Inaudible Sound as a Covert Channel in Mobile Devices

Luke Deshotels
North Carolina State University
alecdeshotels@gmail.com

Abstract

Mobile devices can be protected by a variety of information flow control systems. These systems can prevent Trojans from leaking secrets over network connections. As mobile devices become more secure, attackers will begin to use unconventional methods for exfiltrating data.

We propose two sound-based covert channels, ultrasonic and isolated sound. Speakers on mobile devices can produce frequencies too high for most humans to hear. This ultrasonic sound can be received by a microphone on the same device or on another device. We implemented an ultrasonic modem for Android and found that it could send signals up to 100 feet away. We also determined that this attack is practical with the transmitter inside of a pocket. Android devices with vibrators can produce short vibrations which create isolated sound. These vibrations can be detected by the accelerometer, but they are not loud enough for humans to hear. If performed while the user is not holding the device, the vibrations will not be noticed.

Both covert channels can stealthily bypass many information flow control mechanisms. We propose several simple solutions to these vulnerabilities. In order to guarantee information flow control, sound-based channels must be regulated.

1 Introduction

Users trust mobile devices with an unprecedented amount of sensitive information. Financial credentials, private photographs, location data, correspondence records, and more can be found on a smartphone. This collection of sensitive data has attracted the attention of government agencies, criminals, and other sophisticated attackers.

In order to protect user privacy, several security mechanisms have been proposed that enforce information flow policies and restrict network usage. However, many of these do not consider inaudible sound as a method for data exfiltration. This paper focuses on the case where users have taken steps to secure their data, but attackers are willing to use unconventional means to extract it.

1.1 Inaudible Sound

Sound can be defined as vibrations in matter. We usually perceive sound as vibrations in air. These vibrations are made up of waves with measurable frequencies.

Humans with ideal hearing can perceive sound frequencies within the range of 20hz to 20khz. However, sensitivity to high frequencies declines rapidly with age as shown in Figure 1. Most humans over 18 years of age cannot hear frequencies above 17khz [4]. Ultrasonic sounds are those frequencies too high for humans to hear.

A significant amount of inaudible frequencies can be produced by mobile device speakers. The small speakers on such devices are better at creating high frequencies, so this paper will focus on ultrasonic sound. These inaudible frequencies can also be detected by microphones. Therefore, machines can communicate with sound frequencies that most humans cannot hear.

Hanspatch and Goetz [9] presented methods for bypassing security with ultrasonic sounds. They implemented their experiments on laptop computers. Our paper is intended to present and evaluate proofs of concept for mobile devices such as smartphones and tablets. It also discusses abuses and solutions specific to mobile devices. They did not work with vibrators which are primarily available on mobile devices. They also did not test transmissions through fabric.

The vibrator on many Android devices can produce vibrations that can be felt, but not heard. This is performed by activating the vibrator for a very short amount of time. The vibrations produced can be detected by the accelerometer or microphone of the same device.

These vibrations are still considered sound although

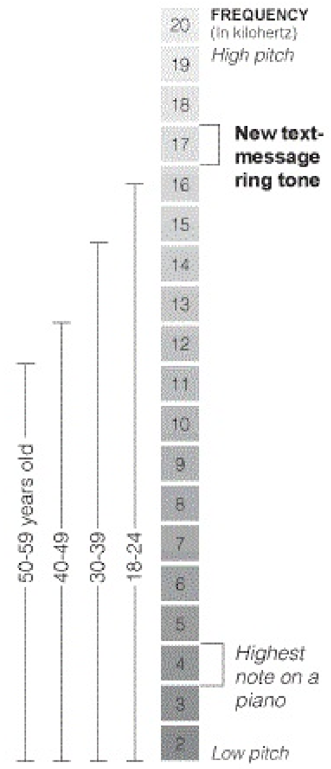

Figure 1: Chart showing hearing ranges by age group. It also references a ring tone used by teenagers to avoid detection by adults in schools [4].

they primarily occur in the device instead of in the air. Some of the vibrations are transferred to the air, but the amplitude is too low for humans to hear them. Therefore, sound produced in this way is also inaudible.

Subramanian et al. [15] explore methods for using non-radio signals for malicious communication in wireless sensor networks. Sound and seismic waves are considered among these methods. However, their implementations are not stealthy and were implemented on wireless sensors. They do not measure the proximity required for transmitting messages via sound. Our paper evaluates the use of inaudible sound to stealthily bypass information flow control on mobile devices. We determine the maximum range at which our implementation can send messages, and we also discuss the implications of this technology for mobile devices.

1.2 Bypassed Security Mechanisms

We will reference the security mechanisms of Android because of its open source nature and the tools available to extend Android security.

Android applications are regulated by a permission system that enforces the principle of least privilege. At install time, users are presented with a list of dangerous permissions the application requires to function correctly. These permissions include network access, access to message history, access to contacts, etc. If the user is willing to trust the application with these permissions, they allow the installation to continue. If not, the installation is aborted. Unfortunately, access to the speakers, vibrator, and accelerometer are not considered dangerous actions and do not require explicit permissions.

Several third party security extensions have been proposed for Android. Many of them do not prevent sensitive data from being exfiltrated as sound.

Kirin [8] allows users to set custom policies that prevent dangerous combinations of permissions. However, as noted previously, these attacks do not require any significant permissions.

TaintDroid [7] tracks sensitive data as it moves within and out of Android devices. TaintDroid can prevent secrets from being exported over network connections, but it does not prevent secrets from being broadcast as sound.

MockDroid [6] provides false data when sensitive data is requested by an application. It can also make an application believe that network access is unavailable. However, an application may be trusted with real data in order to obtain functionality. It can then export the data as sound instead of using the network.

1.3 Comparison to Bluetooth

The attacks performed with ultrasonic sound are similar to those performed with Bluetooth devices because they are both alternatives to conventional internet connections (WiFi, 4G, etc.). Therefore, it is worth noting differences between the two mediums. Bluetooth has a significantly more developed and secure infrastructure than sound on mobile devices. For example, Bluetooth access is listed as a dangerous permission on Android that users must explicitly allow. Bluetooth also utilizes a pairing process so that the user must explicitly allow his device to be paired with another Bluetooth enabled machine. Ultrasonic sound does not require a permission and can emit sounds to anything that can hear them regardless of a pairing process. In the context of this paper, the absence of security for sound makes it more interesting than Bluetooth as a network alternative.

1.4 Contributions

This work provides the following contributions.

- The identification of inaudible sound as a covert channel that allows applications to bypass existing security measures.

- A proof of concept implementation of an ultrasonic modem to covertly exfiltrate data on Android devices.
- Evaluation of the performance of this proof of concept including data on its range, bitrate, and ability to transmit through fabric.
- Discussion of the potential abuses of and solutions to the problem of inaudible sound on mobile devices.

The remainder of this paper proceeds as follows. Section 2 presents methods for abusing inaudible sound and preventing detection. Section 3 describes the design of our implementation and experiments. Section 4 evaluates our experiment results. Section 5 discusses observations, assumptions, and future work. Section 6 discusses solutions to the problem of inaudible sound as a covert channel. Section 7 describes related work. Section 8 concludes. Section 9 acknowledges non-authors that contributed directly to this work. Section 10 discusses the availability of our implementation.

2 Abuses of Sound-Based Channels

Data exfiltration can be performed with inaudible sound in at least two ways. Assume in each case that application A is trusted with secret data but has been secured such that it cannot exfiltrate this data over a network. This can be achieved with several information flow tools or simply by restricting the privileges of the application.

2.1 Intra-Device Communication

The first method is intra-device communication between applications that would otherwise be separated. Application A can not communicate directly with application B because A has secret data. Application B has network access and other privileges, but is never trusted with sensitive data. Application A can transmit the secret as sound and have the colluding application B on the same device receive the message with the microphone. Alternatively, A could use the vibrator to emit inaudible pulses and have B receive the message with the accelerometer. Application B can now access the data and transmit it over the network.

In using the vibrator, there is a moderate risk of detection. If the user is holding the device or is carrying it in a pocket, they may feel the vibrations and become suspicious.

It would be ideal to utilize the vibrator while the user is asleep and away from the device. Malware can assume the user is sleeping if the following conditions are met. First, the device is charging. Second, the accelerometer has not recorded any movement for the past hour. Third, the time is between midnight and 5 AM. These conditions could be optimized, and there may be other useful observations to utilize.

2.2 Inter-Device Communication

Second, an agent may compromise a targeted subject's mobile device with a Trojan horse to perform an inter-device attack. However, the subject has secured his device, such that the Trojan cannot exfiltrate data over the network. Instead of using the network, the Trojan can broadcast the data as ultrasonic sound for the attacker to record. The recording device could be carried by the attacker or hidden in areas where the subject normally spends time. Resourceful attackers also have access to long distance listening equipment such as laser microphones.

The same attack can be applied at a large scale with a popular application A and a network of recording devices. Assume a large corporation or government agency wanted to track security conscious users. This organization could distribute applications that are pre-installed on mobile devices and impossible to uninstall. Such applications are very common and are referred to as bloatware. Users can enhance their privacy with security extensions, but sound based attacks could bypass these. This large organization could also control a wide network of recording devices at various building entrances. These recording devices can pick up sensitive data from users as they pass by and leak data as ultrasonic sound.

3 Design

We implemented proofs of concept for both isolated sound and ultrasonic sound. The isolated sound implementation is fairly crude but serves its purpose. The ultrasonic sound implementation is tested to determine the attack's practical limits.

3.1 Isolated Sound Experiment

We used two Android devices running third party apps to demonstrate that a device could produce isolated sound. The transmitter was a Samsung Galaxy S4 running the Vibration loop [5] application. The receiver was a Google Nexus 7 (2013 edition) running the Accelerometer Monitor [1] application.

The transmitter was set to vibrate for 1 millisecond, wait 500 milliseconds, and then repeat the pattern. The transmitter was placed on top of the receiver. The receiver displayed the accelerometer data with the Accelerometer Monitor application. The transmitter began the defined vibration pattern.

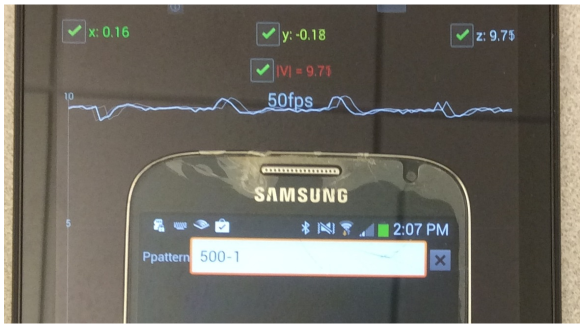

Figure 2: Photo of our isolated sound proof of concept

We took the photo shown in Figure 2. The hills and valleys shown in the photo demonstrate that the accelerometer is detecting significant changes. These changes could be used to encode sensitive data and share it between applications that should not be allowed to communicate. We could not hear the vibrations being produced unless we held our ears less than one foot from the transmitter.

While this proof of concept uses multiple devices, it is intended to provide evidence for an intra-device transmission. If this attack works for two touching devices, it is likely to work with a vibrator and accelerometer in the same device.

3.2 Ultrasonic Sound Experiment

We implemented an ultrasonic modem on Android devices to test its practical limits for inter-device communication. The transmitter uses frequency shift keying (FSK) to transmit digital data, and the receiver decodes the FSK message to obtain the digital data. This technique was tested to determine its maximum bitrate and distance while remaining stealthy. We also experimented to determine the feasibility of this attack from within a pocket.

3.2.1 Materials

FSK functions by encoding binary 1's and 0's as two different frequencies. Each frequency must be predetermined. Our experiments use 18khz as the low frequency and 19khz as the high frequency. These 1's and 0's may be referred to as bits or symbols. If the receiver senses a high tone, it receives a 1. If it receives a low tone, it receives a 0. FSK also uses a set amount of time for each symbol to maintain synchronization and detect repeats of the same symbol. An illustration describing FSK is presented in Figure 3.

The frequencies 18khz and 19khz were chosen for various reasons. Lower frequencies should be able to travel

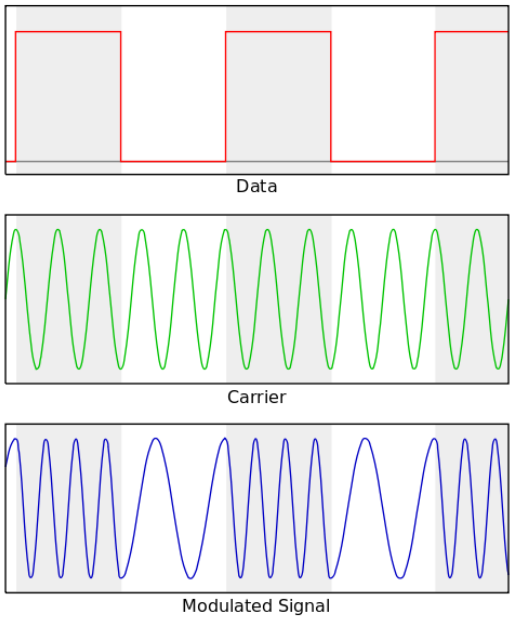

Figure 3: Example of a digital signal represented as analog waves in frequency shift keying. [3].

farther while maintaining their integrity. These frequencies were still too high for the author and experimenters to hear. Hanspatch and Goetz [9] used a frequency of 18,600 Hz. We found that error rates were significantly higher when we used 19khz and 20khz. This is likely due to hardware limitations.

The transmitter software is a Matlab script that creates an audio file in .wav format consisting of three parts. The first is a preliminary tone. This is a consistent tone of a single frequency which helps the receiver roughly identify the signal of interest in a larger recording. We used a preliminary tone consisting of 128 consecutive low frequency (18khz) symbols. The second is a synchronization sequence. This is a complex but predefined pattern of high and low frequency tones. This pattern allows the receiver to pinpoint exactly where the intended message begins. The synchronization pattern was also 128 symbols long. The third is the digital data encoded as a series of high and low frequencies. The digital data is a string of ASCII characters that have been converted to binary symbols. The .wav files produced by this script can be played on arbitrary devices including Android devices.

The receiver software is an Android application that records arbitrary sounds as an audio file. It then decodes that audio file into digital data. The receiver has a priori knowledge of everything but the message contents. This a priori knowledge includes the frequencies used, the length of the preliminary tone, the synchronization pattern, the bitrate, etc. Once the receiver has pinpointed

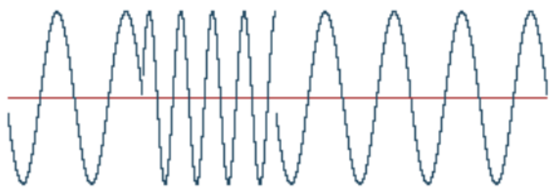

Figure 4: Example of discontinuity between frequency shifts. These abrupt changes can cause audible clicks. [2].

the message contents in the audio file, it can decode it as binary bits based on the high and low signals detected. This binary sequence is then converted into ASCII characters and displayed on the Android device screen.

The .wav files were produced on a PC and exported to be played on a single Nexus 7 (2013). The receiver application was installed on 3 different Nexus 7 (2013) devices.

3.2.2 Optimizations

We also implemented some redundancy in the receiver to make the process more robust. Each audio file recorded is decoded 10 times with a slightly different starting point for each. The starting point is always a point before the preliminary tone begins. Different starting points can affect the decoder's ability to correctly identify the beginning of the message contents. This optimization causes the decoding process to require more time and energy, but it does not affect the transmission cost.

Our second optimization addresses the problem of audible clicks during phase shifts. Each symbol is represented as a cosine wave, but the waves do not connect seamlessly. When transitioning between symbols, there may be an abrupt change in the wave's phase. This is illustrated in Figure 4 and is manifest in the speakers as an audible click. Hanspatch and Goetz [9] also noted this phenomenon. We were able to mitigate this issue by gradually reducing the amplitude at the beginning and end of each symbol. This caused the clicks to become so quiet that they are inaudible.

3.2.3 Calculating Error Rates

The multiple decoding optimization made calculating error rates non-trivial. Some decoding attempts were significantly more accurate than others and a method for prioritizing them was necessary. We addressed this problem by using a predetermined sequence weaved into the message. A comparison to this predetermined sequence was used to determine the value of each decoding attempt. The bit error rate is determined for the decoding attempt with the highest value.

The message we transmitted is the following character string.

```
0@1,2^3^4l5Q6n7C8f9W
```

The numbers 0 through 9 are expected at every other character position. The other characters were randomly chosen and constitute the data we wish to transmit. For example, the message "strawberry" would be encoded as the following string.

```
0s1t2r3a4w5b6e7r8r9y
```

This "strawberry" example will be used for the remainder of this subsection for simplicity. Note that the actual experiments used the previously listed random ASCII sequence.

The number of correct, expected characters in each decoding attempt is calculated to select the most valuable result. The expected characters are then removed from the chosen string leaving the arbitrary data. In our example, this could be the message "strawberry" or a result containing bit errors. Let us assume the message "ttrawberry" was produced after removing the expected characters.

This received message is then converted into its binary ASCII representation and compared to the intended message's binary representation. Comparing the binary code for "strawberry" and "ttrawberry" yields a difference of three bits. These three bit errors out of a possible 80 bits yields a bit error rate of 3.75%.

3.2.4 Bitrate Experiment

We experimented to determine the maximum bitrate our implementation could support at a minimal distance. The minimal distance was obtained by placing two devices back to back without cases.

Once the two devices were in position, the receiver was set to record, and the transmitter played the encoded .wav file. After the file finished playing, the receiver was set to decode. Once the receiver finished decoding, it displayed its results to the screen.

This experiment was performed in a computer lab. Some ambient noise may have come from air conditioning vents, computer fans, a refrigerator, and researchers typing.

3.2.5 Distance Experiment

A similar experiment was performed to measure the maximum distance at which messages could be passed and accurately decoded. This experiment used a consistent bitrate of 8.61 bits per second, but the distance between devices was varied.

In this experiment, the devices were placed in cases that allowed them to be propped up in their tallest orientation. The transmitter's speaker was turned to face the screens of the receiving devices. One Nexus 7 was used to broadcast to three receiving Nexus 7's.

The frequencies, preliminary tone, synchronization tone, and message were all the same as in the bitrate experiment. The sequence of events (record, play tone, decode, evaluate) was also identical to the bitrate experiments.

This experiment was performed in a large hallway in a research building. Ambient noise could have been caused by air conditioning, people walking, and use of doors.

3.2.6 Transmitter in Pocket Experiment

Mobile devices spend a significant amount of time in pockets. Therefore, leaking data via ultrasonic sound is much more practical if the signals can pass through fabric.

In order to test these conditions, we placed the transmitter inside the leg of a pair of denim jeans during transmission. This experiment took place at a distance of 20 feet and under the same conditions as the distance experiments.

4 Evaluation

The results obtained by our experiments suggest that transmitting messages with ultrasonic sound on mobile devices is very practical. The bitrates and distances observed allow for several interesting abuses of this covert channel.

4.1 Bitrate Experiment Results

Our bitrate tests revealed an unexpected limit in our ability to remain stealthy. The Nexus 7 devices produced small amounts of noise outside of the desired frequencies. There were also traces of the clicks which we had reduced by adjusting amplitude during frequency shifts. Neither of these phenomena were audible during transmission at bit rates below 345 bits per second. However, at bitrates above 345 bits per second, the clicks and noise began to form audible tones.

A different Android device using a more sophisticated implementation may be able to achieve higher bitrates. In this implementation, we found the effective bitrate limit of stealthy ultrasonic sound to be 345 bits per second.

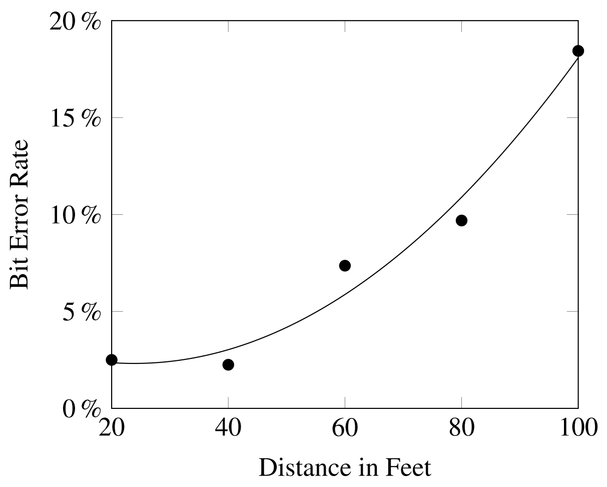

Figure 5: Average bit error rates at 20 foot intervals. The quadratic regression is also plotted.

4.2 Distance Experiment Results

The distance experiment results suggest a decline in transmission reliability as distance increases. Such a decline is expected. We observed a significant increase in error rates at 100 feet.

Figure 5 illustrates our experiment results. These error rates can be overcome with parity schemes, repeated transmissions, and other solutions. We chose to stop experimenting at 100 feet, but transmission at greater distances with the same hardware may also be possible. Our data formed the following formula when we calculated the quadratic regression.

$$errorRate(x) = 0.00271786x^2 - 0.129543x + 3.862$$

This formula has an adjusted R^2 score of 0.950 for the fit of the data to the regression.

During experimentation, we encountered a limitation of our character based decoding. Some attempts to decode transmissions produced strings too short to evaluate. We suspect that an end of string character was encountered which caused the rest of the transmission to be omitted. Figure 6 plots the rates of occurence of this phenomenon. The chart contains an outlier at 60 feet where this error did not occur during experiments. Further testing would likely produce a quadratic regression similar to our bit error rate. This is an implementation specific phenomena, but we still value these results.

4.3 Pocket Experiment Results

Our results suggest that transmission from within a pocket is feasible. With the transmitter inside the leg of a pair of denim jeans, the bit error rate for 20 feet was 1.46%.

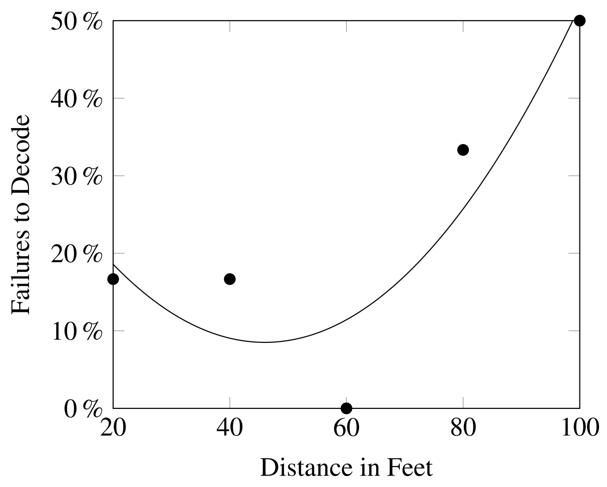

Figure 6: Occurrence rate of failures to decode due to end of string character. The quadratic regression is also plotted.

This error rate is lower than the rate we measured without the denim barrier. We suspect that at higher bitrates and greater distances, the barrier would have a more significant effect. However, these results suggest that an infected phone in a pocket can reliably leak data to a receiver at least 20 feet away.

5 Discussion

We were able to make several informal observations during our experiments. We also made several inferences based on our results.

During experimentation we informally monitored battery use of the tablets. Transmission of the ultrasonic signal did not seem to use a significant amount of power, but decoding messages was resource intensive. An infected device does not need to transmit sound at all times. It could transmit once every few minutes, or a more sophisticated implementation could probe for a receiver before transmitting. With our implementation, the receiver must spend a significant amount of time decoding the transmission it recorded. This leads to more power usage, but the attacker is free to plug in his listening device after recording. For an intra-device attack using ultrasonic sound, power usage of the receiver is more relevant. One solution for intra-device attacks is to perform the transmission and reception while the phone is charging.

The bit rate we chose for distance experiments, 8.61, is fairly low, but it is still sufficient for leaking sensitive data. IDs, social security numbers, credit card numbers, coordinates of locations visited, passwords, and more could be leaked in less than one minute. Higher bitrates are also feasible, but the recording device would require closer proximity to the transmitter.

5.1 Assumptions

We made the following assumptions as conditions of the isolated sound attack being stealthy. First, the device is not on a drum-like object. For example a filing cabinet's thin walls and hollow center will amplify the pulses and make them audible. Second, the device is not touching the user. In this case the user would be able to feel the vibrations. Third, the user's ear is not within 1 foot of the device. Fourth, the user does not have an animal such as a dog that could detect and react to faint sounds. Other similar conditions may exist.

We made the following assumptions as conditions of the ultrasonic sound attack being stealthy. First, people near the device cannot hear 18khz sounds or higher frequencies. Many humans under 18 years of age and some exceptional adults can hear these frequencies. Second, only the intended frequencies are being produced. The Nexus 7 devices also produced very faint noise at frequencies other than 18khz and 19khz. These sounds were only audible if the experimenters ear was within 6 inches of the device's speaker. This noise is likely due to a limitation of the speakers used. We noted that an iPad Mini with Retina Display did not produce such noise. Third, the user does not have an animal such as a dog that could detect and react to ultrasonic sounds. Other similar conditions may exist.

5.2 Future Work

Researchers could address our assumptions, improve our experiment process, and implement the proposed solutions.

Two assumptions can be mitigated. First, by using the highest frequencies available on mobile device speakers (about 20khz) the chances of humans detecting the sounds are significantly lower. This is assuming the hardware used can produce pure tones at these frequencies. Our experiments suggested that Nexus 7 tablets were not suitable for using frequencies above 19khz. Second, devices other than Nexus 7 tablets may be able to produce the frequencies required without faint noise on other frequencies. Our iPad Mini with Retina Display seemed to produce high frequency sounds without noise.

The experiment could be improved by using a more robust, sophisticated transmission protocol. Many techniques exist for wireless communication that could have been applied to the ultrasonic sound implementation. Such techniques include phase shift keying, parity checking, and receivers that respond when messages are received.

A continuous connection between the device and receiver may yield a more accurate maximum distance for ultrasonic communication. This connection could be established between both devices at a close proximity. The space between the devices could be increased until the connection fails and the distance of this event could be recorded.

Finally, practical solutions to inaudible sound abuse should be implemented and distributed. The solutions we propose may help, or researchers may develop better ones.

6 Solutions

There are several potential solutions to the ultrasonic sound problem. First, speakers could be installed that can only produce those frequencies that most people can hear. Second, physical filters could be placed over the speakers that block ultrasonic sounds, but not those with wider wavelengths. Third, an indicator that the speakers are in use could be displayed to the user. This would be similar to the lights that often indicate when a camera is currently recording. Fourth, audio files could be filtered before reaching the speakers to remove any ultrasonic frequencies. Fifth, a trusted application could listen for unusual activity in the range of ultrasonic sound and warn the user if such activity is detected. Sixth, an explicit permission to use the speakers could be added to the Android OS. Seventh, existing security tools could begin to monitor sound as a channel that is a potential means of exfiltrating data.

There are also solutions to isolated sound attacks that use the vibrator. First, the minimum time to use the vibrator could be increased from 1 millisecond to a more audible amount of time. Second, explicit permissions to use the vibrator and accelerometer could be added to the Android OS. Third, the sensitivity of the accelerometer could be reduced so that it cannot detect inaudible vibrations. Fourth, a log of vibrator use might reveal unusual activity upon inspection. Fifth, existing security tools could begin to monitor the vibrator as a channel that is a potential means of exfiltrating data.

7 Related Work

We have identified three major topics related to this article. These topics are sensory malware, sound-based machine communication, and information flow control on Android.

Sensory Malware The sound-based attacks we have proposed could be utilized by sensory malware. Sensory malware is any software that abuses the sensors of the infected device.

PlaceRaider [16] uses the device camera to create a three-dimensional view of the victim's surroundings. This three-dimensional view is created by combining multiple photographs that were taken stealthily by the malware. This view enhances the attacker's ability to extract sensitive data from the victim's surroundings.

TapPrints [10] determines the user's keystrokes by using accelerometer data. Each key being tapped causes a slightly different offset of the device's position depending on the keyboard layout. These patterns can be used to turn the accelerometer into a key logger.

Soundcomber [14] stealthily records and exfiltrates sensitive audio data. This data is recorded when the user speaks during a sensitive phone call. The data can then be exfiltrated through several covert channels that they describe.

Analog to Digital Communication A review of this field is beyond the scope of this paper. Instead, we will list those works most related to our implementation.

Usselman [17] patented frequency shift keying in 1949. Frequency shift keying may not be the most effective method for analog to digital modulation, but it is relatively simple. Therefore, it was an excellent choice for our proof of concept implementation.

Sound is significantly more effective than radio waves in underwater communication. Therefore, audio is the normal method of marine communication. Otnes et al. [13] present a survey of digital underwater communication techniques. These techniques may be useful to implement as future work to improve our implementation.

Information Flow Control on Android As mentioned in the introduction, there are several information flow control tools for Android.

Aquifer [11] prevents users from accidentally leaking files through network connections. It is limited to accidental disclosures and does not prevent malicious information leaks.

Heuser et al. [12] present Android Security Modules (ASM). ASM is a framework to make the creation of security enhancements for Android more efficient. They also list and describe 15 existing Android security extensions. Many of these tools are information flow control enforcers.

8 Conclusion

The production of this article has led the authors to three conclusions.

First, our experiment results suggest that data exfiltration via sound on mobile devices is a practical attack. The ranges supported by our implementation are more than sufficient for an attacker to stealthily record from an infected device. The maximum bitrates we recorded are also sufficient for sharing images and documents in intra-device attacks.

Second, sound will become a more attractive channel for data exfiltration as network security increases. Users are steadily becoming aware of the value of their sensitive data and the security risks of using a mobile device. This trend and frameworks for security tools on Android will make security enhancements even more common. Attackers may need to use unconventional methods for exfiltration as network connections are more closely monitored.

Third, many of the sensors and actuators on mobile devices are grossly underestimated in terms of their impact on security. No explicit permission is required to access the accelerometer despite the many potential ways to abuse it. The same can be said for the speakers and vibrator. These features are also underestimated by existing information flow control systems. These systems do not consider them to be useful sources or sinks for sensitive information.

9 Acknowlegements

We would like to thank those that contributed their knowledge or resources to this paper. Dr. Purush Iyer acted as my official adviser during the design and implementation of this work. Dr. William Enck provided advice and four Nexus 7 devices for testing. Mayank Manjrekar and Santosh Kumar provided us with an FSK modem for Android which we adapted to use ultrasonic frequencies. We also added optimizations and evaluation mechanisms to their implementation. Leia Kagawa and Feifei Wang both assisted with our experiments.

This work was funded by the United States Army Research Office (ARO) grant number W911NF-14-1-0117.

10 Availability

The source code used in our experiments is available under the GPL version 2 license at https://bitbucket.org/ladeshot/ultrasonicfsk.git.

References

[1] Accelerometer monitor - android apps on google play. Web site: https://play.google.com/store/apps/details?id=com.lul.accelerometer [Last accessed: 2014-05-04].

[2] frequency modulation - computer definition. Web site: http://www.moz.ac.at/sem/lehre/lib/ks/lib/fm/frequency-modulation.html [Last accessed: 2014-05-04].

[3] Frequency-shift keying - wikipedia, the free encyclopedia. Web site: http://en.wikipedia.org/wiki/Frequency-shift$_{k}eying[Lastaccessed : 2014-05-04]$.

[4] A ring tone meant to fall on deaf ears - new york times. Web site: http://www.nytimes.com/2006/06/12/technology/12ring.html?$_{r}=2oref=slogin[Lastaccessed : 2014-05-04]$.

[5] Vibration loop - android apps on google play. Web site: https://play.google.com/store/apps/details?id=net.oxdb.Vibration [Last accessed: 2014-05-04].

[6] BERESFORD, A. R., RICE, A., SKEHIN, N., AND SOHAN, R. Mockdroid: trading privacy for application functionality on smartphones. In *Proceedings of the 12th workshop on mobile computing systems and applications* (2011), ACM, pp. 49–54.

[7] ENCK, W., GILBERT, P., CHUN, B.-G., COX, L. P., JUNG, J., MCDANIEL, P., AND SHETH, A. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI* (2010), vol. 10, pp. 1–6.

[8] ENCK, W., ONGTANG, M., AND MCDANIEL, P. Mitigating android software misuse before it happens. *Citeseer* (2008).

[9] HANSPACH, M., AND GOETZ, M. On covert acoustical mesh networks in air. *Journal of Communications 8*, 11 (2013).

[10] MILUZZO, E., VARSHAVSKY, A., BALAKRISHNAN, S., AND CHOUDHURY, R. R. Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), ACM, pp. 323–336.

[11] NADKARNI, A., AND ENCK, W. Preventing accidental data disclosure in modern operating systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 1029–1042.

[12] NADKARNI, A., AND ENCK, W. Asm: A programmable interface for extending android security. In *Technical Report TUD-CS-2014-0063* (2014), Intel CRI-SC at TU Darmstadt, North Carolina State University, CASED / TU Darmstadt.

[13] OTNES, R., ASTERJADHI, A., CASARI, P., GOETZ, M., HUSØY, T., NISSEN, I., RIMSTAD, K., VAN WALREE, P., AND ZORZI, M. *Underwater acoustic networking techniques*. Springer, 2012.

[14] SCHLEGEL, R., ZHANG, K., ZHOU, X.-Y., INTWALA, M., KAPADIA, A., AND WANG, X. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *NDSS* (2011), vol. 11, pp. 17–33.

[15] SUBRAMANIAN, V., ULUAGAC, S., CAM, H., AND BEYAH, R. Examining the characteristics and implications of sensor side channels. In *Communications (ICC), 2013 IEEE International Conference on* (2013), IEEE, pp. 2205–2210.

[16] TEMPLEMAN, R., RAHMAN, Z., CRANDALL, D., AND KAPADIA, A. Placeraider: Virtual theft in physical spaces with smartphones. *arXiv preprint arXiv:1209.5982* (2012).

[17] USSELMAN, G. L. Frequency shift keying, Feb. 8 1949. US Patent 2,461,456.