message is played, the audio beacon

Attack	Goal	Attacker capabilities
Deanonymization	Retrieve identifying information (e.g., the true IP address), especially targeting users from anonymity networks	◇ Set up an ad campaign with a ultrasound service provider ◇ Lure victims to beacon trap ◇ Access to PII collected by the ultrasound service provider
Beacon injection	Pollute the user's profile maintained by the service provider	◇ Set up an ad campaign with a ultrasound service provider, or ◇ Obtain valid beacons from existing campaigns ◇ Stay in physical proximity or lure victims to beacon trap
Information leakage	Obtain/infer information about the users' interest	◇ Stay in physical proximity or lure victims to beacon trap

Table 1. Summary of the attacks that can be launched by exploiting an ultrasound tracking deployment.

is captured by the ultrasound-enabled device, which would then handle it as every other valid beacon (e.g., a uXDT app would *report* the beacon back to the uXDT backend).

5.1 Deanonymization Attack

The first attack we discuss is the *deanonymization attack*. This attack allows an adversary to deanonymize one or more users of an anonymity network or system (e.g., Tor, VPN). For example, this attack could allow a journalist to uncover the identity of a whistle-blower who uses Tor to send and share highly confidential documents. To perform the attack, the journalist would just need to convince (by social engineering or any other means) the whistle-blower to visit a hidden service (hosted on an .onion domain). At this point, the whistle-blower launches the latest version of the Tor browser and visits the hidden service. However, unbeknownst to the whistle-blower, the journalist has set up the hidden service page so that it emits inaudible beacons (i.e., a beacon trap). These beacons get captured by the ultrasound-enabled app running on the non-anonymized victim's device, which then forwards them to service provider's server. When these steps are completed, the attacker is able to establish a link between the victim's anonymized device (i.e., Tor browser) and the non-anonymized one (i.e., smartphone). Next we provide a more systematic explanation of the attack steps and several attack scenarios. Finally, we discuss a proof of concept attack we conducted to validate the feasibility of our attack.

Attack Steps. Figure 4 shows the stages of the attack and how different entities interact with the ultrasound framework. In the first step, the adversary $\boldsymbol{A}$ creates a new campaign with an ultrasound tracking provider and captures the inaudible beacon $\boldsymbol{b}$ associated with it (❶). Then, $\boldsymbol{A}$ creates a beacon trap that emits $\boldsymbol{b}$ (❷). An example of a beacon trap can be seen in Figure 4 where the adversary incorporates the trap in a hidden service to target a Tor user. Subsequently, $\boldsymbol{A}$ lures the anonymous user to visit the trap (❸). This step may involve social engineering or it may simply be that the trap uses a resource that the user is visiting often (i.e., a watering-hole attack). Alternatively, if the adversary is a Tor exit node operator, this step is practically omitted, since $\boldsymbol{A}$ can then inject the source code of the trap on the requested web pages without any need to interact with the user. The process of becoming a Tor exit node is very similar with that of a normal relay, and does not require many resources or a long reputation-building period.

In the next step, the victim uses her anonymous device to load the resource (❹). Once this happens, the beacon trap is triggered and the anonymous device starts to periodically emit $\boldsymbol{b}$ from its speakers (❺). Due to the high frequencies used by inaudible beacons, the user is completely oblivious to the fact that ultrasounds are being emitted and hence does not realize that a deanonymization attack is being carried out. Simultaneously, the ultrasound-enabled device owned by the user (e.g., a smartphone) is actively listening for uBeacons. Once it detects one, it captures the signal and transforms it into a string of characters. After running some integrity and validity checks on the string, the framework reports it (along with a number of unique user/device identifiers) to the provider's server, where it gets stored (❻).

Finally, $\boldsymbol{A}$ obtains the unique identifiers of the user who reported $\boldsymbol{b}$ (❼). There are many possible ways for $\boldsymbol{A}$ to do this. For instance, $\boldsymbol{A}$ can exploit the privacy leaks that RTB systems suffer from (Section 2). Alternatively, in the case, of intelligence agencies (a realistic assumption for attacks against Tor, VPN), they can request the data through a subpoena or a court order. Once the provider responds with the information, the real identity of the user can be trivially uncovered by the adversary. For instance, uXDT frameworks often use the *international mobile equipment identity* (IMEI) number as unique device an identifier.

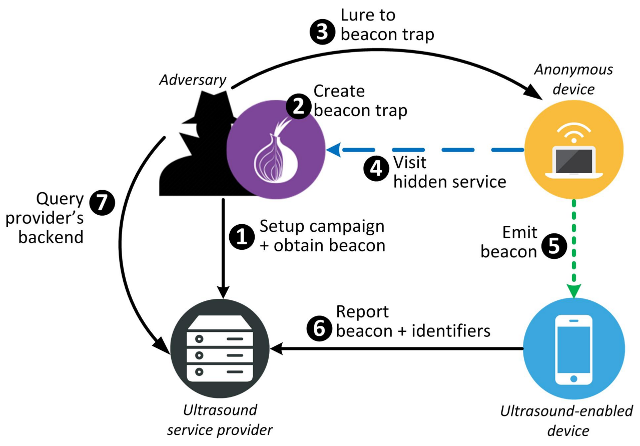
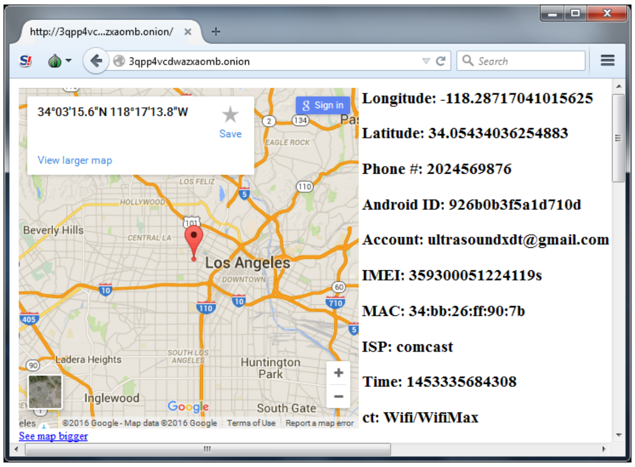

❸ Lure to beacon trap
Adversary
❷ Create beacon trap
Anonymous device
❹ Visit hidden service
Query provider's backend ❼
❶ Setup campaign + obtain beacon
Emit beacon ❺
❻ Report beacon + identifiers
Ultrasound service provider
Ultrasound-enabled device

Fig. 4. Operational steps of the deanonymization attack against a user of an anonymization network.



Longitude: -118.28717041015625

Latitude: 34.05434036254883

Phone #: 2024569876

Android ID: 926b0b3f5a1d710d

Account: ultrasoundxdt@gmail.com

IMEI: 359300051224119s

MAC: 34:bb:26:ff:90:7b

ISP: comcast

Time: 1453335684308

ct: Wifi/WifiMax

Fig. 5. Screenshot of the proof-of-concept web page upon a successful Tor deanonymization. The proof-of-concept attack can extract the user's location, ISP, phone number, Google account ID, and other sensitive information, even though the user is browsing through the Tor browser. (Note that for review anonymity, the details shown in the example do not identify the authors in any way.)

However, the IMEI can be easily attributed to a real person by the telecommunication operators [11, 12]. Moreover, uXDT service providers often store multiple unique identifiers, which make it even easier for $\mathcal{A}$ to infer the identity of the user (e.g., the Google account name).

Proof of Concept. To practically validate our proposed attack and evaluate its effectiveness, we conducted a proof of concept attack against a Tor user. The scenario of our PoC matches the stages illustrated in Figure 4 and closely resembles the *modus operandi* of an attacker in the real world. The attack was launched against a Tor user, who was a member of our team and consented to take part in the deanonymization experiment and to install a uXDT framework on her smartphone. For the purpose of the PoC, the ultrasound-enabled device is an Android Nexus 6 smartphone, running the application History GK [6] (available on Google Play Store), which incorporated the SilverPush uXDT framework.

For the PoC, we developed a malicious service that automates the majority of the steps conducted by the adversary $\mathcal{A}$. In accordance with the stages in Figure 4, the malicious service takes as input a unique beacon code b (❶), and generates a web page that features some dummy content and a JavaScript snippet emitting b (❷). The malicious web page is then made available through a Tor hidden service and the .onion address gets published (❸).

The Tor user then loaded the malicious hidden service from a laptop using the latest version of the Tor browser (6.0.1), configured with the default settings (❹). Once fully loaded, the malicious web page started to periodically emit the uBeacon b from the speakers of the laptop. Instantly, the ultrasound-enabled Android device (which was located on the same desk) detects it (❺) and reports the beacon ID along with some unique identifiers to the backend server of the ultrasound service provider (❻). Originally, these requests were directed to the company's IP addresses, however, by setting up a proxy server, we were able to intercept and analyse them.

At the final stage, $\mathcal{A}$ extracts from the service provider's servers the user information associated with b (❼). This can be done in many ways (e.g., RTB leakage, subpoena). However, since we didn't want our experiments to interfere with production systems serving real clients, we simply extract the data from the intercepted requests. This allowed us to accurately and safely conduct numerous experiments. Upon the retrieval of this information, the identity of the anonymous user has been fully uncovered. Figure 5 contains a visual representation of all the user information that an attacker can extract with a Tor deanonymization attack against SilverPush users.

Discussion. The impact of the deanonymization attack is substantial as it relies on minimal assumptions, which are compatible with the threat models of the great majority of anonymity networks (e.g., Tor) [21]. Moreover, the majority of the ultrasound-enabled apps are vulnerable to this attack as most of the existing frameworks report the uBeacons captured (often along with user data) to an online server operated by the service provider. Hence, we believe that the deanonymization attack poses an immediate threat for the users of anonymity networks and services, and it is critical that mitigation measures are deployed soon by the network operators and service providers.

5.2 uBeacon Injection

In the *uBeacon injection* attack the adversary exploits the lack of authentication in uBeacons to interfere with the uXDT and the proximity marketing mechanisms. For example, imagine an attacker equipped with a simple beacon-emitting device (e.g., a smartphone) walking into Starbucks at peak hour. As a result, all customers with an ultrasound-enabled app installed on their devices will be receiving the uBeacons and unknowingly forward them to the backend server of the service provider (e.g., SilverPush in cases of uXDT, Lisnr in cases of proximity tracking), thus polluting their user profiles and influencing the content served to them.

Attack Steps. As seen in Figure 6, the adversary ***A*** initially needs to acquire one or more valid beacons. To do this, ***A*** sets up an advertising campaign with an ultrasound service provider and captures the beacon associated with it (❶). Alternatively, ***A*** may simply record the beacon associated with a campaign run by another company. The adversary then starts replaying the beacon to one or more ultrasound-enabled devices belonging to different users (❷). This step of the attack can be realized in many ways, depending on the goals of the adversary. For instance, ***A*** can either replay the beacon using a mobile device, build a beacon trap and lure users into it, or even employ a computer or mobile device botnet.

Once the beacon is picked up by the ultrasound-enabled mobile device, the ultrasound framework performs a number of validity checks on it. Since the replayed beacon remains unaltered and no authentication mechanisms are in place, the validity checks always succeed. Subsequently, the mobile device processes the uBeacon, and reports its identifier (and a number of unique identifiers extracted from the device) to the company's backend server (❸). Upon reception, the ultrasound service provider stores the data and analyzes the uBeacon to determine with which ad campaign it is associated. Then, the ultrasound service provider updates the user's profile by considering the latest user interests and activities (❹). This means that each beacon injected by the adversary makes the user's profile less accurate. More specifically, the profile is *polluted* by ***A***, to include an activity or interest that is potentially unrelated to the user. The adversary can also seek to increase influence on the user's profile, by running multiple rounds of the attack and injecting multiple beacons.

Proof of concept. Beacon injection attacks have already been observed in the wild. For instance, users of the shopping application Shopkick [39] (available on Google Play Store), soon after the deployment of the system, started uploading archives with walk-in tones that other users could

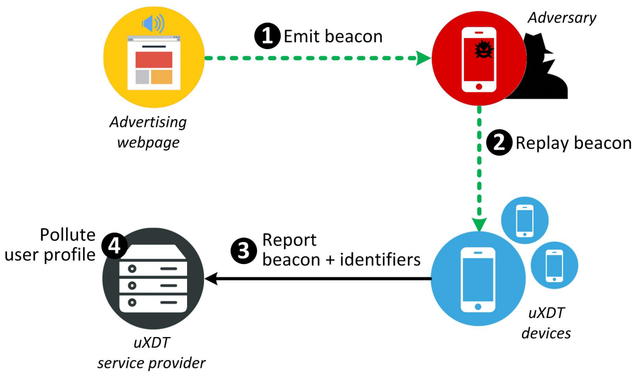


Fig. 6. Operational steps of the beacon injection attack to pollute users' profiles.

replay from their computer speakers to get the reward points. Apart from this, we also designed our own proof of concept, to demonstrate the potential impact of this attack. We chose to simulate a scenario where we launch a phishing attack against a single user owning a device with a uXDT framework. As with the deanonymization attack, no production systems were affected during the experiments. Additionally, for the purposes of the PoC, we developed an Android application that captures and replays inaudible beacons.

In the first stage of the attack, the adversary ***A*** creates a new campaign featuring interstitial ads (i.e., full-page ads), and uploads the banners specifically tailored to the victim. Using our Android application, ***A*** is able to capture the inaudible beacon of the campaign (❶). ***A*** then moves to another room where she is in proximity of the targeted user and uses our application to replay the beacon multiple times, with an interval of 15 seconds (❷). Simultaneously, the uXDT framework running within an app on the target's device *picks up* the emitted beacons and reports each one of them to the backend (❸). The backend then updates (i.e., pollutes) the user's profile based on the beacons injected by the adversary (❹).

At this point, the beacon injection attack has been completed. However, we take some extra steps to better illustrate the potential real-world impact, without influencing the recommendation systems of any commercial deployments. To achieve this, we redirected the beacon reports sent by the user's device to our own backend. Our backend is a simulation of a full recommendation system (such as the one that ad companies would use), and we programmed it to push an ad if a user reports a beacon more than twice within an hour. In other words, if the user watches the same ultrasound-carrying ad more than twice in an hour, then the same ad will appear on the user's uXDT device. As a result, when the app embedding the uXDT framework queries the backend for new ads, our algorithm goes through the polluted user's profile and finds that the beacon corresponding to $\boldsymbol{A}$'s campaign has been reported more than two times. At this point, the ad that is selected (and that would be eventually displayed on the app) is the adversary's malicious ad.

Discussion. The impact of the beacon injection attack depends on a number of factors. The first factor is the way in which the service provider utilizes the user's profiles. If the beacons act as a trigger for their corresponding ad to be shown, the adversary can influence which exact ads will be shown to the ultrasound-enabled device, and even start a new campaign and push malicious content (e.g., targeted malvertising campaigns [31][43], profile pollution attacks [33][49]). On the other hand, if the user's profile is used only as an information source to determine the general interests of the user, the impact is less severe. However, the adversary can still pollute the profile to include interests that the user would find offensive (e.g., adult content, religious affiliations).

The two other factors that determine the impact of the attack are the capabilities and the goals of the adversary. So far, we only discussed realizations of the attack where the adversary targets a single user. However, $\boldsymbol{A}$ may also be able to interfere with a campaign launched by a competitor. In this case, the impact can be severe if the attacker has the means to inject the beacon to a very large number of devices. This can be achieved by employing a large botnet, or by injecting JavaScript as a Tor exit node (as discussed in Section 5.1). By doing this, the adversary could manipulate an uXDT service provider into showing the ad to unrelated users and thus reduce the conversion rate and waste part of the campaign budget. Another instance where a large-scale injection attack could have a detrimental effect is *TV analytics*. In TV analytics, inaudible beacons are often used to track user engagement and behavior. An adversary could easily replay the beacons corresponding to a specific TV program to a large number of ultrasound-enabled devices and influence the results.

We believe that the exact implementation of the ad serving mechanism determines the potential impact too. For example, currently, some companies allow the advertising client to set a frequency cap on clicks and impressions of the ad. This could mitigate some unsophisticated versions of the attack, but they would be hardly effective against adversaries who use a botnet.

The implementation of authentication mechanisms would help prevent uBeacon injection attacks. However, uBeacons carry a very limited volume of information, and suffer from a moderate rate of transmission errors [19]. These would result in a high occurrence of invalid beacons, thus decreasing the efficiency of the technology. Hence, the implementation of authentication mechanisms in ultrasound tracking systems comes with multiple challenges and is far from straightforward. Instead, Shopkick tries to address these attacks using contextual information to detect abnormal walk-ins (e.g., the location reported by GPS is different from the location of the store), but with limited success.

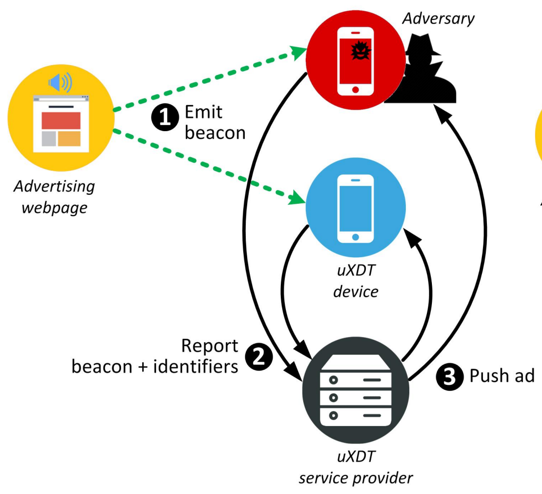
5.3 Information Leakage

In this attack, the adversary exploits the lack of authentication information and the capability of injecting beacons to breach the privacy of a victim user. For instance, an employer could acquire a wealth of private information on the personal interests and activities of the employees. To do this, the employer could eavesdrop beacons emitted from the computer speakers of a specific employee, and then replay them to the employee's uXDT-enabled smartphone. Subsequently, the uXDT framework will report these beacons to the ultrasound service provider and fetch ads related to the interests of the victim user, thus partially exposing the victim's profile. Additionally, depending on the realization of the attack, this can also trigger the establishment of a link between the victim's and the adversary's devices, which entails that beacons captured by one of the devices will influence the ads shown to the other.

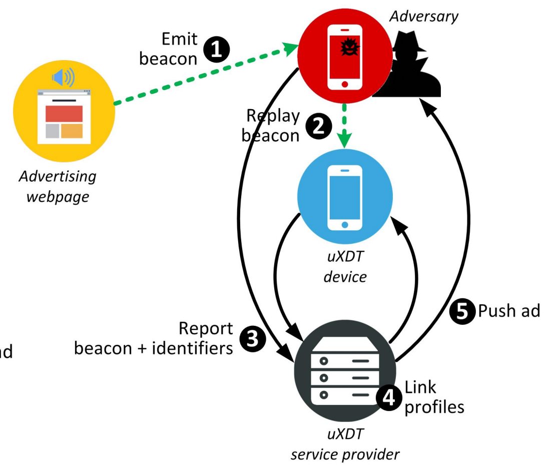
Attack Steps. Depending on the capabilities of the adversary, the attack can be carried out in either a passive or active fashion.

The *passive* version (Figure 7a) of the attack is an evolution of the homonymous attack introduced by Castelluccia et al. [16]. In this scenario, it is assumed that the user operates in two or more environments with different behavioral requirements. As a result, there is clear separation of the activities the user is carrying out in each of these environments. An example of such a setting would be a user who handles two computers, one at home and one at work. The adversary $\boldsymbol{A}$ is assumed to be located in one of the environments (e.g., at work) and is in physical proximity with the user.

On the first step of the attack, the victim uses a computer to browse the Internet while being in physical proximity with $\boldsymbol{A}$. During this browsing session, a number of targeted ads with inaudible beacons are displayed on the user's screen (❶). These targeted ads are related to the activities and the interests of the user in general, and not only those observed in the current environment (i.e., job). Simultaneously, $\boldsymbol{A}$ is handling a uXDT device, which detects the beacons emitted by the user's ads, and reports



(a) Passive information leakage attack



(b) Active information leakage attack

Fig. 7. Operational steps of the information leakage attacks.

them to the uXDT provider's backend (❷). As the adversary $\boldsymbol{A}$ captures and submits the beacons intended for the user, she slowly builds her own profile to match that of the user. Once enough beacons are collected, the two profiles are very similar and the attack reaches its final stage. From this point on, $\boldsymbol{A}$ starts to get served ads related to the interests of the user (❸). $\boldsymbol{A}$ then uses the techniques described in [16, 17] to remove any noise, and build a quite accurate interests profile for the user. To do this, $\boldsymbol{A}$ first applies a number of filters which remove irrelevant ads (e.g., contextual, location-based) and then rebuilds the user's profile by attributing each of the remaining ads to one broad category (e.g., car rental, travel).

The *active* version of this attack exploits the device linking that XDT techniques perform. More specifically, the adversary uses a beacon-emitting device to inject/replay beacons to the victim's uXDT-enabled device, while reporting the same beacons along with meta-data (e.g., location, gender), matching those of the victim to the ultrasound service provider's backend. After a number of beacons have been reported, a pairing between the device of the adversary and the victim will be triggered with high probability. From now on, the attacker and the victim's devices will be linked, and thus will start receiving related advertising content. It should be noted that the details of the linking depend on the specific design of the ultrasound service provider's profiling system.

More formally, the *active* version of the information leakage attack is depicted in Figure 7b and requires five steps. Initially, the adversary $\boldsymbol{A}$ collects a large number of beacons from various sources (❶); however, $\boldsymbol{A}$ does not report any of them to the uXDT service provider's backend. Instead, $\boldsymbol{A}$ starts replaying those beacons to the targeted user's uXDT device (❷). Simultaneously, both the adversary and the user report these beacons to the backend server (❸). As in the previous attacks, each report contains some unique identifiers and the beacon ID. Once a large number of beacons have been submitted, the service provider links the profiles of the user and the adversary with high probability. It should be noted that the actual precision of the attack depends on the profile matching method used by each company. However, according to the literature, companies that utilize behavioural profiling methods (using different tracking technologies) tend to match identically behaving users with high probability [45, 50, 52].

This happens because the provider assumes that the two devices belong to the same person (❹). At this point the attack has been completed and the uXDT service provider starts pushing the same ads in both devices. This means that $\boldsymbol{A}$ is able to get an insight into the user's interests by studying the ads served on $\boldsymbol{A}$'s own device.

Proof of concept. The PoC of this attack is a composition of the PoCs previously described, and, therefore, its description is omitted.

Discussion. The impact of this attack is hard to quantify, as a privacy invasion can have various implications ranging from negligible to severe. Nonetheless, we believe that privacy is strongly related with the rights of the individual, and hence should never be jeopardized, regardless of the consequences.

Moreover, as we mentioned in our introduction, each inaudible beacon corresponds to a specific ad shown to the user. A subtle assumption of the uXDT model is that the inaudible beacons can be liberally made public (i.e., broadcasted) because they do not carry sensitive information. However, as shown in the previous work [16], targeted ads (and by extension beacons) can be exploited by an eavesdropping adversary to infer the user interests and activities.

This information leakage exposes some fundamental flaws in the uXDT technology, which enable an adversary to acquire otherwise private information about the user's interests and activities. Unfortunately, the limited bandwidth offered by inaudible beacons prevents the application from being able to adopt any protection mechanism at a lower level. Instead, solutions should be sought at the application level (i.e., in the business logic).

5.4 Responsible Disclosure

We communicated our findings and concerns to SilverPush, and they acknowledged the aforementioned security vulnerabilities. Subsequently, on a recent announcement [22], SilverPush claimed that they have ceased all collaborations with US developers. However, no information is provided regarding developers from other countries. We also contacted the Tor project to inform them about the deanonymization attack.

6 System Enhancements

In this section, we discuss potential solutions to the security and privacy issues identified earlier, and evaluate their effectiveness and applicability. For instance, a straightforward way to prevent ultrasonic tracking would be a jamming device emitting ambient ultrasonic noise. However, such a solution suffers from numerous drawbacks (e.g., contributes to noise pollution, impractical to carry at all times). Instead, we propose two immediately deployable security mechanisms designed to mitigate the risks for browser and smartphone users. Moreover, we propose a design for a new OS-level API that would allow developers to implement ultrasound-based mechanism without requiring full access to the microphone, thus honouring the principle of least privilege.

6.1 Ultrasound-filtering Browser Extension

We developed an extension for the Google Chrome browser that filters out all ultrasounds from the audio output of the websites loaded by the user. Instead of simply detecting beacons, the extension proactively prevents webpages from emitting inaudible sounds, and thus completely thwarts any unsolicited attempts of uXDT (unless the user opts to allow ultrasounds from a specific tab). As a result, this extension offers protection against the deanonymization and information leakage attacks introduced in Section 5.

From a technical perspective, the extension mediates all the audio outputs of the page and filters out the frequencies that fall within the range used by inaudible beacons. To do this, each time a new webpage is loaded, a JavaScript snippet is inserted and executed in the page. The snippet is using the Web Audio API, which is part of the HTML5 specification. This API represents the audio handling modules of the page as `AudioNodes`. An `AudioNode` can be an audio source (e.g., an embedded YouTube video), an audio destination (e.g., the speakers) or an audio processing module (e.g., a low-pass filter). It also introduces the concept of *audio graphs*, where different `AudioNodes` are linked together to create a path from the audio source to the audio destination. This path can be arbitrarily long and may include multiple filters, which process the signal before it reaches the destination node.

Our extension takes advantage of this `AudioNode` linking capability in order to add a new filter between all the audio sources of the page and the audio destination. More specifically, upon execution, the snippet creates a *high-shelf* filter `AudioNode`, and sets its base frequency to 18kHz and its gain to -70db. As shown in Figure 8, the settings of our extension can be customized to flexibly opt out from specific advertising companies. As a result, the filter attenuates all the frequencies that are higher than 18kHz, while it leaves the lower frequencies unaltered. Subsequently, the snippet identifies all the audio sources of the page and modifies the audio graph so that their signal passes through the high-shelf filter before it reaches the speakers. From this point on, any audio played by the page is first sanitized by the high-shelf filter and then forwarded to the system's speakers. This procedure happens in real-time and it has minimum impact on audible frequencies of the spectrum.

Our extension completely filters out all ultrasonic frequencies, and hence provides complete protection from attacks in this spectrum since the emission of uBeacons is completely blocked (we tested with real-world Android apps that use SilverPush framework). It has only one technical limitation: due to some shortcomings of HTML5, it cannot directly filter a few legacy/obsolete technologies, such as Flash player. We believe that this limitation is alleviated by using the other countermeasures introduced in this section.

6.2 Android Ultrasound Permission

In this section, we describe a modification to the Android OS to implement a mechanism analogous to the one described in the previous section. In particular, we aim at providing the functionality for the end users to selectively filter the ultrasound frequencies of the signal acquired by the smartphone microphone. In general, our idea is to separate permissions for listening to *audible* sound and sound in the high-frequency (i.e., ultrasound) spectrum. This separation forces the applications to explicitly declare that they will capture sound in the inaudible spectrum, and it makes the usage of mechanisms based on ultrasounds as an opt-in feature.

We have modified the Android OS to extend the existing permission, `RECORD_AUDIO`, so that an app that needs to record audio from the microphone can now selectively ask whether it also requires access to the high-frequency spectrum of the signal (by requiring the new `RECORD_ULTRASOUND_AUDIO` permission). Since the vast majority of apps does not require access to the high-frequency spectrum and can function normally even without it, we believe that the mere asking for access of the high-frequency spectrum could be used as a strong signal for requiring more carefully vetting at the app store level.

The implementation of this new permission required the modification of two parts. First, we needed to define a new permission, which can be done by modifying a configuration file.[1] The second modification relates to the AudioFlinger component. AudioFlinger is the main sound server in Android: when an app wants to obtain a data *read* from the microphone, the app communicates through the Binder IPC mechanism to the AudioFlinger component. This component, in turn, reads the stream data from the kernel sound driver (e.g., ALSA, OSS, custom driver) and makes the content accessible to the requesting app (once again through the Binder IPC mechanism).

Our modification in the AudioFlinger component implements the following logic. Consider an app that wants to acquire data from the microphone. If this app has both the `RECORD_AUDIO` *and* the `RECORD_ULTRASOUND_AUDIO` permissions, then the stream is not modified. However, if the requesting app does not request the `RECORD_ULTRASOUND_AUDIO` permission, our patched AudioFlinger would *filter out* frequencies above a certain threshold. The filter implemented by our current prototype is a standard low-pass filter that attenuates signals with frequencies higher than the cutoff frequency.

Our current implementation filters sound in the time domain, and it can thus operate in real-time and only requires a few bytes of extra memory. Furthermore, our patch is not invasive and is constituted by less than one hundred lines of codes.

6.3 Standardization & uBeacon API

We believe that the mechanisms we proposed earlier in this section would significantly increase the security and awareness of the user. However, we acknowledge that these mechanisms have two main limitations. First, they might interfere with benign use cases. Second, and more importantly, they do not address one of the main issues associated to this technology: an app that wants to implement ultrasound-based mechanisms currently needs full access to the microphone, thus leading to significant privacy concerns.

We argue for the standardization of the ultrasound beacon format and we envision a new OS-level API that implements in a single, trusted place to logic for detecting and decoding uBeacons. The first step towards this goal is to specify the format and structure of a uBeacon. Once this process is completed, an API for handling uBeacons can be implemented. In particular, such an API should expose calls for: (1) uBeacon discovery and capturing, (2) uBeacon decoding and integrity validation, and (3) uBeacon generation and emission (targeted towards Android embedded devices). We note that the developers of a similar technology, Bluetooth low energy beacons, are following this direction. In fact, an API for Bluetooth low energy beacons is already provided in Android to regulate the bluetooth usage [2, 4, 5].

From a technical perspective, this API would be implemented by a privileged process (like a system process on Android) and its existence would provide several important benefits. To begin with, an app that wants to implement ultrasound-based mechanisms only needs to acquire access to this API, and no full access to the device's microphone is needed anymore. Thus, the user's sensitive data are not exposed to third-party apps in the first place. Additionally, app developers would not need to require a security-sensitive permission (i.e., microphone access) to make use of ultrasound-based technology, thus avoiding being deemed as "potentially malicious" by the users. An-

1 In AOSP, the file is at the following file path: `./base/core/res/AndroidManifest.xml`

other advantage of such API is that it would act as a central place to detect and monitor ultrasound-based abuses.

However, in order to enforce the use of this API, the ultrasound spectrum must be accessible only to privileged components of the system. To achieve this, the system module handling the microphone should filter out ultrasonic frequencies by default, and the user should be able to grant access to the spectrum on a per-app basis. Our ultrasound permission demonstrates that such filtering is technically feasible.

As a result, since third-party apps would never get access to any signal in the ultrasound spectrum (even when requiring the microphone permission), it is impossible for malicious websites (or other any other source) to make use of non-standard custom beacons and to violate the user's privacy as it happens right now: in fact, even if a custom beacon is encoded in an audio stream, the framework API will not properly detect and decode such beacons. Thus, forcing all third-party developers to make use of the central API will provide the right incentive to implement ultrasound-based mechanisms according to the standard. Finally, such an API would improve the transparency of the beacon capturing process, and ensure the user's full awareness on the information collected.

We acknowledge that the proposed API does not prevent aggressive ad frameworks from performing cross-device tracking. However, we note that the API itself could be used as a central component for monitoring ultrasound-related events and that privacy-conscious users can use the defense mechanisms discussed earlier in this section to completely block ultrasound usage.

7 Related & Future Work

This work is related to the abuse of audio as a physical layer. To the best of our knowledge, we are the first to focus specifically on the security and privacy aspects of inaudible frequencies to link, for various purposes, different devices. Indeed, previous researches have focused mainly on the use of audio—both in the audible and inaudible spectra—as a *covert* channel.

Acoustic covert channels. It has been known for at least ten years that audible and inaudible audio can be used as a carrier for digital data [28]. Nowadays there are even open-source implementations of a software modem [29] and a TCP/IP networking stack [13] based on ultrasonic frequencies as a physical layer. Recently, after the highly debated BadBIOS [25]—a malware prototype that, according to news coverage, can communicate over an inaudible audio covert channel—new research work has been published in this area [27, 46]. A recent work examines how inaudible audio can be used in mobile devices as a covert channel for inter- and intra-device communication [19]. The authors propose an optimized modulation scheme that minimizes audible "clicks" due to the sudden change of frequency that encode the alternating ones and zeros. Alongside, they evaluate the feasibility of using the device's vibrator to encode bits as subtle vibrations that produce no sound when the device is left on a flat surface. The authors also measure the maximum achievable bit and error rate at varying distances, and conclude by briefly listing a series of countermeasures. Moreover, there are also modulation techniques that can be used to create covert channels in the audible spectrum. One such example is the Intrasonics [8] technology, which exploits the brain's natural echo filtering to encode data bits as inaudible echo sequences.

Audio channel access control. Petracca et al. [35] focuses on audible audio on mobile devices as a communication channel that can be abused by an attacker when the user is not listening, and concludes that their proposed attacks are feasible because of the complete lack of access control. More precisely, the authors analyze the feasibility of creating a malicious app that uses the phone speaker, for example, to produce commands that would be picked up and blindly executed by an app that trusts the microphone as an input for commands. The typical examples are voice-activated procedures (e.g., "OK Google, play some music" or "browse on evil.com"). In a similar vein, the microphone can be abused by a malicious application to eavesdrop sensitive communication (e.g., while a screen-reading, text-to-speech application is "speaking" a user's confidential email). Along this line, the authors describe and implement six attacks and propose an SE Linux extension to regulate the access to microphone and speakers. Our long-term solution (i.e., a permission bit specifically tailored for the ultrasonic spectrum) follows the same principle, that is, ensuring that only apps that have the requested permission can operate in the ultrasonic range.

Open Problems. Moreover, as the ecosystem is expanding, it is important to develop efficient mechanisms to keep track of the new applications participating in it, and evaluate their security in a systematic way. Potential solutions to this open and challenging problem could include a combination of heuristics and static analysis techniques. Moreover, another interesting area for future work is the incorporation of lightweight cryptographic mechanisms in uBeacons. Such an advancement could yield very interesting results making some of the current applications more secure, and enabling a whole new range of features.

8 Conclusion

This paper discusses the privacy and security concerns that arise with the emerging of new technologies based on the ultrasound. In particular, we first show that commodity OS (e.g., Android) do not offer proper APIs to implement ultrasound-based mechanisms without violating the principle of least privilege: any app using this technique necessarily needs to require full access to the device's microphone. We also perform the first security analysis of ultrasound-based mechanisms, such as proximity marketing and advanced cross-device tracking techniques used by aggressive advertising and analytics companies. We determine the current schemes to be vulnerable to a series of attacks, including user deanonymization, beacon injection, and information and privacy leakage. We also show that when these emerging techniques are *combined* with existing ones, the effects to the user's privacy can be devastating, especially since some of these uXDT services do not provide an opt-out mechanism option for users.

We conclude our work by proposing a series of mechanisms and modifications to protect browsers and smartphones. We also argue for the standardization of the format of ultrasound beacons and the introduction of a new OS-level API that would act as a single, trusted component to detect and decode ultrasound beacons. We hope this work will raise awareness of this emerging trend in the community. We released the prototypes and source code of our countermeasures, so that users are aware of the threats and can actively protect themselves.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable feedback. We would also like to thank Lara, our research assistant. Vasilios Mavroudis was supported by H2020 PANORAMIX Grant (ref. 653497). This material is also supported by NSF under Award No. CNS-1408632, and by Secure Business Austria. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of NSF or Secure Business Austria.

References

[1] Made in America Festival 1.0.8 app on the Play Store. https://play.google.com/store/apps/details?id=com.lisnr.festival.madeinamericaandroid, 2015.
[2] Bluetooth Low Energy API Level 18. https://developer.android.com/guide/topics/connectivity/bluetooth-le.html, 2016.
[3] Google Cast 1.16.7 app on the Play Store. https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app, 2016.
[4] Google Nearby Messages API. https://developers.google.com/nearby/messages/android/get-beacon-messages, 2016.
[5] Google Proximity Beacon API. https://developers.google.com/beacons/proximity/guides, 2016.
[6] History GK 5.0 app on the Play Store. https://play.google.com/store/apps/details?id=com.gktalk.history, 2016.
[7] Indianapolis Colts Mobile 3.1.1 app on the Play Store. https://play.google.com/store/apps/details?id=com.yinzcam.nfl.colts, 2016.
[8] Intrasonics-Artificial Echo Modulation. http://www.intrasonics.com/, 2016.
[9] McDo Philippines app on the Play Store. https://play.google.com/store/apps/details?id=ph.mobext.mcdelivery, 2016.
[10] Signal360's Use Cases. http://www.signal360.com/#results, 2016.
[11] 3GPP. 3rd generation partnership project, technical specification of international mobile station equipment identities (imei).
[12] A. Andreadis and G. Giambene. The global system for mobile communications. *Protocols for High-Efficiency Wireless Networks*, pages 17–44, 2002.
[13] anfractuosity. Ultrasound Networking. 00003.
[14] Audible magic. https://www.audiblemagic.com/advertising/, 2016.
[15] C. Calabrese. Comments for November 2015 Workshop on Cross-Device Tracking.
[16] C. Castelluccia, M.-A. Kaafar, and M.-D. Tran. Betrayed by your ads! In *Privacy Enhancing Technologies Symposium*, pages 1–17. Springer, 2012.
[17] T. Chen, I. Ullah, M. A. Kaafar, and R. Boreli. Information leakage through mobile analytics services. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, page 15. ACM, 2014.
[18] Copsonic. http://www.copsonic.com/products.html#webtostoretracker, 2016.
[19] L. Deshotels. Inaudible sound as a covert channel in mobile devices. In *Proc. 8th USENIX Conf. Offensive Technologies*, page 16.
[20] A. Detector. Silverpush android apps. https://public.addonsdetector.com/silverpush-android-apps/, November 2015.
[21] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.
[22] Forbes. Silverpush quits creeping world out, ceases tracking tv habits via inaudible 'beacons'. http://www.forbes.com/sites/thomasbrewster/2016/03/21/silverpush-tv-mobile-ad-tracking-killed/, March 2016.
[23] Ftc public discussion on cross-device tracking. https://www.ftc.gov/news-events/audio-video/video/cross-device-tracking-part-1, November 2015.
[24] D. Goodin. Beware of ads that use inaudible sound to link your phone, TV, tablet, and PC.
[25] D. Goodin. Meet "badBIOS," the mysterious Mac and PC malware that jumps airgaps. 00004.
[26] Google. Chromecast guest mode - guest mode faqs. https://support.google.com/chromecast/answer/6109297?hl=en, August 2016.
[27] M. Hanspach and M. Goetz. On Covert Acoustical Mesh Networks in Air. 8(11):758–767.

[28] Honda Electronics. Ultrasonic Aquatic Communication System.

[29] M. Kamal. minimodem - general-purpose software audio FSK modem.

[30] Lisnr. http://lisnr.com/platform, 2016.

[31] Real-time bidding and malvertising: A case study. https://blog.malwarebytes.org/cybercrime/2015/04/real-time-bidding-and-malvertising-a-case-study/, April 2015.

[32] M. Marlinspike. New tricks for defeating ssl in practice.

[33] W. Meng, X. Xing, A. Sheth, U. Weinsberg, and W. Lee. Your online interests: Pwned! a pollution attack against targeted advertising. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 129–140. ACM, 2014.

[34] L. Olejnik, C. Castelluccia, et al. Selling off privacy at auction. 2014.

[35] G. Petracca, Y. Sun, T. Jaeger, and A. Atamli. AuDroid: Preventing Attacks on Audio Channels in Mobile Devices. In *Annual Computer Security Applications Conference*. ACM Press.

[36] E. Ramirez. Transcript - Part 1. In *FTC Cross-Device Tracking Workshop*.

[37] E. Ramirez. Transcript - Part 2. In *FTC Cross-Device Tracking Workshop*.

[38] C. Roeding and A. Emigh. Method and system for location-triggered rewards, July 16 2013. US Patent 8,489,112.

[39] Shopkick. https://www.shopkick.com/, June 2016.

[40] Signal360. http://www.signal360.com/#solution, 2016.

[41] A. Silverman. Colts to begin using lisnr technology to reach fans' mobile devices at games, events. http://www.sportsbusinessdaily.com/Daily/Issues/2016/07/19/Franchises/Colts.aspx, July 2016.

[42] Silverpush. https://www.silverpush.co/#!/audio, 2015.

[43] A. K. Sood and R. J. Enbody. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE security & privacy*, (1):54–61, 2013.

[44] Sophos. Users weighed down by multiple gadgets and mobile devices, new sophos survey reveals. https://www.sophos.com/en-us/press-office/press-releases/2013/03/mobile-security-survey.aspx, March 2013.

[45] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[46] V. Subramanian, S. Uluagac, H. Cam, and R. Beyah. Examining the characteristics and implications of sensor side channels. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2205–2210. IEEE.

[47] Tchirp. http://www.tchirp.com/#theTech, 2016.

[48] P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. Weippl. Spoiled Onions: Exposing Malicious Tor Exit Relays. In *Privacy Enhancing Technologies Symposium*. Springer, 2014.

[49] X. Xing, W. Meng, D. Doozan, A. C. Snoeren, N. Feamster, and W. Lee. Take this personally: Pollution attacks on personalized services. In *Proceedings of the 22nd USENIX Security Symposium*, 2013.

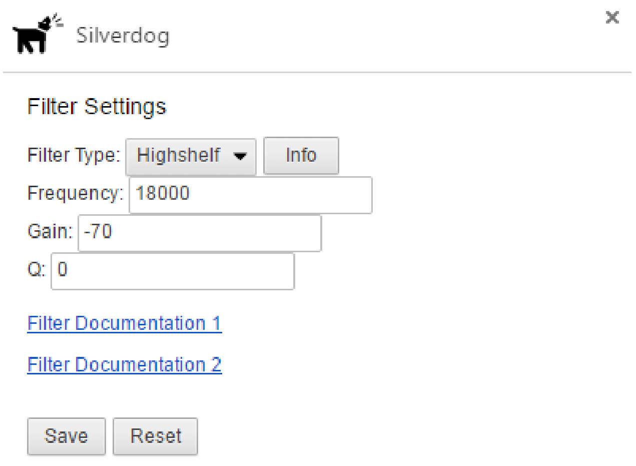
[50] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th international conference on World wide web*, pages 261–270. ACM, 2009.

[51] Y. Yuan, F. Wang, J. Li, and R. Qin. A survey on real time bidding advertising. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 418–423. IEEE, 2014.

[52] W. Zhang, L. Chen, and J. Wang. Implicit look-alike modelling in display ads: Transfer collaborative filtering to ctr estimation. *arXiv preprint arXiv:1601.02377*, 2016.

9 Appendix

In this appendix, we provide some additional screenshots from the countermeasures introduced in Section 6.


Silverdog

Filter Settings

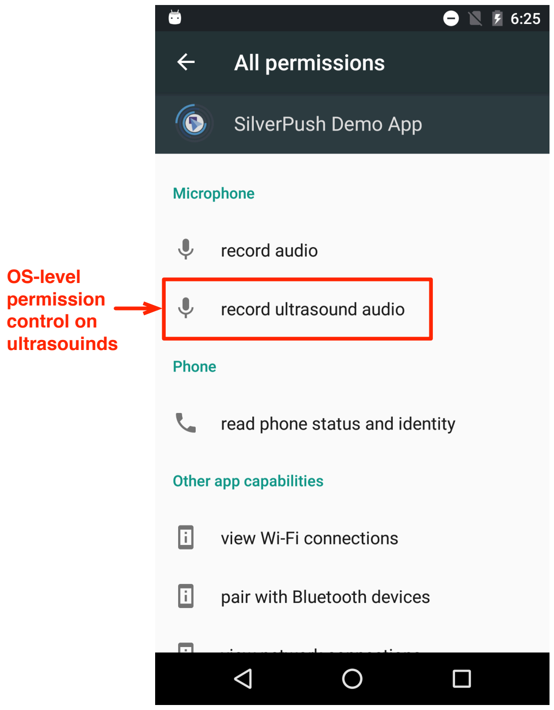
Filter Type: Highshelf

Frequency: 18000

Gain: -70

Q: 0

Fig. 8. Details of the settings page of our beacon-filtering browser extension.


All permissions

SilverPush Demo App

Microphone
- record audio
- record ultrasound audio

Phone
- read phone status and identity

Other app capabilities
- view Wi-Fi connections
- pair with Bluetooth devices

Fig. 9. Screenshot of the permission system to allow fine-grained control on the ultrasonic spectrum. An ultrasound-enabled application needs explicit permission to access the ultrasounds over the audio channel (which is a separate permission from the normal audio).