Analysis of an Ultrasound-Based Physical Tracking System

Mathieu Cunche, Leonardo S Cardoso

Analysis of an Ultrasound-Based Physical Tracking System

Mathieu Cunche[†], Leonardo S. Cardoso[†]

firstname.lastname@insa-lyon.fr
[†]Univ Lyon, INSA Lyon, Inria, CITI, France

May 23, 2018

Abstract

In this paper, we present the analysis of an Ultrasound-based tracking application. By analyzing several mobile applications along with the network communication and sample of the original audio signal, we were able to reverse engineer the ultrasonic communications and some other elements of the system. Based on those finding we show how arbitrary ultrasonic signal can be generated and how to perform jamming. Finally we analyze a real world deployment and discuss privacy implications.

1 Introduction

Ultrasounds can be used to carry signal, the same way radio waves does. Recently, this feature ha s been exploited to create side-channels. A trending application of ultrasound communication is the implementation of ultrasonic beacons that broadcast an identifier in a specific area. Those ultrasonic beacons found the same use as Bluetooth beacons: a mobile application running on a smartphone can pick those signal to localize the user and to trigger events. Beacon technologies are used in the retail industry for geofencing and to push promotional messages to users during their visit of a shop. The advantage of ultrasonic signals is that all smartphone can receive, and as opposed to other technologies based on Bluetooth that requires the corresponding interface to be activated, the microphone is always available.

The use of ultrasounds as side-channel raises privacy concerns as it can be used for cross-device identification, de-anonymization, or location tracking [4, 2].

In this work we focus on a specific ultrasound system called SpotInStore. Using publicly available resources, we have been able to reverse-engineer the Spotinstore system. The goal of this work is to increase the knowledge of those new technologies and to analyze the associated privacy and security risks.

Section2 introduces the SpotInStore system then section 3 presents our analysis of the system. Signal generation and jamming are presented in section 4 and the analysis of the real world deployment is done in section 5. Finally, section 6 concludes.

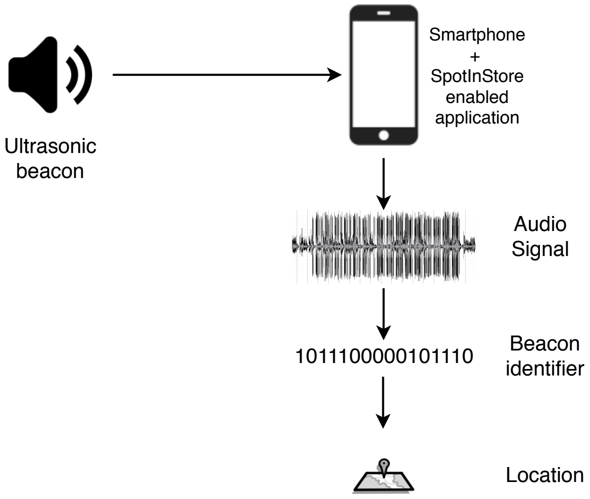


Figure 1: Details of the *SpotInStore* system.

2 The *SpotInStore* system

A French company, *Fidzup*[1], specialized in *in-store tracking technology* has developed an ultrasonic tracking system named *SpotInStore*[2]. This system uses ultrasonic beacon to localize user in the physical world.

2.1 System description

The *SpotInStore* system relies on ultrasonic channel to transmit information from the environment to an agent running on a mobile phone. This system is composed of several elements:

- One or several ultrasonic beacons deployed in the physical space and broadcasting an audio signal;
- An application having the microphone permission running on the subject phone.

Ultrasonic beacons are typically deployed in a shop locally broadcasting a signal that cannot be heard by humans. The mobile application running on the phone can access the microphone of the device to pickup this ultrasonic signal. This ultrasonic signal is then processed by the application to extract an identifier corresponding to the code of the beacon. Each beacon has its own identifier, and this identifier can be associated to a specific location. Thanks to this association between the beacon code and a location, the application can obtain the location of the user. Figure 1 presents the details of the *SpotInStore* system.

[1] `https://www.fidzup.com/en/`

[2] `https://www.fidzup.com/en/live/`

From now on, the application has access to the current location of the user as it would have been provided by the geolocation service of the phone if the application had been granted the geolocation permission. This location information can then be processed by the application but can also be sent to a remote server. In the example advertised by Fidzup, the geolocation information is used for geofencing: triggering specific actions when a user enter or leave a specific area.

The *SpotInStore* system is associated with two mobile applications that are used for demonstration and troubleshooting. In addition, we have identified an application of a shopping center that uses the *SpotInStore* system.

2.2 Demonstration app: *Fidzup Live*

The first application, *Fidzup Live*[3], is used to demonstrate the system when combined with a demonstration video. This video, presents an application scenario in which a user receive coupons through pop-up notification in a mobile application. This video embeds an ultrasonic signal, that can be received by a nearby phone. When watching the video with the FidzupLive mobile app running, pop-up notification appears on the phone simultaneously with those that appear on the video.

2.3 Troubleshooting app: *Fidzup Diagnostic*

The second application, *Fidzup Diagnostic*, is a troubleshooting tool whose purpose is to help identify reception issues in real world deployment. The main feature of this mobile application is the detection and identification of an ultrasonic signal. When launching the listen activity, it capture the audio signal and attempt to extract the ultrasonic signal. As a secondary feature, it seems to be able to identify the location where the detected signal is supposed to be deployed. However, for all the signals we have submitted to the Diagnostic app, the app always failed to identify a location.

2.4 Shopping center app

The SpotInStore system is currently deployed in a shopping center near Paris: *La Vache Noire*[4] at Arceuil. A mobile application[5] dedicated to this shopping center is available both on the Google PlayStore[6] and the Apple AppStore[7]. The geofencing feature of the mobile application is described on their website: *un procédé de "géofencing" pour offrir des promotions et alertes exclusives aux utilisateurs se trouvant dans le centre.* and it adds that no personal information is stored *Dès qu'elles le quittent, la désactivation des messages est automatique et immédiate et aucune donnée nominative n'est enregistrée.* Through a physical visit of the shopping center, we have been able to confirm the presence of ultrasonic signal.

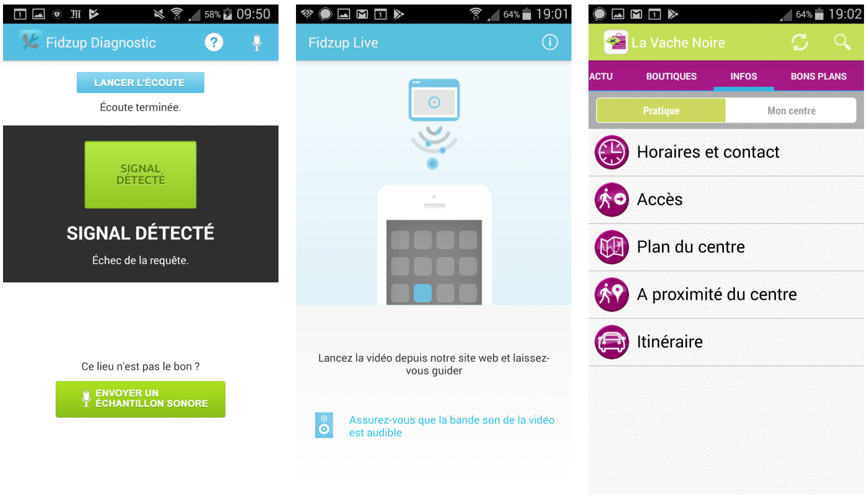
[3] https://play.google.com/store/apps/details?id=com.fidzup.fidzuplive

[4] http://www.la-vache-noire.com/

[5] http://www.la-vache-noire.com/application-mobile-la-vache-noire/

[6] https://play.google.com/store/apps/details?id=com.fidzup.darkmall

[7] https://itunes.apple.com/fr/app/id618120919



Écoute terminée.

SIGNAL DÉTECTÉ

Échec de la requête.

Ce lieu n'est pas le bon ?

Lancez la vidéo depuis notre site web et laissez-vous guider

Assurez-vous que la bande son de la vidéo est audible

(a) Fidzup Diagnostic (b) Fidzup Live (c) *La Vache Noire*

Figure 2: Screenshot of mobile applications featuring the SpotInStore system.

3 Analysis of the system

We performed a reverse engineering of the system by analyzing several elements of the system : the Fidzup Live and Fidzup Diagnostic applications and the. More specifically we analyze the ultrasonic signal and the network communications.

3.1 The ultrasonic signal

Analysis of the ultrasonic signal is easiest if we have access to an actual signal. Luckily the demonstration video used with the Fidzup Live application contains such a signal. We describe the process of our analysis and our first findings.

The first step of the analysis is the acquisition of a sample. This was done by retrieving the demonstration video and extracting the soundtrack with `ffmpeg`. Then using a spectrum analyzer (`Sonic Vizualizer`) we were able to clearly identify the ultrasonic signal centered around 19KHz (see Figure3).

Focusing on the ultrasonic signal of the audio track (see Figure **??**), we can identify a sequence of energy pulse separated by blank spaces. This signal appear to be an *amplitude-shift keying* (ASK) modulation, in which a sequence of bit in which the symbol '1' is represented by transmitting a carrier wave with a fixed amplitude, and the symbol '0' is represented by the lack of a carrier wave (blank).

Furthermore, we can have a rough estimation of the symbol duration by measuring the time between the beginnings of two consecutive 1's. Measuring this value on the previous signal, we get an approximate duration $d = 0.0669$ seconds.

The sequence appears to have a structure as the same pattern repeats itself which suggests that a frame in transmitted in loop by the signal. We further

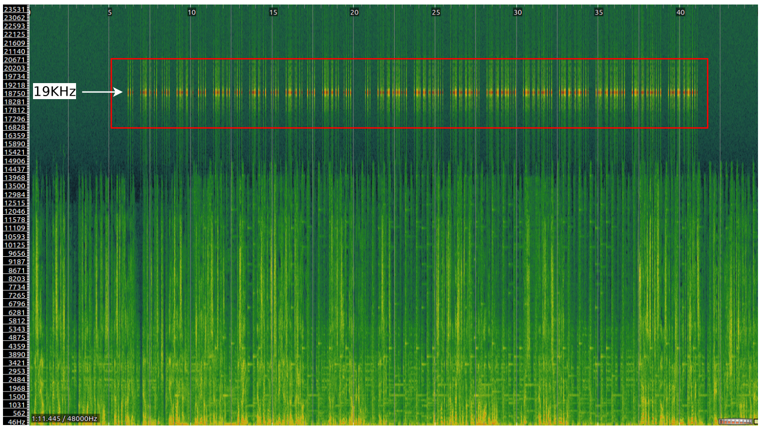


Figure 3: Spectrogram of the audio from the demonstration video. A signal centered on 19KHz is visible from time 6 to time 41.

observe the ultrasonic sequence can be divided into two part, each of them having a different pattern that repeats itself several times. After further analysis, we identified that the length of this pattern is 32 symbols meaning that a frame is formed of 32 bits. Up to this point the structure of this frame and the information it carries remain unknown.

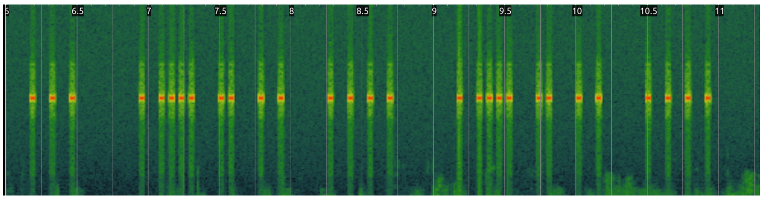


Figure 4: Detail of the ultrasonic signal at 19KHz.

3.2 Analysis of the network traffic

As noted before, the Fidzup Diagnostic mobile application includes a feature to resolve the location associated with a signal received by the app. We assumed that this resolution was not done locally and was made with the help of a remote service. Using a soft Wi-Fi access point (`hostapd`) we captured the traffic coming out of a smartphone running the Diagnostic app.

Studying the traffic when the app is receiving a signal we identified DNS queries made to the domain `api.spotinstore.com`. The corresponding answered showed that this domain is a CNAME record for `spotinprod.herokuapp.com` indicating that the service is hosted on the herokuapp platform.

In a second time we observed that the application send HTTP GET requests. By repeating the operation, we were able to obtain requests to two distinct URLs :

1. `http://api.spotinstore.com/api/v2/mobile/zones/3022.json?signature=3f43f9a8b61e3e0fb8e7ecf96d9b328d&timestamp=1498592150`

2. `http://api.spotinstore.com/api/v2/mobile/zones/3020.json?signature=a32eea6ad7a62c389732d32e7504afcc&timestamp=1498592166`

Apart from the `signature` and `timestamp` parameters, those URLs differ by the last element of the path : `3022` and `3020`. This element looks like a code that could designate a location or a zone, as suggested by the previous element of the path: `zones`. Two distinct codes were observed because the ultrasound signal we used contains two distinct frames, which must correspond to different zones.

The structure of the URL thus appear to be the following :

`http://api.spotinstore.com/api/v2/mobile/zones/<zone_code>.json?signature=<signature>&timestamp=<timestamp>`

where:

- `zone_code` is the code associated to a zone;
- `signature` is a token specific to each request;
- `timestamp` is the unix timestamp at which the request is generated.

The `signature` token appear to be pseudo-random as there is no evident pattern between the various requests. It is likely a token that is generated using a custom algorithm in order to restrict the access to the service.

3.3 Analysis of the mobile applications

For the last part of our analysis, we focused on the mobile applications: Fidzup Diagnostic and Fidzup Live. By decompiling those applications we were able to access information that helped finishing the reverse engineering of the system. The decompilation of the `.apk` was performed using `Dex2jar`[8] to extract the `.class` files and then `Luyten`[9] to analyze those `.class` files.

In the Fidzup Live application we identified a class, `com.fidzup.spotinstore.audio.SpotInStoreAudio` including information about the signal characteristics and the structure of the frames. First we found a set of variable specifying some elements of the signal.

```
basicAudioAnalyzerConfig.frequency = 19000.0f;
basicAudioAnalyzerConfig.samplingRate = 44100.0f;
basicAudioAnalyzerConfig.windowSize = 512;
basicAudioAnalyzerConfig.pulseDuration = 69.66f;
basicAudioAnalyzerConfig.pulseRatio = 0.33333334f;
basicAudioAnalyzerConfig.analysisDuration = (int)(
    basicAudioAnalyzerConfig.pulseDuration * 32.0f * 3.2f);
basicAudioAnalyzerConfig.signalSize = 32;
basicAudioAnalyzerConfig.bitcounts = new int[] { 15, 17, 19, 13,
    11, 21, 23, 9, 7, 25, 27 };
return basicAudioAnalyzerConfig;
```

The most interesting are the following :

- `frequency = 19000.0f;` confirms that the carrier frequency is indeed 19KHz;

[8] https://github.com/pxb1988/dex2jar

[9] https://github.com/deathmarine/Luyten

- `pulseDuration = 69.66f;` confirms the previous approximation of the symbol duration;

- `signalSize = 32;` suggests that the total size of a frame is 32 bits.

In addition we found methods and constants whose names suggested the use of a cyclic redundancy check (CRC), which is common practice in wireless communications.

```
    private static final int CRC_MASK = 255;
    private static final int CRC_POLYNOMIAL = 359;
    private static final int CRC_POLYNOMIAL_SIZE = 9;
    private static final int CRC_REMAINDER_SIZE = 8;
    private static final int DATA_LENGTH = 18;
    private static final int DATA_MASK = 262143;
    private static final int INIT_MASK_W_OFFSET = 2080374784;
    private static final int INIT_VALUE_W_OFFSET = 1409286144;
```

These values indicate the characteristics of the CRC used. For instance that the polynomial used by the CRC is on 9 bits (`CRC_POLYNOMIAL_SIZE`) and its decimal value is 359 (`CRC_POLYNOMIAL`) but that the output is truncated to keep only 8 bits (`CRC_REMAINDER_SIZE`). As a matter of fact the method computing the CRC is available:

```
    private int crc(long n, int i, final int n2, final int n3) {
        final long n4 = n2;
        n <<= 64 - i;
        long n5;
        for (--i; i >= 0; --i) {
            n5 = n;
            if ((n & 1L << 63) != 0x0L) {
                n5 = (n ^ n4 << 64 - n3);
            }
            n = n5 << 1;
        }
        return (int)(n >>> 64 - n3 + 1);
    }
```

Looking at the decompiled code of the Fidzup Diagnostic application, we found a class, `com.fidzup.spotinstore.diagnostic.ListenActivity` that contained information about the construction of the HTTP requests identified during the network traffic analysis. More particularly, we found the following code that confirmed the structured of the request:

```
  private int r;
  ...
  final long n = new Date().getTime() / 1000L;
  final String string = "http://api.spotinstore.com/api/v2/mobile/
      zones/" + this.r + ".json?signature=" + com.fidzup.
      spotinstore.diagnostic.g.a(String.valueOf(this.r) + n + "
      XaSnSXFLaN") + "&timestamp=" + n;
```

In this portion of code we can quickly identified that `this.r` is in fact the zone code and that n contain the timestamp. It is important to note that the zone code is represented by an `int`. We can therefore assume that the zone code is encoded on 32 bits. More interesting, is the generation of the signature. It is obtained by applying the method `a` from the class `com.fidzup.spotinstore.diagnostic.g` to the concatenation of the zone code, the time stamp and a constant string.

The method a, shown below, is in fact the computation of the MD5 digest. More specifically it takes a String as input and returns a String containing the digest in a hexadecimal representation.

```
 public static String a(String s) {
   try {
     final MessageDigest instance = MessageDigest.getInstance("MD5
       ");
     instance.update(s.getBytes());
     final byte[] digest = instance.digest();
     final StringBuffer sb = new StringBuffer();
     for (int i = 0; i < digest.length; ++i) {
 for (s = Integer.toHexString(digest[i] & 0xFF); s.length() < 2; s
      = "0" + s) {}
 sb.append(s);
     }
     return sb.toString();
   } catch (NoSuchAlgorithmException ex) {
   ex.printStackTrace();
     return "";
   }
}
```

The signature is therefore computed as follows : `<signature> = MD5(<zone_code> | <timestamp> | "XaSnSXFLaN" )`

3.4 Wrapping up

Based on our previous analysis we can assert the following:

- The signal is transmitted using an ASK modulation using a carrier frequency of 19KHz with symbol duration of 0.0669 milliseconds.
- The signal is transmitted by 32-bits long frames that are repeated.
- The frame includes a CRC of 8 bits.

Based on those information we attempted to rebuild the internal structure of a frame. Among its 32 bits, we know that 8bits belong to a CRC. In addition we have signal corresponding to the zone code 3020 respectively 3022 that have been transmitted. The binary representation of those codes are the following :

- 3020 : `1011 1100 1100`
- 3022 : `1011 1100 1110`

If we look at the transmitted frame we notice that they include the two previous sequences of bits. Starting from there, we can observer that, in the frame, those sequences are always preceded by the same sequence: `0101 0100 0000`. It make sense to consider that the zone code is in fact encoded on 16 bits and that the last four 0 of this sequence are in fact the most significant bits of the zone code. This leaves an 8 bits sequence `0101 0100` that always precedes the zone code. This sequence is the preamble, a pattern marking the beginning of a frame.

The last 8 bits of the frame are likely the CRC. By recomputing the CRC of the area code, we found the exact value found in the last 8 bits of the frame,

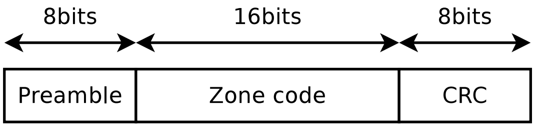


Figure 5: Structure of a frame with an 8 bits preamble, 16 bits payload, and 8 bits CRC.

confirming that this is indeed a CRC computed over the area code and not the full frame (the preamble is not included in the CRC computation).

Finally a frame is composed of an 8 bits constant preamble (`0101 0100` or 0x54), followed by a 16 bit value corresponding to the zone code and ends with an 8 bits CRC of the zone code (see Figure 5).

4 Ultrasonic signal generation and jamming

Having reverse engineered the ultrasonic encoding, we can now generate our own signal in order to interact with the system. Based on the GNURadio framework we created tools to interact with the SpotInStore system.

4.1 Signal generation

In a nutshell, our signal generation tool generate an ASK signal by multiply a carrier signal at 19KHz with a rectangular signal encoding the sequence of bits. In order to get a clean signal, we then use pass-band filter centered on the carrier frequency. The result is then fed to an audio sink that output the signal through the computer speakers. The rectangular signal is created by taking a frame in its binary form and repeating each bit n times. n depending on the sampling rate of the setting and the symbol duration : $n = samplingRate * d$. The GNURadio companion sketch of the generator is presented on Figure 7.

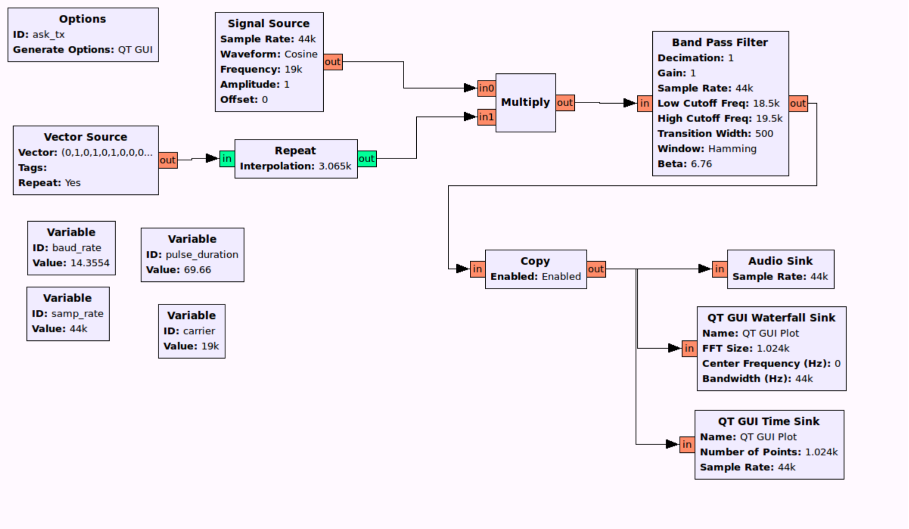
Using this generator, we've been able to generate signals that were recognized by both Fidzup Diagnostic and Fidzup Live applications. In particular, we were able to trigger pop-up in the Fidzup live application by generating signal for the zone code used in the demonstration video (3020 and 3022).

The ability of generating arbitrary code can be used for various purposes. It could be used to annoy users by generating notifications, analyze unknown or locked features of an application, or simply to raise user awareness by demonstrating how those systems are working.

4.2 Ultrasonic jamming

Using this same generator, it is possible to jam SpotInStore ultrasonic signal. The best approach we have identified is to generate an ASK signal corresponding to a random sequence of bits. The two signals will then be merged at the receiver, and the temporal structure will be destroyed rendering the original signal impossible to decode.

Apart from denial of service, jamming can be seen as a mean of privacy protection. Indeed, the jammer will generate a small area were the ultrasonic


Options
ID: ask_tx
Generate Options: QT GUI

Signal Source
Sample Rate: 44k
Waveform: Cosine
Frequency: 19k
Amplitude: 1
Offset: 0

Vector Source
Vector: (0,1,0,1,0,1,0,0,0...
Tags:
Repeat: Yes

Repeat
Interpolation: 3.065k

Variable
ID: baud_rate
Value: 14.3554

Variable
ID: pulse_duration
Value: 69.66

Variable
ID: samp_rate
Value: 44k

Variable
ID: carrier
Value: 19k

Multiply

Band Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 44k
Low Cutoff Freq: 18.5k
High Cutoff Freq: 19.5k
Transition Width: 500
Window: Hamming
Beta: 6.76

Copy
Enabled: Enabled

Audio Sink
Sample Rate: 44k

QT GUI Waterfall Sink
Name: QT GUI Plot
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 44k

QT GUI Time Sink
Name: QT GUI Plot
Number of Points: 1.024k
Sample Rate: 44k

Figure 6: Signal generator in GNURadio.

system is ineffective. Since ultrasonic beacons can expose the user to serious privacy issues, and given the difficulty to control all the channels through which the signal can be received, ultrasonic jamming can be an attractive solution for users.

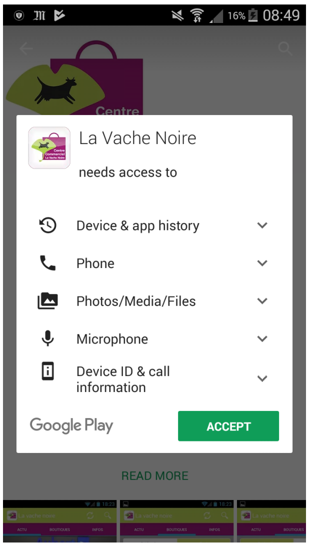
5 Analysis of a real world deployment

The Spotinstore system has been deployed in the shopping center *La Vache Noire*. As part of this deployment, a dedicated mobile featuring the Spotinstore reception framework has been published [10]. We collected audio sample at the shopping center in order to test the application. However the Android application was only working partially on our phone and we weren't able to use the features associated with the ultrasonic beacons.

5.1 Application permissions

Looking at the permissions required by the application, we can see that the microphone permission, required to acquire the ultrasonic signal, is included. However we notice that geolocation permission is not required, suggesting to the user that this application cannot obtain the location of the user. This is obviously not the case as this application feature an ultrasonic localization system. Even if the application can obtain the geolocation of the user in a limited number of locations it still access the location to provide location based services and can potentially collect it. Furthermore is capable of getting a very accurate geolocation. Indeed, as advertised by Fidzup, each ultrasonic beacon can cover a limited area such as a small shop ore an aisle in a larger shop.

[10] \url{https://play.google.com/store/apps/details?id=com.fidzup.darkmall}



La Vache Noire

needs access to

Device & app history

Phone

Photos/Media/Files

Microphone

Device ID & call information

Figure 7: Permissions ask by the Android application of the shopping center *La Vache Noire*.

The case of applications geolocating users through side channels without asking for geolocation permission is not new [1]. Using this kind of technique to obtain the user location without its consent has been considered as a deceptive practice by the Federal Trade Commission of the USA [3].

5.2 Zone codes

Using audio samples collected in the shopping center *La Vache Noire*, we were able to identify the codes broadcast by the ultrasonic beacons. We did so, using the Fidzup diagnostic application: we played the samples next to a phone running the application while monitoring the network traffic of the phone. The zone code was obtained from the GET request sent by the app as in section 3.2:

```
http://api.spotinstore.com/api/v2/mobile/zones/2013.json?
signature=406c4003cc8786cfd94450865dddc308&timestamp=1500657161
```

Over the eight samples, we always obtained the same zone code : `2013`. This suggests that only one zone is configured at this shopping center. The ultrasonic system is likely used to detect if the user is in the shopping center, and the signal is likely broadcast by the audio system of the shopping center.

6 Conclusions

The system studied in this paper relies on a relatively simple coding and modulation scheme for the ultrasonic signal. Using open source tools such as GNURadio it is thus straightforward to analyze the signal and to generate arbitrary codes. We show that jamming of the signal is feasible and that it can be a solution to protect users' against tracking. The study of a real world deployment demonstrates that those systems can be used to track the location of users without asking their permissions.

As we demonstrated in this paper, those ultrasonic signals can be decoded by anybody. This exposes the users to tracking by other entities. For instance, any mobile application having access to the microphone could geolocate users visiting an area equipped with beacons. A malware running on a device could achieve a similar result. The generalization of those systems will mean that more and more areas will be covered by ultrasonic beacons which will increase the potential for unsolicited tracking.

The tool presented in this documents are available online : `https://github.com/cunchem/SIS-Ultrasonic-Beacon`

7 Acknowlegements

The authors would like to thank Rollo Tomassi for telling us about this system and Marc Jeanmougin (@___Mc__) for helping us collecting audio samples.

References

[1] Jagdish Prasad Achara, Mathieu Cunche, Vincent Roca, and Aurélien Francillon. Short Paper: WifiLeaks: Underestimated Privacy Implications of the ACCESS_wifi_state Android Permission. July 2014.

[2] Daniel Arp, Erwin Quiring, Christian Wressnegger, and Konrad Rieck. Privacy threats through ultrasonic side channels on mobile devices. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, pages 35–47. IEEE, 2017.

[3] Federal Trade Commission. Mobile Advertising Network InMobi Settles FTC Charges It Tracked Hundreds of Millions of Consumers' Locations Without Permission, FTC, June 22nd 2016, June 2016.

[4] Vasilios Mavroudis, Shuang Hao, Yanick Fratantonio, Federico Maggi, Christopher Kruegel, and Giovanni Vigna. On the Privacy and Security of the Ultrasound Ecosystem. *Proceedings on Privacy Enhancing Technologies*, 2017(2):95–112, 2017.