Covert Acoustic Channels

Improving Range, Accuracy, and Undetectability

Evan Lynch, Yihua Li, Wei Zhao

May 15, 2014

Contents

Abstract

The ultrasonic acoustic channel is a form of covert channel that can be used to transmit signals across an air gap without detection by the human ear. In this paper, we explore four existing implementations and design a new system that achieves better working range and higher accuracy. Additionally, recognizing that the covert nature of the channel relies on limitations of the human ear, we design and implement a system that extends that undetectability to less limited passive listening devices. We focus on using unmodified commodity hardware to simulate conditions consistent with our security and threat model.

Keywords: ultrasonic, covert channel, acoustic

1 Introduction

In computer security, a covert channel is a type of attack that allows the sending and receiving of information between processes that should not or are not expected to communicate according to the security policy of the system. Lampson who introduced this concept writes, "A covert channel is a parasitic communication channel that draws bandwidth from another channel in order to transmit information without the authorization or knowledge of the latter channel's designer, owner or operator."[1] Various types of covert channels exist. One example is covert storage channels, which utilize a shared resource as a method of communication. Other covert channels make use of information carried by a comunication medium that is not intended to be used to encode messages such as unprescribed time delays in wireless communication.

In this paper, we explore a third type of covert channel which exploits an entirely new medium for communication—sound. Acoustic waves have been mostly used for underwater communications where it is more efficient than radio waves. Over the air, we commonly utilize radio waves for information transmission, but some have proposed to use sound as a method for wireless communication.[2] In essence, audio networking means that transmitters modulate data and play the resulting audio data using the host machine's output audio devices such as speaker. The sound is transmitted over the air over short range. Receivers listen via input devices such as microphones and demodulate incoming signals to recover the transmitted information. As a result, even though two computers are not connected to each other via any established type of network interface, they can still communicate with each other by transmitting and receiving sound over the air using microphone and speakers.

To make the communication covert, the transmission is made in the ultrasonic or near ultrasonic frequency range that humans cannot not hear. As most commodity speakers are only capable of producing sound with a 44.1KHz sample rate, this allows us to use a maximum frequency of about 22KHz by the Nyquist-Shannon sampling theorem. The ultrasonic range for most humans is above 19.5KHz.

Covert channels constitute a major security threat for a sensitive system. Keeping a computer unnetworked and separated from networked computers by an airgap is a common physical security measure. Research shows critical data could be leaked and transmitted via the audio subsystem[6]. In addition, as the speakers and microphones are not commonly considered in system and network security policies, the adversary

could exploit the vulnerability and gain control of the sytem to set up a botnet of devices for stealthy information transmission.

1.1 Threat and Security Model

For our work, we supposed that there is a sensitive system that utilizes an airgap between a networked and an unnetworked computer. The adversary has offline access to both computers so they can install malware that collects information, transmits it with audio, and receives this audio and transmits the information over the standard network. However, they cannot modify the hardware of either computer without risking discovery. Following this model, we tested and implemented acoustic channel designs on commodity laptops without introducing special audio equipment.

Additionally, we assume that the operator of the sensitive system is not initially aware of the covert communication, and so questions of covertness are made assuming a passive and unassuming listener, while questions of detectability are made asssuming an active listener.

2 Previous Work

Previous works have been done on acoustic channel for both underwater environment and over the air. E. M. Sozer, et al, [3] survey the existing network technology and its applicability to underwater acoustic channels, while G. Leus, et al, [4] explores the possibility of making audio communication between unmanned underwater vehicles less detectable by third party by transmitting audio messages at a low signal -to-noise ratio (SNR). The idea of using audio as a networking technique is further explained in [2] and the authors implemented a variant of on-off keying (OOK) modulated over a 21.2-kHz carrier and achieved inaudible data transmission at a bit rate of 8 bps across 3.4 meters in an office. In addition to OOK, other modulation techniques such as binary frequency-shift keying (BFSK), and binary phase shift keying (BPSK) could also be used, and the authors in [5] experimented with transmitting digitally coded signals across an air gap in the lab with all three methods and demonstrated that BPSK can be used to transmit signals with a low bit error rate. Finally, Hanspach, et.al, used acoustic communication to construct a convert acoustical mesh network consisting of malware infected laptops. The system utilized near ultrasonic frequency

range and achieved transmission of up to 19.7 m with rate of about 20 bits/s between two nodes [6].

3 Existing Implementations

We found four existing implementations for which we could acquire the source-code. We set out to explore these existing implementations to learn about the state of the art and then proceed from there. In each of these implementations, we discover a different set of draw-backs. We focus on analyzing each implementation for the following aspects: throughput, accuracy, working range, covertness and detectability. Covertness is defined as how much a system would alert an unsuspecting user, while detecetabilty is defined as how easy it is for a user who actively tries to look for such covert transmissions.

3.1 GNU Radio Implementation

This implementation of ultrasonic networking uses GNU radio, a software development toolkit that provides signal processing blocks to implement software defined radio [7]. The system utilizes Frequency Shift Keying modulation with a carrier frequency of 19 KHz, and is therefore hardly detectable by the human ear. The implementation is also reliable since it uses a packet encoder to convert data into byte stream and adds a checksum and preamble to the stream before transmission. And with each symbol, 9 samples are generated, making the signal more resilient to noise. However, this redundancy limits the throughput of the audio transfer and in our experiment, we found that the characters are transmitted with high latency.

3.2 Quietnet

Quietnet uses straightforward implementations[8]. It uses amplitude modulation to send signals using a fixed frequency (usually in ultrasonic range) by varying the amplitude. It relies on a specific hardcoded encoding that translate a subset of characters in the ASCII into a bit pattern that starts and ends with a one without a consecutive two zeroes in between. Then it uses ‘00’ as the sigal to tell each different character apart while receiving the message. It doesn’t use any error correction code, which makes it not reliable and prone to error propagation. It also uses a hard-coded

absolute threshold to detect the existence of a signal, which makes its working range small, as signals attenuate exponentially with distance. However, because Quietnet does not use any error correction, there is no redundant information being sent. This makes the throughput within working range very high, even though it is susceptible to errors.

3.3 6.02 Audiocom Lab Implementation

We reuse the audiocom lab code for 6.02 class[9]. The lab code focuses on sending and receiving information using audio channel. This implementation uses frequency modulation and Viterbi encoding and decoding. The frequency modulation enables this implementation high throughput because now we can send messages using different frequencies. Using Viterbi encoding and decoding lends this implementation high accuracy and fewer errors. By setting the carrier frequency to be ultrasonic range, we are able to send messages through the audio channel with high accuracy and high throughput. However, because we have a fixed sampling rate, which cannot exceed 48kHz, we experience aliasing effects, which produces an audible hissing sound. This is not desirable as we strive for a covert channel that is not easily discoverable by human.

3.4 JavaScript Implementation

Sonicnet.js is a javascript library that uses Web Audio API to send and receive data as sound [10]. The sound is generated by the Web Audio Oscillator node on the transmitter side and processed through a real-time Analyser node in the receiver side. The implementation utilizes a single-tone multi-frequency signaling (STMF) scheme: the available frequency spectrum is split into ranges with each range corresponding to a specific character for transmission. With the library, the authors implemented a web app for sending emoticons. Each emoticon is represented by a character and a sound is produced and transmitted at the frequency corresponding to that character. Similar to Quietent, sonicnet.js does not use any error correction scheme and is therefore susceptible to errors. The throughput is also low as the system transmits each character in sequence and waits some time in between.

Table 1: Summary of observations

Metric	gnuRadio App	Quietnet	Audiocom Lab	JavaScrip App
Maximum working range for $\geq$ 90% accuracy	1.5m	0.03m	0.5m	0.4m
Behavior when outside range	Fails to detect preamble	Fails to interpret any character correctly	Fails to detect preamble	Fails to detect any character
Throughput within range	1.5 sec/char	0.8 sec/char	0.3 sec/char	1 sec/char
Covertness	High	High	Low	High
Detectability	High	High	High	High

3.5 Summary of Observations

We used two laptops that were bought within the last two years. To test the systems, we set the send laptop to its maximum volume. We summarize our findings in Table 1.

4 Improvements

Following our initial investigation, we decide to first improve on range and accuracy and then on undetectability. Our first implementation *QuietChat* focuses on improving the range and accuracy. Our second implementation *DogWhisper* focuses on improving undetectability.

4.1 Improving Range and Accuracy: *QuietChat*

To improve upon the existing implementations, we notice the following limitations for Quietnet and Audiocom:

1. In Quietnet, a single ultrasonic frequency is used. Bits are sent as chunks of samples over time. Bit 1 is a chunk of this frequency at maximum amplitude, and bit 0 is simply silence. At the receiver's side, a hardcoded amplitude threshold is used to decide whether a bit is 1 or 0. The major problem with

this is that if the range is too far, even a bit 1 will fall below the amplitude threshold. Lowering the threshold will not solve this problem, because doing so will cause problems when the range is close.

2. In Audiocom, frequency modulation is used to encode data. An ultrasonic frequency is used for the carrier. Although this brings all the frequencies in the data samples to ultrasonic range, due to aliasing the speaker emits audible noise. Therefore, we decide to use a simple scheme that combines both techniques from Quietnet and Audiocom. The key idea is to send each bit as a different frequency. To detect which bit was sent, we compare the amplitudes of both frequencies and pick the higher one. With this idea, we can also send more than 2 kinds of bits, with a different frequency each, to help us with synchronization and error correction. To increase range, we also make a tradeoff between throughput and accuracy. We decide that accuracy and range are more important than throughput for ultrasonic covert communications, because what we want to do is be able to transmit data between two nearby machines; the data we transmit may be very small (such as a password), but the accuracy of the data is quite important.

Focusing on improving the range and accuracy of the system, we design a new system called *QuietChat*. In this design, the sender sends audio samples in chunks, just like in Quietnet. Each chunk is a continuous pure frequency with maximum amplitude, representing one of the five kinds of bits: 0, 1, A, B, C. We use bits 0 and 1 to transmit data, bits A and B to synchronize the timing, and C to signal the end of the stream.

4.1.1 Repetition Error Correction

For each bit we want to send, we send five copies of this bit (and thus five chunks, called a group),This allows us to sample each of the five chunks as a bit and take the majority. This is a simple error correction scheme. While reading the stream, we actually only read three chunks per group, because our timing of the chunks may not align with the actual timing of the received signals, but we can synchronize the timing to make sure that the three chunks we read lie completely inside the group. Then we use the three chunks to correct one-bit errors.

4.1.2 Synchronization

To synchronize the timing, at the beginning of the transmission, a preamble sequence of ABABABABABAB is sent (each A and B repeats 5 times as above). At the receiver, who samples chunks continuously, whenever 3 consecutive A's are seen (which are likely the first 3 A's sent in that group), we skip three samples and read three more samples (so that these three new samples are in the center of the second group). If these three new samples are all B's, we skip two samples and read three new samples again, and if these are all A's, we skip two samples and read three new samples yet again. If these are all B's, we have detected a pattern of ABAB and synchronized ourselves to the stream. We send AB multiple times in the preamble to give the receiver at least three chances to detect the ABAB pattern.

From this point on, the receiver repeats by reading three chunks every five chunks, takes the majority of them, ignore any A or B bits (since there can be more preamble bits not yet consumed), and output any 0 or 1 bits, by converting them into ASCII text and printing it out.

4.1.3 Termination

After sending all the bits, the sender sends five bits of C (each repeated 5 times as above) to signal the end of the stream. When the receiver detects 4 consecutive bits of C, it stops and reset itself as if no bits were ever received. Then the receiver continues to wait for the next transmission.

4.1.4 Implementation

We implemented this design in Python on top of PyAudio, and compared our implementation with Quietnet.

4.1.5 Results

We use the same testing configurations as we used before. We mainly compare our results with *Quietnet* in Table 2.

Table 2: Comparison between QuietChat and Quietnet

Metric	QuietChat	Quietnet
Maximum working range for $\geq 90\%$ accuracy	at least 3 m (higher range not tested)	0.03m
Behavior when outside range	Either preamble is not detected, or the text received is partially incorrect (tested by turning down volume)	Fails to interpret any character correctly
Throughput within range	2 sec / char	0.8 sec/char
Covertness	High	High
Detectability	High	High

4.2 Improving Undetectability: *DogWhisper*

Although in the ultrasonic range, all of the strategies for covert communication we have discussed before are easily detected with a cursory examination of the frequency spectrum of sound in the room. A large spike in the frequency domain will be noticeable at the carrier frequency as illustrated in Figure 1. At the expense of throughput, we attempted to design a covert transmission that is less detectable. This system we call *DogWhisper*.

4.2.1 Spread Spectrum Encoding

To attempt to avoid detection by a spectrum-aware though still unwitting observer, we designed and implemented the following spread-spectrum encoding. Each bit of message is transmitted as random noise that has been filtered to be only in the ultrasonic spectrum $19500Hz$ to $21000Hz$. This frequency range has been broken up into ten evenly-spaced bins. To encode a 0 or 1, half of the bins are zeroed out of the noise spectrum. The selection of which bins determines the bit. We call these no-tolerance filters one-pass and zero-pass. Figure 2 shows an idealized output signal for one-pass filtered noise. Modulated noise rather than a cleaner signal is chosen so that it blends more discreetly into the background. Using the entire spectrum allows for a lower power signal

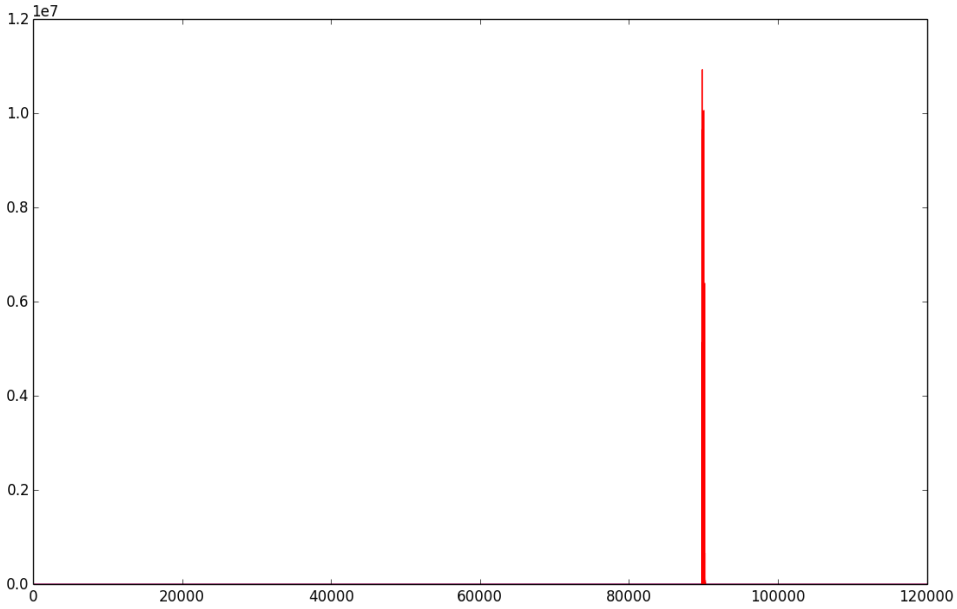

Figure 1: This is the read-out of a frequency analyzer examining the ambient sound in the room while the GNURadio implementation is transmitting. Notice the large spike at the carrier frequency. The frequency axis ranges from 10000 to 23000 Hz.

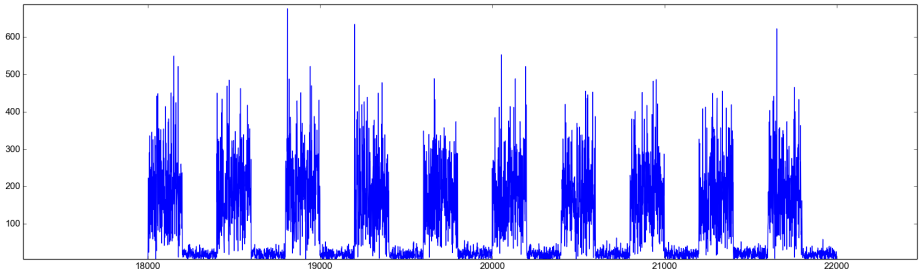

Figure 2: This is a frequency domain view of a simulation which encodes 1.

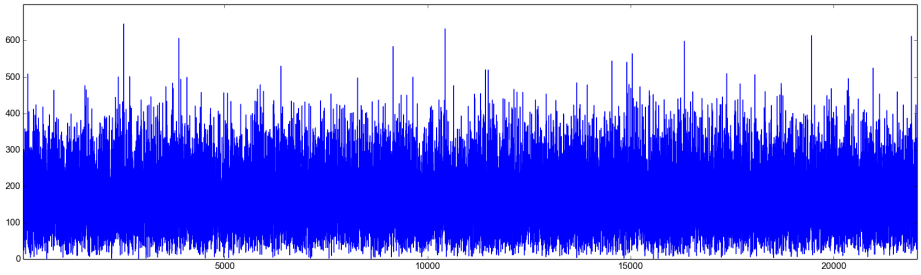

Figure 3: This is a the frequency-domain spectrum of the simulated signal with added noise that was presented to the decoder. The signal is compellingly hidden within the noise although the decoder could still interpret the correct bit ninety-percent of the time.

4.2.2 Decoding

To decode the *DogWhisper* signal, the decoder reads in fixed-length buffers of input from the microphone. For each buffer, the decoder decides if it is hearing a 1 or a 0 and how confident it is about what it is hearing. The input is said to be a 1 if the average power in the one-pass spectrum is greater than the average power in the zero-pass spectrum, and vice versa. The absolute value of the difference determines the confidence. When no signal is transmitted the difference should be small, while when a signal is transmitted, the difference should be large.

If the confidence is above an experimentally derived threshold, the bit is recorded. If it is below the threshold, a '_' character is recorded. This string of characters is then sent to the next step.

In the next step, the incoming bits are passed through a median filter which removes small fluctuations in the recorded characters. This is reasonable because a single bit is transmitted for a period of time longer than several fixed-length input buffers. As an example, if the input is 0000100___0____11111001111___1__1___ this would be smoothed out by the median filter to 0000000_____1111111111_____.

Finally, a state-machine looks for edges between regions of 0, 1, and _ and outputs bits accordingly.

4.2.3 Implementation

Like *QuietChat*, *DogWhisper* is implemented with Python using the PyAudio module. The range of the *DogWhisper* was tested by transmitting the signal from a 2011 MacBook Pro and receiving the signal on a 2010 Lenovo ThinkPad.

4.2.4 Results

Focusing on detectability, we did not make attempts to improve the robustness of the message via error correction, etc. and measured our success transmitting single bits. Initially we tested our design with simulations, hiding the signal in generated noise and then letting the decoder attempt to interpret the signal. Figure 3 shows the spectrum of the simulated signal which could be successfully interpreted for both bits ninety-percent of the time by a simulated decoder.

Using hardware, transmission of bits via *DogWhisper* was markedly more robust than the somewhat similar QuietNet implementation. Bits could be successfully distinguished nearly one-hundred percent of the time when transmitting at close range (up to a meter), even with significant background noise. Even up to three meters, the bits could be successfully distinguished more than seventy-percent of the time. Ranges beyond this were not measured. This first experiment was executed with the transmission produced at top volume. At this volume spectrum, analysis quickly revealed that the goal of hiding the signal in noise was not entirely achieved, although a significant improvement over the GNURadio spike in Figure 1. We determined that our simulation drastically over-estimated the amount of background noise that generally exists in the ultrasonic spectrum. Although our signal could be distinguished in the presense of this noise, without the noise there to begin with, there was little to hide it. However, by reducing the volume of the transmission significantly, we were able to produce a transmission that was still robust up to a range of a meter while being significantly less detectable. Figure 4 shows the spectrum of noise captured during transmission of this lower-power signal.

5 Future Work

Our current implementations manage to improve on some aspects, but either of the implementation manage to improve the system on all aspects. Future work

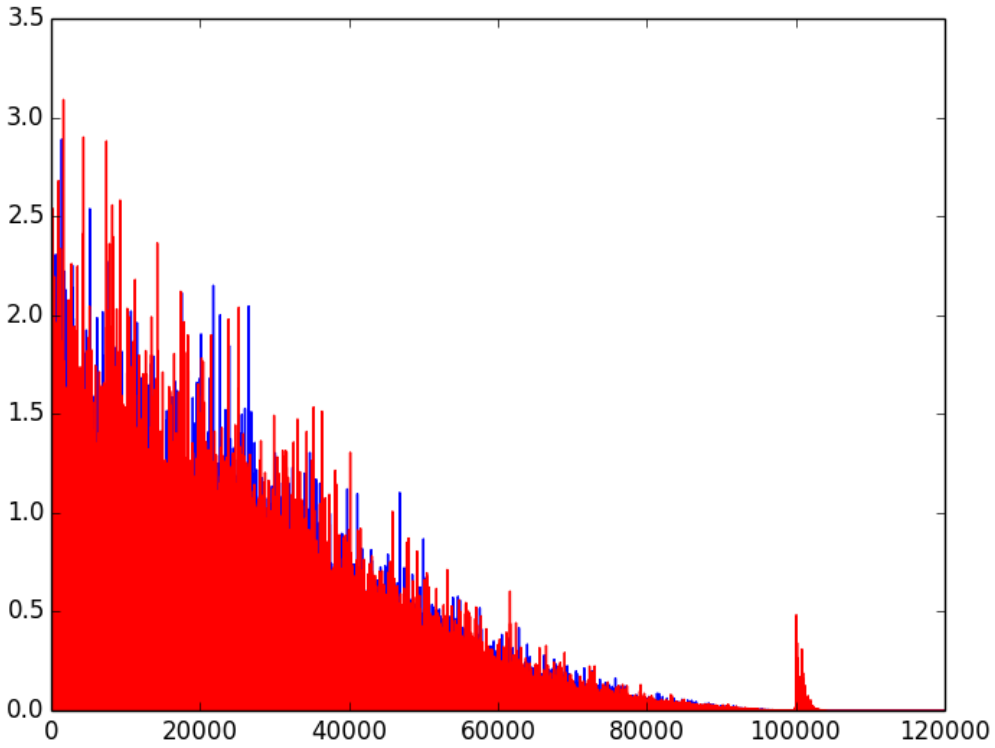

Figure 4: This the spectrum in the room when *DogWhisper* is trans mitting. The x-axis is a linear plot of frequency in the range 10000 to 23000 Hz. The y-axis corresponds to an unnormalized power detected at each frequency. The blue data behind is a control measurement that closely follows the red data except for a notable artifact in the ultrasonic range. Note that the noise profile on the left (which attenuates at higher frequency primarily because of the decreasing effectiveness of our listening hardware at higher frequencies) though present does not even register a pixel on the GnuRadio spectrogram.

can be focused on merging these two implementations to improve range, accuracy and undetectebility. Furthur research needs to be done to improve the throughput while still keeping the maximum working range and accuracy high and detecteb9ilty low.

6 Conclusion

In summary, we first explored different exisiting implementations of ultrasonic acoustic covert channels. Based on our findings, we implemented two new systems, *QuietChat* and *DogWhisper*. *QuietChat* achieves higher robustness with much higher maximum working range and accuracy by using multiple frequencies amplitude modulation and repetition error correction scheme. *DogWhipser* also achieves greated robustness and range and achieves notably lower detectability by employing a spread-spectrum encoding that modulates simulated noise.

References

[1] B. W. Lampson, *A note on the confinement problem.* Commun. ACM, vol. 16, no. 10, pp. 613-615, Oct 1973.

[2] A. Madhavapeddy, D. Scott, A. Tse, and R. Sharp, *Audio Networking: The forgotten wireless technology.* IEEE Pervasive Computing, vol. 4, no. 3, pp. 55–60, July 2005.

[3] Sozer, Ethem M., Milica Stojanovic, and John G. Proakis. *Underwater acoustic networks.* IEEE Journal of Oceanic Engineering, vol. 25, no. 1, pp. 72-83, Jan 2000.

[4] G. Leus and P. van Walree, *Multiband OFDM for Covert acoustic communications*, IEEE J. Sel. A. Commun., vol. 26, no. 9, pp. 1662–1673, Dec 2008 .

[5] C. Li, D. Hutchins, and R. Green, *Short-range ultrasonic digital communications in air*, IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, vol. 55, no. 4, pp. 908–918, 2008.

[6] M. Hanspach, and M. Goetz. *On Covert Acoustical Mesh Networks in Air.* Journal of Communications, vol. 8, no. 11, 2013.

[7] *Ultrasound Networking.* Anfractuosity. N.p., n.d. Web. May 2014.

[8] K. Murphy. *Katee/quietnet.* GitHub. N.p., n.d. Web. May 2014.

[9] H. Balakrishnan. *audiocom602.blogspot.com/* MIT EECS. N.p., n.d. Web. August 2012

[10] B. Smus. *Borismus/sonicnet.js.* GitHub. N.p., n.d. Web. May 2014.